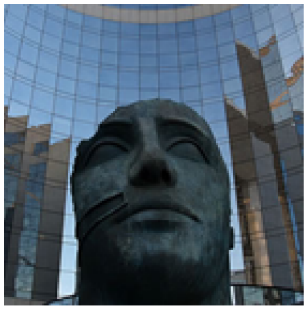


1. Gyakorlat

- **Kerekítési hiba - Hibaterjedés - Kondiciószám**

- **1. példa**

Az alábbi két szám összeadásánál gépi precízió esetén a 10^{-16} hozzáadott érték nem jelenik meg, azonban ha szoftveresen növeljük a precíziót 17-re, akkor már ábrázolható. Használjuk a `vpa` függvényt!

```
>> clc; clear all;  format long;

% A gépi epszilon

>> eps
ans =
   2.220446049250313e-016

>> % a kerekítési egység

>> eru=0.5*eps
eru =
   1.110223024625157e-016


>> x = 1.2345678901234560; y = 10^(-16);

>> z = x + y;

>> z
z = 1.234567890123456

>> z = vpa (x, 17) + vpa (y, 17);
>> vpa (z, 17)

ans = 1.2345678901234561
```

Alternatív megoldás a `digits` függvénnyel,

```
>> x = 1.2345678901234 56; y = 10^(-16);
>> digits (16);
>> z = vpa (x + y);
>> z

z = 1.234567890123456


>> digits (17);
>> z = vpa (x + y)

z = 1.2345678901234560


>> z = vpa (x) + vpa (y)

z = 1.2345678901234561
```

- **2. példa**

Határozzuk meg az $y = x^3 - 5\,x^2 + 6\,x + 0.55$ polinom értékét az $x = 2.73$ helyen 4 jegyű aritmetikával! Ismételjük meg a műveletet a Horner elrendezést alkalmazva! Hasonlítsuk össze a hibákat!

```
>> x = 2.73;
>> digits (4);
>> t1 = vpa (x^3)
t1 = 20.35

>> t2 = vpa (-5*x^2)
t2 = -37.26

>> t3 = vpa (6*x)
t3 = 16.38

>> y = vpa (t1 + t2 + t3 + 0.55)
y = .200 e - 1
```

A Horner elrendezés,

$$y(x) = 0.55 + x\,(6 + x\,(x - 5))$$

```
% A Horner elrendezés szimbolikus előállítása
>> clear x
>> syms x
>> horner (x^3 - 5*x^2 + 6*x + 55/100)
ans = 11/20 + (6 + (-5 + x)*x)*x
```

```
>> x = 2.73;
>> t1 = vpa ((x - 5)*x)
t1 = -6.197
>> y= vpa (0.55 + (t1 + 6)*x)
y = .122 e - 1

% A megoldás végtelen precizióval

y=11917/1000000

% A megoldás 16 jegy pontosan

>> format long
>>y=x^3 - 5*x^2 + 6*x + 0.55
y = 0.011917000000000
```

Tehát a Horner elrendezés nem csupán hatékonyabb algoritmus de a kerekítési hibákra is kevésbé érzékeny!

- **3. példa**

Határozzuk meg a

$$z = 1 - \frac{1.208}{x}$$

kifejezés kiértékelését az x = 1.209 helyen, két különböző algoritmus szerint, négyjegyű aritmetika esetén. Határozzuk meg ezek relatív hibáit, elemezzük az egyes lépéseket!

Algoritmus 1.

$$y = \frac{1.208}{x}$$

$$z = 1 - y$$

Algoritmus 2.

$$y = x - 1.208$$

$$y = \frac{y}{x}$$

```
% algoritmus 1
format long
>> x = 1.209;
% Az 1. lépés
>> y1 = vpa (1.208/x, 4)
y1 =
 .9992

 % Az eredmény 20 jegy precizióval
y1r = vpa (1.208/x, 20)
y1r =
 .99917287014061207610

 % Az 1. lépés relatív hibája %-ban
ey1 = vpa (abs (y1 - y1r)/y1r*100, 4)
ey1 =
 .2715 e - 2

 % A 2. lépés
z1 = vpa (1 - y1, 4)
z1 =
 .8 e - 3

% Az eredmény 20 jegy precizióval
z1r = vpa (1 - y1r, 20)
z1r =
 .82712985938792390 e - 3

 % Az 2. lépés relatív hibája %-ban
ez1 = vpa (abs (z1 - z1r)/z1r*100, 4)
ez1 =
 3.280
```

Tehát ennek az algoritmusnak a relatív hibája 3.28 %!

```
% 2. algoritmus
% Az 1. lépés
>> y2 = vpa (x - 1.208, 4)
y2 =
 .1000 e - 2
```

```
 % Az eredmény 20 jegy precizióval
y2r = vpa (x - 1.208, 20)
y2r =
 .10000000000001119105 e - 2
 % azaz a relatív hiba "zérus"

 % A 2. lépés
z2 = vpa (y2/x, 4)
z2 =
 .8271 e - 3

  % Az eredmény 20 jegy precizióval
z2r = vpa (y2r/x, 20)
z2r =
  .82712985938801646857 e - 3

   % Az 2. lépés relatív hibája %-ban
 >> ez2 = vpa (abs (z2 - z2r)/z2r*100, 4)
ez2 =
 .3610 e - 2
```

Vagyis a második algoritmus relatív hibája csupán 0.0036 %! Vegyük észre, hogy az 1. algoritmus hibáját alapvetően a 2. lépésben a közel azonos számok kivonása okozta!

```
% A megoldás végtelen precizióval
>> 1/1209

% A megoldás 16 jegy pontosan
ans = 8.271298593879239 e - 004
```

4. példa

Gyakran a feladat megoldása rendkívül érzékeny a bemeneti adatokra. Erre egy példa a Wilkinson- polinom gyökeinek meghatározása.

A Wilkinson - polinom

$$W(z) = \prod_{i=1}^{n} (z - i)$$

A polinom n = 7 esetén

$$W = (-7 + z)\,(-6 + z)\,(-5 + z)\,(-4 + z)\,(-3 + z)\,(-2 + z)\,(-1 + z)$$

azaz

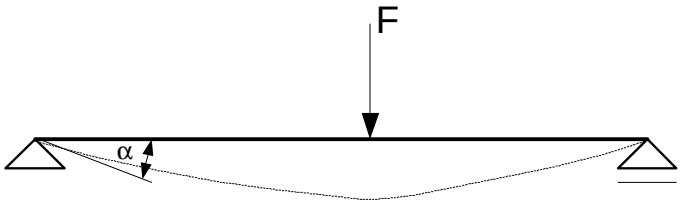
$$W = -5040 + 13\,068\,z - 13\,132\,z^2 + 6769\,z^3 - 1960\,z^4 + 322\,z^5 - 28\,z^6 + z^7$$

```
% A Wilkinson polinom szimbolikus előállítása n = 7 esetén
>> syms W z

>> for i = 1 : 7 W (i) = z - i;
end
>> w = prod (W)
w = (z - 1)*(z - 2)*(z - 3)*(z - 4)*(z - 5)*(z - 6)*(z - 7)
 >> expand (w)
ans = z^7 - 28*z^6 + 322*z^5 - 1960*z^4 + 6769*z^3 - 13132*z^2 + 13068*z - 5040

% A polinom gyökei gépi precizióval
>> roots ([1, -28, 322, -1960, 6769, -13132, 13068, -5040])
ans = 7.000000000000755
5.999999999996947
5.000000000004256
3.999999999997327
3.000000000000800
1.999999999999899
1.000000000000003

% Változtassuk meg a négyzetes tag együtthatóját -13132 - ről -13131 - re
>> roots ([1, -28, 322, -1960, 6769, -13131, 13068, -5040])
ans = 6.918384192904751
        6.254969875558790
        4.475194646764544 + 0.360864025160649 i
        4.475194646764544 - 0.360864025160649 i
        2.841379597921446
        2.036257422919012
        0.998619617166907

% Ezen a kicsiny 0.008 % -os változás esetén már komplex gyökök is fellépnek!

% Szimbolikus megoldás

        >> solve(expand(w))
ans =
 1
 2
 3
 4
 5
```

```
 6
 7


>> solve('z^7-28*z^6+322*z^5-1960*z^4+6769*z^3-13131*z^2+13068*z-5040')
ans =
                          .99861961716688339673327569978879
                          2.0362574229197134072571333588982
                          2.8413795979188513532587220628982
                          6.2549698755633181248422080581749
                          6.9183841929030118732123907860455
4.4751946467641109223481350170972+.36086402516932899962921403367002*i
4.4751946467641109223481350170972-.36086402516932899962921403367002*i

%  Ezen a helyzeten alapvetően a szimbolikus megoldás sem segít! Tehát itt az alapvető probléma az
%  output rendkívüli érzékenysége az input változására, azaz a rosszul kondicionáltság, amely
% párosulva  a kerekítési hibával fatális hibához vezethet!
```

- **5. példa**

Az alábbi L hosszúságú, $A = a^2$ keresztmetszetû (L >> a), közepén F erővel terhelt kéttámaszú tartó anyagának rugalmassági tényezőjét E, kell meghatározni az α szög mérése alapján!



Az α szöget az alábbi formula adja

$$\mathrm{tg}(\alpha) = \frac{F L^2}{16 E I}, \quad \text{ahol} \quad I = \frac{a^4}{12}$$

azaz

$$E(L, a, F, \alpha) = \frac{3 L^2}{4 a^4} F \cot(\alpha)$$

Az egyes jellemzők névleges értékei

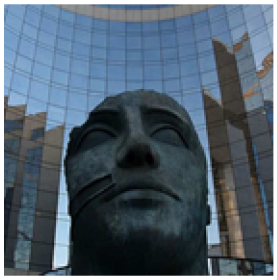
$$L_n = 100;\ a_n = 1;\ F_n = 120;\ \alpha_n = 0.017;$$

A mért jellemzők abszolút hibái,

$$\Delta L = 0.01;\ \Delta a = 0.01;\ \Delta F = 0.96;\ \Delta\alpha = 0.000085;$$

Határozzuk meg ezek alapján a rugalmassági tényezõ értékét és hibáját, valamint az E (L, a, F, α) függvény kondiciószámát!

```
%Szimbolikus számítás
>> syms E L F a alfa G eps
%A függvény a rugalmassági modulusra:
>> E=3*F*L^2/(4*a^4*tan(alfa))
 E =

3/4*F*L^2/a^4/tan(alfa)
% A "függvény" névleges értéke %szimbolikus esetben ezt kifejezésnek nevezzük
>> En = subs(E,{L,a,F,alfa},{100,1,120,0.017})
En =

  5.2936e+007

  %A gradiens vektor
  >> G=[abs(diff(E,L));abs(diff(E,a));abs(diff(E,F));abs(diff(E,alfa))]

G =

                3/2*abs(F*L/a^4/tan(alfa))
                3*abs(F*L^2/a^5/tan(alfa))
                3/4*abs(L^2/a^4/tan(alfa))
 3/4*abs(F*L^2/a^4/tan(alfa)^2*(1+tan(alfa)^2))

 % Az abszolút hiba
 >> h=subs(G',{L,a,F,alfa},{100,1,120,0.017})*[0.01;0.01;0.96;0.000085]
h =

  2.8163e+006

  % A relatív hiba
 >>rh=h/En
  rh =

    0.0532

% A függvény kondiciószáma
>> k=   [100 1 120 0.017]*subs(G,{L,a,F,alfa},{100,1,120,0.017})/En
  k =
     8.0002
```



2. Gyakorlat

Csonkítási hiba - Teljes hiba - Optimális lépésköz

6. példa

Határozzuk meg az

$$y = \exp(x)$$

kifejezés értékét az $x = 0.5$ helyen 4 értékes jegyre, 50 jegyű aritmetikával, alkalmazva az alábbi közelítést,

$$\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

```
>> digits (50);
>> x = 1/2;
>> y3 = vpa (1 + x + x^2/2)

y3 = 1.625000000000000000000000000000000000000000000000

>> y4 = vpa (1 + x + x^2/2 + x^3/factorial (3))

y4 = 1.645833333333333333333333333333333333333333333333

>> y5 = vpa (1 + x + x^2/2 + x^3/factorial (3) + x^4/factorial (4))

y5 = 1.648437500000000000000000000000000000000000000000

>> y6 = vpa (1 + x + x^2/2 + x^3/factorial (3) + x^4/factorial (4) + x^5/factorial (5))

y6 = 1.648697916666666666666666666666666666666666666667

% Az első 4 jegy már nem változik!

% A beépített függvénnyel
>> vpa (exp (x))
```

```
ans = 1.648721270700128194164335582172498106956481933593 8
```

7. példa

Határozzuk meg az exp (-x) kifejezés értékét az x = 8.3 helyen 4 értékes jegyre, az alábbi két formula szerint,

a) $y(x) = \exp(-x) = 1 - x + \frac{x^2}{2} - \frac{x^3}{3!} + - \ldots \rightarrow y_i = \sum_{j=0}^{i} (-1)^j \frac{x^j}{j!} = y_{i-1} + (-1)^i \frac{x^i}{i!}$

b) $y(x) = \exp(-x) = \frac{1}{1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \ldots} \rightarrow y_i = \frac{1}{\sum_{j=0}^{i} \frac{x^j}{j!}} \rightarrow \frac{1}{y_i} = \frac{1}{y_{i-1}} + \frac{x^i}{i!}$

A relatív hiba

a) $\epsilon = \frac{y_i - \exp(-x)}{\exp(-x)}$

```
% Írjunk m-file-t a két formulára

function y=app1(x,eps)
i=1;y=1;s=-x; ex=exp(-x);
while and(abs((ex-(y+s))/ex)>eps,i<100)
   y=y+s;
   i=i+1;
   s=(-1)^i*x^i/factorial(i);
  end
   y=[y,i];

function y=app2(x,eps)
i=1; yn =1; s = x; ex = exp(-x);
while and(abs((ex-1/(yn+s))/ex)>eps,i<100)
  yn=yn+s;
   i=i+1;
   s=x^i/factorial(i);
end
y=[1/yn,i];


format long

>> app1(8.3,0.00001)
ans =
  0.000248514147999  36.000000000000000

 >> app2(8.3,0.00001)
ans =
  0.000248521803 62  23.000000000000000

% A pontos érték
>> exp(-8.3)
```

```
ans =
   2.485168271079519e-004

  % Mint az várható volt, a b) algoritmus kedvezőbb, mivel kevesebb lépésszámmal éri el az előírt
  % hibakorlátot. Ok: az a) esetben a váltakozó előjel, amely közel azonos értékek kivonását jelenti
```

8. példa

Legyen adott az alábbi polinom,

$$f(x) = -0.1\,x^4 - 0.15\,x^3 - 0.5\,x^2 - 0.25\,x + 1.2$$

Határozzuk meg a deriváltat az $x = 0.5$ helyen a *differenciahányados* alapján $h = 0.5$ lépésközt választva. Becsüljük meg a közelítés hibáját!
Mivel

$$f(x+h) = f(x) + f'(x)\,h + f''(\xi)\frac{h^2}{2!}, \quad x \le \xi \le x + h$$

A differenciahányados

$$\Delta f = \frac{f(x+h) - f(x)}{h} = f'(x) + f''(\xi)\frac{h}{2!}$$

Az abszolút hiba becslése tehát

$$\Delta = \left|\frac{f(x+h) - f(x)}{h} - f'(x)\right| = \left|f''(\xi)\frac{h}{2!}\right|$$

```
% Állítsuk elő a függvényt, valamint annak első és második deriváltját szimbolikusan

>> syms f x

>> f = -0.1*x^4 - 0.15*x^3 - 0.5*x^2 - 0.25*x + 1.2;

>> diff (f, x)
ans =
     -2/5*x^3 - 9/20*x^2 - x - 1/4

>> diff (f, x, 2)
 ans =
       -6/5*x^2 - 9/10*x - 1

%A megfelelő anonymous függvények a numerikus számításhoz

>> f = @(x) - 0.1*x^4 - 0.15*x^3 - 0.5*x^2 - 0.25*x + 1.2;
>> df = @(x) - 2/5*x^3 - 9/20*x^2 - x - 1/4;
>> ddf = @(x) - 6/5*x^2 - 9/10*x - 1;

%A derivált közelítése a differenciahányadossal

 >> format long

>> h = 0.5;
>> delf = (f (0.5 + h) - f (0.5))/h
delf = -1.450000000000000
```

```
% A közelítés valóságos abszolút hibája

>> del = abs (delf - df (0.5))
del = 0.537500000000000

% A közelítés hibájának becslése a maradéktag alapján

>> R1 = @(u) h/2*ddf (u);

%mivel R1(u) az u =1 helyen veszi fel abszolútértékének maximumát, (rajzoljuk fel az R1(u)-t !)

>> dele = abs (R1 (1))
dele = 0.775000000000000
```

9. példa

Határozzuk meg a teljes hiba - azaz a kerekítési és a csonkítási hibák összegének - a lépésköztől (h) való függését az alábbi függvény deriváltjának x =2 helyen a differenciahányadossal történő becslése esetén,

$$y(x) = x^3$$

A derivált közelítő értéke:

$$\frac{(2+h)^3 - 2^3}{h}$$

A pontos érték,

$$3 \times 2^2 = 12$$

A teljes hiba,

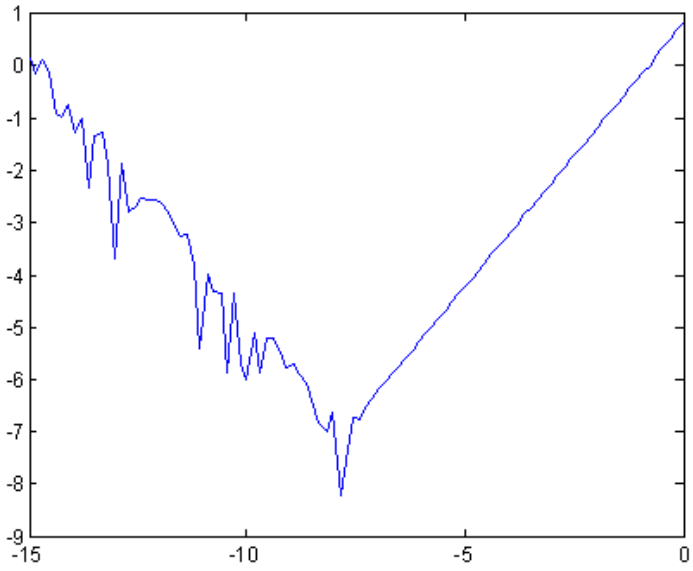
$$H(h) = \left| \frac{(2+h)^3 - 2^3}{h} - 12 \right|$$

A csonkítási hiba becslése a maradéktag alapján,

$$\Delta(h) = \left| \frac{h}{2} y''(x+\xi) \right| = \frac{h}{2} 6(x+\xi) \leq 3h(2+h)$$

```
% A teljes hiba

>> H = @(h) abs (((2 + h).^3 - 8)./h - 12) % Figyeljünk a pontokra, mert később, mint "listable"-ként használjuk!

% A csonkítási hiba becslése

>> estH = @(h) 3*(2 + h)*h;

% A teljes hiba és a csonkítási hiba számítása csökkenő lépésköz esetén

>> format long

>> h = 1 e - 4;

>> H (h)
ans =
    6.000100225662663e-004
```

```
>> estH (h)
ans =
   6.000000300000000 e - 004

% itt a csonkítási hiba domináns

>> h = 1 e - 8;

>> H (h)
ans =
    7.292965165106580 e - 008

>> estH (h)
ans =
     6.000000029999999 e - 008

h = 1 e - 12

H (h)
ans =
    0.001066806988092

estH (h)
ans =
    6.000000000003000 e - 012

% Itt már a teljes hiba jóval nagyobb, mint a becsült csonkítási  hiba, mivel itt már a kerekítési
% hiba domináns, miután igen kis h esetén  (2+h)^3 - 2^3  két közeli szám különbsége!

%Ábrázoljuk a teljes hibát a lépésköz függvényében log-log koordináta rendszerben

>> n = linspace (-15, 0, 100);

>> err = log10 (H (10.^n));
>> plot (n, err)
```



Látható, hogy nagyobb lépésköz esetén a csonkítási hiba, kisebb lépésköz esetén pedig a kerekítési hiba a domináns! Ennek a jelenségnek igen fontos szerepe van a differenciálegyenletek numerikus megoldásánál ! (Vajon miért?)

Megjegyzés

A Maple vagy a *Mathematica* rendszerek alkalmazása esetén, a fenti példánál nem lép fel kerekítési hiba, hiszen

$$\frac{(2+h)^3 - 2^3}{h} = 12 + 6h + h^2$$

amely átalakítást a fenti rendszerek automatikusan elvégeznek!

10. példa

Határozzuk meg az optimális lépésközt az $f(x) = \sin(x)$ függvény deriváltjának elsőrendű véges differenciával való közelítése esetén az $x = \pi/4$ helyen, ha 52 bites mantisszájú aritmetikával rendelkezünk!

A derivált elsőrendű véges differenciával történő közelítése a következő:

$$\delta(x, h) = \frac{f(x+h) - f(x)}{h},$$

Mivel

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + \ldots = f(x) + f'(x)h + f''(\xi)\frac{h^2}{2}$$

ahol

$$x \le \xi \le x + h$$

ezzel

$$\delta(x, h) = f'(x) + f''(\xi)\frac{h}{2} = f'(x) + K_1\frac{h}{2}$$

azaz a csonkítási hiba arányos a h lépésközzel! Azonban a számítás során kerekítési hiba is fellép, ezért a teljes hiba (csonkítási + kerekítési hiba (Δ)),

$$\delta_T(x, h) = \frac{(f(x+h) + \Delta_1) - (f(x) + \Delta_0)}{h} = f'(x) + \frac{\Delta_1 - \Delta_0}{h} + K_1 \frac{h}{2}$$

A kerekítési hiba tehát fordítottan arányos a lépésközzel! Becsüljük a teljes hibát

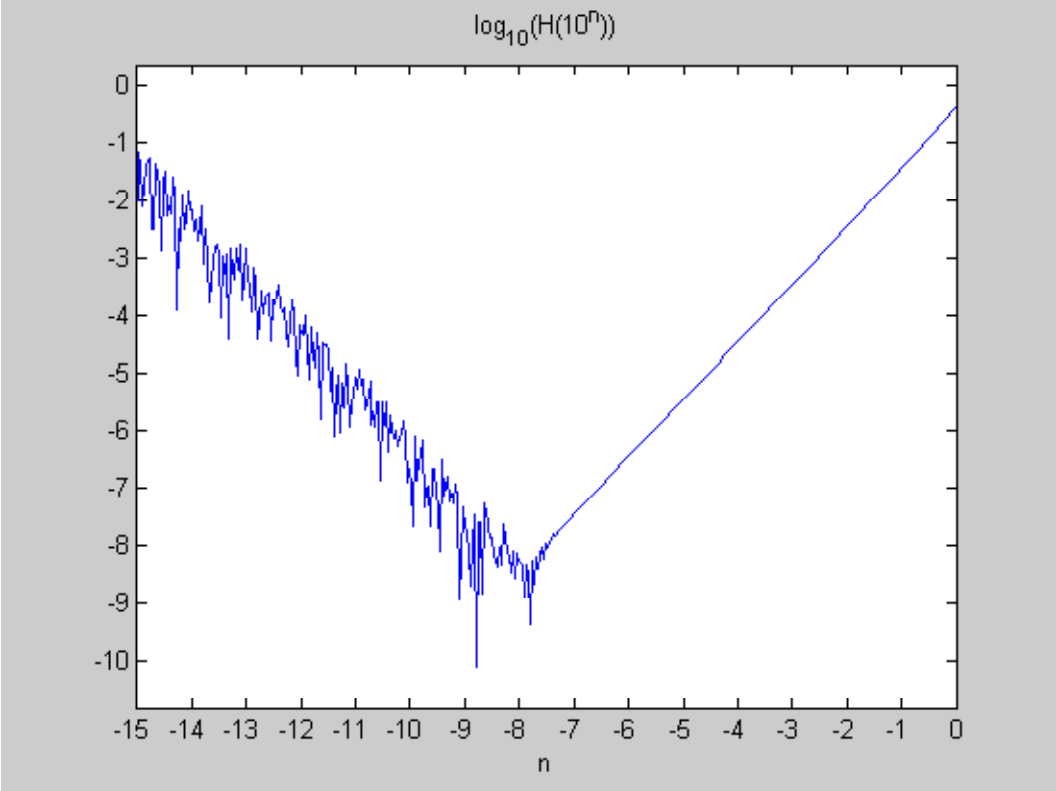
$$|f'(x) - \delta_T(x, h)| \leq \frac{|\Delta_1| + |\Delta_0|}{h} + |K_1| \frac{h}{2} \leq 2 \frac{\epsilon_{ru}}{h} + |K_1| \frac{h}{2}$$

ahol ϵ_{ru} a kerekítési egység, amely a gépi epszilon fele.

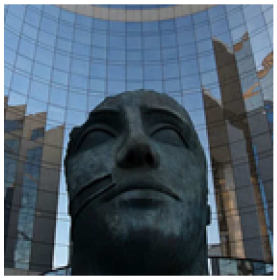
A jobb oldal minimumát adó lépésköz,

$$h_{opt} = 2 \sqrt{\frac{\epsilon_{ru}}{|K_1|}}$$

```
>> clear all

% a Matlaban a gépi epszilon értékét az eps változó tartalmazza, így a kerekítési egység

>> delta = vpa (eps/2, 20)
delta =
      .11102230246251565404 e - 15

%A második derivált

>> K1 = -sin (pi/4)
 K1 =
    -0.7071

 % Az optimális lépésköz

 >> hopt = vpa (2*sqrt (delta/abs (K1)), 20)
 hopt =
      .25060666062048510998e-7

>> log10(hopt)
 ans =
 -7.6010073905155251791714301102974

%A teljes hiba, mint a lépésköz függvénye

 >> H = @(h) abs ((sin (pi/4 + h) - sin (pi/4))/h - cos (pi/4));

% log-log koordináta rendszerben

>> Hlog = @(n) log10 (H (10^n));

 >> ezplot (Hlog, [-15, 0])
```



Az ábra alapján is közelítően a minimum helye ≈ -7.6 = log (h_{opt})!



3. Gyakorlat

Megoldás létezése és egyértelműsége - direkt módszerek - nulltér

11. példa

Vizsgáljuk meg az alábbi egyenletrendszerek megoldásának létezését és a megoldás egyértelműségét!

1. típus

$$A_1 = \begin{pmatrix} 52 & -60 \\ 39 & -25 \end{pmatrix}, \quad b_1 = \begin{pmatrix} 260 \\ 325 \end{pmatrix}$$

```
% 1. eset

>> A1 = [52 - 60; 39 - 25];
>>  b1 = [260 325];

% A kibővített mátrix

>> A1b1 = [52 - 60 260; 39 - 25 325]

A1b1 = 52 - 60 260
          39 - 25 325

% Ezek rangjai

>> rank (A1)
ans = 2

>> rank (A1b1)
ans = 2

% Ezek egyezőek és rank(A)= n = 2, tehát van egyértelmű megoldás és természetesen egyszerűbben

>> det(A1)
ans =
       1040
```

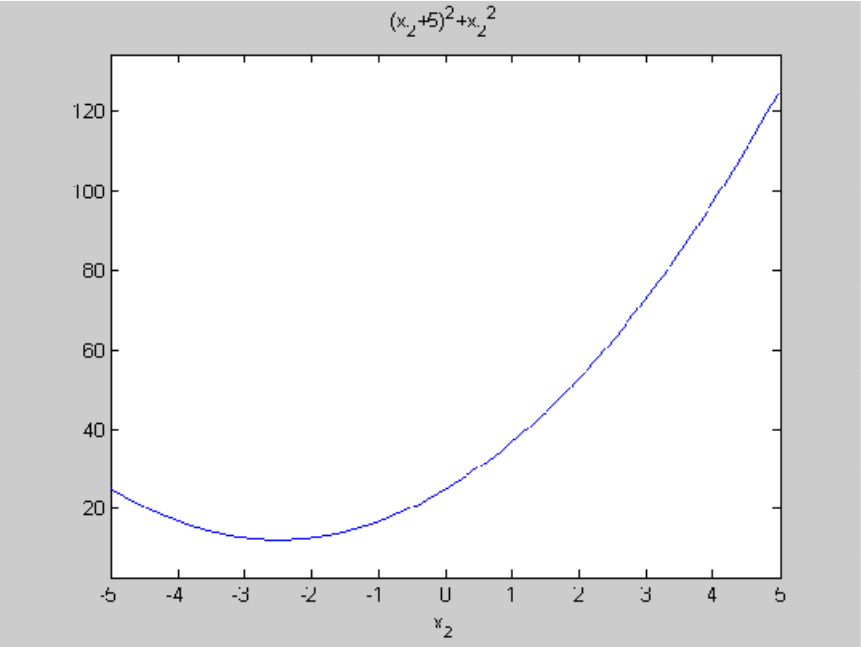
```
% A megoldás
 >> x12 = linsolve (A1, b1')

x12 = 12.5000
       6.5000

% A megoldás ellenőrzése
>> A1 (1 : 2, 1)*x12 (1) + A1 (1 : 2, 2)*x12 (2)
ans = 260
      325
```

2. típus

$$A_2 = \begin{pmatrix} 52 & -52 \\ 39 & -39 \end{pmatrix}, \quad b_2 = \begin{pmatrix} 260 \\ 195 \end{pmatrix}$$

```
% 2. eset

clc; clear all

>> A2 = [52 - 52; 39 - 39];
>> b2 = [260 195];

>> det (A2)
ans = 0

%Tehát az biztos, hogy nincs egyértelmű megoldás!

>> rank (A2)
ans = 1

% azaz az A2 mátrix rang deficites, mivel  A21 = - A22
>> A21 = A2 (1 : 2, 1)
A21 = 52
      39

>> A22 = A2 (1 : 2, 2)
A22 = -52
      -39

%  A kibővített mátrix
>> A2b2 = [52 - 52 260; 39 - 39 195];

%vagy  A2b2=[A2,b2'];

rank (A2b2)
ans = 1

% tehát létezik megoldás csak végtelen sok mivel

% b2 = x1 A21 + x2 A22 = (x1 - x2) A21
```

```
% tehát az  "egyenletrendszer"  mátrixa a két megoldás különbségére

%M = I A21 - azaz M= egységmátrix * A21

>> M = [52 0; 0 39]
M = 52 0
        0 39

% A megoldás nyilván   dx = b21/52 = b22/ 39 = 5

% vagy

dx = linsolve (M, b2')
dx = 5
       5

%Tehát x1 = x2 +5

% Azt a megoldást választjuk, amelynek normája minimális

>> f=@(x2)(x2+5)^2+x2^2;
>> ezplot(f,[-5,5])

>>x2= fminsearch(f,5)
x2 =
   -2.5000
>> x1=x2+5
x1 =c
    2.5000
```



% További lehetőség, hogy *elhagyjuk a redundáns egyenleteknek megfelelő sorokat* az A2 és b2 -ből, %pl most a másodikat, és alkalmazzuk a megkötéses minimalizálásra kapott közvetlen eredményt

```
% azaz  x = A^T (A A^T)^-1 b

 >> A=[52,-52];b=[260];
>> x=A'*inv(A*A')*b
x =
    2.5000
   -2.5000

%Közvetlen megoldás a pszeudoinverz segítségével
>> x2=pinv(A2)*b2'
x2 =
    2.5000
   -2.5000
```

3. típus

$A_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{pmatrix}$, $b_3 = \begin{pmatrix} 6 \\ 9 \end{pmatrix}$

```
% 3.eset

clc; clear all

>> A3=[1 2 3; 2 3 4];
>> b3=[6 9];

>> A3b3=[1 2 3 6; 2 3 4 9];

>> rank(A3)
ans =
     2

% Valóban, hiszen

>> A3(1:2,1)+A3(1:2,3)==2*A3(1:2,2)
ans =
     1
     1

% A kibővített mátrix rangja

>> rank(A3b3)
ans =
     2

% azaz létezik megoldás, de nem tudjuk egyértelmű-e? Nyilván nem, hiszen

>> mn=size(A3)
mn =
     2     3

n=mn(2)
```

```
n =
     3
%vagyis az ismeretlenek száma nagyobb, mint az egyenletek száma: alulhatározott rendszer

rank(A3)<n
ans =
       1

%Most azonban det (A3*A3') = 6 ≠ 0, ezért a minimál normájú megoldás

>> x3=A3'*inv(A3*A3')*b3'
x3 =
    1.0000
    1.0000
    1.0000

% vagy a pszeudoinverzzel

 >> x3=pinv(A3)*b3'
x3 =
    1.0000
    1.0000
    1.0000
```

4. típus

$$A_4 = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 4 & -1 \end{pmatrix}, \qquad b_4 = \begin{pmatrix} 5 \\ 7 \\ 2 \end{pmatrix}$$

```
% 4.eset

clc; clear all

>>A4=[1 2;2 3;4 -1];
>>b4=[5 7 2];

>> rank(A4)
ans =
     2

>> A4b4=[1 2 5;2 3 7; 4 -1 2]

rank(A4b4)
ans =
     3

% azaz nem létezik megoldás

%A rendszer túlhatározott,

>> mn= size(A4);
```

```
>> n=mn(2)
n =
     2
n==rank(A4)
ans =
     1

%de a mátrix rangja egyező az ismeretlenek számával,
% megoldásként az egyenletek hibáinak négyzetösszegét minimalizáljuk

>> f=@(x)norm(A4*[x(1);x(2)]-b4')

>> fminsearch(f,[1,1])
ans =
    0.9424    1.8021

% Direkt megoldási módszerek:

x4=linsolve(A4,b4')
x4 =
    0.9424
    1.8022

x4=A4\b4'
x4 =
    0.9424
    1.8022

x4=inv(A4'*A4)*A4'*b4'
x4 =
    0.9424
    1.8022

x4=pinv(A4)*b4'
x4 =
    0.9424
    1.8022

% Természetesen ez a "megoldás" nem elégíti ki az egyenletrendszert

norm(A4*x4-b4')
ans =
    0.5398
```

5. típus

$$A_5 = \begin{pmatrix} 52 & -52 \\ 39 & -39 \end{pmatrix}, \qquad b_5 = \begin{pmatrix} 260 \\ 325 \end{pmatrix}$$

```
% 5. eset

clc; clear all
```

```
>> A5 = [52 - 52; 39 - 39];
>> b5 = [260 325];

>> det (A5)
ans = 0

%Tehát az biztos, hogy nincs egyértelmű megoldás!

>> rank (A5)
ans = 1

% azaz mint a 2. esetnél az A5 mátrix rangja deficites, mivel  A51 = - A52


%  A kibővített mátrix
>> A5b5 = [52 - 52 260; 39 - 39 325];

rank (A5b5)
ans = 2

% tehát nem létezik megoldás csak a legkisebb négyzetek értelmében
% azonban most, eltérően a 4. esettől és egyezően a 2. esettel A5'*A5 szinguláris!

>> det(A5'*A5)
ans =
    0

% A feladat azonban visszavezethető az alulhatározott esetre (3.eset)

  >> A=A5'*A5
A =
       4225      -4225
      -4225       4225

>> b=A5'*b5'
b =
      26195
     -26195

% most a mátrix és a kibővített mátrix rangja egyenlő,

>> Ab=[4225 -4225 26195;-4225 4225 -26195]
Ab =
       4225      -4225       26195
      -4225       4225      -26195

>> rank(A)
ans =
    1

>> rank(Ab)
```

```
ans =
     1
% azaz   rank(A) < n = 2

% A megoldás tehát, elhagyva a redundáns egyenletet,

>> AA=A(1,1:2)
AA =
       4225       -4225

>> bb=b(1)
bb =
       26195

>> AA'*inv(AA*AA')*bb
ans =

    3.1000
   -3.1000

% Tehát hagyományos értelemben nincs megoldás, de a legkisebb négyzetek értelmében
%viszont végtelen sok megoldás van, a fenti ezek közül a legkisebb normájút választjuk!

% Ezt jól megvilágítja a következő megoldási lehetőség:

% A rezidium - a hibák négyzetösszege

> res=(-260 + 52*x1 - 52*x2)^2 + (-325 + 39*x1 - 39*x2)^2
 res =
    (-260+52*x1-52*x2)^2+(-325+39*x1-39*x2)^2

% Minimalizálásának szükséges feltétele a parcális deriváltak eltűnése, azaz

>> eq1=diff(res,x1)
eq1 =
  -52390+8450*x1-8450*x2

 >> eq2=diff(res,x2)
eq2 =
 52390-8450*x1+8450*x2

 % De ez egy redundáns lineáris rendszer, amelynek végtelen sok megoldása van,
 % mint a 2. esetben.
 % Megoldása megkötéses minimalizálása a megoldás normájának, ahol a megkötés
 % most az egyik egyenlet:

>> A=[-8450,8450]; b=[52390]
b =
       52390

>> x=A'*inv(A*A')*b
x =
```

```
   -3.1000
    3.1000

% Mindez egyszerűbb a pszeudoinverz alkalmazásával
    >> pinv(A5)*b5'
ans =
    3.1000
   -3.1000
```

12. példa

Határozzuk meg az A x = 0 homogén egyenletrendszer triviálistól eltérő megoldását, ha van!

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

```
>> A = [0 1 0 0 0; 0 0 0 1 1; 0 1 0 1 1; 0 0 0 0 0; 1 1 0 0 1];

% A szükséges feltétel

>> det (A)
ans = 0

 >> mn = size (A)
 mn = 5 5

 >> rank (A)
 ans = 3

 %A nulltér bázisvektorainak száma: cols(A) - rank(A)

  >> mn (1) - rank (A)
  ans = 2

  % A bázisvektorok
  >> V = null (A)
  V = 0.5772    0.0130
        0.0000    0.0000
     - 0.0226     0.9997
       0.5772     0.0130
     - 0.5772   - 0.0130

    >> v1 = V (1 : 5, 1)
     v1 = 0.5772
          0.0000
        - 0.0226
          0.5772
        - 0.5772

      >> v2 = V (1 : 5, 2)
```

```
      v2 = 0.0130
           0.0000
           0.9997
           0.0130
         - 0.0130

%    Ezek valóban megoldások
      >> A*v1
       ans = 1.0 e - 015*
          0.2170
         - 0.3331
         - 0.1110
          0
         0.1110

          >> norm (ans)
          ans = 4.2740 e - 016

          >> norm (A*v2)
          ans = 3.2056 e - 016

 % természetesen ezek lineáris kombinációi is megoldások:  α v1 + β v2, azaz végtelen sok  megoldás van.
```

13. példa

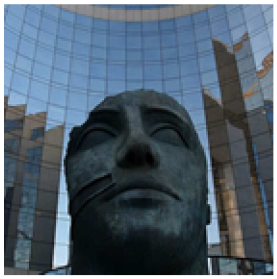
Határozzuk meg az alábbi A mátrix sajátértékeit és sajátvektorait!

$$A = \begin{pmatrix} 1 & 0 & -4 \\ 0 & 5 & 4 \\ -4 & 4 & 3 \end{pmatrix}$$

```
>> clear all
>> syms A lam

>> A = [1 0 - 4; 0 5 4; -4 4 3];

%A sajátértékek meghatározása:

>> expand (det (A - eye (3)*lam))
ans = -81 + 9*lam + 9*lam^2 - lam^3

>> c = [-1 9 9 - 81];

>> lam = roots (c)
lam = 9.0000
      -3.0000
       3.0000

%Határozzuk meg a második sajátértékhez tartozó sajátvektort
% Ez az A2 = A -  (-3) *I mátrixú homogén egyenlet megoldását jelenti,azaz

>> A2 = A - eye (3)*lam (2)
A2 = 4.0000    0          - 4.0000
```

```
         0         8.0000    4.0000
    -4.0000    4.0000    6.0000

>> det(A2)
ans =
     0

% A bázis vektorok száma

3-rank(A2)
ans =
     1

  % A bázisvektor
>> v2=null(A2)

v2 =

   -0.6667
    0.3333
   -0.6667

 % Ez tehát a második sajátvektor. (Persze ennek λ-szorosa is az, λ*v2)
 % Ellenőrzés

  >> A*v2-(-3)*v2

ans =

  1.0e-015 *

    0.8882
   -0.8882
   -0.8882

  >> norm(ans)

ans =

  1.5384e-015

  % A beépített függvények segítségével

  >> [V,D]=eig(A);
>> V
V =
   -0.6667    0.6667   -0.3333
    0.3333    0.6667    0.6667
   -0.6667   -0.3333    0.6667

>> D
D =
```

```
   -3.0000         0         0
         0    3.0000         0
         0         0    9.0000
% A sajátértékek növekvő sorrendben rendezettek.
```



4. Gyakorlat

Felbontások (LU, Cholesky, QR és SVD)

14. példa

Egy építkezéshez 4800 m^3 homok, 5810 m^3 kavics és 5690 m^3 finomkavics szükséges.

Három bányából szállítjuk az anyagot, amelyek összetétele bányánként a következő:

Bánya	Homok %	Finomkavics %	Durvakavics %
1	52	30	18
2	20	50	30
3	25	20	55

Melyik bányából, mennyi anyagot kell szállítani az építkezés igényének kielégítésére? Legyenek az egyes bányákból szállított anyagmennyiségek rendre, x_1, x_2 és x_3 és tegyük fel, hogy $x_i \geq 0$, i =1, 2, 3.

Az anyagmérleg a homokra:

$$4800 = 0.52\,x_1 + 0.20\,x_2 + 0.25\,x_3$$

a finomkavicsra

$$5960 = 0.18\,x_1 + 0.30\,x_2 + 0.55\,x_3$$

a durvakavicsra

$$5810 = 0.30\,x_1 + 0.50\,x_2 + 0.20\,x_3$$

Alkalmazzunk LU felbontást!

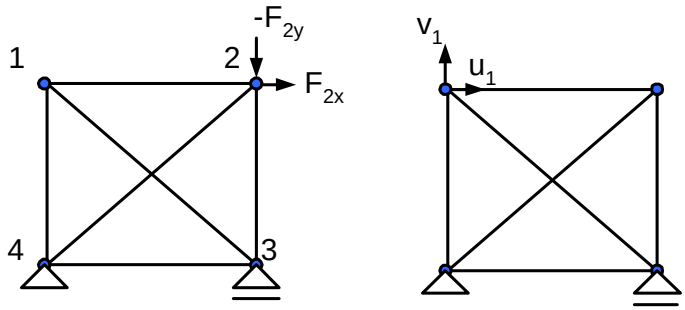
```
>> M = [0.52 0.2 0.25; 0.18 0.3 0.55; 0.3 0.5 0.2]

 M = 0.5200  0.2000  0.2500
     0.1800  0.3000  0.5500
     0.3000  0.5000  0.2000

 % Az alsó (L), felső (U)  háromszögmátrixok és permutációs mátrix (P)

 >> [L U P] = lu (M)

L = 1.0000      0           0
```

```
      0.5769    1.0000    0
      0.3462    0.6000    1.0000


U = 0.5200    0.2000    0.2500
      0         0.3846    0.0558
      0         0         0.4300


P =  1 0 0
      0 0 1
      0 1 0

>> L*U-P*M
ans =
  1.0e-016 *
      0         0    0
      0         0    0
     -0.2776    0    0

 >> norm(ans)
ans =
  2.7756e-017

 % Az első egyenlet L y = P b megoldása:
 % Annak érdekében, hogy a linsolve kihasználja azt, hogy L alsó háromszögmátrix

 >> opts.UT=false;
 >>opts.LT=true;

>> y=linsolve(L,P*b,opts)
y =
  1.0e+003 *
    4.8000
    3.1908
    2.2340

% A második egyenlet U x = y megoldása

>> opts.UT=true;
>> opts.LT=false;

>> x=linsolve(U,y,opts)
x =
  1.0e+003 *
    3.8320
    7.5427
    5.1953

% Ellenőrzés
>> norm(M*x-b)
ans =
```

```
    0

 % Természetesen a linsolve egylépésben is megadja a megoldást az LU felbontást alkalmazva:

 >> linsolve(M,b)
ans =
  1.0e+003 *
    3.8320
    7.5427
    5.1953
```

15. példa

Tekintsük az alábbi rácsos tartót:



A csomópontok elmozdulását leíró egyenletrendszer, $A\,x = F$, ahol

$$A = \begin{pmatrix} 1.35 & -0.35 & -1 & 0 & -0.35 \\ -0.35 & 1.35 & 0 & 0 & 0.35 \\ -1 & 0 & 1.35 & 0.35 & 0 \\ 0 & 0 & 0.35 & 1.35 & 0 \\ -0.35 & 0.35 & 0 & 0 & 1.35 \end{pmatrix} 2.8 \times 10^5; \quad x = \begin{pmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \end{pmatrix} \quad \text{és} \quad F = \begin{pmatrix} 0 \\ 0 \\ 70 \\ 50 \\ 0 \end{pmatrix}$$

Oldjuk meg az egyenletrenszert, kihasználva, hogy az A mátrix pozitív definit és szimmetrikus.

