Matlab 8–9. előadás

A Matlab programozása

Dr. Szörényi Miklós,
Dr. Kallós Gábor

2017–2018

Tartalom

- Algoritmus, pszeudokód, program
  - Programozási alapfeladatok
- Strukturált programozás
  - Vezérlő szerkezetek
- Elágazások
  - Két- és többágú szelekció
  - Feltételek, kiértékelés
- Ciklusok
  - For ciklus
  - Egymásba ágyazott ciklusok
  - While ciklus
- Rekurzió
- Bekérés és kiírás
  - Interaktív grafika
- Paraméterek kezelése
- A kód hatékonysága

Algoritmikus feladatmegoldás

Fontos fogalmak

- Számítási folyamat: a számítógépben zajlik
  - Adatmanipuláció (egy cél érdekében)
  - A működést *programok* vezérlik
- Szeretnénk mi is *programot készíteni*
- *Specifikáció*: kapunk egy feladatot, ill. egy leírást a feladatról
  - Mit kell csinálnia a programnak?
  - Mi lesz a bemenet, és mi lesz a kimenet
- *Algoritmus*: hogyan oldja meg a feladatot a program?
  - A megoldás *lépésről-lépésre* történő leírása
  - Nekünk kell kitalálni, nincs rá teljesen általános módszer
  - Az algoritmus kidolgozása egyszerre tudomány, „művészet” és mérnöki feladat
    - Pl. helyesség-bizonyítás, tulajdonságok számszerűsíthetőek, de: kellenek egyedi ötletek, felismerések, ugyanakkor vannak *szabványok*

Algoritmikus feladatmegoldás

Fontos fogalmak

- Algoritmus
  - Utasítássorozat megengedett lépésekből
  - Mechanikusan (gépiesen) végrehajtható
    - Jó esetben nem kell emberi beavatkozás
  - *Véges sok* lépésből áll, mindig *egyértelműen adott a következő lépés*
  - Minden időpontban *véges sok* memória kell (tehát nem végtelen)
  - A használt adatokat (változókat) az alg. tervezése előtt/közben össze kell gyűjteni!
- Egyszerű algoritmus példák
  - Számsorozat összeadása
  - Prímtényezős felbontás
- *Pszeudokód*: eszköz az algoritmus szöveges leírására
  - Mindig egy adott formalizmust követ (ez lehet „nagyvonalú”)
- *Program(kód)*: a szg. által érthető nyelven, megfelelő szabályokat betartva (szigorúan formálisan!) leírt algoritmus

Algoritmikus feladatmegoldás

Fontos fogalmak

- *Változó*: eltárolt, *megjegyzett érték*, amelynek nevet adunk
  - Vagy: egy hely neve (a memóriában), ahol egy értéket tárolunk
  - A változót lehet (ki)olvasni és lehet neki *értéket adni* (az érték tehát változhat)
    - Értékadáskor a régi érték elveszik
  - A program futása során, egyes időpontokban más-más lehet egy adott változó értéke
    - Ugyanaz a művelet máskor más eredményt adhat
- *Konstans*: egy helyre (a memóriában) fixen beírt érték
  - Értéke *nem változhat*
- *Típus*: értékkészlet és műveletek együttese
  - A változó típusa egyben a vele végezhető műveleteket is meghatározza
  - Klasszikus (egyszerű) típusok: egész, valós, logikai, karakter, szöveg
    - Pl. a logikai típusú változó ÉK-e: {igaz, hamis}, műveletek: „nem”, „és”, „vagy” stb.
  - Klasszikus összetett típusok: tömb (vektor, mátrix), rekord, halmaz stb.

Algoritmikus feladatmegoldás

Fontos fogalmak

- Kifejezések
  - Változókból, konstansokból és műveletekből (ill. relációkból) épülnek fel
  - Meg kell, hogy feleljenek a nyelvtani szabályoknak
  - A műveletek adott típusokon végezhetőek el, és az eredménynek is van valamilyen típusa (utóbbi lehet esetleg más)
    - Pl. egy szám+szám művelet eredménye is szám, de egy szám<szám összehasonlítás logikai eredményt ad
- Deklaráció
  - A változókat sok programozási nyelvben (előre) *deklarálni kell*
  - Erősen típusos programozási nyelvek: minden változót definiálni kell, előre meg kell mondani a típusát és a nevét (pl. C nyelv)
  - Gyengén típusos nyelvek: a változóknak nem kell előre megadni a típusát, sőt egy adott nevű változó típusa is változhat (pl. Matlab)

Strukturált programozás

- A program megengedett elemei:
- Elemi műveletek
  - Kiírás, bekérés
  - Értékadás, valaminek a kiszámolása
- Szekvencia
  - Egymás utáni lépések (a feladat részekre bontható)
  - Számít a sorrend!
- Szelekció (elágazás)
  - Feltételes végrehajtás
  - Logikai kifejezés vizsgálata, igaz ág, hamis ág
  - Minden futtatásnál csak egy konkrét ág hajtódik végre
- Iteráció (ciklus)
  - Programrész ismétlése, amíg egy feltétel fennáll
  - Ciklusmag, ciklusfeltétel (mindig újra kiértékelődik), c.fej, c.változó
  - Elöl- és hátultesztelő ciklus
- *Nem megengedett* például az ugrás (goto) vagy a kiugrás (break)
- Matematikailag bizonyított: minden számítógéppel megoldható feladat megoldható *strukturáltan is*
- Célszerű a kódot megfelelően tördelni
  - Egymásba ágyazott szerkezetek

Programozási alapfeladatok

- Tipikusan egy (adott/bekért) sorozat elemeit dolgozzuk fel
  - A sorozat elemszáma ismert lehet, vagy a végét spec. jel jelzi (pl. −1)
    - Ettől függően for ciklus vagy while ciklus használandó
  - A feladatok egyszerűbb esetben tömbök nélkül (is) megoldhatók
- Összegzés
  - Példa: számok összeadása, faktoriális
  - Összeg vagy szorzat változó használandó
- Számlálás
  - Példa: osztók száma, páros számok száma, adott karakterek száma
  - Darab változó használandó
- Szélsőértékek keresése
  - Példa: maximum- és minimumkeresés
  - Min/max változó használandó
  - Az első elem célszerűen külön kezelendő
- Eldöntési feladatok; szerepel-e egy adott érték a sorozatban
  - Példa: *n* prímszám-e
  - Logikai változó használandó
- Keresés, rendezés

