MATLAB NEDİR?

MATLAB®, temel olarak nümerik hesaplama, grafiksel veri gösterimi ve programlamayı içeren teknik ve bilimsel hesaplamalar için yazılmış yüksek performansa sahip bir yazılımdır. Matlab programının tipik kullanım alanları: Matematik and hesaplama işlemleri / Algoritma geliştirme / Modelleme, simülasyon (benzetim) ve öntipleme / Veri analizi ve görsel efektlerle destekli gösterim / Bilimsel ve mühendislik grafikleri / Uygulama Geliştirme şeklinde özetlenebilir.

MATLAB adı, *MATrix LABoratory (Matrix Laboratuarı)* kelimelerinden gelir. MATLAB, ilk olarak Fortran Linpack ve Eispack projeleriyle geliştirilen ve bu programlara daha etkin ve kolay erişim sağlamak amacıyla 1970'lerin sonlarında yazılmıştır. İlk başlarda bilim adamlarına problemlerin çözümüne matris temelli teknikleri kullanarak yardımcı olmaktaydı. Bugün ise geliştirilen yerleşik kütüphanesi ve uygulama ve programlama özellikleri ile gerek üniversite ortamlarında (başta matematik ve mühendislik olmak üzere tüm bilim dallarında) gerekse sanayi çevresinde yüksek verimli araştırma, geliştirme ve analiz aracı olarak yaygın bir kullanım alanı bulmuştur. Ayrıca işaret işleme, kontrol, fuzzy, sinir ağları, wavelet analiz gibi bir çok alanda ortaya koyduğu **Toolbox** adı verilen yardımcı alt programlarla da özelleştirilmiş ve kolaylaştırılmış imkanlar sağlamış ve sağlamaya da devam etmektedir.
Web adresi: "http://www.mathworks.com" Email: info@mathworks.com

Matlab, komut temelli bir programdır. Command Window penceresinde » işareti Matlab'in komut prompt'unu gösterir ve bu işaret bulunduğu satır komut satırı olarak adlandırılır. Bu işaretin hemen yanında yanıp sönen **I** şeklinde ki işaret komut ve metin yazma cursor'u yani imlecidir. Bu işaretin olduğu yerde klavyeden giriş yapılabilir demektir.

İFADELER (EXPRESSIONS)

Matlab diğer programlama dillerinde olduğu gibi bir giriş (input) olarak çeşitli matematiksel ve metinsel ifadeler sağlar. İfadeleri 4 ana madde altında inceleyebiliriz:

- Sayılar (Numbers)
- Değişkenler (Variables)
- İşleçler (Operators)
- Fonksiyonlar (Functions)

Matlab'de ifadeler genellikle **değişken = ifade** veya basitce sadece **ifade** formundadır.

- Bir ifade = işareti ile bir değişkene atanmamış ise Matlab otomatik olarak sonucu, **ans** adı verilen özel bir değişkende saklar.

```
» b=4*5
b =
   20
» log(2)
ans =
   0.6931
```

- Normalde ifadeler, tek bir satırda yer alırlar. Ancak bir kaç ifade aralarına virgül konarak yazılabilir ve işleme konabilir. Virgülden sonraki boşluk sayısında bır sınırlama yoktur. Komut veya değişken bildirimli ifadelerde ; noktalı virgül işaretini de kullanılabilirsiniz.

```
» x=3.01,  y=(23*7)/5,  z =          3^6
x =
   3.0100
y =
  32.2000
z =
   729
```

MATLAB TEMEL YAZIM NOTASYONU (SYNTAX)

Matlab'de herşey bir dizi (array) olarak işleme konur. Bir dizi, skaler, vektör, matris veya metinsel dizge (karakter dizisi) olabilir.

- 1x1 dizisi skaler (scaler) veya metin dizgesi (string) gösterir. a=3 b=-20.56 c=3e-4 d=2+5*i e='Bir tamsayı giriniz:' gibi. Metin dizgeleri (text string), '…' arasında verilir.
- nx1 veya 1xn dizisi bir vektör (vector) gösterir. x=[0, 2, 4, 6] ifadesinde x, 1x4 boyutunda bir vektördür. [ ] köşeli parantezler içindeki sayıların arasında virgül (,) veya en az bir veya daha fazla boşluk olmalıdır.
- nxm veya mxn dizisi bir matris (matrix) gösterir. [ ] köşeli parantezler içindeki sayı grupları arasında noktalı virgül (;) olmalıdır. ; işareti matrisin sütunlarını ayırır. Sayı gruplarında virgül (,) veya en az bir veya daha fazla boşluk olabilir.

MATLAB'DE KULLANILAN ÖZEL DEĞERLER VE SABİTLER

Değişken Adı	Açıklama
ans	Bir işlemin son cevabı (answer) veya bir ifadenin değeri
pi	pi sabiti: π=3.1415926535897...
i veya j	imajiner birim, √-1
eps (epsilon yani ε)	Kayan-nokta bağıl doğruluğu (relative precision), $2e^{-52}$
realmin	En küçük kayan-nokta (floating-point) sayısı, $2e^{-1022}$
realmax	En büyük kayan-nokta (floating-point) sayısı, $(2-\varepsilon)e^{1024}$ yani $2e^{1024}$'den çok az küçük
inf	Sonsuz (infinity) yani realmax dan daha büyük bir sayı
NaN	Sayı değil (Not-a-Number)

KOMUT SATIRINDA KLAVYE KISAYOLLARI

↑	Ctrl-P	Bir önceki satırı çağırır (Recall previous line)
↓	Ctrl-N	Bir sonraki satırı çağırır (Recall next line)
←	Ctrl-B	Imleç bir karakter sola hareket eder (Move back one character)
→	Ctrl-F	Imleç bir karakter sağa hareket eder (Move forward one character)
ctrl-→	Ctrl-R	Bir kelime sağa hareket eder (Move right one word)
ctrl-←	Ctrl-L	Bir kelime sola hareket eder (Move left one word)
home	Ctrl-A	Satırın basına gider (Move to beginning of line)
end	Ctrl-E	Satırın sonuna gider (Move to end of line)
esc	Ctrl-U	Geçerli satırı iptal eder yani tamamen siler (Clear line)
del	Ctrl-D	Bir karakter sola doğru siler (Delete character at cursor)
backspace	Ctrl-H	Imlecten satırın başına kadar olan kısmı siler (Delete character before cursor)
	Ctrl-K	Imlecten satırın sonuna kadar olan kısmı siler (Delete to end of line)

Format Komutu: Matlab, değerlerin sayısal formatını **format** komutu ile kontrol eder. Bu komut sayıların kaç hane yani kaç ondalıkla veya diğer bir deyişle kaç digit gösterileceğini belirler. Matlab'de tüm hesaplamalar, double precision da yapılır.

Komut	**Açıklama**
format	Varsayılan format (format short ile aynıdır)
format short	5 rakamlı sabit nokta skala formatı
format long	15 rakamlı sabit nokta skala formatı
format short e	5 rakamlı kayan nokta formatı
format long e	15 rakamlı kayan nokta formatı
format short g	5 rakamlı en uygun sabit veya kayan nokta formatı
format long g	15 rakamlı en uygun sabit veya kayan nokta formatı
format bank	Dolar ve sent için sabit format (İki ondalıklı gösterimler için kullanabilirsiniz)
format rat	En küçük tamsayı oranı yaklaşımıyla sayıyı ondalıklı olarak gösterir

TRİGONOMETRİK FONKSİYONLAR

	Trigonometrik		**Hiperbolik**	
Fonksiyon Adı	**Fonksiyon formatı**	**Ters Fonksiyon formatı**	**Fonksiyon formatı**	**Ters Fonksiyon formatı**
sinus	sin(x)	asin(x)	sinh(x)	asinh(x)
cosinüs	cos(x)	acos(x)	cosh(x)	acosh(x)
tanjant	tan(x)	atan(x)*	tanh(x)	atanh(x)
kotanjant	cot(x)	acot(x)	coth(x)	acoth(x)
sekant	sec(x)	asec(x)	sech(x)	asech(x)
kosekant	csc(x)	acsc(x)	csch(x)	acsch(x)

* atan2(y,x), dördüncü bölge değeridir.

Matlab'de trigonometrik fonksiyonlarda *derece* yerine *radian* kullanılır. Bu nokta çok onemlidir. Kısaca $2\pi=360^o$ veya $\pi=180^o$ derecedir. Örneğin derece cinsinden sin(30)=0.5 tir. Oysa Matlab bu işlemin sonucunu

```
» sin(30)
ans =
      -0.99
```

şeklinde verir. Genel kullanım derece cinsinden olduğundan ya radian (**pi** değişkenini kullanarak) cinsinden belirtmeniz ya da $\pi/180$ ile çarpanız gereklidir. Programlama bölümünde örneğin direkt girilen sayının derece olarak algılanıp değerini veren bir fonksiyonda yazılabilir. Örneğimize devam edersek;

```
» sin(pi/6)
ans =
       0.50
```

Ancak bu yöntemde π yani pi'ye çevirmek (örneğin 6.47 açısı gibi) her zaman bu kadar olmayacağından

```
» sin(30*pi/180)
ans =
       0.50
```

komutu yani açıyı parentezler içinde kalmak şartıyla pi/180 ile çarpmak daha pratiktir.

Ör: $x=45^o$ için $(\sin 4x)-(2\cos x)^3$ ifadesinin değerini bulunuz.

1. yol:
» x = pi/3;
» sin(4*x)-(2*cos(x))^3
ans =
-2.83

2.yol:
» x = 45;
» sin(4*x*pi/180)-(2*cos(x*pi/180))^3
ans =
-2.83

LOGARITMIK VE ÜSTEL FONKSIYONLAR

Fonksiyon Adı	Fonksiyon formatı
10 tabanında logaritma ($\log_{10}x$)	log10(x)
2 tabanında logaritma ($\log_2x$)	log2(x)
doğal logaritma (lnx)	log(x)
üstel (e^x)	exp(x)
karekök ($\sqrt{x}$)	sqrt(x)
üsalma (x^n, n herhangibir sayı)	x^n

1- Matlab'de matematikten bildiğimiz doğal logaritma gösterimi **ln** olarak değil doğrudan **log** olarak gösterilmektedir. Yine bildiğimiz gibi lnx=$\log_e$x demektir. Genel yazım formatı bir x değeri için lnx, Matlab'de log(x) şeklindedir.

ln1	**ln10**	**ln2**
» log(1)	log(10)	» log(2)
ans =	ans =	ans =
0	2.3026	0.6931

Matlab'de e sabit sayısı yani e=2.71828 veya kısaca e=2.71 sayısı e olarak tanımlanmamıştır. Bunun yerine bir sonraki konuda göreceğimiz exp(1) fonksiyonu kullanılabilir. Biliyoruz ki lne=1 dir ve bunu Matlab de sağlayalım:

» exp(1)
ans =
2.7183

2- Matlab'de matematikten bildiğimiz normal logaritma 10 tabanındadır ve bir x değeri için genel yazım formatı log10(x) şeklindedir. Ayrıca Matlab, 2 tabanındaki logaritma içinde hazır bir fonksiyon sağlar. Bir x değeri için genel yazım formatı log2(x) şeklindedir. şeklindedir. Doğal logaritma da olduğu gibi negatif sayıların logaritmaları reel sayı değildir ve sıfır için değeri sonsuzdur. 0 ile 1 arasındaki (0 ve 1 dahil değil) değerleri negatiftir. Şimdi sırasıyla log1, log10, log100 ve log 1000, sonra log2, log5, log3/5 ve sonra da log0 ve log(-4) değerlerini bulalım.

log1	**log10**	**log5**
» log10(1)	» log10(10)	» log10(5)
ans =	ans =	ans =
0	1	0.6990

Not: Bilimsel notasyondan bildiğimiz gibi 1.0966e+003=$1.0966.10^3$ demektir.