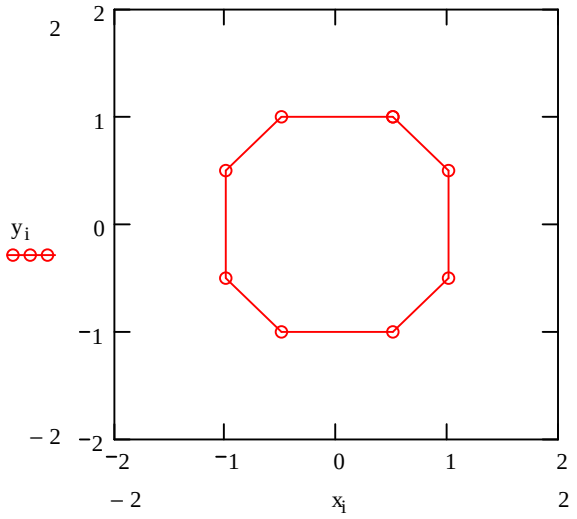
```
>> clear all

  >> A = [1.35 - 0.35 - 1 0 - 0.35;
    -0.35 1.35 0 0 0.35;
    -1 0 1.35 0.35 0;
    0 0 0.35 1.35 0;
    -0.35 0.35 0 0 1.35]

  A = 1.3500    - 0.3500    - 1.0000    0         - 0.3500
      -0.3500    1.3500     0           0          0.3500
      -1.0000    0          1.3500      0.3500     0
       0         0          0.3500      1.3500     0
  -0.3500        0.3500     0           0          1.3500

% >> A=A*2.8*1e5;

% A mátrix szimmetrikus

  >> norm (A - A')
```

```
   ans = 0

% és pozitív definit

   >> [V D] = eig (A);
>> diag (D)
ans =
  1.0e+005 *
    0.5973
    2.8000
    3.6039
    4.7600
    7.1388

% azaz, létezik Cholesky-felbontása

>> L=chol(A)

L =
  614.8170  -159.3970  -455.4200        0  -159.3970
         0   593.7951  -122.2519        0   122.2519
         0         0   394.5213   248.4023  -146.1190
         0         0         0   562.4023    64.5380
         0         0         0         0   558.6870

%ellenőrzés

>> LT=L';
>> norm(LT*L-A)
ans =
  6.2446e-011

% Az első egyenlet megoldása: LT y = b

  >> opts.UT=false;
>> opts.LT=true;

>> y=linsolve(LT,b)
y =
         0
         0
    0.1774
    0.0105
    0.0452

 % A második egyenlet: L x = y

 >> opts.UT=true;
>> opts.LT=false;

>> x=linsolve(L,y)
x =
```

```
  1.0e-003 *
    0.3929
    0.0809
    0.4737
    0.0095
    0.0809

% egy lépésben

>> x=linsolve(A,b)
x =
  1.0e-003 *
    0.3929
    0.0809
    0.4737
    0.0095
    0.0809

% ellenőrzés
>> norm(A*x-b)
ans =
  3.3704e-014
```

16. példa

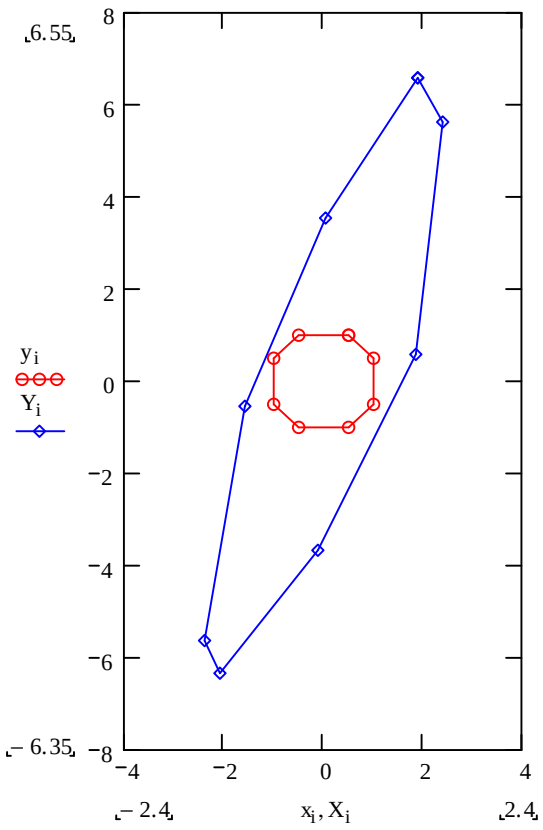
Adottak a síkban az alábbi nyolcszög koordinátái,



$$x = (0.5,\ 1,\ 1,\ 0.5,\ -0.5,\ -1,\ -1,\ -0.5,\ 0.5)^T$$

$$y = (1,\ \ 0.5,\ -0.5,\ -1,\ \ -1,\ -0.5,\ 0.5,\ \ 1,\ \ 1)^T$$

Az alakzat transzformált képének mért koordinátái a következők,



$$X = (1.9,\ 2.4,\ 1.84,\ -0.1,\ -2.1,\ -2.4,\ -1.6,\ 0.05,\ 1.9)^T$$

$$Y = (6.55,\ 5.59,\ 0.573,\ -3.7,\ -6.35,\ -5.65,\ -0.55,\ 3.52,\ 6.55)^T$$

A transzformáció feltételezett alakja:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Határozzuk meg a transzformáció paramétereit!

Minden pontra 2 egyenlet írható fel:

$$x_i = a_{11} X_i + a_{12} Y_i + b_1$$

$$y_i = a_{21} X_i + a_{22} Y_i + b_2$$

i = 1, ..., 8

Az egyenletrendszer mátrixos alakja:

$$M\, c = d$$

ahol

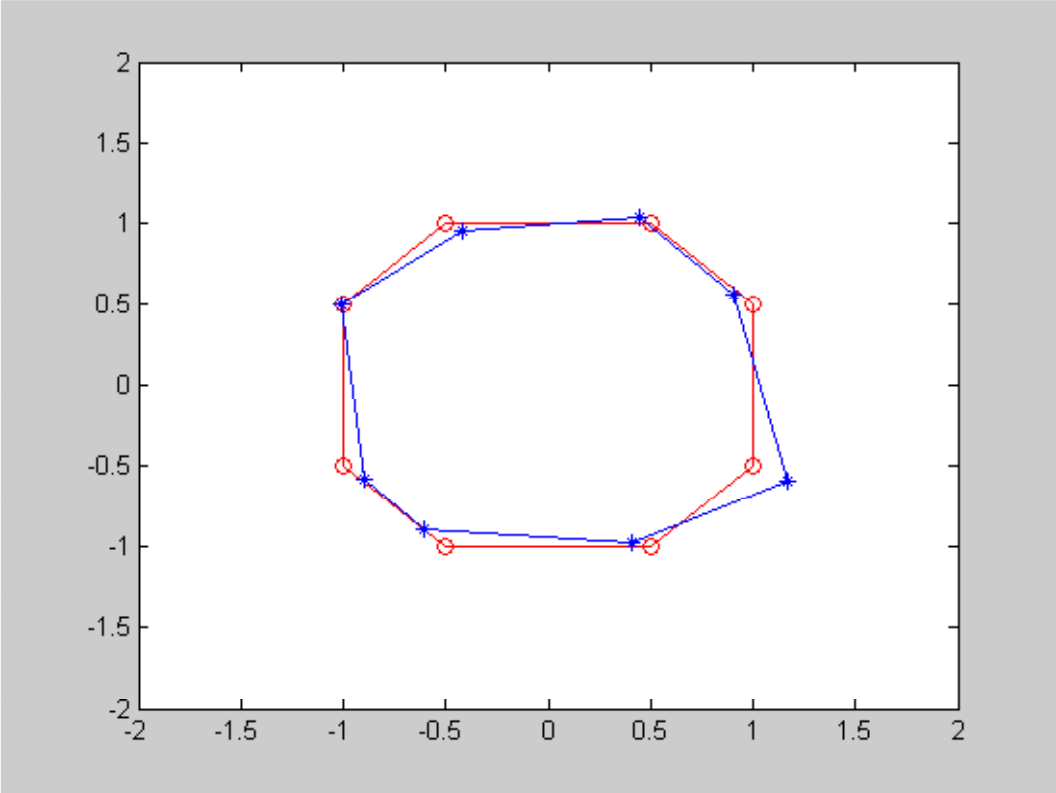
$$M = \begin{pmatrix} X_i & Y_i & 0 & 0 & 1 & 0 \\ . & . & . & . & . & . \\ 0 & 0 & X_i & Y_i & 0 & 1 \\ . & . & . & . & . & . \end{pmatrix},\quad c = \begin{pmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ b_1 \\ b_2 \end{pmatrix},\quad d = \begin{pmatrix} x_i \\ . \\ y_i \\ . \end{pmatrix}$$

```
>> clear all
```

```
>> x = [0.5 1 1 0.5 - 0.5 - 1. - 1. - 0.5 0.5]';
>> y = [1 0.5 - 0.5 - 1 - 1 - 0.5 0.5 1 1]';
>> X = [1.9 2.4 1.84 - 0.1 - 2.1 - 2.4 - 1.6 0.05 1.9]';
>> Y = [6.55 5.59 0.573 - 3.7 - 6.35 - 5.65 - 0.55 3.52 6.55]';

%Megjegyzés: a vektorok a későbbi rajzolás miatt 1-el hosszabbak, de ez nem zavaró!
% Az együtthatómátrix előállítása

>> for i = 1 : 8
M (i, 1 : 6) = [X (i), Y (i), 0, 0, 1, 0];
M (i + 8, 1 : 6) = [0, 0, X (i), Y (i), 0, 1];
d (i) = x (i);
d (i + 8) = y (i);
end

>> M

M =

   1.9000   6.5500        0        0   1.0000        0
   2.4000   5.5900        0        0   1.0000        0
   1.8400   0.5730        0        0   1.0000        0
  -0.1000  -3.7000        0        0   1.0000        0
  -2.1000  -6.3500        0        0   1.0000        0
  -2.4000  -5.6500        0        0   1.0000        0
  -1.6000  -0.5500        0        0   1.0000        0
   0.0500   3.5200        0        0   1.0000        0
        0        0   1.9000   6.5500        0   1.0000
        0        0   2.4000   5.5900        0   1.0000
        0        0   1.8400   0.5730        0   1.0000
        0        0  -0.1000  -3.7000        0   1.0000
        0        0  -2.1000  -6.3500        0   1.0000
        0        0  -2.4000  -5.6500        0   1.0000
        0        0  -1.6000  -0.5500        0   1.0000
        0        0   0.0500   3.5200        0   1.0000


>> d'
ans =
   0.5000
   1.0000
   1.0000
   0.5000
  -0.5000
  -1.0000
  -1.0000
  -0.5000
   1.0000
   0.5000
  -0.5000
  -1.0000
  -1.0000
```

```
   -0.5000
    0.5000
    1.0000

% Ez egy túlhatározott egyenletrendszer
>> rank(M)
ans =
     6

>> rank([M,d'])
ans =
     7

% Alkalmazható a QR felbontás: M = Q*R

>> [Q,R]=qr(M);

>> norm(M-Q*R)
ans =
  8.6926e-015

% ahol R egy felső háromszögmátrix
>> R

R =

   -5.0496   -10.9388         0         0    0.0020         0
         0    7.3109         0         0    0.0006         0
         0         0   -5.0496   -10.9388         0    0.0020
         0         0         0    -7.3109   -0.0000   -0.0006
         0         0         0         0   -2.8284   -0.0000
         0         0         0         0         0    2.8284
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0
         0         0         0         0         0         0

  % Q egy ortonormált mátrix
 >> norm(Q'*Q-eye(16))
ans =
  1.1600e-015

% A jobboldal előállítása
>> B=Q'*d';

% A megoldás
```

```
>>  opts.UT=true;
>>  opts.LT=false;
>> c=linsolve(R,B)

c =
    0.6750
   -0.1275
   -0.4070
    0.2753
    0.0006
    0.0001

%Megoldás lehetséges még
>> c=linsolve(M,d')
c =
    0.6750
   -0.1275
   -0.4070
    0.2753
    0.0006
    0.0001

%vagy

>> c=inv(M'*M)*M'*d'
c =
    0.6750
   -0.1275
   -0.4070
    0.2753
    0.0006
    0.0001

 % a transzformációs modell illeszkedésének szemléltetése
 % Az eredeti nyolcszög
  >> plot(x,y,'ro-')

% A tárgykoordinátákból visszatranszformált nyolcszög
>> for i=1:9
xt(i)=[c(1),c(2)]*[X(i),Y(i)]'+c(5);
yt(i)=[c(3),c(4)]*[X(i),Y(i)]'+c(6);
end

>> hold on
>> plot(xt,yt,'b*-')
```



17. példa

Határozzuk meg a 3. gyakorlat 1.példájának 5.esetében szereplő

$$A_5 = \begin{pmatrix} 52 & -52 \\ 39 & -39 \end{pmatrix}$$

mátrix pszeudóinverzét a mátrix szinguláris értékek szerinti felbontásával!

```
>> clear all
>> A5 = [52 - 52; 39 - 39]

 >> [U S V] = svd (A5)

   U = -0.8000   - 0.6000
        -0.6000    0.8000

   S = 91.9239      0
        0              0.0000

   V = -0.7071     0.7071
         0.7071    0.7071

  % ellenőrzés
   >> norm (A5 - U*S*V')
   ans =
    1.4211 e - 014

 %továbbá
     >> U'*U
```

```
    ans =
     1.0000   0.0000
     0.0000   1.0000

 >> V'*V
 ans =
 1.0000   0
 0          1.0000

 %Az A5 pszeudoinverze

 >> invA5 = V*diag ([1/S (1, 1); 0])*U'
 invA5 =
   0.0062    0.0046
 - 0.0062   - 0.0046

 %A beépített függvénnyel
 >> pinv (A5)
 ans =
    0.0062   0.0046
  - 0.0062  - 0.0046

 % Az is igaz, hogy az A5 szinguláris értékei megegyeznek az A5'*A5 mátrix
 % sajátértékeinek   négyzetgyökével

 >> [V D] = eig (A5'*A5);

>> diag(D)
ans =
         0
      8450

>> sqrt (ans)
ans =
       0
  91.9239

 % Mint azt korábban is láttuk (3.gyakorlat 13. példa) az eig függvény a sajátértékeket nagyság szerint növekvő
sorrendben adja.
```

18. példa

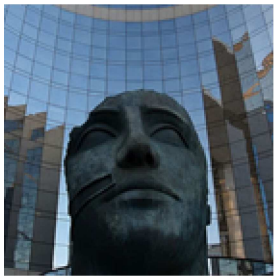
Határozzuk meg az alábbi A mátrix sajátértékeit és sajátvektorait az A mátrix szinguláris értékek szerinti felbontásával!

$$A = \begin{pmatrix} 1 & 0 & -4 \\ 0 & 5 & 4 \\ -4 & 4 & 3 \end{pmatrix}$$

A 3. gyakorlat 13. példájához képest az eltérés csak annyi, hogy egy adott sajátértékhez tartozó homogén egyenlet mátrixának nullterét most az SVD segítségével állítjuk elő.

```
  >> clear all
  >> syms A lam
```

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

>> expand (det (A - eye (3)*lam))
ans =
-81 + 9*lam + 9*lam^2 - lam^3

>> c = [-1 9 9 - 81];
>> lam = roots (c)
lam = 9.0000
        -3.0000
         3.0000

>> A2 = A - eye (3)*lam (2)
A2 =
4.0000    0         - 4.0000
0         8.0000    4.0000
-4.0000   4.0000    6.0000

 %A szinguláris értékek szerinti felbontás
>> [U S V] = svd (A2);
>> S
S = 12.0000    0            0
      0        6.0000       0
      0        0            0.0000
>> V
V = -0.3333      - 0.6667      - 0.6667
      0.6667     - 0.6667        0.3333
      0.6667       0.3333      - 0.6667

% A S -ben csak a 3. szinguláris érték zérus, ezért a nulltér egy bázisvektorral rendelkezik.
% ez a V-beli neki megfelelő,azaz 3. oszlopvektor

>> v1 = V (1 : 3, 3)
v1 = -0.6667
       0.3333
      -0.6667

% Persze a beépített függvény is ezt adta,
      >> null(A2)
ans =
  -0.6667
   0.3333
  -0.6667
```



5. Gyakorlat

Ritka mátrixok megoldása - rosszulkondicionált egyenletrendszer

19. példa

Oldjuk meg az alábbi egyenletrendszert

$$A = \begin{pmatrix} -4 & 4 & 0 & 0 & 0 \\ 4 & -8 & 4 & 0 & 0 \\ 0 & 4 & -8 & 4 & 0 \\ 0 & 0 & 4 & -8 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}; b = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

A rendszer mátrixa egy ún. sávmátrix, amely a ritka mátrixok csoportjába tartozik.

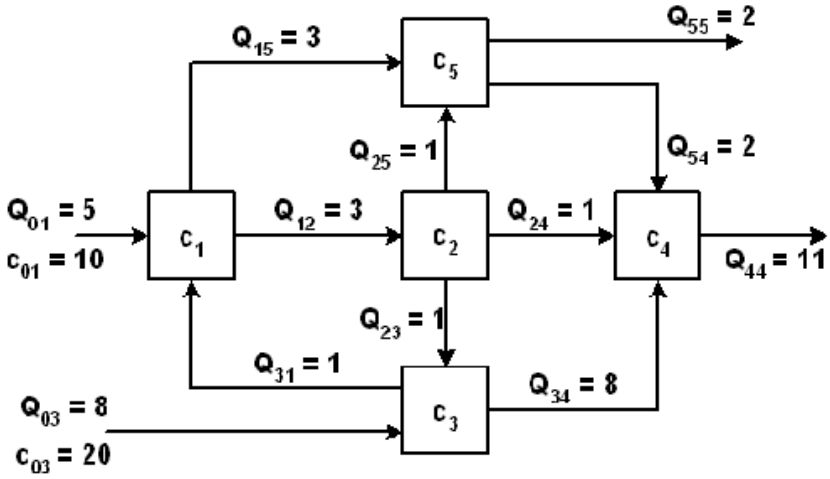
```
>> clear all
>> A = [-4 4 0 0 0; 4 - 8 4 0 0; 0 4 - 8 4 0; 0 0 4 - 8 4; 0 0 0 0 1]

A = -4   4   0   0   0
     4  - 8   4   0   0
     0   4  - 8   4   0
     0   0   4  - 8   4
     0   0   0   0   1
% a nem-nulla elemek száma
>> nonzero = nnz (A)
nonzero = 12

% az összes elemek száma
>> total = prod (size (A))
total = 25

% a mátrix sűrűsége
>> density = nonzero/total
density = 0.4800

% ritka mátrixok tárolási formája
>> AS=sparse(A)

AS =
```

```
   (1,1)       -4
   (2,1)        4
   (1,2)        4
   (2,2)       -8
   (3,2)        4
   (2,3)        4
   (3,3)       -8
   (4,3)        4
   (3,4)        4
   (4,4)       -8
   (4,5)        4
   (5,5)        1

  % ezzel az adatstruktúrával csak bizonyos függvények képesek dolgozni
  % nevezetesen mldivide (AS,b) vagy röviden AS\ b és az inv(AS)
   >> b=[1 2 3 4 5]';

  >> mldivide(AS,b)
ans =
         0
    0.2500
    1.0000
    2.5000
    5.0000

% vagy ami ugyanaz
>> x=AS\b
x =
         0
    0.2500
    1.0000
    2.5000
    5.0000

>> x=inv(AS)*b
x =
         0
    0.2500
    1.0000
    2.5000
    5.0000

% A linsolve és a pinv nem működik ezzel az adatstruktúrával

>> x=linsolve(AS,b)
??? Error using ==> linsolve
Linsolve is currently not supported for sparse inputs


>> x=pinv(AS)*b
??? Error using ==> svd
Use svds for sparse singular values and vectors.
```

```
Error in ==> pinv at 29
  [U,S,V] = svd(A,0);
```

20. példa

Határozzuk meg az egyes vízkezelő reaktorokban a koncentráció értékét az alábbi kapcsolás esetén, tökéletes keveredést (a koncenráció azonos a reaktortér minden pontjában) feltételezve,



A mérlegegyenletek:

$6\,c_1 - c_3 = 50$
$-3\,c_1 + 3\,c_2 = 0$
$-c_2 + 9\,c_3 = 160$
$-c_2 - 8\,c_3 + 11\,c_4 - 2\,c_5 = 0$
$-3\,c_1 - c_2 + 4\,c_5 = 0$

Az egyenletrendszer mátrixos alakja:

$$A = \begin{pmatrix} 6 & 0 & -1 & 0 & 0 \\ -3 & 3 & 0 & 0 & 0 \\ 0 & -1 & 9 & 0 & 0 \\ 0 & -1 & -8 & 11 & -2 \\ -3 & -1 & 0 & 0 & 4 \end{pmatrix}; \; b = \begin{pmatrix} 50 \\ 0 \\ 160 \\ 0 \\ 0 \end{pmatrix}$$

```
%Direkt megoldás beépített függvénnyel
>> clear all
>> A = [6 0 - 1 0 0; -3 3 0 0 0; 0 - 1 9 0 0; 0 - 1 - 8 11 - 2; -3 - 1 0 0 4]

A = 6    0   - 1   0    0
    -3   3    0    0    0
    0  - 1    9    0    0
    0  - 1  - 8   11  - 2
    -3 - 1    0    0    4

>> AS=sparse(A)
AS =
  (1,1)       6
  (2,1)      -3
  (5,1)      -3
```

```
  (2,2)      3
  (3,2)     -1
  (4,2)     -1
  (5,2)     -1
  (1,3)     -1
  (3,3)      9
  (4,3)     -8
  (4,4)     11
  (4,5)     -2
  (5,5)      4

%A feladatot megoldhatjuk direkt módszerrel
>> b = [50 0 160 0 0]';
>> AS\b

ans = 11.5094
      11.5094
      19.0566
      16.9983
      11.5094
```

Jacobi módszer

Az iterációs egyenletek:

$$c_1 = \frac{1}{6} c_3 + \frac{50}{6}$$

$$c_2 = c_1$$

$$c_3 = \frac{1}{9} c_2 + \frac{160}{9}$$

$$c_4 = \frac{1}{11} c_2 + \frac{8}{11} c_3 + \frac{2}{11} c_5$$

$$c_5 = \frac{3}{4} c_1 + \frac{1}{4} c_2$$

Mátrixos alakban:

$$\mathbf{c}(i+1) = AI * \mathbf{c}(i) + bI = \begin{pmatrix} 0 & 0 & \frac{1}{6} & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{9} & 0 & 0 & 0 \\ 0 & \frac{1}{11} & \frac{8}{11} & 0 & \frac{2}{11} \\ \frac{3}{4} & \frac{1}{4} & 0 & 0 & 0 \end{pmatrix} * \mathbf{c}(i) + \begin{pmatrix} \frac{50}{6} \\ 0 \\ \frac{160}{9} \\ 0 \\ 0 \end{pmatrix}$$

Jacobi - módszer függvénye

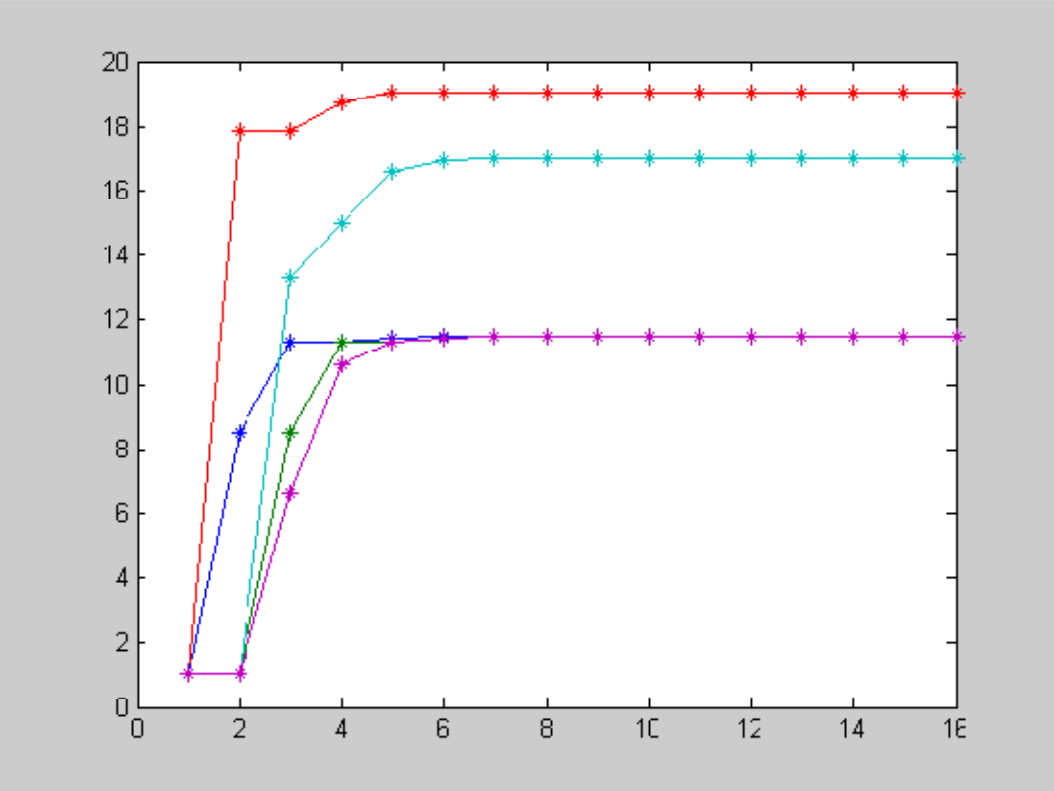
```
function y = jacobi (AI, bI, x0, eps, nmax)
X = [x0];
x1 = AI*x0 + bI;
i = 0;
X = [X, x1];
while and (i <= nmax, norm (x1 - x0) > eps)
x0 = x1;
x1 = AI*x0 + bI;
```

```
X = [X, x1];
i=i+1;
end
y = X;
```

Az iterációs formula előállítása

```
% az iterációs formula  x(i+1) = AI*x(i)+bI állandói: AI és bI
% A Jacobi módszer esetén a B mátrix diagonál mátrix és elemei

>> B=diag(A)
B =
     6
     3
     9
    11
     4

% Ennek inverze
>> Binv=1./B
Binv =
    0.1667
    0.3333
    0.1111
    0.0909
    0.2500

 % Ezzel az AI
 >> AI=eye(5)-diag(Binv)*A
AI =
         0         0    0.1667         0         0
    1.0000         0         0         0         0
         0    0.1111         0         0         0
         0    0.0909    0.7273         0    0.1818
    0.7500    0.2500         0         0         0

% A bI vektor pedig
>> bI=diag(Binv)*b
bI =
    8.3333
         0
   17.7778
         0
         0

% az induló vektor
>> x0=[1 1 1 1 1]';

% a megoldás
>> S=jacobi(AI,bI,x0,1e-6,100);

>> x=S(1:5,length(S))
```

```
x =
   11.5094
   11.5094
   19.0566
   16.9983
   11.5094

  % az iterációk száma
  >> niter=length(S)
niter =
    16

% a megoldás vektor

>> x=S(1:5,niter)
x =
   11.5094
   11.5094
   19.0566
   16.9983
   11.5094

% a megoldás relatív hibája
>> norm(A*x-b)/norm(b)
ans =
  1.2573e-008

% a kovergencia vizualizációja
ST=S';
plot(ST,'*-')
```



Gauss - Seidel módszer

Mint tudjuk ebben az esetben egy adott $x_i^{(k+1)}$ kiszámítása esetén a rákövetkező egyenletekben a többi változó frissített értékével számolunk,

$$x_i^{(k+1)} = f_i\left(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i^{(k)}, \ldots, x_n^{(k)}\right)$$

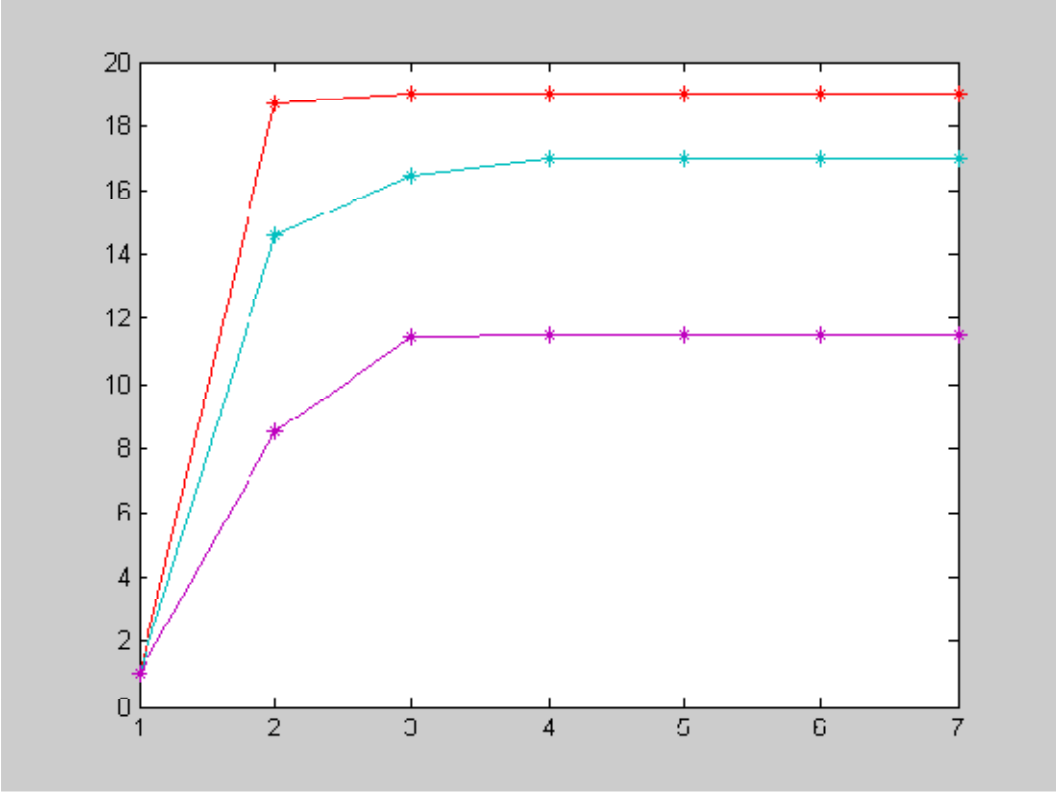
$$x_{i+1}^{(k+1)} = f_{i+1}\left(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i^{(k+1)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)}\right)$$

Az alábbi függvényben ezt a frissítést nem a megfelelő *B* mátrix előállításával oldottuk meg, elkerülendő a *B* inverzének képzését (bár az egy háromszögmátrix), hanem a sorokra vonatkozó ciklusban (j) egy egyszerű értékadó utasítással (x0 = x1), amely minden új $x_i^{(k+1)}$ számítása után frissíti a jobboldalt.

```
function y = gaussseidel (AI, bI, A, b, x0, eps, nmax)
X = [x0];
i = 0;
n = length (x0);
while and (i <= nmax, norm (A*x0 - b) > eps)
x1 = x0;
for j = 1 : n
x1 (j) = AI (j, 1 : n)*x0 + bI (j);
x0 = x1;
end
X = [X, x1];
end
y = X;
```

```
>> S = gaussseidel (AI, bI, A, b, x0, 1 e - 6, 100);

>> x = S (1 : 5, length (S))
```

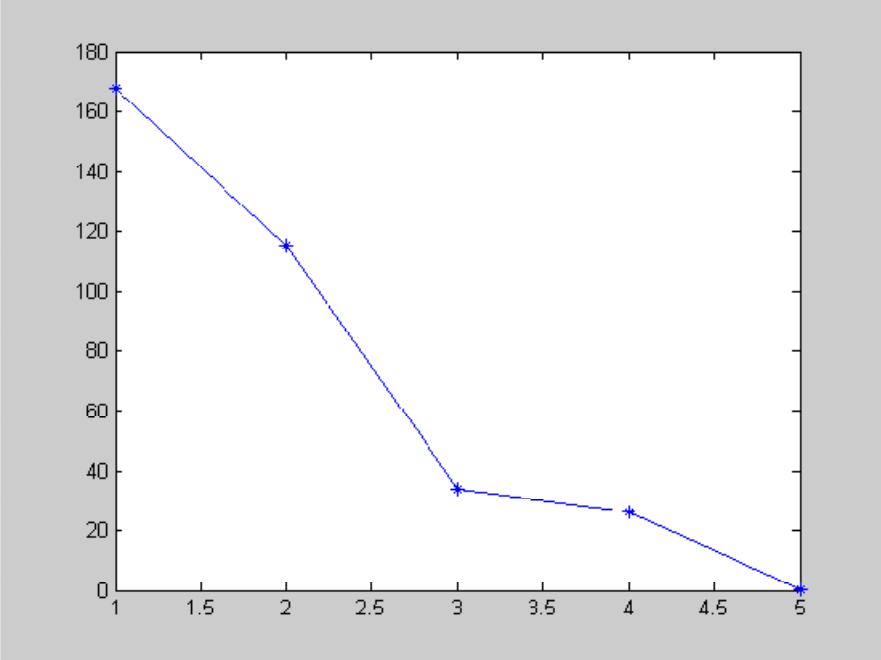
```
x = 11.5094
11.5094
19.0566
16.9983
11.5094

>> niter = length (S)
niter = 7

>> norm (A*x - b)/norm (b)
ans = 4.1509 e - 009

>> ST = S';
>> plot (ST, '*-')
```



Iterációs megoldás beépített függvénnyel

```
%Iterációs megoldás beépített függvénnyel

>> [x,flag,relres,iter,resvec] = gmres (AS, b);

% a megoldás
>> x
x = 11.5094
    11.5094
    19.0566
    16.9983
    11.5094

% a megoldás relatív hibája: norm(b-A*x)/norm(b)
```

```
>> relres
relres = 4.4050 e - 016

% a hibák az egyes iterációs lépések során: norm(b-A*x(i))
>> resvec

resvec = 167.6305
114.7905
34.1596
26.1821
0.0000

>> plot (resvec, ' b*-')
```



Most az iterációs lépések száma csak 5.

Megjegyzés

Előfordul, hogy az inverz mátrix előállítása a folyamat minőségi vizsgálata szempontjából is hasznos lehet, nem csupán akkor ha több jobboldalra akarjuk a számítást elvégezni! Esetünkben

```
>> inv (A)

ans =   0.1698    0.0063    0.0189    0         0
        0.1698    0.3396    0.0189    0         0
        0.0189    0.0377    0.1132    0         0
        0.0600    0.0746    0.0875    0.0909    0.0455
        0.1698    0.0896    0.0189    0         0.2500
```

Ebből kiderül - a negyedik oszlopban lévő zérusok alapján - hogy a 4. reaktor koncentrációjától (c4) nem függ a többi reaktor koncentrációja (nincs visszacsatolása!), amint az az ábrából is látszik. Ilyen jellegű megállapítások fontos szerepet játszanak egy hálózat viselkedésének megítélése szempontjából!

21. példa

Gauss - Seidel relaxációs módszer

A GS módszert módosíthatjuk egy relaxációs tényező bevezetésével,

$$x^{k+1} = (1 - \omega)\, x^k + \omega\, f(x_k)$$

ahol

$$0 < \omega \leq 2$$

Ha az eredeti séma

$$x^{k+1} = f(x_k)$$

divergál, akkor próbálkozhatunk ω <1 értékkel (alulrelaxálás). Ha lassú a konvergencia akkor az ω > 1 választás gyorsíthatja az iterációt (túlrelaxálás).

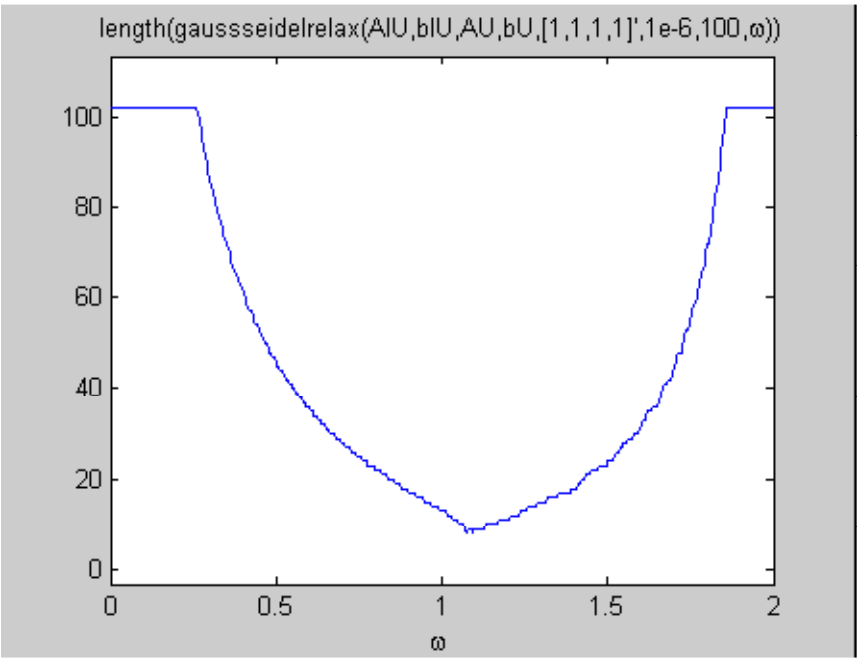
```
function y = gaussseidelrelax (AI, bI, A, b, x0, eps, nmax, omega)
X = [x0];
i = 0;
n = length (x0);
while and (i <= nmax, norm (A*x0 - b) > eps)
x1 = x0;
for j = 1 : n
x1 (j) = (1 - omega)*x0 (j) + omega*(AI (j, 1 : n)*x0 + bI (j));
x0 = x1;
end
X = [X, x1];
i=i+1;
end
y = X;
```

Határozzuk meg az alábbi lineáris egyenletrendszer iterációs megoldásának optimális relaxációs tényezőjét, azaz amelynél adott kezdeti feltétel és előírt hibakorlát mellett az iterációk száma minimális!

$$A = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix};\ b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

```
A=[4 -1 -1 0;-1 4 0 -1;-1 0 4 -1;0 -1 -1 4]

A =
    4   -1   -1    0
   -1    4    0   -1
   -1    0    4   -1
    0   -1   -1    4

>> b=[1 0 0 0]'

b =
    1
    0
    0
    0
```

```
% Az iterációs formula előállítása: AI és bI
>> AI=eye(4)-diag(1./diag(A))*A

AI =
         0      0.2500    0.2500         0
    0.2500         0         0    0.2500
    0.2500         0         0    0.2500
         0      0.2500    0.2500         0

>> bI=diag(1./diag(A))*b

bI =
    0.2500
         0
         0
         0

% A kezdő érték
x0=[1 1 1 1]';

%  Az iterációk száma a Jacobi módszerrel

>> S=jacobi(AIU,bIU,x0,1e-6,100);
>> length(S)
ans =
    22

 % Az iterációk száma a GaussSeidel módszerrel
 >> S=gaussseidel(AIU,bIU,AU,bU,x0,1e-6,100);
>> length(S)
ans =
    13

%Az optimális ω relaxációs tényező meghatározása érdekében definiáljuk az alábbi függvényt

>>niter=@(omega)length(gaussseidelrelax(AIU,bIU,AU,bU,x0,1e-6,100,omega));

%amely az ω függvényében megadja a szüksége iterációk számát. Rajzoljuk fel a függvényt
>> ezplot(niter,[0,2])
```



Látható, hogy a minimum nem $\omega = 1$-nél van, hanem valamivel nagyobb értéknél. Tehát túlrelaxációt alkalmazva az iterációk száma csökkenthető. A minimum

```
>> omegaopt=fminsearch(niter,1)
ans =
   1.0750

%Ezzel   a relaxációs tényezővel a konvergencia a leggyorsabb

>> niter(omegaopt)
ans =
    8
```

22. példa

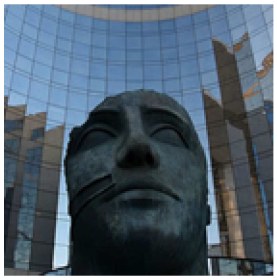
Oldjuk meg az alábbi egyenletrendszert:

$$A = \begin{pmatrix} 64\,919\,121 & -159\,018\,721 \\ \frac{83\,739\,041}{2} & -102\,558\,961 \end{pmatrix}; \qquad b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

```
%Numerikus megoldás gépi pontossággal

A = [64919121 - 159018721; 83739041/2 - 102558961];
  >> b = [1 0];
  >> format long
>> x=A\b'
x =
  1.0e+008 *
   1.060183080071325
   0.432817930017831

>> norm(A*x-b')
ans =
    0
```

```
 % Ez helyes megoldásnak tűnik.

 %Oldjuk meg az egyenletrendszert szimbolikusan is:

>> [x y] = solve (' 64919121*x - 159018721*y = 1', ' 83739041/2*x - 102558961*y = 0');
>> x
x = 205117922
y
y = 83739041
A*[x; y] - b'

ans = 0
        0

% Bár az eredmény teljesen más, ez is helyesnek látszik!

%Nem lenne egyértelmű megoldás?

>> det(A)
ans =
 -0.967370286583900

>> rank(A)xn
ans =
    1

>> rank([A;b])
ans =
    2

% Mivel det(A)≠ 0    létezik egyértelmű megoldás, de ezzel ellentmond,
%hogy rank(A)≠ rank([A;b])!!! Mi történik itt?!

% Alkalmazzunk a gépinél nagyobb precíziót:

>> digits(30);
>> Ap=vpa([64919121 -159018721; 83739041/2 -102558961]);
>> bp=vpa([1 0]);

>> det(Ap)
ans =
 -.50000000000000

 >> rank(Ap)
ans =
 2

>> rank([Ap,bp'])
ans =
2
```

```
%Most már rendben van a dolog! A viselkedés oka:

>> cond(A)
ans =
    1.814453860824375e+016

% vagy
>> vpa(cond(A),30)
 ans =
 18144538608243752.

%A mátrix kondiciószáma igen nagy!

% A precízióvesztés

-log10(2^-53*cond(A))
ans =
   -0.3042

%Azaz gépi precízióval esélyünk sincs a megoldásra

% Tehát ilyenkor vagy szimbolikus megoldást választunk, vagy a gépinél lényegesen nagyobb
%preciziót!

