MATLAB

MATLAB = ,,Matrix laboratory”

Részletes leírás, help: `http://www.mathworks.com/help/`

Az alkalmazás elindítása után (az alapértelmezett beállítással) az ablak 3 részre tagolódik:

- Command Window (parancsablak)
- Current Folder
- Workspace

Az ablak tagolása menüből változtatható:

HOME $\rightarrow$ Layout

Az alapértelmezett tagolás visszállítása:

HOME $\rightarrow$ Layout $\rightarrow$ Default

A parancsablakba utasításokat gépelhetünk, pl:

```
>> 3+4
ans =
      7

>> 3*1.5
ans =
      4.5000

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

A Matlab-ban nem tizedesvessző, hanem ,,tizedespont" van.

Ha másképp nem rendelkezünk, akkor az eredmény az `ans` nevű változóba kerül.

Használhatunk más változókat is, pl.:

```
>> a=3+4
a =
     7

>> a=3; b=4; c=a+b
c =
     7
```

Ha egy értékadó utasítást pontosvesszővel zárunk le, akkor az értékadás végrehajtódik, de az eredmény nem jelenik meg a parancsablakban. Pl.:

```
>> a=3; b=4; c=a+b;
```

A változó értékét ekkor is megkérdezhetjük, nevének begépelésével:

```
>> c
c =
     7
```

Változónevek

- Betűvel kell kezdődniük, tartalmazhatnak betűket, számokat, aláhúzást. Megkülönbözteti a kis- és nagybetűket. Ne használjunk ékezetes betűket!
- Nem lehetnek változónevek a Matlab kulcsszavai (pl. `if`, `end`, stb), az `iskeyword` utasítással felsoroltathatjuk ezeket a kulcsszavakat.
- Figyeljünk rá, hogy ne használjuk változónévként Matlab-függvények neveit (pl. `cos`, `size`, stb). Ha nem vagyunk biztosak benne, hogy egy név létezik-e már, akkor az `exist` függvénnyel ellenőrizhetjük (pl. `exist cos`)
- A `clear` utasítással törölhetünk változókat (pl. `clear a,b` törli az `a` és `b` változókat). A `clear all` utasítással minden változó törlődik.

M-fájlok

A Matlab futtatható állományai az M-fájlok.

- Nyissunk meg a szerkesztőablakban egy új fájlt:

  Kattintsunk a bal felső sarokban a + ikonra, vagy

  New → Script
- Írjuk ide a programunkat

```
% elso kod
  a=3;  b=4;
  c=a+b;
  disp(c)
```

A megjegyzéseinket %-jel mögött helyezhetjük el.

```
% elso kod
  a=3;  b=4;
  c=a+b;
  disp(c)
```

- A megjegyzéseinket %-jel mögött helyezhetjük el.
- A változókat nem szükséges inicializálni.
- Egy sorba akár több utasítás is kerülhet, ha ezeket pontosvesszővel választjuk el.
- Itt is figyeljünk az értékadó utasítások után a pontosvesszőkre, ha lemarad, akkor annak eredménye futás közben megjelenik a parancsablakban.
- A `disp` függvény kiírja az adott változó értékét.

M-fájlok

- Mentsük el a fájlt.

  Olyan könyvtárba mentsünk, amelyet a Matlab el tud érni. Ezek listáját megkaphatjuk, ha a parancsablakba a `path` utasítást gépeljük, vagy menüből:

  HOME $\rightarrow$ Set Path

  A fájl `.m` kiterjesztésű legyen, pl. `proba.m`
- Futtassuk a programunkat.

  Írjuk be a fájl nevét a parancsablakba kiterjesztés nélkül:

  `>> proba`

  vagy menüből

  EDITOR $\rightarrow$ Run

A Matlab függvényekről a parancsablakban a `help` parancs segítségével kérhetünk leírást, pl.

```
>> help disp
```

A parancsablakban a korábbi utasításaink visszahívhatóak a felfele mutató nyíl (↑) nyomogatásával.

Ha a felfele mutató nyilat néhány karakter begépelése után nyomjuk meg, akkor az ilyen karaktersorozattal kezdődő utasításokat hívja vissza.

A létező változóink a Workspace ablakban vannak felsorolva, de felsoroltathatjuk a parancsablakban is a `who` paranccsal:

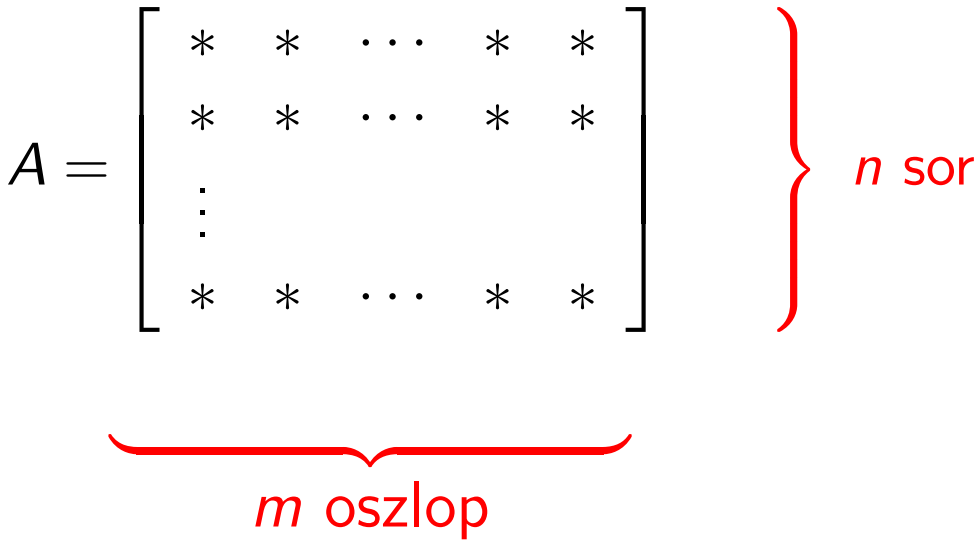
```
>> who
Your variables are:
a  b  c
```

A `whos` parancs részletesebb leírást ad a változókról:

```
>> whos
  Name        Size                Bytes  Class     Attributes
  a           1x1                     8  double
  b           1x1                     8  double
  c           1x1                     8  double
```

Mátrixok, vektorok

A Matlab alapvető változói a mátrixok.
Mátrix = egy kétdimenziós tömb:



Ekkor A egy $n \times m$-es tömb (mátrix)

Sorvektorok

Sorvektor: olyan mátrix, melynek egyetlen sora van:

$$a = \begin{bmatrix} * & * & \cdots & * & * \end{bmatrix}$$

Sorvektorok létrehozása Matlab-ban

szögletes zárójelek között elemeinek felsorolásával, pl:

- az elemeket vesszővel választjuk el:

  `a =[-1.2, 3.1, 4.7, 1.9]`

- vagy az elemeket szóközzel választjuk el:

  `a=[-1.2 3.1 4.7 1.9]`

A vektor elemeinek számozása 1-gyel kezdődik, `a(i)` az a vektor i-edik eleme.

A : (kettőspont) operátor

Ha a vektor elemei szabályos lépésközzel követik egymást, akkor használhatjuk a kettőspont operátort:

- a $b = [1, 2, 3, 4, 5]$ vektor:

  `b = 1:5`

- a $c = [5, 4, 3, 2, 1]$ vektor:

  `c = 5:-1:1`

- a $d = [2, 2.2, 2.4, 2.6, 2.8, 3]$ vektor

  `d=2:0.2:3`

Általában:

```
x=elsoelem:lepeskoz:utolsoelem
```

ahol a lépésköz negatív is lehet, vagy

```
x=elsoelem:utolsoelem
```

ekkor a lépésköz 1.

A linspace függvény

Ha megadott számú, egyenlő lépésközű elemet szeretnénk megadni:

```
x=linspace(elsoelem,utolsoelem,elemekszama)
```

Pl. az

```
e=linspace(1,2,6)
```

utasítás egy 6 elemű vektort ad meg, melynek első eleme 1, utolsó eleme 2, és az elemek egyforma lépésközzel követik egymást:

$$e = [1, 1.2, 1.4, 1.6, 1.8, 2]$$

Ha a linspace függvényt csak két argumentummal hívjuk, akkor 100 elemű vektort kapunk:

```
x=linspace(elsoelem,utolsoelem)
```

Néhány hasznos függvény

- size

  Megadja egy mátrix sorainak és oszlopainak számát.

  Pl. $a = [1, -2, 0, 5]$ esetén `size(a)` értéke: `[1,4]`

- numel

  Megadja a mátrix elemeinek számát.

  Pl. az előző a-val `numel(a)` értéke 4.

- ones

  `ones(n,m)` egy $n \times m$-es, csupa 1-esből álló mátrix.

  Pl. `ones(1,5)` egy 5 elemű, csupa 1-es sorvektor.

- zeros

  `zeros(n,m)` egy $n \times m$-es, csupa 0-ból álló mátrix.

  Pl. `zeros(1,5)` egy 5 elemű, csupa 0 sorvektor.

Oszlopvektorok

Oszlopvektor: olyan mátrix, melynek egyetlen oszlopa van:

$$a = \begin{bmatrix} * \\ * \\ \vdots \\ * \\ * \end{bmatrix}$$

Oszlopvektorok létrehozása Matlab-ban

- szögletes zárójelek között elemeinek felsorolásával, az elemeket pontosvesszővel választjuk el:

  ```
  a =[-1; 3; 0; 1]
  ```

- vagy egy sorvektort transzponálunk:

  ```
  a=[-1, 3, 0, 1]; b=a'
  ```

A vektor elemeinek számozása 1-gyel kezdődik, `a(i)` az a vektor i-edik eleme.

Transzponálás

Sorból oszlopot, oszlopból sort csinál:

- $$a = [-1, 3, 0, 1] \quad \Longrightarrow \quad a' = \begin{bmatrix} -1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$
- $$a = \begin{bmatrix} -1 \\ 3 \\ 0 \\ 1 \end{bmatrix} \quad \Longrightarrow \quad a' = [-1, 3, 0, 1]$$

(valójában a ’ jel konjugált transzponáltat eredményez, a konjugálás nélküli transzponátás: `a.'` **vagy** `transpose(a)`**. Ez csak akkor jelent különbséget, ha a vektor elemei nem valós számok.)**

Példák

`a=[1;2;3]`
vagy
`a=[1, 2, 3]'`
vagy
`a=(1:3)'`
	$a = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

`a=(1:0.2:2)'`
vagy
`a=(linspace(1,2,6))'`
	$a = \begin{bmatrix} 1 \\ 1.2 \\ 1.4 \\ 1.6 \\ 1.8 \\ 2 \end{bmatrix}$

Néhány hasznos függvény

A sorvektoroknál bemutatott függvényeket itt is használhatjuk. (Figyeljünk a különbségekre!)

- size
  Megadja egy mátrix sorainak és oszlopainak számát.
  Pl. $a = [1; -2; 0; 5]$ esetén `size(a)` értéke: `[4,1]`
- numel
  Megadja a mátrix elemeinek számát.
  Pl. az előző a-val `numel(a)` értéke 4.
- ones
  `ones(n,m)` egy $n \times m$-es, csupa 1-esből álló mátrix.
  Pl. `ones(5,1)` egy 5 elemű, csupa 1-es oszlopvektor.
- zeros
  `zeros(n,m)` egy $n \times m$-es, csupa 0-ból álló mátrix.
  Pl. `zeros(5,1)` egy 5 elemű, csupa 0 oszlopvektor.

Vektorok darabolása

Sor- és oszlopvektorokra is:

`v(2)`	a v vektor 2. eleme
`v([2,4])`	a v vektor 2. és 4. eleméből álló vektor
`v(2:4)`	a v vektor 2., 3. és 4. eleméből álló vektor
`v(end)`	a v utolsó eleme
`v(2)=[]`	elhagyja a vektor 2. elemét
`v([2,4])=[]`	elhagyja a vektor 2. és 4. elemét
`v(2:4)=[]`	elhagyja a vektor 2., 3. és 4. elemét

A fenti utasítások eredménye aszerint lesz sor-, vagy oszlopvektor, hogy a v sor-, vagy oszlopvektor volt-e.

Emlékeztető: a vektor elemeinek számozása 1-gyel kezdődik.

Vektorelemek módosítása

Az előzőeket felhasználva módosíthatjuk egy vektor elemeit. Pl.: ha

$$v = [-1, 4, 6, 0, -3, 5]$$

akkor (pirossal jelölve a módosított elemeket)

- `v(2)=-5` $\Longrightarrow$ $v = [-1, \color{red}{-5}, 6, 0, -3, 5]$
- `v([2,4])=[-5,1]` $\Longrightarrow$ $v = [-1, \color{red}{-5}, 6, \color{red}{1}, -3, 5]$
- `v([2,4])=1` $\Longrightarrow$ $v = [-1, \color{red}{1}, 6, \color{red}{1}, -3, 5]$
- `v(2:4)=[-5,-2,1]` $\Longrightarrow$ $v = [-1, \color{red}{-5}, \color{red}{-2}, \color{red}{1}, -3, 5]$
- `v(2:4)=1` $\Longrightarrow$ $v = [-1, \color{red}{1}, \color{red}{1}, \color{red}{1}, -3, 5]$

Ha a vektor egy részét módosítani akarjuk, akkor egy ugyanolyan méretű vektorral kell egyenlővé tennünk, vagy egyetlen számmal. Utóbbi esetben minden módosítandó elemet arra a számra cserél.

Fontos! A `v(9)=4` utasítás eredménye az `v=[-1,4,6,0,-3,5,0,0,4]` vektor (a legkisebb olyan vektor, amelyben van értelme a 9. elemre hivatkozásnak, a nemdefiniált elemeket 0-kal tölti fel. Megváltozik a vektor mérete, erre nem figyelmeztet!)

Vektorok összefűzése

Emléleztető: szögletes zárójelben az elemeket

- vesszővel vagy szóközzel választjuk el $\Longrightarrow$ sorvektor
- pontosvesszővel választjuk el $\Longrightarrow$ oszlopvektor

Ugyanígy építhetünk össze vektorokat is, ha a méretük engedi (az eredménynek ,,téglalap alakúnak" kell lenni).

Hüvelykujj-szabály: ha a szögletes zárójelen belül az utolsó elem után

- vesszővel, vagy szóközzel elválasztva írjuk az új elemet, akkor azt az utolsó elem után helyezi el (ha lehetséges)
- pontosvesszővel elválasztva írjuk az új elemet, akkor sort tör, az új elemet az eddigiek alá helyezi (ha lehetséges)

Vektorok összefűzése

`[a b]` vagy `[a,b]`	két sorvektor egymás után fűzése
`[-4 a 3 -1]`	sorvektor bővítése újabb elemekkel

Példa: Ha

$$\mathtt{a} = \begin{bmatrix} 5 & -1 & 2 \end{bmatrix}, \mathtt{b} = \begin{bmatrix} 2 & -7 & 3 & -1 \end{bmatrix},$$

akkor

$$\mathtt{[a\ b]} = \begin{bmatrix} 5 & -1 & 2 & 2 & -7 & 3 & -1 \end{bmatrix}$$

$$\mathtt{[-4\ a\ 3\ -1]} = \begin{bmatrix} -4 & 5 & -1 & 2 & 3 & -1 \end{bmatrix}$$

Amit nem lehet: Az `a` és `b` vektorokat nem helyezhetem egymás alá, mert nem egyforma hosszúak.

Ha egyforma hosszúak lennének, akkor egymás alá rakhatnánk őket $\Longrightarrow$ egy 2 sorból álló mátrixot kapnánk.

Vektorok összefűzése

`[m;n]` két oszlopvektor egymás után fűzése

`[1;m;-3]` oszlopvektor bővítése újabb elemekkel

Példa: Ha

$$\mathtt{m} = \begin{bmatrix} -2 \\ 1 \\ 6 \end{bmatrix}, \mathtt{n} = \begin{bmatrix} 0 \\ 4 \end{bmatrix},$$

akkor

$$\mathtt{[m;n]} = \begin{bmatrix} -2 \\ 1 \\ 6 \\ 0 \\ 4 \end{bmatrix}, \quad \mathtt{[1;m;-3]} = \begin{bmatrix} 1 \\ -2 \\ 1 \\ 6 \\ -3 \end{bmatrix}$$

Amit nem lehet: Az `m` és `n` vektorokat nem tehetjük egymás mellé, mert nem ugyanolyan hosszúak.

Ha egyforma hosszúak lennének, akkor egymás mellé rakhatnánk őket $\Longrightarrow$ egy 2 oszlopból álló mátrixot kapnánk.

Aritmetikai műveletek vektorokkal

Ha `a` és `b` két ugyanolyan méretű vektor, akkor

- `a+b` ill. `a-b` a két vektor elemenkénti összege, ill. különbsége
- `a+1` az `a` minden eleméhez hozzáad 1-et
- `a.^ 2` az `a` minden elemét négyzetre emeli
- `a.*b` az `a` és `b` vektorok elemenkénti szorzata,
- `a./b` az `a` és `b` vektorok elemenkénti hányadosa
- `1./a` az `a` elemenkénti reciproka

Az utolsó négy esetben a műveleti jel előtti pont a művelet elemenkénti végrehajtását eredményezi. A pont nélküli műveletek mást jelentenek, ld. később.

`sin`, `cos`, `tan`, `exp`, `log`, `sqrt`, `abs`, stb. mind elemenként hajtódik végre.

`NaN` : Not a Number (pl. 0/0, `Inf/Inf`)

Példák

$$\texttt{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \texttt{b} = \begin{bmatrix} -2 & 4 & 1 \end{bmatrix}$$

- a+b $= \begin{bmatrix} -1 & 6 & 4 \end{bmatrix}$
- a+1 $= \begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$
- a.^ 2 $= \begin{bmatrix} 1 & 4 & 9 \end{bmatrix}$
- a.*b $= \begin{bmatrix} -2 & 8 & 3 \end{bmatrix}$
- a./b $= \begin{bmatrix} -0.5 & 0.5 & 3 \end{bmatrix}$
- 1./a $= \begin{bmatrix} 1 & 0.5 & 0.3333 \end{bmatrix}$

Ha a pontot nem szerepel a $*$ előtt, akkor:

```
>> a*b
```

Error using * Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To perform elementwise multiplication, use '.*'.

Néhány hasznos függvény

- `min(x)` és `max(x)` az x vektor legkisebb és legnagyobb eleme
- `sort(x)` az x elemeit növekvő sorrendbe rendezi
- `sort(x,'descend')` az x elemeit csökkenő sorrendbe rendezi
- `flip(x)` az x elemeit fordított sorrendben sorolja fel
- `length(x)` az x vektor sor és oszlop száma közül a nagyobb
- `numel(x)` az x elemeinek száma
- `sum(x)` az x vektor elemeinek összege
- `prod(x)` az x vektor elemeinek szorzata
- `mean(x)` az x vektor elemeinek átlaga
- `x(3)` az x vektor harmadik eleme
- `x(1:3)` az x vektor első három eleme
- `x(3:end)` az x vektor minden elemei a harmadiktól az utolsóig

for-ciklus

```
for ciklusvaltozo=vektor
  utasitasok
end
```

- A `for` és az `end` kulcsszavak határolják a for-ciklust.
- A törzsben szereplő utasítások általában függenek a ciklusváltozó értékétől.
- A ciklusváltozó lehetséges értékei a ciklus fejében szerepelnek, a ,,vektor"-ban felsorolva.
- Az utasításokat a ciklusváltozó minden lehetséges értékére végrehajtja.

```
for k=1:3
    disp(['A ciklusvaltozo erteke most:', num2str(k)]);
end
```

- A ciklusváltozó neve ebben az esetben `k`
- A ciklusváltozó lehetséges értékei az `1:3`, azaz az `[1,2,3]` vektorban vannak felsorolva
- a törzsben egyetlen utasítás szerepel. A disp egy kiiratő (display) függvény, az argumentumában egy vektor áll (ld. szögletes zárójelek!), melynek két eleme van, ezek most sztring (szöveg) típusúak. Az első elem a ,,A ciklusvaltozo erteke most:" szöveg, a második pedig a `k` aktuális számértéke szöveggé alakítva.

Az eredmény:

```
A ciklusvaltozo erteke most:1
A ciklusvaltozo erteke most:2
A ciklusvaltozo erteke most:3
```

```
s=1;
for i=1:2:9
    s=s*i;
    disp(['s erteke:',num2str(s)]);
end
```

- Itt az `i` ciklusváltozó az 1,3,5,7,9 értékeket veszi fel (1-től 2-es lépésközzel lépünk 9-ig).
- A ciklus minden lépésében az `s` értékét szorozza az `i` aktuális értékével (`s` kezdőértéke 1 volt) és kiírja az `s` pillanatnyi értékét.

Az eredmény:

```
s erteke:1
s erteke:3
s erteke:15
s erteke:105
s erteke:945
```

←Az `s` kezdőértékét megszorozta `i=1`-gyel
←Az `s` előző értékét megszorozta `i=3`-mal
←Az `s` előző értékét megszorozta `i=5`-tel
←Az `s` előző értékét megszorozta `i=7`-tel
←Az `s` előző értékét megszorozta `i=9`-cel

1. feladat

Január elsején 50000 Ft-ot helyezünk el a bankszámlánkon, havi 0.5% kamatozású betéten. A kamatot minden hónap utolsó napján hozzáadják a tőkéhez, amit a következő napon kiegészítünk újabb 50000 Ft-tal, és újra lekötjük. Írjon egy kódot, ami megadja, hogy mennyi lesz a tőkénk a 30. hónap végén (a kamat hozzáírása után).

2. feladat

50000 Ft-ot helyezünk el a bankszámlánkon, változó havi kamatozású betéten (kamatos kamatra). Határozza meg mekkora összeget vehetünk fel a 12. hónap végén, ha a havi kamatok a `p` vektorban adottak. (pl. `p=[0.5,0.4,0.3,0.4,0.3,0.5,0.4,0.4,0.5,0.5,0.4,0.4]`)

3. feladat

Legyen $a_0 = 4$, továbbá minden $n \in \mathbb{N}$ esetén $a_n = \frac{3}{2} a_{n-1} - 1$. Írjon egy kódot, mely megadja a_{20} értékét.

Megjegyzés

A Matlab vektorizált utasításai általában gyorsabbak, mint a for-ciklusként megírt kódok.

4. feladat

Írjon egy kódot, melyben generál egy 1000000 elemű véletlen x vektort (használja a `rand` függvényt.) Számítsa ki azt az y vektort, melynek minden koordinátája 1-gyel nagyobb az x megfelelő koordinátájánál. Határozza meg az y vektort for-ciklussal is, illetve az $y = x + 1;$ parancs segítségével is, és mérje le mindkét kódrészlet futási idejét. (Használja a `tic` és `toc` függvényeket.) Vizsgálja meg azt is, hogy mit jelent futási időben, ha az első esetben inicializálja az y vektort (pl. egy, az x-szel megegyező méretű csupa 0 vektornak), illetve ha nem inicializálja. Ne feledkezzen meg a sorvégi pontosvesszőkről!!!
(Teljesen valós képet akkor kap, ha az adott nevű változók nem léteznek a workspace-ben. Ezért a kód minden futtatása előtt adja ki a `clear all` parancsot, illetve érdemes a két kódrészletben más-más változónevet használni `y` helyett.)

Kerekítés tízes számrendszerben

1. feladat

Szabályos kerekítéssel kerekítse a lenti számokat két tizedesjegyre!

$$0.36455, \quad 21.7761, \quad 13.6666, \quad 1.1251$$

Matlab-ban:

`round(x)` a legközelebbi egészre kerekíti x-et
`round(x,n)` n tizedesjegyre kerekíti x-et

Két további, kerekítésre használható függvény: `floor` és `ceil`

A kiíratás formátuma Matlab-ban

A `format` függvény segítségével szabályozhatjuk.

Alapértelmezés: 4 tizedesjegy (`format short`)

Két további lehetőség:
`format long` $\rightarrow$ 16 tizedesjegy
`format shortE` $\rightarrow$ normál alak, 4 tizedesjeggyel

Példa:

```
>> 1/7
ans =
    0.1429
>> format long; 1/7
ans =
   0.142857142857143
>> format shortE; 1/7
ans =
   1.4286e-01
```

A kiíratás formátuma Matlab-ban

Az `1.4286e-01` jelentése: $1.4286 \cdot 10^{-1}$.
Az `1.4286e+02` jelentése: $1.4286 \cdot 10^{2}$

2. feladat

Néhény számítást végeztünk Matlab-ban, a következő eredményeket kaptuk:

```
8.4781e-03,  0.3287,  5.3766e-01,
2.1283e+01,  3.3766e-01,  1.9283e+03
```

Rendezze az eredményeket nagyság szerint növekvő sorrendbe!

Kettes számrendszer

3. feladat

Írja fel a következő számok kettes számrendszerbeli alakját!

$$30, \quad 117, \quad 0.875, \quad 0.5625, \quad 0.96875,$$

$$1.625, \quad 2.75, \quad 3.125,$$

$$0.1, \quad \frac{1}{3}, \quad 1.4, \quad 3.3$$

Mi lesz a felsorolt számokhoz rendelt normalizált lebegőpontos szám, ha 4 hely áll rendelkezésre a mantissza ábrázolására, és szabályosan kerekítjük a számokat?

Gépi számítások

4. feladat

Vizsgálja meg számítógépén a $0.4 - 0.5 + 0.1 == 0$ logikai kifejezés értékét! Magyarázza meg a tapasztalt jelenséget! Mi lesz a $0.1 - 0.5 + 0.4 == 0$ logikai kifejezés értéke? Magyarázza meg a tapasztalt jelenséget!

5. feladat

Olvassa el az Octave/Matlab `eps` függvényének help-jét. Nézze meg az `eps` (azaz az `eps(1)`) értékét.

6. feladat

Vizsgálja meg számítógépén a $2^{66} + 1 == 2^{66}$, $2^{66} + 10 == 2^{66}$, $2^{66} + 100 == 2^{66}$, $2^{66} + 1000 == 2^{66}$ és $2^{66} + 10000 == 2^{66}$ logikai kifejezések értékét! Keresse meg azt a legkisebb $n > 0$ számot, melyre a $2^{66} + n == 2^{66}$ logikai kifejezés értéke hamis. Mennyi az `eps(2^66)` értéke?

7. feladat

(a) Az alábbi algoritmus végrehajtása után mennyi az x elméleti, illetve a gépi számítás után adódó értéke?

```
x=1/3;
for i=1:40
   x=4*x-1;
end
```

(b) Az alábbi algoritmus elméletileg minden $x \geq 0$ esetén az x eredeti értékét adja vissza. Vizsgálja meg mi történik a gyakorlatban, ha az algoritmust $x = 1000$, $x = 100$ kezdőértékkel futtatja!