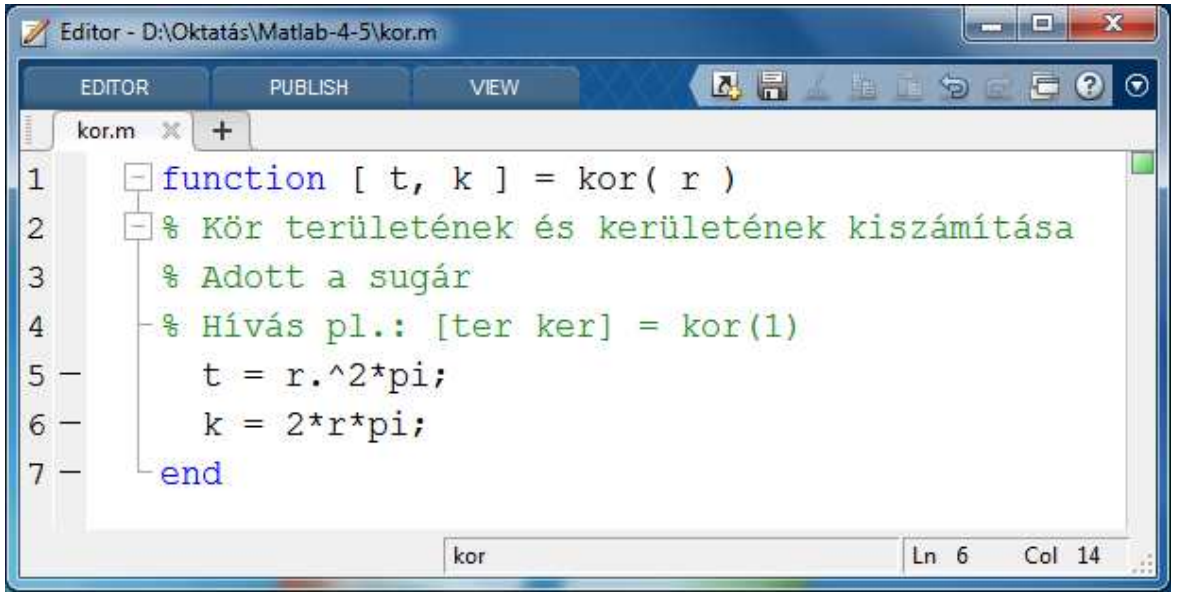
Áttekintés

A Matlab programozási környezetéről

- Saját programozási nyelv
  - Felhasználóbarát, de alacsonyszintű elemek is vannak/használhatók
- Barátságos fejlesztőkörnyezet
  - Szintaktikai kiemelések, fordítási ellenőrzés, javítási javaslatok stb.
- Változók, kifejezések
  - Nem deklaráljuk a változókat
  - Típusok, értékhatárok, műveletek, pontosság, konverziók stb. – tudjuk
  - Logikai műveletek, hasonlítások az ismert módon
- A strukturált programozás elveit követjük
  - Építőelemek: szekvencia, szelekció, iteráció
  - A végrehajtás alapvetően szekvenciális, de párhuzamos szerkezetek is beépíthetők
  - A vezérlőszerkezeteknek van eleje és vége (utóbbi tipikusan: end)
- Célszerűen m-fájlokban tároljuk a kódot
  - Tipikusan nem a parancsablakban dolgozunk
- Hibakeresési lehetőségek

Vezérlő szerkezetek/Szekvencia

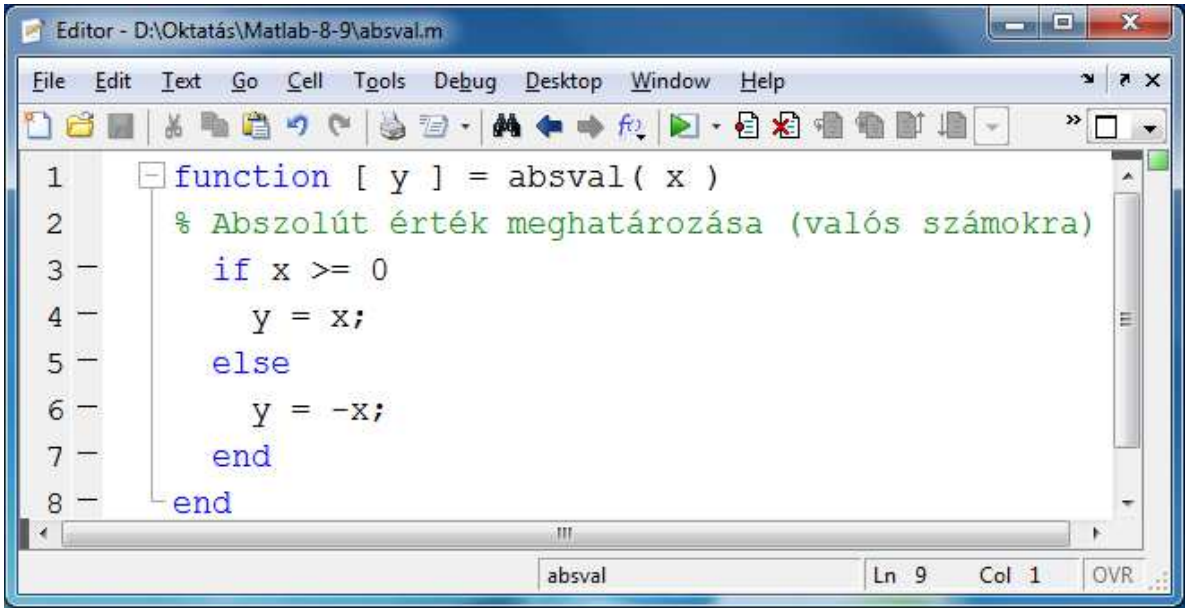
- *Szekvencia*: műveletek egymás utáni végrehajtása (az eredeti feladat felbontásával)
  - Elemi művelet: tovább már nem bontható (tudjuk)
- A csak szekvenciát tartalmazó prg.ok a legegyszerűbbek (általában)
- Szekvencia példa: kör kerülete és területe
  - Feladat: Írjuk át bekérősre (input parancs), ill. bővítsük szép kiírással (*disp* parancs)
- Másik hasonló példa: mágikus mátrix kiírása (script, gyak.)



Vezérlő szerkezetek/Elágazások

„Sima” szelekció

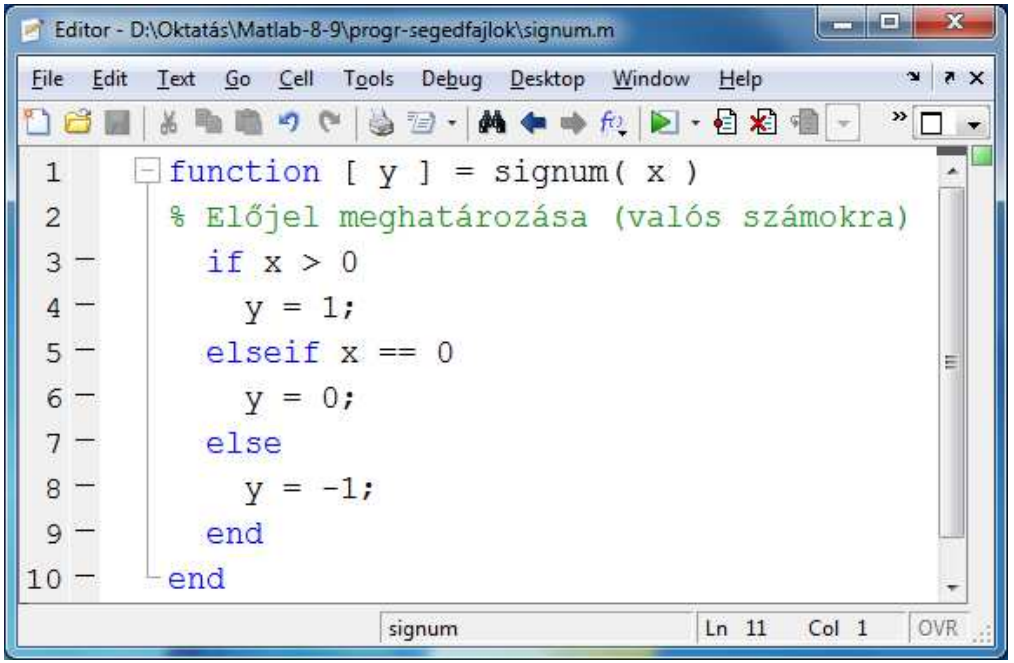
- Motiváció: A „fix” viselkedéstől eltérően, helyzettől függően a program különböző utasításokat hajt(hat) végre
- A klasszikus if-then-else szerkezet szintaktikája:
  `if logikai kifejezés[,|;] utasítás(ok), else utasítás(ok), end`
  - Megj.: A Matlabban nem használjuk a then kulcsszót!
- Példa: Abszolút érték számítása (saját függvény)
  - Elemezzük a működést! (Teszteljük a fv-t, hasonlítsuk össze a beépített változattal)
  - Mi történik komplex input esetén?



