

## **BÖLÜM 1 : MATLAB KULLANIMI ve MATRİS İŞLEMLERİ**

“Matlab” programı ( MATrix LABoratory 'nin ilk üç harfleri alınarak isimlendirilmiştir.) mühendislik uygulamalarının, hesaplamalarının ve simülasyonlarının çoğunun gerçekleştirildiği matris ve matematik tabanlı kompleks bir programdır. Her türlü grafiksel sonuçlar istenilen tarzda alınabildiği için kullanım alanı çok geniştir. Ayrıca MATLAB versiyonlarından en az 6.0 ve üzeri olanlarının kullanılması güncellik açısından daha yararlı olacaktır.

Bu bölümde programı kullanmaya başlamak için giriş komutları, matematiksel fonksiyonlar ve matris operatörleri anlatılacaktır. Ayrıca kılavuzun en son kısmında da en çok kullanılan matris komutları ve fonksiyonları tablo halinde verilmiştir.

help ‘fonksiyon ismi’

komutu yazıldığında yardım istenilen fonksiyon hakkında detaylı bilgiye ulaşılabilmektedir.

help help

yazıldığında ise on-line olarak yardım kılavuzunun nasıl kullanılacağı hakkında bilgilere ulaşılabilmektedir.

### **-- Matris Operatörleri :**

Aşağıda verilen simgeler matris işlemlerinde kullanılmaktadır:

+	Toplama
-	Çıkarma
*	Çarpma
^	Kuvvet alma
‘	Konjüge transpozunu alma

### **-- Mantık ve İlişki Operatörleri :**

<	Küçük	&	Ve
<=	Küçük eşit		Veya
>	Büyük	~	Değil
~=	Eşit değil		

-- Başlangıç olarak komut satırına :

date

yazılırsa program tarafından geçerli olan tarih alınacaktır.Yani :

ans=  
30-Oct-2002

-- MATLAB bir işlemin sonucunu ans= .... şeklinde gösterir. ( ans = answer = cevap)

-- MATLAB programından çıkmak için ise exit veya quit yazmak yeterli olacaktır.

-- En son yazılan komutların hepsine üst ve alt yön tuşlarına dokunarak kolay bir şekilde ulaşılabilir.

-- En son tanımlanan herhangi bir 'x' değeri için yapılan işlemlerden sonra bu 'x' değeri komut satırına yazılıp enter tuşuna basılırsa daha önce neye karşılık olarak tanımlandığı ekrana yazılacaktır.

--  $n \times 1$  veya  $1 \times n$  boyutunda vektör tanımlamak için :

$x=[1 \ 2 \ 3 \ -4 \ -5]$  veya

$X=[1,2,3,-4,-5]$  yazılmalıdır.

Yukarıdaki iki yazım biçiminden okuma kolaylığı olması için ilk yazılan tip kullanılacaktır.

-- Tanımlanan bu satır vektörünü sütun vektörüne dönüştürmek için :

$y=x'$  yazılırsa ekranda görülen değer aşağıdaki gibi olacaktır:

y=  
1  
2  
3  
-4  
-5

-- Matris tanımlamak için aşağıdaki A matrisi verilmiş olsun :

$$A = \begin{bmatrix} 1.2 & 10 & 15 \\ 3 & 5.5 & 2 \\ 4 & 6.8 & 7 \end{bmatrix}$$

Bu matrisi MATLAB'e tanıtmak için şu şekilde yazılmalıdır :

$$A = [1.2 \ 10 \ 15 ; 3 \ 5.5 \ 2 ; 4 \ 6.8 \ 7]$$

Yani her satırın sonunun neresi olduğunu konulan noktalı virgül işareti temsil etmektedir.

-- Örnek olarak aşağıdaki B matrisini tanıtmak için :

$$B = \begin{bmatrix} 1 & e^{-0.02} \\ \sqrt{2} & 3 \end{bmatrix}$$

$$B = [1 \ \exp(-0.02); \text{sqrt}(2) \ 3] \text{ şeklinde yazılmalıdır.}$$

Ekranda ise şu şekilde gözükecektir:

$$B = \begin{bmatrix} 1.0000 & 0.9802 \\ 1.4142 & 3.0000 \end{bmatrix}$$

-- Apostrofi işareti (') matrisin konjüge transpoznesinin alınmasına yarar. Eğer matris reel bir matris ise basit olarak transpoze alım işlemi olarak da tanımlanabilir..

Yeni bir A matrisi tanımlayalım :

$$A = [1 \ 2 \ 3 ; 4 \ 5 \ 6 ; 7 \ 8 \ 9]$$

Ekranda görülecek matris şu şekilde olacaktır :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Bu matrisin transpozmesini almak için :

$C = A'$  yazılırsa ekranda görülecek transpoze değeri :

$$C = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

-- Kompleks sayıların girilmesi için ise  $i^2=-1$  denkleminin kökü  $i$  veya  $j$  olarak tanımlanır.

Örnek olarak  $1+j\sqrt{3}$  değerini tanıtmak için :

$$X = 1 + \text{sqrt}(3)*i \quad \text{veya}$$

$$X = 1 + \text{sqrt}(3)*j \quad \text{olarak yazılmalıdır.}$$

Bu kompleks sayı üstel formatta da yazılabilmektedir :

$$1+j\sqrt{3} = 2 \exp[(\Pi/3)*j]$$

Bu durumda komut satırına aşağıdaki ifade yazılacaktır :

$$x = 2 \exp[\text{pi}/3)*j]$$

$i$  ve  $j$  daha önceden değişken olarak kullanılmışsa tanımlama için  $ii$  ve  $jj$  kullanılacaktır.Yani :

$$ii = \text{sqrt}(-1)$$

$$jj = \text{sqrt}(-1)$$

Dolayısıyla aşağıdaki yazım da mümkün olmaktadır :

$$X = 1 + \text{sqrt}(3)*ii$$

$$X = 1 + \text{sqrt}(3)*jj$$

-- Kompleks matris tanımlamak için aşağıdaki X matrisi verilmiş olsun:

$$X = \begin{bmatrix} 1 & j \\ -j5 & 2 \end{bmatrix}$$

Komut satırına ise şu şekilde girilecektir :

$$X = [1 \ j ; -j5 \ 2]$$

Bu durumda ekranda görülecek değer :

$$X = \begin{bmatrix} 1.0000+0 & 0+1.0000i \\ 0-5.0000i & 2.0000+0 \end{bmatrix}$$

$Y = X'$  komutu yazılırsa :

$$Y = \begin{bmatrix} 1.0000+0 & 0+5.0000i \\ 0-1.0000i & 2.0000+0 \end{bmatrix}$$

iletişi ekranda okunacaktır.

Daha önce de belirtildiği gibi yukarıdaki işlem konjüge transpoze olarak algılanmaktadır. Eğer sadece transpoze alınacaksa (konjügesiz) komut şu şekilde yazılmalıdır :

$$Y.' \quad \text{veya} \quad \text{conj}(Y')$$

Bu durumda ekranda gözükecek değerler :

$$\text{ans} = \begin{bmatrix} 1.0000+0 & 0-1.0000i \\ 0+5.0000i & 2.0000+0 \end{bmatrix}$$

-- Toplama ve çıkarma işlemlerinin yapılması için aşağıdaki M ve N matrisleri verilmiş olsun :

$$M = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix} \quad N = \begin{bmatrix} 1 & 0 \\ 2 & 3 \\ 0 & 4 \end{bmatrix}$$

Bu değerleri ekrana girmek için:

$$M = [2 \ 3 ; 4 \ 5 ; 6 \ 7]$$

$$M = \begin{bmatrix} 2 & 3 \\ 4 & 5 \\ 6 & 7 \end{bmatrix}$$

$$N = [1 \ 0 ; 2 \ 3 ; 0 \ 4]$$

$$N = \begin{bmatrix} 1 & 0 \\ 2 & 3 \\ 0 & 4 \end{bmatrix}$$

Toplama işlemi için:

$$C = M+N$$

$$C = \begin{bmatrix} 3 & 3 \\ 6 & 8 \\ 6 & 11 \end{bmatrix}$$

Eğer x vektörü şu aşağıdaki gibi verilirse :

$$X = \begin{bmatrix} 5 \\ 4 \\ 6 \end{bmatrix}$$

Bu vektörü şu şekilde tanıtmak gerekir :

$$X = [5 ; 4 ; 6]$$

Örneğin  $T = X-1$  gibi işlemi gerçekleştirmek için :

$$T = X-1 \\ T = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$$

-- Matris çarpımı daha önce de belirtildiği gibi \* çarpma operatörüyle yapılmaktadır. Aşağıdaki örnek incelenirse çarpmanın da tanım gereği çarpılan matrislerin boyutlarının uyuşması gerekmektedir. Aksi takdirde çarpma işlemi yapılmayacak ve hata mesajı verilecektir.

```
x = [1 ; 2 ; 3];    y = [4 ; 5 ; 6];    A = [1  1  2 ; 3 4  0 ; 1  2  5]
```

```
x'*y
```

```
ans =
    32
```

```
x*y'
```

```
ans =
     4     5     6
     8    10    12
    12    15    18
```

```
b = A*x
```

```
b =
     9
    11
    20
```

Bunların dışında matris bir skaler değerle de çarpılabilir :

```
5*A
```

```
ans =
     5     5    10
    15    20     0
     5    10    25
```

-- Matris üssü ( expm(A) )  $n \times n$  matrise uygulanır. Matematiksel tanımı ise şu şekildedir:

$$\text{expm}(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots$$

Eğer A kompleks bir matris ise abs(A) değeri de kompleks modül değerler üzerinden hesaplanacaktır. Yine matematiksel ifadesine bakacak olursak :

$$\text{abs}(A) = \sqrt{\text{real}(A)^2 + \text{imag}(A)^2}$$

`angle(A)` ise faz açılarını radyan cinsinden A kompleks matrisi için hesaplamaktadır. Burada tanım değerleri  $-\Pi$  ve  $+\Pi$  arasında kabul edilmektedir.

Sonuç olarak verilen bir K matrisi için aşağıdaki uygulama incelenebilir :

```
A = [2+2*i    1+3*i ; 4+5*i    6-i];
abs(A)
```

```
ans =
    2.8284    3.1623
    6.4031    6.0828
```

```
angle(A)
```

```
ans =
    0.7854    1.2490
    0.8961   -0.1651
```

-- Kompleks bir sayının modülü ve faz açısını bulmak için :

$$z = x + iy = re^{i\theta}$$

```
r = abs(z)
theta = angle(z)
z = r*exp(i*theta)
```

-- Bir vektörün elemanlarının teker teker karesinin alınması işlemi şu şekilde yapılmaktadır :

```
x = [1  2  3];
x.^2
```

```
ans=
    1    4    9
```

Eğer kompleks sayılar mevcut ise :

```
y = [2+5*i    3+4*i    1-i]
y.^2
```

```
ans =
   -21.0000+20.0000i   -7.0000+24.0000i    0-2.0000i
```



2x2 bir kare matris olursa yine aynı şekilde :

```
A = [1 2 ; 3 4];  
A.^2
```

```
ans =  
     1     4  
     9    16
```

-- Eleman elemana çarpma işlemi için çarpma operatörünün önüne bir nokta işareti (.) konmaktadır :

```
x = [1 2 3], y = [4 5 6]  
z = [x.y]  
z = [4 10 18]
```

Bir örnek daha verilirse :

```
      1 2 3      4 5 6  
A =      1 9 8    B =      7 6 5
```

```
C = A.*B
```

```
C =  
     4 10 18  
     7 54 40
```

-- Bir matrisin tüm elemanlarının tek tek karesini almak için :

```
A=[ 1 2 3 4 ]  
  
A.^2  
  
ans =  
  
     1     4     9    16
```

-- Eleman elemana bölme için ise :

```
U = x./y
```

```
U =
    0.2500  0.4000  0.4000
    0.1429  1.5000  1.6000
```

-- Bir matrisin tersini bulmak için `inv(A)` komutu kullanılır :

```
A = [0 1 0;0 0 1;-6 -11 -6];
inv(A)

ans =
   -1.8333   -1.0000   -0.1667
    1.0000    0.0000    0.0000
    0.0000    1.0000    0.0000
```

-- Çeşitli komutlar ve durumlar tek bir sırada virgül (,) veya noktalı virgül ile (;) ayrılarak yazılabilir.

