



7. előadás

Matlab 1.

(Ismerkedés, környezet, adattípusok)

Dr. Szörényi Miklós,
Dr. Kallós Gábor

2014–2015





Tartalom

- A Matlab általános bemutatása
- Matlab környezet
 - Ablakok, súgó rendszer
 - A Matlab, mint számológép (egyszerű Matlab-session példa)
- Mire jó a Matlab? (Ízelítő)
 - Numerikus és szimbolikus számítási segédeszköz
 - Ábrák, animációk készítése
- A Matlab jelkészlete
- Adattípusok
 - Numerikus típusok (egész, valós)
 - Értékhatárok, konverziók
 - Komplex számok
 - Szöveges, logikai, dátum/idő adatok, mátrixok és vektorok
- Értékek megjelenítési formátuma
- Változók, értékadások, kifejezések
- Alapvető parancsok (változók, I/O)

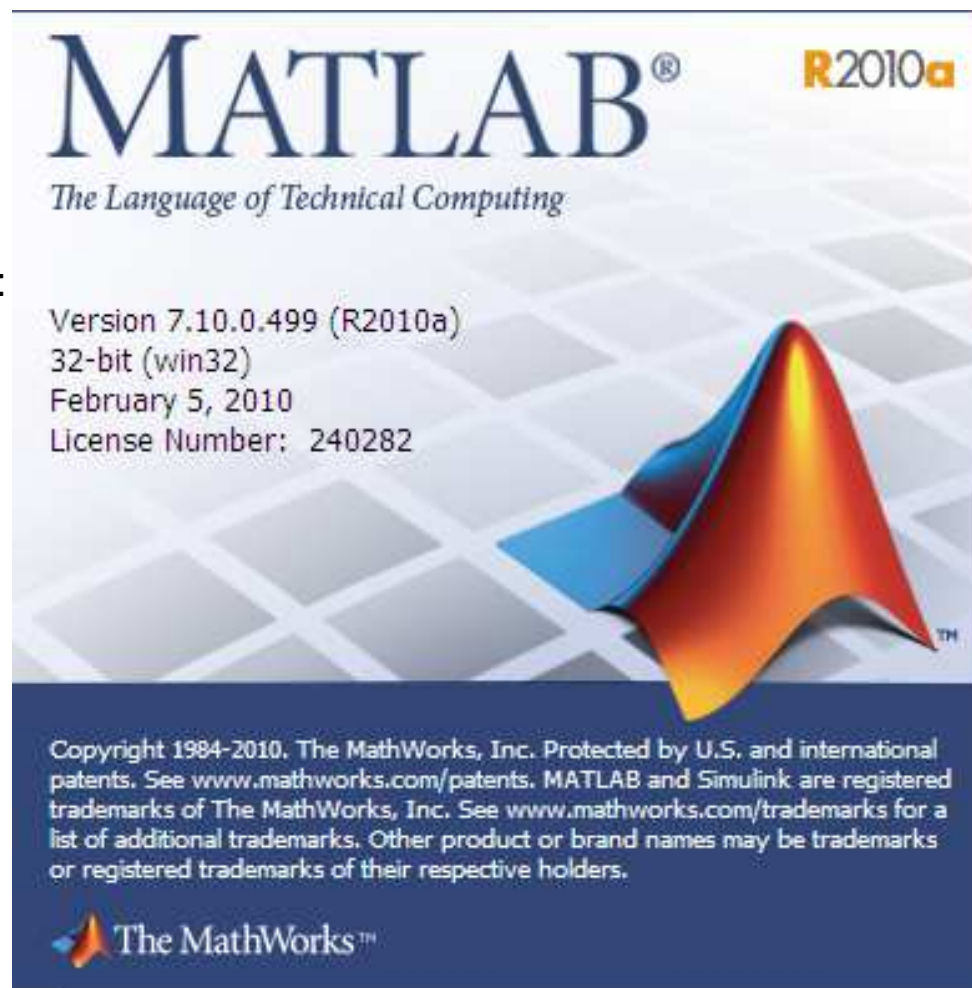




A Matlab általános bemutatása

Matlab (Mathworks)

- Integrált műszaki/technikai/tudományos számítási környezet és univerzális matematikai szoftverrendszer
- Fő profil: *numerikus számítások hatékony elvégzése, ötvözve egy fejlett grafikai/szemléltetési apparátussal és egy magas szintű programozási nyelvvel*
 - További érdekes lehetőségek (pl.): szimuláció és modellezés, interaktív dokumentumok készítése
 - Szimbolikus számítások is végezhetők
- Bátorítás: már viszonylag szerény tudással is sok probléma megoldható, és a megoldások már így is látványosak és igényesek lehetnek





