

Előző gyakorlat - Fel - Következő gyakorlat

## Tartalomjegyzék

- 1 MatLab
  - ◆ 1.1 Kezdeti lépések
    - ◇ 1.1.1 Hozzáférés a programhoz
    - ◇ 1.1.2 Számológép
  - ◆ 1.2 Adattípusok
    - ◇ 1.2.1 A számbábrázolások
    - ◇ 1.2.2 Mátrixok
    - ◇ 1.2.3 Tartományok
  - ◆ 1.3 Műveletek mátrixokkal
    - ◇ 1.3.1 Transzponált
    - ◇ 1.3.2 Összeadás
    - ◇ 1.3.3 Szorzás
    - ◇ 1.3.4 Tagonként vagy mátrixként
  - ◆ 1.4 Változók
  - ◆ 1.5 Indexelés
  - ◆ 1.6 Összegzés
  - ◆ 1.7 Vektorizáció
  - ◆ 1.8 Függvények
  - ◆ 1.9 Feladatok
    - ◇ 1.9.1 Elemi sorműveletek elvégzése
    - ◇ 1.9.2 Mi ez?
    - ◇ 1.9.3 LER
    - ◇ 1.9.4 Még LER
    - ◇ 1.9.5 Nagy mátrix okosan
    - ◇ 1.9.6 Függvény mátrixokon
    - ◇ 1.9.7 Részmátrix
    - ◇ 1.9.8 Részmátrixon függvény
    - ◇ 1.9.9 Minden második oszlop
    - ◇ 1.9.10 Függvény alkalmazás csak adott elemeken
    - ◇ 1.9.11 Segítség magadon
    - ◇ 1.9.12 Numerikus deriválás

## MatLab

Az MatLab program alkalmas különböző matematikai számításokat numerikusan elvégzésére, köztük

- lineáris (mátrix) egyenletek megoldása
- differenciál-egyenletek, megoldása
- szabályozástechnika

Neve a *Matrix Laboratory* rövidítése, a 70-es években kezdte el fejleszteni *Cleve Moler*, a *University of New Mexico* egyetem számítástudományi tanszékének vezetője. Azóta a mathworks cég fizets terméke lett.

Van egy ingyenes alternatívája, az Octave.

## Kezdeti lépések

### Hozzáférés a programhoz

Ha otthonról dolgozunk, akkor a következő lehetőségek legalább egyikével éljünk:

- A BME-től kaphatunk licenst: <https://net.bme.hu/sw/>. Kövessük az utasításokat a honlapon (BME-s hálózaton belülről érhető el).
- Putty-al lépünk be a leibniz-re és írjuk be a terminálba, hogy `matlab`

A Leibniz-en telepítve van, gépteremből is használhatjuk.

### Számológép

Egy terminálban adjuk ki a következő parancsot:

```
matlab -nojvm
```

A MatLab egy fejlettebb számológépként is használható. Írjuk be az alábbiakat:

```
2 + 3
```

majd üssünk Enter-t. Ennek hatására:

```
> 2 + 3  
ans = 5
```

Próbáljuk ki ezeket is:

```
2 - 3  
2 * 3  
2 / 3  
floor (2 / 3)  
mod (2, 3)  
2^3  
sqrt (2)  
log (2)  
log (3)  
log (8) / log (2)  
exp (1)  
pi  
cos (pi / 2)  
(180 / pi) * acos (0.5)
```

Kilépni így lehet

```
exit
```

## Adattípusok

Minden szám alapértelmezésben lebegőpontos, akkor is, ha véletlenül egész:

```
1000 / 9
ans = 111.11
```

Viszont megadhatjuk, hogy egészekként értelmezze a számokat:

```
int32 (1000) / int32 (9)
ans = 111
```

MatLab-ban egy szám mindaddig valós, amíg komplexnek nem bizonyul:

```
sqrt (2)
sqrt (-2)
```