Şimdi de aşağıdaki üstel yazılımlı ifadelerin değerleri bulalım :

3^{12}, $(1/2)^{-4}$, $20^{1/5}$, $10^{3/5}$, $5.62.10^{-5}$,

» 3^12, (1/2)^(-4), 20^(1/5), 10^(3/5), 5.62*10^(-5)

3- Bu konu başlığı altında çok kullanılan bir diğer fonksiyonumuz karekök alma işlemini gerçekleştiren sqrt fonksiyonudur. Genel yazım formatı bir x değeri için sqrt(x) şeklindedir. sqrt fonksiyonunu kullanarak sırasıyla $\sqrt{2}$, $\sqrt{23}$, $\sqrt{144}$, $\sqrt{(3+\sqrt{2})}$ işlemlerini yapalım.

» sqrt(2), sqrt(23)
ans =
1.4142
ans =
4.7958

KARMASIK (KOMPLEKS) SAYI İSLEMLERI

Bilindiği gibi kompleks sayıların tipik genel formatı a + bi , a+bj veya a + ib , a+jb şeklindedir. Matlab dilinde bu notasyon a + bi, a + bj veya a+i*b , a+j*b şeklinde ifade edilir. Bu gösterim şekli aynı zamanda kartezyen gösterim olarak da adlandırılır. Sayılarda ı veya j kullanımı arasında fark yoktur her ikisi de aynı şeyi ifade ederler. Örnek olarak 2-3j karmaşık sayısını ele alalım.

» 2-3j	» 2-i3	» 2-i*3
ans =	??? Undefined function or variable 'i3'.	ans =
2.0000 - 3.0000i		2.0000 - 3.0000i

Temel kompleks sayı işlemleri:

abs	Mutlak değer (Absolute value)
angle	Faz açısı (Phase angle)
conj	Kompleks eşlenik (Complex conjugate)
imag	Kompleks imajiner kısım (Complex imaginary part)
real	Kompleks reel kısım (Complex real part)

x=3+4j sayısı için özetlersek

Komut	*Sonuç*
real(x)	3
imag(x)	4
abs(x)	$\sqrt{(3^2+4^2)}=5$
angle(x)	$\tan^{-1}(4/3)=0.9273$
conj(x)	3-j*4

! abs (absolute) komutu sadece karmaşık sayı işlemlerinde değil diğer tüm mutlak değer alma $|x|$ işlemlerinde kullanılabilir. Fonksiyon adı **abs** olup genel formatı bir x değeri için abs(x) şeklindedir. Ör: » abs(sqrt(3)-1) ans = 0.7321

1) x=2(1+4j)	**2)** $k=(1.2+2.5i)^3$	**3)** a=-2-j ve b=3+√2 olmak üzere s1) -5a+b s2) a/b
» x=2*(1+4*i) x = 2.0000 + 8.0000i	» k=(1.2 + 2.5*i)^3 k = -20.7720 - 4.8250i	» a=-2-j; » b=3+sqrt(2); » s1=-5*a+b s1 = 8.4142 + 2.0000i » s2=a/b s2 = -0.4531 - 0.2265i

YUVARLATMA İŞLEMLERİ

fix : Sıfıra doğru yuvarlatma yapar
floor : -∞ 'a doğru en yakın tamsayıya yuvarlatma yapar
ceil : +∞ 'a doğru en yakın tamsayıya yuvarlatma yapar
round : En yakın tamsayıya yuvarlatma yapar

» fix(-5.1)	» floor(-5.1)	» ceil(-5.1)	» round(-5.1)
ans =	ans =	ans =	ans =
-5	-6	-5	-5

KALAN BULMA İŞLEMLERİ

Matlab'de bölme işlemi sonucu kalan bulma işlemi iki şekilde yapılır:
mod - Modül (Bölme işleminde işaretli kalan)
rem - Bölme işleminde kalan

rem(x,y), eğer y ~= 0 (y≠0) ise x - y.*fix(x./y) demektir. rem(x,0) değeri NaN'dır.
mod(x,y), eğer y ~= 0 (y≠0) ise x - y.*floor(x./y) demektir. mod(x,0) değeri x'dir.

rem fonksiyonu örnekleri:	mod fonksiyonu örnekleri:
» rem(15,2) ans = 1	» mod(15,2) ans = 1

TEMEL İSTATİKSEL İŞLEMLER

max : Verilerin en büyük değerini bulur
min : Verilerin en küçük değerini bulur
length : Veri sayısını bulur
sum : Verilerin toplamını hesaplar
prod : Verilerin çarpımını hesaplar
median : Verilerin ortanca değeri hesaplar
std : Verilerin standart sapmasını hesaplar
mean : Verilerin ortalama değerini hesaplar yani aritmetik ortalama alır
geomean : Verilerin geometrik ortasını hesaplar
harmmean : Verilerin harmonik ortasını hesaplar
sort : Verilerin azalan sırada sıralar

» d=[0.5 1 0.34 2.5 2.5 1.14 3.0 3.4 5 6.5 4.31 5.5] ;

» max(d)	» min(d)	» length(d)	» sum(d)
ans = 6.5000 (En büyük değeri bulur)	ans = 0.3400 (En küçük değeri bulur)	ans = 12 (Vektörün boyunu yani veri sayısını bulur)	ans = 35.6900 (0.5+1.0+0.34+2.5+ …+4.31+5.5 toplamını bulur)

İstatiksel işlemler matrislerde sütün sütun işlem yapar. A bir matris ise sum(A) A matrisinin sütularını ayrı ayrı toplar.

DİZİLER

Matlab'in en temel işlem elemanı ve veri tipi dizilerdir (array). Dizi, en genel matematiksel tanımı ile nümerik ve metinsel değerler topluluğudur. Matlab'de herşey bir dizi olarak işleme konur. Matlab'de üç tip dizi ifadesi bulunmaktadır:

1. Reel ile kompleks sayıları ifade eden çiftkat veya nümerik diziler (double veya numeric array)

2. Nesneleri ve metinsel dizgeleri ifade eden hücre diziler (cell array)
3. Genelleştirme ve çeşitli tipleri ifade eden n-boyutlu diziler (n-dimensional array)

VEKTÖR İŞLEMLERİ

Vektörler, mx1 veya 1xn boyutlu dizilerdir. mx1 boyutlu diziye sütun vektörü denir ve eleman sayısı m tanedir; 1xn boyutlu diziye sütun vektörü denir ve eleman sayısı n tanedir. Matlab'de vektörleri oluşturmanın üç temel yolu vardır:

1. Direkt olarak (köşeli parantez **[…]** kullanma)
2. Eşit aralıklı elemanlar kullanarak (**:** işaretini kullanarak veya **linspace**, **logspace** komutlarıyla)
3. Utility fonksiyonlar kullanarak (**rand**, **randn**, **ones**, **zeros** komutlariyla)

Temel Vektör İşlem Notasyonları

İşlem	**Matlab formu**	**Örnek Uygulama** a=[1 2 3] , b=[-1 2 6]	**Açıklama**
Toplama	a + b	0 4 9	Dizilerin karşılıklı elemanları toplanır.
Çıkarma	a – b	2 0 -3	Dizilerin karşılıklı elemanları çıkartılır.
Çarpma	a .* b	-1 4 18	Dizilerin karşılıklı elemanları çarpılır.
Sağa Bölme	a ./ b	-1.0000 1.0000 0.5000	a dizisinin her bir elemanı, sırasıyla b dizisinin her bir elemanına bölünür.
Sola Bölme	a .\ b	-1 1 2	b dizisinin her bir elemanı, sırasıyla a dizisinin her bir elemanına bölünür.
Üs alma	a .^ b	1 4 729	a dizisindeki her bir elemanın, sırasıyla b dizisindeki elemanlarla üsleri alınır..
Transpoze	a'	1 2 3	Satır vektörünü sütun vektörüne çevirir veya tersini yapar.

* Matlab dilinde nokta işaretli işlemler (**dot** işlemleri) vektörde eleman eleman (elemanter) işlem yapacağını gösterir.
Çarpma: **.*** Bölme: **./** veya **.** ve Üsalma: **.^**

Eşit aralıklı elemanlar kullanarak vektör oluşturma

Bu yöntem ile Matlab'de vektör oluşturma üç şekilde olur:

1- *Vektör elemanları birbirlerini, sabit miktarda artan veya azalan bir değerle (step size) takip ederler.* ***:*** *işleci (colon operator) bu tür bir işlem için en temel bir yöntemdir. Genel sözdizimi formatı:*

f = İlkDeger : DeğişimMiktarı : SonDeger

şeklindedir. Değişim miktarı belirtilmezse IlkDeger'den sonra 1'er er artım olacağını ifade eder. : işaretinden önce veya sonra görüntü netliği için boşluk verebilirsiniz.

Örneğin,

```
» n = 1:10
n =
    1    2    3    4    5    6    7    8    9   10
```

n değişkeni 1, 2, 3, 4 ... ve 10 tamsayılarını üretir diğer bir deyişle elemanları [1 2 3 4 5 6 7 8 9 10] olan bir n satır vektörü gösterir. Görüldüğü gibi artım miktarı belirtilmezse Matlab bunu 1 birim olarak kabul eder.

```
» p = 0.2:0.25:1
p =
   0.2000    0.4500    0.7000    0.9500
```

p değişkeni 0.2 ile 1 arasında 0.25 artımla [0.2 0.45 0.7 0.95] satır vektörünü üretir.