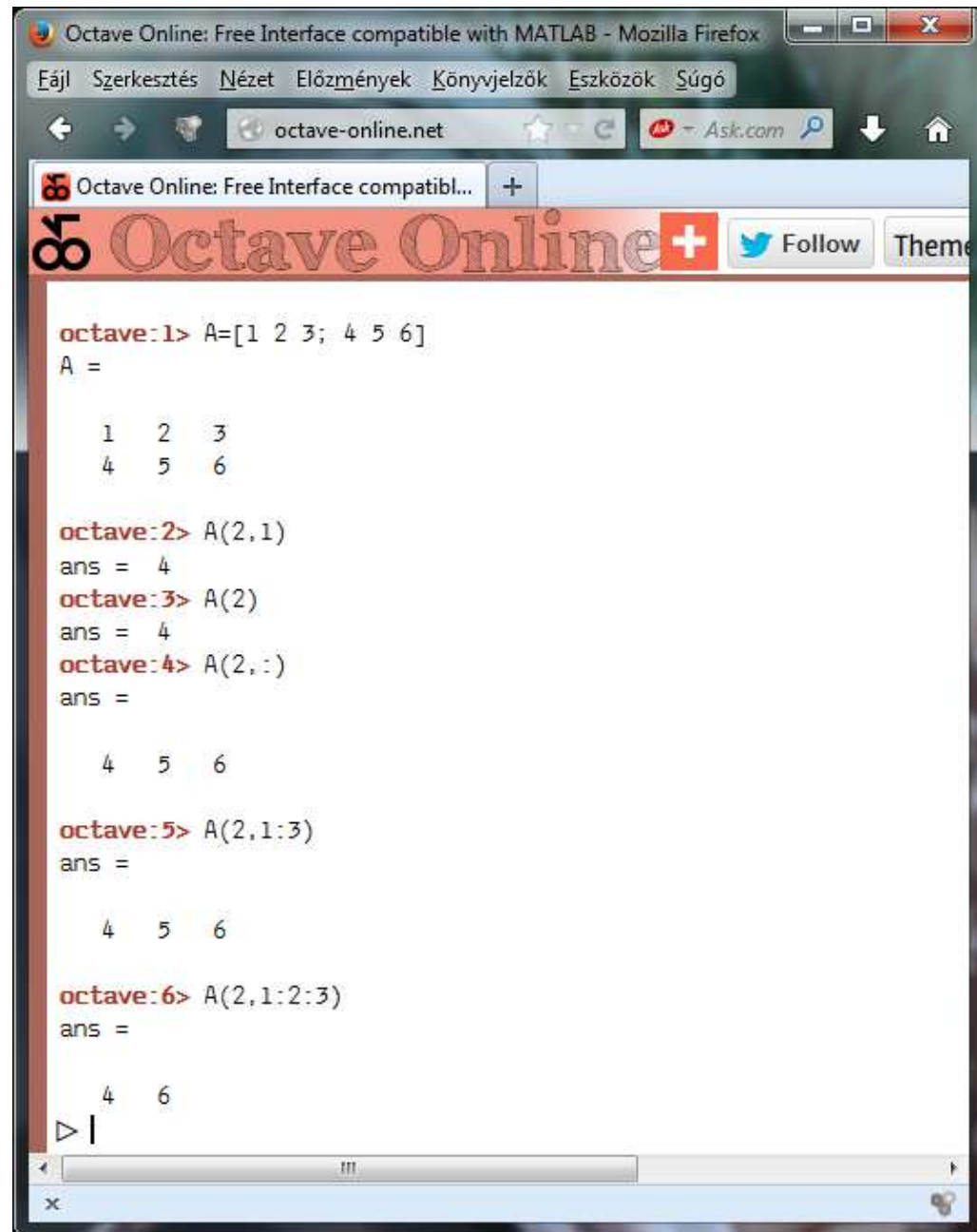
A Matlab általános bemutatása

- A fejlesztés története
 - Az 1970-es évektől folyik a fejlesztés, kezdetben oktatási segédeszköznek szánták (lineáris algebra és numerikus matematika)
 - Később az alkalmazott matematika és a tudományos számítások irányába mozdultak el
 - Napjainkban: univerzális szoftverrendszer (robusztus, nagy tudású)
 - Sokoldalú alkalmazhatóság: a Matlab üzleti siker lett!
- A használat főbb logikai szintjei
 - Interaktív (parancsvezérelt) környezet
 - Több száz beépített parancs, rengeteg mintapéldával
 - Scriptek
 - Programozás
- Bővíthetőség, kiterjeszthetőség
 - Toolboxok (pl. szimbolikus matematika, optimalizáció, szimuláció, szabályozástechnika, jelfeldolgozás, parciális diff.egyenletek, fuzzy logika, neuronhálózatok, statisztika, képfeldolgozás)
 - Saját programok
 - Kommunikációs lehetőség és átjárhatóság (C és más nyelvek)



Az Octave és a Scilab

- A Matlab drága szoftver
 - (Toolboxok...)
 - Student verzió létezik
 - Ez jóval szerényebb áron megkapható
- Ha otthon egyáltalán nincs jogunk a Matlab használatához \Rightarrow ingyenes alternatívák, helyettesítők
 - GNU Octave
 - Scilab
- Az elérhető kompatibilitás igen nagyfokú
 - (A szintaktika pontosabban igazodik az Octave programnál)



```
octave:1> A=[1 2 3; 4 5 6]
A =

    1    2    3
    4    5    6

octave:2> A(2,1)
ans = 4

octave:3> A(2)
ans = 4

octave:4> A(2,:)
ans =

    4    5    6

octave:5> A(2,1:3)
ans =

    4    5    6

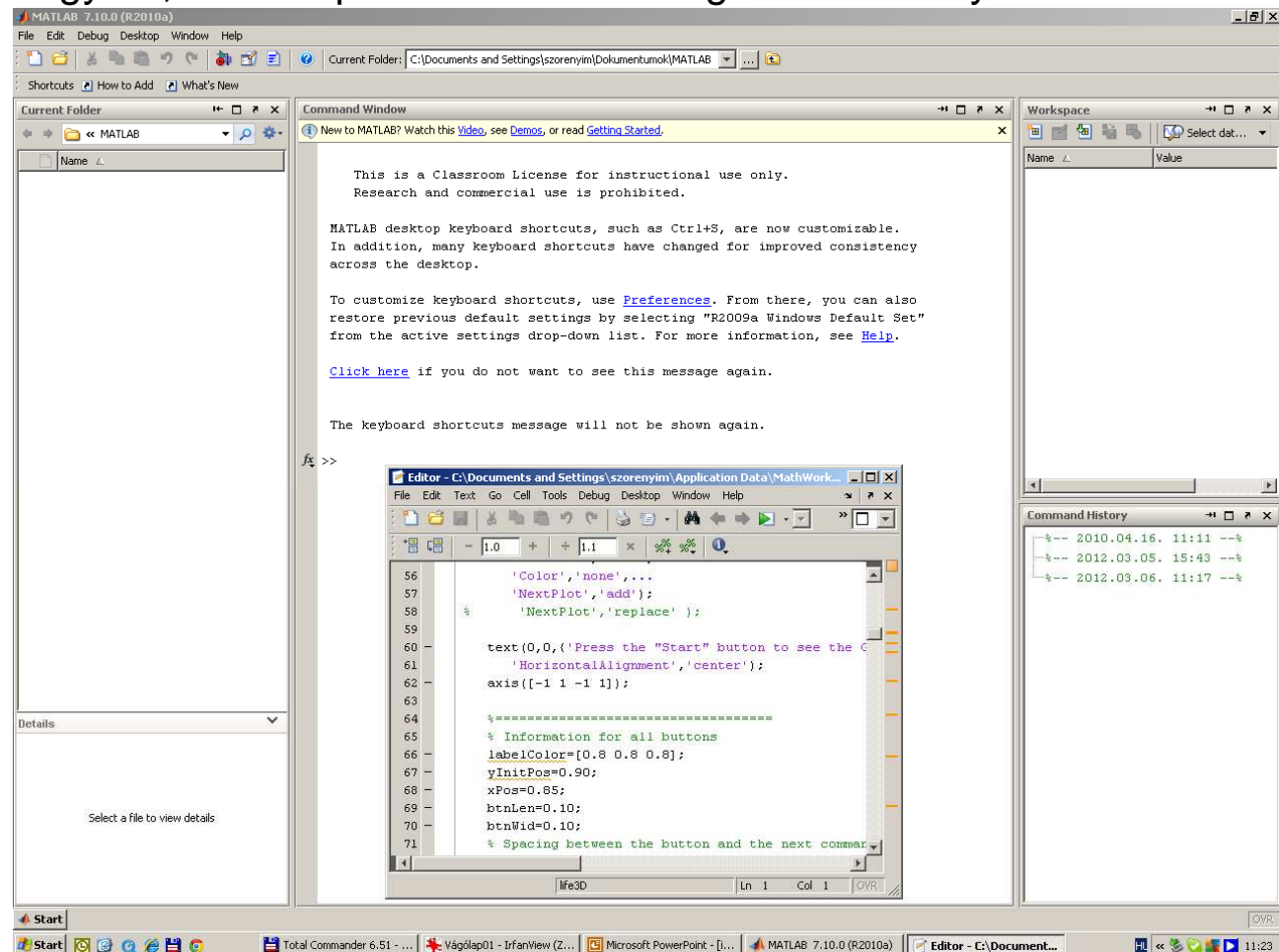
octave:6> A(2,1:2:3)
ans =

    4    6
```