## A számábrázolások

- `double`: dupla lebegő pontos, 64 bit (8 byte)
  - ◆ valós: 8 byte
  - ◆ komplex: 16 byte
- `single`: szimpla lebegő pontos, 32 bit (4 byte)
  - ◆ valós 4 byte
  - ◆ komplex: 8 byte
- `int32`: 32 bites kettes komplement egész (4 byte)
- `int8`: 8 bites kettes komplement egész: -128..127 (1 byte)
- `uint32`: 32 bites előjel nélküli egész (4 byte)
- `uint8`: 8 bites előjel nélküli egész: 0..255 (1 byte)

A méret nagyon is számít:

```
log(single(1.0001))
log(double(1.0001))
int32(100+100)
int8(100+100)
```

## Mátrixok

Az MatLab-ban **minden szám egy mátrix**

- számok: 1x1
- vektorok:
  - ◆ sorvektor: 1xn
  - ◆ oszlopvektor: nx1
- matrix: nxm

Ennek alapos oka van, amit majd később fogunk megérteni és ami a MatLab leglényegéhez vezet bennünket. [Bővebben itt.](#)

Sorvektor:

```
[1 2 3 4]
[1, 2, 3, 4]
```

Oszlopvektor:

```
[1; 2; 3; 4]
```

Ez **nem** oszlopvektor:

```
[[1], [2], [3], [4]]
```

Mátrix:

```
[1 2; 3 4]  
[1, 2; 3, 4]
```

Speciális mátrixok:

- zeros: csupa 0
- ones: csupa 1
- eye: diagonálisban 1, máshol 0
- diag: négyzetes diagonális mátrix, megadott főátlóval

```
zeros (2, 3)  
eye (2, 3)  
ones (3, 1)  
diag ([1, 2, 3, 4])
```

Próbáljuk ki:

```
size (5)  
size ([1, 2, 3])  
size ([1; 2; 3])
```

## Tartományok

A tartományok speciális sorvektorok, próbáljuk ki:

```
1:10
```

Ha nem egyesével akarunk ugrani:

```
1:0.1:2  
1:2:10
```

Komplex számmal nem lehet, mert azok nem rendezhetők!  
Az eredmény mindig `double` lesz, de utána konvertálhatjuk:

```
int32 (1:0.5:10)
```

Leszálló tartományok:

```
4:-1:1
```

Üres tartomány:

```
4:1:1  
4:1
```

## Mátrixok

Diagonális mátrixot megadhatunk így is:

```
> diag(1:4)
ans =
    1    0    0    0
    0    2    0    0
    0    0    3    0
    0    0    0    4
```

## Műveletek mátrixokkal

Mivel minden szám egyben egy  $1 \times 1$ -es mátrix, így ezek mindig használhatóak.

### Transzponált

Transzponált egyszerűen **vesszővel** ( ' ):

```
> [1 2; 3 4] '
ans =
    1    3
    2    4
> _
```

Vagy

```
> (1:4) '
ans =
    1
    2
    3
    4
```

Komplex mátrixokra a vessző adjungálást jelent, azaz a transzponált elemenkénti konjugáltját:

```
> [1 2i; 3i 4] '
ans =
    1 - 0i    0 - 3i
    0 - 2i    4 - 0i
```

Konjugálást így csinálhatunk:  $(2 + 1i)'$

### Összeadás

```
1 + (1:4)
eye(2, 2) + ones(2, 2)
[1; 2; 3; 4] - [4; 3; 2; 1]
```

### Szorzás

Minden szorzás mátrixszorzás:

```
> [1 2; 3 4]*[1 2; 3 4]
ans =
    7    10
   15    22
```

Hatványozás szintén, így az invertálás is:

Tartományok

```
[1 2; 3 4]^2
[1 2; 3 4]^(-1)
```

A szorzásnál a méreteknek kompatibiliseknek kell lenniük:

```
ones (2, 3) * ones (3, 5)
```