```
for i=1:60
   x=sqrt(x);
end
for i=1:60
   x=x^2;
end
```

Mi az oka a tapasztalt jelenségeknek?

8. feladat

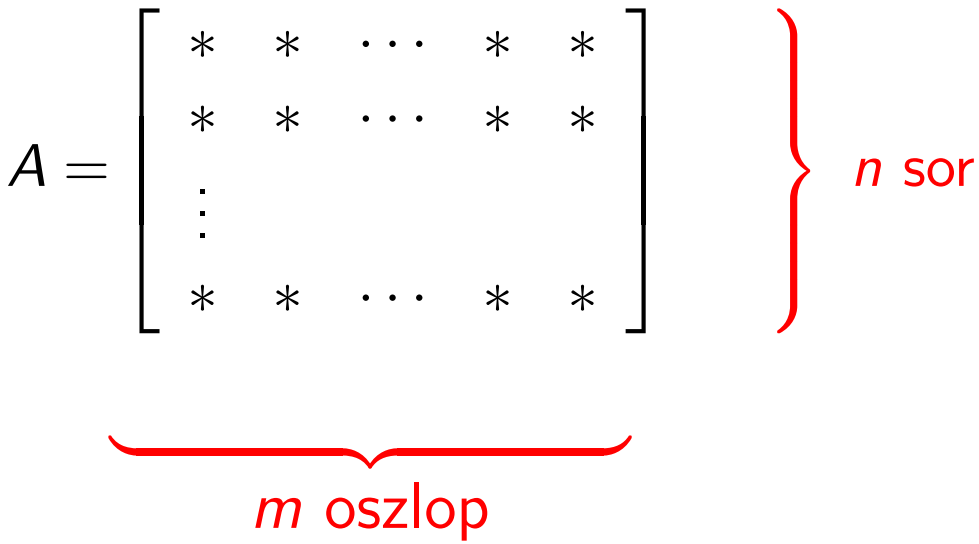
tudjuk, hogy az $y_1, \ldots, y_n$ minta esetén a korrigált tapasztalati szórásnégyzet kiszámításának két lehetséges módja:

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \overline{y})^2 \quad \text{és} \quad s_n^2 = \frac{1}{n-1} \left(\sum_{i=1}^{n} y_i^2 - n(\overline{y})^2 \right),$$

ahol $\overline{y}$ a mintaátlagot jelöli. Az $y_1 = 99.4, y_2 = 100, y_3 = 100.01$ minta esetén alkalmazza mindkét kiszámítási módot a következőképpen. Az alapértelmezett `format short` kiíratást használja, és a második esetben a zárójelben szereplő mindkét kifejezést külön számolja ki, írja fel füzetébe a kapott eredményt, majd azokkal végezze el a maradék műveleteket. Mit tapasztal? Melyik eredmény áll közelebb a valódi értékhez? (A Matlab-ban a korrigált tapasztalati szórásnégyzet a `var`, a mintaátlag a `mean` függvénnyel számolható.)

Mátrixok

Mátrix = egy kétdimenziós tömb:



Ekkor A egy $n \times m$-es mátrix.

A mátrix elemeinek számozása a bal felső sarokban kezdődik, $(1, 1)$-gyel.

`A(i,j)`: az i-edik sor és a j-edik oszlop metszetében lévő elem.

Mátrix létrehozása elemeinek felsorolásával

Az

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

mátrix létrehozása:

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

vagy

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

Az egy sorban álló elemeket vesszővel vagy szóközzel, a sorokat pontosvesszővel választjuk el.

Mátrixok létrehozása vektorok összefűzésével

Ha

$$a = \begin{bmatrix} 1 & -2 & 0 \end{bmatrix}, b = \begin{bmatrix} 2 & -11 & 7 \end{bmatrix},$$

azaz Matlab-ban:

```
a=[1,-2,0]; b=[2,-11,7];
```

akkor `B=[a;b]` eredménye:

$$B = \begin{bmatrix} 1 & -2 & 0 \\ 2 & -11 & 7 \end{bmatrix}$$

Emlékeztető: a szögletes zárójelen belül a pontosvessző eredménye a sortörés.

Ha a két vektor mérete olyan, hogy nem helyezhetők egymás alá (nem ugyanannyi oszlopuk van), akkor hibaüzenetet kapunk.

Mátrixok létrehozása vektorok összefűzésével

Ha

$$m = \begin{bmatrix} -3 \\ 0 \\ 7 \end{bmatrix}, n = \begin{bmatrix} 1 \\ -2 \\ 0 \end{bmatrix},$$

azaz Matlab-ban:

```
m=[-3;0;7]; n=[1;-2;0];
```

akkor `C=[a'  b']` és `D=[m  n]` eredménye:

$$C = \begin{bmatrix} 1 & 2 \\ -2 & -11 \\ 0 & 7 \end{bmatrix} \qquad D = \begin{bmatrix} -3 & 1 \\ 0 & -2 \\ 7 & 0 \end{bmatrix}$$

Emlékeztető: Ha a szögletes zárójelen belül az új elemet szóközzel, vagy vesszővel elválasztva írjuk az utolsó elem után, akkor az új elem az utolsó elem mellé kerül.

Mátrixok bővítése

Az előbb létrehozott mátrixokkal, vektorokkal, azaz ha

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, a = \begin{bmatrix} 1 & -2 & 0 \end{bmatrix},$$

akkor `E=[A;a]` vagy `E=[A;[1,-2,0]]` eredménye

$$E = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & -2 & 0 \end{bmatrix}$$

Tehát: *[mátrix „sortörés” (azaz ;) sorvektor]*

Mátrixok bővítése

Ha

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, m = \begin{bmatrix} -3 \\ 0 \\ 7 \end{bmatrix},$$

akkor az `F=[A m]` vagy `F=[A, m]` eredménye

$$F = \left[\begin{array}{ccc|c} 1 & 2 & 3 & -3 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 7 \end{array}\right]$$

Tehát: *[mátrix szóköz vagy vessző oszlopvektor]*

Mátrixok bővítése

Ha

$$C = \begin{bmatrix} 1 & 2 \\ -2 & -11 \\ 0 & 7 \end{bmatrix} \quad \text{és} \quad D = \begin{bmatrix} -3 & 1 \\ 0 & -2 \\ 7 & 0 \end{bmatrix}$$

akkor `G=[C D]` és `H=[C;D]` eredménye

$$G = \left[\begin{array}{cc|cc} 1 & 2 & -3 & 1 \\ -2 & -11 & 0 & -2 \\ 0 & 7 & 7 & 0 \end{array}\right] \qquad H = \left[\begin{array}{cc} 1 & 2 \\ -2 & -11 \\ 0 & 7 \\ \hline -3 & 1 \\ 0 & -2 \\ 7 & 0 \end{array}\right]$$

Az ones és zeros függvények

- `ones(n,m)` létrehoz egy $n \times m$-es csupa 1-esből álló mátrixot.
  Pl. `ones(3,4)` eredménye

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- `zeros(n,m)` létrehoz egy $n \times m$-es csupa 0-ból álló mátrixot.
  Pl. `zeros(3,2)` eredménye

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

size

`size(A)` egy kételemű sorvektorral tér vissza; az `A` sorainak és oszlopainak számával.

- ha `A` sorvektor, akkor a két visszaadott érték közül az első 1,
- ha `A` oszlopvektor, akkor a két visszaadott érték közül a második 1.

Példák.

1. Adott `A` mátrix esetén készítsük el azt a `B` mátrixot, melynek ugyanaz a mérete, mint `A`-nak, de minden eleme 1.

```
>>B=ones(size(A))
```

2. Adott `A` mátrix esetén készítsük el azt a `b` vektort, melynek ugyanaz a mérete, mint `A` egy sorának, de minden eleme 1.

```
>> [m,n]=size(A);
>> b=ones(1,n)
```

Hivatkozás elemekre, sorokra, oszlopokra, részmátrixokra

A mátrix elemeinek számozása a bal felső sarokban kezdődik, $(1,1)$-gyel.

`A(i,j)`: az i-edik sor és a j-edik oszlop metszetében lévő elem. Ha

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix},$$

akkor pl. `A(2,3)` értéke 7.

`A(i,:)`: egy sorvektor, az A mátrix i-edik sora
`A(:,j)`: egy oszlopvektor, az A mátrix j-edik oszlopa.

Pl. `A(2,:)` értéke `[5 6 7 8]` és `A(:,3)` értéke

$$\begin{bmatrix} 3 \\ 7 \\ 11 \end{bmatrix}$$

Hivatkozás elemekre, sorokra, oszlopokra, részmátrixokra

Több sorra és oszlopra is hivatkozhatunk egyszerre:

- `A(2:3,:)` az A mátrix 2. és 3. sora
- `A([1 3],:)` az A mátrix 1. és 3. sora.

Az előző A mátrixszal:

$$\texttt{A(2:3,:)} = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}, \quad \texttt{A([1,3],:)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- `A(:,[1 3])` az A mátrix 1. és 3. oszlopa
- `A(:,[1 3 4])` az A mátrix 1., 3. és 4. oszlopa

$$\texttt{A(:,[1 3])} = \begin{bmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 11 \end{bmatrix}, \quad \texttt{A(:,[1 3 4])} = \begin{bmatrix} 1 & 3 & 4 \\ 5 & 7 & 8 \\ 9 & 11 & 12 \end{bmatrix}$$

Hivatkozás elemekre, sorokra, oszlopokra, részmátrixokra

Hivatkozhatunk adott sorok és oszlopok metszeteként előálló részmátrixokra:

- `A(2:3,[1 3])` az A mátrix 2. és 3. sorának és 1. és 3. oszlopának metszetéből álló mátrix

Az előbb definiált A mátrix esetén

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix},$$

azaz

$$\texttt{A(2:3,[1 3])} = \begin{bmatrix} 5 & 7 \\ 9 & 11 \end{bmatrix}$$

Mátrixok módosítása

Az előző hivatkozások segítségével felülírhatjuk, elhagyhatjuk a mátrix egyes részeit.

Pl.: `A(2,3)=-1` kicseréli a mátrix (2,3) elemét -1-re.

Vigyázzunk! A `A(2,6)=-1` parancs eredménye:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 0 & 0 \\ 5 & 6 & 7 & 8 & 0 & -1 \\ 9 & 10 & 11 & 12 & 0 & 0 \end{bmatrix}$$

Megváltozik a mátrix mérete, erre nem figyelmeztet!

(Elkészítette a legkisebb olyan mátrixot, melynek része A és amelyben van értelme a fenti értékadásnak. A nemdefiniált elemeket 0-val töltötte fel).

Mátrixok módosítása

Teljes sorokat, oszlopokat is módosíthatunk egyszerre. Az eredeti A mátrixszal (a módosított elemeket pirossal jelölve)

- az `A(:,1)=[-1;-2;-3]` parancs eredménye:

$$A = \begin{bmatrix} -1 & 2 & 3 & 4 \\ -2 & 6 & 7 & 8 \\ -3 & 10 & 11 & 12 \end{bmatrix}$$

Az értékadó utasítás jobb oldalán ugyanolyan típusú vektor áll, mint a módosítandó rész (Ebben az esetben egy 3 elemű oszlopvektor.)

- az `A(:,1)=-1` parancs eredménye:

$$A = \begin{bmatrix} -1 & 2 & 3 & 4 \\ -1 & 6 & 7 & 8 \\ -1 & 10 & 11 & 12 \end{bmatrix}$$

Az értékadó utasítás jobb oldalán egy szám áll. Minden hivatkozott elemet erre cserél.

Mátrixok módosítása

Sorok, oszlopok elhagyása mátrixokból

- `A(i,:)=[]` az i-edik sor elhagyása
- `A(:,j)=[]` a j-edik oszlop elhagyása
- `A([1 3],:)=[]` az 1. és 3. sor elhagyása
- `A(:,[1 3])=[]` az 1. és 3. oszlop elhagyása

Sor- és oszlopcsere

Az i-edik és j-edik sor illeve oszlop cseréje:
`A([i,j],:)=A([j,i],:)`, ill. `A(:,[i,j])=A(:,[j,i])`

Mátrixból vektor

`A(:)` az A mátrix elemei oszlopfolytonosan felsorolva

Aritmetikai műveletek mátrixok között

Ha A és B két azonos méretű mátrix, c pedig egy szám, akkor

- `A+c` a mátrix minden eleméhez hozzáad c-t
- `c*A` a mátrix minden elemét megszorozza c-vel
- `A+B` a két mátrix elemenkénti összege
- `A-B` a két mátrix elemenkénti különbsége
- `A.*B` a két mátrix elemenkénti szorzata
- `A./B` a két mátrix elemenkénti hányadosa
- `A.^2` a mátrix minden elemét négyzetre emeli

Az utolsó három esetben figyeljünk a műveleti jel előtti pontra! Ennek hiányában a *, a / és a ^ három teljesen más műveletet jelent. (Ld. később, a lineáris algebrai résznél.)

Aritmetikai műveletek mátrixok között

Ha A és B nem azonos méretű (és egyik sem szám), akkor a fenti utasítások hibaüzenetet adnak, kivéve, ha az egyik mátrix mérete megegyezik a másik mátrix egy sorának, vagy oszlopának méretével. Pl. ha

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}, B = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}, C = [0, 1, 2, 3]$$

akkor

$$\texttt{A+B} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 3 & 4 & 5 & 6 \\ 6 & 7 & 8 & 9 \end{bmatrix}, \texttt{A+C} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 5 & 7 & 9 & 11 \\ 9 & 11 & 13 & 15 \end{bmatrix},$$

tehát az első esetben A minden oszlopához hozzáadta a B oszlopvektort, a második esetben A minden sorához hozzáadta a C sorvektort.

Néhány hasznos függvény: `numel`, `size`, `sum`, `prod`

- `numel(A)`
  az A elemeinek száma
- `size(A)`
  az A mérete
- `sum(A)` vagy `sum(A,1)`
  egy sorvektorral tér vissza: az A minden oszlopában összeadja az ott álló elemeket.
- `sum(A,2)`
  egy oszlopvektorral tér vissza: az A minden sorában összeadja az ott álló elemeket.
- `sum(A,'all')`
  összeadja az A minden elemét
- `prod`
  szorzatot számol, hívása a `sum` függvényhez hasonló

Néhány hasznos függvény: `max`, `min`

- `max(A)`, vagy `max(A,[],1)`
  egy sorvektorral tér vissza: az A mátrix minden oszlopában veszi az elemek maximumát
- `max(A,[],2)`
  egy oszlopvektorral tér vissza: az A mátrix minden sorában veszi az elemek maximumát
- `max(A,[],'all')`
  egy számmal tér vissza: az A mátrix elemeinek maximumával
- `max(A,B)`
  ahol A és B két azonos méretű mátrix; elemenként veszi a két mátrix maximumát
- `max(A,c)`
  ahol A egy mátrix, c egy skalár; egy mátrixszal tér vissza, elemenként veszi az A és a c maximumát

A `min` függvény ugyanígy, minimumot számol.

Lineáris algebra, mátrixműveletek

- Mátrix szorzása skalárral: `c*A`
  Ha `c` egy skalár, `A` egy mátrix, akkor `c*A`: a mátrix minden elemét megszorozza a skalárral.
- Mátrixok összege: `A+B`
  Ha `A` és `B` azonos méretű mátrixok, akkor `A+B` a két mátrix elemenkénti összege
- Mátrixok szorzata: `A*B`
  Ha az `A` mátrix oszlopainak száma megegyezik a `B` mátrix sorainak számával, akkor `A*B` a két mátrix szorzata (azaz az eredménymátrix ij-edik eleme az `A` mátrix i-edik sorának és a `B` mátrix j-edik oszlopának skaláris szorzata).
- Mátrixok elemenkénti szorzata: `A.*B`
  Ha `A` és `B` azonos méretű mátrixok, akkor `A.*B` a két mátrix elemenkénti szorzata

Lineáris algebra, mátrixműveletek

Példa. (Mátrix szorzása skalárral, mátrixok összege)

$$A = \begin{bmatrix} -1 & 2 & -2 \\ 3 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 5 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

```
>> A=[-1,2,-2;3,0,1];B=[3,5,-1;2,-1,2];
>> 2*A
ans =
    -2     4    -4
     6     0     2

>> A+B
ans =
     2     7    -3
     5    -1     3
```

Lineáris algebra, mátrixműveletek

Példa. (Mátrixok elemenkénti szorzata és szorzata)

$$A = \begin{bmatrix} -1 & 2 & -2 \\ 3 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 5 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

```
>> A.*B
ans =
    -3    10     2
     6     0     2
>> A*B
Error using  *
Incorrect dimensions for matrix multiplication. Check that
the number of columns in the first matrix matches the number
of rows in the second matrix. To perform elementwise
multiplication, use '.*'.
```

A két mátrix elemenkénti szorzata számolható, a szorzatuk nem.

Lineáris algebra, mátrixműveletek

Példa. (Mátrixok elemenkénti szorzata és szorzata)

$$A = \begin{bmatrix} -1 & 2 \\ 3 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 5 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

```
>> A=[-1,2;3,0];B=[3,5,-1;2,-1,2];
>> A.*B
Matrix dimensions must agree.

>> A*B
ans =
     1    -7     5
     9    15    -3
```

A két mátrix elemenkénti szorzata nem számolható, a szorzatuk számolható.

Lineáris algebra, mátrixműveletek

Példa. (Mátrixok elemenkénti szorzata és szorzata)

$$A = \begin{bmatrix} -1 & 2 \\ 3 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 3 & 5 \\ 2 & -1 \end{bmatrix}$$

```
>>  A=[-1,2;3,0];B=[3,5;2,-1];
>> A.*B
ans =
    -3    10
     6     0
>> A*B
ans =
     1    -7
     9    15
```

Mindkét szorzat számolható, de az eredmény más. Figyeljünk rá, hogy melyik műveletet használjuk!

Néhány hasznos függvény

- `det(A)`
  Az `A` mátrix determinánsa
- `inv(A)`
  Az `A` mátrix inverze
- `rank(A)`
  Az `A` mátrix rangja
- `diag(A)`
  Ha `A` egy mátrix, akkor `diag(A)` egy vektor, az `A` főátlója.
  Ha `A` egy vektor, akkor `diag(A)` az a mátrix, melynek főátlója `A`.
- `tril(A)`
  Az `A` mátrix alsóháromszög része.
- `triu(A)`
  Az `A` mátrix felsőháromszög része.

Néhány hasznos függvény

- `dot(a,b)`
  az `a` és `b` vektorok skaláris (belső-) szorzata
- `transpose(A)`
  az `A` mátrix transzponáltja
- `norm(A)` vagy `norm(A,2)`
  Az `A` mátrix (vagy vektor) 2-normája
- `norm(A,1)`
  Az `A` mátrix (vagy vektor) 1-normája
- `norm(A,inf)`
  Az `A` mátrix (vagy vektor) ∞-normája

Lineáris egyenletrendszerek

Lineáris egyenletrendszerek megoldása Matlab-bal

Példa

Oldjuk meg az $Ax = b$ lineáris egyenletrendszert, ha

$$A = \begin{bmatrix} -2 & -1 & 4 \\ 2 & 3 & -1 \\ -4 & -10 & -5 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 1 \\ -12 \end{bmatrix}$$

Megoldás. Használjuk a `backslash` operátort!

```
>>A=[-2 -1 4; 2 3 -1; -4 -10 -5];
>>b=[3; 1; -12];
>>x=A\b
x=
     3
    -1
     2
```

Ügyeljünk rá, hogy a b oszlopvektorként legyen megadva!

Ha az egyenletrendszer kibővített mátrixával meghívjuk az `rref` függvényt:

```
>>rref([A b])
ans=
   1  0  0   3
   0  1  0  -1
   0  0  1   2
```

akkor láthatjuk, hogy a Gauss-Jordan elimináció eredményeként valóban így állítható elő a b vektor az A oszlopvektoraiból, amelyek lineárisan függetlenek, tehát a megoldás egyértelmű.

Példa

Oldjuk meg az $Ax = b$ lineáris egyenletrendszert, ha

$$A = \begin{bmatrix} -4 & -4 & 2 \\ -2 & -7 & 3 \\ 2 & 12 & -5 \end{bmatrix}, \quad b = \begin{bmatrix} -2 \\ 6 \\ -13 \end{bmatrix}$$

Megoldás. Próbálkozzunk ismét a `backslash` operátorral!

```
>>A=[-4 -4 2; -2 -7 3; 2 12 -5];
>>b=[-2; 6; -13];
>>x=A\b
warning: matrix singular to machine precision
x =

   1.93162
  -1.27350
   0.31624
```

A Matlab arra figyelmeztetett, hogy a mátrix szinguláris (valóban, $\det(A) = 0$), de ellenőrizhetjük, hogy `Ax-b` kerekítési hiba nagyságrendű, azaz x-et tekinthetjük megoldásnak.

Próbálkozzunk az `rref` függvénnyel!

```
>>rref([A b])
ans=
   1.0000         0 -0.1000   1.9000
        0  1.0000 -0.4000  -1.4000
        0         0         0         0
```

Azt látjuk, hogy a mátrix oszlopvektorai lineárisan függőek, de a b vektor benne van az oszlopvektorok által felfeszített térben. Tudjuk, hogy ilyenkor az egyenletrendszernek végtelen sok megoldása van, ezek közül egy:

$$x = \begin{bmatrix} 1.9 \\ -1.4 \\ 0 \end{bmatrix}$$

Ha az egyenletrenszer összes megoldását szeretnénk tudni, akkor használjuk a `null` függvényt, amely előállítja a nulltér egy bázisát:

```
>>p=null(A)
p=
   -0.092450
   -0.369800
   -0.924500
```

Ezek szerint a lineáris egyenletrendszer általános megoldása:

$$\begin{bmatrix} 1.9 \\ -1.4 \\ 0 \end{bmatrix} + \lambda p$$

ahol $\lambda \in \mathbb{R}$. (A kapott x megoldás a $\lambda = -0.34207$ konstanshoz tartozik.)

Példa

Oldjuk meg az $Ax = b$ lineáris egyenletrendszert, ahol

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 6 \\ 9 \\ 12 \end{bmatrix}$$

Megoldás. A `backslash` operátorral azt kapjuk, hogy

```
>>x=A\b
x=
   1.0000
   2.7000
```

Könnyen látható, hogy ez **nem megoldása** az egyenletrendszernek.

Az `rref` függvénnyel:

```
>>rref([A b])
ans=
    1  0  0
    0  1  0
    0  0  1
    0  0  0
```

láthatjuk, hogy a harmadik egyenlet jelentése: $0 \cdot x_1 + 0 \cdot x_2 = 1$, azaz az egyenletrendszer **ellentmondásos**.

Ellentmondásos lineáris egyenletrendszerek esetén a backslash operátor egy olyan x vektort ad vissza, melyre az Ax és b vektorok eltérése euklideszi normában a legkisebb (azaz $\|Ax - b\|_2$ minimális). Ilyenkor azt mondjuk, hogy x az egyenletrendszer legkisebb négyzetes értelemben vett megoldása.

1. feladat

Hány egyenletből áll és hány ismeretlent tartalmaz az $Ax = b$ lineáris egyenletrendszer az alábbi esetekben? Oldja meg Matlab-bal az egyenletrendszereket.

(a)

$$A = \begin{bmatrix} 2 & -3 & 1 & 1 \\ -1 & 3 & 4 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$

(b)

$$A = \begin{bmatrix} 2 & 1 \\ -3 & 4 \\ 5 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} -5 \\ 24 \\ -23 \end{bmatrix}$$

(c)

$$A = \begin{bmatrix} 2 & 1 & 5 & 0 \\ -3 & 4 & -13 & 22 \\ 5 & -1 & 16 & -14 \\ 1 & 1 & 2 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 12 \\ 81 \\ -33 \\ 15 \end{bmatrix}$$

2. feladat

A következő feladatot Matlab-bal oldja meg.
Egy műkereskedő három festményt vásárolt. Ha az első és a harmadik festményt 20%-kal, a másodikat pedig 30%-kal a beszerzési ár fölött sikerül eladnia, akkor a bevétele 1.9 millió Ft lesz. Ha a három festményt rendre 40%-kal, 20%-kal és 10%-kal drágábban tudja eladni, mint amennyiért vette, akkor 1.885 millió Ft bevétele lesz. Viszont ha csak 5%-kal, 25%-kal és 20%-kal többért tudja eladni a festményeket, mint amennyiért vette őket, akkor 1.805 millió Ft-ot kap értük. Mennyiért vásárolta az egyes képeket?

3. feladat

Tekintsük a következő mikrogazdasági modellt: földművesek, állattanyésztők és bányászok egy-egy csoportja rendre gabonát, húst és szenet "állít elő". Az előállított termék egy részét minden csoport maga használja fel, egy részét a többiek veszik igénybe, egy részét pedig külső piacon értékesíti. Az alábbi táblázatban látható, hogy az egyes csoportoknak egy egységnyi áru előállításához hány egységnyire van szükségük a többi nyersanyagból, illetve hány egység a külső igény. Matlab-bal határozza meg az előállított termékek mennyiségét úgy, hogy minden igény ki legyen elégítve, és ne keletkezzen felesleg.

	földműves	állattenyésztő	bányász	külső igény
növény	0.1	0.7	0.1	2
hús	0.2	0.1	0.3	3
szén	0.4	0.3	0.1	5

Több jobboldali vektor

Példa

Oldjuk meg az $Ax = b$ és $Ax = c$ egyenletrendszereket, ha

$$A = \begin{bmatrix} -2 & -1 & 4 \\ 2 & 3 & -1 \\ -4 & -10 & -5 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 1 \\ -12 \end{bmatrix}, \quad c = \begin{bmatrix} 17 \\ 1 \\ -42 \end{bmatrix}$$

Megoldás. Mivel a két rendszer mátrixa azonos, ezért megoldhatjuk őket egyszerre.

```
>>A=[-2 -1 4; 2 3 -1; -4 -10 -5];
>>b=[3; 1; -12]; c=[17; 1; -42];
>>x=A\[b c]
x=
     3    -2
    -1     3
     2     4
```

Több jobboldali vektor

Nagyméretű mátrixok esetén a futási időt jelentősen befolyásolhatja, hogy az azonos mátrixszal adott rendszereket egyszerre, vagy külön-külön oldjuk meg:

```
>> A=rand(10000);
>> b=ones(10000,1);
>> c=zeros(10000,1);
>> tic;x=A\[b,c];toc
Elapsed time is 6.116513 seconds.
>> tic;x=A\b; x2=A\c; toc
Elapsed time is 11.571959 seconds.
```

(A fenti eredmény egy Intel Core i5-4590 processzorral, 7.7 GiB memóriával rendelkező gépen született).

Mátrix inverze Matlab-bal

Az `inv` függvénnyel számítható. Ha a mátrix nem négyzetes, vagy a determinánsa 0 (vagy 0-hoz közeli), akkor hibaüzenetet, illetve figyelmeztetést kapunk.

Nagyméretű mátrixok inverzének kiszámítása túl költséges lehet. Csak akkor számoljuk ki, ha ténylegesen szükségünk van az inverzre.

Pl. az $Ax = b$ négyzetes mátrixú lineáris egyenletrendszer megoldása $x = A^{-1}b$ módon kb háromszor annyi műveletbe kerül, mint az $x = A\backslash b$ megoldás.

Gyengén meghatározott lineáris egyenletrendszerek

4. feladat

Oldja meg Matlab-bal az $Ax = b$ lineáris egyenletrendszert, ha

$$A = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 0.98 \end{bmatrix}, \qquad b = \begin{bmatrix} 1.99 \\ 1.97 \end{bmatrix}. \text{ ill. } b = \begin{bmatrix} 1.98 \\ 1.98 \end{bmatrix}.$$

Hasonlítsa össze a relatív eltérést a jobboldali vektorok között, illetve a megoldásvektorok között. Számítsa ki a mátrix kondíciószámát!

5. feladat

Tegyük fel, hogy az $Ax = b$ egyenletrendszert akarjuk megoldani, ahol a b vektor esetlegesen hibával terhelt. Legfeljebb mekkora lehet a megoldás relatív hibája (∞-normában), ha tudjuk, hogz a b vektor relatív hibája legfeljebb 0.01 (∞-normában)?

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

Gyengén meghatározott lineáris egyenletrendszerek

6. feladat

Matlab-bal oldjuk meg a következő 100×100-as lineáris egyenletrendszert:

$$\begin{bmatrix} 1 & -1 & -1 & -1 & \cdots & -1 & -1 \\ 0 & 1 & -1 & -1 & \cdots & -1 & -1 \\ 0 & 0 & 1 & -1 & \cdots & -1 & -1 \\ \vdots & & & & & & \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{99} \\ x_{100} \end{bmatrix} = \begin{bmatrix} -98 \\ -97 \\ -96 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

A mátrix előállításához használhatjuk a ones, triu és eye függvényeket.
Ezután perturbáljuk egy kicsit a jobb oldalt: legyen `b(100)=1.00001`.
Oldjuk meg újra a rendszert! Számítsuk ki a mátrix kondíciószámát!

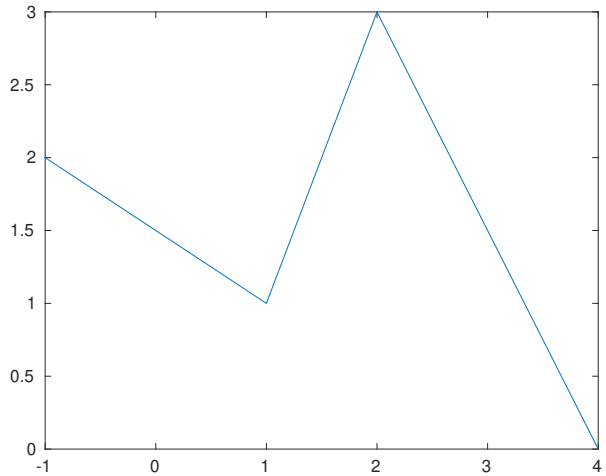
Pontok, vonalak a síkon: `plot`

`plot(x,y)`
Töröttvonallal összekötve ábrázolja azokat a síkbeli pontokat, melyek 1. koordinátája az `x`, 2. koordinátája az `y` vektorban van felsorolva.

Rajzoltassuk ki a $(-1, 2)$, $(1, 1)$, $(2, 3)$, $(4, 0)$ pontokat összekötő töröttvonalat!

1. készítsünk egy vektort a pontok első koordinátáiból: `x=[-1,1,2,4]`
2. készítsünk egy vektort a második koordinátákból: `y=[2,1,3,0]`
3. hívjuk meg a `plot` függvényt.

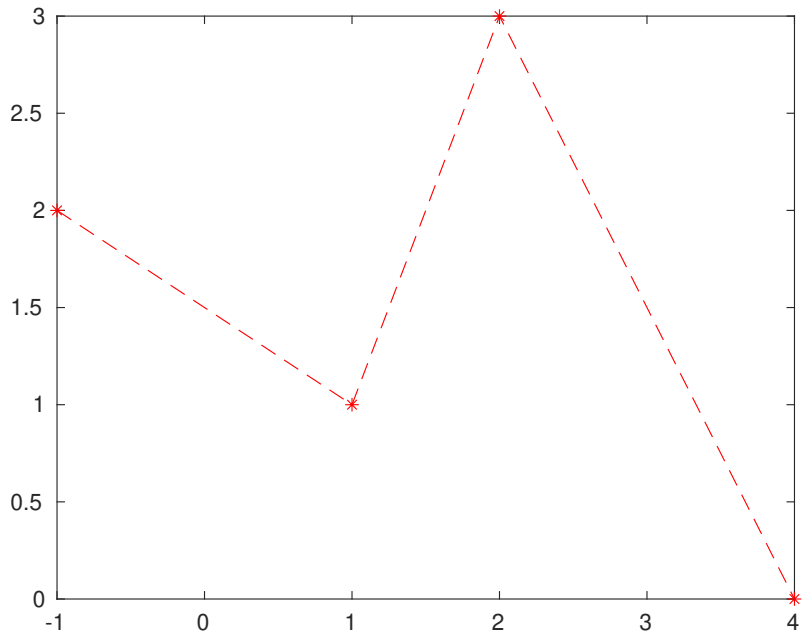
```
>> x=[-1,1,2,4];y=[2,1,3,0];
>> plot(x,y)
```



Pontok, vonalak a síkon: `plot`

A `plot` függvény harmadik argumentumában megadhatjuk az összekötő vonal és a pontokat jelző marker típusát, színét. Pl.