Matlab környezet

Alapértelmezésben a képernyőn: 5 ablakos tagolás (módosítható)

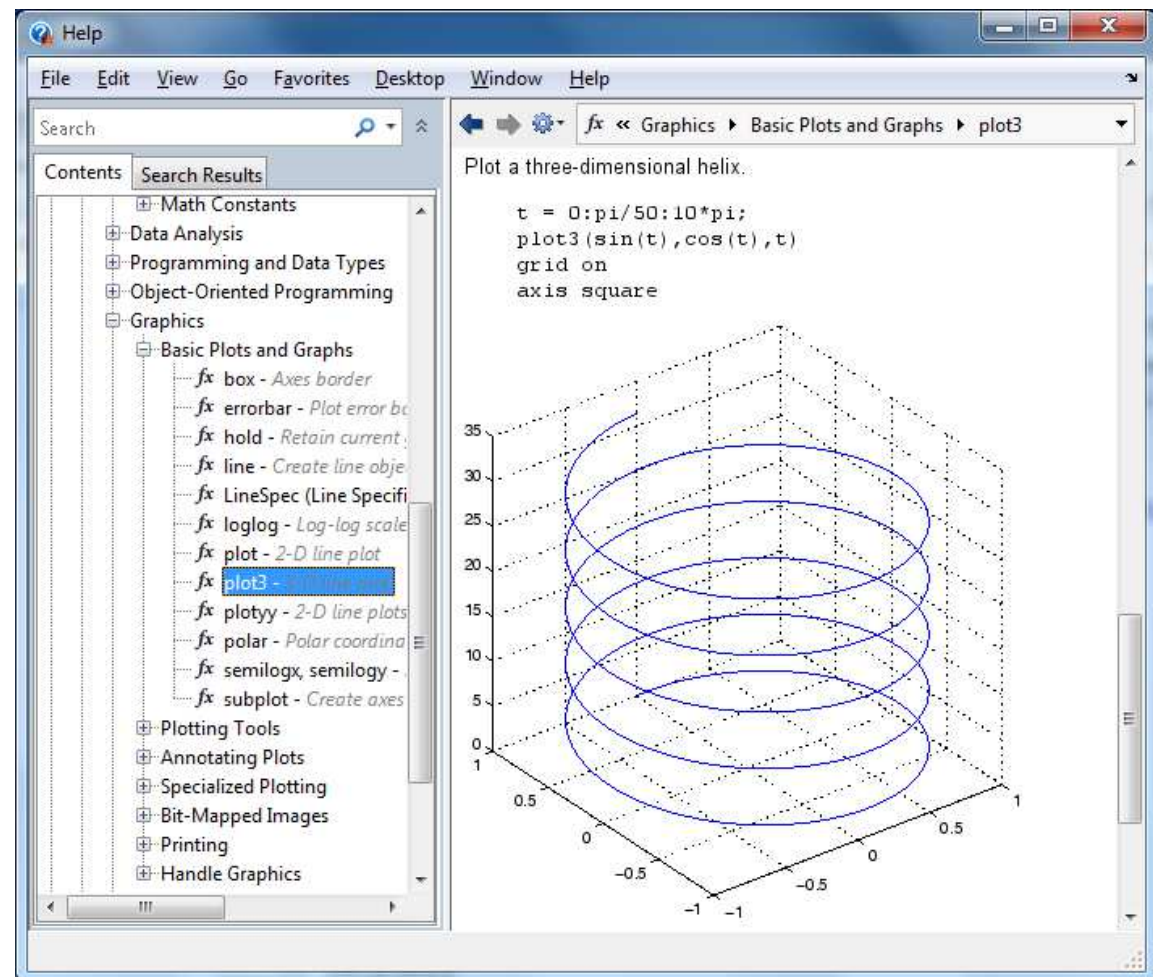
- Parancsablak (Command Window)
- Parancselőzmények (Command History)
 - A javítás egyszerű és gyors, korábbi parancsaink szükség esetén könnyen visszahozhatók
- Munkaterület-változók (Workspace)
 - [Mátrixszerkesztő (Variable Editor)]
- Aktuális könyvtár (Current Folder)
 - A munka kezdetén beállítandó (!)
- Fájlrészletek (Details)
 - [Szövegszerkesztő (Editor)]