>> x=vpa(Ap\bp')
 x =
 205117922.
  83739041.
```



6. Gyakorlat

Zárt és nyílt-intervallum-módszerek, polinom gyökei, nagy preciziójú számítás

23. példa

Egy szabadfelszínű csatorna vízhozamát *Q*, az átlagos áramlási sebesség *U*, a csatorna szélessége *B* és a vízmagasság *H*, egyértelműen meghatározza. Az *U* sebesség viszont kapcsolatba hozható a csatorna globális és lokális geometriai tulajdonságaival, nevezetesen, a csatorna esésével *S* és hidraulikai sugarával *R*, továbbá a csatorna *Manning-féle* surlódási tényezőjével *n*. A hidraulikai sugár viszont kifejezhető a *B* és *H* értékeivel, hiszen az nem más, mint a nedvesített terület és a nedvesített kerület felének hányadosa.
Ezeket figyelembevéve a vízhozam:

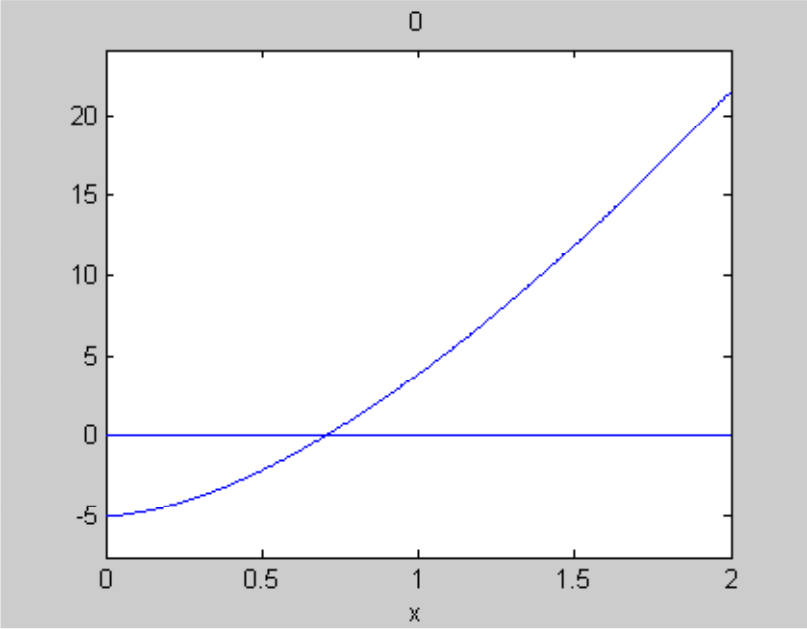
$$Q = \frac{\sqrt{S}}{n} \frac{(B\,H)^{\frac{5}{3}}}{(B + 2\,H)^{\frac{2}{3}}}$$

Határozzuk meg a csatornában a vízszint magasságát *H*-t, az alábbi adatok alapján:

S = 0.0002;
n = 0.03;
$Q = 5\,m^3/s$
B = 20 m

Definiáljuk az *f* (*H*) = 0 nemlineáris egyenletet, rajzoljuk fel és oldjuk meg a *húr-módszer*, a *fixpont-módszer*, a *Wegstein módszer* és a *Newton módszer* felhasználásával!

```
>> format long
>> S = 0.0002; n = 0.03; Q = 5; B = 20;

>> f=@(H)sqrt(S)/n*(B*H)^(5/3)/(B+2*H)^(2/3)-Q;
>> ezplot(f,[0,2])
>> hold on
>> ezplot('0',[0,2])
```



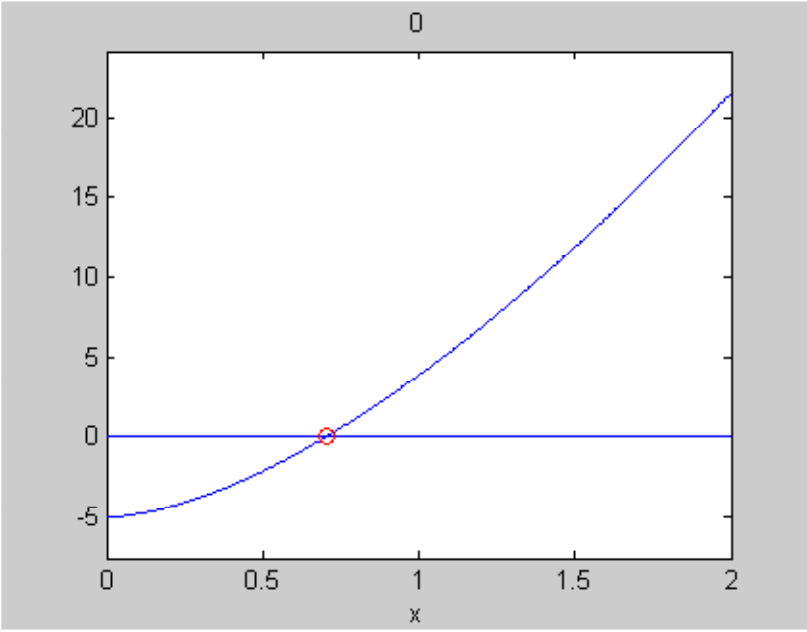
Húr - módszer

A húr és az x tengely metszéspontja,

$$c = \frac{a\ f(b) - b\ f(a)}{f(b) - f(a)}$$

```
function y = hur (f, a, b, eps, N)
c = (a*f (b) - b*f (a))/(f (b) - f (a));
i = 1;
while and (abs (f (c)) > eps, i <= N)
if f (c)*f (a) > 0
a = c;
else
b = c;
end
i = i + 1;
c = (a*f (b) - b*f (a))/(f (b) - f (a));
end
y = [c i];
```

```
%megoldás
>> Hsol = hur (f, 0.5, 1.5, 1e - 6, 100)

Hsol = 0.702293181030230  10.000000000000000

%ellenőrzés
>> f(Hsol(1))
ans =
   -8.692188480097229e-007

%vizuális ellenőrzés
>> plot([Hsol(1)],[f(Hsol(1))],'ro')
```



Newton - módszer

Az iterációs formula,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

```
function y = newton (f, df, x0, eps, N)
x1 = x0;
x2 = x1 - f (x1)/df (x1);
i = 1;
while and (abs ((x2 - x1)/x2) > eps, i <= N)
x1 = x2;
x2 = x1 - f (x1)/df (x1);
i = i + 1;
end
y = [x2, i];
```

```
% a derivált függvény előállítása

>> diff ('sqrt (S)/n*(B*H)^(5/3)/(B + 2*H)^(2/3) - Q', ' H')

ans = 100/9*2^(1/2)*20^(2/3)*H^(2/3)/(20 + 2*H)^(2/3) - 80/9*2^(1/2)*20^(2/3)*H^(5/3)/(20 + 2*H)^(5/3)

>> df=@(H)100/9*2^(1/2)*20^(2/3)*H^(2/3)/(20+2*H)^(2/3)-80/9*2^(1/2)*20^(2/3)*H^(5/3)/(20+2*H)^(5/3)

% a megoldás
>> Hsol=newton(f,df,1.,1e-6,100)
Hsol =
  0.702293256258431   4.000000000000000

 % ellenőrzés
  >> f(Hsol(1))
```

```
ans =
   6.217248937900877e-015
```

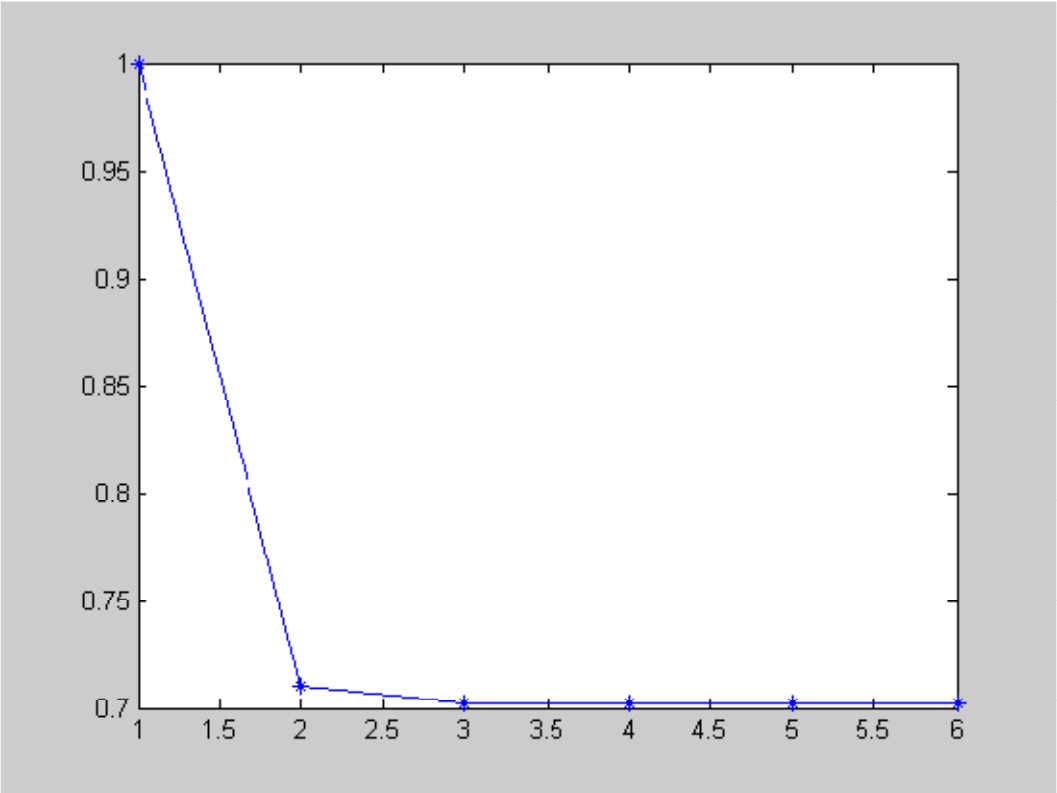
Fixpont - módszer

Hozzuk az egyenletet $x = g(x)$ alakra,

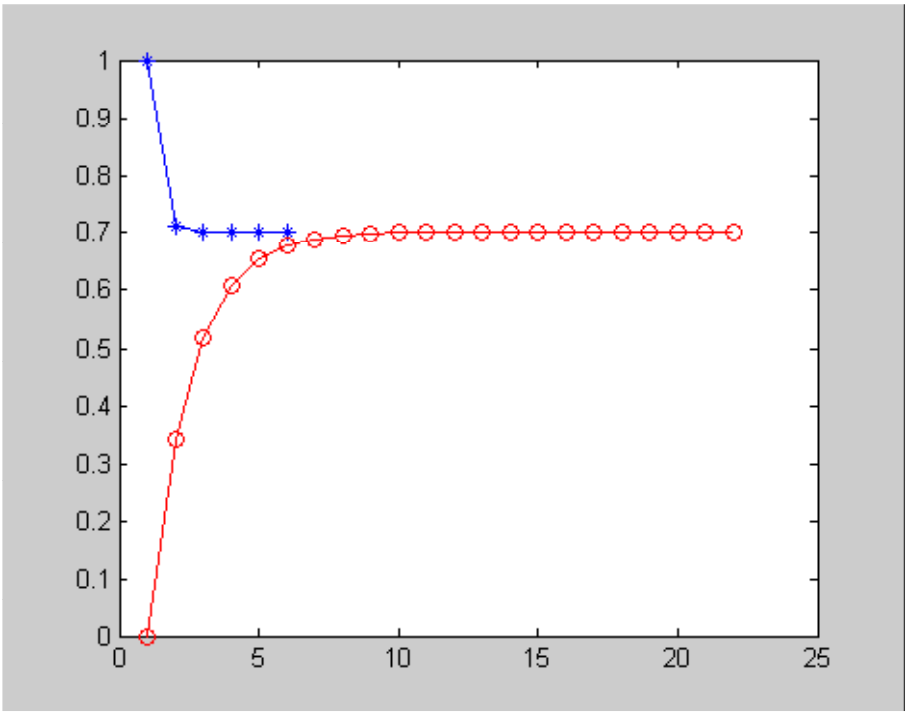
$$H = \left(\frac{n}{\sqrt{S}}\, Q\right)^{\frac{3}{5}} \frac{(B + 2\,H)^{\frac{2}{5}}}{B}$$

```
function y = fixedMR (g, x0, eps, lam, N)
global X Y
xold = x0;
xnew = (1 - lam)*xold + lam*g (xold);
i = 2;
X = [1, 2];
Y = [x0, xnew];
while and (abs ((xnew - xold)/xnew) > eps, i <= N)
xold = xnew;
xnew = (1 - lam)*xold + lam*g (xold);
i = i + 1;
X = [X, i];
Y = [Y, xnew];
end
y = [xnew, i];
```

```
>> g = @(H) (n/sqrt (S)*Q)^(3/5)*(B + 2*H)^(2/5)/B;

global X Y

% a megoldás H0= 1 és lam = 1 esetén

>> Hsol=fixedMR(g,1,1e-6,1,100)
Hsol =
  0.702293259936448   6.000000000000000

 % ellenőrzés
>> Hsol(1)-g(Hsol(1))
ans =
   3.581475893632558e-009

 >> plot(X,Y,'b*-')
```




```
% megoldás H0 = 0 és lam = 0.5 esetén

>> hold on
>> Hsol=fixedMR(g,0,1e-6,0.5,100)

Hsol =
   0.702292678627677  22.000000000000000

>> Hsol(1)-g(Hsol(1))
ans =
   -5.624689118910453e-007

>> plot(X,Y,'ro-')
```



Wegstein - módszer

Az iterációs formula,

$$x_{i+1} = \frac{x_{i-1}\, g(x_i) - g(x_{i-1})\, x_i}{x_{i-1} - x_i + g(x_i) - g(x_{i-1})}$$

ahol induláskor

$$x_1 = g(x_0)$$

```
function y = wegstein (g, x0, eps, N)
i = 1;
x1 = g (x0);
x2 = (x0*g (x1) - g (x0)*x1)/(x0 - x1 + g (x1) - g (x0));
while and (abs ((x2 - x1)/x2) > eps, i <= N)
x0 = x1;
x1 = x2;
x2 = (x0*g (x1) - g (x0)*x1)/(x0 - x1 + g (x1) - g (x0));
i = i + 1;
end
y = [x2, i];
```

```
>> Hsol = wegstein (g, 1, 1 e - 6, 100)

Hsol = 0.702293256255948 3.000000000000000

>> Hsol (1) - g (Hsol (1))

ans = -2.417732680726203 e - 012

% egy másik kezdőértékkel,
```

```
>> Hsol = wegstein (g, 0, 1 e - 6, 100)

Hsol = 0.702293256258606 3.000000000000000

>> Hsol (1) - g (Hsol (1))

ans = 1.706412788848866 e - 013
```

Megoldás a beépített *fzero* függvénnyel

```
>> options = optimset (' Display', ' iter', ' TolFun', 1e - 6);

% egy kezdőértékkel (nyílt-intervallum módszer)
>> fzero (f, 1, options)
ans =
   0.702293256258430

>> f (ans)
ans = 0

%intervallum megadással (zárt-intervallum-módszer)

>> fzero (f, [0.5 1.5], options)
ans =
   0.702293256258430

>> f (ans)
ans = 0

% értelmezzük az itt nem közölt képernyőre kiírt adatokat!

%alternatív information

>>  [x,fval,exitflag,output]=fzero(f,[0.5 1.5])
x =
  0.702293256258430

fval =
    0

exitflag =
    1

output =
   intervaliterations: 0
   iterations: 6
   funcCount: 8
   algorithm: 'bisection, interpolation'
   message: 'Zero found in the interval [0.5, 1.5]'

% illetve
```

```
>> [x,fval,exitflag,output]=fzero(f,1)
x =
   0.702293256258430

fval =
     0

exitflag =
     1

output =
    intervaliterations: 8
    iterations: 5
    funcCount: 21
    algorithm: 'bisection, interpolation'
    message: 'Zero found in the interval [0.68, 1.22627]'
```

24. példa

Az általános gáztörvény

$$p\,v = R\,T$$

amely azonban nem használható nagy nyomáson illetve magas hõmérséklet esetén. Egy alternatíva a *van der Waals egyenlet*,

$$\left(p + \frac{a}{v^2}\right)(v - b) = R\,T$$

ahol az általános gázállandó R = 0.082054.
Az *a* és *b* paraméterek a gáz fajtájától függõek, például az oxigén esetén: *a* =1.360 és *b* =0.03183.

Határozzuk meg a *v* fajtérfogat értékét 700 K és *p* = 50 atm esetén! A kezdőértéket számítsuk ki az általános gáztörvény alapján! Alkalmazzunk beépített függvényeket! Írjunk egy felhasználó barát m-fájlt!

Az *fsolve* alkalmazása

```
function y = vanderwaals (p, T, a, b)
R = 0.082054;
f = @(v) (p + a/v^2)*(v - b) - R*T;
v0 = R*T/p;
y = fsolve (f, v0);
err = abs (f (y));
y = [y, err];
```

```
>> a = 1.360; b = 0.03183; p = 50; T = 700;
>> vanderwaals (p, T, a, b)
Optimization terminated : first - order optimality is less than options.TolFun.ans = 1.157737836806198
0.000000000001194
```

Megoldás, mint polinom gyöke

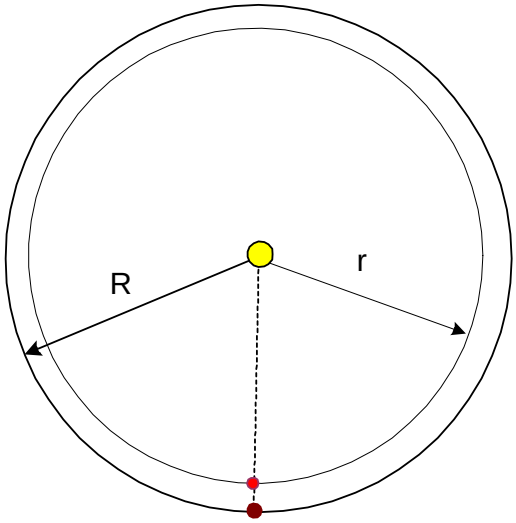
```
% a polinom együtthatóinak meghatározása

>> syms a b p T v R

% a polinom (nullára rendezve és v^2 -tel beszorozva)

>> expand(v^2*((p + a/v^2)*(v - b) - R*T))
ans =
 v^3*p-v^2*p*b+v*a-a*b-v^2*R*T

 % az együtthatók
>> coeffs(ans,v)
 ans =
[     -a*b,        a, -R*T-p*b,        p]
```

```
function y = gaslaw (p, T, a, b)
R = 0.082054;
c = [-a*b, a, -R*T - p*b, p];
% Az együtthatók sorrendjét meg kell fordítani a roots miatt
for i = 1 : 4
h (i) = c (5 - i);
end
gy = roots (h);
% csak a valós gyököt tekintjük
i = 1;
while not (isreal (gy (i)))
i = i + 1;
end
y = gy (i);
```

```
>> a = 1.360; b = 0.03183; p = 50; T = 700;
>> gaslaw(p,T,a,b)
ans =
   1.157737836806174
```

25. példa

A NASA *Wind* nevezetű műholdja mindig a Föld és a Nap közötti összekötő egyenesen van. A Föld és a Nap távolsága *R*, a műhold és a Nap távolsága *r*. Határozzuk meg a műhold Naptól való távolságát!



A műholdra ható erők: a Nap tömegvonzásából származó erő, a Föld tömegvonzási ereje és a Nap körüli keringésből keletkező centrifugális erő egyensúlyban vannak:

$$\frac{G\,M_S\,m}{r^2} = \frac{G\,M_F\,m}{(R-r)^2} + m\,r\,\omega^2$$

$G = 6.67 \times 10^{-11}$ – gravitációs állandó
$M_S = 1.98 \times 10^{30}$ kg – Nap tömege
$M_F = 5.98 \times 10^{24}$ kg – Föld tömege
m – a műhold tömege
$R = 1.49 \times 10^{11}$ m
$T = 3.15576 \times 10^{7}$ sec – műhold keringési ideje
és a keringés szögsebessége

$$\omega = \frac{2\,\pi}{T}$$

Mivel $(R-r)^2$ igen kicsi érték, célszerű beszorozni $r^2\,(R-r)^2$, azaz az egyenlet

$$G\,M_S\,(R-r)^2 - G\,M_F\,r^2 - r^3\,(R-r)^2\,\omega^2 = 0$$

Mivel ez egy polinomiális egyenlet a *r* változóra nézve, a deriváltja is polinom, alkalmazzuk a Newton - módszert!

Megoldás gépi precizióval

```
>> clear all; clc

%adatok
>> G=6.67e-11; MS=1.98e30; MF=5.98e24; R=1.49e11; T=3.15576e7;
>> w=2*pi/T
w =
   1.991021277657232e-007

% a függvény
>> f=@(r)G*MS*(R-r)^2-G*MF*r^2-r^3*(R-r)^2*(2*pi/T)^2;

% a függvény deriváltja
>> diff( 'G*MS*(R-r)^2-G*MF*r^2-r^3*(R-r)^2*(2*pi/T)^2','r')
ans =
 -2*G*MS*(R-r)-2*G*MF*r-12*r^2*(R-r)^2*pi^2/T^2+8*r^3*(R-r)*pi^2/T^2
```

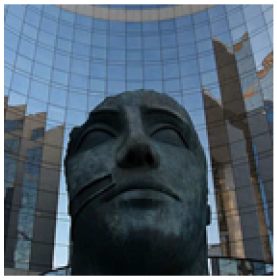
```
 %tehát
>> df=@(r)-2*G*MS*(R-r)-2*G*MF*r-12*r^2*(R-r)^2*pi^2/T^2+8*r^3*(R-r)*pi^2/T^2;

% megoldás
% Egy reális kezdőérték: r0=R
>> rsol=newton(f,df,R,1e-10,100)
rsol =
  1.0e+011 *
   1.476177500961505   0.000000000150000

% ellenőrzés
>> f(rsol(1))
ans =
    1.888946593147858e+023

% Ez igen nagy hiba
% Nagyobb precizióval kellene számolni! De ehhez át kell írni az m-fájlt.
```

Megoldás tetszőleges precizióval

```
function y = newtonvpa (f, df, x0, eps, N,digit)
digits (digit);
x1 = vpa (x0);
x2 = vpa (x1 - f (x1)/df (x1));
i = 1;
% bizonyos beépített függvények nem engedik a vpa használatát, így az and sem!
% ha egy változó szimbolikus vagy mesterségesen rendelünk hozzá preciziót (vpa),
% akkor a double függvénnyel konvertálhatjuk vissza valós változóra
while and (double (vpa (abs (f (x2)))) > double (vpa (eps)), i <= N)
x1 = vpa (x2);
x2 = vpa (x1 - f (x1)/df (x1));
i = i + 1;
end
y = [x2, i];
```

```
>> digit=50;
>> rsolvpa=newtonvpa(f,df,R,1e-10,100,digit)
 rsolvpa =
[ 147617750096.150461886307641713061255048767023759 69,                 17]

 % ellenőrzés
 >> digits(digit);
>> vpa(f(rsolvpa(1)))
 ans =
 -.6e-10

% Ha nagyobb pontosság kell akkor eps  értékét növeljük, de ekkor célszerű a preciziót is növelni!

>>digit=100;
 >> rsolvpa=newtonvpa(f,df,R,1e-13,100,digit)
```

```
rsolvpa =
[
147617750096.150461886307641713061255048767023759687132312902938123283132349559600937759
5436612159964,                                                                          17]

% ellenőrzés
>> digits(digit)
>> vpa(f(rsolvpa(1)))
ans =
 .1000866466507672161944131025290074426788108e-18
```



7. Gyakorlat

Nemlineáris egyenletrendszerek megoldása

26. példa

Keressük meg az alábbi egyenletrendszer valós, pozitív gyökeit!

$$x\,y - x^3 - \frac{1}{4} = 0$$

$$x^2 + y^2 - 3 = 0$$

Rajzoljuk fel az egyenleteket reprezentáló görbéket,

$$y = \frac{\frac{1}{4} + x^3}{x}$$

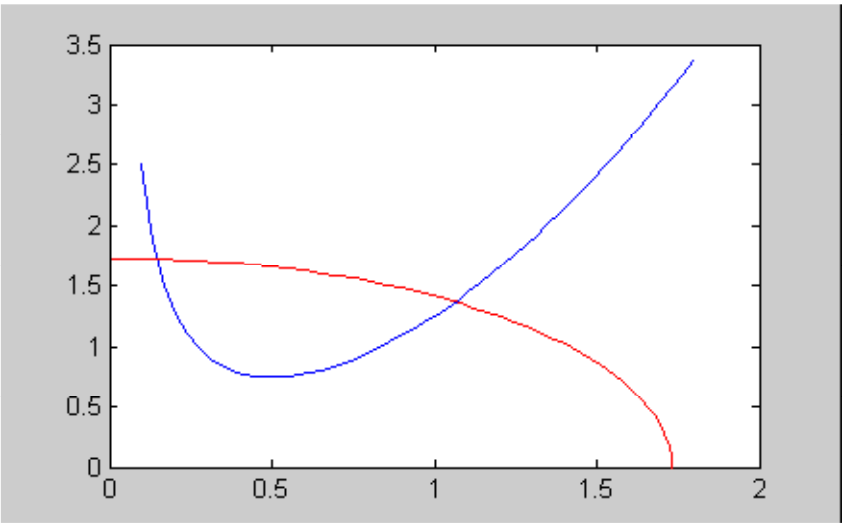
illetve a kör esetén

$$x(t) = \sqrt{3}\,\cos(t) \qquad \text{és} \qquad y(t) = \sqrt{3}\,\sin(t)$$

Megjegyzés

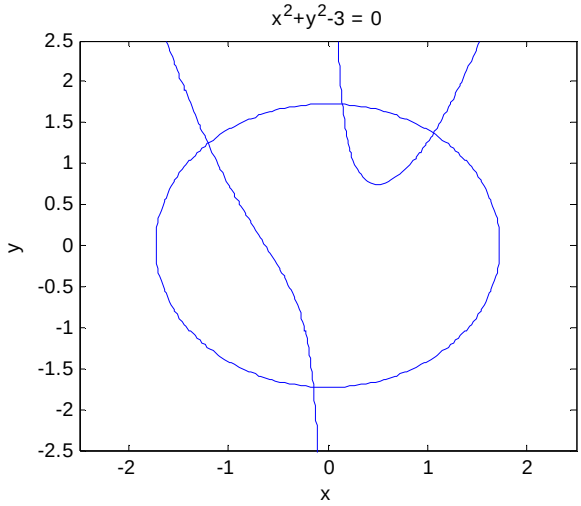
Látni fogjuk, hogy egyszerűbben is el lehet végezni az ábrázolást!

```
x = linspace (0.1, 1.8, 50);
y1 = (1/4 + x.^3)./x;
plot (x, y1)
t = linspace (0, pi/2, 50);
x2 = sqrt (3)*cos (t);
y2 = sqrt (3)*sin (t);
hold on
plot (x2, y2, ' r')
```



Egyszerűbb ábrázolás - minden valós gyököt szemlélteve (explicit függvény zérus szintvonalának megjelenítése),

```
ezplot('x*y-x^3-0.25')
hold on
ezplot('x^2+y^2-3')
```



Többváltozós Wegstein - módszer

Alkalmazzuk a Wegstein - módszer többváltozós formáját, amely egy gradiens nélküli eljárás:

Az *i* - edik változó frissítése a *k*-adik iterációs lépésben,

$$x_i^{(k+1)} = (1 - c_i)\, x_i^{(k)} + c_i \left(g\left(x^{(k)}\right)\right)_i$$

ahol

$$c_i = \frac{1}{1 - \frac{\left(g\left(x^{(k)}\right)\right)_i - \left(g\left(x^{(k-1)}\right)\right)_i}{x_i^{(k)} - x_i^{(k-1)}}}$$

```
function y = wegsteinsys (g, x0, eps)
n = length (x0);
```

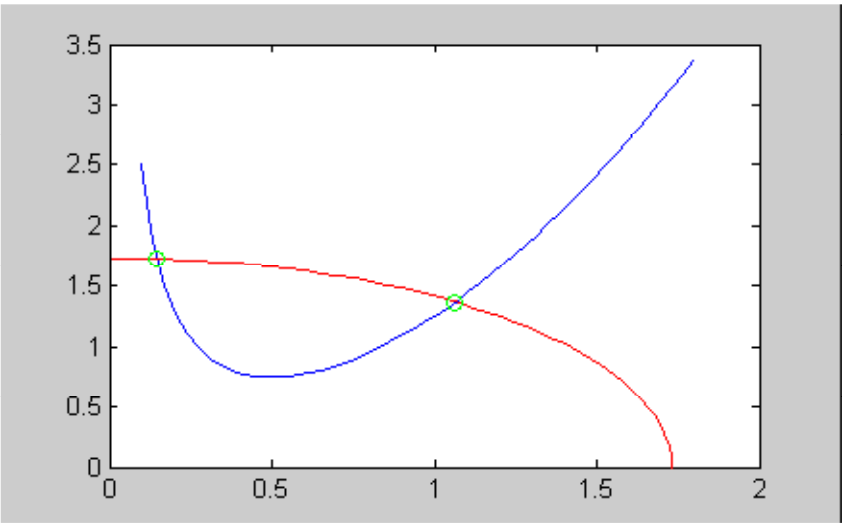
```
x1 = g (x0);
while norm (x1 - x0) > eps
gx0 = g (x0);
gx1 = g (x1);
for i = 1 : n
c (i) = 1/(1 - (gx1 (i) - gx0 (i))/(x1 (i) - x0 (i)));
x2 (i) = (1 - c (i))*x1 (i) + c (i)*gx1 (i);
end
x0 = x1;
x1 = x2';
end
y = x1;
```

Írjuk át az egyenletrenszert $(x,\ y)^T = \mathbf{g}\ (x, y)$ alakba, az első egyenletből *x*-t, a másodikból *y*-t kifejezve

$$g(x,\ y) = \begin{pmatrix} \frac{0.25}{y - x^2} \\ \sqrt{|3 - x^2|} \end{pmatrix}$$

A valós tartományban való maradás érdekében alkalmazunk abszolútértéket.

```
>>g = @(x)[0.25/(x (2) - x (1)^2); sqrt (abs (3 - x (1)^2))]

%Legyen a kezdőérték
>> x0 = [1; 2];
>> xs = wegsteinsys (g, x0, 1 e - 15);
>> xs
xs = 0.1467
       1.7258

%ellenőrzés
>> xs-g (xs)
ans = 1.0 e - 015*-0.0833
0.2220

%Rajzoljuk be a megoldást
>> plot (xs (1), xs (2), ' ro')

% A másik zérushely közelítő értéke
>> x0 = [1.1; 1.4];
>> xs = wegsteinsys (g, x0, 1 e - 15);
>> xs
xs = 1.0639
       1.3668

>> xs-g (xs)
ans = 1.0 e - 011*0.0002

-0.3289
>> plot (xs (1), xs (2), ' go')
```



A példa szemlélteti, hogy a nemlineáris egyenletrendszer megoldásánál a megfelelő kezdőérték megválasztása döntő fontosságú!

Newton - Raphson - módszer

Minden k-adik iterációs lépben szükség van egy lineáris egyenletrendszer megoldásra:

$$J\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = -f\left(x^{(k)}\right)$$

```
function y = newtonstep (x, J, f)
A = J (x);
b = f (x);
dx = linsolve (A, -b);
y = x + dx;
```

Ezzel a többváltozós Newton - módszer

```
function y = newtonsys (f, J, x0, eps, nmax)
x1 = x0;
x2 = newtonstep (x1, J, f);
n = 1;
while and (norm (x2 - x1) > eps, n <= nmax)
x1 = x2;
x2 = newtonstep (x1, J, f);
n = n + 1;
end;
y = x2;
```

```
%Most az egyenletrendszer eredeti alakja alkalmazható
f = @(x)[x (1)*x (2) - x (1)^3 - 1/4; x (1)^2 + x (2)^2 - 3];

%Állítsuk elő a Jacobi mátrixot szimbolikusan

syms u v z F

F = [u*v - u^3 - 1/4; u^2 + v^2 - 3];
z = [u v];
j = jacobian (F, z)
```

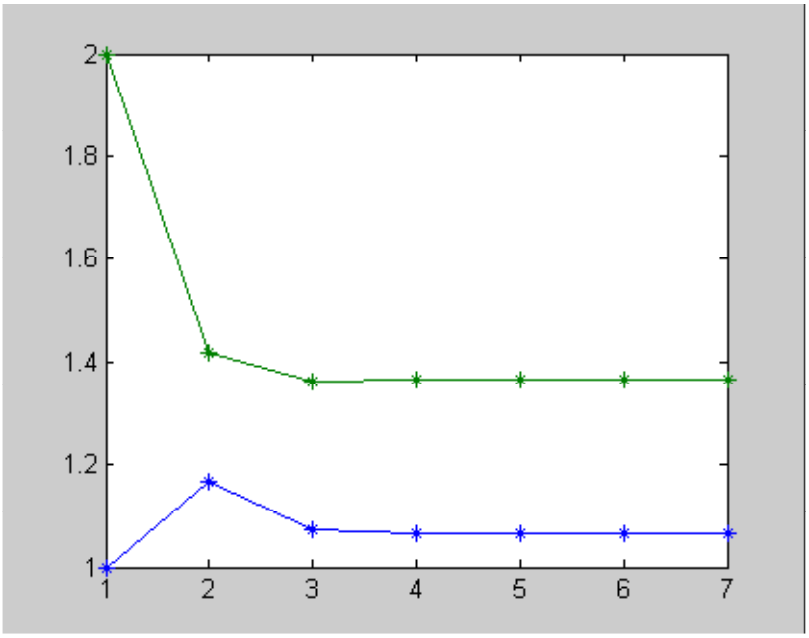
```
j = [v - 3*u^2, u]
    [2*u,     2*v]

%Csináljunk belőle anonymous függvényt
J = @(x)[x (2) - 3*x (1)^2, x (1); 2*x (1), 2*x (2)];

% Az első gyök meghatározása
x0 = [1; 2]

>> xs = newtonsys (f, J, x0, 1 e - 10, 5)
xs = 1.0639
       1.3668

>> f (xs)
ans = 1.0 e - 015*0.2220
0.4441

% A második gyök meghatározása
>> x0 = [0; 2]
x0 = 0
2

>> xs = newtonsys (f, J, x0, 1 e - 10, 10)
xs = 0.1467
       1.7258
>> f (xs)
ans = 0
0
```

Módosítsuk a *newtonsys* függvényt, úgy, hogy az iteráció lépéseit is szemléltetni tudjuk! Célszerű ismét globális változókat alkalmazni!

```
function y = newtonsysM (f, J, x0, eps, nmax)
global X Y
x1 = x0;
x2 = newtonstep (x1, J, f);
X = [1, 2];
Y = [x1, x2];
n = 2;
while and (norm (x2 - x1) > eps, n <= nmax)
x1 = x2;
x2 = newtonstep (x1, J, f);
n = n + 1;
X = [X, n];
Y = [Y, x2];
end;
y = x2;
```

```
>> x0 = [1; 2];
>> global X Y;
>> xs = newtonsysM (f, J, x0, 1 e - 10, 10)
xs = 1.063871320906402
```

```
    1.366813013017094

figure(2)
YT=Y';
plot(X,YT,'*-')
```



Broyden - módszer

Ez egy ún. *kvázi - Newton módszer*, amelynél nincs szükség a Jacobi- mátrix előállítására. Különösen akkor hatékony, ha a Jacobi mátrix csak numerikusan állítható elő, azaz véges differencia módszerrel, analitikusan nem. A k-adik iterációban a Jacobi mátrixnak megfelelő Broyden mátrix frissítése,

$$B^{(k+1)} = B^{(k)} + \frac{\left(\Delta f^{(k)} - B^{(k)}\,\Delta x^{(k)}\right)\left(\Delta x^{(k)}\right)^{T}}{\left(\Delta x^{(k)}\right)^{T}\Delta x^{(k)}}$$

ahol

$$\Delta f^{(k)} = f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)$$
$$\Delta x^{(k)} = \left(x^{(k)} - x^{(k-1)}\right)$$

ezzel

$$B^{(k+1)}\left(x^{(k+1)} - x^{(k)}\right) = B^{(k+1)}\Delta x^{(k+1)} = -\,f\left(x^{(k)}\right)$$

amelyből $\Delta x^{(k+1)}$ számítható.

```
function y = broyden (f, x0, eps, nmax)
n = length (x0);
x = x0;
B = eye (n);
dx = x*0.5;
i = 1;
while and (norm (dx) > eps, i < nmax)
df = f (x + dx) - f (x);
```

```
B = B + ((df - B*dx)*dx')/(dx'*dx);
x = x + dx;
b = f (x);
if or (cond (B) > 1 e3, det (B) < 1 e - 3)
dx = pinv (B)*(-b);
else
dx = linsolve (B, -b);
end
i = i + 1;
end
y = x;
```

Látható, hogy az iterációs lépésben, a lineáris egyenletrendszer megoldását a pszeudóinverz segítségével végezzük, ha a *B* együtthatómátrix kondiciószáma 1000-nél nagyobb, vagy ha a determinánsa kisebb, mint 0.001!

```
f = @(x)[x (1)*x (2) - x (1)^3 - 1/4; x (1)^2 + x (2)^2 - 3];
x0 = [1; 2];

>> xs = broyden (f, x0, 1 e - 15, 20)
xs = 0.1467
     1.7258

f (xs)
ans = 1.0 e - 016*-0.5551
0

>> x0 = [1; 1];

xs = broyden (f, x0, 1 e - 15, 20)
xs = 1.0639
     1.3668

f (xs)
ans = 0
0
```

A *Broyden-módszer*nek az inverz kiszámítása nélküli változata:

$$\Delta x^{(k+1)} = - \left(B^{(k+1)}\right)^{-1} \boldsymbol{f}\left(x^{(k)}\right)$$

Alkalmazva az ún. *Sherman - Morrison* formulát, a $B^{(k+1)}$ mátrix inverze számítható a $\left(B^{(k)}\right)^{-1}$ alapján, azaz

$$B^{(k+1)^{-1}} = B^{(k)^{-1}} + \frac{\left(\Delta x^{(k)} - B^{(k)^{-1}} \Delta f^{(k)}\right)\left(\Delta x^{(k)}\right)^{T}}{\left(\Delta x^{(k)}\right)^{T}\left(B^{(k)^{-1}} \Delta f^{(k)}\right)} \left(B^{(k)}\right)^{-1}$$

Így még lineáris egyenletrendszert sem kell megoldani, csak egy mátrixszorzás kell!

```
function y = broydenSM(f,x0,eps,nmax)
n = length (x0);
x = x0;
BI = eye (n);
dx = x*0.5;
i = 1;
while and (norm (dx) > eps, i < nmax)
```

```
df = f(x + dx) - f(x);
BI = BI +(((dx - BI*df)'*dx)*BI)/(dx'*(BI*df));
x = x + dx;
dx = -BI*f(x);
i = i + 1;
end
y = x;
```

Ezt alkalmazva,

```
>> x0 = [1; 2];
>> xs = broydenSM (f, x0, 1 e - 15, 20)

xs = 0.1467
      1.7258

 >> x = broydenSM(f, x0,1e-10, 100)
x =
  1.063871320903931
  1.366813013017988
```

fsolve

Ez a beépített függvény optimalizációval, minimalizálja az egyenlet hibavektorának normáját, azaz

$$\min_{\mathbf{x}} (\text{norm} (\mathbf{f} (\mathbf{x})))$$

```
>> f = @(x)[x (1)*x (2) - x (1)^3 - 1/4; x (1)^2 + x (2)^2 - 3];

>> x0 = [1; 2];

%Írassuk ki az egyes iterációs lépéseket
>> options = optimset (' Display', ' iter');
>> [x,fval] = fsolve (f, x0, options)
                                                Norm of
Iteration          Func - count        f (x)          step
     0                 3              4.5625
     1                 6              0.169758       0.606676
     2                 9              0.000726985    0.10712
     3                12              1.30958 e - 007  0.0104454
     4                15              5.08352 e - 015  0.000150647
Optimization terminated : first - order optimality is less than options.TolFun.
x = 1.0639
     1.3668
fval = 1.0 e - 007*
-0.6759
0.2269

% pontosabb eredményt kaphatunk a hibakorlát csokkentésével
>> options = optimset (' Display', ' iter', ' TolFun', 1 e - 10);
>> [x,fval] = fsolve (f, x0, options)
                                                Norm of
Iteration          Func - count        f (x)          step
```

```
     0          3          4.5625
     1          6          0.169758          0.606676
     2          9          0.000726985       0.10712
     3          12         1.30958 e - 007   0.0104454
     4          15         5.08352 e - 015   0.000150647
     5          18         1.77494 e - 029   2.9537 e - 008
Optimization terminated : first - order optimality is less than options.TolFun
x = 1.0639
    1.3668

fval = 1.0 e - 014*
-0.3997
0.1332

%Látható, hogy a pontosság növelése drága dolog, ezért feleslegesen ne éljünk vele!
```

Oldjuk meg a feladatot a fenti módszerrel az utasításokat script - file - ba szervezve!

```
%minta a zh feladatok elkészítésére
%
%Név: Rosta Mariann
%Neptunkód: TZ8UI5
%Csoport:05
%
%011 Feladat
%Oldjuk meg az alábbi egyenletrendszert:
%
%f1(x1,x2)=x1*x2-x1^3-1/4
%f2(x1,x2)=x1^2+x2^2-3
%
%Alkalmazzuk az fsolve beépített függvényt.
%Legyen a megoldás hibakorlátja eps=1e-10.
%Írassuk ki az egyes iterációs lépések adatait!
%
%Megoldás:
%
%1) Az egyenletrendszer anonymous függvény formában:
%
f=@(x)[x(1)*x(2)-x(1)^3-1/4; x(1)^2+x(2)^2-3];
%
%2) A kezdetiérték:
%
x0=[1,1];
%
%3) Az opciók beállítása
%
options=optimset('Display','iter','TolFun',1e-10);
%
%4) A függvény hívása
%
display('Az iterációs lépések adatai:')
%
 [x,fval] = fsolve (f, x0, options);
```

```
%
%5) A megoldás
%
display('A megoldás:')
x
%
%6) A megoldás aktuális hibája
%
display('A megoldás hibavektora:')
fval
%
%A script file vége
```

Legyen a script file neve : rosta011.m.

```
>> rosta011
Az iterációs lépések adatai:

                                 Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)              step          optimality     radius
     0             3      1.0625            2.25                1
     1             6      0.0327775         0.424918          0.526            1
     2             9      7.78869e-006      0.0527368       0.00736          1.06
     3            12      5.55012e-013      0.000803591     2.13e-006        1.06
     4            15      2.36962e-026      2.27936e-007    4.03e-013        1.06
Optimization terminated: first-order optimality is less than options.TolFun.
A megoldás:

x =

    1.0639    1.3668

A megoldás hibavektora:

fval =

  1.0e-012 *

   -0.1452
    0.0511
```

solve

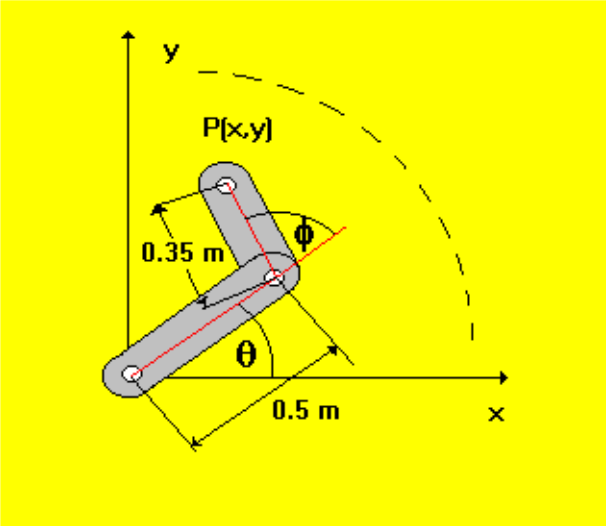
Mivel algebrai egyenletrendszerről van szó, alkalmazhatjuk a *solve* függvényt, amely globális megoldást ad, azaz kezdőérték nélkül megadja az összes megoldást! Ez csak algebrai polinomokból álló rendszernél működik, mivel az eljárás numerikus Gröbner bázist alkalmaz!

