AYDIN BODUR

ePUB

.pdf

e-kitap

MATLAB İLE ÇALIŞMAK

*Matlab'a Giriş
*Diziler
*Polinomlar
*Matrisler
*Grafik Çizimi
*Saat ve Tarih
*Verilen Noktalardan Geçen Eğrinin Denklemini Bulmak
*İnterpolasyon
*Matlab'da Dosyalamak
*Matlab ile İntegral Hesaplamak
*Sembolik Matematik

EMO YAYIN NO: EK/2011/28
ISBN. 978-605-01-0247-5

1954

TMMOB
Elektrik Mühendisleri Odası

TMMOB
Elektrik Mühendisleri Odası
1954

MATLAB İLE ÇALIŞMAK

1.Baskı, Ankara-Aralık 2011
ISBN: 978-605-01-0247-5
EMO Yayın No: EK/2011/28

TMMOB Elektrik Mühendisleri Odası
Ihlamur Sokak No:10 Kat:2 06640 Kızılay Ankara
Tel: (312) 425 32 72 Faks: (312) 417 38 18
http://www.emo.org.tr E-Posta: emo@emo.org.tr

Kütüphane Katalog Kartı

005.3 20 DOĞ 2011

Matlab İle Çalışmak Kitabı; Yayına Hazırlayan: EMO Genel Merkez, --1.bs.--Ankara. Elektrik Mühendisleri Odası, 2011

323 s.:24 cm (EMO Yayın No:EK/2911/28; ISBN:978-605-01-0247-5)

Matlab

Dizgi
TMMOB Elektrik Mühendisleri Odası

Baskı
TMMOB Elektrik Mühendisleri Odası

MATLAB İLE Ç A L I Ş M A K

Profesör Dr. Doğan İbrahim

Başlarken

Bu kitap, Profesör Doğan İbrahim'in 2004 yılında yazdığı, MATLAB kitabının e.kitap olarak tıpkı basımıdır.

MATLAB programı matematik ve bilhassa mühendislik işlemlerinde en yaygın olarak kullanılan programlardan biridir. İlk zamanlar oldukça pahalı olan bu programı sadece profesyonel kuruluşlar alabilmekteydi. Son zamanlarda fiyatında düşüş gösteren MATLAB'ı bugün çok sayıda Üniversite, araştırma merkezi, mühendislik büroları, ve matematikle ilgilenen birçok kuruluş kullanmaktadır. Halen MATLAB'ın öğrenci versiyonları da bulunmaktadır. Fiyatı çok daha düşük olan ve bilhassa matematik ve mühendislik öğrencileri için hazırlanmış olan öğrenci versiyonları bazı kısıntıları dışında esas MATLAB'ın yapabileceği hemen herşeyi yapabilmektedirler.

MATLAB'ın popüler olmasının esas sebebi çok güvenilir olması ve kullanımının oldukça kolay oluşudur. Örneğin, MATLAB, 5 x 5 bilinmeyenli bir lineer denklemi birkaç saniye içerisinde çözebilmektedir. MATLAB'ın diğer önemli bir özelliği de 20 den fazla "toolbox" unun bulunmasıdır. Bu toolbox'lar özel alanlar için geliştirilmiş ve MATLAB'a ilave edilebilen programlar halindedirler. Örneğin, kontrol mühendisliği için "Control Systems Toolbox" bulunmaktadır.

MATLAB, Microsoft Windows ve Macintosh bilgisayarları için geliştirilmiştir. Her iki ortamda da yazılan MATLAB programları birbirleri ile uyumludurlar.

Bu kitap esas olarak matematik ve mühendislik öğrencileri ve pratisyen mühendislerin istekleri göz önünde bulundurularak yazılmıştır. Kitap Üniversite seviyesindeki öğrencilere yönelik olmasına rağmen lise fen kolunda okuyan ve MATLAB programına erişebilen öğrenciler de kitabı faydalı bulacaklardır. Pratisyen mühendisler günlük iş ortamlarında matematik gerektiren işlemlerde kitabı faydalı bulacaklardır. Kitap MATLAB'ın genel kullanımı hakkında toolbox'lar hariç hemen herşeyi ihtiva etmektedir.

Kitap 11 bölümden meydana gelmiştir. İlk bölümde MATLAB'ın kullanımı ve sıkça kullanılan MATLAB komutları örneklerle açıklanmıştır.

İkinci bölümde diziler ve vektörler göz önünde bulundurulmuş ve MATLAB'ı kullanarak program yazma tekniği örneklerle açılanmıştır. Bu bölümde ayrıca fonksiyonlardan bahsedilmiş ve kullanıcı fonksiyonlarının nasıl yaratıldığı çeşitli örneklerle açıklanmıştır.

Kitabın üçüncü bölümünde polinomlar ve MATLAB'ı kullanarak çeşitli polinom işlemlerinın nasıl yapıldığı açıklanmıştır.

Önemli bir konu olan matrislere kitabın dördüncü bölümünde yer verilmiştir. Bu bölümde ayrıca lineer denklemlerin çözümü birçok örneklerle açıklanmıştır.

Bölüm 5 de MATLAB'ı kullanarak grafik şekil çizme tekniği çeşitli örneklerle açıklanmıştır.

Kitabın 6.cı bölümünde MATLAB'ın tarih ve zamanla ilgili fonksiyonlarına ve bu fonksiyonların kullanımlarına yer verilmiştir.

Bölüm 7 de verilen noktalardan geçen herhangibir polinomun denkleminin bulunması açıklanmıştır.

Bilinen noktalar arasında interpolasyon yapma tekniği ise birçok örneklerle bölüm 8 de geniş bir şekilde açıklanmıştır.

MATLAB'ı kullanarak dosya yaratmak ve bu dosyada bilgi saklamak veya mevcut bir dosyadaki bilgileri okumak Bölüm 9 da açıklanmıştır.

MATLAB'ı kullanarak integral hesaplamalarına kitabın 10. cu bölümünde yer verilmiştir.

Son olarak, kitabın 11. ci bölümünde sembolik matematik işlemlerine yer verilmiş ve MATLAB'ı kullanarak sembolik türev ve integral işlemlerinin nasıl yapıldığı örneklerle açıklanmıştır.

Prof.. Dr. Doğan İbrahim
2004

Daha önce Bileşim Yayınlarından çıkan bu kitabı, EMO kanalıyla, bu kez e-kitap olarak sunuyoruz, bu e-kitaplara katkılarından dolayı, EMO yayınları ile uğraşan başta Sn. Emre Metin, Sn.Hakkı Ünlü ve Sn.Orhan Örücü olmak üzere tüm EMO yetkililerine ve kitabın yazarı Sayın Doğan İbrahim'e teşekkür ederiz.

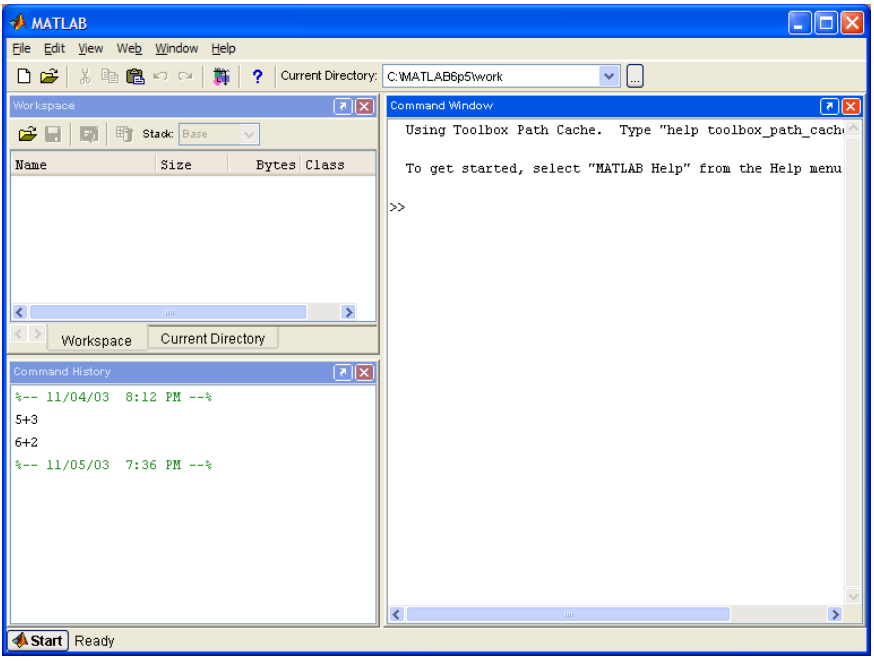
Aydın Bodur

1
MATLAB'A GİRİŞ

1.1 MATLAB'ı Çalıştırmak

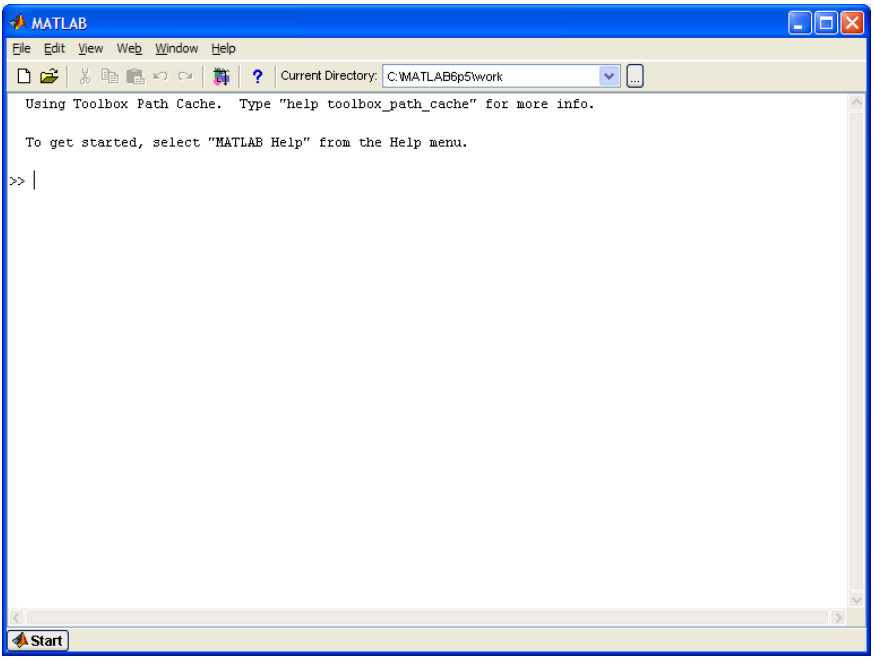
MATLAB'ı kullanmak için MATLAB'ın bilgisayarınıza yüklenmiş olması veya MATLAB'ın bulunduğu bir bilgisayara ağ ile erişebilmeniz gerekmektedir. MATLAB bilgisayarınıza yüklenmiş değilse MATLAB CDROM'unu bilgisayarınıza takıp gerekli işlemleri yapınız ve MATLAB'ı yükleyiniz.

MATLAB'ı başlatmak için desktop'ta bulunan MATLAB ikonunu tıklayınız. Bir müddet sonra karşınıza Şekil 1.1 de gösterilen pencereler çıkacaktır.



Şekil 1.1. MATLAB pencereleri

Burada sol üst köşedeki pencere MATLAB dosyalarının bulundukları yapıyı göstermektedir. Sol alttaki pencere ise yazmış olduğumuz geçmiş komut listesini gösterir. Bizim esas olarak kullanacağımız pencere ise sağdaki penceredir. Şimdi soldaki pencerelerin sağ üst köşelerindeki 'X' işaretlerini tıklayıp bu pencereleri kapayınız. Karşınıza Şekil 1.2 de gösterilen ve bu kitapta kullanacağımız esas MATLAB komut penceresi çıkacaktır.



Şekil 1.2. MATLAB komut penceresi

Şimdi MATLAB'ı kullanmaya hazırız. İlk olarak MATLAB'ı kullanarak basit matematik işlemlerin nasıl yapıldığına bakacağız.

1.2 MATLAB İle Basit İşlemler

MATLAB'ın komut almaya hazır olduğu ">>" karakterleri ile belirtilmektedir. Şimdi 5 ve 3 sayılarını toplayalım

```
>> 5+3  Enter

ans =

        8
>>
```

elde ederiz. Örnekte de görüleceği gibi her komuttan sonra **Enter** tuşuna basmamız gerekmektedir. Daha sonraki örneklerde **Enter** tuşuna basıldığı kabul edilmekte olup bu tuş gösterilmeyecektir. MATLAB son cevabı **ans** diye adlandırılan bir değişkende tutmaktadır. Şimdi bu değişkeni kullanarak işlem yapmaya devam edebiliriz

```
>> ans+4

ans =

        12
>>
```

olur.

MATLAB'ı kullanarak iki skalar değişken arasında yapabileceğimiz en basit işlemler Tablo 1.1 de verilmiştir.

Tablo 1.1 MATLAB skalar aritmetik işlemleri

İşlem	**Matematik şekli**	**MATLAB**
Toplama	a + b	a + b
Çıkarma	a – b	a – b
Çarpma	a x b	a * b
Sağ bölme	a / b	a / b
Sol bölme	b \ a	a \ b
Güç	a^b	a ^ b

MATLAB ile sayılar tam sayı olarak, ondalıklı olarak, veya kayma nokta formatı kullanılarak gösterilebilir. Bazı sayı örnekleri aşağıda gösterilmiştir

2.789 -234 0.003 1.24e3

MATLAB ile kullanılabilecek olan en büyük ve en küçük sayılar $\pm 10^{-308}$ ve $\pm 10^{308}$ arasında olabilir.

İşlem yaparken aritmetik operatörler Tablo 1.2 de gösterilen öncelik sırasını takip ederler.

Tablo 1.2 Aritmetik işlem öncelik sırası

Öncelik	**Aritmetik operatör**
1	Parantez
2	Güç (soldan sağa)
3	Çarpma, bölme (soldan sağa)
4	Toplama, çıkarma (soldan sağa)

Örnek 1.1

Aşağıdaki sayıları MATLAB formatında yazınız:

1.35×10^{4} -12.65×10^{-5} 10^{8}

Çözüm 1.1

1.35e4 -12.65e-5 1e8

Örnek 1.2

Aşağıdaki işlemleri MATLAB kullanarak çözünüz:

i) $2 \times 23 + 3 \times 12 + 5 \times 45$
ii) 2^{2x3}
iii) $2.5 \times 10^{-3} + 2.2 \times 10^{-2}$

Çözüm 1.2

i) >> 2*23 + 3*12 + 5*45

ans =

```
            307
        >>
```

ii)
```
        >> 2^(2*3)

        ans =

            64

        >>
```

iii)
```
        >> 2.5e-3 + 2.2e-2

        ans =

            0.0245

        >>
```

1.3 Değişkenler

Değişkenler matematikte çok yaygın olarak kullanılmaktadırlar. MATLAB'da herhangibir değişkenin bir harf ile başlaması gerekir. Bunun yanında bir değişken içerisinde harfler (a – z arasında), sayılar (0 – 9 arasında) ve alt çizgi (_) karakteri kullanılabilir. Geçerli değişken isimleri şunlar olabilir:

maksimum max2 son_toplam x23

Geçersiz değişken isimlerine örnek olarak ise şunlar gösterilebilir:

2sayac ilk-sayı _23a

MATLAB kullanırken küçük harf ve büyük harfler değişik olarak algılanır. Bu durumda, örneğin, **kitap** ve **Kitap** değişik değişkenlerdir. Bir değişkene değer verirken değişkenin adı

yazılır ve eşittir "**=**" işareti kullanılarak istenilen değer verilir. Aşağıdaki örnekte **ilk** değişkenine 5 değeri verilmiştir:

```
>> ilk = 5

ilk =

        5

>>
```

Herhangibir değişkenin değeri o değişkene yeni değer vererek değiştirilebilir. Yukarıdaki örnekte, **ilk = 8** komutu ile **ilk** değişkeninin yeni değeri 8 olur.

MATLAB'da normal olarak yukarıdaki örnekte de görüleceği gibi ara işlemler ekranda görülür. Birçok uygulamalarda ara işlemleri değil sadece son cevabı görmek isteriz. Ara işlemleri saklamak için işlem sonuna "**;**" karakteri konmalıdır. Aşağıdaki örnekte **ilk** değişkenine 5 değeri, **son** değişkenine ise 3 değeri verilir ve iki değişken toplanıp netice **toplam** diye adlandırılan bir değişkende saklanır:

```
>> ilk = 5;
>> son = 3;
>> toplam = ilk + son

toplam =

        8

>>
```

Değişkenlerin MATLAB'da kullanımı hakkında çeşitli örnekler aşağıda verilmiştir.

Örnek 1.3

Tabanı ve yüksekliği bilinen bir üçgenin alanı şu şekilde hesaplanır:

alan = taban x yükseklik / 2

tabanı 2.3cm ve yüksekliği 10cm olan bir üçgenin alanını hesaplayınız.

Çözüm 1.3

```
>> taban = 2.3;
>> yukseklik = 10;
>> alan = taban * yukseklik / 2

alan =

    11.5000

>>
```

Birden fazla değişkeni ayni satıra yazmak için değişkenleri ara tuşu ile veya virgül ile ayırabiliriz. Örneğin, yukarıdaki işlemi şu şekilde de yapabiliriz:

```
>> taban = 2.3;  yukseklik = 10;
>> alan = taban * yukseklik / 2

alan =

    11.5000

>>
```

Örnek 1.4

Yarıçapı r ve yüksekliği h olan bir silindirin hacmi şu şekilde hesaplanır:

$$V = \pi r^2 h$$

Hacmi 120cm^3 ve yarıçapı 4cm olan bir silindirin yüksekliğini hesaplayınız.

Çözüm 1.4

Silindirin yüksekliğini şu formülle hesaplanabilir:

$$h = \frac{V}{\pi r^2}$$

MATLAB kullanarak yüksekliği şu şekilde hesaplayabiliriz:

```
>> v = 120; r = 4;
>> h = v / (pi*r*r)

h =

    2.3873
>>
```

pi değişkeni MATLAB için ayrılmış bir sembol olup değeri π ya eşittir.

1.4 Sıkça Kullanılan Matematik Fonksiyonları

MATLAB çok çeşitli matematik fonksiyonlarını desteklemektedir. En çok kullanılan matematik fonksiyonlarının listesi Tablo 1.3 de verilmiştir.

Tablo 1.3 Matematiksel fonksiyonlar

Fonksiyon	**Tanımı**
sqrt(x)	Kare kök
round(x)	En yakın tam sayıya tamamla
floor(x)	Aşağıya doğru tamamla
ceil(x)	Yukarıya doğru tamamla
sin(x)	Trigonometrik sinüs
cos(x)	Trigonometrik kosünüs
tan(x)	Trigonometrik tanjant
sinh(x)	Hiperbolik sinüs
cosh(x)	Hiperbolik kosünüs

tanh(x)	Hiperbolik tanjant
abs(x)	Mutlak değer
asin(x)	Ters sinüs
acos(x)	Ters kosünüs
atan(x)	Ters tanjant
pow2(x)	2 nin gücü (2^x)
fix(x)	En yakın tam sayı
round(x)	En yakın tam sayı
rem(x,y)	x/y nin kalanı
exp(x)	E'nin gücü (e^x)

Şimdi bu fonksiyonların kullanımını örneklerle inceleyelim.

floor(x) fonksiyonu x'den küçük olmayan en büyük tam sayıyı verir. Örneğin, **floor(2.8) = 2** ve **floor(-2.8) = -3** olur.

ceil(x) fonksiyonu x'den büyük en küçük tam sayıyı veriri. Örneğin, **ceil(2.8) = 3**, ve **ceil(-3.8) = -3** olur.

fix(x) fonksiyonu x'in ondalık kısmını atıp sadece tam sayı kısmını verir. Örneğin, **fix(2.9) = 2** ve **fix(-2.9) = -2** olur.

round(x) fonksiyonu x'e en yakın olan tam sayıyı verir. Örneğin, **round(2.48) = 2** ve **round(2.5) = 3** olur.

rem(x,y) fonksiyonu x/y işleminin kalanını verir. Örneğin, **rem(22,4) = 2** olur (22/4 = 5 ve kalan = 2).

Örnek 1.5

Hacmi 50cm^3 ve yüksekliği 12cm olan bir silindirin yarı çapını hesaplayınız.

Çözüm 1.5

Silindirin yarı çapı şu formülle hesaplanır:

$$r = \sqrt{\frac{V}{\pi h}}$$

MATLAB kullanılarak problem şu şekilde çözülebilir:

```
>> v = 50; h = 12;
>> r = sqrt(v / (pi*h))

r =

        1.1516

>>
```

Örnek 1.6

$ax^2 + bx + c = 0$ denkleminin bir kökü şu formülle bulunabilir:

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Yukarıdaki formülü kullanarak $x^2 - 4x + 3 = 0$ denkleminin bir kökünü hesaplayınız.

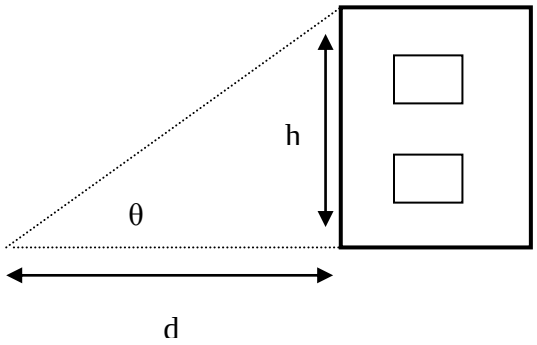
Çözüm 1.6

MATLAB kullanılarak bir kök şu şekilde hesaplanabilir:

```
>> a = 1; b = -4; c = 3;
>> x = (-b + sqrt(b*b – 4*a*c)) / (2*a)

x =

        3

>>
```

Örnek 1.7

Şekil 1.3 de gösterilen binanın yüksekliğini MATLAB kullanarak hesaplayınız. Açının 30º ve binaya olan uzaklığın ise 12 metre olduğunu kabul ediniz.



Şekil 1.3 Örnek 1.7

Çözüm 1.7

Binanın yüksekliği şu formülle hesaplanabilir:

$$\tan\theta = \frac{h}{d}$$

veya

$$h = d\,\tan\theta$$

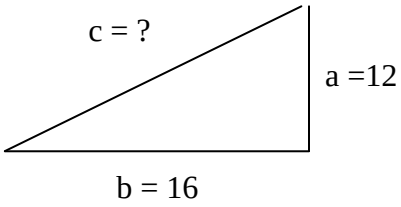
MATLAB'ı kullanarak yüksekliği şu şekilde hesaplayabiliriz:

```
>> h = 12*tan(30*pi / 180)

h =

        6.9282

>>
```

Yani binanın yüksekliği 6.9282 metre olarak hesaplanmıştır. Bu örnekte de göreceğimiz gibi, trigonometrik fonksiyonlar kullanılırken açının radyan olarak belirtilmesi gerekmektedir. Derece olan herhangibir açıyı radyana dönüştürmek için açıyı **pi/180** ile çarpmamız gerekmektedir.

Örnek 1.8

Şekil 1.4 de gösterilen ve bir dik kenarı 12cm ve diğer dik kenarı 16cm olan bir üçgenin üçüncü kenarının (hipotenüs) uzunluğunu hesaplayınız.



Şekil 1.4 Örnek 1.8

Çözüm 1.8

Pitagoras teoremini kullanarak hipotenüsü şu şekilde hesaplayabiliriz:

$$c = \sqrt{a^2 + b^2}$$

MATLAB ile hipotenüsü hesaplarsak

```
>> c = sqrt(12*12 + 16*16)

c =

        20

>>
```

olarak bulunur.

Örnek 1.9

Havaya fırlatılan bir havan topunun gideceği uzaklık şu formül ile hesaplanabilir:

$$d = \frac{V^2 \sin 2\theta}{g}$$

Burada d uzaklık, V topun ilk hızı, θ fırlatılan açı, ve g ise yer çekimi ivmesidir (9.8 m/s^2). Topun ilk hızının 100m/s ve açının 25º olduğunu kabul ederek topun gideceği uzaklığı hesaplayınız.

Çözüm 1.9

Formülü kullanarak topun gideceği uzaklığı şu şekilde hesaplayabiliriz:

```
>> v = 100;  theta = 25;  g = 9.8;
>> d = v*v*sin(2*theta*pi/180) / g

d =

        781.67800

>>
```

1.5 disp Komutu

Daha önce de gördüğümüz gibi satır sonuna "**;**" karakterini koymamakla herhangibir değişkenin değerini ekranda görebiliyoruz. Bu şekilde MATLAB değişkenin ismini yazar ve bir satır atlayarak değişkenin değerini yazar. Birçok durumlarda ekranda daha düzgün ve okunaklı bir çıkış isteriz. **disp** komutunu kullanarak ekrandaki çıkışı kontrol edebiliriz. Bu komutun genel şekli şöyledir:

```
disp(değişken)
```

Burada, değişken yerine herhangibir yazı yazmamız da mümkündür. Örneğin,

```
disp('MATLAB programı');
```

komutu ekrana şu satırı yazar:

```
MATLAB programı
```

Ayni şekilde,

```
disp(' ');
```

Komutu ekranda bir satır atlar. Bazı durumlarda birden fazla değişkeni ekranda ayni satırda göstermek isteriz ve bunun için değişkenleri **disp** komutu içerisinde ve köşeli parentezler arasında birer ara ile yazmalıyız. Örneğin, x,y, ve z değişkenlerini

```
disp( [x y z] );
```

Olarak ayni satırda gösterebiliriz. Ayni satırda yazı ve bir değişkenin değerini göstermek için ilk olarak değişkeni de yazı türüne dönüştürmemiz gerekir. Örneğin, **total** değişkeninin değeri 8 ise:

```
Disp( ['Toplam deger = ',num2str(total)] );
```

Komutu ekranda şu satırı gösterir:

```
Toplam deger = 8
```

Bu örnekte, **num2str** fonksiyonu herhangibir sayıyı yazı (string) şekline dönüştürür.

1.6 format Komutu

format komutu değişkenlerin ekranda nasıl gösterileceğini kontrol eder. Normal olarak tam sayılar çok büyük değillerse oldukları gibi gösterilir. Büyük olan tam sayılar ise kayma noktalı olarak ve noktadan sonra 4 rakam olacak şekilde gösterilir. Örneğin, 1234567890 sayısı 1.23456e+009 olarak gösterilir. 0.001 ve 1000 arasında olan ondalıklı sayılar da normal olarak noktadan sonra 4 rakam olarak gösterilirler. Örneğin, 4.5 sayısı 4.5000 olarak gösterilir. Ayni şekilde 12.34 sayısı 12.3400 olarak gösterilir. 0.001 den küçük ve 1000 den büyük ondalıklı sayılar ise noktadan sonra 4 rakamlı, noktadan önce 1 rakamlı, ve kayma nokta olarak gösterilirler. Örneğin, 1234.5 sayısı 1.2345e+003 olarak gösterilir. Tablo 1.4 de çeşitli sayıların ekranda gösterilişlerine örnekler verilmiştir.

Tablo 1.4 Çeşitli sayıların ekranda gösterilişi

Sayı	**Ekranda gösterilişi**
1200	1200
12345678	12345678
1234567890	1.2345e+009
23.5	23.5000
998.23	998.2300
1234.6	1.2346e+003
0.002	0.0020
0.0015	0.0015
0.0002	2.0000e-004

format komutunu kullanarak değişkenlerin ekranda gösteriliş formatlarını değiştirebiliriz. **format long** komutu değişkenleri noktadan sonra 14 rakamlı olarak gösterir. Ayni şekilde, **format bank** komutu noktadan sonra 2 rakam olarak gösterir. Değişkenin büyüklüğünden emin değilsek **format short g** veya **format long g** komutlarını kullanabiliriz. Tablo 1.5 de çeşitli format komutları verilmiştir.

Tablo 1.5 Çeşitli format komutları

format komutu	Tanımı	Örnek
format	Açılış formatına dön	
format short	Noktadan sonra 4 rakam	3.1416
format long	Noktadan sonra 14 rakam	3.14159265358979
format short e	Noktadan sonra 4 rakam kayma nokta	3.1416e+00
format long e	Noktadan sonra 14 rakam kayma nokta	3.141592653589793e+00
format short g	Noktadan sonra 4 rakam	3.1416
format long g	Noktadan sonra 14 rakam	3.14159265358979
format hex	hexadesimal	400921fb54442d18
format bank	Noktadan sonra 2 rakam	3.14
format compact	Fazlalık yeni satırları gösterme	
format loose	Fazlalık yeni satırları göster	

1.7 input Komutu

input komutu klavyeden bir sayı veya karakter okumak için kullanılır. Bu komutun iki şekli bulunmaktadır:

Klavyeden herhangibir sayı okumak için:

```
a = input('yazı')
```

ve klavyeden bir karakter okumak için ise

```
a = input('yazı', 's')
```

kullanılır. Komut ilk olarak yazı'yı ekranda gösterir ve kullanıcının klavyeden bir sayı veya karakter yazmasını bekler. Örneğin,

```
x = input('Bir sayı giriniz:')
```

komutu ile ekrana

```
Bir sayı giriniz:
```

Yazısı yazılır ve kullanıcının bir sayı girmesi beklenir. Yukarıdaki örnekte girilen sayı ise **x** gibi bir değişkende saklanır.

1.8 pause Komutu

pause komutunun genel şekli **pause(n)** olup bu komut programın n saniye kadar beklemesini sağlar. Eğer pause kendi başına kullanılmışsa, klavyeden herhangibir tuşa basılıncaya kadar program bekler.

Aşağıdaki örnekte program 3 saniye bekler:

```
pause(3)
```

1.9 Program Yazmak

Şimdiye kadar olan örneklerde MATLAB komutlarını direk olarak ekrana yazdık. Bu komutları bir program şeklinde toplayıp MATLAB'a vermemiz de mümkündür. MATLAB'ın program yazmak için çok geniş ve kapsamlı komutları bulunmaktadır. Bu bölümde sadece çok basit program yazma örneklerine bakacağız.

MATLAB programları m-dosyası olarak bilinirler. Bunun sebebi ise programların "**.m**" gibi bir uzantıyla saklanmış olmalarıdır. Yeni m-dosyası yaratmak için üç yol bulunmaktadır:

1. MATLAB içerisinde **edit** komutunu yazarak. Dosya hazır olunca **File -> Save** veya **File -> Save As** yolunu takip ederek dosyayı saklayabiliriz.
2. MATLAB içerisinde **File -> New -> M-file** yolunu takip ederek. Dosya hazır olunca **File -> Save** veya **File -> Save As** yolunu takip ederek dosyayı saklayabiliriz.

3. MATLAB dışında .m uzantılı dosya hazırlayarak

Bir m-dosyasını çalıştırmak için MATLAB çalışırken sadece dosyanın ismini yazmamız yererlidir. Aşağıda birkaç örnek verilmiştir.

Örnek 1.10

Klavyeden girecek olan iki sayıyı toplamak için bir MATLAB programı yazınız. Sayıları kullanıcıdan isteyiniz ve toplamı ekrana yazınız.

Çözüm 1.10

Yukarıda bahsedilen metodların birini kullanarak aşağıdaki programı yazınız ve programa **topla.m** ismini veriniz:

Program listesi aşağıda verilmiştir:

```
disp('Bu program 2 sayiyi toplar');
a = input('Birinci sayi : ');
b = input('Ikinci sayi   : ');
c = a + b;
disp(' ');
disp('Toplam:');
disp(c);
```

Programı çalıştırmak için **topla** komutunu giriniz:

```
>> topla
```

Program çalışınca ekranda şunları göreceksiniz:

```
Bu program 2 sayiyi toplar
Birinci sayi : 3
Ikinci sayi   : 5

Toplam:
        8
>>
```

Örnek 1.11

Klavyeden girilen herhangibir sayının kare kökünü hesaplayan bir MATLAB programı yazınız.

Çözüm 1.11

Programa **karekok** ismi verilmiş ve program listesi aşağıda gösterilmiştir:

```
% Bu program klavyeden girilen
% bir sayinin kare kokunu hesaplar
%
disp('Kare kok programi');                  % baslik
a = input('Sayiyi giriniz: ');              % sayiyi oku
b = sqrt(a);                                % kare kokunu hesapla
disp([num2str(a) ' nin kare koku = ' num2str(b)]);
```

Yukarıdaki programda da görüleceği gibi, açıklama satırları "**%**" karakteri ile başlamakta olup bu satırlara istediğimiz açıklamayı yazabiliriz. Programlarımızda açıklama satırları kullanmak son derece önemli olup bunu bir alışkanlık haline getirmeliyiz.

Programı çalıştırmak için

```
>>> karekok
```

yazınız. Program aşağıdaki işlemleri yapacaktır:

```
Kare kok programi
Sayiyi giriniz: 81
81 nin kare koku = 9
```

1.10 Çalışma Alanı

MATLAB çalışma alanı komutların yazıldığı ortamdır. Normal olarak kullanıcı tarafından yazılan değişkenler ve bu değişkenlerin değerleri bu ortamda saklanır. Çalışma alanı

içerisinde kullanıcının daha verimli çalışabilmesi için çeşitli yardımcı komutlar bulunmaktadır. Bu komutların bazıları aşağıda verilmiştir.

who komutu mevcut değişkenlerin isimlerini gösterir. Aşağıdaki örnekte *total, maksimum, x2* ve *sayac* olmak üzere 4 tane değişken tanımlanmıştır:

```
>> who

Your variables are:

total    maksimum    x2    sayac
```

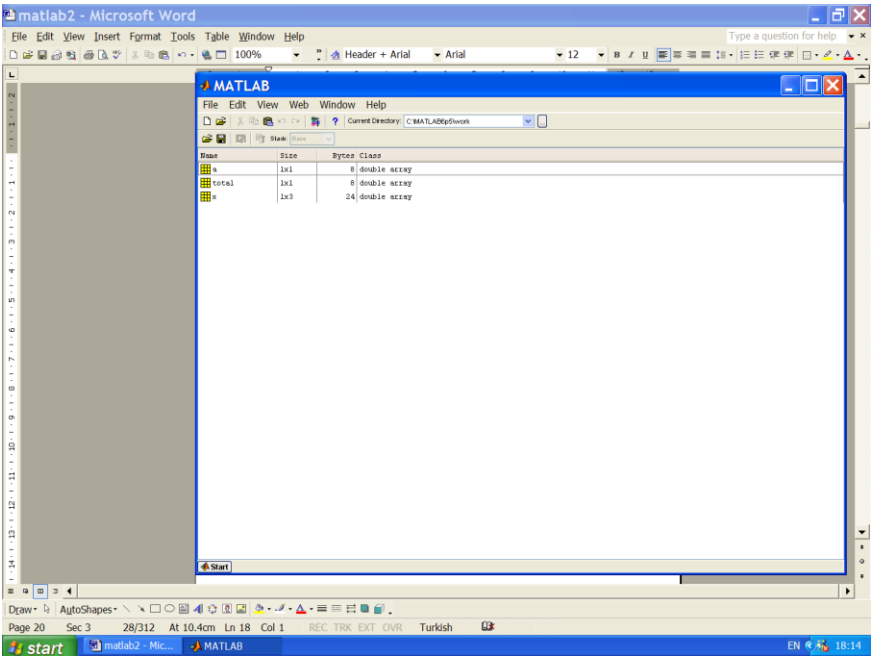
Ayni şekilde, **whos** komutu değişkenlerin isimlerini ve ilave olarak değişkenlerin çeşidi ve bellekte tutmuş oldukları yer hakkında bilgi verir. Örneğin:

```
>> whos
```

Name	Size	Bytes	Class
total	1x1	8	double array
maksimum	1x1	8	double array
x2	1x1	8	double array
sayac	1x1	8	double array

Bu örnekte her değişken 1 skalar büyüklüğünde olup bellekte 8 baytlık bir yer tutmaktadır.

Bellekteki değişkenleri görmenin başka bir yolu da MATLAB'ın *VIEW* menüsünü seçiniz ve oradan da *WORKSPACE* menüsünü seçiniz. Bellekteki değişkenler aşağıda gösterildiği gibi bir tablo şeklinde önünüze çıkacaktır:



clear komutu çalışma alanında olan bütün değişkenleri siler. Sadece bir veya birden fazla değişkeni silmek istiyorsak clear komutundan sonra değişkenlerin isimlerini bir ara vererek yazmamız gerekir. Aşağıdaki örnekte **sayac** değişkeni çalışma alanından silinir:

```
>> clear sayac
```

Ayni şekilde, aşağıdaki örnekte **total** ve **maksimum** değişkenleri çalışma alanından silinirler:

```
>> clear total maksimum
```

Çalışma alanında olan değişkenleri ve bu değişkenlerin değerlerini sabit diskimizde bir dosya içerisinde saklamak için **save** komutunu kullanabiliriz. Aşağıdaki örnekte bütün değişkenler **dosya.mat** isimli bir dosyada saklanırlar.

```
>> save dosya
```

Dosya ismi kullanıcının seçeceği herhangibir isim olabilir. Dosya uzantısı ise MATLAB tarafından otomatik olarak **.mat** olarak seçilir (dosya uzantısını değiştirmek mümkündür, fakat bu işlemin herhangibir avantajı olmadığı için tavsiye edilmemektedir). Sadece bir veya birkaç tane değişkeni dosyada saklamak için dosya adını ve saklamak istediğimiz değişkenleri liste olarak birer ara verip yazmamız gerekir. Aşağıdaki örnekte **a1, a2** ve **a3** değişkenleri **benim_dosya** isimli bir dosyada saklanırlar:

```
>> save benim_dosya a1 a2 a3
```

Burada önemli olan bir nokta, yaratılmış olan dosyanın MATLAB'a has bir dosya olmasıdır. Bu dosyayı herhangi başka bir programda kullanmak mümkün değildir.

Disk üzerinde saklanmış olan bir çalışma alanı dosyasını **load** komutunu kullanarak şimdiki çalışma alanımız haline getirebiliriz. Örneğin,

```
>> load benim_dosya
```

clc komutu çalışma alanını temizler ve bu durumda ekran boş olmuş olur.

home komutu kursoru çalışma alanında sol üst köşeye yerleştirir.

1.11 Komutlar Hakkında Yardım Almak

MATLAB'ı kullanırken bilmediğimiz herhangibir komut hakkında yardım alabiliriz. Bunun için ise **help** ve daha sonra komutun adını yazmamız gerekir. Örneğin, format komutu hakkında yardım almak için **help format** yazmamız gerekir. Bu komutun açıklması ise aşağıda gösterildiği gibidir:

```
>> help format
```

```
FORMAT Set output format.
   All computations in MATLAB are done in double precision.
   FORMAT may be used to switch between different output
   display formats as follows:
     FORMAT         Default. Same as SHORT.
     FORMAT SHORT   Scaled fixed point format with 5 digits.
     FORMAT LONG    Scaled fixed point format with 15 digits.
     FORMAT SHORT E Floating point format with 5 digits.
     FORMAT LONG E  Floating point format with 15 digits.
     FORMAT SHORT G Best of fixed or floating point  5 digits.
     FORMAT LONG G Best of fixed or floating point with 15 digits.
     FORMAT HEX     Hexadecimal format.
     FORMAT +       The symbols +, - and blank are printed
                    for positive, negative and zero elements.
                    Imaginary parts are ignored.
     FORMAT BANK    Fixed format for dollars and cents.
     FORMAT RAT     Approximation by ratio of small integers.

   Spacing:
     FORMAT COMPACT Suppress extra line-feeds.
     FORMAT LOOSE   Puts the extra line-feeds back in.

 Overloaded methods
   help quantizer/format.m

>>
```

MATLAB'da komutlar hakkında yardım almanın diğer bir yolu ise lookfor komutunu kullanmaktır. Lookfor, istenilen komutu içeren her MATLAB komutunun listesini verir. Örneğin,

```
>> lookfor sine
ACOS   Inverse cosine.
ACOSH  Inverse hyperbolic cosine.
ASIN   Inverse sine.
ASINH  Inverse hyperbolic sine.
COS    Cosine.
COSH   Hyperbolic cosine.
SIN    Sine.
SINH   Hyperbolic sine.
TFFUNC time and frequency domain versions of a cosine
modulated Gaussian pulse.
```

xfourier.m: %% Square Wave from Sine Waves
RCOSFIR Design a raised cosine FIR filter.
RCOSFLT Filter the input signal using a raised cosine filter.
RCOSIIR Design a raised cosine IIR filter.
RCOSINE Design raised cosine filter.
rcosdemo.m: %% Raised Cosine Filtering
DSPBLKSINE DSP Blockset Sine Wave block helper function.

>>

1.12 Sorular

1. Aşağıdaki işlemleri önce elde yapınız, sonra MATLAB kullanarak işlemlerin doğruluğunu kontrol ediniz.

   (a) 3 + 2*5 + 2
   (b) 12*2 + 4
   (c) 2 + 3^2 / 2
   (d) 23*2 + 4
   (e) 2 / 4 + 3

2. MATLAB kullanarak aşağıdaki işlemlerin cevabını bulunuz:

   (a) 3 ^ 2 + 1
   (b) $2 \times 10^2 + 3 \times 10^3$
   (c) $4.5 \times 10^{-2} - 4.5$
   (d) 100 x (2 + 4 / 3)

3. Santikrat (C) olarak verilen bir sıcaklığı şu formülü kullanarak Fahrenheit'a (F) dönüştürebiliriz:

   $$F = 9 \times C / 5 + 32$$

   MATLAB'ı kullanarak 80ºC sıcaklığı Fahrenheit'a dönüştürünüz.

4. Aşağıdaki değişkenlerden hangileri MATLAB

değişkeni olamaz, niçin ?

(a) 3,45 (b) 23a (c) a100 (d) 4.34e10
(e) a.4 (f) top sayı (g) a*2 (h) aZalT

5. Aşağıdaki MATLAB komutlarındaki hataları bulunuz.

(a) 23 = a (b) benim sayi = 120 (c) a+1 = a

6. Yarı çapı r ve yüksekliği h olan bir silindirinalanı şu formülle hesaplanmaktadır:

$$A = 2\pi rh$$

Yarı çapı 12cm ve yüksekliği 6 cm olan bir silindirin alanını MATLAB kullanarak hesaplayınız.

7. Şekil 1.3 de gösterilen metod kullanılarak bir ağacın yüksekliği ölçülmek istenmiştir. Ağaca 15 metre uzaktan ağacın en üst bölümü ile 48º bir açı olduğu gözlemlenmiştir. Ağacın yüksekliğini hesaplayınız.

8. Bir üçgenin alanını hesaplayacak MATLAB programını yazınız. Üçgenin tabanı ve yüksekliği klavyeden girilecektir.

9. Fahrenheit sıcaklığı klavyeden okuyan ve Santikrat sıcaklığa dönüştüren MATLAB programını yazınız.

10. **format bank** komutunu açıklayınız. Bu komut ne gibi işlemlerde kullanılabilir ?

11. Klavyeden girilen bir sayının karesini alıp ekranda gösteren bir MATLAB programı yazınız.

2
DİZİLER

2.1 Dizi Nedir ?

Diziler matematikte çok yaygın olarak kullanılmaktadırlar. Genel olarak diziler birden fazla sayının bir araya gelmesiyle meydana gelir. Diziler, vektörler ve matrisler olmak üzere genel olarak ikiye ayrılırlar. Vektör dizileri tek boyutlu dizilerdir. Matrisler ise iki veya daha çok boyutlu olabilirler. Bu bölümde sadece vektör dizilerinin MATLAB'da kullanımını inceleyeceğiz.

2.2 Vektör Dizileri

Aşağıda 5 elemanı olan **x** isimli bir vektör dizisi verilmiştir:

```
x = [1 3 5 7 9]
```

veya

```
x = [1,3,5,7,9]
```

Bu dizi ilk 5 tek sayıyı ihtiva etmektedir. Dizi elemanları köşeli parentez içerisinde olup her eleman arasında bir ara (veya virgül) olmalıdır.

Vektör dizileri üzerine çeşitli matematik işlemleri yapmak mümkündür. Bazı örnekler aşağıda verilmiştir.

2.2.1 Bir Skalar ile İşlem

Herhangibir vektör ve bir skalar arasında toplama, çıkarma, çarpma, bölme ve diğer matematik işlemleri yapabiliriz. Bu işlemlerde vektörün her elemanı skalar ile işlem görür.

Vektör	İşlem	Netice
x = [1 3 5 7 9]	y = x + 2	y = [2 5 7 9 11]
x = [1 3 5 7 9]	y = 2 * x	y = [2 6 10 14 18]
x = [1 3 5 7 9]	y = x – 1	y = [0 2 4 6 8]
x = [2 4 6 8 10]	y = x / 2	y = [1 2 3 4 5]

Vektörler ile matematiksel fonksiyonları kullanarak işlem yapmak da mümkündür.

Örnek 2.1

Birden 10 a kadar olan çift sayıların kare köklerini bulunuz.

Çözüm 2.1

Sayıları x isimli bir vektör olarak gösterirsek

```
x = [2 4 6 8 10]
```

Kare kökünü şu şekilde hesaplayabiliriz

```
y = sqrt(x)
```

y vektörü şu değerleri alır:

```
y = [1.4142    2.0000    2.4495    2.8284    3.1623]
```

Örnek 2.2

0 dan 90 dereceye kadar olan açıların 10 derece ara ile

sinüslerini hesaplayınız.

Çözüm 2.2

Açıları **x** isimli bir vektörde saklarsak:

```
x = [0 10 20 30 40 50 60 70 80 90]
```

Bu açıların sinüslerini şu şekilde hesaplayabiliriz:

```
s = sin(pi / 180*x)
```

Açılar pi/180 ile çarpılıp radyan cinsine dönüştürülmüştür. Daha sonra açıların sinüsü alınmıştır. **s** vektörü şu değerleri alır:

```
s = [0    0.1736    0.3420    0.5000    0.6428    0.7660
     0.8660    0.9397    0.9848    1.0000]
```

2.2.2 Vektör elemanlarının Otomatik olarak Seçilmesi

Yukarıdaki örnekte de görüleceği gibi vektör elemanlarını kendimiz seçtik. MATLAB ile vektör elemanlarını otomatik olarak seçmek mümkündür. Bunun için ise ilk elemanı, elemanlar arası adım miktarını, ve son elemanı belirtmemiz gerekir:

```
Vektör = ilk eleman:adım:son eleman
```

Aşağıdaki örnekte, 0 dan 10 a kadar, 2 şer ara ile sayılar ihtiva eden **x** isimli bir vektör yaratılmıştır:

```
>> x = 0:2:10

x =
        0  2  4  6  8  10
>>
```

Ayni şekilde, 10 dan 0 a kadar 2 şer olarak azalan bir vektörü

şu şekilde yapabiliriz:

```
x = 10:-2:0

x =
        10  8  6  4  2  0
>>
```

Vektör elemanlarını yaratırken son eleman ikinci "**:**" karakterden sonra olan sayıdan büyük olamaz. Örneğin,

```
x = 1:2:6
```

Komutu, x = [1 3 5] vektörünü yaratır.

Vektör elemanlarını yaratırken eğer adım'ı belirtmezsek otomatik olarak 1 alınır:

```
x = 1:10
```

Komutu, x = [1 2 3 4 5 6 7 8 9 10] vektörünü yaratır.

Vektör elemanlarını otomatik olarak yaratarak çok çeşitli tablolar oluşturabiliriz. Bazı örnekler aşağıda verilmiştir.

Örnek 2.3

1 den 5 e kadar olan sayıların kare köklerini bulunuz.

Çözüm 2.3

X isimli bir vektör yaratarak 1 den 5 e kadar olan sayıların kare köklerini şu şekilde bulabiliriz:

```
x = sqrt(1:5)
```

x vektörü şu değerleri alır:

```
x = [1.0000    1.4142    1.7321    2.0000    2.2361]
```

Örnek 2.4

0 dan 50 dereceye kadar olan açıların 10 ar derece ara ile sünüslerini hesaplayınız.

Çözüm 2.4

x isimli bir vektör yaratarak 0 dan 50 dereceye kadar olan açıların sinüslerini şu şekilde bulabiliriz:

```
x = sin(pi/180*(0:10:50))
```

x vektörü şu değerleri alır:

```
x = [0    0.1736    0.3420    0.5000    0.6428    0.7660]
```

Örnek 2.5

Yukarıdaki örneği değiştirerek açıları ve sinüslerini bir tablo şeklinde gösteriniz.

Çözüm 2.5

Açıları ve sinüslerini bir tablo şeklinde göstermek için ilk olarak açıları bir vektörde saklamamız gerekir. Daha sonra bu açıların sinüslerini bulup bir tablo şeklinde gösterebiliriz. Bu örnek için gerekli adımlar aşağıda gösterilmiştir:

```
> x = 0:10:50;
>> sinus = [x'  sin(pi/180*x)']

sinus =

       0         0
  10.0000    0.1736
  20.0000    0.3420
  30.0000    0.5000
  40.0000    0.6428
```

```
    50.0000    0.7660

  >>
```

Bu örnekte açılar ve sinüsleri gösterilirken vektörler transpoz yapılmıştır. Transpoz yapmakla sıra şeklinde olan bir vektör sütun şekline dönüşür. Transpoz yapmak için ise vektörün isminden sonra ‘ karakteri kullanılır.

Örnek 2.6

Yukarıdaki örneği bir MATLAB m-dosyası şeklinde yazınız ve tabloya başlık veriniz. İlk ve son açıları klavyeden okuyunuz. Programa **sinus.m** ismini veriniz.

Çözüm 2.6

Gerekli olan **sinus.m** dosyası aşağıda verilmiştir:

```
%
% Bu program ilk ve son acilari verilen
% ve 10 derece olarak artan acilarin
% sinuslerini hesaplar ve bir tablo seklinde gosterır
%
disp('SINUS TABLOSU');
disp('------------------------');
disp(' ');
ilk = input('Ilk aci : ');
son = input('Son aci : ');
x = ilk:10:son;
sinus = [x' sin(pi/180*x)'];
disp(' ');
disp('    ACI      SINUS');
disp(sinus);
```

Programı çalıştırmak için >>> den sonra **sinus** yazınız. 10 dereceden 90 dereceye kadar olan açıların sinüslerini hesaplayan örnek aşağıda verilmiştir:

```
SINUS TABLOSU
------------------------
```

```
Ilk aci : 10
Son aci : 90

    ACI      SINUS
  10.0000   0.1736
  20.0000   0.3420
  30.0000   0.5000
  40.0000   0.6428
  50.0000   0.7660
  60.0000   0.8660
  70.0000   0.9397
  80.0000   0.9848
  90.0000   1.0000
```

Vektör elemanı yaratmanın başka bir yolu ise **linspace** komutunu kullanmaktır. Bu komut, ilk değeri, son değeri, ve eleman sayısını alır ve gerekli olan elemanları yaratır. Aşağıdaki örnekte 1 den 10 a kadar eşit aralıklarla 5 tane eleman yaratılmıştır:

```
>> x = linspace(1,10,5)

x =
    1.0000    3.2500    5.5000    7.7500   10.0000
>>
```

Örnek 2.7

Havaya fırlatılan bir havan topunun gideceği uzaklık topun ilk hızına ve atılım açısına bağlı olarak şu şekilde değişir:

$$d = \frac{V^2 \sin 2\theta}{g}$$

Burada d uzaklık, V topun ilk hızı, θ fırlatılan açı, ve g ise yer çekimi ivmesidir (9.8 m/s^2). Topun ilk hızının 100m/s olduğunu kabul edersek, açı 0º den 50º ye kadar 5 er derece aralıklarla değiştikçe uzaklığı bir tablo şeklinde gösteriniz.

Çözüm 2.7

MATLAB komutları ve netice aşağıda verilmiştir:

```
>> g=9.8;
>> v=10;
>> theta=0:5:50;
>> d=v*v*sin(2*theta*pi/180)/g;
>> disp([theta'   d'])

      0         0
 5.0000    1.7719
10.0000    3.4900
15.0000    5.1020
20.0000    6.5591
25.0000    7.8168
30.0000    8.8370
35.0000    9.5887
40.0000   10.0491
45.0000   10.2041
50.0000   10.0491
```

2.2.3 Diğer Vektörlerle İşlem

Yukarıdaki örneklerde bir vektör ve skalar arasındaki işlemler incelenmiştir. Bu bölümde birden fazla vektör arasındaki işlemler göz önünde bulundurulacaktır.

Vektörler arasındaki işlemler için vektör operatörlerini kullanmamız gerekir. Tablo 2.1 de vektör aritmetik operatörleri verilmiştir. Burada, operatörden önce bir nokta kullanıldığına dikkat ediniz.

Tablo 2.1 Vektör aritmetik operatörleri

Operatör	**Tanımı**
.*	Vektör çarpımı
./	Vektör bölümü
.\	Ters vektör bölümü
.^	Vektör güç operatörü
+	Vektör toplama
-	Vektör çıkarma

Vektör aritmetik operatörleri verilen iki vektörün karşılıklı elemanlarını alıp gereken işlemi yapar. Bazı örnekler aşağıda verilmiştir.

Bu örnek için, vektörlerin **a** ve **b** olduklarını ve

```
a = [1 2 3 4 5]
b = [2 4 6 8 10]
```

olarak kabul edersek, aşağıdaki vektör işlemlerini yapabiliriz:

İşlem	Netice
a .* b	[2 8 18 32 50]
a ./ b	[.5000 0.5000 0.5000 0.5000 0.5000]
a .\ b	[2 2 2 2 2]
a .^ b	[1 16 729 65536 9765625]
a + b	[3 6 9 12 15]
a – b	[-1 -2 -3 -4 -5]
b – a	[1 2 3 4 5]

Örnek 2.8

Pazarda satılan meyvelerin fiyatının şu şekilde olduğunu kabul edelim:

Meyve	**Fiyatı ($ kilosu)**
Elma	2.5
Armut	3
Banana	4
Seftalı	1.75
Kıraz	2.8
Kayısı	3.4

Pazardan 3 kilo elma, 5 kilo armut, 2 kilo banana, 1 kilo kiraz ve 4 kilo kayısı aldığımızı kabul edersek toplam tutarını hesaplayınız.

Çözüm 2.8

Birim fiyatlarını x gibi bir vektörde ve almış olduğumuz miktarları da y gibi bir vektörde saklarsak:

```
x = [2.5  3  4  1.75  2.8  3.4]
y = [3  5  2  0  1  4]
```

Toplam tutarı bulmak için iki vektörü çarpıp neticeyi toplamamız gerekir:

```
toplam =sum(x .* y)

toplam =

    46.9000
>>
```

olarak hesaplanır. Bu örnekte kullanılan MATLAB **sum** fonkiyonu bir vektörün bütün elemanlarının toplamını verir.

Bir vektörün elemanları 1 den başlayarak ve parentez kullanılarak indekslenir. Örneğin, a = [4 6 9] vektörünü göz önünde bulundurursak, vektörün elemanlarını şu şekilde seçebiliriz:

a(1) birinci eleman (4)
a(2) ikinci eleman (6)
a(3) üçüncü eleman (9)

Örneğin, a(1) + a(3) elemanlarının toplamı 13 olur.

Bir vektörün sadece iki indeks arasında olan elemenlarını istersek şu şekilde yazabilirik:

```
x(n:m)
```

Burada n başlangıç indeksi, m ise bitiş indeksidir. Örneğin, x = [2 4 6 8 10 12] vektörünü göz önünde bulundurursak,

x(2:4) dizisi, [4 6 8] elemanlarını ihtiva eder.

2.2.4 find Komutu

find komutunu kullanarak herhangibir vektör içerisinde belirli şartlara uyan elemanları bulabiliriz. Örneğin,

```
>> x = 0:2:10

x =
        0    2  4  6  8  10
>>
```

vektörünü göz önünde bulunduralım. Bu vektörün 5 den büyük olan elemanlarının indekslerini şu şekilde bulabiliriz:

```
>> z = find(x > 5)

z =
        4   5   6
>>
```

olur. Yani, **x** vektörünün 4, 5 ve 6 indeksli elemanları 5 den büyüktür.

2.2.5 length Komutu

length komutu bir vektörün eleman sayısını verir. Yukarıdaki örnekte, z vektörünün 3 elemanı vardır ve **length(z)** komutu 3 sayısını verir.

2.2.6 Vektör Karşılaştırması

MATLAB'ı kullanarak iki vektörü karşılaştırmamız mümkündür. MATLAB **isequal** fonksiyonu verilen iki vektörü karşılaştırır ve vektörler eşit ise 1, eşit değilse 0 verir. Örneğin,

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

```
>> B = [1 2 3];
>> isequal(a,b)

ans =
        1
>>
```

Ayni şekilde,

```
>> a = [1 2 3];
>> b = [1 2 4];
>> isequal(a,b)

ans =
        0
>>
```

olur.

2.2.7 Vektörlerin Bellekte Saklanması

MATLAB'da bir vektörün her elemanı 8 bayt olarak saklanır. Bu durumda,10 elemanlı bir vektör bellekte 80 bayt yer tutar. Örnek olarak 10 elemanlı, **x** isimli bir vektör yaratalım:

```
>> x = 1:10

x =

   1    2    3    4    5    6    7    8    9   10

>>
```

Vektörün hafızada tuttuğu yeri görmek için **whos** komutunu kullanabiliriz:

```
>> whos
Name      Size                Bytes  Class

  x         1x10                 80  double array

Grand total is 10 elements using 80 bytes
```

>>

2.2.8 max Fonksiyonu

max fonksiyonu bir vektör içerisindeki en büyük elemanı verir. Örneğin,

a = [1 3 5 12 9]

ise

b = max(a) komutu b değişkenini 12 yapar.

Vektör içerisinde en büyük olan sayının indeksini istersek şu komutu yazabiliriz:

[x,k] = max(a)

burada **x** 12 değerini, **k** ise 4 değerini alır (**x** vektörü içerisinde en büyük olan sayının indeksi 4 dür).

2.2.9 min Fonksiyonu

min fonksiyonu bir vektör içerisindeki en büyük elemanı verir. Örneğin,

a = [8 3 2 12 9]

ise

b = min(a) komutu b değişkenini 2 yapar.

Vektör içerisinde en küçük olan sayının indeksini istersek şu komutu yazabiliriz:

[x,k] = min(a)

burada **x** 2 değerini, **k** ise 3 değerini alır (**x** vektörü içerisinde en küçük olan sayının indeksi 3 dür).

2.2.10 sort Fonksiyonu

sort fonksiyonu bir vektör içerisindeki elemanları küçükten büyüğe olmak üzere sıralar. Örneğin,

```
a = [2 7 8 2 5 3 1]
```

ise

```
b = sort(a)
```

Fonksiyonu b vektörüne şu değerleri verir:

```
b = [1 2 3 5 7 8]
```

2.2.11 mean Fonksiyonu

mean fonksiyonu bir vektör elemanlarının ortalamasını bulur. Örneğin,

```
a = [1 3 5]
```

ise

```
b = mean(a)
```

komutu **b** değişkenine 3 değerini verir. (1 + 3 + 5) / 3 = 3.

Örnek 2.9

1 den 10 a kadar olan sayıların ortalamasını bulunuz.

Çözüm 2.9

Sayıları **x** gibi bir vektöre yükleyip **mean** komutunu kullanarak ortalamayı bulabiliriz:

```
x = 1:10;
ortalama = mean(x)
```

Burada **ortalama** değişkeni 5.5000 değerini alır. Bu örneği şu şekilde de çözebiliriz:

```
x = 1:10;
ortalama = sum(x) / 10
```

2.2.12 Kompleks Sayılar

Kompleks sayılar bilhassa mühendislik matematiğinde çok yaygın olarak kullanılmaktadır. Genel olarak bir kompleks sayının gerçek (real) ve gerçek olmayan (imaginary) iki bölümü bulunmaktadır. Gerçek olmayan bölümün önüne i veya j karakteri konmaktadır. Örneğin,

```
2 + 4i  veya 2 + 4j
```

MATLAB'ı kullanarak kompleks sayılar üzerine işlemler yapabiliriz. Örneğin sqrt(4 + 4i) komutu 2.1974 + 0.9102i cevabını verir. Vektör işlemlerinde kompleks sayı kullanmamız mümkündür. Örneğin,

```
A = [2+2i  3+3i]
```

ise

```
b = 2*a  işlemi b vektörüne şu değerleri verir:

b = [4+4i  6+6i]
```

Bir kompleks sayının mutlak değerini bulmak için o sayının gerçek ve gerçek olmayan bölümlerinin karesini alıp toplamamız, ve bulduğumuz sayının kare kökünü almamız gerekir. Örneğin,

```
x + yi
```

sayısının mutlak değeri $\sqrt{x^2 + y^2}$ olarak bulunur. MATLAB

kullanarak bir kompleks sayının mutlak değerini bulmak için **abs** fonksiyonunu kullanabiliriz. Örneğin,

a = [2+4i]

ise

b = abs(a)

komutu **b** değişkenine 4.4721 değerini verir ($\sqrt{2^2+4^2} = 4.4721$).

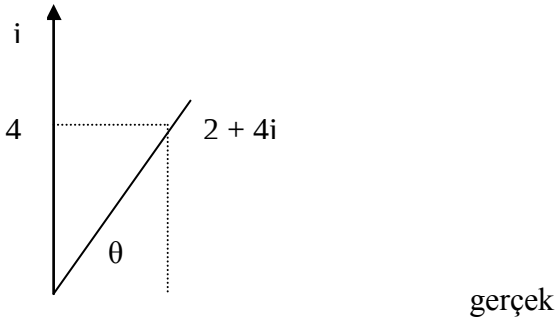
real(z) fonksiyonu z kompleks sayısının gerçek bölümünü verir. Ayni şekilde, **imag(z)** fonksiyonu z kompleks sayısının gerçek olmayan bölümünü verir. **conj(z)** fonksiyonu z kompleks sayısının konjugeytini verir (gerçek olmayan bölümün işareti değişmiş). **angle(z)** fonksiyonu ise, Şekil 2.1 de gösterildiği gibi **z** kompleks sayısının gerçek olmayan ve gerçek olan bölümleri arasındaki açıyı verir.

Kompleks sayı fonksiyonları ile ilgili örnekler aşağıda verilmiştir:

z = 2 + 4i

ise

real(z) fonksiyonu 2 değerini verir, **imag(z)** fonksiyonu 4 değerini verir, **conj(z)** fonksiyonu 2 – 4i değerini verir, ve **angle(z)** fonksiyonu açıyı 1.071 radyan olarak verir (1.071 radyan = 1.071*180/pi = 61.3638º).