Elágazások

Elseif-es szelekció

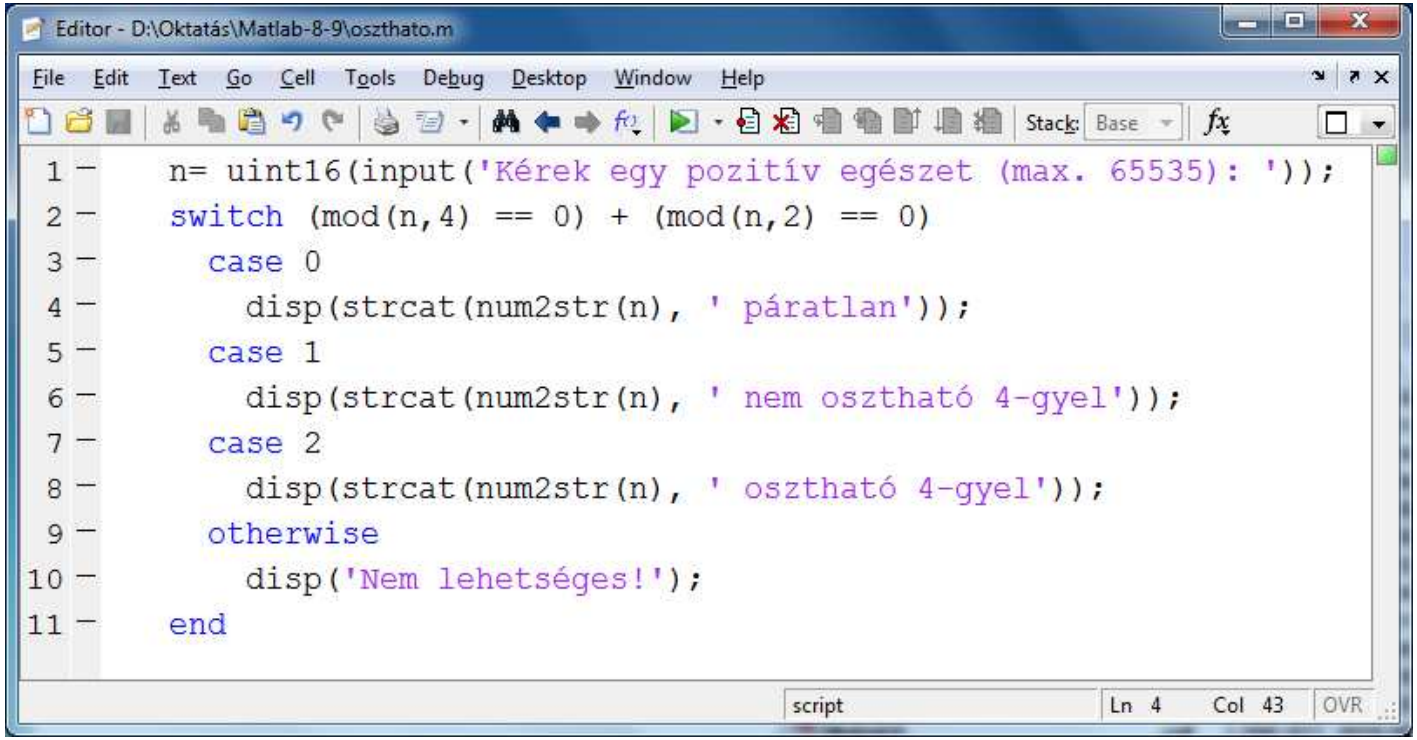
- Újabb ág beépítése szükséges (több mint két alternatíva)
  - Egymást kizáró ágak
  - *else*-be ágyazott újabb *if*-fel is kiváltható, de az *elseif* általában elegánsabb
- Szintaktika értelemszerűen
- Példa: Előjel számítása (saját függvény)
  - Elemezzük a működést! (Teszteljük a fv-t, hasonlítsuk össze a beépített változattal – *sign*)



Elágazások

Switch szerkezet

- Többfelé elágazást tesz lehetővé, egy kifejezés értékétől függően
  - Általában több mint két alternatíva
- Szintaktika: lásd Súgó
  - *switch*, *case* ágak, *otherwise*, a végén *end*
- Példa: Oszthatósági vizsgálat (m-fájl, script)
  - Az *strcat* sztringek összefűzését hajtja végre



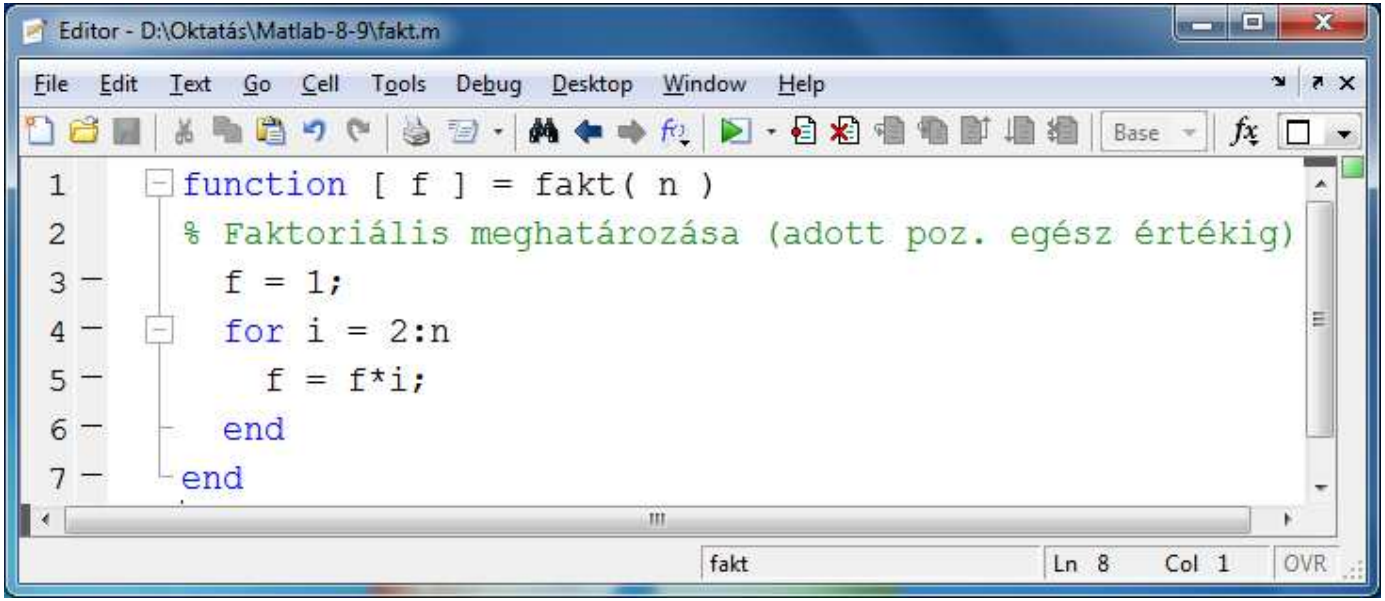
Logikai kifejezések

- Feltételek kiértékelésénél
- Felépítés
  - Relációk (tudjuk már): <, >, <=, ==, ~= stb.
  - (Relációs operátorok: *eq*, *gt*, *le* stb.)
  - Logikai operátorok (tudjuk már): | (*or*), & (*and*), ~ (*not*), *xor*, ||, && (+*any*, *all*)
  - Egyes adatkezelő függvények: *strcmp*, *length*, *size*, *isempty*, *isnumeric*, *isinf*, *isfinite*, *isequal*, *isstr*, *ismember*, *upper*, *lower* stb. (lásd később is)
- Kiértékelés
  - Zárójelezés, precedencia, balról-jobbra (tudjuk)
  - Specialitás: rövidzár (sokszor elég az első op. kiértékelése)
  - A feltétel tömbértékű kifejezés is lehet, ha van benne 0, hamisnak minősül (pl. eye(3) logikai 0, ones(3) logikai 1)
- Példák
  - if isnumeric(b) || (a > b)
  - if ismember(a, set)
  - while (b ~= 0) && (a/b > 18.5)
  - if exist('myfun.m') && (myfun(x) >= y)
  - elseif abs(m - n) == 2

