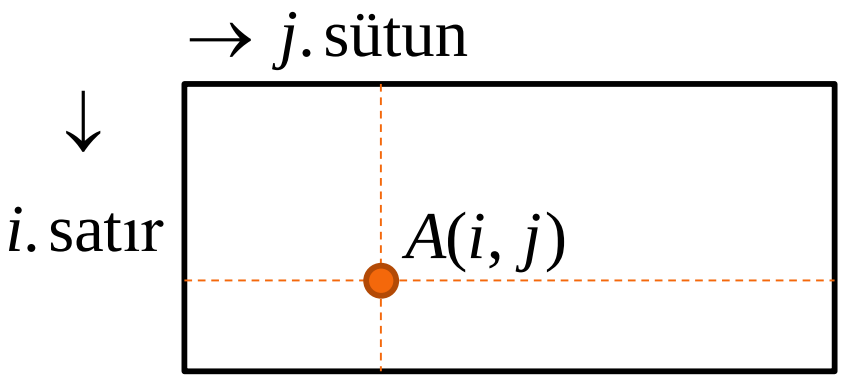
5. diziler ve matrisler

diziler ve matrisler MATLAB'da en önemli veri nesneleridir.

Geçen hafta bir boyutlu dizileri yani, sütun ve satır vektörleri inceledik.

Şimdi iki boyutlu dizileri yani, matrisleri inceleyeceğiz.

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$



Command Window

```
>> A=[1 2 3; 2 0 4; 0 8 5]

A =

     1     2     3
     2     0     4
     0     8     5

>> size(A)        % [N, M] = size(A), N x M matris

ans =

     3     3

fx >>
```

Matris elemanlarına ulaşım

Command Window

```
>> A(1,1)

ans =

     1
```

% Matrisin 11 elemanı

$$A=\begin{bmatrix}\boxed{A_{11}} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}$$

```
>> A(2,3)

ans =

     4
```

% Matrisin 23 elemanı

$$A=\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & \boxed{A_{23}}\\ A_{31} & A_{32} & A_{33}\end{bmatrix}$$

```
>> A(:,2)

ans =

     2
     0
     8
```

% Matrisin 2. sütunu

$$A=\begin{bmatrix}A_{11} & \boxed{A_{12}} & A_{13}\\ A_{21} & \boxed{A_{22}} & A_{23}\\ A_{31} & \boxed{A_{32}} & A_{33}\end{bmatrix}$$

```
>> A(3,:)

ans =

     0     8     5
```

% Matrisin 3. satırı

$$A=\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\\ \boxed{A_{31} & A_{32} & A_{33}}\end{bmatrix}$$

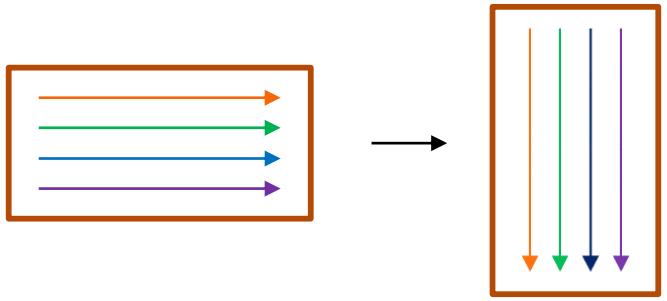
fx

Matrisin transpozesi:

```
Command Window
>> A=[1 2 3 4; 2 0 5 6; 0 8 7 9]

A =

     1     2     3     4
     2     0     5     6
     0     8     7     9

>> A'

ans =

     1     2     0
     2     0     8
     3     5     7
     4     6     9

fx >>
```

% 3 x 4

% 4 x 3



Transpoze operatörü

Temel matrisler üstüne daha fazla bilgi için:

>> help elmat

Temel matrisler ve matris işleme
Temel matrisler:

zeros	-	**tüm elemanları 0 olan matris**
ones	-	**tüm elemanları 1 olan matris**
eye	-	**birim matris**
repmat	-	**dizileri (array) kopyalar ve yanyana koyar**
linspace	-	**lineer aralıklı vektör**
logspace	-	**logaritmik aralıklı vektör**

...

6. Operatörler ve İfadeler

İşlem	Terim bazlı	Matris bazlı
Toplama	+	+
Çıkarma	-	-
Çarpma	.*	*
Bölme (sağ bölme)	./	/
Sol bölme	.\	\
Üs	.^	^

Kompleks eşleniksiz transpoze	.'
Kompleks eşlenikli transpoze	'

Bunlar lineer cebir kurallarına uymalıdır.

>> help /
>> help precedence

Command Window

```
>> a = [1 2 5];
>> b = [4 -5 1];
>> a+b

ans =

     5    -3     6

>> a.*b

ans =

     4   -10     5

>> a./b

ans =

    0.2500   -0.4000    5.0000

>> a.\b

ans =

    4.0000   -2.5000    0.2000
```

% Not: (a ./ b) .* (a .\ b) = [1, 1, 1]

Command Window

```
>> a = [2 3 4 5];
>> a.^2

ans =

     4     9    16    25

>> 2.^a

ans =

     4     8    16    32

>> a+10

ans =

    12    13    14    15

>>
```

% [2^2, 3^2, 4^2, 5^2]

% [2^2, 2^3, 2^4, 2^5]

Command Window

```
>> A = [1 2; 3 4]

A =

     1     2
     3     4

>> [A, A.^2; A^2, A*A]

ans =

     1     2     1     4
     3     4     9    16
     7    10     7    10
    15    22    15    22

>> B = 10.^A;
>> [B, log10(B)]

ans =

          10         100           1           2
        1000       10000           3           4

>>
```