Sorvektor szorozva oszlopvektorral a skalárszorzás, fordítva diádszorzatnak hívjuk:

```
[1, 2, 3] * [1; 2; 3]
[1; 2; 3] * [1, 2, 3]
```

## Tagonként vagy mátrixként

Ha a hatványozást ismételt mátrixszorzásként értelmezi, akkor ez mi?

```
[1 2; 3 4]^0.5
```

És ez mi?

```
sqrt([1 2; 3 4])
```

Bizonyos műveletek *tagonként hatnak* ha egy mátrixra alkalmazzuk, míg mások mátrix-műveletként. De tudunk váltani köztük.

```
> (1:4)^2
error: for A^b, A must be a square matrix
```

Hibát ad, mert két 1x4-es mátrixnak nem értelmes a szorzata. De:

```
> (1:4).^2
ans =
    1    4    9   16
```

Minden műveleti jel olyan, hogy **ha elé pontot rakunk, akkor elemenként hat**. Például az összeadásnál a mátrix összeadás és az elemenkénti összeadás ugyan az, nincsen .+ művelet.

```
> [1 2; 3 4] * [1 2; 3 4]
ans =
    7   10
   15   22
> [1 2; 3 4] .* [1 2; 3 4]
ans =
    1    4
    9   16
```

Hatványozás hasonlóan:

```
> [1 2; 3 4]^(-1)
ans =
   -2.0000    1.0000
    1.5000   -0.5000
> [1 2; 3 4].^(-1)
ans =
    1.0000    0.5000
    0.3333    0.2500
```

A nevesített függvények általában elemenként hatnak:

```
sin (0:0.1:2*pi)
exp ([0,-1;1,0])
```

A m?veleti jelek pedig mátrix m?veletként (\*, ^, /, \)

## Változók

Ahhoz hogy ne csak egy soros dolgokat tudjunk számolni, az adatokat *változókb*an tároljuk.

```
a = 2
b = 3
a + b
```

Mindig van egy `ans` nev? változónak, amiben az *utoljára kiszámolt érték* található.

Ha nincsen érték adva egy változónak, akkor nem tudunk hivatkozni rá:

```
> a / q
error: `q' undefined
```

A pontosvessz?vel (;) **csendes számolást** végezhetünk, ekkor a parancs eredménye nem lesz kiírva:

```
a = 2;
b = 3;
a + b
```

Akkor hasznos, ha az eredményt úgyis elmentjük egy változóba és külön nem akarjuk látni. Kés?bb is megnézhetjük.

A `whos` paranccsal megnézhetjük az aktuálisan tárolt változóinkat.

```
> whos
Variables in the current scope:
  Attr Name      Size      Bytes  Class
  ====  =====
      a          1x1         8  double
     ans          1x1         8  double
      b          1x1         8  double
Total is 3 elements using 24 bytes
```

Egy változó értékét bármikor felülírhatjuk:

```
> a = 2;
> a = [1 2; 3 4];
> whos
Variables in the current scope:
  Attr Name      Size      Bytes  Class
  ====  =====
      a          2x2        32  double
```

## Indexelés

Legyen `M` egy 3x3-as mátrix. Ennek az *i*-edik sorának *j*-edik eleme a következ?.

```
M = rand(3,3);
i = 1;
```

Tagonként vagy mátrixként

```
j = 3;
M(i, j)
```

Mátrixok összefűzése:

```
[M M]
[M; M]
```

Részsorozat kiválasztása a tartományok használatával:

```
l = 0:0.1:1;
l (:)
l (1:11)
l (1:5)
l (5:end)
l (1:2:11)
```

Sűt:

```
l (1:2:11) = 0
```

Rézmátrix hasonlóan, csak két indexszel.

