Közelítő és szimbolikus számítások

1. gyakorlat

# Követelmények, Matlab alapok 1.

*Készítette:*
Gelle Kitti

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

## 1. Követelmények

A gyakorlat folyamán összesen **4 db 10-15 perces ZH-t** fogunk írni, amelyek **10-10 pontot** érnek. Az így elérhető 40 pontból **minimum 20 pontot (50%) el kell érni** ahhoz, hogy a gyakorlat sikeres legyen. Továbbá az első és második ZH-n és a harmadik és negyedik ZH-n is **minimum 6-6 pont** elérése szükséges a kurzus teljesítéséhez.

Egy ZH javítására/pótlására abban az esetben van lehetőség, ha egy részteljesítés nem éri el a minimális pontszámot.

A ZH-k mellett egy *szorgalmi* **otthoni feladat** kiosztására is sor kerül, amellyel egy jegyet lehet javítani (ha az egyébként megszerzett jegy legalább elégséges). Szorgalmi feladat jellegéből adódóan ennek javítására/pótlására nincs lehetőség.

A házi feladat programozást és esszéírást takar. Ezeket a feladatokat a *Matlab* nevű programmal kell elkészíteni.

A ZH-k időpontjai:

- 1. ZH: 3. gyakorlat, szeptember 20.
- 2. ZH: 6. gyakorlat, október 11.
- 3. ZH: 10. gyakorlat, november 8.
- 4. ZH: 13. gyakorlat, november 29.

Ponthatárok:

- 32-40 jeles
- 28-31 jó
- 24-27 közepes
- 20-23 elégséges
- 0-19 elégtelen

## 2. Elérhetőségek

E-mail: kgelle@inf.u-szeged.hu
A levél tárgya legyen: [Köszi]

## 3. Motiváció

Sokakban felmerülhet, miért hasznos a gyakorlaton vett módszereket tanulni? Sok olyan helyzet adódik a gyakorlati életben, amikor nem tudunk egy pontos megoldást adni, vagy azért, mert elméletben nem lehetséges, vagy pedig azért, mert a létező pontos megoldás számításigénye hatalmas, és a legjobb szuperszámítógépekkel is elviselhetetlenül sok időbe telne. Az ilyen helyzetekre például közelítő módszereket szokás használni.

Ezen kívül egy fontos szempont például, hogy a matematikában gyakran használatos integrálás, deriválás műveleteit hatékonyan oldjuk meg a számítógép segítségével. Ez utóbbi azért nagyon fontos, mert sok számítógépes algoritmus használja ezeket (például a gépi tanulás területén). A deriváláshoz használhatunk például szimbolikus módszereket.

Azok a módszerek, amiket a gyakorlaton látni fogtok, többnyire elemi módszerek, nagyobb algoritmusok építőkövei, amelyeket a képfeldolgozás, számítógépes látás, optimalizálás, mesterséges intelligencia, tehát a tudományos programozás területén alkalmaznak (és önmagukban nem fordulnak elő). Ahhoz pedig, hogy egy hatékony algoritmust tudjunk felépíteni, jó ha tudjuk, milyen alkotóelemek állnak a rendelkezésünkre és azok mire használhatók.

## 4. Mi is a Matlab és miért pont ezt használjuk?

Egy kutatás-fejlesztési projektben, ahol új módszereket kell adni egy probléma megoldására, gyakran előfordul, hogy több algoritmust kell implementálni, mire eljutunk a megfelelő eredményeket elérő eljáráshoz. Ennek a kifejlesztési költsége nagyon magas lenne hagyományos nyelvekben (C/C++, Java stb.), mivel sokszor előfordul, hogy algebrai eszközöket kell használnunk, melyek hatékony implementációja is fontos. Ennek egyik módszere a *MATLAB (Matrix Laboratory)* használata.