2

Şekil 2.1 Kompleks sayı, 2 + 4i

2.3 Programlamaya Devam

Kitabın birinci bölümünde MATLAB'ı kullanmak için program yapmaya başlamıştık. Bu bölümde programlamaya devam edip daha ileri programlama konularını inceleyeceğiz.

2.3.1 Programda Karar Vermek

Karar verme her çeşit programda önemli bir kontrol yapısıdır. MATLAB'ı programlarken karar vermek için **if** komutu kullanılır. Bu komut sadece bir satırlık olabilir veya birden fazla satırda kullanılabilir.

Tek satırda if komutu şu şekilde kullanılır:

if *durum komut*, end

Bu örnekte *durum* doğru ise *komut* işlemi yapılır, doğru değilse yapılmaz.

Örneğin,

```
x = 1;
If(x == 1) disp('merhaba'), end
```

Komutu, x değişkeninin değeri 1 olduğu için ekrana **merhaba** yazar.

MATLAB'da karşılaştırma yapmak için Tablo 2.2 de gösterilen operatörler kullanılabilir.

Tablo 2.2 Karşılaştırma operatörleri

Operatör	**Tanımı**

<	Küçükse
<=	Küçük veya eşitse
==	Eşitse
>	Büyükse
~=	Eşit değilse
>=	Büyük veya eşitse

if komutu genellikle programın kolay okunması açısından birden fazla satıra yazılabilir:

```
if durum
        komutlar
end
```

Örneğin, yukarıdaki işlemi şu şekilde yazmak tercih edilmektedir:

```
x = 1;
if(x == 1)
        disp('merhaba')
end
```

Yukarıdaki örnekte durum doğru ise verilen komut işleme girer, durum doğru değilse herhangibir işlem yapılmaz. Birçok uygulamalarda durum doğru ise bir takım işlemler, doğru değilse de başka işlemler yapmak isteriz. Bu tür uygulamalarda, if komutunun şu şeklini kullanabiliriz:

```
İf durum
        Komutlar
else
        Komutlar
end
```

Bazı uygulamalarda ise değişik durumlara göre değişik işlemler yapmak isteriz. Bu tür uygulamalarda **if** komutunun genel şeklini kullanabiliriz:

```
If durum1
        Komutlar
```

```
elseif durum2
        Komutlar
elseif durum3
        Komutlar
…….
else
        Komutlar
end
```

if komutunun kullanımı için örnekler aşağıda verilmiştir.

Örnek 2.10

İki sayı üzerine toplama, çıkarma, çarpma, ve bölme işlemleri yapmak için menü tabanlı ve **hesap_makinesi.m** isimli bir MATLAB programı yazınız. Program sayıları klavyeden okuyacak ve kullanıcının isteğine göre gereken işlemi yapacaktır.

Çözüm 2.10

Program listesi aşağıda verilmiştir:

```
% BASIT HESAP MAKINESI PROGRAMI
% ------------------------------------------------
%
% Bu program klavyeden 2 tane sayi okur.
% Ayni zamanda klavyeden yapilmasi gereken
% islem MENU olarak okunur. MENU da toplama,
% cikarma, carpma ve bolme islemleri bulunmaktadir.
% Program istenilen islemi yapar ve neticeyi ekranda
% gosterir.
%
clc
disp('Hesap Makinesi Programi');
disp('====================');
disp(' ');
birinci = input('Ilk sayiyi giriniz: ');
ikinci = input('Ikinci sayiyi giriniz: ');
disp(' ');
disp('  1. Toplama');
```

```
disp('  2. Cikarma');
disp('  3. Carpma');
disp('  4. Bolme');
disp(' ');
istenen = input('Bir opsiyon seciniz [1-4]: ');
if(istenen == 1)
   netice = birinci + ikinci;
elseif(istenen == 2)
   netice = birinci - ikinci;
elseif(istenen == 3)
   netice = birinci * ikinci;
else(istenen == 4)
   netice = birinci / ikinci;
end

disp(' ');
disp(['netice = ',num2str(netice)]);
```

Programı çalıştırmak için >> komutundan sonra **hesap_makinesi** yazmanız yeterlidir:

Programın çalıştırılmasına örnek aşağıda verilmiştir:

```
Hesap Makinesi Programi
=======================

Ilk sayiyi giriniz: 23
Ikinci sayiyi giriniz: 2

  1. Toplama
  2. Cikarma
  3. Carpma
  4. Bolme

Bir opsiyon seciniz [1-4]: 3

netice = 46
```

2.3.2 Programda Döngü Yaratmak

MATLAB'ı kullanırken programda döngü yapmak oldukça kolaydır. Bunun için normal olarak **for** komutunu kullanırız. Bu komutun genel şekli şöyledir:

```
for x = dizi
        Komutlar
end
```

Burada, **for** ve **end** arasındaki komutlar bütün dizi için yapılmaktadır. Örneğin, komutları 10 defa yapmak istiyorsak şu döngüyü yapmamız gerekir:

```
for x = 1:10
        Komutlar
end
```

Örnek 2.11

1 den 10 a kadar olan sayıların karelerini gösteren ve **for** döngüsünü kullanan kare isimli bir MATLAB programı yazınız.

Çözüm 2.11

İstenilen programın listesi aşağıda verilmiştir:

```
%
% 1 den 10 a kadar olan sayilarin karesi.
% Bu program for dongusunu kullanir
%
format compact
disp('1 den 10 a kadar olan sayilarin kareleri');
for n=1:10
   disp([n,n*n]);
end
```

Program çalıştırılınca ekranda 1 den 10 a kadar olan sayıların karesi bir tablo şeklinde gösterilir:

```
>> kare
```

```
1 den 10 a kadar olan sayilarin kareleri
    1     1
    2     4
    3     9
    4    16
    5    25
    6    36
    7    49
    8    64
    9    81
   10   100
>>
```

for komutu ile döngü yaratırken bazı uygulamalarda döngü adımının 1 den değişik bir sayı olmasını isteyebiliriz. Bir örnek aşağıda verilmiştir.

Örnek 2.12

0 dan 20 ye kadar olan ve 2 olarak artan sayıların karelerini bir tablo şeklinde gösteren bir program yazınız.

Çözüm 2.12

Bu örnek 2.11 e çok benzemekte olup tek farkı for döngüsünde kullanılan dizinin 0 dan 20 ye kadar, 2 aralıklarla artmış olmasıdır.

Program listesi aşağıda verilmiştir:

```
%
% 1 den 20 ye kadar olan ve 2 olarak artan sayilarin karesi.
% Bu program for dongusunu kullanir
%
format compact
disp('0 dan 20 ye kadar olan sayilarin kareleri');
for n=0:2:20
   disp([n,n*n]);
end
```

Program çalıştırılınca ekranda şu tablo gösterilir:

```
>> kare
0 dan 20 ye kadar olan sayilarin kareleri
    0    0
    2    4
    4   16
    6   36
    8   64
   10  100
   12  144
   14  196
   16  256
   18  324
   20  400
>>
```

MATLAB programlarında döngü yaratmanın bir başka şekli de **while** komutunu kullanaraktır. Bu komutun genel şekli şöyledir:

```
while durum
        Komutlar
end
```

Burada, durum doğru olduğu müddetçe **while** ve **end** arasındaki komutlar işlem görmektedir. Tabii ki döngüden çıkmak için **while** komutundan sonra belirtilen durumun döngü içerisinde saplanması gerekmektedir. Aksi halde sonsuz bir döngü elde etmiş oluruz.

while komutuna bir örnek aşağıda verilmiştir:

Örnek 2.13

Örnek 2.10 da verilen hesap makinesi programı bir defa çalıştırıldıktan sonra durur. Bu programı kullanıcı isteğine göre durması için değiştiriniz.

Çözüm 2.13

Yeni programda **while** komutu kullanılarak kullanıcının devam etmek isteyip istememesi kontrol edilmektedir. Programın başında **tekrar** isimli bir değişken EVET kelimesinin 'E' karakterine eşitlenir ve **tekrar** değişkeni 'E' karakterine eşit olduğu müddetçe program döngüsü devam eder. Programın sonunda kullanıcıya devam edip etmemesi sorulur ve kullanıcı 'E' yazdığı müddetçe program devam eder. Kullanıcının 'H' veya başka bir karakter yazması halinde program durur.

Program listesi aşağıda verilmiştir:

```
BASIT HESAP MAKINESI PROGRAMI
% ------------------------------------------------
%
% Bu program klavyeden 2 tane sayi okur.
% Ayni zamanda klavyeden yapilmasi gereken
% islem MENU olarak okunur. MENU da toplama,
% cikarma, carpma ve bolme islemleri bulunmaktadir.
% Program istenilen islemi yapar ve neticeyi ekranda
% gosterir.
% Program kullaniciya bagli olarak durur.
%
tekrar = 'E';
while(tekrar == 'E')
clc
disp('Hesap Makinesi Programi');
disp('==================');
disp(' ');
birinci = input('Ilk sayiyi giriniz: ');
ikinci = input('Ikinci sayiyi giriniz: ');
disp(' ');
disp('  1. Toplama');
disp('  2. Cikarma');
disp('  3. Carpma');
disp('  4. Bolme');
disp(' ');
istenen = input('Bir opsiyon seciniz [1-4]: ');
if(istenen == 1)
   netice = birinci + ikinci;
```

```
elseif(istenen == 2)
   netice = birinci - ikinci;
elseif(istenen == 3)
   netice = birinci * ikinci;
else(istenen == 4)
   netice = birinci / ikinci;
end

disp(' ');
disp(['netice = ',num2str(netice)]);
disp(' ');
tekrar = input('Baska islem yapmak istermisiniz [E-H]: ','s');
end
```

Programın çalışma örneği aşağıda verilmiştir:

```
Hesap Makinesi Programi
=======================

Ilk sayiyi giriniz: 4
Ikinci sayiyi giriniz: 4

  1. Toplama
  2. Cikarma
  3. Carpma
  4. Bolme

Bir opsiyon seciniz [1-4]: 1

netice = 8

Baska islem yapmak istermisiniz [E-H]: H
>>
```

2.3.3 switch Komutu

Bazı uygulamalarda birden fazla durum bulunmakta ve bu durumlara göre değişik işlemler yapmamız gerekmektedir. Bu tip programlar genellikle if-elseif-end komutları kullanılarak yapılabilir. **switch** komutu daha basit ve daha okunaklı olduğu için tercih edilmelidir. **switch** komutunun genel şekli şöyledir:

```
switch değişken
        case durum1
                Komutlar
        case durum2
                Komutlar
        case durum3
                Komutlar
        ctherwise
                Komutlar
end
```

Burada değişken durum1 e eşitse birinci komutlar yapılır, değişken durum2 ye eşitse ikinci komutlar yapılır ve böylece devam eder. Eğer değişken hiçbir duruma eşit değilse **otherwise** dan sonra gelen komutlar işlem görürler.

switch komutu için bir örnek aşağıda verilmiştir.

Örnek 2.14

Örnek 2.10 da verilen hesap makinesi programını **switch** komutu kullanarak tekrar yapınız.

Çözüm 2.14

İstenilen program listesi aşağıda verilmiştir. Burada kullanıcı seçeneği **switch** komutu ile kontrol edilmektedir, ve bu komutun değişkenine **istenen** ismi verilmiştir. Eğer **istenen** 1 ise toplama işlemi yapılır, **istenen** 2 ise çıkarma işlemi yapılır, **istenen** 3 ise çarpma işlemi yapılır ve son olarak **istenen** 4 ise bölme işlemi yapılır.

```
% BASIT HESAP MAKINESI PROGRAMI
% ------------------------------------------------
%
% Bu program klavyeden 2 tane sayi okur.
% Ayni zamanda klavyeden yapilmasi gereken
% islem MENU olarak okunur. MENU da toplama,
% cikarma, carpma ve bolme islemleri bulunmaktadir.
```

```
% Program istenilen islemi yapar ve neticeyi ekranda
% gosterir.
% Bu program switch komutunu kullanmaktadir
%
clc
disp('Hesap Makinesi Programi');
disp('====================');
disp(' ');
birinci = input('Ilk sayiyi giriniz: ');
ikinci = input('Ikinci sayiyi giriniz: ');
disp(' ');
disp('  1. Toplama');
disp('  2. Cikarma');
disp('  3. Carpma');
disp('  4. Bolme');
disp(' ');
istenen = input('Bir opsiyon seciniz [1-4]: ');
switch istenen
   case 1
      netice = birinci + ikinci;
   case 2
      netice = birinci - ikinci;
   case 3
      netice = birinci * ikinci;
   case 4
      netice = birinci / ikinci;
end

disp(' ');
disp(['netice = ',num2str(netice)]);
disp(' ');
```

Programın çalışmasına örnek aşağıda verilmiştir:

```
Hesap Makinesi Programi
====================

Ilk sayiyi giriniz: 2
Ikinci sayiyi giriniz: 6

  1. Toplama
  2. Cikarma
```

3. Carpma
4. Bolme

Bir opsiyon seciniz [1-4]: 3

netice = 12

2.4 fprintf Komutu

MATLAB programlarında ekrana yazmak için disp komutunun kullanımını gördük. *disp* komutu oldukça basit olup kompleks ekrana yazı işlemlerinde yetersiz kalmaktadır. *fprintf* komutu ekrana yazmak için kullanılan çok daha genel ve daha güçlü bir komuttur. *fprintf* komutunu kullanarak ekrana herçeşit yazı ve sayı yazmamız oldukça kolaydır.

Aşağıdaki örnekte ekrana Bu Bir Test Mesajidir… yazısı yazılmaktadır:

```
fprintf('Bu Bir Test Mesajidir…');
```

Yeni satır için \n karakterleri kullanılır. Aşağıdaki örnekte ekrana şu satırlar yazılmaktadır:

```
Satir 1
Satir 2
Satir 3

fprintf('Satir1\nSatir2\nSatir3\n');
```

Ekrana herhangibir değişkenin değerini yazmak için format karakterleri kullanmamız gerekmektedir. En yaygın olarak kullanılan format karakterleri aşağıda verilmiştir:

Format karakteri	Tanımı

%c	karakter
%d	tam sayı (işaretli)
%f	kayan nokta sayı
%g	kayan nokta sayı (sıfırlar atılmış)
%s	karakter dizisi
%u	tam sayı (işaretsiz)
%x	heksadesimal sayı

Aşağıdaki örnekte x sayısının değeri tam sayı olarak ekrana yazılmıştır:

```
x = 12;
fprintf('x sayisi = %d',x);
```

Program çalışınca ekranda şunları görürüz:

```
x sayisi = 12
```

Ayni şekilde, kayma nokta bir sayı için:

```
toplam = 23.34;
fprintf('toplam = %f',toplam);
```

Komutları ekranda şunları gösterir:

```
toplam = 23.340000
```

Eğer %g formatını kullanırsak gereksiz olan sıfırları ekrandan kaldırmış oluruz:

```
fprintf('toplam = %g',toplam);
```

Komutu ekranda

```
toplam = 23.34
```

yazısını gösterir.

fprintf komutunun kullanımına bir örnek aşağıda verilmiştir.

Örnek 2.15

kare.m isimli bir MATLAB programı yazınız ve bu programla ekrandan bir tam sayı okuyup bu sayının karesini hesaplayınız ve cevabı ekrana şu şekilde yazınız (burada n ekrandan okunan sayıyı, m ise bu sayının karesini belirtmektedir):

n sayısının karesi = m

Çözüm 2.15

Gerekli olan program listesi aşağıda verilmiştir. Programda *input* komutu kullanılarak herhangibir sayı okunur, daha sonra bu sayının karesi hesaplanır ve *fprintf* komutu kullanılarak ekranda istenilen şekilde gösterilir:

```
% Bu program ekrandan okunan bir sayinin karesini
% hesaplar ve cevabi ekranda gosterir.
%
n = input('Sayiyi giriniz: ');
m = n*n;
fprintf('%d sayisinin karesi = %d',n,m);
```

Programın çalışmasına bir örnek aşağıda verilmiştir:

```
>> kare
Sayiyi giriniz: 3
3 sayisinin karesi = 9
>>
```

2.5 Kullanıcı Fonksiyonları

Fonksiyonlar programlarımızda sık sık kullandığımız alt programlardır. Örneğin, programımızda bir takım sayıların ortalamalarını sık sık hesaplamak istiyorsak bu ortalama

işlemini bir fonksiyon (veya alt program) olarak yazabiliriz. Daha sonra programımızda bu fonksiyonu kullanıp istediğimiz sayıların ortalamalarını kolaylıkla hesaplayabiliriz. Fonksiyonların genel olarak şu avantajları vardır:

- Fonksiyon sadece bir defa yazılır.
- Fonksiyon hatasız olarak çalışırsa bu fonksiyonu her kullanışımız da hatasız olacaktır.
- Yazmış olduğumuz fonksiyonu diğer programlarımızda da kullanabiliriz.
- Yazmış olduğumuz fonksiyonu başkalarına da verebiliriz.

MATLAB programı ile birlikte çok çeşitli fonksiyonlar verilmektedir. Bu bölümde sadece kullanıcı (veya programcı) tarafından yazılan fonksiyonları inceleyeceğiz.

2.5.1 Bir Fonksiyonun Yapısı

Bir fonksiyon genel olarak şu şekilde yazılabilir:

```
Function [x, y, z] = fonksiyon_adı (giriş argümanları)
% fonksiyon hakkında bilgiler
%
        .....
        ......
        ..... Fonksiyonun algoritması
        ......
        ......
y = dönüş_değeri
```

Her fonksiyon, *function* kelimesi ile başlamalıdır. Bunu takiben fonksiyonun dönüş değişkeni, veya birden fazla ise dönüş değişkenleri bir köşeli parentez içerisinde belirtilmelidir. Fonksiyon geri ana programa dönünce burada belirtilen değerler ile dönecektir. Bazı fonksiyonlarda dönüş değişkenleri olmayabilir. Bu durumlarda dönüş değişkeni ve eşit işareti gerekmez (veya içerisi boş bir köşeli parentez ve eşit işareti kullanılabilir). Daha sonra fonksiyonun adı ve

parentez içerisinde fonksiyonun giriş argümanları belirtilmelidir. Fonksiyonun adı bir karakter ile başlamalı ve bu karakterden sonra istenilen karakterler veya sayılar kullanılabilir. Giriş argümanları ana program tarafından fonksiyona verilen değerleri içermektedir. Bazı fonksiyonlarda giriş argümanı olmayabilir. Bu durumlarda sağ tarafta sadece fonksiyonun adı belirtilmelidir.

Aşağıda bazı örnekler verilmiştir.

Örnek 2.16

Verilen iki sayının ortalamasını bulan ve *ortalama2* diye isimlendirilen bir MATLAB fonksiyonu yazınız. Bu fonksiyonu *oku* isimli bir MATLAB programında kullanarak klavyeden 2 sayı okuyunuz ve bu sayıların ortalamasını ekranda gösteriniz.

Çözüm 2.16

İlk olarak *ortalama2.m* isimli bir dosya yaratırız. Bu dosyanın içindekiler aşağıda verilmiştir:

```
% Bu fonksiyon verilen 2 sayinin ortalamasini hesaplar
%
function y = ortalama2(a, b)
y = (a + b) / 2;
```

Daha sonra *oku.m* isimli ana programımızı yaratırız. Ana programımızda şu satırlar bulunmaktadır:

```
% Bu program klavyeden 2 sayi okur ve ortalama2
% fonksiyonunu kullanarak bu sayilarin ortalamasini
% hesaplar.
%
x = input('Birinci sayi: ');
y = input('Ikinci sayi: ');
z = ortalama2(x, y);
fprintf('Ortalama = %g\n',z);
```

Programı çalıştırmak için oku yazarız. Ana programımız klavyeden 2 sayı okur ve daha sonra ortalama2 fonksiyonuna çağırıp bu sayıların ortalamasını hesaplar. Bulunan netice fprintf komutu ile ekranda gösterilir:

```
>> oku
Birinci sayi: 2
Ikinci sayi: 6
Ortalama = 4
>>
```

Örnek 2.17

ortalama.m diye adlandırılan ve bir vektör olarak verilen sayıların ortalamasını hesaplayan bir fonksiyon yazınız. Bu fonksiyonun kullanımını aşağıdaki sayıların ortalamasını hesaplayarak gösteriniz:

2 3 5 7 8 9

Çözüm 2.17

ortalama.m fonksiyonunun listesi aşağıda verilmiştir. Fonksiyonun tanımından sonra ilk olarak verilen vektörün eleman sayısı hesaplanır ve eğer bu sayı 1 ise (verilen değerler bir vektör değilse) fonksiyon hata verir ve durur. Aksi halde, MATLAB'in *sum* fonksiyonu kullanılarak verilen sayıların toplamı hesaplanır ve eleman sayısına bölünerek istenilen ortalama hesaplanmış olur:

```
% Bu fonksiyon bir vektor olarak verilen sayilarin
% ortalamasini hesaplar
%
function y = ortalama(x)
[d, element_sayisi] = size(x);
if element_sayisi == 1
   disp('Giris vektor olmalidir...');
else
```

```
    y = sum(x) / element_sayisi;
end
```

Yukarıda verilen sayıların ortalamasını şu şekilde hesaplayabiliriz:

```
>> t = [2 3 5 7 8 9];
>> ortalama(t)

ans =

    5.6667

>>
```

Şimdi de birden fazla dönüş veren fonksiyon örneklerine göz atalım.

Örnek 2.18

Bir silindirin alanını ve hacmini hesaplayan ve *silindir* diye adlandırılan bir MATLAB fonksiyonu yazınız. Fonksiyonun kullanımını *ornek.m* isimli bir program yazarak gösteriniz. Programınızda silindirin yarı-çapını ve yüksekliğini klavyeden okuyunuz.

Çözüm 2.18

silindir.m fonksiyonunun listesi aşağıda verilmiştir. Fonksiyon silindirin yarı-çapını (r) ve yüksekliğini (h) giriş olarak alır ve silindirin alanını (A) ve hacmini (V) dönüş değerleri olarak verir:

```
% Bu fonksiyon bir silindirin alanini ve hacmini hesaplar
%
function [A,V] = silindir(r,h)
A = 2*pi*r*h;
V = pi*r*r*h;
```

ornek.m ana programının listesi de aşağıda verilmiştir. Burada silindirin yarı-çapı ve yüksekliği klavyeden okunur.

Daha sonra program silindir fonksiyonunu çağırarak silindirin alanını ve hacmini hesaplar. Bulunan neticeler ise *fprintf* fonksiyonu ile ekrana yazılırlar:

```
% Bu program verilen bir silindirin alanini ve hacmini
% hesaplar. Silindirin yaricapi ve yuksekligi klavyeden
% okunur.
%
disp('BIR SILINDIRIN ALANI VE HACMI');
disp('==========================');
yaricap = input('Silindirin yaricapi: ');
yukseklik = input('Silindirin yuksekligi: ');
[alan,hacim] = silindir(yaricap,yukseklik);
fprintf('Alan = %g    hacim = %g\n', alan,hacim);
```

Programın çalışmasına örnek aşağıda verilmiştir:

```
>> ornek
BIR SILINDIRIN ALANI VE HACMI
==========================
Silindirin yaricapi: 2
Silindirin yuksekligi: 5
Alan = 62.8319    hacim = 62.8319
>>
```

2.5.2 Fonksiyonda Kullanılan local ve global Değişkenler

Herhangibir fonksiyon içerisinde kullanılan değişkenler sadece o fonksiyona ait olup bu değişkenlerin değerleri fonksiyon dışında bilinmemektedir. Ayni şekilde, bir fonksiyon dışında kullanılan değişkenler fonksiyon içerisinde ayni ismi taşısalar bile değerleri bilinmemektedir. Bu durumda, fonksiyon içerisinde ve fonksiyon dışında ayni değişken isimleri kullanılmışsa bu değişkenler birbirleri ile ayni değillerdir. MATLAB'da (eğer değiştirilmemişse) normal olarak bütün değişkenler bu kurala uyarlar ve *local* değişken olarak bilinirler.

MATLAB'da *global* kelimesi kullanılarak herhangibir değişkenin bütün MATLAB içerisinde ayni değeri alması

sağlanabilir. *global* değişkenler, diğer değişkenlerden ayırt edilebilmeleri için hep büyük harfle yazılmaları tavsiye edilmektedir. MATLAB'da *who global* komutu *global* değişkenlerin listesini verir. *clear global* komutu ise bütün *global* değişkenleri *local* yapar. Herhangibir *global* değişkeni *local* yapmak istersek *clear* komutunu ve değişkenin adını yazabiliriz. MATLAB fonksiyonu *isglobal(X)* herhangibir değişkenin *global* olup olmadığını test etmek için kullanılabilir. Örneğin, X değişkeni *global* ise *isglobal(X)* fonksiyonu 1 olur, *global* değilse 0 olur.rilmiştir.

Aşağıda, *global* değişkenlerin kullanımına bir örnek verilmiştir.

Örnek 2.19

Bir üçgenin alanının hesaplayan ve *alan* diye adlandırılan bir fonksiyon yazınız. Üçenin tabanını (TABAN) ve yüksekliğini (YUKSEKLIK) global olarak alınız. Tabanı ve yüksekliği ana programda klavyeden okuyunuz. Ana programı *ucgen.m* isimli bir dosyada saklayınız.

Çözüm 2.19

Alan fonksiyonunun listesi aşağıda verilmiştir. Fonksiyonda TABAN ve YUKSEKLIK değişkenleri global olarak tanımlanmıştır.

```
% Bu fonksiyon bir ucgenin alanini hesaplar
% taban (TABAN) ve yukseklik (YUKSEKLIK)
% global olarak tanimlanmistir.
%
function a = alan
global TABAN YUKSEKLIK
a = TABAN*YUKSEKLIK/2;
```

ucgen.m dosyasının listesi aşağıda verilmiştir. TABAN ve YUKSEKLIK değişkenleri global olarak tanımlanmıştır:

```
% Bu program verilen bir ucgenin alanini hesaplar.
%
global TABAN YUKSEKLIK
disp('BIR UCGENIN ALANI');
disp('================');
TABAN = input('Ucgenin tabani: ');
YUKSEKLIK = input('Ucgenin yuksekligi: ');
alan = ortalama2;
fprintf('Alan = %g\n',alan);
```

Programın çalışmasına bir örnek aşağıda verilmiştir:

```
>> ucgen
BIR UCGENIN ALANI
================
Ucgenin tabani: 3
Ucgenin yuksekligi: 5
Alan = 7.5
```

2.5.3 return Komutu

Normal olarak bir fonksiyon çağrıldığında fonksiyonun sonuna kadar işlem yapar ve daha sonra ana programa döner. Bazı durumlarda fonksiyonun sonuna gelmeden fonksiyondan çıkmak isteyebiliriz ve bunun için de fonksiyonun istediğimiz yerinde *return* komutunu kullanabiliriz.

2.5.4 Bir Fonksiyonun Argüman Sayısı

Bazı durumlarda bir fonksiyonun kaç tane giriş ve kaç tane de çıkış argümanı olduğunu bilmek isteriz. *nargin* komutu bir fonksiyonun kaç tane giriş elemanı olduğunu gösterir. Ayni şekilde, *nargout* komutu bir fonksiyonun kaç tane çıkış argümanı olduğunu belirtir. *nargin* ve *nargout* komutlarını kullanarak daha kullanışlı fonksiyonlar yazmamız mümkündür.

Aşağıdaki örneklerde *nargin* ve *nargout* komutlarının kullanılışı gösterilmiştir.

Örnek 2.20

Paralel bağlanmış dirençlerin toplam direncini hesaplamak için pdirenc isimli bir fonksiyon yazınız. Fonksiyonunuz 5 taneye kadar paralel bağlanmış direncin toplam direncini hesaplamalıdır.

2 tane paralel bağlanmış direncin toplam direnci şu şekilde hesaplanır:

$$1/R_p = 1/R_1 + 1/R_2$$

Ayni şekilde, n tane bağlanmış paralel direncin toplam direnci şu şekilde hesaplanır:

$$1/R_p = 1/R_1 + 1/R_2 + \ldots\ldots + 1/R_n$$

Çözüm 2.20

İstenilen fonksiyonun listesi aşağıda verilmiştir. Fonksiyonun 5 taneye kadar giriş argümanı olabilir. Fonksiyonun başında *nargin* komutu kullanılarak giriş argüman sayısı bulunmuştur. Daha sonra argüman sayısına bağlı olarak toplam paralel direnç bulunmuştur:

```
% Bu fonksiyon 5 taneye kadar paralel baglanmis
% direnclerin toplam direncini hesaplar. Fonksiyon
% giris arguman sayisina gore islem yapmaktadir.
%
function rp = pdirenc(R1, R2, R3, R4, R5)
switch nargin
   case 1
      rp = R1;
   case 2
```

```
    rp = (1/R1) + (1/R2);
    rp = 1/rp;
  case 3
    rp = (1/R1) + (1/R2) + (1/R3);
    rp = 1/rp;
  case 4
    rp = (1/R1) + (1/R2) + (1/R3) + (1/R4);
    rp = 1/rp;
  case 5
    rp = (1/R1) + (1/R2) + (1/R3) + (1/R4) + (1/R5);
    rp = 1/rp;
end
```

Programın çalışmasına bir örnek aşağıda verilmiştir. Bu örnekte değerleri 2 ohm, 4 ohm, ve 6 ohm olan 3 tane direnç paralel olarak bağlanmıştır. Toplam direnç 1.0909 ohm olarak bulunmuştur:

```
>> pdirenc(2,4,6)

ans =

1.0909

>>
```

2.5.5 Kendi Kendine Çağıran (Recursive) Fonksiyonlar

Recursive fonksiyonlar matematiksel işlemlerde kullanılmaktadırlar. Örneğin, n gibi bir sayının faktöryelini şu şekilde hesaplayabiliriz:

$$n! = n \times (n - 1)!$$

Yukarıdaki faktöryel işlemi bir MATLAB recursive fonksiyonu olarak şu şekilde hesaplanabilir:

```
function y = fact(n)
if n > 1
        y = n*fact(n – 1);
```

```
else
        y = 1;
end
```

Recursive fonksiyon programlaması oldukça ileri bir konu olup bu kitapta daha fazla bahsedilmemektedir.

2.6 Sorular

1. Aşağıdaki vektörün elemanlarının toplamını bulunuz:

   ```
   a = [2 4 6 8 10 12]
   ```

2. İlk elemanı 1 ve son elemanı 100 olan ve elemanları ikişer olarak artan **x** isimli bir vektör yaratınız.

3. Soru 2 de yaratmış olduğunuz vektör elemanlarının kare köklerini bulunuz.

4. Aşağıdaki iki vektörün toplamlarını ve çarpımlarını bulunuz:

   ```
   a = [6 5 4 3 2 1]
   b = [1 3 5 7 9 11]
   ```

5. Aşağıdaki kompleks sayı işlemlerini MATLAB kullanarak yapınız. w = 5 + 8i ve z = *3 + 12i olarak kabul ediniz.

(a) w + z	(b) z + w	(c) w – z
(d) z – w	(e) w * z	(f) w / z

6. **for** döngüsü kullanarak $3k^3$ serisinin, k 1 den 10 a kadar değiştikce aldığı değerleri hesaplayınız.

7. Bir MATLAB programı yazıp aşağıdaki ikinci derece denklemin köklerini bulunuz:

$ax^2 + bx + c = 0$

8. Soru 7 deki ikinci derece denklemin köklerini bukmak İçin genel bir MATLAB programı yazınız. a, b, ve c Katsayılarını klavyeden okuyunuz ve denklemin köklerini ekranda gösteriniz.

9. 0 dan 90 dereceye kadar olan açıların sinüslerini ve kosinüslerini hesplayıp bir tablo şeklinde gösteriniz.

10. Aşağıdaki programda q değişkeninin değeri nedir ?

```
q = 0;
for k = 1:2:10
        q = q + 2
end
```

11. 1 den n'e kadar olan tam sayıların ortalamasını hesaplayacak bir program yazınız. n sayısı klavyeden girilecektir.

12. x = [10 8 6 4 2] ve y = [-5 5 –3 3 2] olduğunu kabul edersek, aşağıdaki işlemlerin cevapları nedir ?

(a) z = (x > y) (b) z = (x == y)
(c) z = (x < y) (d) z = (x ~= y)

13. Aşağıdaki tabloda bir fabrikada çalışan bir işçinin bir hafta içerisinde hergün çalışmış olduğu toplam saat verilmiştir. işçinin saatlik ücretinin $10 olduğunu kabul edersek işçinin haftalık maaşını hesaplayınız.

Fabrikada çalışan herhangibir işçinin haftalık maaşını hesaplayacak bir MATLAB programı yazınız.

günler	**Çalışılan saat**
Pazar	5
Pazartesi	6.5
Salı	7

Çarşamba	8
Perşembe	7.5
Cuma	6
Cumartesi	0

14. 0 dan 100 derece Santikrata kadar olan sıcaklıları fahrenheit a dönüştüren ve **for** döngüsü kullanan bir program yazınız.

15. Soru 14 deki programı **while** döngüsü kullanacak şekilde değiştiriniz.

16. Aşağıda verilen vektörlerin toplamını bulunuz:

    a = [2 4 6 8]
    b = [1 3 5 7]
    c = [0 –3 5 12]

17. Klavyeden okunan bir sayının karesini ve küpünü hesaplayan bir MATLAB programı yazınız. Programın girişi şu şekilde olacaktır:

    Sayiyi giriniz:

    Programın çıkışı ise şu şekilde olacaktır:

    n sayisinin karesi = m ve kupu = p

18. Klavyeden bir üçgenin tabanını ve yüksekliğini okuyunuz. Daha sonra bu üçgenin alanını şu şekilde ekranda gösteriniz:

    Tabanı n ve yukseklıgı m olan ucgenın alanı = p

19. Yarı-çapı ve yüksekliği verilen bir silindirin hacmini hesaplayan bir fonksiyon yazınız.

20. Soru 19 da yazdığınız fonksiyonu test etmek için bir program yazınız. Programınızda silindirin yarı-çapını

ve yüksekliğini klavyeden okuyunuz.

21. Bir kenarı verilen bir karenin alanını ve çevresini hesaplayan bir fonksiyon yazınız.

22. Verilen 3 tane sayının ortalamasını, en küçüğünü, ve en büyüğünü hesaplayan bir fonksiyon yazınız.

23. İki kenarı verilen bir dikdörtgenin alanını ve çevresini hesaplayan bir fonksiyon yazınız.

24. 1 den n e kadar olan sayıların toplamını, ve karelerinin toplamlarını hesaplayan bşr fonksiyon yazınız.

25. Soru 24 de yazdığınız fonksiyonu test etmek için bir program yazınız.

3
POLİNOMLAR

Matlab ile çok değişik polinom işlemleri yapmak mümkündür. Bu bölümde önemli polinom işlemlerini inceleyeceğiz.

MATLAB'da polinomlar vektör gibi yazılırlar ve polinomun en büyük kuvveti ilk yazılır. Örneğin,

$$6x^3 + 2x^2 + 4x + 5$$

gibi bir polinom şu şekilde yazılabilir:

```
f = [6 2 4 5]
```

eğer polinomun herhangibir kuvveti eksikse bu kuvvetin katsayısı için 0 koymak gerekir. Örneğin,

$$9x^3 + 3x^2 + 12$$

polinomu şu şekilde yazılabilir:

```
f = [9 3 0 12]
```

3.1 Polinom Çarpımı

Polinom çarpımı iki şekilde olabilir: bir polinomun bir skalar ile çarpımı, veya bir polinomun başka bir polinom ile çarpımı.

Herhangibir polinomu bir skalar ile çarpmak için polinomun her elemanını bu skalar ile çarpmamız gerekir. Örneğin,

$2x^3 + 4x^2 + 3x – 5$

polinomunu 2 ile şu şekilde çarpabiliriz:

```
f = [2 4 3 –5];
g = 2*f
```

elde ettiğimi yeni polinom **g** nin şu elemanları olur:

```
g = [4 8 6 –10]
```

veya

$g = 4x^3 + 8x^2 + 6x – 10$

Herhangibir polinomu başka bir polinom ile çarpmak için **conv** fonksiyonunu kullanmamız gerekir. Örneğin, aşağıdaki **f1** ve **f2** polinomlarını göz önünde bulundurursak,

$f1 = 2x^2 + 4x + 5$
$f2 = x^2 – 2x + 4$

bu iki polinomun çarpımını (g = f1 x f2) şu şekilde hesaplayabiliriz:

```
f1 = [2 4 5];
f2 = [1 –2 4];
g = conv(f1, f2)
```

Bu örnekte **g** vektörünün şu elemanları olur:

```
g = [2    0    5    6    20]
```

veya

$g = 2x^4 + 5x^2 + 6x + 20$

olur. Yani,

$$(2x^2 + 4x + 5)(x^2 – 2x + 4) = 2x^4 + 5x^2 + 6x + 20$$

İki polinomu çarparken polinomların ayni derecede olmaları gerekmez. Aşağıdaki örnekte iki değişik derecede polinomun çarpımı verilmiştir:

$f1 = 3x^2 + x + 4$
$f2 = 2x + 1$

Bu iki polinom şu şekilde çarpılabilir:

```
f1 = [3 1 4];
f2 = [2 1];
g=conv(f1,f2)
```

Burada **g** vektörünün elemanları

```
g = [6 5 9 4]
```

olur. Yani,

$g = 6x^3 + 5x^2 + 9x + 4$

3.2 Polinom Toplamı

Polinom toplaması için herhangibir özel komut yoktur. Toplanacak olan her iki polinomun da ayni derecede olmaları gerekir. Eğer herhangibir polinomun herhangibir kuvveti eksik ise o kuvvetin katsayısı 0 olarak yazılmalıdır.

Aşağıdaki örnekte f1 ve f2 isimli iki polinom toplanır ve netice **g** isimli bir polinomda saklanır:

$f1 = 2x^3 + x^2 – 5x + 12$
$f2 = 6x^3 + 3x^2 + x + 8$

```
f1 = [2 1 –5 12];
f2 = [6 3 1 8];
g = f1 + f2
```

Burada **g** vektörünün şu elemanları vardır:

```
g = [8 4 –4 20]
```

Böylece, toplam polinom şu olur:

$$g = 8x^3 + 4x^2 – 4x + 20$$

3.3 Polinom Çıkarması

Polinom çıkarması için herhangibir özel komut yoktur. Çıkarılacak olan her iki polinomun da ayni derecede olmaları gerekir. Eğer herhangibir polinomun herhangibir kuvveti eksik ise o kuvvetin katsayısı 0 olarak yazılmalıdır.

Aşağıdaki örnekte f2 polinomu f1 den çıkarılır ve netice g gibi bir polinomda saklanır:

$$f1 = 2x^3 + x^2 – 5x + 12$$
$$f2 = 6x^3 + 3x^2 + x + 8$$

```
f1 = [2 1 –5 12];
f2 = [6 3 1 8];
g = f1 - f2
```

Burada g vektörünün şu elemanları vardır:

```
g = [-4 -2 –6 4]
```

Böylece, fark polinom şu olur:

$$g = -4x^3 - 2x^2 – 6x + 4$$

3.4 Polinom Bölümü

Polinom bölümü için **deconv** fonksiyonu kullanılır. Bu fonksiyon hem bölümü ve hem de kalanı hesaplar.

Aşağıdaki örnekte f2 polinomu f1 polinomuna bölünür ne netice **g** gibi bir polinomda saklanır Bölümden kalan ise **r**

polinomunda saklanır.

$f1 = 6x^3 + 3x^2 + 4x + 8$
$f2 = 2x^3 + x^2 – 5x + 12$

```
f1 = [6 3 4 8];
f2 = [2 1 -5 12];
[g,r] = deconv(f1,f2)
```

Burada **g** vektörünün şu elemanları vardır:

```
g = [3]
```

r vektörünün ise:

```
r = [0     0    19   -28]
```

Böylece,

```
g = 3
r = 19x –28
```

olur.

3.5 Bir Polinomun Değeri

MATLAB'I kullanarak herhangibir polinomun değerini hesaplayabiliriz. **polyval(a,x)** fonksiyonu, katsayıları **a** vektöründe bulunan bir polinomun verilen **x** noktasındaki değerini hesaplar. Burada **x** bir vektör olabilir. **polyval** fonksiyonunu kullanan bazı örnekler aşağıda verilmiştir.

Örnek 3.1

Aşağıda verilen polinomun x = 2 olduğunda değerini bulunuz.

$f(x) = 2x^3 + x^2 – 2x + 4$

Çözüm 3.1

```
>> f = [2 1 –2 4];
>> polyval(f, 2)

ans =

   20
>>
```

olur.

Yukarıdaki örneği şu şekilde de çözebiliriz:

```
>> poyval([2 1 –2 4], 2)

ans =

   20
>>
```

Örnek 3.2

Aşağıda verilen polinomun x = 0 2 4 6 8 10 olduğundaki değerlerini bulunuz.

$$f(x) = 2x^3 + x^2 + x + 4$$

Çözüm 3.2

```
>> x = 0:2:10;
>> y = polyval([2 1 1 4], x)

y =
   4    26   152   478   1100   2114

>>
```

olur.

Örnek 3.3

Aşağıdaki polinom çarpımının x = 0, 2 ve 4 olduğunda

değerlerini bulunuz.

$$f(x) = (2x^3 + x^2 + x + 4)(x^2 + x + 1)$$

Çözüm 3.3

İlk olarak iki polinomu çarpabiliriz, daha sonra da verilen x noktalarındaki değerlerini bulabiliriz.

```
> f = conv([2 1 1 4], [1 1 1])

f =
    2     3     4     6     5     4

>> y = polyval(f, [0 2 4])

y =
         4        182       3192

>>
```

Polinomun değerleri şu şekilde bulunmuş olur:

x = 0 y = 4
x = 2 y = 182
x = 4 y = 3192

Örnek 3.4

x 1 den 10 a kadar değiştikçe aşağıdaki polinomun değerlerini hesaplayıp bir tablo şeklinde gösterecek MATLAB programını yazınız.

$$f(x) = 2x^3 + x^2 + x + 4$$

Çözüm 3.4

İstenilen program listesi aşağıda verilmiştir:

```
%
% POLINOM DEGERLERI
% Bu program 2x3 + x2 + x + 4 polinomunun
```

```
% x 1 den 10 a kadar degistikce degerlerini
% hesaplar ve bir tablo seklinde ekranda gosterir
%
x = 1:10;
f = [2 1 1 4];
y = polyval(f, x)
disp('Polinomun degerleri');
disp('===============');
disp('          x          y');
disp([x', y']);
```

Programı çalıştırınca aşağıdaki tabloyu ekranda görürüz:

```
Polinomun degerleri
===============
          x          y
          1          8
          2         26
          3         70
          4        152
          5        284
          6        478
          7        746
          8       1100
          9       1552
         10       2114
```

3.6 Polinom Denklemlerin Kökleri

MATLAB **roots** fonksiyonunu kullanarak hergangibir polinom denklemin köklerini bulabiliriz. Bu fonksiyon mühendislik uygulamalarında son derece önem taşıyan bir fonksiyondur. Bazı örnekler aşağıda verilmiştir.

Örnek 3.5

Aşağıdaki denklemin köklerini bulunuz:

$$x^2 - 7x + 12 = 0$$

Çözüm 3.5

roots fonksiyonunu kullanarak denklemin köklerini kolaylıkla bulabiliriz:

```
>> f = [1 –7 12];
>> roots(f)

ans =
        3
        4
>>
```

Denklemin kökleri x = 3 ve x = 4 olarak bulunur.

Yukarıdaki işlemi şu şekilde de yapabiliriz:

```
x = roots([1 –7 12])
x =
        3
        4
>>
```

Örnek 3.6

Aşağıdaki denklemin köklerini bulunuz:

$3x^3 + 2x^2 + 7x + 4 = 0$

Çözüm 3.6

roots fonksiyonunu kullanarak denklemin köklerini kolaylıkla bulabiliriz:

```
>> roots([3 2 7 4])
ans=

-0.0416 + 1.5110i
-0.0416 - 1.5110i
-0.5836
```

olur. Bu denklemin 2 tane kompleks kökü ve bir tane de gerçek kökü vardır. Kompleks kökleri x = -0.0416 + 1.5110i ve x = -0.0416 – 1.5110i, gerçek kökü ise x = -0.5836 dır.

Örnek 3.7

Aşağıdaki denklemde K katsayısı 0 dan 4 e değiştikçe denklemin köklerini hesaplayan MATLAB programını yazınız:

$$x^3 + 2x^2 - 7x + K = 0$$

Çözüm 3.7

K katsayısını değişirmek için bir vektör kullanabiliriz. İstenilen program listesi aşağıda verilmiştir:

```
%
% Bu program x3 + 2x2 -7x +K denkleminin
% K katsayisi 0 dan 4 e degistikce koklerini
% bulur
%
for K=0:4
   f = [1 2 -7 K];
   y = roots(f);
   disp(['K = ', num2str(K)]);
   disp(y)
end
```

Program çalıştırılınca, K katsayısının değerine göre kökleri şu şekilde buluruz:

```
K = 0
        0
   -3.8284
    1.8284

K = 1
   -3.8737
    1.7240
    0.1497
```

```
K = 2
   -3.9173
    1.5977
    0.3196

K = 3
   -3.9593
    1.4292
    0.5302

K = 4
  -4.0000
   1.0000 + 0.0000i
   1.0000 - 0.0000i
```

3.7 Kökleri Verilen Polinomun Yazılımı

MATLAB **poly** komutunu kullanarak kökleri verilen bir polinomu yazabiliriz. Örneğin, kökleri x = 2 ve x = 3 olan polinomun yazılımı şu şekilde bulunabilir:

```
>> a=[2 3];
>> poly(a)

ans =
    1    -5     6
>>
```

İstenilen polinom, $f(x) = x^2 - 5x + 6$ olur.

Örnek 3.8

Kökleri x = 2, x = 5, x = 3 ve x = 6 olan dördüncü derece polinomu bulunuz.

Çözüm 3.8

poly komutunu kullanarak,

```
>> a=[2 5 3 1];
>> poly(a)

ans =

    1   -11    41   -61    30
>>
```

İstenilen polinom, $f(x) = x^4 - 11x^3 + 41x^2 - 61x + 30$

olur.

3.8 Sorular

1. Aşağıda verilen iki polinomun toplamını bulunuz:

   $f1(x) = 3x^3 + 4x^2 - 2x + 12$
   $f2(x) = -2x^3 + x^2 - 3x - 8$

2. Soru 1 de verilen polinomların çarpımını hesaplayınız.

3. Aşağıdaki iki polinomu toplayınız:

   $f1(x) = x^3 + x^2 - 2x + 5$
   $f2(x) = x^2 + x - 2$

4. Soru 1 de verilen polinomların bölümünü hesaplayınız.

5. Aşağıda verilen f(x) polinomunun x = 2 olduğunda değerini bulunuz:

   $f(x) = x^3 + 2x^2 - 2x + 1$

6. Aşağıdaki polinomun x = 2 ve x = 4 olduğunda değerini hesaplayınız:

$$f(x) = (x^3 + x^2 - x + 2)(2x^3 + x^2 - 3x + 4)$$

7. Aşağıdaki polinomun x = 3 olduğunda değerini hesaplayınız:

$$f(x) = (2x^3 + x^2 + x + 4) / (x^3 + x^2 + 5x + 2)$$

8. Aşağıdaki polinom denklemlerin bütün köklerini bulunuz:

(a) $2x^2 - 3x + 1 = 0$
(b) $x^3 - 4x^2 - 2x + 1 = 0$
(c) $2x^4 - 5x^3 + 2x^2 - 2x + 1 = 0$
(d) $(2x^2 - 2x + 1)(x^2 - 4x + 2) = 0$

9. Kökleri aşağıda verilen polinom denklemleri yazınız:

(a) x = 2.4, x = 2, x = 12
(b) x = -5, x = 4, x = 6
(c) x = 3 + 2i, x = 3 – 2i, x = 5
(d) x = 1, x = 5, x = 8, x = 12

10. x değişkeni 0 dan 10 a kadar birer adımla değiştikçe aşağıdaki polinomun değerini bir tablo şeklinde gösteriniz:

$$f(x) = 5x^3 + 4x^2 - 3x + 2$$

11. Aşağıdaki polinomun K katsayısı 0 dan 10 a kadar değiştikçe polinomum köklerini bulunuz:

$$f(x) = x^3 + 2x^2 + 3x + K$$

12. Kökleri x = 2.5, x = 4, ve x = 8 olan üçüncü derece polinomu bulunuz.

4
MATRİSLER

Matrisler matematikte ve mühendislik işlemlerinde çok yaygın olarak kullanılmaktadırlar. Matrislerin en çok kulanıldıkları alanlardan biri çok bilinmeyenli çoklu denklemlerin çözümündedir. Bu bölümde MATLAB'ı kullanarak çeşitli matris işlemlerinin nasıl yapıldığını inceleyeceğiz.

4.1 Matrislerin MATLAB'da Gösterilişi

Matrisler sıra ve sütunlardan meydana gelmiştir. Örneğin, 2 sırası ve 3 sütunu olan **A** isimli bir matris 2x3 matris olarak bilinir ve şu şekilde gösterilir:

$$A = \begin{bmatrix} 2 & 5 & 0 \\ 3 & 1 & 8 \end{bmatrix}$$

Ayni şekilde, 3 sırası ve 3 sütunu olan bir matris 3x3 olarak bilinir ve şu şekilde gösterilir:

$$A = \begin{bmatrix} 2 & -1 & 4 \\ 7 & 0 & 1 \\ 9 & 2 & 6 \end{bmatrix}$$

MATLAB'da matris gösterirken ilk olarak sıralar ve daha sonra da sütunlar yazılır. Örneğin, yukarıdaki 2x3 matris şu şekilde yazılır:

```
A = [2 5 0; 3 1 8]
```

Ayni şekilde, yukarıdaki 3x3 matrisi şu şekilde yazabiliriz:

```
A = [2 –1 4; 7 0 1; 9 2 6]
```

Bir matrisin elemanlarından başka bir matris elde etmek mümkündür. Örneğin,

$$A = \begin{bmatrix} 2 & -1 & 4 \\ 7 & 0 & 1 \\ 9 & 2 & 6 \end{bmatrix}$$

matrisini göz önünde bulundurursak,

```
B = A(2:3, 1:3)
```

Komutu B matrisini yaratır. Burada B matrisi A matrisinin 2 ve 3 üncü satırlarından ve 1 ve 3 üncü sütunlarından meydana gelmiştir. Yani,

$$B = \begin{bmatrix} 7 & 0 & 1 \\ 9 & 2 & 6 \end{bmatrix}$$

4.2 Matris Transpozu

Transpoz işlemi bir matrisin sıra ve sütunlarının yerini değiştirir. Örneğin,

$$A = \begin{bmatrix} 2 & 5 & 0 \\ 3 & 1 & 8 \end{bmatrix}$$

matrisinin transpozu şu şekildedir:

$$A^T = \begin{bmatrix} 2 & 3 \\ 5 & 1 \\ 0 & 8 \end{bmatrix}$$

MATLAB'da bir matrisin transpozunu üzerine ' karakteri koyarak buluruz. Örneğin, A matrisinin transpozunu A' olarak

gösteririz.

Aşağıdaki örnekte 2x3 bir matrisi göz önünde bulunduralım:

```
A =

    2    5    0
    3    1    8
```

Bu matrisin transpozunu B matrisinde şu şekilde saklayabiliriz:

```
>> B = A'

B =

    2    3
    5    1
    0    8

>>
```

Bir matriste $A^T = A$ ise o matris **simetrik matris** olarak bilinir.

4.3 MATRİS İŞLEMLERİ

Matrisler üzerine her çeşit aritmetik işlem yapmak mümkündür. Bu bölümde önemli matris işlemlerine göz atacağız.

4.3.1 Bir Matrisin Skalar ile Toplamı

Bir matrisi bir skalar ile toplarken matrisin her elemanını skalar ile toplamamız gerekir. Örneğin, aşağıdaki A matrisini göz önünde bulunduralım ve matrisi 2 gibi bir skalar ile toplayalım:

$$A = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix}$$

```
>> A = [3 2; 5 1];
>> B = 2 + A

B =
    5    4
    7    3

>>
```

4.3.2 Bir Matrisin Skalar İle Çarpımı

Bir matrisi bir skalar ile çarparken matrisin her elemanını skalar ile çarpmamız gerekir. Örneğin, aşağıdaki A matrisini göz önünde bulunduralım ve matrisi 3 gibi bir skalar ile çarpalım:

$$A = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix}$$

```
>> A = [3 2; 5 1];
>> B = 3*A

B =
    9    6
   15    3

>>
```

4.3.3 Matrislerin Toplamı

İki matrisi toplamak için matrislerin ayni boyutta olmaları gerekmektedir. Toplama işleminde her iki matrisin de karşılıklı elemanları toplanır ve yeni matris elde edilir. Aşağıda bir örnek verilmiştir:

$$A = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 6 \\ 4 & 12 \end{bmatrix}$$

```
A = [3 2; 5 1];
B = [1 6; 4 12];
C = A + B

C =

    4     8
    9    13

>>
```

olur.

4.3.4 Matrislerin Çıkarması

İki matrisi birbirinden çıkarmak için matrislerin ayni boyutta olmaları gerekmektedir. Çıkarma işleminde her iki matrisin de karşılıklı elemanları birbirlerinden çıkarılır ve yeni matris elde edilir. Aşağıda bir örnek verilmiştir:

$$A = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 6 \\ 4 & 12 \end{bmatrix}$$

```
A = [3 2; 5 1];
B = [1 6; 4 12];
C = A - B

C =
    2    -4
    1   -11

>>
```

olur.

4.3.5 Matrislerin Çarpımı

Matris çarpımı genel olrak 2 şekilde yapılabilir: matrislerin vektör gibi karşılıklı elemanlarının çarpımı, ve normal matris çarpımı. Şimdi her iki çarpma metodunu da örneklerle inceleyelim.

A ve B gibi aşağıdaki iki matrisi göz önünde bulunduralım:

$$A = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 6 \\ 4 & 12 \end{bmatrix}$$

Vektör Çarpımı

Vektör çarpımında her iki matrisin de karşılıklı elemanları çarpılır ve yeni matris elde edilir. Aşağıdaki örnekte A ve B matrisleri vektör olarak çarpılırlar ve netice C gibi yeni bir matriste saklanır.

```
>> A = [3 2; 5 1];
>> B = [1 6; 4 12];
>> C = A .* B

C =
    3    12
   20    12

>>
```

olur.

Burada dikkatimizi çekecek olan bir nokta, A .* B ve B .* A işlemlerinin ayni cevabı vermiş olmalarıdır. Yani,

A .* B = B .* A

Matris Çarpımı

Matris çarpımında birinci matrisin sıraları ikinci matrisin

sütunları ile çarpılır ve böylece yeni matris elde edilir. Matris çarpma işlemi matematikde normal çarpma isareti (**x)** ile gösterilir. Yukarıdaki A ve B matrislerini göz önünde bulundurursak:

$$AxB = \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} x \begin{bmatrix} 1 & 6 \\ 4 & 12 \end{bmatrix} = \begin{bmatrix} 3x1+2x4 & 3x6+2x12 \\ 5x1+1x4 & 5x6+1x12 \end{bmatrix} = \begin{bmatrix} 11 & 42 \\ 9 & 42 \end{bmatrix}$$

olur.

MATLAB ile bu çarpma işlemini şu şekilde yapabiliriz:

```
>> A = [3 2; 5 1];
>> B = [1 6; 4 12];
>> C = A*B

C =
   11   42
    9   42

>>
```

Matris çarpımı yaparken A*B ve B*A nın eşit olmadıklarına dikkat etmemiz gerekir.

Örnek 4.1

Aşağıdaki A ve B matrislerin çarpımını (C = A x B) bulunuz:

$$A = \begin{bmatrix} 1 & 1 & 3 \\ 5 & 1 & 9 \\ 7 & 8 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 2 & 1 \\ 3 & 2 & 0 \\ 4 & 1 & 2 \end{bmatrix}$$

Çözüm 4.1

MATLAB'ı kullanarak çözümü şu şekilde bulabiliriz:

```
 >> A = [1 1 3; 5 1 9; 7 8 1];
>> B = [2 2 1; 3 2 0; 4 1 2];
>> C = A*B

C =

   17    7    7
   49   21   23
   42   31    9

>>
```

İstenilen C matrisi bulunmuş olur:

$$C = \begin{bmatrix} 17 & 7 & 7 \\ 49 & 21 & 23 \\ 42 & 31 & 9 \end{bmatrix}$$

4.4 Özel Matrisler

Bazı matrisler matematikde sık sık kullanıldıkları için MATLAB'da onlar için özel komutlar ayrılmıştır. Özel matrisler olarak bilinen bu matrislerin önemli olanları Tablo 4.1 de verilmiştir.

Tablo 4.1 Özel matrislerin bazıları

eye(n)	nxn birim matris
eye(size(A))	A ile ayni büyüklükte olan birim matris
ones(n)	nxn birler matrisi
ones(m,n)	mxn birler matrisi
zeros(n)	nxn sıfırlar matrisi
zeros(m,n)	mxn sıfırlar matrisi

Birim matrisin (eye) diyagonal elemanları 1, diğer elemanları ise 0 dır. Birim matris genellikle I karakteri ile gösterilir. Bu matrisin başka bir matris ile çarpımı herhangibir şey değiştirmez. Yani, A*I = A. Ayni zamanda, bir matrisin kendi

tersi ile çarpımı birim matris verir: $A.A^{-1} = I$

Örneğin, 3x3 birim A matrisinin şu elemanları vardır:

```
>> A = eye(3)

A =
    1    0    0
    0    1    0
    0    0    1
>>
```

Birler matrisinin bütün elemanalrı da 1 dir. Aşağıda 3x3 birler matrisine örnek verilmiştir:

```
>> A = ones(3)

A =
    1    1    1
    1    1    1
    1    1    1
>>
```

Sıfırlar matrisinin bütün elemanları da 0 dır. Aşağıda 3x3 sıfırlar matrisine örnek verilmiştir:

```
>> A = zeros(3)

A =
    0    0    0
    0    0    0
    0    0    0
>>
```

4.5 Lineer Denklemler ve Matrislerle Çözümü

Matrislerin en çok olarak kullanıldıkları alanlardan biri de lineer denklemlerin çözümündedir. İkili veya üçlü lineer denklemler elde kolay çözülebilirler, fakat daha büyük denklemlerin çözümünde MATLAB çok yardımcı olmaktadır.

4.5.1 Denklem Sayısı ve Bilinmeyen Sayısı Eşit Olan Denklemler

Aşağıda ikili bir lineer denklem örneği verilmiştir:

$$x + y = 2$$
$$2x + 3y = 8$$

Bu denklemin çözümü için birçok yol bulunmaktadır. Örneğin, her iki denklemin de grafiği çizilip kesiştikleri noktalar bulunur. Bu noktalardaki **x** ve **y** değerleri her iki denklemi de sağladığı için denklemin çözüm noktalarıdır.

Yukarıdaki denklem **Gauss** eliminasyon tekniği ile de çözülebilir. Örneğin, birinci denklemi 2 ile çarparsak şu denklemleri elde ederiz:

$$2x + 2y = 4$$
$$2x + 3y = 8$$

Şimdi birinci denklemden ikinci denklemi çıkarırsak x değişkenini elimine etmiş oluruz:

$$2x + 2y = 4$$
$$-2x - 3y = -8$$

Buradan da –y = -4, veya y = 4 olarak bulunur. Şimdi y yi herhangibir denklemde yerine koyarsak x = -2 olur.

Lineer denklemleri MATLAB kullanarak çözmek için ilk olarak denklemi matris şeklinde yazmamız gerekir. Yukarıdaki örnekdeki denklemleri şu şekilde yazabiliriz:

$$\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 8 \end{bmatrix}$$

Bu matris işlemi ise genel olarak şu şekildedir:

$$A.x = b$$

Burada **A** matrisi 2x2 matris, **x** matrisi x ve y yi ihtiva eden 2x1 matris, ve **b** matrisi de 2x1 denklemin sağ tarafını ihtiva eden matristir.

MATLAB ile çözüm matrisini bulmak için **x = A \ b** işlemini yapabiliriz (aşağıdaki işlemde b matrisinin transpozunu aldığımıza dikkat ediniz). Bu işlem Gauss eliminasyon tekniğini kullanmaktadır:

```
>> A = [1 1; 2 3];
>> b = [2 8]';
>> x = A \ b

x =
   -2
    4
>>
```

Böylece x = -2 ve y = 4 olarak bulunur.

Yukarıdaki denklemi ters matris kullanarak da çözmek mümkündür. Genel denklemin matris olarak yazılımını göz önünde bulundurursak:

$$A.x = b$$

Şimdi, her iki tarafı da A^{-1} (A nın tersi) ile çarparsak:

$$A.A^{-1}.x = A^{-1}.b$$

Bir matrisin kendi tersi ile çarpımı birim matrisi (I) vermektedir. Bu durumda,

$$I.x = A^{-1}.b$$

veya

$x = A^{-1}.b$

Olarak bulunur. Böylece, lineer denklemi çözmek için ilk olarak A matrisinin tersini alıp bu matrisi b ile çarpmamız gerekecektir. MATLAB'da bir matrisin tersi **inv** fonksiyonu ile bulunur. Sadece kare matrislerin tersleri olduğu için bu metodla denklem sayısı bilinmeyen sayısına eşit olan lineer denklemleri çözebiliriz. Eğer denklem sayısı bilinmeyen sayısına eşit değilse daha önce bahsedilen ters bölme metodunu kullanmamız gerekir.

