Programlama Ders Notları
2010-2011 Bahar Dönemi

Ertan Pekşen

Kocaeli Üniversitesi
Mühendislik Fakültesi
Jeofizik Mühendisliği Bölümü
14 Şubat 2011

İçindekiler

Bölüm 1

Bölüm 2

Bölüm 3

Bölüm 4

Bölüm 5

BÖLÜM 1

1. MATLAB A GİRİŞ

Matlab ın ilk versiyonu 1970 yılında Cleve Moler tarafından geliştirildi. Matlab bir sayısal hesaplama yazılımıdır (Matematik laboratuarı). Neden MATLAB sorusunun cevabı çok basit: İşlerimizi diğer programlama dillerine göre daha kolay bir şekilde yapabilmek için MATLAB ı kullanacağız. Program yazmak, grafik çizmek kolayca öğrenilip uygulanabilir. Her programlama dilinde olduğu gibi MATLAB da diğer programlama dillerine göre üstünlükleri ve zayıf yanları vardır. Fortran, C, C++, Visual Basic, Visual C, C#, Java vb. gibi diller ya da Surfer, Stanford grafik gibi paket programlarla karşılaştırmak dersin amacı dışındadır.

MATLAB ile programlama yapmak oldukça kolaydır. Örneğin Fortran ve C de olduğu gibi değişkenlerin tiplerini belirtmek problemiyle karşı karşıya kalınmıyor. Komutlar daha çok C diline benzemektedir. Bunun yanında kütüphanesi oldukça geniş. Grafik çizmek, kontur yapmak, 2 boyutlu ve 3 boyutlu şekiller çizmek oldukça kolay. Bir örnek verecek olursak, Fortran da iki matrisin çarpımı için en az üç 'DO' döngüsü gerekirken, MATLAB için sadece bu iki matrisi çarpmak işlemimizi tamamlamaya yetmektedir. Eğer mümkünse Matlab ta yapılan programlarda döngü işlemlerinden kaçınılır. Fakat bazı durumlarda mecburen döngü kurmak durumundayız.

MATLAB ı bilgisayarınıza kurduktan sonra, matlab ile çalışmak için matlab logosunu tıklayarak çalışmaya başlayabiliriz. Bu işlemden sonra, matlab ın sürüm numarasına göre ufak tefek değişiklikler olmasına rağmen, temel işlemler hep aynıdır, matlab işlecinin olduğu pencere matlab için komut penceresidir; işlemleri burada yapacağız.

>> ile Matlab ın komut almaya hazır olduğu anlaşılır. >> help komutu kullanılarak genel yardım ya da >> help plot ile daha özel yardımlar alınarak komutu nasıl kullanacağımız hakkında bilgi elde ederiz.

Eğer basit bir program yazmak istersek. İlk önce >> edit yazıp enter tuşuna basıp programı yazacağımız sayfaya geçmeliyiz. Program herhangi bir text editor ü kullanılarak ta yapılabilir. Önemli olan programın tür eki *.m olmalıdır.

>> edit (ya da fare ile sol üst köşeden boş bir m dosyası açılabilir)

```
close all; % Daha önce açılmış olan şekilleri grafikleri kapat
clear all; % Matlab Belleğinde bulunan değişkenleri siler
 x=-10:1:10;
y=x.^2+3.*x+2;
plot(x,y);
```

Bu programı şimdi deneme.m olarak kayıt edip programdan çıkalım. Programı çalıştırmak için >> deneme yazıp enter tuşuna basmamız yeterli olacaktır. Şimdi sırasıyla Matlab ın özelliklerine inceleyelim.

1.1 Yardım

Matlab ta herhangi bir işlem hakkında bilgi almak için help komutu kullanılır.

>> help

Eğer özel olarak fonksiyonun ismini biliyorsak

>>help mkdir

şeklinde kullanılır. Daha genel yardımlar için

>> help general (genel Matlab işlemleri dir, copy, delete gibi komutlar hakkında)
>> help elmat (temel matris işlemlerinin kullanımı hakkında)
>> help specfun (özel fonksiyonların kullanımı hakkında)
>>help lang (programlama komutları)

>>demo

komutuyla Matlab ın özelliklerini örneklerle görebilirsiniz.

Matlab hesap makinesi şeklinde kullanılabilir.

>>2+3/4*5
ans=5.75

İşlemlerde öncelik sırası aşağıdaki gibidir:

1. Parantez içi işlemler ( )
2. Üst ^
3. Çarpma * ve Bölme / (Soldan sağa öncelik 3*4/5=12/5)
4. Toplama + ve Çıkarma – (soldan sağa 3+4-5= 7-5)

1.2 Sayılar ve Format

Tür	**Örnek**
Doğal Sayılar	1233, -234
Gerçel Sayılar	1.23, -10.34
Karmaşık Sayılar	4.23 -4.78i (i=$\sqrt{-1}$)
Inf	(Sonsuz, sıfıra bölme sonucu oluşur)
NaN	(0/0 belirsizlik, tanımsız, bir sayı değil)

Sayılar üstel biçimde

--1.2341e+03= --1.2341x10^3= --1134.1

yazılabilir. Matlab ta bütün sayısal işlemler çift duyarlılıktadır (double precision).

>> help format

Format ı aşağıdaki şekilde değiştirebiliriz.
>> format short
>>format long e
>>format bank
…

1.3 Değişkenler

Harflerden ve rakamlardan oluşan herhangi bir kombinasyon ilk karakter harf olmak koşuluyla kullanılabilir.

Elec2, Hmag3, EMmag3x,emspec vb.

Rakamla başlayan, özel simgeler ve işaretler (% -+ @ vb.) değişken isimlendirilmesinde kullanılamaz. Net-Cost, %x, 2pay, @y kullanılamayan değişkenler ya da Matlab ta bazı özel anlamları olanlardır.

Size tavsiye olarak şunları söyleyebilirim: yazdığınız programlarda kullandığınız değişken isimlerinde, değişken hakkında bilgi verici özellikte olmasına özen göstermeyi alışkanlık haline getirmeniz, sizin için ve yazdığınız programın başkaları tarafından kolay anlaşılması açısından önemlidir.

Bazı durumlarda aşağıdaki şekilde bir formülü hesap eden bir program yapılması istenebilir.

$$\gamma = \frac{1}{2}\lambda e^{\sqrt{\theta\beta}}$$

Bu durumda klavye de Yunan alfabesi olmadığı için, formülü değişik şeklilerde yazabiliriz. Ben genellikle bu tür durumlarda şu şekilde değişken ismi veririm.

Gamma=(0.5)*lamda*exp(sqrt(teta*beta));

şeklinde ama aynı formül

xTyqwp=(0.5)*htdklqya1xsc*exp(sqrt(lkmmytag*ttQQxys76_wqerq));

şeklinde de olabilir, fakat estetik açıdan pek hoş değil. Hangi değişkenin ne tür bilgi içerdiği hakkında en ufak bir bilgi elde edemezsiniz. Bu nedenle program yazarken, değişkenler hakkında her programın ya da alt programın başında bilgi verilmesi çok önemlidir.

Örnegin, A=k*sin(wt) ile elektrik alan hesaplayalım. Değişken isimlerine örnekler:

ElektrikAlan=k*sin(w*t);
E_alan= sigma*sin(w*t);
E= k*sin(w*t);
…
şeklinde verilebilir.

1.4 Özel Değişkenler

Matlab ta pi , i ve j gibi özel değişkenler vardır. Programlama yaparken i ve j kullanırken dikkat edilmesi gerekir. i ve j nin default (varsayılan) değeri karmaşık sayıdır. Pi ise 3.1415… sayıdır.

1.4 1 Trigonometrik Fonksiyonlar

Trigonometrik fonksiyonlar sin, cos, tan ve cot tır. Bu fonksiyonlarda açı raydan olduğundan, dereceyi radyan cinsinde yazıp kullanmalıyız. Eğer x derece cinsinden açı değeri ise bu açının sinüsü

```
>>y=sin(x*(pi/180))
```

ile hesaplanır. y radyan cinsinden olduğundan eğer gerekliyse sonuç tekrar dereceye (180/pi) çarpanıyla dönüştürülebilir. cos(x), tan(x) ve ters trigonometrik fonksiyonlar (asin(y), acos(y)) da benzer şekilde hesaplanabilir.

1.5 Diğer Temel Fonksiyonlar

Bir sayı için (x) karekök, üs, logaritma (e ve 10 tabanına göre) sqrt(x), exp(x), log(x) ve log10(x) ile hesaplanır.

1.6 Satır ve Sütun Vektörleri

Vektörler [ ] ile oluşturulabilir. Köşeli parantez içinde satır vektörü oluşturmak için iki rakam arasına ya virgül konulur ya da boşluk bırakılır. Örneğe bakalım,

```
>> v1 = [ 1,2, 3, 4]

v1 = 1 2 3 4
```

satır vektörüdür. Sütun vektörü de şu şekilde oluşturulabilir.

>> v2= [1 ; 2; 3; 4]

v2 = 1
     2
     3
     4

Vektörün boyutu **size** komutuyla **size(v)** satır veya sütun olma durumuna göre (1,4) veya (4,1) olarak bulunur. Satır vektörünü, sütun vektöre ‘ işareti ile yapılabilir. Örneğin m=v2’ işlemi bize v2 sütun vektörünü m satır vektörüne çevirecektir. Bu işlemde satırlar ve sütunlar yer değiştirir, vektörün transpozu alınmış olur.

sort komutuyla herhangi bir vektör vec=[3 -2 5 1 0] sıraya dizilecektir.

>> sort(vec)

ans = -2 0 1 3 5

İki nokta üst üste kullanılarak sayı dizileri oluşturulabilir. (a:b:c a sayısından c sayısına kadar b artışla sayılar üretir)

>> 1:4
1 2 3 4
>> -1.4:-0.3:-2

-1.4 -1.7 -2.0

Sırası gelmişken eğer bir komut satırından sonra ; işareti varsa işlem sonucu ekranda görünmez. Program yazımında her satırda c dilinde olduğu gibi ; işareti konur. ; işareti konulmadan da program çalıştırılabilir fakat her değişken ekranda görüneceğinden bu tercih edilen bir durum değildir.

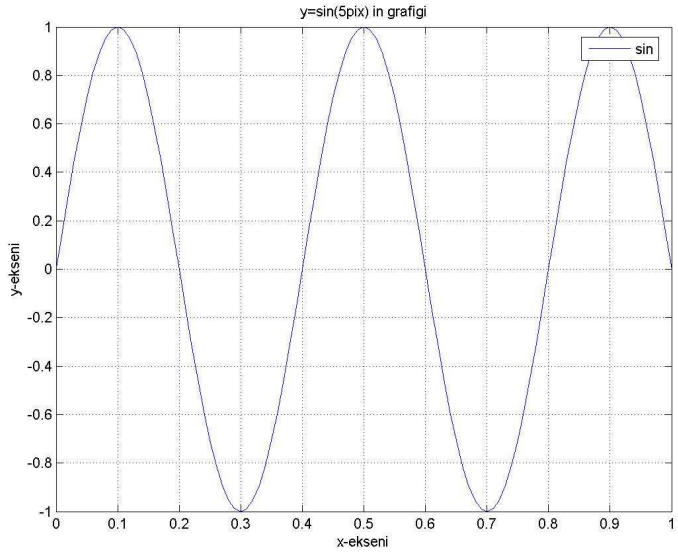
1.7 Grafik

Matlab da grafik çizmek hem eğlenceli, hem zevkli hem de kolay. Şimdi basit plot komutuyla ilgili dört program örneği üzerinde çalışalım. Bu programlarda **plot(x,y)** ve **subplot(abc)** komutları grafik çizmek için kullanıldı. Birinci program listesi:

```
close all
clear all
n=100;
dx=1/n;
x=0:dx:1;
y=sin(5*pi*x);
plot(x,y);
title('y=sin(5pix) in grafigi');
xlabel('x-ekseni');
ylabel('y-ekseni');
```

```
legend('sin');
grid;

print -djpeg graf1
```

bu program çalıştırıldığında ekranda Şekil 1 görünür.



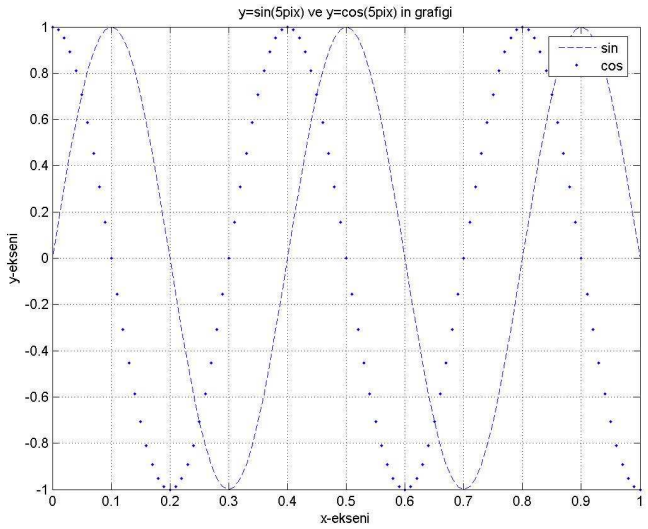
Şekil 1. Sinüs fonksiyonu.

Aynı zamanda bu program, ekranda görünen grafiği graf1.jpeg dosyası olarak çalışılan alt dizine kaydeder. Bu **print** komutu ile gerçekleştirilir.

Benzer şekilde bir diğer program örnek2 nin listesi

```
close all
clear all
n=100;
dx=1/n;
x=0:dx:1;
y1=sin(5*pi*x);
y2=cos(5*pi*x);

plot(x,y1,'--');
title('y=sin(5pix) ve y=cos(5pix) in grafigi');
xlabel('x-ekseni');
ylabel('y-ekseni');
hold on;
plot(x,y2,'.');

legend('sin','cos');
grid;

print -djpeg graf2
```

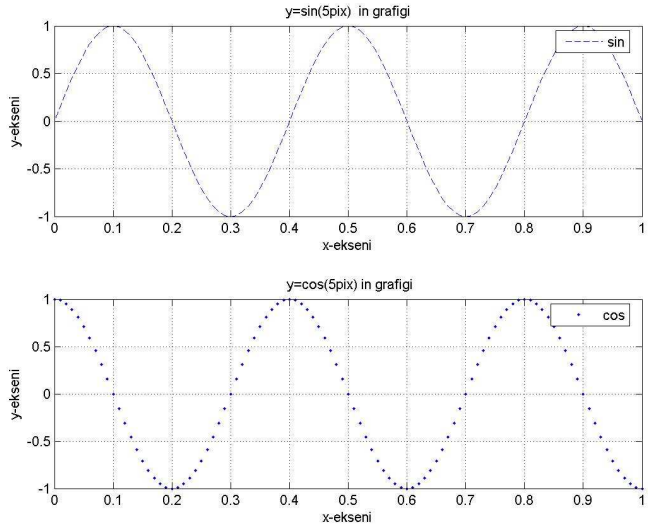
ile verilsin. Bu program sinüs ve kosinüs grafiklerini aşağıdaki şekilde üst üste çizer.