Matlab környezet

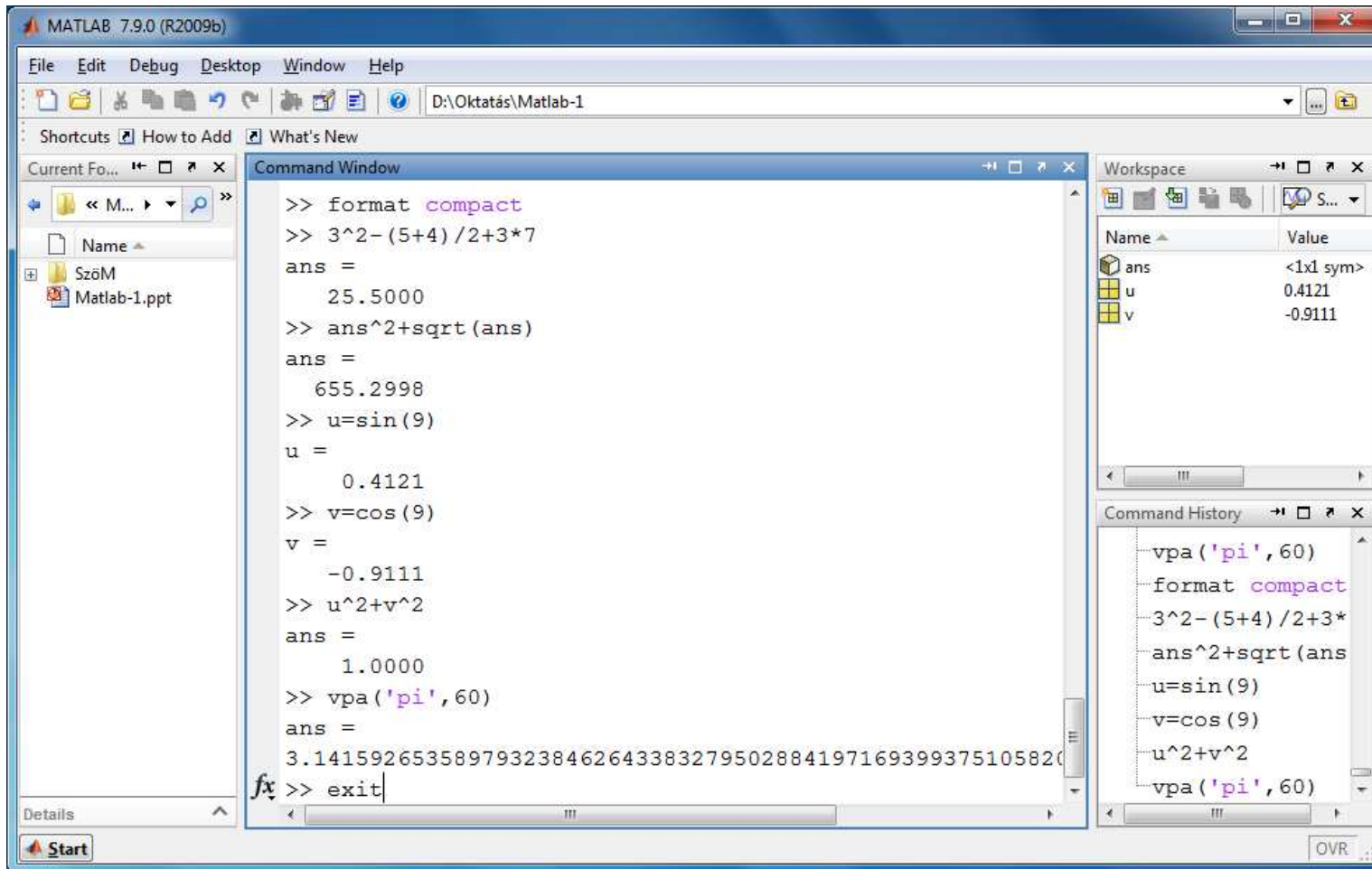
Egyes ablakok megjelenése kérhető

- Ábra (Figure)
 - A grafikus ablakok nem dokkolhatók a munkafelületre
- Súgó rendszer (Help)
 - Szövegközi segítség: `help parancs`
 - Részletes információk (külön ablakban) a kívánt parancsról, és általánosan (könyvszerűen): `help browser`
 - Aktiválása: `helpwin` *parancs* vagy `helpdesk`; *Matlab nélkül, online is elérhető a mathworks.com oldalon*
 - Dokkolhatók a munkafelületre
 - Fejlettebb interaktív help (változat): `doc parancs`
 - Matlab demók: `demo`



Matlab környezet

- Egyszerű Matlab-session példa (a Matlab, mint számológép)
 - A munka kezdetén: munkakönyvtár beállítása (!)
 - Nem kell deklarálni a változókat használat előtt
 - Kilépés előtt: mentés (változók, parancsok)
 - A kilépés lehetőségei: exit, quit, CTRL-Q



The screenshot displays the MATLAB 7.9.0 (R2009b) environment. The Command Window shows the following commands and results:

```
>> format compact
>> 3^2-(5+4)/2+3*7
ans =
    25.5000
>> ans^2+sqrt(ans)
ans =
    655.2998
>> u=sin(9)
u =
    0.4121
>> v=cos(9)
v =
   -0.9111
>> u^2+v^2
ans =
    1.0000
>> vpa('pi', 60)
ans =
3.141592653589793238462643383279502884197169399375105820
fx>> exit
```

The Workspace window shows the following variables and their values:

Name	Value
ans	<1x1 sym>
u	0.4121
v	-0.9111

The Command History window shows the following commands:

```
vpa('pi', 60)
format compact
3^2-(5+4)/2+3*
ans^2+sqrt(ans)
u=sin(9)
v=cos(9)
u^2+v^2
vpa('pi', 60)
```




Mire jó a Matlab?

1. A Matlab, mint numerikus számítási segédeszköz

- Feladat:

Oldjuk meg a következő lineáris egyenletrendszert!

$$\begin{aligned}4x_1 + 2x_2 + 0x_3 &= 10 \\ -3x_1 + 5x_2 + 2x_3 &= -10 \\ -2x_1 + 6x_2 + 1x_3 &= -10\end{aligned}$$

- (Eml.: pl. az inverz mátrixos módszer használható abban az esetben, ha $\det A \neq 0$. *Ha $\det A = 0$, akkor meg kell vizsgálni, hogy az egyenletrendszer összefüggő vagy ellentmondásos.*)

- A Matlab azonban a baloldali osztással direkt megoldást tud adni, inverz mátrix nélkül

- Megoldás:

```
>> A = [4 2 0; -3 5 2; -2 6 1], b = [10; -10; -10], x = A\b
```

- Természetesen előtte $\det A$ is meghatározható (látjuk, hogy nem 0)

- Ellenőrzés:

```
>> A*x
```





Mire jó a Matlab?

2. A Matlab, mint szimbolikus számítási segédeszköz

- Feladat:

Legyen $f(x) = a \cdot e^{-b \cdot x} \cdot \sin(c \cdot x)$

Határozzuk meg $f(x)$ primitív függvényét (határozatlan integrálját), majd deriváljuk a primitív függvényt!

- Megoldás:

```
>> clear, syms a b c x; f = a*exp(-b*x)*sin(c*x)
% A meglévő változók törlése, szimbolikus változók
definiálása
>> int_f = int(f, x)           % primitív függvény
>> diff_int = diff(int_f, x)    % ennek deriváltja
% Itt még nem kaptuk vissza az eredeti függvényt (ez az
alak jóval bonyolultabb), ezért "kényszerített"
egyszerűsítéssel folytatjuk
>> egyszerubb = simple(diff_int); % egyszerűbb alak
>> egyszerubb                  % az egyszerűbb alak kiírása
```



Mire jó a Matlab?

3. Ábrák, animációk készítése

- Feladat: Rajzoljuk fel a lemniszkátát (kétlevelű lóhere)!