```
>> clear all
>> syms x y
>> sol = solve (' x*y - x^3 - 1/4', ' x^2 + y^2 - 3', x, y)

sol = x : [6 x1 sym]
      y : [6 x1 sym]
```

```
% tehát 6 megoldása van az egyenletrendszernek

>> for i=1:6
simplify(sol.x(i))
simplify(sol.y(i))
end

ans =
.14668682796932993359779964791651
ans =
1.7258281995900681786695777518941

ans =
1.0638713209064023525148021141852
ans =
 1.3668130130170939606557897478650

ans =
 .68734505876286124992287921218087e-1+1.52234075320730519077161184345 58*i
ans =
-2.30539736394834250682154038146 02+.45387984337689211354887233247000e-1*i

ans =
-.14313409902258141915585801841528
ans =
-1.72612648137295948757603271601 18

 ans =
-1.20489306160572311694131958612 26
ans =
 1.24427999666248236189374597917 30


ans =
 .68734505876286124992287921218087e-1-1.52234075320730519077161184345 58*i
ans =
-2.30539736394834250682154038146 02-.45387984337689211354887233247000e-1*i

 % tehát valóban 2 valós pozitív, egy valós negatív, egy valós vegyes, és egy komplex konjugált megoldásunk van.
>>
```

27. példa

Az ábrán látható robotkar poziciónálását kell elvégeznünk, azaz egy adott (x_p, y_p) esetén meg kell határozni a θ és ϕ szögeket, amelyek ezt pozíciót eredményezik.



A $P(x, y)$ pont koordinátái a θ és ϕ szögek függvényében:

$$x(\theta, \phi) = 0.5 \cos(\theta) + 0.35 \cos(\theta + \phi)$$

$$y(\theta, \phi) = 0.5 \sin(\theta) + 0.35*\sin(\theta + \phi)$$

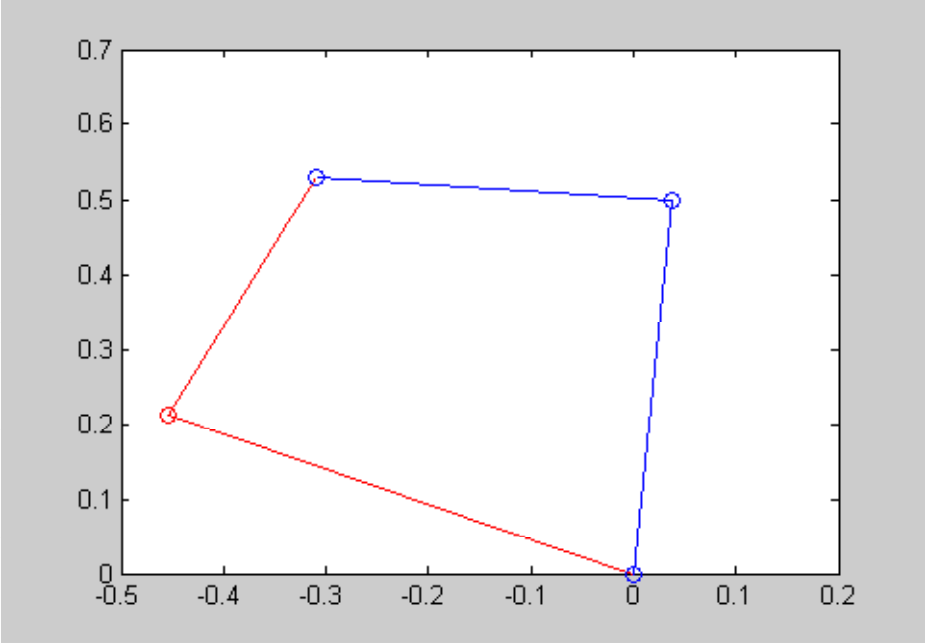
Határozzuk meg a szögek értékét ha a kívánt pozició:

$$x_p = -0.31$$

$$y_p = 0.53$$

```
>> xp = -0.31; yp = 0.53;

 >> x = @(u, v) 0.5*cos (u) + 0.35*cos (u + v);
 >> y = @(u, v) 0.5*sin (u) + 0.35*sin (u + v);

 % a függvény, amelynek zérushelyét keressük
 >> f = @(X)[x (X (1), X (2)) - xp, y (X (1), X (2)) - yp];

% legyen a kezdőérték
>> X0 = [1, -1];

% alkalmazzuk a beépített fsolve függvényt
>> options = optimset('Display', 'off', 'TolFun', 1e-10);
>> [X,fval] = fsolve (f, X0, options)
X =
   2.7065   -1.5579
fval =
  1.0e-010 *
   0.2961   -0.1385

% a hiba normája
>> norm(fval)
ans =
  3.2690e-011

% az eredmény megjelenítéséhez adjuk meg a kar csuklópontjainak koordinátáit,
% mint a szögek függvényét:
```

```
>> Xa = @(u, v)[0, 0.5*cos (u), x (u, v)]';
>> Ya = @(u, v)[0, 0.5*sin (u), y (u, v)]';

%esetünkben
>> xa=Xa (X (1), X (2))
 xa = 0
        - 0.4534
        - 0.3100
  >>ya= Ya (X (1), X (2))
  ya = 0
          0.2107
          0.5300
% rajzoljuk fel a manipulátort
  >> plot (xa, ya, ' ro - ')

 % Nézzük meg van-e másik megoldás is?
% egy másik kezdőpont
  >> X0 = [-1, 1];
>> [X,fval,exitflag,output] = fsolve (f, X0, options);
>> X
X =
   1.4936    1.5579
>> fval
fval =
    0     0

% az iterációk száma
>> output.iterations
ans =
   10

% a szükséges függvénykiértékelések száma

>> output.funcCount
ans =
   31

 % rajzoljuk fel ennek a megoldásnak megfelelő poziciót is

  >> xa = Xa (X (1), X (2))
   xa = 0
          0.0386
        - 0.3100
 >> ya = Ya (X (1), X (2))
  ya = 0
         0.4985
         0.5300

>> hold on
>> plot (xa, ya, ' bo - ')
```

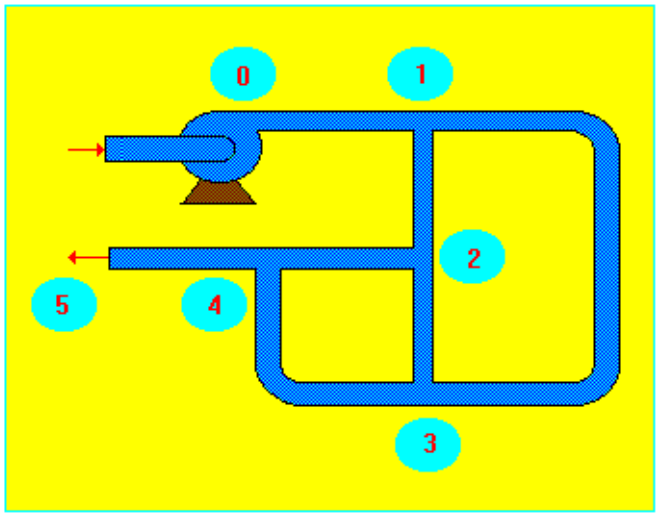


28. példa

Határozzuk meg az alábbi csőhálózat csöveiben a térfogatáramokat, ha azok az egyes csomópontok között az alábbi összefüggéssel számíthatók:

$$Q_{i,j} = b_{i,j} \sqrt{p_i - p_j}$$

ahol i és j a szomszédos csomópontok sorszámai, továbbá $Q_{i,j}$ az i-ből a j-be irányuló térfogatáram, $b_{i,j}$ a csőszakasz ellenállástényezője és p_i a nyomás értéke az i-edik csomópontban.



Az iteráció során a komplex értékek elkerülése miatt célszerű a

$$Q_{i,j} = b_{i,j} \, \text{sign}\left(p_i - p_j\right) \sqrt{\left|p_i - p_j\right|}$$

alakot alkalmazni. Feltesszük, hogy $b_{i,j} = b_{j,i}$.

A megmaradási egyenletek az egyes csomópontokra:

$$F_1\left(p_1, p_2, p_3\right) = Q_{0,1} - Q_{1,2} - Q_{1,3}$$

$$F_2(p_1, p_2, p_3, p_4) = Q_{1,2} - Q_{2,4} - Q_{2,3}$$

$$F_3(p_1, p_2, p_3, p_4) = Q_{1,3} - Q_{3,4} + Q_{2,3}$$

$$F4_3(p_2, p_3, p_4) = Q_{2,4} + Q_{3,4} - Q_{4,5}$$

Adott a belépő és kilépő csonkoknál a nyomás:

$p_0 = 500$
$p_1 = 0.$

valamint az ellenállástényezők értékei:

$b_{0,1} = 0.3;$
$b_{1,2} = 0.1;$

$b_{1,3} = 0.1;$
$b_{2,4} = 0.2;$
$b_{2,3} = 0.2;$
$b_{3,4} = 0.1;$
$b_{4,5} = 0.2;$

Határozzuk meg, hogy a 2 és 3 pontok között milyen irányú az áramlás?!
(feltételezésünk szerint $Q_{2,3} > 0$). Ehhez ki kell számolni a csomópontokban a nyomásokat!

A nyomások kezdeti értékeit vegyük fel lineárisan , azaz $p_1 = \frac{p_0}{2}$, $p_2 = \frac{p_0}{3}$, $p_3 = \frac{p_0}{4}$, $p_4 = \frac{p_0}{5}$.

Mivel a rendszer most elég sok egyenletből áll anonymous függvény helyett m-fájlt írunk:

```
function f = pipenetwork (p)
b01 = 0.3; b13 = 0.1; b12 = 0.1; b24 = 0.2; b23 = 0.2; b34 = 0.1; b45 = 0.2;
p0 = 500; p5 = 0;
f (1) = b01*sign (p0 - p (1))*sqrt (abs (p0 - p (1))) - ...
b12*sign (p (1) - p (2))*sqrt (abs (p (1) - p (2))) - ...
b13*sign (p (1) - p (3))*sqrt (abs (p (1) - p (3)));

f (2) = b12*sign (p (1) - p (2))*sqrt (abs (p (1) - p (2))) - ...
b24*sign (p (2) - p (4))*sqrt (abs (p (2) - p (4))) - ...
b23*sign (p (2) - p (3))*sqrt (abs (p (2) - p (3)));

f (3) = b13*sign (p (1) - p (3))*sqrt (abs (p (1) - p (3))) - ...
b34*sign (p (3) - p (4))*sqrt (abs (p (3) - p (4))) + ...
b23*sign (p (2) - p (3))*sqrt (abs (p (2) - p (3)));

f (4) = b24*sign (p (2) - p (4))*sqrt (abs (p (2) - p (4))) - ...
b45*sign (p (4) - p5)*sqrt (abs (p (4) - p5)) + ...
b34*sign (p (3) - p (4))*sqrt (abs (p (3) - p (4)));
f = f';
```

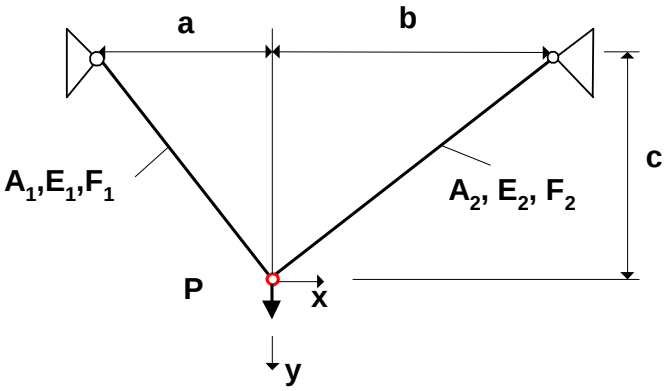
```
% a kezdeti értékek
>> x0 = [1/2 1/3 1/4 1/5]*500
x0 = 250.0000 166.6667 125.0000 100.0000

%alkalmazzuk a Broyden-módszert
>> x = broyden (@pipenetwork, x0', 1 e - 10, 100)

x = 423.1898
```

```
    248.2191
    252.5010
    172.8230

>> norm (pipenetwork (x))
ans = 1.4839 e - 012

% azaz az áramlás  3-as pontból a 2-es felé történik, hiszen p3 > p2!
% vagyis Q23 < 0, mert rosszul vettük fel az irányát.

% a beépített függvénnyel
>> [x,fval]=fsolve(@pipenetwork,x0)
Optimization terminated: first-order optimality is less than options.TolFun.
x =
  423.1898  248.2191  252.5010  172.8230

fval =
  1.0e-006 *

   -0.0059
   -0.1371
    0.1378
   -0.0006

>> norm(fval)
ans =
  1.9443e-007


% Melyik a gyorsabb módszer azonos hibakorlát esetén?

>>  options = optimset('Display', 'off', 'TolFun', 1e-10);
>> tic;x=broyden(@pipenetwork,x0',1e-10,100);toc
Elapsed time is 0.021154 seconds.

>> tic;[x,fval]=fsolve(@pipenetwork,x0',options);toc
Elapsed time is 0.113290 seconds.
```

Megjegyzés: A kezdeti értékek felvétele a műszaki problémák esetén gyakorta nem nehéz, mivel a mérnök általában tudja "műszaki has", tapasztalat és szaktudás alapján, hogy milyen eredmény várható!

29. példa

Adott az alábbi rúdszerkezet feszültségmentes helyzete:



ahol A_i , E_i az i – edik rúd keresztmetszete és rugalmassági modulzsza

Ha a rúdak deformációja nem hanyagolható el az egyensúlyi egyenletekben sem, akkor a közös csomópont elmozdulására az alábbi egyenletrendszer írható fel:

A rúdak megváltozott hossza:

$$S1 = \sqrt{(a + x)^2 + (c + y)^2}$$

$$S2 = \sqrt{(b - x)^2 + (c + y)^2}$$

A rúderők :

$$F1 = \left(S1 - \sqrt{a^2 + b^2}\right) E1\, A1$$

$$F2 = \left(S2 - \sqrt{b^2 + c^2}\right) E2\, A2$$

Az egyensúlyi egyenletek :

$$f_1(x, y) = F1 \frac{a + x}{S1} - F2 \frac{b - x}{S2} + P_x = 0$$

$$f_2(x, y) = -F1 \frac{c + y}{S1} - F2 \frac{c + y}{S2} + P_y = 0$$

Határozzuk meg az alsó csomópont elmozdulását (x, y) ha P_x =0 és P_y = 100 kN.

A szerkezet adatai:

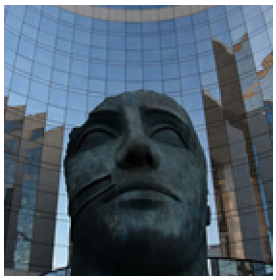
a = 1 m;
b = 1.5 m;
c = 1.2 m;
$E_1 = 2 \times 10^7$ kN/m^2
$E_2 = 0.5 \times 10^7$ kN/m^2
$A_1 = 0.00001$ m^2
$A_2 = 0.00002$ m^2

```
function y = bar (X)
global a b c E1 E2 A1 A2 Px Py;
x = X (1); y = X (2);
S1 = sqrt ((a + x)^2 + (c + y)^2);
S2 = sqrt ((b - x)^2 + (c + y)^2);
F1 = (S1 - sqrt (a^2 + b^2))*A1*E1;
```

```
F2 = (S2 - sqrt (b^2 + c^2))*A2*E2;
f (1) = F1*(a + x)/S1 - F2*(b - x)/S2 + Px;
f (2) = -F1*(c + y)/S1 - F2*(c + y)/S2 + Py;
y = f ‘; %transzponált
```

```
>>global a b c E1 E2 A1 A2 Px Py;
>> a = 1; b = 1.5; c = 1.2; E1 = 2 e7; E2 = 0.5 e7; A1 = 1 e - 5; A2 = 2*A1;
>> Px = 0; Py = 1 e2;
>> format long

>> [x,fval] = fsolve (@bar, [0, 0]')
Optimization terminated : first - order optimality is less than options.TolFun.

x =
-0.009364151198698    0.700508379200921

fval = 1.0 e - 013*
-0.284217094304040   0.142108547152020


%Nézzük meg a Broyden-módszer alkalmazásával is és mérjük az időket!

>> tic;[x,fval] = fsolve(@bar, [0, 0]’);toc
Optimization terminated: first-order optimality is less than options.TolFun.
Elapsed time is 0.014331 seconds.
>> x
x =
  -0.009364151198698
   0.700508379200921

>> fval
fval =
  1.0e-013 *
  -0.284217094304040
   0.142108547152020

 % A Broyden érzékenyebb a kezdőértékre!

>> tic;xs =broyden(@bar,[0.1;0.1],1e-14,500);toc
Elapsed time is 0.006181 seconds.
>> xs
xs =
  -0.009364151198697
   0.700508379200921

>> bar(xs)
ans =
  1.0e-013 *
   0.675015598972095
  -0.284217094304040

% Inverz nélküli Broyden
```

```
>> tic;xs =broydenSM(@bar,[0.1;0.1],1e-14,500);toc
Elapsed time is 0.005066 seconds.
>> xs
xs =
  -0.009364151198699
   0.700508379200921

>> bar(xs)
ans =
  1.0e-012 *
  -0.113686837721616
   0.056843418860808
```



8. Gyakorlat

Lokális optimum megkötés nélkül

30. példa

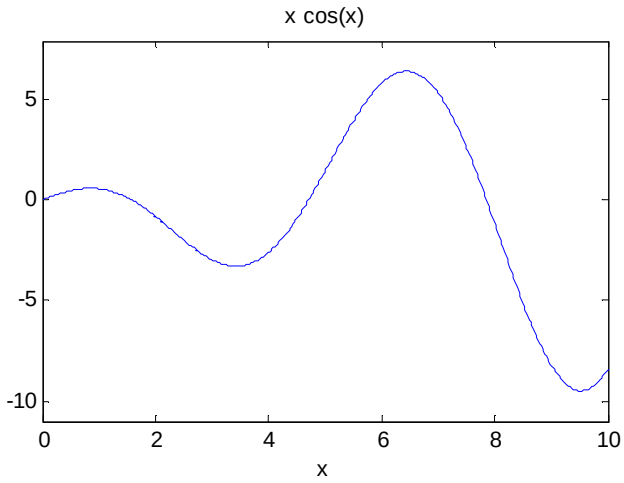
Keressük meg az alábbi függvény egy lokális minimumát illetve maximumát.

$$f(x) = x \cos(x)$$

az $x \in [0, 8]$ intervallumban.

Newton - módszer

```
>> clear all; clc
>> f = @(x) x*cos (x);

>> ezplot (f, [0, 8])
```



Most az iterációs formula,

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

```
%állítsuk elő az első és második deriváltakat
>> diff (' x*cos (x)')
ans =
    cos (x) - x*sin (x)

 % a második derivált
```

```
>> diff (' x*cos (x)', 2)
ans =
    -2*sin (x) - x*cos (x)

% a megfelelő függvények
>> F = @(x) cos (x) - x*sin (x);
>> dF = @(x) - 2*sin (x) - x*cos (x);

%alkalmazzuk a Newton módszert az F(x) = 0 egyenlet megoldására
% 6. gyak. 23 példa

>> xsol = newton (F, dF, 3, 1 e - 6, 10)
xsol =
     3.425618459481985      4.000000000000000

%rajzoljuk fel a megoldást
>> hold on
>> plot (xsol (1), f (xsol (1)), ' ro')

% határozzuk meg a maximumot
>> xsol = newton (F, dF, 6, 1 e - 6, 10)
xsol =
    6.437298179171948     4.000000000000000

>> plot (xsol (1), f (xsol (1)), ' go')

% mindezek beépített függvénnyel
>> fsolve (F, 3)
Optimization terminated : first - order optimality is less than options.TolFun.
ans =
    3.425618459482001

>> fsolve (F, 6)
Optimization terminated : first - order optimality is less than options.TolFun.
ans =
   6.437298179171949
```

Gradiens - módszer

A számítás elvégzése gyorsítható, ha a második deriváltat nem kell kiszámítatni minden lépésnél, azaz

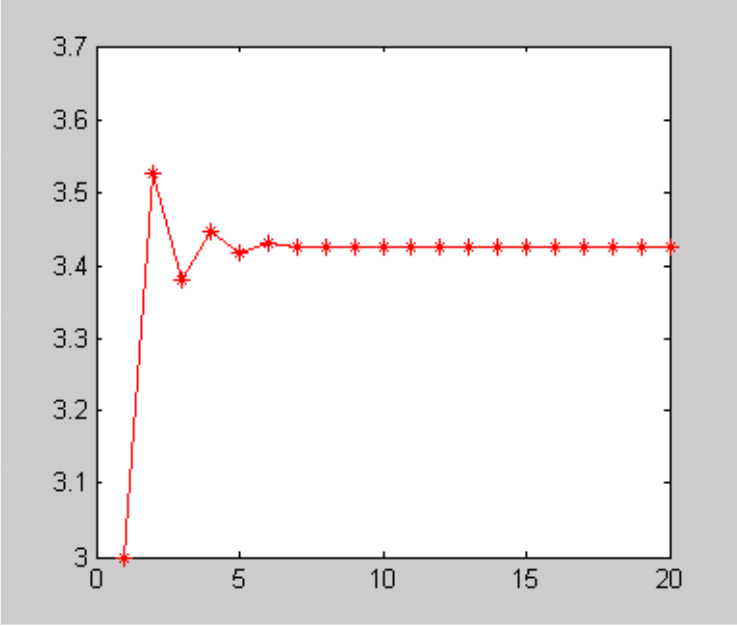
$$x_{k+1} = x_k - \alpha f'(x_k)$$

ahol most legyen α állandó,

$$\alpha = \frac{1}{f''(x_0)}$$

```
% keressük a minimumot x0 = 3
>> alfa = 1/dF (3);
>> g = @(u) u - alfa*F (u)
g =
 @(u) u - alfa*F (u)

>> i = 1;
```

```
>> x (1) = 3;
>> nmax = 15;
>> while and (abs (f (x (i))) > 1 e - 6, i < nmax)
x (i + 1) = g (x (i));
i = i + 1;
end

% az eredmény: az iteráció lépéseinek x koordinátái
>> x'
ans = 3.000000000000000
3.525852146878046
3.379011336322909
3.444949109028904
3.417136314477693
3.429256808336883
3.424042010355576
3.426298582864544
3.425324487087243
3.425745422078296
3.425563607008830
3.425642154189161
3.425608223377631
3.425622881350543
3.425616549266612
3.425608344243065
3.425613576138620
3.425616101948963
3.425617321335818
3.425617910019295

% ábrázoljuk az iteráció alakulását a függvény gráfján
figure(2)
plot(x','r*-')
```



Túllendülést tapasztalunk, tehát lehetett volna α értéke kisebb, pl.

$$\alpha = \frac{\beta}{f''(x_0)}$$

ahol $\beta < 1$

Intervallum - módszer (aranymetszés)

```
function y = golden (f, a, b, r, Tolx, Tolf) %rekurzív verzió
h = b - a;
c = a + (1 - r)*h; % r a felosztás paramétere: egyenlő felosztás esetén: 2/3
d = a + r*h;
fc = f (c);
fd = f (d);
if (abs (h) < Tolx || abs (fc - fd) < Tolf) % leállási feltételeknél: or →||
    if fc <= fd
      y = c;
   else
      y = d;
   end
else
    if fc < fd
      y = golden (f, a, d, r, Tolx, Tolf);
   else
      y = golden (f, c, b, r, Tolx, Tolf);
    end
 end
```

```
% az intervallum
xa=2; xb=5;

% az aranymetszés arány
m=0.618;

xsol = golden (f, xa, xb, m, 1 e - 6, 1 e - 6)
xsol =
   3.425682085397058
```

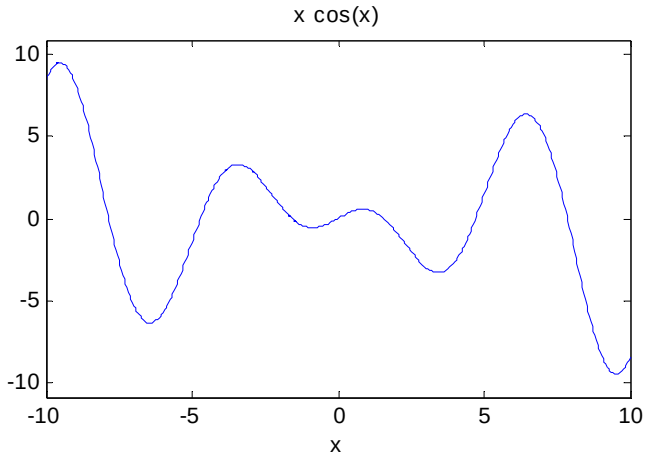
Módosítsuk a *golden* függvényt úgy, hogy az iterációk számát is megkapjuk!

```
function y = golden (f, a, b, r, Tolx, Tolf)
global iter;
iter = iter + 1;
h = b - a;
c = a + (1 - r)*h;
d = a + r*h;
fc = f (c);
fd = f (d);
if (abs (h) < Tolx || abs (fc - fd) < Tolf)
    if fc <= fd
       y = c;
    else
       y = d;
   end
```

```
else
    if fc < fd
       y = golden (f, a, d, r, Tolx, Tolf);
    else
       y = golden (f, c, b, r, Tolx, Tolf);
    end
end
```

```
>>global iter

% Nézzük meg az aranymetszés arány esetén
>> iter = 0;
>> xsol = golden (f, xa, xb, 0.618, 1 e - 6, 1 e - 6)
xsol = 3.425682085397058

>> iter
iter = 16

% ha a felosztás 1/3 - 2/3 azaz
>> iter = 0;
>> xsol = golden (f, xa, xb, 2/3, 1 e - 6, 1 e - 6)
xsol = 3.425969874298517

>> iter
iter = 18

% kevesebb iteráció kell az arnymetszés esetén, és valamivel pontosabb eredményt is kapunk , hiszen

f(3.425682085397058) < f(3.425969874298517)
ans =
    1
```

Határozzuk meg a függvény globális minimumát a [-10, 10] intervallumban!



Alkalmazzunk genetikus algoritmust!

```
>> xmin = ga (f, 1)
Optimization terminated : stall generations limit exceeded.xmin = -0.8570

>> f (xmin)
```

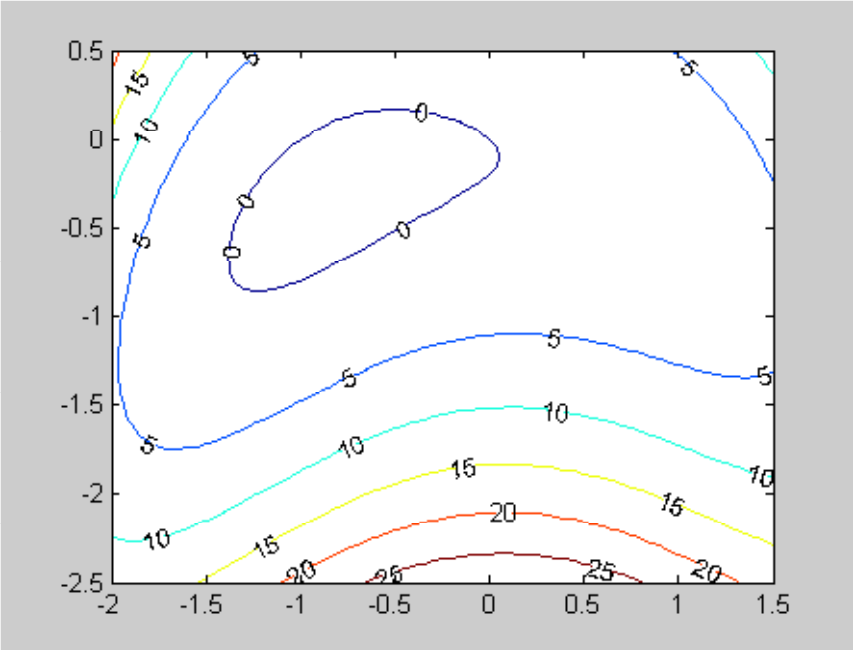
```
ans = -0.5611

>> options = gaoptimset('Generations', 1000, 'PopulationSize', 200);
>> ga(f,1,options)
Optimization terminated: stall generations limit exceeded.
ans =
   9.5297

% javítjuk az eredmény lokális módszerrel

fminsearch(f,ans)
ans =
   9.5294

%Ez jobb

>> f(9.5294)<f(9.5297)
ans =
    1
```

- **31. példa**

Határozzuk meg az alábbi függvény lokális minimumát az x×y =[-2,1.5]×[-2.5. 0.5] intervallumban!

$$F(x, y) = x^4 + 3x^2 y + 5y^2 + x + y$$

```
% ábrázoljuk szintvonalas ábrán a minimum környezetét - "ponttal definiálva"
f = @(x, y) x.^4 + 3*(x.^2).*y + 5*y.^2 + x + y;

>>% Létrehozunk egy x- y rácsot a függvényértékekkel, 0.05 felbontással
>> [X Y]=meshgrid(-2:0.05:1.5,-2.5:0.05:0.5);
>> z = f (X, Y);

>>% a szintvonalak
>>[C, h] = contour (X ,Y, z);

>> % az értékek
>>clabel (C, h)
```



Newton-módszer

Alkalmazzuk a Newton módszert most többváltozós esetre,

$$f(x_{k+1}) \approx f(x_k) + \nabla f(x_k)^T (x_{k+1} - x_k) + \frac{1}{2} (x_{k+1} - x_k)^T H(x_k) (x_{k+1} - x_k)$$

azaz

$$\Delta f = \nabla f(x_k)^T \Delta x + \frac{1}{2} \Delta x^T H(x_k) \Delta x$$

ahol $\nabla f(x)$ az $f(x)$ függvény gradiens-vektora és H a *Hesse-mátrix*, az $f(x)$ függvény második parciális deriváltjainak a mátrixa

$$H(x)_{i,j} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \begin{pmatrix} \frac{\partial}{\partial x_1}\left(\frac{\partial f(x)}{\partial x_1}\right) & \cdots & \frac{\partial}{\partial x_n}\left(\frac{\partial f(x)}{\partial x_1}\right) \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \vdots & \cdots & \vdots \\ \frac{\partial}{\partial x_1}\left(\frac{\partial f(x)}{\partial x_n}\right) & \cdots & \frac{\partial}{\partial x_n}\left(\frac{\partial f(x)}{\partial x_n}\right) \end{pmatrix}$$

A minimum szükséges feltétele alapján, azaz $\Delta f = f(x_{k+1}) - f(x_k)$ -t deriválva $\Delta x = x_{k+1} - x_k$ szerint, a derivált zérus,

$$0 = \nabla f(x_k)^T + H(x_k) \Delta x$$

azaz

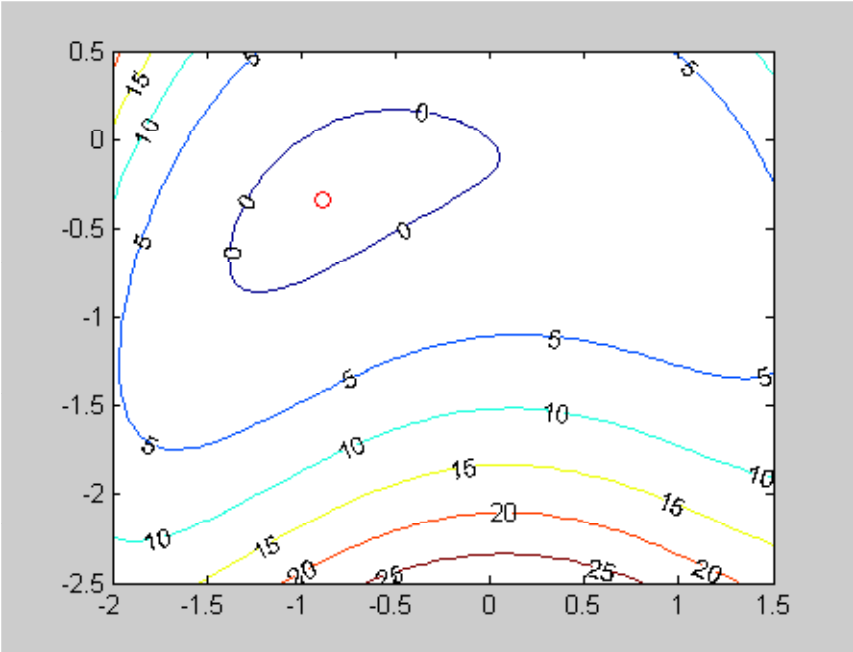
$$H(x_k) x_{k+1} = H(x_k) x_k - \text{grad}(f(x_k))$$

```
function y = gradmulti (f, x0, grad, hesse, eps, nmax)
x1 = x0;
A = hesse (x1);
b = hesse (x1)*x1 - grad (x1);
x2 = linsolve (A, b);
i = 1;
while and (norm (x1 - x2) > eps, i < nmax)
x1 = x2;
```

```
A = hesse (x1);
b = hesse (x1)*x1 - grad (x1);
x2 = linsolve (A, b);
i = i + 1;
end
y = x2;
```

```
% A gradiens és a Hesse mátrix előállítása
>> syms x y

% a parciális deriváltak

>> dfx = diff (f (x, y), x)
dfx =
   4*x^3 + 6*x*y + 1

>> dfy = diff (f (x, y), y)
 dfy =
   3*x^2 + 10*y + 1

%ezekkel a gradiens vektor
>> grad = @(x, y)[4*x^3 + 6*x*y + 1, 3*x^2 + 10*y + 1];

% A második parciális deriváltak
 >> dfxx = diff (dfx, x)
 dfxx =
   12*x^2 + 6*y

 >> dfxy = diff (dfx, y)
 dfxy =
    6*x

>> dfyy = diff (dfy, y)
 dfyy =
   10
% ezekkel a Hesse mátrix

  >> hesse = @(x, y)[12*x^2 + 6*y, 6*x; 6*x, 10];

 % gradmulti az általánosítás miatt vektorváltozokkal (oszlopvektorokkal) dolgozik
 %ezért újra definiáljuk  amit újra kell definiálni

  >> F = @(u) f (u (1), u (2))

  >> G = @(u) grad (u (1), u (2))';

  >> H = @(u) hesse (u (1), u (2));

 %Legyen az ábra alapján az indulóérték

>> u0 = [0.72; -2] ;
```

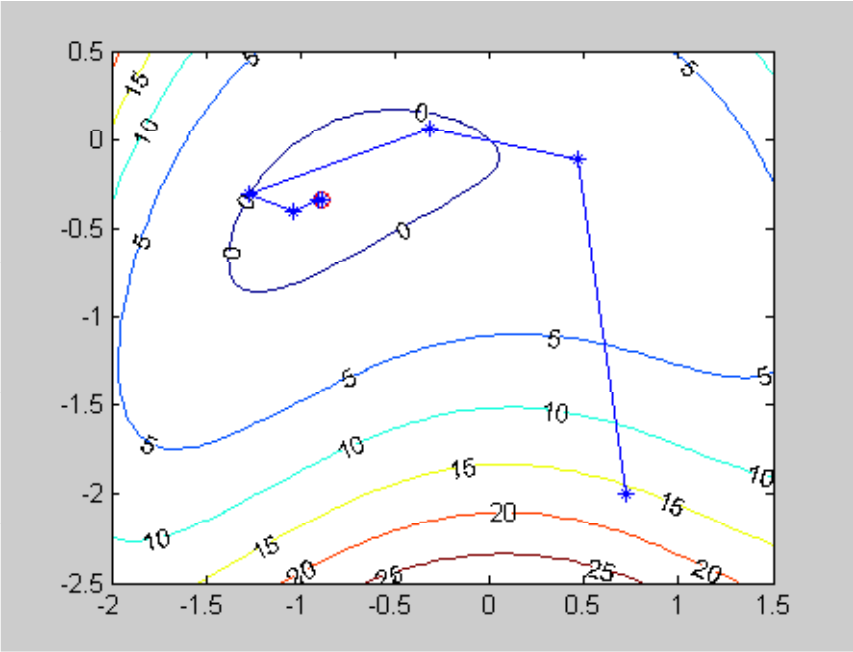
```
% a megoldás
>> usol=gradmulti(F,u0,G,H,1e-6,100)

usol =

  -0.886324206645524
  -0.335671179785745

% rajzoljuk be az ábrába
 >> hold on
>> plot(usol(1),usol(2),'ro')
```



Módosítsuk a *gradmulti* - t, úgy hogy kiadja az iterációs lépések koordinátáit!

```
function y = gradmulti (f, x0, grad, hesse, eps, nmax)
global X Y;
X = [x0 (1)];
Y = [x0 (2)];
x1 = x0;
A = hesse (x1);
b = hesse (x1)*x1 - grad (x1);
x2 = linsolve (A, b);
i = 1;
while and (norm (x1 - x2) > eps, i < nmax)
x1 = x2;
A = hesse (x1);
b = hesse (x1)*x1 - grad (x1);
x2 = linsolve (A, b);
i = i + 1;
X = [X, x2 (1)];
Y = [Y, x2 (2)];
end
y = x2;
```

```
>> clear X Y
>> global X Y;
>> usol = gradmulti (F, u0, G, H, 1 e - 6, 100)

usol = -0.886324206645524
         -0.335671179785745

>> plot (X', Y', ' b*-')
```

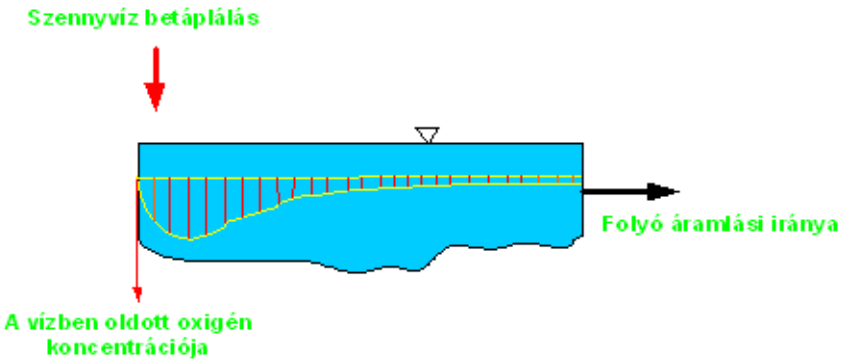


Szimplex módszer - beépített fminsearch

```
>> usol = fminsearch (F, u0, optimset (' TolX', 1 e - 8))

usol = -0.886324212383607
         -0.335671183568364
```

- 32. példa

Környezetvédelmi okokból, a víz élővilágának védelme érdekében a folyók vízben oldott oxigén koncentrációjának minimális értéke nem lehet kisebb, mint egy kritikus minimum!



Az ábra a koncentráció változását mutatja a szennyező anyag tartozkódási idejének függvényében. A tartozkódási idő -elhagyagolható diffuzió mellett- arányos a betáplálástól mért távolsággal. Az arányossági tényező a folyó

átlagos lineáris sebessége az adott szakaszon,

$$c(t) = c_s - \frac{k_d L_o}{k_d + k_s - k_a}\left(e^{-k_a t} - e^{-(k_d+k_s)t}\right) - \frac{S_b}{k_a}\left(1 - e^{-k_a t}\right)$$

ahol

c – oldott oxigén koncentráció, mg/L, függő változó

c_s – a telítési koncentráció, 10 mg/L

t – tartózkodási idő, nap, független változó

L_o – BOD (biochemical oxigen demand) biokémiai oxigénigény a betáplálásnál, 50 mg/L

k_d – BOD lebomlási sebesség, 0.1 1/nap

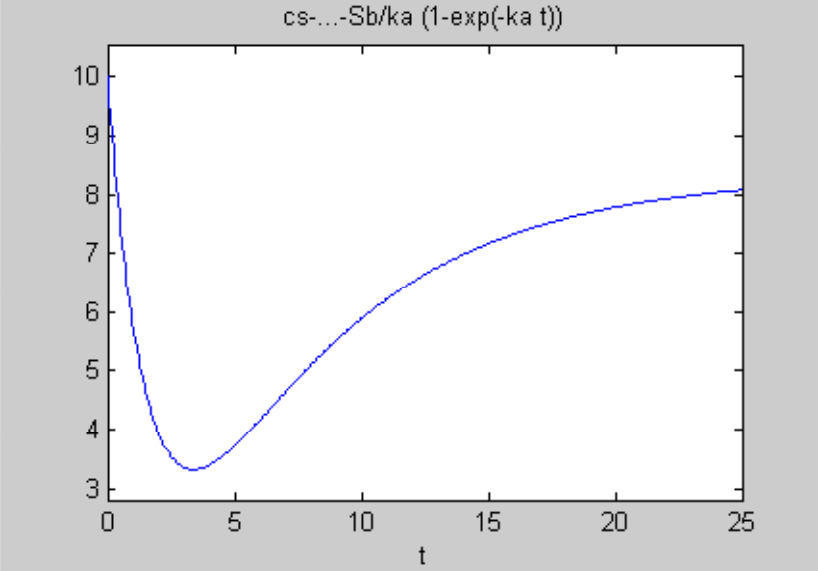
k_s – BOD kiülepedési sebessége, 0.05 1/nap

k_a – átlevegőzési sebessége, 0.6 1/nap

S_b – kiülepedési oxigénigény 1 mg/L/nap

Határozzuk meg az alábbi esetben a folyó minimális oxigén koncentrációját a szennyvíz betáplálás közelében!