```
>> x=[-1,1,2,4];y=[2,1,3,0];
>> plot(x,y,'r*--')
```



Ha csak markert adunk meg, vonaltípust nem, akkor nem köti össze a pontokat.

Markerek

- * csillag
- o kör
- + összeadás jel
- x kereszt
- s négyzet
- d rombusz
- p ötszög
- h hatszög
- < balra mutató háromszög
- > jobbra mutató háromszög
- ∧ felfele mutató háromszög
- ∨ lefele mutató háromszög

Vonaltípusok

- - folyamatos vonal (alapértelmezés)
- : pontozott vonal
- - - szaggatott vonal
- -. szaggatott-pontozott vonal

Színek

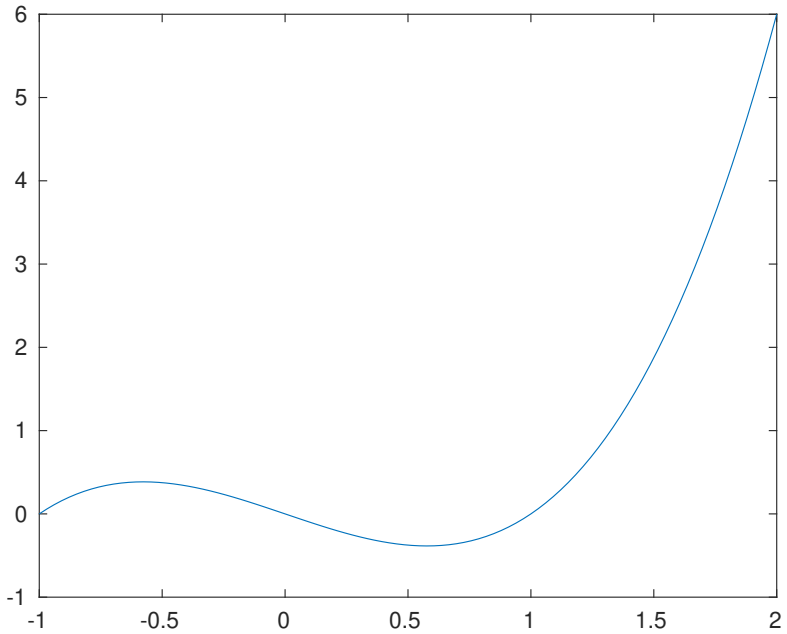
- b kék
- r piros
- g zöld
- k fekete
- w fehér
- y sárga
- m magenta
- c cián

Pontok, vonalak a síkon: `plot`

Ha egy függvényt szeretnénk ábrázolni, akkor számítsuk ki az értékét sok pontban, és a pontpárokat ábrázoljuk.
Pl. az $f(x) = x^3 - x$ függvény a $[-1, 2]$ intervallumon:

```
x=linspace(-1,2);
y=x.^3-x;
plot(x,y)
```

`x`: 100 egyenlő lépésközű pont a $[-1, 2]$ intervallumból
`y`: a függvényértékek az `x`-beli pontokban. Elemenkénti műveletek!



Anoním függvények, function handle

Függvényeket definiálhatunk a következő módon:

```
>> fv=@(x) x.^3-x;
```

Ilyen módon az $fv(x) = x^3 - x$ függvényt definiáltuk, hívása pl.:

```
>> fv(2)
ans =
     6
```

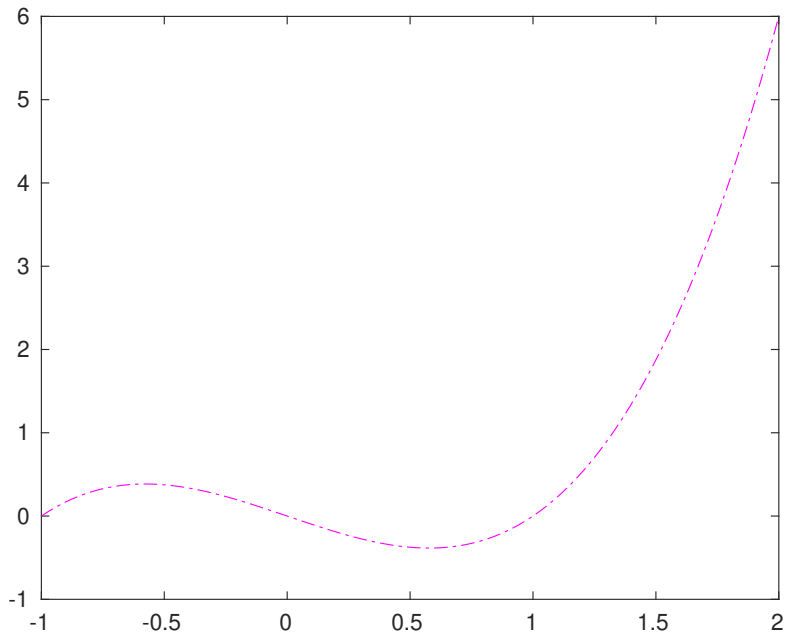
A @ szimbólum után zárójelben szerepelnek a függvény változói (most x), ezt követi a függvény (ez egy ú.n. anoním függvény). Az = baloldalán szereplő változó (most fv) egy ú.n. ,,function handle" típusú változó lesz.

Mivel a függvényt ,,elemenkénti műveletekkel" definiáltuk, ezért akár egy vektorral is hívhatjuk. Ebben az esetben a függvényt a vektor minden elemére kiértékeli, és a függvényértékek vektorát adja vissza.

Pontok, vonalak a síkon: `plot`

Függvény ábrázolásánál is megadhatjuk a vonaltípust és színt is, illetve a függvényt definiálhatjuk function handle-ként:

```
x=linspace(-1,2);
f=@(x) x.^3-x;
plot(x,f(x),'m-.')
```



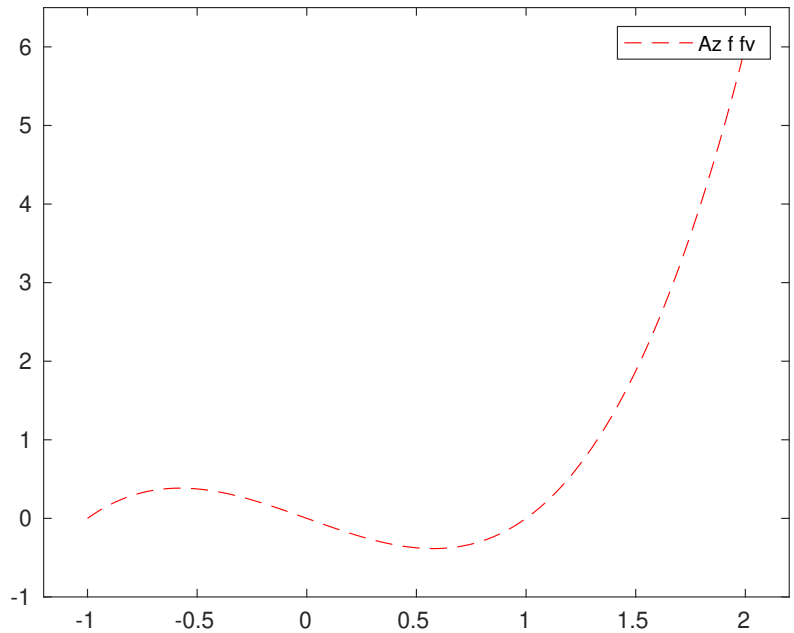
Pontok, vonalak a síkon: figure, axis, legend

A figure utasítás hatására egy új grafikus ablak nyílik. Ennek hiányában, ha van megnyitott grafikus ablak, akkor abba készíti el az ábrát, annak korábbi tartalmát felülírva.

axis([xmin,xmax,ymin,ymax]) beállítja a tengelyek határait (ld. még az xlim és ylim függvényeket).

A legend segítségével magyarázó felíratot készíthetünk.

```
x=linspace(-1,2);
f=@(x) x.^3-x;
figure; plot(x,f(x),'r--')
axis([-1.2,2.2,-1,6.5]);
legend('Az f fv')
```



Tengelyek tulajdonságai: gca

gca: (get current axes) az aktuális ábrán a tengelyek jellemzőit kérdezhetjük le, illetve mósosíthatjuk ezeket.

PI az utolsó ábra esetén:

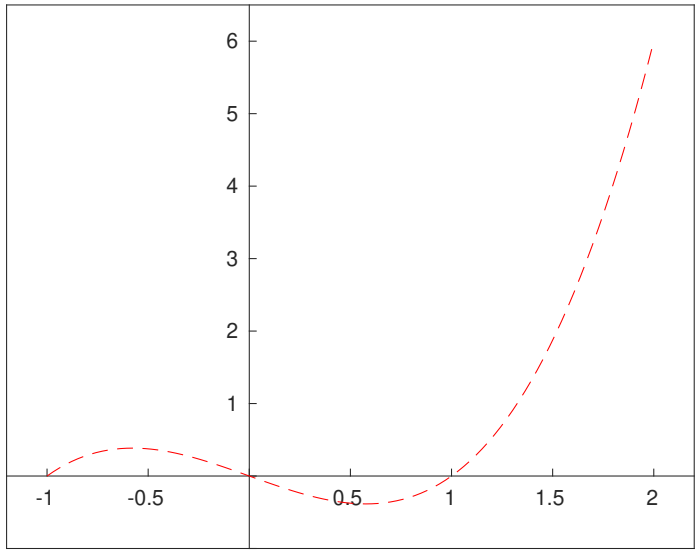
```
>> gca
ans =
  Axes with properties:
              XLim: [-1.2000 2.2000]
              YLim: [-1 6.5000]
            XScale: 'linear'
            YScale: 'linear'
    GridLineStyle: '-'
          Position: [0.1300 0.1100 0.7750 0.8150]
             Units: 'normalized'

  Show all properties
```

Csak akkor működik, ha van nyitott grafikus ablak!

Tengelyek tulajdonságai: gca

Pl. tengelyek pozícionálása (az origóba):

```
x=linspace(-1,2);
f=@(x) x.^3-x;
figure; plot(x,f(x),'r--')
axis([-1.2,2.2,-1,6.5]);
ax=gca;
ax.XAxisLocation='origin';
ax.YAxisLocation='origin';
```



A kód utolsó 3 sora:

ax=gca; *az ax változóba menti a jelenlegi tengely-jellemzőket*

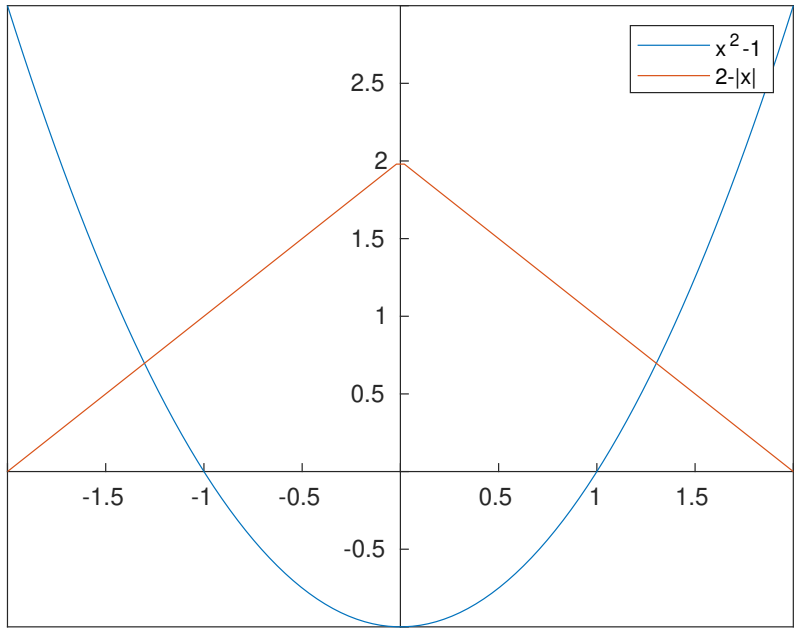
ax.XAxisLocation='origin'; *az x-tengely az origón haladjon át*

ax.YAxisLocation='origin'; *az y-tengely az origón haladjon át*

Több függvény egy ábrán

Ha egyszer hívjuk a `plot` függvényt, több argumentummal:

```
x=linspace(-2,2);
y=x.^2-1;
z=2-abs(x);
figure; plot(x,y,x,z)
legend('x^2-1','2-|x|')
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
```



`plot(x,y,x,z)`: ábrázolja az `(x,y)` és `(x,z)` párokat. Akár még több párt is megadhatunk így. Minden pár után külön megadhatjuk a színt és a vonaltípust. Pl. `plot(x,y,'r--',x,z,'m-.')`

legend: több sztringet is felsorolhatunk, ekkor a rajzolás sorrendjében rendeli hozzá a vonalakhoz a sztringeket.

Több függvény egy ábrán

Az előző ábra a `plot` függvény többszöri hívásával:

```
x=linspace(-2,2);
y=x.^2-1;
figure; plot(x,y)
z=2-abs(x);
hold on;
plot(x,z)
legend('x^2-1','2-|x|')
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
hold off;
```

`hold on` bekapcsolja a ,,rárajzoló" üzemmódot: az aktuális grafikus ablakba rajzol, az ottani eredeti ábra meghagyásával

Több függvény egy ábrán

Az előző kód részletezve:

`x=linspace(-2,2);`	*felveszünk 100 pontot a $[-2,2]$ intervallumban*
`y=x.^2-1;`	*kiszámítjuk x^2-1 értékét az x-beli pontokban*
`figure; plot(x,y)`	*nyitunk egy grafikus ablakot, ábrázoljuk az (x,y) párokat*
`z=2-abs(x);`	*kiszámítjuk $2-\|x\|$ értékét az x-beli pontokban*
`hold on;`	*bekapcsolja a rárajzoló módot*
`plot(x,z)`	*ábrázoljuk az (x,z) párokat*
`legend('x^2-1','2-|x|')`	*magyarázó szövegdoboz*

`ax=gca;`
`ax.XAxisLocation = 'origin';`
`ax.YAxisLocation = 'origin';`
	a tengelyek az origón haladjanak át

`hold off;`	*kikapcsolja a rárajzoló módot*

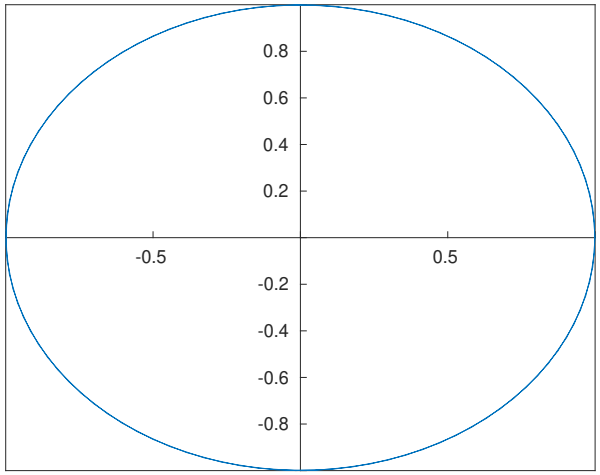
axis equal

A tengelyeken egyforma hosszú egységet választ.

Példa: rajzoljunk egy kört!

Miközben α befutja a $[0, 2\pi]$ intervallumot a $(\cos\alpha, \sin\alpha)$ pontpárok az origó középpontú egység sugarú körön futnak végig:

```
alfa=linspace(0,2*pi);
figure;
plot(cos(alfa),sin(alfa))
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
```

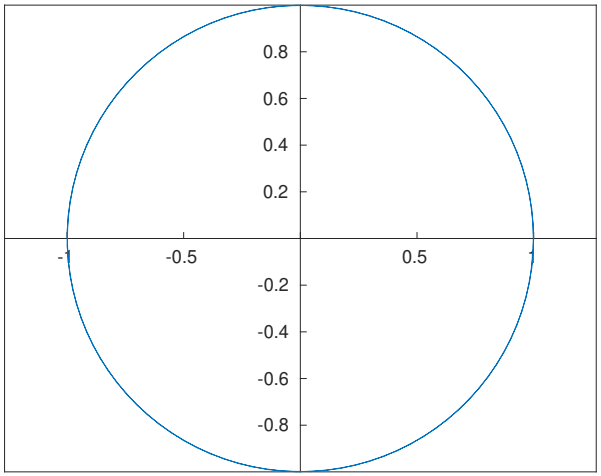


Mivel nem egyforma hosszú az egység a két tengelyen, így az alakzat nem látszik körnek.

axis equal

Adjunk hozzá az előző kódhoz még egy parancsot:

```
alfa=linspace(0,2*pi);
figure;
plot(cos(alfa),sin(alfa))
ax=gca;
ax.YAxisLocation='origin';
ax.XAxisLocation='origin';
axis equal;
```



`axis equal`: egyforma hosszú egységet választ minden tengelyen.

1. feladat

Olvassa be a `salary.xlsx` állományt. Ennek első oszlopában a szakmai tapasztalat (években), másodikban az megfelelő éves fizetések szerepelnek, valamilyen valutában. (Forrás: Kaggle) Ábrázolja az adatokat (diszkrét pontokként): a vízszintes tengelyen az éveket, a függőlegesen a fizetéseket.

Az adatok beolvasásához használhatja a `readtable` függvényt.

Ekkor a beolvasott értékek egy táblázat (`table`) típusú változóba kerülnek. Mivel a táblázat most csak numerikus adatokat tartalmaz, így a `table2array` függvénnyel numerikus tömbbé alakítható.

Az ábra alapján milyen jellegű kapcsolat látszik a szakmai tapasztalat és a fizetések között?

2. feladat

Olvassa be a `baleset.xlsx` állományt. Ebben a magyarországi lakosság számának alakulását találja 1990 és 2022 között, illetve ugyanezen években a személyi sérüléssel járó közúti közlekedési balesetek számát (forrás: KSH). Ábrázolja mindkét adatsort, majd ábrázolja az évek függvényében a 100000 főre jutó balesetek számát is.

3. feladat

Olvassa be a `japan_h_w_man.xlsx` állományt. Ebben többek között 5-17 éves japán fiúk átlagos magassága (cm) és testsúlya (kg) szerepel, városonkénti bontásban. (Forrás: Kaggle) Ábrázolja a testsúlyokat a magasság függvényében! Ábrázolja ugyanezt csak a 6 éves fiúk esetén.

Legkisebb négyzetes közelítések

Legkisebb négyzetes közelítések, polinom illesztése

1. feladat

Határozzuk meg az alábbi adatokat legkisebb négyzetes értelemben legjobban közelítő egyenest!

t_i	1	1.1	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2
f_i	8	8.9	9	9.8	10	11	11.5	11.5	12.5	13	13.7	14

Megoldás. Használjuk a `polyfit` függvényt!

```
p=polyfit(t,f,m)
```

megadja a (t_i, f_i) adatokra legkisebb négyzetes értelemben legjobban illeszkedő legfeljebb m-edfokú polinom együtthatóit a főegyütthatóval kezdve.

Polinom illesztése

1. soroljuk fel a t_i értékeket egy vektorban:

   ```
   >> t=[1 1.1 1.1:0.1:2];
   ```

2. egy másikban a megfelelő f_i értékeket:

   ```
   >> f=[8 8.9 9 9.8 10 11 11.5 11.5 12.5 13 13.7 14];
   ```

3. hívjuk meg a `polyfit` függvényt, a harmadik argumentumban 1-gyel (elsőfokú polinomot illesztünk)

   ```
   >> p=polyfit(t,f,1)
   ```

Az eredmény

```
p=
   5.8235    2.5338
```

A keresett egyenes egyenlete:

$$f(t) = 5.8235t + 2.5338$$

Polinom illesztése

A keresett egyenes egyenlete:

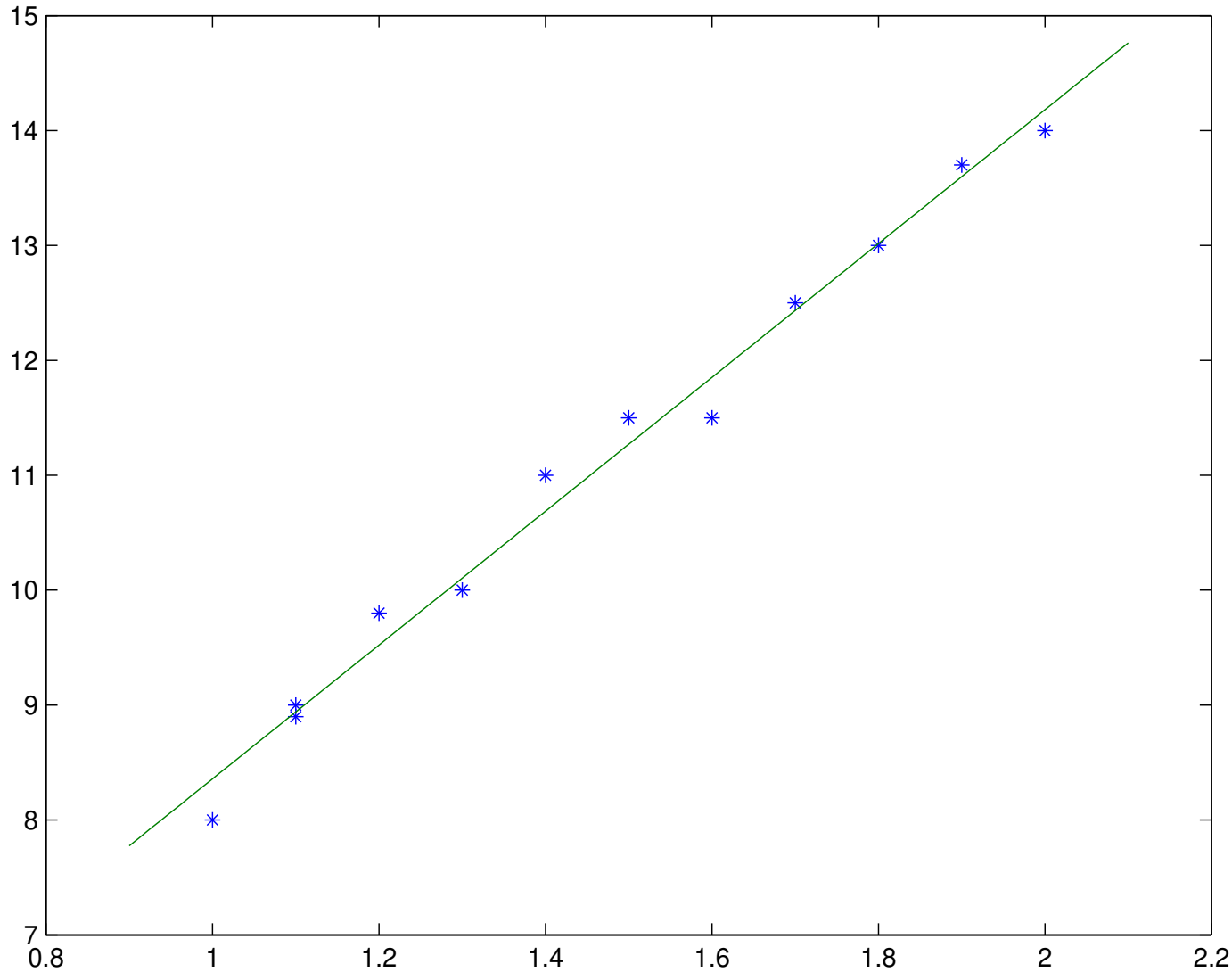
$$f(t) = 5.8235t + 2.5338$$

Ha ábrázolni szeretnénk az adatokat és az illesztett egyenest, akkor

- Vegyünk fel elég sok (pl 100) pontot egy olyan intevallumban, mely tartalmazza a t_i értékeket
- számítsuk ki a polinom értékét a felvett pontokban (használjuk a polyval függvényt)
- ábrázoljuk a megfelelő pontpárokat