- Megoldás:

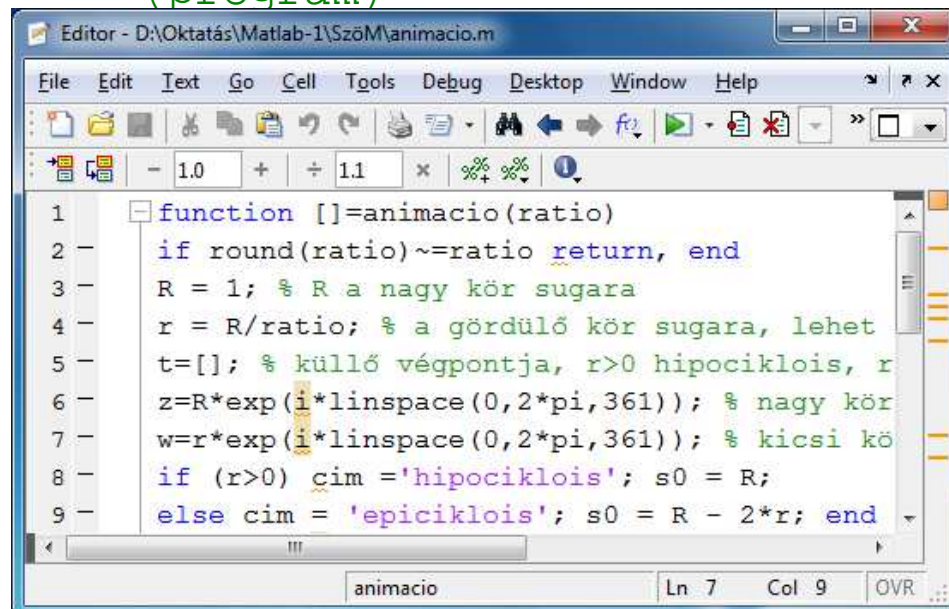
```
>> ezplot(' (x^2+y^2)^2-x^2+y^2 ', [-1,1],[-1,1]); axis square
```

- Feladat: Gördítsünk végig egy nagy körön egy kicsit. Rajzoljuk ki mozgás közben a kicsi kör egy küllőjének végponti pályáját.

- Megoldás:

```
>> animacio(5)
```

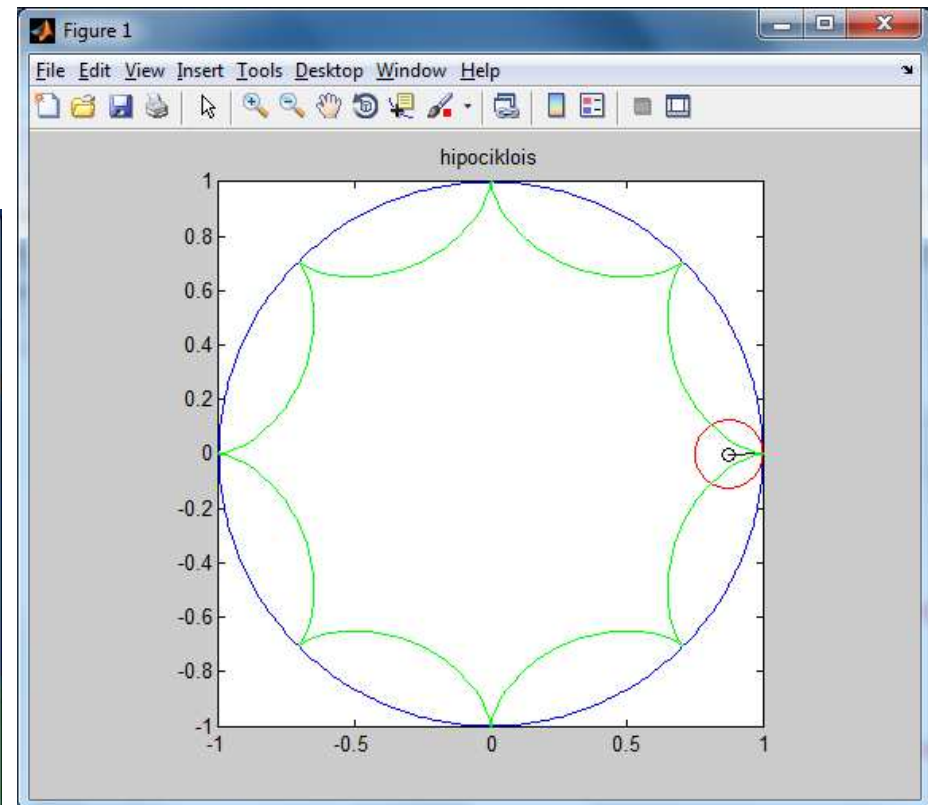
```
% saját függvény M-fájlban  
(program)
```



```

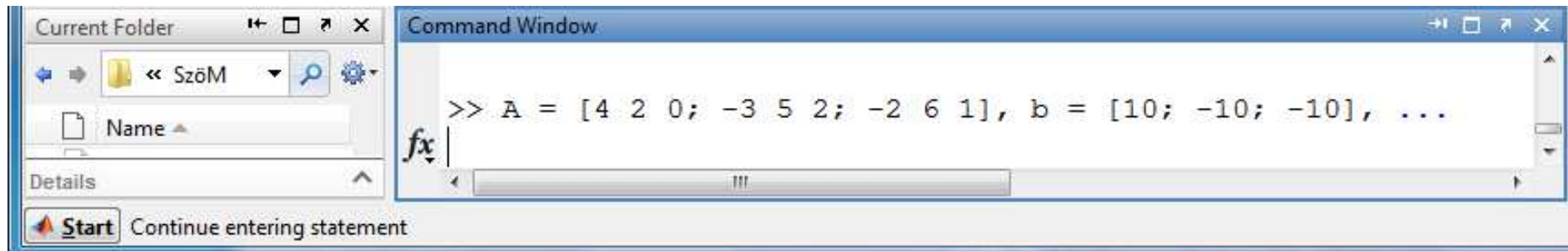
1 function []=animacio(ratio)
2 if round(ratio)~=ratio return, end
3 R = 1; % R a nagy kör sugara
4 r = R/ratio; % a gördülő kör sugara, lehet
5 t=[]; % küllő végpontja, r>0 hipociklois, r
6 z=R*exp(i*linspace(0,2*pi,361)); % nagy kör
7 w=r*exp(i*linspace(0,2*pi,361)); % kicsi kö
8 if (r>0) cim='hipociklois'; s0 = R;
9 else cim='epiciklois'; s0 = R - 2*r; end

```



A Matlab jelkészlete

- A Matlab környezetben használható írásjelek
 - Az angol ABC kis és nagybetűi, amelyek között a Matlab *különbséget tesz!*
 - Magyar és más nemzeti karakterek nem használhatók! (Eredmény: hiba)
 - Számjegyek
 - Space
 - Speciális jelek: `_ . , : ; < > / \ * ^ ~ = () [] { } ' ! @ & | %`
 - A `%` jel után kommentárt írhatunk
 - A `„;”` pontosvessző parancszáró írásjel, ha kiteszük, akkor a parancs eredménye nem kerül a képernyőre; egyébként az eredmény, amely a megjelölt változóba, vagy automatikusan az **ans** (answer) változóba kerül, rögtön ki is jelződik
 - A `„,”` vessző szeparátorjelként szolgál
 - Hosszú parancssor a `„...”` sorozat után új sorban folytatható





Adattípusok

Numerikus adatok (egész, valós)

- A numerikus adatok tárolása automatikusan (alapértelmezetten) az ún. **double** lebegőpontos típusban (lásd IEEE 754 szabvány) történik, de a Matlab további numerikus típusokat is használ (kényszerítéssel)
- Az egyes tárolási típusok határai

típus	intmin('típus')	intmax('típus')	bájt
int8	-128	127	1
uint8	0	255	1
int16	-32768	32767	2
uint16	0	65535	2
int32	-2147483648	2147483647	4
uint32	0	4294967295	4
int64	-9223372036854775808	9223372036854775807	8
uint64	0	18446744073709551615	8
	realmin('típus')	realmax('típus')	
single	1.175494e-038	3.402823e+038	4
double	2.2250738585072e-308	1.79769313486232e+308	8

- Ha egy típusban túlcsoordulás történik, akkor vagy valótlan érték képződik (az ebben a típusban tárolható értékek aktuális határa), vagy egy speciális jelentésű kód, az **Inf**





Adattípusok

Numerikus adatok (egész, valós; folyt.)