Şekil 2. Sinüs ve kosinüs fonksiyonları aynı grafikte.

örnek3 ve örnek 4 programlarında, eğer iki grafik üst üste veya yan yana ayrı çizilmesi gerekiyorsa, subplot komutu kullanılır. Aşağıdaki program ile

```
close all
clear all
n=100;
dx=1/n;
x=0:dx:1;
y1=sin(5*pi*x);
y2=cos(5*pi*x);

subplot(211)
plot(x,y1,'--');
title('y=sin(5pix)  in grafigi');
xlabel('x-ekseni');
ylabel('y-ekseni');
legend('sin');
grid;

subplot(212)
plot(x,y2,'.');
title(' y=cos(5pix) in grafigi');
xlabel('x-ekseni');
ylabel('y-ekseni');
legend('cos');
grid;

print -djpeg graf3
```



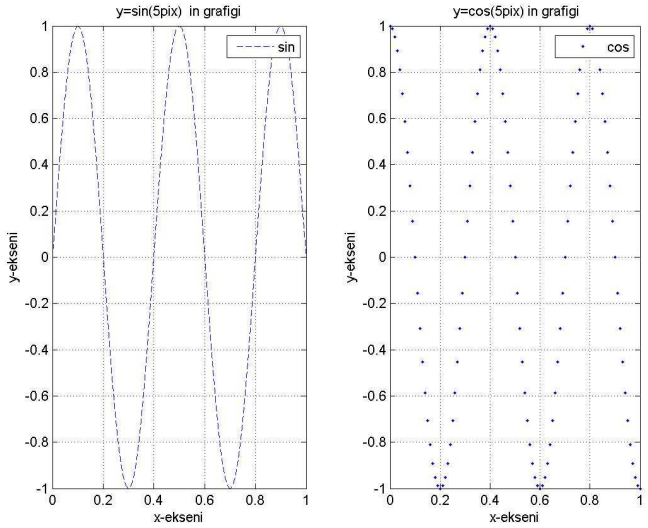
Şekil 3. Sinüs ve kosinüs fonksiyonlarının alt alta çizimi.

şekli elde edilir.

ornek4

```
close all
clear all
n=100;
dx=1/n;
x=0:dx:1;
y1=sin(5*pi*x);
y2=cos(5*pi*x);

subplot(121)
plot(x,y1,'--');
title('y=sin(5pix)  in grafigi');
xlabel('x-ekseni');
ylabel('y-ekseni');
legend('sin');
grid;

subplot(122)
plot(x,y2,'.');
title(' y=cos(5pix) in grafigi');
xlabel('x-ekseni');
ylabel('y-ekseni');
legend('cos');
grid;

print -djpeg graf4
```

Örnek 4 te, iki fonksiyon yan yana çizen program ve Şekil 4 te ise bu programın çalıştırılması durumunda ekranda görünmesi gereken şekil verilmiştir.



Şekil 4. Sinüs ve kosinüs fonksiyonlarının yan yana çizimi.

Bazı durumlarda özellikle kuyu logları uygulamalarında grafikler genellikle örnek 4 te olduğu gibidir. Çalıştığınız probleme göre grafikleri değişik şekillerde gösterebilirsiniz.

1.8 (*) , (.*) , dot ve cross çarpımlar

İki vektör u ve v için skaler çarpım

$$\vec{u} = [u_1, u_2, ..., u_n] \text{ ve } \vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ . \\ . \\ . \\ v_n \end{bmatrix}$$

$$\vec{u}.\vec{v} = \sum_{i=1}^{n} u_i v_i$$

şeklinde tanımlanır ve sonuç bir skalerdir. Bu çarpıma inner veya dot product ta denir.

```
close all
clear all
disp('skaler carpim')

u=[1,2,3]
v=[4;5;6]
```

```
prod=u*v
```

program çalıştırıldığında

>> skaler

skaler carpim
u =
    1   2   3
v =
    4
    5
    6
prod =
        32

Bu işlem Matlab komutu yardımıyla dot(u,v) ile hesaplanabilir. Fakat skaler çarpım için dot komutu kullanılacaksa, u ve v vektörlerinin ikisi ya satır veya sütun vektörü olmalıdır.

Nokta çarpım (.*) ile yapılır. Bu çarpım Hadamard çarpımı olarak ta bilinir. Bu işlemi yapabilmek için vektörlerin tipi ve boyu aynı olmalı. İkisi birden ya kolon vektör ya da sütun vektör olmalıdır. Skaler çarpımda kullandığımız vektörleri bu örnekte de kullanalım. Fakat ikiside aynı tip olmak koşuluyla,

$\vec{u} = [u_1, u_2, ..., u_n]$ ve $\vec{v} = [v_1, v_2, ..., v_n]$

$\vec{u} .* \vec{v} = [u_1 v_1, u_2 v_2, ..., u_n v_v].$

```
close all
clear all
disp('Hadamard carpim')
u=[1,2,3]
v=[4,5,6]
prod=u.*v

disp('Hadamard carpim')
u=[1;2;3]
v=[4;5;6]
prod=u.*v
```

>> dotp
Hadamard carpim

u =
 1   2   3
v =
    4   5   6

```
prod =
    4    10    18
Hadamard carpim
u =
    1
    2
    3
v =
    4
    5
    6
prod =
    4
   10
   18
```

Benzer şekilde iki vektör elemanlarını bölüp üssünü alabiliriz. Bu üç işlem .*, ./ ve .^ program yazmada çok büyük kolaylıklar sağlar ve gerekmedikçe döngü işlemlerinden mümkün olduğunca kaçınılır. Örneklerle bu işlemleri görelim.

$$\vec{u}./\vec{v} = \left[u_1 / v_1, u_2 / v_2, ..., u_n / v_v\right],$$

$$\vec{u}.\hat{}\vec{v} = \left[u_1 \hat{} v_1, u_2 \hat{} v_2, ..., u_n \hat{} v_v\right].$$

```
>> u=[1 2 3]
u =
    1     2     3
>> v=[4 5 6]
v =
    4     5     6

>> u.^v

ans =

    1    32   729
ve
>> u./v

ans =

   0.2500    0.4000    0.5000
```

Ayrıca u ve v vektörleri cross(u,v) ile vektör olarak çarpabiliriz.

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

```
u =
    1     2     3
>> v=[4 5 6]
v =
    4     5     6
>> cross(u,v)

ans =

   -3     6    -3
```

sonucu elde edilir. Sonuç vektördür.

1.9 Matrisler

Matrisler A(m x n) şeklinde gösterilir ve A(m,n) Matlab sunumudur. Burada m satır sayısı, n sütun sayısını gösterir. Örnek

```
disp('Matris')

A=[1,2,3;-2,1,-9]
[m,n]=size(A)
```

```
>> matr
Matris

A =
    1     2     3
   -2     1    -9

m =
    2
n =
    3
```

Burada **size** komutu kullanarak matrisin boyutları bulunabilir. Gerek vektörlerde gerekse matrislerde vektörün ya da matrisin belirli kısımları alınabilir. Örneğin bir v vektörün 3 üncü ve 21 inci elemanlarını vp vektörüne vp=v(3:21) ile aktarabiliriz. Benzer şekilde A matrisinde isteğimiz kolon ve sütunlar arası başka bir matrise kolaylıkla aktarılabilir.

Örneğin A(2:4,6:7).

Matrislerin transpozu vektörlerde olduğu gibi ‘ işareti ile alınır. A matrislerinin satırlarını sütun, sütunları satır yapmak için A=A’ yeterlidir.

Bunların dışında özel matrisler vardır. Bir matris ones(m,n) komutu ile oluşturulur ve bütün elemanları birdir. Sıfır matrisi zeros(m,n) ile oluşturulur ve bütün elemanları

sıfırdır. Birim matris oluşturmak için eye(m) komutundan yararlanılır. Bu durumda oluşan matrisin boyutu m x m olup köşegenlerinde 1 vardır. Köşegen matris oluşturmak veya herhangi bir matrisin köşegenini almak için diag(A) komutu kullanılır. Şimdi bunlarla ilgili uygulamalara bakalım.

```
close all
clear all

n=3;
m=4;

A=ones(n,m)
B=zeros(m,n)
C=eye(n)

D=[1 2 3;3 4 5; 6 7 8]
KK=diag(D)

v=[ 1 2 3 4]
H=diag(v)
```

```
>> matr
A =
   1   1   1   1
   1   1   1   1
   1   1   1   1
B =
   0   0   0
   0   0   0
   0   0   0
   0   0   0
C =
   1   0   0
   0   1   0
   0   0   1
KK =

   1
   4
   8
v =
   1   2   3   4
H =
   1   0   0   0
   0   2   0   0
   0   0   3   0
   0   0   0   4
```

Matrislerle çarpım yapabilmek için boyutlara dikkat etmek gerekir. A(m x n) B(n x k) matrisleri çarpılırsa C(m x k) boyutunda bir matris elde edilir. Bu C=A*B ile yapılabilir.

```
close all
clear all

A=[1 2 ; 3 4 ; 1 5]
B=[1 4 5 6; 1 9 1 4]

C=A*B
```

>> carp

A =

1	2
3	4
1	5

B =

1	4	5	6
1	9	1	4

C =

3	22	7	14
7	48	19	34
6	49	10	26

1.10 Sparse Matrisler

Sparse matris olarak sıfırı bol olan, bazı elemanları sıfırdan farklı olan matrislere denir. Örnek olarak sonlu elemanlarla veya sonlu farklarla çalışıldığında bu tip matrislerle karşılaşırız. Bunu matlab da bir örnek ile açıklayalım. sparse(i,j,v) komutu ile yapılır. İ inci satır j inci sütun daki v elementinin değeri anlamındadır.

```
close all
clear all

i=[1 3 5]
j=[2 3 4]
v=[10 11 12]

S=sparse(i,j,v)

T=full(S)
```

```
i
=
    1     3     5
j =
    2     3     4
v =
   10    11    12
S =
  (1,2)       10
  (3,3)       11
  (5,4)       12
T =
    0    10     0     0
    0     0     0     0
    0     0    11     0
    0     0     0     0
    0     0     0    12
```

burada **sparse** komutu gereksiz olan sıfırları tutmamak için kullanılır. Örneğin 100 x 100 matriste 100 tane değer varsa 900 tane sıfır olmaktadır bu sıfırları bellekte tutmak akıllıca bir iş değildir.

1.11 Karmaşık sayılar

Matlab ta karmaşık sayılarla işlemler oldukça kolaydır. a+ib şeklinde aşağıdaki işlemler hiçbir değişken belirtmesi gerektirmeden kolaylıkla yapılabilir.

```
close all
clear all
a=3+2i
b=4-7i

abmult=a*b
absum=a+b
abdif=a-b
abdiv=a/b
abpow=a^b
```

```
>> complex
a =
   3.0000 + 2.0000i
b =
   4.0000 - 7.0000i
abmult =
  26.0000 -13.0000i
absum =
   7.0000 - 5.0000i
abdif =
```

```
  -1.0000 + 9.0000i
abdiv =
  -0.0308 + 0.4462i
abpow =
  9.7616e+003 -3.4764e+003i
```

1.12 Döngüler

Diğer dillerde olduğu gibi Matlab'ta da döngü oluşturulabilir. Fortran da DO döngüsüne benzer olarak

```
for i=1:n

  işlem

end
```

şeklindendir. Bu işlem i=1 den n e kadar birer adımla yinelenir. Ayrıca

```
for i=1:n
    for j=1:m
        for k=1:t

  işlem
        end
   end
end
```

şeklinde iç içe for döngüleri açmak mümkündür.

1.13 Mantık

Matlab true ve false işlemlerini sırasıyla 1 ve 0 olarak algılar. Bu işlemler koşul gerektiren durumlarda kullanılır. Örneğin bir x değişkeni için bu işlemler

x = = 2 → x 2 ye eşit mi?
x ~= 2 → x 2 ye eşit değil mi?
x > 2 → x 2 den büyük mü?
x < 2 → x 2 den küçük mü?
x >= 2→ x 2 den büyük veya eşit mi?
x <= 2→ x 2 den büyük veya eşit mi?

```
close all
clear all
x=2;
x == 2
```

```
x ~= 2
x > 2
x < 2
x >= 2
x >= 2
```

Bu işlemler while, if, vb. döngülerde kullanılır.

while (koşul)

end

veya birden fazla iç içe while döngüleri açılabilir.

```
while (koşul1)
    while (koşul2)
      while(koşul3)

        işlem

      end
    end
 end
```

Koşullar if komutunda da aşağıdaki şekilde kullanılır.

```
 if (koşul)
       işlem
  else
      işlem
  end
```

```
if (koşul)
       işlem
     elseif  koşul
      işlem
      else
      işlem
  end
```

1.14 Function

Alt programlar, **function** komutu yardımıyla oluşturulabilir. Örneğin 3 kenar uzunluğu bilinen bir üçgenin alanını hesaplayan alt program yazalım.

$$s = (a+b+c)/2$$
$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Burada a,b ve c üçgenin kenar uzunluklarını göstermektedir.

```
function [A]=alan(a,b,c)
s=(a+b+c)/2;
A=sqrt(s*(s-a)*(s-a)*(s-a));
```

alan(a,b,c) yi üçgen alanı hesap etmek için kullanabiliriz. Bu fonksiyonu kullanmak için matlab komut satırında >> alan(3,4,5) yazdığımızda alt program üçgen alanını hesaplar. Bu alt programı kendi yazdığımız programlarda kullanabiliriz.

1.15 **Input** ve **load**

Input ile klavyeden sayı girmemiz gerekiyorsa kullanmamız gereken bir komut. Örneğin: boy=input('Boy') şeklinde kullanılır. Bizim klavyeden girdiğimiz sayı boy değişkenine atanacaktır.

Load ise herhangi bir veri dosyasının okunmasında kullanılır. Örneğin, veri.dat dosyasında 10 satırdan oluşan 3 sütunlu veri olsun. Bu verileri load veri.dat komutu ile okuyup kullanabiliriz. Bu komuttan sonra veri(10,3) matrisi bellekte kullanıma hazırdır. Save komutu kullanarak veriler herhangi bir dosyaya yazdırılabilir.

