Közelítő és szimbolikus számítások

2. gyakorlat

# Matlab alapok 2.

*Készítette:*
Gelle Kitti

Csendes Tibor
Somogyi Viktor
London András
Deák Gábor
*jegyzetei alapján*

# 1. Programozás Matlabban

Eddig a beépített funkciók használatát láthattuk, viszont a Matlab nem csak erre alkalmas, hanem, mint ahogy a bevezetőben elhangzott, a saját algoritmusainkat is megtervezhetjük vele. Ehhez a hagyományos procedurális programozási nyelvekhez hasonlóan tudunk vezérlési szerkezeteket használni, és függvényeket definiálni.

## 1.1 A megszokottól eltérő műveletek és vezérlési szerkezetek jelölései

**Megjegyzés**: A továbbiakban ebben az alfejezetben a szögletes zárójelek közötti kifejezések opcionális lehetőségeket jelölnek, tehát ezeket nem muszáj kitenni.

- `~=` - nem egyenlő
- `~` - negáció
- `&` - elemenkénti logikai ÉS (nem összetévesztendő a hagyományos nyelvekben ugyanezen jelöléssel szereplő bitművelettel)
- `|` - elemenkénti logikai VAGY (nem összetévesztendő a hagyományos nyelvekben ugyanezen jelöléssel szereplő bitművelettel)
- `&&` - logikai ÉS
- `||` - logikai VAGY
- `1` - IGAZ (nem összetévesztendő a hagyományos nyelvekben ugyan ezen jelöléssel szereplő bitművelettel)
- `0` - HAMIS (nem összetévesztendő a hagyományos nyelvekben ugyan ezen jelöléssel szereplő bitművelettel)

## 1.2 Vezérlési szerkezetek

### Szelekciós vezérlés (if–else)

```
if logikai_feltetel
    utasitas(ok)1
[elseif logikai_feltetel2
    utasitas(ok)2]
[else
    utasitas(ok)3]
end
```

### Kezdőfeltételes ismétléses vezérlés (while)

```
while feltetel
    utasitasok
end
```

### Számlálásos ismétléses vezérlés (for)

```
for ciklusvaltozo=kezdo[:lepeskoz]:zaro
    utasitasok
end
```

### Esetkiválasztásos szelekciós vezérlés (switch–case)

```
switch switch_kifejezes
    case case_kifejezes1
        utasitas(ok)1
    [case {case_kifejezes2,case_kifejezes3,...}
        utasitas(ok)2]
    otherwise
        utasitas(ok)3
end
```

## 1.3 Függvények definiálása

A Matlabban a függvényeket .m kiterjesztésű fájlokba írjuk. Minden függvénynek külön fájlba kell kerülnie. Egy új függvényt a File/New/Function menüben tudunk létrehozni. Általános kinézete:

```
function [ kimeno_parameterek ] = nev( bemeno_parameterek )
% fuggveny dokumentacioja

    fuggvenytorzs

end
```

További hasznos parancsok függvényekhez

- `return` - függvényből való visszatérés
- `break`, `continue` - ciklus kényszerített befejezése/ léptetése
- `nargin`, `nargout` - függvény be/kimeneti paramétereinek száma
- `feval` - függvény kiértékelése adott helyen (pl. `feval('fun', 4)`)
- `input` - inputérték bekérése konzolról (pl. `input('Mi a neved?\n', ... 's')` vagy `x=input('x erteke: ')`)
- `fopen`, `fclose` - fájl megnyitása, bezárása
- `fprintf`, `fscanf` - fájlba írás, ill. fájlból olvasás
- `disp`, `sprintf` -változóérték megjelenítés, formázott kiíratás

Szkript: a Matlab parancssorban kiadható utasításokat egy .m kiterjesztésű fájlba menthetjük, az így elkészült szkriptet a fájl nevének begépelésével futtathatjuk

### 1.3.1 Példa

Írjunk egy függvényt, ami kiszámolja a számtani, mértani és harmonikus közepeket egy függvényben. A függvény neve legyen `kozepek`.