- Fontos ismernünk a következőket

- Típuskényszerítés, szándékos konverzió
- Automatikus konverzió
 - Pl. ha a és b közül legalább az egyik egész, akkor közöttük a műveletek csak akkor definiáltak, ha vagy azonos típusúak, vagy az egyikük double; az eredmény pedig az egész típusba kerül
- Túlcsordulás, értékhatárok (legalább becslés szinten)
- Számítási pontosság, gépi epszilon
- NaN (Not a Number) és Inf
- Típus lekérdezése: class fv.

```
Command Window
>> egesz = int8(-10.89)
egesz =
    -11
>> elojel_nelkul = uint8(-10.89)
elojel_nelkul =
     0
>> a = uint8(8), b = -5.2 % b double
a =
     8
b =
   -5.2000
>> a+b % az eredmény uint8
ans =
     3
>> c = uint8(3)/uint8(5) % kerekítés felfelé!
c =
     1
>> 2/0
ans =
   Inf
fx >>
```





Adattípusok

*Lebegőpontos aritmetika

- A legtöbb lebegőpontos szám normalizált alakban tárolódik: $x = \pm(1 + f) \cdot 2^e$
 - Itt $0 \leq f < 1$, azaz (52 biten) $0 \leq 2^{52} f < 2^{52}$, $-1022 \leq e \leq 1023$
 - Mantissza: a pontosságot korlátozza, exponens: az ábrázolható tartományt korlátozza
 - A tárolásnál felhasználható 52 bit f -re, 11 bit e -re és 1 bit az előjelre
 - (Az $(1 + f)$ -ből az 1 nem tárolódik; e helyett $e + 1023$ tárolódik)
- Gépi epszilon (1-eps.): $2^{-52} \sim 2,2 \cdot 10^{-16}$
 - Még éppen $1 + \varepsilon \neq 1$
 - Ez a max. relatív távolság, ami előfordulhat „szomszéd” ábrázolható számok között
 - Másként: a Matlab kerekítési (hiba)szintje 16 decimális jegy
 - Értelmezhető 2-eps, 3-eps stb.!
- Példa: az $1/10$ tárolása
 - Elméleti érték $\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} + \dots$
 - A legközelebbi 52 biten tárolható szám $t = 2^{-4} \left(1 + \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \dots + \frac{9}{16^{12}} + \frac{10}{16^{13}} \right)$
 - Azaz $e = -4$ és $f = \frac{9}{16} + \frac{9}{16^2} + \frac{9}{16^3} + \dots + \frac{9}{16^{12}} + \frac{10}{16^{13}}$
 - Nézzük meg az $1/10$ -et és $1 + \varepsilon$ értéket format hex kijelzéssel! (3fa = 1018)
- További érdekes kérdések: csak nem normalizáltan ábrázolható számok (v. ö.: realmin), különleges kódok (Inf, NaN)





Adattípusok

Komplex számok

- A Matlab a komplex számokat is kezelni tudja, ha azokat normál alakban adják meg: $4 + 3i$
Az i szimbólum az imaginárius egységet jelöli, azaz $i = \sqrt{-1}$
(Helyette a j szimbólum is írható, de a Matlab kicseréli i -re)
- Példa

```
>> z = 4 + 3i, R = abs(z), z*z', fi = angle(z)
```

 - Az `abs` és `angle` függvények polárkoordinátás áttéréshez használhatók
- Az i szimbólum felüldefiniálás esetén sem keveredik össze az i változóval, a használat módja dönti el az értelmezést

```
>> i = 5, z = 4 + 3i, w = 4 + 3*i
```

 - Az i eredeti funkciója a `clear` paranccsal visszaállítható
- Komplex szám exponenciális alakban is megadható:
$$R \cdot \exp(i \cdot fi) = R \cdot \cos(fi) + i \cdot (R \cdot \sin(fi)),$$
ahol R a komplex szám abszolút értéke és fi a radiánban mért szöge
- A Matlab ügyesen tud számolni komplex számokkal (összevonások, egyszerűsítések), például

```
>> a = -7 + 22i, b = 2 + 3i, c = a/b
```
- További komplex számokat kezelő függvények:
`isreal()`, `real()`, `imag()`, `conj()`





Adattípusok

Szöveges adatok

- A Matlab a szöveges adatokat az írásjelekhez rendelt kódszámok sorozatával (sorvektor) tárolja
 - A kódszámok 2 bájtban tárolódnak, azaz a maximális kódszám 65535 lehet
- Az alap kódkészlet az ASCII kódkészlet, ennek nyomtatható része 32-től 127-ig terjed, és az angol ábécé betűit, számjegyeket és a billentyűzet speciális jeleit tartalmazza
 - A sorrend megfelel az ábécé sorrendnek
 - Egyes műveletek is ennek megfelelőek, lásd később is (pl.: `st = 'alma'`, `ujst = st + 1`, `uint8(st)`)
- Megjelenítés példa

```
>> ascii = char(32:127)
>> kod = uint16(ascii)      % más egész vagy valós típus is jó
```
- A magyar ékezetes betűk kódjai nem tükrözik az ábécé sorrendet, a szöveg rendezése a belső kód szerint történik (ami nem tökéletes)
- A rendezés bemutatása

```
>> ekezetes = 'öüóőúéáúíőüóőúéáúí' % sztringkonstans megadása
>> whos ekezetes % infók lekérdezése
>> novekv = sort(ekezetes) % nem tökéletes ábécé szerinti
rendezés
>> novekvkod = uint16(novekv)
```





Adattípusok

Mátrixok és vektorok

- A numerikus adatokat a Matlab – alapértelmezés szerint – mátrixszerkezetbe (számtáblázat) helyezi el
 - Egy mátrixnak n sora és m oszlopa lehet ($n, m \geq 1$)
- A skalár számok egyetlen adatot tartalmazó 1×1 méretű mátrixként tárolódnak
- A vektorok egy sorból vagy egy oszlopból álló mátrixok
- Egy mátrix egy értékadó parancs segítségével az elemeinek [] zárójelpárban történő felsorolásával adható meg, ahol a sorok végét a „;” jelzi
- Egy soron belül az elválasztásra – nem kötelező módon – a vessző jelek használhatók
 - A mátrix típusa az elemei által meghatározott típus lesz (!)
- Példa

```
>> B = [1 2 3; 4 5 6; 7 8 9], C = [1, 2, 3; 4, 5, 6; 7, 8, 9]
>> whos B, C
```