>> save dosya_adı.dat değişkenler -ascii

2. Örnekler

1) İlk program

```
%------------------------------
%Bilgi
%Matlabta yaptigim ilk program
%Ertan Peksen
%21 Subat 2007
%------------------------------
close all
clear all

disp('........Merhaba..........')
```

2) close ve clear komutlarının açıklaması

```
a=a+c
```

3) Seri fonksiyonlardan e fonksiyonun hesaplanması.

```
% Begin
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Aciklama
% Bu fonksiyon e ifadesini istenilen hassasiyette hesaplar.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
clear all
format long e
x=input('x değeri: ');
m=input('m dongu sayısı:');
%tolerans ekle sonra')
m_e=exp(x);
top=1;
for ii=1:m
    top=top+(x^ii)/(factorial(ii));
end
disp(m_e)
disp(top)
disp(['top=',num2str(top)])
%End
```

4) sum ve prod komutlarının kullanılması

```
%Begin
%Bu program toplama islemei yapar
% Ertan Peksen  21 Subat 2007
close all
clear all
n=input('n sayısını girin :')
top=0;
carp=1;
carpt=0;
for ii=1:n
    top=top+1;
end

for jj=1:n
    carp=carp*jj;
    carpt=carp+carpt;
end

disp(top)

disp(carp)

top2=sum(1:n)
carp2=prod(1:n)
```

3. Problemler ve çözümleri

3.1 Sorular

1. 10^(x*i) fonksiyonunun gerçel ve sanal kısımlarını aynı grafikte gösteriniz. Burada x değişkenini 1 ile (3/2)*pi arasında 0.01 aralıklarla değişmektedir. i karmaşık sayı anlamındadır. Verileri **save** komutu kullanarak veri.dat dosyasına ASCII formatta yazınız. (**real(y)** ve **imag(y)** komutları sırasıyla y nin gerçel ve sanal kısımlarını alırlar)

2 **while** ve **rand** komutlarını kullanarak her seferinde klavyeden sayı girilerek zar atışları için program yazınız. (Yol gösterme: input ile klaveyeden 1 sayısını girilecek, zar=floor(1+6*rand(n,2)) komutu bize ikili zar sonuçlarını verecektir.)

3 Fibonacci sayıları 1,1,2,3,5,8,13… şeklindedir. Verilen bir n değeri için n inci Fibonacci değerini hesap eden program yazınız. (Yol gösterme: Bilinmesi gereken ilk iki rakamdır, daha sonraki rakamlar kendinden önce gelen iki sayının toplamına eşittir.)

4 ax^2+bx+c=0 denklemini çözen program yazın. Bu programın giriş verileri a,b ve c olup çıkış ise x1 ve x2 dir.

3.2 Çözümler

1)

```
%-------------------------------
close all
clear all

s=1:0.01:(3/2)*pi;
f=10.^(s.*i);

fr=real(f);
fi=imag(f);

save veri.dat s fr fi –ascii

plot(s,fr,'r--')
hold on
plot(s,fi,'k.')
title('Problem 1')
xlabel('x ekseni')
ylabel('y ekseni')
legend('real','imaginery')
print -djpeg prob1
```

2)

```
close all
clear all
```

```
n=2;
while (n ~= 0)
    m=input('Bir sayi giriniz (kac adet ikili istiyorsunuz): ');
    d=floor(1+6*rand(m,2))
    disp('durmak için 0 tusunu kullanınız devam için herhangi bir tusa
basınız.')
    n=input('Devam için 0 haric bir rakam giriniz: ');
end
```

3)

```
close all
clear all
%Fibonacci sayıları hesap eden program
n=input('Kacıncı Fibonacci sayısı:');
f(1:n)=0;
f(1)=1;
f(2)=1;
if n == 1
disp([num2str(n) ' inci fibonacci sayisisi= ' num2str(f(1)) ' dir.']);
elseif n == 2
disp([num2str(n) ' inci fibonacci sayisisi= ' num2str(f(2)) ' dir.']);
else
for ii=3:n
    f(ii)=f(ii-1)+f(ii-2);
end
disp([num2str(n) ' inci fibonacci sayisisi= ' num2str(f(n)) ' dir.']);
end
```

4)

```
%Begin
close all
clear all
%İkinci derece denklemi cozen program
%ax^2+bx+c=0 tipinde bir denklemin
% a, b ve c katsayılarının girimesi gerekir.

a=input('ax^2 nin katsayısını girin:');
b=input('bx in katsayısı girin:');
c=input('c sayısını girin:');

d=b*b-4*a*c;
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);

disp('x1')
disp(x1)
disp('x2')
disp(x2)
```

4. Matlab Fonksiyonları

(Burada verilen komutlar hakkında daha fazla bilgi ve örnek kullanım için help komutunu kullanınız**)**

% Bilgi için kullanılır
; Bir alt satıra geç
help: komutlar hakkında bilgi ve nasıl kullanılacağı hakkında örnek verir.
edit: matlab komut satırında yeni bir dosya açmak veya var olan dosyaya erişmek için kullanılır.
mkdir: alt dizin yaratmak için kullanılır.
rmdir: alt dizin silmek için kullanılır.
dir ve ls: dosyaları listelemek için kullanılır.
pwd: hangi alt dizinde olduğunuzu öğrenmek için kullanılır.
who ve whos: bellekte bulunan değişkenleri gösterir.
clear: bellekte bulunan değişkenleri siler.
close: açılan şekil pencerelerini kapatır.

input: klavyeden programa veri girmek için kullanılır.
disp: ekranda sonuçları göstermek için kullanılır.
format: çalışılan format ayarlarını değiştirmek için kullanılır.
save: verileri kaydetmek için kullanılır.
load: verileri yüklemek için kullanılır**.**
plot: grafik çizmek için kullanılır.
subplot: grafiklerde kullanılan bir komuttur.
xlabel: grafik x ekseni için bilgi yazısı ve değişken birimi için kullanılır.
ylabel: grafik y ekseni için bilgi yazısı ve değişken birimi için kullanılır.
title: grafiğin ismi
legend: hangi grafiğin hangi işaretle gösterildiğini açıklayan küçük kutu.
print ekranda gösterilen şekli istenilen formatta kayıt etmek için kullanılır.
size: matris boyutlarını bulmak için kullanılır**.**
sort: vektörü küçükten büyüğe sıralar.
dot: skaler çarpım**.**
cross: vektörel çarpım**.**
zeros: sıfır matrisi veya vektörü oluşturmak için kullanılır**.**
eye: birim matris oluşturmak için kullanılır**.**
ones: bir matrisi veya vektörü oluşturmak için kullanılır**.**
diag: matrisin köşegen elemanlarıyla işlem yapmak veya köşegen matris oluşturmak için kullanılır**.**
sparse: sıfırı çok olan matrislerle işlem yaparken kullanılır. Sıfırları bellekte tutmamak için kullanılır.
full: sparse matrisi tam matrise dönüştürür
if : koşullarda uygulana eğer komutu.
for: döngü kurmak için kullanılır (koşulsuz).
while: döngü kurmak için kullanılır (koşullu).
floor: verilen bir reel sayıyı doğal sayıya çevirmek için kullanılır. Ayrıca **ceil** ve **round** komutları da benzer işlemi yaparlar fakat yuvarlatma işleminde farklılıklar vardır.

real: bir karmaşık sayısının reel kısmını almak için kullanılır**.**
imag: bir karmaşık sayının sanal kısmını almak için kullanılır**.**
rand: 0 ile 1 arasında gelişigüzel sayı üretmek için kullanılır**.**
function: alt program yazmak için kullanılır**.**
sum: girilen vektörleri toplar, matris olursa ilk önce satırları toplar eğer sum iki kez kullanılırsa bütün matris elemanları toplanmış olur.
prod: girilen vektörleri çarpar, matris olursa ilk önce satırları çarpar eğer prod iki kez kullanılırsa bütün matris elemanları çarpılmış olur olur.
factorial: faktöriyel almak için kullanılır**.**

BÖLÜM 2

1. Gelişigüzel sayı üretme

rand(n,m): rand komutu 0 ile 1 arasında gelişigüzel sayılar üretir.

Örnekler:

```
>> rand(1)

ans =

   0.7828

>> rand(2)

ans =

   0.9341   0.5185
   0.2213   0.7974

>> rand(1,2)

ans =

   0.0732   0.9448

>> rand(2,1)

ans =

   0.1415
   0.3763
```

floor: Gerçel ve sanal sayılardaki virgülden sonraki kısımları atmaya yarar. Bunu yaparken, söz konusu sayının bulunduğu iki tam sayıdan küçük olanına çevirir. Floor a benzer komutlardan **ceil** ve **round** da küçük farklılıklarla benzer işlemler yapmak için kullanılır.

Örnekler:

```
>> floor(1.2)

ans =

    1

>> floor(1.7)
```

```
ans =

    1

>> ceil(1.2)

ans =

    2

>> ceil(1.7)

ans =

    2

>> round(1.2)

ans =

    1

>> round(1.7)

ans =

    2
```

2. Alt program

Matlabta alt program yazarken, alt programın ilk satırı

function [cıktılar]=isim(girdiler)

şeklindedir. Alt program ismi satırda kullandığınız isimle aynı olmalıdır. Örnek function kullanımı için problem 2 nin çözümüne bakınız.

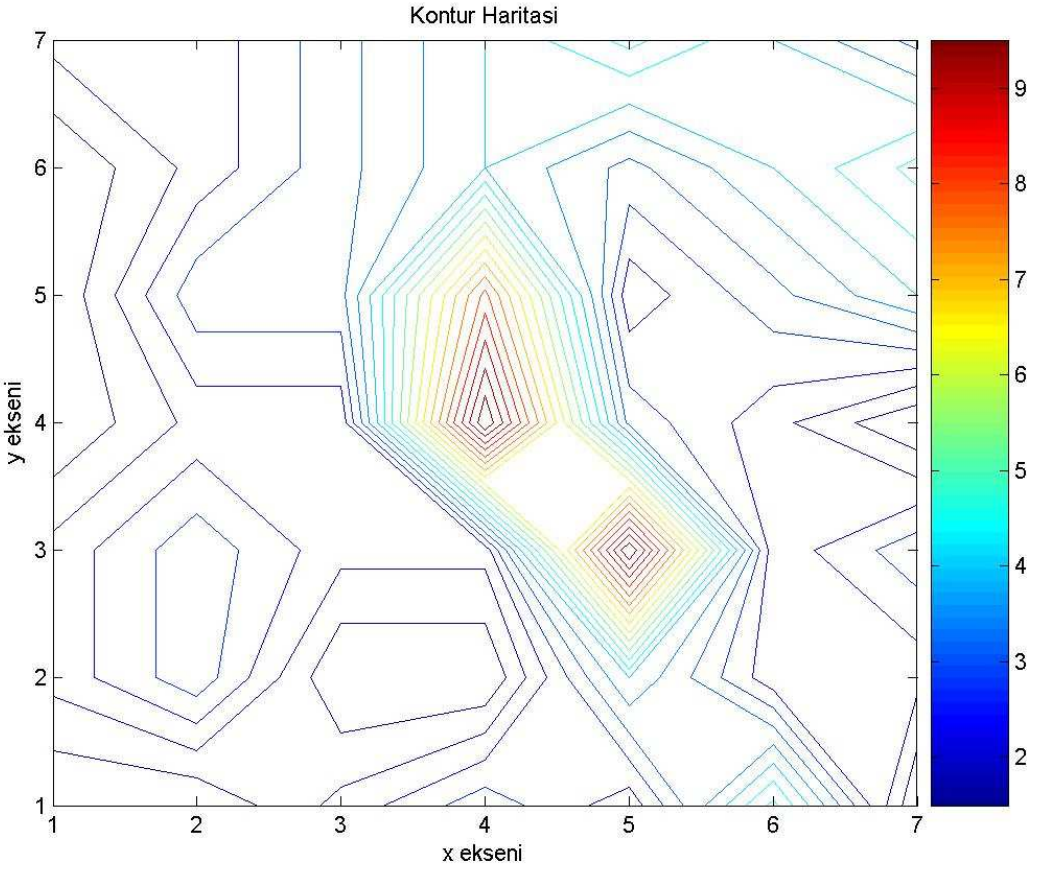
3. Kontur çizimleri

contour ve mesgrid

Yer bilimcileri olarak kontur çizmek bizler için önemlidir ve çoğunlukla kullanırız. Genelde surfer ve benzeri programlar kullanılır. Matlabta konturlar **contour** komutuyla yapılabilir. Contour komutunu kullanmadan önce uygun matris değerlerinin üretilmesi

gerekir. Matris değerleri genelde yer bilimlerinde ölçü alınan noktanın koordinatları olarak düşünülebilir. Yeryüzünü x ve y düzlemi kabul edersek, x-y düzleminde ölçü alınan noktalar
meshgrid komutuyla üretilebilir. Normalde arazide nerelerde ölçü alınırsa bu değerler zaten elimizde vardır. Örneklerle bu komutları anlamaya çalışalım.