Örneğin,

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 5 \end{bmatrix}$$

matrisinin inversi şu şekilde bulunur:

```
>> A = [3 1; 2 5];
>> inv(A)

ans =
    0.3846   -0.0769
   -0.1538    0.2308
>>
```

Böylece,

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 5 \end{bmatrix} \text{ ise,}$$

$$A^{-1} = \begin{bmatrix} 0.3846 & -0.0769 \\ -0.1538 & 0.2308 \end{bmatrix}$$

Örnek 4.2

Aşağıdaki lineer denklemin çözümünü ters matris metodunu kullanarak bulunuz:

6x – 3y + 4z = 41
12x + 5y – 7z = -26
-5x + 2y +6z = 14

Çözüm 4.2

İlk olarak denklemi matris şeklinde yazalım:

$$\begin{bmatrix} 6 & -3 & 4 \\ 12 & 5 & -7 \\ -5 & 2 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 41 \\ -26 \\ 14 \end{bmatrix}$$

burada,

$$A = \begin{bmatrix} 6 & =3 & 4 \\ 12 & 5 & -7 \\ -5 & 2 & 6 \end{bmatrix}, \quad x = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad b = \begin{bmatrix} 41 \\ -26 \\ 14 \end{bmatrix}$$

denklemin köklerini $x = A^{-1}.b$ işlemi ile bulabiliriz:

```
>> A = [6 -3 4; 12 5 -7; -5 2 6];
>> b = [41 -26 14]';
>> x = inv(A)*b

x =
    2.0000
   -3.0000
    5.0000
>>
```

Böylece, kökler, x = 2, y = -3 ve z = 5 olarak bulunmusş olur.

Örnek 4.3

Aşağıdaki lineer denklemin köklerini bulunuz:

3x + 2y - 9z = -65
-9x - 5y + 2z = 16
6x + 7y +3z = 5

Çözüm 4.3

Denklemi matris şeklinde yazmadan direk olarak A ve b matrislerini yazıp köklerini bulabiliriz:

```
>> A = [3 2 -9; -9 -5 2; 6 7 3];
>> b = [-65 16 5]';
>> x = inv(A)*b

x =
    2.0000
   -4.0000
    7.0000
>>
```

Böylece, x = 2, y = -4 ve z = 7 olarak bulunurlar.

Matrisler ile denklem çözerken **b** matrisinin transpozunun alındığına dikkat edilmesi gerekmektedir. Yukarıdaki işlemleri **b** nin transpozunu almadan da yapmamız mümkündür. Bu örnekte **b** matrisi 3x1 olduğuna göre matrisin elemanlarını 3 satır halinde yazarsak transpoz işlemine gerek kalmayacaktır:

```
>> A = [3 2 -9; -9 -5 2; 6 7 3];
>> b = [-65; 16; 5];
>> x = inv(A)*b

x =
    2.0000
   -4.0000
    7.0000
>>
```

Örnek 4.4

Aşağıdaki lineer denklemin köklerini bulunuz:

$$3x + 2y - z + w + 3p = 14$$
$$-x - y + 2z + w + p = 7$$

x + y +3z – w - p = 3
x + y + z + w + p = 9
2x + 2y – z + 3w + 5p = 23

Çözüm 4.4

```
>> A = [3 2 -1 1 3; -1 -1 2 1 1; 1 1 3 -1 -1; 1 1 1 1 1;2 2 -1 3 5];
>> b = [14; 7; 3; 9; 23];
>> x =inv(A)*b

x =
   1.0000
   1.0000
   2.0000
   2.0000
   3.0000
>>
```

Denklemin kökleri, x = 1, y = 1, z = 2, w = 2, ve p = 3 olarak bulunur.

4.5.2 Denklem Sayısı ve Bilinmeyen Sayısı Eşit Olmayan Denklemler

Lineer denklemlerin bazı durumlarda çözümleri olmayabilir, veya çözüm tek olmayabilir. Örneğin, aşağıdaki denklemin sonsuz sayıda çözümü vardır:

2x + y = 10
4x + 2y = 20

Bunun sebebi ise, her iki denklem de aynidir (ikinci denklem birincinin 2 katıdır). Bu gibi durumlarda denklemin tek çözümü olmayıp çözüm bilinmeyenler arasındaki bağıntı olarak gösterilebilir:

x = (10 – y) / 2

Burada, y nin aldığı sonsuz değerlere göre x bilinmeyeni de sonsuz değer alacaktır.

Aşağıda verilen örnekte ise denklemin çözümü yoktur. Bu örnekte, y = 10 – 2x ve y = (20 – 10x) / 5 doğrularu birbirlerine paralel olup kesişmezler:

2x + y = 10
10x + 5y = 20

Verilen bir lineer denklemin çözümü olup olmadığını araştırırken ilk olarak o denklemin determinantını hesaplamamız gerekir. Determinantı hesaplama için çeşitli metodlar vardır. Burada sadece 2x2 bir matrisin determinantının nasıl hesaplandığını göreceğiz.

Aşağıdaki matrisi göz önünde bulundurursak,

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 5 \end{bmatrix}$$

bu matrisin determinantını hesaplarken matrisin elemanlarını iki düz paralel çizgi arasına yazarız ve şu işlemi yaparız:

$$|A| = \begin{vmatrix} 3 & 1 \\ 2 & 5 \end{vmatrix} = 3x5 - 1x2 = 13$$

Bir lineer denklemde A matrisinin determinantı 0 ise o matris tekeş (singular) olarak bilinir. Tekeş bir denklemin ya hiç çözümü yoktur, veya sonsuz sayıda çözümü olabilir. Tekeş bir lineer denklemin çözümünü ters matris metodunu kullanarak (**x = A^{-1}.b**) hesaplamak mümkün değildir. Böyle bir denklemin çözümü için ters matris bölme işlemini (**x = A \ b**) kullanmamız gerekir.

Bir matrisin determinantını MATLAB kullanarak bulmak için

det fonksiyonunu kullanırız. Örneğin, yukarıda verilen A matrisinin determinantını MATLAB ile şu şekilde bulabiliriz:

```
>> det([3 1; 2 5])
ans =
        13
>>
```

olarak bulunur.

Verilen bir lineer denlemin çözümü olup olmadığını araştırırken o denklemi oluşturan matrisin **rank**'ını da bulmamız gerekmektedir. Bir matrisin rankını bulurken o matrisi oluşturan determinanta bakarız ve bu determinant içerisinde 0 olmayan en büyük determinant bize o matrisin rankını verir. Örneğin, yukarıdaki A matrisinin rankı 2 dir. MATLAB kullanarak herhamgibir matrisin rankını, **rank** fonksiyonu ile bulabiliriz:

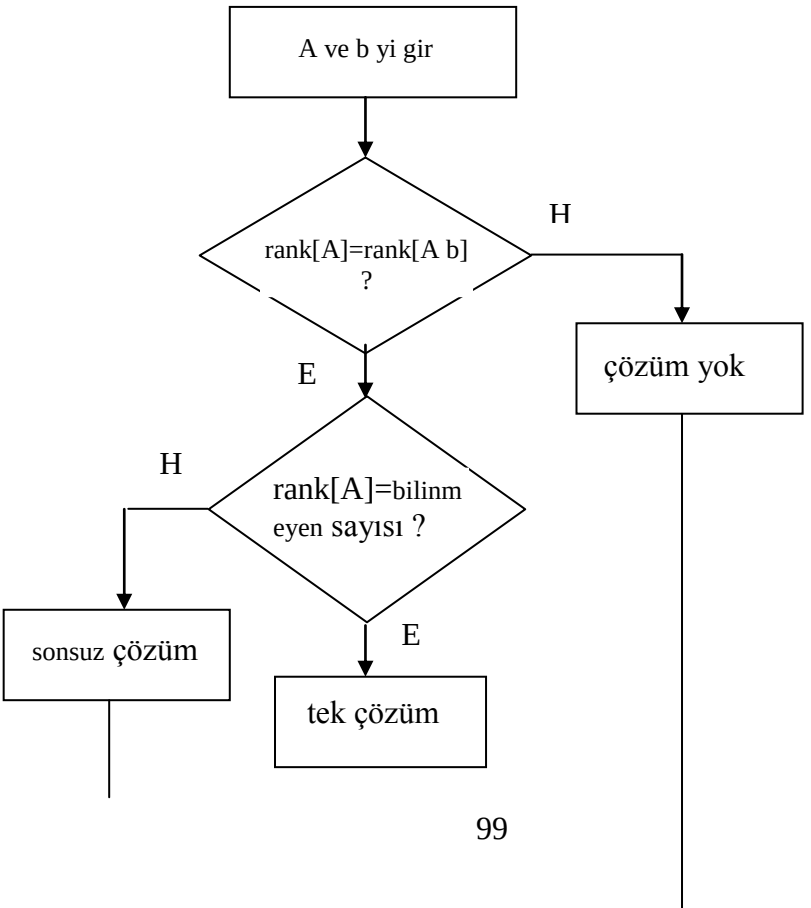
```
>> rank([3 1; 2 5]
ans =
        2
>>
```

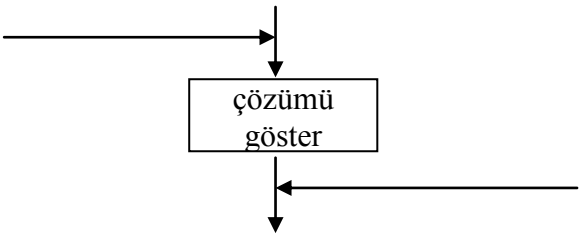
olarak bulunur. Şimdi, bir lineer denklemin çözümü için gerekli olan teoremi yazabiliriz:

A.x = b *lineer denkleminde* ***m*** *tane denklem ve* ***n*** *tane de bilinmeyen olduğunu kabul edelim. Bu denklemin herhangibir çözümünün olması için: 1.* ***rank[A] = rank[A b]*** *koşulunun olması gerekmektedir. Eğer 1 nolu koşul sağlanmışsa ve ayni zamanda* ***rank[A] = bilinmeyen sayısı*** *ise bu durumda denklemin çözümü tektir. Eğer koşul 1 sağlanmışsa, fakat* ***rank[A} < bilinmeyen sayısı****, bu durumda denklemin sonsuz sayıda çözümü bulunmakta ve böylece bilinmeyenler arasında bir bağıntı kurmak mümkündür.*

Şunu da unutmamak gerekir ki eğer $|A| = 0$ ise, yani A matrisinin determinantı 0 ise denklem tekeş olup bu denklemi ters matris (**x = A^{-1}.b**) veya ters bölme (**x = A \ b**) işlemleri

ile çözmek mümkün değildir. Verilen bir nxn kare A matrisinde rank'ın n den küçük olması, $|A| = 0$ olduğunu gösterir. Şekil 4.1 de herhangibir lineer denklemi çözmek için takip edilmesi gereken yol bir akış şeması olarak gösterilmiştir. İlk olarak A ve b matrisleri tanımlanır. Daha sonra [A] ve [A b] matrislerinin rank'ları hesaplanır. Eğer iki rank eşit değillerse denklemin çözümü yoktur. İki rank eşit oldukları zaman, rank'ın bilinmeyen sayısına eşit olup olmadığı araştırılır. Eğer rank bilinmeyen sayısına eşitse denklemin tek çözümü vardır ve bu çözüm ters bölme işlemi ile hesaplanabilir. Eğer rank bilinmeyen sayısından küçükse denklemin sonsuz sayıda çözümü bulunmaktadır. Bu durumda, ters bölme işlemi kullanılarak bir çözüm bulunabilir. MATLAB böyle bir durumda bilinmeyenlerden birinin 0 olduğunu kabul eder ve diğer çözümleri hesaplar. Bir diğer çözüm şekli ise MATLAB **rref** fonksiyonunu kullanarak bilinmeyenler arasındaki bağıntıyı göstermektir. **rref** komutu matrisi *azaltılmış sıra echelon formatı*'na dönüştürür.





Şekil 4.1 Lineer denklem çözüm akış şeması

Lineer denklem çözümü ile ilgili örnekler aşağıda verilmiştir:

Örnek 4.5

Aşağıdaki denklemi çözünüz:

2x –4y + 5z = -4
-4x –2y +3z = 4
2x +6y –8z = 0

Çözüm 4.5

İlk olarak A ve [A b] matrislerinin determinantlarını hesaplayalım:

$$A = \begin{bmatrix} 2 & -4 & 5 \\ -4 & -2 & 3 \\ 2 & 6 & -8 \end{bmatrix} \qquad b = \begin{bmatrix} -4 \\ 4 \\ 0 \end{bmatrix}$$

ve,

$$[Ab] = \begin{bmatrix} 2 & -4 & 5 & -4 \\ -4 & -2 & 3 & 4 \\ 2 & 6 & -8 & 0 \end{bmatrix}$$

MATLAB'ı kullanarak matrislerin rank'larını şu şekilde

hesaplayabiliriz:

```
>> A = [2 -4 5; -4 -2 3; 2 6 -8];
>> b = [-4; 4; 0];
>> rank(A)

ans =

    2

>> rank([A b])

ans =

    2

>>
```

A matrisinin determinantı ise

```
>> det(A)

ans =
          2
>>
```

olarak bulunur.

[A] ve [A b] matrislerinin her ikisinin de rank'ları 2 olarak bulunmuştur. **rank[A] = rank[A b]** olduğu için denklemlerin çözümü vardır. Fakat **rank < bilinmeyen sayısı** olduğu için denklemlerin sonsuz sayıda çözümü olmasi beklenmektedir. Fakat A matrisinin determinantı 0 olduğu için matris tekeştir (singular) ve ters bölme işlemiyle veya matris tersi metodunu kullanarak bir çözüm bulmamız mümkün değildir. Örneğin, ters bölme işlemi ile çözmeye çalışırsak ekranda şu hata mesajını görürüz:

```
>> x = A \ b;
Warning: Matrix is singular to working precision.
(Type "warning off MATLAB:singularMatrix" to suppress this warning.)
```

```
>>
```

Yukarıdaki matrisi çözmek için yarı ters matris metodu diye bilinen ve **pinv** komutunu kullanan tekniği uygulayabiliriz. Gerekli olan komutlar aşağıda verilmiştir:

```
>> x = pinv(A)*b

x =

   -1.2148
    0.2074
   -0.1481

>>
```

Yukarıdaki matrisin sonsuz sayıda çözümü olmasına rağmen, **pinv** komutunu kullanarak bulunan çözüm x vektörünü minimum yapan çözüm olarak bilinmektedir.

Yukarıdaki örnekte genel bir çözüm elde etmek için *azaltılmış sıra echelon formatını* kullanabiliriz. Bunun için de **rref** komutunu kullanmamız gerekir:

```
>> rref([A b])

ans =

     1     0   -0.1   -1.2
     0     1   -1.3    0.4
     0     0    0       0

>>
```

Cx = d genel denklemini düşünürsek, yukarıdaki matrisi [C d] olarak şu şekilde gösterebiliriz:

$$[C \quad d] = \begin{bmatrix} 1 & 0 & -0.1 & -1.2 \\ 0 & 1 & -1.3 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

ve bilinmeyenler arasındaki bağıntıyı şu şekilde yazabiliriz:

x + 0y – 0.1z = -1.2
0x +y –1.3z = 0.4
0x + 0y + 0z = 0

veya

x – 0.1z = -1.2
y – 1.3z = 0.4

olur.

Burada, z değişkeninin değeri bulunursa denklem tam olarak çözüşmüş olur.

Örnek 4.6

Aşağıdaki lineer denklemin çözümünü bulunuz:

X + y + z = 400
10x + 5y = 1600

Çözüm 4.6

İlk olarak [A] ve [A b] matrislerinin determinantlarını hesaplayalım:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 10 & 5 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 400 \\ 1600 \end{bmatrix}$$

ve

$$[A \quad b] = \begin{bmatrix} 1 & 1 & 1 & 400 \\ 10 & 5 & 0 & 1600 \end{bmatrix}$$

Matrislerin ranklarını hesaplarsak:

```
>> A = [1 1 1; 10 5 0];
>> b = [400; 1600];
>> rank(A)

ans =

    2

>> rank([A b])

ans =

    2

>>
```

Bu örnekte rank[A] = rank [A b], fakat rank bilinmeyen sayısından küçüktür. Bu durumda sonsuz sayıda çözüm olmasını bekleriz. Matris ters bölme işlemi ile çözüm bulmaya çalışırsak:

```
>> x = A \ b

x =

  160.0000
         0
  240.0000

>>
```

x = 160, y = 0 ve z = 240 olarak bir çözüm elde etmiş oluruz. MATLAB bu durumda çözüm ararken bilinmeyenlerden bir tanesini 0 yapıp diğer bilinmeyenleri hesaplar. Yukarıdaki çözüm sonsuz çözümlerden biri olduğu için bu çözümün geçerliliği denklemi meydana getiren fiziksel ortama da bağlıdır. Örneğin, y değişkeni fiziksel kuvveti gösteriyorsa, kuvvetin 0 olmasının geçerli bir çözüm olup olmamasına siz karar verebilirsiniz.

4.6 rand Fonksiyonu (Gelişigüzel sayılar üretmek)

rand fonksiyonunu kullanarak 0 ve 1 arasında gelişigüzel (random) sayı üretmemiz mümkündür. Örneğin,

```
>> rand

ans =

    0.6602
>>
```

rand komutunu tekrar yazdığımızda 0 ve 1 arasında yeni bir sayı elde ederiz:

```
>> rand

ans =

        0.7200
>>
```

Bu şekilde, rand komutunu her girişimizde 0 ve 1 arasında yeni bir sayı elde etmiş oluruz. Rand fonksiyonu bilgisayarımızın saatını kullanarak sayı üretmektedir. Saat ise daimi değiştiği için üretilmiş olan sayılar değişik olmaktadır.

Birden fazla gelişigüzel sayı üretmek için rand komutunu birden fazla yazabiliriz. Veya, örneğin, **rand(n,1)** komutunu kullanarak n tane gelişigüzel sayı üretebiliriz. Aşağıdaki örnekte 0 ve 1 arasında 10 tane gelişigüzel sayı üretilmiştir:

```
>> rand(10,1)

ans =
```

```
   0.6979
   0.3784
   0.8600
   0.8537
   0.5936
   0.4966
   0.8998
   0.8216
   0.6449
   0.8180

>>
```

rand(n,m) komutu ile nxm boyutlu ve 0 ile 1 arasında gelişigüzel sayıları ihtiva eden bir matris elde etmiş oluruz:

```
>> rand(3,3)

ans =

   0.3420   0.5341   0.8385
   0.2897   0.7271   0.5681
   0.3412   0.3093   0.3704

>>
```

Ayni şekilde:

```
>> rand(1,5)

ans =

   0.9797   0.2714   0.2523   0.8757   0.7373

>>
```

randn komutunu kullanarak 0 ve 1 arasında, *normal dağılım*a uygun bir sayı üretebiliriz:

```
>> randn

ans =
```

```
   -0.4326

>>
```

Yine ayni şekilde, birden çok normal dağılıma uygun sayı üretmek için:

```
>> randn(10,1)

ans =

   -1.6656
    0.1253
    0.2877
   -1.1465
    1.1909
    1.1892
   -0.0376
    0.3273
    0.1746
   -0.1867

>>
```

komutunu yazabiliriz.

rand veya **randn** komutları ile üretilmiş olan sayılar 0 ve 1 arasındadırlar. Herhangibir aralıkta gelişigüzel sayı üretmek için şunu yapabiliriz:

Örneğin, [n,m] aralığında bir sayı üretmek için ilk olarak **rand** komutunu kullanarak bir sayı üretiniz. Daha sonra bu sayıyı m-n ile çarpınız ve n ilave ediniz. Bu şekilde elde etmiş olduğunuz sayı n ve m arasında olacaktır. Aşağıda bir örnek verilmiştir:

Örnek 4.7

Bu örnekte, 2 ve 10 arasında gelişigüzel (random) bir sayı üreteceğiz.

Çözüm 4.7

İlk olarak rand fonksiyonu ile bir sayı üreteceğiz, daha sonra bu sayıyı (10 – 2) = 8 ile çarpıp 2 ileve edeceğiz.

```
>> a = 8*rand+2

a =

   3.0921

>>
```

Örnek 4.8

2 ve 10 arasında değişen 10 tane gelişigüzel sayı elde ediniz.

Çözüm 4.8

Gerekli olan komutlar aşağıda verilmiştir:

```
>> a = rand(10,1);
>> b = 8*a + 2

b =

   2.0941
   9.1512
   3.5931
   4.3898
   7.2915
   4.2753
   5.7538
   2.5182
   9.9067
   6.6623

>>
```

4.7 Sorular

1. Aşağıda A ve B isimli iki matris verilmiştir:

$$A = \begin{bmatrix} 1 & 0 \\ 12 & 8 \end{bmatrix} \qquad B = \begin{bmatrix} 5 & 10 \\ 9 & 5 \end{bmatrix}$$

Bu matrisler üzerine şu işlemleri yapınız:

(a) C = A + B (b) C = A – B
(c) C = A ve B nin vektör çarpımı
(e) C = A ve B nin vektör bölümü

2. Soru 1 de verilen iki matrisin çarpımını elde bulunuz. Daha sonra MATLAB'ı kullanarak cevabınızı kontrol ediniz.

3. Aşağıda verilen A matrisinin transpozunu bulunuz:

$$A = \begin{bmatrix} 3 & 1 & 6 \\ 1 & 5 & 8 \\ 6 & 8 & 1 \end{bmatrix}$$

Bu matris için ne söyleyebilirsiniz ?

4. Aşağıda verilen iki matrisin çarpımını bulunuz:

$$A = \begin{bmatrix} 3 & 1 & 6 \\ 1 & 5 & 8 \\ 6 & 8 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 11 & 0 \\ 0 & 2 & 8 \\ 7 & 5 & 2 \end{bmatrix}$$

5. Aşağıda verilen denklemlerin çözümlerini bulunuz:

(a)

2x + 2y + z = 5
x – y + z = 1

$x + 2y + z = 4$

(b)

$x + y + 2z = 7$
$x – y – z = - 3$
$2x + 3y + 5z = 18$

(c) $3x + 2y – 9z = -65$
$- 9x – 5y + 2z = 16$
$6x + 7y + 3z = 5$

6. Aşağıdaki iki matrisi örnek alarak AB çarpımının BA çarpımına eşit olmadığını isbatlayınız:

$$A = \begin{bmatrix} 2 & 1 \\ 5 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 6 \\ 2 & 3 \end{bmatrix}$$

7. Aşağıdaki matrisleri göz önünde bulundurarak şu işlemleri
yapınız:

(a) A + B + C
(b) A – B + C

$$A = \begin{bmatrix} 2 & 1 \\ 5 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 6 \\ 2 & 3 \end{bmatrix} \quad C = \begin{bmatrix} -1 & 2 \\ 2 & -1 \end{bmatrix}$$

8. Aşağıdaki matrisi göz önünde bulundurarak şu işlemleri
yapınız:

(a) Matrisin ikinci satırını seçiniz
(b) İkinci satırın elemanlarının toplamını bulunuz
(c) Matrisin üçüncü satırını seçiniz
(d) Ikinci ve üçüncü satırların vektör çarpımını bulunuz

$$A = \begin{bmatrix} 1 & -1 & 4 \\ 12 & 2 & -5 \\ 8 & 4 & 3 \end{bmatrix}$$

9. Aşağıda verilen matris için şu işlemleri yapınız:

(a) Her sütündaki maksimum ve minimum elemanı bulunuz
(b) Her sıradaki maksimum ve minimum elemanı bulunuz

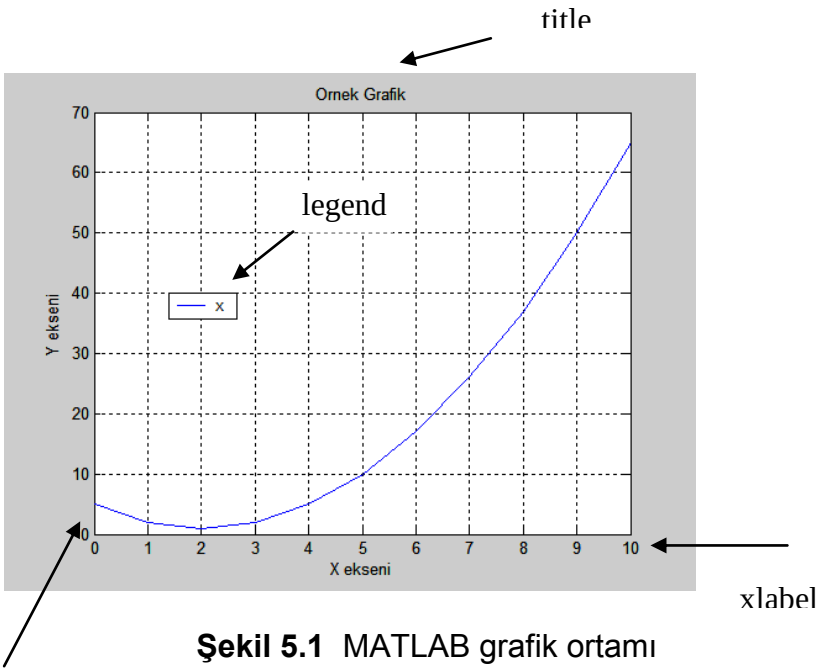
$$A = \begin{bmatrix} 11 & -1 & -6 \\ 12 & 0 & -5 \\ -6 & 4 & 1 \end{bmatrix}$$

5
GRAFİK ÇİZİMİ

Grafik çizimi MATLAB'ın en önemli özelliklerinden biridir. MATLAB'ı kullanarak kolaylıkla iki boyutlu ve üç boyutlu her çeşit grafik çizmemiz mümkündür. Bu bölümde grafik çizme işlemini inceleyeceğiz.

5.1 MATLAB x-y Grafik Ortamı

İlk olarak MATLAB grafik ortamını ve MATLAB'ı kullanarak basit bir x-y grafiğinin nasıl çizilebileceğini göreceğiz. MATLAB grafik ortamı Şekil 5.1 de gösterilmiştir. Burada **x** ekseni yatay eksen ve **y** ekseni dikey eksendir.



Şekil 5.1 MATLAB grafik ortamı

ylabel

x ekseninin tanımı **xlabel** komutu ile belirlenir ve bu komut içerisine yazılan yazı **x** ekseni altına yazılır.

Ayni şekilde, **y** ekseninin tanımı **ylabel** komutu ile belirlenir ve bu komut içerisine yazılan yazı **y** ekseni yanına yazılır.

Grafiğin ismi **title** komutu ile belirlenir ve bu komut içerisine yazılan yazı grafiğin üst dış bölümüne yazılır.

Grafik gridi **grid on** komutu ile gösterilir. **grid off** komutu ise gridi kaldırır.

Grafik komutları yazılırken komutları virgül ile ayırıp ayni satıra yazmak mümkündür. Eğer bir satır sığmıyorsa, satır devam işareti olan üç tane nokta (…) koyup bir sonraki satırda devam edebiliriz.

MATLAB ile grafik çızmek için **plot** komutunu kullanırız. Bu komutun genel şekli **plot(x,y)** olup **x** ve **y** vektör iseler, bu değerlere bağlı olan bir grafik çizilir. Grafiğin düzgün olması için mümkün olduğu kadar noktanın alınmasına dikkat edilmelidir. Grafikle ilgili tanımların grafik çizildikten sonra belirtilmesi gerekmektedir.

Aşağıda bir örnek verilmiştir:

Örnek 5.1

x değeri 0 ve 20 arasında değiştikçe $y = 2\sqrt{x}$ denkleminin grafiğini çiziniz.

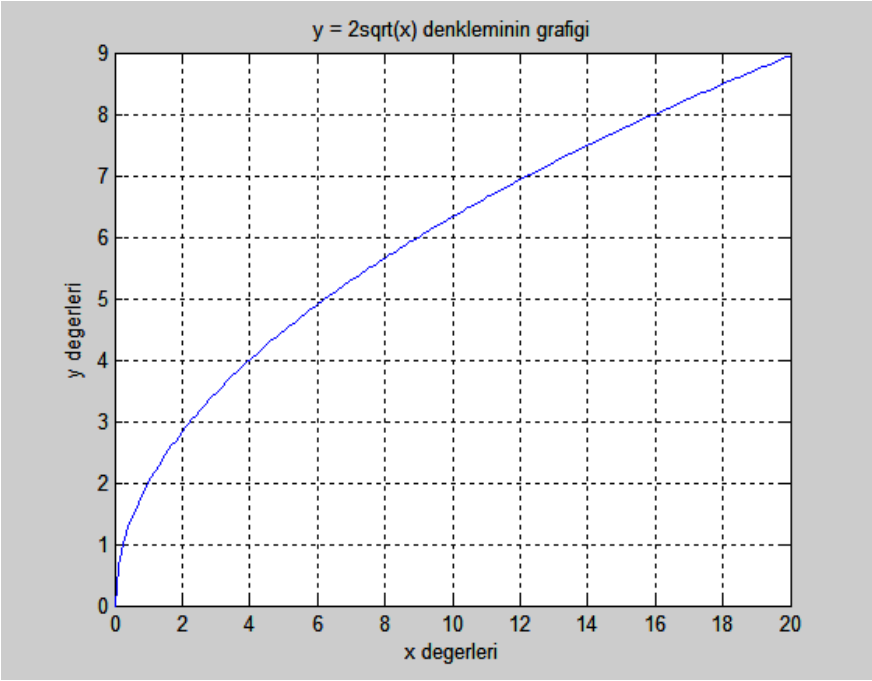
Çözüm 5.1

İlk olarak x değerlerini bir vektörde saklayıp daha sonra da grafiği çizebiliriz.

```
>> x = [0:0.1:20];
```

```
>> y = 2*sqrt(x);
>> plot(x,y)
>> xlabel('x degerleri')
>> ylabel('y degerleri')
>> title('y = 2sqrt(x) denkleminin grafigi')
>> grid on
```

Çizilmiş olan grafik Şekil 5.2 de gösterilmiştir.



Şekil 5.2 Çizilen grafik

Yukarıdaki komutları virgül ile ayırıp ayni satıra yazmamız da mümkündür:

```
>> x = [0:0.1:20];
>> y = 2*sqrt(x);
>> plot(x,y), xlabel('x degerleri'), ylabel('y degerleri'), …
     title('y = 2sqrt(x) denkleminin grafigi'), grid on
```

Grafikten de görüleceği gibi **x** ve **y** eksenlerinin başlangıç ve bitim noktaları ve eksenler üzerindeki işaretler otomatik olarak MATLAB tarafından konmuştur.

Eksenlerin başlangıç ve bitim noktalarını istersek kendimiz de belirtebiliriz. Bunun için, istediğimiz minimum ve maksimum eksen noktalarını şu şekilde belirtmemiz gerekir: **[xmin xmax ymin ymax]**. Genel olarak şu eksen komutları bulunmaktadır:

axis square: Bu komut grafiğin kare şeklinde olması için gerekli olan eksen limit seçimlerini yapar.

axis equal: Bu komut eksen işaretlerinin ve limitlerinin her iki eksende de ayni olmasını sağlar.

axis auto: Genel olarak en çok kullanılan ve ayni zamanda otomatik olarak seçilen eksen komutudur. Bu komut eksen işaretlerini ve eksen limitlerini otomatik olarak en iyi şekilde seçmeyi hedef alır.

Örnek 5.2

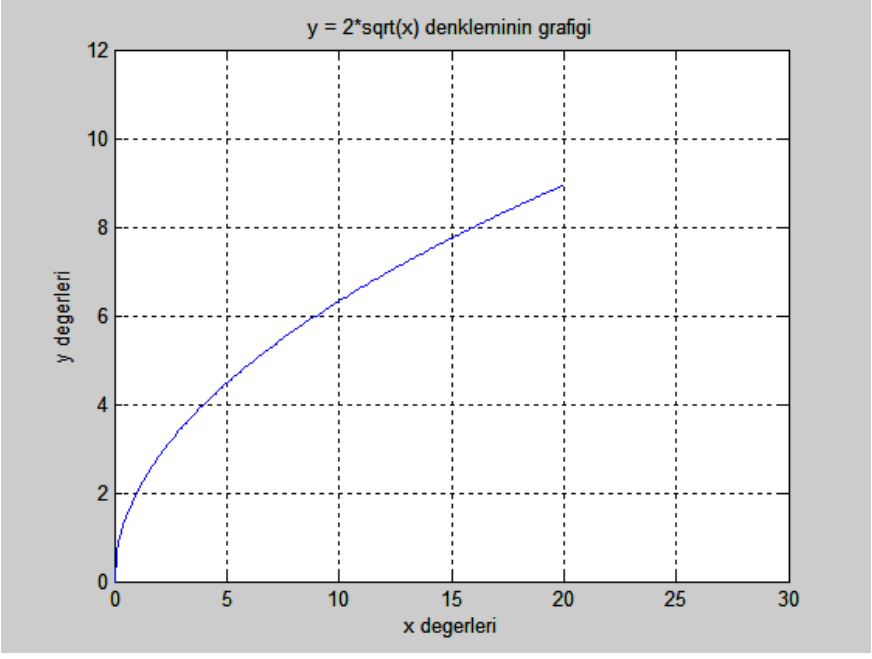
Örnek 5.1 deki grafiği tekrar çiziniz ve x ekseni limitlerini 0 dan 30 a, y ekseni limitlerini ise 0 dan 12 ye alınız.

Çözüm 5.2

Gerekli olan MATLAB adımları Örnek 5.1 e benzer fakat burada eksen uzunluklarını kendimiz seçmekteyiz:

```
>> x = [0:0.1:20];
>> y = 2*sqrt(x);
>> plot(x,y)
>> xlabel('x degerleri')
>> ylabel('y degerleri')
>> title('y = 2sqrt(x) denkleminin grafigi')
>> grid on
>> axis([0 30 0 12])
```

Yeni grafiğimiz Şekil 5.3 de gösterilmiştir.



Şekil 5.3 **x** ve **y** eksen limitleri değiştirilmiş grafik

5.2 Veri İşaretleri ve Çizgi Çeşitleri

Herhangibir deneyden elde ettiğimiz noktaların grafiğini çizerken normal olarak veri noktalarını grafik üzerinde belirli sembollerle belirtiriz ve bu veri noktalarını doğru parçaları ile veya eğrilerle birleştiririz.

MATLAB ile grafik çizerken veri noktalarına koyacağımız sembolleri, eğri şeklini, ve eğrinin rengini seçmemiz mümkündür. Tablo 5.1 de en çok kullanılan veri sembolleri, eğri çeşitleri, ve renkleri verilmiştir.

Tablo 5.1 Veri sembolleri, eğri çeşitleri ve eğri renkleri

Veri sembolü		**Eğri çeşidi**		**Renk**	
Nokta	.	Katı çizgi	-	Siyah	k
Yıldız	*	Noktalı çizgi	--	Mavi	b
Çarpı	x	Noktalı uzun çizgi	-.	Siyan	c
Daire	o	Noktalı çizgi	:	Yeşil	g
Artı	+			Magenta	m
Kare (▯)	s			Kırmızı	r
Elmas (◊)	d			Beyaz	w
5 köşeli yıldız(*)	p			Sarı	y

Örneğin, veri noktalarına küçük çarpı işaretleri koymak için:

```
plot(x,y,'x')
```

Komutunu vermemiz gerekir. Bu komutla grafik çizdiğimizde sadece veri noktalarını görürüz ve bu noktalar herhangibir şekilde birbirlerine bağlanmış değildirler. Veri noktalarını birbirlerine bağlamak için eğri çeşidini de belirtmemiz gerekir. Örneğin, veri noktalarına **x** işareti koyup bu noktaları noktalı çizgilerle birleştirmek için şu komutu vermemiz gerekir:

```
plot(x,y,'x',x,y,':')
```

Çizginin örneğin kırmızı olmasını istersek:

```
plot(x,y,'rx',x,y,':')
```

Komutunu yazmamız gerekir.

Örnek 5.3

Yapılmış olan bir deneyde bir direncin uçlarındaki gerilim ve dirençten geçen akım aşağıdaki tabloda verildiği gibidir. Gerilim ve akım değişimini bir grafik olarak gösteriniz ve veri noktalarına 5 köşeli yıldız koyunuz.

Gerilim (Volt)	Akım (Amper)
10	1
15	1.5
20	2
25	2.5
30	3
35	3.5
40	4
45	4.5
50	5

Çözüm 5.3

Program listesi aşağıda verilmiştir. Gerilim V isimli bir vektörde saklanmış ve 10 dan 50 ye kadar 5 er aralıklı değerler almıştır. Akım ise A isimli bir vektörde saklanmış ve 1 den 5 e kadar 0.5 aralıklarla değerler almıştır. Plt komutunda veri noktalarına 5 köşeli yıldız konması için ‘p’ komutu ilave edilmiştir.

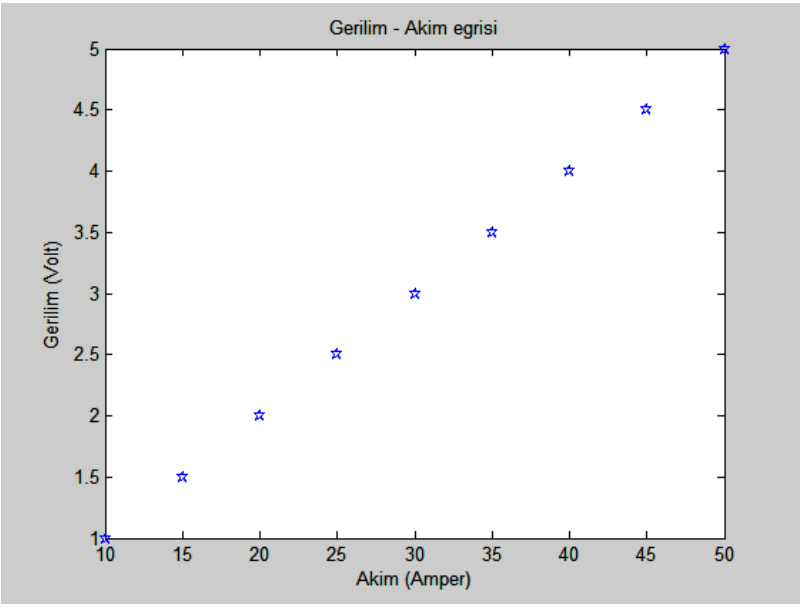
```
>> V=10:5:50;
>> plot(V,A);
>> plot(V,A,'p')
>> V=10:5:50;
>> A = 1:0.5:5;
>> plot(V,A,'p')
>> xlabel('Akim (Amper)');
>> ylabel('Gerilim (Volt)');
>> title('Gerilim - Akim egrisi');
>>
```

Yukarıdaki örnekte veri noktalarına işaretler konmuş fakat bu

noktalar birleştirilmemiştir (Şekil 5.4 e bakınız). Aşağıdaki örnekte bu noktaların nasıl birleştirileceği gösterilmiştir.

Örnek 5.4

Örnek 5.3 deki grafiği çiziniz fakat ilave olarak veri noktalarını çizgiler ile birleştiriniz.



Şekil 5.4 Veri noktaları 5 köşeli yıldızla gösterilmiştir

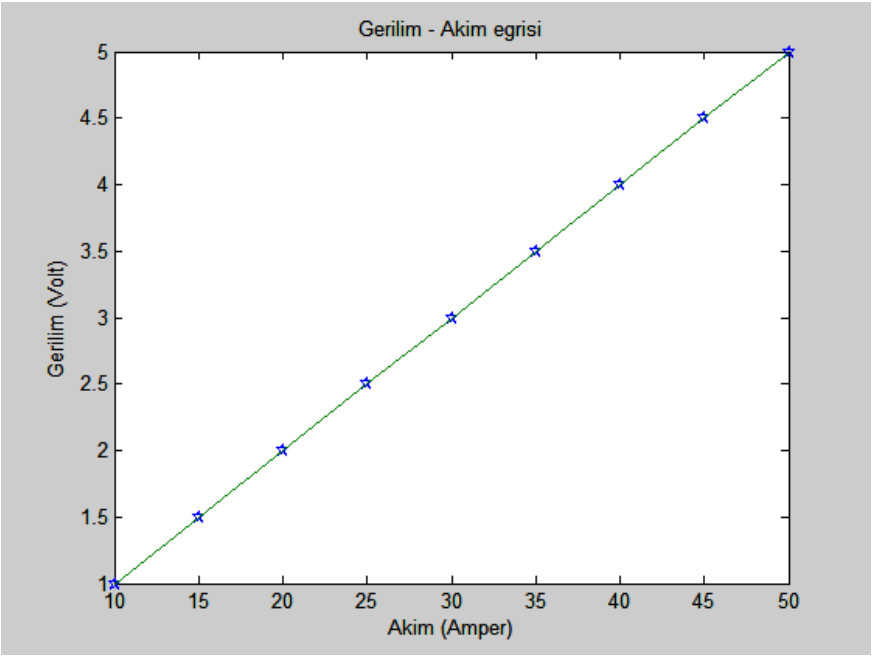
Çözüm 5.4

Yeni program listesi aşağıda verilmiştir. Yeni grafik ise Şekil 5.5 de gösterilmiştir.

```
>> V=10:5:50;
>> plot(V,A);
>> plot(V,A,'p')
>> V=10:5:50;
>> A = 1:0.5:5;
>> plot(V,A,'p',V,A,'-')
>> xlabel('Akim (Amper)');
```

```
>> ylabel('Gerilim (Volt)');
>> title('Gerilim - Akim egrisi');
>>
```

Programda da görüleceği gibi plot komutu değiştirilmiştir. İlk olarak veri noktalarına 5 köşeli yıldız konmuş, daha sonra da bu noktalar çizgilerle birleştirilmiştir.



Şekil 5.5 Veri noktaları çizgilerle birleştirilmiştir

Örnek 5.5

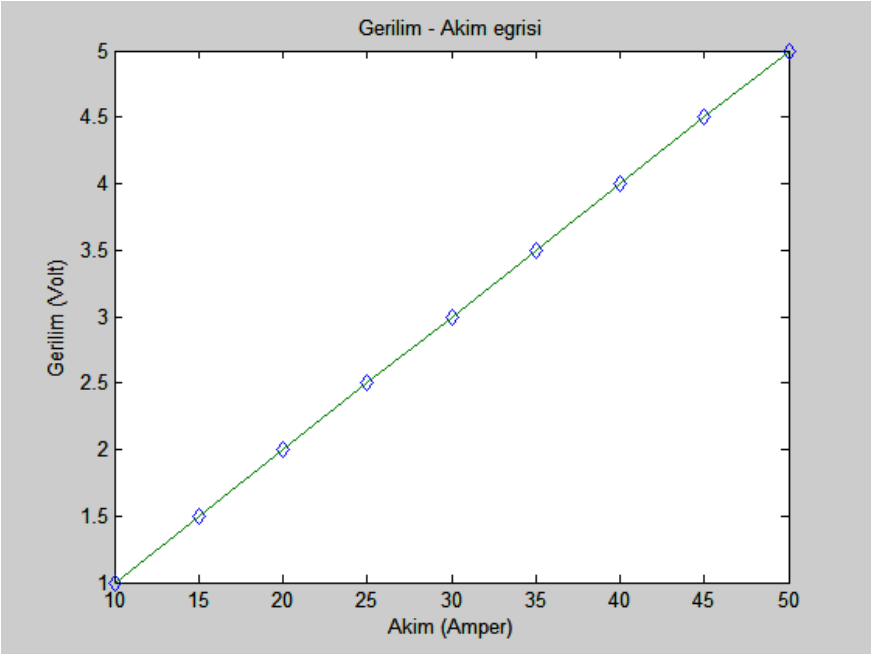
Örnek 5.4 de verilen grafiği tekrar çiziniz fakat veri noktalarına elmas işaretleri koyunuz.

Çözüm 5.5

Yeni program listesi aşağıda verilmiştir:

```
>> V=10:5:50;
>> plot(V,A);
>> plot(V,A,'p')
>> V=10:5:50;
>> A = 1:0.5:5;
>> plot(V,A,'d',V,A,'-')
>> xlabel('Akim (Amper)');
>> ylabel('Gerilim (Volt)');
>> title('Gerilim - Akim egrisi');
>>
```

Yeni grafik Şekil 5.6 da gösterilmiştir.



Şekil 5.6 Veri noktaları elmas işareti ile birleştirilmiştir

Şimdiye kadar çizmiş olduğumuz grafiklerin kapalı bir kutu içerisine alındığını görmüş olduk. **box off** komutu ile bu kutuyu kaldırıp sadece x-y eksenlerini bırakabiliriz. Şekil 5.7 de **box off** komutunun etkisi gösterilmiştir.

Örnek 5.6

Örnek 5.4 deki grafiği tekrar çiziniz ve veri noktalarını kırmızı elmas işaretle gösteriniz.

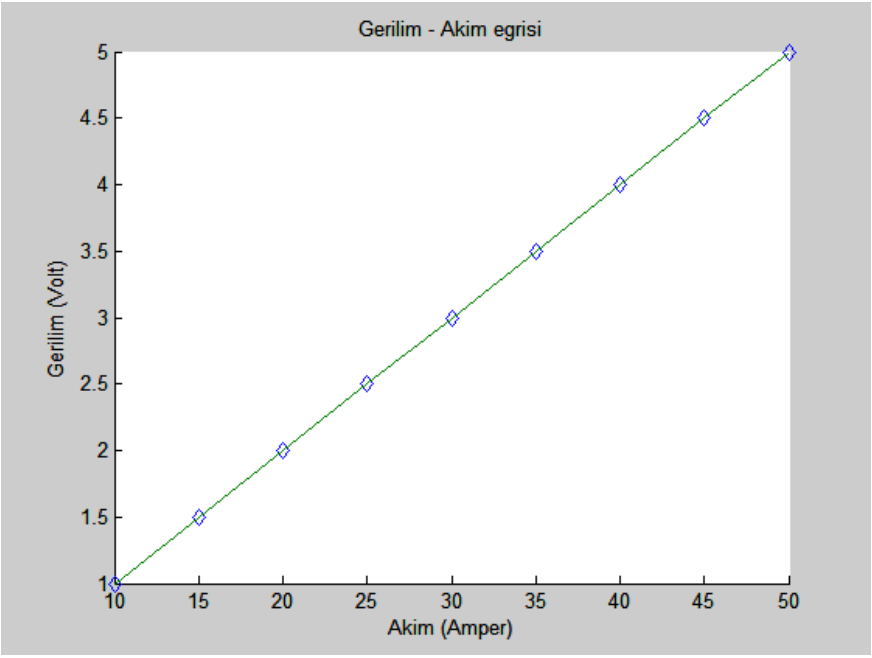
Çözüm 5.6

İstenilen program listesi aşağıda verilmiştir:

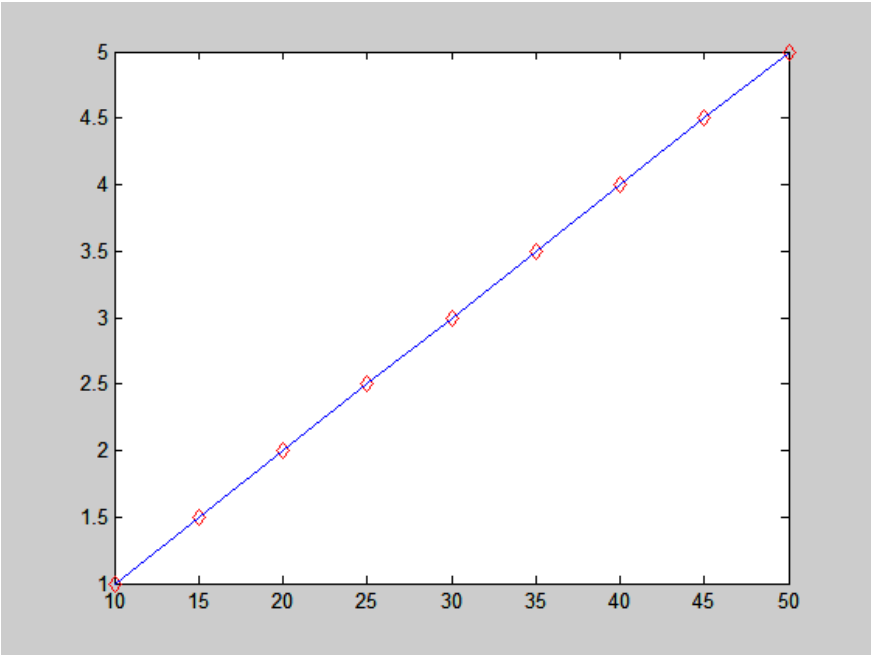
```
>> V=10:5:50;
>> plot(V,A);
>> plot(V,A,'p')
>> V=10:5:50;
>> A = 1:0.5:5;
>> plot(V,A,'rd',V,A,'-')
>> xlabel('Akim (Amper)');
>> ylabel('Gerilim (Volt)');
>> title('Gerilim - Akim egrisi');
>>
```

Bu programın plot komutunda elmas işaretlerin kırmızı renk olması için '**r**' harfi ilave edilmiştir.

Yeni grafik Şekil 5.8 de gösterilmiştir.



Şekil 5.7 **box off** komutu ile grafik etrafındaki kutu kaldırılmış olur



Şekil 5.8 Veri işaretleri kırmızı renk yapılmıştır

5.3 Polinom Çizimi

Polinom grafik çizimi oldukça önemli bir konudur. Daha önce görmüş olduğumuz **polyval** fonksiyonunu kullanarak kolaylıkla istediğimiz polinomun grafiğini çizebiliriz.

Örnek 5.7

X değişkeni –10 dan +10 a kadar değiştikçe aşağıdaki polinomum grafiğini çiziniz.

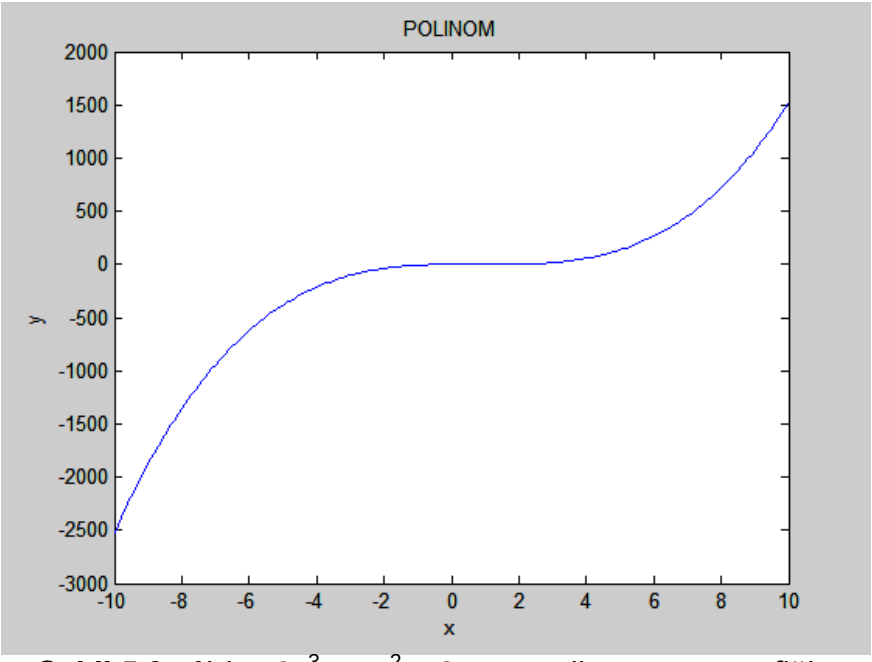
$$f(x) = 2x^3 – 5x^2 + 2x + 1$$

Çözüm 5.7

x değerlerini x isimli bir vektörde saklayıp grafiği kolaylıkla çizebiliriz.

Program listesi aşağıda ve grafik Şekil 5.9 da verilmiştir. Programda x adımları 0.1 olarak alınmıştır:

```
>> x = -10:0.1:10;
>> f = [2 -5 2 1];
>> p = polyval(f,x);
>> plot(x,p), xlabel('x'), ylabel('y'), title('POLINOM')
>>
```



Şekil 5.9 $f(x) = 2x^3 – 5x^2 + 2x + 1$ polinomunun grafiği

Yukarıdaki programı şu şekilde de yazabiliriz:

```
>> x = -10:0.1:10;
>> f = [2 -5 2 1];
>> plot(x, polyval(f,x)), xlabel('x'), ylabel('y'), title('POLINOM')
>>
```

Örnek 5.8

x değişkeni 0 dan 6 ya kadar 0.01 aralıklarında değiştikçe aşağıdaki fonksiyonum grafiğini çiziniz. Grafiğe grid ilave ediniz ve grafiği kutu içerisine almayınız (sadece x-y eksenlerini gösteriniz).

$$f(x) = e^{-1.1x}. \sin(5x + 1)$$

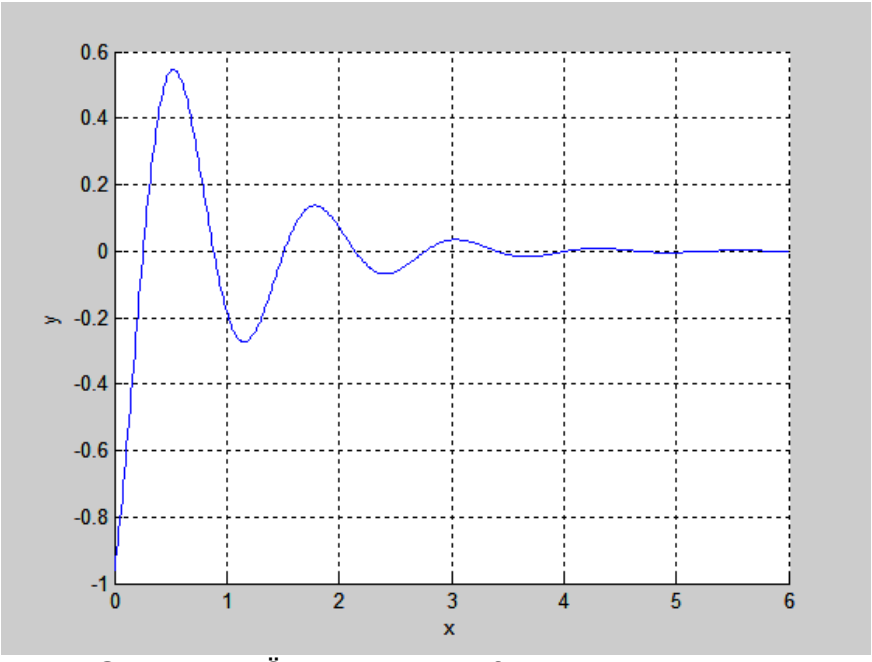
Çözüm 5.8

Programın listesi aşağıda verilmiştir. x değişkeni yatay

eksen değerlerini, f değişkeni de dikey eksen değerlerini saklamaktadır.

Çizilmiş olan grafik Şekil 5.10 da gösterilmiştir:

```
>> x = 0:0.01:6;
>> f = exp(-1.1*x).*sin(5*x+5);
>> plot(x,f), xlabel('x'), ylabel('y'). box off, grid
>>
```



Şekil 5.10 Örnek 5.8 in grafiği

5.4 Birden Fazla Grafiğin Ayni Eksenlerde Çizimi

MATLAB'ı kullanarak birden fazla grafiği ayni eksenlerde çizebiliriz. Böylece, değişik grafikleri kolaylıkla karşılaştırma imkanımız olmuş olur.

Örnek 5.9

Aşağıdaki iki fonksiyonun grafiklerini ayni eksenlere çiziniz:

y(x) = sinh(x)
g(x) = cosh(x)

Çözüm 5.9

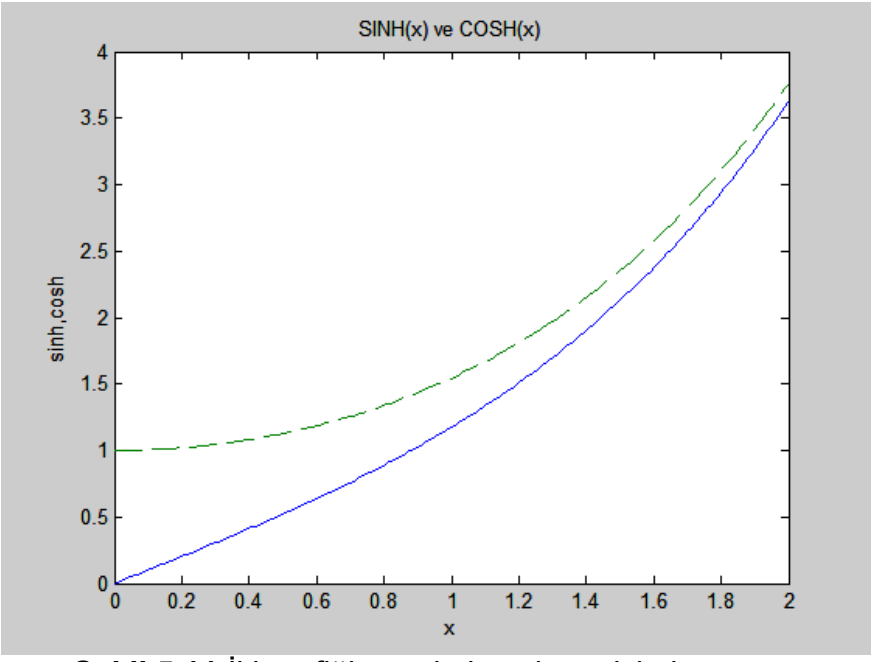
Bu örnekte x ekseni değerleri 0 dan 2 ye kadar ve 0.01 aralıklarla alınmıştır. Daha sonra y ve g fonksiyonlarının x noktalarındaki değerleri bulunmuştur. Plot komutu ile grafik çizilmiştir. Birinci grafikte değişkenler x-y olarak alınmış, ikinci grafikte ise x-g olarak alınmıştırlar. İki grafiğin de ayni olmamaları için ikinci grafik kesik çizgili olarak çizilmiştir.

Program listesi aşağıda verilmiştir. Çizilen grafik ise Şekil 5.11 de gösterilmiştir:

```
>> x = 0:0.01:2;
>> y = sinh(x);
>> g = cosh(x);
>> plot(x,y,x,g,'--')
>>xlabel('x'),ylabel('sinh,cosh'), title('SINH(x) ve COSH(x)')
>>
```

Şekil 5.11 e bakıldığında hangi grafiğin hangi fonksiyona ait olduğu belli değildir. İkinci grafiği kesik çizgili olarak çizdik fakat sadece bu şekile bakılınca hangi grafiğin hangi fonksiyona ait olduğunu söylemek mümkün değildir.

legend komutunu kullanarak grafikleri tanımlamak mümkündür. Bu komut grafik içerisine küçük bir kutu çizip hangi grafiğin hangi fonksiyona ait olduğunu belirtir. Bir örnek aşağıda verilmiştir.



Şekil 5.11 İki grafiğin ayni eksenlere çizimi

Örnek 5.10

Örnek 5.9 da çizilmiş olan grafikleri kolay tanınmaları için legend komutunu kullanarak işaretleyiniz.

Çözüm 5.10

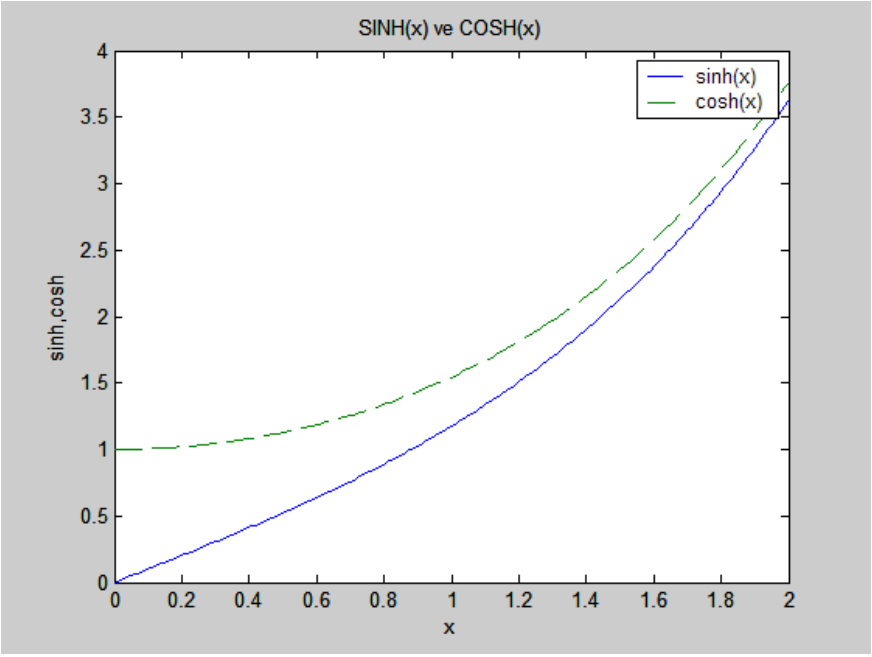
Yeni program listesi aşağıda verilmiştir:

```
>> x = 0:0.01:2;
>> y = sinh(x);
>> g = cosh(x);
>> plot(x,y,x,g,'--')
>>xlabel('x'),ylabel('sinh,cosh'), title('SINH(x) ve COSH(x)')
>> legend('sinh(x)','cosh(x)')
```

legend komutunun, kaç tane grafik varsa o kadar parametresi olmalı ve parametreler tek tırnak içerisine alınmalıdırlar. Birinci parametre plot komutu ile çizilen ilk

grafiği tanımlar, ikinci parametre ise plot komutu ile çizilen ikinci grafiği tanımlar.

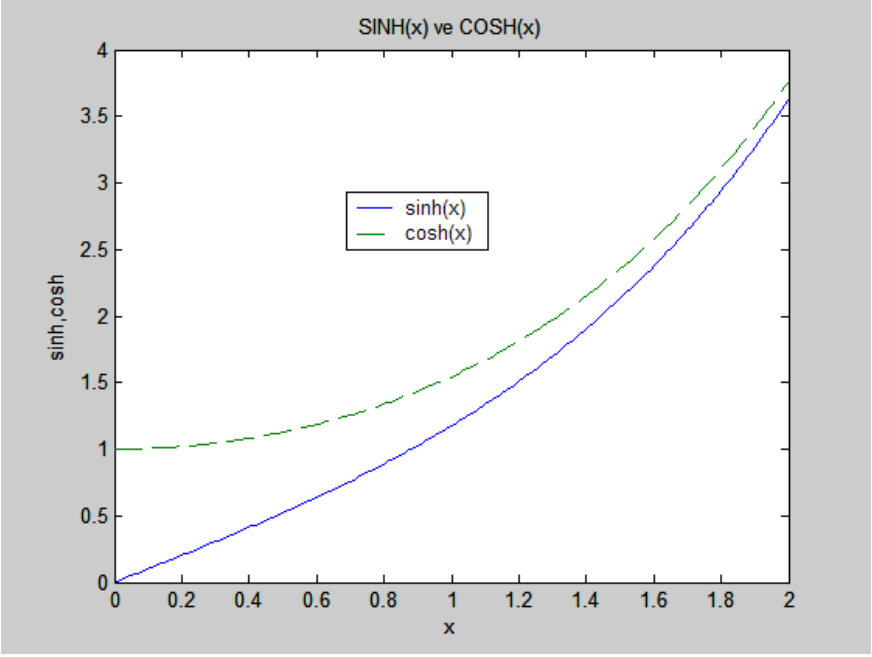
Çizilen grafik Şekil 5.12 de gösterilmiştir. **legend** kutusunun yerini değiştirmek mümkündür. Bunun için farenizi **legend** kutusu üzerine koyunuz ve sol tuşa basarak **legend** kutusunu istediğiniz yere koyunuz. Şekil 5.13 de legend kutusu grafiğin ortasına konmuştur.