A MATLAB speciális programrendszer, amelyet numerikus számítások elvégzésére fejlesztettek ki, valamint magas szintű programozási nyelv. A *The MathWorks* által kifejlesztett programrendszer képes mátrix számítások elvégzésére, függvények és adatok ábrázolására, algoritmusok implementációjára és felhasználói interfészek kialakítására. Szimbolikus számításokra önmagában nem képes, de ez is megoldható ha a *Symbolic Math* eszköztárat külön telepítjük a *Mathworks*ből. Előnye, hogy könnyen felhasználhatunk benne más programozási nyelvekben (C/C++, Java, .NET, Python) íródott programokat.

Gyakran használják irányítástechnikai, képfeldolgozási, számítógépes látási, illetve gépi tanulási algoritmusok fejlesztésére.
Néhány hasonló szoftver:

- *Octave*, amely a GNU terméke és nyílt forráskódú
- *Maple*, melynek főleg szimbolikus eszköztára erős
- *Wolfram Mathematica*, mely szimbolikus programozási nyelvet használ

# 5. Alapvető tulajdonságok

A Matlab alapvető adattípusa a mátrix, megadása szögletes zárójelek között történik:

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

Oszlopok elválasztása: vessző vagy szóköz
Sorok elválasztása: pontosvessző
Ha egy parancs végére pontosvessző kerül, akkor a program nem írja ki az utasítás eredményét, csak eltárolja.

Kommentelni a százalékjel segítségével tudunk:

```
>> v1 = [4 6 7] %ez egy sorvektor
>> v2 = [4;6;7] %ez egy oszlopvektor
```

A második vektorral egyenértékű művelet a `v3 = [3 6 1]'` parancs is, ahol a `'` a transzponálás műveletét jelenti.

Hivatkozás a mátrix elemeire:

```
>> A(2,3)%nev(sor, oszlop)
ans = 7
```

A fenti sorban az `ans` egy ideiglenes változó, amit a Matlab akkor használ, ha egy értékadás bal oldalán nem szerepel semmi. Fontos különbség, hogy a legtöbb programozási nyelvvel szemben a Matlab az indexelést 1-től kezdi.

## 5.1 A mátrix elemeinek elérése és manipulációja

A mátrix elemeit a fentieken kívül sokkal több módon is el lehet érni. Például:

```
>> A([1 3], [2 4])%a matrix 1. es 3. soranak 2. es 4. eleme
>> A([1:3], 3)%a matrix elsotol harmadik sorainak harmadik eleme
>> A(2, [1:3])%a matrix masodik soranak elsotol harmadik elemei
>> A(:, 2)%masodik oszlop
```

```
>> A(2:end, 1)%a masodik sortol az utolsoig az elso oszlop elemei
```

A mátrixokat ki is tudjuk bővíteni (a művelet sor és oszlopellenőrzést végez, tehát hibát írhat ki, ha nem megfelelő méretű a konkatenált mátrix):

```
>> B = [A, [3 2 1]'] %oszlop hozzafuzese
>> C = [A; [3 2 1 0]] %sor hozzafuzese
>> K = [ 1 2 3 4; 5 6 7 8]
>> D = [A; K]
```

Egy sorba több parancsot is írhatunk vesszővel vagy pontosvesszővel elválasztva.

Általánosan a kettőspont operátor:

| | |
|---|---|
| `j:k` | mint a `[j,j+1,...,k]`, és üres, ha $j > k$ |
| `j:i:k` | mint a `[j,j+i,j+2i,...,k]`, és üres, ha $i > 0$ és $j > k$ vagy ha $i < 0$ és $j < k$ |
| `A(:,j)` | `A` $j$. oszlopa |
| `A(i,:)` | `A` $i$. sora |
| `A(:,:)` | a teljes 2D-s tömb. Mátrixoknál ez ugyanaz, mint `A` |
| `A(j:k)` | eredménye: `A(j),A(j+1),...,A(k)` |
| `A(:,j:k)` | eredménye: `A(:,j),A(:,j+1),...,A(:,k)` |
| `A(:,:,k)` | egy 3D-s `A` tömb $k$. lapja |
| `A(i,j,k,:)` | ha `A` egy 4D-s tömb, az eredményvektor a következőket tartalmazza: `A(i,j,k,1)`, `A(i,j,k,2)`, stb |
| `A(:)` | `A` összes eleme egy oszlopban reprezentálva |