Vezérlő szerkezetek/Ciklusok

For ciklus

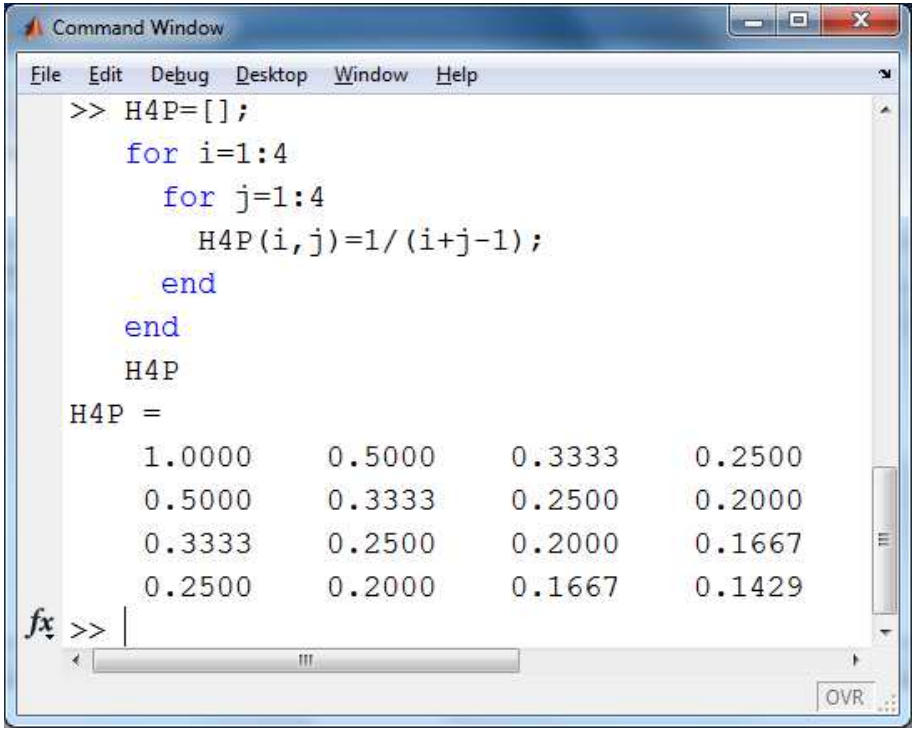
- A legegyszerűbb iteráció (fix lépésszám)
- Motiváció: Egy parancsot vagy parancsok egy csoportját néhányszor/többször meg kell ismételni
- Szintaktika:
  `for ciklusvált = kezdőé:végé[,|;] utasítás(ok), end`
  - A ciklusváltozót nemcsak egyesével léptethetjük, pl. i = 1:2:6
- Példa: Faktoriális (saját függvény)
  - Elemezzük a működést! (Teszteljük a fv-t különböző típusú adatokkal is)



Ciklusok

- Ciklusok egymásba is ágyazhatók
- Egymásba ágyazott for ciklusok tipikus alkalmazása: mátrix adatokkal való feltöltése, elemeinek bejárása, módosítása
- Példa: Hilbert-mátrix előállítása
  - (Matlabban persze *nem ez* a hatékony mo.)
  - Eml.: Tudjuk, hogy létezik beépített parancs
- A kód (4×4-es mátrix):

```
H4P = [];
for i = 1:4
  for j = 1:4
    H4P(i,j) = 1/(i+j-1);
  end
end
H4P
```



- Megj.: A mátrixot manuálisan for ciklusok nélkül is elő tudjuk állítani

```
> H4 = [1./[1:4];1./[2:5];1./[3:6];1./[4:7]]
```

Ciklusok

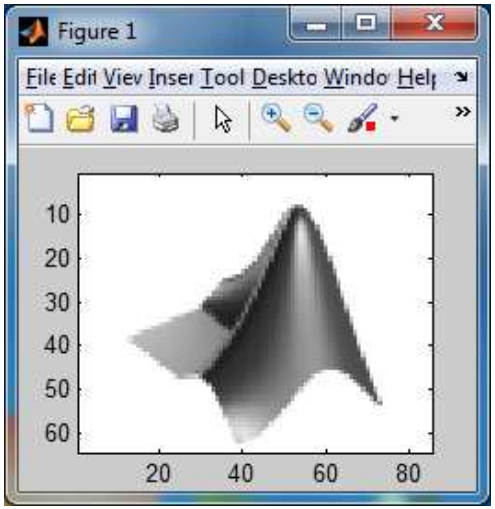
- Egymásba ágyazott ciklusok alkalmazása (folyt.)
- Eml.: Vágási feladat (szürkeárnyalatos képre)
  - Ha a színérték bizonyos korlát felett van, akkor fehér lesz, különben fekete (kontraszt szélsőséges kiemelése)
  - Lásd még: vektorizáció

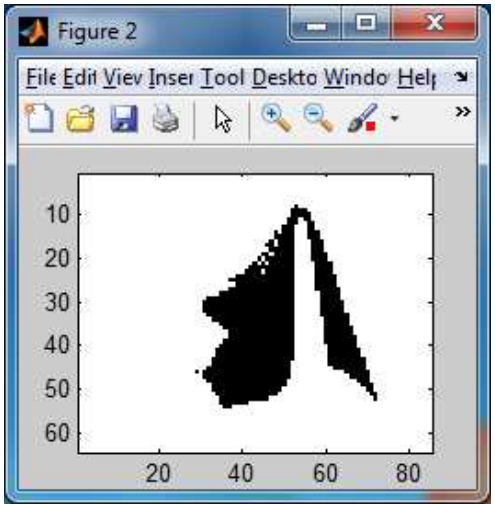
Editor - D:\Oktatás\Matlab-8-9\kepfeld-segedfajlok\threshold.m

File Edit Text Go Cell Tools Debug Desktop Window Help

```
1   function [ result ] = threshold( grayimg, t )
2   % grayimg is the grayscale image, t is the threshold value
3   % each pixel value that are greather than t in the input image
4   % will be changed to 255, while the other are changing to 0
5       result = grayimg;
6       img_size = size(grayimg);
7       h = img_size(1);
8       w = img_size(2);
9       for i = 1:h;
10          for j = 1:w;
11              if (grayimg(i,j) > t)
12                  result(i,j) = 255;
13              else
14                  result(i,j) = 0;
15              end;
16          end;
17      end;
18  end
```