Şekil 5.12 legend komutu ile grafikleri tanımlama

Grafikleri tanımlamanın başka bir yolu da **gtext** komutunu kullanmaktır. Bu komutun genel şekli **gtext('yazı')** olup belirtilmiş olan **yazı**'yı grafiğin istediğimiz yerine yerleştirebiliriz. Bu komut şu şekilde çalışır: grafiği çizdikten sonra gtext komutunu ve içerisine istediğiniz yazıyı yazınız. Enter tuşuna basınca grafik otomatik olarak önünüze çıkacak ve ayni zamanda yatay ve dikey iki tane koordinat çizgisi göreceksiniz. Bu çizgilerin kesiştikleri noktayı yazı'yı yazmak istediğiniz yere getiriniz ve fare tuşunu tıklayınız. Belirtmiş olduğunuz yazı istediğiniz yere yazılacaktır. Örneğin, Şekil

5.11 deki grafik Şekil 5.14 de tekrar çizilmiş ve **gtext('sinh(x) grafigi')** komutu verilerek sinh(x) grafiği yanına **sinh(x) grafigi** yazısı yazılmıştır. Ayni şekilde, **gtext('cosh(x) grafigi')** komutu verilerek cosh(x) grafiği yanına **cosh(x) grafigi** yazısı yazılmıştır.



Şekil 5.13 legend kutusunun yeri fare ile değiştirilmiştir

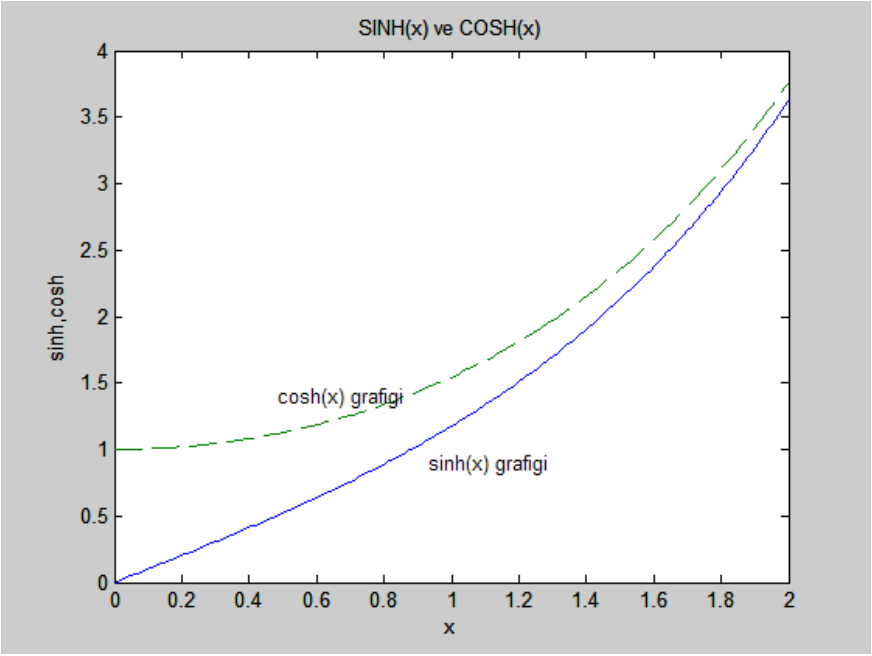
5.5 text Komutu

text komutunu kullanarak grafiğin istenilen x-y koordinatlarına istediğimiz yazıyı yazabiliriz. Bu komutun genel şekli şöyledir: text(x,y,'yazı'). Aşağıda bir örnek verilmiştir.

Örnek 5.11

x değişkeni 0 dan 2π ya kadar değiştikçe sin(2x) fonksiyonunun grafiğini çiziniz ve grafiğin (2.0, 0.5) noktasına

sin(2x) grafiği yazısını yazınız.



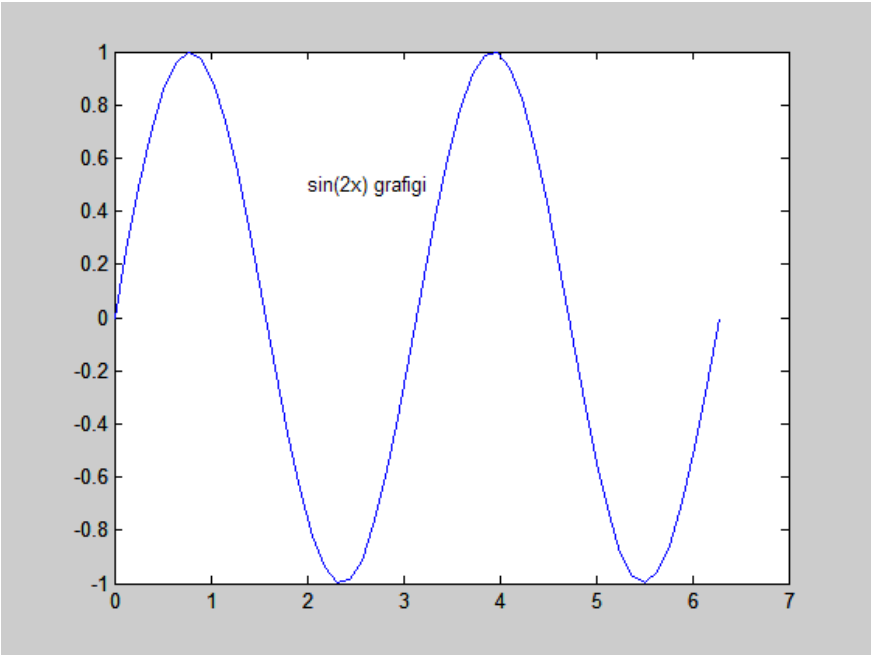
Şekil 5.14 gtext komutu ile grafikleri tanımlama

Çözüm 5.11

Grafiği çizmek için şu komutlar kullanılmıştır:

```
>> x = linspace(0,2*pi,50);
>> y = sin(2*x);
>> plot(x,y)
>> xlabel('x');
>> ylabel('sin(2x)');
>> text(2.0,0.5,'sin(2x) grafigi')
>>
```

Çizilmiş olan grafik Şekil 5.15 de gösterilmiştir.



Şekil 5.15 Grafikte (2.0, 0.5) noktasına **sin(2x) grafiği** yazılmıştır.

5.6 LineWidth Komutu

Bu komut çizilen grafiğin çizgi kalınlığını belirler. Bu komut belirtilmediğinde çizgi kalınlığı 0.5 nokta olarak alınır (1 nokta = 1.72 inç). **LineWidth** komutu plot komutu içerisinde kullanılır. Bir örnek verilmiştir.

Örnek 5.12

Örnek 5.11 de verilen grafiği tekrar çizerek çizgi kalınlığını 3 nokta olarak belirtiniz.

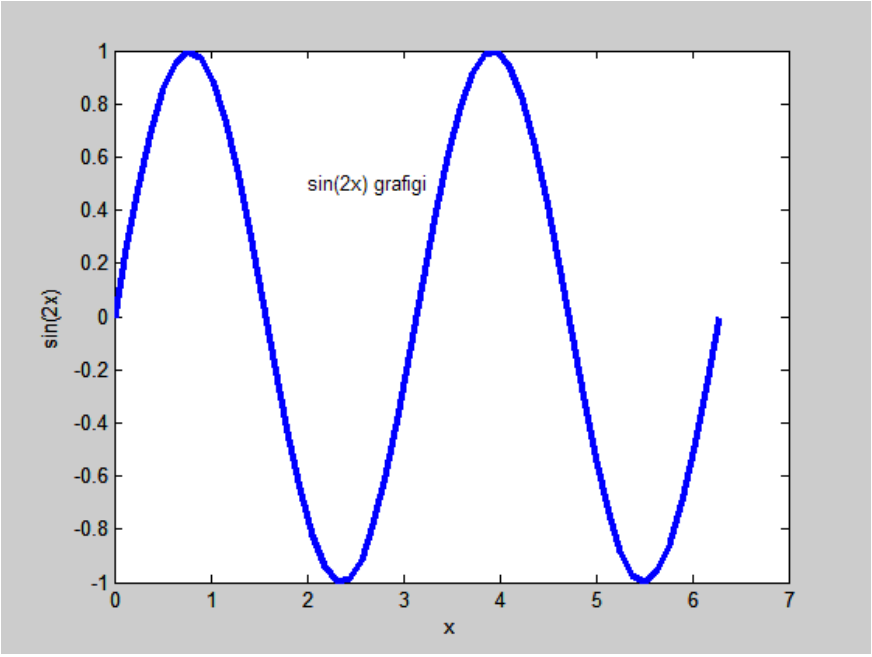
Çözüm 5.12

Gerekli MATLAB komutları aşağıda verilmiştir:

```
>> x = linspace(0,2*pi,50);
>> y = sin(2*x);
```

```
>> plot(x,y,’LineWidth’,3)
>> xlabel('x');
>> ylabel('sin(2x)');
>> text(2.0,0.5,'sin(2x) grafigi')
>>
```

Çizilmiş olan yeni grafik Şekil 5.16 da gösterilmiştir:



Şekil 5.16 **LineWidth** komutu ile çizgisi kalınlaştırılmış grafik

5.7 FontSize Komutu

Bazı uygulamalarda grafiğin başlık fontunu büyütmek isteyebiliriz. Bunun için, title komutu içerisinde **FontSize** komutunu kullanabiliriz. Bu komut kullanılmazsa font büyüklüğü 12 punto olarak alınır. Aşağıda bir örnek verilmiştir.

Örnek 5.13

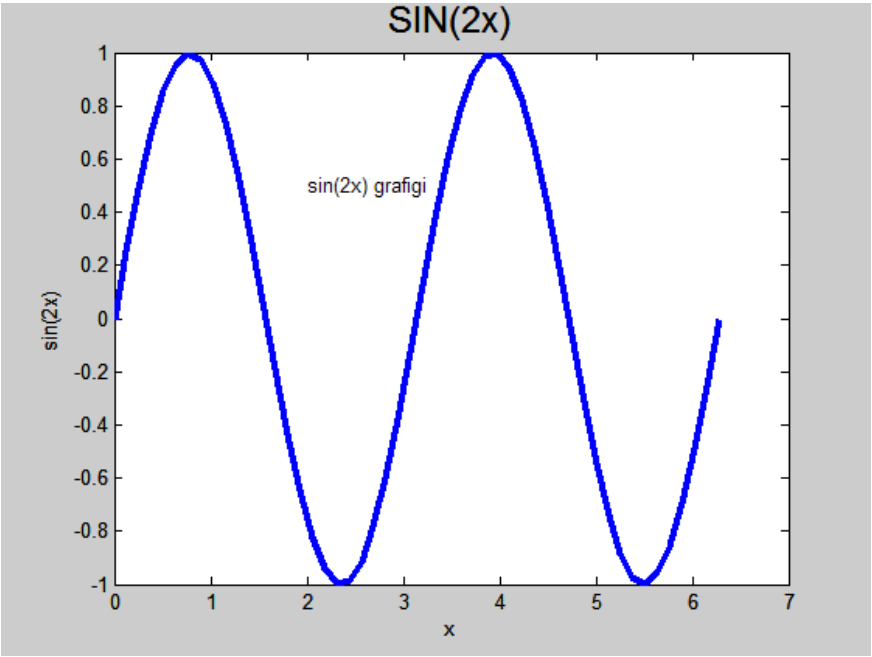
Örnek 5.12 de verilen grafiği tekrar çizerek grafiğe SIN(2x) başlığını yazınız. Başlığın puntosunu 18 olarak belirtiniz.

Çözüm 5.13

Gerekli MATLAB komutları aşağıda verilmiştir:

```
>> x = linspace(0,2*pi,50);
>> y = sin(2*x);
>> plot(x,y,'LineWidth',3)
>> xlabel('x');
>> ylabel('sin(2x)');
>> text(2.0,0.5,'sin(2x) grafigi')
>> title('SIN(2x)','FontSize',18)
>>
```

Çizilmiş olan yeni grafik Şekil 5.17 de gösterilmiştir:



Şekil 5.17 Başlığı 18 punto yapılmış olan grafik

5.8 MarkerSize Komutu

Bu komut, veri noktalarına konan işaretlerin büyüklüğünü ayarlamaktadır. Bu komutun geçerli olması için veri noktalarına işaretler konması gerekmektedir. Aşağıdaki örnekte işaret büyüklüğü 14 punto nokta olarak seçilmiştir.

Örnek 5.14

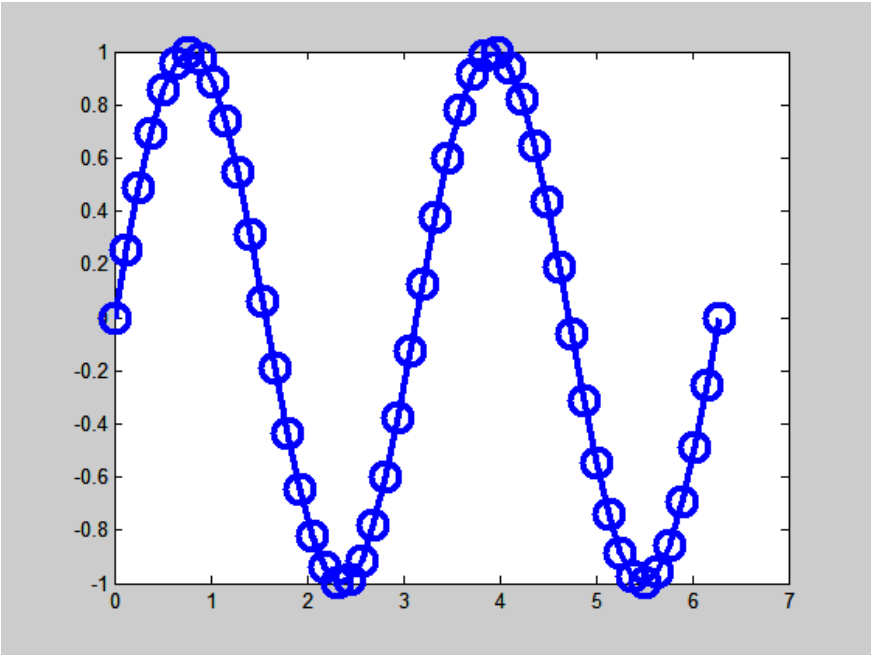
Örnek 5.11 de verilen grafiği tekrar çizerek veri noktalarını 12 nokta büyüklüğünde ‘o’ işaretleri ile birleştiriniz.

Çözüm 5.14

Gerekli MATLAB komutları aşağıda verilmiştir:

```
>> x = linspace(0,2*pi,50);
>> y = sin(2*x);
>> plot(x, y, ’-o’, ’LineWidth’, 3 , ’MarkerSize’,14)
>> xlabel('x');
>> ylabel('sin(2x)');
>> text(2.0,0.5,'sin(2x) grafigi')
>> title(‘SIN(2x)’,’FontSize’,18)
>>
```

Çizilmiş olan grafik Şekil 5.18 de gösterilmiştir.



Şekil 5.18 **MarkerSize** komutu ile veri işaretleri Büyütülmüştür.

5.9 fplot Komutu

Herhangibir fonksiyonun grafiğini çizmek için çok kullanışlı olan bu komut grafiği çizilecek olan fonksiyonu analiz yapar ve grafiği en optimum sayıda noktalar alarak çizmeye çalışır. Grafiğini çizmek istediğimiz fonksiyon **fplot** komutu içerisine doğrudan yazılır. Komutun genel şekli şöyledir:

```
fplot("fonksiyon", [xmin xmax])
```

Örnek 5.15

fplot komutunu kullanarak $f(x) = x^3 + x^2 - 2x + 1$ in grafiğini $x = -10$ ve $x = 10$ aralığında çiziniz.

Çözüm 5.15

İstenilen grafiği çizmek için gerekli olan MATLAB komutları aşağıda verilmiştir.

```
>> f = ‘x^3 + x^2 - 2*x + 1’;
>> fplot(f, [-10 10])
>>
```

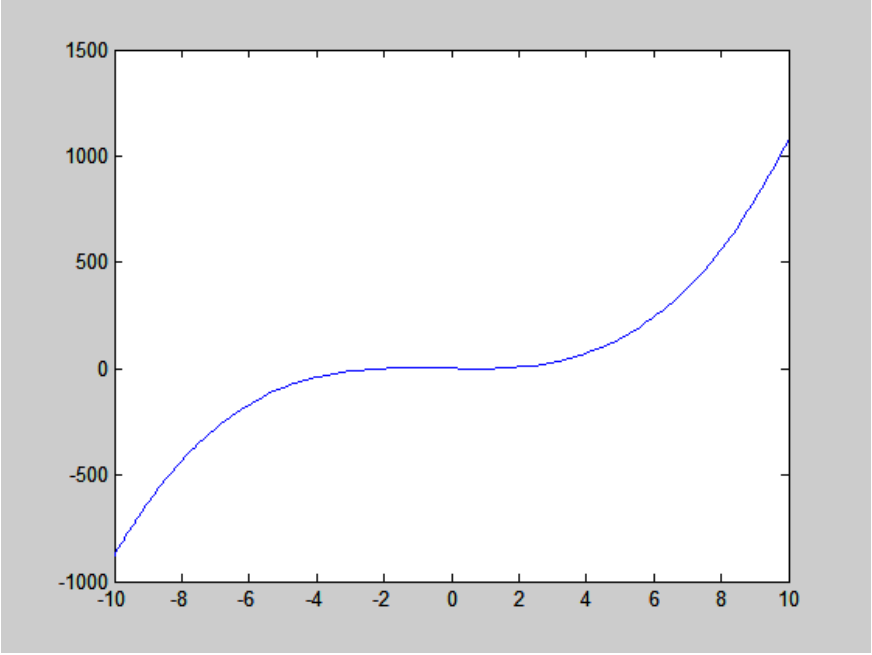
Yukarıdaki komutlar ayni satırda şu şekilde de yazılabilir:

```
>> fplot('x^3 + x^2 - 2*x + 1',[-10 10])
```

Çizilmiş olan grafik Şekil 5.19 da gösterilmiştir:

Örnek 5.16

$f(x) = e^{-0.4x} \sin(2x)$ ın grafiğini x = 0 ve x = 10 arasinda çiziniz.



Şekil 5.19 fplot komutunu kullanarak çizilen grafik

Çözüm 5.16

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> f = 'exp(-0.4*x)*sin(2*x)';
>> fplot(f,[0 10])
>>
```

Çizilmiş olan grafik ise Şekil 5.20 de gösterilmiştir.

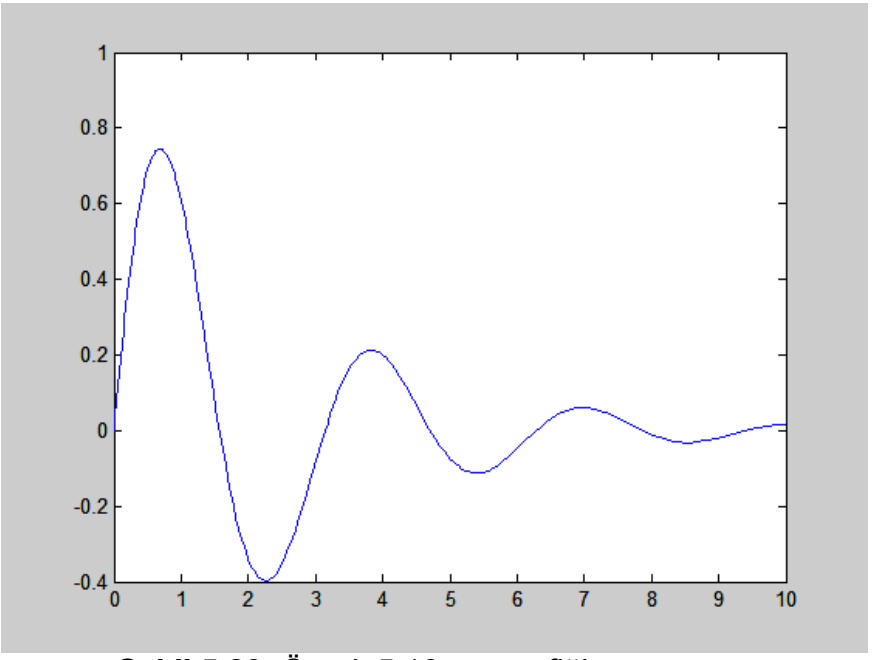
5.10 subplot Komutu – Birden Fazla Küçük Grafik Çizmek

subplot komutunu kullanarak birden fazla küçük boyutta grafikler çizmemiz mümkündür. Bu komut, birden fazla grafiği karşılaştırm istediğimiz durumlarda çok faydalı olmaktadır. **subplot** komutunu kullanarak normal grafik alanını sıra ve sütunlara ayırırız. Daha sonra istediğimiz sırayı ve sütunu seçerek istediğimiz grafiği bu seçilen alana çizeriz. subplot komutunun genel şekli şöyledir:

```
subplot(m, n, q)
```

Burada m toplam sıra numarası, n toplam sütun numarası, ve q ise grafiğik sıra numarasıdır. Örneğin, subplot(2, 3, 4) komutu grafik için 2 sıra ve 3 sütun ayırır ve bir sonraki grafiği 4. üncü ayrılmış yere çizer. Burada 4.üncü ayrılmış yer ikinci sıranın ilk boşluğudur.

Aşağıda bazı örnekler verilmiştir.



Şekil 5.20 Örnek 5.16 nın grafiği

Örnek 5.17

Aşağıdaki 4 grafiği küçük grafikler halinde ayni grafik alanına çiziniz:

sinh(x) tanh(x) sech(x) sinh(2x)

x değerini 0 ve 5 arasında, 0.01 aralıklarla alınız.

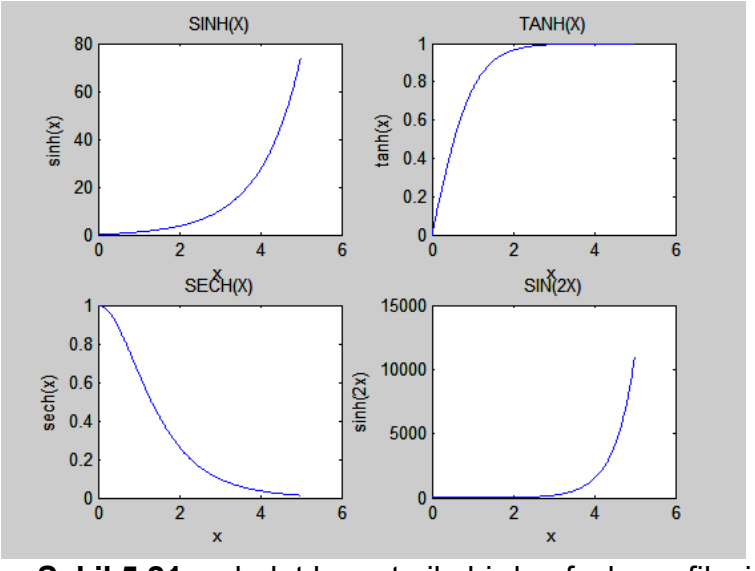
Çözüm 5.17

Bu örnekte 4 grafiği de ayni grafik alanına çizmek için **subplot** komutu kullanılacaktır. Grafikler 2 sıra ve 2 sütun şeklinde çizileceklerdir. İstenilen MATLAB komutları aşağıda verilmiştir:

```
>> x = 0:0.01:5;
>>%
>>% Birinci grafigi ciz
```

```
>>%
>> subplot(2,2,1);
>> plot(x,sinh(x)), xlabel('x'), ylabel('sinh(x)'),title('SINH(X)')
>>%
>>% Ikinci grafigi ciz
>>%
>> subplot(2,2,2)
>> plot(x,tanh(x)), xlabel('x'), ylabel('tanh(x)'), title('TANH(X)')
>>%
>>% Ucuncu grafigi ciz
>>%
>> subplot(2,2,3)
>> plot(x,sech(x)), xlabel('x'), ylabel('sech(x)'), title('SECH(X)')
>>%
>>%Dorduncu grafigi ciz
>>%
>> subplot(2,2,4)
>> plot(x,sinh(2*x)), xlabel('x'), ylabel('sinh(2x)'), title('SINH(2X)')
>>
```

Çizilmiş olan grafikler Şekil 5.21 de gösterilmiştir. **subplot** komutunda kullanılan sıra ve sütun sayısını değiştirmekle grafiklerde olan değişimleri göstermek için aşağıda bir örnek verilmiştir.



Şekil 5.21 subplot komutu ile birden fazla grafik çizimi

Örnek 5.18

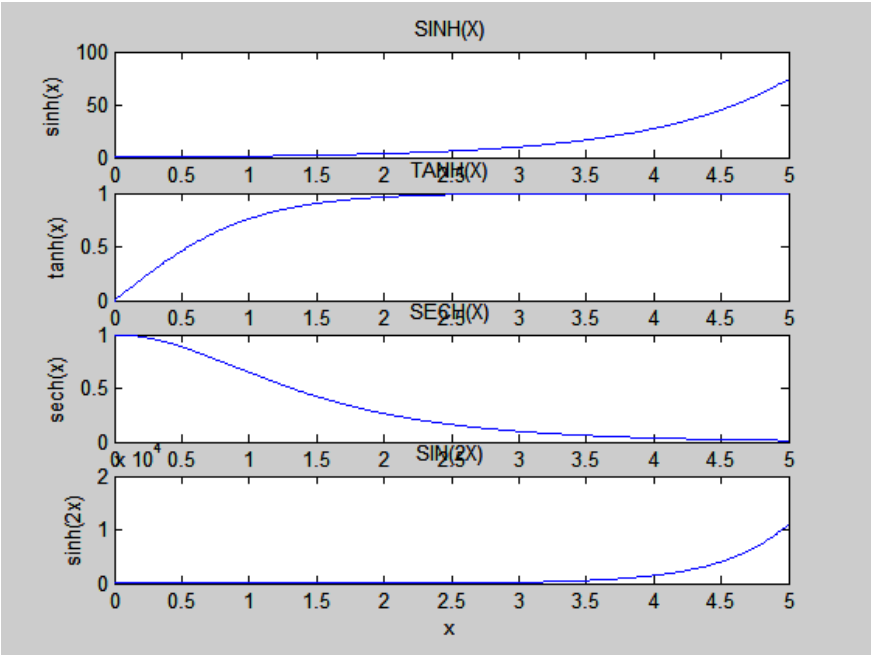
Örnek 5.17 deki grafikleri tekrar çiziniz fakat grafikleri 4 sıra ve 1 sütun şeklinde çiziniz.

Çözüm 5.18

İstenilen MATLAB komutları aşağıda verilmiştir:

```
>> x = 0:0.01:5;
>> subplot(4,1,1);
>> plot(x,sinh(x)), xlabel('x'), ylabel('sinh(x)'),title('SINH(X)')
>>%
>> subplot(4,1,2);
>> plot(x,tanh(x)), xlabel('x'), ylabel('tanh(x)'), title('TANH(X)')
>>%
>> subplot(4,1,3);
>> plot(x,sech(x)), xlabel('x'), ylabel('sech(x)'),title('SECH(X)')
>>%
>> subplot(4,1,4);
>>plot(x,sinh(2*x)),xlabel('x'),ylabel('sinh(2x)'),title('SIN(2X)')
>>
```

Çizilmiş olan grafikler Şekil 5.22 de gösterilmiştir.



Şekil 5.22 **subplot** komutu ile 4 sıralı ve 1 sütunlu grafik çizimi

Örnek 5.19

subplot komutunu kullanarak ayni grafik ortamına üç tane grafik çiziniz. Üst grafik 5Hz sinüs dalgası, orta grafik 10Hz sinüs dalgası, ve alttaki grafik de iki sinüs dalgasının toplamını göstersin. Her iki sinüs dalgalarının da yükseklikleri 5 olsun.

Çözüm 5.19

İlk olarak zaman 0 dan 1 saniyeye kadar, 0.01 aralıkları ile **zaman** isimli bir vektörde saklanmıştır. Daha sonra **frekans1** 5Hz olarak tanımlanmış ve ilk sinüs dalgasının değerleri **sinus1** vektöründe saklanmıştır. Burada **sinus1** in değerleri şu şekilde hesaplanmıştır:

sinus1 = A.sin(2πf/t)

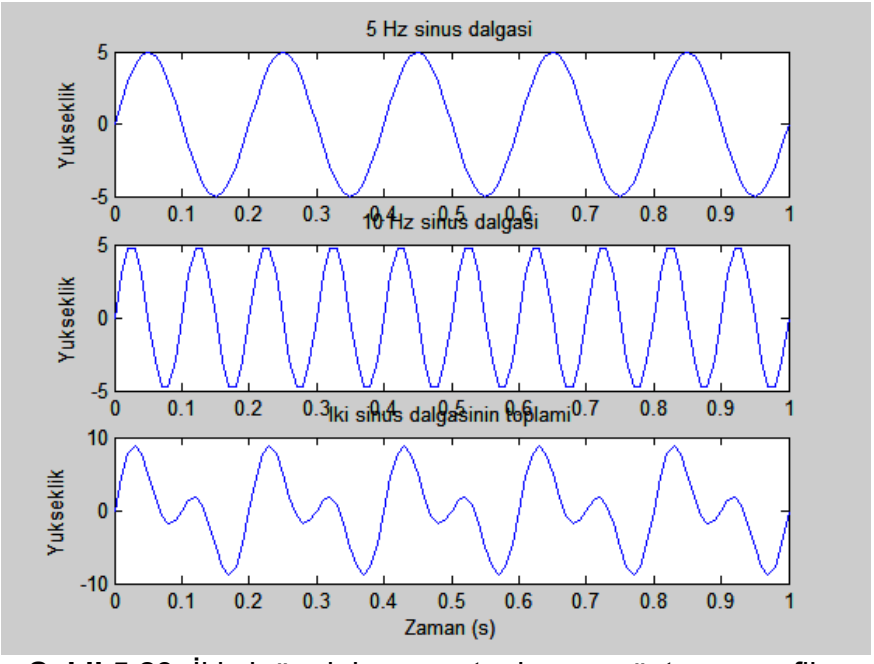
A katsayısı dalganın yüksekliği, **f** dalganın frekansı, ve **t** ise zamandır (saniye).

Daha sonra ikinci sinüs dalgasının frekansı 10Hz yapılıp bu dalganın değerleri **sinus2** isimli bir vektörde saklanmıştır. Son olarak, **subplot** komutu kullanılarak her iki sinüs dalgası ve bu dalgaların toplamının grafikleri çizilmiştir.

İstenilen MATLAB komutları aşağıda verilmiştir:

```
>> zaman = [0: 0.01: 1];
>> frekans1 = 5;
>> sinus1 = A*sin(2*pi*frekans1*zaman);
>> frekans2 = 10;
>> sinus2  = A*sin(2*pi*frekans2*zaman);
>>%
>>% Simdi grafikleri ciz
>>%
>>% Birinci sinus dalgasi
>>%  ===============
>> subplot(3,1,1)
>> plot(zaman,sinus1)
>> title('5 Hz sinus dalgasi')
>> ylabel('Yukseklik')
>>%
>>% Ikinci sinus dalgasi
>>% ==============
>> subplot(3,1,2);
>> plot(zaman,sinus2)
>> title('10 Hz sinus dalgasi')
>> ylabel('Yukseklik')
>>%
>>% Simdi iki sinus dalgasinin toplamini ciz
>>%  ============================
>> subplot(3,1,3);
>> plot(zaman,sinus1+sinus2)
>> title('Iki sinus dalgasinin toplami')
>> xlabel('Zaman (s)')
>> ylabel('Yukseklik')
>>
```

Grafiklerin çizimi Şekil 5.23 de gösterilmiştir.



Şekil 5.23 İki sinüs dalgasının toplamını gösteren grafik

Örnek 5.20

Verilen iki sinüs dalgası üzerine işlemler yapıp bu dalgaların grafiklerini çizecek bir MATLAB m-dosyası programı yazınız. Dalgaların yüksekliklerini, her iki dalganın frekansını, ve yapılacak olan işlem klavyeden girilmelidir. Dalgaları 1 saniyelik bir yatay eksen kullanarak çiziniz ve her dalga için veri aralığının 0.01 saniye (10ms) olduğunu kabul ediniz. Dalgaları birbiri altına ayni grafik ortamına çiziniz.

Dosyaya **grafikler.m** ismini veriniz.

Çözüm 5.20

Bu örnek için ilk olarak dalga yüksekliklerini, her dalganın frekansını, ve yapılacak olan işlemi klavyeden okumamız gerekir. Daha sonra istenilen işlemi yapıp grafiklerimizi çizebiliriz.

grafikler.m dosyasının listesi aşağıda verilmiştir. Programın başında, herhangibir 0 ile bölme işleminden doğacak olan hatayı önlemek için "*warning off MATLAB:divideByZero*" komutu yazılmıştır. Daha sonra ekran temizlenip her iki dalgaların da yğkseklikleri ve frekansları istenmiştir. Bunu takiben, dalgalar üzerine yapılacak olan işlem klavyeden istnemis ve daha sonra grafikler çizilmiştir.

```
%
% Bu program verilen iki sinus dalgasi uzerine
% islemler yapar ve her iki sinus dalgasini da ve
% neticeyi de ayni grafik ortaminda cizer. Dalgalarin
% yukseklikleri, frekanslari, ve yapilacak islem
% klavyeden girilmektedir.
warning off MATLAB:divideByZero
clc
%
% Birinci dalganin yuksekligini ve frekansini
% klavyeden oku.
%
a1 = input('Birinci dalganin yuksekligi: ');
f1 = input('Birinci dalganin frekansi (Hz): ');
disp(' ');
%
% Simdi ikinci dalganin yuksekligini ve frekansini
% klavyeden oku.
%
a2 = input('Ikinci dalganin yuksekligi: ');
f2 = input('Ikinci dalganin frekansi (Hz): ');
%
% Simdi zaman araligini zaman vektorunde sakla ve
% her iki dalganin da degerlerini sinus1 ve sinus2
% gibi iki vektorde sakla.
%
zaman = [0: 0.01: 1];
sinus1 = a1*sin(2*pi*f1*zaman);
sinus2 = a2*sin(2*pi*f2*zaman);
%
% Simdi yapilacak olan islemi klavyeden okuyunuz
%
disp(' ');
```

```
disp(' ISLEMI SECINIZ');
disp(' ==============');
disp('1. Dalgalari topla');
disp('2. Dalgalari cikar');
disp('3. Dalgalari carp');
disp('4. Dalgalari bol');
disp(' ');
secim = input('Secenek: ');
%
switch secim
case 1
   netice = sinus1 + sinus2;
case 2
   netice = sinus1 - sinus2;
case 3
   netice = sinus1 .* sinus2;
case 4
   netice = sinus1 ./ sinus2;
end
%
% Ilk iki sinus dalgasinin grafiklerini ciz
%
subplot(3,1,1);
plot(zaman,sinus1);
ylabel('yukseklik');
grid on
%
% Basligi ilk grafigin ustune yaz
%
switch secim
case 1
  title('Iki Sinus dalgasinin Toplami');
case 2
  title('Iki Sinus dalgasinin farki');
case 3
  title('Iki Sinus dalgasinin Carpimi');
case 4
  title('Iki Sinus dalgasinin Bolumu');
end
%
% Ikinci grafigi ciz
%
subplot(3,1,2);
```

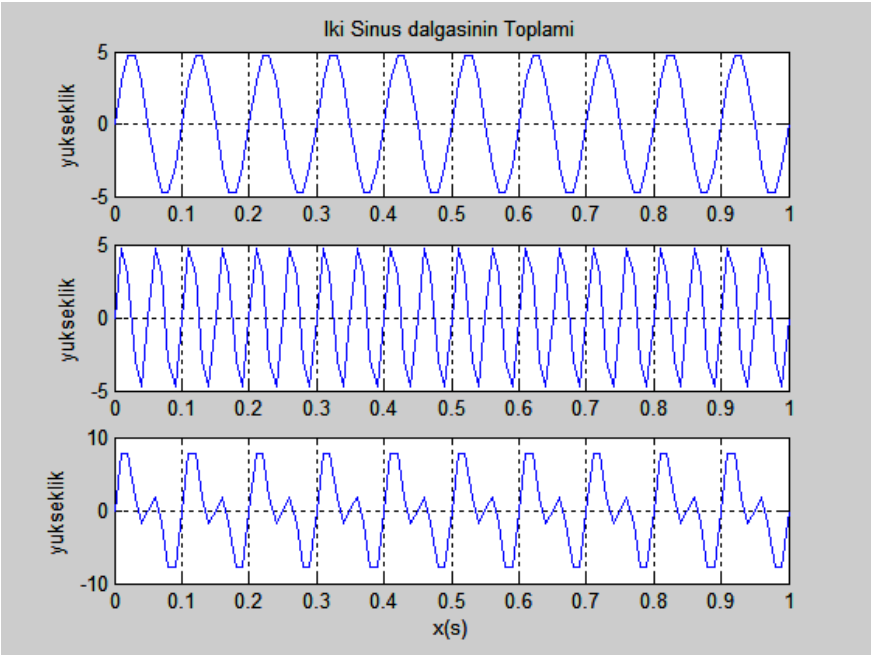
```
plot(zaman,sinus2);
ylabel('yukseklik');
grid on
%
% Neticeyi ciz
%
subplot(3,1,3);
plot(zaman,netice);
ylabel('yukseklik');
grid on
xlabel('x(s)')
disp("Grafikler cizilmistir…");
```

Program, >> komutundan sonra **grafikler** yazarak çalışır.

Programın çalışmasına örnekler aşağıda verilmiştir:

```
Birinci dalganin yuksekligi: 5
Birinci dalganin frekansi (Hz): 10

Ikinci dalganin yuksekligi: 5
Ikinci dalganin frekansi (Hz): 20

 ISLEMI SECINIZ
 ==============
1. Dalgalari topla
2. Dalgalari cikar
3. Dalgalari carp
4. Dalgalari bol

Secenek: 1
Grafikler cizilmistir...
>>
```

Çizilmiş olan grafik Şekil 5.24 de gösterilmiştir.

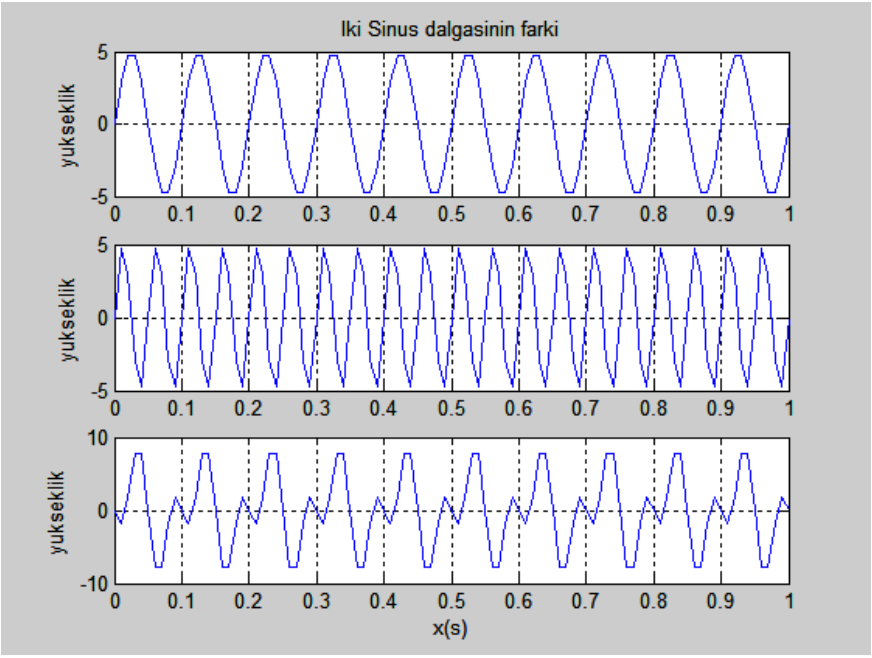


Şekil 5.24 Sinüs dalgalarının toplamı

Başka bir örnek:

```
Birinci dalganin yuksekligi: 5
Birinci dalganin frekansi (Hz): 10

Ikinci dalganin yuksekligi: 5
Ikinci dalganin frekansi (Hz): 20

 ISLEMI SECINIZ
 ==============
1. Dalgalari topla
2. Dalgalari cikar
3. Dalgalari carp
4. Dalgalari bol

Secenek: 2
Grafikler cizilmistir...
>>
```

Bu örnek için çizilen grafik Şekil 5.25 de gösterilmiştir.

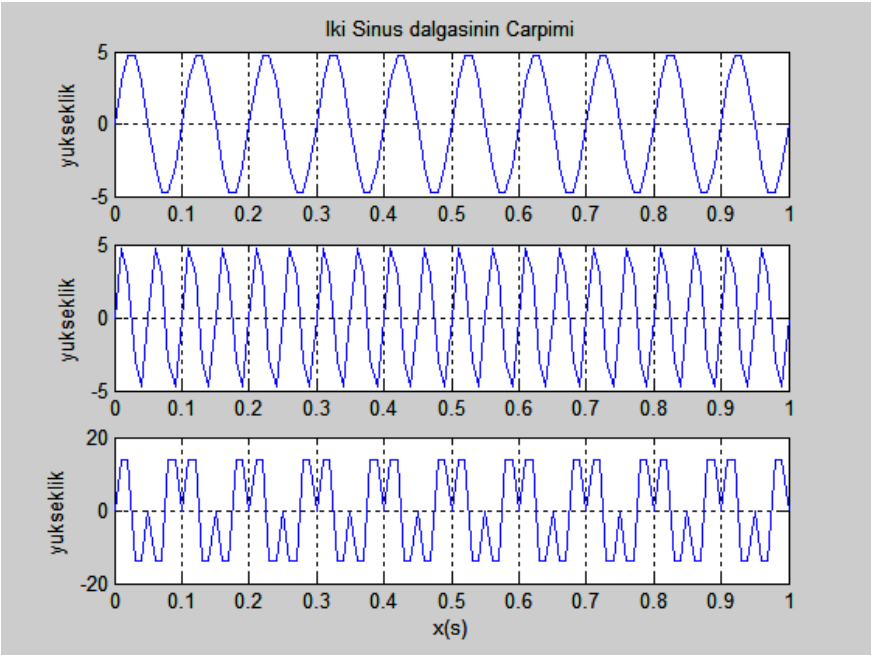


Şekil 5.25 Sinüs dalgalarının farkı

Sinüs dalgalarını çarpımınını gösteren örnek ise:

```
Birinci dalganin yuksekligi: 5
Birinci dalganin frekansi (Hz): 10

Ikinci dalganin yuksekligi: 5
Ikinci dalganin frekansi (Hz): 20

 ISLEMI SECINIZ
 ==============
1. Dalgalari topla
2. Dalgalari cikar
3. Dalgalari carp
4. Dalgalari bol

Secenek: 3
Grafikler cizilmistir...
>>
```

Bu örneğin grafiği Şekil 5.26 da gösterilmiştir.



Şekil 5.26 Sinüs dalgalarının çarpımı

5.11 print Komutu

print komutu çizilmiş olan bir grafiği yazıcıya gönderip kağıt üzerinde kopyesini çıkarmak için kullanılmaktadır. Bu komut direk olarak MATLAB çalışma alanından yazılabilir. Yazıcıya gönderilen grafik **orient** komutu kullanılarak portre olarak (**portrait**), uzun olarak (**tall**), veya yan olarak (**landscape**) çizilebilir:

```
>> orient portrait
```
veya
```
>> orient landscape
```
veya
```
>> orient tall
```

MATLAB'da yazıcıya grafik göndermenin bir başka yolu da grafik penceresi içerisinden **File -> Print** komutunu seçmektir.

Çizilmiş olan herhangibir grafiği örneğin bir word dosyasına aktarmak istersek, grafik penceresinden **Edit -> Copy Figure** komutunu seçmemiz gerekir. Normal olarak bir grafik bitmap formatı olarak kopyalanır. Fakat, grafik penceresinden **Edit -> Copy Options** komutunu seçerek başka bir formatı seçmemiz de mümkündür.

5.12 hold Komutu

Normal olarak **plot** komutu kullanıldığında mevcut eksenler silinir ve yeni baştan eksenler çizilir. **hold on** komutunu kullanarak, eksenleri silmeden grafik ortamımıza yeni grafikler ilave etmemiz mümkündür. Aşağıa bir örnek verilmiştir:

Örnek 5.21

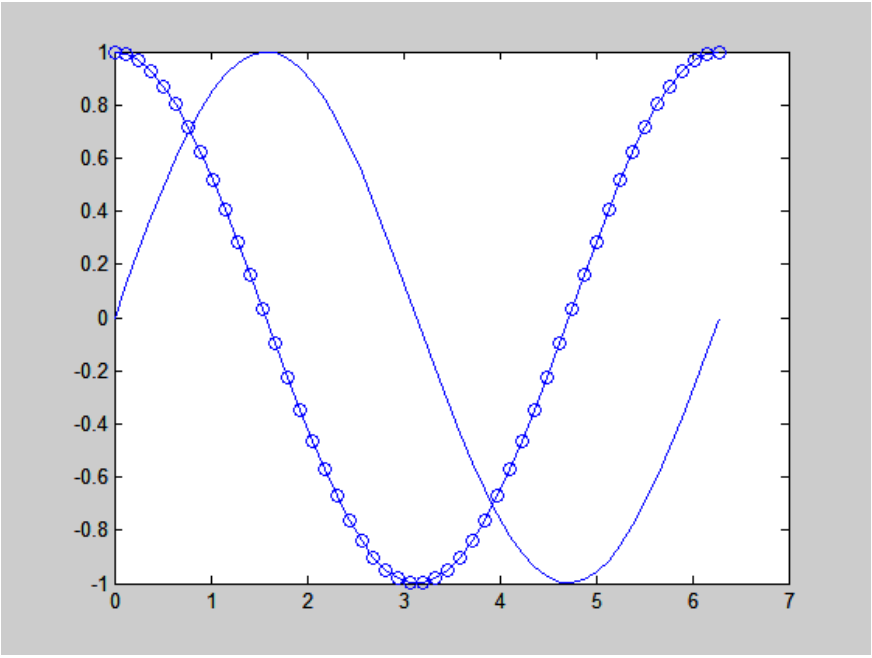
Hold on komutunu kullanarak, ilk olarak sinüs grafiği, daha sonra da kosinüs grafiğini çiziniz.

Çözüm 5.21

İstenilen MATLAB komutları aşağıda verilmiştir:

```
>> x =linspace(0,2*pi,50);
>> y=sin(x);
>> z=cos(x);
>> plot(x,y)                    % İlk grafigı cız
>> hold on                      % Eksenleri silme
>> plot(x,z,'-o')               % Ikıncı grafıgı cız
>> hold off
>>
```

Çizilen grafik Şekil 5.27 de gösterilmiştir.



Şekil 5.27 **hold on** komutu kullanılarak iki **plot** komutuyla iki grafik çizilmiştir.

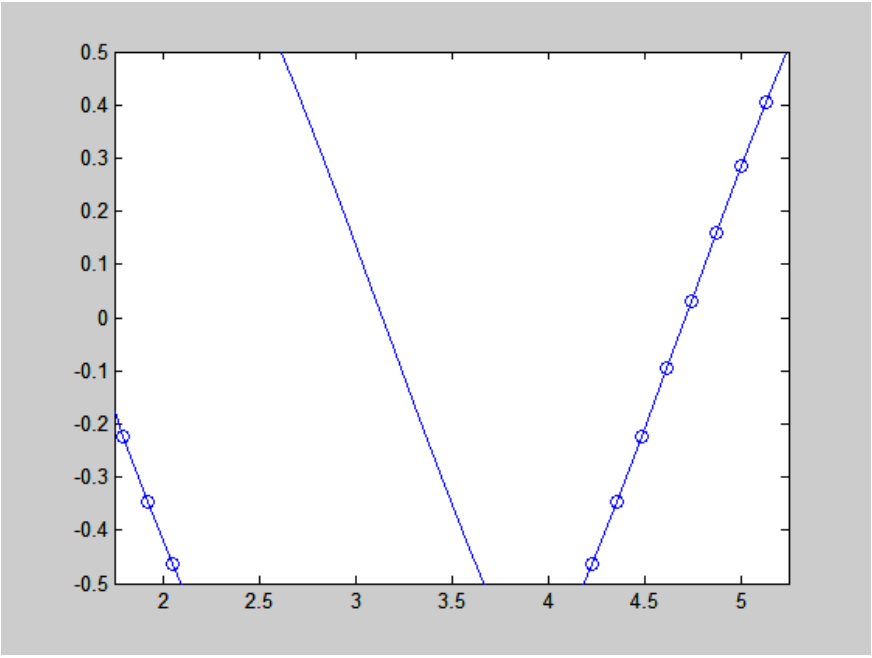
5.13 zoom Komutu

zoom komutu ile herhangibir grafiği büyütebiliriz. İlk olarak zoom modunun **on** yapılması gerekmektedir. Daha sonra, **zoom(n)** komutuyla grafiği büyütebiliriz. Burada **n** grafiğin kaç defa büyütüleceğini belirtir. **zoom out** komutu ise grafiği eski orijinal büyüklüğüne koymaktadır.

Örneğin, Şekil 5.27 de çizilen grafiği şu komutlarla 2 katı büyütebiliriz:

```
>> zoom on
>> zoom(2)
>> zoom off
```

Büyütülmüş olan yeni grafik Şekil 5.28 de gösterilmiştir.



Şekil 5.28 zoom(2) komutu ile 2 katı büyütülmüş grafik

5.14 plotyy Komutu

plotyy komutu ile normal olarak 2 grafik çizeriz ve birinci grafiğin y ekseni grafiğin solunda, ikinci grafiğin ise y ekseni grafiğin sağıda olur. Bu komutun genel şekli şöyledir:

```
plotyy(x1,y1, x2,y2)
```

xı-y1 grafiği sol dikey eksenini kullanır, x2-y2 grafiği ise sağ dikey ekseni kullanır.

Aşağıda bir örnek verilmiştir.

Örnek 5.22

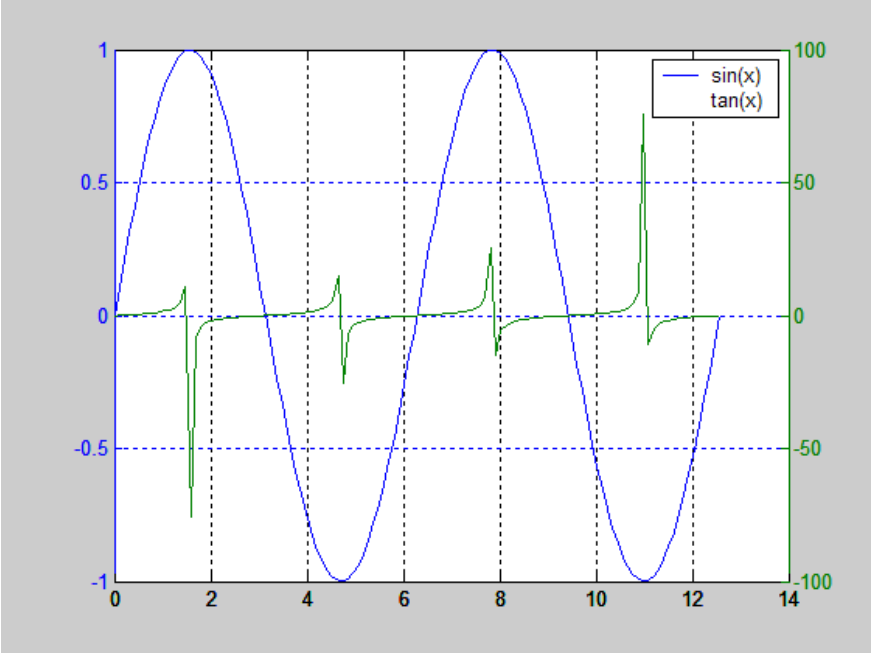
plotyy komutunu kullanarak sin(x) ve tan(x) in 0 dan 4π ya kadar grafiklerini çiziniz.

Çözüm 5.22

Bu çizim için gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> x =linspace(0,4*pi,120);
>> y=sin(x);
>> z=tan(x);
>> plotyy(x,y,x,z)
>> legend('sin(x)','tan(x)')
>> grid on
>>
```

Çizilmiş olan grafik ise Şekil 5.29 da gösterilmişir.



Şekil 5.29 plotyy komutu ile iki grafik çizimi

5.15 ginput Komutu

ginput komutunu kullanarak bir grafiğin herhangibir noktasının koordinatlarını bulabiliriz. Bu komutu yazınca önümüze otomatik olarak grafik çizilir ve faremiz yoluyla istediğimiz noktaya gideriz. Faremizin sol tuşunu tıklayınca bulunduğumuz noktanın koordinatlarını ekranımızda görürüz. Bu komutun genel şekli şöyledir:

```
[x,y] = ginput(n)
```

Burada **n** kaç tane noktanın koordinatını istediğimizi belirtir. Eğer **n** belirtilmemişse **Enter** tuşuna basıncaya kadar istediğimiz kadar noktanın koordinatlarını bulabiliriz.

Aşağıdaki örnekte grafik üzerindeki bir noktanın koordinatlarının nasıl bulunduğu gösterilmiştir:

```
 >> x =linspace(0,4*pi,100);
>> y=sin(x);
>> plot(x,y)
>> [x,y] = ginput(1)
```

[x,y] = ginput(1) komutunu verince karşımıza Şekil 5.30 da gösterilen grafik ve çıkar ve faremizi kullanarak koordinatlarını istediğimiz noktaya gideriz. Faremizin sol tuşunu tıklayınca bulunduğumuz noktanın koordinatlarını ekranda şu şekilde görürüz:

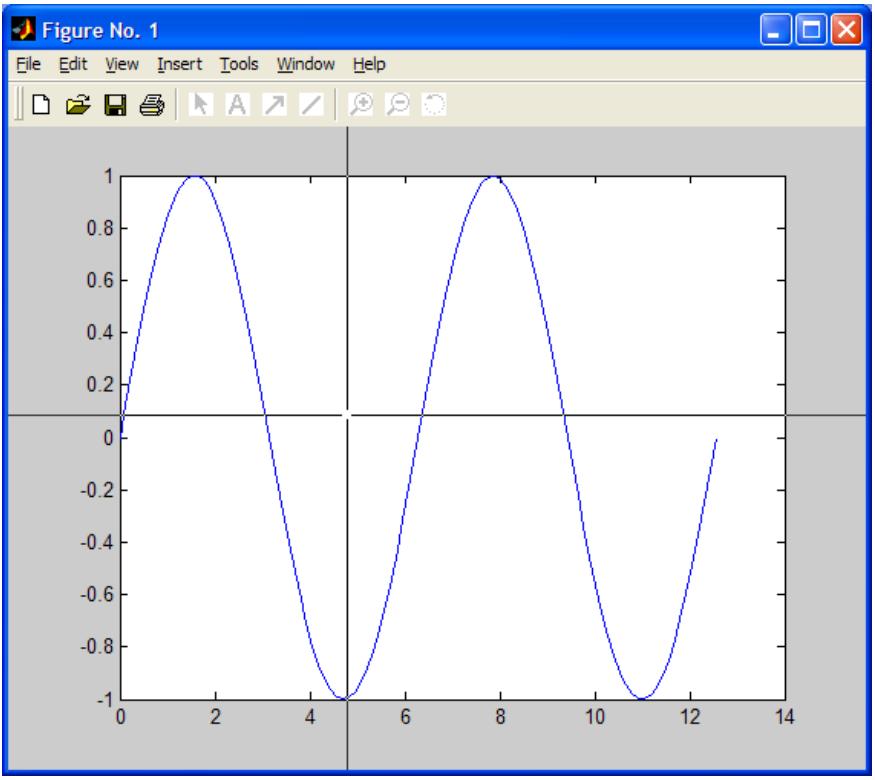
```
x =
   5.3065

y =

   0.1023
>>
```

Aşağıdaki örnekte ise 3 değişik noktanın koordinatları bulunmuştur:

```
[x,y] = ginput(3)

x =
    3.7581
    7.4677
   11.3387

y =
    0.6520
    0.1667
   -0.5292

>>
```

Burada, bulunmuş olan üç noktanın koordinatlar şunlardır:

(3.7581,0.6520) (7.4677,0.1667) (11.3387,-0.5292)



Şekil 5.30 **ginput** komutu ile istenilen bir noktanın koordinatlarını bulmak

5.16 Logaritmik Grafik Çizimi

Birçok mühendislik uygulamalarında logaritmik grafik çizimi oldukça önem taşımaktadır. MATLAB'I kullanarak çeşitli formatlarda logaritmik grafikler çizebiliriz:

- Her iki eksenin de logaritmik olduğu grafikler
- Dikey eksenin lineer, yatay eksenin logaritmik olduğu grafikler
- Dikey eksenin logaritmik, yatay eksenin lineer olduğu grafikler.

Şimdi bu değişik logaritmik grafik çizimlerine bazı örnekleri inceleyeceğiz.

Örnek 5.23

Aşağıdaki x ve y değerlerini her iki ekseni de logaritmik olan bir grafik üzerine çiziniz:

x = 2, 15, 50, 150, 280, 350, 520
y = 0.1, 0.3, 0.7, 1.2, 1.5, 1.8, 2.0

Çözüm 5.23

Gerekli olan MATLAB komutları aşağıda verilmiştir. Her iki ekseni de logaritmik olan bir grafik çizmek için **loglog** grafik komutunu kullanırız:

```
>> x = [2 15 50 150 280 350 520];
>> y = [0.1 0.3 0.7 1.2 1.5 1.8 2.0];
>> loglog(x,y)
>> xlabel('x')
>> ylabel('y')
>> title('Logaritmik Grafik')
>> grid on
```

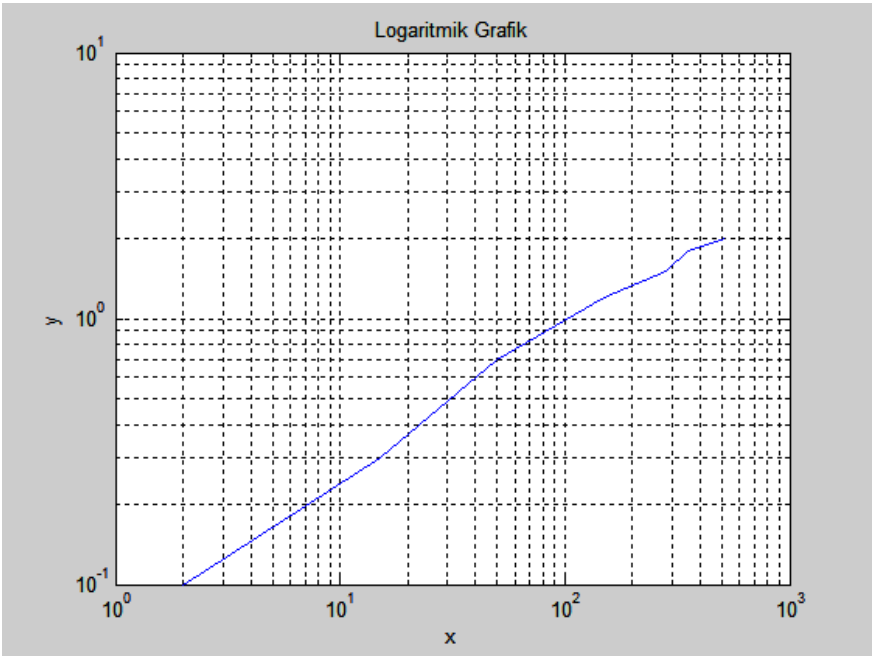
>>

Şekil 5.31 de çizilen grafik gösterilmiştir.

Örnek 5.24

Aşağıdaki x ve y değerlerini dikey ekseni logaritmik ve yatay ekseni lineer olan bir grafik üzerine çiziniz:

x = 2, 15, 50, 150, 280, 350, 520
y = 0.1, 0.3, 0.7, 1.2, 1.5, 1.8, 2.0



Şekil 5.31 Her iki ekseni de logaritmik olan grafik çizimi

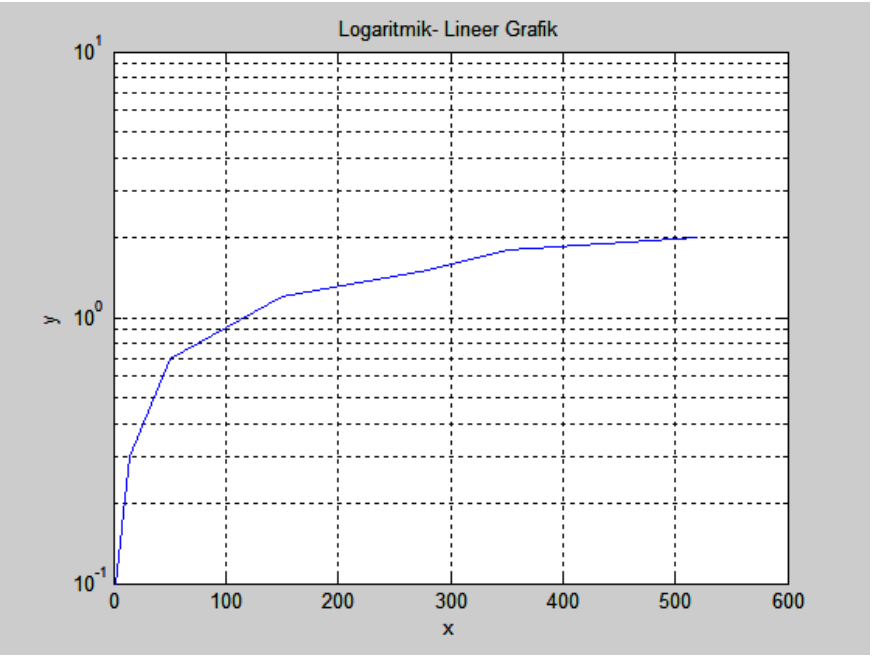
Çözüm 5.24

Gerekli olan MATLAB komutları aşağıda verilmiştir. Dikey ekseni logaritmik ve yatay ekseni lineer olan bir grafik çizmek için **semilogy** grafik komutunu kullanırız:

>> x = [2 15 50 150 280 350 520];

```
>> y = [0.1 0.3 0.7 1.2 1.5 1.8 2.0];
>> semilogy(x,y)
>> xlabel('x')
>> ylabel('y')
>> title('Logaritmik- Lineer Grafik')
>> grid on
>>
```

Çizilmiş olan grafik ise Şekil 5.32 de gösterilmiştir.



