# MATLAB matematikai alkalmazások

```
>> format long
>> format compact
```

**fzero** : egyváltozós függvény gyökének keresése
Az első paraméter a függvény képlete vagy neve, a második paraméter lehet az intervallum (2 hosszú vektor), ahol a gyököt keressük vagy az iterációs numerikus módszer kezdeti értéke.

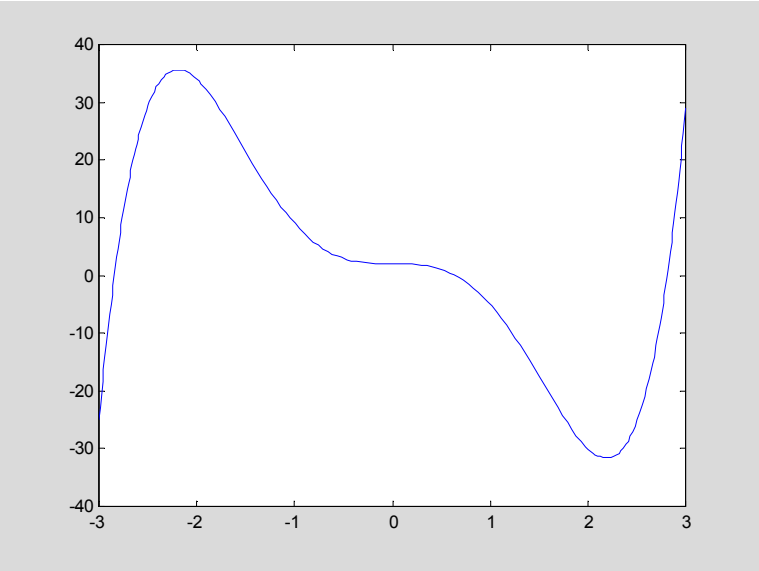
```
>> fzero('cos',[1,2])
Zero found in the interval: [1, 2].
ans =
   1.57079632679490

>> fzero('cos',[-1,1])
??? Error using ==> fzero
The function values at the interval endpoints must differ in
sign.

>> fzero('x^3-2',0)
Zero found in the interval: [-1.28, 1.28].
ans =
   1.25992104989487
```

---
```
function y=f1(x)
y=x.^5-8*x.^3+2;
```
---
Megjegyzés: az előbbi függvény vektoriálisan is meghívható.

```
>> fplot('f1',[-3,3])
```



```
>> x=fzero('f1',0)
Zero found in the interval: [-0.9051, 0.9051].
x =
   0.64113504097055

>> x=fzero('f1',-4)

Zero found in the interval: [-2.72, -4.9051].
x =
  -2.84375920373960

>> x=fzero('f1',4)
Zero found in the interval: [2.72, 4.9051].
x =
   2.81249009945444
```

**trapz:** határozott integrál számítása trapéz szabállyal
Az intervallumnak vesszük egy `x` beosztását, és a beosztás pontjaiben legeneráljuk a függvényértékeket az `y` vektorba. A két vektort kell átadni a `trapz` parancsnak.

$$\int_0^{\pi} \sin(x)dx = 2$$

```
>> x=0:0.1:pi;
>> y=sin(x);
>> trapz(x,y)
ans =
   1.99746892659093

>> x=0:0.01:pi;
>> y=sin(x);
>> trapz(x,y)
ans =
   1.99998206504367
```

**quad:** Határozott integrál közelítése (adaptív Simpson-szabállyal)
Az első paraméter a függvény képlete illetve neve, a második és harmadik paraméter az intervallum kezdete és vége. Megadható egy negyedik paraméterben, hogy milyen pontosságú legyen a közelítő érték.

```
>> quad('sin',0,pi)
ans =
   1.99999999619084

>> quad('sin',0,pi,1e-10)
ans =
   1.99999999999988
```

```
>> quad('f1',0,1)
ans =
   0.16666666666667
```

**dblquad:** kétváltozós függvény integrálja
A meghívása hasonló a `quad` függvényéhez, nézzük egy példát:

$$\iint_A xy-3x-y\text{^}3\,dxdy, \quad A=[0,1]\times[0,1]$$

```
---------------------------------------
function z=f2(x,y)
z=x*y-3*x-y^3;
---------------------------------------
```

```
>> dblquad('f2',0,1,0,1)
ans =
  -1.50000000000000
```