2- **linspace** ve **logspace** komutlarını kullanmak. Bu durumda başlangıc ve bitiş noktaları arasında kaç nokta olacağını siz belirtirsiniz.

linspace *komutunun genel sözdizimi formatı:*

linspace(x1, x2, n)

şeklindedir. x1, aralığın IlkDeger ile x2, SonDeger değerleridir. n, IlkDeger ile SonDeger arasındaki nokta sayısıdır. Eğer n belirtilmezse iki nokta arası lineer olarak 100 eşit parçaya ayrılır. **linspace,** lineer aralıklı bir vektör üretir. **linspace** özellikle eğri çizimlerinde ve eğri uydurma ilerinde çok yararlıdır.

logspace *komutunun genel sözdizimi formatı:*

logspace(x1, x2, n)

şeklindedir. n, IlkDeger (x1) ile SonDeger (x2) arasındaki nokta sayısıdır. Eğer n belirtilmezse 10^{x1} ile 10^{x2} arası logaritmik olarak eşita aralıklı 50 satır vektörü üretir. **logspace,** logaritmik aralıklı bir vektör üretir ve aslında logaritmik ölçekte **linspace** komutunun rolunu oynar. Bir vektörde logaritmik aralıklı elemanlar özellikle üstel fonksiyonlarla (log-log ve semilog grafikler gibi) ilgili iseniz çok yararlıdır. Sistem frekans cevabı, Bode diyagramları vb gibi logaritmik ölçek gerektiren grafik çizimlerinde kullanabilirsiniz.

Utility fonksiyonlarla üretilen utility vektörler

1- rand *fonksiyonunu kullanmak. Bazen sadece bir özelliği veya bir şeyi denemek ve durumunu gözlemek için bir sayı vektörü oluşturmak isteyebilirsiniz. İşte* ***rand*** *uniform olarak dağılmış rastgele sayılı vektörler üretir.*

rand *fonksiyonu için genel sözdizimi formatı:*

f = a + (b-a)*rand(m,n)

şeklindedir. Burada f vektörü, a ile b sayıları arasında uniform olarak dağılmış rastgele sayılardan oluşur. m ve n vektör boyutunu belirler, tabiki en az biri m=1 veya n=1 olmalıdır. m=1 ise n sütun sayıda satır vektörü, n=1 ise m satır sayıda sütun vektörü üretilir. Sadece **rand** komutunun kullanımı ile 0 ile 1 arasında rastgele sayılar üretirsiniz. Örneğin a=1 ile b=5 arasında yani 1 ile 5 arasında rasgele 7 sayı üretmek istiyorsanız

```
» r = 1 + 4*rand(1,7)
r =
   1.0470    4.5756    1.7966    2.1949    3.6458    2.1376    2.8769
```

2- ones *ve* ***zeros*** *fonksiyonlarını kullanmak. Bu fonksiyonlardan* ***ones*** *ile elemanları sadece 1'lerden oluşan,* ***zeros*** *ile elemanları sadece 0'lardan oluşan bir vektör üretilir.* Genellikle ones, aynı değerli bir vektör oluşturmak için; zeros ise script ve fonksiyon işletimini hızlandırmak için kullanılır.

ones *fonksiyonu için genel sözdizimi formatı:*

f = k*ones(m,n) veya **f = k*ones[m,n]**

şeklindedir. Burada m ve n vektör boyutunu belirler, tabiki en az biri m=1 veya n=1 olmalıdır. m=1 ise n sütun sayıda satır vektörü, n=1 ise m satır sayıda sütun vektörü üretilir. k=1 için elemanları sadece 1 olan vektör, k≠1ve 0 için elemanları k olan bir vektör elde edilir.

zeros *fonksiyonu için genel sözdizimi formatı:*

f = zeros(m,n) veya **f = zeros[m,n]**

şeklindedir. Burada m ve n vektör boyutunu belirler, tabiki en az biri m=1 veya n=1 olmalıdır. m=1 ise n sütun sayıda sıfırlardan oluşan satır vektörü, n=1 ise m satır sayıda sıfırlardan oluşan sütun vektörü üretilir.

```
» dortler = 4*ones(1,4)
dortler =
      4.00      4.00      4.00      4.00
```

komutu 4 elemanlı her bir elemanı 4 olan vektör üretir.

```
» V_sıfır = zeros(1,4)
V_sifir =
       0       0       0       0
```

komutu 4 elemanlı her bir elemanı sıfır olan vektör üretir.

Vektör Bilgilerini Elde Etmek (bir f vektörü için)

size(f): komutu vektörün 1xn veya nx1 olarak kaç n boyutunda olduğunu verir. **Size** komutunun ilk değeri satır sayısını son değeri sutun sayısını verir.

length(f): komutu vektörün uzunluğunu diğer bir deyişle boyunu yani vektörün kaç elemanı olduğunu gösterir. Bu komut yerine **max(size(A))** komutu da kullanılabilir.

f(n): komutu vektörün n. elemanını (n=1,2,3,...) gösterir. f(5), f vektörünün 5. elemanını gösterir.

f(1:5): komutu vektörün ilk beş elemanını gösterir.

Bir vektörün ilk elemanı örneğimizdeki f vektörü için f(1) ile ve son terimi f(length(f)) ile bulunur.

MATRİSLER

Matlab'de matrisleri oluşturmanın üç temel yolu vardır:

1. Direkt olarak (köşeli parantez **[…]** kullanma)
2. Utility fonksiyonlar kullanarak (**eye**, **ones**, **zeros, rand, randn** komutlarıyla)
3. Özel matrisler (**pascal**, **hilbert** vb fonksiyonlarla)

Utility fonksiyonlar kullanarak (rand, ones, zeros, eye komutlarıyla) matris oluşturma

a) rand *fonksiyonunu kullanmak. Bazen sadece bir özelliği veya bir şeyi denemek ve durumunu gözlemek için rastgele sayılardan oluşmus bir matris oluşturabilirsiniz. İşte* ***rand*** *uniform olarak dağılmış rastgele sayılı matrisler üretir.*

rand *fonksiyonu için genel sözdizimi formatı:*

F = a + (b-a)*rand(m,n)

şeklindedir. Burada F matrisi, a ile b sayıları arasında uniform olarak dağılmış rastgele sayılardan oluşur ve mxn, matris boyutunu belirler. Sadece **rand(k)** komutunun kullanımı ile kxk boyutunda 0 ile 1 arasında rastgele sayılı matris üretirsiniz. Diğer bir ifade ile **rand(k)** ile **rand(k,k)** aynı işleve sahiptir.

Örneğin a=-5 ile b=5 arasında yani -5 ile +5 arasında rasgele sayılı 2x4 (iki satır 4 sütunlu) bir matris üretmek istiyorsanız

```
» a = -5 + 10*rand(2,4)
a =
  -0.5490   -0.3401    3.4622   -2.9735
   4.3181   -0.8135    0.2515    1.7214
```

b) ones *ve* ***zeros*** *fonksiyonlarını kullanmak. Bu fonksiyonlardan* ***ones*** *ile elemanları sadece 1'lerden oluşan,* ***zeros*** *ile elemanları sadece 0'lardan oluşan matrisler üretilir.*

ones *fonksiyonu için genel sözdizimi formatı:*

F = k*ones(m,n) veya **F = k*ones([m,n])**

şeklindedir. Burada mxn matris boyutunu belirler. F matrisi, k=1 için elemanları sadece 1 olan vektör, k≠1 ve k≠0 için elemanları k olan mxn boyutunda bir matris gösterir. ones(t), tüm elemanları 1 olan boyutu txt olan kare matris üretir.

zeros *fonksiyonu için genel sözdizimi formatı:*

F = zeros(m,n) veya **F = zeros([m,n])**

şeklindedir. Burada mxn matris boyutunu belirler ve F, elemanları sadece 0 olan mxn boyutunda bir matris gösterir. ones(t), tüm elemanları 1 olan boyutu txt olan kare matris üretir. zeros(t), tüm elemanları 0 olan boyutu txt olan kare matris üretir.

Genellikle **ones**, aynı değerli bir matris oluşturmak ve bazı işlemlerde yardımcı bir araç olarak ; **zeros** ise script ve fonksiyon işletimini hızlandırmak, mühendislik işlemlerinde *sparce* matris oluşturmak ve yine bazı işlemlerde yardımcı araç olarak kullanılır.

» e = ones(3) e = 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 komutu 3x3 boyutunda her bir elemanı 1 olan kare matris üretir. » e = ones(2,3) e = 1.00 1.00 1.00 1.00 1.00 1.00 komutu 2x3 boyutunda her bir elemanı 1 olan bir matris üretir.	» g = zeros(3) g = 0 0 0 0 0 0 0 0 0 komutu 3x3 boyutunda her bir elemanı 0 olan kare matris üretir. » g = zeros(2,3) g = 0 0 0 0 0 0 komutu 2x3 boyutunda her bir elemanı 0 olan bir matris üretir.

c) eye *fonksiyonunu kullanmak. Bu fonksiyon ile birim matris oluşturulur. Bilindiği gibi birim matris, birinci diyagoneli 1 olan diğer tüm elemanları 0 olan bir matristir.* Genelde büyük I harfi ile temsil edilir. ***eye*** *fonksiyonu için genel sözdizimi formatı:*

F = eye(m,n) veya **F = eye([m,n])**

şeklindedir. Burada mxn matris boyutunu belirler. **eye**(t), boyutu txt olan kare matris üretir. Birim matris genelde kare matris olarak kullanılır.

```
» I = eye(3)
I =
     1.00        0        0
        0     1.00        0
        0        0     1.00
```

Özel matrisler (pascal, magic, hilbert, spiral, meshgrid vb fonksiyonlarla) oluşturma

a) **pascal(k)** fonksiyonu: Pascal üçgeni elemanlarından oluşur. Ör. » pascal(4)

b) **magic(k)** fonksiyonu: kxk boyutunda 1'den k^2 'ye kadar sayılardan oluşan k=2 hariç eşit satır, sütun ve diagonal toplamına sahip bir kare matris oluşturur. Ör. » magic(3)

c) **hilb(k)** fonksiyonu kxk boyutunda her bir elemanı 1/(i+j-1) ile belirlenen hilbert matrisi olarak adlandırılan bir kare matris oluşturur. Matrisin genel elemanı hilb(ij) olup i satır,j sütun numarasını gösterir. Ör. >>hilb(5)

MATRİS BİLGİ ALMA FONKSİYONLARI

Matlab'de yerleşik bazı matrisler ilgili fonksiyonlar ile matrisler hakkında bilgi alabiliriz. Bu bilgiler arasında matrisin determinat'ı, rank'ı, matrisin boyutu, matrisin veri özellikleri, diagonel elemanları vb. sayılabilir.

```
      1   4   7
A = -2   5  -8
      3   6   1
```

matrisi üzerinde bazı işlemler yapalım.

```
» A =[1   4   7
     -2   5  -8
      3   6   1]
```

Matrisin transpozesi : Bir A matrisinin transpozesi A' ile bulunur. ' transpoze operatörü ile matrisin satır ve sütunları yer değiştirir. (A')'=A dır yani transpozenin transpozesi matrisin kendisidir.

```
» TranspozeA = A'
TranspozeA =
    1   -2    3
    4    5    6
    7   -8    1
```

Matrisin determinantı: Bir A matrisinin determinantı **det**(A) fonksiyonu ile bulunur. Matris, kare matris olmalıdır.

```
» A = [1 4 7; -2 5 -8; 3 6 1];
» detA=det(A)
detA =
  -224
```

Matrisin tersi: Bir A matrisinin tersi **inv**(A) fonksiyonu ile bulunur. Matris, tekil (singular) ve kare (square) matris olmalıdır. Konu dışı olmakla beraber karesel olmayan matrislerin tersini alabilmek için **pinv** komutu kullanılır.

```
» A = [1 4 7; -2 5 -8; 3 6 1];
» TersA=inv(A)
TersA =
   -0.2366   -0.1696    0.2991
    0.0982    0.0893    0.0268
    0.1205   -0.0268   -0.0580
» B = [1 0 3; 4 0 6; 2 0 5];
» inv(B)
Warning: Matrix is singular to working precision.
ans =
   Inf   Inf   Inf
   Inf   Inf   Inf
   Inf   Inf   Inf
```

Görüldüğü gibi eğer matrisiniz singular yani tekil değilse kare matris de olsa tersini alamazsınız. B matrisinin determinantı da **det(B)**=0'dır zaten. Benzer olarak H=[1 2 -1 ; 3 2 3; 2 2 1] matrisini deneyiniz.