```
clear all; clc

cs = 10; kd = 0.1; ka = 0.6; ks = 0.05; Lo = 50; Sb = 1;
c = @(t) cs - kd*Lo/(kd + ks - ka)*(exp (-ka*t) - exp (-(kd + ks)*t)) - Sb/ka*(1 - exp (-ka*t));
ezplot (c, [0, 25])
```



Newton - módszer

```
>> syms t
>> df = diff (c (t), t)
df =
    -23/3*exp (-3/5*t) + 5/3*exp (-3/20*t)

>> ddf = diff (c (t), t, 2)
ddf =
     23/5*exp (-3/5*t) - 1/4*exp (-3/20*t)

>> F = @(t) - 23/3*exp (-3/5*t) + 5/3*exp (-3/20*t)
```

```
>> dF = @(t) 23/5*exp (-3/5*t) - 1/4*exp (-3/20*t)

>> cmin = newton (F, dF, 4, 1 e - 6, 20)
cmin =
    3.391236229988999 5.000000000000000
```

Beépített fzero függvénnyel

Most az *fsolve* helyett alkalmazzuk az *fzero* függvényt:

```
>> cmin = fzero (F, 4)
cmin =
    3.391236229988999
```

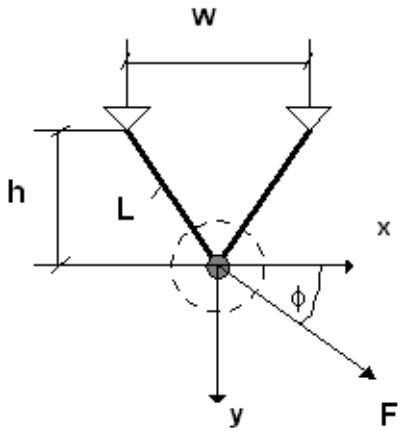
Aranymetszés

```
>> global iter
>> iter = 0;
>> cmin = golden (c, 0, 4, 0.618, 1 e - 6, 1 e - 8)
cmin =
  3.391228414375473

>> iter
iter = 19
```

Megjegyzés: Általában egy feladat megoldását igyekezzünk többféle módon is előállítani!

- **33. példa**

Tekintsük az alábbi kerékpárváz háromcsuklós tartóként definiált modelljét:



A szerkezet potenciális energiája:

$$V(x, y) = \frac{E\,A}{L}\left(\frac{w}{2\,L}\right)^2 x^2 + \frac{E\,A}{L}\left(\frac{h}{L}\right)^2 y^2 - F_x \cos(\phi) - F_y \sin(\phi)$$

ahol

E – rugalmassági modulusz, 2×10^{11} Pa

A – keresztmetszeti terület, 0.0001 m2

w – fesztáv, 0.44 m

L – hossz, 0.56 m

h – vázmagasság, 0.52 m

F – terhelés, 5×10^4 N

ϕ – pozíció, $0 < \phi < 90^0$ legyen $\phi = 30^0$

Határozzuk meg a csukló elmozdulását (x, y)!

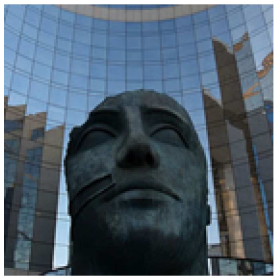
Simplex módszer - Nelder- Mead

```
>> E = 2 e11; A = 0.0001; w = 0.44; L = 0.56; h = 0.52; F = 5 e4; fi=30;
>> V = @(x) E*A/L*(w/2/L)^2*x (1)^2 + E*A/L*(h/L)^2*x (2)^2 - F*x (1)*cos (fi*pi/180) - ...
F*x (2)*sin (fi*pi/180);

% a kezdő iterációs érték legyen x=[0 0]
>> fminsearch (V, [0, 0], optimset (' Display', ' iter', ' TolFun', 1 e - 8))

Iteration              Func - count          min f (x)              Procedure
0                          1                    0
1                          3                 - 10.4808           initial simplex
2                          5                 - 20.5074           expand
3                          7                 - 37.1926           expand
4                          9                 - 54.8522           expand
5                         11                 - 83.6357           expand
6                         12                 - 83.6357           reflect
.....................................................................................................
32                        63                 - 90.1153           contract inside
33                        65                 - 90.1153           contract inside
34                        67                 - 90.1153           contract inside
35                        69                 - 90.1153           contract inside
36                        71                 - 90.1153           contract outside
.....................................................................................................
41                         81                - 90.1153           contract inside

Optimization terminated : the current x satisfies the termination criteria using
OPTIONS.TolX of 1.000000 e - 004
and F (X) satisfies the convergence criteria using
OPTIONS.TolFun of 1.000000 e - 008


ans =
   0.003927868944563           0.000405911242272
```

Newton - módszer -beépített: fminunc

```
>> xsol = fminunc (V, [0, 0])
Warning : Gradient must be provided for trust - region method;
using line - search method instead.
 > In fminunc at 281
Optimization terminated : relative infinity - norm of gradient less than options.TolFun.
```

```
xsol =
    0.003927882975903     0.000405909709162
```



10. Gyakorlat

Lokális, globális és spline interpoláció egy és kétváltozó esetén, szabályos és szabálytalan eloszlású pontokon.

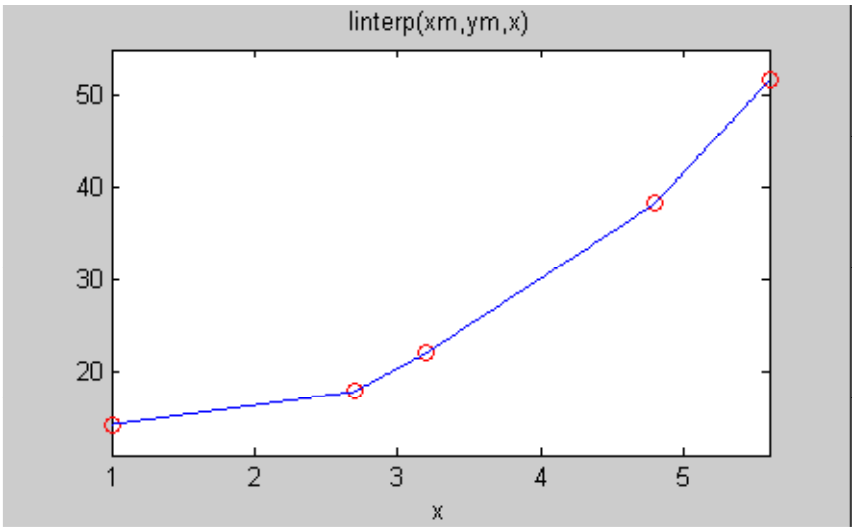
39. példa

Adottak alábbi *adatpontok* (x, y) koordinátái:

x = (1, 2.7, 3.2, 4.8, 5.6), y= (14.2, 17.8 ,22 ,38.3, 51.7).

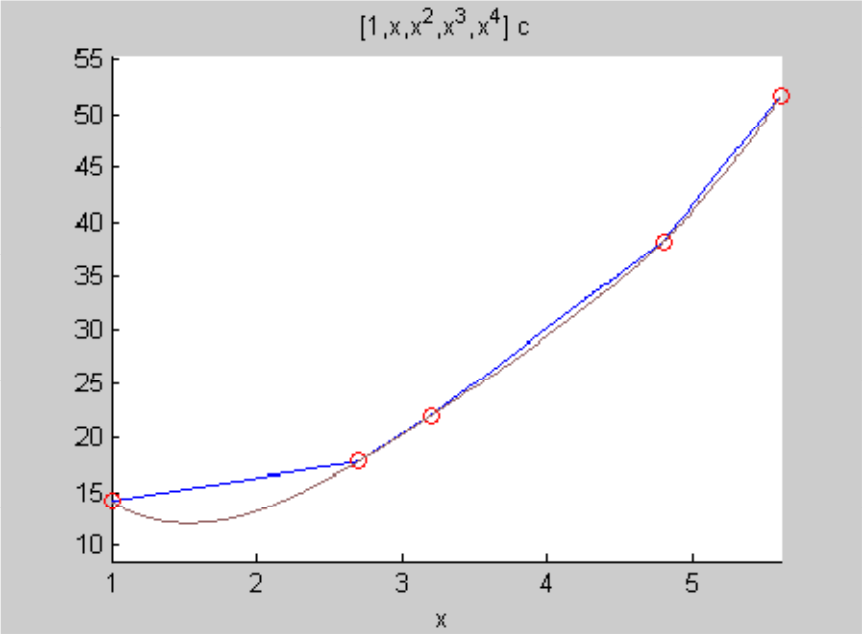
Alkalmazzunk lineáris és polinomiális interpolációt és állítsuk elő a két interpolációs eljárás közötti eltérés függvényét a x∈[1, 5.6] tartományban.

```
>> xm = [1 2.7 3.2 4.8 5.6];
>> ym = [14.2 17.8 22 38.3 51.7];

% az adatok vizualizációja
>> plot(xm,ym,'ro');

% a lineáris interpoláció beépített függvénnyel
>> yL=@(x)linterp(xm,ym,x);
>> hold on
>> ezplot(yL,[1,5.6]);
```



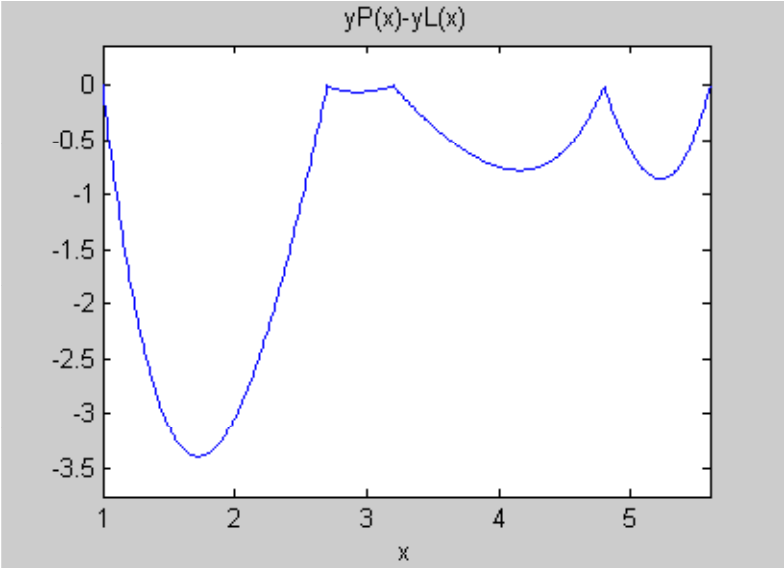

```
>>% A polinomiális interpoláció meghatározása "gyalog" módszerrel
>>A együtthatók meghatározására szolgáló lineáris egyenletrendszer mátrixának
```

```
>>% előállítása oszloponként
>> n=length(xm);
>> M(1:5,1)=ones(size(xm));
>> for j=2:5
M(1:n,j)=xm.^(j-1);
end
>> M
M =
    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    2.7000    7.2900   19.6830   53.1441
    1.0000    3.2000   10.2400   32.7680  104.8576
    1.0000    4.8000   23.0400  110.5920  530.8416
    1.0000    5.6000   31.3600  175.6160  983.4496

%Mint tudjuk ez egy Vandermonde tipusú mátrix, amely  n > 6 - 8 eesetén rosszul-kondicionált!
% már most is (n=5) meglehetősen nagy a kondiciószáma:
>> cond(M)
ans =
  6.1948e+004

%Az együtthatók meghatározása
>> c=linsolve(M,ym')
c =
   34.9600
  -36.1836
   18.6885
   -3.5208
    0.2558

% Az interpolációs függvény anonymous függvénye
>> yP=@(x)[1 x x^2 x^3 x^4]*c;

>>  ezplot(yP,[1,5.6]);
```




```
% Az eltérés függvény
>> d=@(x)yP(x)-yL(x)
```

```
d =
    @(x)yP(x)-yL(x)
>> figure(2)
>> ezplot(d,[1,5.6])
```




```
% Az együtthatók meghatározása beépített függvény segítségével
>> p = polyfit (xm, ym, n-1)
p = 0.2558 - 3.5208 18.6885 - 36.1836 34.9600
% ezzel a megfelelő anonymous függvény
>> y=@(x)polyval(p, x)
```

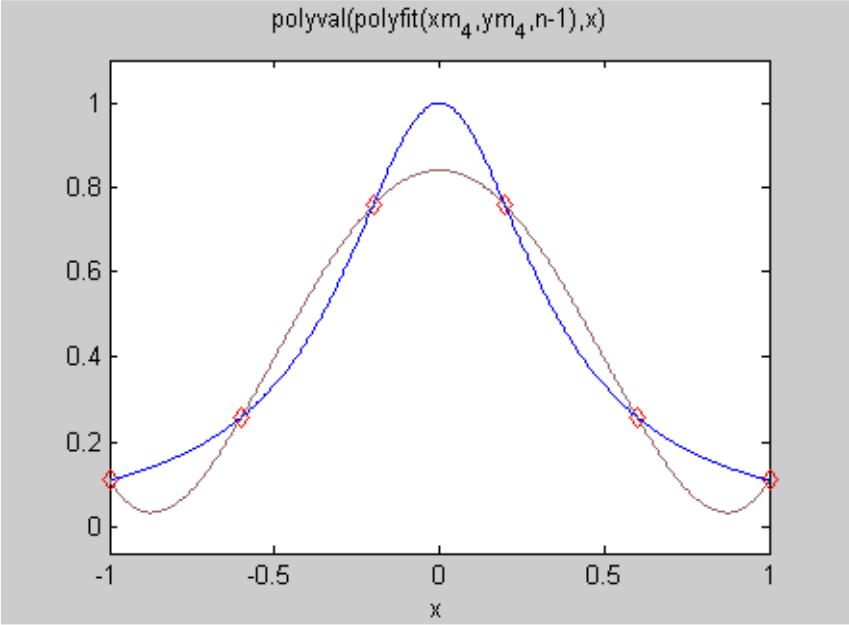
40. példa

Tekintsük az alábbi *függvény közelítését* a [-1, 1] intervallumban, 4 illetve 10 belső osztópont esetén.

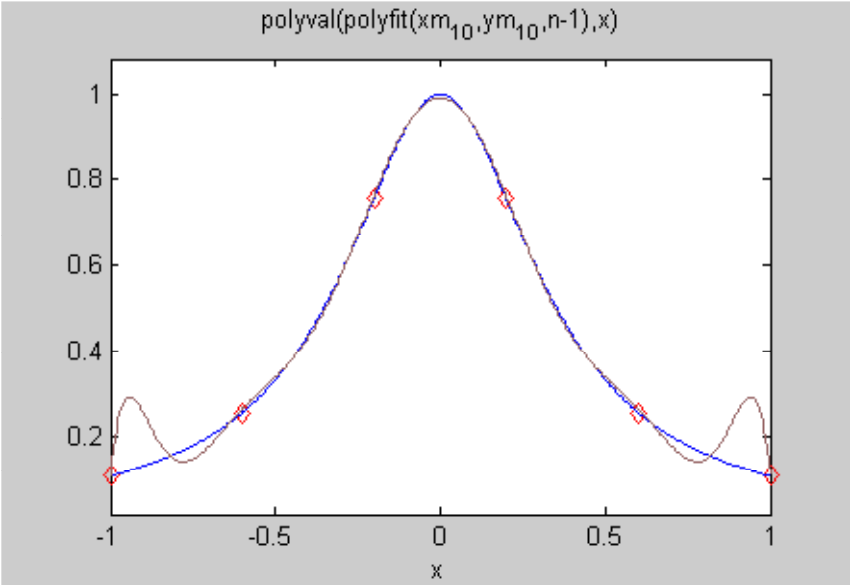
$$f(x) = \frac{1}{1 + 8\,x^2}$$

```
%Az eredeti függvény
>> f = @(x) 1/(1 + 8*x^2);

% A pontok koordinátái 4 belső pont esetén
>> d = 2/5;
>> for i = 1 : 6
xm4 (i) = -1 + (i - 1)*d;
ym4 (i) = f (xm1 (i));
end;

% a polinomiális interpoláció
>> n = length (xm4) ;
>> p4 = @(x) polyval (polyfit (xm4, ym4, n-1), x);

%ábrázolás
>> plot (xm4, ym4, ' rd');
```

```
>> hold on
>> ezplot (f, [-1, 1]);
>> ezplot (p4, [-1, 1]);
```




```
% A hiba jelentős, talán több belső pont segít csökkenteni a hibát?!
% ismételjük meg az előbbieket 10 belső pontra!
>> d = 2/11;
>> for i = 1 : 12
xm10(i)=-1 +(i -1)*d;
ym10(i)= f(xm10(i));
end
>> n=length(xm10);
>> p10=@(x)polyval(polyfit(xm10, ym10, n-1),x);
>> figure(2)
>> plot (xm4, ym4, 'rd');
>> hold on
>> ezplot (f, [-1, 1]);
>> ezplot (p10, [-1, 1]);
```



Valóban jobb a közelítés az intervallumon belül, de annak szélein "hullámzás" lép fel (*Runge-jelenség*), mivel n =11!

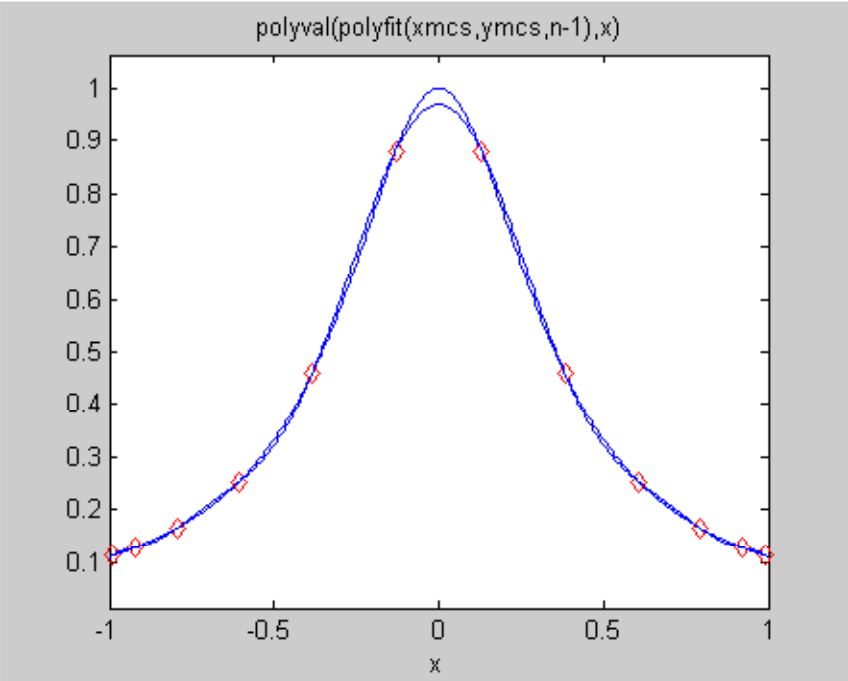
A probléma egyik megoldása- ha az alappontokat tetszőlegesen felvehetjük - a *Csebisev- osztópontok* alkalmazása:

$$x_k = \cos\left(\frac{2\,N + 1 - 2\,k}{2\,(N+1)}\,\pi\right), \quad k = 0,\ 1,\ \ldots,\ N$$

ahol a belső pontok száma N -1, továbbá

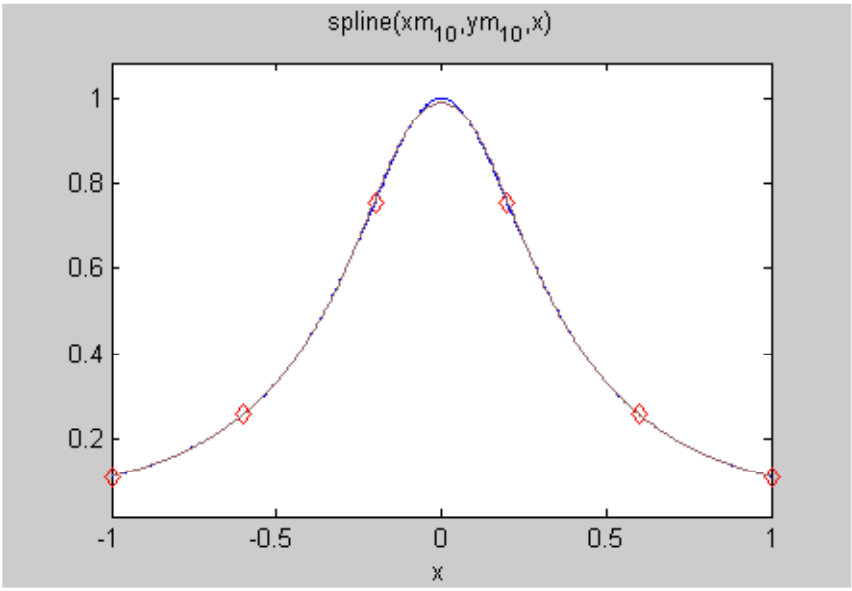
$$x_0 = -1 \quad \text{és} \qquad x_N = 1$$

```
% esetünkben a Csebisev pontok
>> for i = 1 : 12
xmcs (i) = cos ((2*11 + 1 - 2*(i - 1))/(2*12)*pi);
ymcs (i) = f (xmcs (i));
end
>> n = length (xmcs)
n = 12

%a megfelelő  interpolációs polinom
>> pcs = @(x) polyval (polyfit (xmcs, ymcs, n - 1), x);

% ábrázoljuk
>> figure (3)
>> plot (xmcs, ymcs, ' rd');
>> hold on
>> ezplot (f, [-1, 1]);
>> ezplot (pcs, [-1, 1]);
```



Gyakran azonban az adatpontok helye adottság, pl mérések esetén. Ekkor más interpolációs módszert alkalmazunk, amely kicsit lokális és egy kicsit globális is:
Spline interpoláció!

```
%beépített függvényt alkalmazva
>> sp = @(x) spline (xm10, ym10, x);

>> figure (4)
>> plot (xm4, ym4, ' rd');
>> hold on
>> ezplot (f, [-1, 1]);
>> ezplot (sp, [-1, 1]);
```



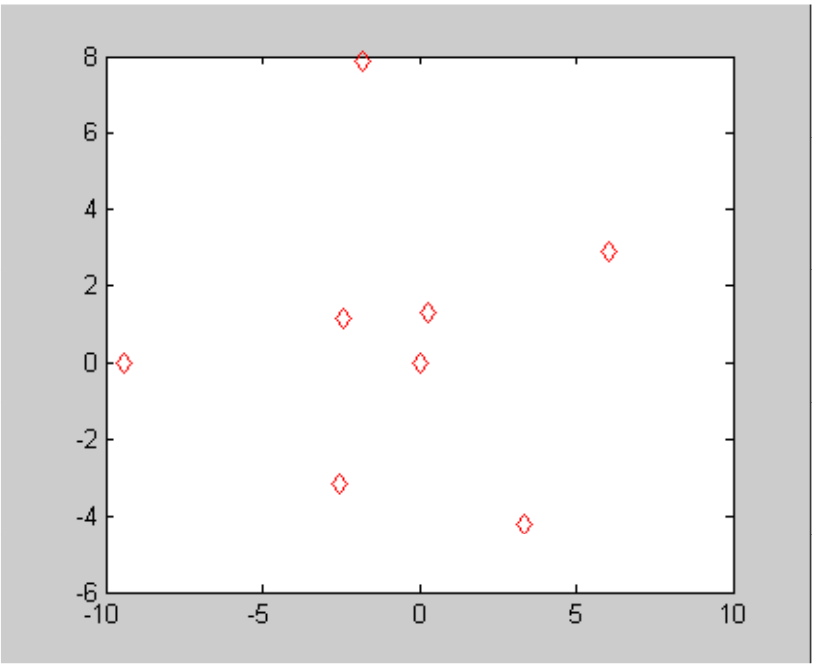

```
% igen jó közelítést ad - az sem tudjuk melyik görbe melyik?!
>> sp (0)
```

```
ans = 0.9888

>> f (0)
ans = 1
```

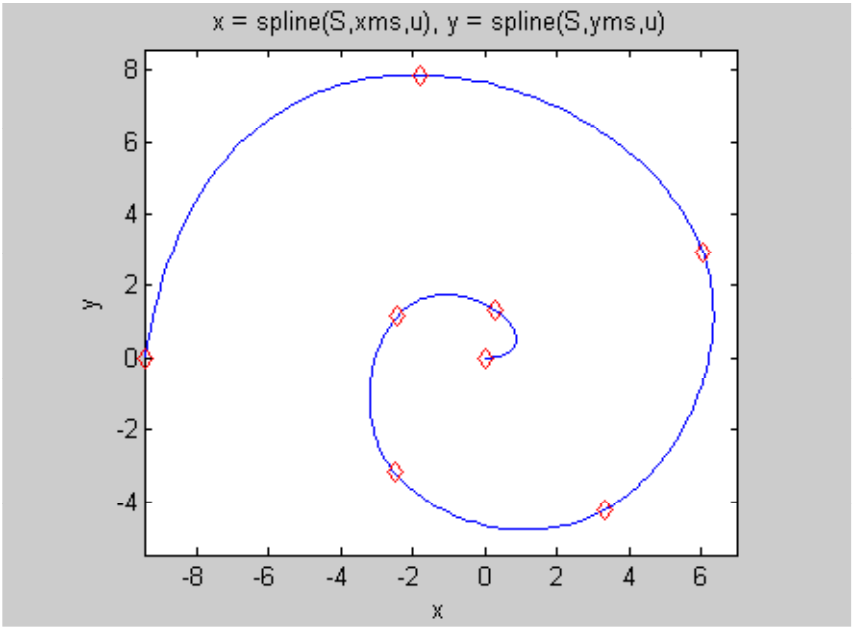
41. példa

A spline interpoláció jól alkalmazható olyan görbék esetében is, amelyeknél a függvény adott x_i értéknél nem egy, hanem több különböző y_i értéket vesz fel, pl. egy körvonal. Mint tudjuk, ilyen esetekben a görbe paraméteres alakját alkalmazzunk. Alkalmazzunk spline interpolációt egy spirális görbe közelítésére, annak ismert diszkrét pontjai alapján!

```
% a spirális paraméteres egyenletei -(ponttal definiálva, azaz vektoros formában)
>> xa=@(t)t.*cos(t);
>> ya=@(t)t.*sin(t);

% legyen a pontok száma: n = 8
>> n=8;

% lépésköz a paraméterben 3 körülfordulást tekintve
>> dt=3*pi/(n-1);

%a diszkrét pontok előállítása
>> t=0;
>> for i=1:n
xms(i)=xa(t);
yms(i)=ya(t);
t=t+dt;
end;

% ábrázoljuk a pontokat
>> plot(xms,yms,'rd');
```

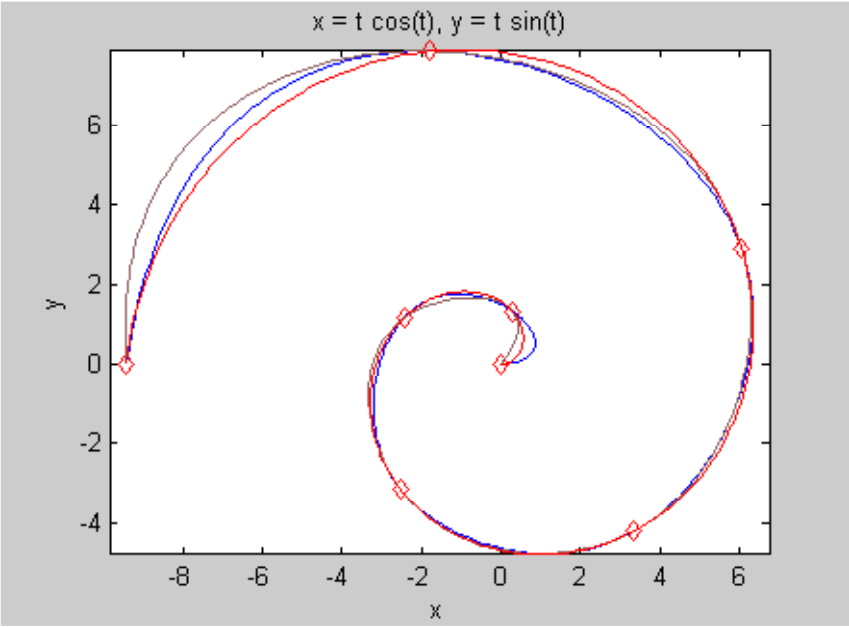



```
% az interpolációhoz paraméterként válaszhatjuk a pontok sorszámát:
>>S(1:8)=1:8
S =
    1    2    3    4    5    6    7    8

% az interpolációs függvények
>> xS=@(u)spline(S,xms,u);
>> yS=@(u)spline(S,yms,u);

%rajzoljuk fel
>> hold on
>> ezplot(xS,yS,[1,8])
```



Egy másik lehetőség ha paraméterként a görbe ívhosszát választjuk!

```
% az ívhossz értékeit tartalmazó vektor - a közelítés a szomszédos pontok közötti húr
% hosszával történik
T (1) = 0;
>> for i = 2 : n
T (i) = T (i - 1) + sqrt ((xms (i) - xms (i - 1))^2 + (yms (i) - yms (i - 1))^2);
end;

% az interpolációs függvények
>> xT = @(u) spline (T, xms, u);
>> yT = @(u) spline (T, yms, u);

% ábrázolás - barna színnel
>> ezplot(xT,yT,[0,T(n)])

%ábrázoljuk a valódi görbét is - piros színnel
>> ezplot(xa,ya,[0,3*pi])
```



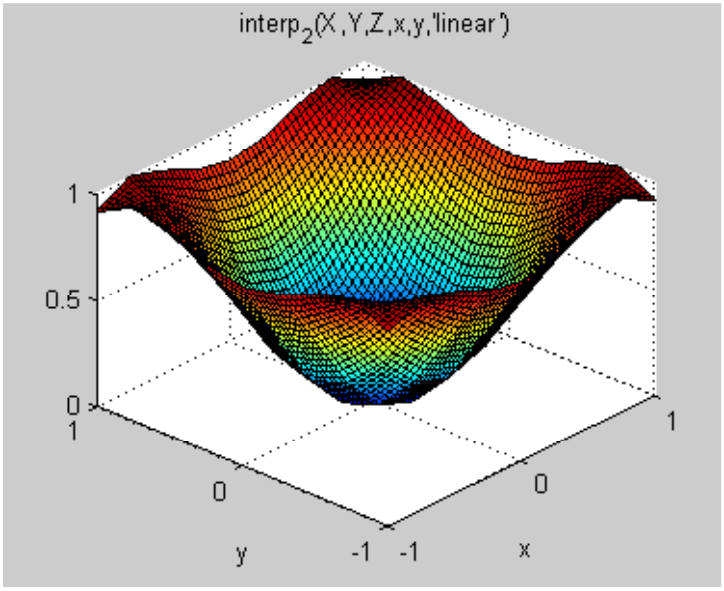
Látszik, hogy esetünkben a kis szögeknél (nagy sugár) az első, míg nagy szögeknél a második típusú paraméter ad jobb értéket! Mi erre a magyarázat?
(Az ívhossz, mint paraméter a legjobb, de kis szögeknél (távol lévő pontok) a húrral történő közelítés jelentősen eltér a valódi ívhossztól)

42. példa

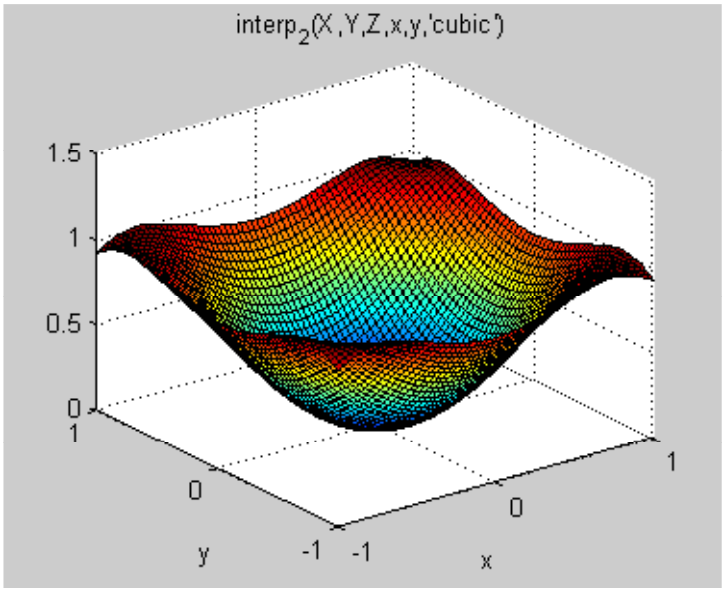
Közelítsük a $z(x,y) = \sin(x^2+y^2)$ függvényt az $x \times y = [-1,1] \times [-1, 1]$ tartományon bilineáris és köbös-spline interpolációval! Jellemezzük az egyes módszerek hibáját egy $\Delta x \times \Delta y = 0.25 \times 0.25$ négyzetrácson számított értékek alapján!

```
clear all;clc

% függvény - vektorosan, azaz ponttal definiálva
z = @(x, y) sin (x.^2 + y.^2);

% az alaphálózat létrehozása
```

```
[X, Y] = meshgrid (-1 : 0.25 : 1);

% a függvényértékek
Z = z (X, Y);

% a bilineáris interpolációs függvény
zBL = @(x, y) interp2 (X, Y, Z, x, y, ' linear');

ezsurf (zBL, [-1, 1, -1, 1])
```

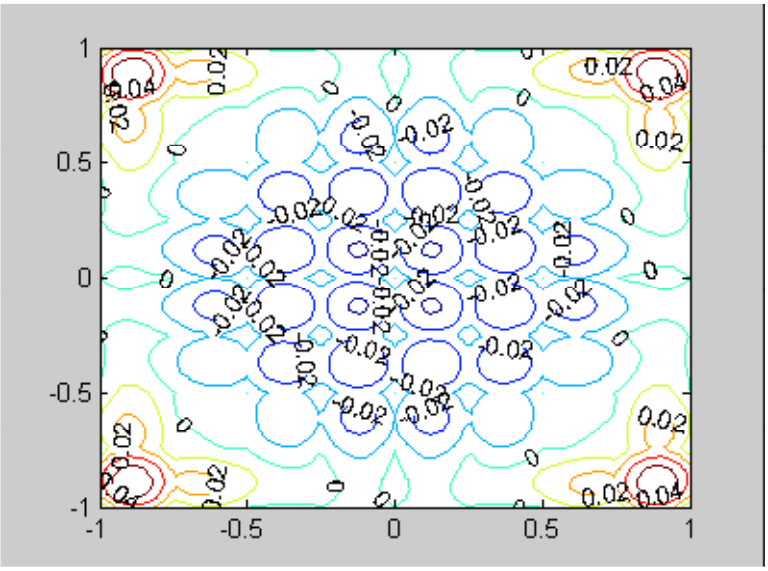



```
% köbös spline esetén
>> zCS = @(x, y) interp2 (X, Y, Z, x, y, ' cubic');
>> ezsurf (zCS, [-1, 1, -1, 1])
```

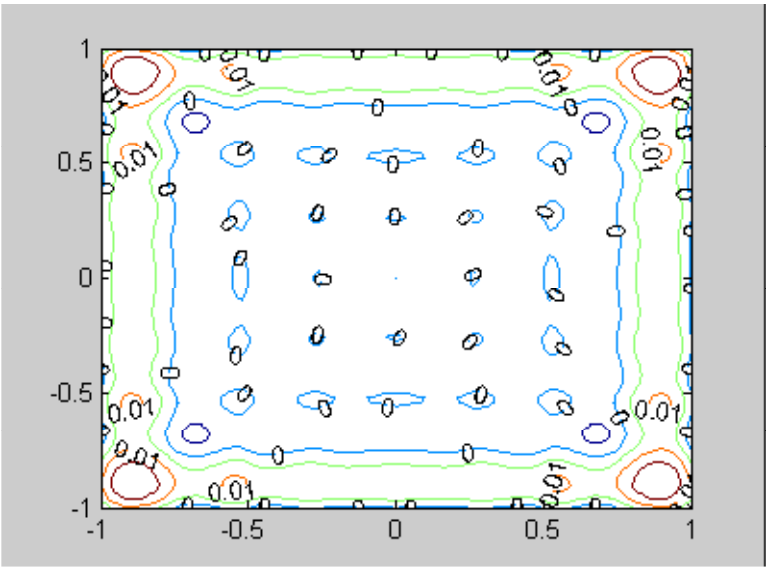



```
% a bilineáris interpoláció hibája
>> errBL=@(x,y)z(x,y)-zBL(x,y);

% a köbös spline esetén
```

```
>> errCS=@(x,y)z(x,y)-zCS(x,y);

% a hibák eloszlását szintvonalas ábrán adjuk meg. Ehhez új rácsot kell generálni.
%(miért nem jó a régi? - mert zérus hibát kapnánk- miért? - mert interpoláció!)

>> [Xerr, Yerr] = meshgrid (-1 : 0.02 : 1);

% a bilineáris eset
>> ZBL = errBL (Xerr, Yerr);
>> [C,h] = contour (Xerr, Yerr, ZBL);
% a clabel(C,h) helyett a ritkább felíratozást eredményező függvény alkalmazzuk
>> set (h, ' ShowText', ' on', ' TextStep', get (h, ' LevelStep')*2);
```




```
% A köbös spline esetén hasonlóan járunk el
errCS = @(x, y) z (x, y) - zCS (x, y);
ZCS = errCS (Xerr, Yerr);
[C, h] = contour (Xerr, Yerr, ZCS);
set (h, ' ShowText', ' on', ' TextStep', get (h, ' LevelStep')*2);
```



Látható, hogy a köbös spline lényegesen kisebb hibát ad!

43. példa

Alkalmazzuk az RBF módszert, az alábbi *Franke-féle g*(x,y) teszt függvény közelítésére, az $x\times y\in[0,\,1]^2$ tartományban, n =100 véletlenszerűen elhelyezkedő adatpont: (x_i,y_i) és $g_i=g(x_i,y_i)$ esetén.

$$g(x,\,y)=\frac{3}{4}\,e^{-1/4\left((9\,x-2)^2+(9\,y-2)^2\right)}+\frac{3}{4}\,e^{-(1/49)\,(9\,x+1)^2-(1/10)\,(9\,y+1)^2}+\frac{1}{2}\,e^{-1/4\left((9\,x-7)^2+(9\,y-3)^2\right)}-\frac{1}{5}\,e^{-(9\,x-4)^2-(9\,y-7)^2}$$

A közelítő RBF bázisfüggvény típusa legyen a vékonylemez RBF spline:

$$\varphi(r)=r^2\log(r)$$

illetve multikvadratikus RBF spline

$$\varphi(r)=\sqrt{1+(cr)^2}$$

ahol c állandó (paraméter) és

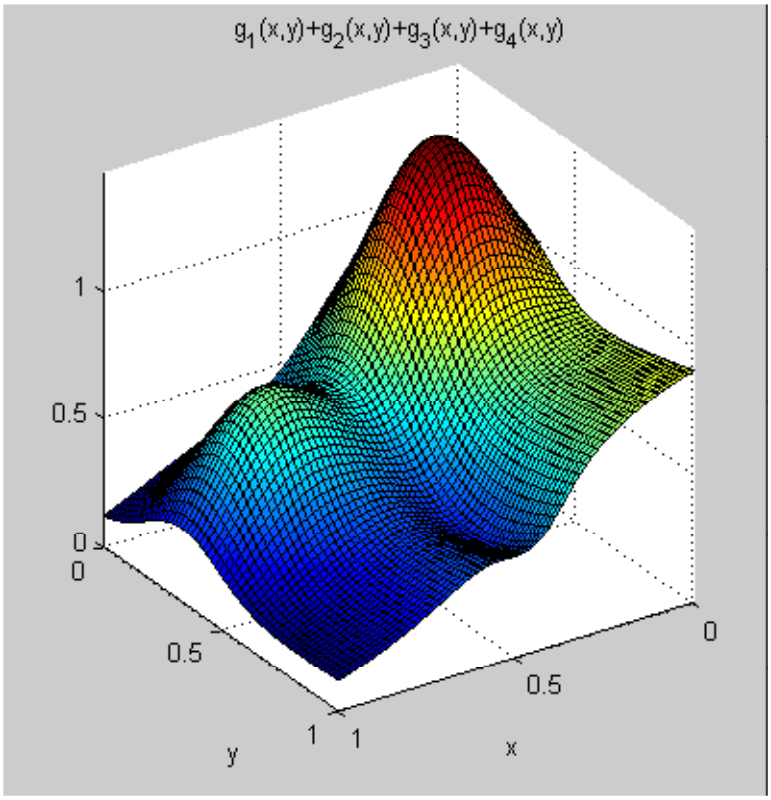
$$r_i(x,\,y)=\left\|\begin{pmatrix}x\\y\end{pmatrix}-\begin{pmatrix}x_i\\y_i\end{pmatrix}\right\|,\qquad i=1,\,2,\,\ldots,\,n$$

így a közelítés alakja

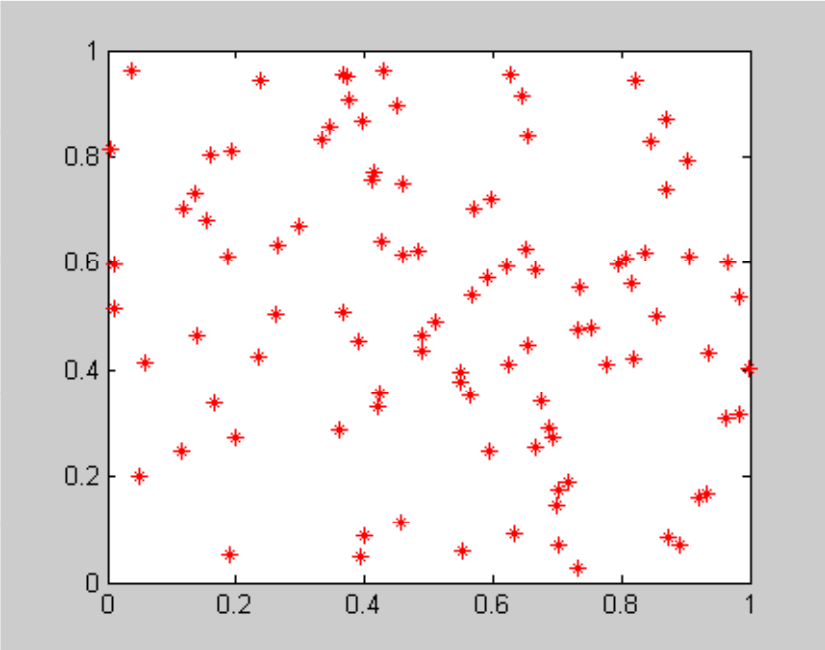
$$g(x,\,y)\approx\sum_{i=1}^{n}k_i\,\varphi\left(\left\|\begin{pmatrix}x\\y\end{pmatrix}-\begin{pmatrix}x_i\\y_i\end{pmatrix}\right\|\right)$$

Tehát a feladat a k_i együtthatók meghatározása!