**fmin:** egyváltozós függvény minimumhelyének keresése az adott intervallumon

```
>> x=fmin('f1',0,3)
x =
   2.19089544395934
```

**fminsearch:** többváltozós függvény minimumhelyének keresése
A legegyszerűbb meghívása: első paraméter a függvény neve, a második paraméter az iterációs módszer kezdeti értéke:

```
-------------------------------------
function z=f3(x)
z=(x(1)-1)^2+3*(x(2)-x(1))^2;
-------------------------------------
```

f3 minimumhelyének keresése a [0,0] pontból indítva:

```
>> x=fminsearch('f3',[0,0])
x =
   0.99996746085006   0.99997360469626
```

**eig:** sajátérték, sajátvektor meghatározás

```
>> A=[3,4,1;2,0,2;-1,1,1]
A =
     3     4     1
     2     0     2
    -1     1     1
```

```
>> format short
>> eig(A)
ans =
    4.3166
   -2.3166
    2.0000

>> [V,D]=eig(A)
V =
    0.9218    0.5074   -0.7071
    0.3468   -0.7707   -0.0000
   -0.1734    0.3854    0.7071
D =
    4.3166         0         0
         0   -2.3166         0
         0         0    2.0000
```

D diagonálisában az A mátrix sajátértékei találhatók, V oszlopvektorai a megfelelő sajátértékekhez tartozó sajátvektorok.

**Differenciálegyenletek megoldása**

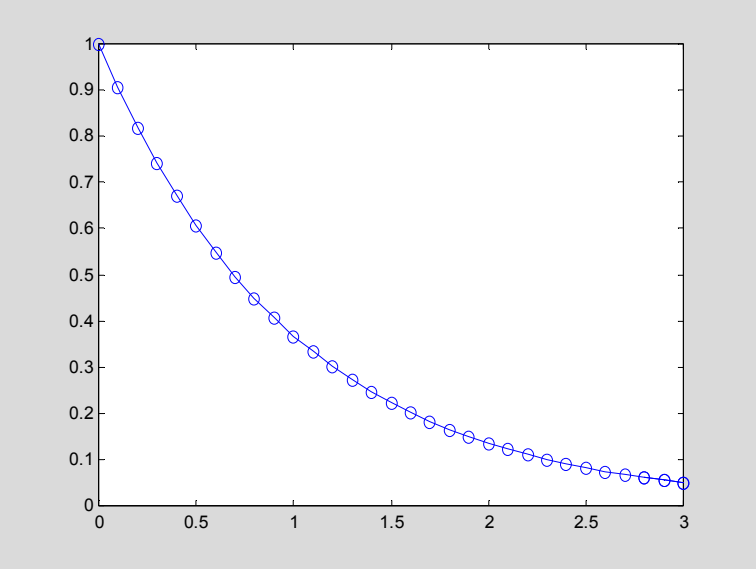
Az

$x' = f(t,x), \quad x(t_0) = z, \qquad x : R \to R^n,\ f : R \times R^n \to R^n$

alakú differenciálegyenlet-rendszerek megoldását például az **ode45** függvénnyel számolhatjuk ki. Ennek meghívása: első paraméter: az egyenlet jobb oldalát definiáló `f` függvény neve, második paraméter: azon t értékek vektora, ahol a numerikus közelítést ki szeretnénk számolni (a vektor első eleme a kezdeti időpont), a harmadik paraméter a `z` kezdeti érték.
Hasonló, más numerikus módszert használó differenciálegyenlet megoldó parancsok: `ode23`, `ode113`, `ode15s`, stb.

x'=-x, x(0)=1

```
-------------------------------------
function y=de1(t,x)
y=-x;
-------------------------------------
>> tt=0:0.1:3;
>> ode45('de1',tt,1)
```