Şekil 5.32 Dikey ekseni logaritmik, yatak ekseni lineer olan grafik

Örnek 5.25

Aşağıdaki x ve y değerlerini dikey ekseni lineer ve yatay ekseni logaritmik olan bir grafik üzerine çiziniz:

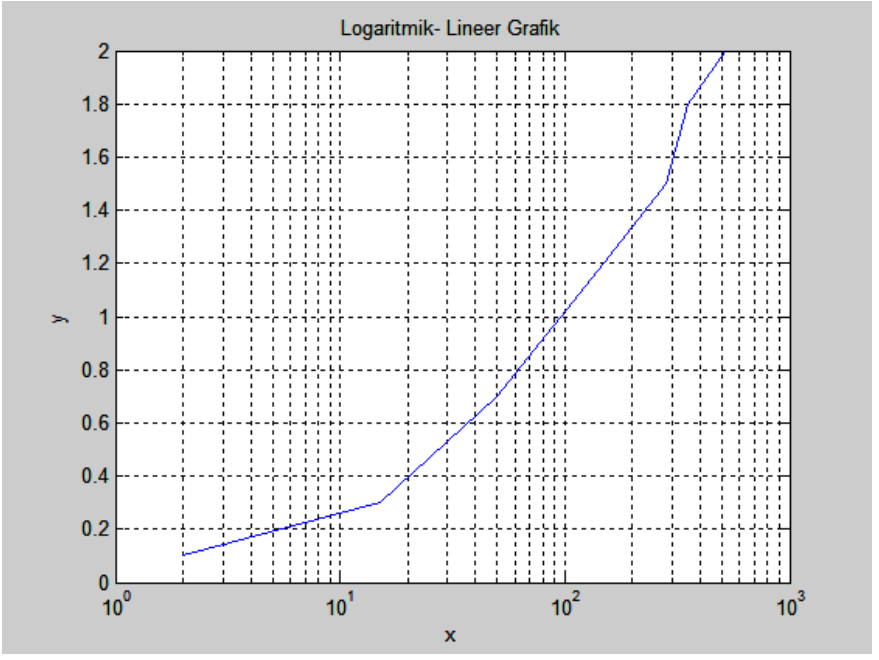
x = 2, 15, 50, 150, 280, 350, 520
y = 0.1, 0.3, 0.7, 1.2, 1.5, 1.8, 2.0

Çözüm 5.25

Gerekli olan MATLAB komutları aşağıda verilmiştir. Dikey ekseni logaritmik ve yatay ekseni lineer olan bir grafik çizmek için **semilogx** grafik komutunu kullanırız:

```
>> x = [2 15 50 150 280 350 520];
>> y = [0.1 0.3 0.7 1.2 1.5 1.8 2.0];
>> semilogx(x,y)
>> xlabel('x')
>> ylabel('y')
>> title('Logaritmik- Lineer Grafik')
>> grid on
>>
```

Çizilmiş olan grafik ise Şekil 5.33 de gösterilmiştir.



Şekil 5.33 yatay ekseni logaritmik, dikey ekseni lineer olan grafik

Örnek 5.26

Aşağıdaki fonksiyonun lineer, ve üç değişik şekilde logaritmik olmak üzere dört değişik eksende grafiğini çiziniz. x in 0 ve 10 arasında değiştiğini kabul ediniz:

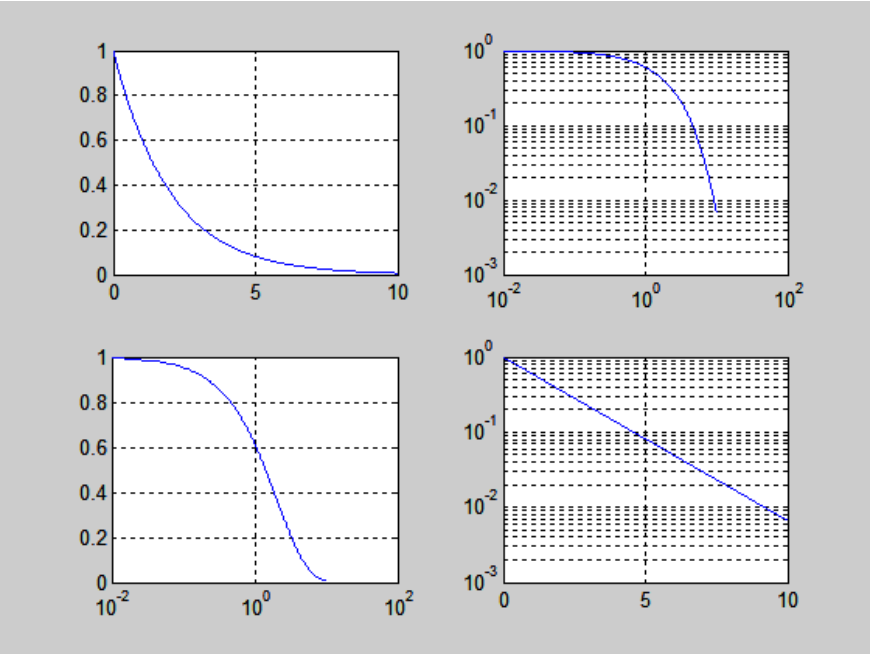
$$y = e^{-0.5x}$$

Çözüm 5.26

Fonksiyonun lineer-lineer, logaritmik-logaritmik, logaritmik-lineer,ve lineer-logaritmik olmak üzere dört grafiğini çizmek için şu MATLAB komutları kullanılmıştır:

```
>> x=0:0.01:10;
>> y=exp(-0.5*x);
>> subplot(2,2,1);
>>%
>> % lineer-lineer grafik
>>%
>> plot(x,y);
>> grid on
>> subplot(2,2,2);
>>%
>>% logaritmik – logaritmik grafik
>>%
>> loglog(x,y);
>> grid on
>> subplot(2,2,3);
>>%
>>% logaritmik – lineer grafik
>>%
>> semilogx(x,y);
>> grid on
>> subplot(2,2,4);
>>%
>>% lineer – logaritmik grafik
>>%
>> semilogy(x,y);
>> grid on
>
```

Çizilmiş olan grafikler Şekil 5.34 de gösterilmiştir.



Şekil 5.34 Bir fonksiyonun dört değişik grafiğinin Çizimi

5.17 Pay (Pie) Grafiği Çizimi

Pay grafiği bir daire şeklinde olup bu daire değişkenlerin büyüklüklerine bağlı olarak parçalara ayrılmıştır. Bu tip grafikler daha çok birkaç tane değişken olan durumlarda kullanılırlar. Bazı örnekler aşağıda verilmiştir.

Örnek 5.27

100 kişiye hafta sonlarını nasıl geçirdikleri sorusu sorulduğunda şu cevaplar alınmıştır:

	Yapılan aktivite	Yapanların sayısı
1.	Televizyon görme	55
2.	Futbol oynama	25

3.	Kitap okuma	10
4.	Araba sürme	5
5.	Yürümek	5

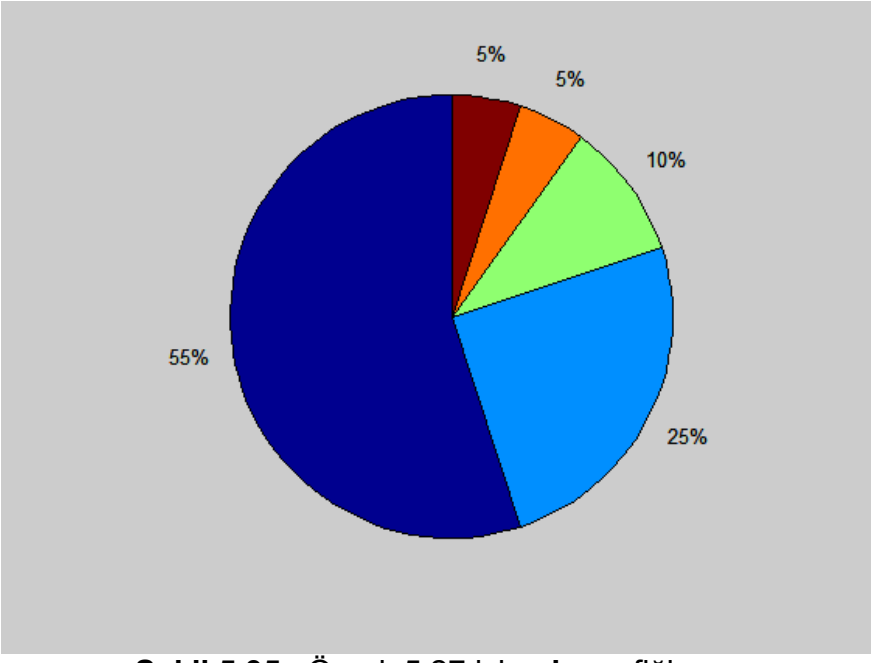
Hafta sonları aktivitilerini gösteren bir pay grafiği çiziniz.

Çözüm 5.27

MATLAB **pie** komutu verilen değerlere uygun bir pie grafiği çizer. Gerekli komutlar aşağıda verilmiştir:

```
>> x = [55 25 10 5 5];
>> pie(x)
>
```

Çizilen grafik Şekil 5.35 de gösterilmiştir.



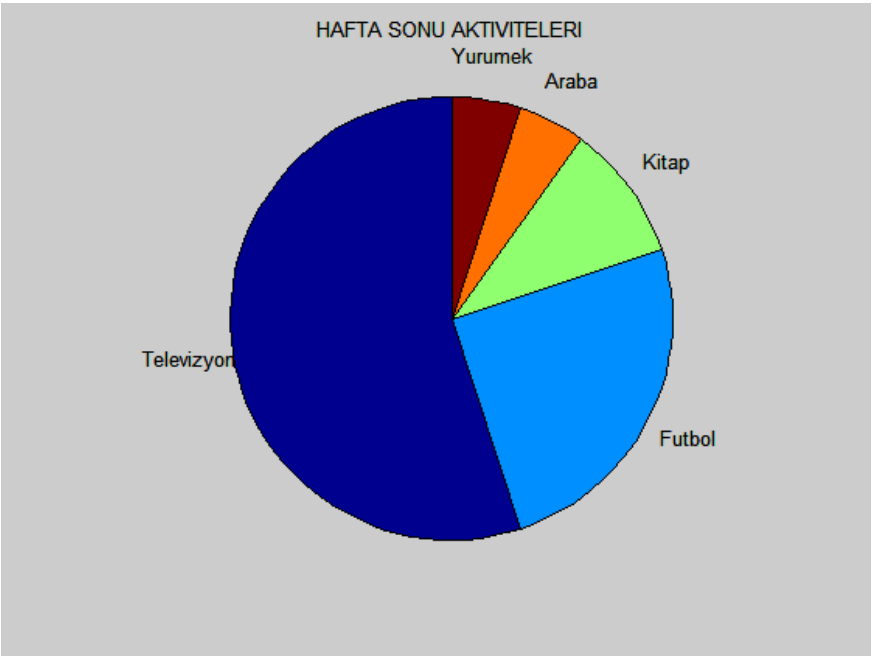
Şekil 5.35 Örnek 5.27 için **pie** grafiği

Yukarıda çizilen grafikte sadece yüzdelik sayılar olup bu sayıları neye karşılık geldikleri belli değildir. Bunu belirtmek için her bölümün tanımını bir değişkende tutabiliriz.

Aşağıdaki. Örnekte her bölümün tanımı **tanımlar** diye bir değişkende saklanmış ve bu değişken **pie** komutuna verilmiştir:

```
>> x = [55 25 10 5 5];
>> tanimlar = {'Televizyon','Futbol','Kitap','Araba','Yurumek'};
>> pie(x,tanimlar)
>> title('HAFTA SONU AKTIVITELERI')
>>
```

Çizilmiş olan yeni pie grafiği Şekil 5.36 da gösterilmiştir.



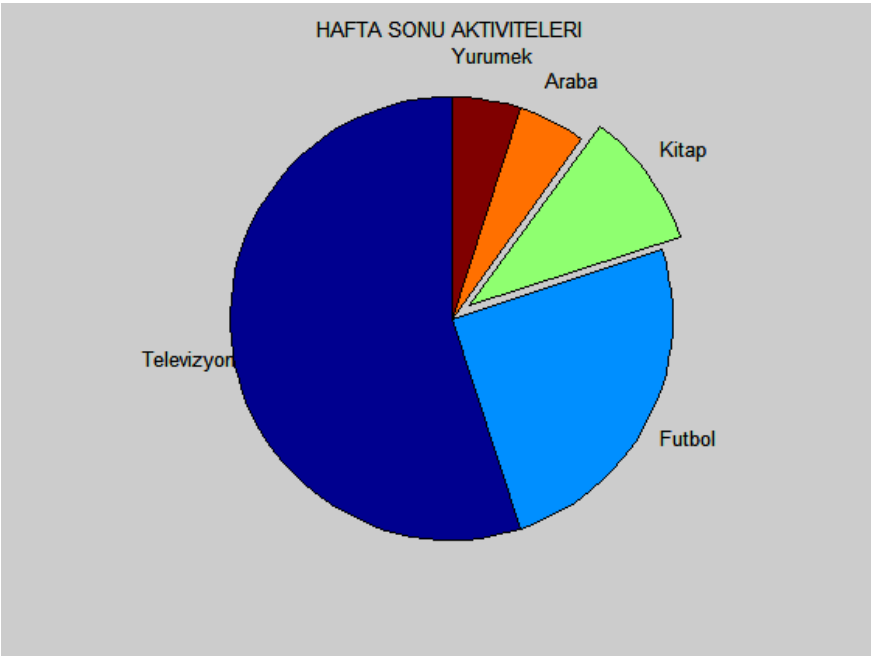
Şekil 5.36 Bömlümleri tanımlanmış pie grafiği

Bir pie grafiğinde belirli bir dilimü belirtmek için onu hafif dışarıya çıkmış şekilde çizebiliriz. Bunun için, bir vektör yaratırız ve belirtmek istediğimiz dilimleri 1, diğer dilimlere ise 0 sayılarını koruz. Aşağıdaki örnekte, **Kitap** dilimi (3 üncü dilim) hafif dışarıya çizilmiştir:

```
>> x = [55 25 10 5 5];
>> tanimlar = {'Televizyon','Futbol','Kitap','Araba','Yurumek'};
```

```
>> dilimler = [0 0 1 0 0];
>> pie(x,dilimler,tanimlar)
>> title('HAFTA SONU AKTIVITELERI')
>>
```

Çizilmiş olan yeni grafik Şekil 5.37 de gösterilmiştir.

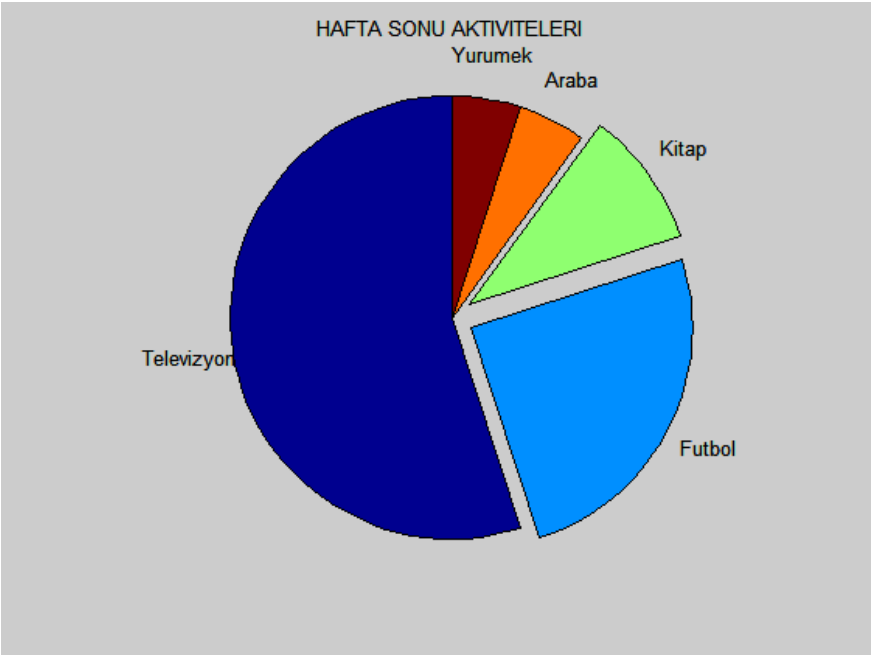


Şekil 5.37 Kitap dilimi hafif dışarıya alınmıştır

Birden fazla dilimi dışarıya almak için o dilimler için dilimler vektörüne 1 koymamız gerekir. Aşağıdaki örnekte 2.nci ve 3.üncü dilimler hafif dışarıya alınmıştırlar:

```
>> x = [55 25 10 5 5];
>> tanimlar = {'Televizyon','Futbol','Kitap','Araba','Yurumek'};
>> dilimler = [0 1 1 0 0];
>> pie(x,dilimler,tanimlar)
>> title('HAFTA SONU AKTIVITELERI')
>>
```

Şekil 5.38 de yeni grafik gösterilmiştir.



Şekil 5.38 Birden fazla diliim dışarıya alınmıştır

Örnek 5.28

Bir grup insana, en çok yedikleri meyveler hakkında sorulan sorular sonunda şu sonuçlar çıkmıştır:

Meyve Adı	Yenen miktar
Elma	4500
Armut	4000
Banana	3200
Kiraz	2000
Karpuz	800
Kavun	500

Bu miktarları bir pie grafiği şeklinde gösteriniz ve grafiğin dilimlerine meyvelerin isimlerini yazınız.

Çözüm 5.28

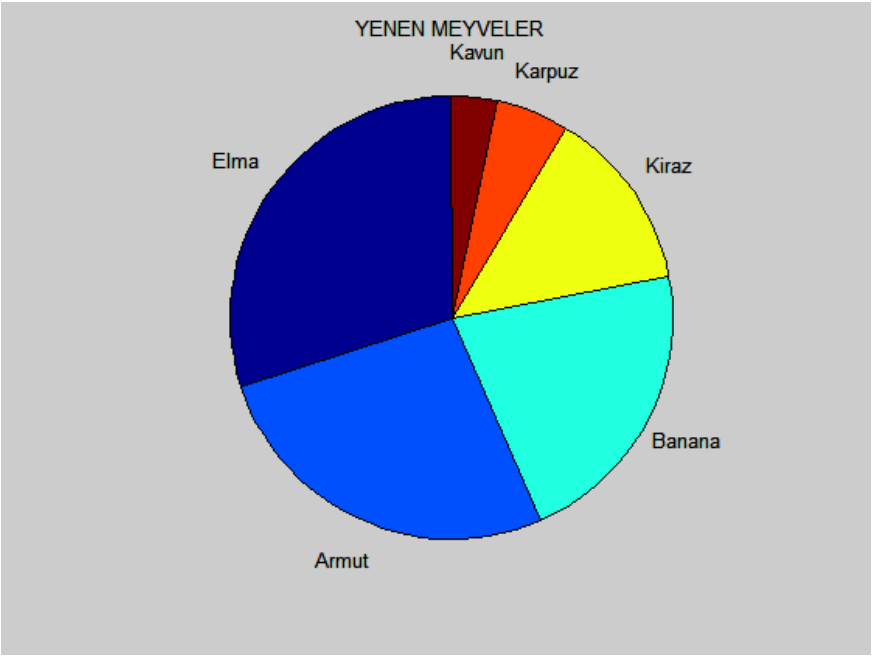
Bu örnekte, **pie** grafiği çizebilmek için ilk olarak her yene miktarın toplam yenen miktara olan oranlarını bulmamız gerekir. Daha sonra ise grafiği çizebiliriz.

Aşağıdaki örnekte **yenen** vektörü yenen meyve miktarlarını tutar, **oran** vektörü her meyvenin yeme oranını tutar, ve **tanımlar** vektörü ise meyvelerin isimlerini tutar.

Gerekli olan komutlar aşağıda verilmiştir:

```
>> yenen = [4500 4000 3200 2000 800 500];
>> oran = yenen / sum(yenen);
>> tanimlar = {'Elma','Armut','Banana','Kiraz','Karpuz','Kavun'};
>> pie(oran,tanimlar)
>> title('YENEN MEYVELER')
>>
```

Çizilmiş olan grafik Şekil 5.39 da gösterilmiştir.



Şekil 5.39 Meyve yeme oranını gösteren **pie** grafiği

5.18 Bar (Sütun) Grafiği

Sütun grafiğinde değişkenler dikey sütunlar halinde gösterilir. Bu grafiği çizmek için bar komutu kullanılır. Aşağıda bir örnek verilmiştir.

Örnek 5.29

Örnek 5.28 de verilen meyve yeme tablosunu kullanarak bir sütun grafiği çiziniz.

Çözüm 5.29

Çizim için gerekli olan komutlar aşağıda verilmiştir:

```
>> yenen = [4500 4000 3200 2000 800 500];
>> bar(yenen)
>> title('YENEN MEYVE MIKTARI')
>>
```

Çizilmiş olan sütun grafiği Şekil 5.40 da gösterilmiştir.

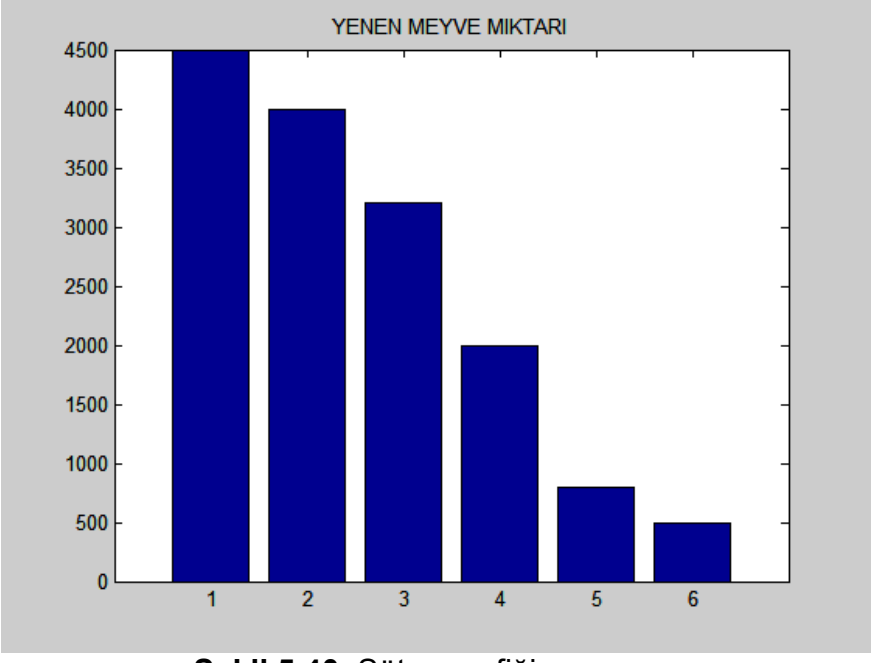
barh komutunu kullanarak grafiği yatay çizmemiz de mümkündür. Yukarıdaki programda **bar(yenen)** komutunu **barh(yenen)** komutuyla değiştirirsek Şekil 5.41 de gösterilen grafiği elde etmiş oluruz.

5.19 Stairs (Adım) Grafiği

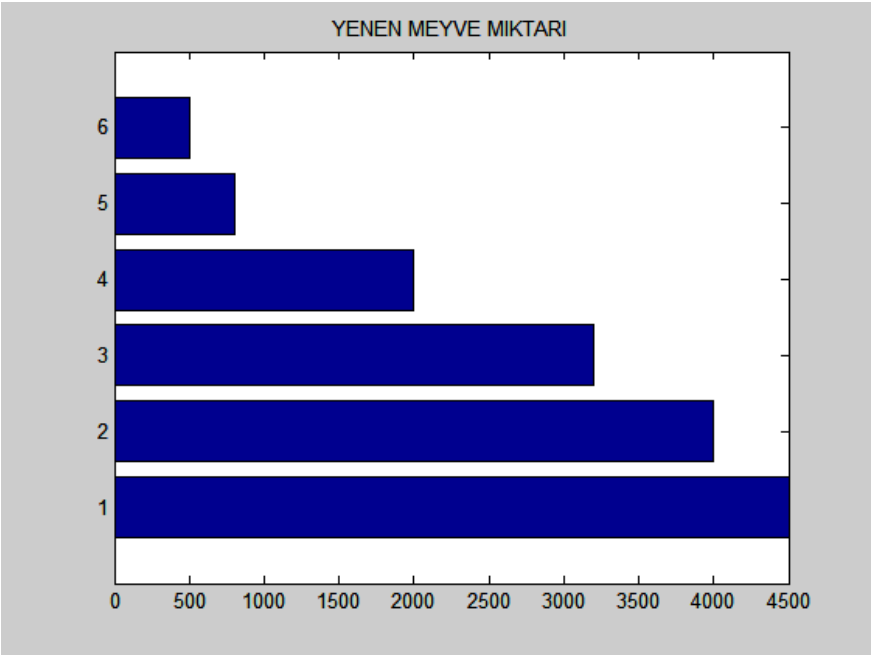
Bu tip grafikler sayısal değerlerin zaman içerisinde değişimini göstermek için kullanılırlar. Aşağıda bir örnek verilmiştir:

Örnek 5.30

x değerleri 0 dan 20 ye kadar 0.25 aralıklarıyle değiştikçe sin(x) grafiğini adım grafik olarak çiziniz.



Şekil 5.40 Sütun grafiği



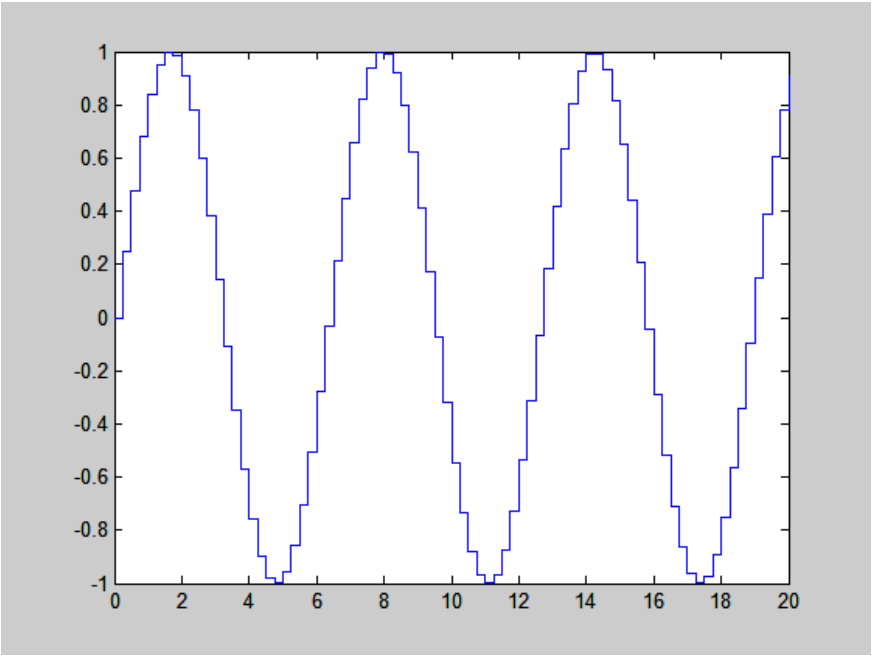
Şekil 5.41 yatay sütun grafiği

Çözüm 5.30

Gerekli olan komutlar aşağıda verilmiştir:

```
>> x = 0:0.25:20;
>> stairs(x,sin(x))
>>
```

Çizilen grafik Şekil 5.42 de gösterilmiştir.



Şekil 5.42 sin(x) fonksiyonunun adım (stairs) grafiği

5.20 Compass (Pusula) Grafiği

compass grafiği genellikle yön ve miktar içeren değişkenler için kullanılır. Bu grafik, bir pusula şekli çizer ve değişkenleri bu pusulada gösterir. Değişkenlerin miktarları oklarla gösterilir, açıları ise pusula açı sistemine göre çizilir. Aşağıda bir örnek verilmiştir.

Örnek 5.31

Aşağıdaki kompleks sayıları bir compass grafiğinde gösteriniz:

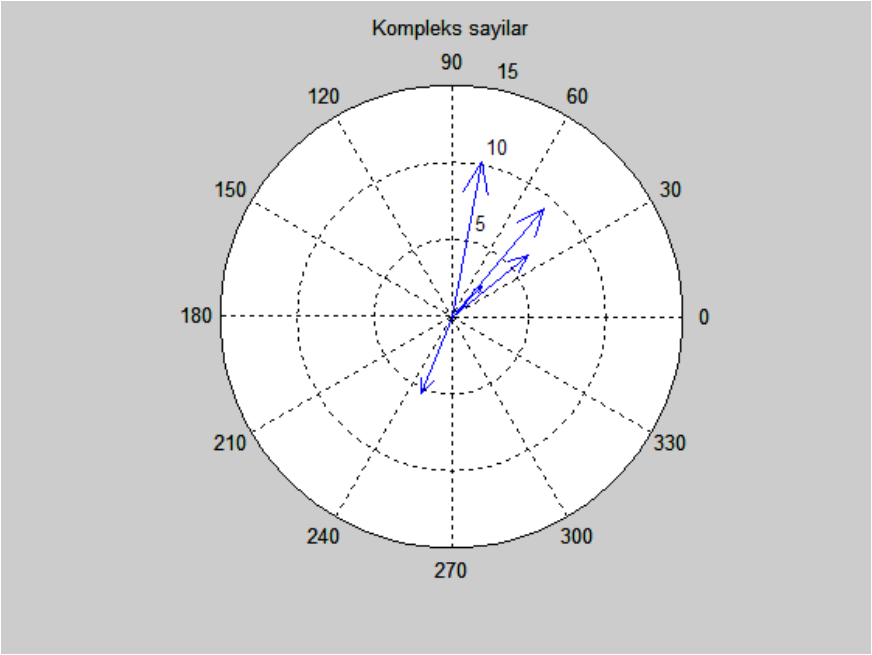
2+2i 5+4i 6+7i 2+10i -2-5i

Çözüm 5.31

Gerekli olan komutlar aşağıda gösterilmiştir:

```
>> x = [2+2i 5+4i 6+7i 2+10i -2-5i];
>> compass(x)
>> title('Kompleks sayilar')
>>
```

Çizilen grafik ise Şekil 5.43 de gösterilmiştir.



Şekil 5.43 compass (pusula) grafiği

5.21s tem Grafiği

stem grafiği x ekseninden uzanan çizgiler şeklinde grafik çizer. Değişkenlerin değerleri çizgilerin sonunda birer küçük dairelerle belirtilir.

Örnek 5.32

Sinüs dalgasının 0 ve 2π arasında değişimini bir stem

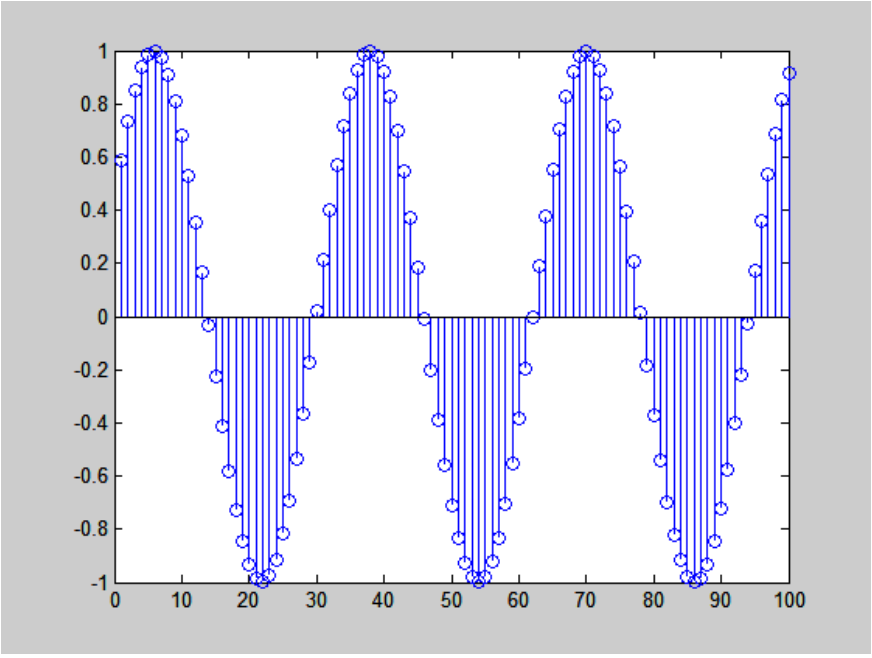
grafikle gösteriniz.

Çözüm 5.32

Gerekli olan komutlar aşağıda verilmiştir:

```
>> x = linspace(0.2*pi,20);
>> stem(sin(x))
>>
```

Çizilen grafik ise Şekil 5.44 de gösterilmiştir.



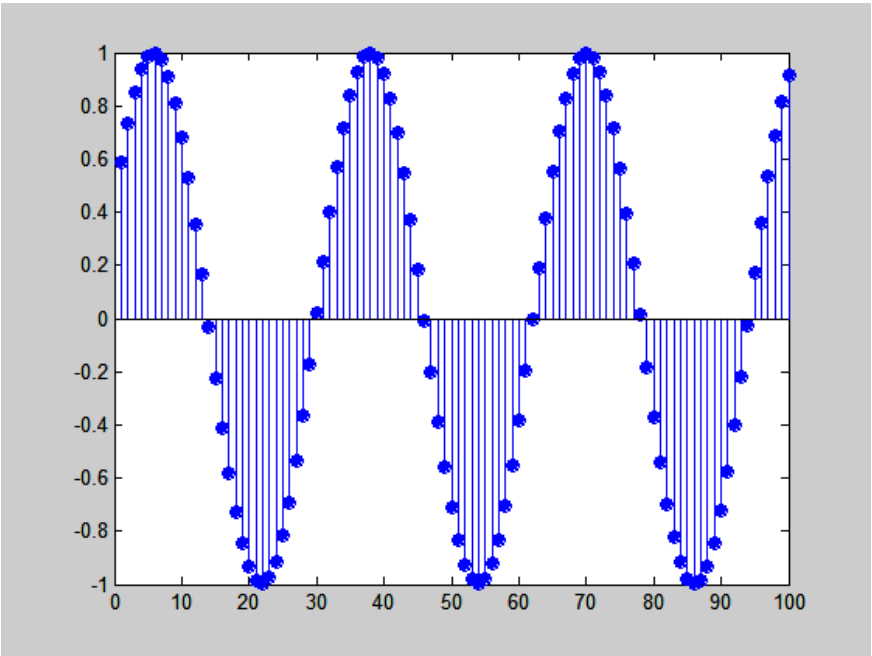
Şekil 5.44 stem grafiği ile sin(x) fonksiyonunun çizimi

Yukarıdaki şekilde y değerlerini belirten dairelerin içi boştur. **stem** komutunda **fill** komutunu kullanarak bu dairelerin içini doldurabiliriz. Bunun için, aşağıdaki komutları vermemiz gerekecektir:

```
>> x = linspace(0.2*pi,20);
>> stem(sin(x),'fill')
```

>>

Çizilmiş olan yeni grafik Şekil 5.45 de gösterilmiştir.

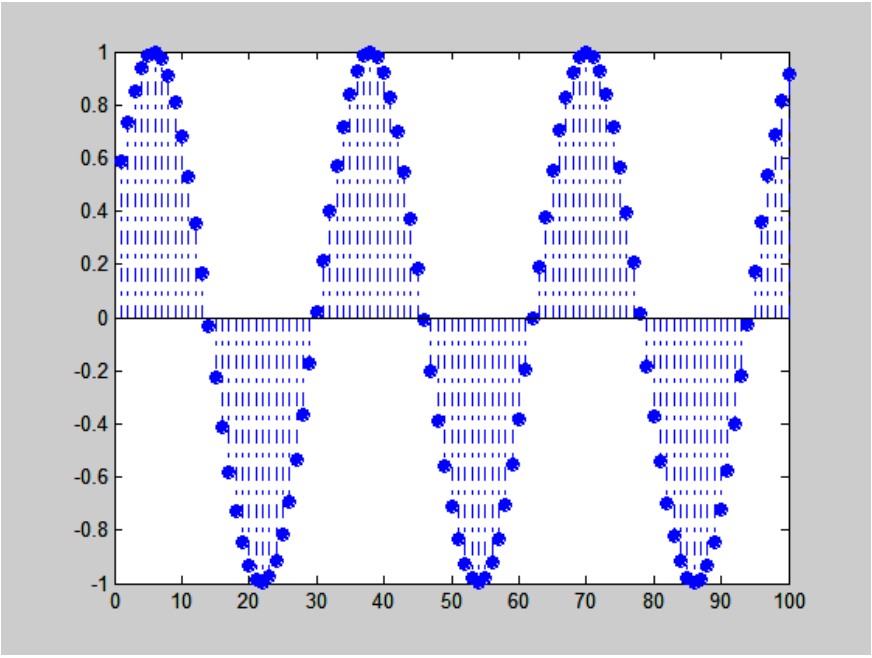


Şekil 5.45 stem grafiğinde içi doldurulmuş daireler

Yukarıda çizmiş olduğumz stem grafiklerindeki çizgileri noktalı çizgi yapmak için stem komutu içerisine '-.' karakterlerini koymamız gerekir:

```
 >> x = linspace(0.2*pi,20);
>> stem(sin(x),'fill','-.')
>>
```

Çizilmiş olan yeni grafik Şekil 5.46 da gösterilmiştir.



Şekil 5.46 noktalı çizgilerden meydana gelmiş stem grafiği

5.22 hist (Histogram) Grafiği

hist komutu veri değişimlerini gösteren bir histogram çizer. Aşağıda bir örnek verilmiştir.

Örnek 5.33

Aşağıdaki veri değerlerine göre bir histogram çiziniz:

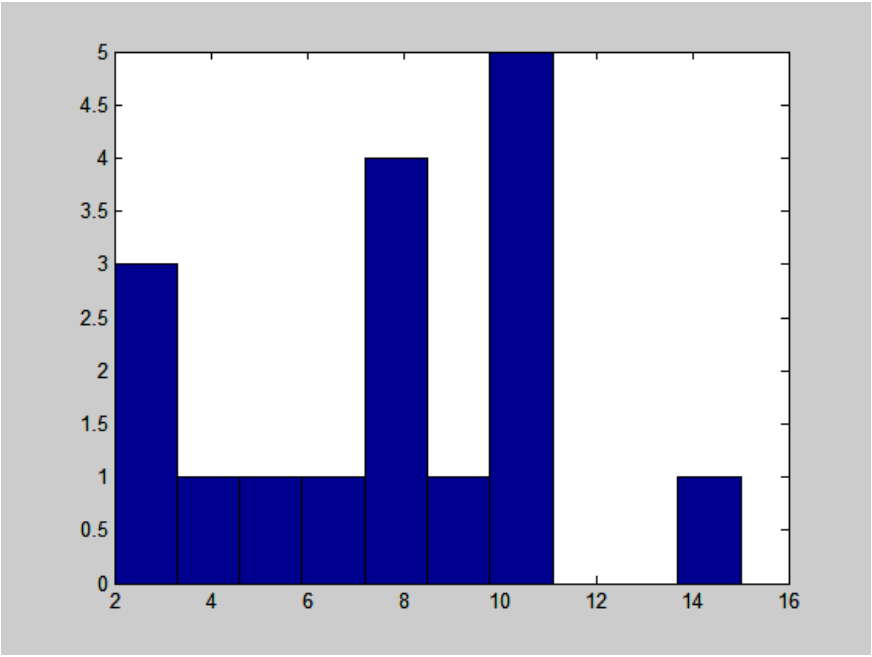
2 2 2 4 5 6 8 8 8 8 9 10 10 10 10 10 15

Çözüm 5.33

Gerekli olan komutlar aşağıda verilmiştir:

```
>> x = [2 2 2 4 5 6 8 8 8 8 9 10 10 10 10 10 15];
>> hist(x)
>>
```

Çizilen grafik Şekil 5.47 de gösterilmiştir.



Şekil 5.47 Histogram grafiği

5.23 Polar Grafik

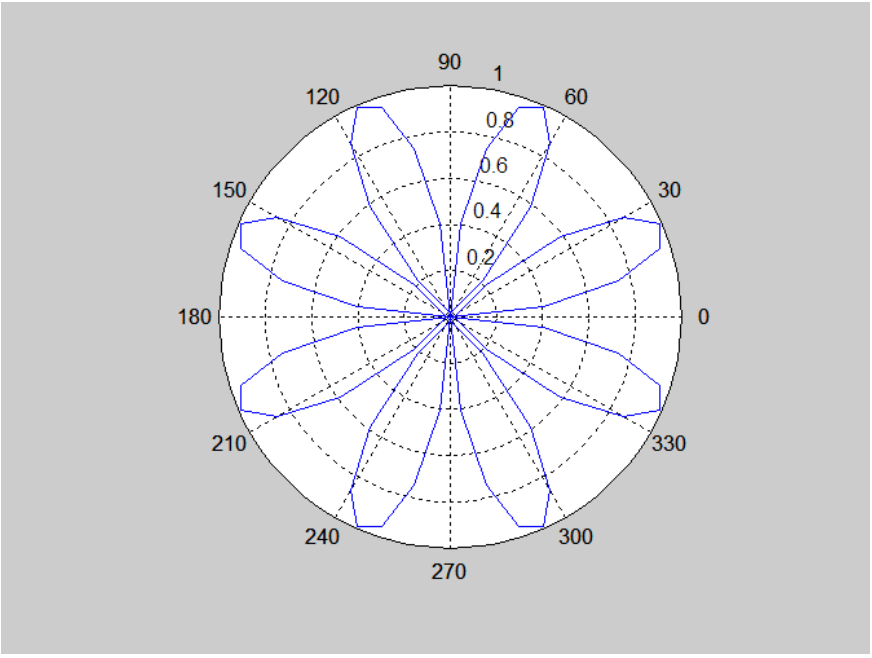
(x, y) gibi bir nokta polar koordinatlarda şu şekilde gösterilir:

$$x = r\cos(ø)$$
$$y = r\sin(ø)$$

Burada ø açısı 0 dan 360º kadar değişmektedir. MATLAB *polar(ø, r)* komutu şiddeti r ve açısı ø olan bir polar grafik çizer. Örneğin,

```
>> x=0:pi/30:2*pi;
>> polar(x,sin(4*x)),grid on
```

komutları Şekil 5.48 de gösterilen polar grafiği çizer.



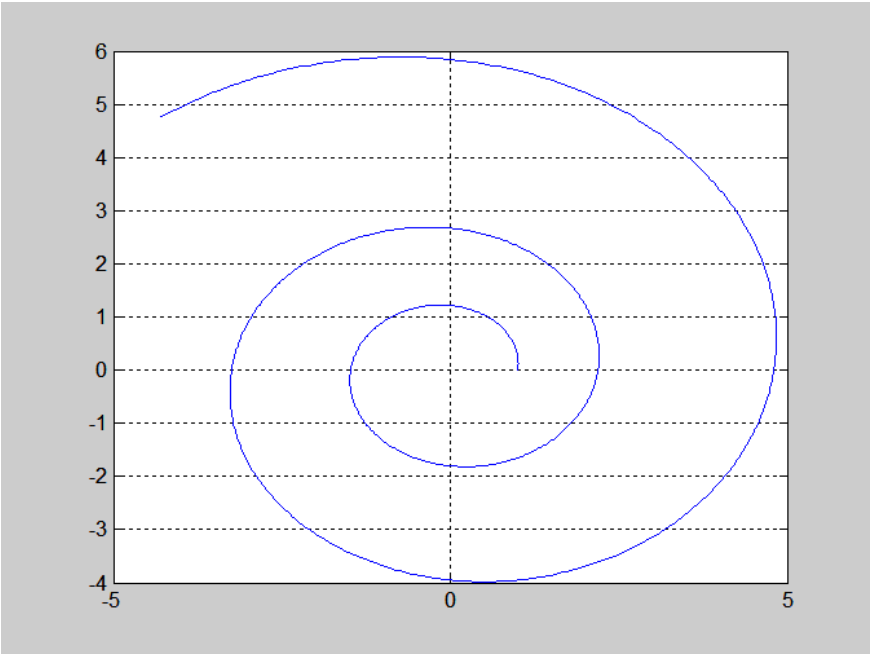
Şekil 5.48 Polar grafik

5.24 Kompleks Sayı Grafiği

Eğer y değişkeni kompleks sayı ise, plot(y) komutu bu değişkenin gerçek olmayan bölümünü dikey eksene, gerçek olan bölümünü ise yatay eksene çizer. Örneğin,

```
>> z = 0.1 + 1.2i;
>> n = [0:0.01:10];
>> plot(z.^n), grid on
```

komutları Şekil 5.49 da gösterilen grafiği çizer.



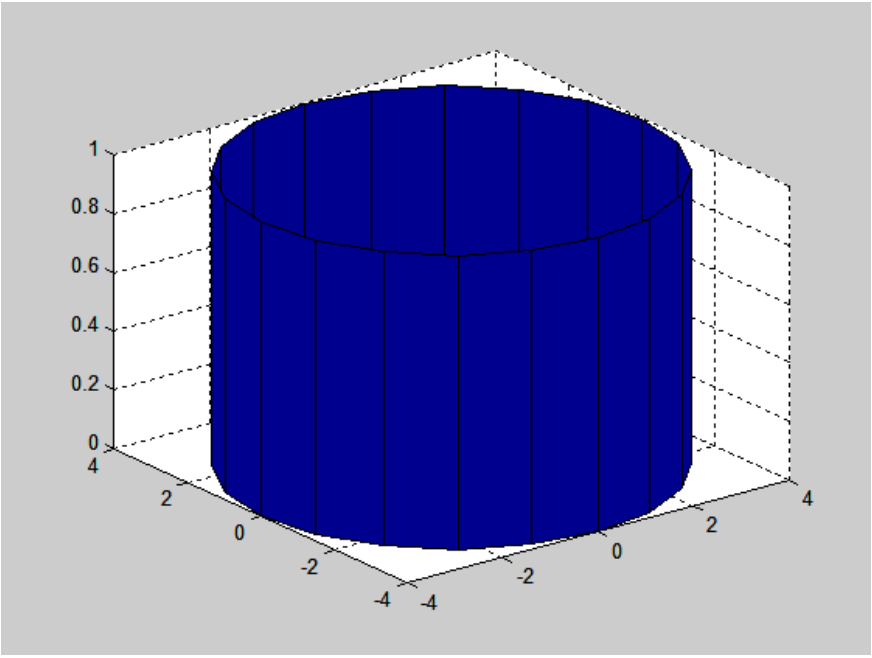
Şekil 5.49 Kompleks sayı grafiği

5.25 3 Boyutlu Silindir Grafiği

cylinder(r) komutu yarıçapı r olan ve birim yüksekliğinde olan bir silindir grafiği çizer.

Örneğin,

cylinder(4) komutu, Şekil 5.50 de gösterilen ve yarıçapı 4 olan, bir silindirin grafiği çizer.



Şekil 5.50 cylinder komutu ile silindir çizimi

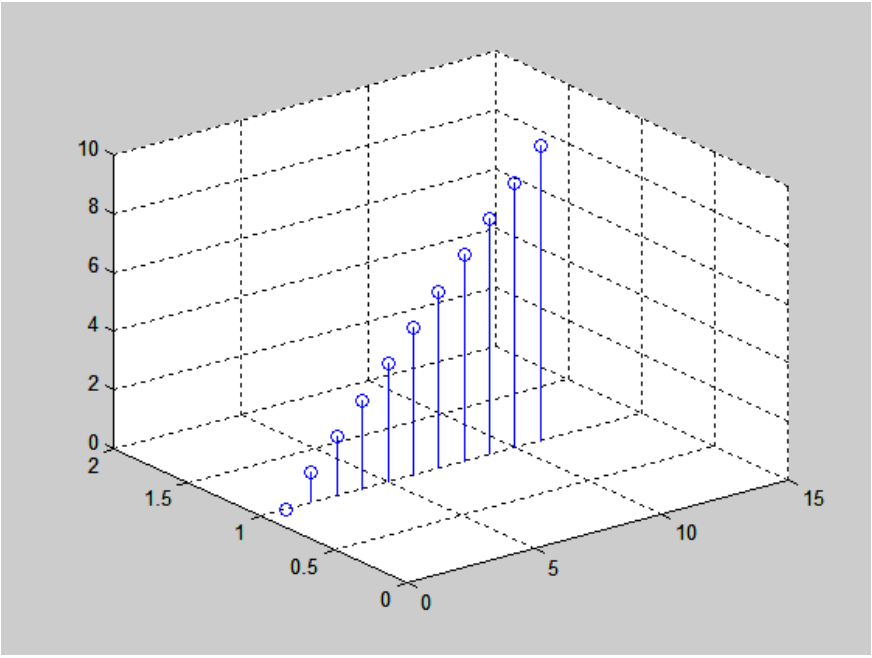
5.26 3 Boyutlu stem Grafiği

stem3 komutu 3 boyutlu stem grafiği çizer.

Örneğin,

```
>> x=0:10;
>> stem3(x)
>>
```

komutları Şekil 5.51 da gösterilen 3 boyutlu stem grafiğini çizer.



Şekil 5.51 3 boyutlu **stem** grafiği

Örnek 5.34

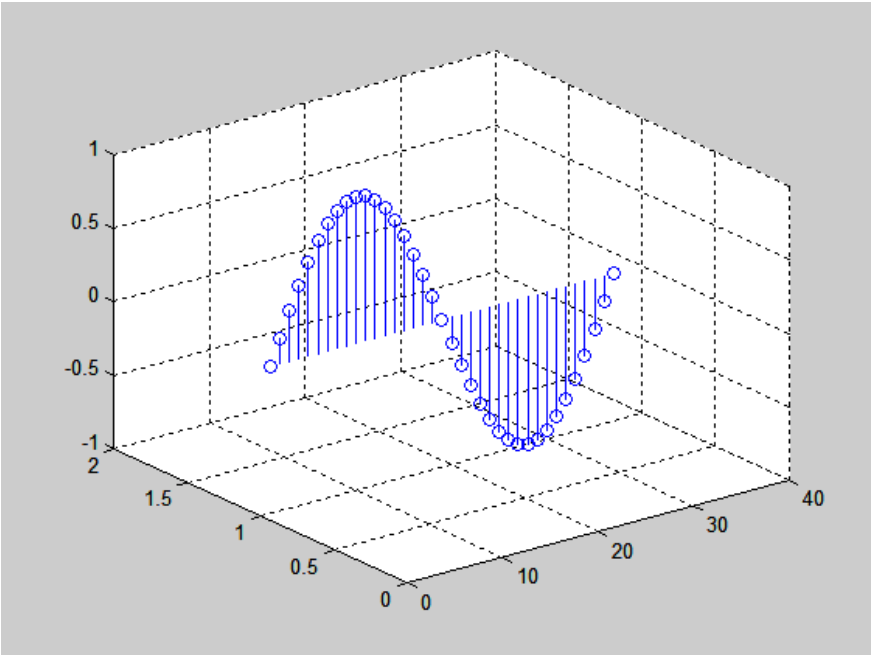
sin(x) grafiğinin 0 ve 360 derece arasında değişimini 3 boyutlu **stem** grafiği ile gösteriniz.

Çözüm 5.34

İstenilen komutlar aşağıda verilmiştir:

```
>> x=0:10:360;
>> y=sin(2*x*pi/180);
>> y=sin(x*pi/180);
>> stem3(y)
>>
```

Çizilmiş olan 3 boyutlu **stem** grafiği Şekil 5.52 de gösterilmiştir.



Şekil 5.52 sin(x) grafiğinin 3 boyutlu stem grafiği olarak çizimi

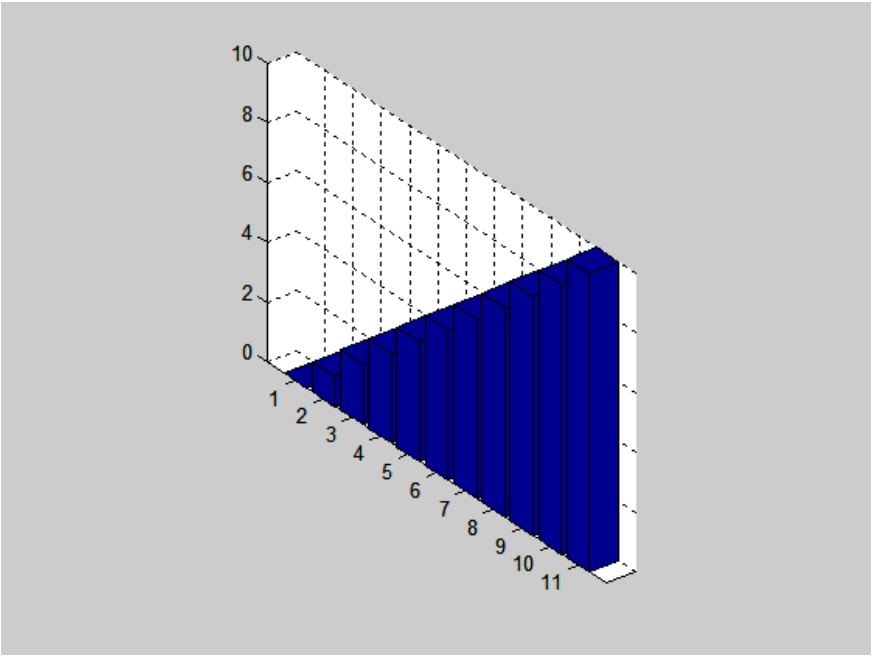
5.27 3 Boyutlu Bar Grafiği

bar3 komutu 3 boyutlu bar (sütun) grafiği çizer.

Örneğin,

```
>> x=0:10;
>> bar3(x)
>
```

komutları Şekil 5.53 de gösterilen 3 boyutlu sütun grafiği çizer.



Şekil 5.53 3 boyutlu bar (sütun) grafiği

5.28 3 Boyutlu plot Komutu

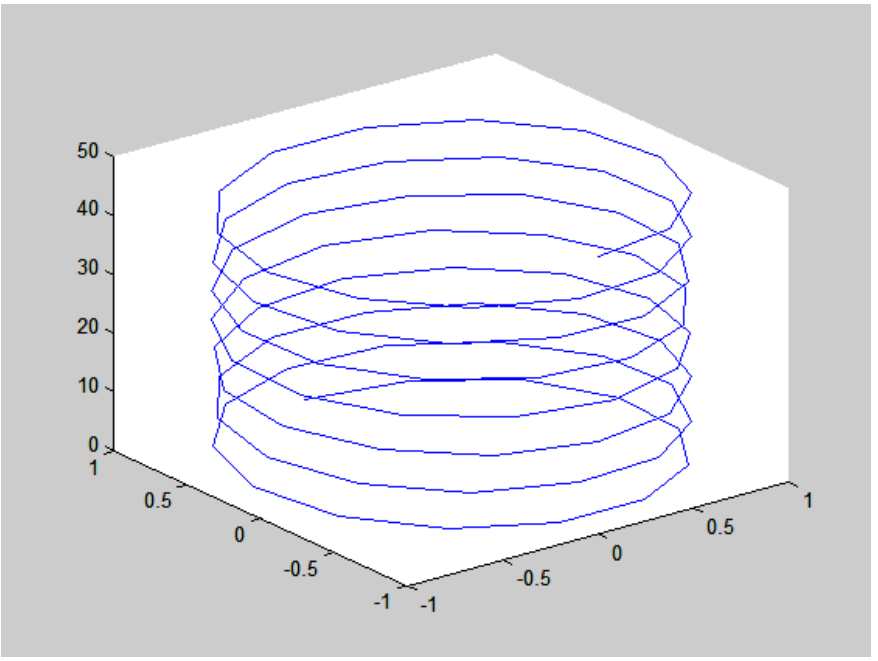
MATLAB'da **plot3** komutunu kullanarak 3 boyutlu grafikler çizebiliriz. Bu komutun genel şekli şöyledir:

```
plot3(x,y,z)
```

Örneğin, aşağıdaki 3 boyutlu **plot3** komutu ile bir helix grafiği çizebiliriz:

```
>> z = linspace(0,15*pi);
>> plot3(sin(z),cos(z),z)
>>
```

Çizilmiş olan 3 boyutlu grafik Şekil 5.54 de gösterilmiştir.



Şekil 5.54 3 boyutlu plot komutu ile çizilmiş grafik

Örnek 5.34

Aşağıdaki fonksiyonların 3 boyutlu grafiğini çiziniz:

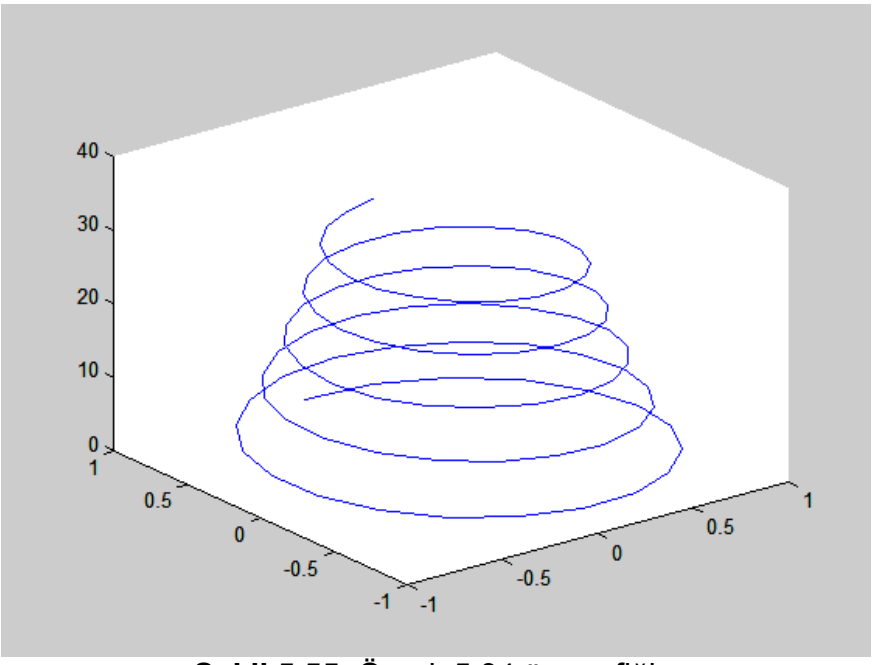
$x = e^{-0.02t} \sin(t)$
$y = e^{-0.02t} \cos(t)$
$z = t$

Çözüm 5.34

Gerekli olan komutlar aşağıda verilmiştir:

```
>> t = linspace(0,10*pi,100);
>> plot3(exp(-0.02*t).*sin(t),exp(-0.02*t).*cos(t),t)
>>
```

Çizilmiş olan grafik Şekil 5.55 de verilmiştir.



Şekil 5.55 Örnek 5.34 ün grafiği

5.29 Sorular

1. Aşağıdaki fonksiyonların grafiklerini çiziniz

   (a) sinx – cosx
   (b) sin2x
   (c) x^2 –x + 1
   (d) $e^{-0.1x}\sin x$

2. Aşağıda verilen x-y değerlerine göre bir grafik çiziniz:

   x = [0 1 2 3 4 5 6 7 8 9 10]
   y = [0 3 5 7 9 11 13 15 17 19 21]

3. Soru 2 de çizilen grafikte veri noktalarını ‘o’ işaretleri ile gösteriniz.

4. $2e^{-0.1x}$ fonksiyonunun grafiğini –2 ve 2 arasında

(a) lineer- lineer
(b) logaritmik-logaritmik
(c) lineer-logaritmik
(d) logaritmik-lineer

eksenlere ve ayni grafik ortamında çiziniz.

5. xtanx = 7 denkleminin grafiğini çiziniz ve denklemin köklerini bulunuz.

6
SAAT VE TARİH

MATLAB ile çeşitli saat ve tarihle ilgili fonksiyonlar kullanabiliriz. Bu fonksiyonların bazıları sadece saat ile, bazıları ise adece tarih ile ilgilidirler.

Bu bölümde sıkça kullanılan MATLAB saat ve tarihle ilgili fonksiyonlara bir göz atacağız ve her fonksiyon için örnekler vereceğiz:

6.1 calendar Fonksiyonu

Bu fonksiyon, bulunduğumuz ayın tarihini 6x7 bir dizi olarak gösterir. Dizinin ilk sütunu Pazarı, son sütunu ise Cumartesini gösterir.

Örneğin, bulunduğumuz ay ve senenin Ocak 2004 olduğunu kabul edersek bu ayın takvimini şu şekilde gösterebiliriz:

```
>> calendar

      Jan 2004
 S   M   Tu   W   Th   F   S
 0   0   0    0   1    2   3
 4   5   6    7   8    9   10
11  12  13   14  15   16  17
18  19  20   21  22   23  24
25  26  27   28  29   30  31
 0   0   0    0   0    0   0

>>
```

Yukarıdaki örnekte 1 Ocak Perşembe gündü.

calendar(y,m) fonksiyonunu kullanarak verilen herhangibir sene ve ayın takvimini bulabiliriz. Burada *y* seneyi ve *m* ise ayı göstermektedir. Aşağıda bir örnek verilmiştir.

Örnek 6.1

10 Şubat 1970 de doğduğunuzu kabul edersek doğduşunuz ay ve yılın takvimini bulunuz.

Çözüm 6.1

Şubat 1970 yılının takvimini şu şekilde bulabiliriz:

```
>> calendar(1970,2)

          Feb 1970
S     M     Tu    W     Th    F     S
1     2     3     4     5     6     7
8     9     10    11    12    13    14
15    16    17    18    19    20    21
22    23    24    25    26    27    28
0     0     0     0     0     0     0
0     0     0     0     0     0     0

>>
```

Bu durumda, doğduğunuz gün Salı gündü.

Örnek 6.2

Mart 2001 yılında hangi günlerin Pazar olduğunu bulunuz.