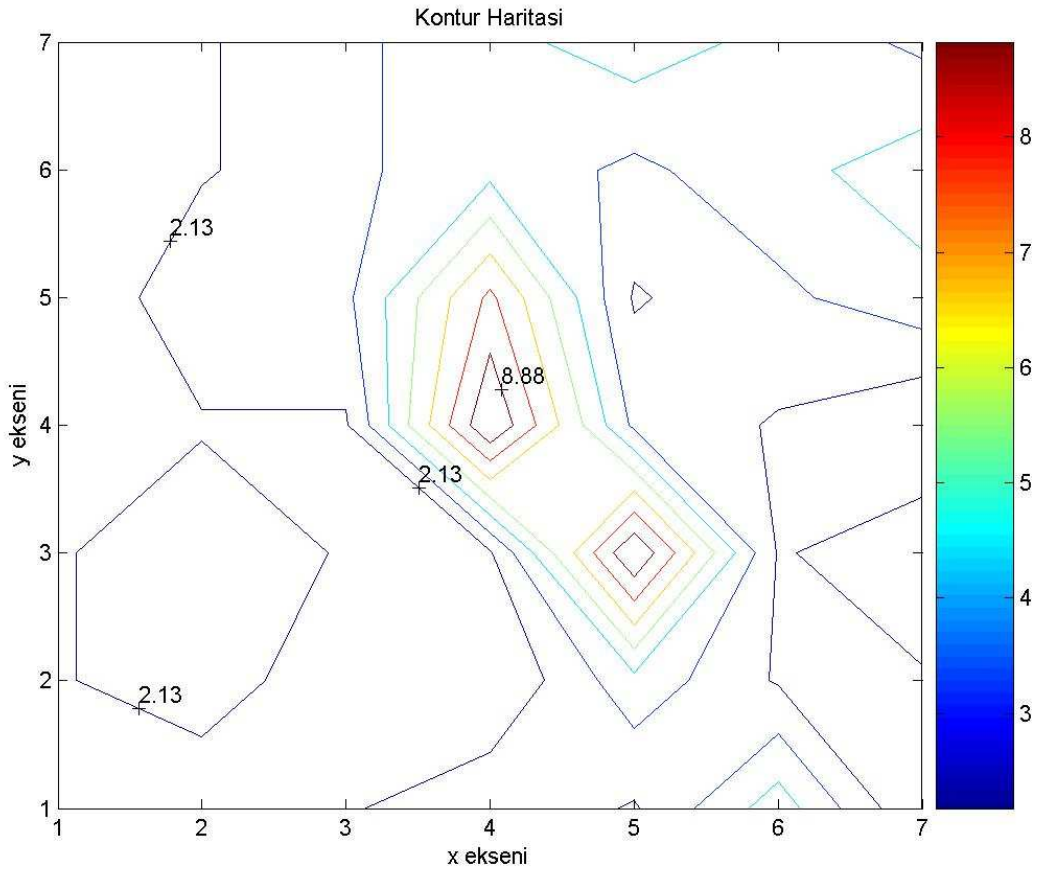
```
close all
clear all

x=1:7;
y=1:7;

[X,Y]=meshgrid(x,y);

Z=[1 1 2 3 2 5 1;
   2 3 1 1 4 2 2;
   2 3 2 2 10 2 3;
   1 2 2 10 3 2 1;
   1 3 3 8 2 3 4;
   1 2 3 4 3 4 5;
   2 2 3 4 5 4 3 ];

contour(X,Y,Z,20);
colorbar

title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')

%print -djpeg kontur.jpg
```



şekli elde edilir. Contour(X,Y,Z,20) X,Y koordinatlarına karşılık gelen Z değerlerinin kontur haritasıdır. Bunların matris boyutları birbirlerine eşit olmalıdır. X ve Y 10 x 10 boyutunda ise Z de 10 x 10 boyutunda olmalıdır. 20 sayısı ise konturların kaç tane olması gerektiğini tanımlar. Konturların daha sık veya seyrek olmasını bu sayı ile belirtebiliriz. Aşağıdaki örnekte clabel komutu kullanarak nasıl kontur değerlerinin üzerine yazıldığını görebiliriz. Eğer clabel(C,'manual') komutu kullanılırsa fare ile kontur çizgileri üzerinde istediğimiz konturun değerini yazabiliriz.

```
close all
clear all

x=1:7;
y=1:7;

[X,Y]=meshgrid(x,y);

Z=[1 1 2 3 2 5 1;
   2 3 1 1 4 2 2;
   2 3 2 2 10 2 3;
   1 2 2 10 3 2 1;
   1 3 3 8 2 3 4;
```

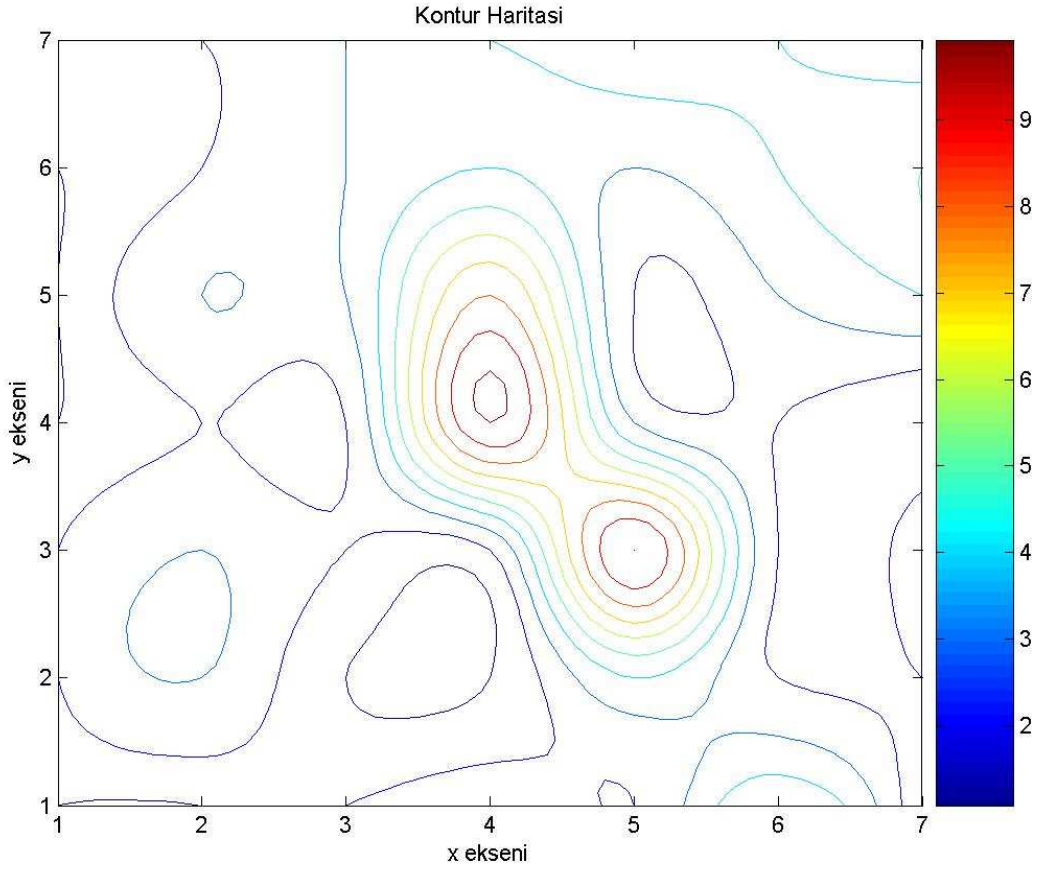
```
   1 2 3 4 3 4 5;
   2 2 3 4 5 4 3 ];

C=contour(X,Y,Z,7);
clabel(C)
colorbar

title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')
```



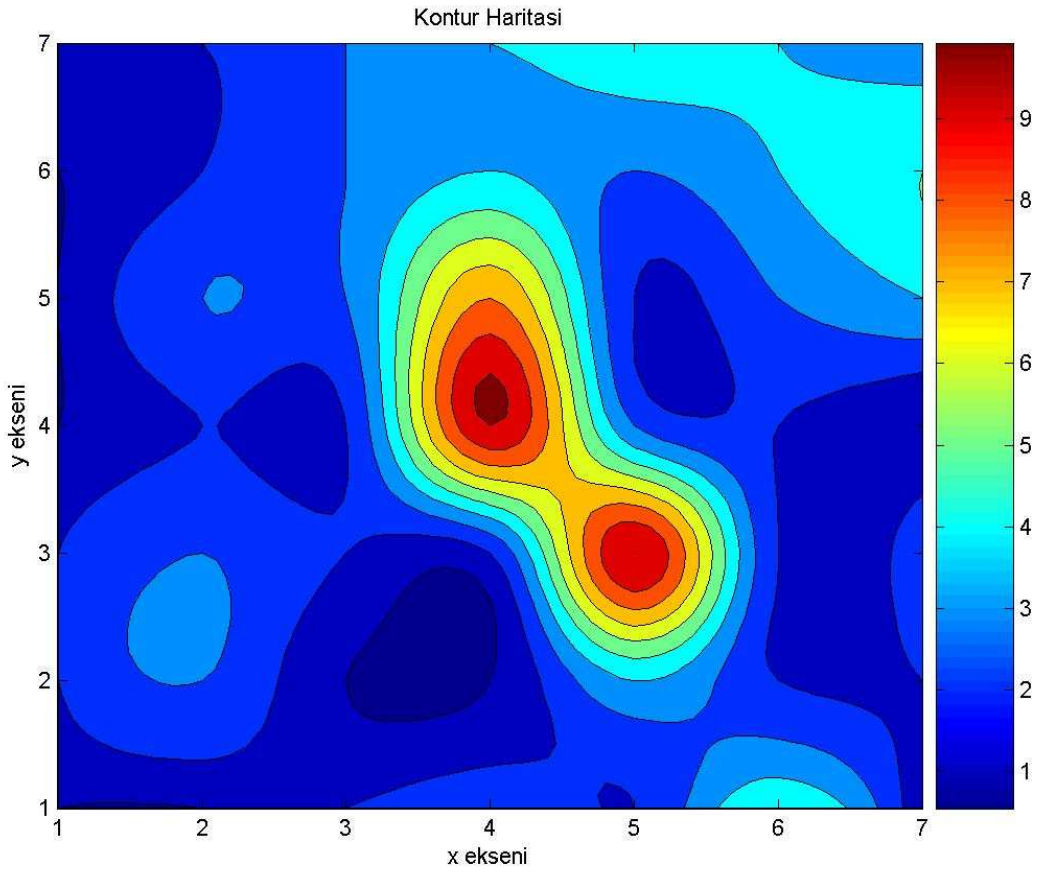
Kontur kenarlarını yumuşatmak için interp2 komutu kullanılır. Aşağıdaki örneği inceleyelim

```
close all
clear all

x=1:7;
y=1:7;
[X,Y]=meshgrid(x,y);
Z=[1 1 2 3 2 5 1;
```

```
    2 3 1 1 4 2 2;
    2 3 2 2 10 2 3;
    1 2 2 10 3 2 1;
    1 3 3 8 2 3 4;
    1 2 3 4 3 4 5;
    2 2 3 4 5 4 3 ];
[xi,yi]=meshgrid(1:0.1:7,1:0.1:7);
zi=interp2(X,Y,Z,xi,yi,'cubic');
contour(xi,yi,zi);
colorbar

title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')
```

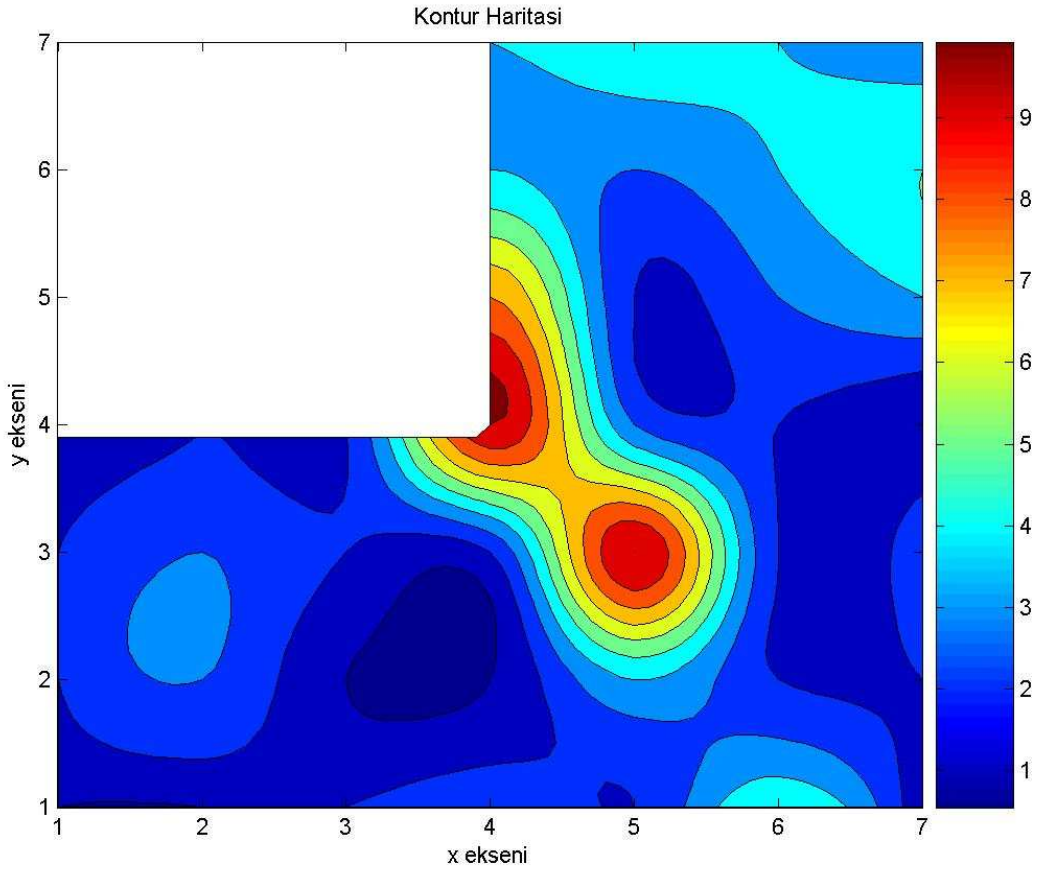


```
close all
clear all
x=1:7;
y=1:7;
[X,Y]=meshgrid(x,y);
Z=[1 1 2 3 2 5 1;
   2 3 1 1 4 2 2;
   2 3 2 2 10 2 3;
   1 2 2 10 3 2 1;
   1 3 3 8 2 3 4;
   1 2 3 4 3 4 5;
   2 2 3 4 5 4 3 ];
[xi,yi]=meshgrid(1:0.1:7,1:0.1:7);
zi=interp2(X,Y,Z,xi,yi,'cubic');
contourf(xi,yi,zi);
colorbar
title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')
```



Bazı durumlarda ölçü alınan arazi tam kare olmayabilir. Ölçü sahasında bina ve benzeri yapılardan dolayı ölçü alamadığımız yerler için, z matrisinde ölçü yerlerine **NaN** yazmamız yeterlidir. Bu bir sayı değildir anlamındadır.

```
close all
clear all
x=1:7;
y=1:7;
[X,Y]=meshgrid(x,y);
Z=[1 1 2 3 2 5 1;
   2 3 1 1 4 2 2;
   2 3 2 2 10 2 3;
   1 2 2 10 3 2 1;
   1 3 3 8 2 3 4;
   NaN NaN 3 4 3 4 5;
   NaN NaN 3 4 5 4 3];
[xi,yi]=meshgrid(1:0.1:7,1:0.1:7);
zi=interp2(X,Y,Z,xi,yi,'cubic');
contourf(xi,yi,zi);
colorbar
title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')
```