% Not: A^2 = A*A

% $B = \begin{bmatrix} 10^1 & 10^2 \\ 10^3 & 10^4 \end{bmatrix}$

7. Fonksiyonlar

```
>> help elfun                % temel fonksiyonların dizisi
```

Bazı temel hazır fonksiyonlar:

sin(x),	cos(x),	tan(x),	cot(x)
asin(x),	acos(x),	atan(x),	acot(x)
sinh(x),	cosh(x),	tanh(x),	coth(x)
asinh(x),	acosh(x),	atanh(x),	acoth(x)

exp(x), log(x), log10(x), log2(x)

fix(x), floor(x), ceil(x), round(x)

sqrt(x), sign(x), abs(x)

sum(x), prod(x), cumsum(x), cumprod(x)

ve daha fazlası:

```
size(x),        length(x),       class(x)

sinc(x)         % sin(pi*x)/(pi*x)

max(x),       min(x),      sort(x)

mean(x),      std(x),          % istatistik
median(x),  mode(x)

rand, randn,              % rastgele sayı üreticisi
randi,  rng               % rng ile başlar

filter,   conv,   fft     % DSP fonksiyonları

clock,  date

factorial(n),   nchoosek(n,k)     % ayrık matematik
```

Çoğu fonksiyon skaler veya dizi ve matris girdilerini kabul eder ve dizinin her öğesinde çalışır.

$$x = [x_1, x_2, x_3, \ldots]$$

$$f(x) = [f(x_1), f(x_2), f(x_3), \ldots]$$

Command Window

```
>> x = [0, pi/4, pi/3, pi/2, pi];
>> sin(x)

ans =

         0    0.7071    0.8660    1.0000    0.0000

>> sin(sym(x))

ans =

[ 0, 2^(1/2)/2, 3^(1/2)/2, 1, 0]

fx >>
```

% açık çözümü görmek için
% sembolik araç çubuğunu kullanır

Command Window

```
>> x = [2.1, 2.8, -3.1, -3.5, 4.5];
>> y = exp(x)

y =

    8.1662   16.4446    0.0450    0.0302   90.0171

>> z = log(y)

z =

    2.1000    2.8000   -3.1000   -3.5000    4.5000

>> [fix(x); floor(x); ceil(x); round(x)]

ans =

     2     2    -3    -3     4
     2     2    -4    -4     4
     3     3    -3    -3     5
     2     3    -3    -4     5

fx >>
```

Örnek: sum(x) fonksiyonunu kullanarak aşağıdaki geometrik seri eşitliğini doğrulayın.

$$\frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^3}+\ldots+\frac{1}{2^N}=1-\frac{1}{2^N}$$

$$\sum_{n=1}^{N}\frac{1}{2^N}=1-\frac{1}{2^N}$$

toplama notasyonu

Command Window

```
>> format long;
>> N=8; n = 1:N;
>> sum(1./2.^n)

ans =

   0.996093750000000

>> 1-2^(-N)

ans =

   0.996093750000000
```

% n = [1, 2, ..., 8]
% [1/2^1, 1/2^2, ..., 1/2^8]
% ./ ve .^ işlemlerine dikkat

y = cumsum(x) – x'in elemanlarının kümülatif toplamı

$$y(1) = x(1)$$

$$y(2) = x(1) + x(2)$$

$$y(3) = x(1) + x(2) + x(3)$$

...

$$y(n) = \sum_{i=1}^{n} x(i) = x(1) + x(2) + \ldots + x(n)$$

x = [y(1), diff(y)] % ters işlem

cumsum – Örnek 1

```
>> format long;
>> N=8; n = 1:N;                    % n bir satır vektördür
>> y = cumsum(1./2.^n);
>> z = 1-1./2.^n;                   % y, z denk olmalı
>> fprintf('%d    %10.8f    %10.8f\n',[n;y;z]);
1    0.50000000    0.50000000
2    0.75000000    0.75000000
3    0.87500000    0.87500000
4    0.93750000    0.93750000
5    0.96875000    0.96875000
6    0.98437500    0.98437500
7    0.99218750    0.99218750
8    0.99609375    0.99609375
fx >>
```

fprintf 3x8'lik [n; y; z] matrisi üstünde sütun bazlı işlem yapar yani,

$$\begin{bmatrix} n_1 & n_2 & n_3 & \dots \\ y_1 & y_2 & y_3 & \dots \\ z_1 & z_2 & z_3 & \dots \end{bmatrix}$$

cumsum – Örnek 2

% aaron.dat

Yıl	ti	H	Yıl	ti	H
1954	1	13	1964	11	24
1955	2	27	1965	12	32
1956	3	26	1966	13	44
1957	4	44	1967	14	39
1958	5	30	1968	15	29
1959	6	39	1969	16	44
1960	7	40	1970	17	38
1961	8	34	1971	18	47
1962	9	45	1972	19	34
1963	10	44	1973	20	40

```
A = load('aaron.dat');

ti = A(:,2);
H = A(:,3);
yi = cumsum(H);

p = polyfit(ti,yi,1)

t = linspace(1,20,101);
y = polyval(p,t);
```


```
plot(t,y,'r-',ti,yi,'b.','markersize',18);
xlabel('\it{t}'); ylabel('toplam');
xlim([0 21]); xticks(0:2:21); yticks(0:100:800); grid on;
title('Hank Aaron sayı koşusu çıktıları')
legend('lineer regresyon', 'data', 'location', 'se');
print -dmeta aaron.wmf
```

Rastgele sayılar, min, max, mean, std, sort

```
Command Window
>> seed = 127; rng(seed);
>> x = randn(5,3)

x =

    0.0294   -1.0928    1.6686
   -1.5732   -0.1697   -0.4750
   -1.1899    0.5751   -0.7604
    1.8115    0.6548   -1.1189
    0.0426   -0.0969    0.1698

>> min(x), max(x), mean(x), std(x)

ans =

   -1.5732   -1.0928   -1.1189


ans =

    1.8115    0.6548    1.6686


ans =

   -0.1759   -0.0259   -0.1032


ans =

    1.3248    0.7051    1.0972
```

Üreticiyi başlatır

Sıfır ortalamalı, birim varyanslı, gauss rastgele sayılardan oluşan 5x3'lük matris

```
>> help rng
>> help rand
>> help randn
>> help randi
```

sütun bazlı hesaplama

MATLAB'da sütun baskındır.