threshold Ln 18 Col 4 OVR





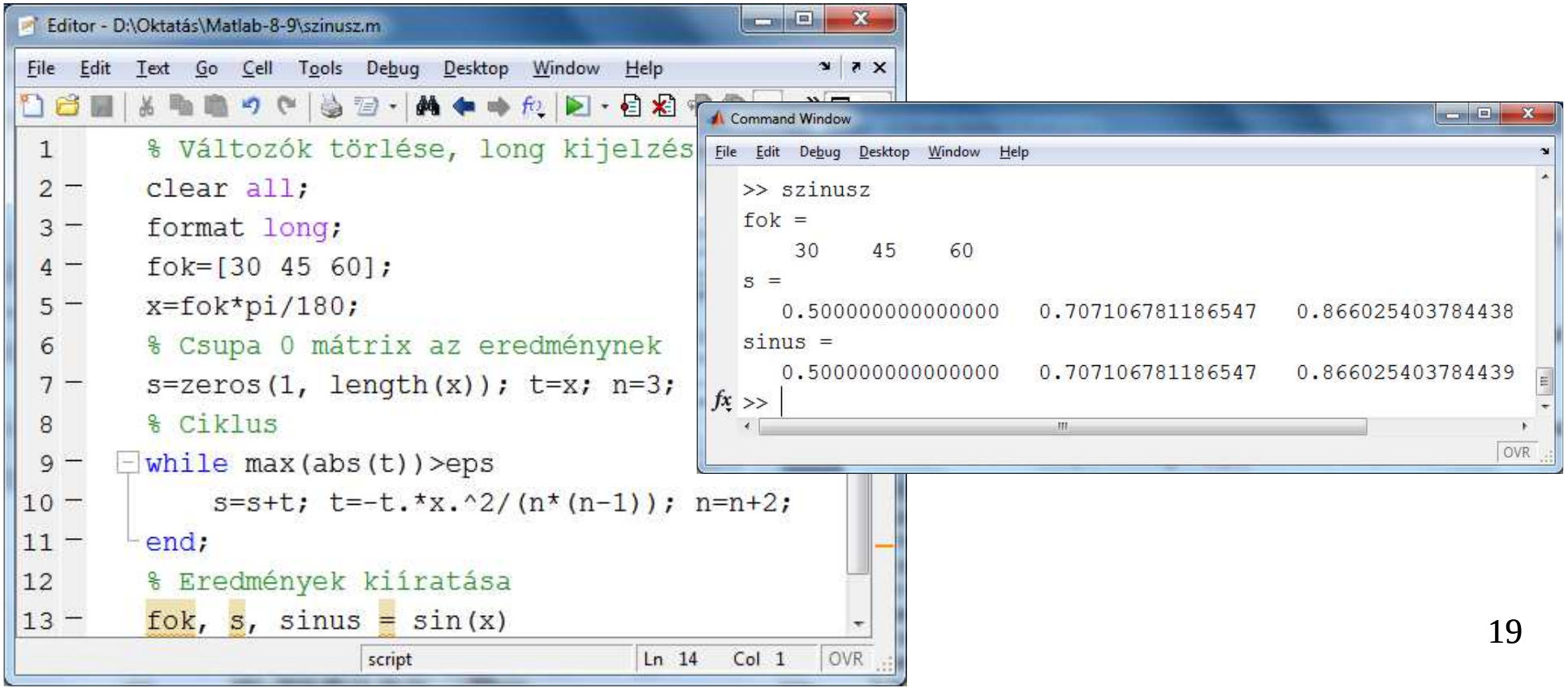
Ciklusok

While ciklus

- Motiváció: Nem tudjuk/akarjuk előre megmondani, hogy pontosan hányszor kell lefuttatni a ciklusmagot
  - Ez függhet például az inputtól
- Ilyenkor bennmaradási feltétel fogalmazható meg (igaz feltétel)
  - A ciklusváltozót (ha van) nekünk kell módosítani
- Lehetséges hiba: végtelen ciklust írunk
  - Ctrl+C-vel tudjuk leállítani (jó esetben)
- Szintaktika:
  `while logikai kif.[,|;]utasítás(ok), end`
  - Persze a cikluson kívül (előtte és utána) megfelelő tevékenységek adandók meg
- Példák
  - Gépi epszilon meghatározása
    ```
    > eps = 1; while (1+eps) > 1, eps = eps/2; end, eps = eps*2
    ```
    - Tudjuk: *eps* már eredetileg beépített változóként is létezik (és itt felülírjuk)
  - Melyik Hilbert mátrix determinánsa lesz először kisebb eps-nél?
    ```
    > i=1; while det(hilb(i))>eps; i=i+1; end, i
    ```

Ciklusok

- While ciklus példa: szinusz értékek előállítása Taylor-sorral (script)
  - Képlet: $$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$
- Lépések
  - A következő tag előjelváltó, x*x-szel szorzódik, n*(n-1)-gyel osztódik
  - Ha a kívánt pontosságot (*eps*) elértük, kiszállunk



Ciklusok

Specialitások

- For ciklusnál lépésköz megadható (tudjuk)
- A ciklusváltozó tetszőleges sorvektoron is végigmehet (és elemeinek értékeit felveheti)
  - Így akár oszlopvektorok is kerülhetnek a ciklusváltozóba
- Példa 1.

```
> for i=[pi, inf, nan] i, end
```

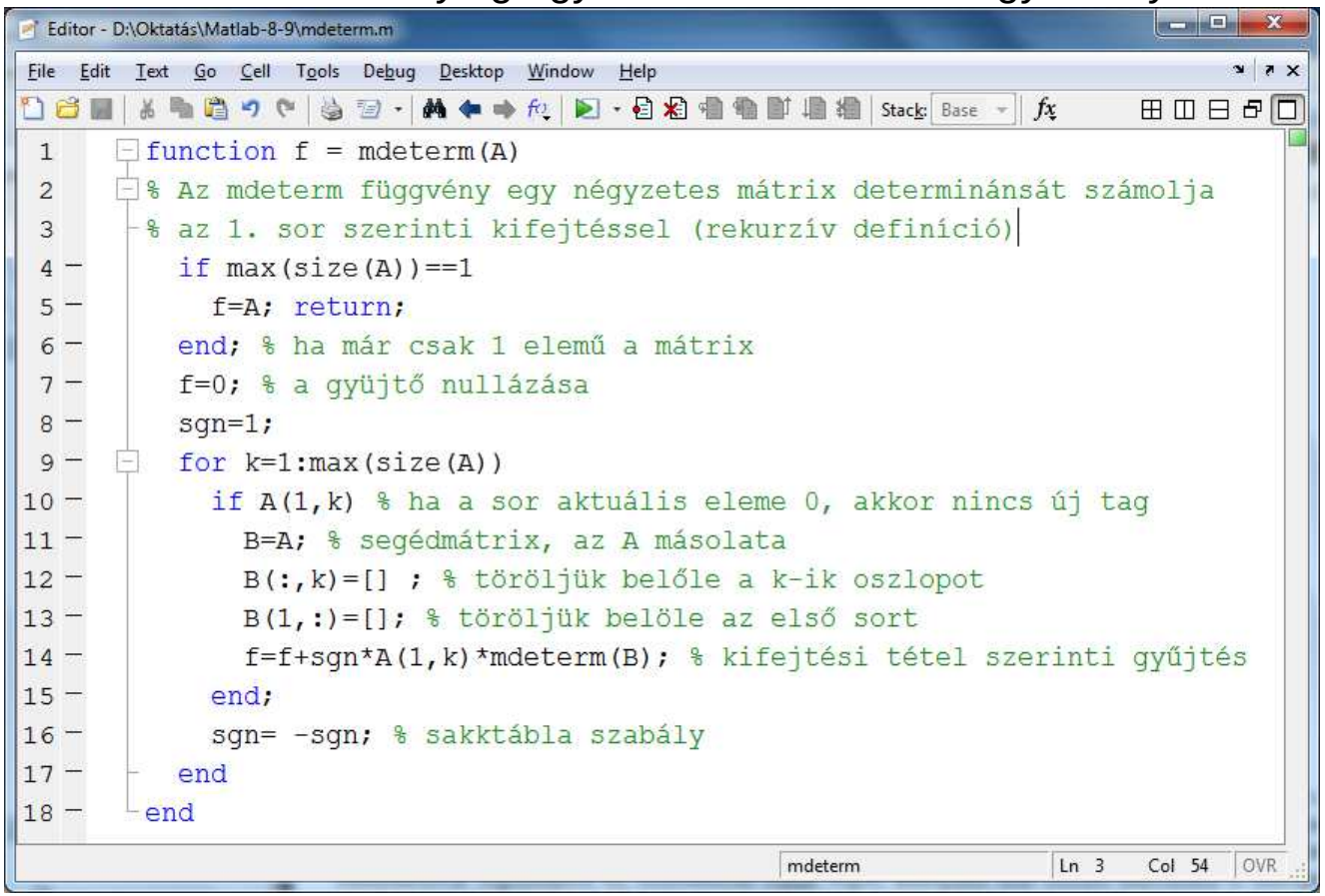
  - Mi történne akkor, ha pontosvesszőt raknánk az elemek közé? (Mi lenne i értéke?)
- Példa 2.

```
> A = rand(2,4), for x=A x, end;
```

  - Mi kerül most az x változóba?
- A Matlabban ciklusok párhuzamos végrehajtására is van lehetőség (*parfor* szerkezet), de ehhez kell a Parallel Computing Toolbox
  - Lásd: Súgó
- Nem strukturált vezérlő eszközök is léteznek a Matlabban
  - Break, continue (nem tanuljuk)

Rekurzió

- A függvény ilyenkor saját magát hívja
  - A megfelelő változók (adatok) veremben tárolódnak, ennek mélysége korlátozott (módosítható)
- Klasszikus, rekurzívan megoldható feladatok: faktoriális számítás, determináns kifejtés, Hanoi tornyai
  - A rekurzív szerkezet sok esetben viszonylag egyszerűen kiváltható hagyományos iterációval (lásd elektr. jegyzet)
- Példa: Determináns meghatározása rekurzív módon (saját fv.)
- Elemezzük az eljárást!
- Ellenőrizzük a helyes működést, és hasonlítsuk össze a beépített *det* függvénnyel



Adatbekérés és kiírás

Bekérés

- Szintaktika: változó = *input*()
  - A parancs az argumentumként megadott üzenet kiadása után a felhasználói begépelést eltárolja a változóban
  - Példa (*switch* szerkezetnél):
```
> n = uint16(input('Kérek egy pozitív egészet (max. 65535): '));
```
- Grafikus input: ginput
  - Egérkoordinátákat és gombot (egér vagy billentyű) fogad és tárol

