GİRİŞ

MATLAB, bilimsel çalışmalar ve mühendislik hesaplarında kullanılabilecek matris esaslı etkileşimli bir sistemdir. Aslında programlamaya çok yatkın olan MATLAB ile karmaşık sayısal problemler gerçek anlamda bir program yazmaksızın kolaylıkla çözülebilir. MATLAB adı İngilizce MATrix LABoratory (MATris LABoratuarı) kelimelerinden elde edilmiş bir kısaltmadır.

Bu notların amacı, MATLAB kullanmaya başlayanlara yardımcı olmaktır. Küçük bir el kitabı olarak düşünülmüştür. Notları okurken bir taraftan da bilgisayarda çalışmak ve çeşitli örnekler yapmak faydalı olacaktır.

Ayrıntılı açıklamalar için anında yardım kullanmanız mümkündür. Daha sonra açıklanacağı gibi MATLAB'a girdikten sonra `help` komutu ile anında yardım alınabilecek fonksiyonlar listesini görebilirsiniz. `help` *`fonksiyonadı`* komutu size özel bir fonksiyon hakkında ayrıntılı bilgi verecektir. Mesela `help eig` komutu ile özdeğer fonksiyonu **`eig`** hakkındaki tüm kullanım detaylarına ulaşmak mümkündür. `demo` komutu ile MATLAB'ın bazı özelliklerini inceleyebilirsiniz.

MATLAB'ın gücü ve etkisi bu notlarda gösterilebilenlerin çok ötesindedir. Konuyu daha detaylı irdeleyen kitaplar vardır. MATLAB'ın, WINDOWS ortamında çalışan çeşitli sürümleri vardır, ancak bu sürümler arasındaki farklar burada anlatılanlar itibariyle önemli değildir.

1. MATLAB'a giriş

WINDOWS ortamında çalışırken masa üstündeki MATLAB simgesini tıklatarak giriş yapılabilir.

2. Matris Girişi

MATLAB'da bütün değişkenler matris olarak gözetilirler. Boyutu 1×1 olan bir matrise skaler adı verilir. O halde mesela 5 sayısı bir skaler veya boyutu 1×1 olan bir matristir. Bütün değişkenlerin bir matris olarak gözetileceği unutulmamalıdır. Tek satırlı veya tek sütunlu matrislere vektör adı da verilir. Bir skalere de tek elemanlı vektör olarak bakılabilir.

Yukarda söylenenleri birer örnekle gösterelim.

Skaler veya 1×1 boyutlu matris veya tek elemanlı vektörler:

```
5   -3   0   1e6    0.55   -6.6666  -1.2e-2   3*0.4/2   6^2.5
```

Tek satırlı matrisler veya vektörler:

```
[1 2 3]    [-1 0 3*6.7]    -4:4     [5 exp(1) -402,0]
```

Tek sütunlu matrisler veya vektörler:

```
[3      [-2;5;5;cos(0)]
 5
 0]
```

Matrisler

```
[1 2 3;4 5 6;7 8 9]
```

```
[1 2 3
 4 5 6
 7 8 9]
```

```
[
 1:3
 4:6
 7:9
]
```

Buradaki üç işlem de aynı sonucu verir.

Bir satır içindeki elemanlar birbirlerinden boşlukla veya virgülle ayrılmalıdır. Sayıların üstel yazılımında boşluk verilmemelidir, `3.49e-7` gibi.

Değişkenler

`a=[1 2 3;4 5 6;7 8 9]` ifadesi 3×3 lük bir matris oluşturacak ve bunu `a` değişkenine atayacaktır. Değişken isimleri mutlaka alfabetik bir karakterle başlamalıdır. Bunun dışında önemli bir sınırlama yoktur. İşte bir kaç örnek:

```
a1=10:-1:0;  matlab_degişkeni=[a1;2*a1]; x2x=[ ];   cs=cos(0)
nk='Newton Kanunu'; NK='Newton Yasasi'
```

Matris boyutlarının bulunması

MATLAB'da her değişkenin bir matris olarak gözetildiği daha önce açıklanmıştı. Bir matrisin, bir vektörün ya da bir skalerin boyutunu bulmak için `size` fonksiyonu kullanılır. Örnekler:

```
» size(4)
ans =
     1     1

» x=1:10;
» size(x)
ans =
     1    10

» y=x';
» size(y)
ans =
    10     1

» z=[1 2;0 0];
» size(z)
ans =
     2     2

» d1000=[1:1000;2:1001];
» size(d1000)
ans =
      2    1000
```

Yukarıdaki `size` fonksiyonu yerine `length` fonksiyonu kullanılırsa bir matrisin en büyük boyutu elde edilir. Yani `length(x), max(size(x))` ile eşdeğerdir. Deneyiniz.

Matris elemanları

Matris elemanlarını bulmak için ilgili satır ve sütunların numaraları parantez içinde virgülle ayrılmış olarak verilmelidir. Mesela `a=[1 2 3;4 5 6;7 8 9]` olmak üzere a(2,3), a matrisinin ikinci satır üçüncü sütunundaki elemanı yani 6 yı verecektir. Aynı şekilde `x=4:7` olarak tanımlanırsa x(1), 4 olacaktır. `y=[0;2;4;6]` olarak ifade edilirse y(4), 6 olacaktır.

Örnekler:

```
» d=5;
» d(1,1)
ans =
     5

» d(1)
ans =
     5
```

```
» r=[0.5 7.6 0 -1];
» r(2)
ans =
    7.6000

» r(1,2)
ans =
    7.6000

» r(2,1)
???   Index exceeds matrix
dimensions.
```

(indis matris boyutlarını aşıyor)

```
» r(3)=4
r =
    0.5    7.6    4    -1

» r(1:3)
    0.5    7.6     4

» r(6)=-3
r =
    0.5  7.6  4  -1  0  -3
```

r(5) için 0 yerleştirilmiş olduğuna dikkat ediniz.

```
» r([2 2])
    7.6000    7.6000

» y=r'
y =
    0.5000
    7.6000
    4.0000
   -1.0000
         0
   -3.0000

» y(2)
ans =
    7.6000

» y(2,1)
ans =
    7.6000

» y(1,2)
???   Index exceeds matrix
dimensions.

» y(5)=12
y =
    0.5000
    7.6000
    4.0000
   -1.0000
   12.0000
   -3.0000

» a=[1 2 3;4 5 6;7 8 9];

a =

     1     2     3
     4     5     6
     7     8     9

» a(2,3)
ans =
     6

» a(2,:)
ans =
     4     5     6

» a(:,3)
ans =
     3
     6
     9

» a(2:3,[1 3])
ans =
     4     6
     7     9

» a(1:2,3)
ans =
     3
     6

» a([1 2],3)
ans =
     3
     6

» a(1:2,3)=[-3;-6]
a =
     1     2    -3
     4     5    -6
     7     8     9
```

Matris indisleri vektör ya da matris olabilir. `a([1 2;2 3])`, veya `p=[1 2;2 3]` olmak üzere `a(p)` komutlarını deneyiniz. `a([1 2;2 3])` ile `a([1 2],[2 3])` arasındaki farkı anlamak için bu ikisini ayrı ayrı mutlaka deneyiniz. `a(:,2)` komutunda üst üste iki noktanın (:), ilgili matrisin ikinci sütunundaki tüm satırları gösterdiğine dikkat ediniz.

Yukarıdaki son örnek matris elemanlarının nasıl değiştirilebileceğini göstermektedir. a(:) ve b=a(:) komutlarıyla neler bulursunuz, deneyiniz?

3. Matris İşlemleri

MATLAB'la aşağıdaki matris işlemlerini yapmak mümkündür:

+	toplama
–	çıkarma
•	çarpma
^	üs
′	transpoz
\	sağdan bölme
/	soldan bölme

Bu matris işlemleri skalerlere de uygulanabilir. Matris işlemlerinde matris boyutları uyumlu değilse bir hata mesajı belirecektir. Ancak bir skalerle bir matris arasındaki işlemlerin çoğunda herhangi bir sorun ortaya çıkmayacaktır. İki matris arasındaki işlemler bir kaç türlü olabilir. Mesela iki matrisi eleman eleman çarpmakla matris olarak çarpmak farklı şeylerdir (matris çarpımını hatırlayınız). İki satır matrisinin (vektörünün) matris çarpımı her zaman bir hata verecektir (`x=1:3` ve `y=4:6` olarak tanımlayınız ve `x*y` işlemini yapmaya çalışınız). x ve y satır matrislerini eleman eleman çarpmak için `x.*y` yazmak gerekecektir. Aynı şekilde x vektörünün karesini almak için ya `x.*x` veya `x.^2` yazmak gerekir (sonuç `[1 4 9]`).

Söylediklerimizi örneklemeye çalışalım.

a) Skaler-skaler işlemleri

```
» 3+5
ans =
    8

» -4+5*2
ans =
    6

» 4-8/2*5
ans =
   -16
» cos(0)+3^2
ans =
    10

» -4^2
ans =
   -16

» (-4)^2
ans =
    16

» 9\18
ans =
    2
```

b) Vektörlerle işlemler

Vektör işlemleri terimi ile eleman eleman yapılan işlemler anlaşılmalıdır. Bunun için toplama ve çıkarma hariç olmak üzere bir işlem işaretinden önce . (nokta) işareti kullanmak gerekir. Ancak çarpmada bir skalerle bir vektör arasındaki işlemde buna gerek yoktur. Bölmede ise vektörü bir sayıya bölerken değil bir sayıyı vektöre bölerken nokta işaretine

ihtiyaç vardır. Örnekleri dikkatle inceleyiniz. Üs alırken her durumda nokta işaretine ihtiyaç vardır.

```
» 3+[1 2 3]
ans =
     4     5     6

» 1:3+1
ans =
     1     2     3     4

» (1:3)+1
ans =
     2     3     4

» a=[1 2 3 4]
» 5*a
ans =
     5    10    15    20

» a*5
ans =
     5    10    15    20

» a/0.5
ans =
     2     4     6     8

» 2/a
??? Error using ==> /
```
(/ yanlış kullanılıyor)
```
Matrix dimensions must
agree.
```
(matris boyutları uyumlu olmalı)

```
» a/2
ans =
    0.5   1   1.5    2

» (1:4)^3
??? Error using ==> ^
    Matrix must be square.
```
(^ yanlış kullanılıyor)
(Matris kare olmalı)

```
» (2:2:10)^2
??? Error using ==> ^
    Matrix must be square.
```

Yukarıdaki işlemlerde + işaretini - işareti ile değiştirerek tekrar yapınız.

Görüldüğü gibi işlemlerin bazılarında hata mesajı belirmektedir. Mesela `2/a` için hata mesajı çıkmıştır.

```
» a=[2 4 8]

» a/2
ans =
     1     2     4

» a./2
ans =
     1     2     4

» 2/a
??? Error using ==> /
Matrix dimensions must
agree.

» 2./a
ans =
    1    0.5    0.25


» a.\2
ans =
    1   0.5000    0.2500

» 2\a
ans =
     1     2     4

» 2.\a
ans =
     1     2     4

» (1:3)+(6:8)
ans =
     7     9    11

» x=[1 2 3];
» y=[4 5 6];
» z=x.*y
z =
     4    10    18

» z=x./y
z =
    0.25   0.4    0.5

» z=x.\y
z =
    4   2.5    2

» z=y./x
z =
```

```
    4   2.5    2

» z=y.\x
z =
    0.25    0.4    0.5

» x=[1;2;3];
» y=[4;5;6];
» z=x.*y
z =
     4
    10
    18

» z=x./y
z =
    0.2500
    0.4000
    0.5000

» z=x.\y
z =
    4.0000
    2.5000
    2.0000

» z=y./x
z =
    4.0000
    2.5000
    2.0000

» z=y.\x
z =
    0.2500
    0.4000
    0.5000

» p=[1 2 3]; q=[4 3 0];
» z=p.^q
z =
     1     8     1
```

Üs bir skaler olabilir:

```
» z=p.^2
z =
     1     4     9
```

Taban da bir skaler olabilir:

```
» z=2 .^[p q]
z =
    2  4  8  16  8  1
```

Yukarıdaki işlemde 2 den sonra bir boşluk vermekte fayda vardır

Bir matris ya da vektörün transpozu satır ve sütunun yer değiştirmesi olarak tanımlanır. Bu anlamda bir matris ya da vektörün transpozunu almak için ‘ işareti kullanmak gerekir. İşte bir kaç örnek:

```
» v1=0:2:6
v1 =
     0     2     4     6

» v=v1'
v =

     0
     2
     4
     6

» v2=[0 1 2 -1]'
v2 =
     0
     1
     2
    -1

» v3=v2'
v3 =
     0     1     2    -1
```

c) Matrislerle işlemler

Yukarda satır ve sütun matrislerinin transpozu verilmiştir. Bir matrisin transpozuyla ilgili aşağıdaki örnekleri inceleyiniz.

```
» m=[1:3;4:6]
m =
```

```
     1     2     3
     4     5     6

» n=m'
n =

     1     4
     2     5
     3     6

» k=[1 0;1 -9]'
k =
     1     1
     0    -9
```