Command Window

```
>> [m,i]=min(x), min(min(x))

m =

   -1.5732   -1.0928   -1.1189


i =

     2     1     4


ans =

   -1.5732

>> sort(x)

ans =

   -1.5732   -1.0928   -1.1189
   -1.1899   -0.1697   -0.7604
    0.0294   -0.0969   -0.4750
    0.0426    0.5751    0.1698
    1.8115    0.6548    1.6686
```

Her sütunun minimumu

Minimumların satır indexleri

Genel minimum

```
x =

    0.0294   -1.0928    1.6686
   -1.5732   -0.1697   -0.4750
   -1.1899    0.5751   -0.7604
    1.8115    0.6548   -1.1189
    0.0426   -0.0969    0.1698
```

i=2

min, max, sort matris girdilerinde sütun bazlı hesap yapar.

Her sütunu artan sırada sıralar.

sort(x, 'ascend')
sort(x, 'descend')

```
Command Window
>> [x2,i2] = sort(x(:,2), 'descend')

x2 =

    0.6548
    0.5751
   -0.0969
   -0.1697
   -1.0928


i2 =

     4
     3
     5
     2
     1

>> x_sort = x(i2,:)

x_sort =

    1.8115    0.6548   -1.1189
   -1.1899    0.5751   -0.7604
    0.0426   -0.0969    0.1698
   -1.5732   -0.1697   -0.4750
    0.0294   -1.0928    1.6686

fx >>
```

Yalnızca 2. sütunu azalan sırada sıralar.

x2 = sıralanmış sütun,
i2 = sıralama indeksi

% 2. sütuna göre x'i sıralar.

% sortrows kullanılabilir:
sortrows(x,-2)

Üç metodla kendi fonksiyonunuzu oluşturabilirsiniz:

1. **Fonksiyon tanımlama (function-handle), @(x)**
2. **inline**
3. **M-dosyası (M-file)**

örnek: $f(x) = e^{0.5x}\sin(5x)$

```
>> f = @(x) exp(-0.5*x).*sin(5*x);

>> g =  inline('exp(- 0.5*x).*sin(5*x)')

% yukarıdaki yazılan satırları düzenleyip kaydetmek
function y = h(x)
   y = exp(-0.5*x).*sin(5*x);
```

Parametreler fonksiyona nasıl dahil edilir?

örnek: $f(x) = e^{-ax}\sin(bx)$

% metod 1: Önce a, b tanımlanır, sonra da f:

```
a=0.5; b=5;
F = @(x) exp(-a*x).*sin(b*x);
```

% metod 2: Parametreler değişken olarak tanımlanır

```
F = @(x,a,b) exp(-a*x).*sin(b*x);
```

% Bu, f(x,a,b) fonksiyonunu tanımlar
% f(x, 0.5, 5), metod 1'de tanımlanan f(x) ile eşdeğerdir.

8. Grafik

MATLAB eğrilerin ve yüzeylerin çizilmesi ve görselleştirilmesi için geniş imkanlara sahiptir. Detaylarına sonra gireceğiz.

Fonksiyonların ve (x, y) çiftlerinin 2 boyutlu grafikleri

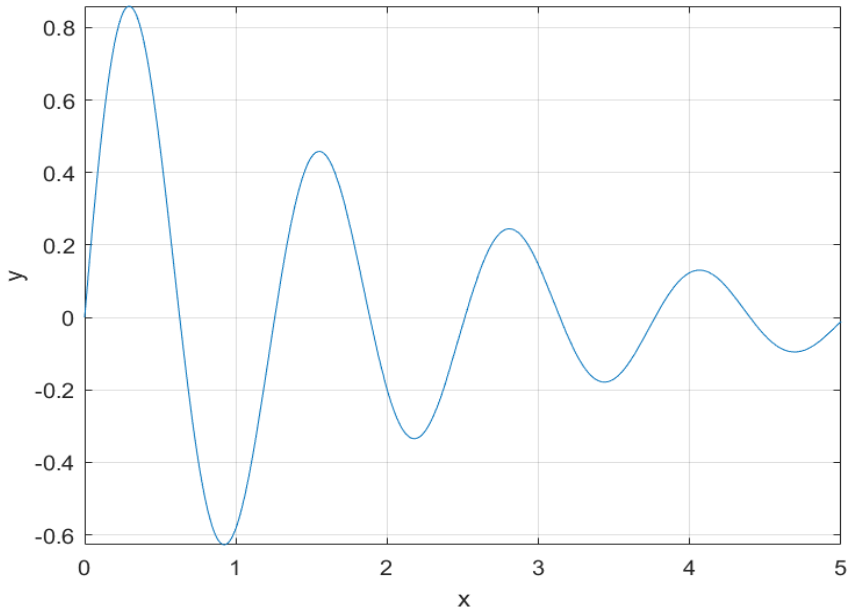
plot, fplot, ezplot

fonksiyonları ile yapılabilir.

```
>> help plot          % 2 boyutlu grafik
>> help fplot         % fonksiyon grafiği
>> help ezplot        % basit fonksiyon grafiği
```

Eğer bir f(x) fonksiyonu function-handle (@(x)) ya da inline ile tanımlanmışsa grafiği hızlı bir şekilde fplot, ezplot ile çizilebilir. Sadece tanım aralığını belirlemek gerekir. Örneğin:

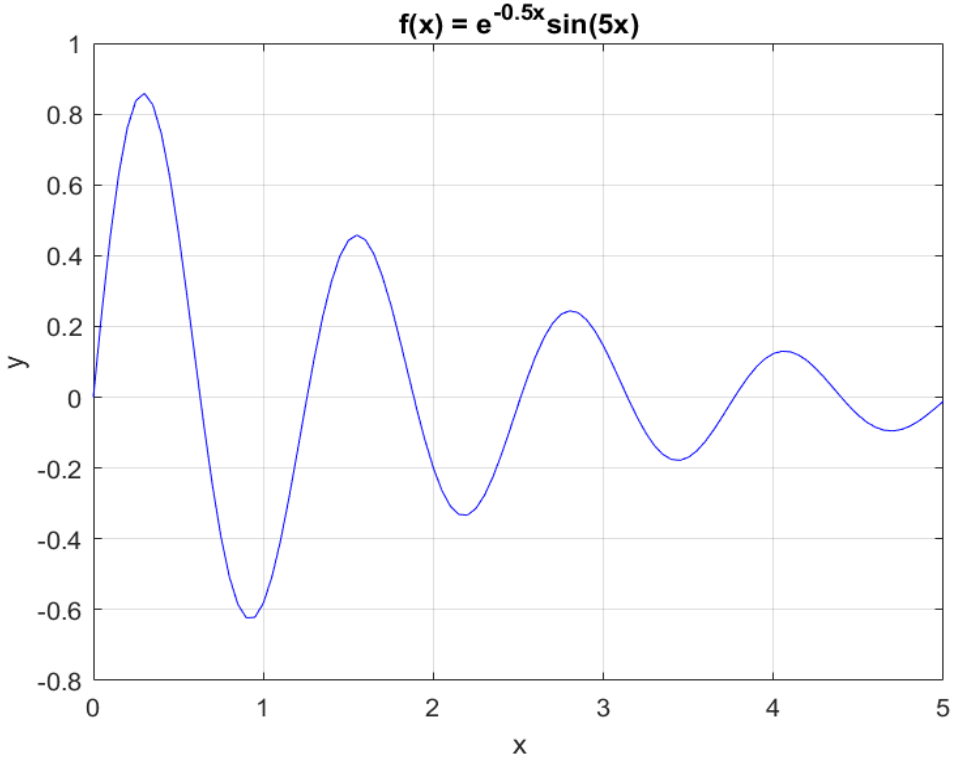
```
>> f = @(x) exp(-0.5*x).*sin(5*x);
>> fplot(f,[0,5]);                    % [0, 5] aralığı üzerinde çizer.
```



Figür penceresi (figure window) açılır ve grafik üstünde düzenlemeye izin verilir, örneğin; x, y eksenlerinin etiketlendirilmesi, başlık eklenmesi, grid eklenmesi, rengin değiştirilmesi, grafiğin WMF, PNG, EPS, ... gibi formatlarda kaydedilmesi...

plot fonksiyonu

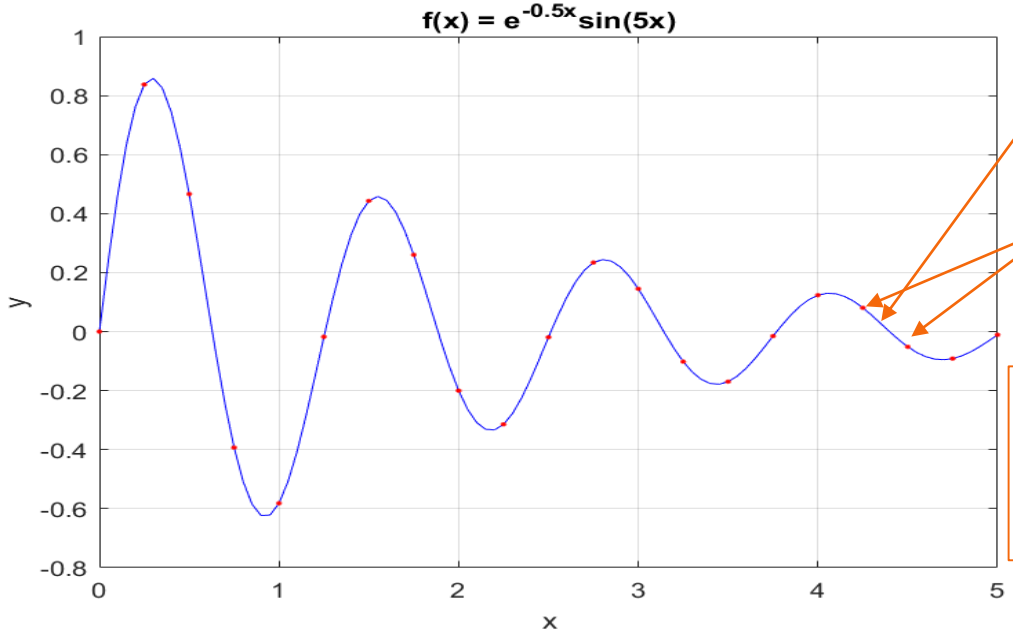
```
>> x = linspace(0,5,101);
>> y = f(x);
>> plot(x,y,'b-')                    % mavi düz çizgi
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x}sin(5x)');
```



Grafik açıklamaları yukarıda da gösterildiği gibi birbirinden ayrı komutlarla veya figür penceresinde (figure window) grafik editörü (plot editor) yardımıyla yapılabilir.

aynı plot'ta birden fazla grafik

```
>> x5 = x(1:5:end);
>> y5 = y(1:5:end);
>> plot(x,y,'b-',x5,y5,'r.');
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x}sin(5x)');
```