```
% ábrázoljuk a tesztfüggvényt:

>> g1 = @(x, y) 3/4*exp (-1/4*((9*x - 2)^2 + (9*y - 2)^2));
>> g2 = @(x, y) 3/4*exp (-1/49*((9*x + 1)^2 - 1/10*(9*y + 1)^2));
>> g3 = @(x, y) 1/2*exp (-1/4*((9*x - 7)^2 + (9*y - 3)^2));
>> g4 = @(x, y) - 1/5*exp (-(9*x - 4)^2 - 1/10*(9*y - 7)^2);
>> z = @(x, y) g1 (x, y) + g2 (x, y) + g3 (x, y) + g4 (x, y);
>> ezsurf (z, [0, 1])
```




```
>> n=100;
>> xr = rand (n, 1); yr = rand (n, 1);
>> plot (xr, yr, ' r*');
```



Az egyes közelítő függvények:

vékonylemez RBF

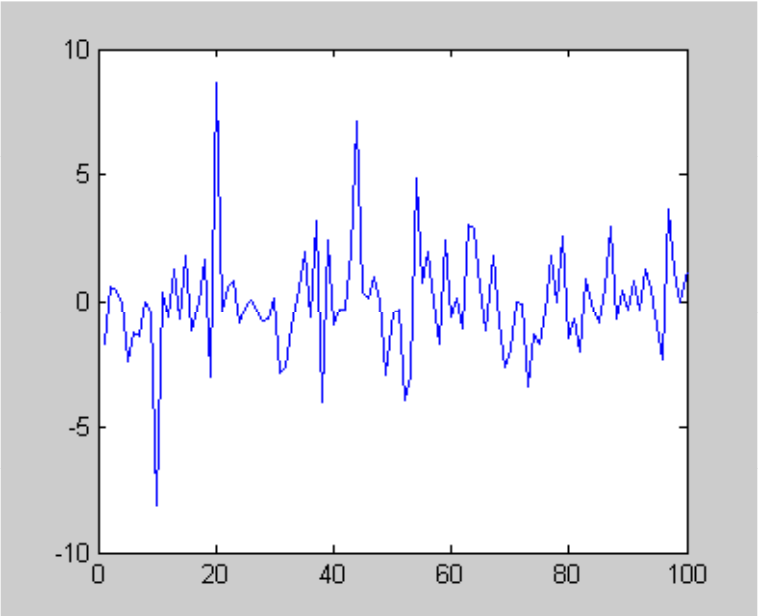
```
function y = vlemez (u, ui)
if norm (u - ui)~= 0
y = norm (u - ui)^2*log (norm (u - ui));
```

```
else
y = 0;
end
```

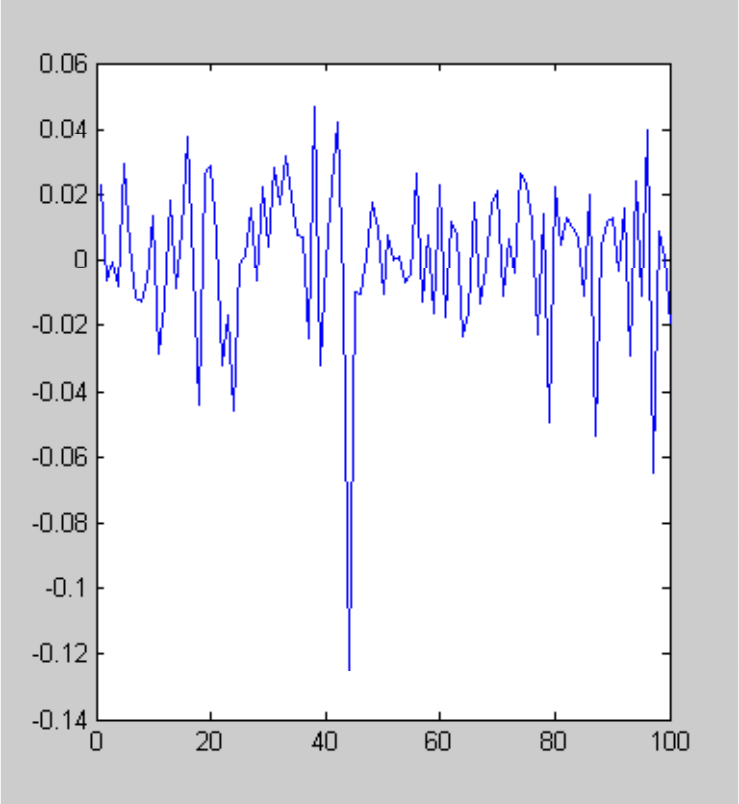
multikvadratikus RBF

```
function y = multiq (u, ui, c)
if norm (u - ui)~= 0
y = sqrt(1+(c* (u - ui))^2);
else
y = 0;
end
```

```
% az adatpontok előállítása
>> for i = 1 : n
YR (i) = z (xr (i), yr (i));
end;

% az x-y koordináták összefoglalása egy XYR mátrixba
>> XYR (1 : 100, 1) = xr;
>> XYR (1 : 100, 2) = yr;

%az együtthatók (k_i) meghatározása a vékonylemez RBF esetén
%az egyenletrendszer mátrixa
>> for i=1:n
for j=1:n
ZRVL(i,j)=vlemez([XYR(i,1),XYR(i,2)],[XYR(j,1),XYR(j,2)]);
end;
end;

% az együtthatók
>> kVL=linsolve(ZRVL,YR');

%ábrázoljuk őket
>> figure(3)
>> plot(kVL)
```




```
%az együtthatók (k_i) meghatározása a multikvadratikus RBF esetén, c = 13 választással
%megjegyezzük, hogy a c értéke jelentősen befolyásoljhatja a közelítés minőségét!
%az egyenletrendszer mátrixa
>>c=13;
>> for i=1:n
for j=1:n
ZRMQ(i,j)=multiq([XYR(i,1),XYR(i,2)],[XYR(j,1),XYR(j,2)],c);
end;
end;

% az együtthatók
>> kMQ=linsolve(ZRMQ,YR');

%ábrázoljuk őket
>> figure(4)
>> plot(kMQ)
```

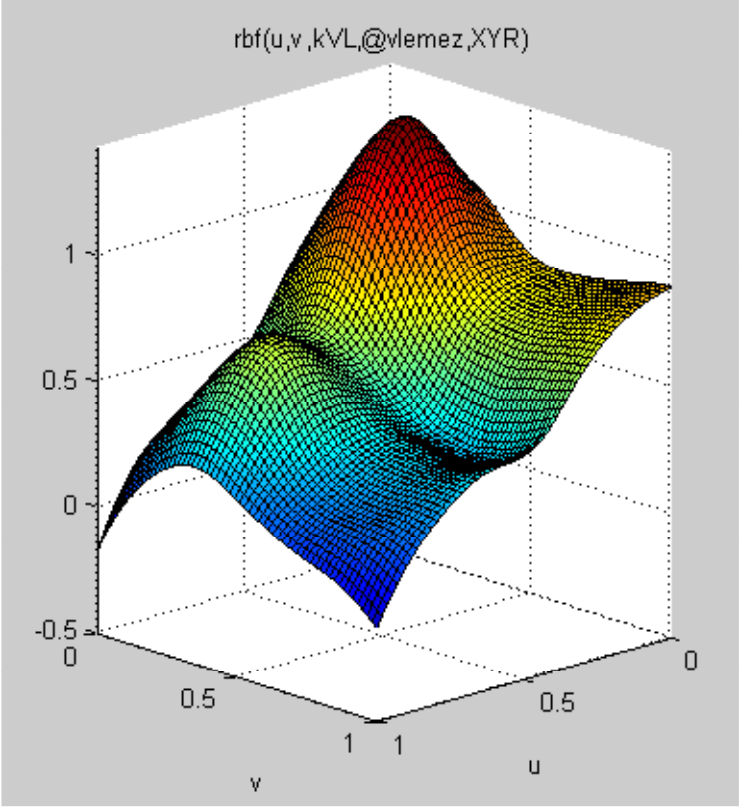


Vegyük észre, hogy az együtthatók most a (-1, 1) intervallumba esnek.
Definiáljuk az RBF közelítő függvényt általánosan, tetszőleges φ bázisfüggvény esetére:

```
function y =rbf (u, v,k, fi, XYR)
n=length(k);
y = 0;
for i = 1 : n
 y = y + k (i)*fi ([u, v], [XYR (i, 1), XYR (i, 2)]);
end;
```

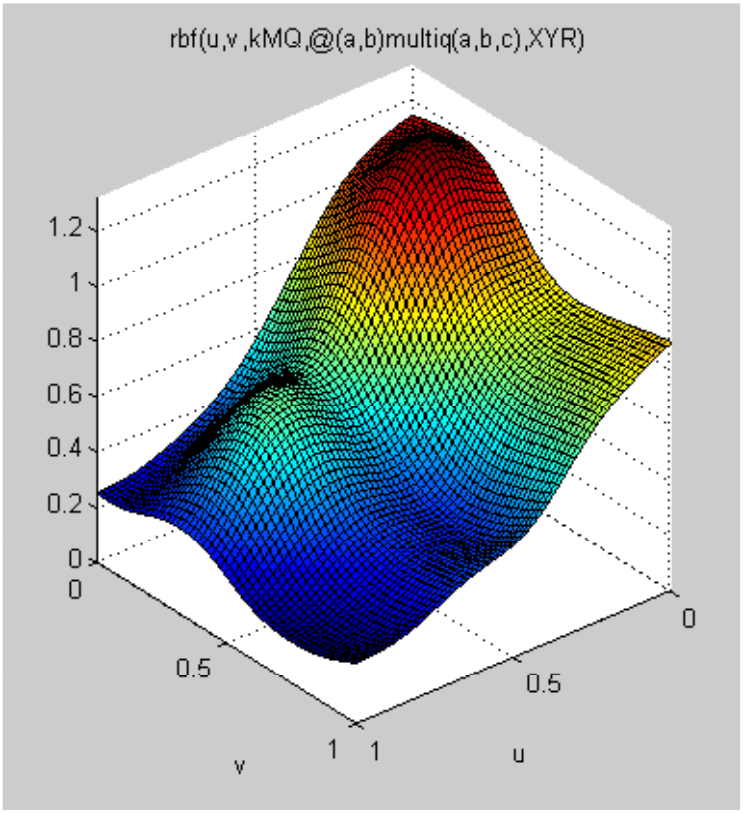
Ezzel a vékonylemez RBF típusú közelítés függvénye

```
>> rbfVL = @(u, v) rbf (u, v, kVL, @vlemez, XYR);

>> ezsurf (rbfVL, [0, 1, 0, 1])
```



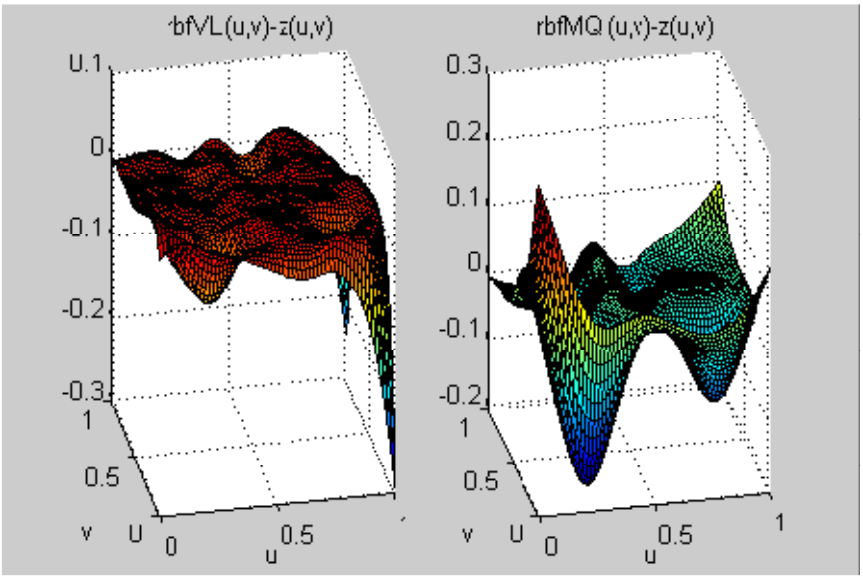
A multikvadratikus esetben figyelemmel kell lennünk arra, hogy az *rbf* függvény definicójában a *fi* függvénynek csak 2 változója van, amíg ebben a neki megfelelő *multiq* függvénynek 3, hiszen a 3. a *c* paraméter értéke. Ilyen esetekben a bemenetként szereplő függvényt (most *multiq*) átdefiniáljuk a szükséges változósra, amikor azt az hívó függvény (most *rbf*) paraméter listájába beírjuk. Az ilyen problémát korábban globális változóval oldottuk meg!

```
% c értékkel rendelkezik!
>> rbfMQ = @(u, v) rbf (u, v, kMQ, @(a, b) multiq (a, b, c), XYR);

>> ezsurf (rbfMQ, [0, 1, 0, 1])
```



A hibafüggvények:

```
>> errVL = @(u, v) rbfVL (u, v) - z (u, v);
>> errMQ = @(u, v) rbfMQ (u, v) - z (u, v)

>> subplot (1, 2, 1)
>> ezsurf (errVL, [0, 1, 0, 1])
>> subplot (1, 2, 2)
>> ezsurf (errMQ, [0, 1, 0, 1])
```



Látható, hogy a vékonylemez RBF a tartomány sarkait kivéve egyenletesebb és kisebb hibát ad, mint a multikvadratikus RBF. Persze ha megnézzük az alappontok eloszlását látjuk, hogy éppen ez a két sarok "üres", azaz alulreprezentált.

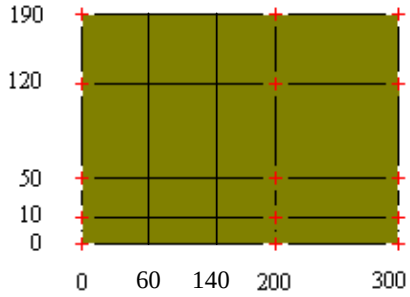
Megjegyzés: Mivel általában a beépített álvéletlenszám generátorok nem kielégítőek, a véletlen, egyenletes eloszlásúnak szánt pontok generálására, a valóságos egyenletes eloszlás jobb közelítésére, az ún. *Halton* -pontokat használják, lásd később a *Monte-Carlo* módszernél.

44. példa

Tekintsük a következő tereprendezési feladatot: Mennyi földet kell, hozni vagy elvinni, ha sík terepet akarunk kialakítani z = 135 m magasan. A szabályos elrendezésű pontokban mért magasságértékek egy digitális terepmodell esetén,

$$z = \begin{pmatrix} 135 & 131.7 & 136.4 & 139.8 & 119.8 \\ 134.8 & 133 & 139.7 & 135.5 & 120 \\ 124.4 & 130 & 140 & 139.2 & 122.3 \\ 131.1 & 133.8 & 138.12 & 137.7 & 121.1 \\ 133.2 & 137.1 & 143 & 135 & 123.3 \end{pmatrix}$$

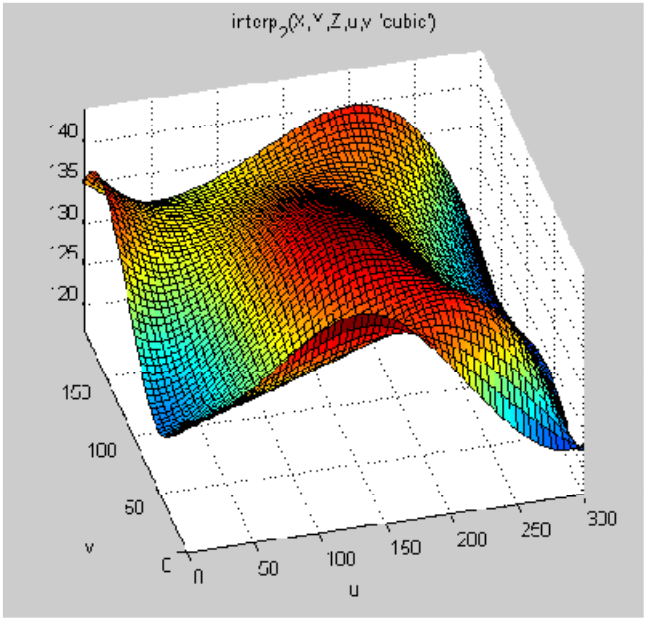
A mérési pontok koordinátái



A digitális terepmodellt köbös spline-interpolációval állítjuk elő.

```
% a koordináták
>> x = [0 60 140 200 300];
>> y = [190 120 50 10 0]; % itt a sorrendre ügyeljünk!

% a rács koordinátái
>> [X,Y] = meshgrid (x, y);

% a rácspontokban mért értékek


>> Z = [135, 131.7, 136.4, 139.8, 119.8;
        134.8, 133, 139.7, 135.5, 120;
        124.4, 130, 140, 139.2, 122.3;
        131.1, 133.8, 138.12, 137.7, 121.1;
        133.2, 137.1, 143, 135, 123.3]

Z = 135.0000 131.7000 136.4000 139.8000 119.8000
134.8000 133.0000 139.7000 135.5000 120.0000
124.4000 130.0000 140.0000 139.2000 122.3000
131.1000 133.8000 138.1200 137.7000 121.1000
133.2000 137.1000 143.0000 135.0000 123.3000

% köbös spline interpolációt alkalmazva
>> F = @(u, v) interp2 (X, Y, Z, u, v, ' cubic');
```

```
% ellenőrzés
>> F (0, 0)
ans = 133.2000

>> F(300,190)
ans =
  119.8000

% terep geometriája
>> ezsurf (F, [0, 300, 0, 190])
```



A teljes rendelkezésre álló földtérfogat

$$V_t = \int_0^{190} \int_0^{300} F(x, y)\, dx\, dy$$

A szükséges földtérfogat

$$V_{sz} = z\ 300 \times 190$$

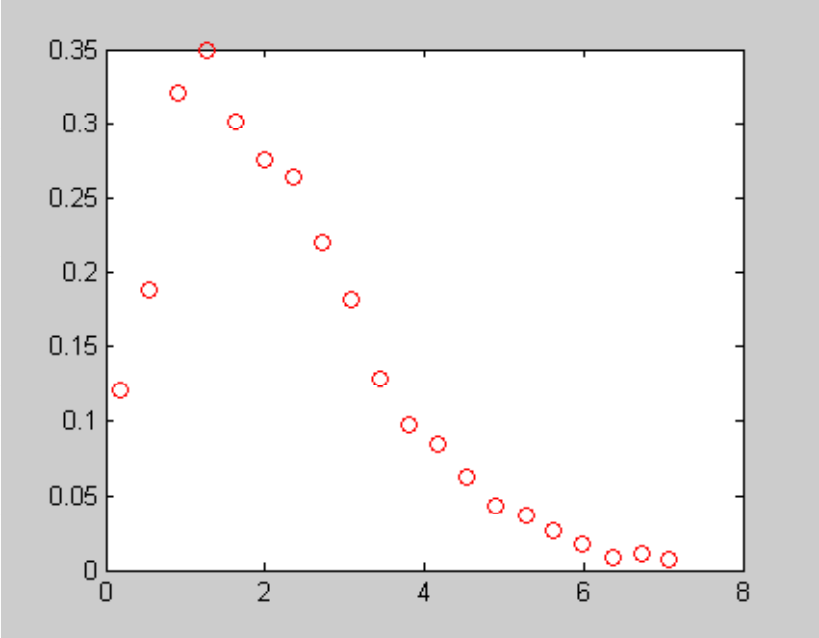
```
%A kettős integrál numerikus közelítése a Simpson szabály alapján
>> format long

>> Vt = dblquad(F, 0, 300, 0, 190)
Vt=
   7.614030545588052e+006

 % a szükséges térfogat
 >> Vsz=135*300*190
Vsz =
    7695000

% a pótlandó mennyiség
    >> Vsz-Vt
ans =
   8.096945441194810e+004
```

Polinomiális regresszió: egy és két változó esetén, nemlineáris regresszió

45. példa

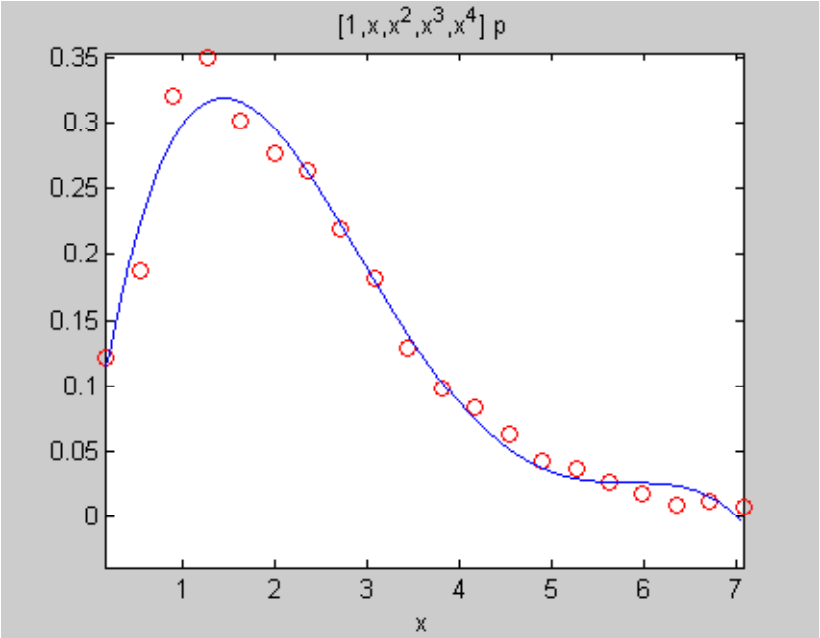
Adott a dataxy.txt adafile-ban egy (x_i,y_i) adatsor. Közelítsük az adatokat egy 4. fokú algebrai polinommal!

```
% Az adatfile beolvasása:
>> xym=load('c:\dataxy.txt')
xym =
  0.182000000000000   0.121000000000000
  0.545000000000000   0.188000000000000
  0.908000000000000   0.321000000000000
  1.270000000000000   0.350000000000000
  1.630000000000000   0.301000000000000
  2.000000000000000   0.276000000000000
  2.360000000000000   0.264000000000000
  2.720000000000000   0.220000000000000
  3.090000000000000   0.181000000000000
  3.450000000000000   0.129000000000000
  3.810000000000000   0.098000000000000
  4.180000000000000   0.084000000000000
  4.540000000000000   0.063000000000000
  4.900000000000000   0.043000000000000
  5.270000000000000   0.037000000000000
  5.630000000000000   0.027000000000000
  5.990000000000000   0.017000000000000
  6.360000000000000   0.009000000000000
  6.720000000000000   0.011000000000000
  7.080000000000000   0.007000000000000

  % a függő és független változók vektorának szétválasztása és a mérési pontok ábrázolása
  >> [n m]=size(xym)
n =
   20
m =
  2
>> xm=xym(1:n,1);
>> ym=xym(1:n,2);

>> plot(xm,ym,'ro')
```




```
% a polinom fokszáma
np=4;

% a polinom együthatóinak meghatározására szolgáló egyenletrendszer mátrixa
>> for i = 1 : n
for j = 1 : np + 1
X (i, j) = xm (i)^(j - 1);
end
end

% a mérések súlymátrixa legyen egységmátrix
>> W = eye (n);

% a túlhatározott egyenletrendszer megoldása - a polinom együtthatói
>> p = inv (X'*W*X)*X'*W*ym
p = 0.040839149916552
0.454783054886466
-0.234696306041646
0.040104244370439
-0.002281848304302

% az együtthatók meghatározása beépített függvénnyel
>> np=4;pp=polyfit(xm,ym,np);
>> pp'
ans =
  -0.002281848304290
   0.040104244370264
  -0.234696306040812
   0.454783054885077
   0.040839149917092

% itt sorrend fordított!
```

```
%definiáljuk a regressziós polinomot
>> yp4 = @(x)[1 x x^2 x^3 x^4]*p;

% beépített függvénnyel
ypp4=@(x)polyval(pp,x);

%rajzoljuk el
>> hold on
>> ezplot (yp4, [min (xm), max (xm)])
```



A regresszió lokális minősítése a rezidiumok értékei alapján történhet:

$$r_i = ym_i - yp4\,(xm_i), \quad i = 1, \ ..., \ n$$

Globális minősítése pedig ezek szórásával:

$$syx = \sqrt{\sum_{i=1}^{n} \frac{r_i^2}{n - (np + 1)}}$$

A becsült paraméterek megbízhatóságának jellemzése a szórásuk alapján történhet. Ezeket a becsült paramétervektor kovariancia mátrixának (C) főátlóbeli elemeinek négyzetgyöke adja:

$$C = syx^2 \left(X^T X\right)^{-1}$$

azaz

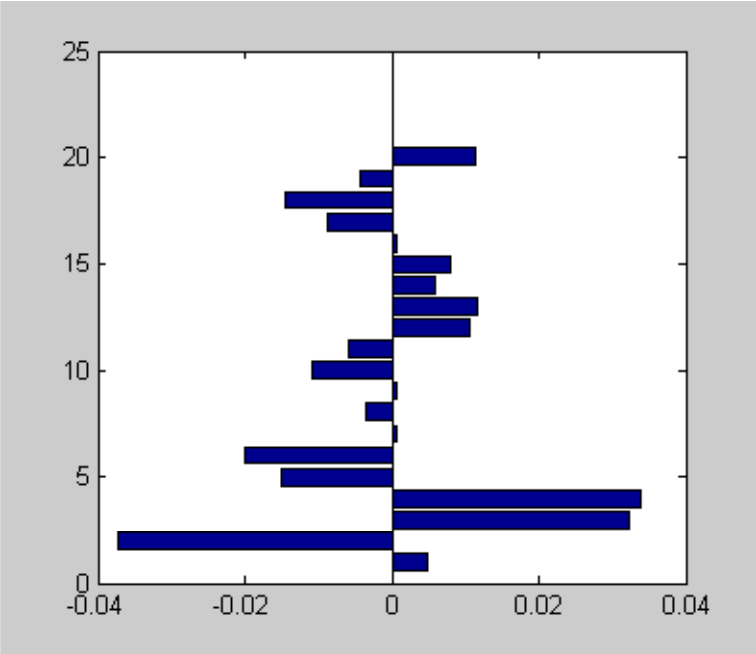
$$\sigma p_i = \sqrt{C_{i,i}} \qquad i = 1, \ .., \ np$$

ahol X mátrix a már alkalmazott, az együtthatók meghatározásra szolgáló egyenletrendszer mátrixa:

$$X_{i,j} = xm_i^{j-1}$$

```
% a rezidiumok és szórásuk számítása
s=0;
>> for i = 1 : n
```

```
r (i) = ym (i) - yp4 (xm (i));
s = s + (ym (i) - yp4 (xm (i)))^2;
end

% a rezidiumok ábrázolása
>> figure (2)
>> barh (r)
```




```
% a szórásuk
>> sxy = sqrt (s/(n - np + 1))
sxy =
    0.017456758078599

 % a kovariancia mátrix
 >> C=sxy^2*inv(X'*X)
    C =
   0.0004   -0.0007    0.0003   -0.0001    0.0000
  -0.0007    0.0015   -0.0008    0.0002   -0.0000
   0.0003   -0.0008    0.0005   -0.0001    0.0000
  -0.0001    0.0002   -0.0001    0.0000   -0.0000
   0.0000   -0.0000    0.0000   -0.0000    0.0000

% a paraméterek szórása
>> format long

>>szigmap=sqrt(diag(C))
szigmap =
  0.020223207745413
  0.038756551943921
  0.021680397065253
  0.004484114312649
  0.000306407316730
```

46. példa

A magyar Duna vízgyűjtőjének területei Ausztriában és Bajorországban találhatók. Ha itt jelentős mennyiségű csapadék esik, akkor a Dunán árhullám vonul végig. Feladatunk, a budapesti tetőző vízállás előrejelzése a következő két adat alapján:

x_1: az árhullámot kiváltó csapadék, amely 15 osztrák illetve bajor csapadékjelző állomás adatainak középértéke, mm.

x_2: a Duna vízállása Budapestnél az árhullámot kiváltó esőzés kezdetekor, cm.

y: a becsülendő érték, a Budapestnél, az árhullám tetőzéséhez tartozó vízszint, cm.

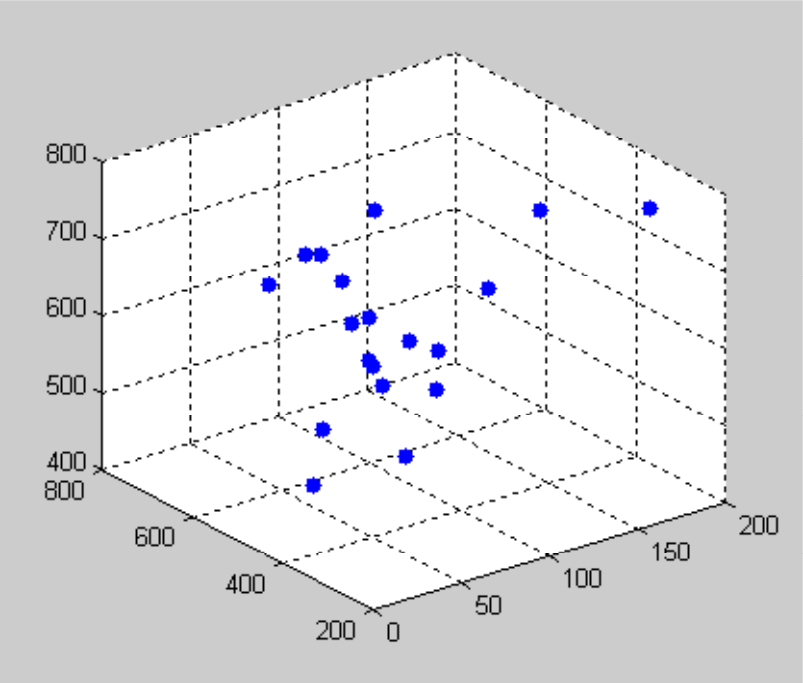
Az eddigi (1896 óta) adatokat az alábbi táblázat tartalmazza, ahol egy sor, egy adott időponthoz tartozó x_1, x_2 és y értékeket jelenti.

x_1 (mm)	x_2 (cm)	y (cm)
58	405	590
52	450	660
133	350	780
179	285	770
98	330	710
72	400	640
72	550	670
43	480	520
62	450	660
67	610	690
64	380	500
33	460	460
57	425	610
62	560	710
54	420	620
48	620	660
86	390	620
74	350	590
95	570	740

Alkalmazzunk másodfokú polinomiális regressziós felületet:

$$z(u, v) = p_1 + p_2\, u + p_3\, v + p_4\, u^2 + p_5\, u\, v + p_6\, v^2$$

```
% az adatok
x1=[58 52 133 179 98 72 72 43 62 67 64 33 57 62 54 48 86 74 95]';
x2=10*[40.5 45 35 28.5 33 40 55 48 45 61 38 46 42.5 56 42 62 39 35 57]';
y=10*[59 66 78 77 71 64 67 52 66 69 50 46 61 71 62 66 62 59 74]';

%ábrázoljuk a pontokat
scatter3(x1,x2,y,'filled')
```



% az együtthatók meghatározása

% az együttható mátrix

```
>> X=[ones(size(x1)),x1,x2, x1.*x1, x1.*x2, x2.*x2];
>> X
X =
     1    58   405   3364   23490   164025
     1    52   450   2704   23400   202500
     1   133   350  17689   46550   122500
     1   179   285  32041   51015    81225
     1    98   330   9604   32340   108900
     1    72   400   5184   28800   160000
     1    72   550   5184   39600   302500
     1    43   480   1849   20640   230400
     1    62   450   3844   27900   202500
     1    67   610   4489   40870   372100
     1    64   380   4096   24320   144400
     1    33   460   1089   15180   211600
     1    57   425   3249   24225   180625
     1    62   560   3844   34720   313600
     1    54   420   2916   22680   176400
     1    48   620   2304   29760   384400
     1    86   390   7396   33540   152100
     1    74   350   5476   25900   122500
     1    95   570   9025   54150   324900
```

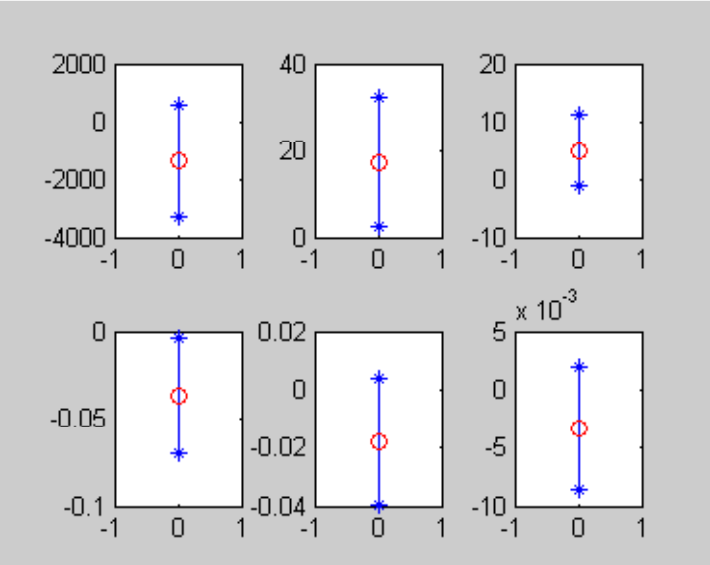
% az együtthatók meghatározása

```
>> [p pint h]=regress(y, X);
```

% az együtthatók értékei

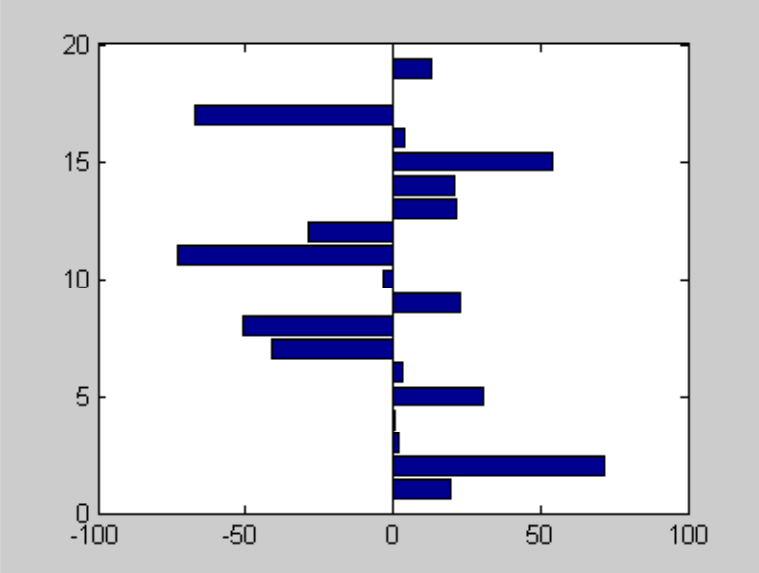
```
>>p
```

```
p =
  1.0e+003 *
  -1.339904885689538
   0.017188397349307
   0.004957238409773
  -0.000037295105311
  -0.000017919964216
  -0.000003338772032

  % Az együtthatók alsó és felső korlátai  95% -os megbízhatóság szint mellett:

>>pint
pint =
  1.0e+003 *

  -3.269183333997813   0.589373562618736
   0.002340256646102   0.032036538052511
  -0.001312086954542   0.011226563774088
  -0.000070244391302  -0.000004345819321
  -0.000039771928137   0.000003931999706
  -0.000008609100730   0.000001931556665

  % ábrázoljuk a paraméterek megbízhatósági intervallumait
  >> for i=1:6
subplot(2,3,i)
plot([0;0],[pint(i,1);pint(i,2)],'b*-');
hold on
plot(0,p(i),'ro');
end
```



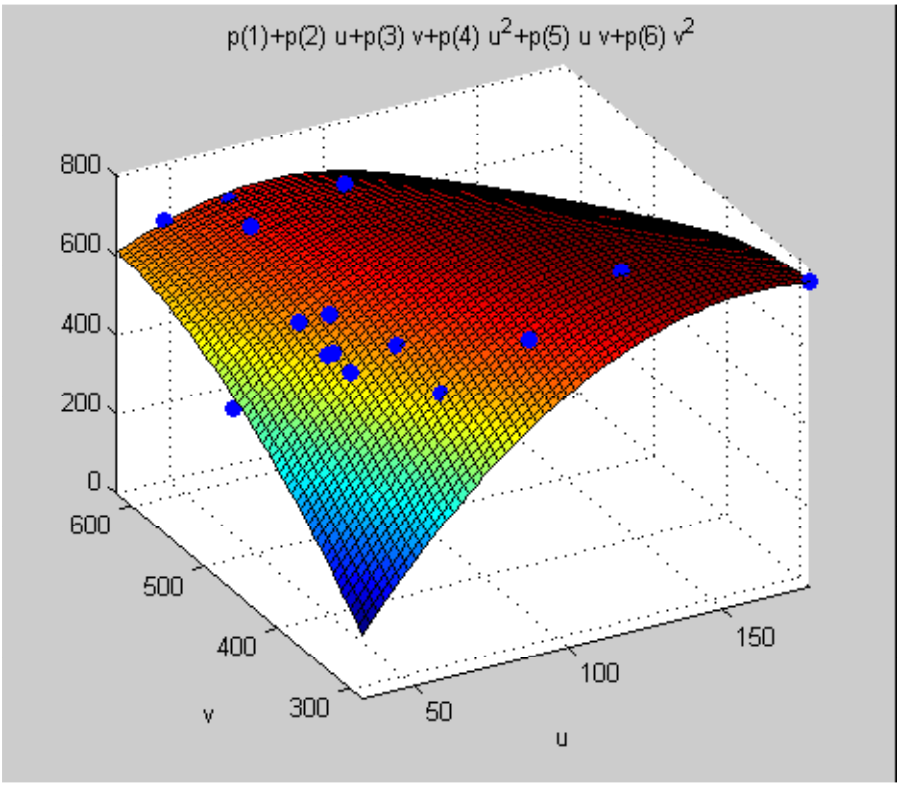

```
   % a rezidiumok az egyes pontokban
>> h
h =
  19.339059734133798
  71.625403041930895
```

```
 1.701620834112418
 0.720015424927169
 30.859378673286074
 3.081233114503789
-41.193900243456255
-50.590849686875231
 22.897688573742698
 -3.469428283934803
-73.210177861353372
-28.518306856102981
 21.688541209143636
 20.753192219589209
 53.827998688440061
 4.024025710486740
-66.922841401639971
 0.284682225068309
 13.102664884000887

 >>bar(h)
```



Ábrázoljuk a regressziós felületet:

```
>> z=@(u,v)p(1)+p(2)*u+p(3)*v+p(4)*u^2+p(5)*u*v+p(6)*v^2
>> hold on
>> ezsurf(z,[min(x1),max(x1),min(x2),max(x2)])
```



47. példa

Röntgen átvilágítással végzett vizsgálatokban a kép torzul, ha a sugarak a képernyőre nem merőlegesek. A torzulás mértékének meghatározása és a kép korrekciója céljából gömb alakú próbatestet világítanak meg és meghatározzák a képernyőn megjelenő ellipszis paramétereit. A z_1- z_2 koordináta-rendszer origóját a (p_1, p_2) pontba eltolva az ellipszis egyenlete:

$$(z_1 - p_1, z_2 - p_2)\begin{pmatrix} p_3 & p_4 \\ p_4 & p_5 \end{pmatrix}\begin{pmatrix} z_1 - p_1 \\ z_2 - p_2 \end{pmatrix} - 1 = 0$$

Az ellipszis mért pontjai $\{z_{1,i}, z_{2,i}\}$ az ellip.txt file-ban találhatók. Határozzuk meg a p_i, i=1,..5 paramétereket regresszióval!

A probléma most nemlineáris regresszió, hiszen a *paraméterek* szorzata, azaz a *paraméterek nemlineáris kifejezése* szerepel, továbbá *implicit*, mivel a rendelkezésre álló modell alakja nem

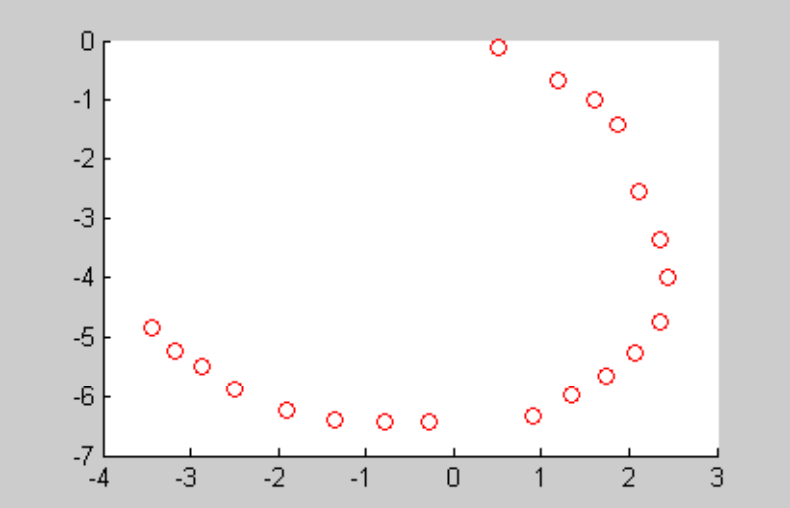
$$z_2 = f(z_1, p)$$

hanem

$$g(z_1, z_2, p) - 1 = 0$$

```
%adatok beolvasása
>> z1z2 = load (' c : \ellip.txt');
>> size (z1z2)
ans = 20 2

% a mérések száma
n=20;

% a pontok megjelenítése
>> z1 = z1z2 (1 : n, 1);
```

```
>> z2 = z1z2 (1 : n, 2);
>> scatter (z1, z2, ' ro')
```



1) Oldjuk meg a feladatot először az implicit kifejezés közvetlen minimalizálásával, azaz, minimalizáljuk a

$$G(p) = \sum_{i=1}^{n} \left(g\left(z_{1,i}, z_{2,i}, p\right) - 1\right)^2$$

kifejezést, vagyis a paraméterbecslést az explicit kifejezésre vonatkozó legkisebb négyzetek értelmében oldjuk meg.

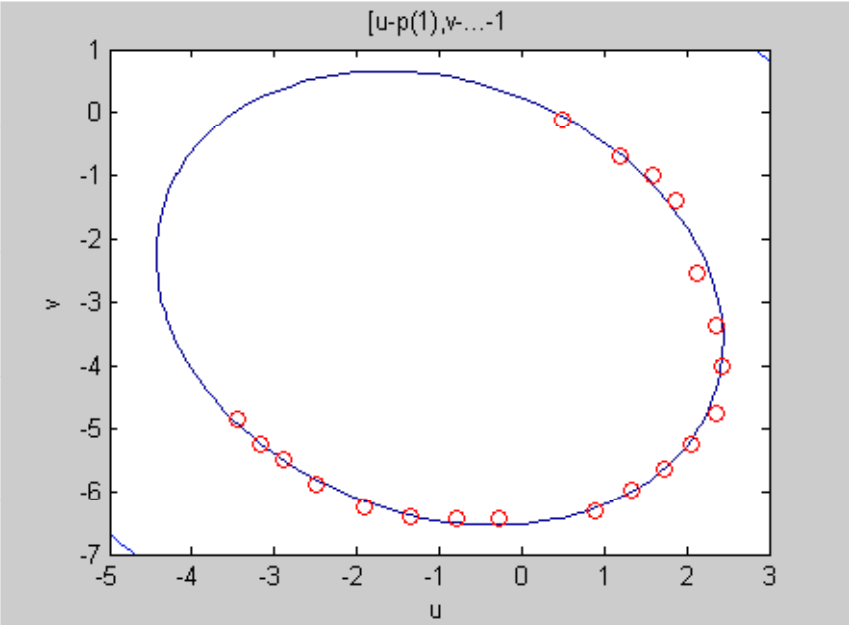
Definiáljuk a célfüggvényt:

```
function y = ellip (p)
global Z1 Z2
n = length (Z1);
s = 0;
for i = 1 : n
s0 = [Z1 (i) - p (1), Z2 (i) - p (2)]*([p (3), p (4); p (4), p (5)]*[Z1 (i) - p (1), Z2 (i) - p (2)]') - 1;
s = s + s0^2;
end
y = s;
```

Mivel nem ismerjük a paramétereknek még közelítő értékeit sem, *gloális minimalizálás*t alkalmazunk a *genetikus algoritmus* felhasználásval.

```
% a globális változókat definiáljuk és értéket adunk nekik

global Z1 Z2

>> Z1=z1;
>>Z2=z2;

% az opcionális értékek átállítása: Generations: 100 →10000,  PopulationSize: 20→200,
%UseParallel: 'never'→'always' ez utóbbi nyilván csak több prcesszoros vagy magos gépen,
% a laborban 2 magos van.
```

```
>> options = gaoptimset (' Generations', 10000, ' PopulationSize', 200, 'UseParallel','always');

psolG = ga (@ellip, 5, options)'
Optimization terminated: average change in the fitness value less than options.TolFun.

psolG =

  -1.452469770838021
  -1.171178924694129
   0.087263012437653
   0.015170352531300
   0.031834628462305

% Az általános módszer, hogy a globális minimalizációval csak a globális optimum közelébe %férközünk, de nem erőltetjük a pontosabb eredményt, mert ez a módszer nagyon
% számításigényes!
% A hozzávetőleges eredményt, egy lokális módszer kezdeti értékeként megadva, amely módszer általában négyzetesen konvergál (ez mit jelent?) pontosítjuk.