-- Çıkış formatını istediğimiz uzunlukta elde edebiliriz. Eğer matris elemanları tamsayı ise bu durum sonuçta bir değişiklik yapmayacaktır. Bunun için aşağıdaki komutları kullanmak gerekmektedir :

```
format short
format long
```

```
x = [1/3    0.00002];
x

x =
    0.3333    0.0000

format short; x

x =
    0.3333    0.0000

format long; x

x =
0.3333333333333333    0.000020000000000000
```

-- 1'den 5'e kadar sayıları 0.5'lik aralıklarla yazdırmak istersek iki nokta'yı (:) kullanmak yeterli olacaktır :

```
t =  
    1    2    3    4    5
```

```
t = 1:0.5:3
```

```
t =  
    1.0000    1.5000    2.0000    2.5000    3.0000
```

Düzgün azalan biçimde yazdırırsak :

```
t = 5:-1:2
```

```
t = 5  4 3 2
```

-- Bir matrisin i. satırını veya j. sütununu görüntülemek için aşağıda tanımlanan A matrisini komutlarıyla inceleyelim :

Aşağıdaki A matrisinin 2. satırı görüntülemek için : A(i , :)

```
A = [0  1  0;0  0  1;-6 -11 -6]  
A(2 , :)
```

```
ans =  
    0    0    1
```

A matrisinin 3. sütununu görüntülemek için :

```
A(:, 3)
```

```
ans =  
    0  
    1  
   -6
```

-- Bir matrisin (i,j) ninci elemanını bulmak için :

```
k = A(3,3)
```

```
k = -6
```

-- Bir matrisin boyutlarını öğrenmek için size(A) komutu, rankını bulmak için rank(A) kullanılır.

```
A=
    2    3    2
    5    4    1
    2    6    8
```

```
size(A)
```

```
ans =
     3     3
```

```
rank(A)
```

```
ans =
     3
```

-- Bir matrisin determinantını bulmak için det(A) komutu kullanılır.

```
A=
    2    3    2
    5    4    1
    2    6    8
```

```
det(A)
```

```
ans =
   -18
```

-- Bir matrisin normunu bulmak için ise norm(x) yazmak gerekmektedir. Matematiksel norm ifadesini verecek olursak :

$$\text{norm}(x) = \sum(\text{abs}(x).^2)^{0.5}$$

```
x = [2  3  6]
norm(x)
```

```
ans =
     7
```

-- Bir matrisin özdeğerlerini bulmak için eig(A) komutu kullanılır :

```
A = [0 1 ; -1 0]
eig(A)
```

```
ans =
    0+1.0000i
    0-1.0000i
```

-- Öz vektörleri bulmak da tek satırlık bir işlem gerektirmektedir. Aslında özvektörleri bulmak için verilen  $[X,D] = \text{eig}(A)$  komutu aynı zamanda öz değerleri de bulduğu için her iki bilgiye aynı anda ulaşma imkanı olmaktadır :

```
A = [0 1 0 ; 0 0 1 ; -6 -11 -6]
[X,D] = eig(A)

X =
   -0.5774    0.2182   -0.1048
    0.5774   -0.4364    0.3145
   -0.5774    0.8729   -0.9435

D =
   -1.0000    0    0
    0   -2.0000    0
    0    0   -3.0000
```

Burada X sonuç matrisinin her bir sütunu verilen A matrisinin bir öz değerini göstermektedir.

D sonuç matrisinin diyagonalindeki (köşegenindeki) elemanların her biri de verilen A matrisinin özdeğerlerini göstermektedir.

Verilen eş boyutlu farklı iki A ve B gibi matrisin genelleştirilmiş öz değerlerini ve öz vektörlerini bulmak için ise  $[X,D] = \text{eig}(A,B)$  komutu yazılmalıdır.

-- Karakteristik denklemi bulmak için poly(A) komutu kullanılır.

```
A = [0 1 0 ; 0 0 1 ; -6 -11 -6]

p = poly(A)

p =
    1.0000    6.0000   11.0000    6.0000
```

Burada görülen sonuç katsayıları karakteristik denklemin katsayılarıdır. Yani :

$$s^3 + 6s^2 + 11s + 6 = 0$$

-- Bir polinomun köklerini bulmak için roots(a) komutu yazılmalıdır. Yukarıdaki karakteristik denklemin köklerini bulmak istersek :

$$r = \text{roots}(p)$$

$$r = \begin{matrix} -3.0000 \\ -2.0000 \\ -1.0000 \end{matrix}$$

-- Polinomların çarpımı için conv(a,b) komutu kullanılır.

$$\begin{aligned} a(s) &= s^2 - 20.6 \\ b(s) &= s^2 + 19.6s + 151.2 \end{aligned}$$

a(s) ve b(s) polinomlarını çarpmak için :

$$\begin{aligned} a &= [1 \ 0 \ -0.26]; \quad b = [1 \ 1.96 \ 151.2] \\ c &= \text{conv}(a,b) \end{aligned}$$

$$c = \begin{matrix} 1.0e+003 \\ 0.0010 & 0.0196 & 0.1306 & -0.4038 & -3.1147 \end{matrix}$$

Dolayısıyla çarpım sonucu şu şekilde yazılabilir :

$$c(s) = s^4 + 19.6s^3 + 130.6s^2 - 403.8s - 3114.7$$

-- Bir polinomda herhangi bir tamsayı değerini hesaplatmak için polyval(c) komutu kullanılır :

$$p(s) = 3s^2 + 2s + 1$$

$$\begin{aligned} p &= [3 \ 2 \ 1]; \\ \text{polyval}(p,5) \end{aligned}$$

$$\begin{aligned} \text{ans} &= \\ 86 \end{aligned}$$

-- 1 ve 0 sayılarının istenilen matrisel boyutta çabuk olarak üretilebilmesi için ones(m,n) ve zeros(m,n) komutları kullanılabilir :

```
ones(2,2)
```

```
ans =
```

```
1    1
1    1
```

```
zeros(3,3)
```

```
ans =
```

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

-- Birim matris de eye(n) komutuyla istenilen boyutta oluşturulabilir :

```
eye(5)
```

```
ans =
```

```
1    0    0    0    0
0    1    0    0    0
0    0    1    0    0
0    0    0    1    0
0    0    0    0    1
```

-- Bir matrisin köşegenindeki elemanları listelemek için diag(A) komutu kullanılır :

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

```
diag(A)
```

```
ans =
```

```
1
5
9
```

Köşegenin elemanları haricindeki matris bileşenleri 0 olarak göstermek istersek :

```
diag(diag(A))
```

```
ans =
```

```
1    0    0
0    5    0
0    0    9
```

Köşegen matrisi oluşturmayla alakalı aşağıdaki diğer örnekler de incelenebilir :

```
diag(1:5)
```

```
ans =
```

1	0	0	0	0
0	2	0	0	0
0	0	3	0	0
0	0	0	4	0
0	0	0	0	5

```
diag(0:4)
```

```
ans =
```

0	0	0	0	0
0	1	0	0	0
0	0	2	0	0
0	0	0	3	0
0	0	0	0	4

```
[diag(1:5) - diag(0:4)]
```

```
ans =
```

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

-- Bir matrisi rastgele olarak oluşturmak için rand(n) komutu kullanılır.0 ile 1 arasındaki sayıları alır.

```
rand(4)
```

```
ans =
```

0.3654	0.6739	0.3603	0.0493
0.1400	0.9994	0.5485	0.5711
0.5668	0.9616	0.2618	0.7009
0.8230	0.0589	0.5973	0.9623



## ALIŞTIRMALAR

**1-** Sin(x) 'i ilk 10 terim kullanarak bulan bir Matlab programı yazınız. X açısı değerinin derece olarak kullanıcıdan alıp sin(x)'i bulunuz ?

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + x^9/9! - \dots$$

**2-** Yerden  $V_0$  hızıyla ve  $\Theta$  açısıyla fırlatılan bir topun  $t = 0,1,2,\dots,10$  saniye boyunca hareket bilgilerini veren Şema T,  $V_x$ ,  $V_y$  bilgilerini her saniyede görüntüleyebilmelidir.

Not1: Gerekli formüller aşağıda sıralanmıştır. ( $g = 10 \text{ m/s}^2$ , Yer çekimi ivmesi)

$$V_x = V_0 \cos(\Theta) ; V_y = V_0 \sin(\Theta) - gt ; V = (V_x^2 + V_y^2)^{1/2}$$

Not2: Topun tüm hareketi boyunca yerden yeterince yüksekte olduğunu varsayınız.

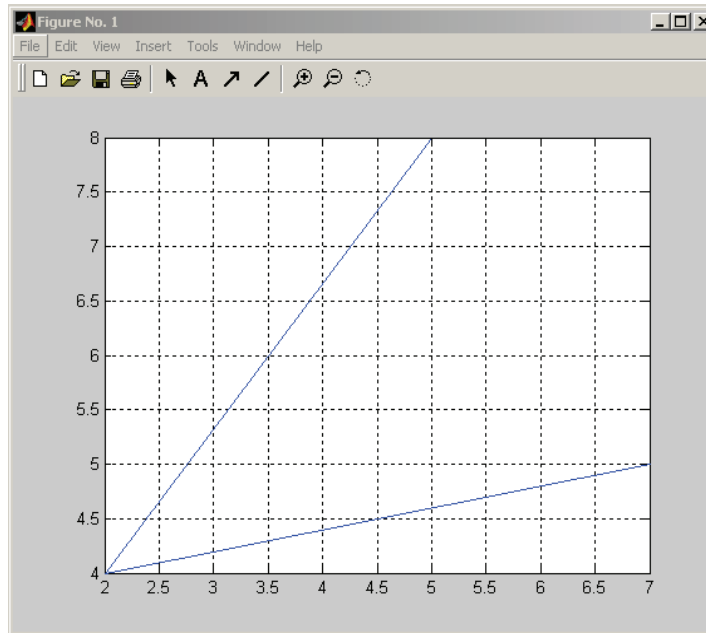
## **BÖLÜM 2 : GRAFİK ve EĞRİ ÇİZİMLERİ**

-- x ve y vektörleri aynı boyutta ise bu vektörleri ekrana çizdirmek için plot(x,y) komutu kullanılır.

```
A =[ 7  2  5];  
B =[ 5  4  8 ];  
plot(A,B);  
grid
```

Bu durumda grafik ekrana aşağıda gösterildiği gibi otomatik olarak çizilecektir :

Ayrıca plot(X,Y,'x') komutu çizilen eğriyi 'x' karakterini kullanarak çizmektedir.



-- Aşağıda grafik çizimiyle ilgili bazı özellikler sıralanmıştır :

**x=3:0.5:10**

Seçilen bir parametreye göre (burada x parametresi seçilmiş) çizdirilmesi planlanan eğrinin sınırları yukarıdaki gibi yazılır. 3 ve 10 değerleri çizdirilmek istenen aralığı, ortadaki 0.5 değeri artış miktarını göstermektedir.

**grid**

Grafik arka yüzünün ölçekli olarak gösterilmesini sağlar.

**title('...')**

Çizilen grafiğe başlık yazmak için kullanılır.

**xlabel('...')**

Çizilen grafiğin x-eksenine istenilen açıklamayı yazmak için kullanılır.

**ylabel('...')**

Çizilen grafiğin y-eksenine istenilen açıklamayı yazmak için kullanılır.

**text('X,Y,'text')**

Grafik ekranı üzerine istenilen koordinatlar dahilinde herhangi bir açıklama yazmak için kullanılır.

**. + \* o x**

İstenildiği takdirde çizilen eğrinin düz çizgi olarak değil de farklı karakterlerle çizdirilebilir. Bunlar için ise yukarıda gösterilen nokta, artı, yıldız, yuvarlak ve x karakterleri kullanılır. Bu karakterleri plot() komutu içerisinde '+' şeklinde yazmak yeterli olacaktır.