Matematikten bilindiği gibi, matris çarpımı matrislerin karşılıklı elemanlarının çarpımı değildir. Matris çarpımı için sadece * işaretini kullanmak gerekir. Matris elemanlarını karşılıklı çarpmak içinse **.*** işaretlerinin birlikte kullanılması gerekir. Örnekleri inceleyiniz ve kendi kendinize satır ve sütun matrisleri de oluşturarak matris çarpımları gerçekleştiriniz.

```
» x=[1 2;3 4]

x =
     1     2
     3     4

»  y=[0 -1;2 10]

y =
     0    -1
     2    10

» x*y
ans =
     4    19
     8    37

» x.*y
ans =
     0    -2
     6    40

» v1=0:2:6; v2=1:2:7;
» z=v1*v2
??? Error using ==> *
```

Inner matrix dimensions must agree. (Birinci matrisin sütun sayısı ikinci matrisin satır sayısına eşit olmalı)

```
» z=v1*v2'
z =
    68

» z=v1'*v2
z =
     0     0     0     0
     2     6    10    14
     4    12    20    28
     6    18    30    42

» z=v2*v1'
z =
    68

» z=v2'*v1
z =
     0     2     4     6
     0     6    12    18
     0    10    20    30
     0    14    28    42
```

Karmaşık Sayılar:

MATLAB'daki tüm işlemlerde ve fonksiyonlarda karmaşık sayıları kullanmak mümkündür. Karmaşık sayılar i ya da j kullanılarak girilir. Karmaşık sayılar i=(-1)^0.5 şeklindeki tanımla verilebileceği gibi, mesela jj=(-1)^0.5 şeklindeki bir tanımla da girilebilir. Herhangi bir çalışma alanında (workspace) karmaşık sayılar örneğin i ile verilmişse, i değişkenine yapılacak başka bir atamadan sonra karmaşık sayılarla işlem için i nin tekrar kullanılamayacağı unutulmamalıdır. Aşağıda karmaşık sayılarla ilgili bir kaç örnek verilmiştir.

```
» i
ans =
        0 + 1.0000i

» z1=1+3i
z1 =
```

```
   1.0000 + 3.0000i

» z2=2-4i
z2 =
   2.0000 - 4.0000i

» z=z1+z2
z =
   3.0000 - 1.0000i

» z=z1*z2
z =
  14.0000 + 2.0000i

» z1/z2
ans =
  -0.5000 + 0.5000i

» z1^2
ans =
  -8.0000 + 6.0000i

» (1+i)*[2-3i 5 4i]
ans =
   5 - 1.0i   5 + 5i  -4 + 4i

» [i  1+i  2].*[i  1-i  i-1]
ans =
  -1   2   -2+2i

» x=[1+i 1-i;-2-i 4];
» y=[3 2-3i;4-5i 6-2i];
» x*y
ans =
   2 - 6i    9 - 9i
  10 - 23i  17 - 4i

» x.*y
ans =
   3 + 3i  -1 - 5i
 -13 + 6i  24 - 8i

» x
x =
   1 + 1i   1 - 1i
  -2 - 1i   4

» x'
ans =
   1 - 1i  -2 + 1i
   1 + 1i   4

» x.'
ans =
   1 + 1i  -2 - 1i
   1 - 1i   4
```

Son iki transpoz işlemine dikkat ediniz. `x'` eşlenik alarak transpoz işlemini yaparken, `x.'` bildiğimiz anlamda transpoz almaktadır.

Çalışma alanı (workspace) bilgileri:

Bir çalışma yapılırken verilen ve elde edilen değişkenler bir çalışma alanında saklanmıştır. Bunların neler olduğunu görmek için who komutunu girmek gerekir. Daha detaylı bilgi elde etmek içinse whos komutu kullanılır. Bilgisayarımızı yeni açtığımızı varsayalım ve bazı değişkenler tanımlayarak bu komutları kullanalım.

```
» clear
» x=12;y=[-2 4 6 -1 0];
» z=(1+i)*x;
» m=[1 2;3 0];
» [p q]=size(m);

» [p q]
ans =
     2     2

» who

Your variables are:

ans       p         x         z
m         q         y
```

Sürüm 4.2 de:

```
» whos
      Name      Size       Elements     Bytes    Density   Complex

      ans      1 by 2             2        16       Full      No
        m      2 by 2             4        32       Full      No
        p      1 by 1             1         8       Full      No
        q      1 by 1             1         8       Full      No
        x      1 by 1             1         8       Full      No
        y      1 by 5             5        40       Full      No
        z      1 by 1             1        16       Full      Yes

Grand total is 15 elements using 128 bytes
```

Sürüm 5.2 de:

```
» whos
  Name      Size          Bytes  Class

  ans       1x2              16  double array
  m         2x2              32  double array
  p         1x1               8  double array
  q         1x1               8  double array
  x         1x1               8  double array
  y         1x5              40  double array
  z         1x1              16  double array (complex)

Grand total is 15 elements using 128 bytes
```

Buradaki değişkenleri çalışma alanından silmek için `clear` komutu kullanılır. `clear` komutu yalnız başına kullanılırsa tüm değişkenler silinir. `clear` *`değişken_adları`* ise sadece verilen değişkenlerin silinmesini sağlar. Yukardaki değişkenlerden bazılarını silerek `who` ya da `whos` komutunu tekrar kullanınız.

Çalışma alanının saklanması

Daha önce söylendiği gibi programdan çıkmak için `quit` ya da `exit` komutları kullanılır. Ancak programdan çıkmadan önce elde ettiğimiz değişkenleri daha sonra kullanmak üzere bir çalışma alanında saklamak isteyebiliriz**.** Bunun için `save` *`çalışma_alanı_adı`* komutu kullanılır. Ancak saklamak istediğimiz çalışma alanının farklı bir klasöre (directory'ye) yerleştirilmesini istiyorsak bunun belirtilmesi gerekir. Böyle bir işlemi yaptıktan diyelim ki üç gün sonra daha önceki çalışma alanını elde etmek ve oradaki değişkenleri kullanmak istiyorsunuz. Bunun için de `load` *`çalışma_alanı_adı`* komutu kullanılır. Örnekleri inceleyiniz.

```
» cd
C:\MATLAB\BIN

» cd \mak

» x=1;y=-10:2:10;z=1+i;
» m=[-2:1;2:5;9:-1:6];
» p=m';

» who
```

```
Your variables are:
m     p     x     y     z

» clear x
» who
Your variables are:
m     p     y     z

» save ders
```

Bu komutlar sonunda `mak` adlı klasörde `ders.mat` adlı bir dosya oluşur. `ders.mat` adlı dosyanın uzantısı olan `mat` program tarafından oluşturulur. Şimdi üç gün sonra yeniden çalışmaya başladığımızı varsayalım. `ders.mat` adlı çalışma alanını elde etmek için şu işlemlere ihtiyaç vardır:

```
» cd

C:\MATLAB\BIN

» cd \mak

» load ders

» who
Your variables are:
m         p         y         z

» yeni=y.^2;

» size(yeni)

ans =
     1    11

» who
Your variables are:
ans       p         yeni
m         y         z
```

Disk Yönetimi

MATLAB'da bazı DOS komutlarını kullanmak mümkündür. Bunların başlıcaları şunlardır: dir, chdir veya cd, delete, type. Şimdi MATLAB'ın çalıştığı klasörden DERS adlı klasöre geçerek bu komutları örnekleyelim. Bu klasörde matrisp.m ve seqpow.m adlı MATLAB'la yazılmış iki program vardır. Bu programların yapısı hakkında daha sonra bilgi verilecektir.

```
» cd

C:\MATLAB\BIN

» cd \ders

» dir
.          ..         matrisp.m seqpow.m
```

dir komutu uzantısına bakmaksızın tüm dosyaları listeleyecektir. Eğer sadece MATLAB'a ilişkin dosyalar olan m ve mat uzantılı dosyaları elde etmek istersek what komutunu kullanmalıyız.

```
» type seqpow.m
% seqpow.m
% bir sayı dizisinin muhtelif kuvvetlerinin hesabı
z=1;
while z==1
 a=input('diziyi gir  ');
 b=input('en çok kaçıncı kuvvet  ');
 n=max(size(a));
 c=zeros(b,n);
    for i=1:b
      c(i,:)=a.^i;
    end
 c
 z=input(' yeni hesap için 1, çıkmak için sıfır yaz  ');
end
```

Şimdi `delete` komutuyla `matrisp.m` adlı dosyayı silelim ve yeniden dir komutunu deneyelim.

```
» delete matrisp.m

» delete seqpow
File not found or permission denied.

» dir

.         ..        seqpow.m
```

`seqpow.m` adlı dosyayı da silmek istiyoruz ancak uzantısı verilmediği için hata oluşuyor. Bir çalışma alanındaki değişkenlerin `clear`, bir klasördeki program ya da diğer adıyla fonksiyonların silinmesi için `delete` komutunun kullanıldığını unutmayınız. Ekranın temizlenmesi için `clc`, daha sonra görülecek olan grafiklerin silinmesi için `clg` veya `clf` komutlarının kullanıldığını da not ediniz.

Çıktı formatı

MATLAB'da bütün işlemler çift duyarlılıkla yapılır (double precision). Ancak bir işlemin sonucunun ekranda gösterimi için bir kaç seçenek vardır. Eğer bir matrisin bütün elemanları tam sayı ise sayıların ondalıklı kısımları gösterilmez. Örneğin

```
 » s=[0 1 -1 6/3]
```

şeklindeki bir tanımla her zaman

```
s =
     0     1    -1     2
```

elde edilir. Ancak elemanlardan herhangi birisi tam sayı değilse, sayının ekranda gösteriliş biçimi olarak çeşitli seçenekler vardır. Bunlar sırasıyla `format short`, `format short e`, `format long`, `format long e`, `format hex` ve `format +` olarak sıralanabilir. `format hex`, sayının hegzadesimal gösterimi anlamındadır. `format +` ile matrisin pozitif elemanları +, negatif elemanları -, sıfır olan elemanları boşlukla temsil edilir. Diğerlerini örnekleri inceleyerek kavrayabiliriz.

```
» s=[4/3 1.2345e-6 ]

s =
    1.3333    0.0000

» format short e
» s
s =
  1.3333e+000  1.2345e-006

» format long
» s
s =
   1.33333333333333   0.00000123450000
```

```
» format long e
» s
s =
    1.333333333333333e+000    1.234500000000000e-006

» format hex
» s
s =
   3ff5555555555555   3eb4b6231abfd271

» format +
» s
s =
++
```

Normal gösterime geçmek için `format short` ya da kısaca `format` komutu yeterlidir.

```
» format
» s
s =
    1.3333 0.0000
```

s matrisinin ikinci elemanı sıfır olarak gözükmektedir. Ancak bu sadece ekrandaki görüntüdür. Bellekte sayı gerçek haliyle saklanmaktadır. Bunu anlamak için sayının kendisine ve 100 ile çarpımına bakalım .

```
» s(2)
ans =
  1.2345e-006

» 100*s(2)
ans =
  1.2345e-004
```

format short ile format short g arasındaki farkı gözlemek için aşağıdaki komutları dikkatle izlemek yueterlidir.

```
» format short g
» s

s =
       1.3333  1.2345e-006
```

Eğer bir matrisin en büyük elemanı 1000 den büyük veya 0.001 den küçükse tüm elemanlara uygulanması gereken ortak bir faktör belirir:

```
» s=[5/3 1.2345e-3 -1.25e3]
s =
  1.0e+003 *

    0.0017    0.0000   -1.2500
```

Son olarak `format compact` ve `format rat` komutundan söz edilebilir. format compact sayıların ekranda gösteriminde satırlar arasında boşluk verilmesini önler. format rat ise sayıları bayağı kesir biçiminde elde etmek için kullanılır.

```
» format rat
» s

s =
     5/3         45/36452   -1250
```

Yardım Kullanımı

MATLAB içinde her türlü konuda yardım kullanmak mümkündür. Herhangi bir çalışma alanında `help` komutuyla hangi hususlarda yardım alınabileceği görülebilir. Özel bir işaret ya da fonksiyon için de yardım kullanmak mümkündür. Bunun için `help` *konu* (örneğin `help [` veya `help inv`) komutu yeterlidir. İşte bir kaç örnek:

```
» help inv
 INV     Matrix inverse.
    INV(X) is the inverse of the square matrix X.
    A warning message is printed if X is badly scaled or
    nearly singular.

» help polyfit
 POLYFIT Polynomial curve fitting.
    POLYFIT(x,y,n) finds the coefficients of a polynomial
    p(x) of degree n that fits the data, p(x(i)) ~= y(i),
    in a least-squares sense.

    See also POLY, POLYVAL, ROOTS.

» help sin
 SIN     Sine.
         SIN(X) is the sine of the elements of X.
```

Basit matematiksel fonksiyonlar

Basit matematiksel fonksiyonlar matrislere eleman eleman uygulanır. Bunları önce liste halinde gösterip daha sonra birer örnekle açıklamaya çalışalım. Bu arada 3.1415926 şeklindeki pi sayısının MATLAB'da `pi` yazılarak elde edildiğini hatırlayalım.