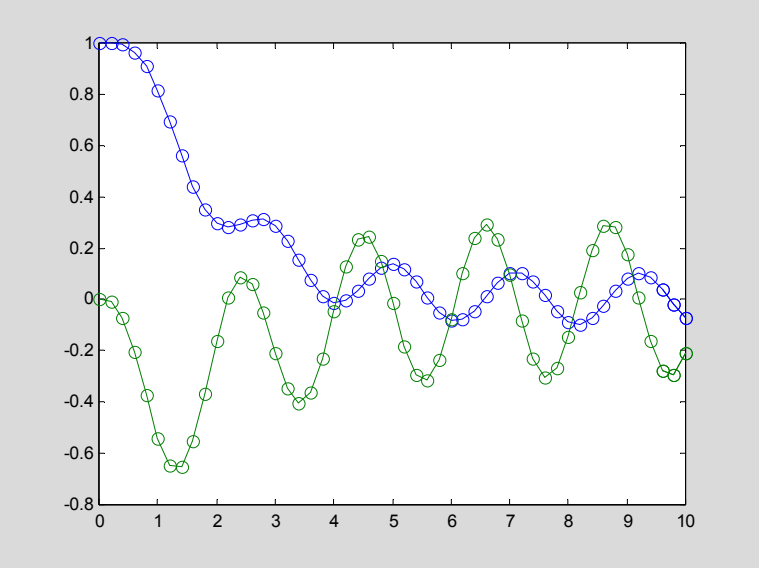



$$x_1' = x_2$$
$$x_2' = -x_1 - 2x_2 + \cos(3t)$$
$$x_1(0) = 1, \quad x_2(0) = 0, \quad t \in [0,10]$$

Az egyenlet jobb oldalát definiáló függvény:

```
-----------------------------------------------
function y=de2(t,x)
y=[x(2);-x(1)-2*x(2)+cos(3*t)];
-----------------------------------------------
```

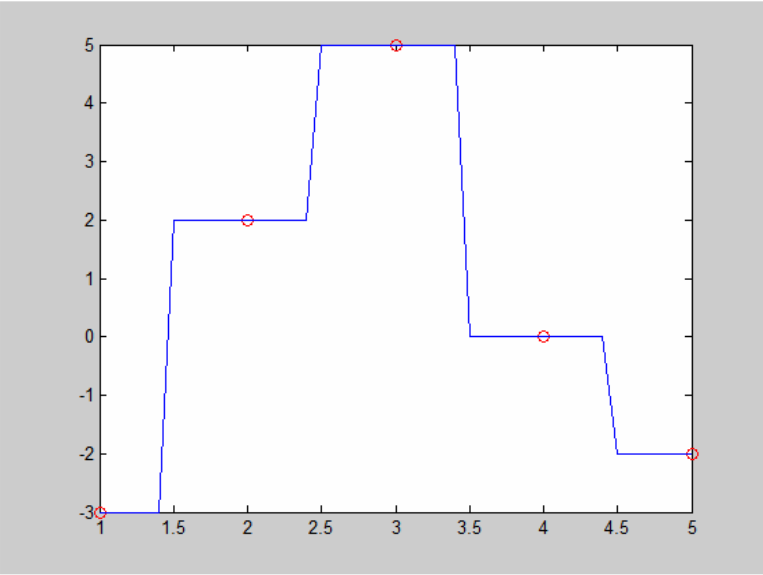
```
>> tt=0:0.2:10;
>> ode45('de2',tt,[1;0])
```