>> p=fminunc(@ellip,psolG)
Warning: Gradient must be provided for trust-region method;
  using line-search method instead.
> In fminunc at 281
Optimization terminated: relative infinity-norm of gradient less than options.TolFun.

p =

  -0.998282546212933
  -2.939503356154787
   0.087635714846724
   0.016139985640131
   0.079958906267817

% az ábrázolás érdekében a fenti paraméterekkel definiáljuk az ellipszis anonymous függvényét
 % (implicit!)
>> Elli = @(u, v)[u - p (1), v - p (2)]*([p (3) p (4); p (4) p (5)]*[u - p (1); v - p (2)]) - 1;

%majd berajzoljuk az ellipszist a mérési pontokat tartalmazó ábrába
>>hold on
>> ezcontour (Elli, [-5, 3, -7, 2])
% az ábrát módosítottuk a Property Editorral! (Alternatív felrajzolás ezplot segítségével!)
```



2) Oldjuk meg a feladatot beépített függvénnyel. Miután ez csak explicit alakra jó, úgy teszünk,mintha ez egy 2 független változós függvény lenne, azaz

$$y = g(z_1, z_2, p)$$

ahol most az $\{y_i, z_{1,i}, z_{2,i}\}$ hármasokat ismerjük. Persze tudjuk, hogy $y_i = 1$ minden i – re!

Írjunk egy m – file – t erre a $g(z_1, z_2, p)$ függvényre, amely most a kimeneti vektort y – t állítja elő.

Normál (explicit) esetben erre is lennének mérési értékek. Most ezek rendre 1-egyel egyenlők!

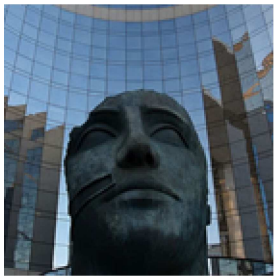
```
function y = funi (p, X)
[n m] = size (X);
for i = 1 : n
z (1) = X (i, 1);
z (2) = X (i, 2);
y (i) = [z (1) - p (1), z (2) - p (2)]*[p (3) p (4); p (4) p (5)]*[z (1) - p (1); z (2) - p (2)];
end
y = y';
```

```
% egyesítjük a "bemeneti" változók mérési vektorait
>> X = [z1, z2]

X = 0.500000000000000 - 0.120000000000000
1.200000000000000 - 0.680000000000000
1.600000000000000 - 1.000000000000000
1.860000000000000 - 1.400000000000000
2.120000000000000 - 2.540000000000000
2.360000000000000 - 3.360000000000000
2.060000000000000 - 5.250000000000000
1.740000000000000 - 5.640000000000000
1.340000000000000 - 5.970000000000000
0.900000000000000 - 6.320000000000000
-0.280000000000000 - 6.440000000000000
-0.780000000000000 - 6.440000000000000
-1.360000000000000 - 6.410000000000000
-1.900000000000000 - 6.250000000000000
```

```
-2.500000000000000 - 5.880000000000000
-2.880000000000000 - 5.500000000000000
-3.180000000000000 - 5.240000000000000
-3.440000000000000 - 4.860000000000000
2.440000000000000  - 4.000000000000000
2.360000000000000  - 4.750000000000000

% megadjuk a kimeneti vektor értékeit- normál esetben ezt is mérnénk
>> Y=ones([n 1])
Y =
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1

  % az iteráció kezdő vektora a paraméterekre legyen

>> p0=[1 1 1 1 1]';

% alkalmazzuk a beépített függvényt
>>  [p h,J,Cov]=nlinfit(X,Y,@funi,p0);

% a paraméterek
>> p
p =
  -0.998280736392196
  -2.939505108777681
   0.087635783862025
   0.016139948641816
   0.079958964594072

% összehasonlításként ezek értékei az előző módszerrel
  -0.998282546212933
  -2.939503356154787
   0.087635714846724
   0.016139985640131
```

```
   0.079958906267817

% a rezidiumok:  Δi  = 1 – g (z1, z2, p), i=1,...,n
>> h
h =
   0.031265198035486
   0.007950817477609
  -0.055085351183465
  -0.047515363392199
   0.094883045957904
   0.043085001370410
  -0.018422120459345
  -0.001522978004102
   0.015251419963825
  -0.022399367715049
   0.056174555129819
   0.040714896765768
  -0.015041280371455
  -0.043916755955089
  -0.031540283281025
   0.009942793397593
  -0.002316346634109
   0.031232949262774
  -0.008234754224959
  -0.054190917996997

% a g - függvény Jacobi-mátrixa a megoldás helyén  Ji,j = ∂ g (zi, p) / ∂ pj  |p=popt
>> J
J =
  Columns 1 through 3

  -0.353617988828641  -0.499257441856869   2.244813339798184
  -0.458232105215374  -0.432298651613198   4.832391501691203
  -0.518011600319744  -0.394036485754325   6.751007594077858
  -0.550670627235291  -0.338461722335939   8.169708054618928
  -0.559442897933925  -0.164547255028697   9.723608514712828
  -0.575039131485491  -0.041161125724963  11.277978169856544
  -0.461449387409508   0.270768694888440   9.353016101186748
  -0.392773298643533   0.343466325335900   7.498123226395746
  -0.312012193344661   0.409151196691280   5.467507133926451
  -0.223594609685263   0.479325614415708   3.603429432190049
  -0.012899817191214   0.536605529111211   0.515911959134142
   0.074736311808236   0.552745229298565   0.047641843513318
   0.175425839488859   0.566669725960357   0.130848509038255
   0.264908154748916   0.558513639155078   0.813116784129216
   0.358128131410414   0.518711427439842   2.255192645359605
   0.412465417295543   0.470208562443933   3.540907357562166
   0.456654450613739   0.438313567673140   4.759945288044213
   0.489959065566916   0.385937174646985   5.962044827846965
  -0.568402095900496   0.058604374267812  11.821701388942081
  -0.530170764945515   0.181125615390620  11.277978169856544
```

```
  Columns 4 through 5

   8.448791914699438   7.949668190868716
   9.934051217261432   5.105410724125751
  10.078770819132822   3.761720743354606
   8.800702838062241   2.370108267263276
   2.491595081257721   0.159612710880340
  -2.824200421093565   0.176807134844067
 -14.132170814064120   5.338338185385464
 -14.789311488994205   7.292616021087794
 -14.172182240559595   9.183835728449454
 -12.834145047584395  11.427674812057438
  -5.028586677165976  12.253391069224676
  -1.528102271364344  12.253391069224676
   2.510755842989538  12.044262005218888
   5.970325439734439  10.959306995751032
   8.831626609726614   8.646448535611242
   9.636280046577838   6.556080388158323
  10.038071149738476   5.292228497380260
   9.378608326668759   3.688260349559317
  -7.292463674433896   1.124627173145924
 -12.160191344582934   3.277853780164697

% a kovariancia mátrix - a megoldás helyén (p=p_opt), ebből számolható a paraméterek szórása:
% σp_i=√(Cov_i,i)

% most ez is adódik:
>> Cov
Cov =
  Columns 1 through 3

   0.011149265052132  -0.010395592975332   0.000371384384606
  -0.010395592975332   0.011040884321830  -0.000327961635510
   0.000371384384606  -0.000327961635510   0.000015514311829
  -0.000255026363541   0.000253175908239  -0.000007965468979
   0.000284321219949  -0.000321708762024   0.000007600767860

  Columns 4 through 5

  -0.000255026363541   0.000284321219949
   0.000253175908239  -0.000321708762024
  -0.000007965468979   0.000007600767860
   0.000007186031686  -0.000007182634837
  -0.000007182634837   0.000011822156805

% a szórások
>> sqrt(diag(Cov))
ans =
    0.1056    0.1051    0.0039    0.0027    0.0034
```



11. Gyakorlat

Véges differencia, trapéz és Simpson szabály, Gauss-Legendre kvadratúra, Richardson - féle extrapoláció, Monte-Carlo módszer.

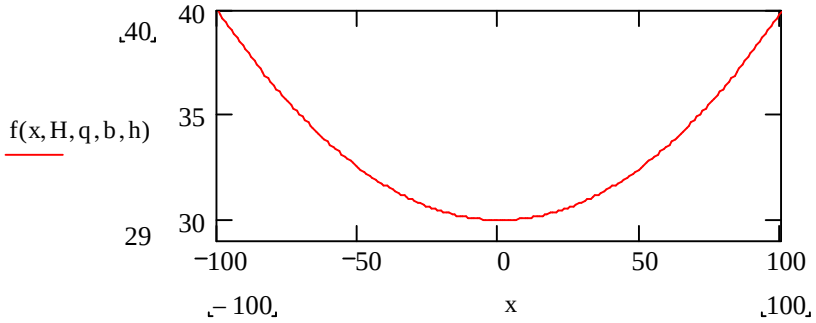
48. példa

Adott egy *q* [N/m] fajlagos súlyú szabadvezeték, amelyet két egymástól *b* távolságra lévő *h* magasságú oszlophoz rögzítünk úgy, hogy a feszítőerő vízszintes komponense *H*. A kábel alakját az alábbi koszinus - hiperbolikusz függvény írja le:

$$f(x,\ H,\ q,\ b,\ h) = \frac{H}{q}\left(\cosh\left(\frac{q}{H}\,x\right) - \cosh\left(\frac{q}{H}\,\frac{b}{2}\right)\right) + h$$

Határozzuk meg a kábel hosszát, ha

$H = 10^3$ N, $q = 2$ N/m, $b = 200$ m, $h = 40$ m.

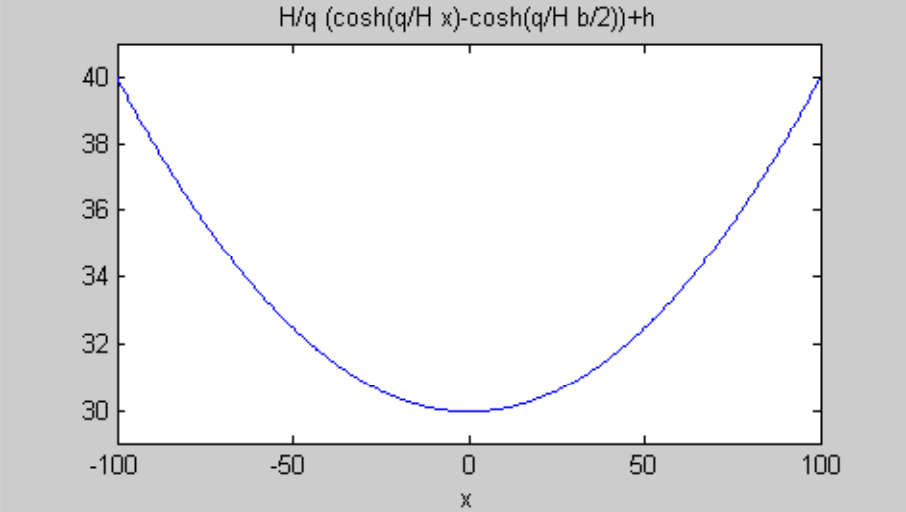


Oldjuk meg a feladatot numerikusan és ellenőrizzük a hibáját szimbolikus számítással!

Tehát legyen F(x) egy egyváltozós függvény:

$$F(x) = f(x,\ 1000,\ 2,\ 200,\ 40)$$

```
>> H = 1000; q = 2; b = 200; h = 40;

%definiáljuk a függvényt
>> F = @(x) H/q*(cosh (q/H*x) - cosh (q/H*b/2)) + h;

>> ezplot (F, [-100, 100])
```



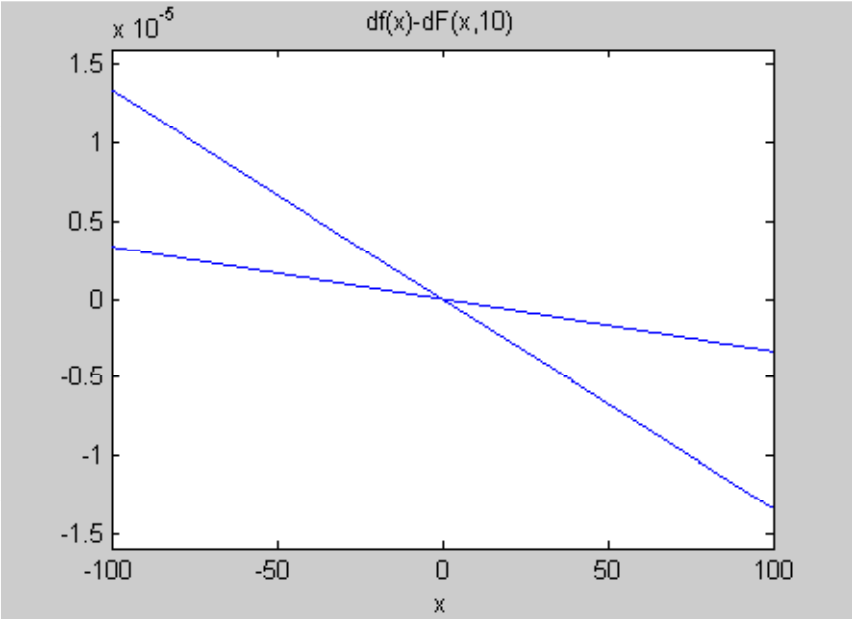
Egy *f*(x) függvény ívhosszát [a, b] intervallumon az alábbi módon határozhatjuk meg:

$$s(a, b) = \int_a^b \sqrt{1 + (f'(x))^2}\, dx$$

A deriváltfüggvényt a másodrendű hibájú , $O(h^2)$ differencia módszerrel közelítjük, azaz

$$dF(x, \Delta) = \frac{F(x + \Delta) - F(x - \Delta)}{2\,\Delta}$$

```
% a deriváltfüggvény numerikus közelítése az x helyen, mint Δ lépésköz függvénye
>> dF = @(x, D) (F (x + D) - F (x - D))/D/2;

%az analítikus derivált függvény előállítása szimbolikusan
>> syms x
>>>> diff (F(x), x)
ans =
   sinh(1/500*x)

%azaz
>> df = @(x) sinh (1/500*x)

% állítsuk elő a derivált numerikus közelítésének hibáját Δ=10 és Δ=5 esetén, mint a hely %függvényét

>> d10 = @(x) df (x) - dF (x, 10)
>> d5 = @(x) df (x) - dF (x, 5)

%ábrázoljuk a hibafüggvényeket
>> ezplot (d5, [-100, 100])
>> hold on
>> ezplot (d10, [-100, 100])
```



Az integrálást először a trapézszabály alapján végezzük,

```
function s = trap (f, a, b, n)
s = 0;
d = (b - a)/n;
for i = 1 : n - 1
s = s + f (a + i*d) ;
end
s = d/2*(2*s+f(a)+f(b));
```

Legyen a lépésköz Δ=10 m

```
>>D = 10;

>>% az integrandusz
>> dfI = @(x) sqrt (1 + dF (x, D)^2);

>> format long
>> LT = 2*trap (dfI, 0, 100, 100/D)
    LT =
    2.013428916602308 e + 002
```

Ha az integrálásnál az analítikus derivált függvényt használjuk

```
>> dfIa = @(x) sqrt (1 + df (x)^2);
>> LTa = 2*trap (dfIa, 0, 100, 100/D)
LTa =
  2.013427136964378 e + 002

% az ebből származó eltérés mm-nél kisebb!   Tehát elegendő a derivált közelítésénél a Δ=10.

% Numerikus integrálás beépített Simpson szabály alapján

 % a beépített függvény megkívánja, hogy az integrandusz függvény, dfI (x) tömbre is alkalmas legyen,
 % tehát vektoros definició, ponttal,
```

```
>> dfI=@(x)sqrt(1+dF(x,D).^2);

>> LS=2*quad(dfI,0,100)
LS =
   2.013361796274642e+002

% jelentős az eltérés: a trapézszabály hibája másodrendű, míg a Simpson módszeré negyedrendű.
```

Javítsuk a trapéz szabály eredményét *Richardson*- féle extrapolációval:

Legyen egy numerikus közelítés $\phi(\Delta)$ eredménye $O(\triangle^{p_1})$ hibával adott, akkor a

$$\phi\left(\frac{\Delta}{m}\right) + \frac{\phi\left(\frac{\Delta}{m}\right) - \phi(\Delta)}{m^{p_1} - 1}$$

kifejezés, a közelítést $O(\triangle^{p_2})$ hibával adja, feltéve, hogy

$$\phi(\Delta) = L + \Delta^{p_1} c_1 + \Delta^{p_2} c_2 + \ldots$$

alakban felírható, ahol L a pontos érték és c_i állandók.

A trapéz szabály esetén $p_1=2$ és $p_2 =3$. Legyen m =2.

```
% a Richardson féle  javítás
>>m=2;
>> LT = 2*(trap (dfI, 0, 100, 100/(D/m)) + ...
(trap (dfI, 0, 100, 100/(D/m)) - trap (dfI, 0, 100, 100/D))/(m^2 - 1))
LT =
 2.013361796386174 e + 002

 % a szimbolikus integrálás

 >>Ls = 2*int('sqrt(1+(sinh(1/500*x))^2)',x,0,100)
 Ls =
-500*(2+exp(-2/5)+exp(2/5))^(1/2)*(exp(-1/5)-exp(1/5))/(exp(-1/5)+exp(1/5))

% véges precíziójú numerikusra (lebegőpontos) áttérve:
>> Ls=double(Ls)
Ls =
   2.013360025410940e+002

 % Látható, hogy a Simpson szabály csak kicsit jobb, mint a trapézszabály
 % a Richardson  féle  javítással.
```

Az egyik legegyszerűbb, de egyben igen hatékony módszer a *Gauss - Legendre*- féle kvadratúra. Esetünkben először a a határokat a

$$-\frac{b}{2} \le x \le \frac{b}{2}$$

intervallumból a $[-1 \le \eta \le 1]$ intervallumba kell lineárisan transzformálni, azaz

$$x = \frac{b}{2}\eta \quad \rightarrow \quad dx = \frac{b}{2}d\eta$$

tehát az integrálandó függvény

$$f(\eta) = \sqrt{1 + \left(\sinh\left(\frac{1}{500}\frac{b}{2}\eta\right)\right)^2}\ \frac{b}{2}$$

```
>> b = 200;
% a derivált függvény
>> df = @(eta) sinh (1/500*b/2*eta);

% az integrandusz
>> dfIa = @(eta) sqrt (1 + df (eta)^2)*b/2;

% a kétpontos  Gauss-Legendre kvadratúra
>> LGL = dfIa (-1/sqrt (3)) + dfIa (1/sqrt (3))
LGL =
    2.013348154734078 e + 002

%amely pontosabb, mint a trapéz szabály Δ=10 m lépésközzel és analítikus deriválttal.
```

Foglaljuk össze a különböző megoldási módszerek eredményét az alábbi táblázatban. Mindegyik esetben a numerikus differenciálás és integrálás esetén a lépésköz egységesen Δ=10 m.

Deriválás	Integrálás	Eredmény
$O(h^2)$	trapéz	201.342891
analítikus	trapéz	201.342713
$O(h^2)$	Simpson	201.336179
$O(h^2)$	trapéz + Richardson	201.336002
analítikus	analítikus	201.336002
analítikus	Gauss – Legendre 2 belső pont	201.334815

Megjegyzések:

(1) Ha egy feladat szimbolikusan elvégezhető, végezzük el szimbolikusan, mert csak egyszer kell és esetenként mérnökileg értelmezhető összefüggést kapunk!
(2) Csak olyan pontosságú módszert alkalmazzunk, amely a feladat fizikai szempontjából értelmes, pl. ne akarjuk meghatározni egy szoba hőmérsékletét 0.001 fok pontossággal bármennyire is csábító a lehetőség.

49. példa

Írjunk rekurzív függvényt a trapéz - szabály adaptív alkalmazására! Egy sorozatot állíthatunk elő, csökkenő lépésközökkel, például

$$s_0 = Tr(h_0),\ s_1 = Tr(h_1),\ \ldots,\ s_k = Tr(h_k),\ \ldots.$$

ahol

$$h_0 > h_1 > \ldots > h_k > \ldots$$

Legyen az előírt hibakorlát ϵ, akkor a rekurzió báziskritériuma,

$$|s_k - s_{k+1}| \leq \epsilon$$

Alkalmazzunk lépésköz csökkentésére felezést, azaz

$$h_{k+1} = \frac{h_k}{2}$$

```
function s = tra (f, a, b, h0, eps)
h1 = h0;
h2 = h0/2;
n1 = (b - a)/h1;
n2 = (b - a)/h2;
I1 = tr (f, a, b, n1);
I2 = tr (f, a, b, n2);
if abs (I2 - I1) < eps
I = I2;
else
I = tra (f, a, b, h2, eps);
end
s = I;
```

Teszteljük a függvényt az alábbi integrállal:

$$G = \int_0^{30} 200 \left(\frac{z}{5+z}\right) \exp\left(-\frac{z}{15}\right) dz$$

```
% a gépi pontosságú exakt megoldás:

>> g = @(z) 200*z/(5 + z)*exp (-z/15);
>> Gexact = simplify (int (g (z), 0, 30))
Gexact =
 -1000*(-3*exp (2) + exp (7/3)*Ei (1, 1/3) + 3 - exp (7/3)*Ei (1, 7/3))*exp (-2)

% numerikus alakban
>> GexactN=double(Gexact)
GexactN =
   1.480568480085906e+003
```

Megjegyzés:

Az Ei (x) az exponenciális integrál

$$\text{Ei}(x) = \gamma + \ln(x) + \sum_{n=1}^{\infty} \frac{x^n}{n\, n!}$$

és γ az Euler - állandó

$$\gamma = \lim_{n \to \infty} \sum_{k=1}^{n} \frac{1}{k} - \ln(n) = 0.57721566 \ldots$$

Mielőtt alkalmaznánk az adaptív lépésközű rekurzívan megírt trapéz-módszerünket, próbálkozzunk beépített függvénnyekkel:

```
% Beépített Simpson- szabály

>> LS = quad (g, 0, 30, 10^(-3))
LS =
```

```
   1.480215355980118 e + 003

% csökkentsük a lépésközt:
>> LS = quad (g, 0, 30, 10^(-4))
LS =
   1.480518211371748 e + 003

>> LS = quad (g, 0, 30, 10^(-5))
LS =
   1.480547455903976 e + 003

>> LS = quad (g, 0, 30, 10^(-6))
Warning : Maximum function count exceeded; singularity likely.
 > In quad at 106
LS = 1.520408578472959 e + 003

% hiába csökkentjük a lépésközt, az eredmény nem hogy javulna, de még rosszabbá válik!
% A kerekítési hiba dominánssá vált a csonkítási hibával szemben!
% Nem lehet a módszerrel jobb eredményt elérni, mint 5 értékes jegy pontosságot!

% A Gauss-Lobato kvadratura tovább bírja, de ez sem jobb!

>> LQ=quadl(g,0,30,10^(-6))
LQ =
   1.483689242814975e+003
>> LQ=quadl(g,0,30,10^(-7))
LQ =
   1.480677508618293e+003
>> LQ=quadl(g,0,30,10^(-8))
LQ =
   1.480584674448347e+003

>> LQ=quadl(g,0,30,10^(-9))
Warning: Maximum function count exceeded; singularity likely.
> In quadl at 104
LQ =
   1.551006154822670e+003
```

Most nézzük mire megyünk az adaptív trapéz módszerrel?

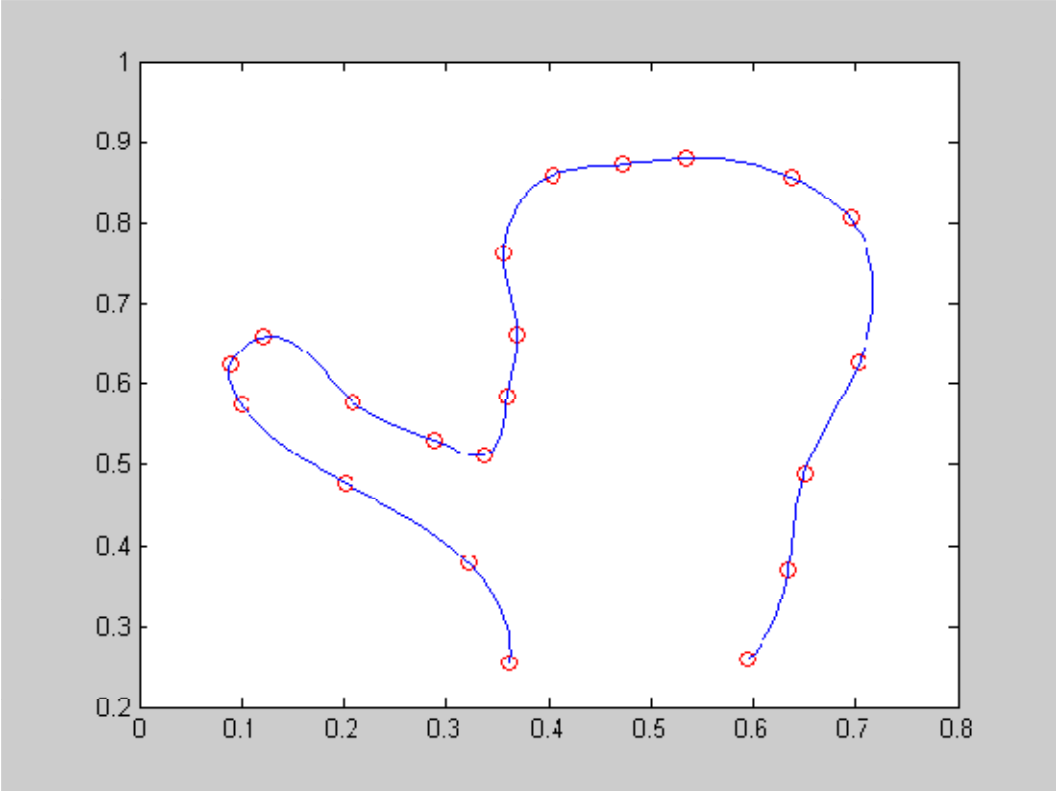
```
% Alkalmazzuk az adaptív trapézszabályt h0= 0.5 induló lépésközzel, különböző hibakorlátok esetén

>> LTA = tra (g, 0, 30, 0.5, 10^(-3))
LTA =
   1.480568269330506 e + 003

>> LTA = tra (g, 0, 30, 0.5, 10^(-4))
LTA =
   1.480568466913674 e + 003

>> LTA = tra (g, 0, 30, 0.5, 10^(-5))
LTA =
   1.480568476792829 e + 003
```

```
>> LTA = tra (g, 0, 30, 0.5, 10^(-6))
LTA =
   1.480568479880091 e + 003

% Gyakorlatilag 9 értékes jegyre pontos - persze lassabb, mivel rekurzív.
```

50. példa

Adottak egy egyújjas kesztyű konturjának pontjai (x_i,y_i) a glove.txt file-ban. Rajzoljuk fel a kesztyű konturját!

```
>> xy = load (' c : \glove.txt');
>> n = length (xy);

 >> X = xy (1 : n, 1);
 >> Y = xy (1 : n, 2);
 >> plot (X, Y, ' ro')

 % alkalmazzunk spline interpolációt, paraméterként az ívhosszat közelítve a húrral
 >> T (1) = 0;
 >> for i = 2 : n
T (i) = T (i - 1) + sqrt ((X (i) - X (i - 1))^2 + (Y (i) - Y (i - 1))^2);
  end

 % ezzel a paraméteres egyenletek
>> xT = @(u) spline (T, X, u);
>> yT = @(u) spline (T, Y, u);

% a kontur felrajzolása vektorosan, azaz pontpárokkal

>> tpT = linspace (0, T (length (T)), 200);
>> xpT = xT (tpT);
>>  ypT = yT (tpT);

>> hold on
>> plot (xpT, ypT, ' b - ');
```



Határozzuk meg a területét!

A terület, mint poligon területe (*előjeles!*)

Tudjuk, hogy egy háromszög területe

$$T_3 = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = \frac{1}{2}\left(-x_2\,y_1 + x_3\,y_1 + x_1\,y_2 - x_3\,y_2 - x_1\,y_3 + x_2\,y_3\right)$$

Ezt a következő módon is előállíthatjuk

$$T_3 = \frac{1}{2}\left((x_1,\, x_2,\, x_3)\begin{pmatrix} y_2 \\ y_3 \\ y_1 \end{pmatrix} - (x_2,\, x_3,\, x_1)\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}\right)$$

Általánosítva poligonra

$$T_n = \frac{1}{2}\left((x_1,\, x_2,\, x_3\, \ldots\, x_n)\begin{pmatrix} y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \\ y_1 \end{pmatrix} - (x_2,\, x_3,\, \ldots,\, x_n,\, x_1)\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{pmatrix}\right)$$

```
>>abs( (X'*Y ([2 : n, 1]) - X ([2 : n, 1])'*Y))/2
ans =
  0.2158
```

Egyszerű kvadraturával

```
% előállítunk egy h ×h rácsméretű rastert, amely lefedi az ábrát
>> h = 0.05;
```

```
>> [u v] = meshgrid (min (X) : h : max (X), min (Y) : h : max (Y));

% meghatározzuk, hogy ezen rácspontok közül, melyek vannak a polygon területén
%  (igaz(1) -hamis(0))

>> K = inpolygon (u, v, X, Y)

K = 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 0 0
0 0 0 0 0 1 1 1 1 1 1 0 0
0 0 0 0 1 1 1 1 1 1 1 1 0
0 0 0 1 1 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 1 1 1 0
0 1 1 1 0 0 1 1 1 1 1 1 0
0 1 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 1 1 1 1 1
0 0 0 0 0 0 0 1 1 1 1 0 0

% a K mátrix nemnulla elemeinek száma szorozva a rács területével adja  a közelítő értéket:
>> h^2*nnz (K)
ans =
   0.2175
```

Monte - Carlo módszer

Ez nagyon hasonló az előbbi módszerhez, de most nem alkalmazunk szabályos rácsot, hanem helyette egyenletes eloszlású véletlen elhelyezkedésű pontokkal fedjük le a befoglaló szabályos tartományt.

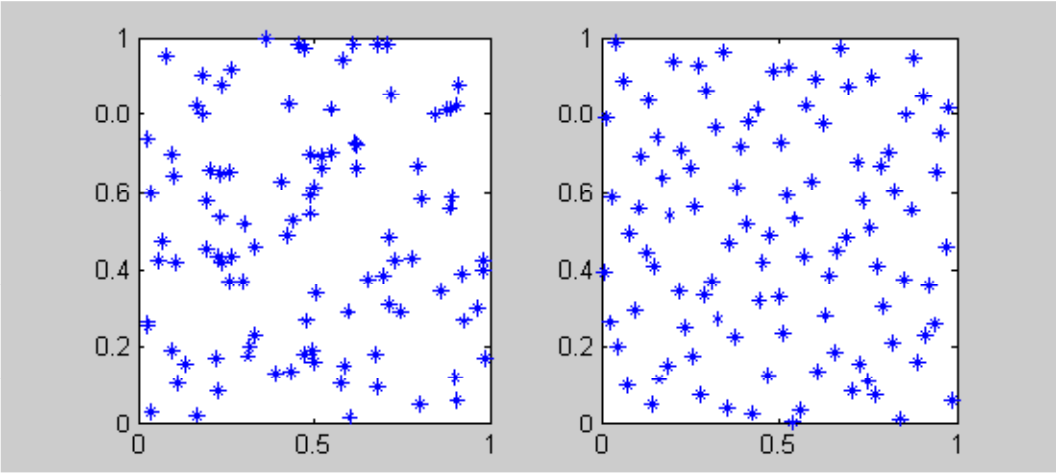
```
% egy másik ábrába csak a kontur vonalát rajzoljuk fel
figure(2)
plot (xpT, ypT, 'b-');

% a poligont(!) befoglaló négyszög tartomány
x1 = min (X); x2 = max (X); y1 = min (Y); y2 = max (Y);
XX = [x1 x1 x2 x2 x1];
YY = [y1 y2 y2 y1 y1];

>>hold on
>>plot (XX, YY, ' r*-')

% generáljunk 128 darab pontot

>> xyH=haltonseq(128,2);
```

A Halton módszer $\xi \in [0, 1]$ intervallumban állít elő egyenletes eloszlású véletlen számokat ezért szükség van az alábbi transzformációra

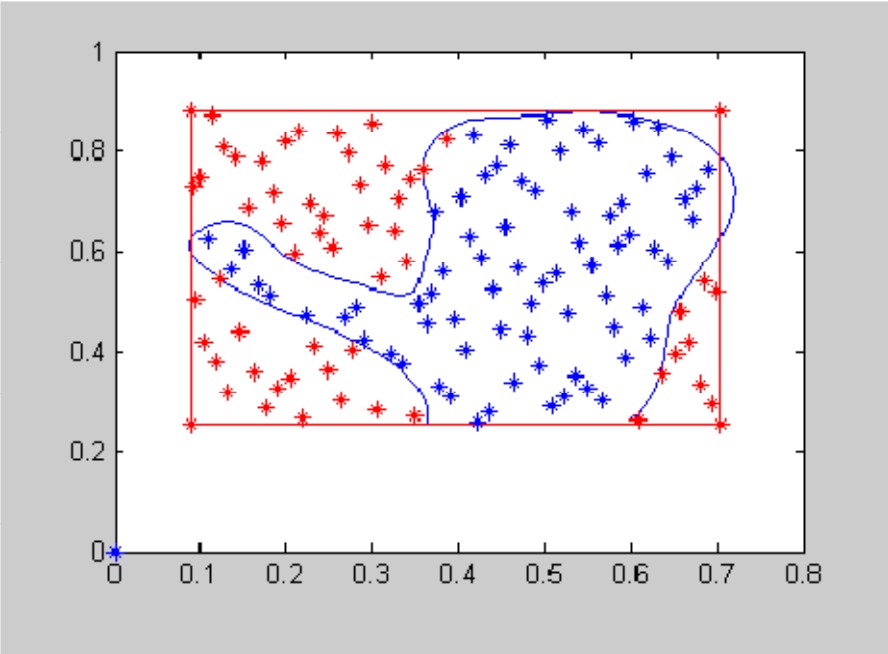
$$x = x_{min} (1 - \xi) + x_{max} \xi$$

```
% a Halton pontok és a beépített véletlenszám generátor eredménye n = 100
>> subplot(1,2,1)
>> n=100;
```

```
% Matlab beépített
>> xyr=rand(n,2);
>> plot(xyr(:,1),xyr(:,2),'b*')

% a Halton-pontok
>> xyH=haltonseq(100,2);
>> subplot(1,2,2)
>> plot(xyH(:,1),xyH(:,2),'b*')
```




```
>>xr = x1*(ones (128, 1) - xyH (1 : 128, 1)) + x2*xyH (1 : 128, 1);
>>yr = y1*(ones (128, 1) - xyH (1 : 128, 2)) + y2*xyH (1 : 128, 2);
>>plot (xr, yr, ' r*')

% A pontok tagsági értéke - ha a tartományba esik (1) különben  (0)
>> k=inpolygon(xr,yr,X,Y);

% rajzoljuk be ezeket a pontokat
>> xR=xr.*k;
>> yR=yr.*k;
>> plot(xR,yR,'b*')  % a tartományon kívüli pontokat az origóba rajzoltuk
```



$$\text{A} = \text{befoglaló négyzet területe} \times \frac{\text{kék pontok száma}}{\text{összes pontok száma}}$$

```
>> nnz (k)/128*(x2 - x1)*(y2 - y1)
ans =
   0.2189
```

Magyarázat

A terület közelítő értéke

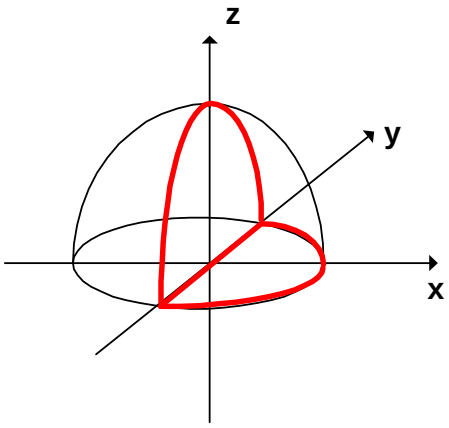
$$\text{A} = \int_{\text{A}} dA \approx \frac{n_A}{n_T} A_T$$

ahol A_T az A területet befoglaló terület $(A \subseteq A_T)$ és

n_T az egyenletes eloszlású véletlen (Halton) pontok száma az A_T tartományban.

51. példa

Határozzuk meg egy egységnyi sugarú negyedgömb térfogatát!



A negyedgömbben lévő P (x, y, z) pontok az alábbi feltételeket kell, hogy kielégítsék:

$$0 \le x \le 1$$

$$-1 \le y \le 1$$

$$0 \le z \le 1$$

és

$$x^2 + y^2 + z^2 \le 1$$

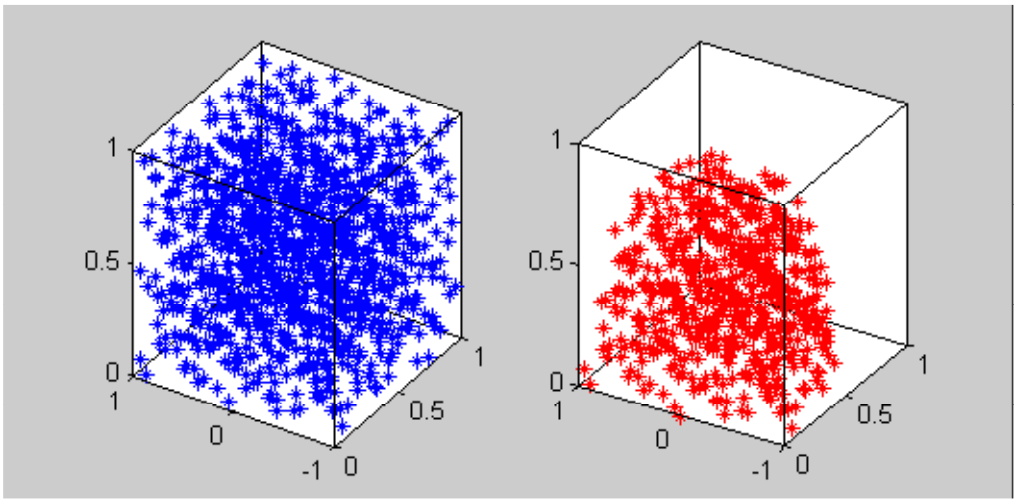
```
% a pontok generálása
x1 = 0; x2 = 1; y1 = -1; y2 = 1; z1 = 0; z2 = 1;
nT = 1024;
xyzH = haltonseq (nT, 3);

A tartomány transzformációja:

xr = x1*(ones (1024, 1) - xyzH (1 : 1024, 1)) + x2*xyzH (1 : 1024, 1);
yr = y1*(ones (1024, 1) - xyzH (1 : 1024, 2)) + y2*xyzH (1 : 1024, 2);
zr = z1*(ones (1024, 1) - xyzH (1 : 1024, 3)) + z2*xyzH (1 : 1024, 3);

subplot (1, 2, 1)
plot3 (xr, yr, zr, ' b*')
```

A tagsági függvény

```
function y = member (xr, yr, zr)
n = length (xr);
for i = 1 : n
if xr (i)^2 + yr (i)^2 + zr (i)^2 < 1
y (i) = 1;
else
y (i) = 0;
end
end
```

```
% az indikátor vektor: értéke 1 ha a pont a tartományhoz tartozik
>> f = member (xr, yr, zr);

% a tartományba eső pontok száma
>> nV = sum (f)
nV =
    537

>> subplot (1, 2, 2)
>> plot3 (xr.*f', yr.*f', zr.*f', ' r*')

% a befoglaló tartomány
>> VT = 1*2*1
VT =
       2

% a  tartomány becsült értéke
>> V = nV/nT*VT
V =
   1.0488
```

```
% a valóságos értéke
>> Vexact = pi/3
Vexact =
    1.0472
```



A hiba becslése

$$d = V_T \sqrt{\frac{\overline{f^2} - \overline{f}^2}{n_T}}$$

```
% f  vektor átlaga
>> fa = sum (f)/nT
fa =
  0.5244

% a négyzetének átlaga
>> f2a = sum (f.*f)/nT
f2a =
   0.5244

% a becsült hiba
>> d = VT*sqrt ((f2a - fa^2)/nT)
d =
    0.0312

% valóságos hiba
>> dr = abs (V - Vexact)
dr = 0.0016
```

52. példa

Határozzuk meg az alábbi integrál értékét *Monte - Carlo* módszerrel,

$$I = \int_{-4}^{4}\int_{0}^{6}\int_{-1}^{3}\left(x^3 - 2\,y\,z\right) dx\,dy\,dz$$

Vegyük észre, hogy most nem egy tartomány mérőszámának, hanem egy tartományon értelmezett $\rho(x,y,z)$ függvény integráljának közelítéséről van szó!

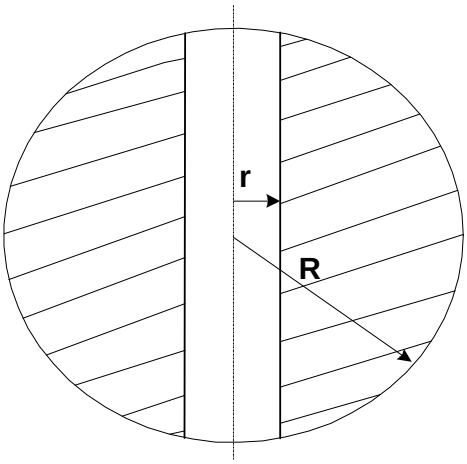
```
>> nT = 1024;
>> xyzH = haltonseq (nT, 3);
>> x1 = -1; x2 = 3; y1 = 0; y2 = 6; z1 = -4; z2 = 4;
>> xr = x1*(ones (1024, 1) - xyzH (1 : 1024, 1)) + x2*xyzH (1 : 1024, 1);
>> yr = y1*(ones (1024, 1) - xyzH (1 : 1024, 2)) + y2*xyzH (1 : 1024, 2);
>> zr = z1*(ones (1024, 1) - xyzH (1 : 1024, 3)) + z2*xyzH (1 : 1024, 3);

% az integrálandó függvény
>> ro = @(x, y, z) x.^3 - 2*y.*z;

% a függvényértékek vektora
>> f = ro (xr, yr, zr);

% a tartomány
>> VT = 8*6*4
VT = 192

% az integrál közelítő értéke
>> I = VT/nT*sum (f)
I =
   970.6509

% A pontos érték
>> Iexact=triplequad(ro,-1,3,0,6,-4,4)
Iexact =
  960.0000
```

53. példa

Határozzuk meg az alábbi hengeres üreggel rendelkező origó középpontú gömb tömegét

a) ha a sűrűség állandó, $\rho = 1$
b) ha a sűrűség a sugár növekedésével lineárisan növekszik,

$$\rho(x, y, z) = \sqrt{x^2 + y^2 + z^2}$$



Legyen r = 0.3 m és R = 1 m. Ha sűrűség állandó $\rho = 1$ akkor a tömeg:

$$\text{Ma} = \frac{4}{3}\pi\left(\text{R}^2 - \text{r}^2\right)^{\frac{3}{2}}$$

```
% generáljunk 1024 Halton pontot a [-1×1]^3 kockában

>> nT=1024;
>> VT=2*2*2;
>> R=1; r=0.3;
>> Ma=4/3*pi*(R^2-r^2)^(3/2)
Ma=
  3.63622

xyzH = haltonseq (1024, 3);

x1 = -1; y1 = -1; z1 = -1; x2 = 1; y2 = 1; z2 = 1;

xr = x1*(ones (1024, 1) - xyzH (1 : 1024, 1)) + x2*xyzH (1 : 1024, 1);
yr = y1*(ones (1024, 1) - xyzH (1 : 1024, 2)) + y2*xyzH (1 : 1024, 2);
zr = z1*(ones (1024, 1) - xyzH (1 : 1024, 3)) + z2*xyzH (1 : 1024, 3);
```

A pont a tartományban van ha a következő két feltétel egyszerre teljesül

$$\sqrt{x^2 + y^2 + z^2} < \text{R}$$
$$\sqrt{x^2 + y^2} > \text{r}$$

Ennek megfelelően a tagsági függvényünk

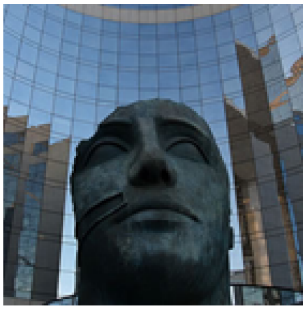
```
function y = member (xr, yr, zr)
n = length (xr);
for i = 1 : n
if and (sqrt (xr (i)^2 + yr (i)^2 + zr (i)^2) < 1, sqrt (xr (i)^2 + yr (i)^2) > 0.3)
y (i) = 1;
else
y (i) = 0;
end
end
```

```
% az indikátor vektor - eleme 1 ha a pont a tartományban van, 0 különben
>> f = member (xr, yr, zr);
```

```
% a tartományba eső pontok száma
>> nV = nnz (f)
nv =
 459

% a) eset

% a tömeg egységnyi sűrűség esetén
>> M = nV/nT*VT
M =
   3.5859

% a hiba
   >> Ma -M
ans =
    0.0503
```

- ha a sűrűség változik

```
% b) eset

% az integrálandó függvény
>> ro =
 @(x, y, z) sqrt (x.^2 + y.^2 + z.^2);

% a tartományban felvett függvényértékek
 >> F=ro(xr.*f',yr.*f',zr.*f');

 % az integrál közelítő értéke
 >> VT*sum(F)/nT
ans =
    2.7902
```



12. Gyakorlat

Kezdetiérték problémák: Euler, Runge-Kutta módszerek, javítás Richardson-féle extrapolációval, differenciálegyenlet-rendszerek

54. példa

Meghatározandók egy gömb alakú részecske mozgásának foronomiai görbéi a $t \in [0,T]$ intervallumban egy mozdulatlan közegben, ha feltételezzük, hogy a mozgást csak a közegellenállás korlátozza,

$$\frac{d\,v(t)}{dt} = -k\,v(t)\,|v(t)|$$

A kezdeti feltétel:

$$v(0) = v_0$$

Adatok : T=10, v_0 =2, k=0.2

Euler módszer

Az alábbi kézdetiérték probléma

$$\frac{dx(t)}{dt} = f(t, x(t))$$

$$x(0) = x_0$$

megoldására szolgáló Euler módszer

$$x_{i+1} = x_i + f(t_i, x_i)\,h$$

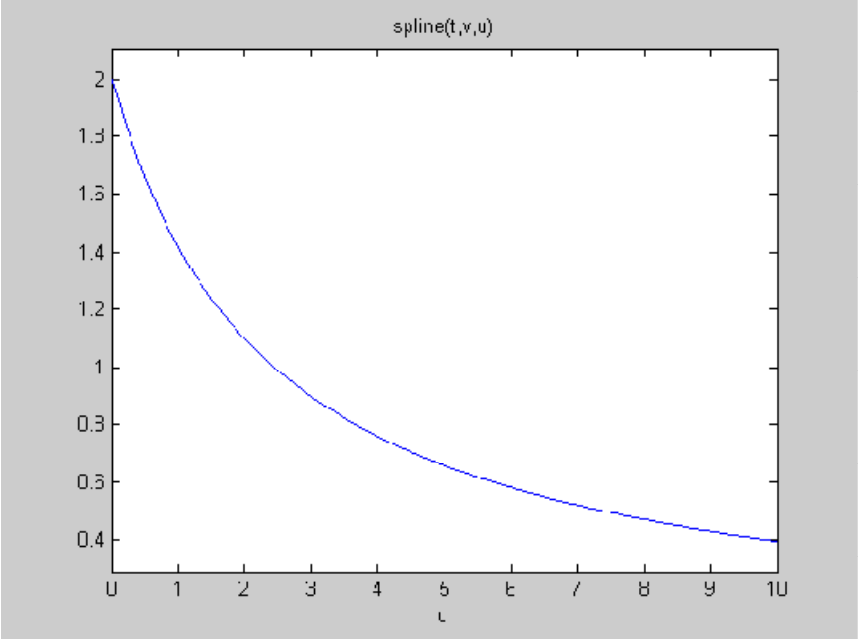
ahol *h* a lépésköz és

$$x(t_i) \approx x_i$$

```
function[t x] = euler (x0, a, b, h, f)
n = round ((b - a)/h);
t (1) = a;
x (1) = x0;
for i = 1 : n - 1
x (i + 1) = x (i) + h*f (t (i), x (i));
t (i + 1) = t (i) + h;
end
```

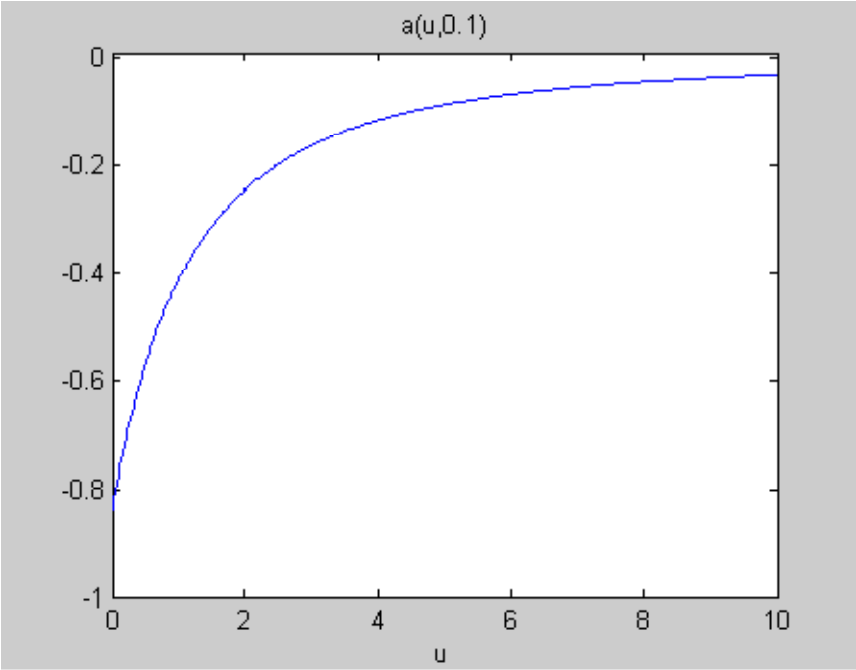
A sebesség: v = v(t)

```
% Legyen h=0.1
>> f = @(t, v) - 0.2*v*abs (v);
% most ugyan f() autonom, azaz nem függ t-től, de az euler()-t általánosan kell megírni!