Matrisin rankı: Bir A matrisinin rankı **rank**(A) fonksiyonu ile bulunur.

```
» A = [1 4 7; -2 5 -8; 3 6 1];
» rankA=rank(A)
rankA =
    3
```

Matrisin boyutu: Bir A matrisinin boyutu yani kaça kaç olduğu **size**(A) fonksiyonu ile bulunur. Verilen cevapta yan yana iki sayıdan birincisi satır sayısını, ikincisi sütun sayısını gösterir.

```
» A = [1 4 7; -2 5 -8; 3 6 1];
» boyutA=size(A)
boyutA =
    3    3
```

yani A matrisi 3x3 boyutlu bir matristir. Daha açık bir şekilde görmek için aşağıdaki komutta kullanılır.

Matrisin diagonel elemanları: Bir A matrisinin ana diagonel elemanları **diag**(A) fonksiyonu ile bulunur.

```
» A = [1 4 7; -2 5 -8; 3 6 1];
» ana_diag=diag(A)
ana_diag =
    1
    5
    1
```

Matrisin özdeğerleri ve özvektörleri: Matrisin öz değerleri ve özvektörleri **eig**(A) fonksiyonu ile bulunur.

```
» A = [1 4 7; -2 5 -8; 3 6 1];
» OzDegerler=eig(A)
OzDegerler =
  -2.9650
   4.9825 + 7.1219i
   4.9825 - 7.1219i
```

Hem ozvektörleri hem de özdeğerlerini bulmak için

```
» [v,d]=eig(A)
v =
  -0.8965            0.4700 - 0.2917i   0.4700 + 0.2917i
   0.1810            0.1587 + 0.6635i   0.1587 - 0.6635i
   0.4044            0.4735 - 0.0669i   0.4735 + 0.0669i
d =
  -2.9650                 0                 0
       0            4.9825 + 7.1219i        0
       0                  0            4.9825 - 7.1219i
```

komutu kullanılır. v, sütunlar olarak özdeğer vektörlerini veya özvektörler matrisini ; d, ana diagonel boyunca özdeğerleri gösterir.

MATRİS İNDEKSLEME VE KOLON (:) NOTASYONU

Bir A matrisini düşünürsek

- **A(k) gösterimi** k=1,2 ... olmak üzere k. elemanını verir. Eleman sırası ilk sütun elemanlarından başlar ikinci sütun elemanları vb şeklinde devam eder.
- **A(i,j) gösterimi** i. satır j. sütun elemanını verir.
- **A(i:j) gösterimi** eleman sırası ilk sütun elemanlarından başlayarak i. eleman ile j. eleman arasındaki elemanları verir yani [A(i) A(i+1) ... A(j)] şeklinde bir vektör oluşturur.
- **A(:,j) gösterimi** A matrisinin j. sütununu verir.
- **A(:,i:j) gösterimi** [A(:,i);A(:,i+1);...;A(:,j)] şeklinde i. sütundan j. sütuna kadar sütunlardan oluşan bir alt matris verir.
- **A(i,:) gösterimi** A matrisinin i. satırını verir.
- **A(i:j,:) gösterimi** [A(i,:);A(i+1,:);...;A(j,:)] şeklinde i. satırdan j. satıra kadar satırlardan oluşan bir alt matris verir.

Örnek olarak aşağıdaki 5x5 boyutlu yani 25 elemanlı Amatrisi üzerinde duralım:

```
» A= [ 4 2 3 -1 2; -4 1 -2 2 7; 2 0 3 9 5; 1 -7 3 5 7; 5 -1 0 -2 8]
A =
    4    2    3   -1    2
   -4    1   -2    2    7
    2    0    3    9    5
    1   -7    3    5    7
    5   -1    0   -2    8
```

» A(3) ans = 2 A matrisinin 3. elemanı 2'dir.	» A(18) ans = 9 A matrisinin 18. elemanı 9'dur.	» A(1,4) ans = -1 A matrisinin 1. satır 4. sütun kesişimindeki eleman -1'dir.	» A(1:7) ans = 4 -4 2 1 5 2 1 A matrisinin 1. elemandan 7. elemana kadar olan elemanları verir.

» A(2,:) ans = -4 1 -2 2 7 A matrisinin 2. satırını verir.	» A(3:5,:) ans = 2 0 3 9 5 1 -7 3 5 7 5 -1 0 -2 8 A matrisinin 3. 4. ve 5. satırlarından oluşan alt matrisi verir.	» A(:,4) A = -1 2 9 5 -2 Matrisin 4. sütununu verir.	» A(:,1:3) ans = 4 2 3 -4 1 -2 2 0 3 1 -7 3 5 -1 0 A matrisinin 1., 2. ve 3. sütunlarından oluşan alt matrisi verir.

PROGRAMLAMA

Daha önceki bölümlerde komutlarımızı interaktif modda yani komut satırında yazıyor ve işletiyorduk. Programlama mantığı çerçevesinde artık kodlarımızı bir dosyaya yazıp bu dosya üzerinden çalıştıracağız (run) yani işleteceğiz. Bu dosyalar, m dosyaları (m-files) olarak adlandırılır ve genel olarak *.m şeklinde gösterilir. m kelimesi Matlab'in ilk harfinden gelir. Tıpkı C++ dilindeki .c veya Microsoft Word programındaki .doc gibi.

M-dosyalarının iki türü vardır:

- Düzyazı (script) m dosyaları
- Fonksiyon (function) m dosyaları

Fonksiyonların genel yazım formatı

function [out1,out2,... ,outN] = fonksiyon_adı (in1,in2,...,inM)

şeklindedir. Bu format yani fonksiyon tanım satırı m-dosyasının ilk satırında olmalıdır. Fonksiyon *fonksiyon_adı* ile çağrılır ve *fonksiyon_adı.m* olarak saklanır. (in1,in2,...,inM) giriş argümanlarını, (out1,out2,... ,outN) ise çıkış argümanlarını gösterir. M ve N ıfır veya herhangi bir pozitif tamsayı olabilir. Örnek olarak function [Anapara,faiz] = kredi(P,oran,süre) veya function P = fm_mod(V,T) verilebilir.

M-Dosyalarında Değişken Kullanımı

- Bir değişkeni atamadan önce atamanın sağ tarafında bir değeri olmalıdır.
- Bir değişkenin üzerine başka bir değer atarsanız eski değişkenin değeri atanan yeni değişken değeri olur.
- Değişken adları bir harfle başlamalı, sonra herhangi bir harf rakam ve altçizginin kombinasyonu gelebilir. Matlab büyü k küçük harfleri ayırd eder yani A ile a ayrı değişkenlerdir.
- Değişken uzunluğu herhangi bir uzunlukta olabilir ancak Matlab değişken adının ilk 63 karaterini dikkate alır. Diğer bir deyişle maksimum değişken adı uzunluğu 63 karakter olmalıdır.

- Bir değişkene vereceğiniz adın daha once kullanılıp kullanılmadığını kontrol etmek için isvarname fonksiyonu kullanılır. Eğer verdiğiniz ad geçerli ise 1, değilse 0 değerini gösterir.

BİR MATLAB PROGRAMIN ANATOMİSİ

Kullanıcıdan belirli bir formatta bir A matrisini girmesini ve çıkışta girilen bu A matrisinin transpozesini (devriğini) gösteren ve her bir adımı ve aşamayı açıklayan bir program yazınız.

1. Adım: Matlab'i çalıştırınız. Sonra Matlab ekranında **File*New*M file** komutuyla veya komut satırından **edit** komutunu yazıp Enter'a basarak etkileşimli m dosyası yazmak için **Editor/Debugger** ekranına geçiniz. Tekrar belirtmeliyim ki herhangi bir metin (text) editöründe de (örneğin Windows'un Notepad-Not Defteri metin editörünü veya Word programını da kullanabilirsiniz) Matlab m dosya kodunuzu yazabilirsiniz. Ancak progrmaın çeşitli aşamalarını test etmek ve hataları anında düzeltip yeni sonuçları görmek için Matlab'ın kendi editörünü kullanmak daha etkin ve verimli bir yoldur.