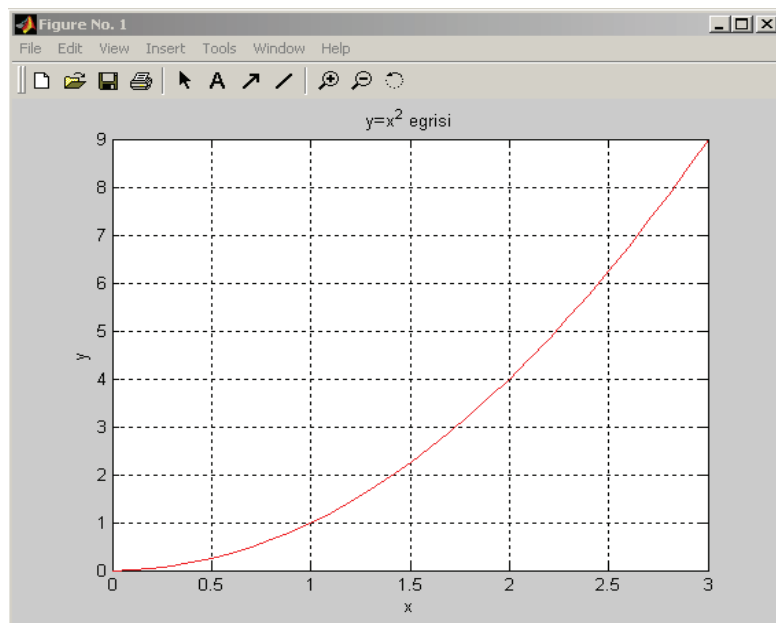
**r g b w i**

Çizilen eğrinin rengi de yukarıda gösterilen kısaltmalarla değiştirilebilir. Burada 'r' kırmızı renk (red), 'g' yeşil renk (green), 'b' mavi renk (blue), 'w' beyaz renk (white) ve 'i' ise (invisible) olarak kısaltılmıştır.

**Not :** Bu özellikler ve daha farklı görüntü özellikleri grafik ekranı üzerindeki "Insert" ve "Tools" menüleri aracılığıyla komut satırını kullanmadan da yapılabilmektedir.

-- Aşağıdaki örnekte ise  $y = x^2$  eğrisini 0 ve 3 aralığında çizdirelim :

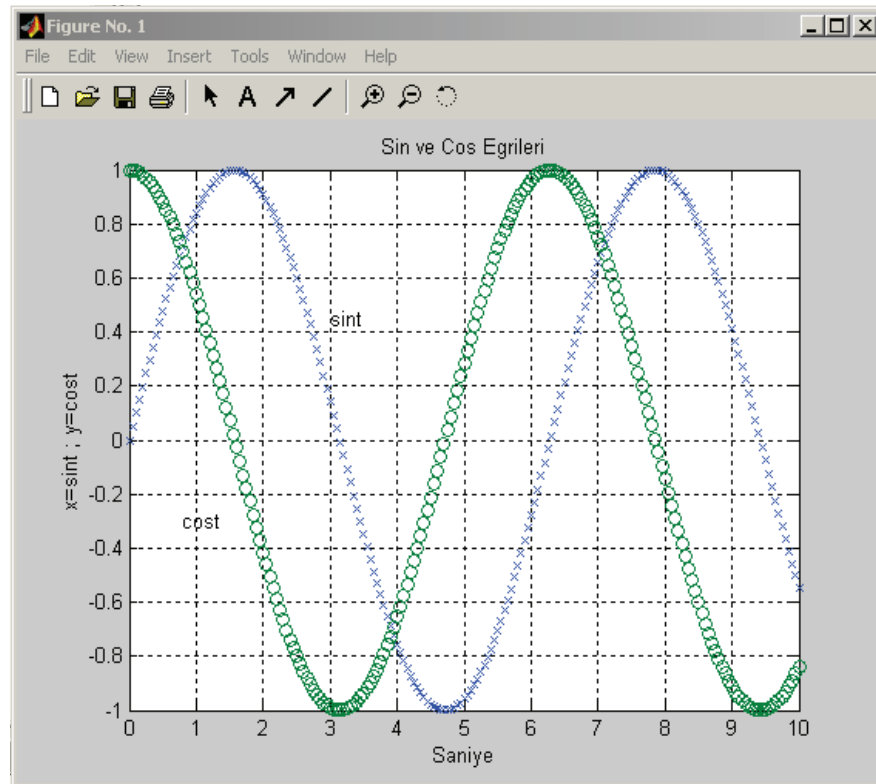
```
x = 0:0.1:3;  
y = x.^2;  
plot(x,y,'r');  
title('y=x2 eğrisi');  
xlabel('x');  
grid;  
ylabel('y')
```



-- Birden fazla eğriyi tek bir grafik ekranı üzerinde görmek için çizdirilmesi istenen eğriler aynı plot(...) komutu içinde yazılmalıdır.

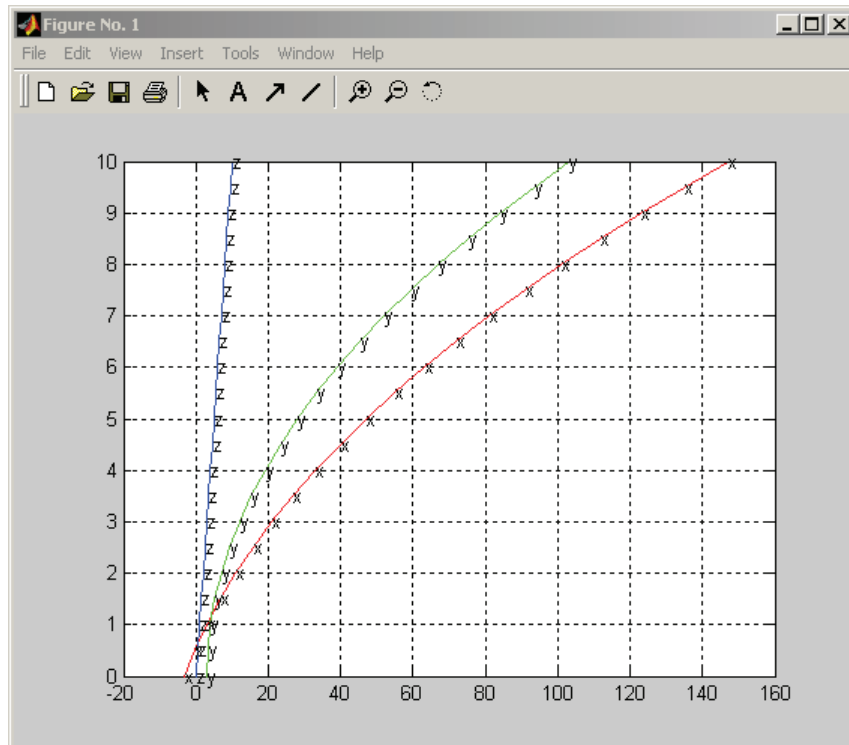
-- Birden fazla eğriyi üst üste çizme uygulaması olarak aşağıdaki örnekte  $\sin(x)$  ve  $\cos(x)$  eğrileri tek bir grafik ekranı üzerinde çizdirilmiştir :

```
t = 0:0.05:10;  
x = sin(t);  
y = cos(t);  
plot(t,x,'x',t,y,'o');  
grid;  
title('Sin ve Cos Eğrileri');  
xlabel('Saniye');  
ylabel('x=sint ; y=cost');  
text(3,0.45,'sint');  
text(0.8,-0.3,'cost')
```



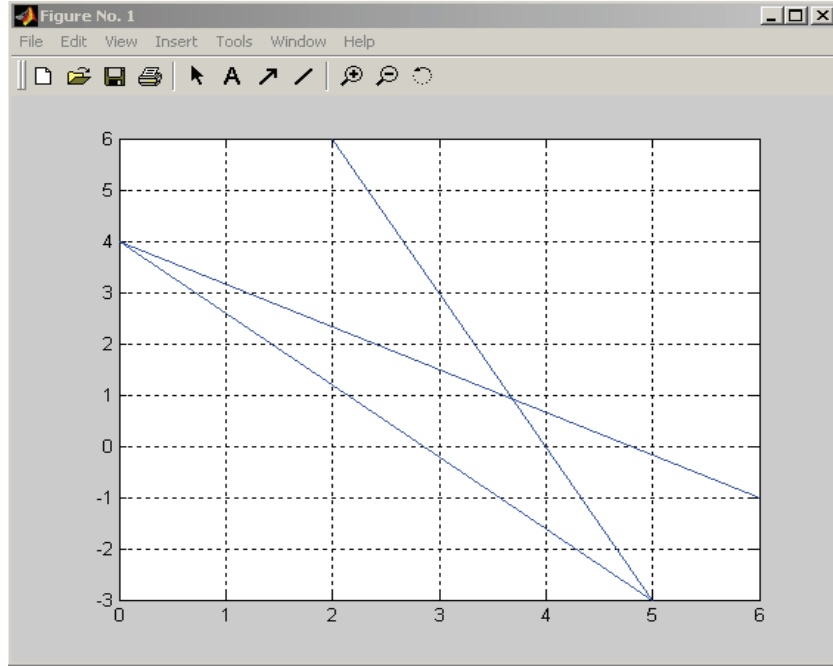
-- Aşağıdaki örnekte ise 3 farklı eğri çizdirilmiştir :

```
t=0:0.5:10;  
x=t^2+5*t-3;  
y=t.^2+3;  
z=t;  
plot(x,t,'r',y,t,'g',z,t,'b');  
grid;  
title('3 Farklı Grafigin Cizimi');  
xlabel('Giris Degerleri');  
ylabel('Cikis Degerleri');  
text(x,t,'x');  
text(y,t,'y');  
text(z,t,'z')
```



-- Kompleks vektörlerin çiziminde plot(z) ifadesi kullanılır. Çizim işleminde ise reel ve imajiner kısımlar ayrı ayrı ikili noktalar olarak kabul edilir :

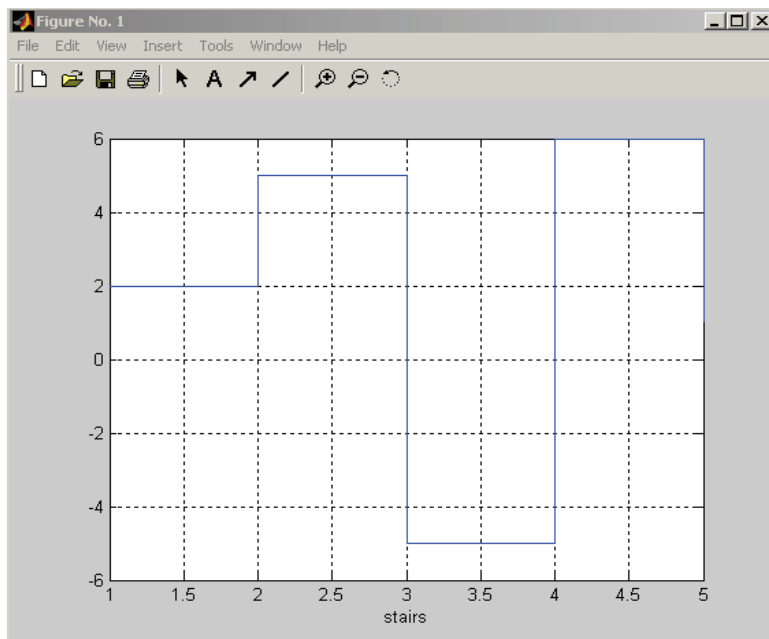
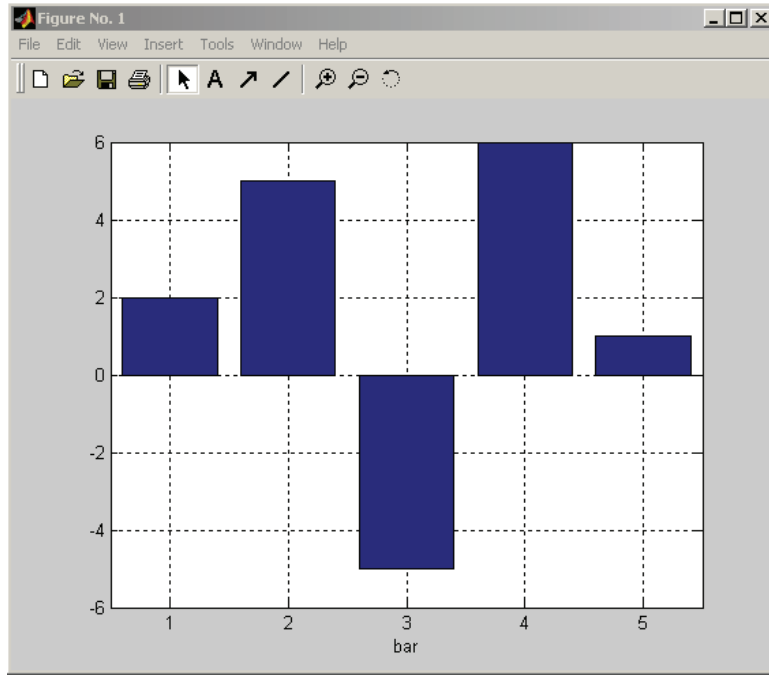
```
C=[2+6i 5-3i 4i 6-i]  
plot(C)  
grid
```