```
>> xx=linspace(0.9,2.1);
>> yy=polyval(p,xx);
>> figure; plot(t,f,'*',xx,yy)
```

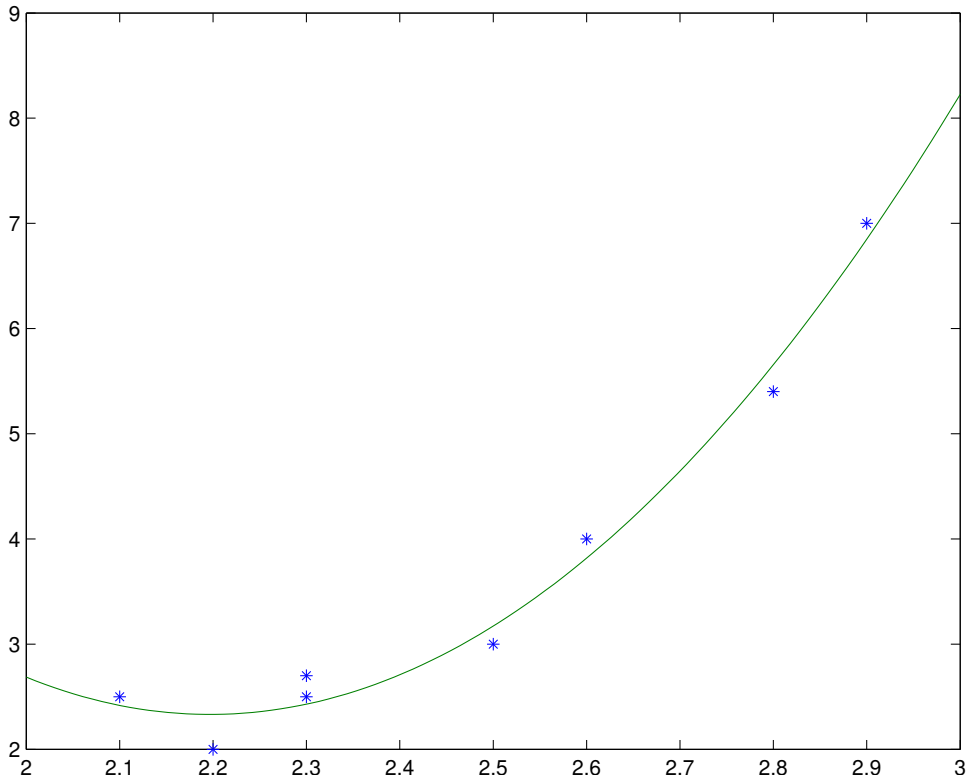
A `polyval` függvény a `p` együtthatójú polinom értékeit adja az `xx`-ben adott helyeken.


15
14
13
12
11
10
9
8
7
0.8
1
1.2
1.4
1.6
1.8
2
2.2

2. feladat

Határozzuk meg az alábbi adatokat legkisebb négyzetes értelemben legjobban közelítő másodfokú polinomot!

t_i	2.1	2.2	2.3	2.3	2.5	2.6	2.8	2.9
f_i	2.5	2	2.5	2.7	3	4	5.4	7



3. feladat

Egy fél méter magas, téglatest alakú víztartályt egyenletes sebességgel töltenek fel vízzel. Amikor a tartályban 3 cm magasan áll a víz Péter elhatározza, hogy megméri a vízszint változását az idő függvényében. A következő méréseket végezte:

t_i (min)	0	2	4	6	8	10	12
f_i (cm)	3	4	5	5.5	6.5	7	8

Becsülje meg milyen magasan lesz a víz 20 perccel azután, hogy Péter elindította a mérést! Mikor indították el a tartály feltöltését? Kb mikor lesz tele a tartály?

4. feladat

Egy ipari mérlegen egy nagyobb mennyiségű gabona van, amit valaki egyenletes sebességgel lapátol a mérlegről zsákokba. Miután elkezdte a munkát, időnként megnézzük mennyit mutat a mérleg. Az alábbi értékeket láttuk:

idő (min)	1	15	20	28
tömeg (kg)	980	605	470	250

Becsüljük meg mennyi gabona volt a mérlegen 23 perccel azután, hogy elkezdét zsákokba lapátolni.
Becsüljük meg mennyi ideig tart, amíg az összes gabonát zsákokba rakják, illetve eredetileg mennyi gabona volt a mérlegen.

5. feladat

Olvassa be a `salary.xlsx` állományt. Ábrázolja az adatokat (a szakmai tapasztalat függvényében a fizetéseket). Határozza meg az adatokat legkisebb négyzetes értelemben legjobban közelítő egyenes paramétereit és ábrázolja az egyenest az adatokkal közös ábrán. Adjon becslést arra, hogy 1.7, 2.5 és 6.5 év szakmai gyakorlattal mekkora éves fizetésre lehet számítani.

6. feladat

Olvassa be a `japan_h_w_man.xlsx` állományt. Ábrázolja a testsúlyokat a magasság függvényében! Határozza meg az adatokat legkisebb négyzetes értelemben legjobban közelítő egyenes egyenletét, majd határozza meg a legjobban illeszkedő másodfokú polinomot is. Mindkét esetben számítsa ki a közelítés hibáját. Melyik közelítés tűnik jobbnak?

Numerikus Matematika

Baran Ágnes

Gyakorlat
Legkisebb négyzetes közelítések 2.

Legkisebb négyzetes közelítések

Példa

Határozzuk meg az alábbi adatokat legkisebb négyzetes értelemben legjobban közelítő

$$F(t) = x_1 + x_2 \cos(\pi t) + x_3 \sin(\pi t)$$

alakú modell paramétereit!

t_i	0.1	0.5	1.2	1.5	2	2.1	2.4	3	3.2
f_i	3.9	2.6	-0.8	0.3	3.2	3.8	3.2	-0.7	-0.9

Megoldás. A paramétereket az

$$A^T A x = A^T f$$

Gauss-féle normálegyenlet megoldása szolgáltatja.

$$A^T A x = A^T f$$

ahol

$$A = \begin{bmatrix} 1 & \cos(\pi t_1) & \sin(\pi t_1) \\ 1 & \cos(\pi t_2) & \sin(\pi t_2) \\ \vdots & & \\ 1 & \cos(\pi t_9) & \sin(\pi t_9) \end{bmatrix}, \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_9 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Állítsuk elő a megadott adatokból az A mátrixot:

```
>> t=[0.1 0.5 1.2 1.5 2 2.1 2.4 3 3.2]';
>> f=[3.9 2.6 -0.8 0.3 3.2 3.8 3.2 -0.7 -0.9]';
>> A=[ones(9,1), cos(pi*t), sin(pi*t)];
```

Figyeljünk az A mátrix előállításánál: oszlopokat kell egymás mellé tennünk!

- Oldjuk meg a normálegyenletet!

  Ha az f is oszlopvektorként adott:

```
>> x=(A'*A)\(A'*f)
x =
    1.4372
    2.0310
    1.1711
```

  A legjobban illeszkedő adott alakú modell tehát:

$$F(t) = \underbrace{1.4372}_{x(1)} + \underbrace{2.0310}_{x(2)}\cos(\pi t) + \underbrace{1.1711}_{x(3)}\sin(\pi t)$$

  Ha f-et nem oszlopvektorként adtuk meg, akkor a normálegyenlet megoldásánál hibaüzenetet kapunk.

- Ábrázoljuk az adatokat és az illesztett modellt!
  - Vegyünk fel sok (pl 100) pontot egy olyan intervallumban, ami tartalmazza a mérési helyeinket. Most pl. a $[0, 3.3]$ intervallum megfelelő:

    ```
    >> xx=linspace(0,3.3);
    ```

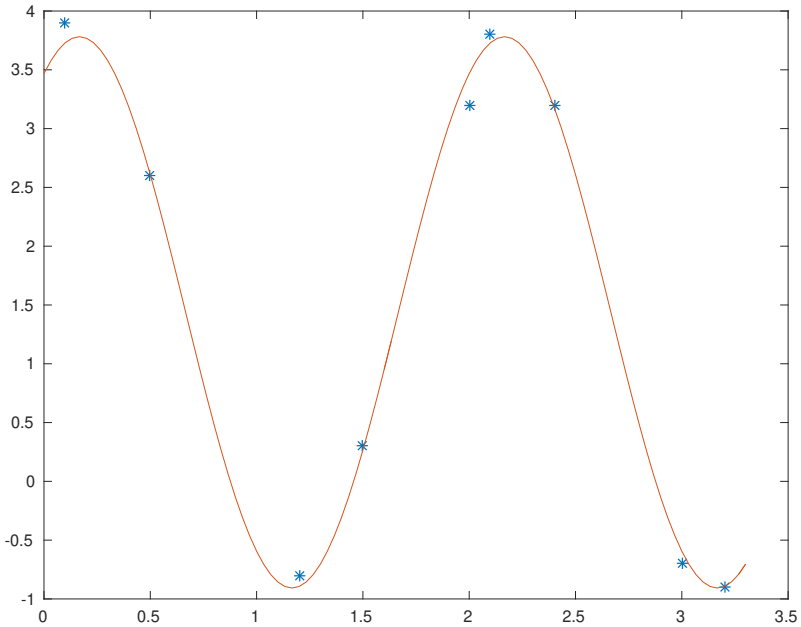
  - Az `xx` vektor elemeiben számoljuk ki a modell értékét. (A modell paraméterei az `x` vektorban vannak!)

    ```
    >> yy=x(1)+x(2)*cos(pi*xx)+x(3)*sin(pi*xx);
    ```

  - Ábrázoljuk az eredeti adatokat és az illesztett függvényt.

    ```
    >> plot(t,f,'*',xx,yy)
    ```

```
t=[0.1 0.5 1.2 1.5 2 2.1 2.4 3 3.2]';
f=[3.9 2.6 -0.8 0.3 3.2 3.8 3.2 -0.7 -0.9]';
A=[ones(9,1), cos(pi*t), sin(pi*t)];
x=(A'*A)\(A'*f);
xx=linspace(0,3.3);
yy=x(1)+x(2)*cos(pi*xx)+x(3)*sin(pi*xx);
figure; plot(t,f,'*',xx,yy)
```



Feladatok

(1) Határozza meg az alábbi adatokat legjobban közelítő

$$F(t) = a + \frac{b}{t}$$

alakú modell paramétereit!

t_i	1	1.2	1.4	1.4	1.5	1.7	1.9	2	2.1	2.2
f_i	4.2	3.8	3.4	3.3	3.3	3	2.8	2.8	2.75	2.7

(2) Határozza meg az alábbi adatokat négyzetesen legjobban közelítő

$$F(t) = x_1 \sin(t) + x_2 \sin(2t) + x_3 \sin(3t)$$

alakú modell paramétereit!

t_i	0.1	0.5	1.2	1.5	2	2.1	2.4	3	3.2	3.4	3.8	4	4.2	4.6	5
f_i	1	4.1	3	1	-1.5	-1.6	-1.7	-0.4	0.1	0.7	1.6	1.8	1.6	0.2	-2.5

3. feladat

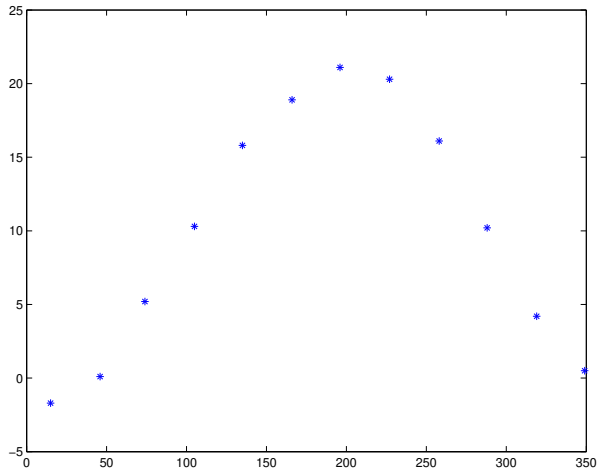
Havi középhőmérsékletek átlagai Budapesten (1901-1950)

t_i	15	46	74	105	135	166	196	227	258	288	319	349
f_i	-1.7	0.1	5.2	10.3	15.8	18.9	21.1	20.3	16.1	10.2	4.2	0.5

Illesszünk az adatokra

$$F(t) = x_1 + x_2 \cos\left(2\pi \frac{t-14}{365}\right)$$

alakú modellt.



4. feladat

Olvassa be a `trees.xlsx` állományt (Forrás: R programcsomag). Ez 31 fa (kései meggy) alábbi adatait tartalmazza: a törzs átmérője inch-ben (kissé félrevezetően: Girth oszlop), a fa magassága láb-ban (Height oszlop), a faanyag térfogata köbláb-ban (Volume oszlop). Ábrázolja a faanyag mennyiségét az átmérő függvényében, illetve a faanyag mennyiségét a magasság függvényében. A `plot3` függvény segítségével ábrázolja a faanyag mennyiségét az átmérő és a magasság függvényében. Illesszen egyenest a (Girth,Volume) adatokra, azaz becsülje meg a

$$\text{Volume} \sim x_1 + x_2 \cdot \text{Girth}$$

kapcsolatban az x_1, x_2 paramétereket! Végezze el a

$$\text{Volume} \sim x_1 + x_2 \cdot \text{Girth} + x_3 \cdot \text{Height}$$

közelítést is. Mindkét esetben számítsa ki az átlagos négyzetes hibát.

Példa

- Töltsük be a `carsmall` adathalmazt:

  ```
  >> load carsmall.mat
  ```

  Az adatállomány 100 autó adatait tartalmazza, most a `Horsepower`, `Weight`, `Acceleration`, `MPG` változókat fogjuk használni (a Wokspace-ben látjuk, hogy ezek 100 elemű oszlopvektorok). Az első három változóból próbáljuk megbecsülni a negyedik értéket.

- Ellenőrizzük, hogy az egyes vektorokban vannak-e hiányzó értékek!

  Használjuk az `isnan` függvényt, pl.

  ```
  >> sum(isnan(MPG))
     ans =
          6
  ```

  Csak azoknak az autóknak az adatait kellene megtartanunk, ahol mind a négy érték ismert. (Használhatjuk az `any` függvényt.)

- Ha töröljük azokat a sorokat, ahol van hiányzó érték akkor egy 93 elemű adathalmazunk maradt.
- Ábrázoljuk az MPG értékeket a súly függvényében!
- Keressük az MPG és Weight értékek közötti kapcsolatot lineáris függvény formájában:

$$\text{MPG} \sim x_1 + x_2 \cdot \text{Weight}$$

Ezt megoldhatjuk a `polyfit` függvénnyel, a Gauss-féle normálegyenlet megoldásával és a `fitlm` függvénnyel is.

- a polyfit függvénnyel: a (Weight,MPG) adatokra illesztünk egyenest.

  Figyeljünk, ne az eredeti Weight, MPG vektorokat használjuk, abban hiányzó értékek vannak!

  Emlékezzünk: a `polyfit` az illesztett egyenes együtthatóit a főegyütthatóval kezdve sorolja fel.

  Számítsuk ki az átlagos négyzetes hibát (a megfelelő képlet alkalmazásával, vagy használhatjuk a `immse` függvényt is)
- Oldjuk meg a feladatot a Gauss-féle normálegyenlettel is.

- a `fitlm` függvénnyel:

```
ans =
Linear regression model:
    y ~ 1 + x1
Estimated Coefficients:
                    Estimate         SE          tStat
                   ----------    ----------    -------

    (Intercept)       49.238        1.6504      29.834
    x1             -0.0086118    0.00053775    -16.014

Number of observations: 93, Error degrees of freedom: 91
Root Mean Squared Error: 4.16
R-squared: 0.738,  Adjusted R-Squared: 0.735
F-statistic vs. constant model: 256, p-value = 3.24e-28
```

Itt az együtthatókon kívül más információkat is kaptunk, ezekről ld. Statisztikából.

Ezután a Weight változón kívül használjunk még egy magyarázóváltozót, a Horsepower-t:

$$\text{MPG} \sim x_1 + x_2 \cdot \text{Horsepower} + x_3 \cdot \text{Weight}$$

Ebben az esetben is határozzuk meg az átlagos négyzetes hibát.

Numerikus matematika

Baran Ágnes

Gyakorlat
Lagrange-interpoláció

Lagrange-interpoláció

Példa

Határozzuk meg a $(-2,-5),(-1,3),(0,1),(2,15)$ pontokra illeszkedő minimális fokszámú polinomot!

Megoldás. Készítsük el az osztott differenciák táblázatát!
Az első két oszlopba az alappontok és a megfelelő függvényértékek kerülnek:

-2	-5
-1	3
0	1
2	15

Számítsuk ki az elsőrendű osztott differenciákat!

$$
\begin{array}{r|r|l}
-2 & -5 & \\
 & & \frac{3-(-5)}{-1-(-2)} = 8 \\
-1 & 3 & \\
 & & \frac{1-3}{0-(-1)} = -2 \\
0 & 1 & \\
 & & \frac{15-1}{2-0} = 7 \\
2 & 15 &
\end{array}
$$

Számítsuk ki a másodrendű osztott differenciákat!

$$
\begin{array}{c|c|c|c}
-2 & -5 & & \\
 & & 8 & \\
-1 & 3 & & \frac{-2-8}{0-(-2)} = -5 \\
 & & -2 & \\
0 & 1 & & \frac{7-(-2)}{2-(-1)} = 3 \\
 & & 7 & \\
2 & 15 & &
\end{array}
$$

Számítsuk ki a harmadrendű osztott differenciát!

$$
\begin{array}{c|cccc}
-2 & -5 & & & \\
 & & 8 & & \\
-1 & 3 & & -5 & \\
 & & -2 & & \frac{3-(-5)}{2-(-2)} = 2 \\
0 & 1 & & 3 & \\
 & & 7 & & \\
2 & 15 & & &
\end{array}
$$

A táblázat felső élét használva írjuk fel a polinomot!

-2	-5			
		8		
-1	3		-5	
		-2		2
0	1		3	
		7		
2	15			

$$L_3(x) = -5 + 8(x+2) - 5(x+2)(x+1) + 2(x+2)(x+1)x$$

Megj.:

A Lagrange-polinom nem függ az adatok sorrendjétől, így választhattuk volna a táblázat alsó élét is:

-2	-5			
		8		
-1	3		-5	
		-2		2
0	1		3	
		7		
2	15			

$$L_3(x) = 15 + 7(x-2) + 3(x-2)x + 2(x-2)x(x+1)$$

Mindkét esetben

$$L_3(x) = 2x^3 + x^2 - 3x + 1$$

Lagrange-interpoláció

1. feladat

Írja fel az alábbi pontokra illeszkedő minimális fokszámú polinomot!

(a) $(-3,-6)$, $(-2,-17)$, $(-1,-8)$, $(1,-2)$, $(2,19)$,

(b) $(-3,-31)$, $(-2,-8)$, $(1,1)$, $(2,24)$,

(c) $(-2,-13)$, $(-1,-4)$, $(1,2)$,

(d) $(-2,-5)$, $(-1,3)$, $(0,1)$, $(2,15)$,

(e) $(-1,4)$, $(1,2)$, $(2,10)$, $(3,40)$,

(f) $(-2,38)$, $(-1,5)$, $(1,-1)$, $(2,-10)$, $(3,-7)$,

(g) $(-2,-33)$, $(-1,-2)$, $(1,6)$, $(2,7)$, $(3,-18)$.

2. feladat

Horner-algoritmussal határozza meg a $p(-3)$ értéket, ha

$$p(x) = -x^5 + 3x^3 - 4x^2 - 3x + 5$$

Lagrange-interpoláció Octave/Matlab-bal

A `polyfit` függvény

`polyfit(x,f,n-1)` Ha x és f n-elemű vektorok, akkor megadja annak a legfeljebb $(n-1)$-edfokú polinomnak az együtthatóit, amely illeszkedik az (x_i, f_i), $i = 1, \ldots, n$ adatokra.

Példa

Határozzuk meg a $(-2,-5), (-1,3), (0,1), (2,15)$ pontokra illeszkedő minimális fokszámú polinomot!

Megoldás.

```
>>x=[-2, -1, 0, 2];
>>f=[-5, 3, 1, 15];
>>p=polyfit(x,f,3)
p=
   2.0000 1.0000 -3.0000 1.0000
```

Ábrázoljuk a pontokat és az illesztett függvényt!

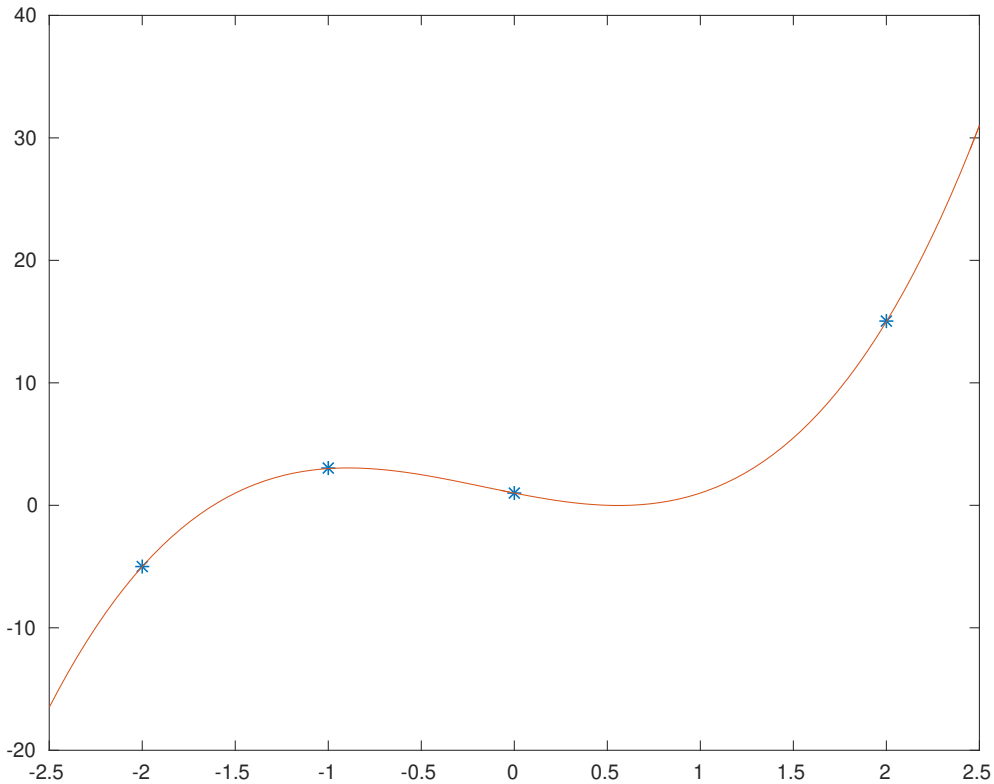
```
x=[-2, -1, 0, 2];
f=[-5, 3, 1, 15];
p=polyfit(x,f,3);
xx=linspace(-2.5,2.5);
yy=polyval(p,xx);
figure; plot(x,f,'*',xx,yy)
```

A `polyval` függvény:

```
yy=polyval(p,xx);
```

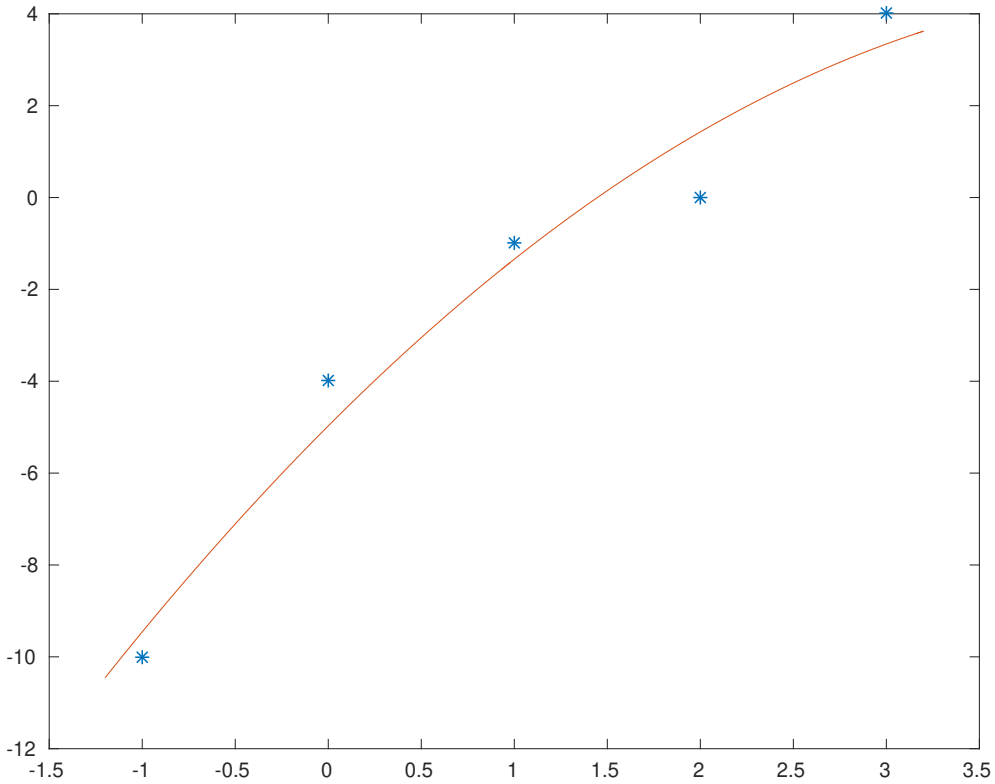
a `p` együtthatójú polinom értékeit adja az `xx` vektor koordinátáiban.
(`p`-ben a polinom együtthatói a főegyütthatóval kezdve szerepelnek)

```
x=[-2, -1, 0, 2];
f=[-5, 3, 1, 15];
p=polyfit(x,f,3);
xx=linspace(-2.5,2.5);
yy=polyval(p,xx);
figure; plot(x,f,'*',xx,yy)
```



Fontos! Ha a `polyfit` függvényben nem megfelelően írjuk elő a polinom fokszámát, akkor a polinom nem feltétlenül illeszkedik az adatokra.