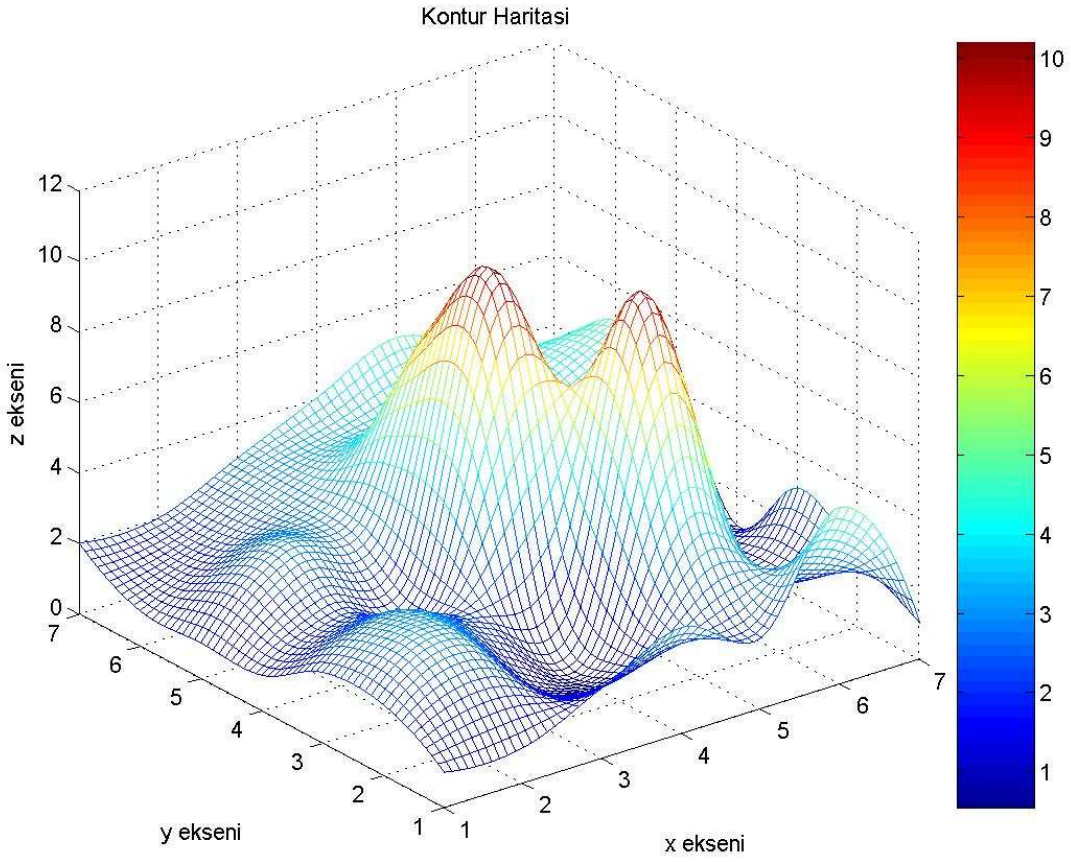


mesh komutunu kullanarak yukarıdaki örnek iki boyutlu çizimleri üç boyutlu olarak çizmeye çalışalım.

```
close all
clear all
x=1:7;
y=1:7;
[X,Y]=meshgrid(x,y);

Z=[1 1 2 3 2 5 1;
   2 3 1 1 4 2 2;
   2 3 2 2 10 2 3;
   1 2 2 10 3 2 1;
   1 3 3 8 2 3 4;
   1 2 3 4 3 4 5;
   2 2 3 4 5 4 3 ];
[xi,yi]=meshgrid(1:0.1:7,1:0.1:7);
zi=interp2(X,Y,Z,xi,yi,'cubic');
mesh(xi,yi,zi)
```

```
colorbar
title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')
zlabel('z ekseni')
rotate3d
```

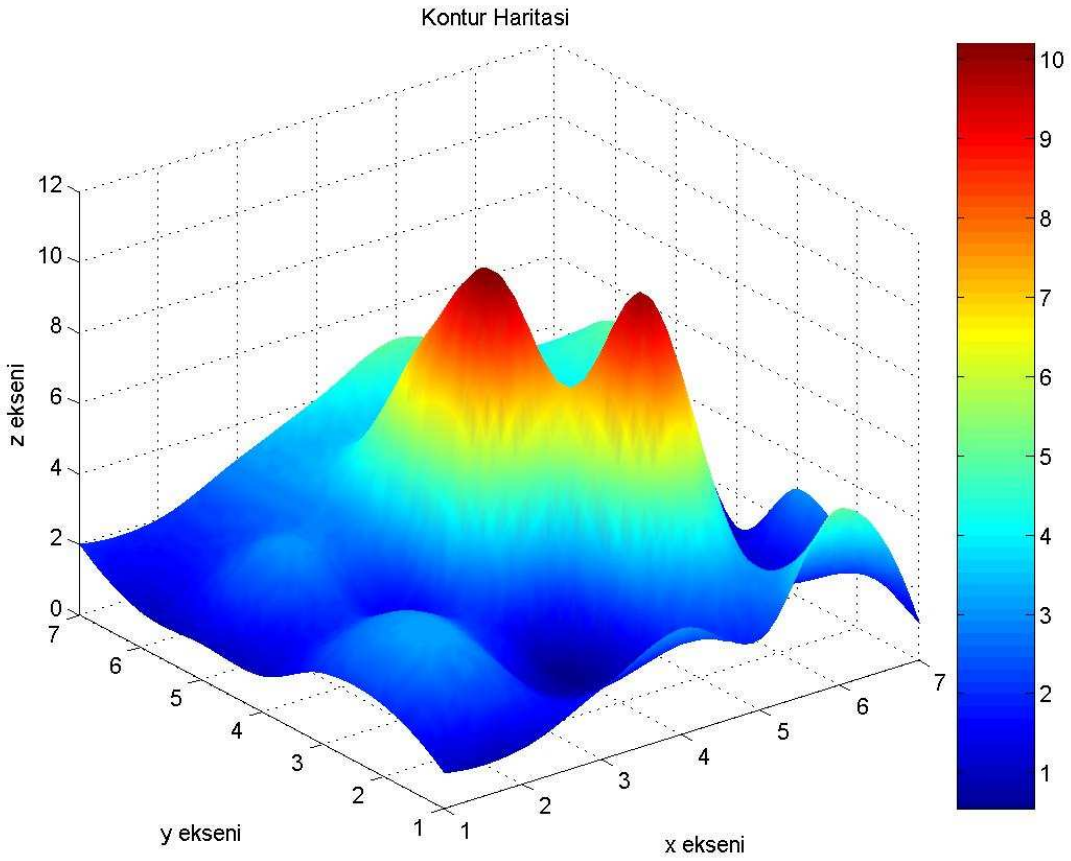
Bu programı çalıştırdığımızda aşağıdaki şekil elde edilecektir. Rotate3d komutu ile bu fonksiyonu istediğimiz açıdan fare yardımıyla bakabiliriz.



mesh yerine **meshc** komutuda kullanılabilir bu durumda, şekil aynı olacak ve xy düzlemine kotur çizilecektir. Sadece **mesh** komutu yerine **meshc** komutu yazarak denebilir**. Surf** ve **surfc** komutları da **mesh** komutu gibi kullanılır.

```
close all
clear all
x=1:7;
y=1:7;
[X,Y]=meshgrid(x,y);
Z=[1 1 2 3 2 5 1;
   2 3 1 1 4 2 2;
   2 3 2 2 10 2 3;
```

```
    1 2 2 10 3 2 1;
    1 3 3 8 2 3 4;
    1 2 3 4 3 4 5;
    2 2 3 4 5 4 3 ];
[xi,yi]=meshgrid(1:0.1:7,1:0.1:7);
zi=interp2(X,Y,Z,xi,yi,'cubic');
surf(xi,yi,zi)
colormap jet
shading interp
colorbar
title('Kontur Haritasi')
xlabel('x ekseni')
ylabel('y ekseni')
zlabel('z ekseni')
rotate3d
```



Problemler:

1 **while** ve **rand** komutlarını kullanarak her seferinde klavyeden sayı girilerek zar atışları için program yazınız. (Yol gösterme: input ile klaveyeden 1 sayısını girilecek, zar=floor(1+6*rand(n,2)) komutu bize ikili zar sonuçlarını verecektir.)

2 ax^2+bx+c=0 denklemini çözen program yazın. Bu programın giriş verileri a,b ve c olup çıkış ise x1 ve x2 dir.

Çözümler:

1) zar (rand komutu)

çözüm:

```
close all
clear all
n=2;
while (n ~= 0)
   m=input('Birsayi giriniz (kac adet ikili istiyorsunuz): ');
   d=floor(1+6*rand(m,2));
   disp('-----------')
   disp('Zar sonuclari:')
   disp(d)
   disp('-----------')
   disp('programı durdurmak için 0 tusuna basınız devam icin 0 haric herhangi bir tusa
basınız.')
   n=input('devam ya da bitir: ');
end
disp('Program durdu.')
```

```
close all
clear all
n=0;
while (n == 0)
   m=input('Birsayi giriniz (kac adet ikili istiyorsunuz): ');
   d=floor(1+6*rand(m,2));
   disp('-----------')
   disp('Zar sonuclari:')
   disp(d)
   disp('-----------')
   disp('programı durdurmak için 0 haric herhangi bir tusa basınız.')
   n=input('devam ya da bitir: ');
end
disp('Program durdu.')
```

2) ikinci derece denklemi (alt program)

```
%Begin
```

```
close all
clear all
%İkinci derece denklemi cozen program
%ax^2+bx+c=0 tipinde bir denklemin
% a, b ve c katsayılarının girimesi gerekir.

a=input('ax^2 nin katsayısını girin:');
b=input('bx in katsayısı girin:');
c=input('c sayısını girin:');

d=b*b-4*a*c;
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);

disp('x1')
disp(x1)
disp('x2')
disp(x2)
```

```
%Begin
close all
clear all
%İkinci derece denklemi cozen program
%ax^2+bx+c=0 tipinde bir denklemin
% a, b ve c katsayılarının girimesi gerekir.

a=input('ax^2 nin katsayısını girin:');
b=input('bx in katsayısı girin:');
c=input('c sayısını girin:');

[x1,x2]=coz(a,b,c);

disp('Kokler')
disp('--------------')
disp(x1)
disp(x2)
```

alt program

```
function [x1,x2]=coz(a,b,c)
d=b*b-4*a*c;
x1=(-b+sqrt(d))/(2*a);
x2=(-b-sqrt(d))/(2*a);
```

BÖLÜM 3

3. 1 Formatlı okuma ve yazma

Bu hafta dersimizde formatlı yazma ve formatlı okuma komutlarını öğreneceğiz. Dosya açmak için fopen komutu kullanılır. Örnek olarak çalıştığınız alt dizinde orbits.dat verileri olsun. Bu verileri okumak en kolay şekilde load komutuyla yapıldığını önceki derslerde kısaca bahsetmiştik. Orbits.dat aşağıdaki şekilde olsun. Type komutuyla orbits.dat ın içeriği görülebilir.

```
>> type orbits.dat

1.23 0.43
2.45 1.56
3.78 3.89
```

Herhangi bir dat dosyasındaki verileri

```
>> load('orbits.dat')
>> load orbits.dat
```

komutları ile okuyabiliriz. Save komutunu verileri kaydetmek için kullanabiliriz. **Save dosya ismi değişkenler –ascii** komutuyla isteğimiz isimde -ascii formatında ya da –mat vb. formatlarda dosyaya yazılabilir.

```
>> xk=1:4

xk =

    1     2     3     4

>> orbits

orbits =

   1.2300    0.4300
   2.4500    1.5600
   3.7800    3.8900

>> save my.out orbits xk -ascii
>> type my.out

  1.2300000e+000  4.3000000e-001
  2.4500000e+000  1.5600000e+000
  3.7800000e+000  3.8900000e+000
  1.0000000e+000  2.0000000e+000  3.0000000e+000  4.0000000e+000
```

Eğer verileri yazdırmayı isteğiniz dosyanın tür ekini belirtmez iseniz bu otomatik olarak *.mat olarak okunma veya yazma anlamındadır. Örneği inceleyelim, dosyamızın tür ekini mat olarak kayıt edip okuyalım (default (varsayılan) değeri).

```
>> save my.mat orbits xk
```

veya
>> save my orbits xk

Şimdi çalıştığımız alt dizinde olan dosyaları ls veya dir komutuyla görebiliriz.

>> ls

. .. my.mat my.out orbits.dat

Daha sonra belleği clear komutuyla temizleyelim.

>>clear all

Sonra verileri dosyadan okuyalım.

>> load my

load komutu kullanıldığında matlab otomatik olarak tür eki mat olan dosyayı okuyacaktır.
Değişkenleri görmek için whos komutu kullanılabilir.

```
>> whos
  Name        Size                Bytes  Class

  orbits      3x2                    48  double array
  xk          1x4                    32  double array
```

Denemek için orbits ve xk değişkenlerini yazıp ekranda görebiliriz.

```
>> orbits

orbits =

    1.2300    0.4300
    2.4500    1.5600
    3.7800    3.8900

>> xk


xk =

     1     2     3     4
```

>>whos –file my.mat dediğimizde sadece bu dosya içindeki değişkenler ekranda görünecektir.

>> load my.out komutu çalışmayacaktır.

My.out dosyasında satır ve sütun sayıları farklı biri matris diğeri vektör iki veri grubu bulunduğu için çalışmaz.

Şimdi aşağıdaki programla orbits.dat ı formatlı şekilde okuyalım.
Aşağıdaki programın ismi den1.m dir.

```
close all
clear all

fid=fopen('orbits.dat','r');

Orbit=fscanf(fid,'%f',[2 Inf])
Orbit=Orbit'

fclose(fid);
```

Programı çalıştırdığımızda

```
>> den1

Orbit =

   1.2300    2.4500    3.7800
   0.4300    1.5600    3.8900


Orbit =

   1.2300    0.4300
   2.4500    1.5600
   3.7800    3.8900
```

elde ederiz. Burada %f yerine aşağıdaki semboller de gelebilir.

%d Tam sayı
%f reel sayılar
%e üstel sayılar
%s karakter vb.

Ayrıca %4f ile sayının basamak sayısını belirtebiliriz. Yalnız burada bildirilen rakam, en büyük basamak sayısını içerdiği için dikkat edilmesi gerekir. Örnek olarak bizim orbits.dat ı doğru okuyabilmek için her ne kadar 1.23 gibi üç basamaklı sayı gibi görünse de format olarak en az 5 byte a ihtiyac vardır. İki basamak . ve işaret (- ve +) için bellekte yer ayırmayı unutmayınız. Aksi taktirde formatlı okuma hata yapacaktır. Örnek olarak den1.m dosyasındaki %f ifadesini sırasıyla %2f, %3f, %4f ve %5f olarak deneyiniz.

Fscanf komutundaki ‘r’ read oku anlamındadır.

Fprintf komutunu inceleyelim. Aşağıdaki program x ve y vektörlerini fddeneme.dat dosyasına yazmak için kullanılabilir.