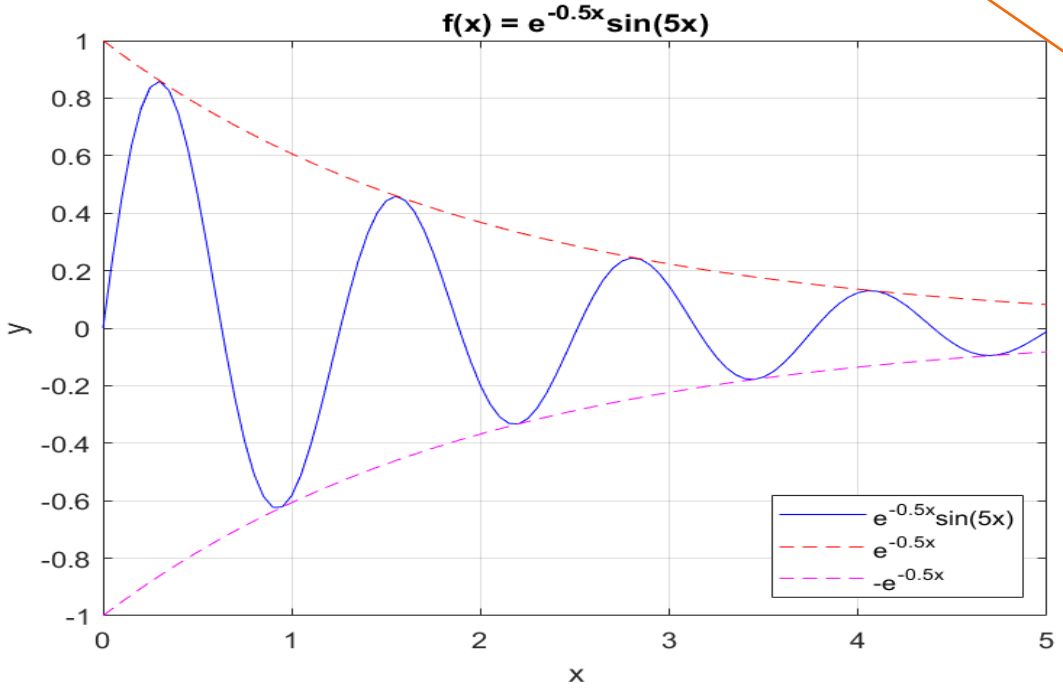


(x, y) mavi düz çizgi olarak çizildi

(x5, y5) kırmızı noktalar olarak çizildi

Birden fazla (x,y) çifti (aynı uzunluk (size) olması şart değil) farklı çizgi stilleri ile çizilebilir.

```
>> e = exp(-0.5*x);
>> plot(x,y,'b-',x,e,'r--',x,-e,'m--');
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x}sin(5x)');
>> legend('e^{-0.5x}sin(5x)', 'e^{-0.5x}',...
    '-e^{-0.5x}', 'location', 'SE');
```



bir sonraki satırdan devam etmek için üç nokta

south-east

Birden fazla eğri çizmek ve açıklama (legend) eklemek

legend, plot editöründen de eklenebilir.

9. Fonksiyon maksimumu ve minimumu

Mühendisler her zaman tasarımlarını mümkün olan en iyi çözümleri bularak optimize etmeyi severler. Bu, genellikle tasarım parametrelerinin bazı fonksiyonlarını minimize veya maksimize etmekle ilgilidir.

Bir f(x) fonksiyonunun [a, b] aralığında bir minimumu (veya maksimumu) olsun. Bunu bulmak için aşağıdaki üç metod kullanılabilir:

1. **min (veya max) fonksiyonunun kullanıldığı grafik metodu**
2. **fminbnd hazır fonksiyonunun kullanılması**
3. **fzero fonksiyonunun kullanılması (f(x)'in türevinin bilinmesi koşuluyla)**

(çok değişkenli fonksiyonlar için fminsearch kullanılır)

Üç metod için MATLAB uygulaması

```
f = @(x) ...                    % Fonksiyonu tanımla
                                % f(x) vektör girişini kabul etmeli
                                % ve vektör çıktısı oluşturmalı
```

1

```
x = linspace(a,b,N);                % N'nin büyük olması önemli
[fmin,imin] = min(f(x));            % fmin = minimum değer
xmin = x(imin);                     % minimumun x'teki değeri
plot(x,f(x),xmin,fmin,'o');         % görüntü
```

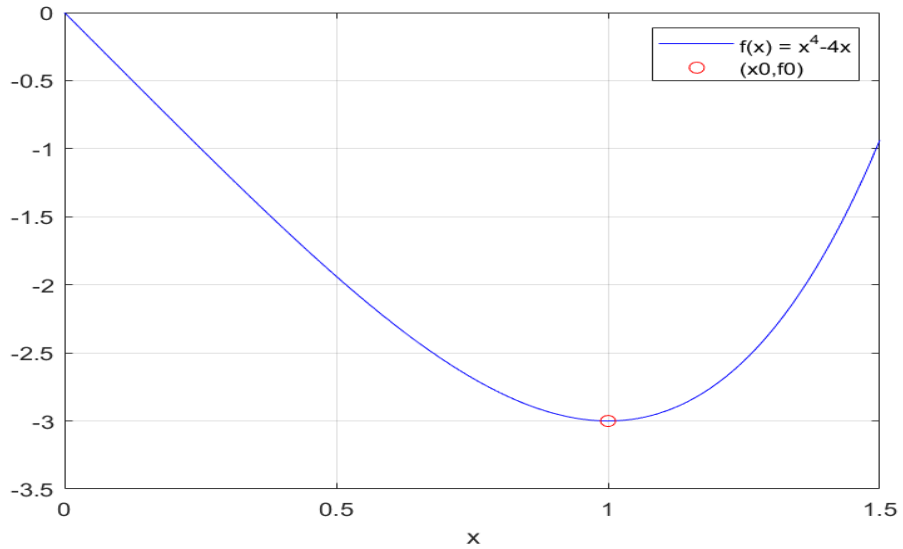
2

```
[xmin,fmin] = fminbnd(f,a,b);       % [a,b] 'de arar
```

3

```
F = @(x)                % f(x)'in türevini tanımla ya da sembolik
                          toolbox kullan
xmin = fzero(F,x0); % x0 civarında arar
fmin = f(xmin);     % f(x)'nin minimum değeri
```

```
f = @(x) x.^4 - 4*x;

x = linspace(0, 1.5, 151);

[f0, i0] = min(f(x)); x0 = x(i0);

plot(x, f(x), 'b-', x0, f0, 'ro');
xlabel('x'); grid;
legend('f(x) = x^4-4x', '(x0,f0)');
```