```
x=[-1 0 1 2 3]; f=[-10 -4 -1 0 4]; p=polyfit(x,f,2);
xx=linspace(-1.2,3.2); ff=polyval(p,xx);
figure; plot(x,f,'*',xx,ff)
```



3. feladat

Közelítse az

$$f(x) = e^x - \sin(\pi x)$$

függvényt a $[0, 1]$ intervallumon egy másodfokú polinommal. Ábrázolja az eredeti és az illesztett függvényt közös ábrán.

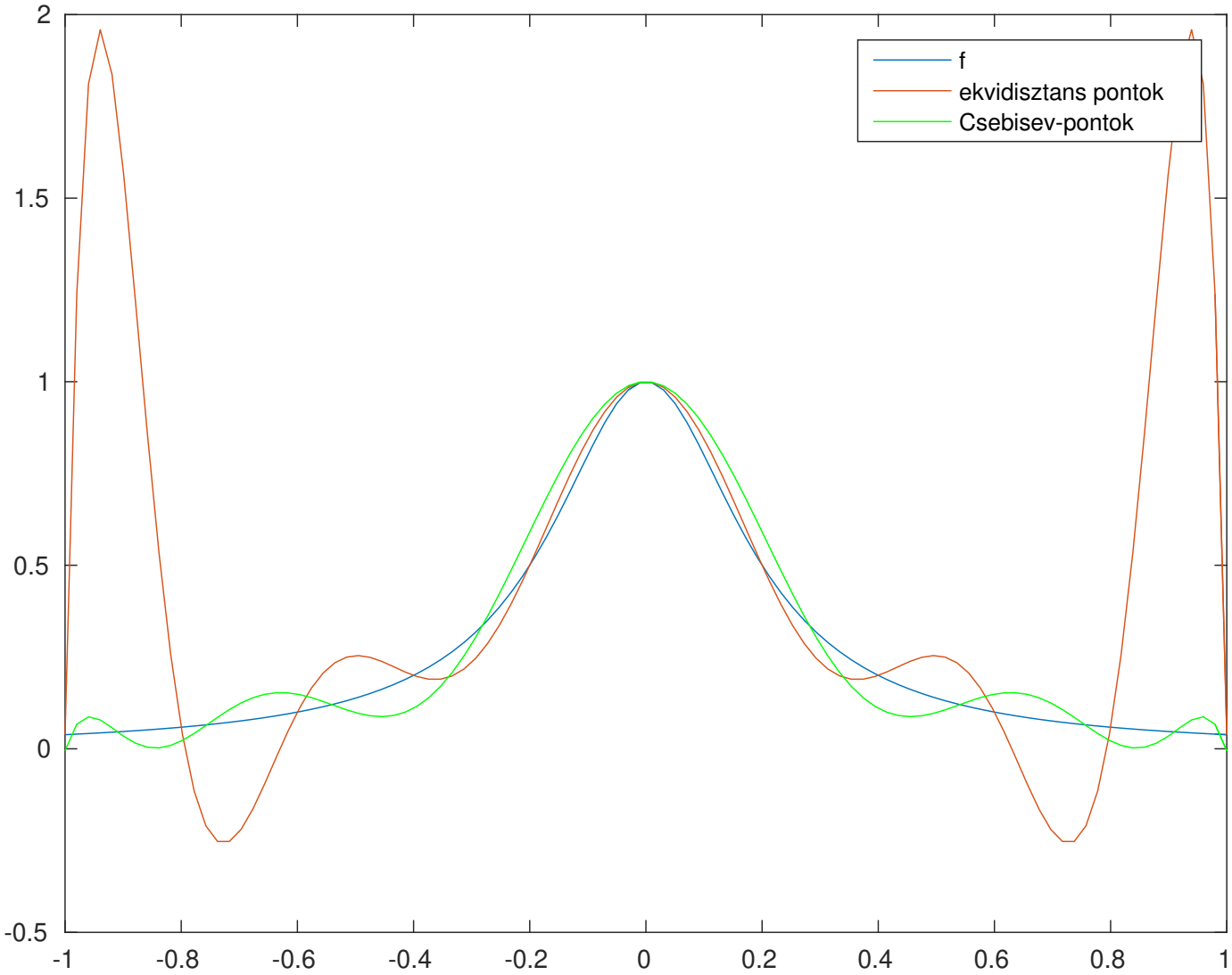
4. feladat

Tudjuk, hogy egy test méterben számolva s_0 utat tett meg, egyenletes v_0 (m/s) sebességgel, majd ezután egyenletesen gyorsítani kezdett a (m/s^2) gyorsulással. A gyorsulás kezdetétől számítva a 2., 4. és 5. másodperc végén az összes megtett út rendre 16, 38 és 52 m. Határozza meg s_0, v_0 és a értékét.

5. feladat (Szorgalmi)

Rajzoltassuk ki közös ábrára az alábbi 3 függvényt:

- az
$$f(x) = \frac{1}{1 + 25x^2}$$
függvényt a $[-1, 1]$ intervallumon
- az f függvény
$$-1, -0.8, -0.6, \ldots, 0.6, 0.8, 1$$
egyenlő lépésközű (ekvidisztáns) alappontokhoz tartozó Lagrange-polinomját
- az f függvény
$$x_k = \cos\left(\frac{2k-1}{22}\pi\right), \quad k = 1, 2, \ldots, 11$$
alappontokhoz (Csebisev-pontok) tartozó Lagrange-polinomját.


f
ekvidisztans pontok
Csebisev-pontok
2
1.5
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

6. feladat (Szorgalmi)

Írjon egy kódot, mely adott alappontok és függvényértékek esetén az osztott differenciák táblázatát határozza meg.
Tesztelje a kódját a legelső példával, illetve az első feladatban adott pontpárokkal.

7. feladat

Írjon egy Matlab-függvényt, mely Horner-algoritmus segítségével kiszámítja egy polinom értékét egy megadott x helyen. (Ne használja a polyval függvényt.) A függvény bemenetei: a polinom együtthatóit tartalmazó vektor és az x érték.
Módosítsa a kódját úgy, hogy egyszerre több helyen is ki tudja számolni a helyettesítési értéket (ekkor x vektorként adott).

8. feladat

Egy pontszerű testet a $(0, y_0)$ pontból, a vízszintessel α szöget bezáró irányban felfelé v_0 $[m/s]$ kezdősebességgel elhajítunk, akkor a test pályája:

$$y = y_0 + x \tan \alpha - \frac{g}{2v_0^2 \cos^2 \alpha} x^2,$$

ahol g $[m/s^2]$ a nehézségi gyorsulás.
Ha tudjuk, hogy a test pályája áthalad a $(1, 2.3957)$, $(2, 2.4280)$, $(4, 1.4027)$ pontokon, akkor mi lesz a földetérési helyének x koordinátája? (Feltételezzül, hogy a talaj vízszintes).

Használhatja a `roots` függvényt.

Numerikus matematika

Baran Ágnes

Gyakorlat
Interpoláció 2.

Hermite-interpoláció

1. feladat

Határozzuk meg az alábbi adatokra illeszkedő minimális fokszámú polinomot, illetve az illesztett polinom értékét a megadott x_0 pontban!

(a)

x_i	-1	1
$f(x_i)$	-1	3
$f'(x_i)$		0

$x_0 = 0$

(b)

x_i	-1	2
$f(x_i)$	7	1
$f'(x_i)$	-11	-2

$x_0 = 1$

(c)

x_i	-2	1
$f(x_i)$	23	2
$f'(x_i)$	-25	
$f''(x_i)$	18	

$x_0 = -1$

(d)

x_i	-2	2
$f(x_i)$	16	-20
$f'(x_i)$	-21	-29

$x_0 = 0$

2. feladat

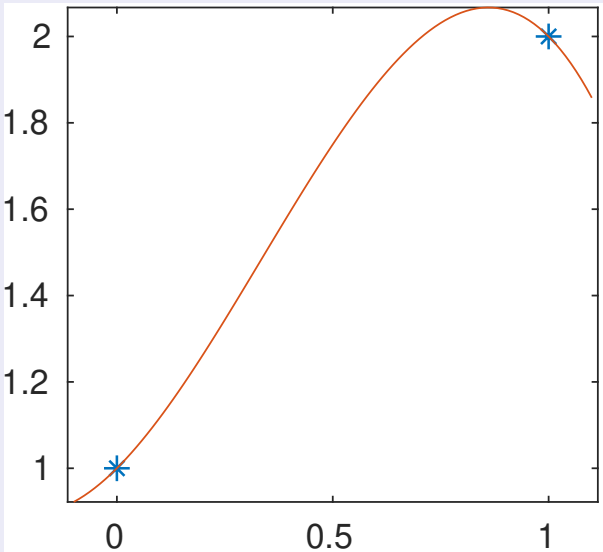
Írja fel az $f(x) = \cos(x) - 3x$ függvény $x_0 = 0$-beli érintőjének az egyenletét!

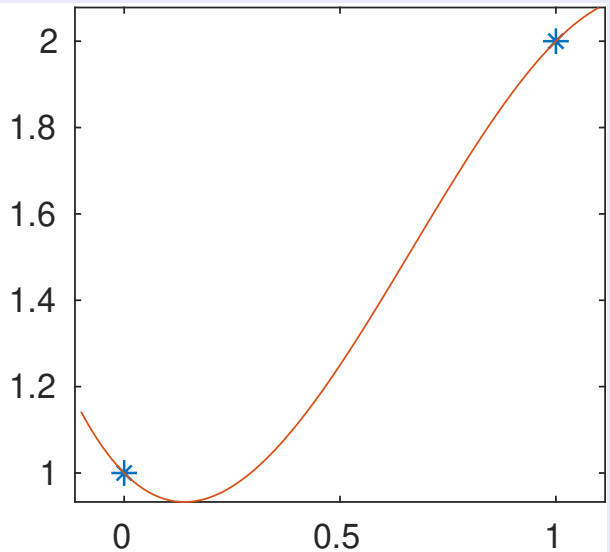
3. feladat

Meghatároztuk az alábbi adatokra illeszkedő minimális fokszámú polinomot:

x_i	0	1
$f(x_i)$	1	2
$f'(x_i)$	1	-1

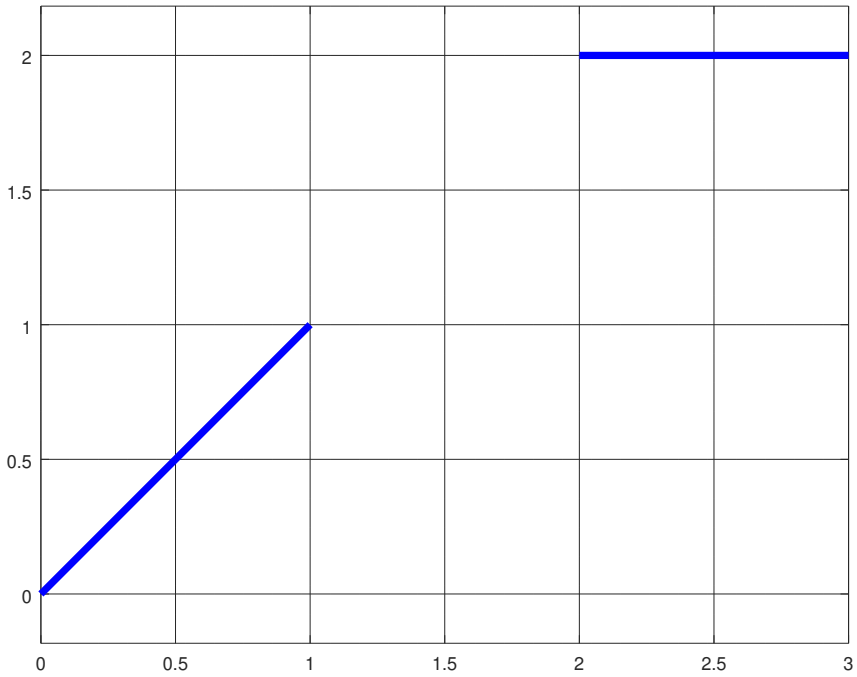
A pontokat és a kapott polinomot ábrázoltuk. Matlab használata nélkül döntse el, hogy melyik ábrát kaptuk az alábbiak közül. (Válaszát indokolja.)





4. feladat

Az ábrán kékkel jelölt két útszakaszt szeretnénk összekötni úgy, hogy a végeredményként kapott út vonalában ne legyen törés. Adja meg az összekötő útszakaszt leíró függvényt. Rajzoltassa ki az összekötő útszakaszt, az eredeti szakaszokkal együtt.



Spline interpoláció Octave/Matlab-bal

Példa

Határozzuk meg az alábbi adatokhoz tartozó harmadfokú spline-t!

x_i	-2	-1	0	1	2	3
S	4	1	7	4	12	9
S'	15					8

Megoldás. Használjuk a `spline` függvényt!

```
p=spline(x,y)
```

Előállítja a szakaszonként harmadfokú spline együtthatóit. Itt `x` az alappontok vektora, az `y` vektor első és utolsó koordinátája a két végpontban adott deriváltérték, a többi koordináta a függvényértékek.

```
>>x=-2:3; y=[15 4 1 7 4 12 9 8]; p=spline(x,y)
p =
  scalar structure containing the fields:
    form = pp
    breaks =
      -2  -1   0   1   2   3

    coefs =
       19.0000  -37.0000   15.0000    4.0000
      -12.0000   20.0000   -2.0000    1.0000
       11.0000  -16.0000    2.0000    7.0000
      -12.0000   17.0000    3.0000    4.0000
       15.0000  -19.0000    1.0000   12.0000

    pieces =  5
    order =  4
    dim =  1
```

A spline együtthatói: `p.coefs`

Figyeljünk arra, hogy a polinomok együtthatóit a részintervallumok kezdőpontjaihoz viszonyítva kapjuk!

Az 5 illesztett polinom:

$$p_1(x) = 19(x+2)^3 - 37(x+2)^2 + 15(x+2) + 4$$
$$p_2(x) = -12(x+1)^3 + 20(x+1)^2 - 2(x+1) + 1$$
$$p_3(x) = 11x^3 - 16x^2 + 2x + 7$$
$$p_4(x) = -12(x-1)^3 + 17(x-1)^2 + 3(x-1) + 4$$
$$p_5(x) = 15(x-2)^3 - 19(x-2)^2 + (x-2) + 12$$

Ellenőrizzük az illeszkedési feltételeket!

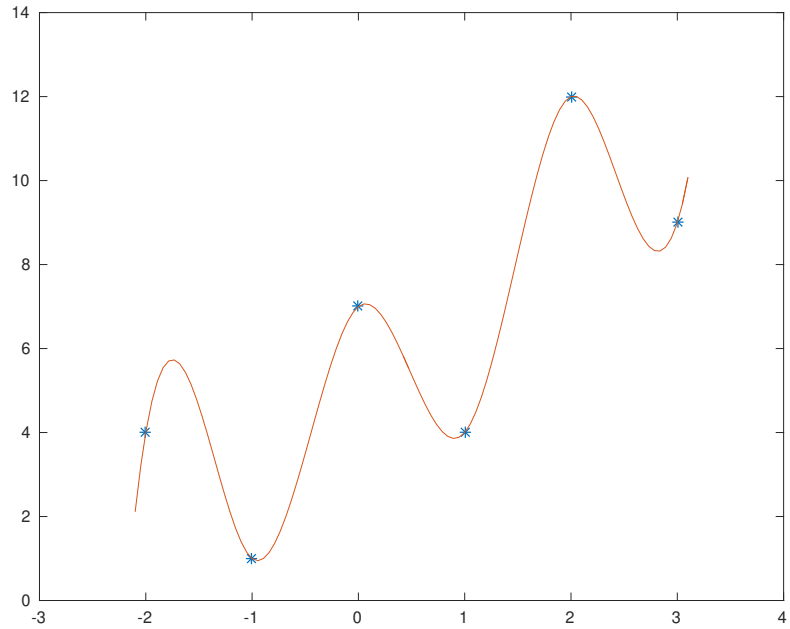
Ha nem az együtthatókat szeretnénk tudni, hanem a spline értékét valamely pont(ok)ban, akkor

`yy=spline(x,y,xx)`

ahol `x` és `y` az előbbi vektorok, `xx` azon pontok vektora, ahol a helyettesítési értéket keressük. Ekkor `yy`-ba kerülnek a kiszámolt függvényértékek.

```
>> x=-2:3;
>> y=[15 4 1 7 4 12 9 8];
>> xx=linspace(-2.1,3.1);
>> yy=spline(x,y,xx);
>> plot(x,y(2:end-1),'*',xx,yy)
```

```
x=-2:3;
y=[15 4 1 7 4 12 9 8];
xx=linspace(-2.1,3.1);
yy=spline(x,y,xx);
plot(x,y(2:end-1),'*',xx,yy)
```



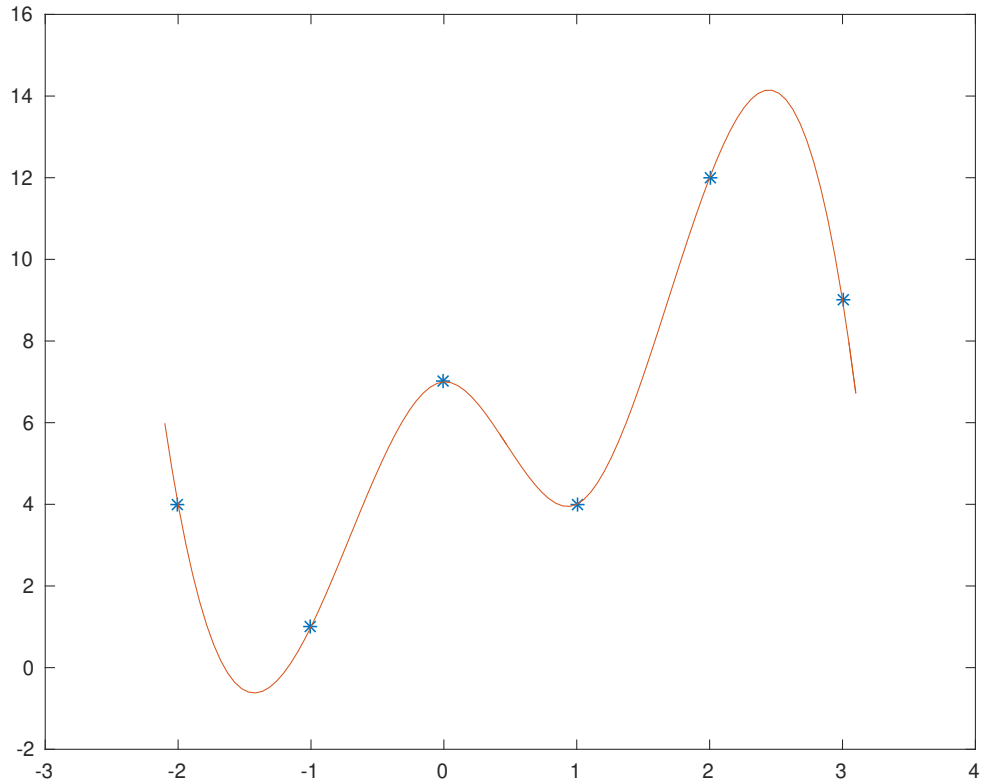
Az így kapott függvény teljesíti az illeszkedési feltételeket, és az első két deriváltja folytonos.

Megjegyzés

Ha a `spline` függvényt olyan `x` és `y` vektorokkal hívjuk, amelyek ugyanannyi koordinátát tartalmaznak, akkor a hiányzó két feltételt az Octave/Matlab azzal helyettesíti, hogy az első és utolsó két részintervallum találkozásánál a harmadik deriváltat is folytonosnak tekinti.

```
x=-2:3;
y=[4 1 7 4 12 9];
xx=linspace(-2.1,3.1);
yy=spline(x,y,xx);
plot(x,y,'*',xx,yy)
```

```
x=-2:3;
y=[4 1 7 4 12 9];
xx=linspace(-2.1,3.1);
yy=spline(x,y,xx);
plot(x,y,'*',xx,yy)
```



5. feladat

Rajzoltassuk ki közös ábrán az alábbi 3 függvényt:

- az

$$f(x) = \frac{1}{1 + 25x^2}$$

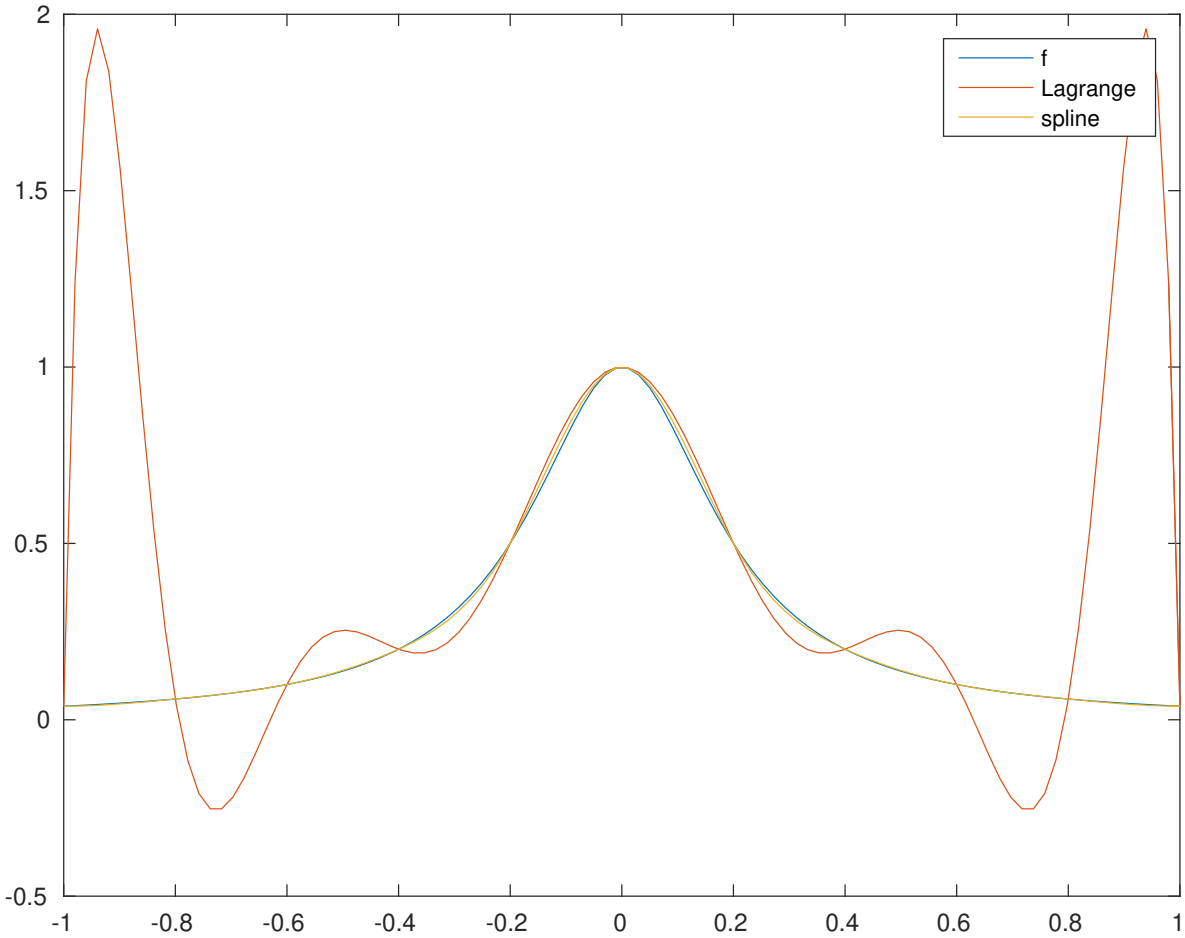
  függvényt a $[-1, 1]$ intervallumon
- az f függvény

$$-1, -0.8, -0.6, \ldots, 0.6, 0.8, 1$$

  egyenlő lépésközű (ekvidisztáns) alappontokhoz tartozó Lagrange-polinomját
- az f függvény

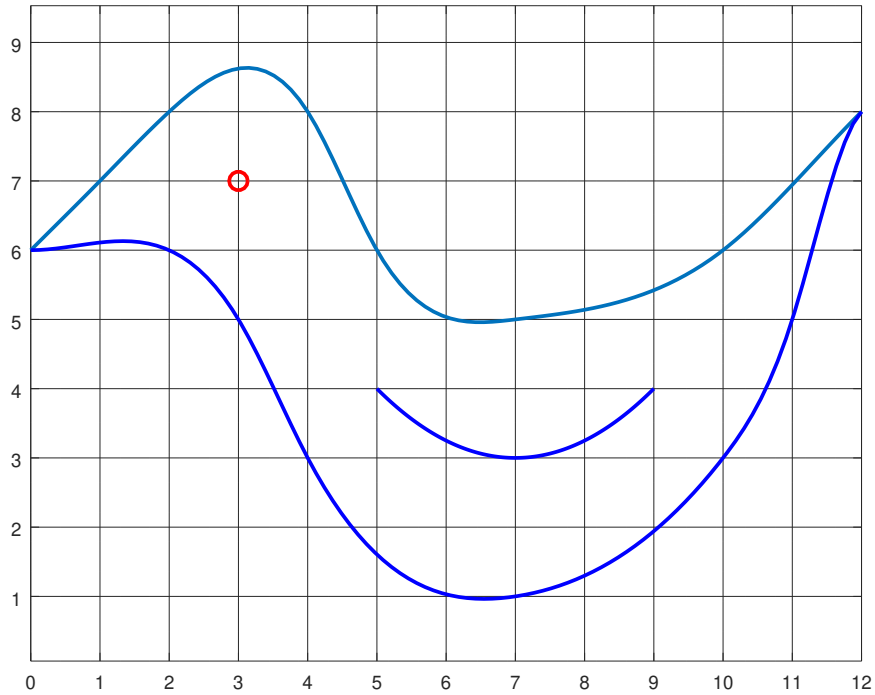
$$-1, -0.8, -0.6, \ldots, 0.6, 0.8, 1$$

  alappontokhoz tartozó harmadfokú spline polinomját. (A végpontokban a deriváltértékeket tekintsük 0-nak.)

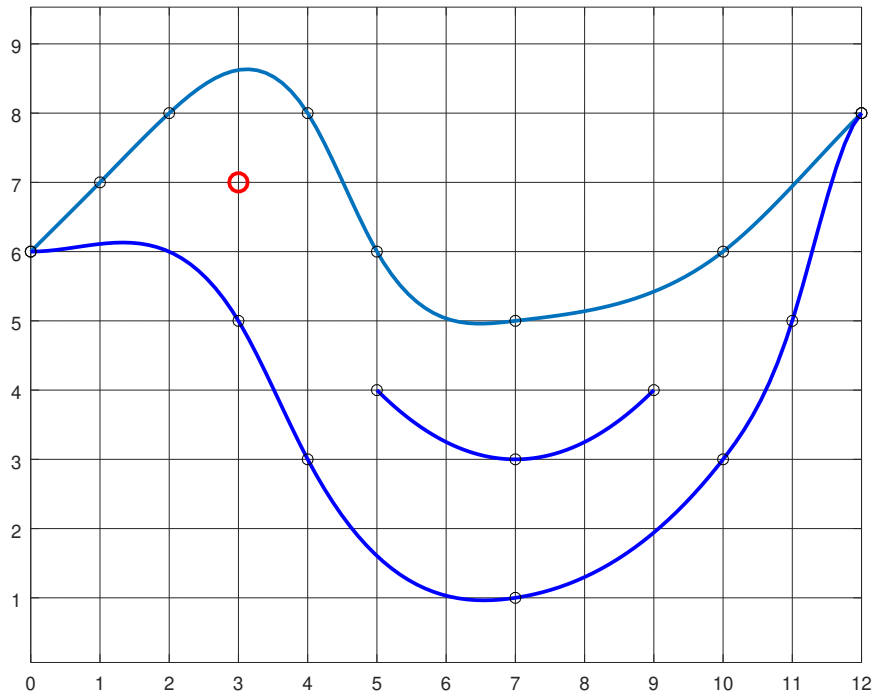

f
Lagrange
spline
2
1.5
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

6. feladat (szorgalmi)

Készítse el Octave-val az ábrán látható rajzot.



Útmutatás: használja a bejelölt (egész koordinátájú) pontokat és a `spline` függvényt.



MATLAB, függvények.

Baran Ágnes

March 24, 2024

Függvények írása

A Matlab-függvények szerkezete:

```
function [kimenovaltozok]=fvneve(bemenovaltozok)
  utasitasok
end
```

- A függvény a `function` és `end` kulcsszavak között helyezkedik el
- a függvény bemenő változói a kerek zárójelek között vanank felsorolva, vesszővel elválasztva
- a függvény által visszaadott értékek a szögletes zárójelben vannak felsorolva, vesszővel elválasztva
- a bemenő értékekből az `utasitasok`-ban megadott módon határozza meg a visszaadott értékeket

Ha a fenti függvényt egy külön m-fájlban írtuk meg, akkor fvneve.m néven kell elmenteni. Ha a függvényt egy kód részeként hoztuk létre, akkor az m-fájl végén kell állnia (ekkor akár több ilyen függvény is lehet a fájl végén). A függvény ilyenkor csak az adott m-fájlon belül hívható (ú.n. lokális függvény).

Példák

```
function  y=fv1(x)
   y=x.^2-1;
end
```

Ekkor a `y=fv1(x)` utasítás eredménye a $x^2 - 1$ kifejezés értéke, ahol x akár vektor is lehet. Utóbbi esetben a függvény elemenként hajtódik végre és y is vektor (ezt az teszi lehetővé, hogy a függvényben minden művelet végrehajtható vektorokra is, mivel a négyzetreemelés jele elé kitettük a . jelet)
Például `y=fv1(-3)` esetén:

```
y =
     8
```

`y=fv1([1,2,3])` esetén:

```
y =
     0     3     8
```

Példák

```
function  [s,p]=fv2(x,y,z)
    s=x+y+z;
    p=x*y*z;
end
```

Ennek a függvénynek 3 bemenő paramétere van, ezeket vesszővel elválasztva soroljuk (akkor is, ha nem azonos típusúak, pl. egy skalár egy vektor és egy mátrix lenne).
A függvény 2 értékkel tér vissza: a bemenetként megadott 3 szám összegével és szorzatával.
Ha több visszatérési érték van, akkor ezeket a függvény létrehozásakor és hívásakor szögletes zárójelben soroljuk (akkor is, ha nem azonos típusúak).