2. Adım: Program kodunu yazalım.

```
% ***** Bu program bir matrisin transpozesini bulur. *****

% Kullanici bir A matrisini girer ve sonuçta bu matrisin
% transpozesi gösterilir.

disp('Asagidaki formatta istediginiz bir kare matrisi giriniz.')
disp(' ')
disp(' [x x x; x x x; ... ] Burada x numerik bir sayi olmalidir.' )
disp(' ')

A = input('A matrisini giriniz: ')
B = zeros(size(A)); % B matrisinin her elemanini
                                % A ile ayni boyutta sifira ayarlanır.
M = size(A,1); % A'nin satir sayisini bulur
N = size(A,2); % A'nin sütun sayisini bulur

% Matrisin transpoze alma islemi
if N ~= M
  disp('Satir sayisi sütun sayisina esit degildir. Baska bir matris giriniz')
  break
end

for i = 1:M
  for j = 1:N
    B(j,i) = A(i,j);
  end
end

A_transpoze=B

% Display the A matrix
disp('A matrisi: ')
A
disp(' ')

% Display the B matrix
disp('A matrisin transpozesi: ')
A_transpoze
disp(' ')
```

3. Adım: Kodu kaydediniz. Yazdığınız Matlab kodunu Ctrl+S veya **File*Save** komutunu ile gelen pencerede bir ad altında kaydediniz. Örneğin burada mtranspoze.m şeklinde kaydettik. (Eğer başka br metin editörünü kullanıyorsanız dosya uzantısının mutlaka .m şeklinde olmasına dikkat ediniz. Oysa Matlab editöründe .m yazmaya gerek yoktur zira bu uzantı otomatik olarak verilir)

4. Adım: Programı çalıştırınız. Yazmış yani kodlamış ve m dosyası olarak kaydettiğiniz dosyanın Matlab'in şu anki geçerli yolu (path) ile aynı yani şu an aktif olan klasörde olmalıdır. M-dosya editörü Matlab 6.0 versiyonu için yazılan tüm m dosyalarını varsayılan olarak **work** adlı klasör altında kaydeder ve Matlab'i ilk açtığınızda work klasörü aktif yani geçerli klasör olarak ekrana gelir. Bu nedenle kolaylık olsun diye ilk başlarda m dosyalarınızı **work** klasöründe varsayılan kaydedebilirsiniz. İleriki aşamalarda bir konu başlığı altında ayrı bir klasör açıp m dosyalarınızı buraya kaydetmeniz ve yine bu klasör altından çalıştırmanız gerekecektir. Şu an dosyamız **work** klasörü altında ve Matlab geçerli çalışma klasörü **work** klasörü olduğundan komut satırına **mtranspoze** yazıp programı çalıştıralım.

```
» mtranspoze
Asagidaki formatta istediginiz bir kare matrisi giriniz.
 [x x x; x x x; ... ] Burada x numerik bir sayi olmalidir.
```

A matrisini giriniz:
ekranına kare matris olmak üzere istediğiniz bir A matrisini giriniz.
A matrisini giriniz: [1 -4;3 5]
A matrisi:
A =
1 -4
3 5

A matrisin transpozesi:
A_transpoze =
1 3
-4 5

İŞLEÇLER

Matlab'de en temel olarak üç tür işleç vardır:

ARİTMETİK İŞLEÇLER

İşleç	Açıklama		
+	Toplama	^	Üs alma
-	Çıkartma	'	Transpoze
.	Ondalık noktası	.*	Dizi Çarpma(eleman-eleman)
=	Atama	./	Dizi sağa bölme (eleman-eleman)
*	Çarpma	.\	Dizi sola bölme (eleman-eleman)
/	Sağa Bölme	.^	Dizi üs alma (eleman-eleman)
\	Sola Bölme	.'	Dizi transpoze

İLİŞKİSEL İŞLEÇLER (RELATIONAL OPERATORS)

Matlab, diziler arasında karşılaştırma yapmak için 6 tane ilişkisel işlece sahiptir. Önemli bir nokta = ve == işleçleridir. = işleci, değişken atama (assigment) ve yerleştirme (replacement) işlevi görür oysa == işleci, matematiksel olarak eşittir anlamındadır.

İlişkisel İşleçler

İlişkisel işleç sembolü	Anlamı	Yardım (help)
<	Küçüktür (Less than)	lt
<=	Küçük eşittir (Less than or equal)	le
>	Büyüktür (Greater than)	gt
>=	Büyük eşittir (Greater than or equal)	g
==	Eşittir (Equal)	eq
~=	Eşit değildir (Not equal)	ne

MANTIKSAL İŞLEÇLER (LOGICAL OPERATORS)

Mantıksal işleçler

İşleç sembolü	Adı	Tanımı
~	NOT (Değil)	P bir dizi ise ~P, P ile aynı boyutlu, yeni bir dizi üretir. Bu yeni dizi P sıfır değilse 0'lardan oluşur; P sıfır ise 1'lerden oluşur.
&	AND (Ve)	P ve Q birer dizi ise P&Q, P ve Q ile aynı boyutlu, yeni bir dizi üretir. Bu yeni dizi P ve Q herikisi de sıfır olmayan elemanlara sahip ise 1'lerden oluşur; P veya Q sıfır ise 0'lardan oluşur.
|	OR (Veya)	P ve Q birer dizi ise P | Q, P ve Q ile aynı boyutlu, yeni bir dizi üretir. Bu yeni dizi P veya Q'da en az bir eleman sıfır değilse 1'lerden oluşur; P ve Q sıfır ise 0'lardan oluşur.

find Fonksiyonu

find fonksiyonu çok yararlı bir işleve sahiptir. Verilen mantıksal koşula göre yani sınamaya göre (koşulu ve sınamayı sağlayan) istenen -sıfırdan farklı olan- verilerin ve elemanların değerlerini değil indislerini (indices) elde etmenizi sağlar. find(a>50) komutu a'nın 50 den büyük olduğu a indislerini belirler.

ŞART DEYİMLERİ (CONDITIONAL STATEMENTS)

if DEYİMİ

if (eğer) deyimin genel formatı:

if mantıksal ifade
deyim

```
end
```

şeklindedir. Eğer **mantıksal ifade** doğru ise **deyim** de belirtilen işlem yapılır ve **end** ile işlem sona erdirilir.

```
a=input('Bir a degeri giriniz: ');
if a < 50
   s = 5*a;
end
s
```

else DEYİMİ

else (başka) deyiminin genel formatı:

```
if mantıksal ifade
     deyim takımı-1
else
     deyim takımı-2
end
```

şeklindedir. Eğer **mantıksal ifade** doğru ise **deyim takımı-1**'de belirtilen, yanlış ise yani değilse **deyim takımı-2**'de belirtilen işlem yapılır ve **end** ile işlem sona erdirilir. **else deyim takımı-2** şeklindeki yazılım da doğrudur ancak **else** den sonra en az bir boşluk bırakılmalıdır.

```
a=input('Bir a degeri giriniz: ');
if a < 50
          s = 5*a;
else
          s = 2*a;
end
s
```

elseif DEYİMİ

elseif (eğerbaşka) deyiminin genel formatı

```
if mantıksal ifade-1
     deyim takımı-1
elseif mantıksal ifade-2
     deyim takımı-2
else
     deyim takımı-3
end
```

şeklindedir. Eğer **mantıksal ifade-1** doğru ise **deyim takımı-1**'de belirtilen işlem, daha sonra başka bir **mantıksal ifade-2** verilir ve eğer bu ikinci şart doğru ise **deyim takımı-2**'de belirtilen işlem yanlış ise **deyim takımı-3**'de belirtilen işlem yapılır ve **end** ile işlem sona erdirilir.

```
a=input('Bir a degeri giriniz:  ');
if a < 50
      s = 5*a;
elseif a < 100
   s = 2*a;
else
   s = a/2;
end
s
```

for DÖNGÜSÜ

Bir **for** döngünün genel formatı:

```
for döngüdeğişkeni=ifade
                deyimler
end
```

şeklindedir. Örneğin ifade döngüdeğişkeni=m:j:n şeklinde ise burada m:j:n , j artım miktarlı ilk değeri m son değeri n olan bir sayı aralığıdır. Her **for** döngüsü **end** ile kapatılmalıdır.

```
for i = 1:5
```

```
   x(i)= i^2 ;
end
x

» orfor
x =
    1    4    9   16   25   11
```

while DÖNGÜSÜ

Bir while (süresince veya iken) döngüsünün genel formatı

```
while ifade
          deyimler
end
```

şeklindedir. Her **while** döngüsü de **if** döngüsü gibi **end** ile kapatılmalıdır.

```
x=2;
while x<50
  disp(x);
  x=x^3-x^2;
end
x
```

komut satırına orwhile dosya adını yazıp Enter'a basalım.

```
» orwhile
     2
     4
    48
x =
      108288
```

continue **ve** break **Yapısı**

continue ifadesi **for** ve **while** döngülerinde verilen şarta göre şart sağlandığında işlem bir sonraki iterasyona geçer, **break** ise işlemi sona erdirir. Eğer içiçe for veya while döngüsünde kullanılırsa **continue** ifadesinin geçtiği ve **end** ile sonlandırılan işlem atlanıp bir sonraki içiçe döngüye geçilir. Aşağıdaki örnekte *devam.m* kodu çıktısında görüleceği gibi negatif değerli sayılar işleme konmamıştır.

```
t=[-5,5,-4,4,-3,3-2,2,-1,1];
for k=1:length(t)
  if t(k)<0
    continue
  end
  t(k)= log10(t(k))
end
t
```

Test:

```
>> devam
t =
  -5.0000    0.6990   -4.0000    0.6021   -3.0000         0    0.3010   -1.0000         0
```

break ifadesi **for** ve **while** döngülerinde verilen şart veya durum sağlandığında işlem burada kesilir yani program işletmeyi durdurur. Eğer içiçe for veya while döngüsünde kullanılırsa sadece **break** ifadesinin geçtiği döngü sona erdirilir ve varsa bir sonraki içiçe döngüye geçilir. Aşağıdaki örnekte *kes.m* kodu çıktısından görüleceği gibi m=-25 yani ilk negatif değerini aldığında işlem kesilmiştir.

```
for t=1:10
  m=100-t^3
  if m<0
    break
  end
  n=sqrt(m)
end
m
```

Test: >> kes

```
m =
   99
```

```
n =
    9.9499
....
m =
   -25
```

switch-case Yapısı

switch-case (değiştir-durum) yapısı, yukarıda gördüğümüz **if**, **else** ve **elseif** yapılarının kullanımına bir alternatif getirir. Aslında **switch-case** ile yapılan herşey **if** yapılarıyla da yapılır ama **switch-case** ile yazılan programlar daha okunabilir bir özelliğe sahiptir. Genel formatı

switch giriş ifadesi (skaler veya karakter dizgesi)
case ifadesi
    deyim grubu-1
case ifadesi
    deyim grubu-2
.
.
otherwise
    deyim grubu-n
end

şeklindedir. Giriş ifadesi**,** her bir **case** değeri ile karşılaştırılır. Her bir **case** değeri ayrı bir satırda olmalıdır.

Örnek: sindeg.m adlı bir dosyada, girilen bir açı değerinin hangi bölgede olduğu ve sinüs değerinin ne olduğunu bulmak için aşağıdaki kodu giriniz.

```
angle=input('Bir aci giriniz: ');
switch fix(angle/90)
case 0
  disp('I. Bolge ve pozitif')
case 1
  disp('II. Bolge ve pozitif')
case 2
  disp('III. Bolge ve negatif')
case 3
  disp('IV. Bolge ve negatif')
otherwise
  disp('0 ile 360 arasinda bir deger giriniz')
end
```

sindeg.m dosyasını komut satırından çalıştırınız ve giriş değeri olarak 135 girdiğimizi varsayalım.

```
» sindeg
Bir aci giriniz: 135
II. Bolge ve pozitif
```

FONKSİYON FONKSİYONLARI

Fonksiyon fonksiyonları, diğer fonksiyonları giriş argümanı olarak kabul eden fonksiyonlardır. Tüm diğer ayrıntılı bilgiler için » **help funfun** komutunu kullanınız.