Basit matematiksel fonksiyonlar (x bir matrisi gösteriyor)	
abs(x)	mutlak değer veya karmaşık sayının modülü
angle(x)	faz açısı (karmaşık sayıda), sonuç radyan cinsinden
sqrt(x)	kare kök
real(x)	karmaşık sayının gerçel kısmı
imag(x)	karmaşık sayının sanal kısmı
conj(x)	karmaşık sayının eşleniği
round(x)	en yakın tam sayıya yuvarlama
fix(x)	sıfıra doğru yuvarlama
floor(x)	$-\infty$ a doğru yuvarlama
ceil(x)	$+\infty$ a doğru yuvarlama
sign(x)	işaret fonksiyonu (sayı pozitifse 1, negatifse -1, sıfırsa 0 verir)

rem(x,a)	x deki bir sayının a ya bölümünden kalan
exp(x)	e^x
log(x)	e tabanına göre logaritma
log10(x)	10 tabanına göre logaritma

Şimdi yukarıdaki fonksiyonları birer örnekle açıklamaya çalışalım:

```
» x=[-2.25 4 -9i 3+4i]
x =
   -2.2500    4.0000    0-9.0000i    3.0000+4.0000i

» abs(x)
ans =
     2.2500     4.0000     9.0000     5.0000

» angle(x)*180/pi
ans =
     180.0000          0   -90.0000    53.1301
```

Sonucun derece cinsinden bulunması için 180/pi ile çarpıldığını not ediniz.

```
» sqrt(x)
ans =
     0+1.5000i    2.0000    2.1213-2.1213i    2.0000+1.0000i

» real(x)
ans =
     -2.2500     4.0000          0     3.0000

» imag(x)
ans =
     0     0    -9     4

» conj(x)
ans =
     -2.2500    4.0000    0+9.0000i    3.0000-4.0000i

» k=[2  -3  4.1   -4.1   4.4  -4.4   4.5  -4.5  4.9  -4.9  4.999];

» round(k)
ans =
     2    -3     4    -4     4    -4     5    -5     5    -5     5

» fix(k)
ans =
     2    -3     4    -4     4    -4     4    -4     4    -4     4

» floor(k)
ans =
     2    -3     4    -5     4    -5     4    -5     4    -5     4

» ceil(k)
ans =
     2    -3     5    -4     5    -4     5    -4     5    -4     5

» sign([1 2 0 -4 -2.44])
ans =
```

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

Karmaşık sayı halinde, yani `X` bir karmaşık sayı ise,
`sign(X) = X ./ abs(X).`
Bunun bir doğrultunun birim vektörünü bulurken yapılan işlemle benzerliğini not ediniz.

```
» sign(3-4i)
ans =
   0.6000 - 0.8000i

» m=[4 5]; b=[2 3];
» rem(m,b)
ans =
     0     2
```

4 ü 2 ye bölünce 2 çıkar **0** kalır, 5 i 3 e bölünce 1 çıkar **2** kalır. Benzer bir fonksiyon `mod` olup detaylar için `help mod` komutundan yararlanabilirsiniz.

```
» x=[0 0.5 1 2 ]

» exp(x)
ans =
    1.0000    1.6487    2.7183    7.3891

» exp(-x)
ans =
    1.0000    0.6065    0.3679    0.1353

» p=[0.25 1 exp(1) 10 1e2]
p =
    0.2500    1.0000    2.7183   10.0000  100.0000

» log(p)
ans =
   -1.3863         0    1.0000    2.3026    4.6052

» log10(p)
ans =
   -0.6021         0    0.4343    1.0000    2.0000
```

Trigonometrik Fonksiyonlar	
sin	sinüs
cos	kosinüs
tan	tanjant
asin	arksinüs
acos	arkkosinüs
atan	arktanjant
atan2	arktanjant (açının birim çemberdeki yeri - aşağıdaki açıklamaya bakınız)
sinh	hiperbolik sinüs
cosh	hiperbolik kosinüs

tanh	hiperbolik tanjant
asinh	hiperbolik arksinüs
acosh	hiperbolik arkkosinüs
atanh	hiperbolik arktanjant

Trigonometrik hesaplarda açıların radyan olarak verilmesi gerektiği unutulmamalıdır. Aynı şekilde ters trigonometrik hesaplar sonucunda da sonuçlar radyan cinsinden çıkacaktır. Aşağıdaki işlemlerde açılar önce derece olarak tanımlanmış daha sonra radyana çevrilmiştir.

```
» d=[0 30 60 90 120 150 180];
» r=pi/180*d;

» sin(r)
ans =
      0    0.5000    0.8660    1.0000    0.8660    0.5000    0.0000

» cos(r)
ans =
    1.0000    0.8660    0.5000    0   -0.5000   -0.8660   -1.0000

» tan([0 30 60 120 150 180]*pi/180)
ans =
         0    0.5774    1.7321   -1.7321   -0.5774    0.0000

» s=[0 1 -1 0.5 3^0.5/2];

» 180/pi*asin(s)
ans =
         0   90.0000  -90.0000   30.0000   60.0000

» 180/pi*acos(s)
ans =
   90.0000         0  180.0000   60.0000   30.0000

» 180/pi*atan(s)
ans =
         0   45.0000  -45.0000   26.5651   40.8934
```

atan2 fonksiyonunun genel yapısı `atan2(y,x)` şeklindedir. Birim çemberi düşünelim. y ve x, işaretleriyle göz önüne alınarak açının pozitif x ekseninden ölçülen değeri elde edilir. Örnek:

```
» aci=180/pi*atan2(1,1)
aci =
    45

» aci=180/pi*atan2(-1,1)
aci =
   -45

» aci=180/pi*atan2(1,-1)
aci =
   135

» aci=180/pi*atan2(-1,-1)
aci =
  -135

» aci=180/pi*atan2(-1,0)
aci =
   -90

» aci=180/pi*atan2(0,-1)
aci =
   180
```

Önce y eksenindeki değerin verildiğini not | ediniz.

```
» sinh([1 -1 0 2])
ans =
    1.1752   -1.1752         0    3.6269

» cosh([1 -1 0 2])
ans =
    1.5431    1.5431    1.0000    3.7622

» tanh([1 -1 0 2])
ans =
    0.7616   -0.7616         0    0.9640

» asinh([1 -1 0 2])
ans =
    0.8814   -0.8814         0    1.4436

» acosh([1 -1 0 2])
ans =
    0     0 + 3.1416i     0 + 1.5708i   1.3170

» atanh([1 -1 0 2])
Warning: Divide by zero
Warning: Log of zero
ans =
  Inf   NaN  -Inf  NaN  0   0.5493+1.5708i

(Dikkat: sıfırla bölme)
(Dikkat: sıfırın logaritması)
(Inf: sonsuz)
NaN: sayı değil)
```

İlişki Operatörleri

MATLAB'ın ilişki operatörleri sayıların karşılaştırılmasında kullanılır. Bunlar şu şekilde sıralanabilir:

İlişki Operatörleri	
==	eşit
~=	eşit değil
<	küçük
>	büyük
<=	küçük eşit
>=	büyük eşit

İlişki operatörleri elemanları karşılaştırır ve 0 ve 1 lerden oluşan aynı boyutta bir matris verir. Karşılaştırmada soruya 'evet' cevabı veriyorsak sonuç 1, 'hayır' cevabı veriyorsak 0 dır. Bunu önermemiz doğruysa sonuç 1, yanlışsa 0 dır diyerek de açıklayabiliriz.

Şu örnekleri inceleyelim:

```
» P=1:9,Q=9-P

P =
     1     2     3     4     5     6     7     8     9


Q =
     8     7     6     5     4     3     2     1     0
```

Şimdi P vektörünün 4 ten büyük elemanlarını elde edelim

```
» d=P>4
d =
     0     0     0     0     1     1     1     1     1
```

P nin 4 ten büyük elemanları için 1 bulunurken, 4 ten küçük elemanları için 0 elde edilmiştir. "P ile Q nün karşılıklı elemanları arasında eşit olan var mı" sorusuna cevap arayalım. Bunun için `esitmi` adlı bir değişken tanımlayalım

```
» esitmi=P==Q
esitmi =
     0     0     0     0     0     0     0     0     0
```

Burada = ile == işaretlerinin iki farklı maksat için kullanıldıklarını gözden kaçırmayalım. P ile Q nün karşılıklı elemanları arasında birbirine eşit eleman olmadığı için sonuçta hep 0 bulunmuştur. `P==3` ile `Q==3` komutlarını deneyiniz, neler buluyorsunuz?

P>2 komutu ile sıfır ve birlerden oluşan bir vektör bulunur. Bu işlemin sonucunu başka bir vektörle işleme sokabiliriz. İşte basit bir örnek:

```
» yeni=Q-(P>2)
yeni =
     8     7     5     4     3     2     1     0    -1
```

Şimdi şu komutu dikkatle inceleyelim:

```
» yepyeni=Q-P>2

yepyeni =
     1     1     1     0     0     0     0     0     0
```

Görüldüğü gibi `yeni` ile `yepyeni` değişkenleri farklıdır. Birincide önce P>2 işlemi yapılıp bulunan Q dan çıkartılırken, ikincide önce Q dan P çıkartılmış daha sonra 2 ile karşılaştırma yapılmıştır.

Söz bu noktaya gelmişken MATLAB'da işlemler arasındaki öncelik sırasını belirtmekte de yarar vardır:

MATLAB operatörleri öncelik sırası

```
^   .^   '   .'
*   /    \   .*   ./   .\
+   -
:   >    <   >=   <=   ==   ~=
|   &
```

Bu tabloda öncelik satıra göre belirlenir. O halde üs alma işlemi çarpma işleminden önce yapılır. Her satırdaki işlemlerin birbirine göre önceliği yoktur. İkinci satırı göz önüne alalım. Bir işlemde önce çarpma varsa çarpma, bölme varsa bölme yapılır. 2*3-4/2*5 gibi bir ifadede işlem sırası çarpma, bölme, çarpma ve çıkarma şeklinde oluşur. Bütün bunların yanında parantez içi işlemlerin en büyük önceliğe sahip olduğunu hatırlayalım.

Aşağıdaki örnek sıfırla bölme işleminden kurtulmak için başvurulan pratik bir yöntemi ve basit bir uygulamayı göstermektedir:

```
Q=Q+(Q==0)*eps
Q =
  8.0000 7.0000 6.0000 5.0000 4.0000 3.0000 2.0000 1.0000 0.0000
```

Burada Q nün son elemanı olan 0 özel bir MATLAB sayısı olan eps ile değiştirilmiştir. eps nin değeri yaklaşık olarak 2.2204e-016 dır. Bunun sıfır olarak gözükmesi kullanılan format short dolayısıyladır. Şu komut durumu açıklamaya yetecektir:

```
» Q(9)
ans =
  2.2204e-016
```

x sıfıra giderken $\lim \frac{\sin(x)}{x}$ ifadesi 1 değerini alır. Bu bağıntıyı şu şekilde örnekleyebiliriz:

```
» x=(-3:3)/3

x =
   -1.0000   -0.6667   -0.3333         0    0.3333    0.6667    1.0000

» sin(x)./x
Warning: Divide by zero.
ans =

    0.8415    0.9276    0.9816       NaN    0.9816    0.9276    0.8415
```

sin(0)/0 tanımsız olduğu için bir uyarı alıyoruz. Şimdi 0 erine eps değerini yerleştirerek yeniden deneyelim:

```
» x=x+(x==0)*eps;
» sin(x)./x

ans =
    0.8415   0.9276   0.9816   1.0000    0.9816    0.9276    0.8415
```

Şimdi x=0 için sin(x)/x doğru bir şekilde elde edilmiştir.