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

Egy sor kihagyása:

```
A ([1, 3], :)
```

Vagy rézmátrix kiválasztása:

```
S = ones (8, 8);
S (3:6, 3:6) = -1
```

Vagy adott indexekre:

```
S = ones (8, 8);
S ([1 2 8], [2 6]) = 8
```

Mátrix kilapítása:

```
A = eye (3, 3);
A (:)
```

## Összegzés

- A mátrix sorait összegzi a `sum( , 1)`
- A mátrix oszlopait összegzi a `sum( , 2)`

Azon dimenzió szerint összegez, ami a második paraméter. Ha nincsen második paraméter, akkor 1-nek tekinti. Az összes elemet kilapítással összegezzük:

```
sum(A(:))
```



## Vektorizáció

Az MatLab-ban általában egyszerre sok dolgot számolunk, nem csak egy értéken értékelünk ki egy függvényt. Például az  $X$  mátrix minden sorának számoljuk ki a normáját ( $X$  lehet  $n \times 3$ -as, ahol  $n$  nagyon sok):

```
X = rand (10, 3);
sqrt (sum (X.^2, 2))
ans =
    0.99105
    0.86977
    1.29362
    0.91697
    1.26149
    0.84024
    1.45410
    1.19791
    1.01153
    1.07420
```

Belülről kifelé haladva elemezzük a függvényeket:

- $X.^2$ : kiszámolja az elemenkénti négyzetet
- $\text{sum}(\text{ , } 2)$ : összegzi a mátrix sorait egy oszlopvektorba (a 2 azt jelenti, a tömb második dimenziója, azaz a 2. index mentén)
- $\text{sqrt}$ : elemenként gyököt von

Számoljuk ki a  $2x^2 - 3x + 1$  függvényértékeket, ahol  $x$  egy sorvektor:

```
x=0:0.1:1;
fx=2.*x.^2 - 3.*x + 1
```

A **vektorizáció** lényege, hogy ahol lehet mátrix és vektor műveletekre vezessük vissza a számításainkat, mert **1000 darab számpár összeszorzása lassabb, mint két darab 1000 hosszú vektor szorzása!**

## Függvények

Írjuk be a következőket egy `f.m` nevű fájlba!

```
function fx = f (x)
    fx = 1 / (x^2 + 1);
end
```

Ha a fájl neve megegyezik a függvénnel és abban a mappában van, ahonnan a programot indítottuk, akkor így használhatjuk:

```
> f(3)
ans = 0.10000
```

Függvények megadása:

```
function <<az eredmény>> = <<a függvény neve>> (<<változók>>)
    ...
end
```

A függvény hasáiban bármit számolhatunk, de a végén adjunk értéket <<az eredmény>> változónak. A függvény hasáiban érdemes csendes számítást végezni, itt használjunk mindenütt pontosvesszőt (;) a sor végén!

Egy másik függvény:

```
function R = remove_last (x)
    R = x (1:end-1);
end
```

Ezt egy `remove_last.m` fájlba tegyük. Példa:

```
> remove_last (1:5)
ans =
     1     2     3     4
```

## Feladatok

### Elemi sorműveletek elvégzése

Hozzunk létre egy egyjegyű nemnegatív egészekből álló  $4 \times 5$ -ös mátrixot, majd cseréljük ki két sorát, szorozzuk be a második sorát 2-vel és adjuk az első sorának kétszeresét a harmadik sorhoz! (A mátrix redukált lépcsős alakjának meghatározására azért a `rref` függvényt használjuk!)

### Mi ez?

Figyeljük meg a következőket.

```
A=[1 2 3; 4 5 6; 7 8 9];
B=[9 8 7; 6 5 4; 3 2 1];
trace (A*B')
A(:)' * B(:)
```

Mi a `trace (A*B')`?