Ha tudni szeretnénk, hogy mekkora méretű a mátrixunk, akkor ezt a `size(A)` hívással deríthetjük ki, amely egy vektort ad vissza:

```
>> size(A)
ans =
     3     4
```

Hasonló művelet a `length(A)`, mely a paraméterül kapott mátrix dimenziói közül a nagyobbat adja vissza:

```
>> length(A)
ans = 4
```

## 5.2 Aritmetikai operátorok

Természetesen, mivel a program adattípusa a mátrix, ezért a műveletek is mátrixműveletekként vannak definiálva. A végrehajtható műveleteket két csoportba soroljuk:

1. normál mátrixműveletek: `= + - * ^ / \`
2. elemenkénti műveletek: `.* .^ ./`

```
>> N = [1 2; 3 4];
>> K = [5 6; 7 8];
>> N*K
ans = 19 22
      43 50
>> N.*K
ans = 5 12
     21 32
```

A jobboldali osztás, `A/B` eredménye olyan `Y` (amennyiben létezik), hogy `YB=A`, illetve a baloldali osztás, `A\B` eredménye olyan `X` (amennyiben létezik), hogy `AX=B`.

## 5.3 Logikai operátorok

- `~` - NEM
- `&` - elemenkénti ÉS
- `|` - elemenkénti VAGY
- `&&` - ÉS
- `||` - VAGY

## 5.4 Sorozatok

A Matlab segítségével sorozatokat is megadhatunk. Ezek használatát korábban már láttuk, de most már tudni is fogjuk, hogy mik ezek. A sorozatok megadásának is alapvetően két módja van.

1. A `:` operátorral adjuk meg `a:b:c` alakban, ahol `a` a kezdőérték, `b` a lépésköz, `c` pedig a befejező érték.

   ```
   >> s1 = 0:2:10
   s1 = 0     2     4     6     8     10
   ```

2. A `linspace(a,b,n)` függvénnyel, ahol `a` a kezdőérték, `b` a befejezőérték, `n` pedig az elemek száma.

```
>> s2 = linspace(0,10,5)
s2 = 0    2.5000    5.0000    7.5000    10.0000
```

## 5.5 Speciális mátrixok

- `eye(n,m)` - $n \times m$ méretű egységmátrix
- `ones(n,m)` - $n \times m$ méretű, csupa egyeseket tartalmazó mátrix
- `zeros(n,m)` - $n \times m$ méretű, csupa nullákat tartalmazó mátrix
- `rand(n,m)` - $n \times m$ méretű, a $[0, 1]$ intervallumból kikerülő, véletlenszerű értékeket tartalmazó mátrix
- `diag(x)` - egy diagonális mátrixot hoz létre, ahol `x` egy $1 \times n$ vektor (így a létrehozott diagonális mátrix $n \times n$ méretű lesz).

Mindegyik függvény hívható egy paraméterrel is, ilyenkor egy $n \times n$ méretű mátrixot kapunk.

## 5.6 Beépített konstansok

- `Inf` - végtelen
- `NaN` - Not a Number
- `pi` - $\pi$
- `tic, toc` - `toc` a legutolsó `tic` utasítás óta eltelt idő másodpercben
- `clock` - sorvektor, melynek tartalma a pillanatnyi idő

## 5.7 Fontosabb matematikai függvények

**1. Megjegyzés.** Matlabban a függvényeknek többféle visszatérési értékük is lehet akár a kapott paramétertől függően, akár attól, hogy mit várunk vissza.
**2. Megjegyzés.** Sok függvény, ha vektort vagy mátrixot kap paraméterül, akkor elemenként hajtja végre a műveleteket (ezeknek érdemes utánanéznünk a használatuk előtt).

- `floor(x)` - alsó egészrész
- `ceil(x)` - felső egészrész
- `round(x)` - kerekítés
- `abs(x)` - abszolútérték
- `sqrt(x)` - négyzetgyök
- `max(x)` - vektor esetén legnagyobb elemet adja vissza, mátrix esetén minden oszlopból a maximális elemet. Ha a visszatérési értéke `[X, I]` alakú, akkor az `I` vektorban a maximális elemek indexei lesznek (mátrix esetén sorindex)
- `min(x)` - a `max` függvénnyel analóg módon működik.
- `sum(x)` - vektor esetén `x` elemeinek összege, mátrix esetén az oszlopok elemeinek összege
- `prod(x)` - analóg a `sum` függvénnyel, csak itt az elemek (vagy mátrix esetén oszlopok) szorzatát kapjuk.