Örnek: min fonksiyonunu kullanarak bir eğrinin minimumunu bulmak

f0 y=f(x) dizisinin minimumudur

i0 minimumun dizideki İndeksidir yani, f0 = y(i0)

x0 y'nin minimumundaki x değeridir

Değerler: x0 = 1, f0 = -3

fminbnd fonksiyonunu kullanarak f(x)'in minimumunu bulmak

fminbnd ve fzero function handle (@(x))'ı girdi olarak kabul eder.

```
f = @(x) x.^4 - 4*x;                    % [0, 1.5] aralığındaki
[x1,f1] = fminbnd(f,0,1.5);             % f(x)'in minimumunu bulur.
```

fzero fonksiyonunu kullanarak f(x)'in minimumunu bulmak (F(x) = df(x)/dx türevi gerekli)

```
F = @(x) 4*x.^3 - 4;                        % f(x)'in türevi
x2 = fzero(F, 0.5); f2 = f(x2);

[x0,x1,x2; f0,f1,f2]                        % üç metodu karşılaştırır
```

```
ans =
    1.0000    1.0000    1.0000
   -3.0000   -3.0000   -3.0000
```

fminbnd kullanarak f(x)'in maksimumu nasıl bulunur?

```
f = @(x) ...                    % Fonksiyonu tanımla
                                % f(x) vektör girişini kabul etmeli
                                % ve vektör çıktısı oluşturmalı

[xmax,fmin] = fminbnd(@(x) -f(x),a,b);
fmax = -fmin;



% fminbnd(-f,a,b) -> bu şekilde bir yazıma izin yok.
% alternatif olarak, f(x)'in negatifi şu şekilde tanımlanır

g = @(x) -f(x);    % yani, g(x) = -f(x)

[xmax,fmin] = fminbnd(g,a,b);
fmax = -fmin;
```

10. String, cell array, fprintf

- Karakterler ve diziler (strings)
- Dizilerin sıralanması
- num2str kullanımı
- Dizilerin strcmp ile karşılaştırılması

- Hücre dizileri (cell arrays)
- Cell x içerik dizinleme

- fprintf – özet ve örnekler

MATLAB Data Sınıfları

Karakterler | **Mantıksal** | **Nümerik** | **Sembolik** | **Cell** | **Yapı**

- **Nümerik**
  - **Tam Sayı**
    - **İşaretli**
    - **İşaretsiz**
  - **Kayan Nokta**
    - **Single precision**
    - **Double precision**

Daha fazlası

- **Java sınıfları**
- **Kullanıcı tanımlı sınıflar**
- **Function handles**

Hücre (Cell) ve Yapı (Structure) dizileri (array) aynı dizide farklı tipte datalar olarak depolanabilir.

Karakterler ve diziler (strings)

String karakterlerden oluşan dizidir.

Karakterler dahili olarak ASCII (American Standard Code for Information Interchange) kodları olarak adlandırılan standart numaralar ile temsil edilir. (Wikipedia'dan ASCII table)

Command Window

```
>> c = 'A'

c =

    'A'

>> x = double(c)

x =

    65

>> char(x)

ans =

    'A'

>> class(c)

ans =

    'char'

>>
```

% 'A' için ASCII kodu

char() bir karakter dizisi oluşturur

>> doc char
>> doc class

```
Command Window
>> s = 'ABC DEFG'

s =

    'ABC DEFG'

>> x = double(s)

x =

    65    66    67    32    68    69    70    71

>> char(x)

ans =

    'ABC DEFG'

>> size(s)

ans =

     1     8

>> class(s)

ans =

    'char'

fx >>
```

ASCII kodları

ASCII kodlarını karakterlere çevirir

s 8 karakterli bir satır vektördür

```
>> s(2), s(3:5)

ans =

    'B'

ans =

    'C D'

fx >>
```

Dizilerin (strings) sıralanması

Command Window

```
>> s = ['Albert', 'Einstein']

s =

    'AlbertEinstein'

>> s = ['Albert', ' Einstein']

s =

    'Albert Einstein'

>> s = ['Albert ', 'Einstein']

s =

    'Albert Einstein'

>> size(s)

ans =

     1    15

>>
```

Öndeki ve sondaki boşlukları korur

```
>> doc strcat
>> doc strvcat
>> doc num2str
>> doc strcmp
>> doc findstr
```

Dikey sıralama

```
Command Window
>> s = ['Apple'; 'IBM'; 'Microsoft']
Dimensions of arrays being concatenated are not consistent.

>> s = ['Apple     '; 'IBM       '; 'Microsoft']

s =

  3×9 char array

    'Apple     '
    'IBM       '
    'Microsoft'

>> size(s)

ans =

     3     9

fx >>
```

vertcat kullanımı hatası: CAT argümanlarının boyutları uyumlu değil

En uzun string uzunluğuna eşitlemek için boşluklar eklendi.

Dikey sıralama

```
>> s = strvcat('Apple', 'IBM', 'Microsoft');
>> s = char('Apple', 'IBM', 'Microsoft');

s =
Apple
IBM
Microsoft

>> size(s)

ans =
     3    9
```

strvcat, char
İkisinde de gerektiği gibi boşluklar eklenir.

Tavsiye:
dikey sıralamak için char,
yatay sıralamak için [ ] kullan.

num2str

```
Command Window
>> a = [143.87, -0.0000325, -7545]';
>> s = num2str(a)

s =

  3×9 char array

    '   143.87'
    '-3.25e-05'
    '    -7545'

>>
>> s = num2str(a,4)

s =

  3×9 char array

    '    143.9'
    '-3.25e-05'
    '    -7545'

>> s = num2str(a,'%12.6f')

s =

  3×12 char array

    '  143.870000'
    '   -0.000032'
    '-7545.000000'
```

s = num2str(A)
s = num2str(A, duyarlılık (precision))
s = num2str(A, format)

max. 4 basamak

özel format

Dizilerin karşılaştırılması

String karakterler dizisidir, bu yüzden **s1==s2** koşulu **s1** ve **s2**'nin aynı uzunlukta olmasını gerektirir.