U ve V matrislerini ele alalım:

```
U =
     1     2     4
     1     1     1
     2     3     1

V =

     2     2     2
     2     2     2
     2     2     2
```

Aşağıdaki örnekleri inceleyerek ilişki operatörleri bölümünü kapatabiliriz.

```
» U>V
ans =
     0     0     1
     0     0     0
     0     1     0

» U==V
ans =
     0     1     0
     0     0     0
     1     0     0

» U~=V
ans =
     1     1     1
     1     0     1
     0     1     1

» U>=V
ans =
     0     1     1
     0     0     0
     1     1     0

» U>1
ans =
     0     1     1
     0     0     0
     1     1     0


» U<=1
ans =
     1     0     0
     1     1     1
     0     0     1

» Dd=U(:);D=Dd'
D =
     1     1     2     2     1     3     4     1     1

» D~=1
ans =
     0     0     1     1     0     1     1     0     0
```

Bir vektör içindeki negatif sayıları sıfırla değiştirmenin basit bir yolu olarak şu örneği inceleyebiliriz.

```
y =
     1    -3     2     0     7     4    -1     5     4

» p=(y>=0).*y
p =
     1     0     2     0     7     4     0     5     4
```

Şimdi de sıfırları 0.5 ile değiştirelim:

```
» p=p+0.5*(p==0)
p =
     1   0.5     2   0.5     7     4   0.5     5     4
```

Mantıksal Operatörler

Önce bunları sıralayalım, daha sonra örnekleri ele alalım.

Operatör	Tanım ve açıklama (a ve b birer sayıyı gösteriyor)	
&	VE	a&b, a ve b nin her ikisi de sıfırdan farklıysa sonuç 1, biri sıfırsa sonuç 0
|	VEYA	a|b, a ve b den biri sıfırdan farklıysa sonuç 1, her ikisi de sıfırsa sonuç 0
~	DEĞİL	~a, a sıfırsa sonuç 1, sıfırdan farklıysa sonuç 0

Tablodaki tanımlar a ve b birer sayı kabulüyle verilmiştir. Bu operatörlerin matris ve vektörlerle de kullanılabileceklerini önemle hatırlatalım. Bu durumda operatörlerin matrislerin karşılıklı elemanları üzerinde işleme girecekleri açıktır. A ve B matrislerini şöyle tanımlayalım:

```
A =
     1     3    -2
     0     1     4
     0    -1     0

B =
     1     0     3
     0     2    -2
    -1     0     1
```

Yukarıdaki operatörleri A ve B matrislerini göz önüne alarak uygulayalım. Uygulamanın karşılıklı elemanlar üzerinde olduğunu bir kere daha hatırlayalım.

```
» A&B
ans =
     1     0     1
     0     1     1
     0     0     0

» A|B
```

```
ans =
     1     1     1
     0     1     1
     1     1     1
» ~A
ans =
     0     0     0
     1     0     0
     1     0     1
```

```
» ~B
ans =
     0     1     0
     1     0     0
     0     1     0
```

Ayrıca şu örnekleri de ilave etmek faydalı olacaktır.

```
» [1&1 1&0 0&1 0&0]
ans =
     1     0     0     0

» [1|1 1|0 0|1 0|0]
ans =
     1     1     1     0

» ~[1 0 -2 2]
ans =
     0     1     0     0
```

Mantıksal fonksiyonlar

Fonksiyon	**Tanım ve açıklama (a ve b birer sayıyı gösteriyor)**
xor(a,b)	a ve b den yalnız biri sıfırdan farklıysa sonuç 1, her ikisi de sıfırsa veya her ikisi de sıfırdan farklıysa sonuç 0
any(y)	y vektörünün içinde sıfırdan farklı bir eleman varsa sonuç 1, aksi takdirde 0
all(y)	y vektöründeki bütün elemanlar sıfırdan farklıysa sonuç1, aksi takdirde 0

any ve all fonksiyonları matrisler için de kullanılabilir. Bu durumda matrisin her sütunu için bu fonksiyonlar ayrı ayrı uygulanmış olur. Örnekleri dikkatle izlemek konunun anlaşılması için yeterli olacaktır:

```
A =
     1     3    -2
     0     1     4
     0    -1     0
```

```
B =
     1     0     3
     0     2    -2
    -1     0     1
```

```
» xor(A,B)
ans =
     0     1     0
     0     0     0
     1     1     1

» any(A)
```

```
ans =
     1     1     1


» any(any(A))
ans =
     1

» all(A)
ans =
     0     1     0

» all(all(A))
ans =
     0
```

Şimdiye kadar anlatılanlar özellikle if ve while gibi komutlarla birlikte kullanılırlar. İlerde programlama konusunu tartışırken bu hususlar üzerinde ayrıca durulacaktır. Ancak konuyu kapatmadan boş matris ve find komutu üzerinde durmakta yarar vardır. Boş matris elde etmek için kullanılacak komut şudur:

```
» bm=[]
bm =
     []
```

Boş matris sıfırlar matrisi değildir. Bunun için yukarda tanımlanmış olan bm boş matrisinin boyutlarına bakalım:

```
» size(bm)
ans =
     0     0
```

Şimdi de bir sıfırlar matrisi oluşturup bunun boyutlarını bulalım.

```
» sm=zeros(2)
sm =
     0     0
     0     0

» size(sm)
ans =
     2
```

Boş matrisin çok kullanıldığı bir durum küçük bir program parçacığı olarak aşağıda verilmiştir. (Bu program parçasını MATLAB komut satırındayken de yazabilirsiniz, aşağıda ilk satırın başındaki MATLAB komut işaretine dikkat ediniz)

```
» if length(bm)==0
for k=1:3
bm=[bm k^2]
end
end

bm =
     1
bm =
```

```
     1     4
bm =
     1     4     9
```

İlişki operatörlerini kullanarak matris ve vektörlerden belli şartları sağlayan elemanları seçmek mümkündür. İşte çarpıcı bir örnek:

```
» sec=[-1 5 0 2 8 4 3 6 -4 0 1 2]
sec =
  -1     5     0     2     8     4     3     6    -4     0     1   2
```

Şimdi sec adlı değişkende 1 den büyük ve 5 e eşit ya da küçük olanları tespit etmeye çalışalım:

```
» bs=(sec>1)&(sec<=5)
bs =
   0     1     0     1     0     1     1     0     0     0     0   1
```

Şimdi bs yi kullanarak yukarıdaki şartları sağlayan sayıları çıkartalım:

```
» sec(bs)
ans =
     5     2     4     3     2
```

Şu husus ilginç bir noktayı açıklamaya yarayacaktır: Bir değişkenin mantıksal bir işlemin sonucu olup olmadığını anlamak için `islogical` komutu kullanılır. Sonuç 1 ise önerme doğru, 0 ise yanlıştır. Yukarıdaki bs değişkeni mantıksal bir işlem sonucunda elde edilmiştir ve bu sebeple sec adlı değişkenle birlikte kullanılabilmiştir.

```
» islogical(bs)
ans =
     1
```

Öte yandan mantıksal bir işlemin sonucu olmayan 0 ve 1 lerden oluşmuş vektörler eleman seçiminde kullanılamazlar. Bir örnekle açıklamaya çalışalım:

```
qq =
     8    -1    -2     7     1     0

» birvesıfır=[0 1 1 0 0 1];
» qq(birvesıfır)
??? Index into matrix is negative or zero.  See release notes on
changes to logical indices. (Matris indisleri bir veya sıfır.
Mantıksal indislere ilişkin değişiklikler için çıkartılan notlara
bakınız)

» islogical(birvesıfır)
ans =
     0

» bs1ve0=qq<=0
bs1ve0 =
     0     1     1     0     0     1
» qq(bs1ve0)
ans =
```

```
    -1    -2     0

» islogical(bs1ve0)
ans =
     1

» birvesıfır==bs1ve0
ans =
     1     1     1     1     1     1
```

Şu hususu not edelim: `birvesıfır` ile `bs1ve0` değişkenleri aynı sayılardan oluşmuştur, yani eşittirler. Ancak ikinci değişken mantıksal bir işlemin sonucuyken birincisi böyle bir işlemden elde edilmiş değildir, yani `islogical(bs1ve0)` için 1, `islogical(birvesıfır)` için 0 elde ediyoruz. Dolayısıyla 1 ve 0 lardan oluşan sayıların matris indisi olarak kullanımlarında farklı sonuçlar elde edilmiştir.

Bir başka önemli fonksiyon olan find geniş bir kullanım alanına sahiptir. find komutu ile bir matris içinde sıfırdan farklı olan elemanların indisleri elde edilir. İşte birkaç örnek:

```
» p =[     8     0    -2     7     1     0];
» find(p)
ans =
     1     3     4     5
```

p nin içinde sıfırdan farklı elemanların indisleri elde edildi. Yani p nin 1 3 4 ve 5 inci sıradaki elemanları sıfırdan farklıdır.

```
» p<=1
ans =1   0   1   1
» find(p<=
     0     1     1)
ans =
     2     3     5     6
```

GRAFİK

MATLAB'da grafik çizmek oldukça basittir. İki ve üç boyutlu grafik çizmek için kullanımı çok kolay bir çok komut vardır. Bunlardan bazıları şunlardır:

GRAFİK	
plot	lineer x - y grafiği
loglog	x ve y eksenleri logaritmik
semilogx	yarı logaritmik (x ekseni logaritmik)
semilogy	yarı logaritmik (y ekseni logaritmik)
polar	polar grafik
mesh	3 - boyutlu grafik
contour	contour grafik
bar	çubuk şeklinde grafik

stairs	basamaklı grafik

Grafik elde edildikten sonra eksenlerin hangi büyüklükleri gösterdiğini ve grafiğin adını vermek, ızgara çizgilerini çizmek ve grafiğin istediğimiz bir yerine bir metni yerleştirmek mümkündür. Bunun için kullanılan komutlar şunlardır:

title	grafiğin adı
xlabel	x ekseninin etiketi
ylabel	y ekseninin etiketi
grid	ızgara çizgileri
text	grafik üzerine metin
gtext	fareyle konumlandırılan metin

Diğer grafik kontrol komutları:

axis	manuel eksen ölçeği
hold (on-off)	ekrandaki grafiği tut
subplot	birden çok grafik için ekranı böl
ginput	mouse konumlu girdi
figure	yeni grafik penceresi aç

Şimdi bu komutların bazılarının kullanılışları üzerinde durabiliriz. Burada ele alınmayanlar için help komutundan yararlanılabileceğini yineleyelim.

Bir deney yaptığımızı ve aşağıdaki değerleri bulduğumuzu düşünelim.

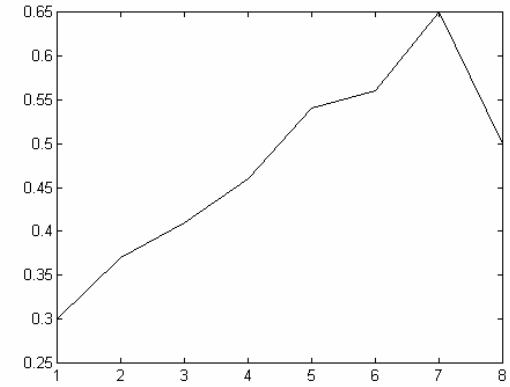
Deney No	**Ölçülen Değer**
1	0.3
2	0.37
3	0.41
4	0.46
5	0.54
6	0.56
7	0.65
8	0.50

Şimdi bu tabloyu bir grafik halinde göstermek için önce ölçülen değerleri d adlı bir vektörde toplayalım:

```
» d=[0.3 0.37 0.41 0.46 0.54 0.56 0.65]
```

Bunu, x ekseni deney numarasını, y ekseni ölçülen değerleri göstermek üzere grafik halinde görmek için şu komutu kullanabiliriz.

```
» plot(d)
```



Şekil 1 Ölçüm sonuçlarının deney numarasına göre grafiği

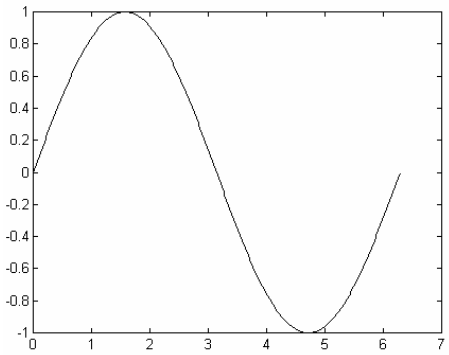
Görüldüğü gibi x ekseninde bu eksen için herhangi bir belirleme yapılmadığı halde 1 den 8 e kadar sayılar vardır. Bu, kısaca programın eğriyi d deki her değeri indis numarasına göre göstererek çizdiği şeklinde açıklanır. Aynı grafiği elde etmek için `plot(1:8,d)` komutu da kullanılabilirdi.