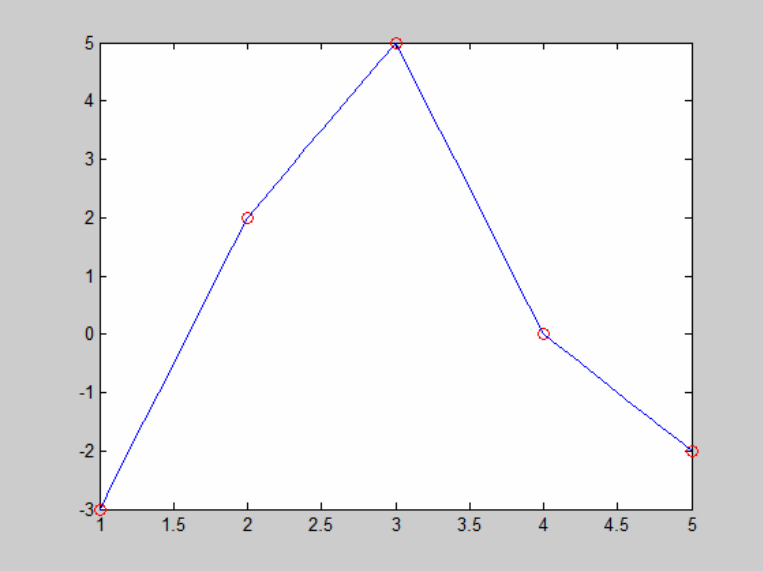
**Spline interpoláció**

A lineáris interpoláció feladatában lineáris függvényekkel kötünk össze megadott pontokat. Az `interp1` függvény első két paraméterében az input pontok x illetve y koordinátáit adjuk meg, a harmadik paraméterben pedig azon x értékek koordinátáit adjuk meg, ahol az interpoláló fügvényt ki szeretnénk értékelni, a függvény outputja az interpolációs értékeket tartalmazó vektor. A negyedik paraméterben megadhatjuk a 'nearest' ,'linear' (default), 'cubic', opciókat, amely az interpolációs módszert határozza meg: a 'nearest' szakaszonként konstans, a 'linear' szakaszonként lineáris, a 'cubic' pedig szakaszonként harmadfokú polinommal köti össze a megadott adatokat.

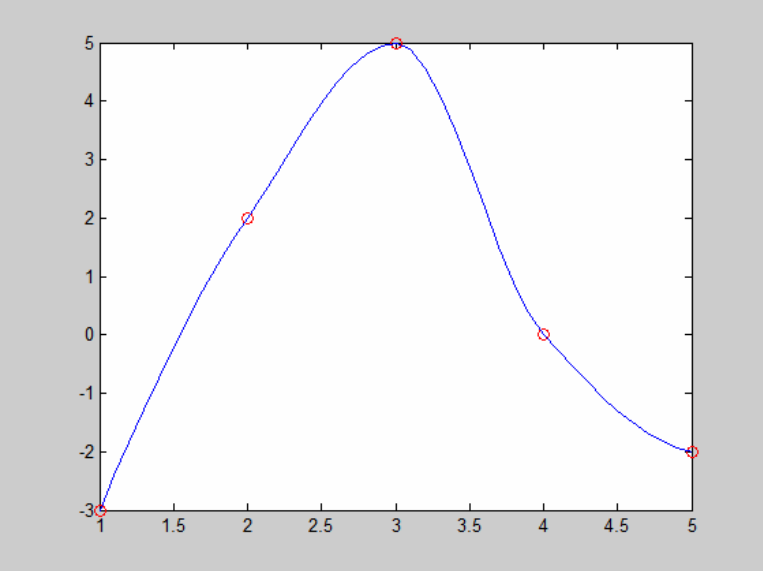
```
>> x =[1, 2, 3, 4, 5]
>> x =
     1     2     3     4     5
>> y=[-3, 2,5,0,-2]
y =
    -3     2     5     0    -2
>> t=1:0.1:5;
>> z=interp1(x,y,t,'nearest');
>> plot(x,y,'or',t,z,'b')
```



```
>> z=interp1(x,y,t,'linear');
>> plot(x,y,'or',t,z,'b')
```




```
>> z=interp1(x,y,t,'cubic');
>> plot(x,y,'or',t,z,'b')
```