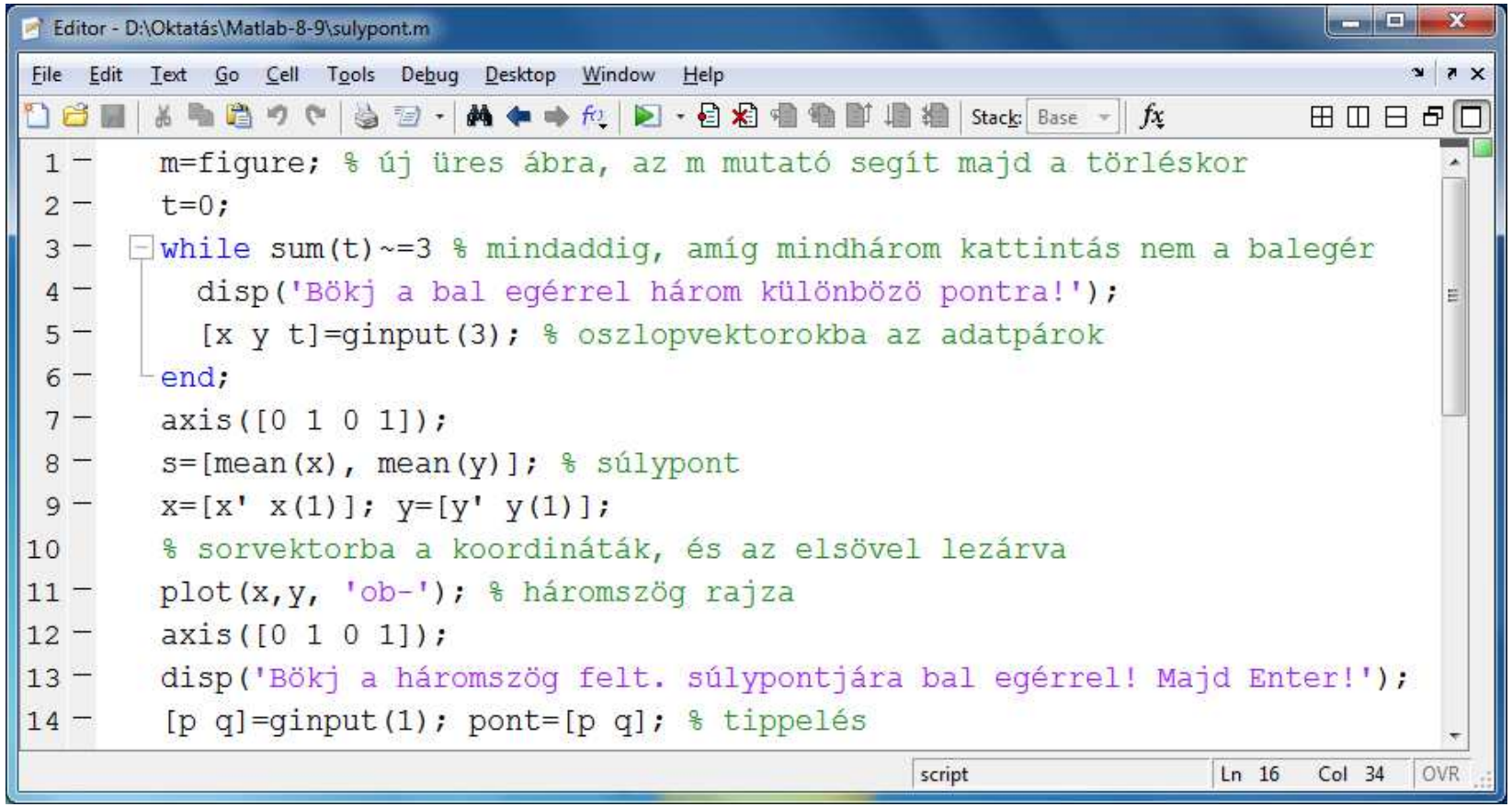
Kiírás

- A *disp* parancs egy változó értékét írja ki a változó neve nélkül
  - Konverzió szükséges lehet (pl. *num2str*)
  - Példa (*switch* szerkezetnél):
```
> disp(strcat(num2str(n), ' osztható 4-gyel'));
```
- C-s jellegűek az *fprintf* és *sprintf* függvények
  - *fprintf* – szöveges adatfolyamot állít elő, ami ha nem fájlba irányul, akkor a Command Window-ba kerül
  - *sprintf* – szintén szöveget állít elő, de az az ans változóba kerül
- Grafikus ábrán a *text* függvénnyel lehet szöveget elhelyezni (tudjuk)

Interaktív grafika

Feladat (játék):

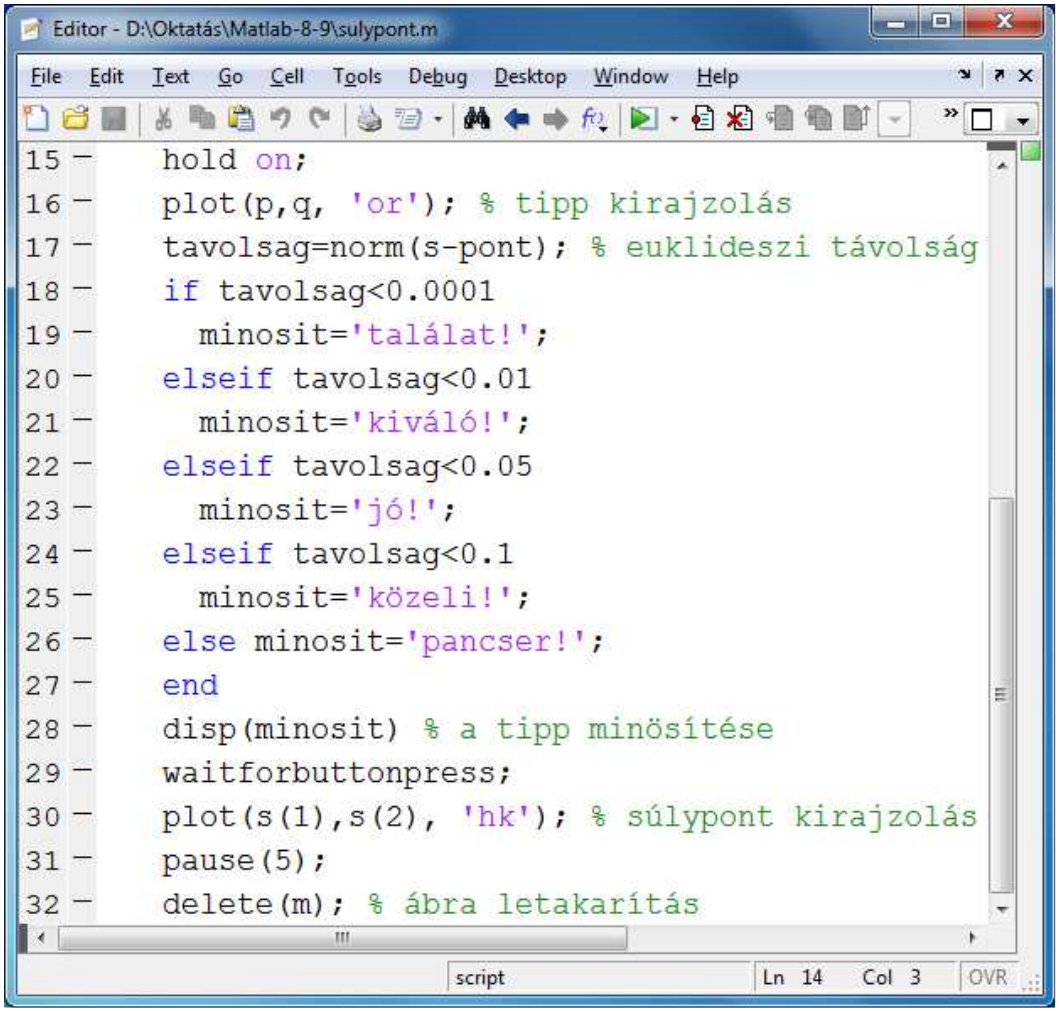
- Egy üres ábra felületén megadunk az egérrel három pontot. A felrajzolt háromszög súlypontjára kell a lehető legjobban tippelve kattintani. A súlypont megcélzása után felrajzoljuk a tippelés helyét, a tényleges súlypontot és minősítést is adunk.