Şimdi de y=sin(x) fonksiyonunun grafiğini elde edelim. Önce peryodun 2π olduğunu hatırlayalım. 0 ile 2π arasında x için muhtelif değerler seçmek gerekir. Bu değerler grafiğin yatay eksenini oluşturacaktır. Daha sonra her x değerine karşı gelen sinüs değeri hesaplanmalıdır. Bunlar da grafiğin y eksenini oluşturacaktır. Her (x_i,y_i) çifti eksen takımında işaretlenerek birer çizgi ile birleştirilirler. O halde hassas eğriler elde etmek için olabildiğince çok (x_i,y_i) çiftine ihtiyaç vardır. Hatırlanacağı gibi trigonometrik fonksiyonların argümanlarını radyan olarak vermek gerekmektedir. Ancak bunları derece olarak değerlendirmek daha kolaydır. Dolayısıyla sinüs fonksiyonunun grafiğini çizmek için önce birer derece aralıkla 0 dan 360 dereceye kadar uzanan bir vektör elde edip daha sonra bunu radyana çevirmek mümkündür. Aşağıdaki örneği dikkatle takip edelim:

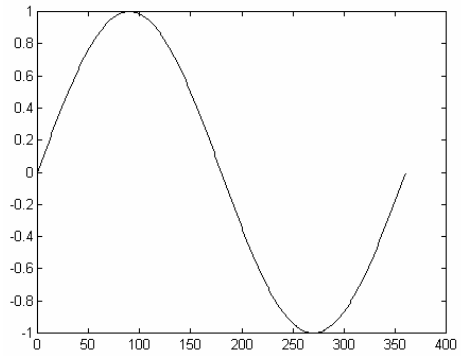
```
» xd=0:360;
» x=xd*pi/180;
» y=sin(x);
» plot(x,y);
```

Grafiğin yatay eksenindeki değerler radyan olarak gösterilmiştir. Bunları derece olarak göstermek için gerekli komut şöyledir:

```
» plot(xd,y);
```



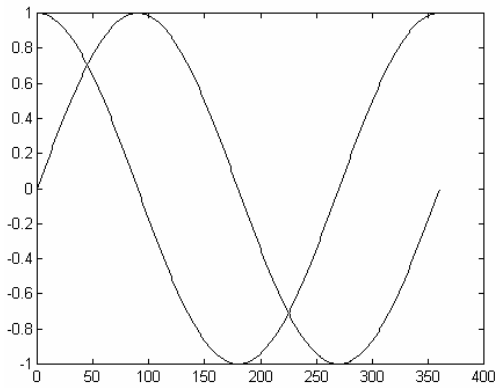
Şekil 2 Sinüs fonksiyonunun grafiği. Yatay eksendeki açılar radyan



Şekil 3 Sinüs fonksiyonunun grafiği. Yatay eksendeki açılar derece

Şimdi sinüs ve kosinüs fonksiyonlarını aynı grafik üzerinde gösterelim:

```
z=cos(x);
plot(xd,y,xd,z)
```



Şekil 4 Sinüs ve kosinüs fonksiyonları aynı grafik üzerinde

Bu örnek aynı anda birden çok grafiğin çizilebileceğini göstermektedir. Bunun başka bir yolu `hold` fonksiyonunu kullanmaktır. `hold` fonksiyonu mevcut bir grafiği koruyarak bir sonraki çizimin aynı grafik üzerinde gösterilmesini sağlar. Bu moddan çıkmak için `hold off` komutunu kullanmak gerekir.

```
» plot(xd,y)
» hold
Current plot held
» plot(xd,z)
```

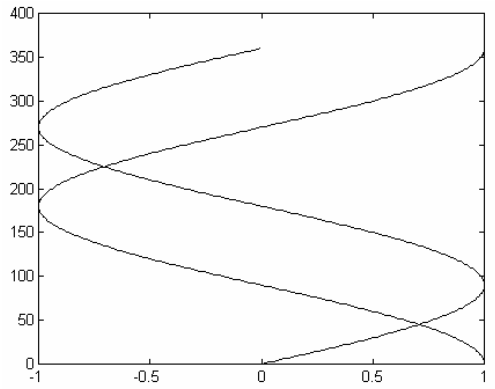
Sonuç yukarıdaki grafiğin aynısı olacaktır. Daha fazla detay için `help hold` komutunu kullanınız.

Eğer plot komutunun argümanlarından birisi matris diğeri vektör ise `plot` komutu yatay eksene vektörü, düşey eksene matrisin sütunlarını yerleştirerek çizim yapar.

```
m=[y;z];
plot(x,m)
```

Sonuç bir önceki grafiğin aynısıdır.

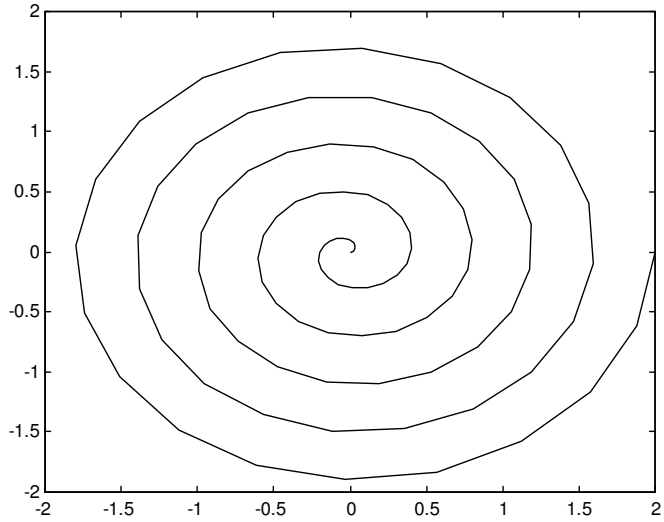
Eğer argümanların yeri değiştirilirse, şekil 90 derece dönecektir. Bunun için `plot(m,x)` komutunu kullanmak gerekir. Görünüm Şekil 5 teki gibidir.



Şekil 5 Sinüs ve kosinüs fonksiyonları. Açılar düşey eksende.

`plot` komutunun kullanımıyla ilgili başka detaylar da vardır. Daha fazla bilgi için `help plot` komutunu deneyiniz.
`linspace(m,n)` komutu ile m ile n sayıları arasında lineer olarak eşit aralıklı dağılmış 100 adet sayı elde edilir. `linspace(m,n,k)` komutu ile sayı adedini k olarak belirleme imkanı vardır. Bu fonksiyonları kullanarak karmaşık sayıların grafik komutuyla kullanımlarını ele alalım.

```
» clear i                          % i kompleks
» r=linspace(0,2);                 % r vektörünü oluşturur
» theta=linspace(0,10*pi);         % theta açısını oluşturur
» [x, y]=pol2cart(theta,r);        % polar koordinatları kompleks
» z=x+i*y;                         % sayıya dönüştürür
» plot(z)                          % kompleks z yi çizer
```



Şekil 6 plot(z) veya plot(real(z),imag(z)) komutları, z kompleks.

Şimdi çizgiler, işaretler ve renklere ilişkin hususları gözden geçirdikten sonra grafikle ilgili bazı özelliklere tekrar döneceğiz.

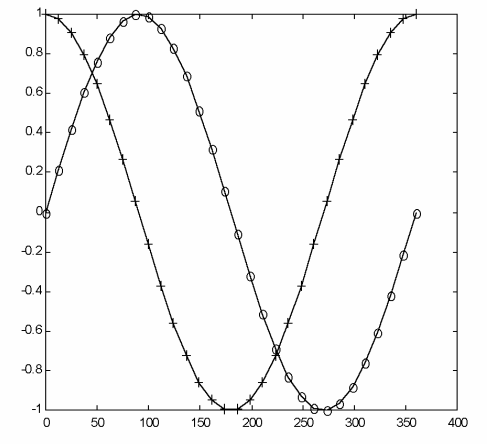
Çizgiler, işaretler ve renkler:

Yukarıdaki örneklerde, MATLAB grafikleri çizerken kesiksiz renkli çizgi seçmiştir (Burada siyah olarak basılmıştır). `plot` komutu içinde ilave argümanlar kullanarak çizgi biçimini ve renkleri değiştirmek mümkündür. Bunun için kullanılabilecek seçenekler şunlardır:

Sembol	Renk	Sembol	Çizgi Biçimi	Sembol	Çizgi Biçimi
y	sarı	.	nokta	s	Kare
m	mor	o	çember	d	Elmas
c	açık mavi	x	x işareti	v	üçgen (yukarı)
r	kırmızı	+	artı	^	üçgen (aşağı)
g	yeşil	*	yıldız	<	üçgen (sağ)
b	mavi	-	kesiksiz çizgi	>	üçgen (sol)
w	beyaz	:	noktalı çizgi	p	Beşgen
k	siyah	-.	çizgi-nokta	h	Altıgen
		- -	kesikli çizgi		

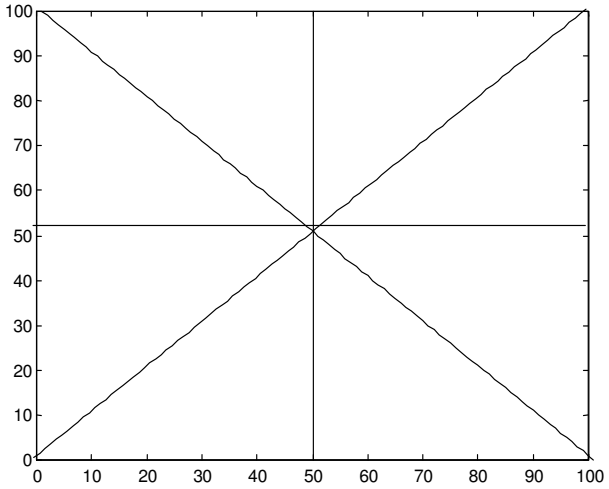
Baskı tekniği burada renkli grafik gösterimine imkan vermemektedir. Ancak aşağıdaki komutu bilgisayarınızda kullanırsanız hem renklerin, hem de işaretlerin nasıl kullanıldığını görebilirsiniz. xd, y ve z nin çalışma sahanızda bulunduğundan emin olunuz. Şekil 7, siyah ve beyaz kullanılarak elde edilen bu grafiği göstermektedir.

```
>>plot(xd,y,'g:',xd,z,'r-',xd,y,'wo',xd,z,'c+')
```



Şekil 7 Renkler ve işaretlerin kullanılışı.

```
» plot(0:100,'c')
» hold
Current plot held
» plot(100:-1:0,'b')
» plot(0:100,50*ones(1,101),'g')
» plot(50*ones(1,101),0:100,'m')
```



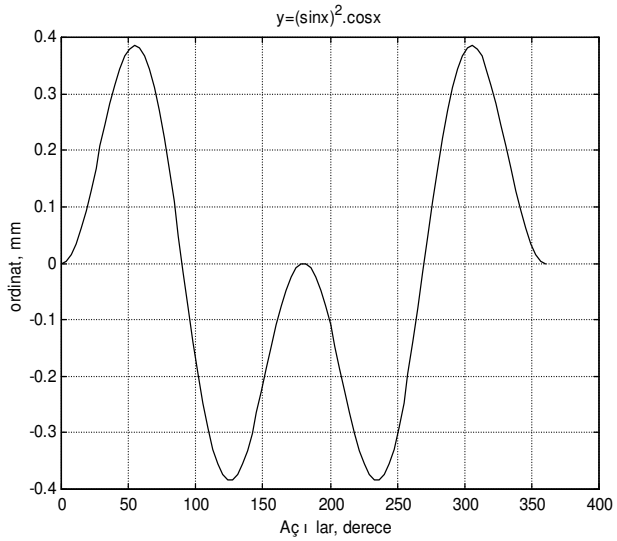
Şekil 8 Grafik için değişik bir örnek. Yatay ve düşey çizgilerin nasıl elde edildiğine dikkat ediniz. Bu çizgiler eksen çizgileri olarak kullanılabilirler.

Izgara çizgileri ve eksen etiketleri

Grafikleri kolay değerlendirme açısından ızgara çizgileri faydalıdır. Şimdi bir örnek fonksiyon alıp grafiğini elde delim ve ızgara çizgilerini `grid` komutuyla yerleştirelim. Bu komut `plot` komutuyla aynı satırda olabileceği gibi ayrı bir satır halinde de verilebilir.

$y=\sin^2 x.\cos x$

```
» x=linspace(0,2*pi);
» y1=sin(x);
» y2=cos(x);
» y=y1.^2 .*y2;
» plot(x*180/pi,y)
» grid,shg
```



Şekil 9 Izgara çizgileri ve eksen etiketleri

Bu grafiğin x eksenine "Açılar, derece", y eksenine "ordinat, mm" yazalım. Bunun için

```
» xlabel('Açılar, derece')
» ylabel('ordinat, mm')
```

komutları kullanılır. Bu iki komut tek satırda da verilebilir. Şeklin başlığı olarak da "$y=\sin^2 x.\cos x$" yerleştirelim. Bunun için de şu komutu kullanırız:

```
» title('y=(sinx)^2.cosx')
```

Parantez içindeki ifade ile şeklin başlığındaki ifadeyi karşılaştırınız, Şekil 9. Burada x ekseni için kullanılan ifadede Türkçe karakterler vardır, ancak bunlar zaman zaman beklenmedik sorunlara yol açmaktadır. Zorunlu olmadıkça şekil içinde Türkçe karakterler kullanmaktan kaçınmakta fayda vardır.