- `sin(x)`, `cos(x)`, `tan(x)`, `exp(x)`, `log(x)`, `log10(x)`, ...

## 5.8 Hasznos parancsok

- `help parancsnev` - súgó meghívása az adott parancsra
- `doc parancsnev` - a MATLAB HTML fájljainak elérése
- `clc` - parancsképernyő törlése
- `home` - a kurzort a parancsablak bal felső sarkába viszi
- `disp(a)` - az argumentum tartalmának képernyőre íratása
- `error('string')` - hibaüzenet képernyőre írása
- `input('string')` - bemenő adat kérése
- `clear [nev/all]` - törli a változókat a munkaterületről
- `exit,quit` - MATLAB bezárása

## 5.9 Függvények ábrázolása

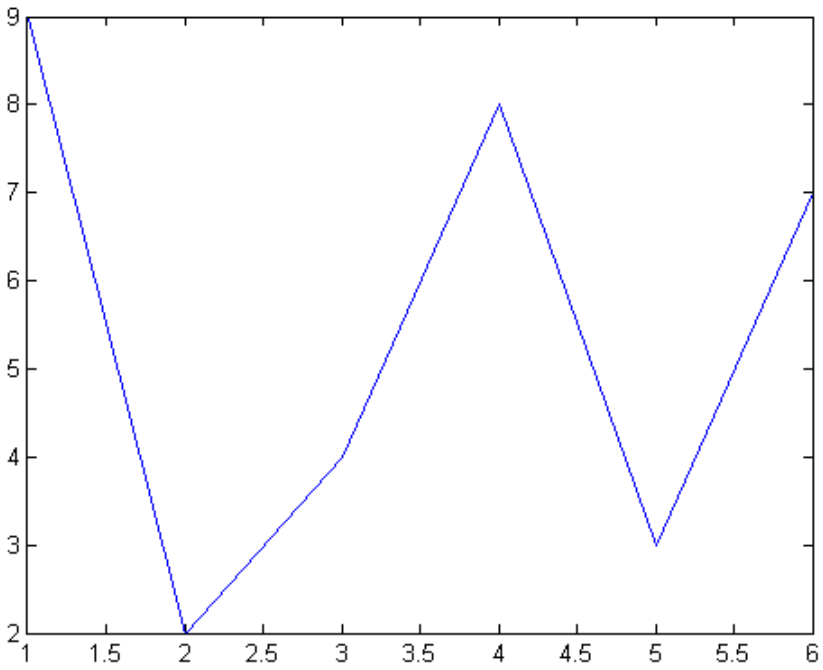
Alapvetően a `plot` függvény segítségével tudunk ábrázolni függvényeket. A `plot`-nak többféle bemenete lehetséges, és ettől függően generál kimenetet.
Az 1. Ábrán azt láthatjuk, hogy mi lesz a kimenete a `plot` parancsnak, ha a `plot([9 ... 2 4 8 3 7])` paraméterezéssel hívjuk meg.

A következőben a szinusz függvényt számoltatjuk ki vele a $[0, 2\pi]$ intervallumban, $\pi/100$-as lépésközökkel (ez látható a 2. Ábrán:

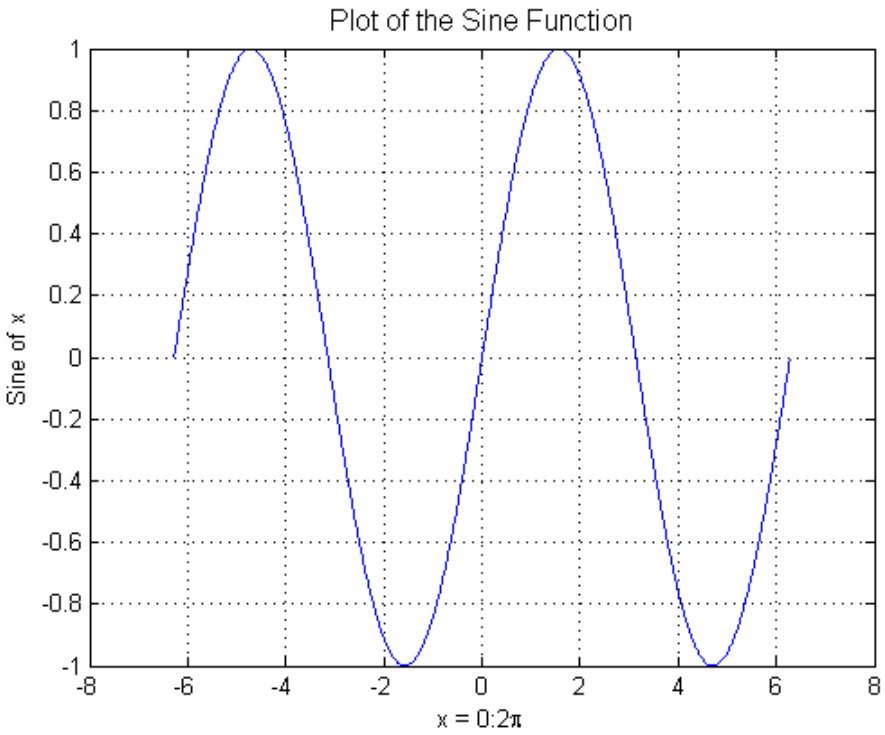
```
>> plot(sin(0:pi/100:2*pi)).
```

Egy kicsit bonyolultabb ábrázolás:

```
>> x = -2*pi:0.01:2*pi; %az abrazolando intervallum
>> y = sin(x); %az abrazolt fuggvenyertekek
>> plot(x,y)
>> xlabel('x = 0:2*\pi')
>> ylabel('x szinusza')
>> title('A szinusz fuggveny abrazolasa','FontSize',12)
>> grid on %bekapcsoljuk a racsozas megjeleniteset
```




**1. Ábra.**: Paraméterként egy vektort adtunk meg




**2. Ábra.**: A szinusz függvény ábrázolása

Egyszerre megadhatunk akár több függvényt is (ennek a kimenetét a 3. Ábrán nézhetjük meg):

```
>> x = -pi:0.01:pi;
>> y1 = sin(x);
>> y2 = cos(x);
>> y3 = x.^2;
>> plot(x, y1, 'r', x, y2, 'g', x, y3, 'b')
>> legend('sin(x)','cos(x)','x^2')%jelmagyarazat
```

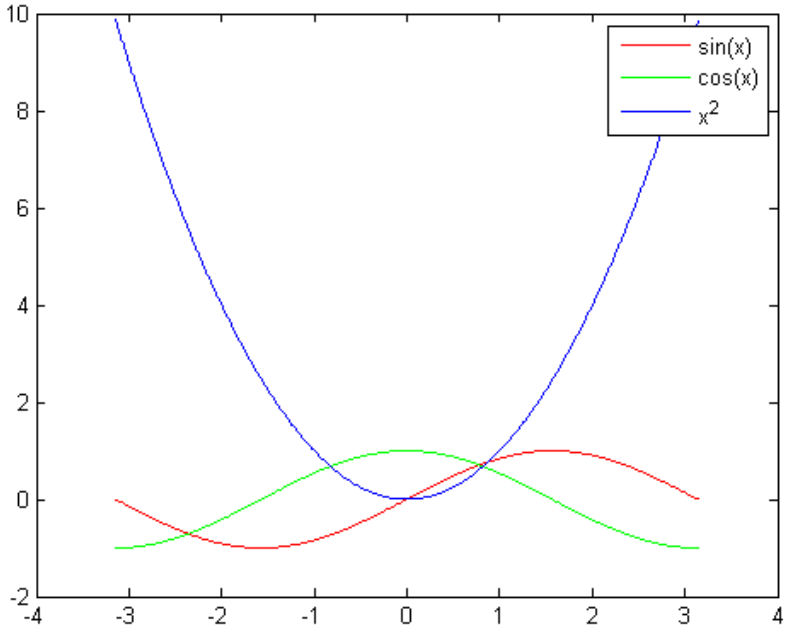
### 5.9.1 A `plot` utasítás paraméterezése