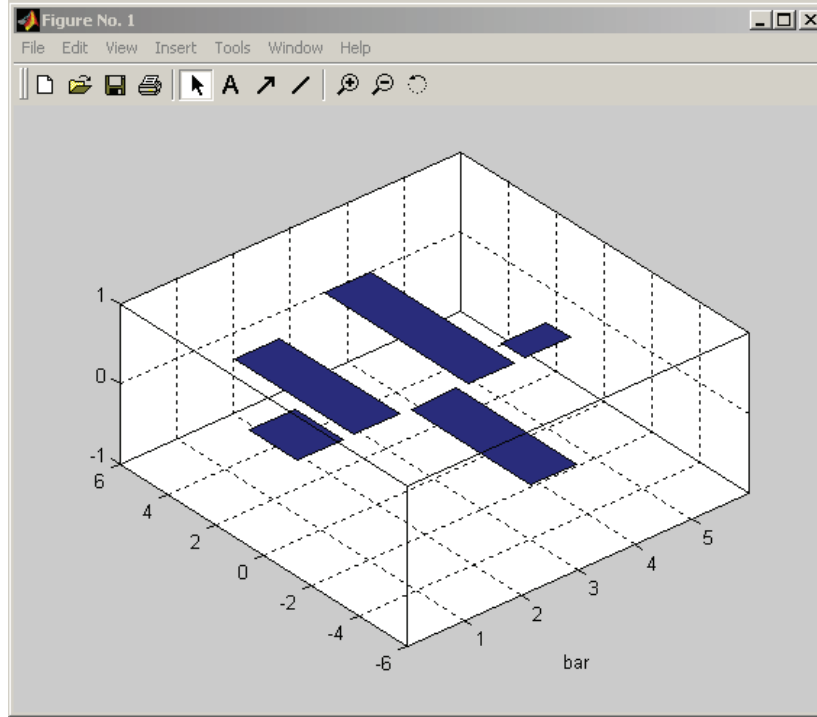
**Not :** loglog(X) komutu hem x eksenini hem de y eksenini logaritmik ölçeklendirmeyi kullanarak X'in grafiğini çizdirir

-- Bir A vektörünü “ bar grafiklerini ” kullanarak çizdirmek için bar(A) komutu kullanılır. “ Basamak ” fonksiyonu şeklinde çizilecek ise stairs(A) komutu kullanılır. Her iki çizime ait örnek grafikler aşağıda ayrı ayrı verilmiştir :

```
A = [ 2 5 -5 6 1 ]  
bar(A);  
grid;  
xlabel('bar');  
stairs(A);  
xlabel('stairs')
```



-- Ayrıca grafik ekranındaki menülerden yararlanarak çeşitli görüntü değişiklikleri yapılabilir. Örnek olarak “ Tools ” menüsünde “ Rotate-3D ” seçeneği kullanılarak mouse yardımıyla iki üstteki “ bar ” grafiğinin görüntüsü aşağıdaki gibi elde edilebilir.



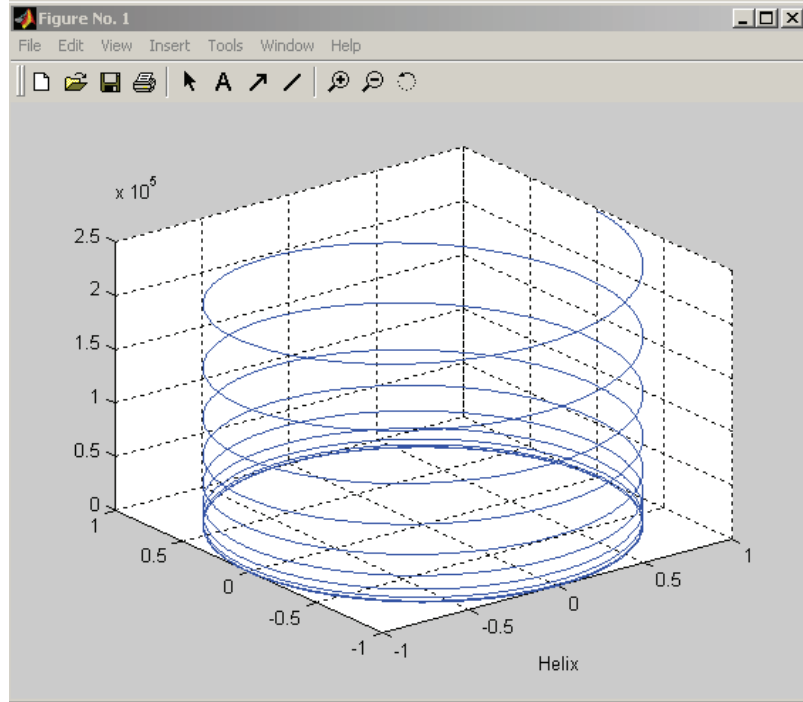
-- Grafik çiziminde grafik çizgi tipleri, işaretler ve renkler aşağıdaki tabloda sıralanmıştır :

Sembol	Renk(RGB)	Çizgi stili	Sembol	Nokta stili	
Y	sarı(110)	.	nokta	-	Çizgi
M	magenta(101)	O	yuvarlak	:	Noktalı
C	cyan(011)	X	çarpı işareti	-.	çizgili ve noktalı
R	kırmızı(100)	+	artı işareti	- -	kesik çizgili
G	yeşil(010)	*	yıldız		
B	mavi(001)	S	karekök		
W	beyaz(111)	D	baklava		



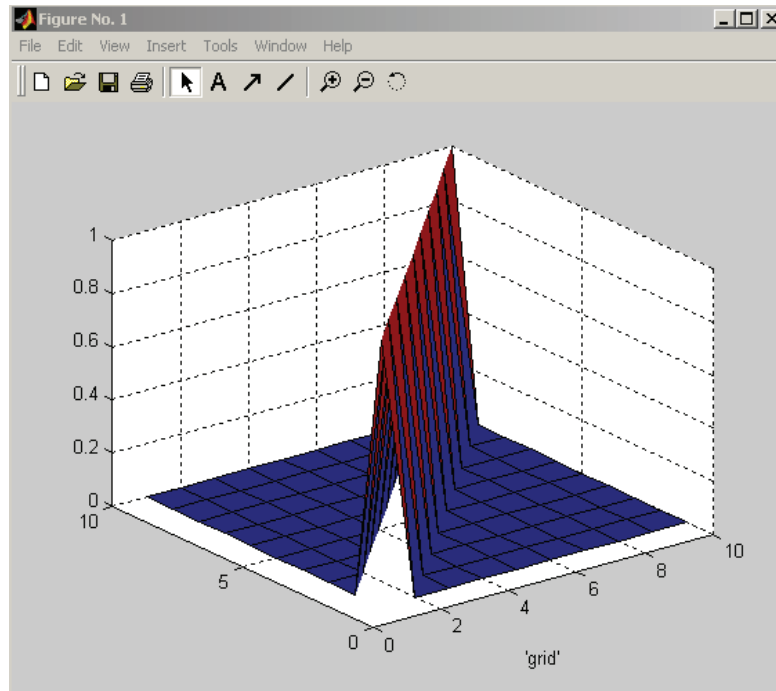
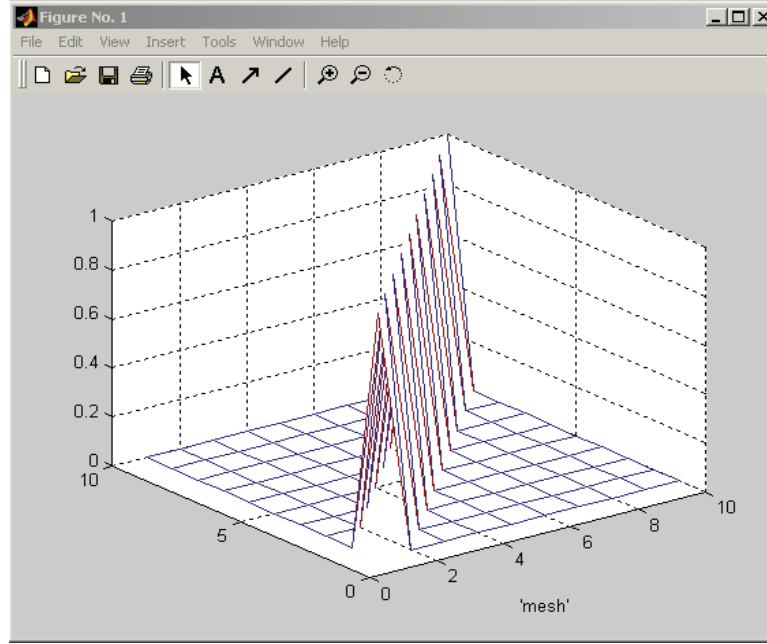
-- '3-D Line' (3 Boyutlu düz çizgi) çizimi için plot3(...) komutu kullanılır .Aşağıda heliks çizimi programı verilmiştir :

```
t=0.01:0.01:20*pi;  
x=cos(t);  
y=sin(t);  
z=t.^3;  
plot3(x,y,z);  
xlabel('Helix');  
grid
```



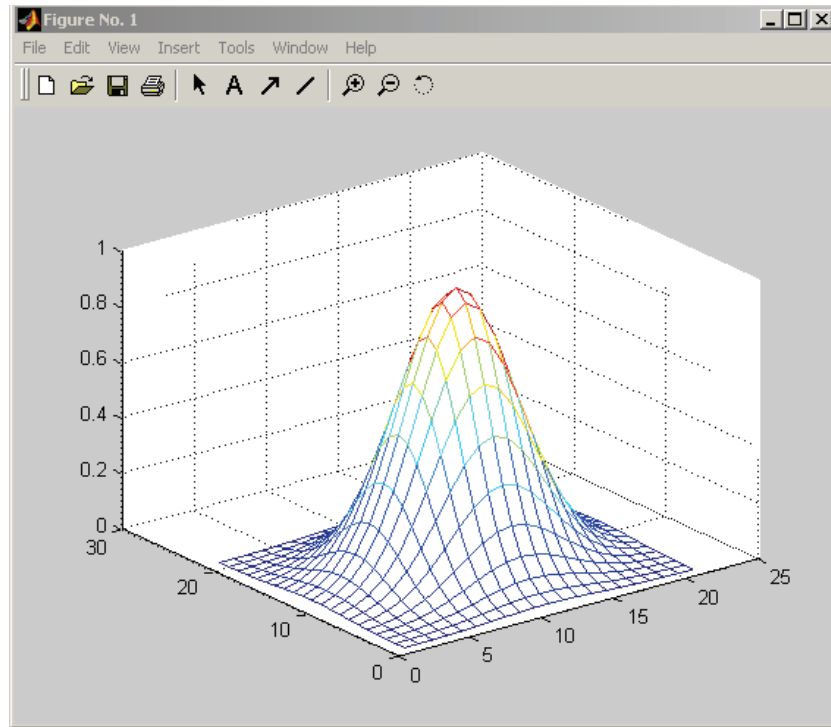
-- 3 boyutlu ağ ve yüzey çizimlerinde kullanılan komutlardan biri `mesh(...)` komutudur. Bu komut verilen girişi z bileşeni olarak algılar ve dikdörtgen x-y düzlemi üzerinde z eksenini boyunca çizim yapar. `surf(...)` komutu ise aynı işi yüzey olarak yapar. Aşağıdaki komut satırlarının çizim görüntüleri yine alt tarafında verilmiştir.

```
mesh(eye(10));  
grid  
  
surf(eye(10));  
grid
```