Grafik içine metin yerleştirmek

Hazırladığımız grafikler üzerine açıklayıcı bilgiler yerleştirmek istediğimizde kullanabileceğimiz üç komut vardır. Bunlar text, gtext ve legend komutlarıdır. Önce şu iki fonksiyonu tanımlayarak grafiklerini elde edelim.

$T_d=2500 + 300.\text{Sin}\theta$; $T_ç=A + 500.\text{Sin}2\theta$

İşte gerekli komutlar:

```
» q=(0:360)*pi/180;
» Td=2500 + 300*sin(q);   Tç=2500 + 500*sin(2*q);
» plot(q*180/pi,Td,'-.k',q*180/pi,Tç,'k')
» grid,shg
» axis([0 360 2000 3000]),shg
```

Grafik Şekil 10 da gösterilmiştir. `axis` komutu grafik eksenlerinin ölçeklerini ayarlamak için kullanılır. Genel yapısı `axis([xmin xmak ymin ymax])` şeklindedir. Burada axis komutu kullanılmadan x ekseninin limitleri 0 - 400 olarak belirlenmişken kullanıldıktan sonra 0 - 360 olarak değişmiştir. Bu değişiklikleri gözlemek için yukardaki komutların bazılarına shg ilave edilmiştir. Bu komut şeklin ekranda fare kullanılmaksızın aktif hale getirilmesini

sağlar. Ekranı ikiye bölüp yarısını matlab komut sayfası, yarısını grafik ekran olarak kullanarak da grid ve axis komutlarının etkilerini anında gözlemek mümkündür.

Önce eksen yazılarını ve başlığı yerleştirelim:

```
»xlabel('ACILAR, derece'),ylabel('Tork, Nm'),title('Kuvvet Analizi')
```

Şimdi grafik üzerinde (100,2950) konumuyla belirli yere "Makine Dinamiği" yazısını yerleştirmeye çalışalım. Bunun için text komutu kullanılır.

```
» text(100,2950, 'Makina Dinamiği')
```

O halde `text` komutunda ölçüler şekil üzerinden alınmalıdır.

`text` komutunda koordinatların verilmesi zarureti vardır. Bu külfetten kurtulmak için `gtext` komutu kullanılabilir. `gtext` komutu ile istenilen metin fare ile yerleştirilir. İşte örnek:

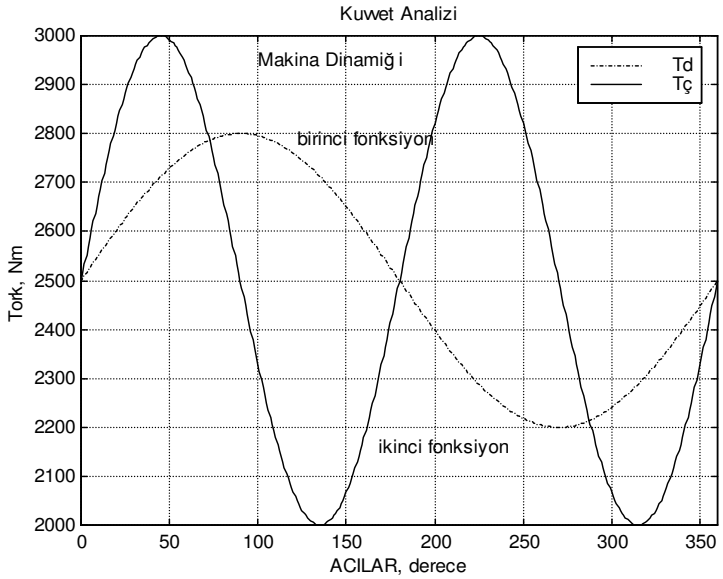
```
» gtext('birinci fonksiyon')
» gtext('ikinci fonksiyon')
```

Bu komutlar girildiğinde grafik kendiliğinden aktif hale gelir ve fare nerede tıklanırsa metin oradan başlanarak yerleştirilir. gtext komutu ile yerleştirilen bir metni değiştirme imkanı yoktur. Dolayısıyla kullanımda dikkatli olmak gereklidir.

`legend` komutu da grafik üzerine bir etiket yerleştirmek amacıyla kullanılabilir.

```
» legend('Td','Tç')
```

komutu ile Td ve Tç nin yerleştirilişi şekil üzerinden izlenebilir. Renkli grafikler halinde hangi rengin hangi fonksiyona ait olduğunun belirtilmesi legend komutu ile sağlanabilir. Aşağıdaki grafikte önce Td çizildiği için legend komutu önce kesikli çizgiyi, sonra kesiksiz çizgiyi göstermiştir (yukarıdaki plot komutuna bakınız). Siz grafiği renkli çizerek legend komutunu kullanmayı deneyiniz.



Şekil 10 text, gtext ve legend kullanarak grafik üzerine metin yerleştirilmesi

`legend` için genel yapıyı `legend('metin1','metin2',...)` olarak verebiliriz. `legend` ile yerleştirilen metin fare ile tutularak istenilen yere sürüklenebilir. Yeni legend komutu eskisini değiştirir. Bu komutun kullanımıyla ilgili başka detaylar da vardır. `help legend` ile detayları incelemek faydalı olacaktır.

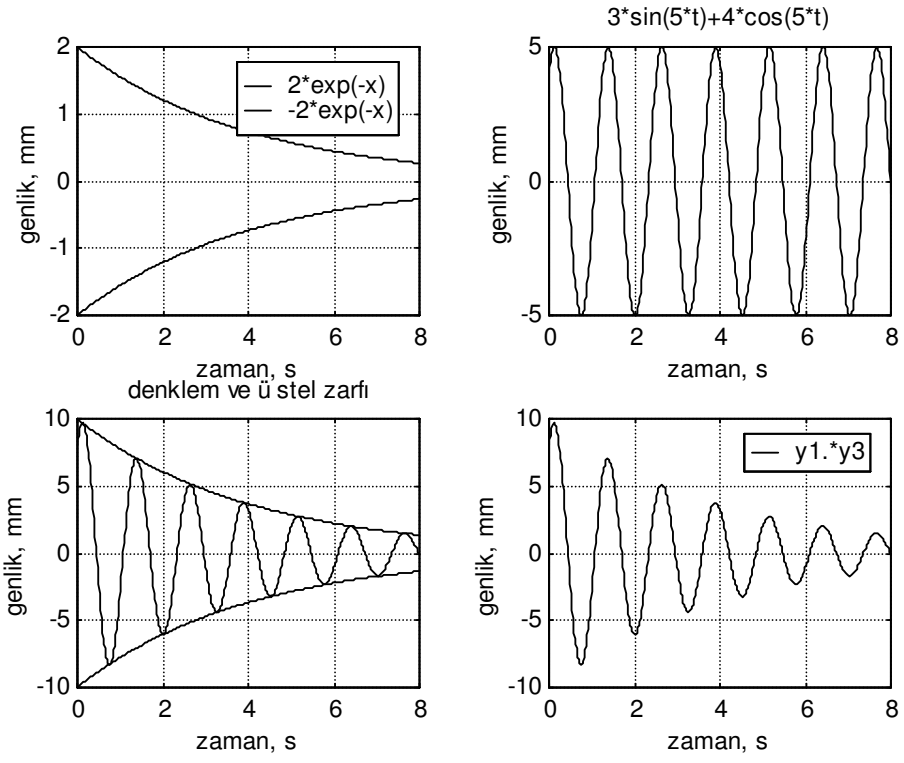
Subplot Komutu

Subplot komutu bir grafik ekranını bölmeye yarar. Kullanımı subplot(m,n,p) şeklindedir. Ekranı m tane satır n tane sütundan oluşan p tane grafik ekranı haline getirir. Mesela subplot(2,2,1) komutu ekranı dörde böler ve bunlardan birincisini aktif hale getirir. Şöyle bir tablo yararlı olabilir.

subplot(2,2,1)	subplot(2,2,2)
subplot(2,2,3)	subplot(2,2,4)

Aşağıdaki komutlar bir denklemin terimlerinin ayrı ayrı grafiklerini elde etmekte kullanılmıştır. Hangi grafiği aktif hale getirmek istiyorsak o ekranla ilgili subplot komutunun kullanıldığına dikkat ediniz. Şimdi, sönümlü titreşim hareketini gösteren şu denklemi alalım:

$$y = 2e^{-0.25t}(3\sin 5t + 4\cos 5t) = 10e^{-0.25t}\sin(5t + tg^{-1}\frac{4}{3})$$



Şekil 11 subplot kullanılarak elde edilmiş bir grafik

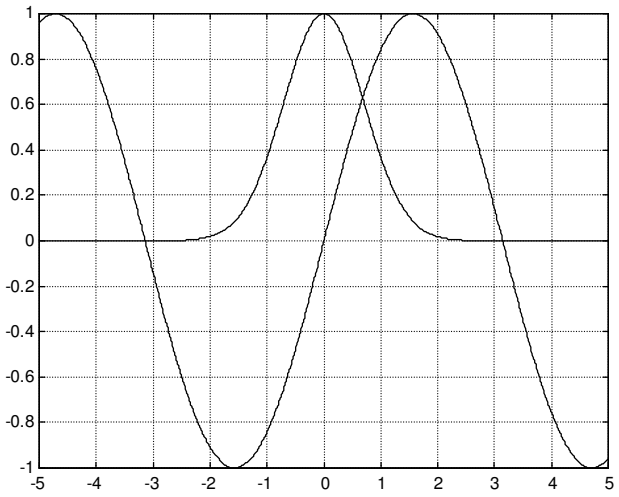
Bu denklemi, 0<=t<=8 aralığında çizelim.

```
» t=linspace(0,8,500);

» subplot(2,2,1)
» y1=2*exp(-0.25*t);   y2=-2*exp(-0.25*t);
» plot(t,y1,t,y2),shg
» legend('2*exp(-x)','-2*exp(-x)')
» xlabel('zaman, s'),ylabel('genlik, mm'),grid

» subplot(2,2,2)
» y3=3*sin(5*t)+4*cos(5*t);
» plot(t,y3),shg
» xlabel('zaman, s'),ylabel('genlik, mm'),grid
» title('3*sin(5*t)+4*cos(5*t)')

» subplot(2,2,3)
» plot(t,5*y1,t,5*y2,t,y1.*y3),shg
» xlabel('zaman, s'),ylabel('genlik, mm'),grid
» title('denklem ve üstel zarfı'),shg

» subplot(2,2,4)
» plot(t,y1.*y3),grid,shg
» xlabel('zaman, s'),ylabel('genlik, mm')
» legend('y1.*y3')

» subplot
```

Yalnız başına `subplot` komutu ekranın eski haline dönmesini sağlar. Yani `subplot` komutu ile `subplot(1,1,1)` komutları aynı işleve sahiptirler. Dördüncü grafikle ilgili işlemler tamamlandıktan sonra diyelim ikinciye dönmek mümkündür. İkinciyi aktif kılmak için `subplot(222)` yazmak gerekir. Sayılar arasına virgül koyma mecburiyeti yoktur.