Interaktív grafika

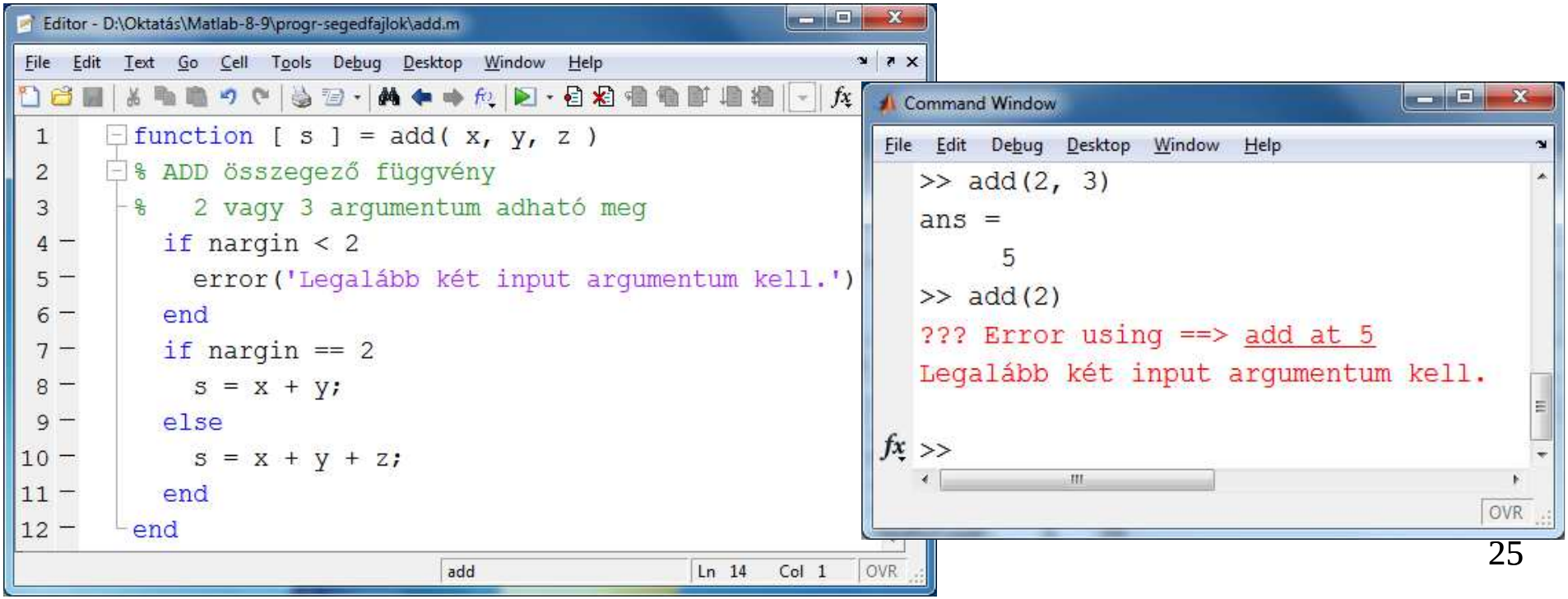
Játék (folyt.), elemzés

- *ginput* szintaktika:
  [x_koordináták, y_koordináták, billentyűkódok] = ginput(n)
  - Ahol n a kért kattintások száma, az output elemek pedig oszlopvektorok.
    A beolvasott koordináták a grafika kirajzolt intervallumába esnek.
  - A kapott billentyűkódok:
    1 – bal …
    2 – középső …
    3 – jobb egérgombot nyomtunk
    egyéb – a lenyomott billentyű ASC kódja
- Figyeljük meg a ciklusszervezést, a szelekciót, a kiírásokat és a grafikai és egyéb elemek használatát!



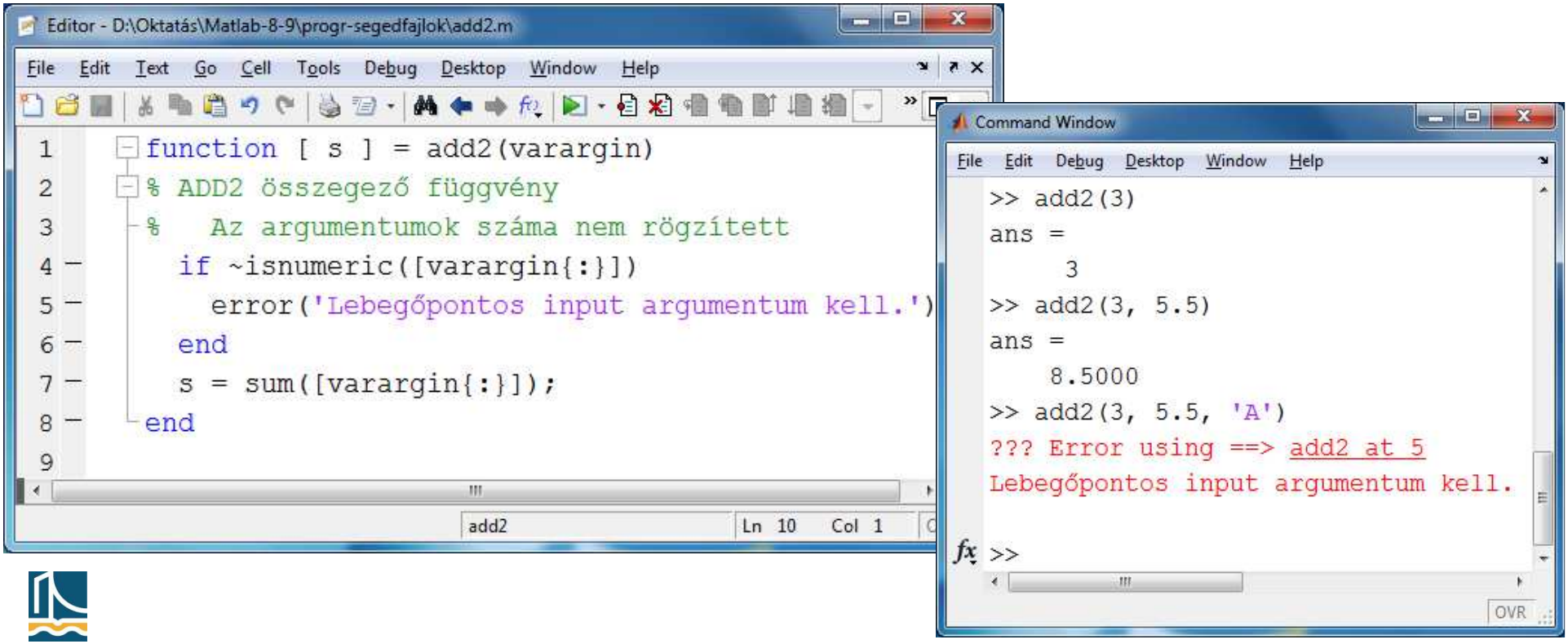
Paraméterek kezelése

- A Matlab függvényekben a paraméterek (aktuális) száma és esetleg típusa is bizonyos keretek között változhat
  - Például nem kötelező mindig minden aktuális paramétert megadni
- Fontosabb paraméterkezelő függvények
  - *nargin*, *nargout* – input/output argumentumok száma
- *error* – hibakezelés



Paraméterek kezelése

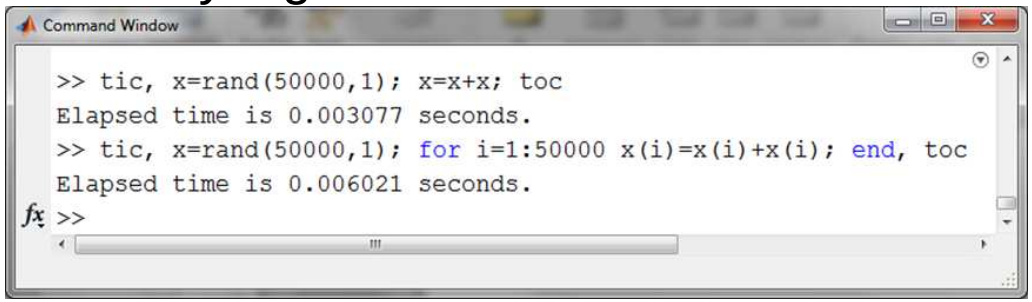
- Fontosabb paraméterkezelő függvények (folyt.)
  - *varargin* – változó számú input paraméterek listáját adja (hasonló: *varargout*); az eredmény cellatömb
    - A varargin{:} kifejezés vesszővel elválasztott listát ad vissza, konverzió kell
  - Az argumentumok típusát/értékét sok esetben ellenőrizni kell; erre szolgál többek között: *isnumeric*, *ischar* (*isstr*), *isinf*, *isnan* stb.