--  $z=\exp(-x^2-y^2)$  fonksiyon yüzeyini  $[-2,2] \times [-2,2]$  tanım aralığında 3 boyutlu olarak çizdirelim :

```
x=-2:0.2:2;  
y=x;  
[x,y]=meshgrid(x,y);  
z=exp(-x.^2-y.^2);  
mesh(z)
```



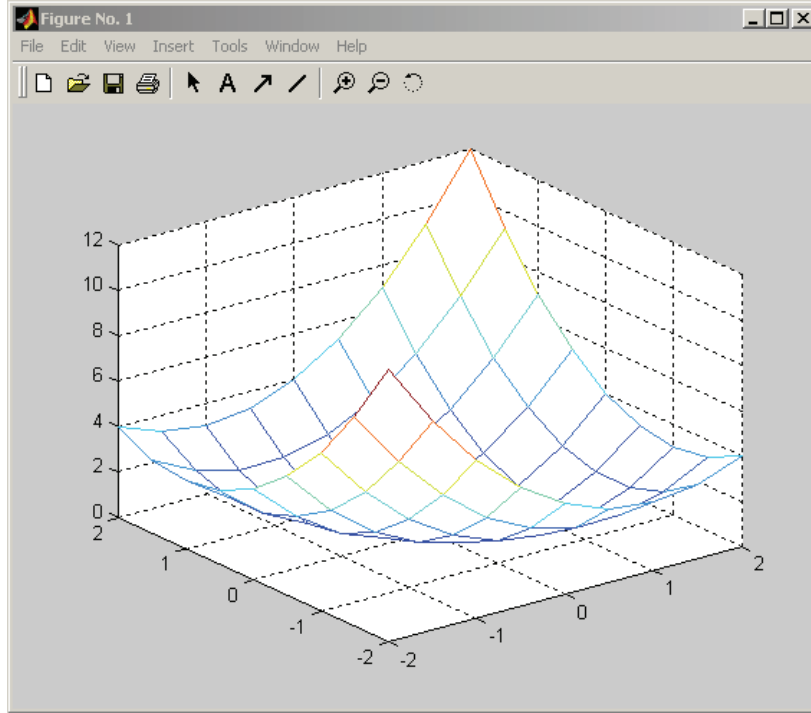
-- Ayrıca view komutu yardımıyla da küresel ve kartezyen koordinatlar ekranda görüntülenebilir.

```
view  
ans =
```

0.7934	-0.6088	0	-0.0923
0.3044	0.3967	0.8660	-0.7835
0.5272	0.6871	-0.5000	8.3031
0	0	0	1.0000

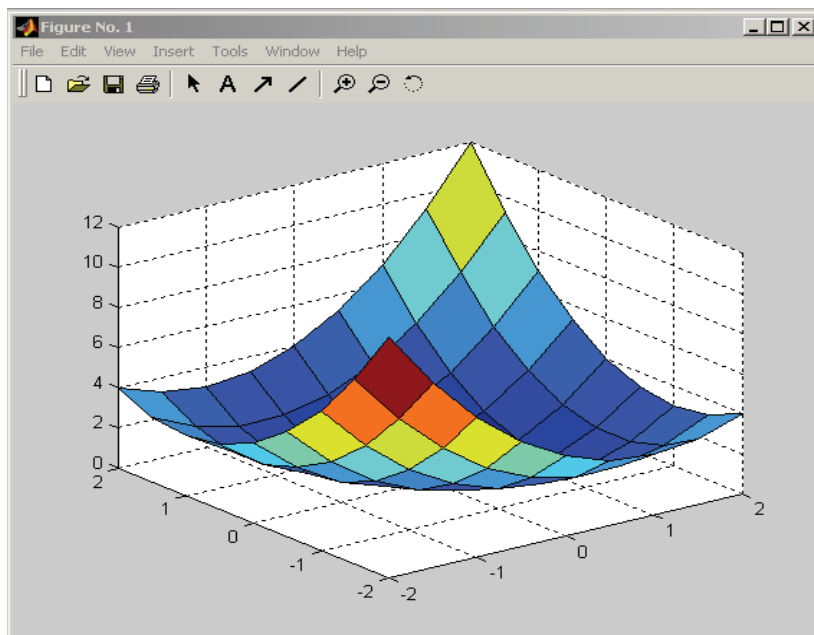
-- Örnek olarak  $z=x^2+y^2+xy$  yüzeyini  $-2 < x < 2$  ve  $-2 < y < 2$  aralığında çizdirelim :

```
[X,Y]=meshgrid(-2:0.5:2,-2:0.5:2);  
Z=X.^2+Y.^2+X.*Y;  
mesh(X,Y,Z)
```



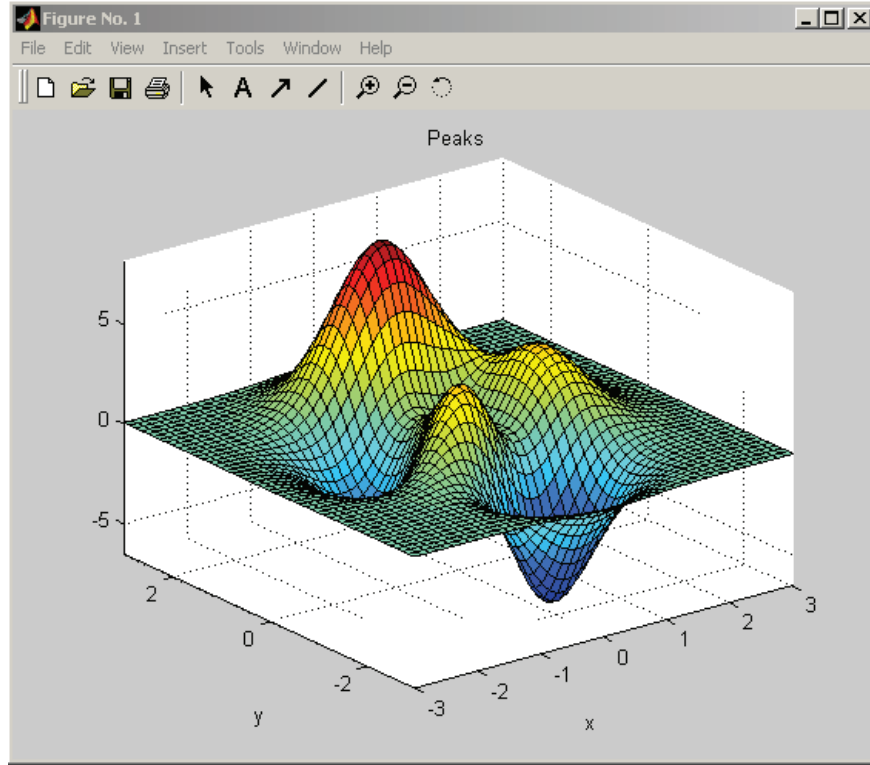
-- Yukarıdaki örnekte çizim fonksiyonu olarak  $\text{mesh}(X,Y,Z)$  yerine  $\text{surf}(X,Y,Z)$  çizim fonksiyonu kullanılırsa grafik yüzeyi aynı fakat her bir karesi farklı renklere boyanmış şekilde çizilecektir:

```
surf(X,Y,Z)
```



-- Herhangi bir yüzey grafiğinde tepe ve alt tepe (minimum ve maximum) değerlerini göstererek yapılan çizimlerde peaks(...) komutu kullanılır :

```
[X,Y]=meshgrid(-3:0.125:3);  
peaks(X,Y)
```



## ALIŞTIRMALAR

- 1-**  $x = t^3 - 2t + 9$  ,  $y = 6t^5 - t$  ,  $z = t^2 + 7$  eğrilerini tek bir grafik ekranında çizdiriniz.
- 2-**  $A = [ 5 \ 8 \ -2 \ 6 \ 4 \ 0 \ 7 ]$  giriş verilerini bar grafik ekranında çizdiriniz.
- 3-**  $z = 2x^2 + y$  yüzeyini, 0.2 artım değeriyle  $x = (-2, 2)$  ve  $y = (-2, 2)$  aralığını kullanarak çizdiriniz.
- 4-**  $z = e^{-2x} + 4x^3$  grafiğini (2,50) aralığında çizdiriniz.
- 5-**  $x = 9\sin(t)$  ,  $y = 2\tan(3t) + \cos(t)$  grafiklerini (0,10) aralığında 0.05 artımla çizdiriniz.

## **BÖLÜM 3 : MATLAB ile PROGRAMLAMA**

Bu bölümde MATLAB yazılımını başlangıçta kolaylıkla kullanabilmek için gerekli olan komut ve fonksiyonlar verilecektir.

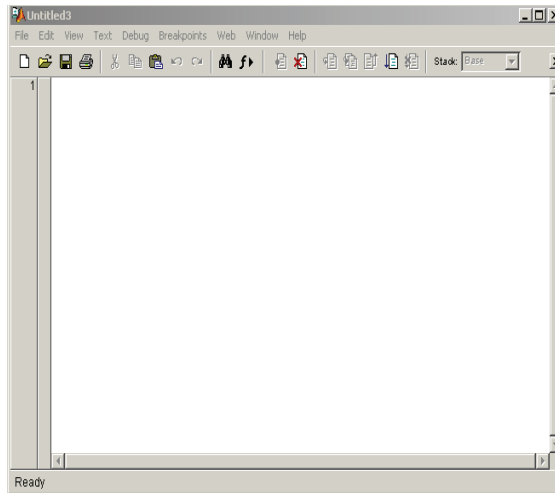
Şu ana kadar MATLAB ile yaptığımız uygulamalarda, belirli bir işlemi gerçekleştirmek üzere gerekli olan komut ya da fonksiyonları komut satırından, >> sembollerinden sonra tek tek girerek icra ediyorduk. Oysa, MATLAB komut ya da fonksiyonlarından gerçekleştirmek istediğimiz bir işle ilgili özel bir grubu, bir dosyaya kaydederek, bu dosya isminin çağırılmasıyla icra ettirebiliriz.

Bir deyimler ya da komutlar grubunu içeren bu tip bir dosyaya MATLAB’de M-dosyası (M-File) adı verilir. Bir komut grubu içerdiği için tanım itibariyle bir program dosyasıdır. Dolayısıyla programlama M-dosyaları oluşturularak yapılır.

-- Komut satırına “helpwin” komutu girilirse veya MATLAB’in Help menüsünden “Help Window” seçeneği seçilirse karşımıza gelecek yardım seçeneklerinden istenilen konu hakkında bilgi alınabilmektedir.

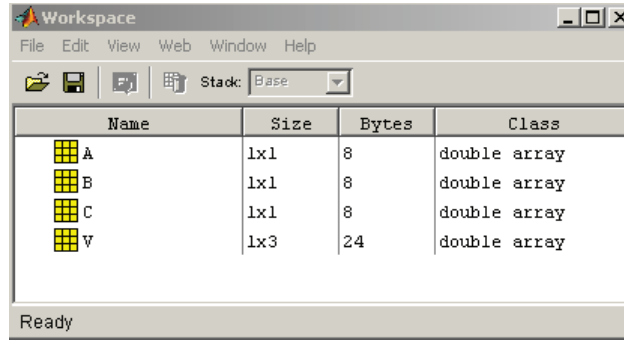
-- “**M-File**” **Oluşturma** : Programlama işlemi M-File (Program Dosyası) dosyalarında yapılır. Bu nedenle program yazarken en çok kullanılacak olan işlem M-File oluşturma işlemidir. Bunun için “File” menüsünden “New” seçilir ve daha sonra “M-File” seçeneği seçilir. Böylece yeni bir programlama ekranı elde edilir. Programlama işlemi bittikten sonra “M-File” dosyasının “File” menüsünden “Save As” seçeneği seçilir. Yazılan program buradaki “work” çalışma dosyasının içine istenilen isimle kaydedilir. Kaydedilen dosyaların uzantısı **.m** olarak kaydedilir. (Örnek: **sonuc.m**) İstenen dosyayı çalıştırmak için ise komut satırında dosya ismini yazıp “Enter” tuşuna basmak yeterlidir. Ayrıca “work” menüsüne girilip oradan da çalıştırılabilir.