Zoom Komutu

Zoom yakın plan gösterimi olarak anlamlandırılabilir. Bir grafiği elde ettikten sonra herhangi bir kısmına daha yakından bakmak için fare ilgili nokta üzerinde sol tuş kullanılarak tıklatılır. Sol tuşa her basış grafiğin o bölgesini büyütecektir. Eski hale gelmek için farenin sağ tuşu kullanılmalıdır. Şimdi şu denklemin köklerini aradığımızı düşünelim:



Şekil 12 zoom kullanarak bir denklemin köklerinin yaklaşık tayini

$$e^{-x^2} - \sin x = 0$$

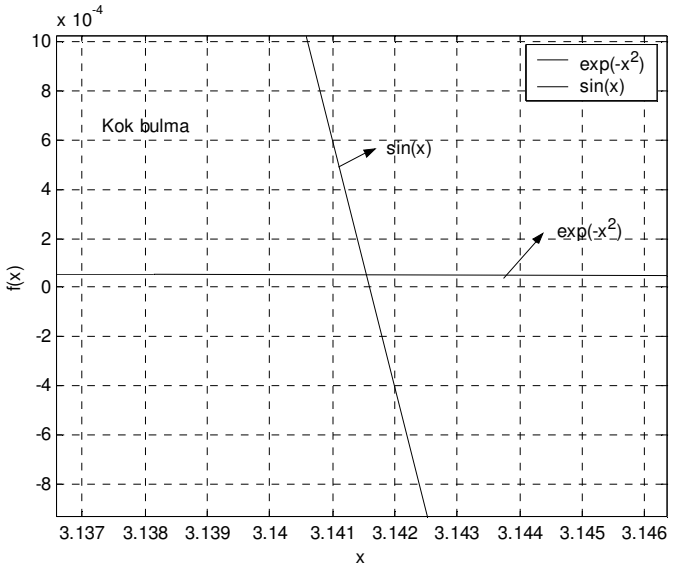
Üstel terim ve harmonik terim ayrı ayrı çizilerek kesim noktaları bulunursa denklemin kökleri bulunmuş olur. Ancak kökleri belirlemek için Şekil 12 deki hassasiyet yeterli değildir. zoom kullanarak x=3 ile x=4 arasındaki kök daha hassas bir şekilde görülebilir. Sonucu Şekil 13 den izleyiniz.

```
» x=linspace(-5,5,1000);

» y5=exp(-x.^2);
» y6=sin(x);
» plot(x,y5,x,y6)

» grid
» zoom
```

Şimdi grafikte eğrilerin x=3 ile x=4 arasındaki kesim noktasında fareyi tıklayarak kesim noktasını daha hassas bir şekilde belirleyebiliriz. Ard arda ikikere tıklayarak eğrinin ilk haline dönebiliriz. Farenin sağ tuşu da bu iş için kullanılabilir. MATLAB'ın 5.3 ve daha sonraki sürümlerinde zoom özelliği grafik ekran üzerinde de kullanılabilir. Ayrıca grafiğin edit edilmesine olanak sağlayan özellik, grafik etiketlerinin yerleştirilmesine, renklerin değiştirilmesine ve özel renk tanımlamaya, eksenlerin lineer ya da logaritmik olarak tanımlanmasına ve eksen ölçeklerinin değiştirilmesine imkan vermektedir. Izgara çizgilerinin de eksenlere göre seçilmesi yine grafiğin edit edilmesi ile mümkün olmaktadır.

```
» xlabel('x'), ylabel('f(x)')
» legend('exp(-x^2)','sin(x)')
```

Bu komutlardan sonra yine grafik ekran üzerindeki özellikleri kullanarak şekil içerisine metin yerleştirmek çizgi ve ok kullanmak mümkün olmaktadır. Şekil 13 deki oklar, 'Kok bulma', 'sin(x)' ve 'exp(-x^2)' ifadeleri bu şekilde yerleştirilmiştir.



Şekil 13 zoom kullanarak köklerin daha hassas tayini. Bu şekil, şekil 12'deki x=3 ile X=4 arasının büyütülmüşüdür.

Şekil 13, kökün yaklaşık 3.1415 olduğunu göstermektedir. Bunun da hemen hemen pi sayısını verdiğini not ediniz. Şekil 12 de şeklin simetrisinden bir diğer kökün –3.1415 olduğunu görebiliriz. 0 ile 1 arasındaki kök ise yine `zoom` komutunu veya grafik üzerindeki düğmeleri kullanarak 0.6806 olarak elde edilir.

ginput

Grafik üzerindeki noktaların koordinatını elde etmek için kullanılan bir komuttur.

```
» [x,y]=ginput(n)
```

komutuyla grafik üzerinde bulunan n tane noktanın koordinatlarını fareyi kullanarak elde etmek mümkündür. n noktayı tamamlamadan işlemi bitirmek için enter tuşunu kullanmak gerekir. Bu komut n parametresini vermeden kullanılırsa enter tuşuna basana kadar x ve y için veri toplamaya devam eder. Şekil 12 ya da Şekil 13'deki kesim noktalarının koordinatlarını elde etmek için bu komutu deneyiniz. Hassasiyetin nasıl değiştiğini her iki şekilde de kesim noktalarının koordinatlarını karşılaştırarak inceleyiniz. Bu amaçla, Şekil 12 için `[x1,y1]=ginput(3)` ve Şekil 13 için `[x2,y2]=ginput(3)` ile elde edilen x1 ve x2 değerlerini kullanabilirsiniz.

Bir grafik çizdikten sonra onu kaybetmeden başka bir grafik elde etmek için figure komutunu kullanmamız gerekir. İşte örnek: İlk komut bütün grafik pencerelerini kapatmaya yarar.

```
» close all
» x=pi/180*linspace(0,360);

» y1=sin(x).^2;
» plot(x,y1)
```

Şimdi eskisi kalmak üzere yeni bir grafik elde edelim.

```
» figure
» y2=sin(x);
» plot(x,y2)
```

Bir grafik daha oluşturup öncekileri de muhafaza etmek için

```
» h3=figure;
» plot(x,y1.*y2)
```

Daha önce iki grafik bulunduğu için h3'ün değeri 3 dür. Şimdi hangi grafiği aktif hale getirmek istiyorsak ya fareyi onun üzerine sürükleyip tıklamalıyız ya da aşağıdaki komutları kullanmalıyız.

```
» figure(1)
» figure(2)
» figure(h3)
```

Bu komutlar grafikleri aktif hale getirmek için kullanılır. Pencerelerden diyelim ki ikincisini kapatmak istiyoruz. Bunun için `close(2)` komutunu kullanmak gerekir. Kapatmaksızın bir grafik penceresinin sadece içeriğini silmek için `clf` komutu kullanılır.

Matlab komut sayfasını silmek için `clc` komutu kullanılır.

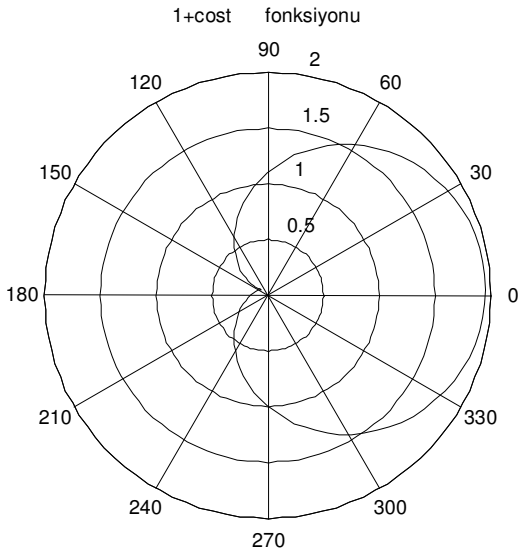
Polar Grafik

Polar koordinatları kullanarak çizilen grafikler bazı durumlarda çok daha kullanışlı olmaktadır. Bu tür grafikler `polar(θ,r)` komutu ile çizilir. θ radyan olarak verilmelidir. İşte bir örnek:

$r = 1 + \cos(t)$

fonksiyo

```
» t=li
» r=1+
» pola
```

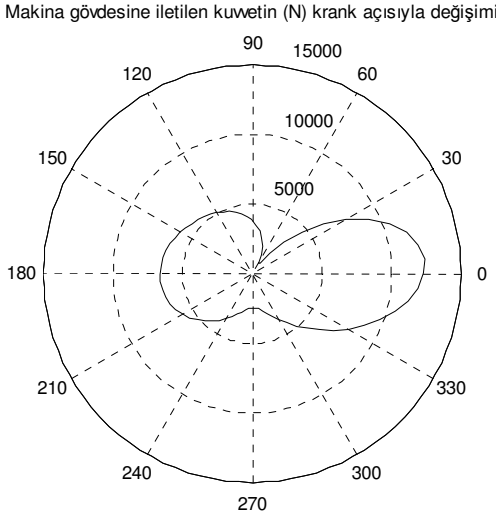


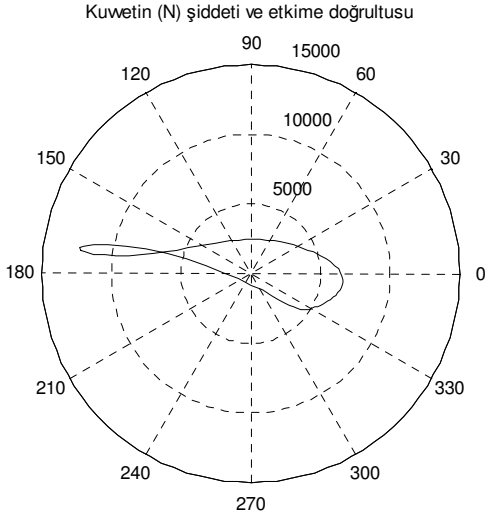
Polar Grafik

Polar grafik, makine tasarımında kinematik büyüklükler değiştikçe farklı değerler alan bir bileşke kuvvetin değişimini izlemekte etkin bir biçimde kullanılabilir. Yukarıdaki fonksiyonu plot komutunu kullanarak da elde ediniz.

Aşağıda bir dört kol mekanizmasında gövdeye iletilen kuvvetin yeğinliğinin (şiddetinin) krank açısıyla değişimi polar fonksiyonu kullanılarak gösterilmiştir. Açılar radyan olarak girilmekte ancak polar fonksiyonu bunu derece olarak göstermektedir. Açıların 0 ile 360 derece arasında gösterildiğine dikkat ediniz. İçten dışa doğru da kuvvetin yeğinliği artmaktadır. Bu şekle bakarak 120 derecelik krank açısında yaklaşık 5000 N luk bir kuvvetin gövdeye iletildiğini söyleyebiliriz. Ancak bu kuvvetin hangi doğrultuda etkidiğini bu şekle bakarak söyleyemeyiz.

Bir sonraki şekilde ise gövdeye iletilen kuvvetin yeğinliği ve bunun doğrultusu hakkında bilgiler verecek bir başka polar grafik gösterilmiştir. Bu şekle bakarak kuvvetin hangi krank açısında ilgili değeri aldığını belirtemeyiz. Sadece kuvvetin gövdeye hangi doğrultuda etki ettiğini görebiliriz. Son şekil ilk iki şekli aynı grafik üzerinde göstermektedir.





Logaritmik Grafik

Üç türlü logaritmik grafik imkanı vardır.

semilogx(x,y): x ekseninde 10 tabanlı logaritmik eksen kullanılır ve plot(log10(x),y) ile aynı sonucu verir.
semilogy(x,y): y ekseninde 10 tabanlı logaritmik eksen kullanılır ve plot(x,log10(y)) ile aynı sonucu verir.
loglog(x,y): x ve y eksenlerinde 10 tabanlı logaritmik eksen kullanılır ve plot(log10(x),log10(y)) ile aynı sonucu verir.

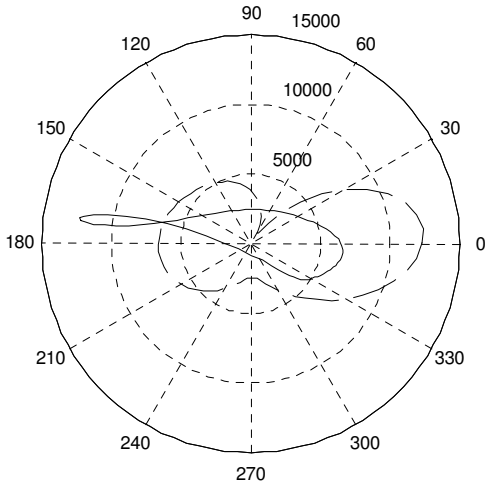
Konuya iyi bir örnek, radyan frekansı ω, genliği F olan harmonik bir zorlama altında hareket eden tek serbestlik dereceli bir titreşim sistemi göz önüne alınarak verilebilir. Frekans oranı r ile gösterilirse büyütme faktörü (M diyelim), k yayın direngenliği, ξ sönüm oranı olmak üzere

$$\frac{y}{(F\ /\ k)} = \frac{1}{\sqrt{(1-\frac{\omega^2}{\omega_0^2})^2 + 4\xi^2\frac{\omega^2}{\omega_0^2}}}$$

denkleminden hareketle,

$$M = \frac{1}{\sqrt{(1-r^2)^2 + 4\xi^2 r^2}}$$

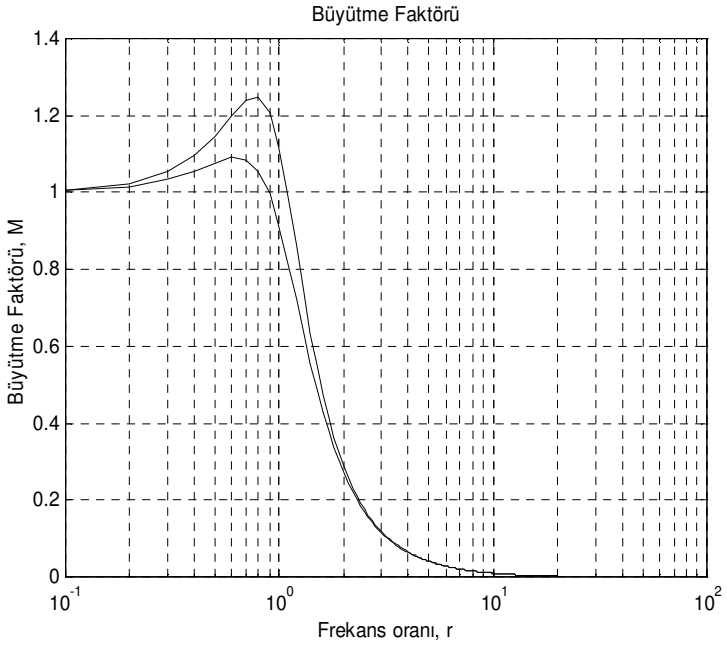
şeklinde yazılabilir. Şimdi logaritmik ölçekle çeşitli grafikler elde edebiliriz. Bunlara Bode diyagramları denildiğini de hatırlayalım.