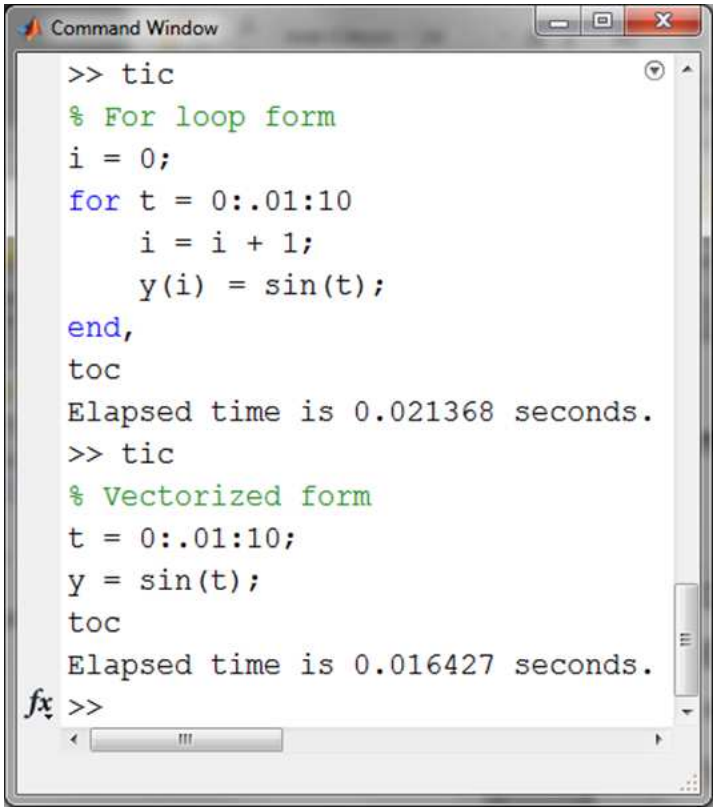


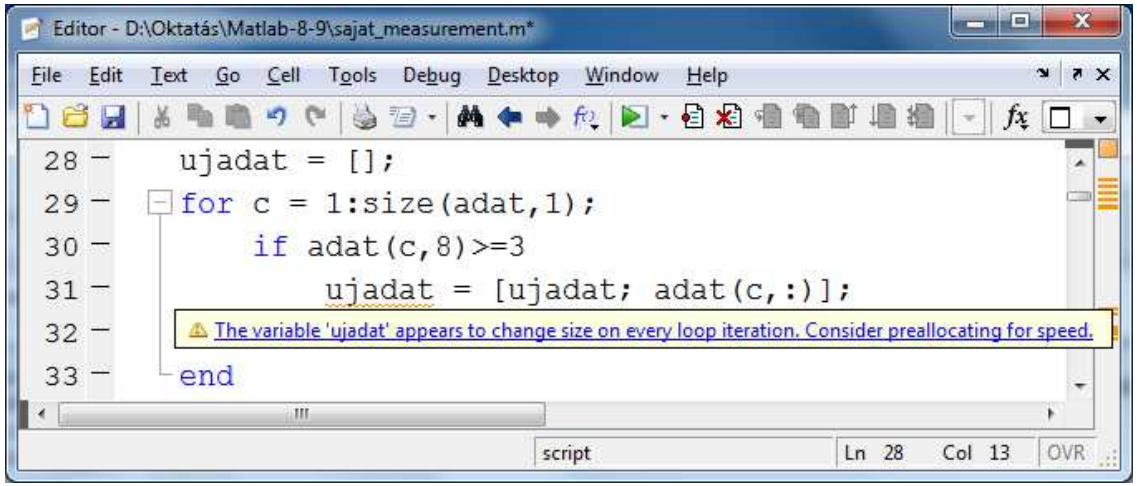
A Matlab kód hatékonysága

Néhány általános jó tanács a kód hatékonyságának növeléséhez

- *for* ciklusok vizsgálata, „vektorizáció”
  - A ciklus nem hatékonyan olvas/ír
  - A Matlab ellenben nagyon hatékony a mátrixokkal és a vektorokkal végzett műveletek végrehajtásában
- Memória előfoglalás
- Warning üzenetek elemzése és kiküszöbölése



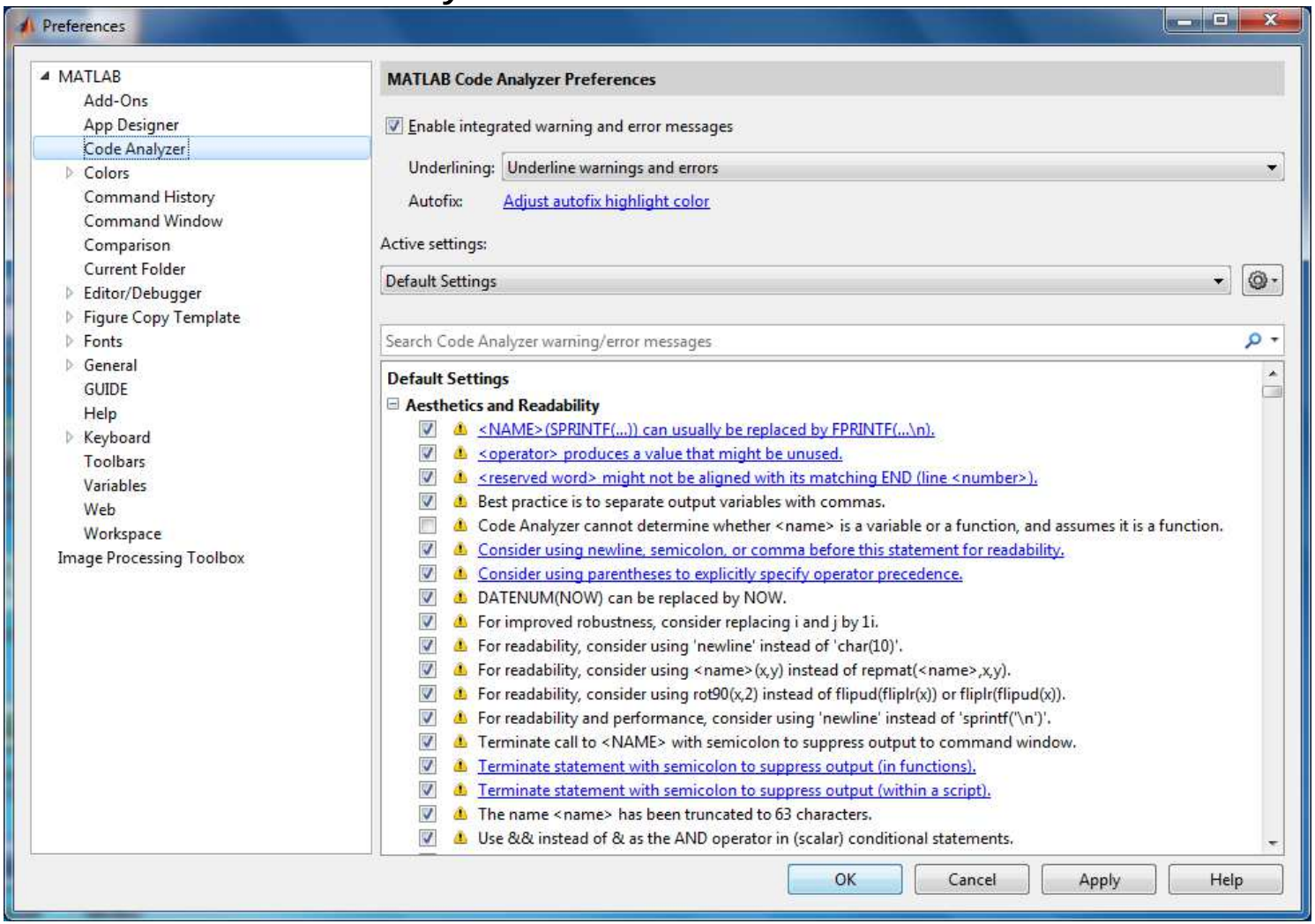




A Matlab kód hatékonysága

Fordító (kódelemző) üzenetek

- Érdemes őket tanulmányozni!



Haladó programozás, „verseny”

Euler-projekt

- Fokozatosan (viszonylag gyorsan) nehezedő feladatok
- Egy idő múlva komoly kihívás!