## LER

Számoljuk ki a következő lineáris egyenletrendszer megoldását:

$$\begin{aligned} x + 2y &= 3 \\ 4x + 5y &= 6 \end{aligned}$$

1. megoldás:

```
A = [1 2; 4 5]
b = [3; 6]
x = A^-1 * b
```

2. megoldás:

```
x = A \ b
```

## Még LER

Oldjuk meg a következ? egyenletrendszereket:

$$\begin{aligned}x + 5y &= 1 \\ 2x + 4y &= 2\end{aligned}$$

$$\begin{aligned}x + 5y &= 1 \\ 2x + 4y &= 2 \\ 5x - 6y &= -1\end{aligned}$$

$$\begin{aligned}x + 2y + 5z &= 1 \\ 5x + 4y + 6z &= 2\end{aligned}$$

## Nagy mátrix okosan

- Készítsük el a következ? mátrixot okosan! (Minél kevesebb karaktert használva.)

```
1 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2
2 2 3 2 2 2 2 2 2 2
2 2 2 4 2 2 2 2 2 2
2 2 2 2 5 2 2 2 2 2
2 2 2 2 2 6 2 2 2 2
2 2 2 2 2 2 7 2 2 2
2 2 2 2 2 2 2 8 2 2
2 2 2 2 2 2 2 2 9 2
2 2 2 2 2 2 2 2 2 10
```

- És most ezt:

```
0 1 0 0 0 0 0 0 0 0
-1 0 1 0 0 0 0 0 0 0
0 -1 0 1 0 0 0 0 0 0
0 0 -1 0 1 0 0 0 0 0
0 0 0 -1 0 1 0 0 0 0
0 0 0 0 -1 0 1 0 0 0
0 0 0 0 0 -1 0 1 0 0
0 0 0 0 0 0 -1 0 1 0
0 0 0 0 0 0 0 -1 0 0
```

- Sakktáblaszabály

```
1 -1 1 -1 1
-1 1 -1 1 -1
1 -1 1 -1 1
-1 1 -1 1 -1
1 -1 1 -1 1
```

## Függvény mátrixokon

Írjunk függvényt, mely az adott mátrix minden elemére alkalmazza a  $2\sin^2x + 1$  függvényt.

## Részmátrix

Írjunk függvényt, mely egy 5x5-ös mátrix 2. és 4. sorából és 1., 3. és 5. oszlopából álló mátrixot adja.

## Résmátrixon függvény

Írjuk meg az előző két függvény kombinációját, mely az adott mátrix 2. és 4. sorából és 1., 3. és 5. oszlopából álló mátrixon alkalmazza a  $2\sin^2x + 1$  függvényt.

## Minden második oszlop

Írjunk függvényt, mely tetszőleges mátrix minden második oszlopából álló mátrixot adja vissza. (Segítség, a `size` sorvektorban megadja a mátrix dimenzióját.)

## Függvény alkalmazás csak adott elemeken

Írjunk függvényt, mely a kapott mátrix csak minden második oszlopán hajtja végre a  $2\sin^2x + 1$  függvényt. (Az eredmény mátrix dimenziója ugyanaz, mint a kapott mátrix.)

## Segítség magadon

- <https://www.mathworks.com/matlabcentral/answers>
- <http://www.mathworks.com/help/matlab/>
- Google

## Numerikus deriválás

Deriváljuk az  $f(x) = 2x^2 - 3x + 1$  függvényt numerikusan! Adott egy  $x$  sorvektor, ami az abszcissa értékeket tartalmazza,  $fx$  pedig a hozzájuk tartozó függvényértékeket.

```
x = 0:0.1:1
fx = 2.*x.^2 - 3.*x + 1
```

Ekkor a függvény numerikus deriváltja:

```
df = (fx(2:end) - fx(1:end-1)) ./ 0.1
```

Nem egyenletes lépésközzel pedig:

```
df = (fx(2:end) - fx(1:end-1)) ./ (x(2:end) - x(1:end-1))
```

Előző gyakorlat - Fel - Következő gyakorlat