Command Window

```
>> s1 = 'short'; s2 = 'shore';
>> s1 == s1

ans =

  1×5 logical array

   1   1   1   1   1

>> s1 == s2

ans =

  1×5 logical array

   1   1   1   1   0

>> s1 = 'short'; s2 = 'long';
>> s1 == s2
Matrix dimensions must agree.

fx >>
```

eq kullanımı hatası
Matris boyutları uyumlu değil

Dizilerin karşılaştırılması

Eşit uzunlukta olmayan dizileri karşılaştırmak ve ikili bir sonuç almak için strcmp kullanılmalı

```
Command Window
>> s1 = 'short'; s2 = 'shore';
>> strcmp(s1,s1)

ans =

  logical

   1

>> strcmp(s1,s2)

ans =

  logical

   0

>> s1 = 'short'; s2 = 'long';
>> strcmp(s1,s2)

ans =

  logical

   0
```

>> doc strcmp
>> doc strvcmpi

büyük küçük harfe duyarlı değil

Kullanışlı Dizi Fonksiyonları

sprintf	- biçimlendirilmiş diziyi yazar
sscanf	- biçimlendirilmiş diziyi okur
deblank	- sondaki boşlukları kaldırır
strcmp	- dizileri karşılaştırır
strcmpi	- dizileri karşılaştırır
strmatch	- olası eşleşmeleri bulur
upper	- büyük harfe dönüştürür
lower	- küçük harfe dönüştürür
blanks	- boşluk dizisi
strjust	- diziyi sola/sağa yaslar, ortalar
strtrim	- baştaki/sondaki boşlukları kaldırır
strrep	- dizileri değiştirir
findstr	- bir diğeri içinde başka bir dizi bulur

Hücre Dizileri (Cell Arrays)

Hücre dizileri (cell arrays) her türlü datayı kapsar:
vektörler, matrisler, diziler (strings), yapılar (structures), diğer hücre dizileri, fonksiyonlar

Bir hücre { } parantezleri içine farklı türde nesneler yerleştirerek oluşturulur, örneğin A, B, C, D keyfi nesneler olmak üzere

```
c = {A, B, C, D};        % 1x4 hücre
c = {A; B; C; D};        % 4x1 hücre
c = {A, B; C, D};        % 2x2 hücre
```

```
c{i,j} i,j hücresindeki dataya ulaşır
c(i,j) i,j pozisyonundaki hücre nesnesine ulaşır
```

Hücre X içerik dizinleme

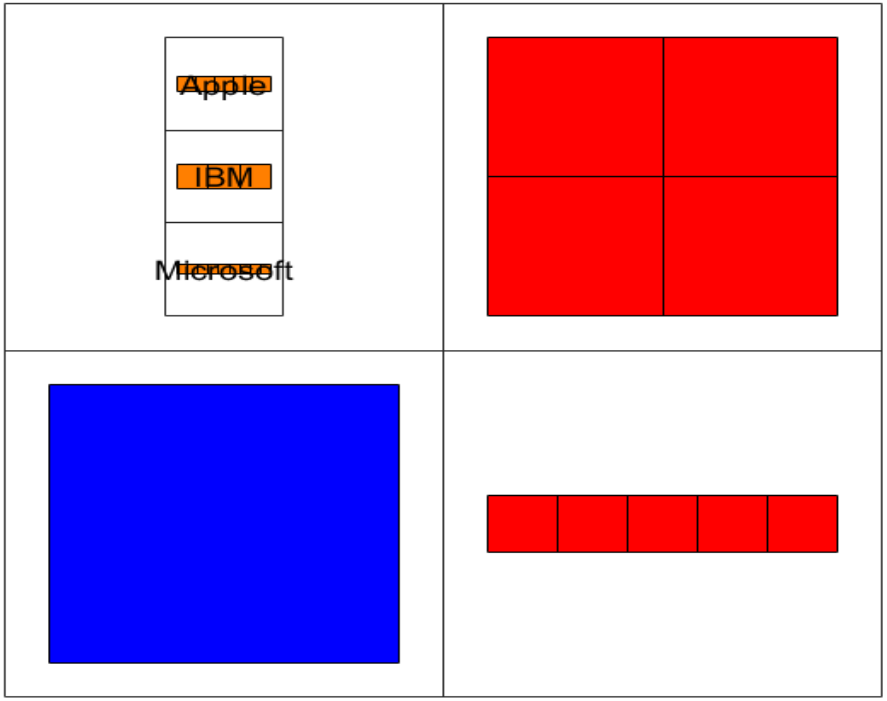
```
A = {'Apple'; 'IBM'; 'Microsoft'};      % hücre
B = [1 2; 3 4];                         % matris
C = @(x) x.^2 + 1;                      % fonksiyon
D = [10 20 30 40 50];                   % satır vektör
```

```
>> c = {A, B; C, D}
```

% 2x2 hücre dizisi tanımla

```
c =

  2×2 cell array

    {3×1 cell   }    {2×2 double}
    {@(x)x.^2+1}    {1×5 double}

>> cellplot(c);
>>
```

Hücre dizisinin görsel gösterimi



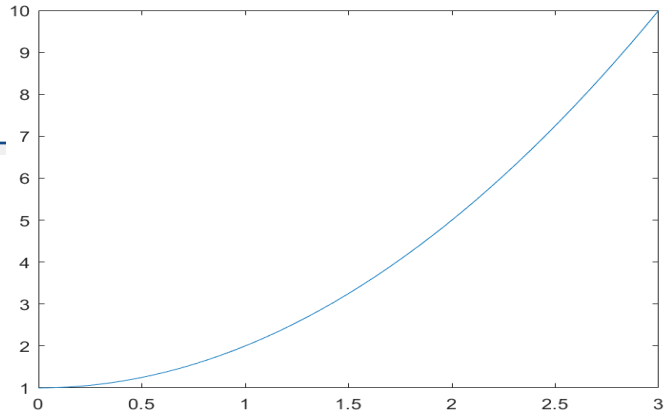
```
>> celldisp(c)

c{1,1}{1} =
Apple
c{1,1}{2} =
IBM
c{1,1}{3} =
Microsoft

c{2,1} =
    @(x)x.^2+1

c{1,2} =
     1     2
     3     4

c{2,2} =
    10    20    30    40    50
```