Çözüm 6.2

calendar fonksiyonunu kullanarak Mart 2001 yılı takvimini *takvim* gibi bir dizide saklayabiliriz. Daha sonra bu dizinin birinci sütunundaki elemenları gösterirsek Mart 2001 yılının Pazar günlerini göstermiş oluruz.

Gerekli olan komutlar şunlardır:

```
>> takvim = calendar(2001,3);
>> takvim(1:7)

ans =

0    4    11    18    25    0    0

>>
```

veya, 2001 yılı Mart ayının 4, 11, 18 ve 25 inci günleri Pazar günlerdi.

6.2 cputime Fonksiyonu

cputime fonksiyonu MATLAB programını çalıştırdıktan sonra ne kadar zaman için kullandığımızı saniye olarak gösterir. Bu fonksiyonu kullanarak bir işlemin ne kadar zaman aldığını kolaylıkla bulabiliriz. Bunun için cputime fonksiyonunu işlemden hemen önce kullanıp zamanı bir değişkende saklarız. Daha sonra işlem bitince yine bu fonksiyonu kullanıp zamanı yine okuruz ve ilk okuduğumuz zamandan çıkarıp işlemin ne kadar zamanda yapıldığını hesaplayabiliriz. Gerekli olan adımlar aşağıda gösterilmiştir:

```
t = cputime;
işlem
cputime – t
```

Örnek 6.3

30 derecenin trigonometrik sinüsünün MATLAB kullanılarak ne kadar zamanda hesaplandığını bulunuz.

Çözüm 6.3

cputime fonksiyonunu kullanarak geçen zamanı şu şekilde hesaplayabiliriz:

```
>> t = cputime;
>> sin(30*pi/180);
>> cputime - t

ans =

3.7190

>>
```

Yukarıdaki örnekte hesaplama 3.7190 saniye almıştır. Bu işlem bilgisayarımızın hızına bağlı olup sizin bilgisayarınızda büyük bir olasılıkla değişik bir değer verecektir.

6.3 MATLAB'da Tarih Formatı

MATLAB'da tarihi 3 değişik formatta göstermemiz mümkündür: *karakter dizisi, seri tarih sayısı*, ve *tarih vektörü*.

6.3.1 Karakter Dizisi

Karakter dizisi en yaygın olarak kullanılan format olup bu formatta tarih gün-ay-yıl olarak gösterilir. Örneğin, '10-Sep-1980'. Burada günler 1 den 31 e kadar herhangibir tam sayı olabilir. Aylar 1 den 12 ye kadar bir tam sayı veya ayın ilk 3 karakteri olabilir. Yıl 2 veya 4 rakam olarak belirtilebilir ve eğer yıl belirtilmemişse şimdiki yıl kabul edilir. Gün-ay-yıl formatına ilaveten istenilirse saat-dakika-saniye de gösterilebilir.

Şimdiki yılın 10 Mayıs 2002 olduğunu kabul edersek, aşağıdakilerin hepsi de ayni neticeyi verirler:

'10-May-2002'
'10-May'
'May 10, 2002'

‘5/10/2002’

’10-May-2002, 13:10’

6.3.2 Seri Tarih Sayısı

MATLAB kendi iç işlemlerinde seri tarih sayısı formatını kullanır. Bu formatda tarih bir sayı olarak belirtilir. Bu sayının başlangıcı (1. nci seri tarih sayısı) 1 Ocak 0000 senesi olarak alınmıştır. Örneğin, 10 Ocak 2000 yılı seri tarih sayısı olarak 730495 olarak gösterilir (1-1-0000 tarihinden 10-1-2000 tarihine kadar 730495 gün geçmiştir).

MATLAB’da datenum fonksiyonunu kullanarak verilen bir tarih karakter dizisini seri tarih sayısına dönüştirebiliriz.

Örnek 6.4

3 Aralık 2000 yılını seri tarih sayısına dönüştürünüz.

Çözüm 6.4

datenum fonksiyonunu kullanarak gerekli dönüşümü yapabiliriz:

```
>> datenum('12-3-2000')

ans =

730823

>>
```

İstenilen seri tarih sayısı 730823 olarak bulunur.

Ayni şekilde, datestr fonksiyonunu kullanarak verilen bir seri tarih sayısını tarih karakter dizisine dönüştürebiliriz.

Örnek 6.5

730824 seri tarih sayısının tarih karakter dizisi olarak eşidini bulunuz.

Çözüm 6.5

datestr fonksiyonunu kullanarak istenilen dönüşümü yapabiliriz:

```
>> datestr(730824)

ans =

04-Dec-2000

>>
```

Cevap, 4 Aralık 2000 olarak bulunur.

6.3.3 Tarih Vektörü

Tarih vektörü bazı MATLAB fonksiyonlarında kullanılmaktadır. Bir tarih vektörü şu elemanlardan meydana gelmiştir:

[yıl ay gün saat dakika saniye]

örneğin, 5 Ekim 1998 tarihini MATLAB'da şu 3 değişik şekilde gösterebiliriz:

karakter dizisi:	05-Oct-1998
seri tarih sayısı:	730033
tarih vektörü:	1998 10 05 0 0 0

datevec fonksiyonu verilen bir tarih karakter dizisini veya seri tarih sayısını elemanlarına ayırır. Örneğin,

```
>> d='10-oct-2000';
>> datevec(d)

ans =

2000        10        10         0         0         0

>>
```

veya, 731000 seri tarih sayısının tarih elemanlarını şu şekilde bulabiliriz:

```
>> datevec(731000)

ans =

2001         5        29         0         0         0

>>
```

Bu örnekte 29 Mayıs 2001 tarihi belirtilmiştir.

6.4 clock Fonksiyonu

clock fonksiyonu şimdiki tarih ve saatı tarih vektörü olarak vermektedir. Örneğin, şimdiki tarihin 10 Ocak 2004 ve saatın 13:09:35 olduğunu kabul edersek,

```
>>c = clock

c =

  1.0e+003 *

    2.0040    0.0010    0.0100    0.0130    0.0090    0.0356

>>
```

6.5 now Fonksiyonu

Bu fonksiyon şimdiki tarih ve saatı seri tarih sayısı olarak verir. Aşağıdaki örnekte şimdiki tarih ve saat ilk olarak seri tarih sayısı olarak bulunur ve daha sonra karakter dizisine dönüştürülür. Şimdiki tarihin 10 Ocak 2004 ve saatın 13:15:15 olduğunu kabul edersek:

```
>> d=now

d =

7.3196e+005

>> datestr(d)

ans =

10-Jan-2004 13:15:15

>>
```

6.6 weekday Fonksiyonu

Bu fonksiyon haftanın gününü sayı olarak ve ayni zamanda karakter dizisi olarak vermektedir. Pazar günü 1 olarak kabul edilmiştir. Örneğin, 10 Ocak 2004 tarihi Cumartesidir. MATLAB'ı kullanarak bu günün ne olduğunu şu şekilde bulabiliriz:

```
>> [n,s]=weekday('10-Jan-2004')

n =
        7

s =

        Sat
>>
```

6.7 eomday Fonksiyonu

Bu fonksiyon, verilen herhangibir senede, verilen herhangibir ayın son gününü belirtir. Fonksiyon şu şekilde kullanılmaktadır:

```
eomday(yıl,ay)
```

Örneğin, 2004 senesi Şubat ayı 29 çekmektedir:

```
>> e = eomday(2004,2)

e =

29

>>
```

6.8 date Fonksiyonu

Bu fonksiyon şimdiki tarihi gün-ay-yıl formatında vermektedir. Örneğin, şimdiki tarihin 10 Ocak 2004 olduğunu kabul edersek:

```
>> date

ans =

10-Jan-2004

>>
```

6.9 etime Fonksiyonu

Bu fonksiyon, verilen t1 ve t2 gibi iki tarih vektörü arasındaki zaman farkını saniye olarak vermektedir. Fonksiyonun kullanılışı şu şekildedir:

```
etime(t2,t1)
```

etime fonksiyonu genellikle herhangibir işlemin ne kadar zamanda yapıldığını hesaplamak için kullanılmaktadır. Bunun için ilk olarak işlem başında zaman vektörü okunur. İşlemden hemen sonra ise zaman vektörü tekrar okunur ve etime fonksiyonu kullanılarak işlemin ne kadar zamanda yapıldığı hesaplanır. Aşağıda bir örnek verilmiştir.

Örnek 6.6

Aşağıdaki döngü işleminin kaç saniyede yapıldığını hesaplayınız:

```
for j = 1:20000
        b = j*j
end
```

Çözüm 6.6

İşlemin kaç saniyede yapıldığını şu şekilde hesaplayabiliriz:

```
t1 = clock;
for j = 1:20000
        b = j*j;
end
t2 = clock;
etime(t2, t1)

ans =

6.6720
```

Bu örnekte döngü işlemi 6.672 saniyede tamamlanmıştır.

Yukarıdaki işlemleri şu şekilde de yapabiliriz:

```
t = clock;
for j = 1:20000
        b = i*i;
end
etime(clock, t)

ans =

5.1400
```

6.10 tic, toc Fonksiyonları

tic ve toc fonksiyonları clock ve etime fomksiyonlarını kullanarak bir işlemin ne kadar zamanda yapıldığını hesaplarlar. Bu fonksiyonlar bir kronometre gibi çalışırlar. tic fonksiyonu kronometreyi çalıştırır, toc fonksiyonu ise kronometreyi durdurur ve geçen zamanı saniye olarak gösterir. Aşağıda bir örnek verilmiştir.

Örnek 6.7

Aşağıdaki döngü işleminin kaç saniyede yapıldığını hesaplayınız:

```
for j = 1:20000
        b = j*j*j
end
```

Çözüm 6.7

İşlemin kaç saniyede yapıldığını şu şekilde hesaplayabiliriz:

```
tic;
for j = 1:20000
```

```
        b = j*j*j;
end
toc

elapsed_time =

    10.421
```

6.11 Tarih Ve Saat Programlaması

MATLAB ile tarihi ve saatı kullanan m-dosyaları yazmamız mümkündür. Bu konuda birkaç örnek aşağıda verilmiştir.

Örnek 6.8

Klavyeden karakter dizisi olarak girilen bir tarihin haftanın hangi günü olduğunu hesaplayıp ekranda gösteren bir MATLAB programı yazınız. Programın ismini *haftanın_gunu.m* koyunuz.

Çözüm 6.8

Program listesi aşağıda verilmiştir. Program ilk olarak klavyeden tarihi olur. Daha sonra *weekday* fonsiyonu kullanılarak haftanın günü bulunur ve ekranda gösterilir.

```
%
% HAFTANIN GUNU
% =============
%
% Bu program klavyeden tarihi bir karakter dizisi olarak alir
% ve bu tarihe karsilik gelen haftanin gununu hesaplayip
% ekranda gosterir.
%
gun = input('Tarihi karakter dizisi olarak giriniz: ');
[n,s] = weekday(gun);
%
```

```
% Simdi haftanin gununu ekranda goster
%
switch n
   case 1
      disp('Pazar')
   case 2
      disp('Pazartesi')
   case 3
      disp('Sali')
   case 4
      disp('Carsamba')
   case 5
      disp('Persembe')
   case 6
      disp('Cuma')
   case 7
      disp('Cumartes')
end
```

Programın çalışmasına örnek aşağıda verilmiştir. Burada tarih 11 Ocak 2004 olarak verilmiş ve gün Pazar olarak gösterilmiştir.

```
>> haftanin_gunu
Tarihi karakter dizisi olarak giriniz: '11-Jan-2004'
Pazar
>>
```

Örnek 6.9

Klavyeden girilen iki tarih arasında kaç gün olduğunu hesaplayıp ekranda gösteren bir program yazınız. Programa *gunler.m* dosya ismini veriniz.

Çözüm 6.9

İstenilen program listesi aşağıda verilmiştir. Programda ilk olarak her iki tarih de okunur. Daha sonra bu tarihler seri tarih sayısına dönüştürülür ve her iki tarih arasındaki fark hesaplanıp *fprintf* fonksiyonu kullanılarak ekranda gösterilir.

```
%
```

```
% IKI TARIH ARASINDAKI GUNLER
% ==========================
%
% Bu program klavyeden girilen iki tarih arasinda kac gun
% oldugunu hesaplar ve ekranda gosterir.  Ikinci tarihin
% birinci tarihten buyuk olmasi gerekir, aksi halde netice
% negatif olarak cikar.
%
tarih1 = input('Birinci tarihi giriniz: ');
tarih2 = input('Ikinci tarihi giriniz : ');
n1 = datenum(tarih1);
n2 = datenum(tarih2);
n = n2 - n1;
fprintf('Gun sayisi = %g\n',n)
```

Programın çalışmasına bir örnek aşağıda verilmiştir. Bu örnekte 10 Ocak 2004 ve 12 Ocak 2004 arasındaki gün sayısı 2 olarak hesaplanmış ve ekranda gösterilmiştir.

```
>> gunler
Birinci tarihi giriniz: '10-jan-2004'
Ikinci tarihi giriniz : '12-jan-2004'
Gun sayisi = 2
```

6.12 Sorular

1. 20 Kasım 1990 tarihini tarih vektörüne dönüştürünüz.

2. 730212 seri tarih sayısının hangi tarih olduğunu hesaplayınız.

3. Verilen herhangibir tarih ve 29 Ekim Cumhuriyet bayramı arasında kaç gün olduğunu hesaplayınız.

4. Atatürk'ün ölüm tarihi olan 10 Kasım 1938 in haftanın hangi günü olduğunu hesaplayınız.

5. 1 den 100 e kadar olan sayıların toplamını hesaplayan

bir MATLAB programı yazınız. Daha sonra bu işlemin bilgisayarınızda ne kadar zamanda yapıldığını hesaplayıp ekranda gösteriniz.

6. tic ve toc fonksiyonlarının ne şekilde kullanıldıklarını bir örnekle açıklayınız.

7. Verilen bir yıl içerisinde hangi günlerin Cumartesi olduğunu bulup ekranda gösteren bir MATLAB programı yazınız.

8. 2004 yılı Şubat ayının takvimini gösteren MATLAB komutu nedir ?

10. Doğum tarihinizden şimdiye kadar kaç gün geçtiğini hesaplayan bir MATLAB programı yazınız.

11. 2000 ve 2020 yılları Şubat aylarının kaç gün olduklarını hesaplayıp tablo şeklinde gösteren bir MATLAB programı yazınız.

7
VERİLEN NOKTALARDAN GEÇEN EĞRİNİN DENKLEMİNİ BULMAK

Birçok deneylerde elde etmiş olduğumuz noktalardan en iyi şekilde geçecek olan bir eğrinin denklemini bulmak isteriz. Bulmuş olacağımız eğri her noktadan geçmeyebilir, fakat verilen noktalara en iyi şekilde uyan bir eğri olmalıdır. En iyi uyan eğriyi bulurken eğrinin her noktaya olan uzaklığı, yani hata göz önünde bulundurulur ve bu hatanın minimize edilmesine çalışılır. MATLAB'ı kullanarak verilen noktalardan geçen eğrinin denklemini bulurken genellikle bu eğriyi bir polinom olarak tanımlarız ve bu polinomun katsayılarını bulmaya çalışırız.

n derecede bir polinomun *n+1* tane katsayısı olup böyle bir polinom şu şekilde yazılabilir:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$$

Genel olarak elimizde *n+1* tane veri noktası varsa bu noktaların hepsinden de *n* dereceli bir polinom geçirilebilir. Fakat bu yaklaşım pratikte mümkün olmayıp genellikle çok daha küçük dereceli bir polinom çözümü aranmaktadır.

MATLAB fonksiyonu polyfit'i kullanarak verilen noktalardan geçen bir eğrinin katsayılarını bulabiliriz. Bu fonksiyonun kullanımı ve çeşitli örnekler aşağıdaki bölümlerde verilmiştir.

7.1 polyfit Fonksiyonu

polyfit fonksiyonun kullanış şekli şöyledir:

p = polyfit(x, y, n)

fonksiyon, (x, y) noktalarından geçecek ve *n* dereceli olan en uygun eğrinin katsayılarını vermektedir. Burada *p* vektöründe katsayılar alçalan derece sırasına göre verilmektedir. Kısacası, *p* vektörü şu katsayıları ihtiva etmektedir:

$p = [a_n \ a_{n-1} \ ..... \ a_1 \ a_0]$

polinomun katsayılarını bulduktan sonra, daha önce de gördüğümüz polyval fonksiyonunu kullanarak bulmuş olduğumuz polinomun değerlerini ve deneyden elde etmiş olduğumuz verileri karşılaştırarak (genellikle ayni eksenlere her iki grafiği de çizerek) polinomun uygunluğunu tesbit etmiş oluruz. Eğer polinom verilen noktalara uygun değilse, polinomun derecesini artırarak yeni katsayılar elde ederiz ve böylece işlem devam etmiş olur.

polyfit ve polyval fonksiyonlarının kullanımını gösteren ve verilen noktalardan geçen bir eğrinin denklemini bulan çeşitli örnekler aşağıda verilmiştir.

Örnek 7.1

Laboratuvarda yapılan bir deneyde x ve y değişkenleri arasında şu bağıntı elde edilmiştir:

X:	0	1	2	3	4	5	6
Y:	5	8	7	6	9	11	14

Bu noktalardan geçecek çeşitli derecelerdeki eğrilerin denklemlerini bulunuz ve bulmuş olduğunuz eğrilerin grafiklerini çizerek bu grafikleri veri noktalarının grafiği ile karşılaştırınız.

Çözüm 7.1

<u>Birinci derece polinom:</u>

İlk olarak veri noktalarına birinci derece bir polinom (bir doğru) çizelim. Gerekli MATLAB komutlaru şunlardır:

```
>> x=[0 1 2 3 4 5 6];
>> y=[5 8 7 6 9 11 14];
>> d=polyfit(x,y,1)

d =

1.2500    4.8214

>>
```

istediğimiz polinomun denklemi p(x) = 1.25x + 4.8214 olur.

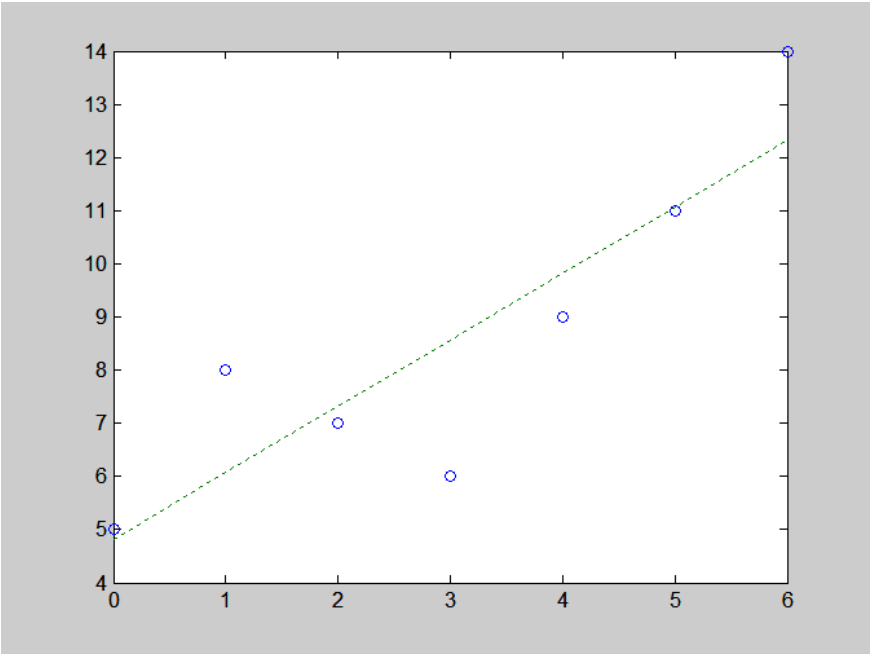
Şimdi bu polinomun x = 0 ve x = 6 arasındaki değerlerini bulup grafiğini çizelim ve veri noktalarının grafiği ile karşılaştıralım:

```
>> z = 0:6;
>> s = polyval(d, z);
>> plot(x,y,'o',z,s,':')
>>
```

Şekil 7.1 de verilen grafiği elde ederiz. Burada veri noktaları ‘o’ işareti ile, polinom ise ‘-‘ işareti ile gösterilmiştir. Şekilden de görüleceği gibi, veri noktalarına birinci derece polinom çizmek oldukça hatalı olur. Şimdi, ikinci derece bir polinomu deneyelim.

<u>İkinci derece polinom:</u>

İkinci derece polinom için gerekli olan MATLAB komutları aşağıda verilmiştir (burada normal olarak x ve y vektörlerini tekrar tanımlamaya gerek yoktur):



Şekil 7.1 Veri noktaları ve p(x) = 1.25x + 4.8214 polinomu

```
>> x=[0 1 2 3 4 5 6];
>> y=[5 8 7 6 9 11 14];
>> d=polyfit(x,y,2)

d =

    0.2738   -0.3929    6.1905

>>
```

istediğimiz polinomun denklemi

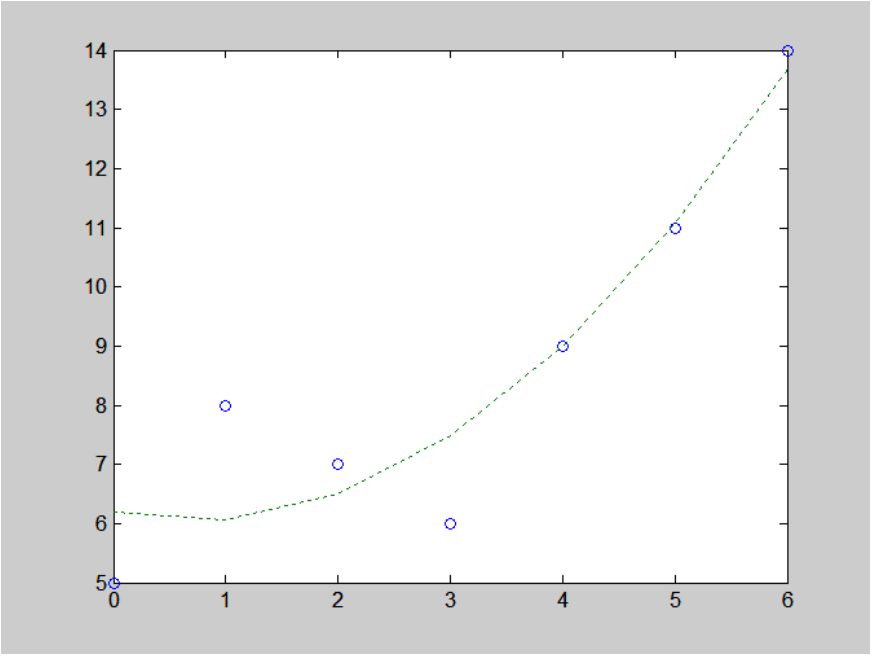
$$p(x) = 0.2738x^2 – 0.3929x + 6.1905 \text{ olur.}$$

Şimdi bu polinomun x = 0 ve x = 6 arasındaki değerlerini bulup grafiğini çizelim ve veri noktalarının grafiği ile karşılaştıralım:

```
>> z = 0:6;
```

```
>> s = polyval(d, z);
>> plot(x,y,'o',z,s,':')
```

Şekil 7.2 de verilen grafiği elde ederiz. Bu grafikten de görüleceği gibi ikinci derece polinomu da pek iyi netice vermemiştir. Şimdi üçüncü derece polinomunu deneyelim.



Şekil 7.2 Veri noktaları ve
$p(x) = 0.2738x^2 – 0.3929x + 6.1905$ polinomu

Üçüncü derece polinom:

Üçüncü derece polinom için gerekli olan MATLAB komutları aşağıda verilmiştir (burada normal olarak x ve y vektörlerini tekrar tanımlamaya gerek yoktur):

```
>> x=[0 1 2 3 4 5 6];
>> y=[5 8 7 6 9 11 14];
>> d=polyfit(x,y,3)

d =
```

```
    0.1111   -0.7262    1.8294    5.5238

>>
```

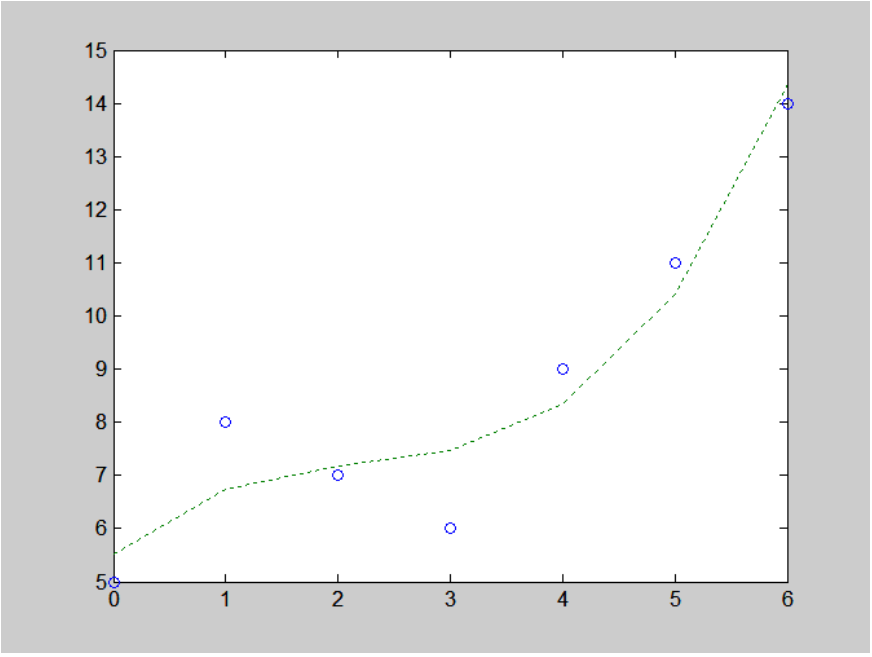
istediğimiz polinomun denklemi

$$p(x) = 0.1111x^3 – 0.7262x^2 + 1.8294x + 5.5238 \text{ olur.}$$

Şimdi bu polinomun x = 0 ve x = 6 arasındaki değerlerini bulup grafiğini çizelim:

```
>> z = 0:6;
>> s = polyval(d, z);
>> plot(x,y,'o',z,s,':')
```

Şekil 7.3 de verilen grafiği elde ederiz. Bu grafikten de görüleceği gibi üçüncü derece polinomu da pek iyi netice vermemiştir. Şimdi dördüncü derece polinomunu deneyelim.



Şekil 7.3 Veri noktaları ve
$p(x) = 0.1111x^3 – 0.7262x^2 + 1.8294x + 5.5238$

Dördüncü derece polinom:

Dördüncü derece polinom için gerekli olan MATLAB komutları aşağıda verilmiştir (burada normal olarak x ve y vektörlerini tekrar tanımlamaya gerek yoktur):

```
>> x=[0 1 2 3 4 5 6];
>> y=[5 8 7 6 9 11 14];
>> d=polyfit(x,y,4)

d =

-0.0909    1.2020   -4.7652    6.4268    5.0563
```

istediğimiz polinomun denklemi

$p(x) = -0.0909x^4 + 1.2020x^3 – 4.7652x^2 + 6.4268x + 5.0563$ olur.

Şimdi bu polinomun x = 0 ve x = 6 arasındaki değerlerini bulup grafiğini çizelim:

```
>> z = 0:6;
>> s = polyval(d, z);
>> plot(x,y,'o',z,s,':')
```

Şekil 7.4 de verilen grafiği elde ederiz. Bu grafik diğerlerine kıyasla daha iyi netice vermesine rağmen, son olarak bir de verilen noktaladan beşinci derecede bir polinom geçirmeye çalışalım.

Dördüncü derece polinom:

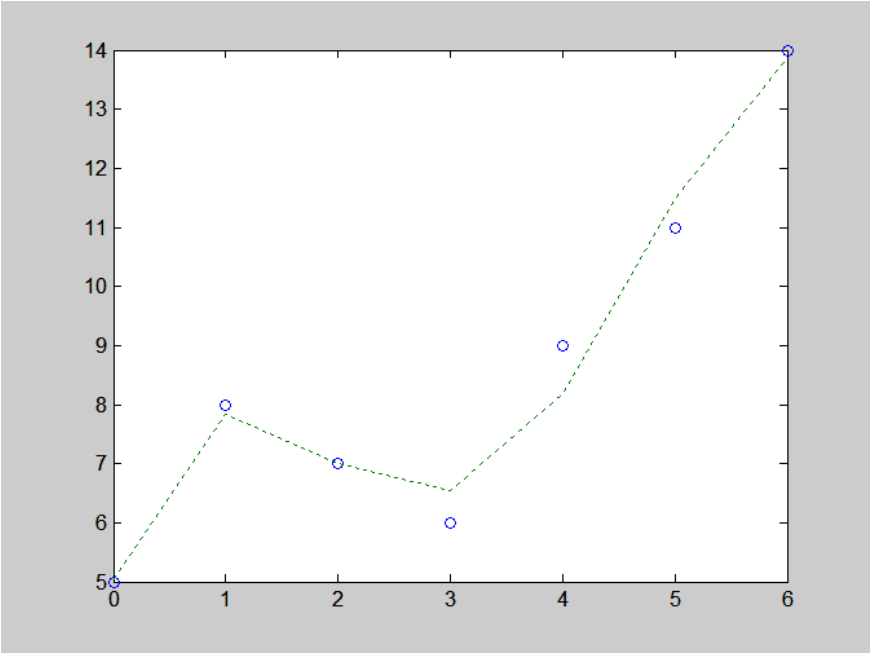
Dördüncü derece polinom için gerekli olan MATLAB komutları aşağıda verilmiştir (burada normal olarak x ve y vektörlerini tekrar tanımlamaya gerek yoktur):

```
>> x=[0 1 2 3 4 5 6];
>> y=[5 8 7 6 9 11 14];
>> d=polyfit(x,y,4)

d =
```

```
0.0292   -0.5284    3.4867   -9.5777    9.7795    4.9729

>>
```

istediğimiz polinomun denklemi ise

$p(x) = 0.0292x^5 - 0.5284x^4 + 3.4867x^3 - 9.5777x^2 + 9.7795X + 4.9729$ olur.

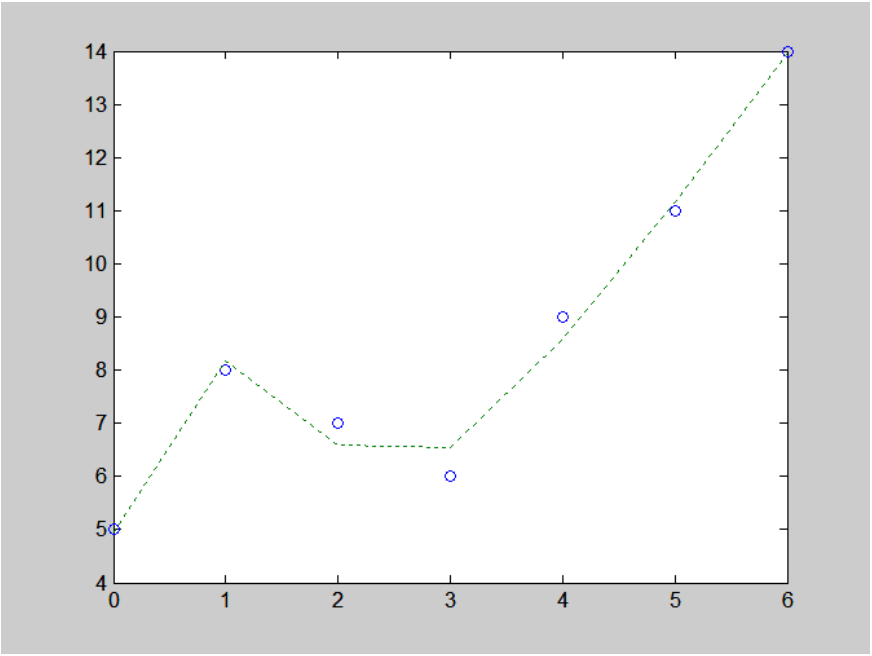


Şekil 7.4 Veri noktaları ve
$p(x) = -0.0909x^4 + 1.2020x^3 - 4.7652x^2 + 6.4268x + 5.0563$

Şimdi bu polinomun x = 0 ve x = 6 arasındaki değerlerini bulup grafiğini çizelim:

```
>> z = 0:6;
>> s = polyval(d, z);
>> plot(x,y,'o',z,s,':')
```

Şekil 7.5 de verilen grafiği elde ederiz. Şekilden de görüleceği gibi, bu grafik veri değerlerine oldukça uygundur.

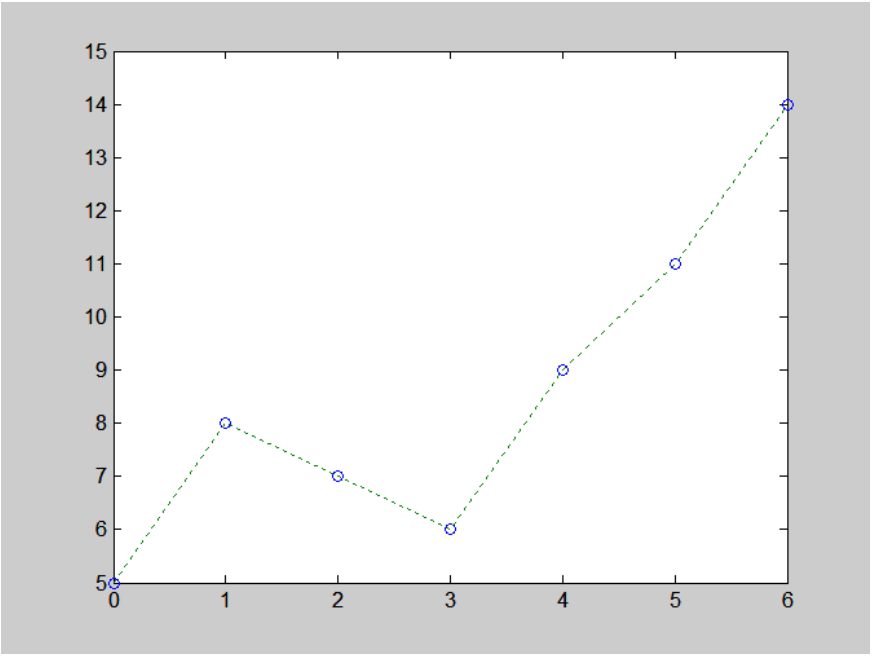


Şekil 7.5 Veri noktaları ve
$p(x) = 0.0292x^5 – 0.5284x^4 + 3.4867x^3 – 9.5777x^2 + 9.7795X + 4.9729$

Daha önce de bahsedildiği gibi, n+1 veri noktasına n dereceli bir polinomu tam olarak yerleştirebiliriz. Bu örnekte 7 tane veri noktası olduğu için, 6 dereceli bir polinomun tam olarak yerleşmesi gerekir. Şekil 7.6 da 6 dereceli bir polinomun veri noktalarına tam olarak yerleştiği gösterilmiştir.

Örnek 7.2

Klavyeden girilen veri noktalarına bir polinom yerleştirmek için MATLAB programı yazınız. Polinomun derecesi, ve veriler klavyeden girilecektir. Veri noktalarını ve elde ettiğiniz polinomu bir grafikte gösteriniz. Ayrıca, elde ettiğiniz polinomun katsayılarını ekranda gösteriniz. Programı *polinom.m* isimli bir dosyada saklayınız.



Şekil 7.6 6 dereceli bir polinom veri noktalarına tam olarak yerleşmektedir.

Çözüm 7.2

İstenilen program aşağıda verilmiştir. Program ilk olarak istenilen polinom derecesini klavyeden okur. Daha sonra X ve Y değerleri yine klavyeden okunur. Program polyfit fonksiyonunu kullanarak verilen noktalardan geçecek polinomun katsayılarını hesaplar ve ekranda gösterir. Daha sonra verilerin minimum ve maksimum noktaları bulunur ve x değerleri üretilerek polinomun ve verilerin grafikleri çizilir.

```
%
%   VERILEN NOKTALARA BIR POLINOM YERLESTIREN
%   PROGRAM
% =============-==============================
%
% Bu program klavyeden polinom derecesini ve verileri
% okur ve bu verilere istenilen derecede bir polinom
% yerlestirir.  Daha sonra grafik cizilir ve
```

```
% polinomun ne kadar iyi oldugu gorulebilir. Program
% ayrica polinomun katsayilarini ekranda gosterir.
%
n = input('Istenilen polinomun derecesi: ');
disp('X - Verilerini giriniz...');
x = input('')
disp('Y - Verilerini giriniz...');
y = input('')
m = length(x);
d = polyfit(x,y,n);
disp('Polinom katsayilari:');
d
z = linspace(min(x),max(x),m);
s = polyval(d,z);
plot(x,y,'o',z,s,':')
grid on
title('Verilen Noktalara Polinom Cizimi')
```

Programın çalışmasına bir örnek aşağıda verilmiştir. Bu örnekte X ve Y değerleri şu şekilde alınmış ve verilere ikinci derecede bir polinom yerleştirilmiştir:

X:	1	2	3	4	5
Y:	1	4	9	17	26

Klavyeden girilenler, kolay ayırt edilmeleri için altları çizilmiştir:

```
>> polinom
Istenilen polinomun derecesi: 2
X - Verilerini giriniz...
[1,2,3,4,5]

x =

1    2    3    4    5

Y - Verilerini giriniz...
[1,4,9,17,26]

y =

1    4    9   17   26
```

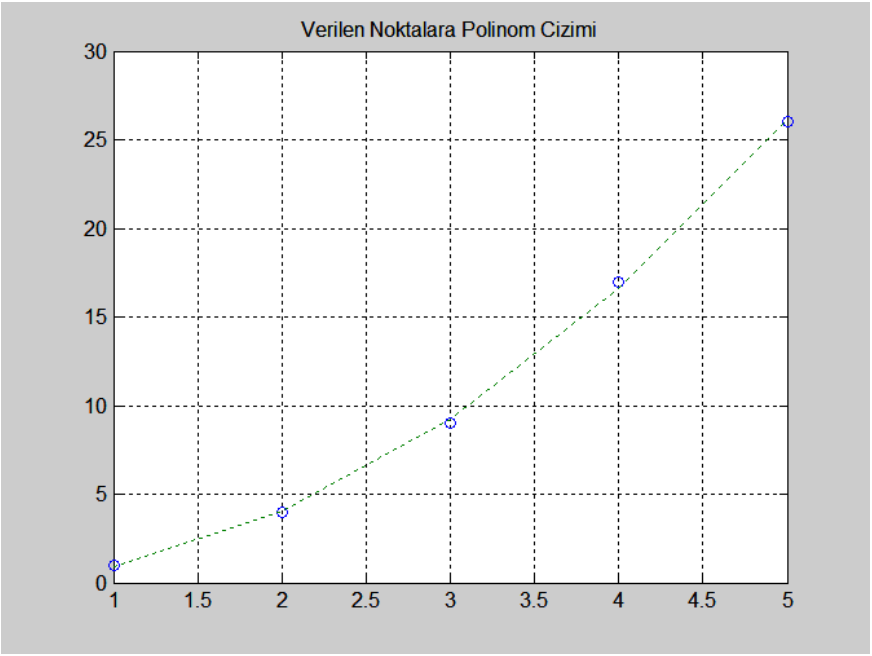
Polinom katsayilari:

d =

1.0714 -0.1286 0.0000

Bu örnekte, istenilen polinom $p(x) = 1.0714x^2 – 0.1286x$ olur.

Program ayrıca Şekil 7.7 de gösterilen grafiği çizmiştir.



Şekil 7.7 Program tarafından çizilen grafik

Verilen noktalara bir polinom çizerken seçilecek olan polinomun derecesi bilinmemektedir. Örnek 7.1 de gösterildiği gibi normal olarak, değişik polinom dereceleri alınarak elde edilen neticeler karşılaştırılır. Aşağıdaki örnekte, subplot komutu kullanılarak dereceleri değişik olan 4 grafik çizilmiştir. Bu grafiklere bakarak hangi polinomun daha uygun olduğuna hemen karar verebiliriz.

Örnek 7.3

Laboratuvarda yapılan bir deneyde aşağıdaki X ve Y değerleri elde edilmiştir:

X: 0 1 2 3 4 5
Y: 0 20 60 70 80

Bu noktalardan bir polinom geçirmek istiyoruz fakat polinomun derecesini bilmiyoruz, fakat polinom derecesinin 1 ve 4 arasında olmasını istiyoruz. polyfit komutunu kullanarak dereceleri 1 den 4 e kadar değişen 4 polinom elde ediniz ve subplot komutunu kullanarak bu polinomların grafiklerini çiziniz.

Çözüm 7.3

İstenilen MATLAB komutları aşağıda verilmiştir:

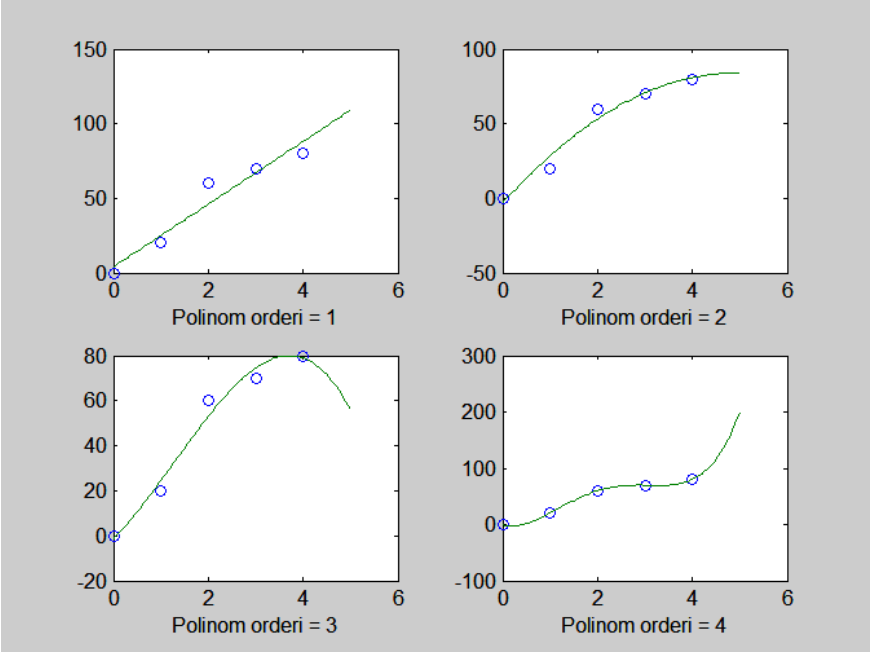
```
>> x = [0 1 2 3 4];
>> y = [0 20 60 70 80];
>> z = 0:0.05:5;
>> for j = 1:4
        disp(['Polinom katsayilari, derece = ',num2str(j)])
        poli = polyfit(x,y,j)
        deger = polyval(poli,z);
        subplot(2,2,j), plot(x,y,'o',z,deger);
        xlabel(['Polinom orderi = ',num2str(j)])
end
```

Program aşağıdaki polinom katsayılarını verir:

```
Polinom katsayilari, derece = 1

poli =

21.0000    4.0000

Polinom katsayilari, derece = 2

poli =
```

```
-3.5714   35.2857   -3.1429

Polinom katsayilari, derece = 3

poli =

-1.6667    6.4286   20.9524   -1.1429

Polinom katsayilari, derece = 4

poli =

3.3333  -28.3333   71.6667  -26.6667   -0.0000
```

X ve Y verileri girildikten sonra program bir döngü yapar ve bu döngü içerisinde polinom derecesi 1 den 4 e değiştikçe polinom katsayıları hesaplanır, polinomların değerleri hesaplanır ve polinomların grafikleri çizilir. Şekil 7.8 de program tarafından çizilen grafik gösterilmiştir.



Şekil 7.8 Programın çizdiği grafikler

7.2 Verilen Noktalara Otomatik Olarak Polinom Yerleştiren MATLAB Programı

MATLAB'ı kullanarak verilen noktalardan değişik derecede polinomlar geçirip neticeyi de hemen görebiliriz. Bunun için MATLAB'ın "Basic Fitting" diye bilinen MENU opsiyonunu seçmemiz gerekir. Bu opsiyon grafik çizdiğimiz zaman ortaya çıkan formda bulunmaktadır. Şimdi bu opsiyony kullanarak verilen noktalardan nasıl değişik derecede polinomlar geçireceğimizi görelim.

Örnek 7.4

Laboratuvarda yapmış olduğumuz bir deney neticesinde X ve Y değişkenleri arasında şu bağıntı bulunmuştur:

X:	0	2	4	6	8	10	12
Y:	0	5	17	37	67	102	146

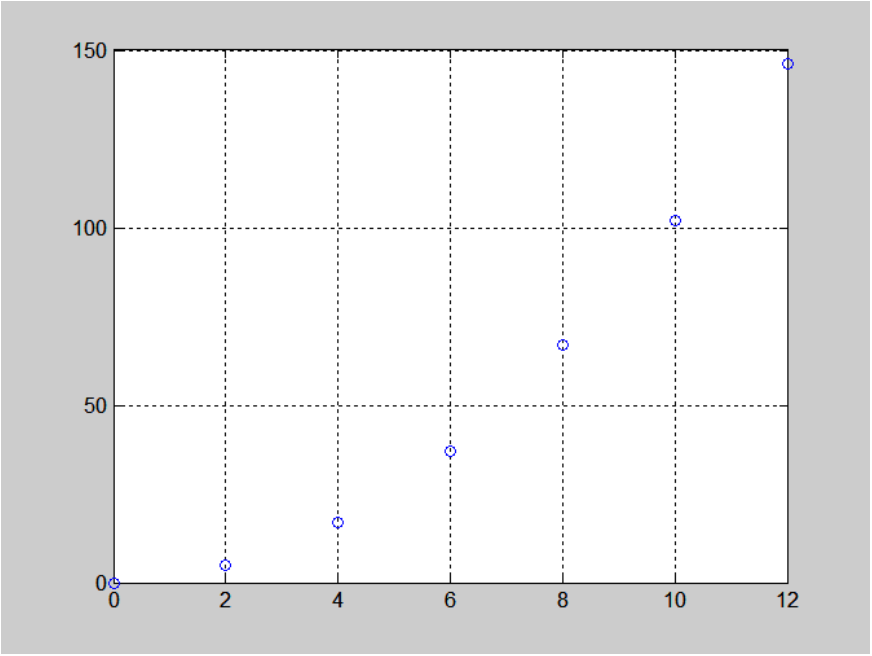
MATLAB'ın "Basic Fitting" opsiyonunu kullanarak bu noktalardan geçen polinomun derecesini ve katsayılarını bulunuz.

Çözüm 7.4

"Basic Fitting" opsiyonunu kullanmak için ilk olarak verilen noktların grafiğini çizmemiz gerekecektir. Bu işlem için gerekli olan MATLAB komutları şunlardır:

```
>> x = [0 2 4 6 8 10 12];
>> y = [0 5 17 37 67 102 146];
>> plot(x,y,'o'); grid on
```

Çizilmiş olan grafik Şekil 7.9 da gösterilmiştir. Bu grafikte veri noktaları 'o' karakteri ile gösterilmiştirler.

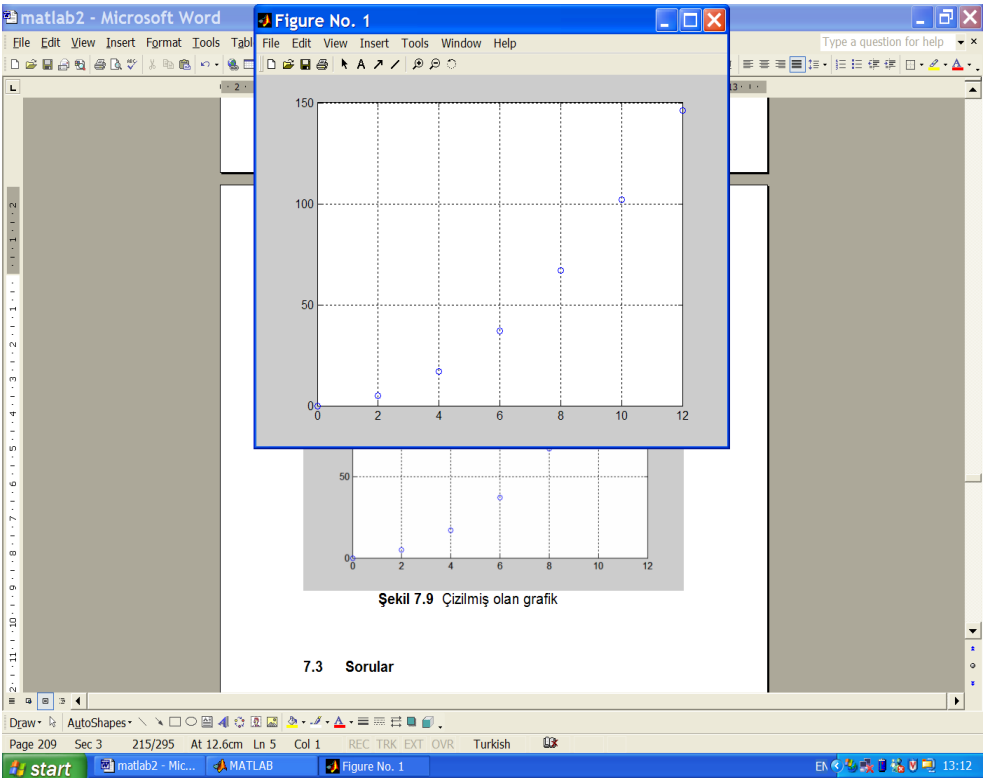


Şekil 7.9 Çizilmiş olan grafik

Grafiğin çizilmiş olduğu forma bakarsak Şekil 7.10 da gösterildiği gibi olup formun ismi Figure No. 1 olarak verilmiştir. Bu form üzerindeki menü seçeneklerine bakarsak, soldan sağa olmak üzere şunları görürüz:

File Edit View Insert Tools Window Help

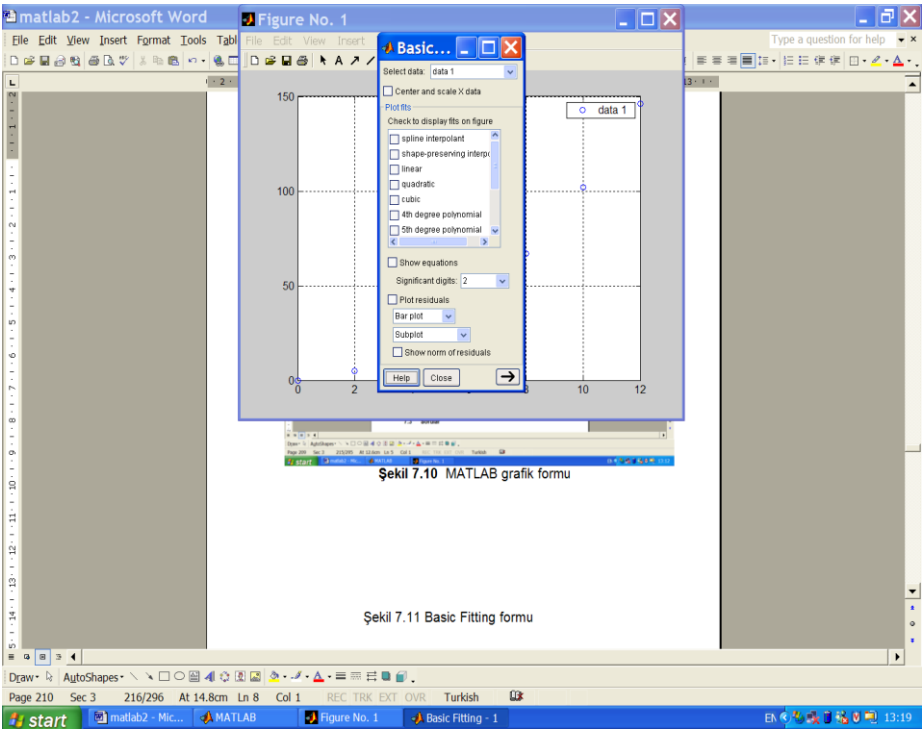
Veri noktalarımızdan polinom geçirmek için **Tools** menüsünü ve ordan da **Basic Fitting** menüsünü seçmemiz gerekir. Bu seçeneklerden sonra Şekil 7.11 de gösterilen formu elde ederiz.



Şekil 7.10 MATLAB grafik formu

Basic Fitting menüsü 2 bölümden meydana gelmektedir. Formun ikinci bölümünü elde etmek için alt sağ taraftaki ok işaretini tıklayınız ve Şekil 7.12 de gösterilen formu elde edeceksiniz.

Şimdi Şekil 7.12 de gösterilen formu kullanarak veri noktalarından geçirmek istediğiniz polinom derecesini tıklayınız. Örneğin, noktalardan ikinci derecede bir polinom geçirmek için **quadratic** butonunu tıklayınız. Otomatik olarak grafik üzerindeki veri noktalarınızdan bir grafik geçecektir. Şimdi **show equations** butonunu tıklarsanız veri noktalarınızdan geçen polinomun katsayılarını formun sağdaki ikinci bölümünde göreceksiniz. Ayni zamanda grafiğinizin sol üst köşesine polinomun denklemi otomatik olarak yazılacaktır. Şekil 7.13 ve Şekil 7.14 de polinom katsayıları ve polinomun grafiği gösterilmiştir.



Şekil 7.11 Basic Fitting formu

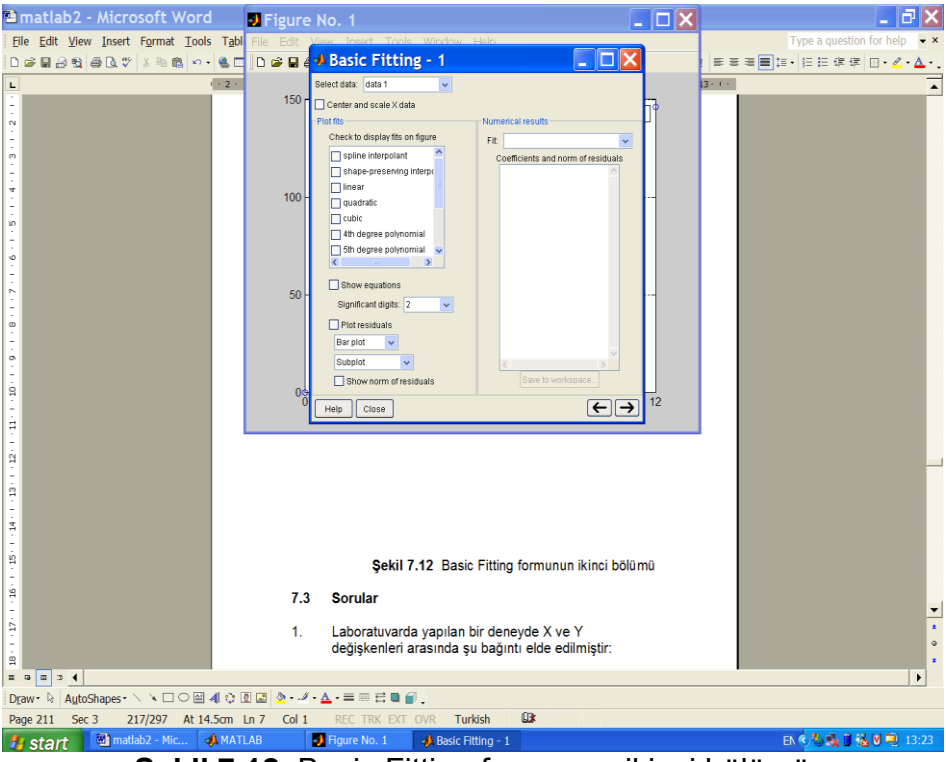
Veri noktalarından geçen polinomun denklemi şu şekilde verilmiştir:

$$y = 0.98x^2 + 0.39x + 3.1e\text{-}14$$

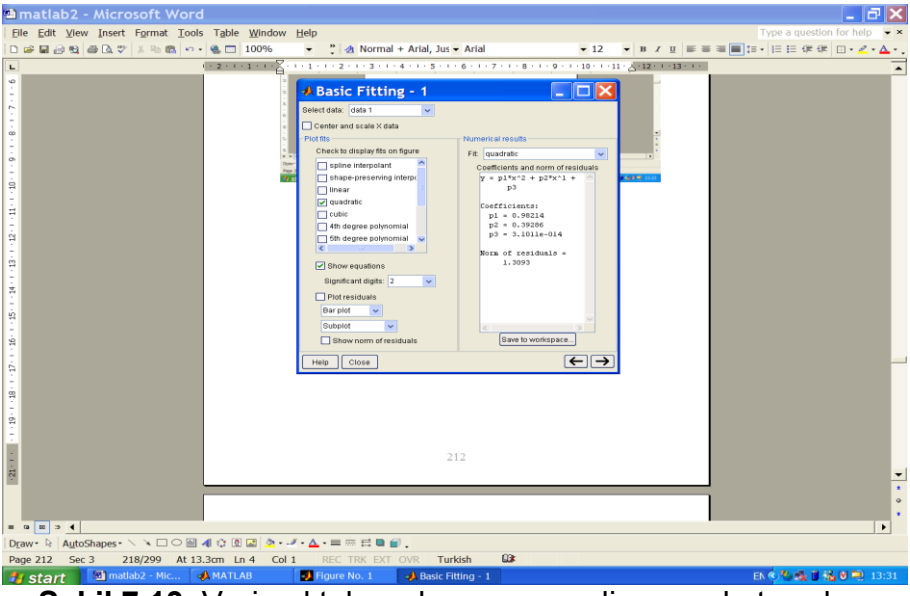
Son term pratik olarak sıfır olduğu için polinomu şu şekilde yazabiliriz:

$$y = 0.98x^2 + 0.39x$$

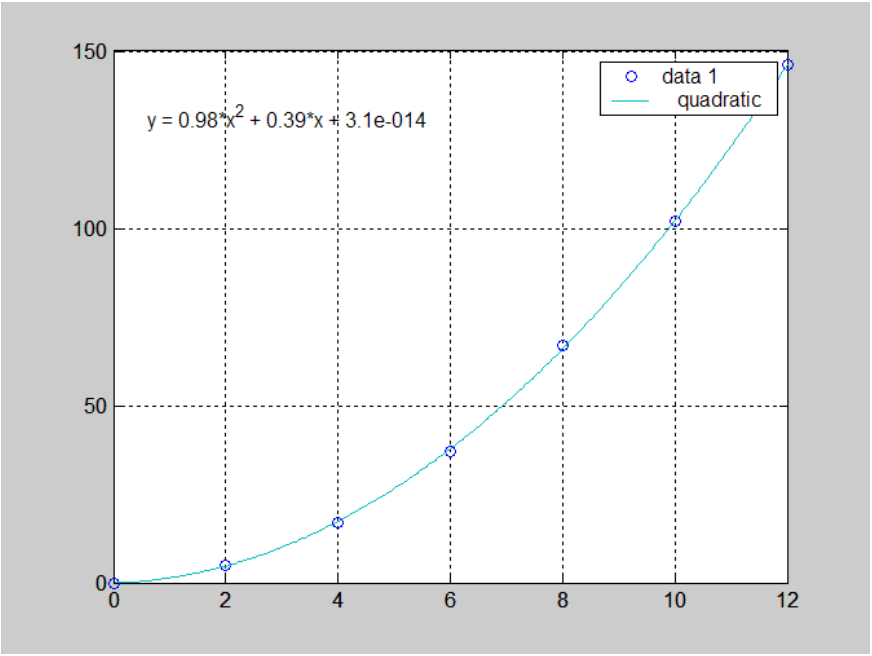
Şekil 7.14 den de görüleceği gibi bulmuş olduğumuz bu polinom veri noktalarımıza gayet iyi bir şekilde uymaktadır.



Şekil 7.12 Basic Fitting formunun ikinci bölümü



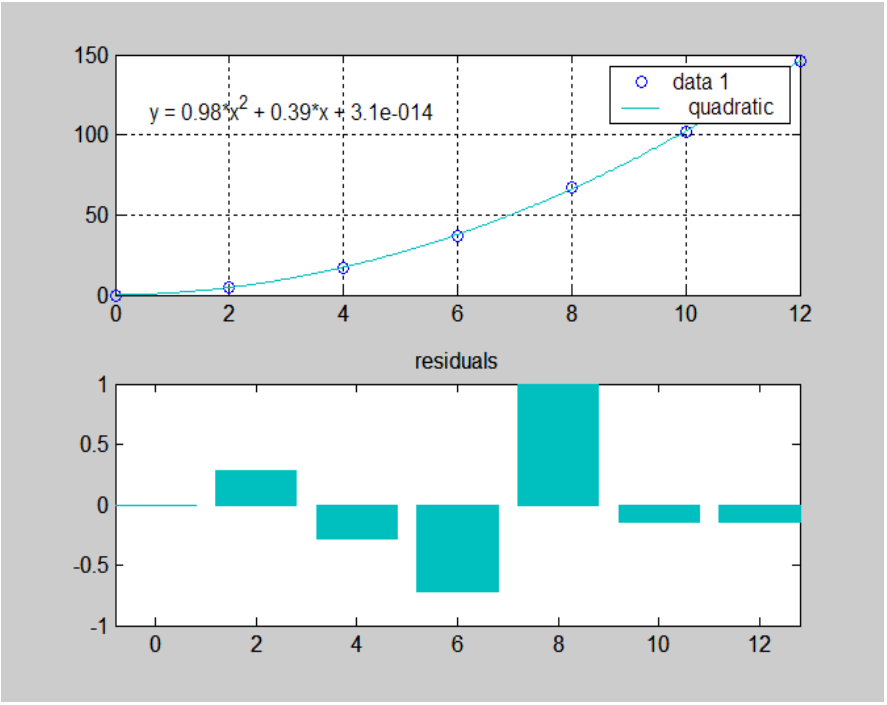
Şekil 7.13 Veri noktalarından geçen polinomun katsayıları



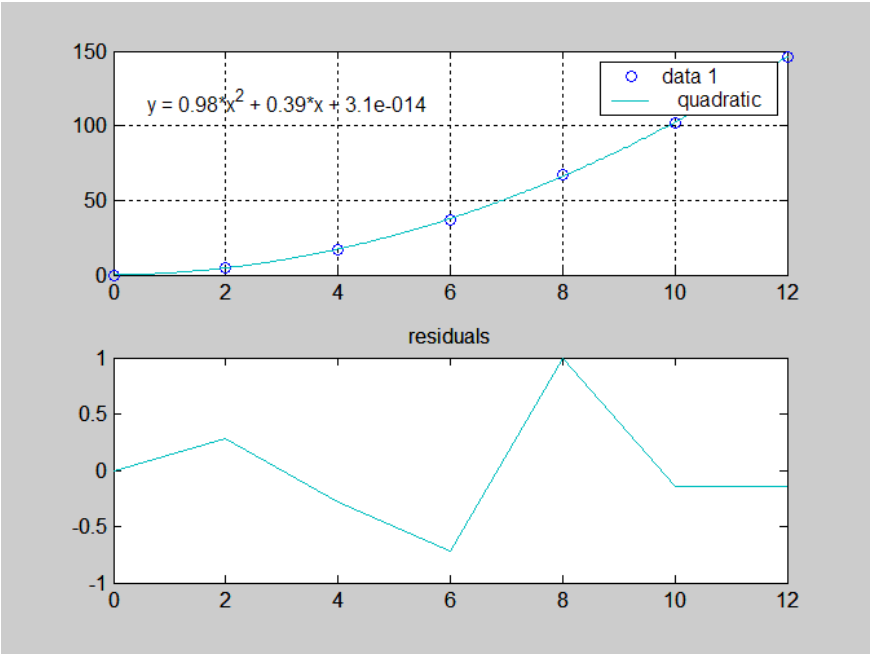
Şekil 7.14 Veri noktalarından geçen polinom

Bulmuş olduğumuz polinomun veri noktalarına uyumluluğunu incelemek için plot residuals butonunu tıklamamız gerekir. Şekil 7.15 de gösterilen ve elde ettiğimiz bar grafiği bize her veri noktasındaki hata derecesini göstermektedir.