Fonksiyon Adı	**Tanımı**
fzero	Tek değişkenli bir fonksiyonun çözümünü bulur.
fmin	Tek değişkenli bir fonksiyonun yerel minimumlarını bulur.
fplot	Dizge ile tanımlanmış fonksiyonun grafiğini çizer.
ezplot	Matematiksel ifadelerin grafiklerinin kolaylaştırılmış çizimini yapar.
quad	Bir fonksiyonunun nümerik olarak integral değerini bulur.
ode23	Düşük dereceli diferansiyel denklem çözümü yapar.
feval	Bir fonksiyonun verilen bir değerini bulur. Dizge ile tanımlanmış fonksiyonu işletir
inline	INLINE fonksiyon nesnesini yapılandırır.

input **Fonksiyonu**

input fonksiyonu kullanıcıdan bir veri girişi istendiğinde kullanılır. Genel sözdizimi yani formatı:

```
kullanıcı_girişi = input('prompt')
kullanıcı_girişi = input('prompt','s')
```

şeklindedir. s takısı giriş olarak bir karakter dizge girişi yani genelde metinsel bir ifade gerektiğinde kullanılır. prompt, geçerli çalışma ortamında değişkenler kullanarak işleme konacak herhengi bir ifade olabilir. input('*prompt*') kullanıcının klavyeden yapacağı bir giriş için bekler ve girilen değeri kullanıcı_girişine döndürür. input('*prompt*','s') ise bir değişken adı veya sayısal değerden ziyade girilen dizgeyi bir metin değişkeni olarak döndürür.

```
» x=input('Herhangi bir sayi giriniz: ');
Herhangi bir sayi giriniz:
```

M-DOSYALARINDA HATA GÖSTERIMI

disp **Fonksiyonu**

disp fonksiyonu, program işleyişindeki veya veri girişi hata uygulamalarında genelde metin dizgesel olarak görev yapar.

```
sayi=input('* 0 ile 1 arasinda bir sayi giriniz * : ');
if sayi > 1
   disp('Hata! Sayi 0 ile 1 arasinda olmalidir.')
else
   disp(' ')
   disp('Rasyonel sayi karsiligi= ')
   disp(rats(sayi))
end
» rasyonelkar
* 0 ile 1 arasinda bir sayi giriniz * : 4
Hata! Sayi 0 ile 1 arasinda olmalidir.
```

error **Fonksiyonu**

error fonksiyonu, disp fonksiyonunun tek farkı hata mesajından önce bir Error satırı içermesidir.

```
sayi=input('* 0 ile 1 arasinda bir sayi giriniz * : ');
if sayi > 1
   error('Dikkat! Sayi 0 ile 1 arasinda olmalidir.')
else
   disp(' ')
   disp('Rasyonel sayi karsiligi= ')
   disp(rats(sayi))
end
» rasyonelkar
* 0 ile 1 arasinda bir sayi giriniz * : 4
??? Error using ==> rasyonelkar
Dikkat! Sayi 0 ile 1 arasinda olmalidir.
```

warning **Fonksiyonu**

warning fonksiyonunun, aslında çok geniş uygulama alanları vardır. Buradaki kullanımı sadece bir örnekleme olması içindir.

```
sayi=input('* 0 ile 1 arasinda bir sayi giriniz * : ');
if sayi > 1
   warning ('Dikkat! Sayi 0 ile 1 arasinda olmalidir.')
else
   disp(' ')
   disp('Rasyonel sayi karsiligi= ')
   disp(rats(sayi))
end

» rasyonelkar
* 0 ile 1 arasinda bir sayi giriniz * : 4
Warning: Dikkat! Sayi 0 ile 1 arasinda olmalidir.
> In E:\bin\rasyonelkar.m at line 5
```

KARAKTER DİZGE (STRING) İŞLEMLERİ

Karakter dizgeleri veya sadece dizge (string) , iki tek tırnak arasındaki ifade edilen gerçekte ASCII kod tablosunda sayısal kodlarla belirtilen ilk 127 karakterden oluşan dizilerdir (character array). Dizgenin uzunluğu, dizgedeki karakter sayısıdır. ASCII karakterlerinin 32 ile 127 arasında olanları yani 0:255 bölgesi tamsayı ve basılabilir karakterdedir.

```
» gir='Programa  hosgeldiniz';
» gir
gir =
Programa  hosgeldiniz
```

Buradaki gir değişkenindeki her bir harf ASCII kod tablosundaki sayısal bir koda sahiptir.

```
» kod=double(gir)
```

```
kod =
  Columns 1 through 12
    80   114   111   103   114    97   109    97    32    32   104   111
  Columns 13 through 21
   115   103   101   108   100   105   110   105   122
```

double komutuyla **gir** değişkenindeki her bir harfin kod karşılığını görebilirsiniz. Buna göre p: 80, r: 114 veya o: 111 kodlarına karşılık düşmektedir. **double** fonksiyonunu kullanarak karakter veri tipinden double veri tipine dönüştürülebilir.

Tam tersi ASCII karşılığı verilen bir dizinin karakter karşılığını bulmak için **char** komutu kullanılır.

```
» char(kod)
ans =
Programa  hosgeldiniz
```

Çok Boyutlu Dizge Gösterimleri

Birden fazla karakter dizgesini birarada göstermek için dizgeleri, vektörler gibi bir arada kullanılabiliriz.

```
» k = strcat('Programa  hosgeldiniz' ,'program kodu: ' , '120885')
k =
Programa  hosgeldiniz  program kodu: 120885

» k=strvcat('Programa  hosgeldiniz' ,'program kodu: ' , '120885')
k =
Programa  hosgeldiniz
program kodu:
120885
```

Sayı-Dizge Dönüşümü

num2str komutu, kayan nokta sayısını dizgeye dönüştürür; int2str komutu ise sadece bir tamsayıyı, dizgeye dönüştürür.
Örnek olarak

```
» x = 236
x =
    236
```

ifadesi bir tamsayı tanımlar. Oysa

```
» xs=int2str(x)
xs =
236
```

komutu, 236 tamsayısını 236 dizgesine dönüştürür.

Dizge-Sayı Dönüşümleri

Matlab'de dizgeleri sayıya dönüştürmek için iki temel komut vardır. str2num fonksiyonu, normalde ASCII karakterdeki dizgeyi sayıya dönüştürür; eval fonksiyonu ise nümerik formdan sayı içeren dizgeye dönüştürür. Her iki fonksiyon da temelde aynı işlevi görür.

```
» c=str2num('236')
c =
   236
```

komutuyla 236 dizgesi 236 sayısına dönüşmüştür.

PROGRAM ÇIKIŞI ve SONUÇ GÖSTERME İŞLEMLERİ

Bir programın sonucunu yani çıkış verileri yazdırmak veya ekranda belirli bir formatta görüntülemek için üç temel komutumuz vardır: format, disp ve fprintf komutları.

disp KOMUTU

Genel yazım formatı:

```
        disp(x)
```

şeklindedir. **disp** komutu bir dizi veya metni görüntüler. Dizi görüntülemede dizinin adı yazılmaz ve boş bir dizi görüntülenmez. x bir karakter dizgesi ise metin olarak görüntülenir. Örnek olarak bir a dizisi alaalım.

```
» a=[1 2 3 4 5]
a =
    1    2    3    4    5
» disp(a)
    1    2    3    4    5
```

disp(' ') komutu kendisinden önce ve sonra gelecek satır arasında bir satır boşluk sağlar. ' ' tırnakları arasındaki boşluk sayısının önemi yoktur. Ancak aynı komutu bir kere daha alt alta kulanır

sprintf KOMUTU

sprintf komutu, formatlı veriyi bir dizgeye yazmak için kulanılır. Genel yazım formatı

[s,errmsg] = sprintf(format,A,...)
şeklindedir. Belirlenmiş format dizgesinin kontrolu altında onu s dizge değişkenine döndürür. errmsg isteğe bağlı bir çıkış argümanıdır ve bir hata oluştuğunda veya boş bir matris girildiğinde bir hata mesajına döndürür. sprintf , çıkışı bir dosyaya yazma işlevi dışındaki fprintf komutunun kullanımı ile aynıdır. fprintf ya bir dosyaya ya da ekrana çıkış verir oysa sprintf, veriyi ve sonucu bir dizge değişkenine döndürür. format dizgesinin kullanımı fprintf komutunda anlatıldığı gibidir. Aşağıda bir kaç örnek verilmiştir.

Komut	**Sonuç**
sprintf('%0.5g', (1+sqrt(5))/2)	1.618
sprintf('%d', round(pi))	3
sprintf('%s', 'merhaba')	merhaba

fprintf KOMUTU

fprintf komutu, formatlı program çıkışı ekranda göstermek veya elde etmek ve bunu bir dosyaya yazdırmak için kulanılır. Genel yazım formatı

fprintf('format' ,A, ...)

şeklindedir. 'format' stringinde belirtilen formatta A dizisi veya ek dizi argümanlarının elemanlarını gösterir. Daha basit bir gösterim ile fprintf('format' ,liste, ...) burada liste virgülle ayrılan değişken adlarını listeler. Komuttaki format stringinin genel formatı:

%[-/+/0][sayı1.sayı2] Kod

şeklindedir. Burada % işareti sabittir ve mutlaka bulunmalıdır. Köşeli parantez içindeki kısımlar isteğe bağlıdır. Bayrak (flag) olarak adlandırılan [-/+/0] gösteriminde - çıkışın sola hizalı olacağını, + önünde sürekli + işaretinin olacağını ve 0 da sayı alanının boşluktan ziyade 0'la doldurulacağını gösterir. Sayı alanı ve kesinliği yani ondalığı olarak adlandırılan ikinci kısımdaki sayı2, ondalık göstergesi olan noktanın sağındaki rakam sayısını, sayı1 yazdırılacak rakamların minimum alan genişliğini gösterir. Kod, kontrol ve biçim kodlarını daha doğrusu dönüşüm karakterlerini içerir. %işaretinden sonra yazılacak ifadelerin arasında boşluk bırakılmamalıdır.