 - Nem konzekvens megadásnál: hibaüzenet
 - (Pl.: az 1. sorban 3 elem, a 2. sorban 2 elem)
- Természetesen egy mátrix más módokon is feltölthető (képletek segítségével vagy akár külső fájlból is; lásd később)
- (Cellatömb adattípus)





Adattípusok

Logikai adatok

- A logikai **igaz** és **hamis** értékek tárolására a szokásos 1 és 0 numerikus értékeket használja a Matlab
 - Ezek is mátrixok
- Példák

```
>> sin(pi/4) == sqrt(2)/2
>> 'Matlab' == 'Matek '
ans =
     1     1     1     0     0     0
```
- A 2. esetben írásjelenkénti hasonlítás és értékelés történik
 - Mit kapunk, ha elhagyjuk a szóközt a Matek szó után?
- [Numerikus értékek, sőt vektorok és mátrixok is kiértékelhetők logikailag (szelekciós szerkezet vagy logical függvény)]
 - Skalár esetében (és a logical függvénnyel) minden nem 0 érték igaznak számít, és csak a 0 számít hamisnak
 - Mátrix esetében csak akkor **true** a logikai értékelés (szelekció), ha minden eleme nem 0, egyébként **false** a logikai érték (lásd jegyzet)]
- Logikai értékekkel feltölthetők mátrixok és vektorok
- Példa

```
>> B = true(2, 3)
>> whos B
```
- (További példák a mátrixműveleteknél)





Adattípusok

Dátum és idő adatok

- A dátum-idő tárolására a Matlab double típusban tárolt valós számot használ: az 1 jelenti a 0000 január 1-et, és törtszámokkal adhatjuk meg a napon belüli időpontot pl. 0.625 a délután 3 órának felel meg (lásd **datetime** fv.)
- A dátum-idő megjelenítése az Excelhez hasonlóan többféle lehet
- A **now()** függvény a pillanatnyi dátum-időt adja vissza, amit a **datevec()** és **datestr()** függvényekkel átalakíthatunk:

```
>> most=now, dt_str=datestr(most), dt_vec=int16(datevec(most))
most = 7.3567e+005 % napok száma 0-tól 2014. márc-ig
dt_str = 09-Mar-2014 21:38:01
dt_vec = 2014 3 9 21 38 2 % év, hó, nap,
óra, perc, másodp.
```

- A **dt_vec**-hez hasonló sorvektort eredményez a **clock** beépített változó is:

```
>> int16(clock)
```

- Az eltelt idő mérését belső változók, műveletek (**tic**, **toc**, **cputime**) támogatják:

```
>> x=sqrt(3); tic, for i=1:10^8 x=x+1.0000001; end, toc; ...
>> x=sqrt(3); tic, for i=1:10^8 x=x/1.0000001; end, toc;
```

- Az összeadás és a szorzás műveleti időigénye közel azonos, az osztásé lényegesen több





Változók, védett alapszavak

- A Matlab változói betűvel kezdődő betűszámsorok lehetnek (aláhúzás karaktert is tartalmazhatnak, angol ábécé)
- De: a kulcsszavak védettek, nem használhatók
 - Ezek: break, case, colon, continue, else, elseif, end, for, if, otherwise, switch, while
- Parancsszavakat, függvényeket, belső változókat sem célszerű adatazonosítóként felhasználni (ha mégis felvesszük (törlés): `clear név`)
 - Pl. clear, dir, exit, format, help, load, save, eps, realmin, realmax, intmin, intmax, pi, Inf, NaN, ...
- A munkaterületi változók listázása: **who** parancs, részletes lista: **whos** parancs (maszkolva is használható)

```
>> whos
```

Name	Size	Bytes	Class	Attributes
ans	1x1	8	double	
dt_str	1x20	40	char	
dt_vec	1x6	12	int16	
szoveg	1x6	12	char	
uint32max	1x1	4	uint32	
x	1x1	1	logical	
z	1x1	16	double	complex

- A munkaterületi változók törlése: **clear** parancs (szintén maszkolható)

```
>> who e* % e betűvel kezdődő változók nevei  
>> clear e* % e betűvel kezdődő változók törlése
```





Értékek megjelenítési formátuma

- Az egész értékek (a double típusban tárolt is) megjelenítése 9 jegyű számokig pontos, ennél több jegy esetén a rendszer a tudományos formátumot használja
- Példa

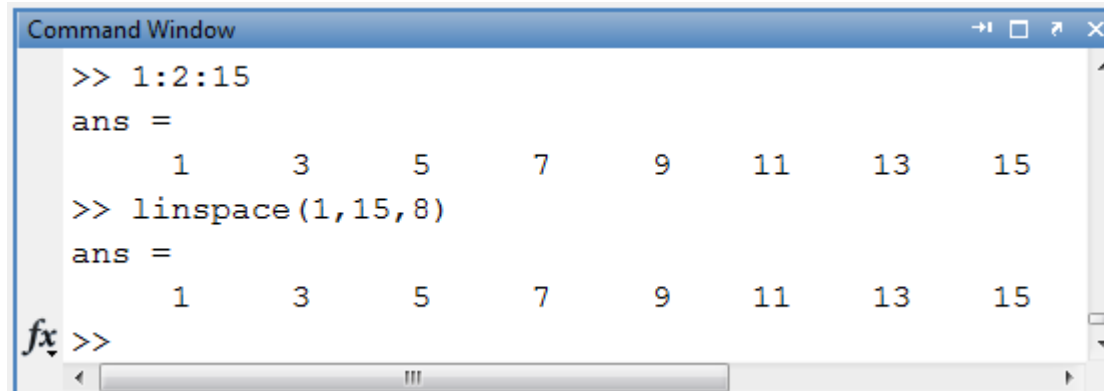
```
>> 123456789, 1234567890
ans =
    123456789
ans =
    1.2346e+009
```
- Lebegőpontos értékeknél a **format long** parancs 15-16 értékes jegyű kiírást biztosít
- A **format short** parancs visszakapcsol az alapértelmezett „short” kijelzésre (4 tizedes)
- Az exponens rész csak akkor kerül kiírásra, ha nem 0. Az exponens mindenkor kiírását az e paraméter biztosítja:

```
>> format long e; pi
ans =
    3.141592653589793e+000
```
- Néhány további formátum:
 - **format compact**, elnyomja a felesleges és extra soremeléseket
 - **format hex**, a tárolt érték hexadecimális megjelenítése