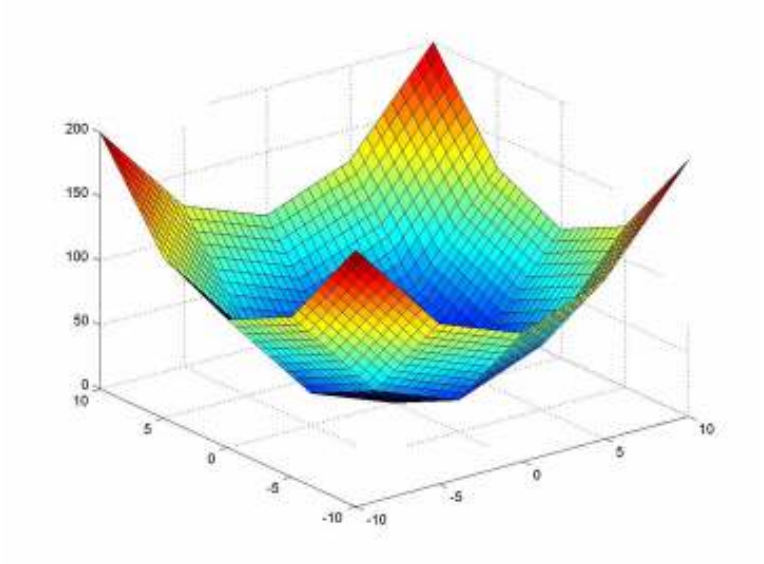


Az `interp2` függvény kétdimenziós függvények értékeit interpolálja.
Használatára a következő parancsfájl mutat példát.

```
-----------------------------------------
% 2 dimenziós interpoláció   : intpol2.m

[x,y]=meshgrid(-10:5:10,-10:5:10);        % egy durva rács a [-5,5]x[-5,5]
négyzeten
z=x.^2+y.^2;                              % függvényértékek a rácspontokban
[xi,yi]=meshgrid(-10:0.5:10,-10:0.5:10);% egy finomabb rács a rajzoláshoz
zi=interp2(x,y,z,xi,yi);                  % a megadott mérési adatok
interpolációja
surf(xi,yi,zi)
-----------------------------------------
```

```
>> intpol2
```



**Polinom interpoláció**

Adott n+1 db pont: $(x_i, y_i)$, i=0,1,...,n. Keressünk egy olyan n-edfokú

$p(t) = a_0 + a_1 t + \cdots + a_n t^n$

polinomot, amelynek grafikonja átmegy a megadott pontokon, azaz teljesíti az

$a_0 + a_1 x_0 + \cdots + a_n x_0^n = y_0$

$a_0 + a_1 x_1 + \cdots + a_n x_1^n = y_1$

$\vdots$

$a_0 + a_1 x_n + \cdots + a_n x_n^n = y_n$

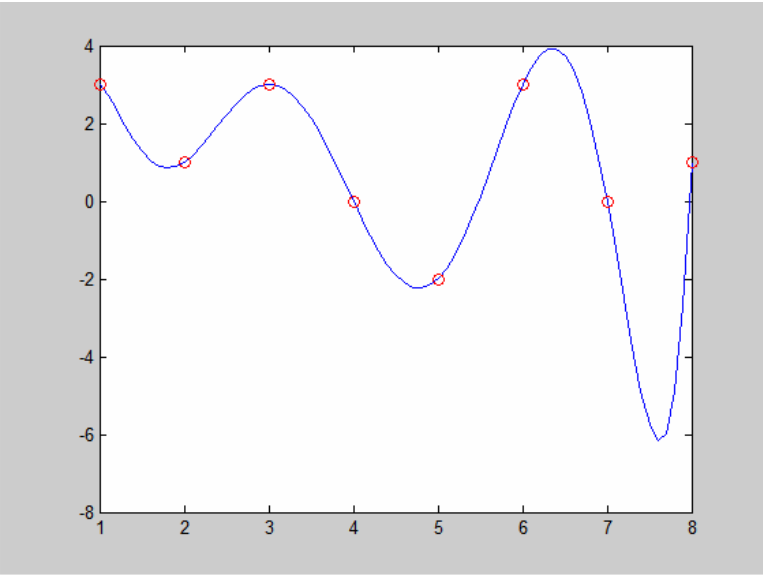
egyenletrendszert. Ez egy $Ea = y$ alakú lineáris egyenletrendszer az ismeretlen együtthatók, $a = (a_0, a_1, \ldots, a_n)^T$ meghatározására, ahol

$$E = \begin{pmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix} \quad \text{és} \quad y = (y_0, y_1, \ldots, y_n)^T .$$

---

```
% adatok:
x=[1; 2; 3; 4; 5; 6; 7; 8];
y=[3; 1; 3; 0; -2; 3; 0; 1];
% együtthatómátrix generálása
E=[ones(8,1), x, x.^2, x.^3, x.^4, x.^5, x.^6, x.^7];
a=E\y;
% az interpolációs polinom és a megadott adatok ábrázolása
t=1:0.1:8;
z=a(1)+a(2).*t+a(3).*t.^2+a(4).*t.^3+a(5).*t.^4+a(6).*t.^5+a(7).*
t.^6+a(8).*t.^7;
plot(t,z,'-b',x,y,'or')
```

---



## Egyenes illesztés

Adott n+1 db pont: $(x_i, y_i)$, i=0,1,...,n. Keressünk egy olyan lineáris $p(t) = a_0 + a_1 t$ polinomot, amely a legkisebb hibával illeszkedig a megadott adatokra, azaz ahol a $\sum_{i=0}^{n} (a_0 + a_1 x_i - y_i)^2$, ú.n. legkisebb négyzetes hiba minimális.