```
>>[s,p]=fv2(-5,1,4)
s =
     0
p =
   -20
```

Ha a függvény hívásakor nem adunk meg kimenő változókat (vagy csak egyet adunk), akkor a két visszatérési érték közül az elsőt adja:

```
>>fv2(-5,1,4)
ans =
     0
```

Ha csak a második visszatérési értékre vagyunk kiváncsiak, akkor

```
>>[~,p]=fv2(-5,1,4)
p =
     -20
```

function handle

A function handle egy függvényekkel kapcsolatos adattípus.
Szükségünk lehet rá, pl. ha

- egy függvényt át akarunk adni egy másik függvénynek (pl a Matlab `integral` függvényének az integrálandó függvényt)
- parancssorban szeretnénk függvényt definiálni

Létrehozása: a @ szimbólum után következik

- egy már létező függvény neve, pl. az előbb létrehozott függvényekkel `af1=@fv1` vagy `af2=@fv2`
- egy ú.n. anoním függvény, pl. `af3=@(x) x.^3-x` vagy `af4=@(x,y) x+y-x.*y`

Hívása:

- `af1(-3)` vagy `af1([1,2,3])` vagy `[s,p]=af2(-5,1,4)`
- `af3(-3)` vagy `af3([1,2,3])` vagy `af4(6,1)`

Az előző példában miért

`af1([1,2,3])` és `af2(-5,1,4)`?

Az egyik esetben miért használtunk szögletes zárójelet, a másikban miért nem?

Az `af1` (illetve az `fv1`) egy egyváltozós függvény. Ez az egy változó lehet akár vektor is (`fv1`-ben elemenkénti műveleteket használtunk!), de a függvényt csak egyetlen változóval hívhatjuk. Ez most az `[1,2,3]` vektor. Ha elhagynánk a szögletes zárójelet, akkor 3 skalárral próbálnánk hívni a függvényt.

Az `af2` (illetve az `fv2`) egy háromváltozós függvény, 3 értékkel kell hívnunk. Ezek most a -5, 1 és 4. Ha szögletes zárójelbe tennénk a 3 értéket, akkor az egyetlen változó lenne (egy vektor).

Ha

`af3=@(x) x.^3-x` és `af4=@(x,y) x+y-x.*y`

akkor `af3` egy egyváltozós függvény (ahol a változó akár vektor is lehet, mert a hatványozás előtt használtuk a pontot). Hívhatjuk egyetlen számmal, ekkor egy számot ad vissza, vagy egy vektorral, ekkor a vektor minden elemére kiértékeli a függvényt (és az értékek vektorát adja vissza).

`af4` egy kétváltozós függvény. Hívhatjuk 2 számmal:

```
>> af4(6,1)
ans =
     1
```

de két (egyforma méretű) vektorral is, mert a szorzást elemenként definiáltuk. Ekkor a visszaadott érték is vektor:

```
>> af4([6,-1],[1,2])
ans =
     1     3
```

Kiértékelte a függvényt a $(6, 1)$ és $(-1, 2)$ párokra.

Numerikus matematika

Baran Ágnes

Nemlineáris egyenletek

Nemlineáris egyenlet gyökei Matlab-bal: `fzero`

Példa: Határozzuk meg a

$$4\cos x = x$$

egyenlet $[-2\pi, 2\pi]$-be eső gyökeit!

Rendezzük 0-ra az egyenletet:

$$4\cos x - x = 0$$

Az egyenlet megoldása az

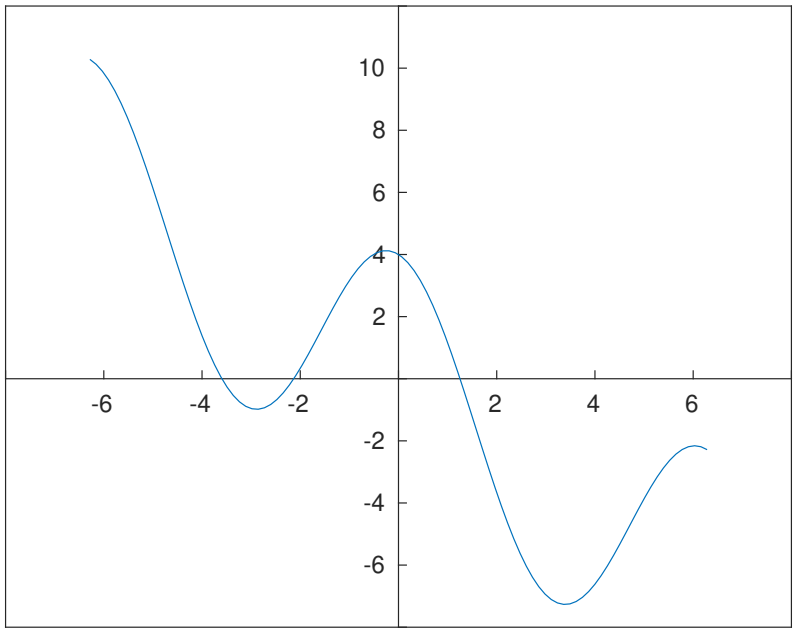
$$f(x) = 4\cos x - x$$

függvény zérushelyeinek megkeresését jelenti.

Nemlineáris egyenlet gyökei Matlab-bal: `fzero`

A Matlab a függvény gyökét egy iterációval közelíti, ehhez szüksége van egy kiinduló közelítésre.

Ilyen kezdeti közelítésre pl. a függvény ábrázolásával tehetünk szert:



(A függvény gyöke: ahol a gráfja metszi az x-tengelyt.)

Az ábra alapján ebben az intervallumban 3 gyök van: -4, -2 és 1 környékén.

Nemlineáris egyenlet gyökei Matlab-bal: `fzero`

Ha `f` egy function handle típusú változó, `x0` egy kezdeti közelítés, akkor

`fzero(f,x0)`

az `f` egy gyökének közelítésével tér vissza.

```
>> f=@(x) 4*cos(x)-x;
>> fzero(f,-4)
ans =
   -3.5953
>> fzero(f,-2)
ans =
   -2.1333
>> fzero(f,1)
ans =
    1.2524
```

A három gyök közelítése: -3.5953, -2.1333, 1.2524

Nemlineáris egyenlet gyökei Matlab-bal: `fzero`

Ha az `fzero` függvényt 2 vissztérési értékkel hívjuk, akkor nem csak a gyök közelítését, hanem ezen a helyen a függvény értékét is visszaadja:

```
>> [gyok,fvertek]=fzero(f,-4)
gyok =
   -3.5953
fvertek =
     0
>> [gyok,fvertek]=fzero(f,-2)
gyok =
   -2.1333
fvertek =
   4.4409e-16
```

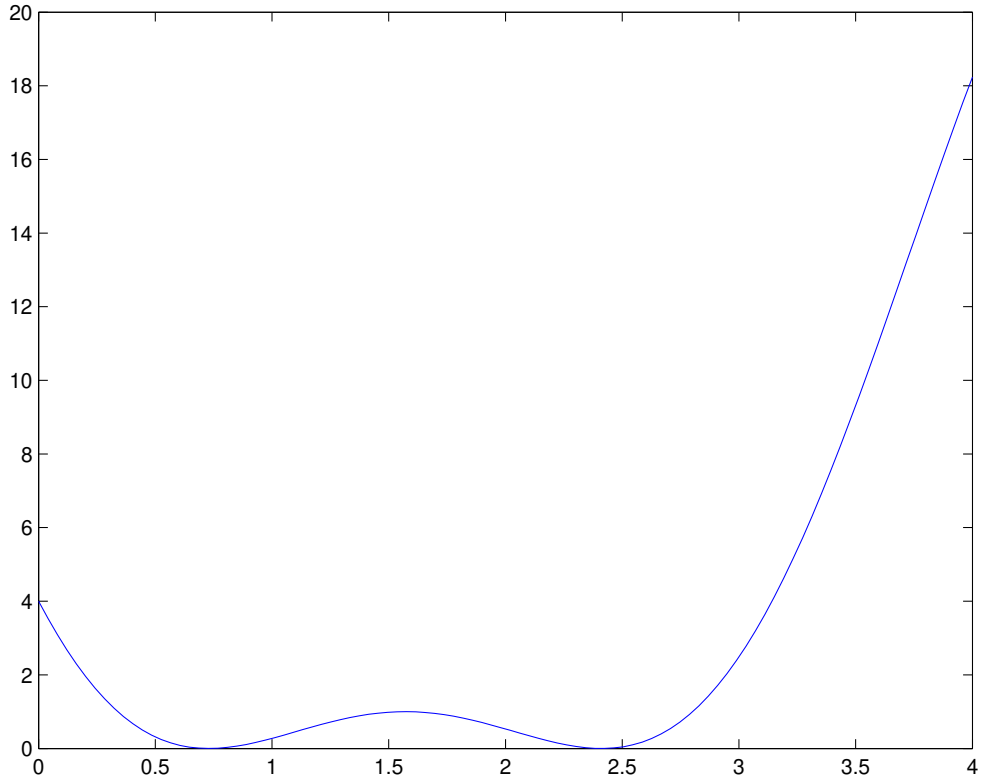
Látjuk, hogy
-3.5953-ben az f értéke 0,
-2.1333-ben nem 0, de 0-hoz nagyon közeli ($4.4409 \cdot 10^{-16}$).

Nemlineáris egyenlet gyökei Matlab-bal: `fzero`

Ha a függvény nem vált előjelet a gyök környezetében, akkor az `fzero` függvény nem találja meg a gyököt:

```
>> f=@(x) 13-9*cos(x).^2-12*sin(x);
>> x=fzero(f,0)
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search.
(Function value at -Inf is NaN.)
Check function or try again with a different starting value.

x =

   NaN
```

Ábrázoljuk az f függvényt a $[0, 4]$ intervallumon!



Az ábra alapján azt sejtjük, hogy a függvénynek 0.5 és 2.5 környezetében van 1-1 gyöke, ahol a függvény nem vált előjelet. Az is látszik, hogy itt a függvénynek minimuma van.

Nemlineáris egyenlet gyökei Matlab-bal: `fsolve`

Ha az `fzero` függvény helyett az `fsolve` függvényt használjuk:

```
>> f=@(x) 13-9*cos(x).^2-12*sin(x);
>> x=fsolve(f,0)
Equation solved.

fsolve completed because the vector of function values is
near zero as measured by the value of the function tolerance,
and the problem appears regular as measured by the gradient.

<stopping criteria details>

x =
    0.7277
```

Így megkaptuk az egyik gyök közelítését. Egy másik lehetőség, ha ilyenkor a függvény minimumhelyét próbáljuk megkeresni.

fminbnd

- `x=fminbnd(f,xmin,ymin)`
- `[x,fval,exitflag,output]=fminbnd(f,xmin,xmax)`

Megkeresi az `f` függvény $[xmin, xmax]$ intervallumbeli minimumát.

```
>> f=@(x) 13-9*cos(x).^2-12*sin(x);
>> [x,fval]=fminbnd(f,0,1)
x =
    0.7297
fval =
   9.2491e-11
>> [x,fval]=fminbnd(f,2,3)
x =
    2.4119
fval =
   1.7231e-13
```

A függvényérték mindkét esetben 0-hoz nagyon közeli, így a gyökök közelítését kaptuk.

Polinomok gyökei: `roots`

Polinomok gyökeinek közelítésére a `roots` függvényt használhatjuk:

```
r=roots(p)
```

ahol a `p` vektorban a polinom együtthatóit kell felsorolni a főegyütthatóval kezdve.

Példa

Közelítsük a $p(x) = 2x^3 - x + 1$ polinom gyökeit.

```
>> p=[2 0 -1 1]
>> r=roots(p)
r =

  -1.00000 + 0.00000i
   0.50000 + 0.50000i
   0.50000 - 0.50000i
```

1. feladat

(a) Közelítse a $3x = \cos(x)$ egyenlet gyökeit!

(b) Közelítse a $3x^3 - 12x + 4 = 0$ egyenlet gyökeit!

(c) Közelítse az $e^x = \sin(x)$ egyenlet gyökeit!

(d) Közelítse az $\ln(x) = 2 - x$ egyenlet gyökét!

(e) Közelítse a $\cos^2(x) + 2\sin(x) = 2$ egyenlet gyökét!

(f) Közelítse az $x^4 - x^3 - 2x^2 - 2x + 4 = 0$ egyenlet gyökeit!

Nemlineáris egyenletrendszer megoldása: `fsolve`

Példa: Oldjuk meg az alábbi egyenletrendszert!

$$x^2 + \frac{2y^2}{x} = 5$$
$$x^2 - xy = -1$$

Rendezzük 0-ra az egyenleteket:

$$x^2 + \frac{2y^2}{x} - 5 = 0$$
$$x^2 - xy + 1 = 0$$

Hozzunk létre egy függvényt, mely egy kételemű vektorral tér vissza: az előző rendszer bal oldalán álló kifejezések értékével. A függvénynek 1 változója legyen, egy vektor, azaz $x \mapsto x_1$, $y \mapsto x_2$.

```
f=@(x) [x(1)^2+2*x(2)^2/x(1)-5,x(2)^2-x(1)*x(2)+1];
```

Nemlineáris egyenletrendszer megoldása: `fsolve`

```
f=@(x) [x(1)^2+2*x(2)^2/x(1)-5,x(2)^2-x(1)*x(2)+1];
```

Hívjuk meg az `fsolve` függvényt:

```
[gyok,fvertek]=fsolve(f,x0)
```

ahol `x0` a gyökök kezdeti közelítését tartalmazó vektor.

Egy nemlineáris egyenletrendszernek több gyöke is lehet, a megfelelő kezdeti értékek megkeresése sokszor nem egyszerű feladat.

Más-más kezdeti közelítésből indulva más-más gyököket kaphatunk.

Az is előfordulhat, hogy egy adott kezdeti közelítésből indulva a Matlab nem talál gyököt.

Nemlineáris egyenletrendszer megoldása: `fsolve`

```
>> [gyok,fvertek]=fsolve(f,[1,1])
```

A parancs végrehajtása után a Matlab az alábbi értékekkel tér vissza:

```
gyok =
    2.0000    1.0000
fvertek =
   1.0e-08 *
    0.8598    0.7406
```

Tehát az egyenletrenszer gyökei 4 tizedesjegyre kerekítve (az eredeti jelölésekkel) $x = 2$ és $y = 1$. A Matlab a kiszámolt értékeket visszhelyettesítette az átrendezett egyenletek bal oldalán álló kifejezésekbe. Az így kapott értékek: $0.8598 \cdot 10^{-8}$ és $0.7406 \cdot 10^{-8}$.

Most kézzel is könnyen ellenőrizhető, hogy $x = 2$ és $y = 1$ pontos megoldások.

2. feladat

Közelítse az alábbi egyenletrendszer gyökét a $[-1,1]^2$ tartományon.

$$\sin(x_1 + 2x_2) + x_1 x_2 = 0$$
$$\cos(x_2 - 1) - \sin(x_1) = 0$$

3. feladat

Közelítse az alábbi egyenletrendszer gyökét a $[-\pi,\pi]^2$ tartományon.

$$\begin{aligned} -4x_1 + \cos(2x_1 - x_2) &= 3 \\ -3x_2 + \sin x_1 &= 2 \end{aligned}$$

4. feladat

Mutassa meg, hogy az $3x^3 - 12x + 4 = 0$ egyenletnek van gyöke a $[0, 1]$ intervallumban. Vizsgálja meg az $x_0 \in [0, 1]$,

$$x_{k+1} = \frac{3x_k^3 + 4}{12}, \quad k = 0, 1, ...$$

eljárás konvergenciáját! Írjon egy Matlab-kódot, amely kiszámolja az iteráció első néhány lépését! Módosítsa a kódot úgy, hogy olyan k értékre álljon le, amelyre $|x_k - x_{k-1}| < \varepsilon$, ahol $\varepsilon > 0$ adott.

Numerikus matematika

Baran Ágnes

Optimalizálás 1.

Egyváltozós függvények minimalizálása

Az `fminbnd` függvény:

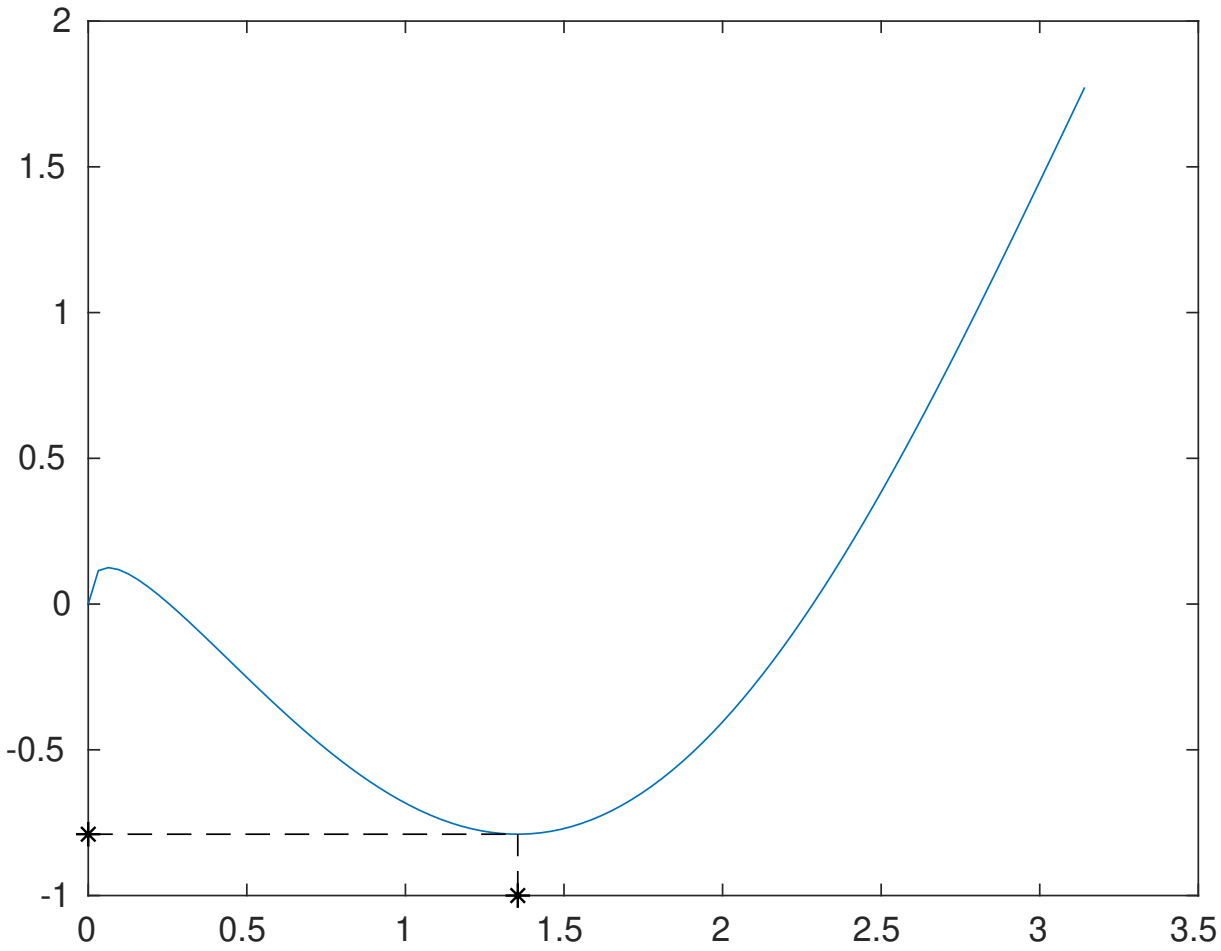
```
[xopt,fopt]=fminbnd(f,a,b)
```

Az f függvény $[a, b]$ intervallumbeli egyik lokális minimumhelyének és minimumának közelítését adja.
`xopt`: a lokális minimumhely közelítése
`fopt`: a lokális minimum közelítése

Példa

Keressük meg az $f(x) = \sqrt{x} - 2\sin(x)$ függvény $[0, \pi]$-beli minimumhelyét.

```
f=@(x) sqrt(x)-2*sin(x);
[xopt,fopt]=fminbnd(f,0,pi)
    xopt =  1.3543
    fopt = -0.78957
```


2
1.5
1
0.5
0
-0.5
-1
0
0.5
1
1.5
2
2.5
3
3.5

Az `fminsearch` és `fminunc` függvények:

- `[xopt,fopt]=fminsearch(f,x0)`
- `[xopt,fopt]=fminunc(f,x0)`

Az f függvény lokális minimumhelyének közelítését (`xopt`) és minimumának közelítését (`fopt`) adja, az `x0` kezdőpontból indítva a keresést.

Mindkettő alkalmas többváltozós függvények minimalizálására is.

```
f=@(x) sqrt(x)-2*sin(x);
[xopt,fopt]=fminsearch(f,0.5)
   xopt =  1.3542
   fopt = -0.78957
xopt,fopt]=fminunc(f,0.5)
   xopt =  1.3543
   fopt = -0.78957
```

Az f függvény maximumát megkereshetjük úgy, hogy a $-f$ függvény minimumát keressük.

Kezdeti közelítésre pl. a függvény ábrázolásával tehetünk szert.

1. feladat

(a) Határozza meg az $f(x) = x^2 + \cos(3x)$ függvény összes $[0, 6]$ intervallumbeli lokális minimumhelyét.

(b) Határozza meg az $f(x) = x^2 + \cos(3x)$ függvény összes $[0, 6]$ intervallumbeli lokális maximumhelyét.

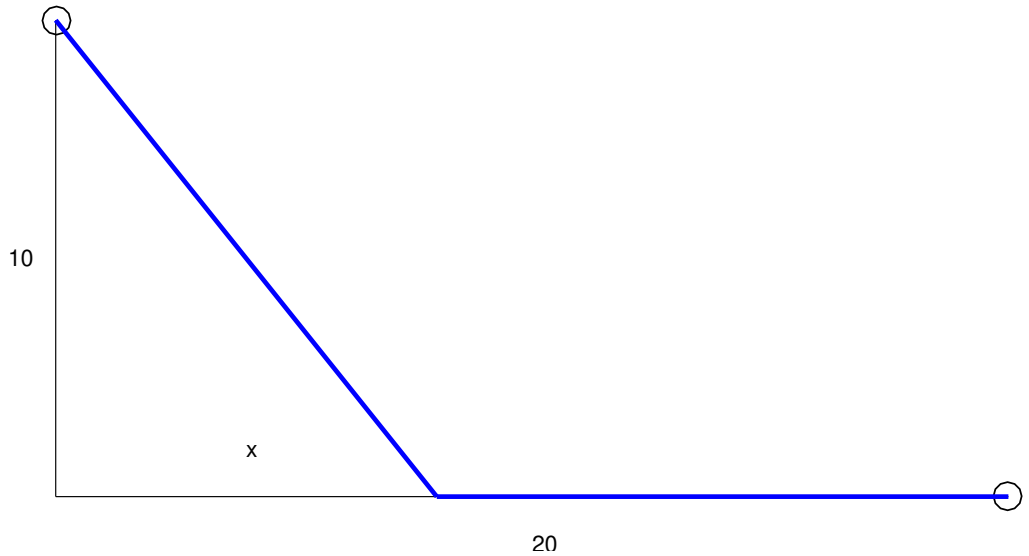
2. feladat

(a) Határozza meg az $f(x) = \sin(2x)\sin(3x)$ függvény összes $[0, 5]$ intervallumbeli lokális minimumhelyét.

(b) Határozza meg az $f(x) = \sin(2x)\sin(3x)$ függvény összes $[0, 5]$ intervallumbeli lokális maximumhelyét.

3. feladat

A parttól 10 km-re fekvő sziget áramellátását szeretnénk biztosítani egy olyan áramellátó központból, amely közvetlenül a parton helyezkedik el, 20 km-re a partnak a szigethez legközelebbi pontjától. Ha 250 ezer Ft-ba kerül 1 km víz alatti vezeték elhelyezése, és 100 ezerbe 1 km vezeték telepítése a szárazföldön, akkor határozzuk meg a minimális költségű útvonalat.



4. feladat

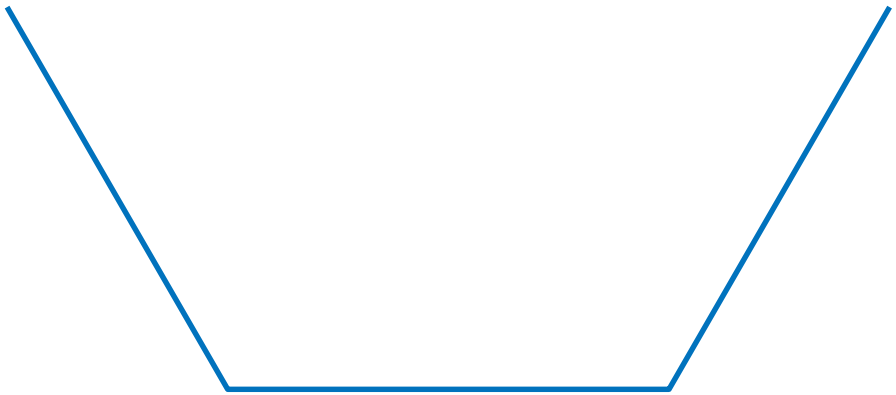
Egy 1 l űrtartalmú, henger alakú konzervdobozt szeretnénk készíteni. Határozza meg a doboz méreteit úgy, hogy adott vastagságú lemezből készítve a lehető legkevesebb anyagra legyen szükség az elkészítéséhez.

5. feladat

Egy 15 cm-szer 20 cm-es kartonlapból egy fedél nélküli dobozt szeretnénk hajtogatni (a lap 4 sarkából 1-1 négyzetet kivágva, a keletkező ,,füleket" felhajtva). Adja meg a doboz méretét úgy, hogy annak térfogata maximális legyen.

6. feladat

Egy 30 cm széles lemezből szeretnénk csatornát hajtogatni úgy, hogy a lemez két szélén 10-10 cm-t valamilyen szögben felhajtunk. Határozza meg a szöget úgy, hogy a csatornába a lehető legtöbb víz férjen.



Alkalmazott matematika

Baran Ágnes

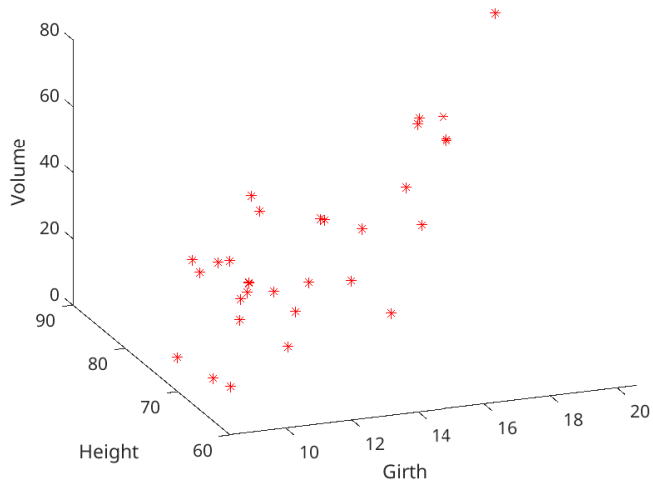
Matlab alapok
Grafika 2.

Pontok, vonalak 3 dimenzióban: plot3

A `plot3` használata hasonló a `plot` használatához, csak a pontok koordinátáit 3 vektorban kell megadni.

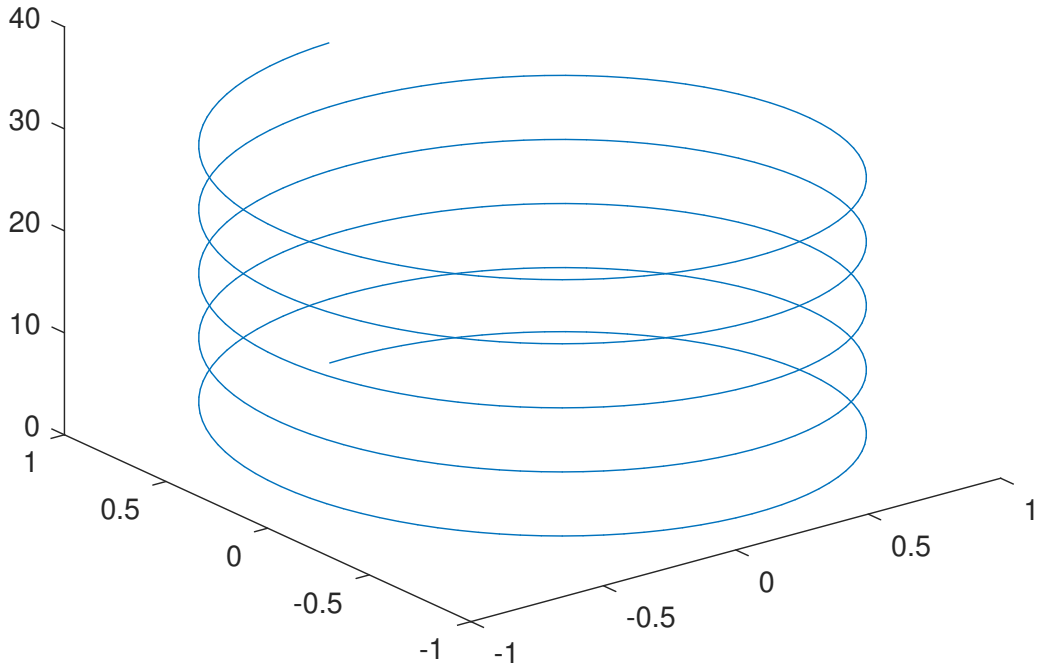
Olvassuk be a `trees.xlsx` állományt és ábrázoljuk a Volume értékeket a Girth és Height értékek függvényében. Lássuk el a tengelyeket a megfelelő cimkékkel.