Műveleti jelek és függvények

- (Lásd még később is)
- **Logikai műveletek**, függvények azonos méretű mátrixokra, elempáronként hajtódnak végre
 - `==, <, >, <=, >=, ~=, &, |, ~, xor(A,B), any(a), all(a)`
- **Logikai műveletek skalárokra**
 - `&&, ||`
- **Mátrix-aritmetikai műveletek**
 - `+, -, *, /` (osztás jobbról), `\` (osztás balról), `^` (jobb o.: skalár; szorzás saját magával), `'` (konjugált transzponálás)
- **Tömb-aritmetikai műveletek** (elempáronként!)
 - `.*, ./, .\`, `mátrix.^mátrix`, `mátrix.^konstans`, `konstans.^mátrix`, `mátrix.'` (csak transzponálás)
- **Sorozatképzés**
 - `1:10, 5:2:15, 0:0.01:10`
 - A `linspace` paranccsal is megvalósítható



```
Command Window
>> 1:2:15
ans =
     1     3     5     7     9    11    13    15
>> linspace(1,15,8)
ans =
     1     3     5     7     9    11    13    15
fx >>
```




Értékadások, kifejezések, hasonlítások

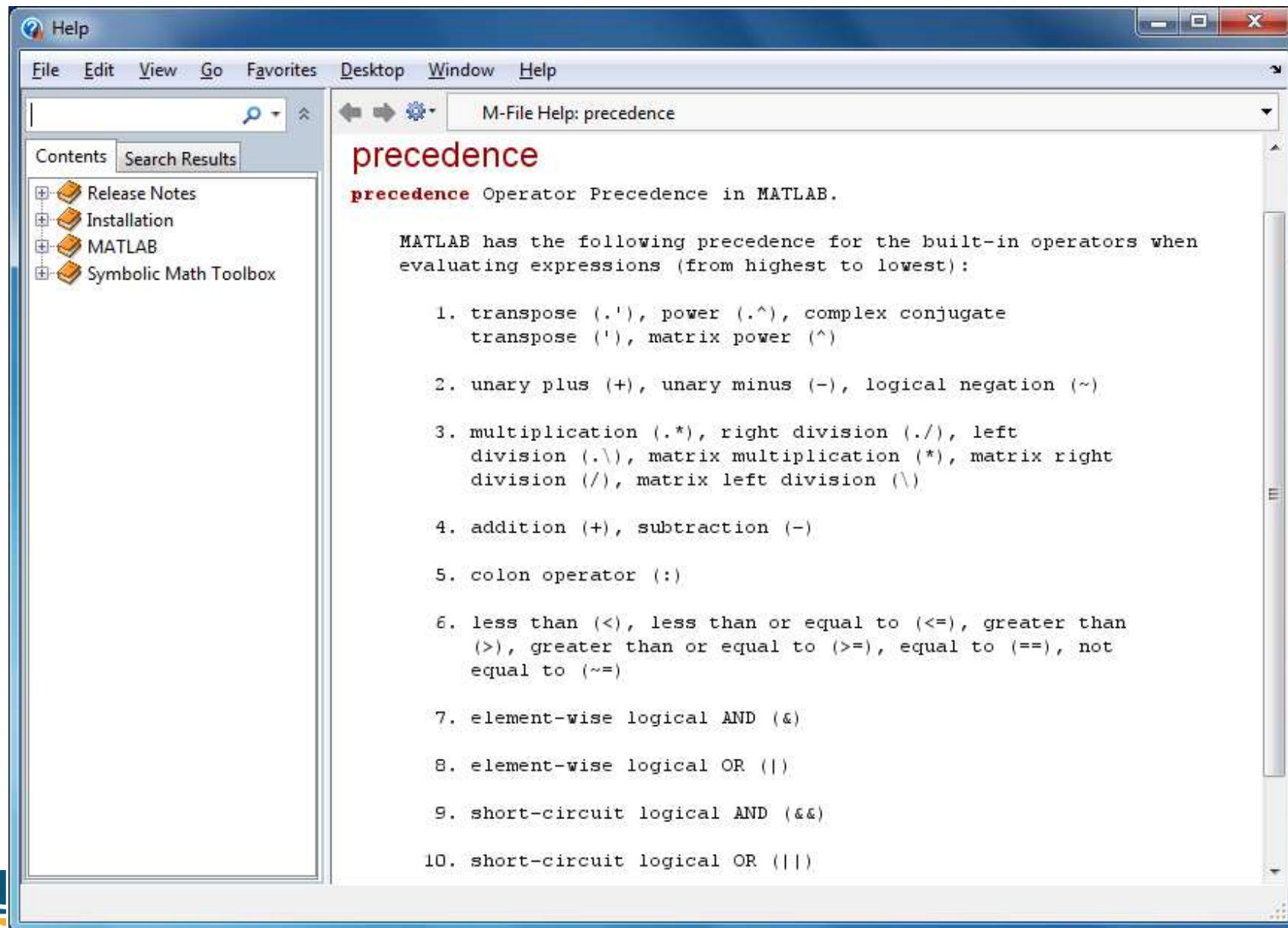
- A létrehozott változók értékeivel, konstansokkal, beépített függvényekkel kifejezéseket írhatunk be, amelyeket a rendszer kiértékel (már tudjuk)
 - A precedencia szabályok a megszokottak, ill. a súgóban megnézhetők
- **% értékadások**

```
>> sz = 1/3 - 1/2, st = sin(pi/6), e = exp(1), e2 =  
exp(2), egys = log(e)  
>> sz2 = sym('1/2')-sym('1/3')
```
- **% hasonlítások**

```
>> st > sz  
>> e2 == e*e      % nem számol pontosan!!  
>> ell = e2 - e*e  
>> s_pi = single(pi), s_pi == pi  
>> különbs = pi - s_pi  
>> 'alma' == 'alfa'      % írásjelenként értékel (tudjuk)  
>> strcmp('alma', 'alMa') % kisbetű-nagybetűre érzékeny  
>> strcmpi('alma', 'alMa') % kisbetű-nagyb.re nem érzékeny  
>> abs(sin(pi/4) - 0.5*sqrt(2)) < eps % pontossági ell.  
■ Értelmezzük a következőt:  
>> 3<0<2
```



Értékadások, kifejezések, hasonlítások



Alapvető parancsok (változók, I/O)

- A létrehozott változók értékeit bináris vagy szöveges fájlba lehet menteni és onnét visszatölteni (aktuális könyvtár)
- Példa 1.


```
>> save test.mat % a kiterjesztés elhagyható
>> clear % változók törlése
>> load test.mat
>> save e_valt.mat e*
% csak az e-vel kezdődő változók
```
- Példa 2.


```
>> save a.dat a -ascii
% egy változó mentése txt fájlba (többet is lehet)
>> load a.dat
% egy (!) változó visszatöltése txt fájlból, a fájlnev azonosítja a változót!
```