>> [t v] = euler (2, 0, 10, 0.1, f);

%alkalmazzunk spline interpolációt
vp = @(u) spline (t, v, u);
>> ezplot (vp,[0,10])
```



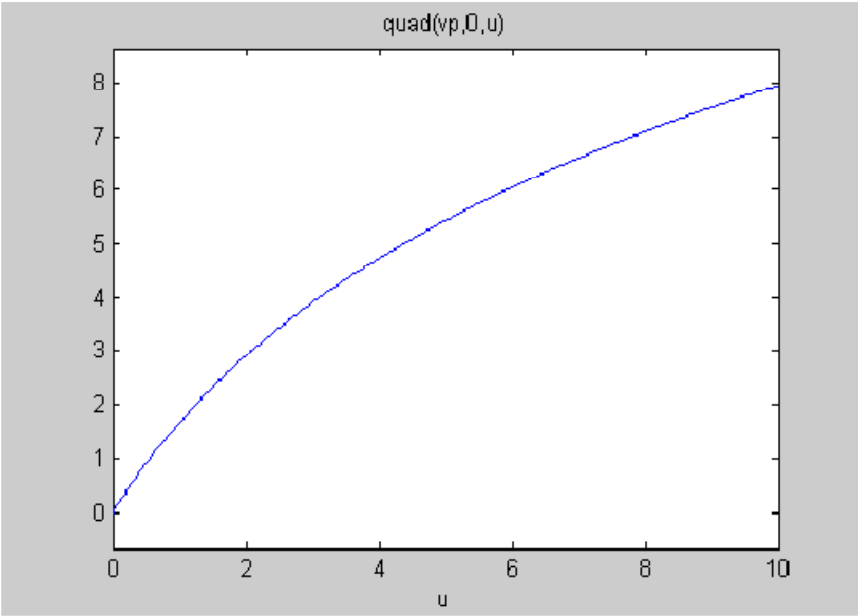
A gyorsulás a = a(t) másodrendű differencia-hányados segítségével

```
>> a = @(u, h) (vp (u + h) - vp (u - h))/(2*h)

% legyen itt is h=0.1
>> ezplot (@(u) a (u, 0.1), [0, 10])
```



A változó sebességgel megtett út: s=s(t)

```
s = @(u) quad (vp, 0, u);

ezplot (s, [0, 10])
```



Mennyire jó ez a közelítés? Oljuk meg más módszerekkel is a mozgásegyenletet!

Analítikus megoldás

```
% az analítikus megoldás
>> syms v
>> dsolve (' Dv = -0.2*v*v', ' v (0) = 2')   % itt egy kis könnyítést tettünk
ans = 10/(5 + 2*t)

% a sebesség függvény
>> vexact = @(u) 10/(5 + 2*t)

% ezzel a pontos érték
 >> vexact (10) ans = 0.4000
```

Euler csökkenő lépésközzel

```
% h = 0.05
   >> [t v] = euler (2, 0, 10, 0.05, f);
>> v (length (v))
ans = 0.3990

% h= 0.01
>> [t v]=euler(2,0,10,0.01,f);
>> v(length(v))
ans =
    0.3998

% h= 0.001
>> [t v] = euler (2, 0, 10, 0.001, f);
>> v(length(v))
ans=
    0.4000

% ellenőrzés: h= 0.0005
>> [t v] = euler (2, 0, 10, 0.0005, f);
>> v(length(v))
ans =
    0.4000

% nem változik az eredmény

% a szükséges idő
>> tic;[t v] = euler (2, 0, 10, 0.001, f);dt=toc;
>> dt
dt =
    1.3628
```

Runge-Kutta módszer

```
% megoldás beépített függvénnyel

function dv = velo (t, v)
dv (1) = -0.2*v (1)*abs (v (1));

>> options=odeset('RelTol',1e-4,'AbsTol',[1e-5]);
>> >> tic;[t v]=ode45(@velo,[0,10],[2],options);dt=toc;

>> v(length(t))
ans =
   0.4000

>> dt
dt =
   0.9528
```

Euler módszer Richardson extrapolációval

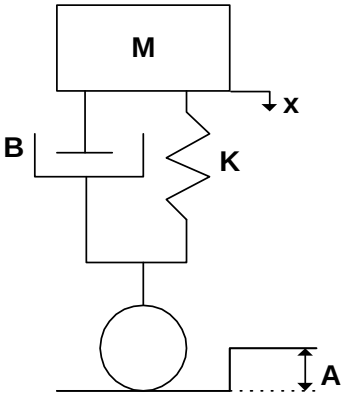
Az Euler módszer esetén $p_1=1$ és $p_2=2$. Legyen m =2. A feladat megoldásához írjunk egy m-file-t,

```
function F = rich (h)
f = @(t, v) - 0.2*v*abs (v);
[t v]=euler(2,0,10,h,f);
F=v(length(t));
```

```
% ellenőrizzük
>> rich(0.1)
ans =
   0.3980

% így h=0.1 és h/2 = 0.05 esetén a megoldás
>> tic; sol=rich(0.05) + (rich(0.05) - rich(0.1))/(2^1-1); dt=toc;
>> sol
sol =
   0.4000

% az idő
>> dt
dt =
   0.0031

% a Richardson ismét bizonyított!
```

55. példa

Egy autó rúgózásának szimulációját végezzük az alábbi egyszerű modell alapján :



A függőleges irányú mozgás mozgásegyenlete:

$$M \frac{d^2}{dt^2} x(t) + B \frac{d}{dt} x(t) + K (x(t) - A) = 0$$

A kezdeti feltételek:

$$x(0) = 0 \qquad \text{és} \qquad \frac{d}{dt} x(0) = 0$$

Alakítsuk át a másodrendű egyenletet elsőrendű rendszerré.
Legyen

$$x_1 \hat{=} x(t)$$

továbbá

$$\frac{d}{dt} x_1(t) \hat{=} x_2(t)$$

ezzel

$$M \frac{d}{dt} x_2(t) + B x_2(t) + K x_1(t) = K A$$

vagy

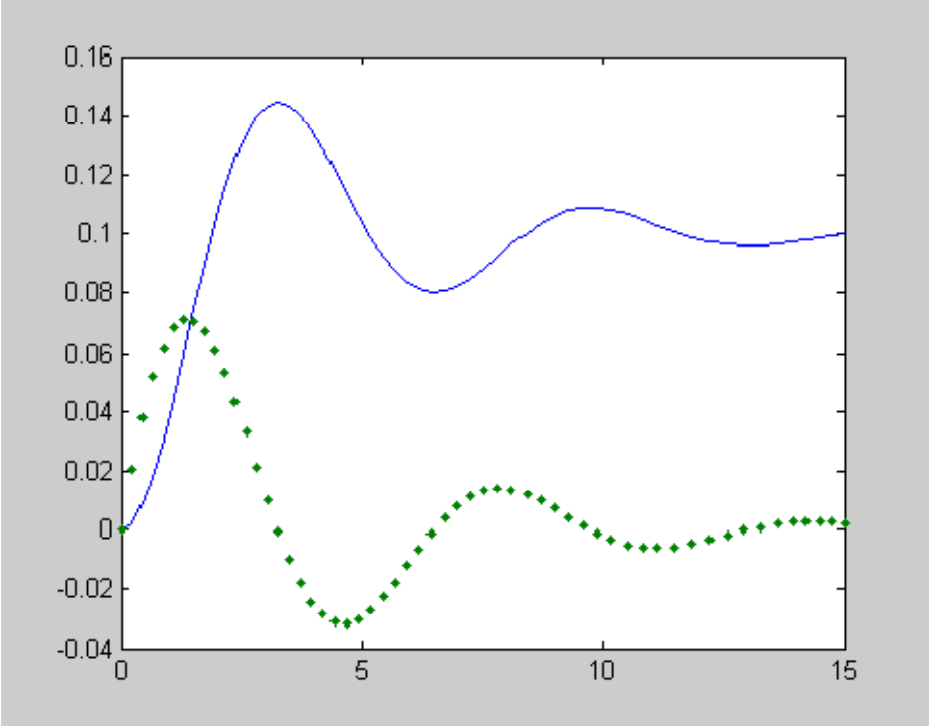
$$\frac{d}{dt} x_2(t) = \frac{1}{M} (K A - B x_2(t) - K x_1(t))$$

Az adatok: M=1000 kg, K= 1000 kg/s^2, A=0.1 m, B= 500 kg/s

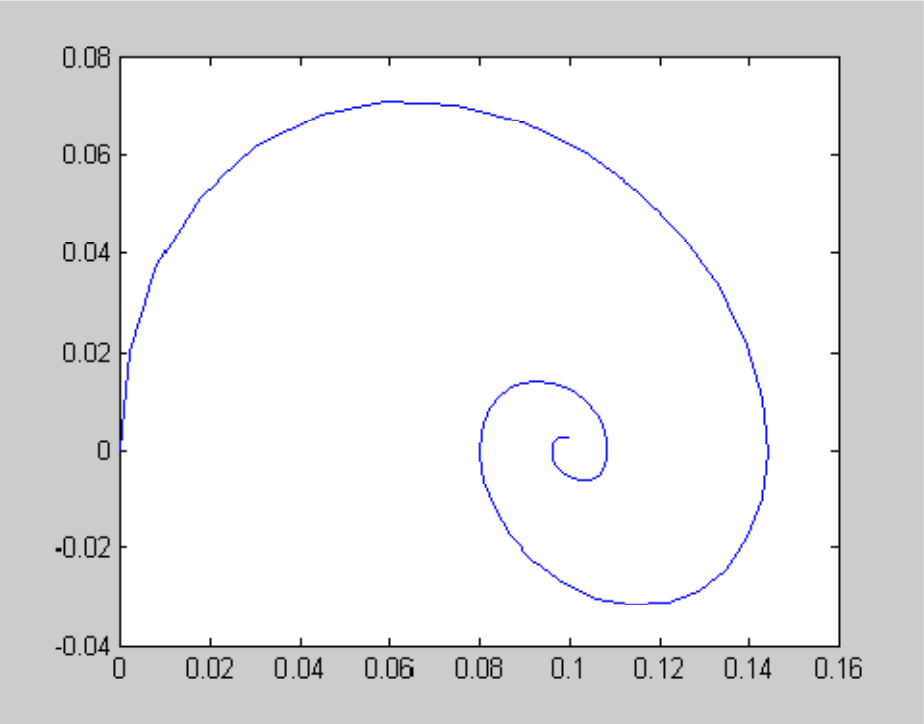
Alkalmazzuk ismét a Runge - Kutta módszert. A jobboldalnak megfelelő függvény most egy kételemű vektor **x** és a deriváltak vektora **dx**.

```
function dx = rugo (t, x)
dx = zeros (2, 1); %"deklarálni" kell a dx vektort
dx (1) = x (2);
dx (2) = (1000*0.1 - 500*x (2) - 1000*x (1))/1000;
```

```
>>options = odeset (' RelTol', 1 e - 4, ' AbsTol', [1 e - 4 1 e - 4]);

% legyen az időintervallum t∈[0 15]
>> [t,x] = ode45 (@rugo, [0 15], [0  0], options);

% a kocsiszekrény elmozdulása és sebessége az idő függvényében
>> plot (t, x (: , 1), ' - ', t, x (: , 2), '.')
```




```
% a mozgás ábrázolása a fázissíkon - az idő most paraméter
>> plot (x (: , 1), x (: , 2))
```



A folytonos függvényként is definiálhatjuk a megoldást, pl. az elmozdulást

```
>>fx=@(u)interp1(t,x (:, 1),u,'spline');
>> ezplot(fx,[0,15]);
```

Peremérték feladatok: Shooting módszer, véges differenciák és globális maradék módszere, szimbolikus megoldás.

56. példa

Tekintsük a következő másodrendű nemlineáris egyenletet:

$$(1 + y)\,\frac{d^2}{dx^2}\,y(x) + \left(\frac{d}{dx}\,y(x)\right)^2 = 0$$

az alábbi peremfeltételek mellett

$$y(0) = 0 \quad y(1) = 1$$

Elsőrendű rendszerré alakítva

$$y_2 = \frac{d}{dx}\,y_1$$

és

$$(1 + y_1)\,\frac{d}{dx}\,y_2 + {y_2}^2 = 0$$

A peremfeltételek

$$y_1(0) = 0 \qquad y_1(1) = 1$$

Oldjuk meg a feladatot, visszavezetve azt kezdetiérték probléma ismételt megoldására (shooting-módszer).

A kezdeti érték feladat:

$$d\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_2 \\ -\frac{y_2^2}{(1+y_1)} \end{pmatrix}$$

A "kezdeti" értékek - értékek az egyik peremen (x =0)

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}(0) = \begin{pmatrix} 0 \\ u \end{pmatrix}$$

ahol η ismeretlen! Helyette viszont ismert a feltétel a másik peremen, (x=1)

$$y_1(1) = 1$$

A megoldás elve, hogy megoldjuk a kezdeti érték problémát egy felvett $y_2(0) = u$ értékre, majd ellenőrizzük a

$$y_1(1) = b(u) \stackrel{?}{=} 1$$

feltétel fentállását. Ha eltérés van akkor módosítjuk az η értékét, azaz megoldjuk a

$$b(u) - 1 = 0$$

egyenletet.

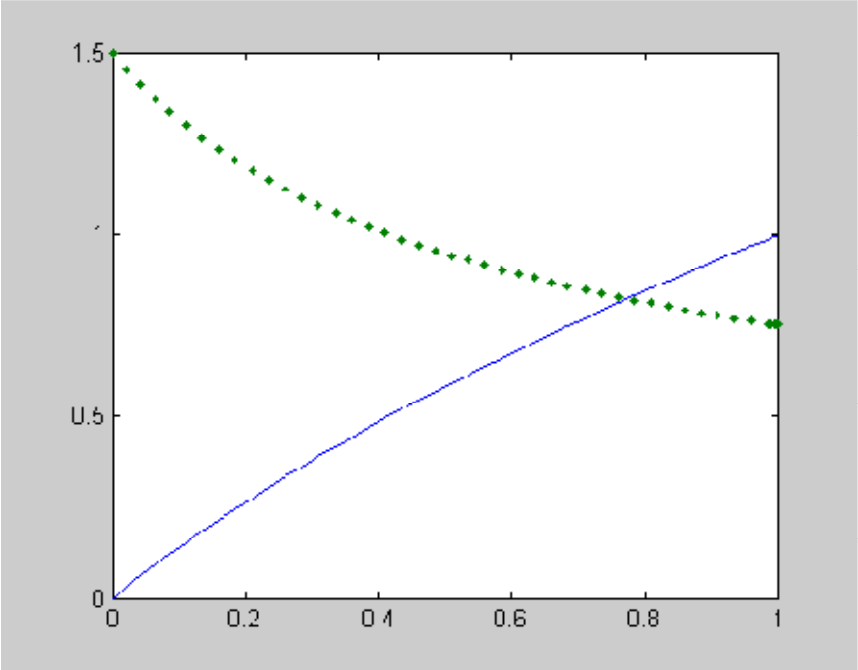
A differenciálegyenlet- rendszer jobboldala:

```
function dy = perem (x, y)
dy = zeros (2, 1);
dy (1) = y (2);
dy (2) = -y (2)^2/(1 + y (1));
```

Az ismeretlen kezdeti feltétel számítása felvett *u* értékre:

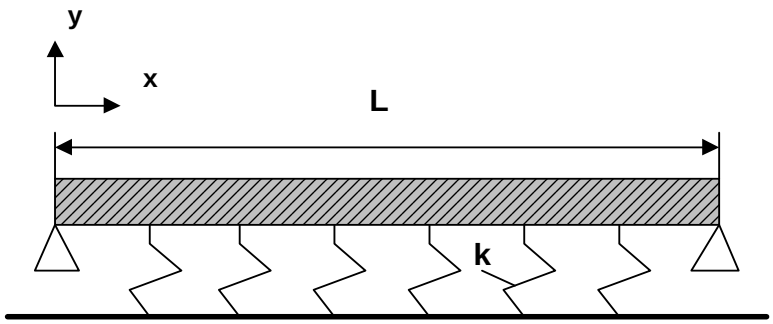
```
function y10=b(u)
options =odeset('RelTol', 1e-4, 'AbsTol', [1e-4 1e-4]);
[x,y]=ode45(@perem,[0,1],[0,u],options);
y10=y(length(y));
```

```
% a függvény, amelynek zérushelye adja a megoldást
>> c = @(u) b (u) - 1;

% a keresett kezdetiérték
>> y10=fzero(c,1)
y10 =
    1.5000

  % ezzel már megoldhatjuk  a peremérték problémát kezdetiérték feladatként
 >> options = odeset ('RelTol', 1e-4 , 'AbsTol', [1e-4 1e-4]);
>> [x,y]=ode45(@perem,[0 1], [0  y10], options);
>> plot(x, y (: , 1), ' - ', x, y (: , 2), '.')
```



57. példa

Legyen a feladat egy rugalmasan ágyazott tartó lehajlásának, szögelfordulásának, nyomatékának és nyíróerőinek a tartó hossza mentén történő változásának meghatározása.



A fenti rugalmasan ágyazott tartó alakját leíró differenciálegyenlet:

$$\frac{d^4}{dx^4} y(x) + \frac{k}{E I} y(x) = \frac{w}{E I}$$

ahol *w* a megoszló teher intenzitása.

A peremfeltételek:

$$y(0) = 0 \qquad y(L) = 0$$

$$\frac{d^2}{dx^2} y(0) = 0 \qquad \frac{d^2}{dx^2} y(L) = 0$$

Vezessük be az alábbi dimenziótlan változókat:

$$z = \frac{x}{L} \qquad \phi = \frac{E I}{w L^4} y \qquad \gamma^4 = \frac{k L^4}{E I}$$

Ezekkel

$$\frac{d^4}{dz^4} \phi(z) + \gamma^4 \phi(z) = 1$$

$$\phi(0) = 0 \qquad \phi(1) = 0 \qquad \frac{d^2}{dz^2} \phi(0) = 0 \qquad \frac{d^2}{dz^2} \phi(1) = 0$$

Elsőrendű differenciálegyenlet - rendszer formájában:

$$\frac{d}{dz} \phi_1 = \phi_2$$

$$\frac{d}{dz} \phi_2 = \phi_3$$

$$\frac{d}{dz} \phi_3 = \phi_4$$

$$\frac{d}{dz} \phi_4 + \gamma^4 \phi_1(z) = 1$$

A peremfeltételek

$$\phi_1(0) = 0 \qquad \phi_1(1) = 0$$

$$\phi_3(0) = 0 \qquad \phi_3(1) = 0$$

A hiányzó kezdetiértékek

$$\phi_2(0) = ? \qquad \phi_4(0) = 0?$$

Oldjuk meg a feladatot a shooting - módszerrel $\gamma = 0.5$ esetén:!

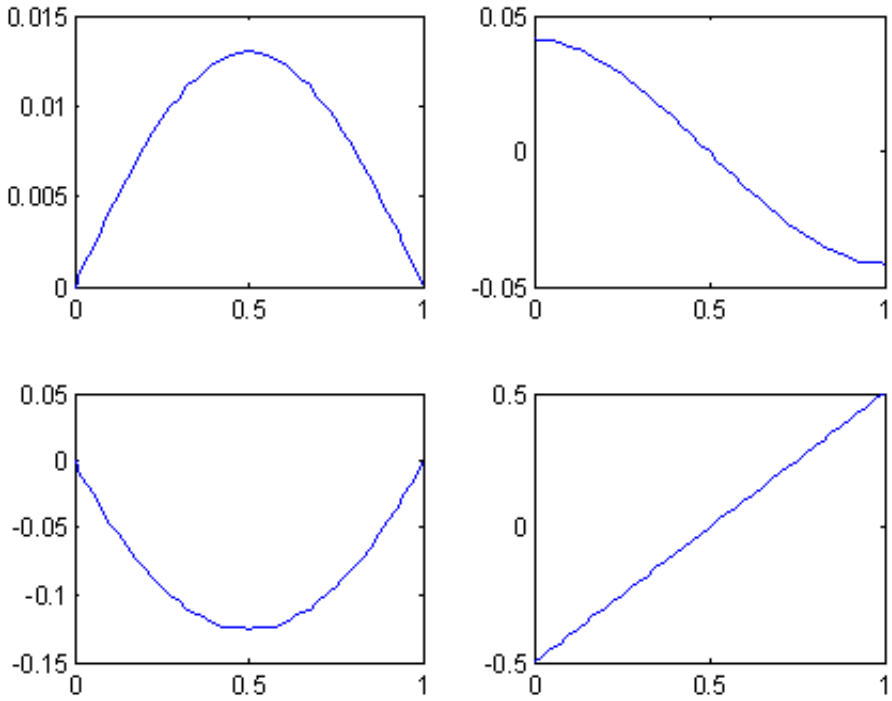
A differenciálegyenletrendszer megadása $\gamma = 0.5$ esetén:

```
function dx = tarto (t, x)
dx = zeros (4, 1);
dx (1) = x (2);
dx (2) = x (3);
dx (3) = x (4);
dx (4) = 1 - 0.5^4*x (1);
```

A függvény, amely megadja a ϕ_1 (1) és ϕ_3 (1) értékeit hiányzó kezdeti értékek ϕ_1 (0) és ϕ_3 (0) felvétele esetén,

```
function y = ingazas (fi)
options = odeset (' RelTol', 1 e - 4, ' AbsTol', [1 e - 5 1 e - 5 1 e - 5 1 e - 5]);
[t, x] = ode45 (@tarto, [0 1], [0 fi (1) 0 fi (2)], options);
n = length (t);
y = [x (n, 1); x (n, 3)];
```

```
% mivel az előírt érkek zérusok, ennek a függvénynek a zérushelyei adják a keresett kezdetiértékeket
>>fi = fsolve (@ingazas, [0, 0])
Optimization terminated : first - order optimality is less than options.TolFun.
fi =
   0.0416    - 0.4997

% ezekkel megoldva a kezdetiérték problémát
>> options = odeset (' RelTol', 1 e - 4, ' AbsTol', [1 e - 5 1 e - 5 1 e - 5 1 e - 5]);
>> [z,F] = ode45 (@tarto, [0 1], [0 fi (1) 0 fi (2)], options);

% ellenőrzés
>> n = length (z);
>>y = [F (n, 1); F (n, 3)]

y = 1.0 e - 008*
0.0289
0.7648

% az eredmény megjelenítése
>> subplot (2, 2, 1)
>> plot (t, x (: , 1))
>> subplot (2, 2, 2)
>> plot (t, x (: , 2))
>> subplot (2, 2, 3)
>> plot (t, x (: , 3))
>> subplot (2, 2, 4)
>> plot (t, x (: , 4))
```



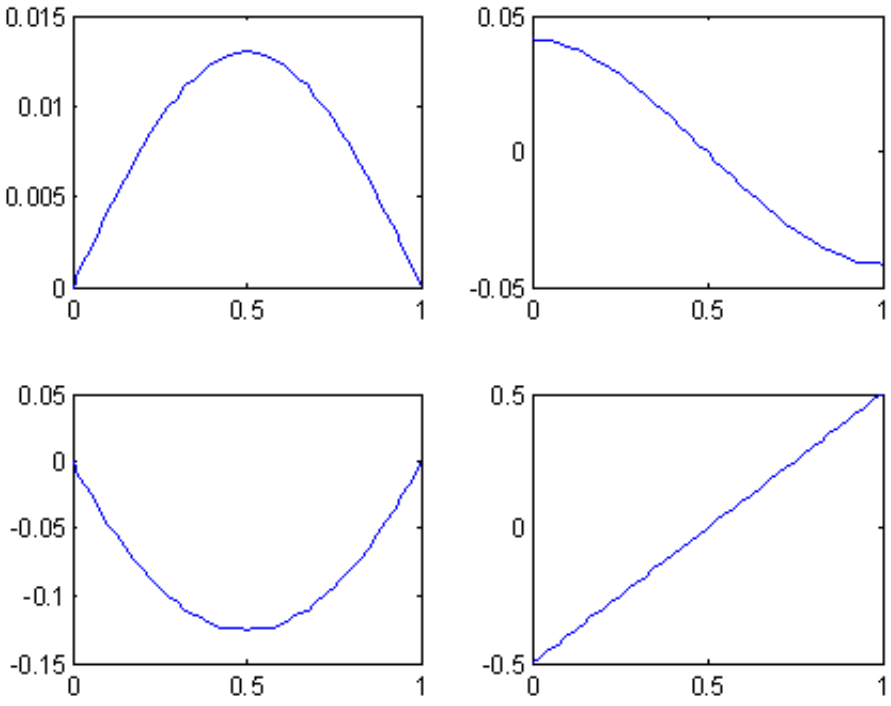
Oldjuk meg a feladot a Matlab beépített függvényével (*bvp4c*)!

A peremértékek megadása

```
function res = perem (xa, xb)
res = [xa (1) - 0; xb (1) - 0; xa (3) - 0; xb (3) - 0];
```

```
% a kezdeti értékek megadása
>> solinit=bvpinit(linspace(0,1,10),[0,0,0,0]); % ez végzi el az iterációt, mint a shooting módszernél

%opciók a hibakorlátra
>> options=bvpset('RelTol',1e-4);

%a peremértékfeladot megoldó függvény hívása
>> sol=bvp4c(@tarto,@perem,solinit,options);

%a független és függő változók értékei
>> x=sol.x;
```

```
>> length(x)
ans =
    15

Y=sol.y;
>> size(Y)
ans =
     4    15

%a peremértékek ellenőrzése
>> Y(1,15)
ans =
     0
>> Y(3,15)
ans =
     0

%az eredmények grafikus megjelenítése

subplot(2,2,1)
plot(x,Y(1,:))
subplot(2,2,2)
plot(x,Y(2,:))
subplot(2,2,3)
plot(x,Y(3,:))
subplot(2,2,4)
plot(x,Y(4,:))
```



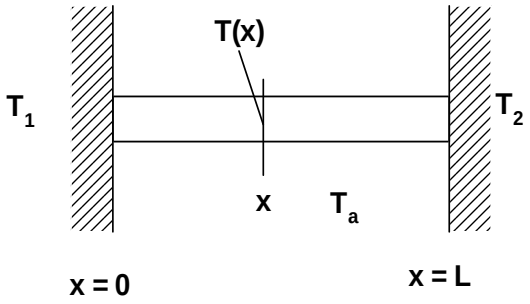
58. példa

Határozzuk meg a hőmérséklet eloszlását egy L = 10 hosszúságú rúd hossza mentén, ha a véghőmérsékletek:

$$T(0) = 40 \quad \text{és} \quad T(L) = 200$$

A környezet hőmérséklete: $T_a = 30$ és a hőátadási tényező értéke k = 0.01. A hőmérséklet eloszlását leíró egyenlet

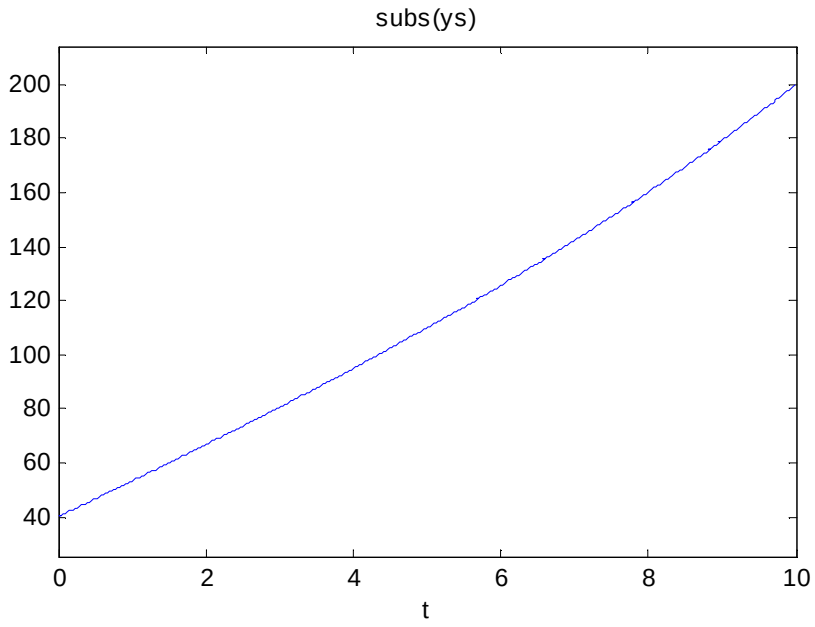
$$\frac{d^2}{dx^2} T(x) + k\,(T_a - T(x)) = 0$$



Szimbolikus megoldás

Mivel ez egy *lineáris* egyenlet a peremfeltétel probléma megoldható szimbolikusan is

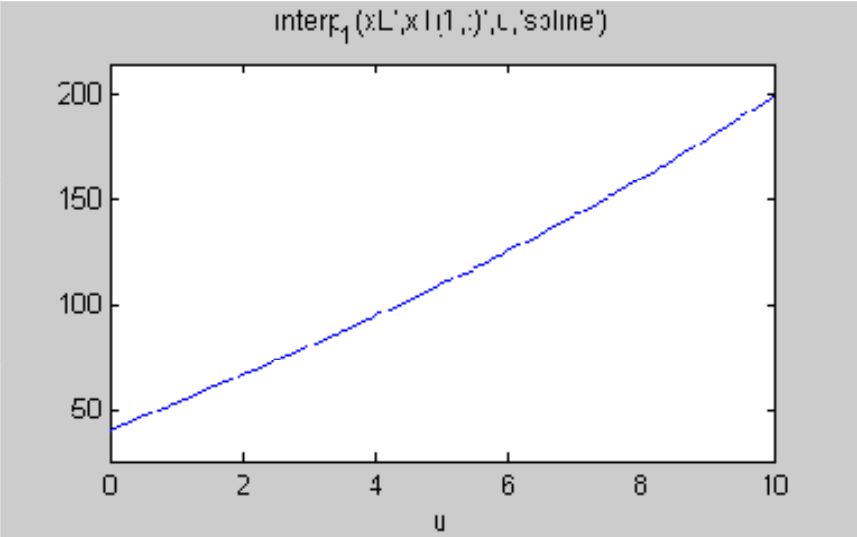
```
>> clear all

% megadjuk a numerikus értékeket
>> k = 0.01; Ta = 30;

% a független változót deklaráljuk
>> syms t

% megoldjuk az egyenletet szimbolikusan
>> ys = dsolve (' D2T + k*(Ta - T) = 0', ' T (0) = 40', ' T (10) = 200');

% a kapott eredmény hosszú és kezelhetettlen ezért rendezetten íratjuk ki (pretty)
>> pretty (simplify (ys))

% a numerikus értékeket behelyettesítve, definiáljuk a numerikus anonymous függvényt
 >> yn = @(x) subs (ys)

% ábrázoljuk a profilt
 >>ezplot(yn,[0,10])
```



Beépített függvénnyel (bvp4c)

```
% a differenciálegyenlet-rendszer
function dx=rud(t,x)
dx=zeros(2,1);
dx(1)=x(2);
dx(2)= 0.01*(x(1)-30);

% peremfeltételek
function res=peremr(xa,xb)
res=[xa(1 )- 40; xb(1) - 200];

% független változó intervalluma, lépésköz és a kezdeti feltételek becslése
>> solinit=bvpinit(linspace(0,10,20),[40,0]);

% opciók
>> options=bvpset('RelTol',1e-4);

% megoldás
 sol=bvp4c(@rud,@peremr,solinit,options);

 %eredmények
 >> xL=sol.x;
```

```
>> xT=sol.y;

% ellenőrzése a peremfeltételnek
>> xT(1,20)
ans =
  200

% a megoldás interpolációs függvénye
>> T=@(u)interp1(xL',xT(1,:)',u,'spline');
>> ezplot(T,[0,10]);
```



Véges differenciák módszere

Osszuk fel a kérdése intervallumot N = 4 részre és közelítsük a profilt ezekben az osztópontokban:

$$T_i \approx T(i\,\Delta x), \quad i = 0,\ 1,\ \ldots,\ 4$$

ahol $\Delta x = L/N = 10/4 = 2.5$. A második derivált közelítése a belső pontokban, (i=1, 2, 3):

$$\frac{d^2}{dx^2} T(i\,\Delta x) \approx \frac{T_{i+1} - 2\,T_i + T_{i-1}}{\Delta x^2}$$

ennek megfelelően a differencia egyenletek ezekben a pontokban figyelembevéve a peremfeltételeket

$$i = 1 \qquad \frac{T_2 - 2\,T_1 + T(0)}{\Delta x^2} + k\,(T_a - T_1) = 0$$

$$i = 2 \qquad \frac{T_3 - 2\,T_2 + T_1}{\Delta x^2} + k\,(T_a - T_2) = 0$$

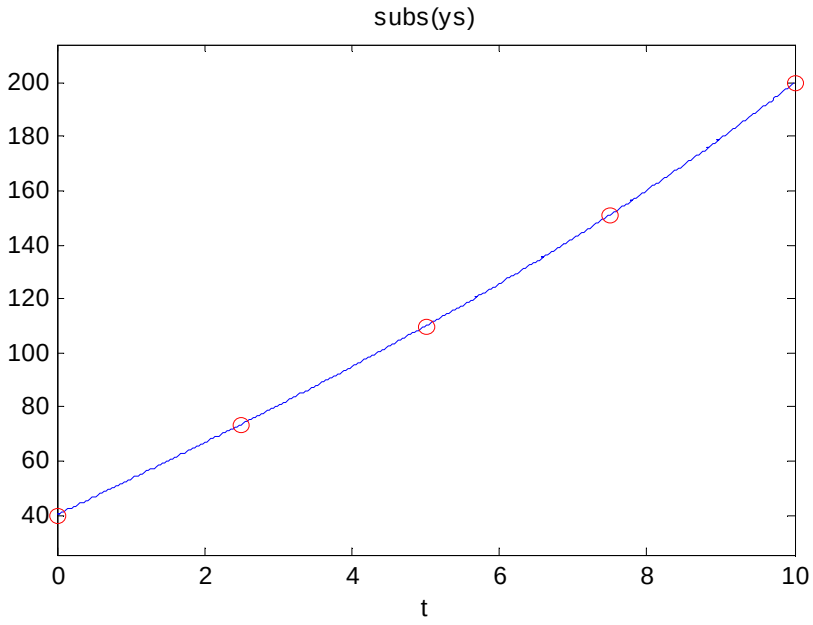
$$i = 3 \qquad \frac{T(L) - 2\,T_3 + T_2}{\Delta x^2} + k\,(T_a - T_3) = 0$$

Ez egy lineáris egyenletrendszer a T_i (i = 1, 2, 3) ismeretlenekre :

$$\begin{pmatrix} -\left(\frac{2}{\Delta x^2}+k\right) & \frac{1}{\Delta x^2} & 0 \\ \frac{1}{\Delta x^2} & -\left(\frac{2}{\Delta x^2}+k\right) & \frac{1}{\Delta x^2} \\ 0 & \frac{1}{\Delta x^2} & -\left(\frac{2}{\Delta x^2}+k\right) \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} = -\begin{pmatrix} k\,T_a + \frac{T(0)}{\Delta x^2} \\ k\,T_a \\ k\,T_a + \frac{T(L)}{\Delta x^2} \end{pmatrix}$$

```
% az adatok
>> dx = 2.5; T0 = 40; TL = 200;

% célszerű bevezetni, mint új változót
>> idx2 = 1/dx^2
idx2 =
    0.1600

% az együttható mátrix elemei
>> A = [-(2*idx2 + k), idx2, 0;
        idx2, -(2*idx2 + k), idx2;
           0, idx2, -(2*idx2 + k)]

A =
   -0.3300    0.1600    0
    0.1600  - 0.3300    0.1600
    0          0.1600  - 0.3300

% a jobboldal vektora
>> b = -[k*Ta + T0*idx2; k*Ta; k*Ta + TL*idx2]

b =
  -6.7000
  -0.3000
  -32.3000

% az egyenletrendszer megoldása
>> Tsol = linsolve (A, b)

Tsol = 73.5691
       109.8614
       151.1449

% a kapott diszkrét pontokban a hőmérséklet értékek

>> Tp = [T0; Tsol; TL]
```

```
Tp =
40.0000
73.5691
109.8614
151.1449
200.0000

% a megfelelő helykoordináták
>> xp = [0 dx 2*dx 3*dx 4*dx]'
xp = 0
     2.5000
     5.0000
     7.5000
     10.0000

% rajzoljuk fel a pontokat az szimbolikus megoldás eredményeként adódó  profilra
>> plot (xp, Tp, ' ro')
```



Globális maradék módszere

Legyen a peremfeltételeket kielégítő hőmérséklet profil másodfokú:

$$T(x) = \frac{1}{L}(T(L)\,x + (L - x)\,T(0)) + x\,(x - L)\,a$$

ahol *a* egy ismeretlen állandó. Ezt az állandót úgy határozzuk meg, hogy a közelítő profilt behelyettesítve a differenciálegyenletbe, a kapott *x*-től függő kifejezés (lokális maradék) négyzetének integrálja (itt most abszolútértékének) a teljes tartományra (globális maradék), minimális legyen.
Mivel

$$\frac{d^2}{dx^2} T(x) = 2\,a$$

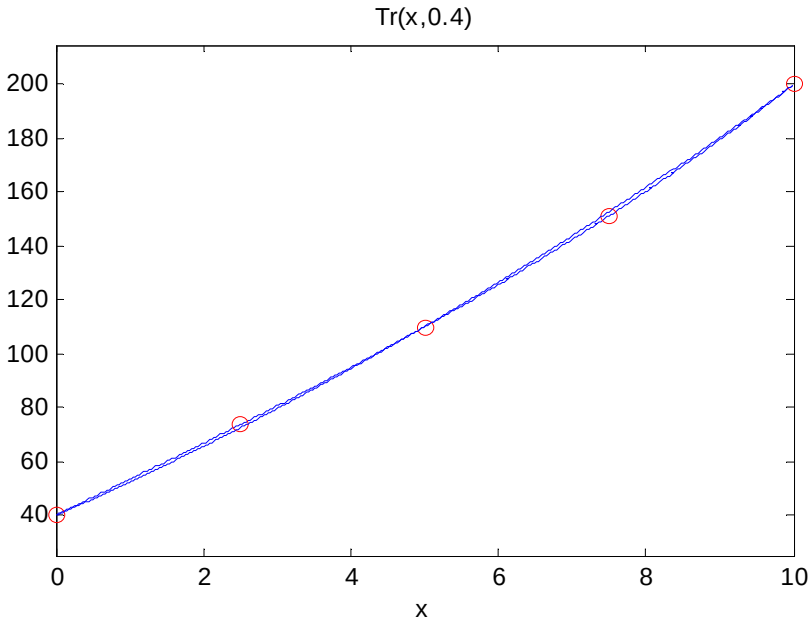
a lokális maradék

$$R(x, a) = 2a + k(T_a - T(x))$$

Tehát minimalizálandó függvény:

$$G(a) = \int_0^L |(R(x, a))\, dx| \quad \rightarrow \min_a$$

```
% a közelítő profil a-val, mint parameterrel - vektoros definició szükséges a későbbi integrálás miatt
>> Tr = @(x, a) (1/10)*(200*x + (10 - x)*40) + x.*(x - 10)*a;

% A lokális maradék abszolútértéke
>> R = @(x, a) abs (2*a + 0.01*(30 - Tr (x, a)));

% A minimalizálandó függvény
>> G=@(a)quad(@(x)R(x,a),0,10);

% a megoldás
>> fminsearch(G,1)
ans =
    0.4000

% rajzoljuk be az ábrába a profilt
>> ezplot(@(x)Tr(x,0.4),[0,10])
```



```
% nézzük meg az egyes megoldások hibáit pontosabban
% a véges differenciák diszkrét pontjait spline-nal közelítjük

>> Tv = @(x) spline (xp, Tp, x)

% ennek lokális hibája
 >> errv = @(x) yn (x) - Tv (x)

 % a globális maradék módszer lokális hibája
 >> errR = @(x) yn (x) - Tr (x, 0.4)

 % rajzoljuk fel a hibafüggvényeket
 >>figure(2)

>> subplot (1, 2, 1)
>> ezplot (errv, [0, 10])

>>hold on

 >> subplot (1, 2, 2)
 >> ezplot (errR, [0, 10])
```