Kaçış (Kontrol) Kodları		**Dönüşüm (Biçim) Kodları**	
Karakter	**Tanımı**	**Belirteç**	**Tanımı**
\n	Yeni bir satıra başlar (return or Enter)	%e	Küçük harfle bilimsel notasyon
\r	Yeni satırın başı (linefeed) Carriage return	%E	Büyük harfle bilimsel notasyon
\b	Geriboşluk karakteri (backspace)	%f	Sabit nokta (ondalık) notasyon
\t	Yatay sekme karakteri (Tab)	%g	En kısa olmasına göre %e veya %f
\f	Formfeed karakteri	%G	%g ile aynı ancak E kullanır
''	Apostrof karakteri (iki tek tırnak)	%s	karakter dizgesi tanımlar
\\	Ters bölü karakteri (backslash)	%c	Tekil karakter
%%	Yüzde işareti karakteri	%d	İşaretli ondalık notasyon

```
» fprintf('Islem sonucu %6.2f bulunmustur. \n', 813);
Islem sonucu    813 bulunmustur.
» fprintf('Islem sonucu %6.4f bulunmustur. \n', 813);
Islem sonucu   0813 bulunmustur.
```

Ör: 1'den 5'e kadar sayıların karesini sayılar sol tarafta kareleri sağ tarafta olacak şekilde gösteriniz. m-dosya adı kareselgos.m olsun

```
disp(' ')
disp('sayi    karesi  ')
disp('====   ======== ')
for i = 1:5
  karesi = i^2;
  sayi=i;
  fprintf('%2.0f  \t %2.0f\n',sayi,karesi)
end
```

```
» kareselgos
sayi        karesi
====   ========
 1           1
 2           4
 3           9
 4          16
 5          25
```

Ör: Bir küpün bir kenarını girerek alanını ve hacmini veren kup.m adlı bir program yazınız. Rastgele değerlerle test ediniz.

```
disp('KUP Bilgileri')
a = input('Kupun bir kenarini giriniz: ');
b=6*a^2;
c=a^3;
disp(['Alani: ' num2str(b) 'm^2' ])
disp(['Hacmi: ' num2str(c) 'm^3' ])
```

```
» kup
KUP Bilgileri
Kupun bir kenarini giriniz: 4
Alani: 96m^2
Hacmi: 64m^3
```

tic, toc **fonksiyonu**

Bu fonksiyon bir işlemin ne kadar süre zaman aldığını ölçmek için kronometre gibi çalışır işlemin başında sanki kronometreye basılır ve işlem basında tekrar basılır ve aradaki zaman ölçülür. Genel yazım formatı

```
tic
   çeşitli ifadeler
toc
```

şeklindedir. tic kronometreyi başlatır. toc ise tic in kullanıldığı andan itibaren geçen süreyi yazdırır. sure adlı değişken ise geçen süreyi saniye olarak toc'a döndürür.

```
tic;
x=0:0.001:10;
y=(sin(x).*cos(x))./(sin(x)-cos(x));
toc;
```

kodunu bir sureolc.m dosyasına yazalım ve çalıştıralım.

```
elapsed_time =
   0.0500
```

Görüleceği gibi işlem toplam olarak 0.05 saniye sürdü. elapsed_time =geçen süre

PROGRAMI .EXE HALİNE GETIRMEK

Örnek olarak ikinci dereceden denklemin kokleirni bulan bir programı .exe yapalım. En önemli nokta program kodunun fonksiyon m-dosyası şeklinde olması ve Turkce karaketer kullanılmamasıdır.

```
f unction [x1,x2]=iddkokleri
disp('ax^2+bx+c=0 seklindeki 2. dereceden denklem cozumu')
disp(' ')

%Katsayilari girme
a=input ('a katsayisini giriniz: ');
b=input ('b katsayisini giriniz: ');
c=input ('c katsayisini giriniz: ');
disp(' ')

if a==0
  disp('Denklem 2. dereceden degildir sifirdan farkli bir a katsayisi giriniz')
end
clc
Katsayilar=[a b c]

% Dikstriminant bulma
D=b^2-4*a*c;

% Kokleri bulma
x1=( -b + sqrt(D) )/(2*a);
x2=( -b - sqrt(D) )/(2*a);

% Kokleri sinama
if D < 0
  disp('Denklemin iki kompleks koku vardir.')
  x1
  x2
else if D == 0
    disp('Denklemin iki esit koku vardir.')
    x1,x2
    else
      disp('Denklemin iki reel ayrik koku vardir.')
      x1
```

```
    x2
  end
end
```

Programı yazıp fonksiyon adıyla aynı adlı olarak kaydettikten sonra komut satırından

>> mcc -m iddkokleri

yazılır ve Lcc compiler seçeneği secilerek m dosyanız iddkokleri.exe haline getirilir.

İKİ BOYUTLU GRAFİKLER

Matlabda en basit grafik çizdirme komutu, **plot** komutudur**. Plot** komutu, iki boyutlu doğru (çizgi) grafiği çizdirir. Örneğin X ile Y, iki aynı boyutlu vektör ve X'deki sayılar x-ekseni (absis) üzerinde Y'deki sayılar y-ekseni (ordinat) üzerinde olsun. **Plot** komutu X in her noktası için karşılık gelen Y değerlerini çizdirir. Diğer bir deyişle, (X(1),Y(1)), (X(2),Y(2)), (X(3),Y(3)) vb noktalar çizdirilecek ve daha sonra da tüm bu noktalar birleştirilecektir. **Plot** komutunun nasıl bir işlem yaptığını bir örnek üzerinde görelim. Önce iki basit vektör oluşturalım:

» x_nok = [1 2 3 4 5];
» y_nok = [25 0 20 5 15];

Daha sonra bu iki vektörü çizdirmek için komut yoluna şu komutu yazalım:

» plot(x_nok, y_nok)

xlabel ('text')	Grafiğin x-eksenini adlandirir. Genelde text, data adı ve/veya birimi olur.
ylabel ('text')	Grafiğin y-eksenini adlandirir. Genelde text, data adı ve/veya birimi olur.
title ('text')	Grafiği adlandırır yani grafiğe başlık verir.

Bazen özellikle çizgi grafiklerde grafiğin daha anlaşılır olması için diğer bir deyişle okunabilirliliğini artırmak ve görünümünü daha belirgin yapmak için klavuz çizgileri (yani alt zemin ızgarası) eklemek gerekebilir. Bu işlem için komut satırına

» grid on (veya sadece **grid** yazılır)

yazılır. Bu durumdaki grafik Şekil-4'de görülmektedir. Eklenmiş klavuz çizgilerini kaldırmak için ise **» grid off** komutu kullanılır.

ÇİZGİ ve İŞARETLEME SEÇENEKLERİ

Çizimin görünümünü değiştirmek isterseniz Matlab'de bir çok çeşitli seçenekler vardır. Çizimin rengini, işaretleyici sembolu ve çizgi tipini kendiniz belirleyebilirsiniz. Bu işlemin genel komutu

plot(x,y,'s')

şeklindedir. Burada x ve y veri vektörlerinden sonra gelen üçüncü argüman olan kesme işaretleri arasında yer alan **s,** Tablo-I'deki üç sütundan (renk, işaretleyici sembolu, çizgi tipi) herhangi biri ya da hepsinin bir kombinazsyonu olabilir. Bu üçüncü argümanın kullanımı sadece isteğe bağlıdır. Ancak tek grafiğikte verilerin dağılımı daha iyi anlamak ve belirli bir peryotta olayın oluşumu kontrol etmek istediğinizde işaretleyiciler iyi bir seçenek olabilir. Ayrıca birden fazla grafiği aynı düzlemde göstermek istediğinizde de bazı **s** kombinasyonlarını kullanmak zorunlu olabilir.

Color (Renk)	Indicator
Blue (Mavi)	b
Green (Yesil)	g
Red (Kirmizi)	r
Cyan (Turkuaz)	c
Magenta (Mor)	m
Yellow (sari)	y
Black (Siyah)	k
White (Beyaz)	w

Marker symbol (Isaretleyici sembolu)	Indicator
Point (Nokta)	.
Plus (Arti)	+
Star (Yildiz)	*
Circle (Daire)	o
x-mark (x isareti)	x
Square (Kare)	s
Diamond (Elmas)	d
triangle (down) (Asagi bakan ucgen)	v
triangle (up) (Yukari bakan ucgen)	^
triangle (left) (Sola bakan ucgen)	<
triangle (right) (saga bakan ucgen)	>
Pentagram (Besgen)	p
Hexagram (Altigen)	h

Line style (Cizgi tipi)	Indicator
Solid (Duz cizgi)	-
Dashed (Kesikli cizgi)	--
Dotted (Noktali cizgi)	:
Dash-dot (Kesikli-niktalicizgi)	-.

ıutunu
ğıdaki

```
hold on          % Grafiği dondurur
plot(...)        % Yeni bir grafik ekler
.
hold on          % Grafiği dondurur
plot(...)        % Yeni bir grafik ekler
.
hold off         % Dondurulan grafiği (grafikleri) serbest bırakır
```

Grafik başlıkları, eksen adlandırmaları ve göstergeler için **hold on** komutunu kullanmaya gerek yoktur. Hold on komutu bir önceki grafiği dondurur ve aynı düzlemde diğer grafiklerin çizlmesine izin verir ve **plot** komutu ile çizilecek olan ikinci grafik bir önceki geçerli eksenleri kullanır. Ancak eğer çizilecek ikinci grafiğin eksen scalası daha geniş ise ona göre otomatik olarak ayarlama yapar.

GRAFİĞE GÖSTERGE EKLEME (LEGEND KOMUTU) İŞLEMLERİ

Gösterge eklemek için kullanılan komut **legend** komutudur. Genel format dizimi:
legend(string1,string2,string3, ... , Pos)
şeklindedir.

legend(string1, string2, string3, ...) komut dizimi gösterge kutusunu, varsayılan olarak grafiğin sağ üst köşesine yerleştirir.
Genel komut dizimindeki **Pos** (Position demektir) ifadesi, gösterge kutusunu belirlenmiş bir konuma yerleştirir. **Pos** ifadesinin alacağı değerler şunlardır:
0 = Otomatik "en iyi" yer (Bu durumda gösterge kutusu verileri kapatmayacak olası en iyi yere yerleştirilir)
1 = Sağ üst köşe (varsayılan değer)
2 = Sol üst köşe
3 = Sol alt köşe
4 = Sol sağ köşe
-1 = Grafiğin sağına yerleştirir
Bunların dışında gösterge kutusunun üzerinde iken fare işaretcisinin sol tuşuna basarak ki bu durumda dört yönlü bir ok çıkar- elle istediğiniz yere taşıyabilir. Ayrıca gösterge kutusunu çift tıklayarak etiket düzenlemesi de yapabilirsiniz. **legend off:** komutu ise gösterge kutusunu yerleştirilen yerden kaldırır.

Örnek:

```
t = 0:pi/100:2*pi;
y1 = sin(t);
y2 = sin(t-0.25);
y3 = sin(t+0.25);
plot(t,y1,t,y2,t,y3)
xlabel('t');
title('Ötelenmis Sinüs Fonksiyonlari');
legend('sin(t)','sin(t-0.25)','sin(t+0.25)',0)
```

Aynı Düzlemde Birden Fazla Bağımsız Grafik Çizdirmek

Aynı düzlem üzerinde ve aynı eksen takımını kullanarak tek bir grafik penceresinde birden fazla ilişkiyi (grafiği) grup halinde üzerinde göstermek için **subplot** komutu kullanılır. Genel formati:
subplot(*m,n,p*)
seklindedir. Burada m satır, n sütun sayısını gösterir ve m*n tane matris düzeninde grafik çizilebilir. Bu komut dikdörtgen şeklinde m*n grafik alanı oluşturur. p yani index 1 ile m*n arasında olmalıdır. p, satırda soldan sağa doğrui sütunda yukarıdan aşağı doğru belirleme yapar yani grafikler bu sıraya göre yerleştirilir.