Ayrıca fprintf komutunun içinde değişkenlerin format tipi ve virgülden sonra kaç basamak istediğimizi de belirtebiliriz.

```
close all
clear all

x=1:10;
y=3.5*x;
fid=fopen('fpdeneme.dat','wt');

n=length(x);

for ii=1:n
   fprintf(fid,' %5.1f   %5.3f \n', x(ii), y(ii));
end


fclose(fid);
```

Bu programı çalıştırdığımızda sonuç fpdeneme.dat dosyasına yazılacaktır. Sonucu görmek için type komutu kullanılabilir.

```
>> type fpdeneme.dat

   1.0   3.500
   2.0   7.000
   3.0   10.500
   4.0   14.000
   5.0   17.500
   6.0   21.000
   7.0   24.500
   8.0   28.000
   9.0   31.500
  10.0   35.000
```

Ya da verileri ekrana formatlı olarak

```
close all
clear all

x=1:10;
y=3.5*x;

n=length(x);

for ii=1:n
   fprintf(1,' %5.1f   %5.3f \n', x(ii), y(ii));
end
```

Bu dosyayı prog1 olarak kayıt edip programı çalıştırırsak sonuç aşağıdaki şekildedir.

```
>> prog1
   1.0   3.500
   2.0   7.000
   3.0   10.500
   4.0   14.000
   5.0   17.500
   6.0   21.000
   7.0   24.500
   8.0   28.000
   9.0   31.500
  10.0   35.000
```

3.2 switch komutu

Switch komutuyla birden çok seçenekli durumlarla karşılaştığımızda aşağıdaki şekilde kullanabiliriz. Bu işlemleri if komutuyla da yapmak mümkündür. Komutun yapısı aşağıdaki şekildedir.

```
switch durum( bir sayı veya yazı olabilir)
   case 1
    case2
     case n
   otherwise
end
```

örnek 1:

```
close all
clear all
gun=input('gun:');

switch gun;
    case 1
        gun='pazartesi'
    case 2
        gun='salı'
    case 3
        gun='carsamba'
    case 4
       gun='persembe'
    case 5
        gun='cuma'
    case 6
        gun='cumartesi'
    otherwise
        gun='pazar'
end
```

```
>> swden
```

```
gun:7

gun =

pazar

>> swden
gun:6

gun =

cumartesi
```

örnek:2

```
close all
clear all

method = input('gir:','s');

switch method
   case {'ertan','peksen'}
      disp('İsim veya soy isim dogru')
   case 'ali'
      disp('Ali ismi dogrudur.')
   case 'veli'
      disp('Veli ismi dogrudur')
   otherwise
      disp('Boyle bir isim bulunamadı')
end
```

programı dent.m olarak kayıt edip çalıştıralım. Klavyeden girilen ismin doğru olup olmadığını kontrol ediyor.

```
>> dent
gir:ertan
İsim veya soy isim dogru
>> dent
gir:peksen
İsim veya soy isim dogru
>> dent
gir:hasan
Boyle bir isim bulunamadı
```

Şeklinde sonuçlar görmeliyiz.

3.3 Çok sık kullanılan önemli komutlar

max(x), **min(x)**, **sort(x)**,**prod**(x),**sum**(x), **mean**(x),**median**(x), **conv**(x) ve **std**(x) komutları işlerimizi oldukça kolaylaştırmaktadır. Örneklerle bu komutları inceleyelim.

Min ve max

```
>> x=[4 2 5 7 1 8 9]

x =

   4    2    5    7    1    8    9

>> max(x)

ans =

   9

>> min(x)

ans =

   1
```

Eğer bir matrisin en büyük değeri bulunmak isteniyorsa

```
> A=rand(4,4)

A =

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

>> max(max(A))

ans =

   0.9501

>> min(min(A))

ans =

   0.0185
```

şeklinde kullanılır.

sort

```
>> x

x =

   4    2    5    7    1    8    9

>> sort(x)
```

```
ans =

    1     2     4     5     7     8     9
```

prod

```
>> x=[1 2 3 4]

x =

    1     2     3     4

>> prod(x)

ans =

   24

>> mean(x)

ans =

   2.5000

>> sum(x)

ans =

   10
```

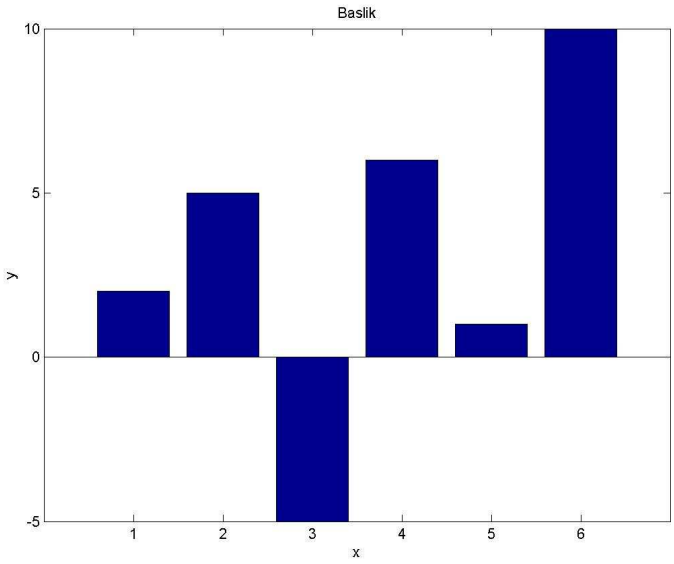
mean ve median

```
>> x=[1 1.3 1.5]

x =

   1.0000    1.3000    1.5000

>> mean(x)

ans =

   1.2667

>> median(x)

ans =

   1.3000

>> std(x)
```

```
ans =

   0.2517

>> x

x =

   1.0000    1.3000    1.5000
-------------------------------------------------------

>> a=[1 2 3]

a =

    1     2     3

>> b=[1 2]

b =

    1     2

>> c=conv(a,b)

c =

    1     4     7     6
```

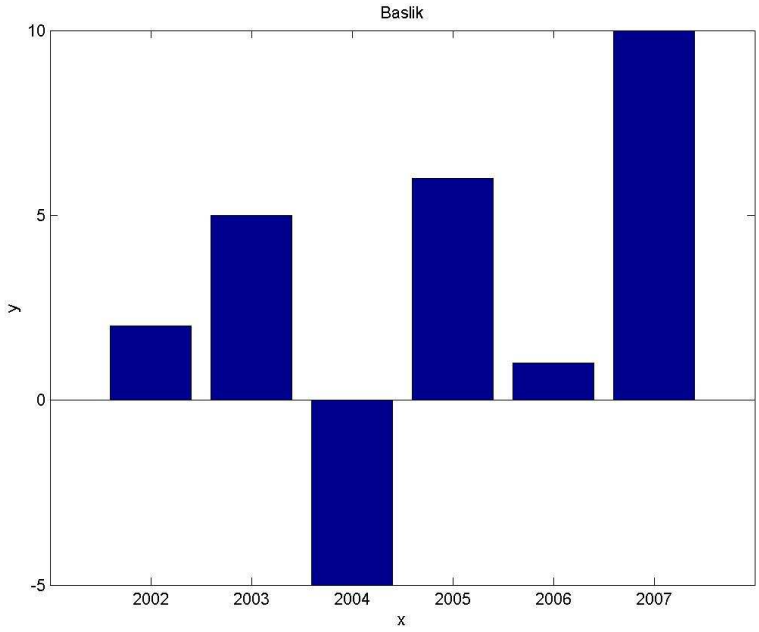
3.4 bar ve pie grafikleri

```
>> A=[ 2 5 -5 6 1 10]
A =
    2     5    -5     6     1    10
>> bar(A)
```



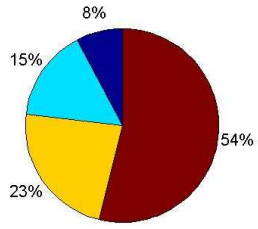
veya daha özel olarak yatay eksen yılları göstermek üzere bar komutu şu şekilde kullanılabilir.

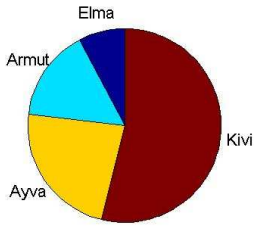
```
close all
clear all

A=[ 2 5 -5 6 1 10];
x=2002:2007;
bar(x,A);
xlabel('x')
ylabel('y')
title('Baslik')
```

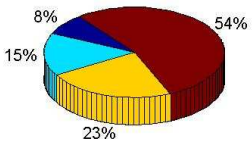


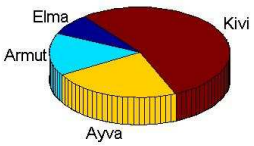
pie komutu ise aşağıdaki şekilde kullanılabilir.

```
close all
clear all

V=[10 20 30 70]
subplot(2,2,1)
pie(V)
subplot(2,2,2)
pie(V,{'Elma','Armut','Ayva','Kivi'})
subplot(2,2,3)
pie3(V)
subplot(2,2,4)
pie3(V,{'Elma','Armut','Ayva','Kivi'})
```


8%
15%
54%
23%


Elma
Armut
Kivi
Ayva


8%
54%
15%
23%


Elma
Kivi
Armut
Ayva

BÖLÜM 4

4.1 Sayısal İntegral

Aşağıdaki basit fonksiyonu ele alalım.

$$y(x)=\int_0^3 e^{2x}dx$$

Bağıntının integrali aşağıdaki şekilde hesaplanabilir.

$$Y(x)=\frac{1}{2}e^{2x}\Big|_0^3=\frac{1}{2}\left(e^6-1\right)=201{,}214=201{,}2143967463676...$$

elde edilir. Bazı durumlarda, verilen fonksiyonun integrali kolayca alınamaz. Bu gibi durumlarda çoğunlukla fonksiyon verilen aralıkta sayısal olarak hesaplanabilir. Matlab ta sayısal integral alma **quad**, **quadl** komutlarıyla yapılabilir. Aşağıdaki programda bu analitik fonksiyonu Matlab ın sayısal integral alma komutunun sonucuyla karşılaştıracağız. Ana program aşağıdaki şekildedir.

```
close all
clear all

a=0;
b=3;
integral_analtic=0.5*(exp(6)-1)
integral_1=quad('growth',a,b)
integral_2=quadl('growth',a,b)
```

şeklindedir.

Alt program (function) ise

```
function y=growth(x)
y=exp(2*x);
```

ile verilmiştir.

Eğer ana programa prob_int ismi verilmiş ise program çalıştırıldığında

```
>> prob_int
integral_analtic =
   2.012143967463676e+002
integral_1 =
   2.012143967613353e+002
integral_2 =
   2.012143967464380e+002
```

değerleri elde edilir. Sonuçlardan anlaşılacağı üzere Matlab ın sayısal integral komutları oldukça iyi sonuçlar vermektedir.

İkili integral **dblquad** komutu ile aşağıdaki şekilde sayısal olarak hesaplanabilir.

Ana program

```
close all
clear all

Q = dblquad(@integrnd,0,3,0,3)
```

function

```
function z = integrnd(x, y)
z =exp(2*x+2*y);
```

program çalıştırıldığında aşağıdaki sonucu üretir.

```
Q =

    4.048723345802940e+004
```

4.2 Üçlü integral

triplequad komutuyla aşağıdaki şekilde hesaplanabilir.

Ana program

```
close all
clear all

tic
Q = triplequad(@integrnd3,0,0.5,0,0.5,0,0.5)
toc
```

Alt program

```
function z = integrnd3(x, y, z)
z =exp(2*x+2*y+2*z);
```

sonuç

```
Q =

    6.341517673266393e-001
```

tic ve **toc** komutları ile programda geçen süreyi hesaplayabilirsiniz.

4.3 Hareketli Görüntü

Matlabta hareketli görüntülerde getframe ve movie komutları kullanılır. Genel olarak hareketli bir görüntü

```
for j=1:n
       cizim komutları (plot, surf …)
       M(j) = getframe;
     end
 movie(M)
```

program yapısında elde edilebilir.

Aşağıdaki örneği inceleyelim.

```
close all
clear all
x=0:0.1:2*pi;
jj=0;
y1=sin(x);
for ii=1:100
    jj=jj+0.1;
y1=sin(x-jj);
plot(x,y1);
M(ii)=getframe;
end
movie(M)
```

Bu programı çalıştırdığımızda ekranda sinüs eğrisinin hareket ettiğini görebiliriz. Benzer şekilde üç boyutlu bir fonksiyon aşağıdaki programı çalıştırmakla elde edilebilir. En son satırda kullanılan, movie2avi komutu filmi avi formatında deneme dosyasına kayıt eder. Bu dosyanızı artık Windows Media Player dan izleyebilirsiziniz.

```
close all
clear all

Z = peaks;
surf(Z);
axis tight
set(gca,'nextplot','replacechildren');

for j = 1:5
    surf(sin(2*pi*j/5)*Z,Z);
    F(j) = getframe;
end

movie(F,5)
%movie2avi(F,'deneme.avi')
```

4.4 Fare Kullanımı

Matlabta fare (Mouse) **ginput** komutuyla kullanılır. Ayrıca **fill** komutuyla fare ile çizdiğimiz şekli boyayabiliriz. Jeofizik modellemede, örneğin Talwani yöntemiyle hesaplanan gelişigüzel şekilli yapıların gravite ve manyetik anomali değerlerini hesap eden program yazmak istediğimizde bu komutları kullanabiliriz. Aşağıdaki örnek programı inceleyelim.

```
close all
clear all
n=input('kac koseli:')
axis([0 10 0 10])
hold on
for  ii=1:n
    [xi,yi] = ginput(1);
    plot(xi,yi,'ro')
    xx(ii)=xi;
    yy(ii)=yi;
end
axis([0 10 0 10])
plot(xx,yy,'k');
plot([xx(1) xx(end)],[yy(1) yy(end)],'k');
fill(xx,yy,'r')
```

Örnek:

$$E = \frac{r}{3\varepsilon_0}\rho_v a_r \qquad 0 < r \le a \quad (1)$$

$$E = \frac{a^3}{3\varepsilon_0 r^2}\rho_v a_r \qquad r > a \quad (2)$$

Yukarıdaki bağıntılar içi dolu bir kürenin merkezinden r kadar uzaklıkta elektrik alan hesap edilmesinde kullanılabilir. (1) ve (2) bağıntılarını kullanarak elektrik alanı küre içinde ve dışında hesap ediniz ve grafiğini çiziniz? (a=10 m, r=0-100 m, $\varepsilon_0 = 8.854\times10^{-12}$ F/m, $\rho_v = 1.3\times10^{-9}$ C/m^3)

Çözüm:

```
close all
clear all
epsilon_0=8.854e-12;
%a=input('a değerini giriniz')
a=10;
r=0:1:100;
rho_v=1.3e-9;
[n,m]=size(r);
for ii=1:m
if r(ii) > a
```

```
E(ii)=(a*a*a*rho_v)/(3* epsilon_0*r(ii)*r(ii));
else
E(ii)=(r(ii)*rho_v)/(3* epsilon_0);
end
end
plot(r,E)
title('Yüklü küre içinde ve disinda elektrik alan')
xlabel('r (m)')
ylabel('E (V/m)')
grid
```

BÖLÜM 5

5. Sembolik Matlab
5.1 Sembolik Türev

Matlab sayısal hesaplamaların yanı sıra sembolik hesaplamalar da yapılabilir. Örnek olarak çok uzun fonksiyonların türev, limit ya da integrallerini almak için kullanılabilir. Syms komutu kullanarak ilk önce sembolik değişken tanımlanır. Aşağıdaki örnekte f=sin(5x) fonksiyonun nasıl türevinin alındığını inceleyelim. Programın ismi sem1.m olsun.

```
close all
clear all

syms x
f=sin(5*x);
d=diff(f)
```

```
>>sem1

d =

5*cos(5*x)
```

olarak elde edilir.