İlk etapta örnek olarak oluşturulmuş olan boş bir M-File dosyası aşağıda gösterilmiştir:



## A) GENEL AMAÇLI KOMUTLAR

-- “ **workspace** ” **komutu** : Çalışma esnasında çalışma alanında (belleğin kullanıcıya ayrılan kısmı) mevcut değişkenler hakkındaki ayrıntılı bilgi aşağıda gösterildiği gibi bir pencere içinde verilir. Pencere içindeki değişkenlerin üzerine çift tıklayarak o değişkenle ilgili bilgilere ulaşıp değişiklikler yapılabilir.



-- “ **clear** ” **komutu** : Bellekte o anda mevcut bulunan değişkenleri bellekten siler.

-- “ **save** ” **komutu** : M-file dosyasının kaydedilmesi yukarıda da anlatıldığı gibi “File” menüsünden yapılır. MATLAB komut satırında ise “ save ” komutu kullanılırsa o esnada bellekte bulunan değişkenleri, istenilen dosya ismiyle ve uzantısı **.mat** olacak şekilde kaydedilir.(Örnek: **sayilar.mat**)

```
>> a=1
a =
    1
>> b=2
b =
    2
>> save sayilar
```

Yukarıda a ve b sayıları sayilar.mat dosyası olarak kaydedilmiştir.

-- “ **load** ” **komutu** : Diskte saklı bir dosya içindeki değişkenleri tekrar belleğe yükler.

```
>> load sayilar
>> who
Your variables are:
a b
```

Yukarıda “sayilar” dosyası “load” komutuyla belleğe tekrar yüklenmiş ve “who” komutuyla bu dosyanın değişkenleri görüntülenmiştir.



-- “**dir**” komutu : Bellekte kayıtlı olan dosyaları listeler.

(Not: Bir programı doğru olarak çalıştırmak için, icra ettirmeden önce “clear” komutuyla mevcut değişkenler silinebilir.)

```
>> dir
.          Oy.m          sayilar.mat          simple-report.sgml
..         diary         simple-report.html
```

-- “**type**” komutu : Bir .m uzantılı dosyanın içeriğini komut satırında görüntüler.

-- “**edit**” komutu : Bir M-dosyasının içeriğinde değişiklik yapma imkanı sağlar.

-- “**open**” komutu : Uzantısı ile belirtilen dosyayı açar.

-- Her yazılım türünde olduğu gibi MATLAB’de de, işletim sistemi kontrolüne geçmeden işletim sisteminin görevi olan bazı işlemleri gerçekleştirebilmek mümkündür. Bu tür işlemler aşağıdaki tabloda özetlenmiştir.

Komutun Adı	Komutun İşlevi
cd	Aktif dizini değiştirir.
dir	Aktif dizinin içindekileri listeler.
ls	Aktif dizinin içindekileri listeler.
delete	Belirtilen dosyayı siler.
type	Belirtilen dosyanın içeriğini listeler.

-- “**clc**” komutu : Komut satırını tamamen siler.

## **B) DEĞİŞKEN ATAMA**

“ C ” ve “ PASCAL ” gibi programlama dillerinde, programın ana gövdesinin oluşturulmasına başlamadan hemen önce, programdaki tüm değişkenlerin hangi tip değişken olduklarını belirtmek ve programın bilgisayar belleğinden uygun miktarda alanı bu değişkenler için tahsis etmesini sağlamak gerekir. MATLAB’de değişkenler, kendilerine ait bir isim ve onlara atanacak değerler yardımıyla oluşturulurlar. Önceden değişken tipini belirtmeksizin, değişkene verilen değere bağlı olarak MATLAB, uygun değişken tipini belirler ve bilgisayar belleğinden yeteri kadar yeri bu değişkene tahsis eder.

Birinci bölümde temel atamalarla ilgili bazı bilgiler (sabit, değişken, matris, dizi, vektör tanımlama...) verilmişti.

### **-- Mantık ve İlişki Operatörleri :**

==	Eşittir	&	and	Ve
~=	Eşit değil	&	and	Ve
<	Küçük	~	not	Değil
<=	Küçük eşit			
>	Büyük			
>=	Büyük eşit			

-- **global( ) komutu** : Farklı M-dosyaları için aynı değişken tanımlanacaksa o değişken global(x) olarak tanımlanabilir.

-- **disp('') komutu** : İstenen açıklamayı görüntüler.

```
>> disp('Programlamaya Giris')
Programlamaya Giris
```

-- **input('') komutu** : Kullanıcıdan klavye aracılığıyla programcı tarafından girilmesi istenen değişken istenir ve ilgili değişkene atanır.

```
>> Yas=input('Yasinizi giriniz :')
Yasinizi giriniz :23
Yas =
    23
```

-- **“ fprintf ” komutu** : Bir açıklama ifadesiyle birlikte bir veya birden fazla değer görüntülenebilmesini sağlar.

```
>> a=231565465;
>> fprintf('Hesap = %d ',a)
Hesap = 231565465.000000
```

**Not :** “ fprintf ” fonksiyonu, kompleks sayıların sadece reel kısmını gösterir. Bu nedenle kompleks sayı uygulamalarında “ disp ” fonksiyonunu kullanılmalıdır.

“ fprintf ” fonksiyonunda kullanılan çeşitli “ biçim tipleri ” aşağıda gösterilmiştir :

%d : Virgüllü sayıları 10’un kuvvetleri şeklinde gösterir.

%f : Kayan noktalı şekilde gösterir, aksi belirtilmedikçe virgülden sonra 6 basamak gösterir.

%e : Sayıyı üstel şekilde gösterir.

-- **linspace ve logspace komutları :** İlk değeri ve son değeri belirtilen bir diziyi lineer veya logaritmik olarak belirtilen sayıdaki elemanı kullanarak yapılandırır. Belirtilen aralığı otomatik olarak verilen eleman sayısına göre böler ve her böldüğü sayıyı görüntüler.

fonksiyon(ilk\_değer , son\_değer , eleman\_sayısı)

```
>> B=linspace(0,10,6)
```

```
B =
```

```
0    2    4    6    8   10
```

-- **Başlangıç, son değer ve artış miktarı belli dizilerin atanması :**

```
>> dizi=10:5:30
```

```
dizi =
```

```
10    15    20    25    30
```

--**Hazır fonksiyon özelliklerini kullanarak oluşturulan diziler için kullanılan komutlar :**

zeros(n,m) : nxm boyutunda 0’lardan oluşan matris üretir.

ones(n,m) : nxm boyutunda 1’lerden oluşan matris üretir.

eye(n,m) : nxm boyutunda birim matris üretir.

length(x) : “x” dizisinin satır sayısını verir.

size(x) : “x” matrisinin boyutlarını (satır ve sütun) verir.

format short : İşlem sonuçlarını virgülden sonra 4 basamaklı olarak gösterir.

format long : İşlem sonuçlarını virgülden sonra 14 basamaklı olarak gösterir.

**Not:** MATLAB’in yapısında önceden tanımlanmış bazı özel sabit veya açıklama değerler :

pi (Pi sayısı) ; i,j (Kompleks i sayısı) ; eps (Epsilon:İki sayı arasındaki en küçük fark) ; Inf (Sayı/Sıfır belirsizliği ve diğer belirsizlikler karşısında üretilen tanımsızlık cevabı : Infinite)

-- **Vektör ve matrislerin tanımlanması :**

```
>> A=[ 2 3 5 9 -2 ]
A =
     2     3     5     9    -2
```

3x3 boyutunda bir matrisi tanımlamak için :

```
>> C=[3 6 9 ; 8 2 4 ; 0 8 3 ]
C =
     3     6     9
     8     2     4
     0     8     3
```

-- “ **who** ” **komutu** : Çalışma alanındaki o esnada mevcut olan değişkenlerin isimlerini listeler. Ayrıca “ whos ” komutu değişkenler hakkında daha ayrıntılı bilgi verir.

```
>> who
Your variables are:
A B C V
```

-- **length( )** **komutu** : Girilen bir vektörün uzunluğunu (eleman sayısını) görüntüler.

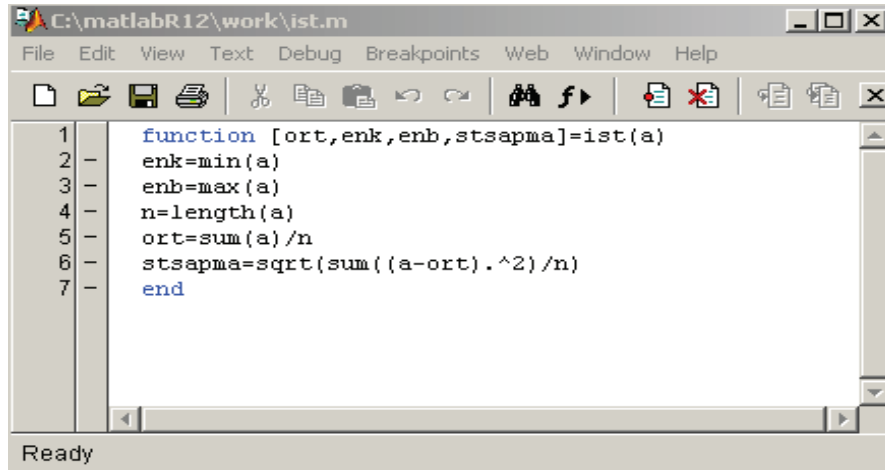
```
>> A=[ 5 8 7 2 5 9 6 ]
A =
     5     8     7     2     5     9     6
>> length(A)
ans =
     7
```

**C) FONKSİYON OLUŞTURMA VE DİĞER KOMUTLAR:**

-- **sum( ), min( ), max( ), mean( )** **komutları** : Yandaki komutlar sırasıyla bir vektörün elemanlarının toplamını, en küçük elemanını, en büyük elemanını ve ortalamasını bulur.

-- “ **function** ” **komutu** : Fonksiyon tanımlamak için kullanılır. Bu özellik aşağıda örnek üzerinde açıklanmıştır :

Örnek olarak kullanıcı tarafından girilen n adet rakamın (Bu rakamların MATLAB’de vektör formunda girilmesi gerekmektedir) ortalamasını, en küçük elemanını, en büyük elemanını ve standart sapmasını bulacak bir fonksiyonu ist(a) adıyla oluşturalım :

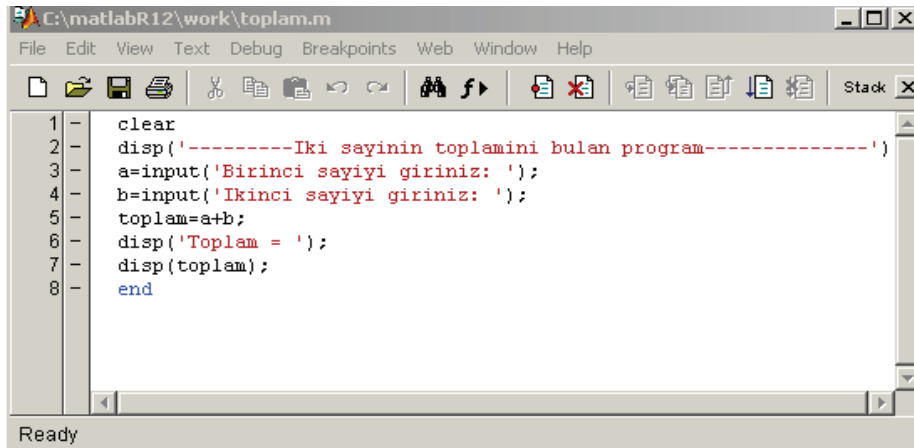


**Not:** Herhangi bir .m dosyası penceresinde “ f ” butonuna mouse ile bir defa dokunulduğunda (yukarıdaki şekilde de görülmektedir) “work” dizini altında kayıtlı olan bütün fonksiyonlar listelenir.