Gerekli komu

```
» ksi=0.3;
» r=0:0.1:
» M=1./sqr
» semilogx
```

ksi=0.2 için d
Logaritmik öl



Programlamaya Giriş

Bazı işlemleri ardışık olarak ve çok sayıda yapmak gerekir. Bu durumda işlemlerin bir program şeklinde bilgisayara tanıtılması gerekir.

Şimdi basit bir problemle programlamayı tanıtmaya çalışalım. İki vektör düşünelim. a=[1 7 9 11], b=[2 8 10 12] olsun. Şimdi bu iki vektörün elemanlarını sırayla herbirinden birer eleman alarak yeniden sıralayıp yeni bir vektör elde edelim. Bu işlemi a ve b nin eleman sayısı kaç olursa olsun yapacak bir dizi işlemi bilgisayara tanıtmak için bir 'program' yazmak gerekir. Aşağıda böyle bir program gösterilmiştir. Programda bir çok döngü ve akışı kontrol eden deyimler vardır. Bunları da sırasıyla inceleyeceğiz.

```
% sirayakoy.m

% Bu program önce a sonra b vektöründen birer eleman alır
% Daha sonra bu işleme devam eder.
% a=[3 5 13 11] ve b=[4 6 14 12] ise c=[3 4 5 6 13 14 11 12] olarak bulunur.
% Program a ile b nin eleman sayısı farklı ise bir uyarı verecektir.

a=input(' a vektörünü giriniz ');
b=input(' b vektörünü giriniz ');

if length(a)~=length(b)
   error(' a ile b nin eleman sayıları farklı olmamalı')
end

c=[];

for i=1:size(a,2)
   c=[c a(i) b(i)]
end
```

Programın detaylarını göstermeden önce a ve b nin farklı bazı değerleri için elde edilen sonuçları gösterelim.

```
» sirayakoy
 a vektörünü giriniz [3 5 13 11]
 b vektörünü giriniz [4 6 14 12]

c =
     3     4
c =
     3     4     5     6
c =
     3     4     5     6    13    14
c =
     3     4     5     6    13    14    11    12
```

Programın sondan ikinci satırını şöyle yazdığımızı varsayalım, yani bir noktalı virgül ekleyelim, `c=[c a(i) b(i)];`
Şimdi programı bir daha çalıştıralım.

```
» sirayakoy
 a vektörünü giriniz [3 5 13 11]
 b vektörünü giriniz [4 6 14 12]
```

Görüldüğü gibi programın adımları gözükmüyor. Üstelik sonucu görmek için c yi sormamız gerekecektir:

```
» c
c =
```

```
     3     4     5     6    13    14    11    12
```

Programlamanın detaylarını daha sonraya bırakarak yukardaki programın bazı özelliklerini açıklamaya çalışalım.

Açıklama satırları (% işareti ile başlayan satırlar)

sirayakoy adlı programda % işareti ile başlayan satırlar programı kullanacaklar için açıklamaları içermektedir. Bu işaretle başlayan satırlar işlem satırları değil açıklama satırlarıdır. Özellikle programın nasıl çalıştıracağı, girdilerin nasıl düzenleneceği, çıktıların neler olduğu gibi hususlar bu şekilde belirtilir.

input Komutu

Daha sonraki iki satırda input komutuyla programa veri girişi sağlanmaktadır. input komutunun kullanılışına ilişkin bazı örnekler gösterelim.

```
» M=input('bir sayı giriniz M=')
bir sayı giriniz M=10
M =
    10
```

Noktalı virgülün (;) etkisini aşağıdaki örnekte izleyelim:

```
» q=input('q matrisini giriniz q=');
q matrisini giriniz q=[2 -9;0.67 2*1.92]
```

input komutu birden fazla değişkeni tanımlamak için de kullanılabilir:

```
» [m,n]=input('q matisinin boyutunu belirtiniz ')
q matisinin boyutunu belirtiniz size(q)
m =
     2
n =
     2
```

Sayısal olmayan bir veriyi okutmak için iki yol vardır: Birincisi veriyi tırnak içinde yazmaktır:

```
» kedikap=input('kedi mi büyük kaplan mi? ');
kedi mi büyük kaplan mi? 'kaplan'
```

İkincisi verinin karakter tipi (string) veri olduğunu input komutu bünyesinde tanımlamaktır:

```
» takim=input('kedi mi büyük kaplan mi? ','s');
kedi mi büyük kaplan mi? kaplan
```

Şimdi sirayakoy programındaki diğer komutları ele alalım:

Koşul deyimleri

MATLAB'da kararlar if deyimi kullanılarak verilir. Genel kullanımı şöyle açıklanabilir:

```
if mantıksal bir ifade
   deyimler
end
```

Konunun başındaki programa dönelim. Eğer a ile b nin boyutları aynı değilse (~=), yani önerme doğruysa, başka bir ifade ile `length(a)~=length(b)` işleminin sonucu 1 ise if deyiminin altındaki ifadeler işlenecek, aksi takdirde if deyiminin altındaki ifadeler göz önüne

alınmayacaktır. Unutmayalım ki ‘mantıksal bir ifade’nin sonucu ya 1 ya da 0 dır. Bu sonucun 1 ve 0 lardan oluşan bir dizi olabileceğini de not edelim. İşte bir örnek:

Bir matris düşünelim. Bu matrisin birinci satırındaki bütün elemanlar 10 dan büyükse ‘matris işe yarar’, değilse ‘matris işe yaramaz’ yazan bir program parçacığı oluşturalım. Matris ‘işe yaramaz’ ise birinci satırın bütün elemanlarını 10 ile çarpalım. Programın adı exif olsun.

```
%exif
if A(1,:)>10
   'matris işe yarar'
else
   'matris işe yaramaz'
   A(1,:)=A(1,:)*10;
end
A
```

Şimdi programı çalıştıralım. Bunun için önce bir A matrisi tanımlayalım:

```
A =

    10     9     8     9
     2     0     4     7
     6     5     6     2
     5     0     8     4

» exif
ans =
matris işe yaramaz
A =
   100    90    80    90
     2     0     4     7
     6     5     6     2
     5     0     8     4
```

A matrisinin tanımını program içinde de yapabileceğimizi not edelim. Burada if else yapısını da tanımış oluyoruz. Eğer if deyiminin izleyen önerme doğruysa hemen altındaki komutlar yerine getiriliyor ve end deyiminin altındaki komutlara sıra geliyor; önerme doğru değilse if deyimin altındaki komutlar atlanıyor ve else deyiminin altındaki komutlar yerine getiriliyor. Bunların değişik bir kullanılışı için de şu örneği inceleyelim:

```
%exifelse
disp('if elseif else ornegi')
disp('')
n=size(a,2);
for i=1:n
   for j=1:n
      if i==j
        a(i,j)=2;
      elseif abs(i-j)==1
        a(i,j)=-1;
      else
        a(i,j)=0;
      end
   end
end
disp('a matrisi')
disp(' ')
disp(a)
```

Bu program bir a matrisinin diyagonal elemanlarını 2, diyagonalin altındaki ve üstündeki elemanları –1, diğerlerini sıfır yapmaktadır. Önce bir a matrisi tanımlayalım:

```
a =
   100    90    80    90
     2     0     4     7
     6     5     6     2
     5     0     8     4
```

Şimdi programı çalıştırabiliriz.

```
» exifelse
if elseif else ornegi
a matrisi

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

Programdaki `disp` komutu parantez içindeki ifadenin ekranda görülmesini sağlar.

FOR-WHILE Döngüleri

sirayakoy adlı programda geçen önemli komutlardan biri de for komutudur. for- end arasındaki komutların tekrar tekrar yapılmasını istediğimizde bu yapıya ihtiyaç vardır. Söz konusu program c boş vektörüne a ve b den birer eleman alarak ilk adımı tamamlamaktadır. Böylece elde edilen c vektörüne ikinci adımda a ve b den birer eleman daha katılmakta ve işlem a ve b nin eleman sayıları kadar tekrarlanmaktadır.

Şimdi başka bir örnek ele alalım. Programla ilgili detaylar başlangıçta verilmiştir. İç içe kullanılan for döngülerine dikkat ediniz. Burada kullanılan bir başka yeni komut pause komutudur.

```
% exfor_m.m
% Bu program bir a matrisinin işaretini değiştirip
% c matrisini elde etmektedir. Ayrıca a matrisinin
% transpozunu bulmak için de for döngüsünü kullanmaktadır.
% Bir a matrisinin transpozu için a' komutu yeterlidir
a=round(10*rand(3));
for i=1:max(size(a))
  for j=1:max(size(a))
    b(i,j)=a(j,i);
    c(i,j)=-a(i,j);
  end
end
disp('ilk matris')
a
disp('devam etmek için bir tuşa bas')
pause;
disp('işareti değiştirilmiş matris')
c
disp('matrisin transpozu')
b
```

```
» exfor_m
ilk matris

a =
     4     9     4
     6     7     9
     8     2     9
```

```
devam etmek için bir tuşa bas
işareti değiştirilmiş matris

c =
    -4    -9    -4
    -6    -7    -9
    -8    -2    -9

matrisin transpozu

b =
     4     6     8
     9     7     2
     4     9     9
```

İç içe geçmiş for döngülerinde önce içteki döngü tamamlanır. Buna bir örnek verelim:

```
%exFORFOR
% Bu program iç içe for kullanımına
% bir örnektir.
h=[];
for m=1:3
   for n=1:2
     h=[h;[m,n]];
   end
end
h
```

» exforfor

h =

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

`for` döngüsüne benzer diğer bir deyim olan `while` programlamada geniş bir kullanım alanına sahiptir. Şimdi bir kaç örnek verelim. Komuttan sonraki önerme doğru olduğu müddetçe (1) işlemler devam edecektir.

```
%bravo.m
% Ekrandan girilen 1 ile 10 arasındaki bir sayının bil-
% gisayar tarafından rastgele seçilen bir sayıya eşit
% olması halinde ekranda 'BRAVO' şeklinde mesaj verecek
% ve programı durduracak aksi halde ekrandan sayı giril-
% mesine devam edecek program

while round(10*rand(1))~=input('sayıyı giriniz    ')
end
'bravo'
```

Lineer olmayan denklem takımlarının çözümünde kullanılan yöntemlerden biri Newton_Raphson yöntemidir . while komutu bu yöntemin programlanmasında etkin bir şekilde kullanılabilir.

Matlabla programlamanın bir diğer yolu function dosyalarının kullanımıdır. Function dosyaları programlar içinde tekrarlı olarak kullanılabilir. Bir function dosyası çalıştırıldıktan

sonra program içindeki hiç bir değişken bellekte kalmaz. Oysa şimdiye kadar ele aldığımız programlarda değişkenler bellekte tutulurlar.

Şimdi function dosyası olarak yazılmış bir program inceleyelim:

ln(1+x) fonksiyonunun Maclaurin serisi

$\ln(1+x)=\Sigma(-1)^{k+1}x^k/k \;,\quad k=1-\infty$

```
function y=macl(x)
% Bu program ln(1+x) fonksiyonunun Maclauren serisidir.
% ln(1+x)=((-1)^k+1)x^k/k  ,    k=1- ¥
lntop=0;k=1;
while abs((x^k)/k)>=eps
   lntop=lntop+((-1)^(k+1))*((x^k)/k)
   k=k+1
end
disp('lntop')
disp(lntop)
disp('k=iterasyon sayısı')
disp(k)
```

Programı çalıştırırken x=0 civarında kalmaya dikkat edilmelidir, aksi takdirde program sonuç vermeyecektir, (niçin?).

X=0.5 için sonuç şöyledir:

```
» macl(0.5)
lntop
    0.4055
k=iterasyon sayısı
    47
```

Bu sonucu log(1.5) yazarak kontrol ediniz.