Şekil 7.15 deki grafiği istersek daha değişik şekilde gösterebiliriz. Örneğin, **line plot** opsiyonunu seçersek hata derecesini Şekil 7.16 da gösterildiği gibi çizgi grafiği olarak da görebiliriz.



Şekil 7.15 Polinomun veri noktalarına uyumluluğu



Şekil 7.16 Hatanın çizgi grafiği olarak gösterilmesi

Örnek 7.5

Yukarıdaki örnekteki veri noktalarından üçüncü derece bir polinom geçiriniz. Polinomun grafiğini ve noktalara olan uyumluluğunu gösteriniz.

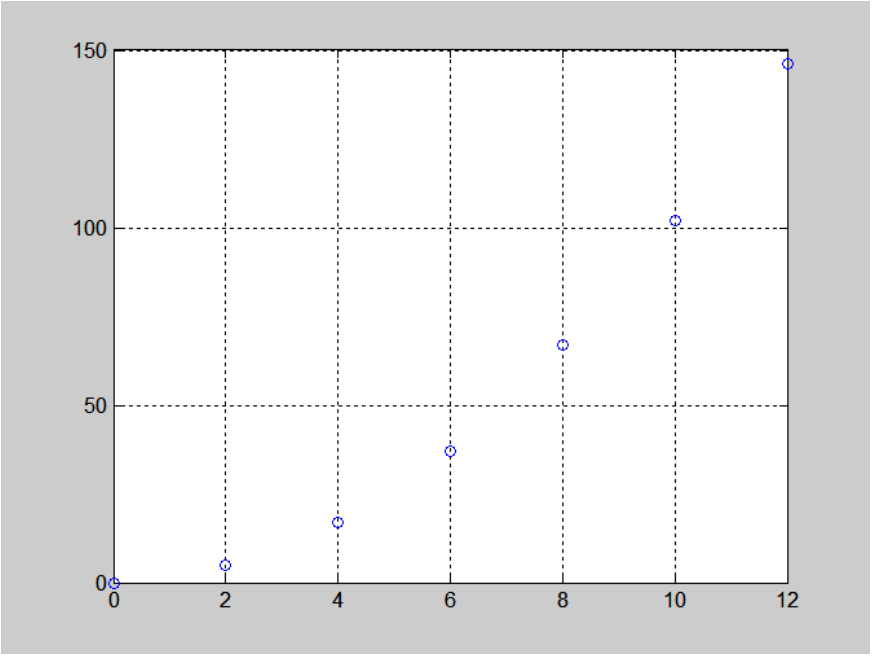
Çözüm 7.5

Daha önce olduğu gibi ilk olarak veri noktalarının grafiğini çizeriz ve bunun için gerekli olan MATLAB komutları şunlardır:

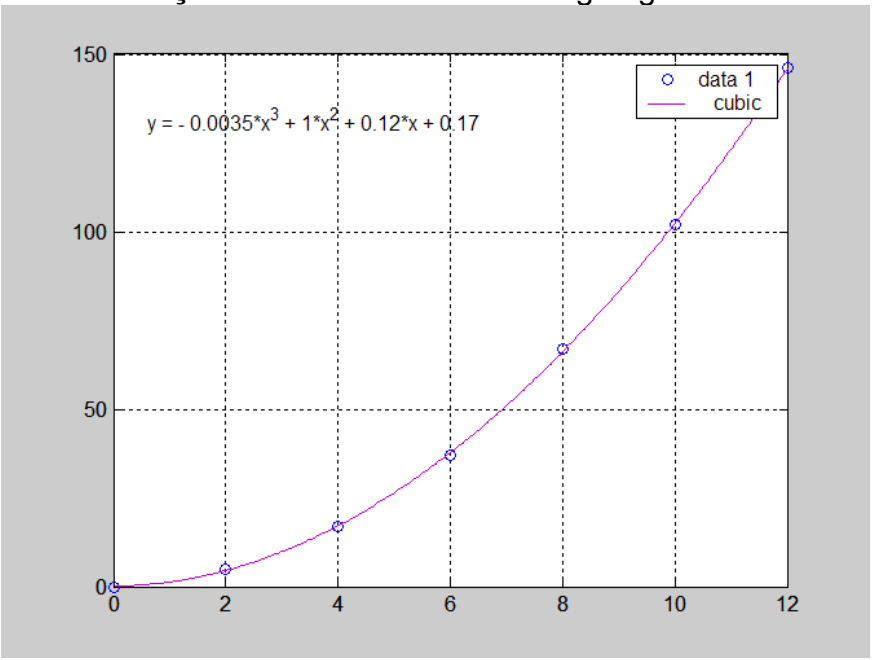
```
>> x = [0 2 4 6 8 10 12];
>> y = [0 5 17 37 67 102 146];
>> plot(x,y,'o'); grid on
```

Grafik Şekil 7.17 de gösterilmiştir. Şimdi Basic Fitting opsiyonunu seçerek veri noktalarımızdan üçüncü derecede (cubic) bir polinom geçiririz. Elde edilen polinomun grafiği ve denklemi Şekil 7.18 de gösterilmiştir. Polinomun veri

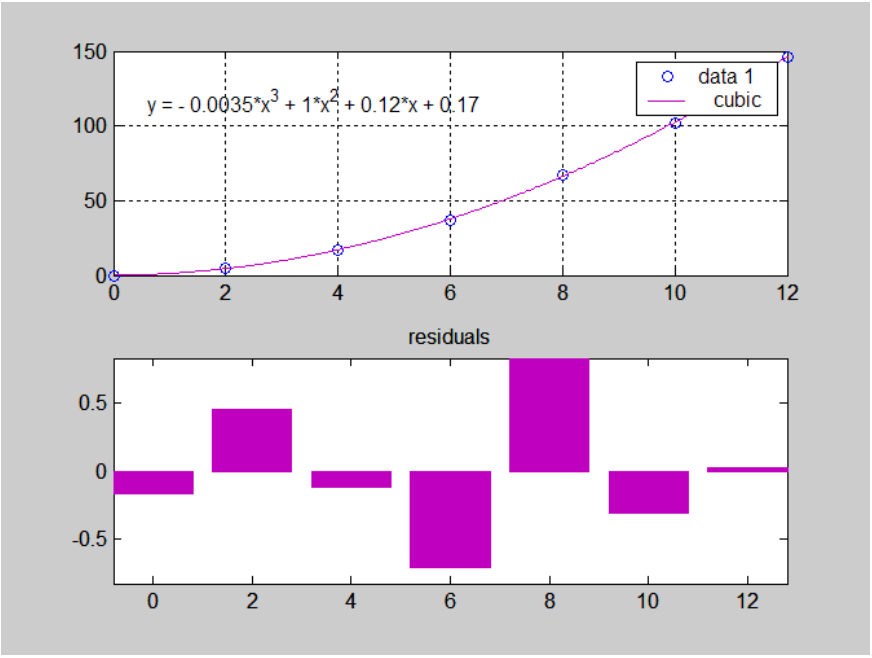
noktalarına olan uyumluluğu ise şekil 7.19 da gösterilmiştir.



Şekil 7.17 Veri noktalarının grafiği



Şekil 7.18 Veri noktalarından geçen üçüncü

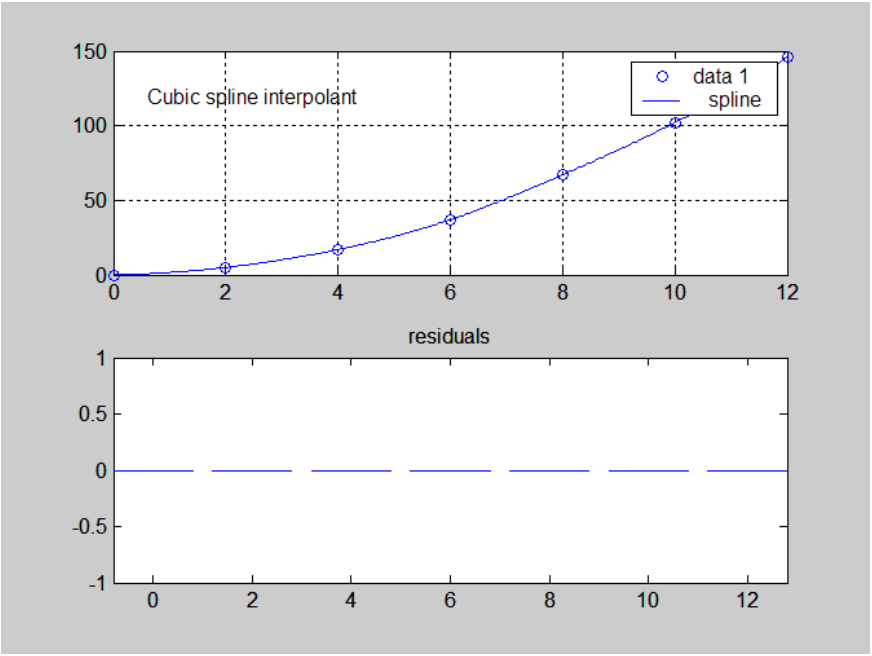


Şekil 7.19 polinomun veri noktalarına uyumluluğu

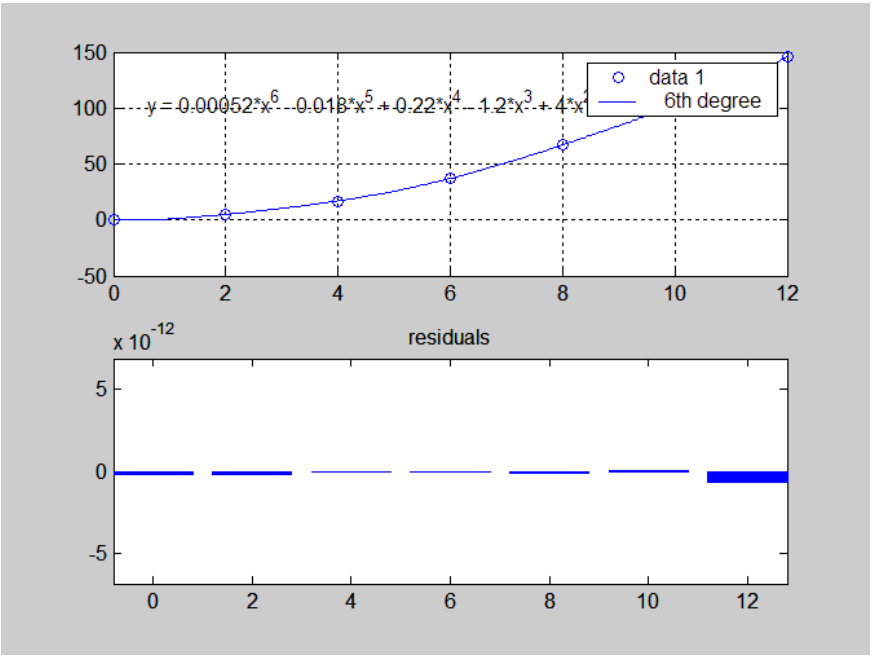
Bu örnekte elde ettiğimiz polinom veri noktalarına daha uygundur. Polinomun denklemi ise şu şekilde verilmiştir:

$$y = -0.0035x^3 + x^2 + 0.12x + 0.17$$

Şekil 7.20 de ve Şekil 7.21 de görüldüğü gibi veri noktalarımıza spline tipi bir polinom veya altıncı derece bir polinom geçirirsek hata tamamıyle ortadan kalkmış olur.



Şekil 7.20 Veri noktalarımızdan spline geçirmek



Şekil 7.21 Veri noktalarımızdan 6. derecede polinom geçirmek

7.3 Veri Noktalarının İstatiksel Analizi

Birçok durumlarda herhangibir deneyden elde ettiğimiz verilerin istatiksel analizini yapmak isteriz. Örneğin, verilerin ortalaması ve standard sapması en çok istediğimiz istatiksel bilgiler arasındadır.

MATLAB'ı kullanarak verilerin istatiksel analizini çok kolaylıkla elde ederiz. Bunun için, yine ilk olarak verilerin grafiğini çizeriz ve **Tools** menüsünden **Data Statistics** opsiyonunu seçeriz. Karşımıza verilerin istatiksel analizini gösteren bir tablo çıkacaktır. Aşağıdaki örnekte bu noktalar açıklanmıştır.

Örnek 7.6

Örnek 7.5 deki verilerin istatiksel analizini yapınız.

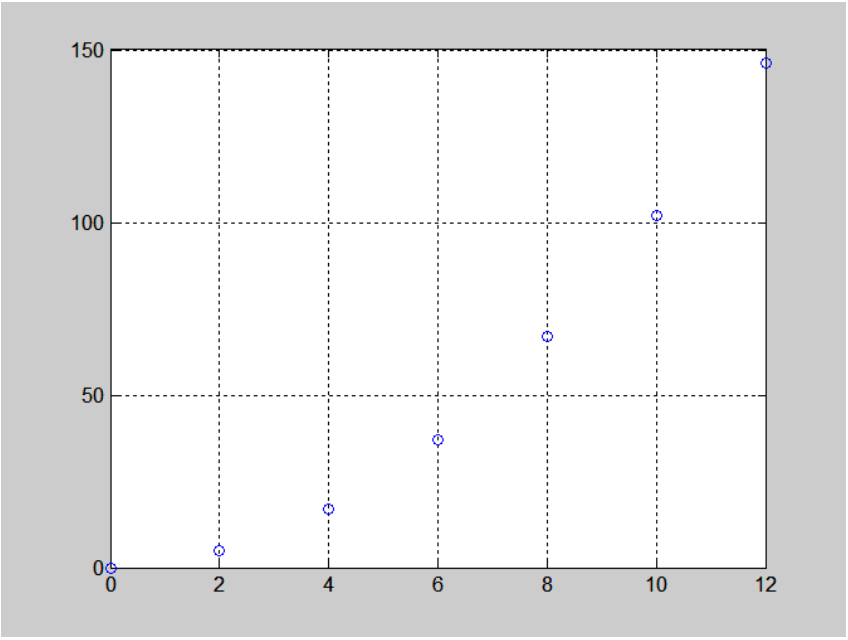
Çözüm 7.6

İlk olarak, Şekil 7.22 de görüleceği gibi verilerin grafiğini çizeriz:

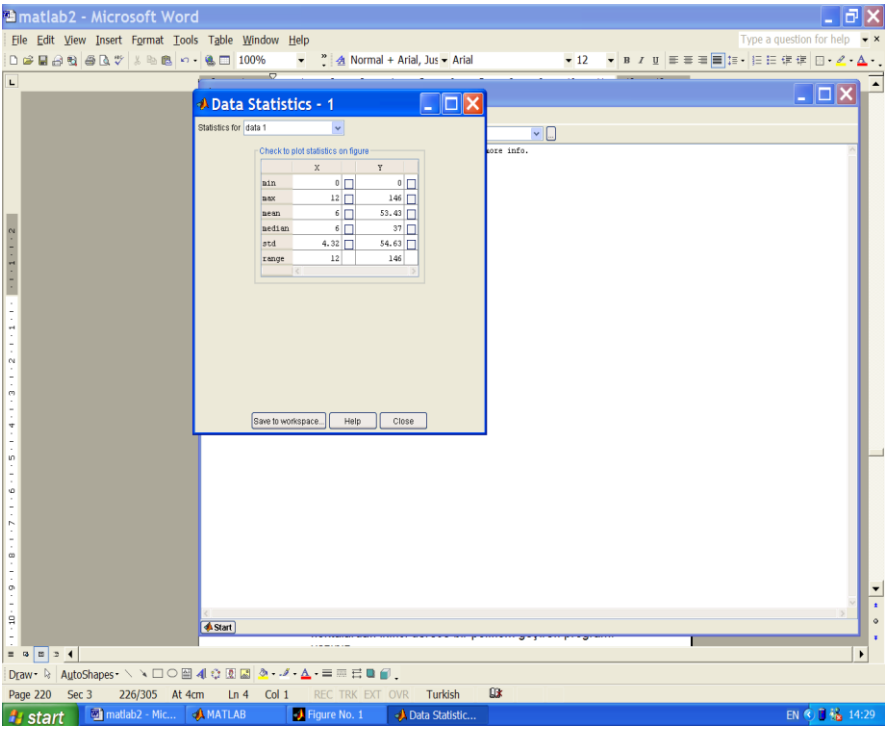
```
>> x = [0 2 4 6 8 10 12];
>> y = [0 5 17 37 67 102 146];
>> plot(x,y,'o'); grid on
```

Daha sonra grafik formundan **Tools** menüsünü ve **Data Statistics** opsiyonunu seçeriz. Önümüze Şekil 7.23 de gösterilen tablo çıkacaktır. Bu tablodan şu özeti yapabiliriz:

	X	**Y**
Ortalama	6	53.43
Standard sapma	4.32	54.63
Medyan	6	37
Minimum	0	0
Maximum	12	146



Şekil 7.22 Verilerin grafiği



Şekil 7.23 Verilerin istatiksel analizi

7.4 Sorular

1. Laboratuvarda yapılan bir deneyde X ve Y değişkenleri arasında şu bağıntı elde edilmiştir:

X:	0	1	2	3	4	5	6
Y:	6	9	5	12	15	18	24

   Bu noktalardan geçecek dördüncü derece polinomunu bulunuz.

2. X ve Y değerlerini klavyeden okuyan ve bu noktalardan ikinci derece bir polinom geçiren programı yazınız.

3. X ve Y değerlerini ve bu noktalardan geçmesi istenen polinomun derecesini klavyeden okuyunuz ve polinomun katsayılarını hesaplayınız.

4. Aşağıdaki noktalardan üçüncü derece bir polinom geçiriniz ve polinomun katsayılarını hesaplayınız:

X: 0	1	2	3	4	5
Y: 0	1	4	9	18	29

5. Soru 4 de elde ettiğiniz polinomun ve veri noktalarının grafiğini çiziniz. Üçüncü derece polinom verilen noktalara uygunmu ?

6. 1 den 5 e kadar olan soruları MATLAB'ın Basic Fitting opsiyonunu kullanarak tekrar yapınız ve doğruluklarını kontrol ediniz.

8
İNTERPOLASYON

Bazı deneylerde almış olduğumuz veriler doğru olmasına rağmen herhangi iki veri noktası arasındaki değerin ne olduğunu öğrenmek isteriz. Örneğin, bir saniye aralıklarla bir dakika içerisinde 60 tane veri aldığımızı kabul edelim. Eğer örneğin 2.5 saniyede olan veri değerini öğrenmek istersek bunun bir çözüm yolu 7.nci bölümde de gördüğümüz gibi ilk olarak noktalardan geçen eğrinin denklemini bulup bu eğrinin 2.5 saniyedeki değerini hesaplayabiliriz. Bir diğer metod ise interpolasyon tekniği kullanarak verilen iki nokta arasında herhangibir yerde verinin değerini hesaplamaktır.

İnterpolasyon genel olarak kompleks ve matematiksel bir işlemdir. Fakat MATLAB'ı kullanarak çok kolaylıkla interpolasyon yapabiliriz.

İnterpolasyon tek boyutlu veya çok boyutlu olabilmektedir. Bu bölümde sadece tek boyutlu interpolasyon metodlarını göreceğiz. MATLAB'da tek boyutlu interpolasyon için interp1 fonksiyonunu kullanabiliriz. Bu fonksiyonun kullanımı ve çeşitli örnekler aşağıdaki bölümde verilmiştir.

8.1 interp1 Fonksiyonu

interp1 fonksiyonu verilen noktalar arasında interpolasyon yapıp bu noktalar arasında herhangibir yerde istenilen değeri verir. Bu fonksiyonun kullanımı şu şekildedir:

```
ai = interp1(x, y, xi)
```
veya
```
ai = interp1(x, y, xi, 'metod')
```

burada x ve y veri değerleri, xi ise y'nin değerlerini bulmak istediğimiz x noktalarıdır. interp1 fonksiyonunda ayrıca kullanmak istediğimiz metodu da belirtebiliriz. Şu metodlar geçerlidir:

nearest:	en yakın veri interpolasyonu
linear:	lineer interpolasyon
spline:	üçüncü derece 'spline' interpolasyon
cubic:	üçüncü derece (küp) interpolasyon

Şimdi bu değişik interpolasyon metodlarının kullanımlarını örneklerle inceleyelim.

8.1.1 nearest Metodu İle İnterpolasyon

Bu metod istenilen noktaya en yakın olan veri noktasını verir.

Örnek 8.1

Laboratuvarda yapılmış olan bir deneyde X ve Y değerleri arasında aşağıdaki bağıntı elde edilmiştir:

X:	-3	-2	-1	0	1	2	3
Y:	9	4	1	0	1	4	9

'nearest' interpolasyon metodunu kullanarak x = 0.5 ve x = 1.6 olduğunda y nin değerlerini bulunuz.

Çözüm 8.1

Gerekli olan MATLAB komutları aşağıda verilmiştir:

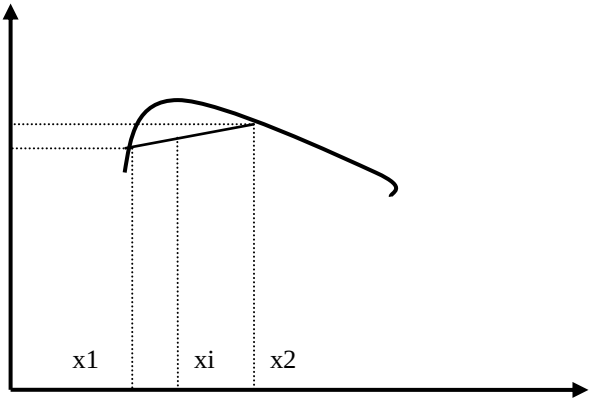
```
>> x = [-3 -2 -1 0 1 2 3];
>> y = [9 4 1 0 1 4 9];
>> z = interp1(x,y,[0.5 1.6],'nearest')
```

```
z =

1    4

>>
```

Çözüm olarak, x = 0.5 için y = 1, ve x = 1.6 için ise y = 4 değerleri bulunur. Doğru cevap y = 0.25 ve y = 2.56 dır. İnterpolasyonla bulunan değerler görüleceği gibi oldukça hatalı olup 'nearest' metodunun burada geçersizdir.

8.1.2 linear Metodu İle İnterpolasyon

Bu metod istenilen x değerinin solundaki ve sağındaki veri noktalarını alıp bu noktalardan bir doğru geçirir ve bu doğrunun istenilen x değerine karşılık gelen y değerini verir. Şekil 8.1 linear interpolasyon metodunun nasıl çalıştığını göstermektedir. Linear metodu sadece birinci derece bir polinoma uyan veri noktaları için tam doğru cevabı vermektedir.



Şekil 8.1 linear interpolasyon metodu
(xi, y değerini bulmak istediğimiz noktadır)

Örnek 8.2

Yukarıdaki örnekteki X ve Y değerlerini göz önünde

bulundurarak x = 0.5 ve x = 1.6 değerlerindeki y değerlerini linear interpolasyon kullanarak hesaplayınız.

Çözüm 8.2

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> x=[-3 -2 -1 0 1 2 3];
>> y = [9 4 1 0 1 4 9];
>> z=interp1(x,y,[0.5 1.6],'linear')

z =

0.5000    2.8000

>>
```

linear interpolasyonu kullanarak bulmuş olduğumuz değerler doğru cevaba daha yakın olmasına rağmen hata yine büyüktür.

8.1.3 spline Metodu İle İnterpolasyon

Bu metod her iki veri noktasından geçen üçüncü derece bir polinom bulur ve bu polinomu kullanarak istenilen noktadaki y değerini hesaplar.

Örnek 8.3

Yukarıdaki örnekteki X ve Y değerlerini göz önünde bulundurarak x = 0.5 ve x = 1.6 değerlerindeki y değerlerini spline interpolasyon kullanarak hesaplayınız.

Çözüm 8.3

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> x=[-3 -2 -1 0 1 2 3];
>> y = [9 4 1 0 1 4 9];
```

```
>> z=interp1(x,y,[0.5 1.6],'spline')

z =

0.2500    2.5600

>>
```

Burada, spline interpolasyonu kullanarak bulmuş olduğumuz değerler tam doğru cevabı vermektedirler.

8.1.4 cubic Metodu İle İnterpolasyon

Bu metod ile veri noktalarından geçen üçüncü derece bir polinom bulunur ve bu polinom kullanılarak istenilen noktadaki y değeri hesaplanır.

Örnek 8.4

Yukarıdaki örnekteki X ve Y değerlerini göz önünde bulundurarak x = 0.5 ve x = 1.6 değerlerindeki y değerlerini cubic interpolasyon kullanarak hesaplayınız.

Çözüm 8.4

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> x = [-3 -2 -1 0 1 2 3];
>> y = [9 4 1 0 1 4 9];
>> z=interp1(x,y,[0.5 1.6],'cubic')

z =

0.3125    2.5480

>>
```

Burada da cevap hatalı olarak verilmiştir.

Örnek 8.5

Laboratuvarda yapılan bir deneyde X ve Y değişkenleri arasında aşağıdaki bağıntı elde edilmiştir:

X:	-3	-2	-1	0	1	2	3
Y:	27	8	1	0	1	8	27

'nearest', 'linear', 'spline' ve 'cubic' metodlarını kullanarak x = 0.1, x = 0.2, x = 0.3 ve x = 0.4 olduğunda y değişkeninin değerlerini bulunuz. Bulmuş olduğunuz değerleri y nin doğru değerleri ile mukayese ediniz.

Çözüm 8.5

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> x = [-3 -2 -1 0 1 2 3];
>> y = [-27 -8 -1 0 1 8 27];
>> z = interp1(x,y,[0.1 0.2 0.3 0.4],'nearest')

z =

0    0    0    0

>> z = interp1(x,y,[0.1 0.2 0.3 0.4],'linear')

z =

0.1000    0.2000    0.3000    0.4000

>> z = interp1(x,y,[0.1 0.2 0.3 0.4],'spline')

z =

0.0010    0.0080    0.0270    0.0640

>> z = interp1(x,y,[0.1 0.2 0.3 0.4],'cubic')

z =

0.0933    0.1760    0.2527    0.3280
```

```
>>
```

Doğru cevap:

```
x = 0.1     y = 0.001
x = 0.2     y = 0.008
x = 0.3     y = 0.027
x = 0.4     y = 0.064
```

Bu örnekten de göreceğimiz gibi, ‘spline’ metodu tam doğru cevabı vermiştir.

Örnek 8.6

Bir MATLAB programı yazarak klavyeden X ve buna karşılık gelen Y verilerini okuyunuz. Daha sonra verilen bir x değerine karşılık gelen y değerini ‘spline’ interpolasyon metodunu kullanarak hesaplayınız. Programınızı *interpolasyon.m* isimli bir dosyada saklayınız.

Çözüm 8.7

İstenilen program listesi aşağıda verilmiştir. Programın başında X ve Y verileri okunur. Daha sonra istenilen x değeri klavyeden girilir. Program ‘spline’ interpolasyon metodunu kullanarak y değerini bulur ve fprintf fonksiyonunu kullanarak ekranda gösterir.

```
%
% "SPLINE" METODUNU KULLANARAK INTERPOLASYON
% ==========================================
%
% Bu programda X ve buna karsilik gelen Y degerleri
% klavyeden okunur. Daha sonra istenilen x degeri
% okunur ve bu x degerine karsilik gelen y degeri
% spline interpolasyon metodu kullanilarak hesaplanir.
%
disp('INTERPOLASYON PROGRAMI');
disp('======================');
disp('X - Verilerini giriniz...');
```

```
x = input('')
disp('Y - Verilerini giriniz...');
y = input('')
xint = input('Istenilen x degerini giriniz: ');
z = interp1(x,y,xint,'spline');
fprintf('y degeri = %g\n',z)
```

Programın çalışmasına bir örnek aşağıda verilmiştir. Bu örnekte X = [1 2 3 4] ve Y = [1 4 9 16] olup x = 2.5 olduğunda y nin değerini bulmak isteriz. Aşağıdaki örnekte, ayırt edilmeleri için klavyeden girilenlerin altları çizilmiştir:

```
>> interpolasyon
INTERPOLASYON PROGRAMI
======================
X - Verilerini giriniz...
[1 2 3 4]

x =

1    2    3    4

Y - Verilerini giriniz...
[1 4 9 16]

y =

1    4    9   16

Istenilen x degerini giriniz: 2.5
y degeri = 6.25
>>
```

Örnek 8.8

Bir deneyde X ve Y değişkenleri arasında aşağıdaki bağıntı elde edilmiştir:

X:	5	10	20	30	40
Y:	4000	12000	24000	40000	62000

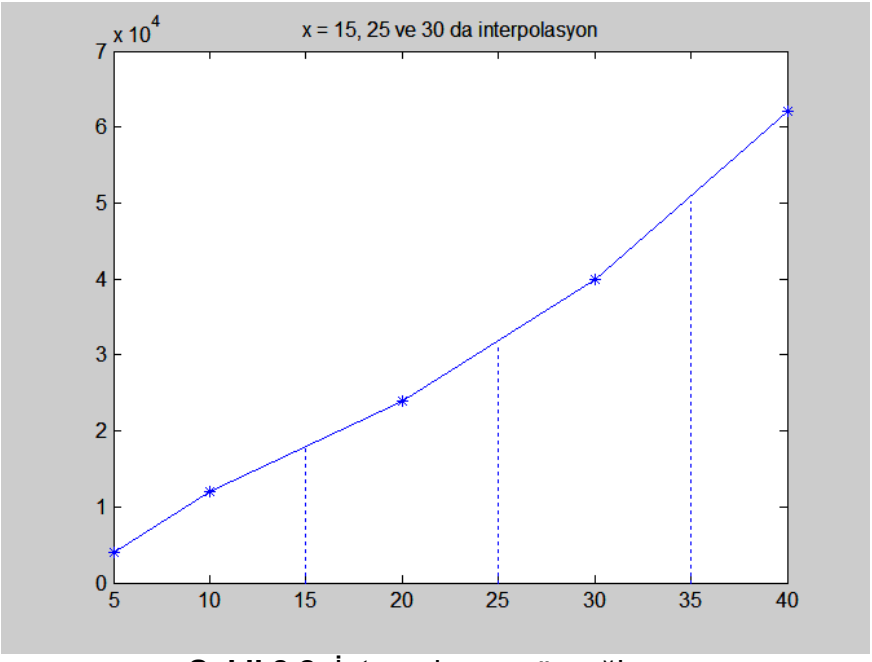
x = 15, 25 ve 35 olduğunda ‘spline’ metodunu kullanarak y değerlerini hesaplayınız. X-Y grafiğini çizerek interpolasyon yapılan noktaları belirtiniz.

Çözüm 8.8

Gerekli olan MATLAB komutları aşağıda verilmiştir. İlk olarak X ve Y değerleri girildikten sonra X-Y grafiği çizilir. Daha sonra ‘spline’ interpolasyon metodu kullanılarak verilen x = 15, x = 25 ve x = 35 değerlerine karşılık gelen y değerleri hesaplanır. Burada y değerleri 18079, 31224, ve 50276 olarak hesaplanmıştırlar. Daha sonra interpolasyon yapılan x noktaları grafikte gösterilmiştir:

```
>> x = [5 10 20 30 40];
>> y = [4000 12000 24000 40000 62000];
>> plot(x,y,'*-')
>> hold on
>> xint = [15 25 35];
>> z = interp1(x,y,xint,'spline')

z =

1.0e+004 *

         1.8079    3.1224    5.0276

>> plot([15 15],[0 z(1)],'b:')
>> plot([25 25],[0 z(2)],'b:')
>> plot([35 35],[0 z(3)],'b:')
>> title('x = 15, 25 ve 30 da interpolasyon')
>>
```

Yukarıdaki programın çizmiş olduğu grafik Şekil 8.2 de gösterilmiştir. Grafikte x = 15, x = 25 ve x = 35 noktaları dikey doğrularla belirtilmiştir.



Şekil 8.2 İnterpolasyon örneği

8.2 Sorular

1. Bir deney sonucu aşağıdaki X ve Y verileri elde edilmiştir:

X:	2	4	6	8	10
Y:	6	12	18	24	30

   x = 3 ve x = 5 noktalarına karşılık gelen y değerlerini 'linear' interpolasyon metodunu kullanarak hesaplayınız. Cevabınızın doğruluğunu kontrol ediniz.

2. Yukarıdaki örneği 'spline' interpolasyon metodunu

kullanarak tekrar ediniz. Cevabınızda bir değişiklik oldumu ? açıklayınız.

3. İnterpolasyon yapmak için bir MATLAB programı yazınız. Programınızda X ve Y verileri klavyeden girilecektir. Ayni zamanda kullanıcının interpolasyon metodunu 'nearest', 'linear', 'spline', ve 'cubic' olarak seçme imkanı olmalıdır. İnterpolasyon yapmak istediğiniz x değerleri de klavyeden girilecektir. Programınız y değerlerini hesaplayıp ekranda gösterecektir.

4. Aşağıdaki verileri kullanarak Soru 3 de yazmış olduğunuz programın doğruluğunu kontrol ediniz:

X:	0	1	2	3	4	5
Y:	0	1	8	27	64	125

5. $f(x) = 2x^2 + 4x - 5$ fonksiyonunun değerini x = 0 ve x = 10 arasında, 0.5 aralıklarla hesaplayınız. Daha sonra, x = 0.1 ve x = 2.7 olduğunda fonksiyonun değerlerini 'spline' interpolasyon metodunu kullanarak hesaplayınız.

9
MATLAB'DA DOSYALAMAK

Şimdiye kadar yapmış olduğumuz örneklerde MATLAB ile giriş-çıkış yaparken giriş olarak klavyeyi ve çıkış olarak ise ekranı kullandık. Bu bölümde bir dosyanın nasıl yaratıldığına, dosyaya nasıl bilgi yazıldığına, dosyadan nasıl bilgi okunduğuna ve dısyanın nasıl kapatıldığına bakacağız. Bir dosyadan bilgi okumak için bu dosyanın MATLAB tarafından yaradılmış olması gerekmez. Örneğin Excel veya başka bir program tarafından yaradılmış olan herhangibir dosyayı MATLAB'da açıp okuyabiliriz. Ayni şekilde, MATLAB tarafından yaradılmış olan bir dosya başka bir program tarafından açılıp okunabilmektedir.

MATLAB'ın desteklediği çok değişik dosya çeşidi bulunmaktadır. Tablo 9.1 de MATLAB tarafından kullanılabilecek olan dosya uzantıları verilmiştir.

Bu bölümde text dosyaların nasıl yaradıldıklarını, ve bu dosyalara nasıl bilgi yazılıp okunabileceğine bakacağız.

9.1 Dosya Açıp Kapatmak

MATLAB'ı kullanarak bir dosyanın içindekilerini okumak için, veya bir dosyaya yazmak için ilk olarak istenilen dosyayı açmamız gerekir. Bu işlem ise fopen komutu ile yapılır ve bu komutun genel formatı şu şekildedir:

```
f = fopen(dosya-adı, açılış-kodu)
```

Burada dosya-adı açmak istediğimiz dosyanın tam adıdır. Eğer dosyada olanları okumak istiyorsak dosyanın disk üzerinde önceden bulunması gerekir. Dosya açılış-kodu ise dosyayı ne maksatla açmak istediğimizi belirtir ve Tablo 9.2

de geçerli olan açılış-kodları verilmiştir.

Tablo 9.1 MATLAB Dosya Çeşitleri

Dosya uzantısı	**Dosya çeşidi**
.MAT	MATLAB çalışma alanı
.CSV	Virgül ile ayrılmış sayılar
.DLM	Belirli bir karakter ile ayrılmış text
.TAB	Tab karakteri ile ayrılmiş text
.XLS	Excel dosyası
.WK1	Lotus 123 dosyası
.AVI	Video dosyası
.TIFF	TIFF resim dosyası
.PNG	PNG resim dosyası
.BMP	BMP resim dosyası
.GIF	GIF resim dosyası
.PCX	PCX resim dosyası
.WAV	WAV ses dosyası

Tablo 9.2 Dosya açılış-kodları

Açılış-kodu	**Tanımı**
‘r’	Dosyayı okumak için aç. Dosyanın önceden disk üzerinde olması gerekir, aksi halde hata verir.
‘r+’	Dosyayı okumak ve yazmak için aç. Dosyanın önceden disk üzerinde olması gerekir, aksi halde hata verir.
‘w’	Dosyaya yazmak için aç. Dosya önceden disk üzerine varsa silinir ve tekrar açılır, dosya önceden yoksa yeni dosya açılır.
‘w+’	Dosyayı okumak ve yazmak için aç. Dosya önceden disk üzerine varsa silinir ve tekrar açılır, dosya önceden yoksa yeni dosya açılır.
‘a’	Dosyaya yazmak için aç. Dosya önceden disk üzerinde varsa, yeni bilgileri mevcut dosyanın sonuna ilave et. Dosya önceden yoksa, yeni dosya aç.

‘a+’	Dosyayı okumak ve yazmak için aç. Dosya önceden varsa, yeni bilgileri dosyanın sonuna ilave et. Dosya önceden yoksa, yeni dosya aç.

MATLAB’da ayni zamanda istediğimiz kadar dosya açabiliriz. Dosya açarken sol tarafta kullanılan değişken (yukarıdaki örnekte f) dosyanın indeksi olarak bilinir ve bu değişken dosya açılışında hata olup olmadığını belirtir. Bu değişken -1 olduğunda hata olduğunu belirtir. 3 veya üzerinde olduğunda ise dosyanın hatasız açıldığını belirtir.

Açık olan herhangibir dosyayı kapatmak için fclose komutu kullanılır. Bu komutun formayı şu şekildedir:

```
s = fclose(f)
```

Burada f dosyayı açarken kullanmış olduğumuz dosya indeksidir. Eğer dosya hatasız olarak kapatılmışsa s değişkeni 0 olur. Dosya kapatırken hata olmuşsa s değişkeni -1 değerini alır.

Örnek 9.1

Bulunduğunuz folderin ismini ve bu folderdeki dosya listesini ekranda gösteriniz. Daha sonra, “dosya1.txt”, “dosya2.txt”, ve “dosya3.txt” isimli 3 tane dosya yaradınız ve dosyaların hatasız yaradıldıklarını kontrol ediniz. Dosyaları kapatınız ve folderinizdeki dosyaların isimlerini yeniden listeleyiniz.

Çözüm 9.1

pwd komutu bulunduğumuz folderin ismini verir:

```
>> pwd

ans =

C:\MATLAB6p5\work
```

```
>>
```

folderimizdeki dosyaları görmek için **dir** komutunu kullanırız:

```
>> dir

.  ..

>>
```

folderimiz boştur. Şimdi istenilen 3 dosyayı yaratalım ve hata olup olmadığına bakalım:

```
>> f1 = fopen('dosya1.txt','w')

f1 =

        3

>> f2 = fopen('dosya2.txt','w')

f2 =

        4

>> f3 = fopen('dosya3.txt','w')

f3 =

        5

>>
```

Burada her 3 dosyada hatasız olarak açılmıştırlar çünkü dosya açılırken sol taraftaki değişken 3 ün üzerinde bir sayı vermiştir.

Şimdi açmış olduğumuz dosyaları kapatalım:

```
>> s1 = fclose(f1)

s1 =
```

```
        0

>> s2 = fclose(f2)

s2 =

        0


>> s3 = fclose(f3)

s3 =

        0

>>
```

Burada her 3 dosya da hatasız olarak kapatılmıştır.

Şimdi folderimizdeki dosya listesine bakarsak yaratmiş olduğumuz yeni dosyaların orda olduklarını görürüz:

```
>> dir
.  ..          dosya1.txt      dosya2.txt      dosya3.txt
>>
```

9.2 Dosyaya Yazmak

Açılmış olan bir dosyaya yazmak için daha önce de gördüğümüz fprintf komutunu kullanırız. Bu komut içerisinde ilk olarak dosyanın indeksini ve daha sonra da dosyaya yazmak istediklerimizi belirtiriz.

Dosyaya yazmak için örnekler aşağıda verilmiştir.

Örnek 9.2

C folderinde *dosyam.txt* isimli bir dosya yaratınız ve bu

dosyaya aşağıdaki cümleleri yazınız:

> Dosya yaratirken ilk olarak fopen komutunu kullanarak dosyayı acmamız gerekir. Daha sonra istediklerimizi bu dosyaya yazabiliriz.

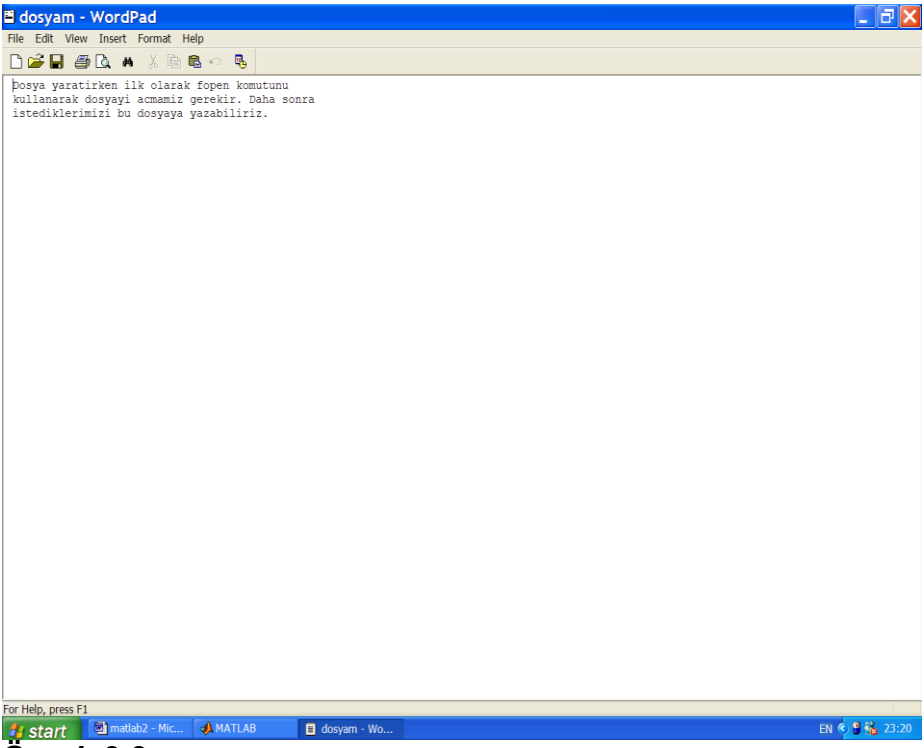
Dosyayı kapatınız ve Wordpad'ı kullanarak dosyanın içindekileri gösteriniz.

Çözüm 9.2

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> f = fopen('c:\dosyam.txt','w');
>> fprintf(f,'Dosya yaratirken ilk olarak fopen komutunu\n');
>> fprintf(f,'kullanarak dosyayi acmamiz gerekir. Daha sonra\n');
>> fprintf(f,'istediklerimizi bu dosyaya yazabiliriz.\n');
>> s = fclose(f);
>>
```

Wordpad'ı kullanarak dosyamızı açarsak şunları görürüz:



Örnek 9.3

1 den 10 a kadar olan sayıların karelerini alıp C: folderinde *kareler.txt* isimli bir dosyada saklayınız

Çözüm 9.3

Gerekli olan MATLAB komutları aşağıda verilmiştir. İlk olarak dosya açılmış ve dosyaya başlık yazılmıştır. Daha sonra 1 den 10 a kadar bir döngü içerisinde sayıların kareleri hesaplanıp dosyaya yazılmıştır:

```
>> f = fopen('c:\kareler.txt','w');
>> fprintf(f,'1 den 10 a kadar olan sayilarin kareleri\n');
>> for j = 1:10
        fprintf(f,'%d    %d\n', j, j*j);
end
>> fclose(f)
```

Program çalışınca C: folderinde kalerer.txt isimli bir dosya yaratılır ve dosya içerisine şunlar yazılır:

```
1 den 10 a kadar olan sayilarin kareleri
1     1
2     4
3     9
4     16
5     25
6     36
7     49
8     64
9     81
10     100
```

Örnek 9.4

0 dan 30 dereceye kadar olan açıların sinüslerini bir tablo şeklinde sinus.txt isimli bir dosyaya yazan MATLAB programını yazınız.

Çözüm 9.4

Gerekli olan MATLAB komutları aşağıda verilmiştir. Dosya açıldıktan sonra 0 dan 30 dereceye kadar olan açıların sinüsleri hesaplanıp bir tablo şeklinde dosyaya yazılmıştır:

```
>> f = fopen('c:\sinus.txt','w');
>> for j = 0:30
        s = j*pi/180.0;
        fprintf(f,'%g   %g\n', j, sin(s));
end
>> fclose(f);
>>
```

Program çalışınca aşağıdakiler sinus.txt dosyasına yazılır:

```
0   0
1   0.0174524
2   0.0348995
3   0.052336
```

```
4   0.0697565
5   0.0871557
6   0.104528
7   0.121869
8   0.139173
9   0.156434
10   0.173648
11   0.190809
12   0.207912
13   0.224951
14   0.241922
15   0.258819
16   0.275637
17   0.292372
18   0.309017
19   0.325568
20   0.34202
21   0.358368
22   0.374607
23   0.390731
24   0.406737
25   0.422618
26   0.438371
27   0.45399
28   0.469472
29   0.48481
30   0.5
```

9.3 Dosyayı Okumak

Bir dosyadan okuyabilmek için o dosyanın disk üzerinde bulunması ve dosya içerisinde bilgi olması gerekmektedir. Dosyadan okumak için **fscanf** komutunu kullanırız. Bu komut içerisinde okuyacağımız dosyanın indeksini ve okunan bilginin hangi değişkenlerde saklanacağını belirtmemiz gerekir.

Herhangibir dosyadan bilgi okurken dosyanın sonuna geldiğimizi **feof** fonksiyonunu kullanarak anlayabiliriz. Dosyanın sonuna geldiğimizde **feof** fonksiyonu 1 olur.

Dosyadan okumak için örnekler aşağıda verilmiştir:

Örnek 9.5

Yukarıdaki örnekte (Örnek 9.2) *sinus.txt* dosyasını açınız ve dosyadanın ilk 5 satırındakileri okuyup ekranda gösteriniz.

Çözüm 9.5

Gerekli olan MATLAB komutları aşağıda verilmiştir. İlk olarak dosya açılmış ve daha sonra 1 den 5 e kadar olan bir döngü içerisinde dosyadaki sayılar **fscanf** komutu ile okunup **fprint** komutu ile ekrana yazılmıştır:

```
>> f = fopen('c:\sinus.txt','r');
>> for j=1:5
        p = fscanf(f,'%g',1);
        q = fscanf(f,'%g\n',1);
        fprintf('%g   %g\n',p,q)
end
fclose(f);
```

Program çalışınca ekranda aşağıdakiler gösterilir:

```
0   0
1   0.0174524
2   0.0348995
3   0.052336
4   0.0697565
```

Örnek 9.6

Yukarıdaki örneği tekrarlayınız fakat dosyada olan herşeyi ekrana yazınız.

Çözüm 9.6

Gerekli olan MATLAB komutları aşağıda verilmiştir. Burada **foef** fonksiyonunu kullanarak dosyanın sonuna gelip gelmediğimizi kontrol etmekteyiz:

```
>> f = fopen('c:\sinus.txt','r');
>> while ~feof(f)
        p = fscanf(f,'%g',1);
        q = fscanf(f,'%g\n',1);
        fprintf('%g    %g\n',p,q)
end
fclose(f);
```

Program çalışınca aşağıdaki satırlar ekranda gösterilir:

```
0   0
1   0.0174524
2   0.0348995
3   0.052336
4   0.0697565
5   0.0871557
6   0.104528
7   0.121869
8   0.139173
9   0.156434
10   0.173648
11   0.190809
12   0.207912
13   0.224951
14   0.241922
15   0.258819
16   0.275637
17   0.292372
18   0.309017
19   0.325568
20   0.34202
21   0.358368
22   0.374607
23   0.390731
24   0.406737
25   0.422618
```

26 0.438371
27 0.45399
28 0.469472
29 0.48481
30 0.5

Örnek 9.7

c:\test.txt isimli bir dosyada şu satırlar yazılıdır:

Dosyanın birinci satiri
Dosyanin ikinci satiri
Dosyanın ucuncu satiri

Dosyayı açıp satırları okuyunuz ve ekranda gösteriniz.

Çözüm 9.7

Gerekli olan MATLAB programı aşağıda verilmiştir. Burada **fgetl** isimli bir komut kullanılmıştır. Bu komut dosyadan bir satır okur ve komutun sol tarafındaki değişkende saklar. Program dosyadaki satırları bir bir okur ve ekranda gösterir:

```
>> f = fopen('c:\test.txt','r');
>> while ~feof(f)
        satir = fgetl(f);
        fprintf('%s\n',satir)
end
```

Program çalışınca ekranda şu satırları görürüz:

```
Dosyanin birinci satiri
Dosyanin ikinci satiri
Dosyanin ucuncu satiri
```

9.4 textread Fonksiyonu

textread fonksiyonu herhangibir dosyadan belirli bir formatta bilgi okumak için kullanılır. Bu fonksiyonun genel kullanım şekli şöyledir:

[A,B,C,…] = textread('dosya-adı', format)

Burada verilen dosyadan belirtilen formatta bilgi okunur ve bu bilgi A, B, C gibi değişkenlerde saklanır. textread fonksiyonu dosyanın başından sonuna kadar okumaktadır. Dosyadan okurken %d işaretli tam sayı için, %u tam sayı için, %f kayan nokta sayı için, %c herhangibir karakter okumak için ve %q bir karakter dizisi okumak için kullanılır. Aşağıda birkaç örnek verilmiştir.

Örnek 9.8

C: folderinde bulunan *deney.txt* dosyasında şu sayılar bulunmaktadır:

```
1  1
2  8
3  27
4  64
5  125
```

textread fonksiyonunu kullanarak bu sayıları okuyunuz ve ekranda gösteriniz.

Çözüm 9.8

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> [x,y] = textread('c:\deney.txt','%d %d');
>> [x,y]
```

```
ans =

1     1
2     8
3    27
4    64
5   125

>>
```

Yukarıdaki örnekten de görüleceği gibi *deney.txt* dosyasında bulunan sayılar okunup *x* ve *y* değişkenlerinde saklanmıştırlar. Burada dikkat edilmesi gereken bir nokta *textread* fonksiyonunu kullanırken dosyayı açıp kapamaya gerek olmayışıdır.

textread fonksiyonunun önemli kullanış alanlarından biri bir dosyada bulunan verileri grafih halinde göstermek içindir. Bir örnek aşağıda verilmiştir:

Örnek 9.9

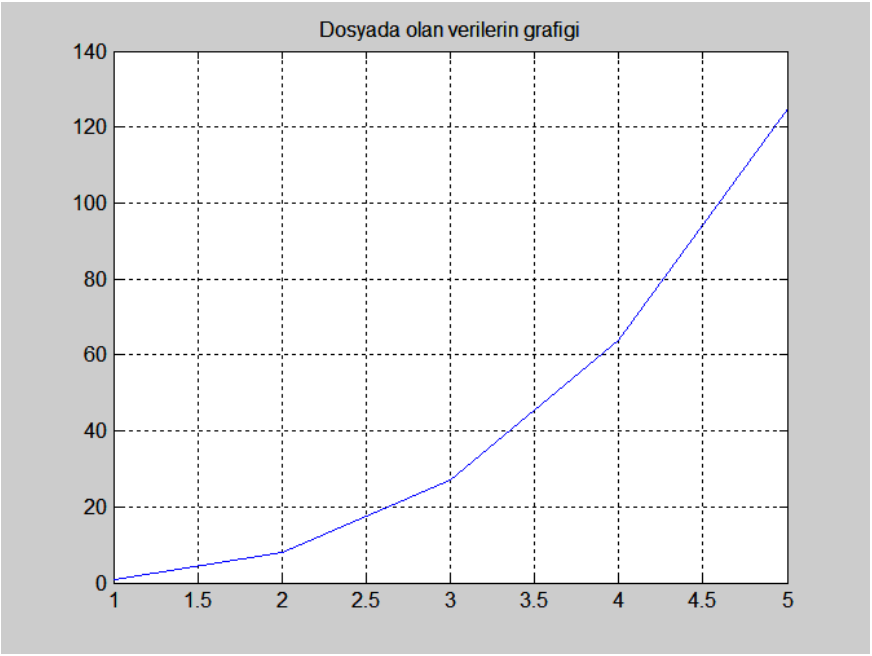
Yukarıdaki örnekteki *deney.txt* dosyasındaki verileri bir grafik olarak gösterecek MATLAB komutlarını yazınız.

Çözüm 9.9

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> [x,y] = textread('c:\deney.txt','%d %d');
>> plot(x,y); title('Dosyada olan verilerin grafigi'); grid on
>>
```

Dosyadaki veriler *x* ve *y* vektörlerine yüklendikten sonra istenilen grafik çizilmiştir. Şekil 9.1 de çizilen grafik gösterilmiştir.



Şekil 9.1 Çizilen grafik

9.5 dlmwrite Fonksiyonu

dlmwrite fonksiyonunun kullanım şekli şöyledir:

dlmwrite(dosya-adı, A, ayırıcı)

Bu fonksiyon A matrisini belirtilen dosyaya yazar ve matriks elemanlarını *ayırıcı* ile ayırır. Normal olarak ayırıcı olarak virgül kullanılır.

Dlmwrite fonksiyonu 'spreadsheet' uygulamalarında sık olarak kullanılan ve virgül ile ayrılan veri formatı ile uyumludur. Bu fonksiyonun kullanımına bir örnek aşağıda verilmiştir:

Örnek 9.10

A matrisinde şu elemanların olduğunu kabul edelim:

1	10	100
2	20	200
3	30	300

Bu matriksin elemanlarını dlmwrite komutunu kullanarak *dlmornek.txt* isimli bir dosyada saklayınız.

Çözüm 9.10

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> A = [1 10 100; 2 20 200; 3 30 300];
>> dlmwrite('c:\dlmornek.txt', A, ',')
>>
```

Bu komutlardan sonra C folderinde dlmornek.txt isimli bir dosya yaratılır ve bu dosyaya aşağıdaki sayılar yazılır (burada sayıların birer virgül ile ayrıldıklarına dikkat ediniz):

```
1,10,100
2,20,200
3,30,300
```

9.6 dlmread Fonksiyonu

dlmread fonksiyonu *dlmwrite* fonksiyonunun tersi olup belirli bir ayırıcı ile ayrılmış olan sayıları bir dosyadan okur ve verilen bir matrisde saklar. Fonksiyonun kullanım şekli şöyledir:

```
A = dlmread(dosya-adı, ayırıcı)
```

dlmread fonksiyonu genellikle ‘spreadsheet’ gibi

programlardan veri okumak için kullanılmaktadır.

Örnek 9.11

Yukarıdaki örnekte yaratılmış olan *dlmornek.txt* dosyasınındaki sayıları okuyup ekranda gösteriniz.

Çözüm 9.11

Gerekli olan MATLAB komutları aşağıda verilmiştir. Burada *dlmread* fonksiyonu kullanılmış ve birer virgül ile ayrılmış olan sayılar okunup ekranda gösterilmiştir:

```
>> b = dlmread('c:\dlmornek.txt',',');
>> b

b =

1   10   100
2   20   200
3   30   300

>>
```

dlmwrite ve dlmread fonksiyonları normal olarak birinci sıra ve birinci sütundan okuyup yazmaktadırlar. Bazı uygulamalarda ilk birkaç sırada başlık olmaktadır. Bu gibi durumlarda bu sıraları atlayıp esas verilerin olduğu sıradan başlamamız gerekmektedir. Bunun için ise dlmwrite ve dlmread fonksiyonlarının aşağıda gösterilmiş olan genel şekillerini kullanmamız gerekmektedir:

dlmwrite(dosya-adı, matris, ayırıcı, ilk-sıra, ilk sütun)

ve

A = dlmread(dosyadı, ayırıcı, ilk-sıra, ilk-sütun)

Burada dikkat edilmesi gereken bir nokta, ilk-satır ve ilk-sğtun değerlerinin 0 dan başlamış olmalarıdır.

Aşağıda bir örnek verilmiştir:

Örnek 9.12

dlmornek.txt isimli bir dosyada aşağıdaki veriler bulunmaktadır:

```
Son elde edilen neticeler
Veriler asagida verilmistir
1,1
2,4
3,9
4,16
5,25
6,36
```

Esas verileri okuyup VERI isimli bir matrise yükleyiniz ve daha sonra bu matrisin içindekilerini ekranda gösteriniz. Burada esas verilerin ikinci sütundan başladığına dikkat ediniz (ilk sütun 0. cı sütundur).