İkinci program aşağıdaki şekilde yine çarpımın türevi uygulanarak $e^x \cos(x)$ fonksiyonun türevini almaktadır. (programın ismi sem2.m)

```
close all
clear all

syms x
f=exp(x)*cos(x);
d1=diff(f)
d2=diff(f,2)
```

```
>> sem2

d1 =

exp(x)*cos(x)-exp(x)*sin(x)



d2 =

-2*exp(x)*sin(x)
```

Aşağıdaki x,y ve z değişkenine bağlı fonsiyonun z ye göre türevini alan sembolik matlab program aşağıdaki şekildedir. (program ismi sem3.m)

```
close all
clear all

syms x y z
f=1/(x*x+y*y+z*z)^(3/2);
d=diff(f,z);
aa=simplify(d)
pretty(aa)

>> sem3

aa =

-3/(x^2+y^2+z^2)^(5/2)*z
```

$$-3\,\frac{z}{\left(x^2+y^2+z^2\right)^{5/2}}$$

5.2 **Sembolik Limit**

Limit işlemi **limit** komutu yardımıyla aşağıdaki programlarda verilen örnekler biçiminde yapılabilir. Limit komutunda sağdan ve soldan limit almak mümkündür. **(limit(f,x,0,'right') veya limit(f,x,0,'right'))**

(program ismi sem4.m)

```
close all
clear all

syms x
f=(1+1/x)^x;
limit(f,x,inf) % x sonsuza giderken

>> sem4

ans =

exp(1)
```

(program ismi sem5.m)

```
close all
clear all

syms x
```

```
f=(x*x-7)/(3*x*x*x-9*x*x+3*x-8);
limit(f,x,3) % x 3 e giderken
```

```
>> sem5

ans =

2
```

5.3 **Sembolik İntegral**

Sembolik integral int komutu yardımıyla aşağıdaki şekilde alınabilir.
int(f,x)
int(f,x,altsınır,üstsınır)

Belirsiz integral örneği.

```
close all
clear all

syms x
f=x*x+1;
a=int(f,x)

pretty(a)
```

>> sem7

a =

1/3*x^3+x

$$1/3\,x^3 + x$$

Belirli integral örneği.

```
close all
clear all

syms x
f=x*x+1;
a=int(f,x,0,1)
```

```
>> sem8

a =

4/3
```

BÖLÜM 6

6.1 En Küçük Kareler Yöntemi

En küçük kareler yöntemiyle verileri bir doğruya yaklaştırmak istenildiğinde doğrunun katsayıları verilerden aşağıdaki formül ile hesaplanabilir. y=a+bx doğrusuna veriler yaklaştırılmaya çalışıldığında a ve b katsayıları için a ve b formülü

$$b = \frac{n\sum xy - \sum x \sum y}{n\sum x^2 - \left(\sum x\right)^2}$$

$$a = \frac{\sum y - b\sum x}{n}$$

ile bulunabilir. Aşağıdaki veri grubu için a ve b katsayılarını hesaplayıp çizen program yazalım.

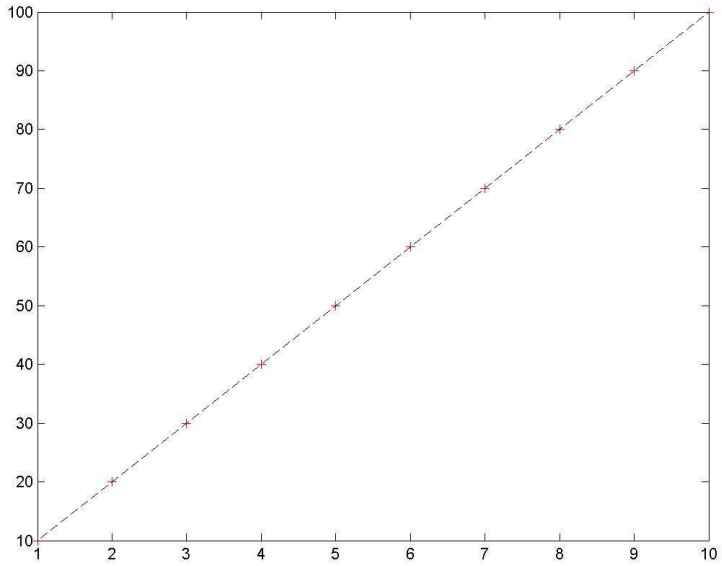
Örnek 1:

```
1 10
2 20
3 30
4 40
5 50
6 60
7 70
8 80
9 90
10 100
```

```
close all
clear all

load arazi.dat;

x=arazi(:,1);
y=arazi(:,2);

plot(x,y,'r+')
axis([0 10 0 10])
hold on

n=length(x);

top1=x.*y;
s_top1=n*sum(top1);

topx=sum(x);
topy=sum(y);
```

```
xx=x.*x;
topxx=n*sum(xx);
topx2=topx*topx;

b=(s_top1-topx*topy)/(topxx-topx2);

a=(topy-b*topx)/n;
 disp('a:')
disp(a)
disp('b')
disp(b)

yh=a+b.*x;

plot(x,yh,'k--')
```

```
>> programismi
a:
     0

b:
    10
```

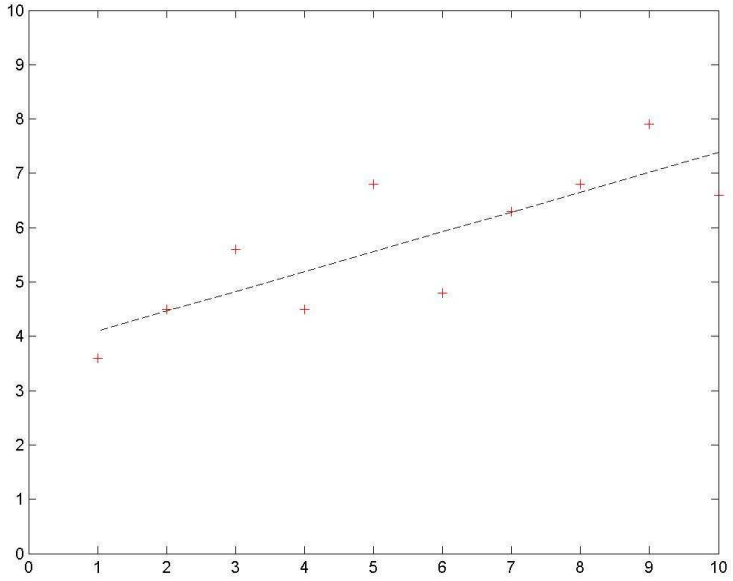


Örnek 2:

Aşağıdaki veri grubu kullanılarak a ve b katsayılarını bulalım.

```
1 3.6
2 4.5
3 5.6
4 4.5
5 6.8
6 4.8
7 6.3
```

```
8 6.8
9 7.9
10 6.6

>> programismi

a:
   3.7333

b:
   0.3648
```



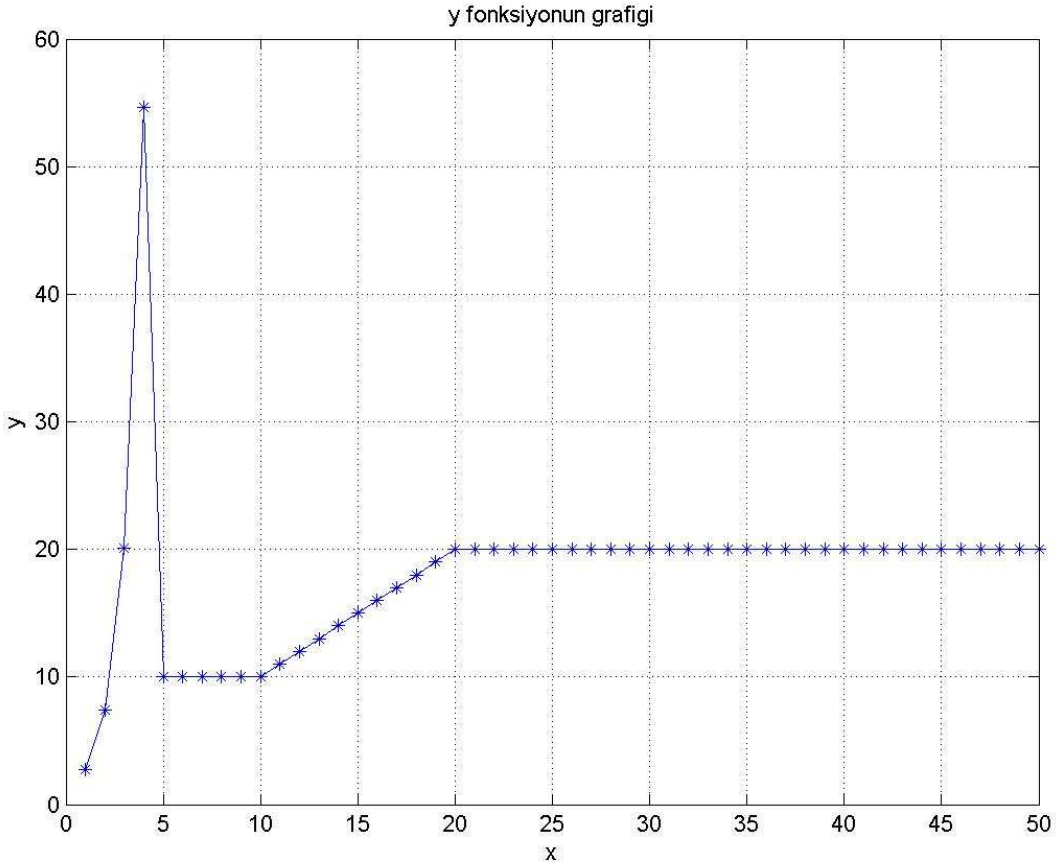
Problem

Aşağıdaki fonksiyonu 1 dan 50 ye kadar 1 er artırarak değişim grafiğini çiziniz.

$$y = f(x) = \begin{cases} e^x & 0 \le x < 5 \\ 10 & 5 \le x < 10 \\ x^2 & 10 \le x < 20 \\ 20 & 20 \le x \le 50 \end{cases}$$

```
close all
clear all

x=1:50;

n=length(x);

for ii=1:n
    if (x(ii) >= 0) & (x(ii) < 5)
```

```
        y(ii)=exp(x(ii));
    elseif (x(ii) >= 5) & (x(ii) < 10)
        y(ii)=10;
    elseif (x(ii) >= 10) & (x(ii) < 20)
        y(ii)=x(ii);
    else
        y(ii)=20;
    end
end

plot(y)
hold on
plot(y,'*')
grid
xlabel('x')
ylabel('y')
title('y fonksiyonun grafigi')
```



6.2 **Matlabta Derleme**

Matlabta yazdığımız programlar **mcc** komutu ile derlenebilir. Derleme işlemi yaparken dikkat etmemiz gereken tek işlem, ana programın ilk satırına **function**

yazmaktır. Aşağıda klavyeden girilen bir sayının karesini alan oldukça basit bir programı derleyelim.

Program ve alt program aşağıdaki şekildedir.

```
function sayi
n=input('Bir sayi giriniz:');
sonuc=kare(n);
disp('Sayinin karesi:')
disp(sonuc)
```

Bu programı sayi.m olarak kayıt edelim. Aaşğıdaki alt programla da sayının karesini alalım ve ismini kare.m olarak kayıt edelim.

```
function kk=kare(m)
kk=m*m;
```

Bu işlemlerden sonra Matlab komut satırında

>> mcc –m sayi

yazıp enter tuşuna bastığımızda sayi.m programı sayi.exe haline gelir. Programı denemek için MS-DOS penceri açılır ya da Matlab komut satırında

>>!sayi

yazarak programı deneyebiliriz.
Burada ! işareti MS-DOS işletim sisteminde çalış anlamında kullanılır.

6.3 Karışık sorular ve çözümleri

1. Yeraltında bulunan küre şeklinde bir cismin gravite anomalisi

$$g_z = \frac{4\pi\Delta\rho R^3 z}{3\left(x^2 + z^2\right)^{(3/2)}}$$

bağıntısı ile hesaplanabilir. Burada $\Delta\rho = \rho_{küre} - \rho_{back}$ yoğunluk farkıdır. R kürenin yarıçapıdır (m). z küre merkezinin derinliğidir (m). x uzunluğu profil uzunluğudur (m). Toplam x eksenini -x:dx:x olarak alınız. G sabitini ise 6.6732×10^{-8} dyne cm^2/g^2 alınız. Aşağıdaki verilenlere göre küre anomalisinin grafiğini çizen program yazınız. Düşey ekseni gmax ile normalize ediniz. Gmax aşağıdaki formülle

$$g_{\max} = \frac{4\pi\Delta\rho R^3}{3z^2}$$

hesaplanabilir.