```
T=readtable('trees.xlsx');
figure; plot3(T.Girth,T.Height,T.Volume,'r*')
xlabel('Girth'); ylabel('Height'); zlabel('Volume')
```



Pontok, vonalak 3 dimenzióban: plot3

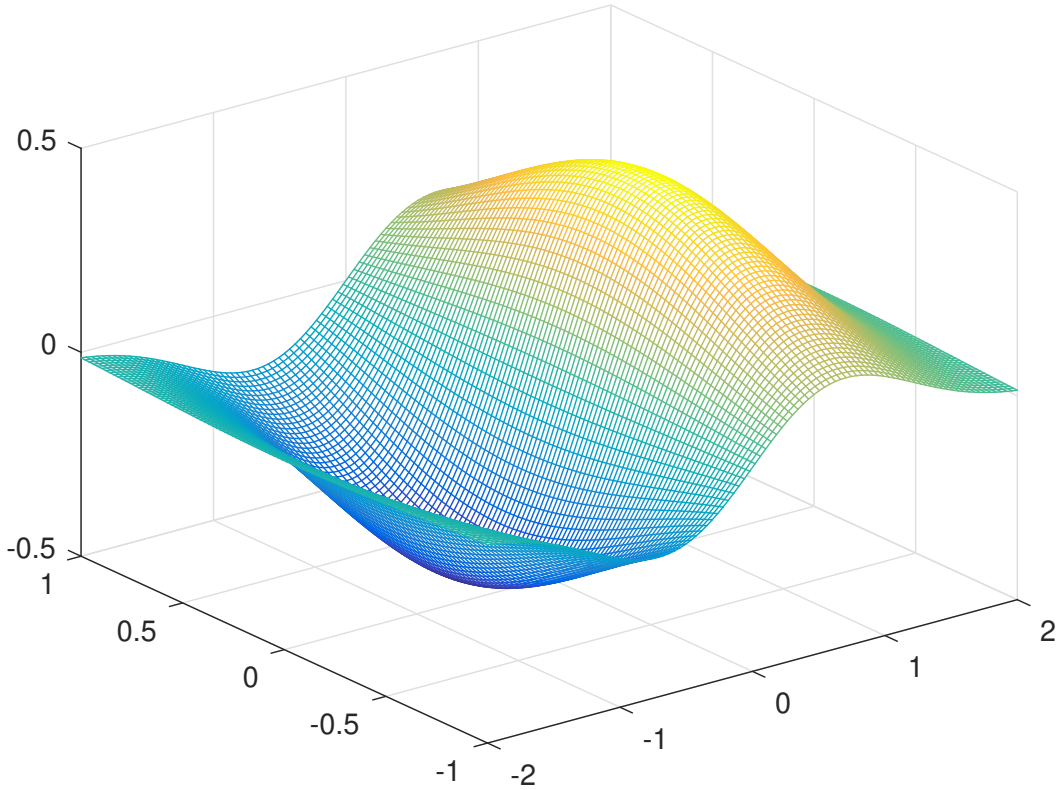
```
t = 0:pi/50:10*pi;
st = sin(t);
ct = cos(t);
figure; plot3(st,ct,t)
```



Felületek: fmesh

Példa

Ábrázoljuk az $f(x,y) = xe^{-x^2-y^2}$ függvényt a $[-2,2] \times [-1,1]$ tartomány felett!



Felületek: fmesh

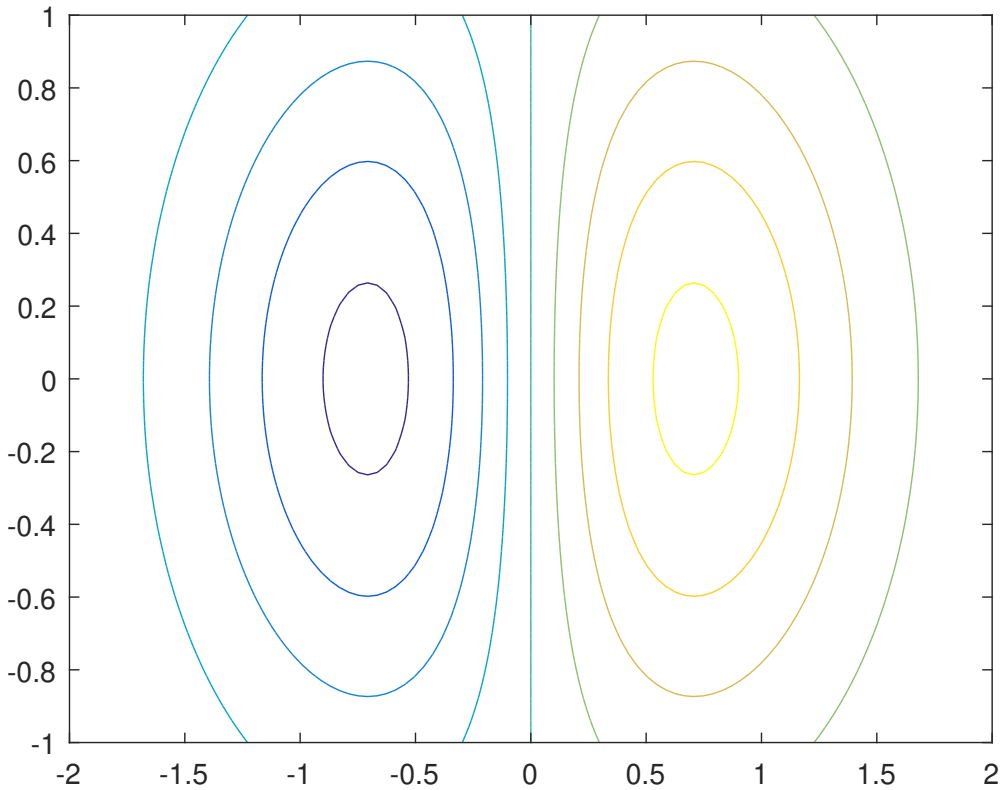
- Definiáljuk a függvényt egy function handle típusú változóként.
- Hívjuk meg az `fmesh` függvényt, első argumentumában az ábrázolni kívánt függvénnyel, második argumentumában (opcionális) a változókra vonatkozó határokkal. Ha a második argumentum hiányzik, akkor a tartomány a $[-5,5] \times [-5,5]$.

```
f = @(x,y) x.*exp(-x.^2-y.^2);
figure; fmesh(f,[-2,2,-1,1])
```

Szintvonalak: fcontour

Hívása megegyezik az `fmesh` hívásával.

```
f = @(x,y) x.*exp(-x.^2-y.^2);
figure; fcontour(f,[-2,2,-1,1])
```



Felületek: mesh

A mesh használatához előbb "be kell rácsoznunk" a sík egy tartományát, ehhez a `meshgrid` függvényt használhatjuk, pl.:

```
>> x=0:15; y=0:10;
>> [X,Y]=meshgrid(x,y);
```

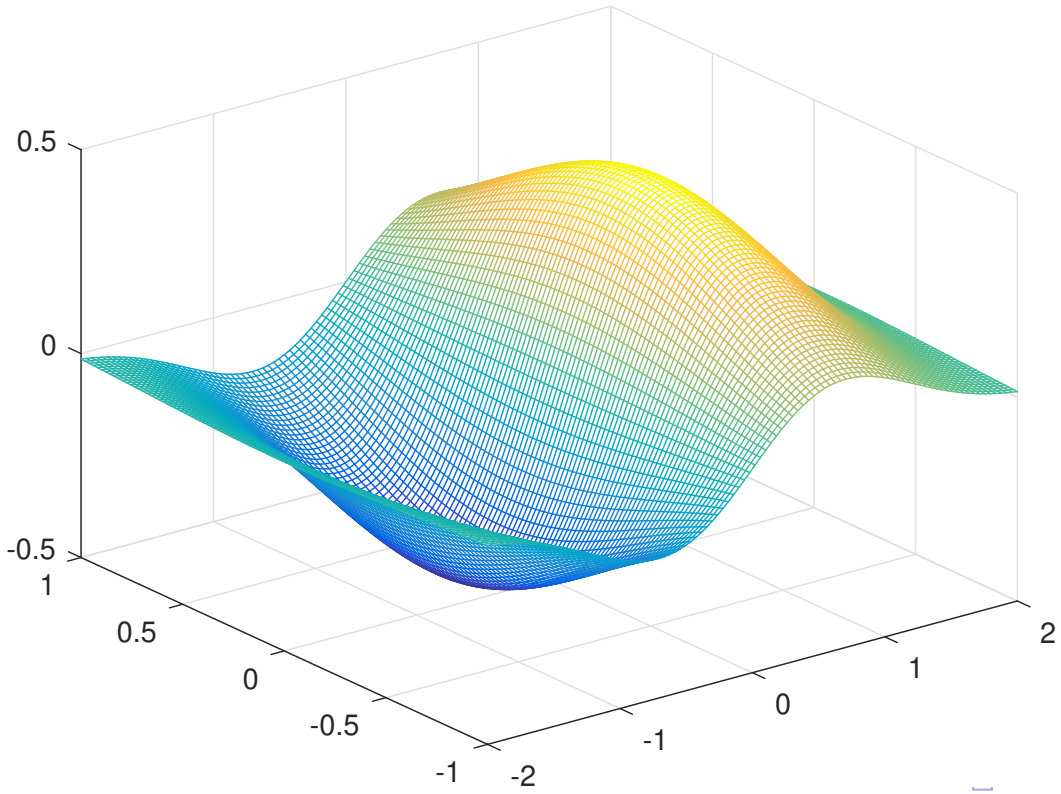
Ekkor `X` és `Y` is 11×16-os mátrix:

$$X = \begin{bmatrix} 0 & 1 & \dots & 14 & 15 \\ 0 & 1 & \dots & 14 & 15 \\ \vdots & & & & \\ 0 & 1 & \dots & 14 & 15 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 1 & \cdots & 1 & 1 \\ \vdots & & & & \\ 10 & 10 & \dots & 10 & 10 \end{bmatrix}$$

(Az `X` és `Y` mátrixokat ,,egymásra helyezve" megkapjuk az összes lehetséges (x_i, y_j) párt)
Ezután kiszámoljuk az (X_i, Y_i) pontokban a függvény értékét és ábrázoljuk (pl a `mesh` vagy `surf` függvénnyel)

Felületek: mesh

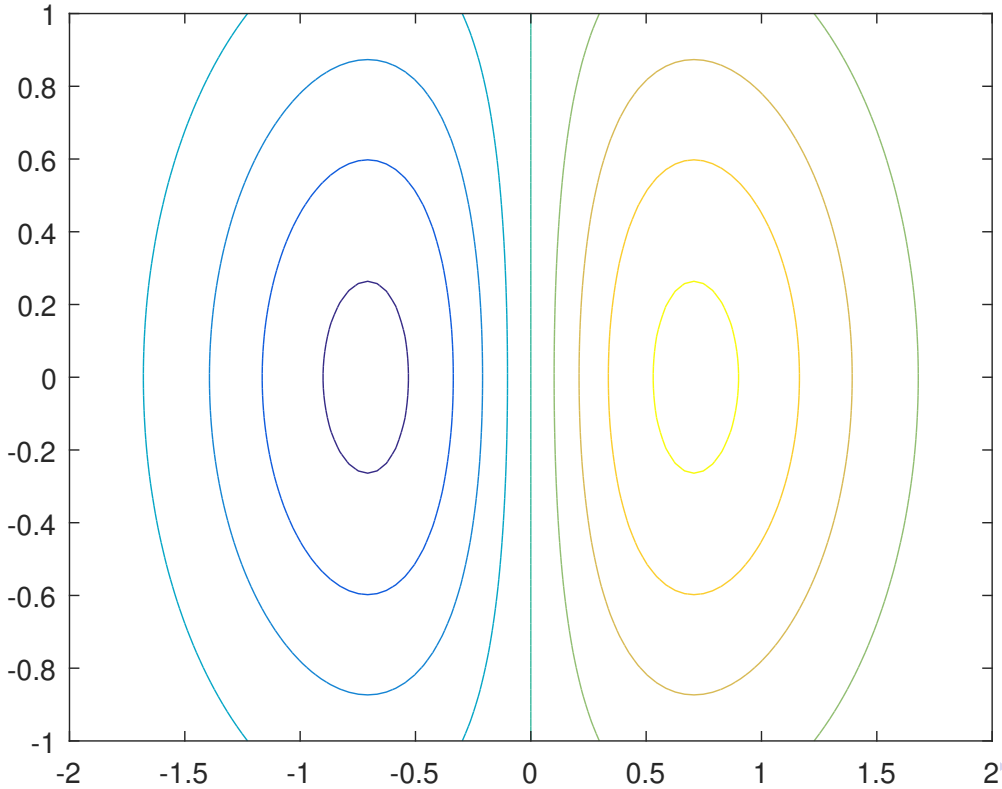
```
x=linspace(-2,2);
y=linspace(-1,1);
[xx,yy] = meshgrid(x,y);
zz = xx.*exp(-xx.^2-yy.^2);
figure; mesh(xx,yy,zz)
```



Szintvonalak: contour

Használata megegyezik a `mesh` használatával.

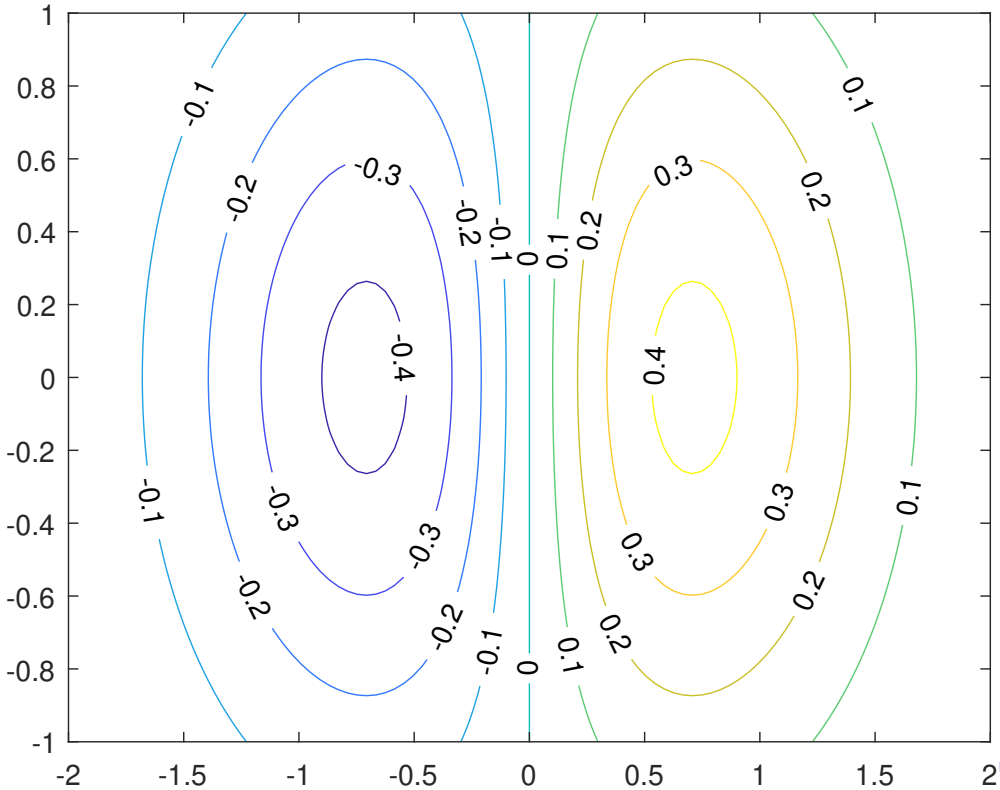
```
x=linspace(-2,2);
y=linspace(-1,1);
[xx,yy] = meshgrid(x,y);
zz = xx.*exp(-xx.^2-yy.^2);
figure; contour(xx,yy,zz)
```



Szintvonalak: contour

A szintvonalakra ráírathatjuk a megfelelő függvényértékeket is.

```
x=linspace(-2,2);
y=linspace(-1,1);
[xx,yy] = meshgrid(x,y);
zz = xx.*exp(-xx.^2-yy.^2);
figure; contour(xx,yy,zz,'ShowText','on')
```

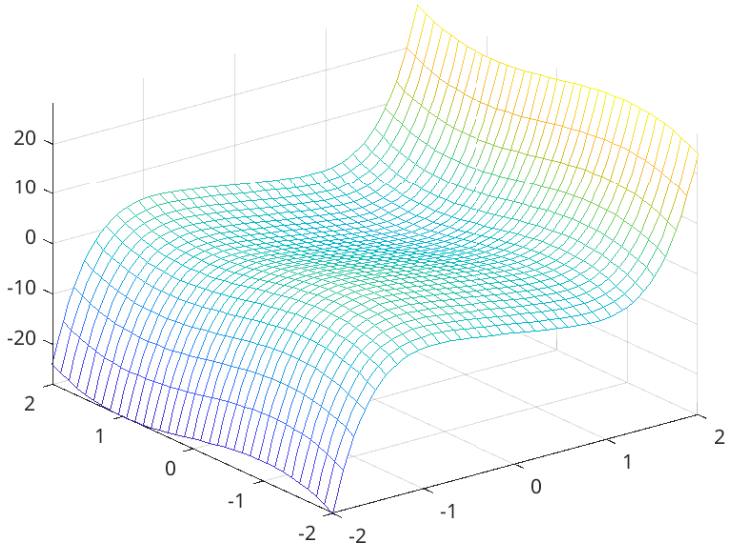


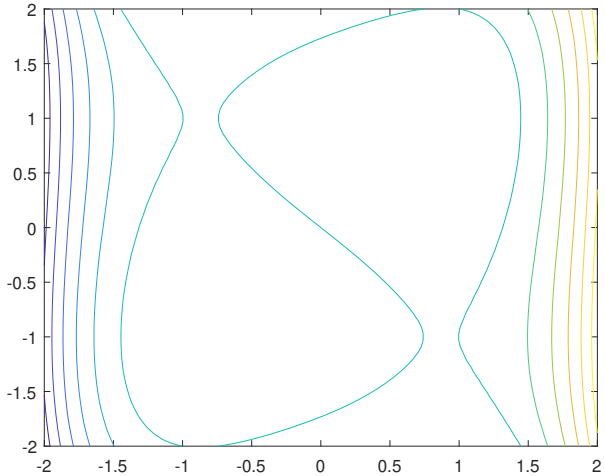
Szintvonalak

Példa

Ábrázoljuk az $f(x,y) = x^5 + y^3 - 3x - 3y$ függvényt, illetve a szintvonalait a $[-2,2]^2$ tartományon.

```
f=@(x,y) x.^5+y.^3-3*x-3*y;
figure;fmesh(f,[-2,2])
figure;fcontour(f,[-2,2])
```

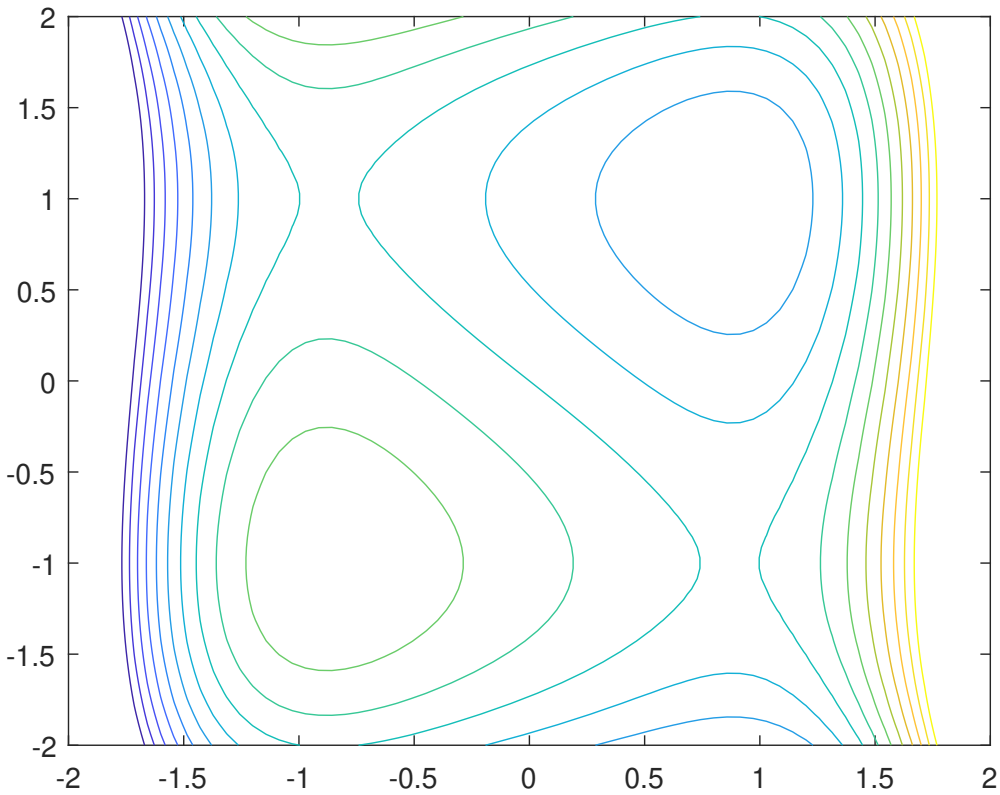




Szintvonalak

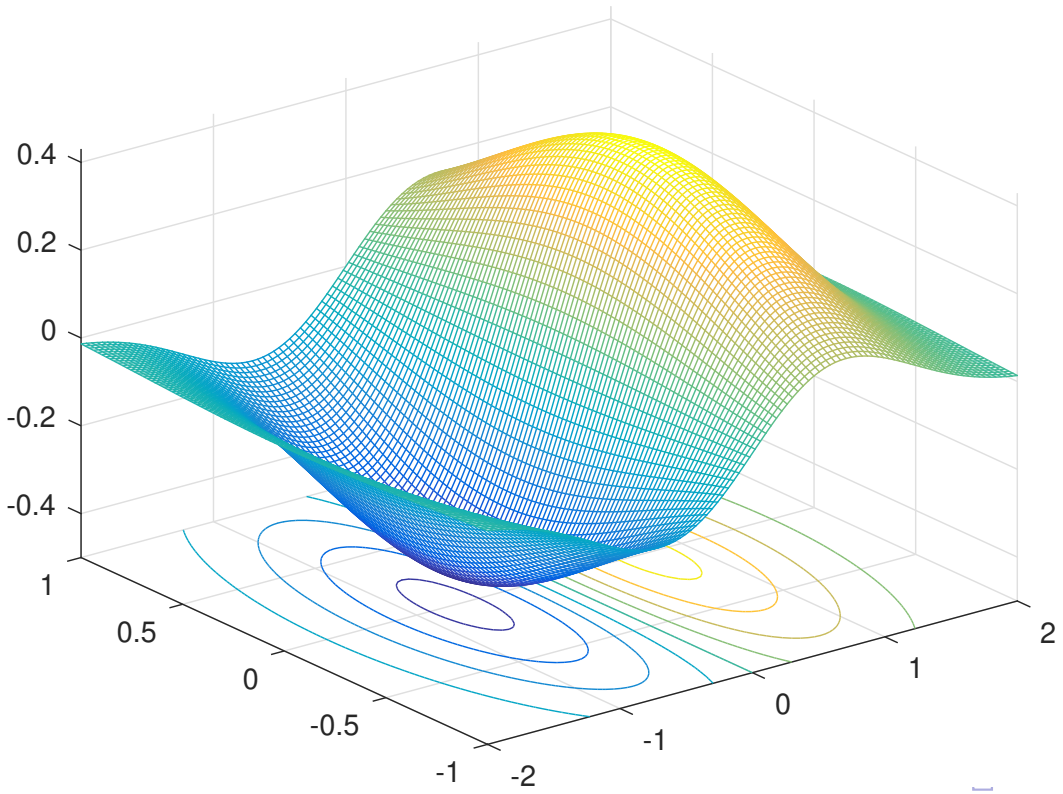
Mivel a függvény a tartomány közepén sokkal lassabban változik, mint a szélén, így a szintvonalak ott ,,ritkák". A `contour` függvényben előírhatjuk milyen függvényértékekhez kérjük a szintvonalakat.

```
figure;fcontour(f,[-2,2],'LevelList',linspace(-10,10,15))
```



Felületek és szintvonalak: meshc

```
x=linspace(-2,2);
y=linspace(-1,1);
[xx,yy] = meshgrid(x,y);
zz = xx.*exp(-xx.^2-yy.^2);
figure; meshc(xx,yy,zz)
```



Numerikus matematika labor

Baran Ágnes

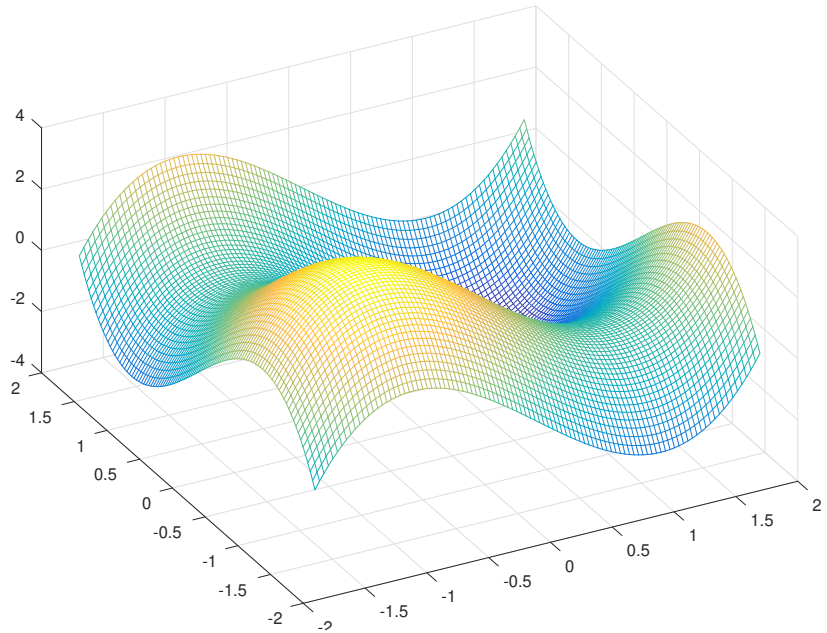
Optimalizálás 2.

Felületek ábrázolása

Példa

Rajzoltassuk ki az $f : \mathbb{R}^2 \to \mathbb{R}$, $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ felületet a $[-2, 2] \times [-2, 2]$ tartomány felett.