Ha felírjuk az

$a_0 + a_1 x_0 = y_0$

$a_0 + a_1 x_1 = y_1$

$\vdots$

$a_0 + a_1 x_n = y_n$

egyenletrendszert, ennek általában nincs megoldása, ha n>1. Ez egy ú.n. túldefiniált $Ea = y$ lineáris egyenletrendszer, ahol

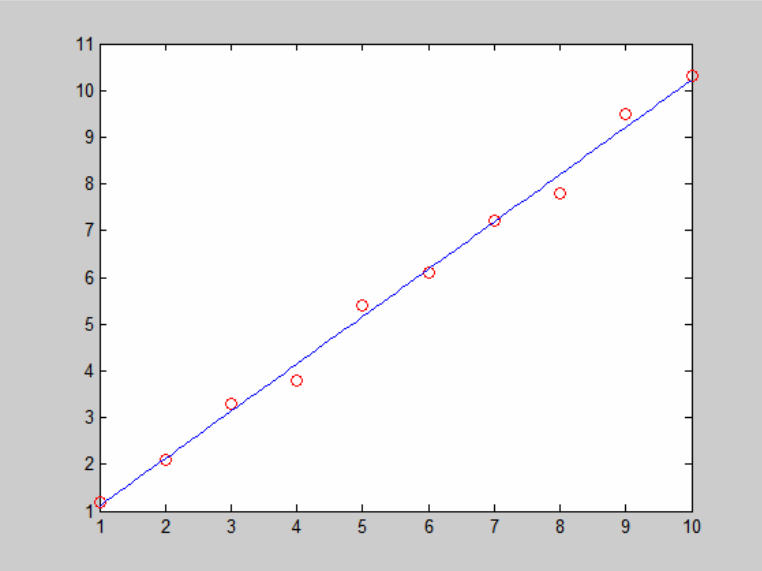
$$E = \begin{pmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}_{(n+1)\times 2} , \quad \text{és} \quad y = (y_0, y_1, \ldots, y_n)^T .$$

A Matlab az `E\y` paranccsal ennek a legkisebb négyzetes megoldását, azaz a fentebb definiált feladat megoldását adja vissza.

---

```
x=[1; 2; 3; 4; 5; 6; 7; 8; 9; 10;];
y=[1.2; 2.1; 3.3; 3.8; 5.4; 6.1; 7.2; 7.8; 9.5; 10.3];
E = [ones(size(x)) x];
a=E\y
t=1:0.1:10;
z=a(1)+a(2).*t;
plot(x,y,'or',t,z,'-b')
```

---



**Polinom illesztés**

Adott n+1 db pont: $(x_i, y_i)$, i=0,1,...,n. Keressünk egy olyan $p(t) = a_0 + a_1 t + \cdots + a_m t^m$ m-edfokú polinomot, amely a legkisebb hibával illeszkedig a megadott adatokra, azaz ahol a $\sum_{i=0}^{n} (a_0 + a_1 x_i + \cdots + a_m x_i^m - y_i)^2$, ú.n. legkisebb négyzetes hiba minimális.

Az

$$a_0 + a_1 x_0 + \cdots + a_m x_0^m = y_0$$
$$a_0 + a_1 x_1 + \cdots + a_m x_1^m = y_1$$
$$\vdots$$
$$a_0 + a_1 x_n + \cdots + a_m x_n^m = y_n$$

egyenletrendszernek általában nincs megoldása, ha n>m. Ez is egy túldefiniált $Ea = y$ lineáris egyenletrendszer, ahol

$$E = \begin{pmatrix} 1 & x_0 & \cdots & x_0^m \\ 1 & x_1 & \cdots & x_1^m \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^m \end{pmatrix}_{(n+1)\times(m+1)} , \quad \text{és} \quad y = (y_0, y_1, \ldots, y_n)^T .$$

A Matlab az `E\y` paranccsal kapjuk itt is vissza a legkisebb négyzetes megoldást.

---

```
x=[-4; -2; -1; 0; 1; 2; 3];
y=[3; 7.2; 6.8; 5.3; 0.8; -5.4; -11.1];
E = [ones(size(x)) x x.^2];
a=E\y
t=-4:0.1:3;
z=a(1)+a(2).*t+a(3).*t.^2;
plot(x,y,'or',t,z,'-b')
```

---