Verilenler:

$\rho_{küre} = 0$

$\rho_{back} = 2.4\ \text{g/cm}^3$

$R = 20\ \text{m}$

$z = 80\,\text{m}$

$dx = 10\,\text{m}$

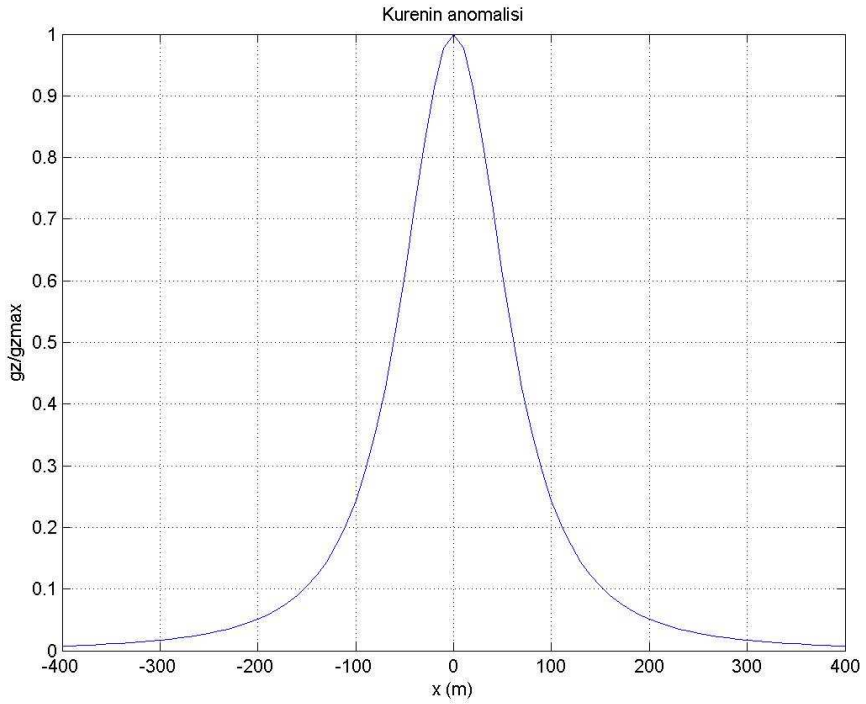
$x = 400\,\text{m}$

program:

```
close all
clear all
%
kure_y=input('kurenin yogunlugu(g/cm3):');
backg_y=input('cevre kayac yogunlugu(g/cm3):');
z=input('Kurenin merkez derinligi(m):');
R=input('Kurenin yarı capi:');
%
sabit=6.6732*1e-8;
%
delta_y=kure_y-backg_y;
%
x=-400:10:400;
%
gz=(sabit*4*delta_y*pi*R*R*R*z)./(3*(x.*x+z*z).^(3/2));
gzmax=(sabit*4*delta_y*pi*R*R*R)/(3*z*z);
gzm=gz/gzmax;
plot(x,gzm)
axis([-400 400 0 1])
xlabel('x (m)')
```

```
ylabel('gz/gzmax')
title('Kurenin anomalisi')
grid
print -djpeg kure
```

```
 >> kure
kurenin yogunlugu(g/cm3):0
cevre kayac yogunlugu(g/cm3):2.4
Kurenin merkez derinligi(m):80
Kurenin yarı capi:20
```



2. Bir öğrencinin vize notuna göre finalden geçmesi için alması gereken en az notu hesap eden program yazınız. Geçme notu vizenin %40 ı ve final sınavının %60 ını alarak hesaplanır. Geçme notu ise 50 dir. Vize notu 50 ve daha yüksek alan öğrenci için hesaplama yapmayınız. Programa vize notu klavyeden girilecek. Sonuçta ekranda öğrencinin dersi geçmesi için alması gereken en az final notu olmalı.

```
close all
clear all

vize=input('Vize notunuzu giriniz:')

if (vize > 100) | (vize < 0)
```

```
    disp('yanlis vize notu')
    disp('tekrar deneyiniz')
elseif (vize < 50)
    n_min=round((50-0.4*vize)/0.6);
    disp('Bu dersten gecmeniz icin en az')
    disp(n_min)
    disp('notunu almanız gereklidir.')
else
    disp('vize 50 den büyük')
    disp('Fnalden 50 almak yeterli.')
end
```

3. (x,y,z) koordinatları verilen iki nokta arasındaki uzaklık

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

ile hesaplanabilir. Bu iki nokta arasındaki uzaklığı hesap eden alt program yazınız (**function**).

```
function A=uzak(x1,y1,z1,x2,y2,z2)
A=sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
```

```
>> uzak(1,2,0,3,2,1)

ans =

    2.2361
```

4. 2 x 2 lik kare matrisin tersi

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \begin{bmatrix} d/(ad-bc) & -b/(ad-bc) \\ -c/(ad-bc) & a/(ad-bc) \end{bmatrix}$$

ile hesaplanabilir. Klavyeden girilen 2 x 2 lik matrisin tersini verilen eşitliğe göre hesaplayınız. Sonucun doğruluğunu A matrisini A matrisinin tersiyle çarparak birim matris elde eden bir program yazınız.

```
close all
clear all

disp('Matris elemanlarını giriniz:')
%Matris 2 X 2
% |a b|
% |c d|
```

```
a=input('a:');
b=input('b:');
c=input('c:');
d=input('d:');

A=[a b ; c d];
payda= a*d-b*c;
at=d/payda;
bt=-b/payda;
ct=-c/payda;
dt=a/payda;

disp('A matrisi:')
disp(A)
disp('A matrisinin tersi:')
At=[at bt ; ct dt];
disp(At)
disp('Sonuc A*inv(A)=birim matris')
br=A*At
```

```
>> ters
Matris elemanlarını giriniz:
a:1
b:0
c:2
d:3
A matrisi:
    1    0
    2    3

A matrisinin tersi:
    1.0000         0
   -0.6667    0.3333

Sonuc A*inv(A)=birim matris

br =

    1    0
    0    1
```

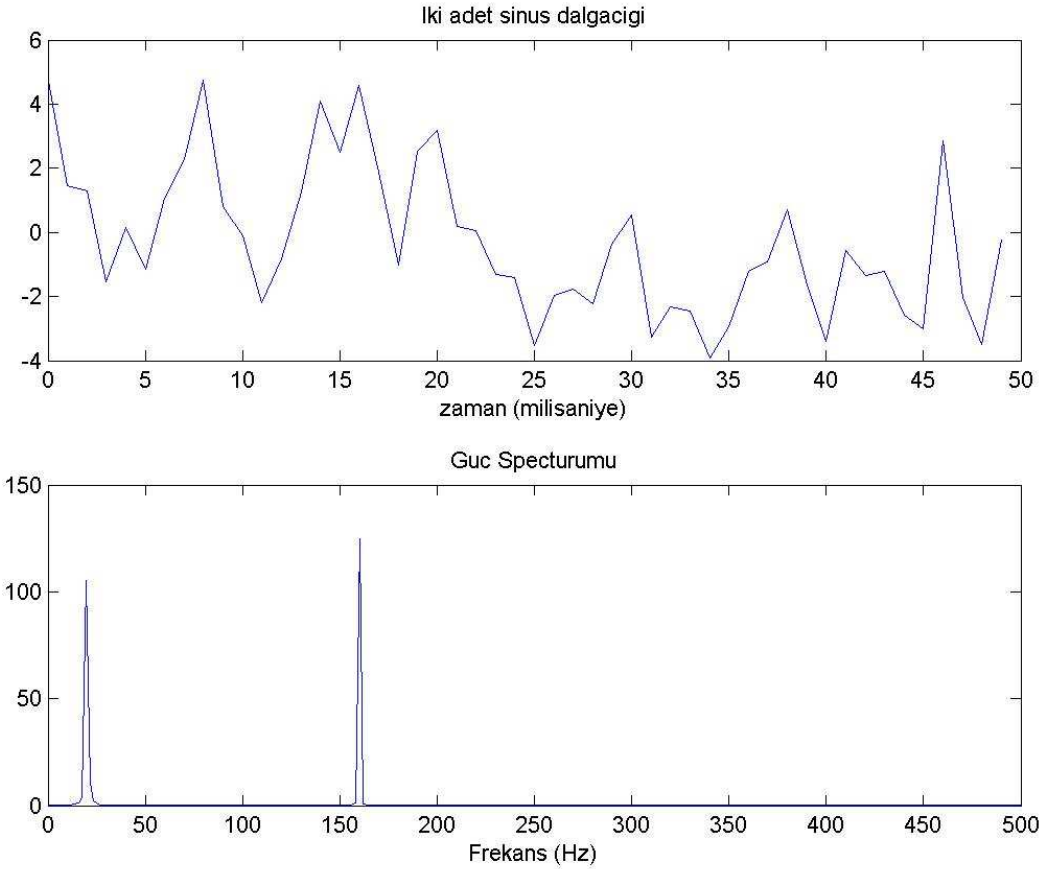
EKLER

Ek-1 Fourier dönüşümü

Jeofizikte sık olarak kullanılan işlemlerden bir tanesi Fourier transformudur. Bu işlem ile zaman ortamındaki veriler frekans ortamına dönüştürülebilir. Aşağıdaki programda iki adet sinüs fonksiyonu kullanarak üretilen sinyale gürültü eklenmiştir. Daha sonra bu verinin Fourier transformu fft komutu kullanılarak alınıp, güç spektrumu çizilmiştir.

Şimdi aşağıdaki örnek programı inceleyelim.

```
close all
clear all

t=0:0.001:0.6; % t 0.001 ms zaman aralikligi
%x=sin(2*pi*50*t)+sin(2*pi*120*t);
x=sin(2*pi*20*t)+sin(2*pi*160*t);%iki sinus egrisi
%(orijinal sinyal)
y=x+2*randn(size(t));% sinyale gurultu ekle

subplot(211) %orijinal sinyali ciz
plot(1000*t(1:50),y(1:50)) %ms saniyeye çevirmek için 1000 le carpılır
title('Iki adet sinus dalgacigi')
xlabel('zaman (milisaniye)')

%-----------------------
yf=fft(x,512); % Fourier trasnformu bu komutla alınır
% yf in hem real hem sanal kısmı var
powyf = (1/512)*abs(yf).*abs(yf);% Guc spektrurmu burada hesaplanır
f=1000*(0:256)/512;% frekans ekseni (yarısını çizmek yeterli)

subplot(212) % fft sonucuna gore guc spektrumunu ciz
plot(f,powyf(1:257))
title('Guc Specturumu')
xlabel('Frekans (Hz)')

%print -djpeg sekil
```



Şekil1. Zaman verisi ve Güç spektrumu.

Ek-2 Sismik Kırılma: İki Tabakalı Yeraltı Modeli

Bu dersimizde sismik kırılma verilerinden yeraltı tabakalarının hızlarını bulacağız. Veri.dat dosyasında uzaklık (m) ve zaman (ms) verileri vardır. Bu verilerden iki tabakalı yeraltı modeline ait yer altı hızlarını (v1 ve v2) ve tabaka kalınlığını bulacağız. Sismik zaman verilerinin eğimi 1/hız a eşittir. Bu bilgiden yararlanarak, verilerden fare yardımıyla noktalar seçerek verilerden doğrular geçirip, eğimden hız hesaplanması yapılabilir. Bu seçim programın en önemli özelliği, noktayı yanlış yerde işaretlemek parametrelerin yanlış hesaplanmasına neden olacaktır. Veriler 18 kanallı bir cihazdan alındığını varsayınız. Dolayısıyla 18 adet x ve t değerleri vardır.

Programın ilk adımında **load** komutuyla veri.dat dosyası okunur ve ilk kolon x değişkenine ikinci kolon zaman değişkenine atanır. Daha sonra şekilden fare ile bizim bir kırılma noktası seçmemiz gerekiyor. Bu işlemi yaptığımızda, program iki tabakalı yer altı modelinin ilk tabakasının hızını verir. Daha sonra ikinci tabakanın hızı fare yardımıyla verilerin kalan kısmından bir doğru geçirmek için iki nokta seçilerek yapılabilir. Böylece yeraltı modeline ait parametreler V1, V2 hızları ile tabaka kalınlığı hesaplanmış olur.

Veri.dat

```
X(m) zaman(ms)
5 8.30
10 16.70
15 25.00
20 32.30
25 41.70
30 50.00
35 58.30
40 66.70
45 70.00
50 72.10
55 74.20
60 76.30
70 78.10
75 79.90
80 84.50
85 86.60
90 88.10
95 90.30
```

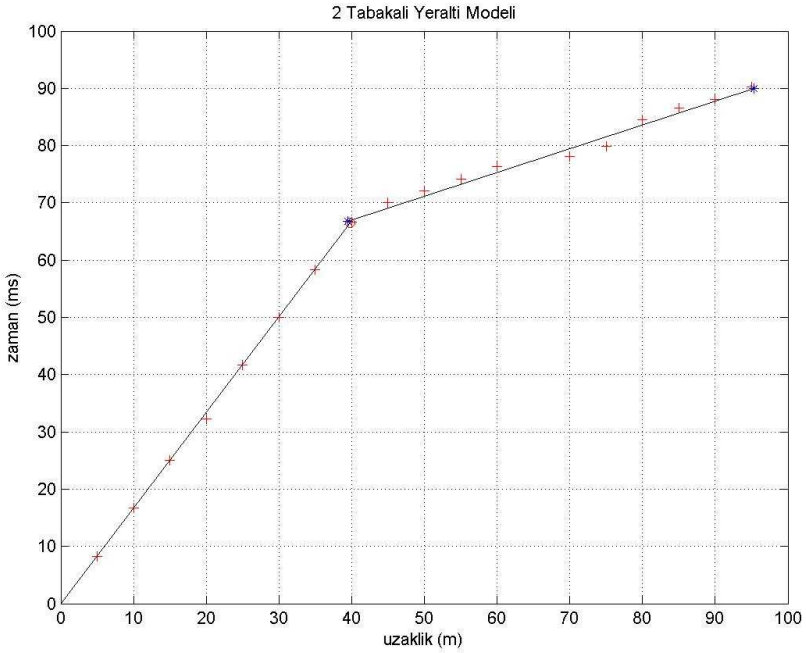
Phiz.m

```
close all
```

```
clear all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%veri yükle
%Birinci kolon uzaklık (m) ikinci kolon zaman (ms)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load hiz.dat
%
xx=hiz(:,1);
zaman=hiz(:,2);
%veriyi ciz
plot(xx,zaman,'r+')
xlabel('uzaklik (m)')
ylabel('zaman (ms)')
hold on
grid
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Birinci tabakanın hizini bulmak için fare ile nokta belirle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[xi,yi]=ginput(1);
plot(xi,yi,'ro');
% (0,0) baslangic noktasi
 x1_m1=0;
 y1_m1=0;
 %Fare ile girilen noktanın koordinati
 x1_max1=xi;
 y1_max1=yi;
%egimden hız hesabı
slope1=(x1_max1)/(y1_max1)*1000;
disp('Birinci tabakadaki Vp hizi(m/s):')
disp(slope1)
%Dogru cizimi
plot([x1_m1 x1_max1], [y1_m1 y1_max1],'k')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%İkinci tabakanin hızını bulmak icin fare ile iki nokta girilmesi
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ikinci nokta
[xi2,yi2]=ginput(1);
plot(xi2,yi2,'o');
%
 x1_max2=xi2;
 y1_max2=yi2;
% Hız hesabı
slope2=((x1_max2-x1_max1)/(y1_max2-y1_max1))*1000;
disp('İkinci tabakadaki Vp hizi (m/s):')
disp(slope2)
%cizim
plot([x1_max1 x1_max2], [y1_max1 y1_max2],'k')
title('2 Tabakali Yeralti Modeli')
v1=slope1;
v2=slope2;
% Tabaka kalinligi hesabi
kalinlik=(xi)/2*sqrt((v2-v1)/(v2+v1));
disp('Tabaka kalinliği (m)')
disp(kalinlik)
```

Tabaka parametreleri aşağıdaki şekilde bulunabilir.

```
Birinci tabakadaki Vp hizi(m/s):
  597.5085

İkinci tabakadaki Vp hizi (m/s):
  2.3739e+003

Tabaka kalinliği (m)
   15.3659
```



Kaynaklar

http://www.mathworks.com
http://www.maths.dundee.ac.uk/software