Örnek:

```
x_deg=[-10:.05:10];
dogru=5.*x_deg;
parabol=x_deg.^2;
ustel=exp(x_deg);
mutlak_deger =abs(x_deg);
subplot(2,2,1);plot(x_deg, dogru);title('Dogru Grafigi');
subplot(2,2,2);plot(x_deg, parabol);title('Parabol Grafigi');
subplot(2,2,3);plot(x_deg, ustel);title('Üstel Grafigi');
subplot(2,2,4);plot(x_deg, mutlak_deger);title('Mutlak Deger Grafigi');
```

figure KOMUTU

figure(n) n=1,2,3,...n

şeklinde komut ile iki (veya daha cok) n tane ayrı grafiği her biri ayrı pencerede olmak üzere çizdirebilirsiniz. Ayrıca her bir grafik birbirinden bağımsız olduğundan her bir grafiğin x degerlerini vb özelliklerini de değiştirebilirsiniz.

Örnek:

```
>> figure(1)
>> x=-pi:pi/10:pi;
>> y1=cos(x);
>> plot(x,y1,'r+');
>> xlabel('x');
>> ylabel('y');
>> title('y =cosx Grafigi');
>> figure(2)
```

```
>> y2=sin(x);
>> plot(x,y2,'b--');
>> xlabel('x');
>> ylabel('y');
>> title('y =sinx Grafigi');
```

pause KOMUTU

pause komutu programınızı geçici olarak dondurmanızı sağlar. Devam etmek için klavyeden herhangi bir tuşa basabilirsiniz yada berlitilen sure sonunda otomatik olarak devam kendiliginden devam eder. **pause(n)** komutu n saniye kadar gecici durdurma yapar

Örnek:

```
>> x=-pi:pi/10:pi;
>> y1=cos(x);
>> plot(x,y1);
>> xlabel('x');ylabel('y');
>> title('y =cosx Grafigi');
>> pause(3)        %Sadece pause ile deneyiniz ve herhengi bir klavye tusuna basarak devam ediniz
>> y2=cos(2*x);
>> plot(x,y2);
>> xlabel('x');ylabel('y');
>> title('y =cos2x Grafigi');
```

axis ([xmin xmax ymin ymax])

axis komutu ile varolan grafik üzerinde hem x-ekseninin hem de y-ekseninin geçerli sınırlarını değiştirebilir ve istediğiniz bir eksen ölçeği belirleyebilirsiniz.

```
>> axis([0 10 –1 1]);
```

komutu x-eksenini 0 ile 10 arasında y-eksenini -1 ile 1 arasında ölçeklendirir.

axis equal: Eksenlerin çentik artımlarını her iki eksen üzerinde eşit olacak şekilde ayaralar.
axis square: Dikdörtgen bir grafik alanı yerine kare görünümlü bir grafik kutusu yapar.
axis normal: Orijinal eksen takımına döndürür.

GRAFİKLERE SEMBOL ve YUNAN KARAKTERLERİNİ EKLEMEK

Mühendislik ve bilimsel grafiklerde **xlabel**, **ylabel** ve **title** adlandırmalarında ya da grafik üzerinde **legend**, **text** ve **gtext** kullanımlarında hatta eksen centiklerinin adlandirmasina kadar $\alpha,\beta,\omega,\lambda$ gibi Yunan alfabesi (greek veya latin karakterler) harflerini veya km^2, $\leq$, ∞, $\rightarrow$ gibi sembolleri eklemek gerekebilir. Bunların Matlab kod sistemindeki kullanımları Tablo'da verilmiştir. Matlab de bu işlemler *text strings* (yazı dizgeleri) olarak adlandırılır.

Yunan Karakterleri ve Semboller

Krakter	Sembol	Krakter	Sembol	Krakter	Symbol
\alpha	α	\upsilon	υ	\sim	~
\beta	β	\phi	ϕ	\leq	≤
\gamma	γ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	♣
\epsilon	ε	\omega	ω	\diamondsuit	♦
\zeta	ζ	\Gamma	Γ	\heartsuit	♥
\eta	η	\Delta	Δ	\spadesuit	♠
\theta	θ	\Theta	Θ	\leftrightarrow	↔
\vartheta	ϑ	\Lambda	Λ	\leftarrow	←
\iota	ι	\Xi	Ξ	\uparrow	↑
\kappa	κ	\Pi	Π	\rightarrow	→
\lambda	λ	\Sigma	Σ	\downarrow	↓
\mu	μ	\Upsilon	Υ	\circ	°
\nu	ν	\Phi	Φ	\pm	±
\xi	ξ	\Psi	Ψ	\geq	≥
\pi	π	\Omega	Ω	\propto	∝

\rho	ρ	\forall	∀	\partial	∂
\sigma	σ	\exists	∃	\bullet	•
\varsigma	ς	\ni	∋	\div	÷
\tau	τ	\cong	≅	\neq	≠
\equiv	≡	\approx	≈	\aleph	ℵ
\Im	ℑ	\Re	ℜ	\wp	℘
\otimes	⊗	\oplus	⊕	\oslash	∅
\cap	∩	\cup	∪	\supseteq	⊇
\supset	⊃	\subseteq	⊆	\subset	⊂
\int	∫	\in	∈	\o	o
\rfloor	⌋	\lceil	⌈	\nabla	∇
\lfloor	⌊	\cdot	·	\ldots	...
\perp	⊥	\neg	¬	\prime	′
\wedge	∧	\times	x	\0	∅
\rceil	⌉	\surd	√	\mid	|
\vee	∨	\varpi	ϖ	\copyright	©
\langle	⟨	\rangle	⟩		

Tablo'da verilen tüm karakterler ayrıca aşağıdaki özellikler (ki Matlab dilinde stream modifier (özel değiştiriciler) olarak adlandırılır) ile beraber kullanılabilir:

- \bf{} - Koyu yazıtipi (bold font)
- \it{} - İtalik yazıtipi (italics font)
- \sl{} - yazıtipi (oblique font, nadiren kullanılır)
- \rm - Normal yazıtipine dönüş
- \füntname{fontname} – Kullanılacak yazıtipi (font) ailesinin adını bülirler
- \fontsize{fontsize} – FontUnits olarak yani punto olarak yazıtipi boyutunu belirler.
- _{...}- Parantez içindeki karakterler ya da yazı altindis olarak gösterilir.
- ^{...}- Parantez içindeki karakterler ya da yazı üstindis olarak gösterilir.

Tablo'da verilen yunan karakterlerini ve sembollerini üç şekilde ifade edebiliriz[1]:
Eşitlik : $\beta=3$ için '\beta=3' (equal)
Üst indis : β^3 için '\beta^3' (subscript)
Alt indis : β_3 için '\beta_3' (superscript)
Ayrıca bu gösterim mantığı m^3 veya K_{12} gibi normal yazı işlemleri için de geçerlidir.

Alt indis veya üst indislerde birden fazla karakter kullanılacak ise bu durumda { } şeklindeki parantez kullanılmalıdır.
x ekseninde "-χ_o^1 ışıldamasına göre yarılanma değerleri " görünmesini istersek

```
>> xlabel ('-\gamma{_0^1} ışıldamasına göre yarılanma değerleri ')
```

komutu kullanılır.

y ekseninde "Hız katsayısı: $K_o=10^{-12}$ " görünmesini istersek

```
>> ylabel ('Hız katsayısı: K_o=10^{-12}')
```

komutu kullanılır.

LOGARİTMİK GRAFİKLER

Komut	x-eksen Ölçeği	y-eksen Ölçeği	Grafik tipi
loglog (x,y)	logarithmic	logarithmic	log(x) karşı log(y)
semilogy(x,y)	linear	logarithmic	log(x) karşı y
semilogx(x,y)	logarithmic	linear	x karşı log(y)

Logaritmik grafikler genel olarak çok buyuk veya çok degerli verileri anlasilabilir bir olcege uyarlar veya lineer olmayan verilere uygun bir çizim zemini sağlar.

1

Örnek:

```
n = [ 3 5 9 17 33 65 ]';
sn = [ 2.57e-1   6.46e-2   1.51e-2    3.96e-3    9.78e-4    2.45e-4 ]' ;
loglog( n, sn, 'x')      % log ölçeği: Logaritmik koordinatlar
xlabel('n (dk)');ylabel('s_n tanecik sayisi ');title('Bir maddenin tanecik ayrisim egrisi ')
```

İKİ Y-EKSENLİ GRAFİK ÇİZMEK

Bazi durumlarda verileri y-ekseninin hem sol hem de sağ tarafında olceklemek gerekebilir. Bu işlem için **plotyy** komutu kullanılır. Genel formati:

```
plotyy(x1,y1,x2,y2)
```

şeklinde olan bu komut ile X1'e karşı Y1 grafiğini y-ekseninin sol tarafına, X2'ye karşı Y2 grafiğini y-ekseninin sağ tarafına çizilir.

ezplot ve fplot komutu kullanımı

ezplot ve fplot komutlarının genel formatı

ezplot ('fun', xmin, xmax, ymin, ymax) veya **ezplot('fun', [xmin,xmax,ymin,ymax])**

ezplot ('fun', xmin, xmax, ymin, ymax) veya **fplot ('fun', [xmin,xmax,ymin,ymax])**

şeklindedir. Burada **fun**, çizdirilecek fonksiyonu yani daha doğrusu karakter dizgesini temsil eder. **xmin** ve **xmax,** fonksiyonun x ekseninde; **ymin** ve **ymax,** fonksiyonun y ekseninde çizim aralığını belirler. Yukarıda komut formlarına göre ezplot('tan(sin(x))-sin(tan(x))') veya ezplot tan(sin(x))-sin(tan(x)) komutlarının her ikiside doğrudur. Ancak fonksiyonun önünde sabit bir değer varsa veya x değerleri için özel bir sınır belirtilecek ise bu durumda 2. form kullanılmalıdır.

Örnek: y=sin(x)/(1+x^2) fonksiyonunun grafiğini çizelim.

```
>>ezplot( 'sin(x)/(1+x^2)' )          veya          >>ezplot sin(x)/(1+x^2)
>> fplot( 'sin(x)/(1+x^2)' , [0 5])
```

diğer bir örnek olarak

```
>> fplot('[ cos(x), 1-x^2/2, 1-x^2/2+x^4/24]', [-pi,pi])
```