A `plot(x, y, '@#')` parancsban a @ jelenti a vonal típusát, ami lehet:

- `-` — folytonos
- `--` — szaggatott
- `:` — pontvonal
- `.` — pontok
- `+` — plusz
- `*` — csillag
- `o` — kör

A # jel pedig a vonal színét, ami többek között lehet:

- r — red, vörös
- b — blue, kék
- g — green, zöld
- w — white, fehér
- y — yellow, sárga




**3. Ábra.**: Több függvény ábrázolása egyszerre

A Matlab a következő plot utasítással automatikusan felülírja az előző plot utasítást. Ha szeretnénk az előző ábrát megtartva, új ablakba rajzolni, akkor a figure() utasítással nyithatunk meg új ablakot a következő plot utasítás számára.

```
>> plot(x,y, 'go')%kirajzolas zold korokkel
>> figure();
>> plot(x,y,'m--')%...lila szaggatott vonallal
```

Több ábrát a hold on utasítás kiadásával tudunk elhelyezni egy ablakban. Ez az utasítás megtartja a grafikus ablakot a későbbi rajzok számára egészen a hold off utasítás kiadásáig, amely újra "elengedi" az ablakot.

```
>> hold on, plot(x,y, 'go')
>> plot(x,y,'m--'), hold off
```

A tengelyhatárok megadása:

- `xlim([xmin,xmax])` és
- `ylim([ymin,ymax])`, vagy
- `axis([xmin,xmax,ymin,ymax])`

A `grid` paranccsal egy olyan rácsot illeszthetünk a koordináta rendszerre, amely illeszkedik a tengelyek beosztására. Bekapcsolás: `grid on`, kikapcsolás: `grid off`.

Feliratok elhelyezése:

- `xlabel(str), ylabel(str)` - tengelyfeliratok
- `title(str)` - ábra elnevezése
- `text(x,y,str)` - a rajz bármelyik, koordinátával megadott pontjára feliratot, szöveget helyezhetünk (gtext)
- `legend(str1, str2)` - magyarázó szöveg

## 5.10 Feladatok

1. Az `A = [1 2 3; 4 5 6; 7 8 9]` mátrixban mit ad eredményül az `A(2, [1 3])` utasítás?

2. Hozz létre egy oszlopvektort `v` néven, amibe tedd bele 1-től 10-ig a számokat 0.2-es lépésközzel!

3. Generálj egy $10 \times 10$ méretű mátrixot, amely véletlenszerűen tartalmaz nullákat és egyeseket, majd számold meg, hogy mennyi egyes van bennük!

4. Egy `v2` oszlopvektorba számold ki `v` eleminek négyzetgyökét, `v3` vektorba 10-es alapú logaritmusát!

5. Egy `negyzet` nevű változóba számold ki `v` elemei négyzetének az összegét!

6. Ábrázold az eddig létrehozott vektorokat egy közös ábrán vektoronként különböző színű pontokkal!

7. Ábrázold a természetes alapú logaritmus függvényt az $[1, 10]$ intervallumon!

8. Ábrázold a kettes, tízes és természetes alapú logaritmusokat az $[1, 10]$ intervallumon! Az egyes függvények legyenek különböző színűek, és legyen jelmagyarázat is.

9. Legyen adott két vektor: `vec1 = [3 1 4]` és `vec2 = [2 1 6]`. Számold ki a négyzetes hibájukat, amit a következő képlet ad meg:

$$\sum_{i=1}^{n} (x_i - y_i)^2$$

10. Írj egy kifejezést, ami kiszámolja egy $n$ hosszú vektorban megadott sorozat elemeire a következő képletet:

$$\sum_{i=1}^{n} \frac{1}{2} \left( n + \frac{1}{n^2 + 2n} \right)^2$$