{ } ile içerik dizinleme

```
>> c{1,1}

ans =

  3×1 cell array

    {'Apple'    }
    {'IBM'      }
    {'Microsoft'}

>> c{2,1}

ans =

  function_handle with value:

    @(x)x.^2+1

fx >>
```

{ } ile içerik dizinleme

```
>> c{1,1}{3}

ans =

    'Microsoft'

>> c{1,1}{3}(6)

ans =

    's'

>> x = [1 2 3];
>> c{2,1}(x)

ans =

     2     5    10

>> fplot(c{2,1},[0,3]);
>>
```




```
>> c{1,2}(2,:)

ans =

     3     4

>> c{1,2}(1,2)

ans =

     2

>> c{2,2}(3)

ans =

    30

>>
```

Command Window

```
>> c(2,2)

ans =

  1×1 cell array

    {1×5 double}

>> class(c(2,2))

ans =

    'cell'

>> c{2,2}

ans =

    10    20    30    40    50

>> class(c{2,2})

ans =

    'double'
```

hücre

hücre içeriği

hücre dizinleme ( )
içerik dizinleme { }

```
>> D = cell2mat(c(1,2))

D =

     1     2
     3     4

>> class(D)

ans =

    'double'

>> E = num2cell(D)

E =

  2×2 cell array

    {[1]}    {[2]}
    {[3]}    {[4]}

>> size(E)

ans =

     2     2

>> size(c(1,2))

ans =

     1     1

>>
```

num2cell - nümerik diziyi hücre diziye çevirir
cell2mat - hücre diziyi nümerik diziye çevirir

Neden?

fprintf – özet ve örnekler

```
fprintf('format_specs', variables);
```

yazdırma formatı özellikleri

yazdırılacak olan değişkenlerin, dizilerin ya da matrislerin listesi

```
>> doc fprintf
>> doc sprintf
```

Örnek 1

100*pi ‘yi yazdırmanın farklı yolları

```
fprintf('%10.6f\n', 100*pi)        314.159265
fprintf('% 10.6f\n', 100*pi)        314.159265
fprintf('%-10.6f\n', 100*pi)       314.159265
fprintf('%+10.6f\n', 100*pi)       +314.159265
fprintf('%10.0f\n', 100*pi)                314
fprintf('%#10.0f\n', 100*pi)              314.
fprintf('%010.0f\n', 100*pi)       0000000314
```

10 boşluk

```
%10.6f      % genişlik 10, ondalık basamak sayısı 6
% 10.6f     % baştan boşluk bırakır
%-10.6f     % sola yaslar
%+10.6f     % + ya da – işaretini yazar
%10.0f      % ondalık basamakları yazmaz
%#10.0f     % ondalık basamak noktasını koyar
%010.0f     % boşluklara sıfır koyar
```

flag

genişlik ve duyarlılık (precision)

dönüşüm karakterleri:

d, i	tamsayı formatı
f	sabit-nokta formatı
e, E, g	üstel format
c,s	karakter veya dizi (string)
x	onaltılı format

Örnek 2

```
a = [1; -2; 3; 4];
b = [10; 20; -30; 40];
c = [100; 200; 300; -400];
```

İlk sütunu hizalamak için en az **%6.3f** 'e İhtiyaç vardır.

```
>> [a, b, c]

ans =
     1    10   100
    -2    20   200
     3   -30   300
     4    40  -400
```

```
fprintf('%9.3f %9.3f %9.3f\n', [a, b, c]');
```

```
 1.000   10.000  100.000
-2.000   20.000  200.000
 3.000  -30.000  300.000
 4.000   40.000 -400.000
```

vektörleştirilmiş versiyon

döngü versiyonu

```
for i=1:4
    fprintf('%9.3f %9.3f %9.3f\n', a(i), b(i), c(i));
end
```

Örnek 3 – metin ve sayılar

```
a = [1; -2; 3; 4];
b = [10; 20; -30; 40];
s = {'a', 'bb', 'ccc', 'dddd'};

for i=1:4
    fprintf('%9.3f %9.3f %4s\n', a(i), b(i), s{i});
end
```

String'lerin hücre dizileri

Kıvrımlı parantez

```
    1.000    10.000    a
   -2.000    20.000   bb
    3.000   -30.000  ccc
    4.000    40.000 dddd
```

max. 4 karakter

```
N = [num2cell([a';b']);s];
fprintf('%9.3f %9.3f %-4s\n', N{:})
```

sola yaslı

```
    1.000    10.000 a
   -2.000    20.000 bb
    3.000   -30.000 ccc
    4.000    40.000 dddd
```

```
fprintf('%9.3f %9.3f %-4s\n', N{:})
```

Bir satırdaki üç girdinin yazdırılması için formatlandırılmış özellikleri kullan

```
    1.000    10.000 a
   -2.000    20.000 bb
    3.000   -30.000 ccc
    4.000    40.000 dddd
```

```
>> N{:}
ans =
     1
ans =
    10
ans =
    'a'
ans =
    -2
ans =
    20
ans =
    'bb'
ans =
     3
ans =
   -30
ans =
    'ccc'
ans =
     4
ans =
    40
ans =
    'dddd'
```

```
>> N

N =

  3×4 cell array

    {[ 1]}    {[-2]}    {[  3]}    {[   4]}
    {[10]}    {[20]}    {[-30]}    {[  40]}
    {'a' }    {'bb'}    {'ccc'}    {'dddd'}

fx >>
```