- A számtani közép képlete:

$$\frac{1}{n}\sum_{i=1}^{n} x_n, \text{ ahol } x_i \in \mathbb{R} \text{ és } n \in \mathbb{Z}^+$$

- A mértani közép képlete:

$$\sqrt[n]{\prod_{x=1}^{n} x_i}, \text{ ahol } x_i \in \mathbb{R}^+ \text{ és } n \in \mathbb{Z}^+$$

- A harmonikus közép képlete:

$$\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}, \text{ ahol } x \neq 0 \text{ és } n \in \mathbb{Z}^+$$

**Megjegyzés.** A lenti kódban lehetnek $\neq$ jelek, amiket matlabban valójában ~= -nek kell írni, csak a LaTeX sajátosságai miatt jelennek meg így.

```
function[ szamtani, mertani, harmonikus ] = kozepek1( x )
% Magyarazo szoveg, a lookfor 'szoveg' utasitas itt keresi a 'szoveg'
% karakterlancot, illetve a help kozepek parancs is ezt irja ki
%
% Szamtani kozep: A = (x(1)+x(2)+...+x(n))/n, ahol x(i)eR,
% i=1,2,...,n
% Mertani kozep: G = (x(1)*x(2)*...*x(n))^(1/n), ahol x(i)≥0,
% i=1,2,...,n
% Harmonikus kozep: H = n/(1/x(1)+1/x(2)+...+1/x(n)), ahol x(i)≠0,
% i=1,2,...,n

    if( nargin ≠ 1 )
        error('A fuggveny 1 bemeno parameterrel dolgozik!')
    elseif( isempty(x) )
        error('A fuggveny csak nem ures vektorral dolgozik!')
    elseif( prod(double(x > 0)) == 0 )
        error('A fuggveny csak pozitiv elemeket tartalmazo vektorral ...
           dolgozik!')
    end

    [m n] = size(x);
    if (m > 1 && n > 1)
        error('A fuggveny csak vektorral dolgozik')
    end

    n = length(x);

    szamtani = 0;
    mertani = 1;
    harmonikus = 0;

    for i=1:n
        szamtani = szamtani + x(i);
        mertani = mertani * x(i);
        harmonikus = harmonikus + 1/x(i);
    end

    szamtani = szamtani / n;
    mertani = mertani^(1/n);
    harmonikus = n/harmonikus;

end
```

### 1.3.2 Példa

Ugyan azt a függvényt készítjük el, mint az előző feladatban, csak kicsit rövidebben.

```
function [ szamtani, mertani, harmonikus ] = kozepek2( x )
% Szamtani kozep: A = (x(1)+x(2)+...+x(n))/n, ahol x(i)eR,
% i=1,2,...,n
% Mertani kozep: G = (x(1)*x(2)*...*x(n))^(1/n), ahol x(i)≥0,
% i=1,2,...,n
% Harmonikus kozep: H = n/(1/x(1)+1/x(2)+...+1/x(n)), ahol x(i)≠0,
% i=1,2,...,n

    if( nargin ≠ 1 )
        error('A fuggveny 1 bemeno parameterrel dolgozik!')
    elseif( isempty(x) )
        error('A fuggveny csak nem ures vektorral dolgozik!')
    elseif( prod(double(x > 0)) == 0 )
        error('A fuggveny csak pozitiv elemeket tartalmazo vektorral ...
           dolgozik!')
    end

    [m n] = size(x);
    if (m > 1 && n > 1)
        error('A fuggveny csak vektorral dolgozik')
    end

    n = length(x);

    szamtani = sum(x)/n;
    mertani = prod(x)^(1/n);
    harmonikus = n/sum(1./x);

end
```

## 2. Feladatok

1. Írj egy olyan eljárást, ami kiszámolja a szigmoid függvény. Az eljárásnak egy paramétere van, és amennyiben ez vektor vagy mátrix, úgy elemenként hajtja végre a szigmoid függvény kiszámolását!
   A szigmoid függvény képlete:
   $$\text{sigm}(x) = \frac{1}{1 + e^x}$$

2. Írj egy olyan eljárást, ami kiszámolja a *p-norma* alapú távolságot. A programnak három paramétere van, egy `x` és `y` vektor, amik közötti távolságot számoljuk, illetve egy `p` szám, amely a távolság típusát határozza meg. A p-norma alapú távolság képlete:
   $$\left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

3. Írj egy függvényt a $a \cdot x^2 + b \cdot x + c = 0$ másodfokú egyenlet megoldására. Bementi paraméterek: $a, b, c$. A megoldás kezelje a speciális eseteket is: többszörös megoldás, komplex megoldás.

4. Írj egy függvényt, aminek bemeneti paramétere egy rand(n) mátrix. Hívd meg a cos függvényt százszor egymástól függetlenül a bemeneti mátrixra. A visszatérési érték egy 100 hosszú sorvektor legyen, ami a végrehajtási időket tartalmazza (ld. `timeit()` függvény). Számold ki a minimális, maximális és átlagos időtartamot, illetve a mért idők szórását is. Az eredményt az *eredmenyek.txt* fájlba írja ki a program.