Çözüm 9.12

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> VERI = dlmread('c:\dlmornek.txt',',',2,0);
>> VERI

VERI =

1     1
2     4
3     9
4    16
5    25
6    36

>>
```

Bu örnekte dosyadaki sayılar 2. satırdan (0 dan başlayarak) ve ilk sütundan (0 dan başlayarak) başlayarak okunmuştur.

9.7 EXCEL'den Veri Okumak

EXCEL en yaygın olarak kullanılan 'spreadsheet' programlarından biridir. MATLAB'ın *xlsread* fonksiyonunu kullanarak EXCEL'de olan herhangibir bilgiyi MATLAB'a aktarabiliriz. *xlsread* fonksiyonu son derece kullanışlı olup bu fonksiyon ile EXCEL'in herhangibir sayfasında olan bilgiyi çok kolaylıkla MATLAB'a aktarabiliriz.

xlsread fonksiyonunun kullanış şekli şöyledir:

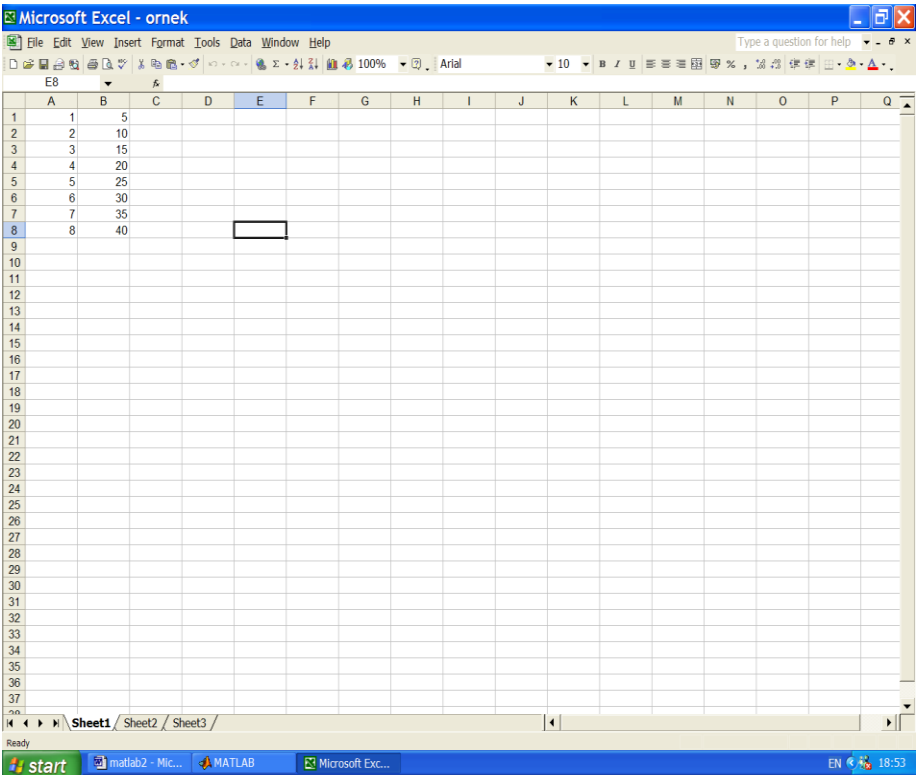
```
xlsread(dosya-adı, sayfa-adı)
```

Burada dosya-adı verilerin bulunduğu EXCEL dosya adı (.xls dosya uzantısı ile) ve sayfa-adı ise verileri almak istediğimiz sayfanın adıdır. *xlsread* fonksiyonunu kullanırken okumak istediğimiz EXCEL dosyasının uzantısının *.xls* olması gerekmektedir, aksi halde bu dosyayı okumamız mümkün olmamaktadır.

xlsread fonksiyonunun kullanımına bir örnek aşağıda verilmiştir.

Örnek 9.13

C: folderinde bulunan *ornek.xls* isimli bir EXCEL dosyasında Şekil 9.2 de olan verilerin olduğunu kabul edelim. Bu verileri MATLAB'da *sayfa1* isimli bir matrise okuyunuz. Daha sonra okunmuş olan verileri ekranda gösteriniz.



Şekil 9.2 *ornek.xls* EXCEL dosyası

Çözüm 9.13

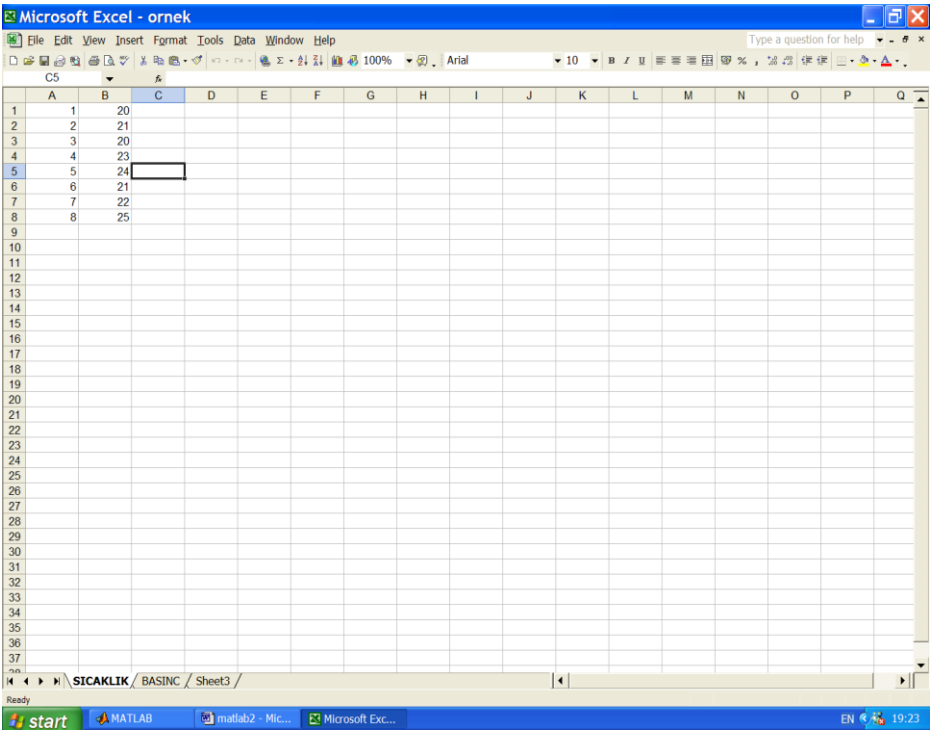
Gerekli olan MATLAB komutları aşağıda verilmiştir. Burada ilk olarak EXCEL dosya adı (c:\ornek) belirtilir ve daha sonra verileri okumak istediğimiz sayfanın ismi (Sheet1) belirtilir. Aşağıdaki örnekte okunan bilgiler *sayfa1* isimli bir matrisde saklanır. Daha sonra okunan bilgiler ekranda gösterilir:

```
>> sayfa1 = xlsread('c:\ornek','Sheet1');
>> sayfa1

sayfa1 =

1     5
2    10
```

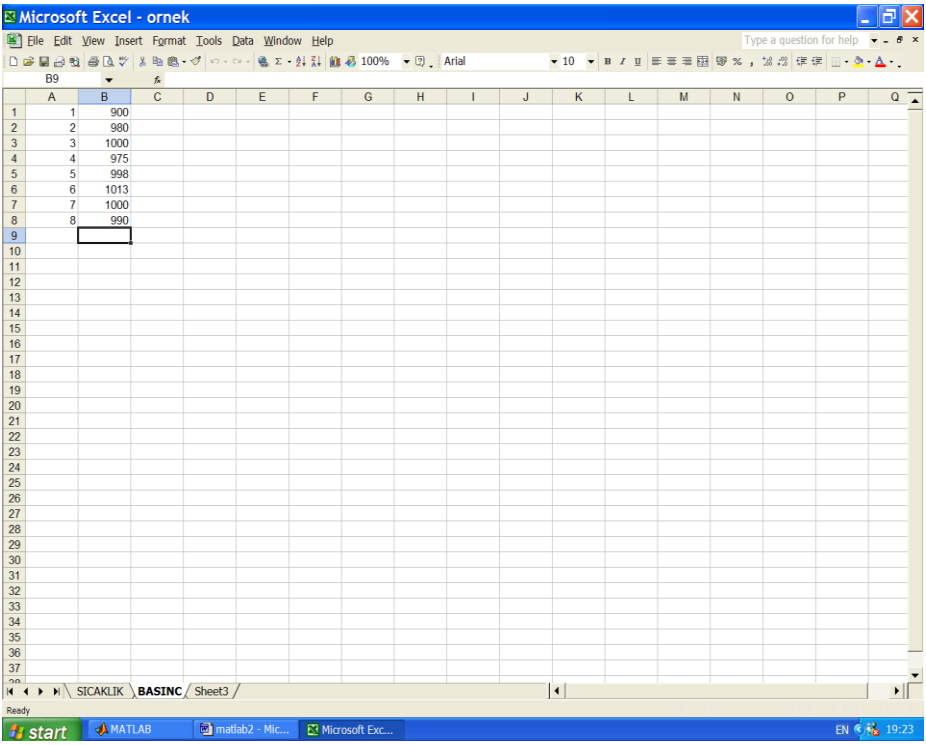
```
3    15
4    20
5    25
6    30
7    35
8    40

>>
```

Örnek 9.14

C: folderinde *ornek.xls* isimli bir EXCEL dosyasında 2 EXCEL sayfası bulunmaktadır. Birinci sayfa SICAKLIK ve ikinci sayfa ise BASINC olarak adlandırılmıştırlar. Bu sayfalardaki veriler Şekil 9.3 ve Şekil 9.4 de verilmiştir. Birinci sayfadaki bilgileri SICAKLIK_BUGUN ve ikinci sayfadaki bilgileri ise BASINC_BUGUN isimli matrislerde saklayacak MATLAB komutlarını yazınız.



Şekil 9.3 SICAKLIK sayfası



Şekil 9.4 BASINC sayfası

Çözüm 9.14

Gerekli olan MATLAB komutları aşağıda verilmiştir. İlk olarak SICAKLIK sayfasındaki veriler okunup SICAKLIK_BUGUN isimli bir matrisde saklanmıştır. Daha sonra BASINC sayfasındaki veriler okunup BASINC_BUGUN isimli bir matrisde saklanmıştır:

```
>> SICAKLIK_BUGUN = xlsread('c:\ornek','SICAKLIK');
>> BASINC_BUGUN = xlsread('c:\ornek','BASINC');
>>
>> SICAKLIK_BUGUN

SICAKLIK_BUGUN =

1    20
```

```
2    21
3    20
4    23
5    24
6    21
7    22
8    25

>> BASINC_BUGUN

BASINC_BUGUN =

1        900
2        980
3       1000
4        975
5        998
6       1013
7       1000
8        990

>>
```

Örnek 9.15

Yukarıdaki EXCEL örneğinde SICAKLIK ve BASINC verilerini okuyup bu verilerin bir grafiğini çiziniz.

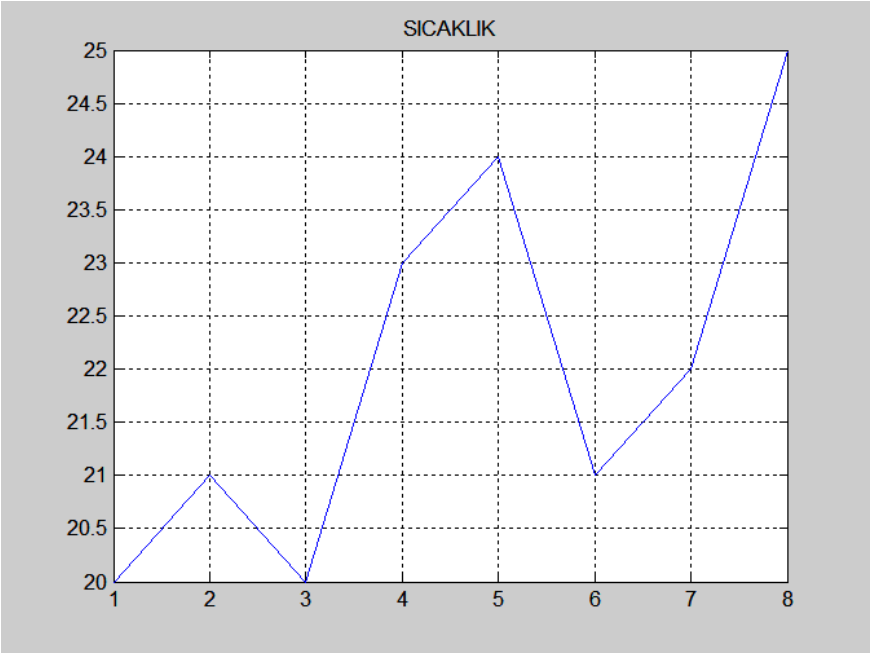
Çözüm 9.15

Gerekli olan MATLAB komutları aşağıda verilmiştir. İlk olarak SICAKLIK sayfası okunur ve veriler SICAKLIK_BUGUN isimli bir matrisde saklanır. Daha sonra ise bu matrisin birinci sütunu *x* vektöründe, ikinci sütunu ise *y* vektöründe saklanırlar. *plot* komutu kullanılarak ise istenilen grafik çizilir:

```
>> SICAKLIK_BUGUN = xlsread('c:\ornek','SICAKLIK');
>> x = SICAKLIK_BUGUN(:,1);
>> y = SICAKLIK_BUGUN(:,2);
>> plot(x,y); title('SICAKLIK'); grid on
```

```
>>
```

Çizilmiş olan grafik şekil 9.5 de gösterilmiştir.

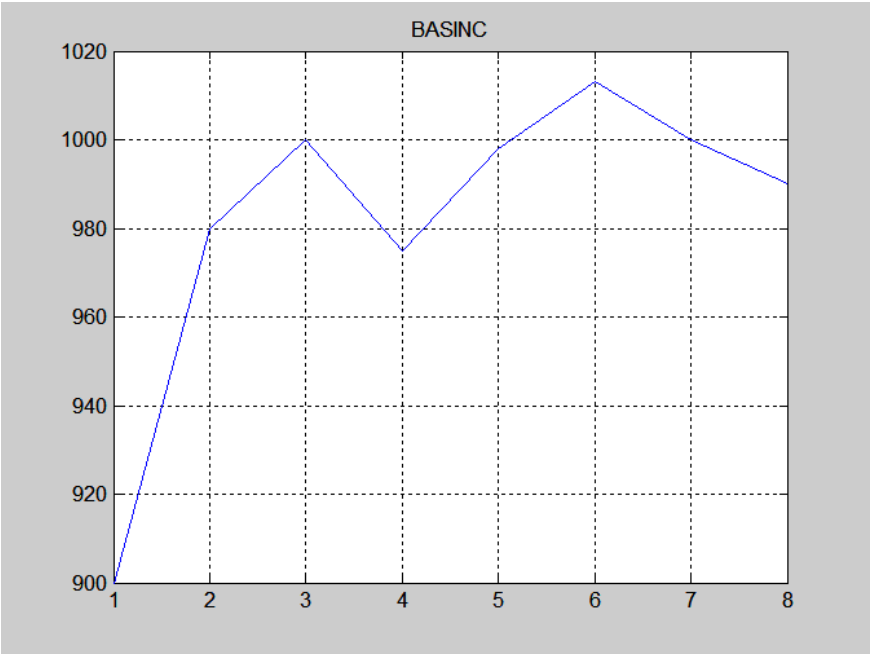


Şekil 9.5 SICAKLIK grafiği

Daha sonra ise BASINC verileri okunur ve bu veriler BASINC_BUGUN isimli bir matrisde saklanır ve bu verilerden *x* ve *y* değerleri bulunur:

```
>> BASINC_BUGUN = xlsread('c:\ornek','BASINC');
>> x = BASINC_BUGUN(:,1);
>> y = BASINC_BUGUN(:,2);
>> plot(x,y); title('BASINC'); grid on
>>
```

plot komutu kullanılarak istenilen grafik çizilir (Şekil 9.6).



Şekil 9.6 BASINC grafiği

Bazı uygulamalarda ‘spreadsheet’ dosyasını açmazdan önce bu dosya hakkında bilgi bulmamız gerekebilir. MATLAB fonksiyonu xlsfinfo bu maksat için kullanılabilir. Xlsfinfo fonksiyonunun kullanılış şekli şu şöyledir:

```
[cesit, sayfalar] = xlsfinfo(dosya-ismi)
```

Burada ‘spreadsheet’ dosyasının ne çeşit bir dosya olduğu cesit değişkeninde, dosyadaki sayfa isimleri ise sayfalar değişkeninde saklanır. Yukarıdaki örneği göz önünde bulundurursak, ornek.xls dosyası hakkında şu bilgileri elde ederiz:

```
>> [cesit, sayfalar] = xlsfinfo('c:\ornek')

cesit =

        Microsoft Excel Spreadsheet
```

```
sayfalar =

    'SICAKLIK'    'BASINC'

>>
```

Bu örnekte, ‘spreadsheet’ normal ‘Microsoft Excel Spreadsheet’ olup örnekte SICAKLIK ve BASINC olmak üzere 2 sayfa bulunmaktadır.

9.8 Sorular

9.1 *ornek.txt* isimli bir dosya açınız ve şu yazıları bu dosyada saklayınız:

Bu dosyada son elde edilen neticeler bulunmaktadır.
Neticeler 2 satır olarak yazılmıstır.
Birinci satirda zaman, ikinci satirda ise veriler bulunur.

9.2 Yukarıdaki dosyadaki yazıları okumak için bir MATLAB Programı yazınız. Okuduğunuz yazıları ekranda gösteriniz.

9.3 1 den 10 a kadar olan sayıların kareköklerini hesaplayınız ve bir tablo şeklinde *karekok.txt* isimli bir dosyada saklayınız.

9.4 9.3 deki sayıları bir MATLAB programı ile okuyunuz ve bu sayıların karelerini alıp ekranda gösteriniz.

9.5 Aşağıdaki sayıları MATLAB komutları kullanarak bir dosyada saklayınız:

1	10
2	20
3	30
4	40
5	50

6 60

9.6 9.5 deki sayıları *textread* komutunu kullanarak okuyunuz ve bu verilerin grafiğini çiziniz.

9.7 9.5 deki sayıları *dlmwrite* komutunu kullanarak bir dosyada saklayınız.

9.8 *dlmread* komutunu kullanarak 9.7 de yaratmış olduğunuz dosyadaki sayıları okuyup bu sayıları ekranda gösteriniz.

9.9 ornek.xls isimli bir EXCEL dosyası yaratınız ve aşağıdaki sayıları bu dosyada saklayınız:

2	4
3	9
4	16
5	25
6	36
7	49
8	64

9.10 *xlsread* fonksiyonunu kullanarak 9.9 da yaratmış olduğunuz dosyadaki sayıları okuyunuz.

9.11 9.10 da okumuş olduğunuz sayıların grafiğini çiziniz.

9.12 9.9 da yaratmış olduğunuz dosyanın çeşidini ve bu dosyada bulunan sayfa isimlerini gösteren MATLAB komutunu yazınız.

10
MATLAB İLE İNTEGRAL HESAPLAMAK

MATLAB programı oldukça güçlü olup her türlü matematiksel işlemi yapacak güce sahiptir. Bu bölümde MATLAB'ı kullanarak integral ve türev işlemlerinin nasıl yapıldığını göreceğiz.

10.1 İntegral

MATLAB'ı kullanarak verilen herhangibir fonksiyonun verilen aralıklardaki integralini hesaplayabiliriz. *quad* ve *quadl* fonksiyonları bu maksat için kullanılabilir. *quad* fonksiyonu adaptiv-recursiv Simpson algoritmasını kullanır. *quadl* fonksiyonu ise Lobatto algoritmasını kullanır.

quad ve *quadl* fonksiyonlarının kullanımlarına örnekler aşağıda verilmiştir.

Örnek 10.1

$f(x) = x^2 + 2x + 1$ fonksiyonunun $x = 0$ ve $x = 1$ arasındaki integralini hesaplayınız.

Çözüm 10.1

quad fonksiyonunu kullanarak istenilen integrali şu şekilde hesaplayabiliriz. Burada ilk olarak fonksiyon bir karakter dizisi olarak yazılmalıdır. Daha sonra quad fonksiyonu kullanılır ve istenilen alt limit ve üst limit belirtilir:

```
>> f = 'x.^2+2*x+1';
>> quad(f,0,1)
```

```
ans =

2.3333

>>
```

İstenilen cevap 2.333 olur.

quadl fonksiyonunu kullanırsak yine ayni cevabı elde ederiz:

```
>> quadl(f,0,1)

ans =

2.3333

>>
```

Örnek 10.2

sin(x) trigonometrik fonksiyonun x = 0 ve x = pi arasındaki integralini hesaplayınız.

Çözüm 10.2

Quad fonksiyonunu kullanarak istenilen integrali şu şekilde hesaplayabiliriz:

```
>> f='sin(x)';
>> quad(f,0,pi)

ans =

2.0000

>>
```

10.1.1 Trapezoidal (Yumuk) İntegral Algoritması

Trapezoidal integral en basit hesaplayabileceğimiz integral algoritmasıdır. Bu integralin hesaplanması burada örnek

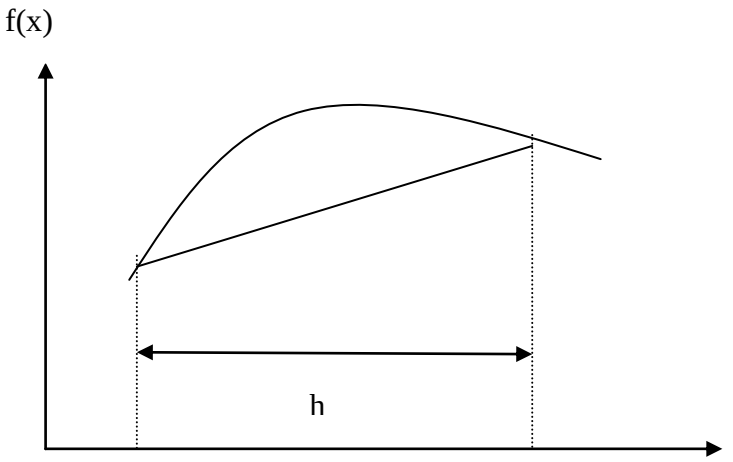
olarak verilmiştir ve MATLAB quad veya quad8 fonksiyonları tavsiye edilmektedir.

Şekil 10.1 de görüleceği gibi fonksiyon istenilen a ve b noktaları arasında eşitlikleri h olan dikey bölümlere ayrılır. Daha sonra aşağıdaki formül kullanılarak istenilen integral hesaplanmış olur:

$$S = h/2 \{ f(a) + f(b) + 2[f(x_1) + f(x_2) + \ldots + f(x_{n-1})] \}$$

ve

$$x_i = a + ih$$



Şekil 10.1 Trapezoidal algoritması

Örnek 10.3

Trapezoidal algoritmasını kullanarak verilen bir fonksiyonun integralini hesaplayan bir MATLAB fonksiyonu yazınız. Daha sonra, $f(x) = x^2 + 2x + 1$ fonksiyonunun $x = 0$ ve $x = 1$ arasındaki integralini hesaplayınız.

Çözüm 10.3

İstenilen MATLAB fonksiyonunun listesi aşağıda verilmiştir. Burada x ve y değişkenleri vektör olarak kullanılmıştırlar. Fonksiyonun herhangibir noktadaki değeri ise MATLAB feval fonksiyonu kullanılarak hesaplanmıştır:

```
% Trapezoidal algoritmasini kullanarak verilen
% bir fonksiyonun integralini hesaplayan
% MATLAB fonksiyonu.
%
function y = trapezoidal(f, a, b, h)
n = (b - a) / h;
x = a + [1:n-1]*h;
y = sum(feval(f, x));
y = h/2 * (feval(f, a) + feval(f, b) + 2*y);
```

İstenilen fonksiyonun integrali aşağıda hesaplanmıştır (bu örnekte h = 0.1 olarak alınmıştır):

```
>> f = inline('x.^2 + 2*x +1');
>> trapezoidal(f,0,1,0.1)

ans =

2.3350
```

h değerini daha küçük alırsak daha hassas değerler elde ederiz. Örneğin, h = 0.001 olarak alırsak:

```
>> trapezoidal(f,0,1,0.001)

ans =

2.3333

>>
```

Burada bulduğumuz değer Örnek 10.1 de bulduğumuz değerin aynisidir.

10.2 Sorular

1. $f(x) = x^2 + 2x + 10$ fonksiyonunun x = 0 ve x = 4 arasındaki integralini quad fonksiyonunu kullanarak hesaplayınız.

2. f(x) = 2sin(x) fonksiyonunun x = 0 ve x = pi/2 arasındaki integralini quad fonksiyonunu kullanarak hesaplayınız.

3. f(x) = sin(x) + cos(x) fonksiyonunun x = 0 ve x = pi/4 arasındaki integralini hesaplayınız.

4. Soru 3 deki integrali elde yaparak bulduğunuz cevabı MATLAB cevabı ile mukayese ediniz.

5. f(x) = 1 / x fonksiyonunun x = 1 ve x = 2 arasındaki integralini quad ve quad8 fonksiyonlarını kullanarak hesaplayınız.

11
SEMBOLİK MATEMATİK

Şimdiye kadar olan MATLAB örneklerimizde sadece sayılar ile işlem yaptık ve netice olarak sayı elde ettik. Bu bölümde sembolik matematik işlemlerin nasıl yapıldığını göreceğiz. Sembolik işlemlerde netice sayı olmayıp karakterlerden meydana gelen bir sembol olabilmektedir. Örneğin, x^2 polinomunun türevini sembolik olarak bulduğumuzda cevap 2*x olur. Ayni şekilde, (x^2 –1) in çarpanlarını sembolik olarak (x – 1)* (x + 1) olarak bulabiliriz.

MATLAB ile sembolik işlem yapabilmek için "Symbolic Math Toolbox" veya MATLAB "Student Edition" versiyonumuz olması gerekir. MATLAB'da sembolik işlemler "Waterloo Maple Software Inc." firmasının geliştirmiş olduğu "Maple V" yazılım paketini kullanmaktadır.

11.1 MATLAB'da Sembol Yaratmak

MATLAB'da sembol yaratmak için **sym** fonksiyonunu veya **syms** komutunu kullanmamız gerekir.

sym fonksiyonu ile, fonksiyon her kullanıldığında sadece bir tane sembol yaratabiliriz. Aşağıdaki örnekte x ve y isimli değişik iki sembol yaratılmıştır:

```
>> x = sym('x')
>> y = sym('y')
```

syms komutu daha genel olup bu komutu kullanarak ayni satırda istediğimiz kadar sembol yaratabiliriz. Aşağıdaki örnekte x, y, u ve v isimli 4 sembol yaratılmıştır:

```
>> syms x, y, u, v
```

Sembollerle işlem yaparken normal aritmetik operatörleri (* + - / ^) ve aritmetik fonksiyonları kullanabiliriz. Örneğin,

```
>> syms a b c d
>> a = d + 1;
>> b = d + 2;
>> c = a * b

c =

(d+1)*(d+2)

>>
```

olur. Bu örnekte, ilk olarak a sembolüne d + 1 değeri verildi, sonra b sembolüne d + 2 değeri verildi. İki sembolün çarpımı ise (d+1)*(d+2) olarak bulunmuştur.

findsym komutunu kullanarak verilen bir sembolik işlemde sembol olup olmadığını bulabiliriz. Örneğin, yukarıdaki işlemi göz önünde bulundurursak:

```
>> findsym(c)

ans =

d

>>
```

olur. Eğer verilen bir sembolik işlemde sembol yoksa MATLAB aşağıda gösterildiği gibi hata verir:

```
> syms a b c
>> findsym(e)
??? Undefined function or variable 'e'.

>>
```

Yukarıdaki işlemde **e** gibi bir sembol olmadığı için MATLAB hata vermiştir. **findsym** komutunun bir diğer kullanım şekli de **findsym(işlem, n)** dir. Burada n, işlem içerisinde x en yakın n tane sembol bulmaya çalışır. Aşağıdaki örnekte, x e en yakın olarak y sembolü verilmiştir:

```
>> syms a b y z
>> findsym(a+b+y+z,1)

ans =

y

>>
```

11.2 Sembollerle İşlem Yapmak

collect komutu ayni kuvvette olan sembolleri bir arada toparlar. Aşağıdaki örnekte a, b, ve c isimli üç tane sembol tanımlanmıştır. Daha sonra a sembolüne bir işlem eşitlenmişir. **collect** komutunu a ya eşitlenmiş olan sembollerin ayni kuvvette olan katsayılarını toparlar:

```
>> syms a b c
>> a = (b - 1)^2 + b^2 + 1;
>> c = collect(a)

c =

2*b^2-2*b+2

>>
```

expand komutu, bir sembole eşitlenmiş olan bir işlemi açıp ayni kuvvette olan terimleri toparlar. Aşağıdaki örnekte **a** sembolüne eşit olan $(b - 1)^2$ işleminin tam açılımı bulunur:

```
>> a = (b - 1 )^2;
```

```
>> c = expand(a)

c =

b^2-2*b+1

>>
```

Ayni şekilde,

```
>> a = (b + 2) ^ 3;
>> c = expand(a)

c =

b^3+6*b^2+12*b+8

>>
```

olur.

factor komutu bir işlemde bulunan sayıları çarpanlarına ayırır. Aşağıdaki örnekte b^2 - 2b + 1 işlemi çarpanlarına ayrılmış ve cevap $(b - 1)^2$ olarak bulunmuştur:

```
>> a = b^2 -2*b+1;
>> c = factor(a)

c =

(b-1)^2

>>
```

simplify komutu sembollerle yapılan bir işlemi basitleştirmeye çalışır. Aşağıdaki örnekte a sembolündeki işlem basitleştirilmiştir:

```
>> a = 2*b^2 + 3*b + 4*b^2 +5 - 4*b;
>> c = simplify(a)
```

```
c =

6*b^2-b+5

>>
```

numden komutu bir sembolün pay ve paydasını bulur. Aşağıdaki örnekte, a sembolüne (b – 1) / (b + 1) işlemi eşitlenmiştir. Daha sonra bu işlemin payı **x** sembolüne eşitlenmiş, paydası ise **y** sembolüne eşitlenmiştir:

```
>> syms a b x y
>> a = (b - 1) / (b + 1);
>> [x,y] = numden(a)

x =

b-1


y =

b+1

>>
```

Bir sembolün değerini bulmak için **subs** komutunu kullanırız. Bu komutun genel şekli şöyledir:

```
subs(işlem, x, n)
```

Burada x sembolüne n değeri verilerek işlemin değeri bulunur. Aşağıdaki örnekte a sembolünün x = 2 olduğunda almış olduğu değer bulunmuş ve bu değer k değişkeninde saklanmıştır:

```
>> a = x^2 + 2*x - 1;
>> k = subs(a,x,2)

k =

    7
```

```
>>
```

11.3 Sembollerle Grafik Çizimi

ezplot komutunu kullanarak bir sembolde tanımlanmış olan bir fonksiyonun grafiğini çizebiliriz. **ezplot** komutunun genel şekli şöyledir:

```
ezplot(sembol, [xmin xmax])
```

Burada xmin ve xmax x ekseni limitlerini belirtir.

Aşağıda bir örnek verilmiştir.

Örnek 11.1

Sembol kullanarak x^2 - 4x + 1 fonksiyonunun x = -5 ve x = 5 arasında grafiğini çiziniz.

Çözüm 11.1

İlk olarak fonksiyonu bir sembole eşitlememiz gerekir. Daha sonra da **ezplot** komutunu kullanarak grafik çizebiliriz.

Gerekli olan MATLAB komutları aşağıda verilmiştir:

```
>> syms f x
>> f = x^2-4*x+1;
>> ezplot(a,[-5 5])
>>
```

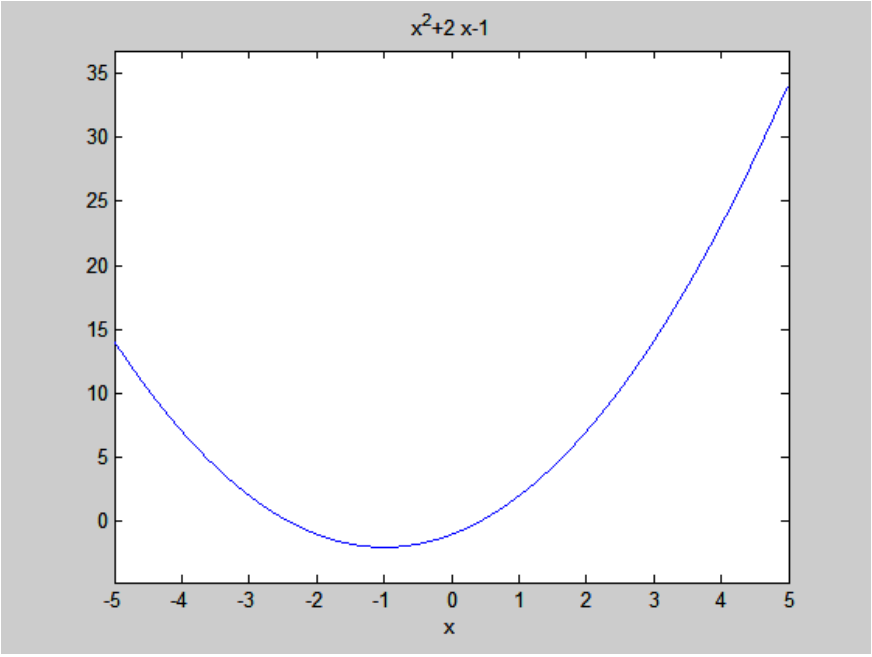
Çizilmiş olan grafik ise Şekli 11.1 de gösterilmiştir.

Örnek 11.2

Sembol kullanarak sin(x) + cos(x) fonksiyonunun 0 ve 2π arasında grafiğini çiziniz.

Çözüm 11.2

Gerekli olan komutlar aşağıda verilmiştir:



Şekil 11.1 ezplot komutu kullanılarak çizilmiş grafik

```
>> syms f x
>> f=sin(x) + cos(x);
>> ezplot(f,[0 2*pi])
>>
```

Çizilmiş olan grafik ise Şekil 11.2 de gösterilmiştir.

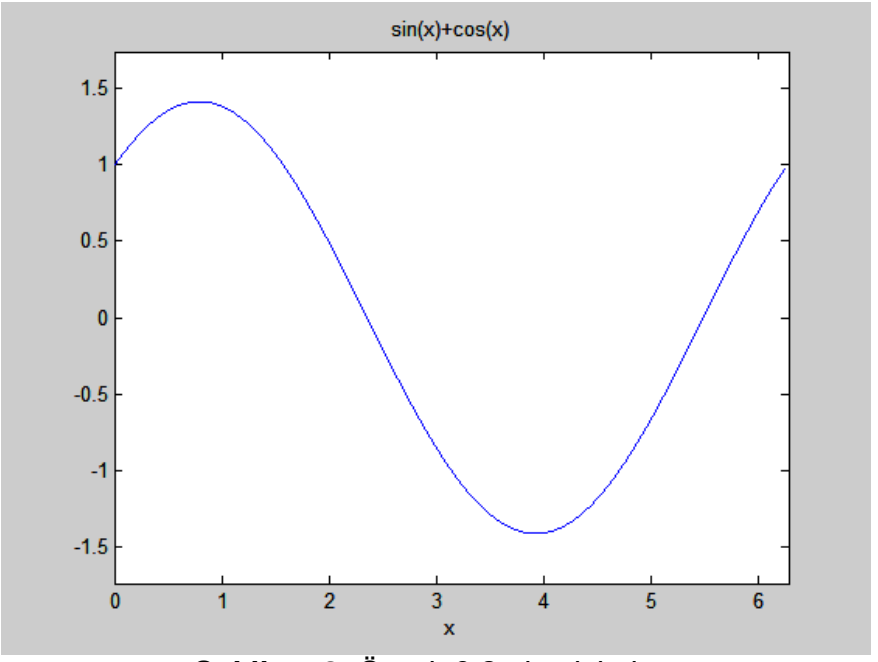
11.4 Sembol Kullanarak Denklem Çözümü

MATLAB'da sembolleri kullanarak denklem çözmemiz mümkündür. Bunun için solve komutunu kullanırız. Bazı örnekler aşağıda verilmiştir:

Örnek 11.3

Aşağıdaki denklemin köklerini sembol kullanarak bulunuz:

$x^2 – 7x + 12 = 0$



Şekil 11.2 Örnek 6.2 nin çizimi

Çözüm 11.3

Denklemin sembolik olarak çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> syms x y
```

```
>> y = 'x^2-7*x+12';
>> solve(y)

ans =

[ 3]
[ 4]

>>
```

Burada denklemin kökleri x = 3 vex = 4 olarak bulunurlar.

Örnek 11.4

Aşağıdaki denklemin köklerini sembol kullanarak bulunuz:

$$x^2 + 4x + 5 = 0$$

Çözüm 11.4

Denklemin sembolik olarak çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> y = 'x^2 + 4*x + 5';
>> solve(y)

ans =

[ -2+i]
[ -2-i]

>>
```

Denklemin kökleri kompleks olup x = -2+i ve x = -2-i olarak bulunurlar.

Sembolik matematik kullanarak lineer denklem setlerini çözmemiz de mümkündür. Aşağıda bir örnek verilmiştir:

Örnek 11.5

Aşağıdaki denklemi sembol kullanarak çözünüz:

$$x + 2y = 7$$
$$2x + 3y = 11$$

Çözüm 11.5

Yukarıdaki denklemi çözmek için gerekli olan komutlar şunlardır:

```
>> denklem1 = 'x + 2*y = 7';
>> denklem2 = '2*x + 3*y = 11';
>> [x,y] = solve(denklem1,denklem2)

x =

1


y =

3

>>
```

Denklemin çözümü, x = 1, y = 3 olarak bulunur.

11.5 Sembolik Türev

MATLAB'I kullanarak sembolik türev işlemleri yapabiliriz. Türev işlemi için diff komutu kullanılır. Bazı örnekler aşağıda verilmiştir.

Örnek 11.6

Aşağıdaki polinomun x değişkenine göre türevini hesaplayınız:

$$f(x) = x^3 + x^2 + 3x + 5$$

Çözüm 11.6

İlk olarak fonksiyonu sembol olarak yazmamız gerekir. Daha sonra ise **diff** komutunu kullanarak fonksiyonun türevini hesaplayabiliriz.

Gerekli olan komutlar aşağıda verilmiştir:

```
>> f = 'x^3 + x^2 + 3*x + 5';
>> diff(f)

ans =

3*x^2+2*x+3

>>
```

Çözüm $3x^2 + 2x + 3$ olarak bulunur.

Örnek 11.7

Aşağıdaki fonksiyonun türevini hesaplayınız:

$$f(x) = x \sin(x)$$

Çözüm 11.7

Gerekli olan komutlar aşağıda verilmiştir:

```
>> f = 'x*sin(x)';
>> diff(f)

ans =
```

```
sin(x)+x*cos(x)

>>
```

Böylece, x.sin(x) fonksiyonunun türevi sin(x) + x.cos(x) olarak bulunmuş olur.

Örnek 11.8

Aşağıdaki fonksiyonun türevini hesaplayınız:

$$f(x) = \frac{2x^2}{x^2 + 3x + 1}$$

Çözüm 11.8

Gerekli olan işlemler aşağıda verilmiştir:

```
>> f = '(2*x^2)/(x^2 + 3*x + 1)';
>> diff(f)

ans =

4*x/(x^2+3*x+1)-2*x^2/(x^2+3*x+1)^2*(2*x+3)

>>
```

Böylece istenilen türev $\frac{4x}{x^2 + 3x + 1} - \frac{2x^2(2x+3)}{(x^2 + 3x + 1)^2}$ olarak bulunmuş olur.

MATLAB'ı kullanarak bir fonksiyonun birden fazla türevini de hesaplayabiliriz. Bunun için, diff komutunun şu şekli kullanılır:

```
diff(işlem,n)
```

Burada n kaçıncı türevi istediğimizi belirtir.

Aşağıda birkaç örnek verilmiştir.

Örnek 11.9

Aşağıdaki polinomun ikinci ve üçüncü türevlerini hesaplayınız:

$$f(x) = 2x^3 + 4x^2 + 5x + 8$$

Çözüm 11.9

Verilen fonksiyonun ikinci türevi şu şekilde hesaplanabilir:

```
>> f = '2*x^3 + 4*x^2 + 5*x + 8';
>> diff(f,2)

ans =

12*x+8

>>
```

Böylece, fonksiyonun ikinci türevi 12x + 8 olarak hesaplanır.

Üçüncü türev ise şu şekilde hesaplanabilir:

```
>> diff(f,3)

ans =

12

>>
```

Üçünci türev ise 12 ye eşit olmuş olur.

Örnek 11.10

Aşağıdaki fonksiyonun ikinci türevini hesaplayınız:

f(x) = x sin(x) + cos(x)

Çözüm 11.10

Verilen fonksiyonun ikinci türevi şu şekilde hesaplanabilir:

```
>> f = 'x*sin(x) + cos(x)';
>> diff(f,2)

ans =

cos(x)-x*sin(x)

>>
```

Böylece fonksiyonun türevi cos(x) – xsin(x) olarak bulunmuş olur.

11.6 Sembolik İntegral

MATLAB'ı kullanarak sembolik intehral hesapları yapmamız mümkündür. Bunun için int fonksiyonunu kullanırız. Aşağıda bazı örnekler verilmiştir.

Örnek 11.11

Aşağıdaki fonksiyonun integralini hesaplayınız:

$$\int (x^2 + 1)dx$$

Çözüm 11.11

Yukarıdaki fonksiyonun integrali şu şekilde hesaplanabilir:

```
>> f = 'x^2 + 1';
>> int(f)

ans =

1/3*x^3+x

>>
```

Cevap $\frac{x^3}{3} + x$ olarak bulunmuş olur.

Örnek 11.12

Aşağıdaki fonksiyonun integralini hesaplayınız:

$$\int [x\sin(x)]dx$$

Çözüm 11.12

Yukarıdaki fonksiyonun integrali şu şekilde hesaplanabilir:

```
>> f = 'x*sin(x)';
>> int(f)

ans =

sin(x)-x*cos(x)

>>
```

İntegral sin(x) – x.cos(x) olarak bulunmuş olur.

Örnek 11.13

Aşağıdaki fonksiyonun integralini hesaplayınız:

$$\int \frac{2}{x} dx$$

Çözüm 11.13

Yukarıdaki fonksiyonun integrali şu şekilde hesaplanabilir:

```
>> f = '2/x';
>> int(f)

ans =

2*log(x)

>>
```

Fonksiyonun integrali 2log(x) olarak bulunur.

MATLAB'I kullanarak limitleri verilmiş olan bir integralin cevabını hesaplayabiliriz. Bu uygulamalar için int komutunun şu şeklini kullanırız:

int(işlem, alt-limit, üst-limit)

Aşağıda bazı örnekler verilmiştir.

Örnek 11.14

Aşağıdaki fonksiyonun integralini hesaplayınız:

$$\int_{2}^{4} (x^2 + 4x + 1) dx$$

Çözüm 11.14

Yukarıdaki fonksiyonun integrali şu şekilde hesaplanabilir:

```
>> f = 'x^2 + 4*x + 1';
>> int(f,2,4)
```

```
ans =

134/3
```

Örnek 11.15

Aşağıdaki fonksiyonun integralini hesaplayınız:

$$\int_0^{\pi} (x \sin x) dx$$

Çözüm 11.15

Yukarıdaki fonksiyonun integrali şu şekilde hesaplanabilir:

```
>> f = 'x*sin(x)';
>> int(f,0,pi)

ans =

pi

>>
```

Böylece integralin cevabı π olmuş olur.

11.7 Sembolik Limit Hesaplanması

MATLAB'ı kullanarak herhangibir fonksiyonun limitini bulabiliriz. Bunun için **limit** fonksiyonu kullanılır. Fonksiyon şu şekilde kullanılır:

```
limit(işlem,a)
```

Burada **a** istediğimiz limit değeridir.

Aşağıda bazı örneler verilmiştir.

Örnek 11.16

Aşağıdaki fonksiyonda x değişkeni 3 olurken fonksiyonun limitini hesaplayınız:

$$\lim_{x \to 3} \frac{x-3}{x^2-9}$$

Çözüm 11.16

Verilen fonksiyonun limiti şu şekilde hesaplanabilir:

```
>> limit((x-3)/(x^2-9),3)

ans =

1/6

>>
```

Cevap 1/6 olarak bulunur.

Örnek 11.17

Aşağıdaki fonksiyonda x değişkeni 3 olurken fonksiyonun limitini hesaplayınız:

$$\lim_{x \to 2} (x^2 + 5x + 1)$$

Çözüm 11.17

Verilen fonksiyonun limiti şu şekilde hesaplanabilir:

```
>> limit(x^2 + 5*x + 1,2)
```

```
ans =

15

>>
```
olarak bulunur.

MATLAB ile bir fonksiyonun limitini hesaplarken eğer x değişkeni 0 ise bunu limit komutu içerisinde belitmemiz gerekmez. Örneğin,

$$\lim_{x \to 0} (x^2 + 5x + 1)$$

hesaplarken şu komutları kullanabiliriz:

```
>> syms x
>> limit(x^2 + 5*x + 1)

ans =

1

>>
```

Cevap 1 olarak bulunur.

11.8 Sembolik Diferansiyel Denklem Çözümü

Diferansiyel denklemler matematik ve bilhassa kontrol mühendisliği dallarında çok yaygın olarak kullanılmaktadırlar. MATLAB ile herhangibir diferansiyel denklemi çözmek oldukça kolaydır.

Genel olarak birinci derecede bir diferansiyel denklem şu şekilde yazılabilir:

$$\frac{dy}{dt} = f(t, y)$$

Burada **t** bağımsız değişken, **y** ise **t** nin bir fonksiyonudur. Diferansiyel denklemlerin çözümünde genel olarak sabit katsayılar ortaya çıkmaktadır. Bu sayıların değerleri denklemin ilk durumlarını (uç durumlarını) bildiğimiz zaman bulunabilirler. MATLAB'ı kullanarak denklemin genel çözümünü ve bu katsayıları da bulmamız mümkündür.

İkinci derecede bir diferansiyel denklem şu şekilde yazılmaktadır:

$$\frac{d^2y}{dt^2} = f(t, y, \frac{dy}{dt})$$

Böyle bir denklemin çözümünde ise iki tane sabit sayı ortaya çıkmaktadır. Bu sayılar yine uç noktları bilinince kolaylıkla bulunabilirler.

MATLAB ile diferansiyel denklem çözümünde **dsolve** komutu kullanılır. Bu komut içerisine diferansiyel denklem, ve bilinirse uç noktalarındaki değerler de yazılır. Böylece denklemin tam çözümü bulunmuş olur. Burada önemli olan nokta, denklemi yazarken eşittir işaretinin de konması gerekmektedir. Ayni zamanda, $\frac{dy}{dt}$ yerine **Dy** sembolü ve $\frac{d^2y}{dt^2}$ yerine ise **D2y** sembolü konmalıdır.

Diferansiyel denklem çözümü üzerine bazı örnekler aşağıda verilmiştir.

Örnek 11.18

Aşağıdaki birinci derece diferansiyel denkleminin çözümünü bulunuz:

$$\frac{dy}{dt} + 3y = 10$$

Çözüm 11.18

Bu denklemi çözmek için **dsolve** komutunu kullanmalıyız. Gerekli olan komutlar aşağıda verilmiştir. **dsolve** komutunu kullanırken y değişkenini sembol olarak tanımlamaya gerek yoktur:

```
>> dsolve('Dy + 3*y = 15')

ans =

5+exp(-3*t)*C1

>>
```

Böylece cevap $5 + C_1e^{-3t}$ olarak bulunur.

Örnek 11.19

Örnek 11.18 de verilen denklemde y(0) = 2 olduğunu kabul edip denklemin tam çözümünü bulunuz.

Çözüm 11.19

Denklemin uç noktasını da belirtmek için gerekli olan komut aşağıda verilmiştir:

```
>> dsolve('Dy + 3*y = 15, y(0) = 2')

ans =

5-3*exp(-3*t)

>>
```

Uç noktası da göz önünde bulundurularak denklemin tam çözümü $5 + 3e^{-3t}$ olarak bulunur.

Örnek 11.20

Aşağıdaki birinci derece diferansiyel denkleminin çözümünü bulunuz:

$$\frac{dy}{dt} = \cos(2t)$$

Çözüm 11.20

Denklemin çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> dsolve('Dy = cos(2*t)')

ans =

1/2*sin(2*t)+C1

>>
```

Denklemin çözümü $\frac{1}{2}\sin(2t) + C_1$ olarak bulunur.

Örnek 11.21

Örnek 11.20 deki denklemde y(1) = 1 olduğunu kabul edip denklemin tam çözümünü bulunuz.

Çözüm 11.21

Denklemin çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> dsolve('Dy = cos(2*t), y(1) = 1')

ans =
```

```
1/2*sin(2*t)-1/2*sin(2)+1

>>
```

Örnek 11.22

Aşağıdaki birinci derecedeki diferansiyel denklemin çözümünü bulunuz:

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

Çözüm 11.22

Denklemin çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> dsolve('Dy = 1 + y^2')

ans =

tan(t+C1)

>>
```

Cevap tan(t + C1) olarak bulunur.

Denklemde y(0) = 1 olduğunu kabul edersek C1 katsayısının değerini ve denklemin tam çözümünü şu şekilde bulabiliriz:

```
>> dsolve('Dy = 1 + y^2, y(0) = 1')

ans =

tan(t+1/4*pi)

>>
```

Örnek 11.23

Aşağıdaki ikinci derece diferansiyel denklemin çözümünü

bulunuz:

$$\frac{d^2y}{dt^2} = 4y$$

Çözüm 11.23

Denklemin çözümü için gerekli olan komut aşağıda verilmiştir:

```
>> dsolve('D2y = 4*y')

ans =

C1*sinh(2*t)+C2*cosh(2*t)

>>
```

Örnek 11.24

Örnek 11.23 de verilen denklemde y(0) = 1 ve y(1) = 2 olduğunu kabul edrsek denklemin tam çözümünü bulunuz.

Çözüm 11.24

Denklemin çözümü için gerekli olan komut aşağıda verilmiştir:

```
>> dsolve('D2y = 4*y', 'y(0) = 1', 'y(1) = 2')

ans =

-(cosh(2)-2)/sinh(2)*sinh(2*t)+cosh(2*t)

>>
```

Örnek 11.25

Aşağıdaki ikinci derece diferansiyel denklemin çözümünü bulunuz:

$$\frac{d^2y}{dt^2} - \frac{dy}{dt} - 2y = 0$$

y(0) = 0 ve y(1) = 1 olarak kabul ediniz.

Çözüm 11.25

Denklemin çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> dsolve('D2y - Dy - 2*y = 0', 'y(0) = 0', 'y(1) = 1')

ans =

-1/(-exp(-1)+exp(2))*exp(-t)+1/(-exp(-1)+exp(2))*exp(2*t)

>>
```

olarak bulunur.

simple komutunu kullanarak denklemi daha basit bir şekilde yazabiliriz:

```
>> simple(ans)

ans =

(-exp(-t)+exp(2*t))/(-exp(-1)+exp(2))
```

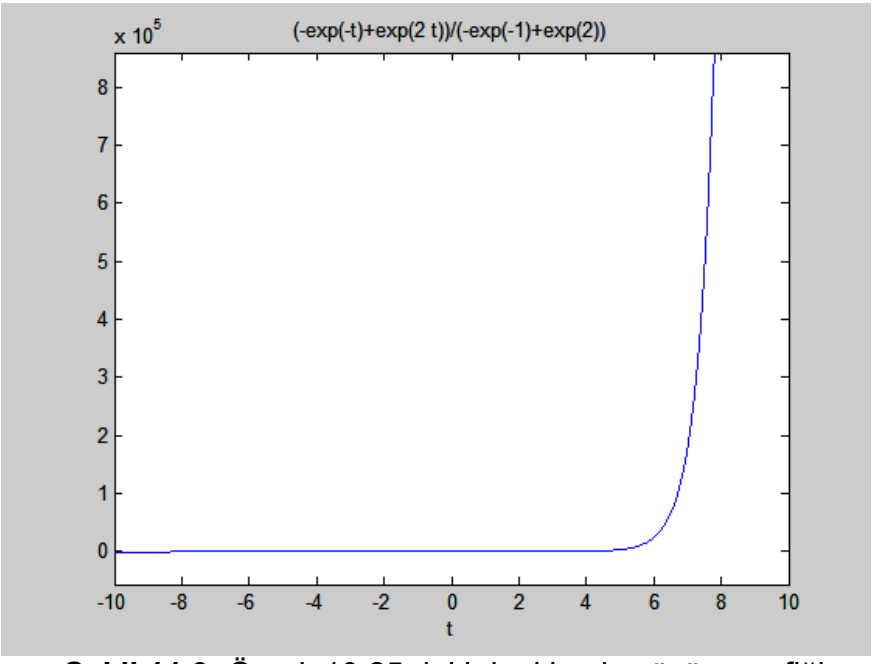
Ayni şekilde, **pretty** komutunu kullanarak denklemi daha okunaklı olarak yazmamız mümkündür:

```
>> pretty(ans)

                    -exp(-t) + exp(2 t)
                    -------------------
                     -exp(-1) + exp(2)
>>
```

Bulmuş olduğumuz cevabın **ezplot** komutunu kullanarak grafiğini çizmemiz mümkündür:

```
>> ezplot(ans, [-10 10])
>>
```

Çizilmiş olan grafik Şekil 11.3 de gösterilmiştir:



Şekil 11.3 Örnek 10.25 deki denklemin çözüm grafiği

Örnek 11.26

Bu örnekte birden fazla birbirine bağlı diferansiyel denklemlerin nasıl çözüldüğüne bir örnek verilmiştir.

Ağaşıdaki diferansiyel denklemerin çözümünü bulunuz:

$$\frac{dx}{dt} = 2x + 5y$$

$$\frac{dy}{dt} = -6x + 2y$$

Çözüm 11.26

Yukarıdaki denklemlerin çözümü için gerekli olan komutlar aşağıda verilmiştir:

```
>> [x,y] = dsolve('Dx = 2*x + 5*y', 'Dy = -6*x + 2*y')

x =

1/6*exp(2*t)*(6*cos(t*30^(1/2))*C1+30^(1/2)*sin(t*30^(1/2))*C2)


y =

1/5*exp(2*t)*(-30^(1/2)*sin(t*30^(1/2))*C1+5*cos(t*30^(1/2))*C2)

>>
```

olarak bulunur.

11.9 Sembolik Toplama

MATLAB'I kullanarak bir dizi sayıların toplamını kolaylıkla bulabiliriz. *symsum* fonksiyonu verilen iki aralıktaki sayıların toplamını hesaplar. Bu fonksiyonun kullanış şekli şöyledir:

```
S = symsum(E, a, b)
```

Burada E istenilen dizi, a ve b ise istediğimiz aralığı belirtirler. Netice S gibi bir değişkende saklanır.

symsum fonksiyonunun kullanımına birkaç örnek aşağıda verilmiştir.

Örnek 11.27

1 den 10 a kadar olan sayıların toplamını bulunuz.

Çözüm 11.27

Gerekli olan MATLAB komutları şunlardır:

```
>> syms k n
>> symsum(k, 0, 10)

ans =

55

>>
```

Cevap 55 olarak bulunur.

Örnek 11.28

1 den 3 e kadar olan sayıların karelerinin toplamını bulunuz.

Çözüm 11.28

Gerekli olan MATLAB komutları şunlardır:

```
>> syms k n
>> symsum(k^2, 1, 3)

ans =

14

>>
```

Cevap 14 bulunur.

Örnek 11.29

1 den n e kadar olan sayıların toplamını hesaplayan formülü yazınız.

Çözüm 11.29

Gerekli olan MATLAB komutları şunlardır:

```
>> syms k n
>> symsum(k, 1, n)

ans =

1/2*(n+1)^2-1/2*n-1/2

>>
```

Yukarıdaki formülü daha basit olarak şu şekilde yazabiliriz:

```
>> pretty(collect(ans))

                    2
                1/2 n  + 1/2 n
>>
```

Cevap olarak istenen formülü $1/2n^2 + 1/2n$ buluruz. Formülü şu şekilde de yazabiliriz:

$$\frac{n(n+1)}{2}$$

11.10 Sorular

1. MATLAB kullanarak aşağıdaki Limitleri çözünüz

   (a) $\lim_{x \to 0} \frac{x^2 - 1}{x + 1}$

   (b) $\lim_{x \to 0} x^2 + 2x + 1$

(c) $\lim_{x \to 1} x - 1$

2. Matlab kullanarak aşağıdaki Limitleri hesaplayınız.

(a) $\lim_{x \to 0} \frac{\sin x}{x^2}$

(b) $\lim_{x \to \pi} x \sin x$

3. Matlab kullanarak aşağıdaki fonksiyonların türevlerini bulunuz

(a) f(x) = x^2 *2x - 5
(b) f(x) = x^3 + 5x^2 *4x –1
(c) f(x) = $\frac{x}{x^2 + 1}$
(d) f(x) = $\sqrt{x - 2}$

4. MATLAB kullanarak aşağıdaki fonksiyonların integrallerini bulunuz

(a) $\int (x^2 + 5x - 2) dx$

(b) $\int x \sin x dx$

(c) $\int \frac{1}{x - 2} dx$

(d) $\int (x \sin x + x \cos x) dx$

5. MATLAB kullanarak aşağıdaki fonksiyonların integrallerini bulunuz

(a) $\int_0^1 (x+1)dx$

(b) $\int_1^3 (x^2+2x+10)dx$

(c) $\int_2^5 (\sqrt{x-1})dx$

6. Aşağıdaki diferansiyel denklemleri MATLAB ile çözünüz

(a) $\frac{dy}{dt}+2y=16$

(b) $\frac{dy}{dt}-4y=1$

(c) $\frac{dy}{dt}+2y=\sin 2t$

(d) $\frac{dy}{dt}+2y=2y+1$

7. Aşağıdaki diferansiyel denklemleri MATLAB ile çözünüz.

(a) $\frac{d^2y}{dt^2}=3y$

(b) $\frac{d^2y}{dt^2}=3y+4$

8. 1 den 100 e kadar olan sayıların toplamını bulunuz.

9. 1 den 10 a kadar olan sayıların küplerinin toplamını

bulunuz.

10. 1 den 10 a kadar olan sayıların karelerinin toplamını bulmak için gerekli olan formülü yazınız.

EK 1

EN ÇOK KULLANILAN MATLAB KOMUT VE FONKSİYONLARI

Lojik ve Karşılaştırma Operatörleri

==	Eşit
~=	Eşit değil
<	Küçük
<=	Küçük veya eşit
>	Büyük
>=	Büyük veya eşit
&	AND operatörü
|	OR operatörü
~	NOT operatörü

Çalışma Alanı Komutları

clc	Komut penceresini temizle
clear	Çalışma alanındaki bütün değişkenleri sil
help	MATLAB komutları hakkında yardım iste
lookfor	Belirli bir komut hakkında yardım iste
quit	MATLAB'dan çık
who	Çalışma alanındaki bütün değişkenleri listele
whos	Çalışma alanındaki değişkenleri uzun olarak listele

Giriş - Çıkış

disp	Ekranda bir değişkeni veya yazı göster
format	Ekranda sayı gösterme şeklini formatla
fprintf	Ekrana veya bir dosyaya yaz
input	Klavyeden oku

Vektör Fonksiyonları

cat	Karakter dizilerini birleştir
find	Sıfır olmayan indeksleri bul
length	Eleman sayısı
linspace	Regular aralıklı vektör yarat
logspace	Logaritmik aralıklı vector yarat
max	En büyük elemanı bul
min	En lüçük elemanı bul
size	Vektör size
sort	Her sütünu sıraya koy
sum	Her sütunu topla

Özel Matrisler

eye	Birim matris yarat
ones	Birler matrisi yarat
zeros	Sıfırlar matrisi yarat

Lineer Denklem Çözme Fonksiyonları

det	Bir matrisin determinantı
inv	Bir matrisin tersi
pinv	Bir matrisin pseudo tersi
rank	Bir matrisin rankı
rref	Azaltılmış sıra echelon formatı

Kompleks Sayı Fonksiyonları

abs(x)	x in mutlak değeri
angle(x)	Kompleks sayı x in açısı
conj(x)	x in konjugati
imag(x)	x kompleks sayısının gerçek olmayan bölümü
real(x)	x kompleks sayısının gerçek bölümü

Trigonometrik Fonksiyonlar

acos(x)	Ters kosinüs
asin(x)	Ters sinus
atan(x)	Ters tanjant
atan2(y,x)	Ters tanjant (4 bölgeli)
cos(x)	Kosinüs
cot(x)	Kotenjant
sec(x)	Secant
sin(x)	Sinus
tan(x)	Tanjant

Hiperbolik Fonksiyonlar

acosh(x)	Hiperbolik cosh tersi
asech(x)	Hiperbolik sech tersi
asinh(x)	Hiperbolik sinh tersi
atanh(x)	Hiperbolik tanh tersi
cosh(x)	Hiperbolik cosh
coth(x)	Hiperbolik coth
sech(x)	Hiperbolik sech
sinh(x)	Hiperbolik sinh
tanh(x)	Hiperbolik tanh

Polinom Fonsiyonları

conv	Iki polinomun çarpımı
deconv	Iki polinomun bölümü
poly	Köklerden polinom denkleminin yazılımı
polyval	Polinom hesaplaması
roots	Denklemin kökleri

Grafik Fonksiyonları

axis	Grafik ekseni
fplot	Fonksiyon grafiği
ginput	Kursor koordinatlarını oku

grid	Grafik gridi
plot	Grafik çiz
print	Grafiği yazıcıya gönder
Title	Grafik baş yazısı
axes	Eksen objeği yarat
gtext	Fare ile label yerleştirme
legend	Fare ile legend yerleştirme
set	Grafik obbjelerinin özelliklerini set yap
subplot	Alt grafik çiz
xlabel	X ekseni başlığı
ylabel	Y ekseni başlığı

Grafik Çeşitleri

bar	Bar grafik
loglog	Logaritmik grafik
plotyy	Sol ve sağ eksen grafiği
polar	Polar grafik
semılogx	Yarı logaritma grafiği (x logarithmic)
semılogy	Yarı logaritma grafiği (y logaritmik)
stairs	Sıra grafiği
stem	Stem grafiği

Program Kontrolu

case	switch komutu seçeneği
else	if komutu seçeneği
elseif	if komutu seçeneği
end	for, while ve if komutlarının bitişini gösterir
for	for komutu
if	if komutu
otherwise	switch komutu seçeneği
switch	switch komutu
while	while komutu

İntegral Fonksiyonları

quad	Simpsons algoritması ile integral
quadl	Lobatto algoritması ile integral
trapz	Trapezoidal (yumuk) algoritması ile integral

Sembol Fonksiyonları

collect	Ayni derecede olan değişkenleri toparla
expand	Güçlerini aç
factor	Çarpanlarını bul
poly2sym	Polinom katsayısını sembole dönüştür
simple	Bir işlemin en kısa gösterilişini bul
simplify	Bir işlemi basitleştir
subs	Yerine koy

Sembol Hesaplama Fonksiyonları

diff	Bir semblde türev almak
int	Bir sembolde integral almak
limit	Bir sembolde limit almak
symsum	Bir sembolde toplama
taylor	Bir sembolde Taylor serisini bulmak

İnterpolasyon Fonksiyonları

interp1	Tek değişkenli interpolasyon
interp2	Iki değişkenli interpolasyon
spline	Spline polinomu interpolasyonu

KAYNAKÇA

1. B.D. Hahn
   Essential Matlab For Scientists and Engineers
   Butterworth-Heinemann Publishers, 2002

2. D. Hanselman and B. Littlefield
   The Student Edition of Matlab
   The MathWorks Inc, 1997.

3. M.A. Natick
   Using Matlab
   The MathWorks Inc., 1997

4. F.P. Beer and E.R. Johnston
   Vector Mechanics For Engineers: Dynamics
   6^{th} Edition, McGraw-Hill, 1997.

5. W.J. Palm
   Introduction to Matlabl6 For Engineers
   McGraw-Hill, 2001.

DİZİN

MATLAB İLE ÇALIŞMAK

AYDIN BODUR

e-kitap

EMO Yönetim Kurulu 42. Dönem'de(Kasım 2010) bir yayın portalı oluşturdu. Bu yayın portalı üzerinde,daha önce de sürdürmekte olduğumuz, basılı dergilerimizin İnternet sürümleri, basılı kitaplarımızın tanıtımları ve çevrim içi satın alma olanakları ile doğrudan İnternet üzerinden bilgisayarınıza indirebileceğiniz e-kitapları çok düşük bedellerle edinebilme olanağına sahip olacaksınız.

İnternet sitemiz üzerinden e-kitap dağıtım hizmetini, yakında hizmete girecek olan EMO Yayın Portalı'nın öncülü olan, sitemizin yayın bölümünde yer alan e-kitaplarla uzunca bir süredir veriyorduk. Yayınlarımızı izleyenler hatırlayacaktır, ilk e-kitabımız, EMO üyesi Arif Künar'ın "Neden Nükleer Santrallere Hayır" kitabının PDF baskısıydı. Hükümetin Akkuyu'da nükleer santral kurma inadı maalesef hala kırılamadı. Dört yıl önce bastığımız bu kitap hala güncel!....

EMO'nun İnternet sitesi üzerinden hizmete giren bu yeni sitemizde yeni e-kitaplarla hizmete açıldı. Sizlerde varsa yayınlamak istediğiniz kitaplarınızı, notlarınızı bize iletebilirsiniz. Bu yayınlar yayın komsiyonumuzun değerlendirmesinden sonra uygun bulunursa yayınlanacak ve eser sahibine EMO ücret tarifesine göre ücret ödenecektir. E-Kitaplar tarafımızdan yayınlandıkça üyelerimize ayrıca eposta ile iletilecektir.

Saygılarımızla

Elektrik Mühendisleri Odası
42. Dönem Yönetim Kurulu

EMO YAYIN NO: EK/2011/28
ISBN. 978-605-01-0247-5

TMMOB Elektrik Mühendisleri Odası
Ihlamur Sokak No:10 Kat:2 Kızılay/Ankara
Tel: (312) 425 32 72 Faks: (312) 417 38 18
http:www.emo.org.tr E-Posta: emo@emo.org.tr