Şimdi de programı komut satırından icra ettirelim :

```
>> A=[5 3 6 9 73 6 5];
>> ist(A)
enk =
     3
enb =
    73
n =
     7
ort =
  15.2857
stsapma =
  23.6203
ans =
  15.2857
```

-- “ **type** ” **komutu** : Bir .m uzantılı dosyanın içeriğini komut satırında görüntüler. Örnek olarak 2 sayının toplamını yapan “toplam” isimli basit bir program yazalım. Aşağıda M-File görüntüsünde gösterilmiştir :

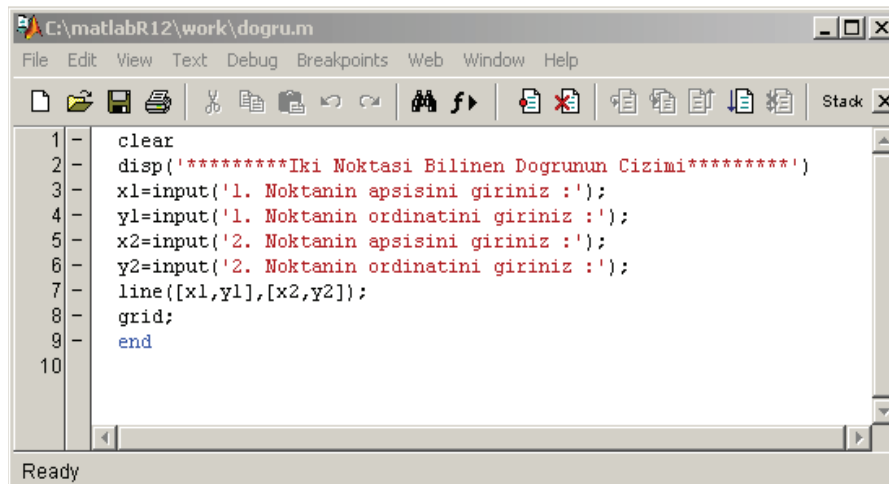


```
1 clear
2 disp('-----İki sayının toplamını bulan program-----')
3 a=input('Birinci sayıyı giriniz: ');
4 b=input('İkinci sayıyı giriniz: ');
5 toplam=a+b;
6 disp('Toplam = ');
7 disp(toplam);
8 end
```

MATLAB komut satırına “type toplam” yazılırsa :

```
>> type toplam
clear
disp('-----İki sayının toplamını bulan program-----')
a=input('Birinci sayıyı giriniz: ');
b=input('İkinci sayıyı giriniz: ');
toplam=a+b;
disp('Toplam = ');
disp(toplam);
end
>>
```

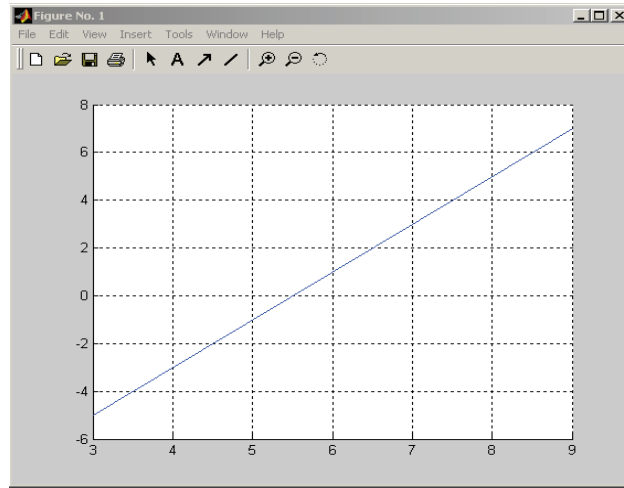
**Örnek Program:** “ İki noktası bilinen doğrunun çizimi ” programı aşağıda gösterildiği gibi dogru.m dosyası adı altında yazılmıştır. Çizilmesi istenen doğrunun iki noktası kullanıcıdan istenmekte ve grafik otomatik olarak çizdirilmektedir.



```
1 clear
2 disp('*****İki Noktası Bilinen Doğrunun Çizimi*****')
3 x1=input('1. Noktanın apsisini giriniz: ');
4 y1=input('1. Noktanın ordinatını giriniz: ');
5 x2=input('2. Noktanın apsisini giriniz: ');
6 y2=input('2. Noktanın ordinatını giriniz: ');
7 line([x1,y1],[x2,y2]);
8 grid;
9 end
10
```

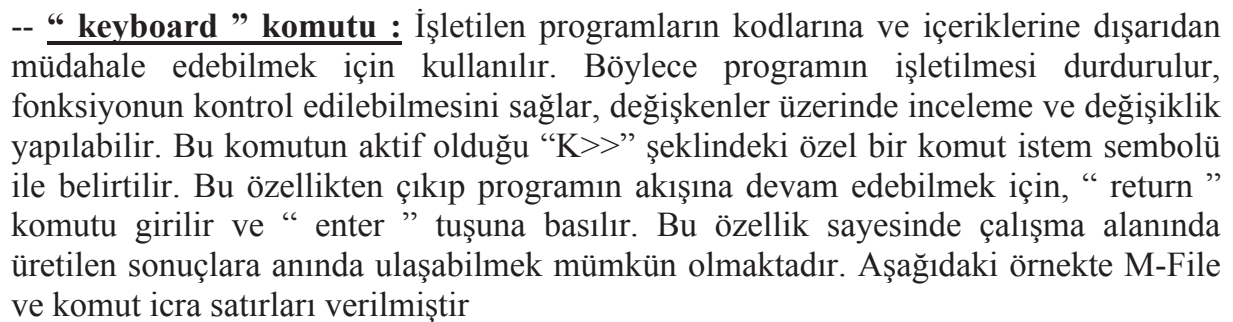
Bu program komut satırında çalıştırılırsa aşağıda gösterildiği gibi icra edilir ve grafik de otomatik olarak yine aşağıda gösterildiği gibi çizdirilir.

```
>> dogru
*****İki Noktasi Bilinen Dogrunun Cizimi*****
1. Noktanin apsisini giriniz :3
1. Noktanin ordinatini giriniz :9
2. Noktanin apsisini giriniz :-5
2. Noktanin ordinatini giriniz :7
>>
```



**Örnek Program:** “ 3 farklı partinin yapılan genel seçimler sonucu aldıkları oy oranlarını pasta grafik dilimi üzerinde gösteren grafiğin çizimi ” programı aşağıda gösterildiği gibi Oy.m dosyası adı altında yazılmıştır. Çizilmesi istenen pasta dilimi grafiği için kullanıcıdan bu üç partiye ait yüzdelik oy oranı rakam olarak istenmekte ve grafik otomatik olarak çizdirilmektedir.

```
C:\matlabR12\work\Oy.m
File Edit View Text Debug Breakpoints Web Window Help
[Icons] Stack: Base
1 - clear
2 - disp('*****');
3 - disp('Partilerin Oy Oranlari Grafigi');
4 - disp('*****');
5 - A=input('A Partisinin aldigi oyun yuzdelik dilimdeki rakamini giriniz: ');
6 - B=input('B Partisinin aldigi oyun yuzdelik dilimdeki rakamini giriniz: ');
7 - C=input('C Partisinin aldigi oyun yuzdelik dilimdeki rakamini giriniz: ');
8 - V=[A B C];
9 - pie(V);
10 - title('Oy Oranlari');
11 - legend('A Partisi','B Partisi','C Partisi')
12
ist.m Oy.m
Ready
```

[illegible]41

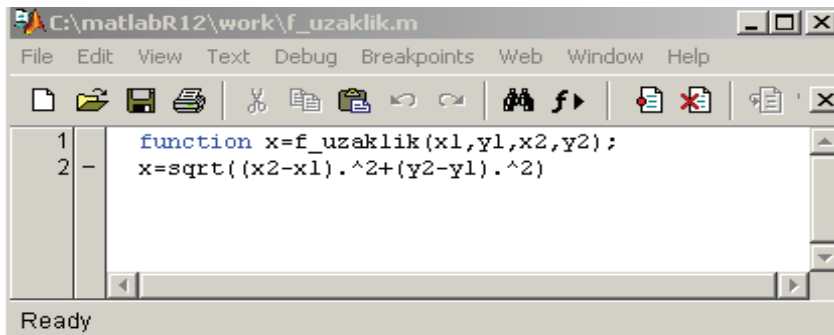


Bu program komut satırında çalıştırılırsa aşağıda gösterildiği gibi icra edilir

```
Birinci sayiyi giriniz: 4
ikinci sayiyi giriniz: 2
K>> c=10
c =
    10
K>> return
d =
    80
K>> return
>>
```

**Örnek Program:** İki nokta arasındaki uzaklığı bulan basit bir programı, önce fonksiyon yapısını kullanarak sonra da aynı programı bu fonksiyon yapısını bellekten çağırma işlemini uygulayarak icra ettirelim :

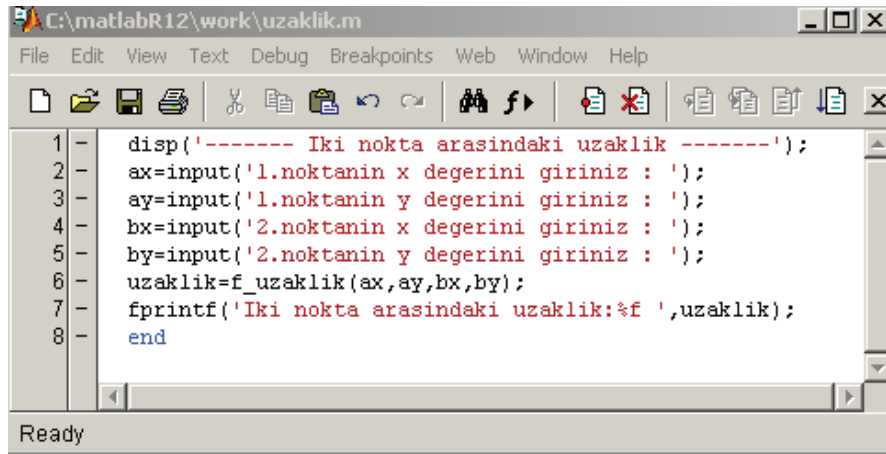
a) Fonksiyon yapısını kullanarak :



Komut satırından icra edilirse :

```
>> f_uzaklik(2,6,3,0)
x =
    6.0828
ans =
    6.0828
```

b) Fonksiyon yapısını bellekten çağırarak :



```
C:\matlabR12\work\uzaklik.m
File Edit View Text Debug Breakpoints Web Window Help
[Icons]
1 - disp('----- Iki nokta arasindaki uzaklik -----');
2 - ax=input('1.noktanin x degerini giriniz : ');
3 - ay=input('1.noktanin y degerini giriniz : ');
4 - bx=input('2.noktanin x degerini giriniz : ');
5 - by=input('2.noktanin y degerini giriniz : ');
6 - uzaklik=f_uzaklik(ax,ay,bx,by);
7 - fprintf('Iki nokta arasindaki uzaklik:%f ',uzaklik);
8 - end
Ready
```

Komut satırından icra edilirse :

```
>> uzaklik
----- Iki nokta arasindaki uzaklik -----
1.noktanin x degerini giriniz : 2
1.noktanin y degerini giriniz : 5
2.noktanin x degerini giriniz : 4
2.noktanin y degerini giriniz : 3
x =
    2.8284
Iki nokta arasindaki uzaklik:2.828427
```

## **D) DÖNGÜ ve ŞARTLI İFADE UYGULAMALARI :**

### **1.ŞARTLI İFADELER :**

-- **“ if ” yapısı :** “ if ” komutunun MATLAB’de 3 farklı şekli mevcuttur :

a) **if** koşul  
    deyim1  
    deyim 2  
    deyim\_n  
**end**

Koşul doğru ise deyim1, deyim1, ... , deyim\_n, ile belirtilen deyimler grubu icra edilir ve programın kontrolü end’i izleyen deyime geçer; koşul yanlış ise bu durumda deyim1, deyim2, ..., deyim\_n ile belirtilen deyimler grubu icra edilmeden kontrol end’i izleyen deyime geçecektir.