```
xx=linspace(-2,2);
yy=xx;
[X,Y]=meshgrid(xx,yy);
Z=X.^3+Y.^3-3*X-3*Y;
figure; mesh(X,Y,Z)
```

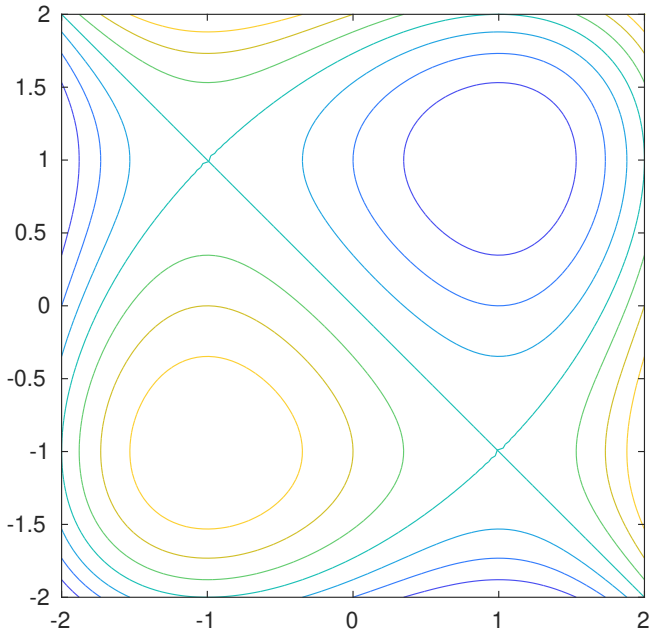


Megjegyzés: a `mesh` helyett használhattuk volna az `fmesh` függvényt is (akkor az ábrázolandó függvényt másképp kell definiálni).

Példa

Rajzoltassuk ki az $f : \mathbb{R}^2 \to \mathbb{R}$, $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ felület szintvonalait a $[-2, 2] \times [-2, 2]$ tartomány felett.

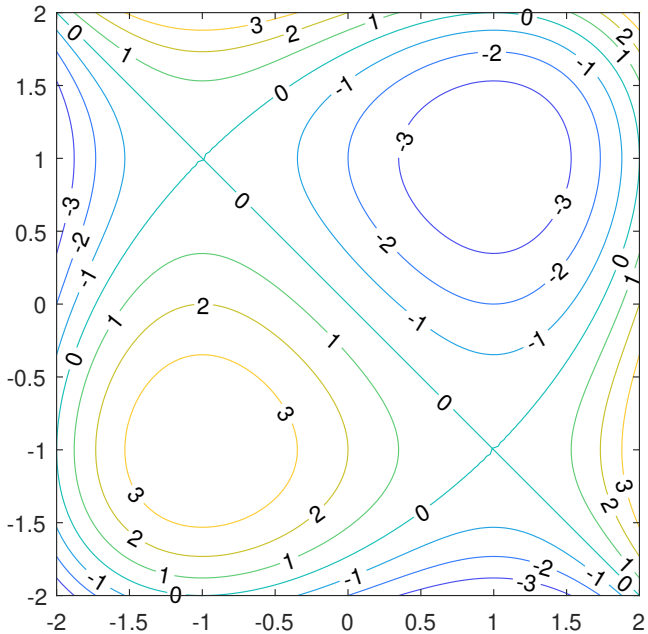
```
xx=linspace(-2,2);
yy=xx;
[X,Y]=meshgrid(xx,yy);
Z=X.^3+Y.^3-3*X-3*Y;
figure; contour(X,Y,Z)
axis equal
```



Megjegyzés: a `contour` helyett használhattuk volna az `fcontour` függvényt is (akkor az ábrázolandó függvényt másképp kell definiálni).

A szintvonalakra ráírathatjuk a ,,magassági számokat" is:

```
xx=linspace(-2,2);
yy=xx;
[X,Y]=meshgrid(xx,yy);
Z=X.^3+Y.^3-3*X-3*Y;
figure; contour(X,Y,Z,'ShowText','on')
axis equal
```



Többváltozós függvények minimalizálása

Példa

Határozzuk meg az $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ függvény gradiensét és a stacionárius pontjait.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 3x_1^2 - 3 \\ 3x_2^2 - 3 \end{bmatrix}$$

Stacionárius pont: ahol $\nabla f(x) = 0$.

A függvénynek 4 stacionárius pontja van:

$$(-1,-1), \quad (-1,1), \quad (1,-1), \quad (1,1)$$

Példa

Rajzoltassuk ki az $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ függvény szintvonalait és a gradiens mezőt a $[-2, 2] \times [-2, 2]$ tartomány felett.

A gradiens mező rajzolásához nagyobb beosztású rácsot használjunk, pl. itt 11-11 pontot veszünk fel mindkét tengelyen:

```
xx=linspace(-2,2,11); yy=xx;
[X,Y]=meshgrid(xx,yy);
```

Ezekben a pontokban meg kell adnunk a gradiensvektor koordinátáit:

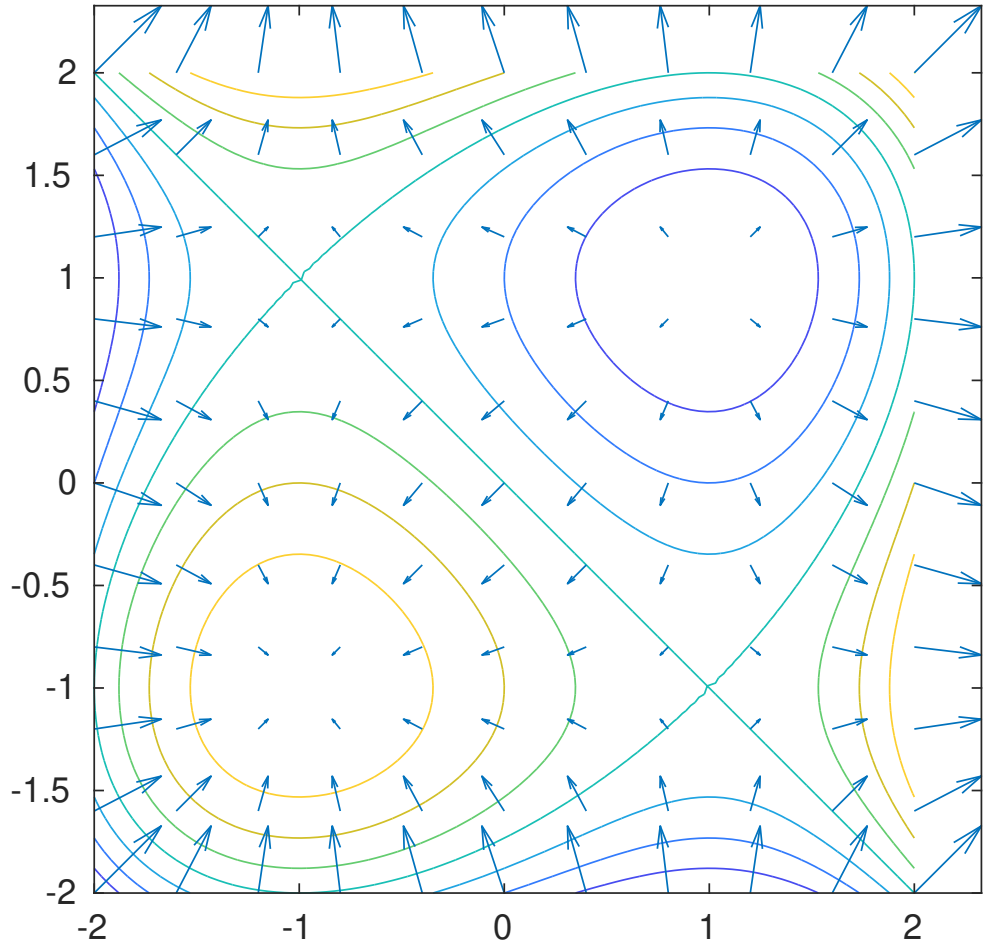
```
dX=3*X.^2-3;
dY=3*Y.^2-3;
```

A nyilak kirajzolásához a `quiver` függvényt használjuk. (Első két input érték a nyilak talppontjának x- és y-koordinátái, második két input érték a nyilak hegyének x- és y-koordinátái.

```
quiver(X,Y,dX,dY)
```

A szintvonalak és a gradiensmező egyben:

```
%a szintvonalak
xx=linspace(-2,2); yy=xx;
[X,Y]=meshgrid(xx,yy);
Z=X.^3+Y.^3-3*X-3*Y;
figure; contour(X,Y,Z)
axis equal
%a gradiensmezo
xx=linspace(-2,2,11); yy=xx;
[X,Y]=meshgrid(xx,yy);
dX=3*X.^2-3;
dY=3*Y.^2-3;
hold on; quiver(X,Y,dX,dY)
```

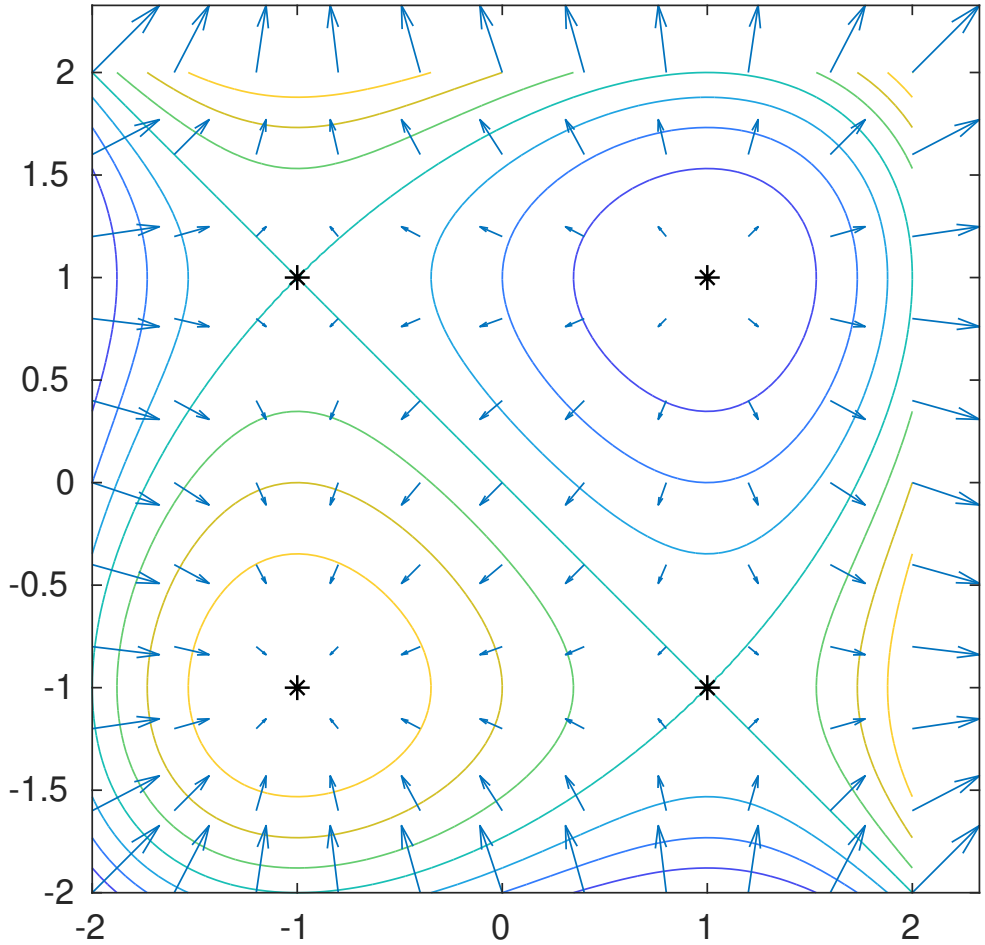


Az $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ függvény szintvonalai és a gradiensmező.

A gradiensmező kirajzolásához használhatjuk a beépített `gradient` függvényt is. (Ekkor nem kell kiszámolnunk a gradienst, a Matlab numerikusan közelíti azt)

```
%a szintvonalak
xx=linspace(-2,2); yy=xx;
[X,Y]=meshgrid(xx,yy);
Z=X.^3+Y.^3-3*X-3*Y;
figure; contour(X,Y,Z)
axis equal

%a gradiensmezo
xx=linspace(-2,2,11); yy=xx;
[X,Y]=meshgrid(xx,yy);
Z=X.^3+Y.^3-3*X-3*Y;
[dX,dY]=gradient(Z);
hold on; quiver(X,Y,dX,dY)
```

Tegyük rá az ábrára a stacionárius pontokat is!

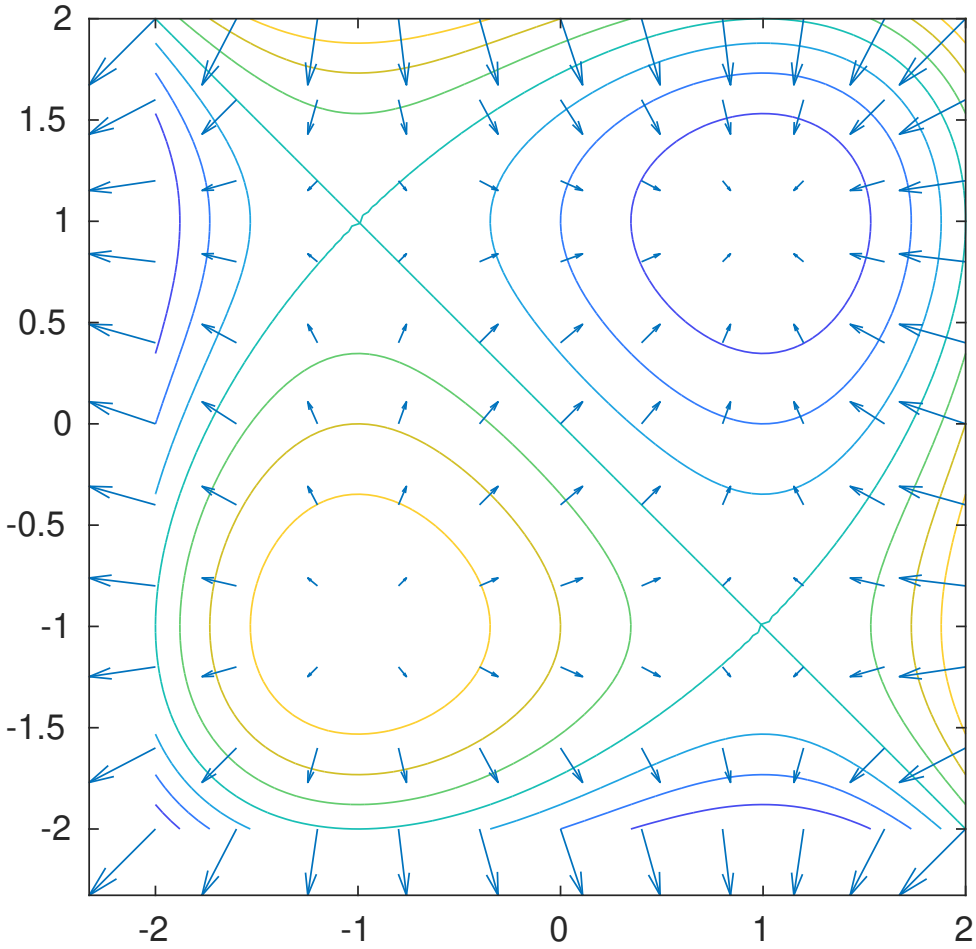


Az előző ábrán megfigyelhetjük, hogy

- a gradiensvektor merőleges az adott pontbeli szintvonalra
- a vektorok hossza a gradiens nagyságát, az iránya a gradiens irányát mutatja
- a stacionárius pontokban a gradiensvektor hossza 0

A gradiensvektor az adott pontban a legmeredekebb emelkedés irányába mutat, a (-1)-szerese (a negatív gradiens) pedig a legmeredekebb csökkenés irányába.

Ha a gradiensmező helyett a negatív gradiensmezőt rajzoltatjuk ki, akkor a nyilak a csökkenés irányába mutatnak.



Az $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ függvény szintvonalai és a negatív gradiensmező.

Példa

Határozzuk meg az $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ függvény stacionárius pontjainak típusát!

A függvény Hesse-mátrixa:

$$H(x) = \begin{bmatrix} 6x_1 & 0 \\ 0 & 6x_2 \end{bmatrix}$$

Ha $x = (-1, -1)$, akkor

$$H(x) = \begin{bmatrix} -6 & 0 \\ 0 & -6 \end{bmatrix},$$

így $\Delta_1 < 0$, $\Delta_2 > 0$, tehát ez a pont lokális maximumhely.

Ha $x = (-1, 1)$, akkor

$$H(x) = \begin{bmatrix} -6 & 0 \\ 0 & 6 \end{bmatrix},$$

így $\Delta_1 < 0$, $\Delta_2 < 0$, tehát ez a pont nyeregpont.

Ha $x = (1, -1)$, akkor

$$H(x) = \begin{bmatrix} 6 & 0 \\ 0 & -6 \end{bmatrix},$$

így $\Delta_1 > 0$, $\Delta_2 < 0$, tehát ez a pont nyeregpont.

Ha $x = (1, 1)$, akkor

$$H(x) = \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix},$$

így $\Delta_1 > 0$, $\Delta_2 > 0$, tehát ez a pont lokális minimumhely.

1. feladat

Számolja ki az alábbi függvények gradiensét és a stacionárius pontjaikat. Rajzoltassa ki a felületeket, illetve egy másik ábrán a függvény szintvonalait és a negatív gradiens mezőt. Ezek alapján mit tud mondani a stacionárius pontok típusáról?

(a) $f(x_1, x_2) = 10 - x_1^2 - x_2^2$

(b) $f(x_1, x_2) = x_1^2 x_2 - 2x_1 x_2^2 + 3x_1 x_2 + 2$

(c) $f(x_1, x_2) = x_1^3 - x_1^2 x_2^2 - x_1 + x_2^2$

Többváltozós függvények minimalizálása Matlab-bal

Az `fminsearch` vagy az `fminunc` függvényeket használhatjuk.

Példa: Keressük meg az $f(x) = x_1^3 + x_2^3 - 3x_1 - 3x_2$ függvény egy lokális minimumhelyét.

Mindkét függvény hívásához meg kell adnunk a minimumhely egy kezdeti közelítését.

```
>> f=@(x) x(1)^3+x(2)^3-3*x(1)-3*x(2);
>> [xopt,fopt]=fminsearch(f,[0.5,0.5])

xopt =
    0.99996    1.00000

fopt =
   -4.0000
```

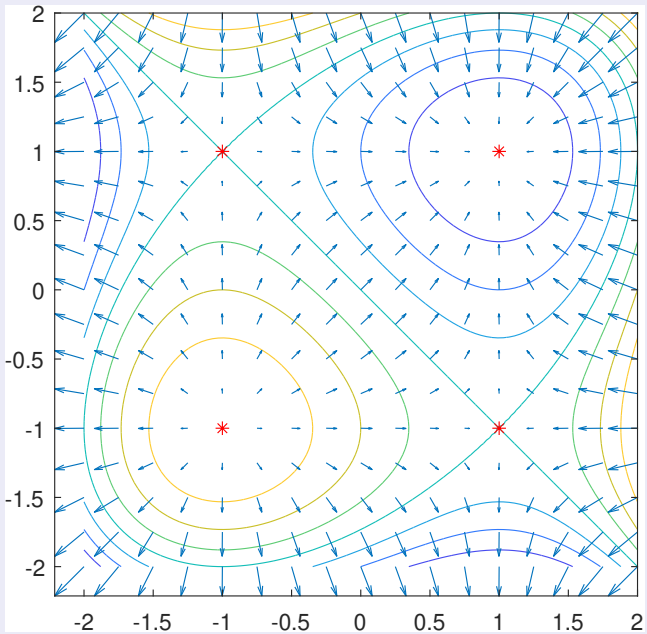
Az `fminunc` függvénnyel:

```
>> f=@(x) x(1)^3+x(2)^3-3*x(1)-3*x(2);
>> [xopt,fopt]=fminunc(f,[0.5,0.5])

xopt =
   1.0000   1.0000

fopt = -4.0000
```

A kezdővektor megválasztása erősen befolyásolhatja az algoritmus sikeres lefutását.

2. feladat

Vizsgáljuk meg milyen eredményt kapunk az `fminunc` `fminsearch` függvényeket az $f(x) = x_1^3 - 3x_1 + x_2^3 - 3x_2$ függvénnyel és az alábbi kezdővektorokkal meghívva:



$$x_0 = [-0.5, -0.5],$$

$$x_0 = [-0.5, 0],$$

$$x_0 = [-1, -0.5],$$

$$x_0 = [-1.5, -1.5]$$

3. feladat

Matlab segítségével határozza meg azt a téglatestet, melynek térfogata 1000 cm^3 és éleinek összhossza minimális.

4. feladat (szorgalmi)

Egy cég 20000 $-t költött egy új termék kifejlesztésére. A termék előállítási költsége darabonként 2 $. Egy piackutató szerint, ha a cég R $-t költene reklámra, és ezután a terméket darabonként A $-ért árulná, akkor a kereslet

$$2000 + 4\sqrt{R} - 20A$$

darab lenne. Mennyit érdemes reklámra költeni, és milyen áron érdemes kínálni a terméket, ha a hasznot maximalizálni szeretnék? Használja a Matlab-ot!

5. feladat

Rajzoltassa ki a megadott tartomány felett az alábbi kétváltozós függvényeket, a szintvonalaikat, a negatív gradiensmezőt és közelítse az adott tartományon belül a minimumhelyüket.

- $f(x_1, x_2) = \frac{1}{6}x_1^3 - x_1 + \frac{1}{4}x_1x_2^2$, ha $x \in [-2.5, 2.5]^2$
- $f(x_1, x_2) = \sin(x_1)\cos(x_2)$, ha $x \in [0, 2\pi) \times [0, 2\pi)$
- $f(x_1, x_2) = x_2(1 - x_1^2 - x_2^2)$, ha $x \in [-1.5, 1.5]^2$

6. feladat (szorgalmi)

Egy téli üdülőövezetben a mentőhelikopter bázisállomását úgy szeretnénk elhelyezni, hogy az n adott síközponttól mért legnagyobb távolsága minimális legyen. Írjon egy Matlab-függvényt, melynek input paramétere az az $A \in \mathbb{R}^{n\times 2}$ mátrix, melynek soraiban a síközpontok koordinátái találhatóak, output paramétere pedig a mentőhelikopter bázisállomásának koordinátáit tartalmazó kételemű vektor.

7. feladat (szorgalmi)

Adott egy bolthálózat n üzletének elhelyezkedése. Helyezzük el az áruraktárat úgy, hogy az üzletektől vett távolságainak összege minimális legyen. Írjon egy Matlab-függvényt, melynek input paramétere az az $A \in \mathbb{R}^{n\times 2}$ mátrix, melynek soraiban az üzletek koordinátái találhatóak, output paramétere pedig az áruraktár koordinátáit tartalmazó kételemű vektor.

Numerikus matematika

Baran Ágnes

Numerikus integrálás

Numerikus integrálás Matlab-bal

Egyváltozós függvények integrálására pl az `integral` függvényt használhatjuk.

Példa

Matlab segítségével számítsuk ki az

$$\int_0^3 x\sqrt{1+x}dx$$

integrál értékét!

Megoldás.

```
>> f= @(x) x.*sqrt(1+x);
>> integral(f,0,3)
ans=
    7.7333
```

Az `integral` függvény hívása:

```
>> integral(fv,xmin,xmax)
```

ahol `fv` az integrálandó függvény (`fv` egy function handle típusú változó), `xmin` és `xmax` az alsó és felső határ.

Az `integral` függvény az `fv` függvényt vektor argumentummal fogja meghívni. Figyeljünk rá, hogy az `fv` ennek megfelelően legyen megadva (elemenkénti operátorok!).

Az `integral` függvény adaptív kvadratúrát használ, és alapértelmezésként 10^{-10} abszolút, vagy 10^{-6} relatív hibával számítja ki az integrál értékét.

A hibahatárok átállíthatóak:

```
>> integral(f,0,3,'RelTol',1e-8,'AbsTol',1e-13)
```

Az előző példában nem feltétlenül szükséges külön létrehozni a `f` változót:

```
>> integral(@(x) x.*sqrt(1+x),0,3)
```

Ha a függvényt korábban egy m-fájlban definiáltuk, pl.

```
function y=myfnc(x)
   y=x.*sqrt(1+x)
end
```

akkor az `integral` függvénynek átadhatjuk a függvény nevét is (function handle-ként):

```
>> integral(@myfnc,0,3)
```

Hasonló a helyzet a Matlab beépített függvényeivel:

```
>> integral(@sin,0,pi)
```

Impropius integrálok

- Az integrálás határai lehetnek $-\infty$ és ∞ is:

```
>> f= @(x) exp(-x);
>> integral(f,0,inf)
ans =  1.0000
```

- Az sem probléma, ha a függvény az intervallum végpontjaiban nincs értelmezve:

```
>> f= @(x) 1./sqrt(1-x.^2);
>> integral(f,-1,1)
ans=
   3.1416
```

Ha nem ismert a függvény

Előfordulhat, hogy nem ismerünk egzakt képletet az integrálandó függvényre, csak bizonyos pontokban ismerjük az értékeit. Ilyenkor a `trapz` Matlab-függvényt használhatjuk.

Példa

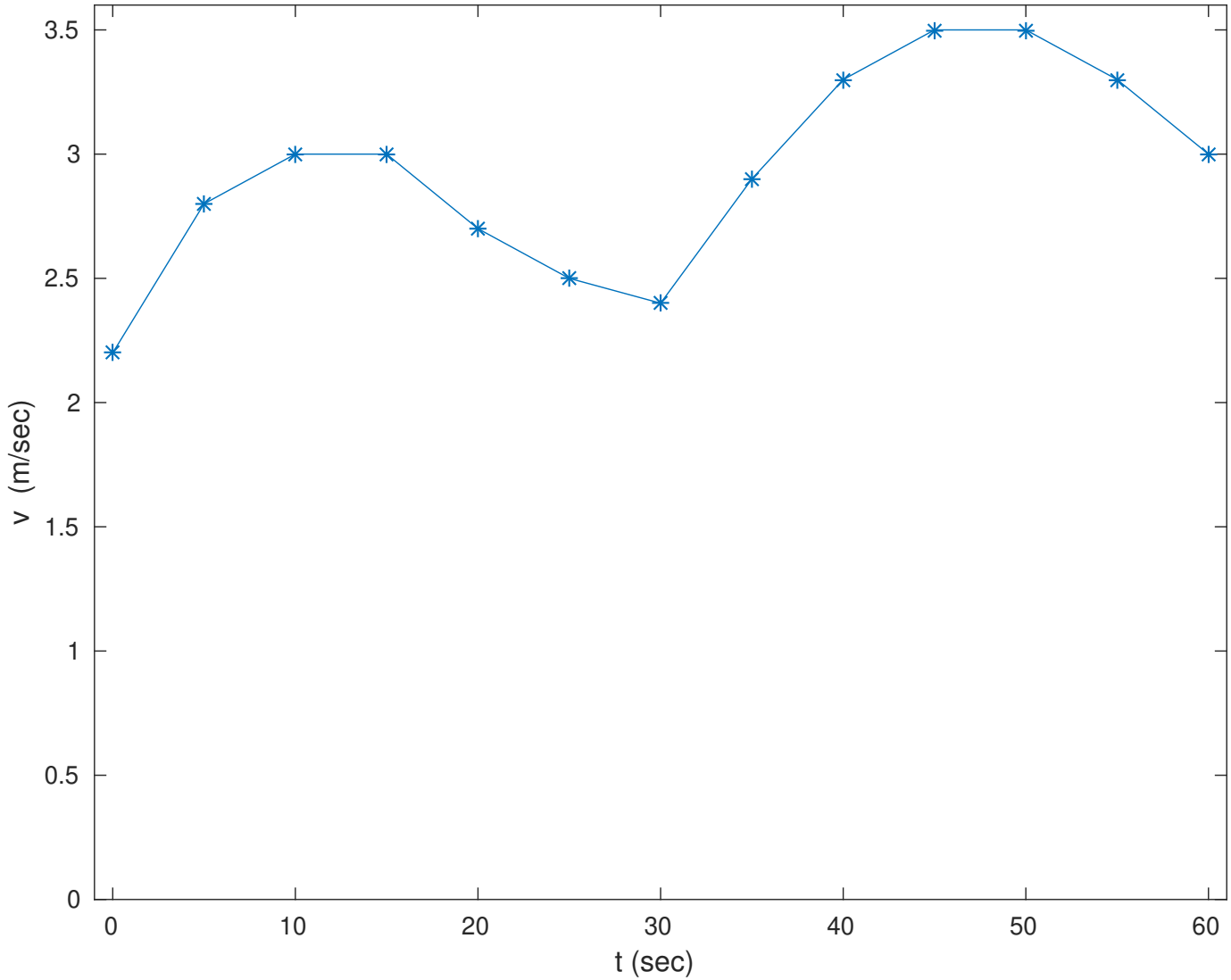
Egy jármű sebességét 1 percen kereszül mértük 5 másodperces időközönként:

t (sec)	0	5	10	15	20	25	30	35	40	45	50	55	60
v (m/sec)	2.2	2.8	3	3	2.7	2.5	2.4	2.9	3.3	3.5	3.5	3.3	3

Becsüljük meg a jármű által megtett utat!

Megoldás. Tudjuk, hogy az a idő alatt megtett út:

$$S = \int_0^a v(t)dt$$


3.5
3
2.5
2
1.5
1
0.5
0
v (m/sec)
0
10
20
30
40
50
60
t (sec)

A `trapz` függvény segítségével az integrál becslése:

```
>> x=0:5:60;
>> f=[ 2.2 2.8 3 3 2.7 2.5 2.4 2.9 3.3 3.5 3.5 3.3 3];
>> trapz(x,f)
ans =
     177.5000

>> y=cumtrapz(x,f);
```

Ekkor

$$y = (0, 12.5, 27, 42, 56.25, 69.25, 81.5, 94.75, 110.25, 127.25, 144.75, 161.75, 177.5)$$

Az `y` vektor i-edik koordinátája az i-edik időpillanatig megtett utat mutatja.

1. feladat

Matlab segítségével számítsa ki az alábbi határozott integrálok értékét!

(a)

$$\int_{-\pi/2}^{\pi/2} x \sin(x^2) dx$$

(b)

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$$

(c)

$$\int_{-1}^{1} \sqrt{1-x^2} dx$$

2. feladat

Közelítse az

$$\int_0^{10} x \sin(5x) dx$$

integrált az `integral` függvénnyel, illetve a `trapz` függvénnyel úgy, hogy alappontoknak az

- `xi=0:10` pontokat
- `xi=[0 0.5:9.5 10]` pontokat

választja. Próbálja megmagyarázni a tapasztalt jelenséget (ábrázolja az integrálandó függvényt a megadott intervallum felett). Növelje az alappontok számát a `trapz` függvény esetén.

3. feladat

Írjon 1-1 Matlab függvényt, mely adott f, a, b és m esetén kiszámítja

$$\int_a^b f(x)dx$$

közelítését összetett trapéz, illetve összetett Simpson-képlettel, akkor, amikor az $[a, b]$ intervallumot m részintervallumra osztjuk.