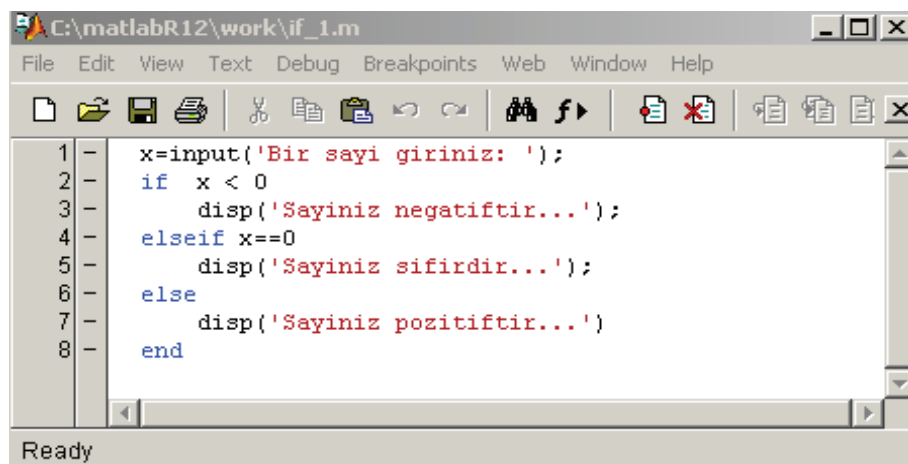
```
b) if koşul
    deyim1
    deyim 2
    deyim_n
else deyim_n+1
    deyim_n+2
    deyim_m
end
```

Koşul doğru ise deyim1, deyim1, ... , deyim\_n, ile belirtilen deyimler grubu icra edilir ve programın kontrolü end'i izleyen deyime geçer; koşul yanlış ise bu durumda da sadece else' i izleyen, deyim1\_n+1, deyim\_n+2, ... , deyim\_m ile belirtilen deyimler grubu icra edilecek ve kontrol end' i izleyen deyime geçecektir.

```
c) if koşul1
    deyim1
elseif koşul2
    deyim2
elseif koşul3
    deyim3
...
elseif koşul_n
    deyim_n
else
    deyim_n+1
end
```

Bu yapı içerisinde kontrol edilen koşullardan herhangi biri doğru ise onunla ilişkili deyim icra edilir ve kontrol end' i izleyen deyime geçer. Koşulların hepsi de yanlışsa, kontrol else' i izleyen deyim\_n+1'e geçer ve bu deyim de icra edildikten sonra kontrol end'i izleyen deyime geçecektir.

**Örnek Program:** Kullanıcı tarafından bir sayı istenip bu sayının pozitif, negatif veya 0 mı olduğunu sorgulayan ve ekrana yazdıran program aşağıda incelenebilir :



```
C:\matlabR12\work\if_1.m
File Edit View Text Debug Breakpoints Web Window Help
[Icons]
1 - x=input('Bir sayı giriniz: ');
2 - if x < 0
3 -     disp('Sayınız negatiftir...');
4 - elseif x==0
5 -     disp('Sayınız sifirdir...');
6 - else
7 -     disp('Sayınız pozitifdir...')
8 - end
Ready
```

Komut satırından icra edilirse :

```
Bir sayi giriniz: 55
Sayiniz pozitifdir...
>> if_1
Bir sayi giriniz: -9
Sayiniz negatiftir...
>> if_1
Bir sayi giriniz: 0
Sayiniz sifirdir...
```

-- “ **switch - case** ” **yapısı** : İkiiden fazla durumu kontrol etmek için, if – elseif – else – end yapısına alternatif olarak kullanılan bir kontrol yapısıdır.

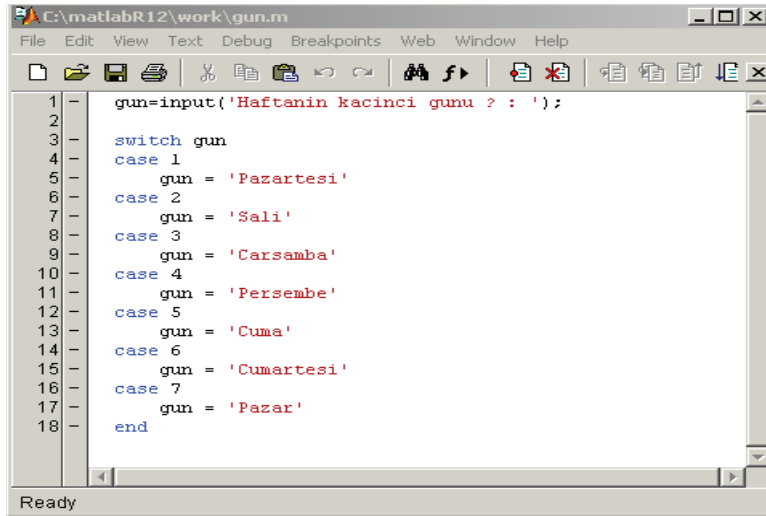
İfadenin değeri kontrol edilir ve buna göre farklı bir deyim veya deyimler grubu icra edilir. Örneğin, ifadenin değeri değer1 ise, deyim1 icra edilir ve kontrol end’i izleyen deyime geçer.

İfadenin değeri , değer2, değer3 ya da değer4 ’e eşitse bu durumda deyim2 icra edilir ve kontrol end’i izleyen deyime geçer.

İfadenin değeri, case’lerle kontrol edilen değerlerden hiç birine eşit değilse bu durumda da otherwise sözcüğünü izleyen deyim\_n+1 icr edilecektir.

```
switch(ifade)
case değer1
    deyim1
case değer2,değer3,değer4
    deyim2
    ...
case değer_n
    deyim_n
otherwise
    deyim_n+1
end
```

**Örnek Program:** Haftanın kaçınıcı gününün ne olduğunu bulan basit bir program aşağıda incelenebilir :



```
1 - gun=input('Haftanın kacinci gunu ? : ');
2 -
3 - switch gun
4 - case 1
5 -     gun = 'Pazartesi'
6 - case 2
7 -     gun = 'Sali'
8 - case 3
9 -     gun = 'Carsamba'
10 - case 4
11 -     gun = 'Persembe'
12 - case 5
13 -     gun = 'Cuma'
14 - case 6
15 -     gun = 'Cumartesi'
16 - case 7
17 -     gun = 'Pazar'
18 - end
```

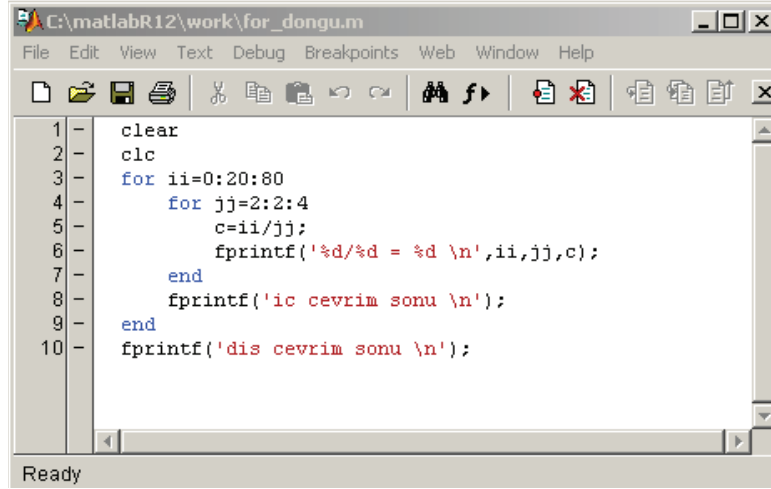
```
>> gun
Haftanın kacinci gunu ? : 5
gun =
    Cuma
```

## **2.DÖNGÜ İFADELERİ :**

-- “ **for** ” **döngüsü** : Parametre değeri başlangıç değerinden başlayarak ve her seferinde artım değeri kadar arttırılarak son değere erişene kadar değiştirilir. Parametrenin her değeri için, deyim1, deyim2, deyim\_n şeklinde belirtilen ve for-end sözcükleri arasında yer alan deyimler grubu icra edilir. Parametrenin değeri son değeri aşınca, programın kontrolü end’i izleyen deyime yani çevrimin dışına çıkacaktır.

```
for parametre=başlangıç:artım:son_değer
    deyim1
    deyim2
    ...
    deyim_n
end
```

**Örnek Program:** Arka arkaya bölme işlemlerinin yapıldığı aşağıdaki for döngüsü programını incelenebilir :



```
1 - clear
2 - clc
3 - for ii=0:20:80
4 -     for jj=2:2:4
5 -         c=ii/jj;
6 -         fprintf('%d/%d = %d \n',ii,jj,c);
7 -     end
8 -     fprintf('ic cevrim sonu \n');
9 - end
10 - fprintf('dis cevrim sonu \n');
```

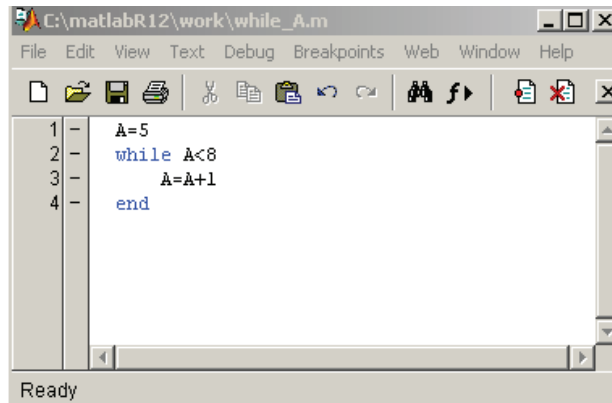
Komut satırından icra edilirse :

```
0/2 = 0
0/4 = 0
ic cevrim sonu
20/2 = 10
20/4 = 5
ic cevrim sonu
40/2 = 20
40/4 = 10
ic cevrim sonu
60/2 = 30
60/4 = 15
ic cevrim sonu
80/2 = 40
80/4 = 20
ic cevrim sonu
dis cevrim sonu
```

-- “ **while** ” **döngüsü** : Belirli bir üst sınıra kadar istenilen işlemleri tekrarlayarak yapar. Koşul doğru olduğu sürece, deyim1, deyim2, ... , deyim\_n şeklinde belirtilen deyimler grubunu icra eder. Koşul yanlış olduğu anda, end'i izleyen deyime yani çevrim dışına çıkar.

```
while koşul
    deyim1
    deyim2
    ...
    deyim_n
end
```

**Örnek Program:** A=5 ilk değerinden başlayarak A<8 olduğu sürece A'ya 1 eklemek için aşağıdaki program incelenebilir :



Komut satırından icra ettirilirse :

```
A =
    5
A =
    6
A =
    7
A =
    8
```

## **ALIŞTIRMALAR**

**1-** Kenarları kullanıcı tarafından istenen bir üçgenin çeşitkenar, ikizkenar veya eşkenar üçgen mi olduğunu bulan MATLAB programını yazınız.

**2-** “  $ax^2+bx+c$  ” ikinci dereceden denkleminin köklerini bulduran MATLAB proramını yazınız.

**3-** Bir otomobil, durgun halden harekete başlayarak 10 dakika boyunca hızlanıyor, hızı 60 km/saat oluyor. Sonra 15 dakika boyunca sabit hızla hareketine devam ediyor ve 10 dakika boyunca yavaşlayarak hızı 0 oluyor. Dışarıdan girilen herhangi bir t anında otomobilin hızını veren bir MATLAB programı yazınız.

**4-** Kullanıcıdan doğum gününü soran ve bu kullanıcın kaç yıl, kaç ay ve kaç yıl yaşadığını bulan bir MATLAB programı yazınız.