



# 10–11. előadás

Matlab 4–5.

(Függvények, függvényábrázolás)

Dr. Szörényi Miklós,  
Dr. Kallós Gábor

2014–2015





## Tartalom

- Példák: clockex, házikó transzformációi
- Elemi függvények
- Saját függvények definiálása és hívása
  - Egyszerű definíció (parancssori)
  - M-fájlos definíció
- Függvényábrázolás
  - A grafika alapjai
  - Egyváltozós függvények
    - plot, subplot
    - fplot, ezplot
  - Szimbolikus megadású függvények
  - Többváltozós függvények, 3D grafika
- Függvényvizsgálat: zérushely, szélsőérték
- Határozott integrál



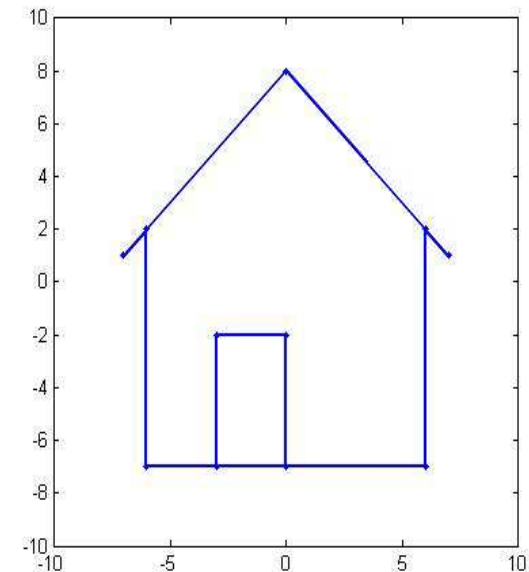
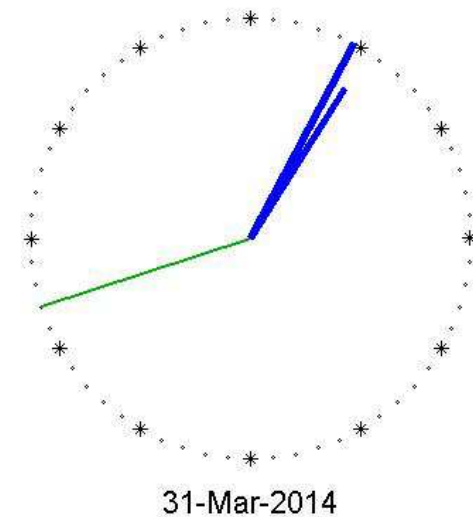


## Példák

- clockex: analóg óra
  - Grafikus objektumokat használ
  - Készítő: Cleve Moler
    - Részletek: clockex.m fájl
- Házikó transzformációi
  - Síkbeli pontsorozatot összekötünk
    - „Csinosítva”: dot2dot.m fájl
    - Az ábra még hangolható
  - Transzformációk mátrixszal
    - Forgatás (lásd saját fv.)
  - Új pozícióban kirajzolás

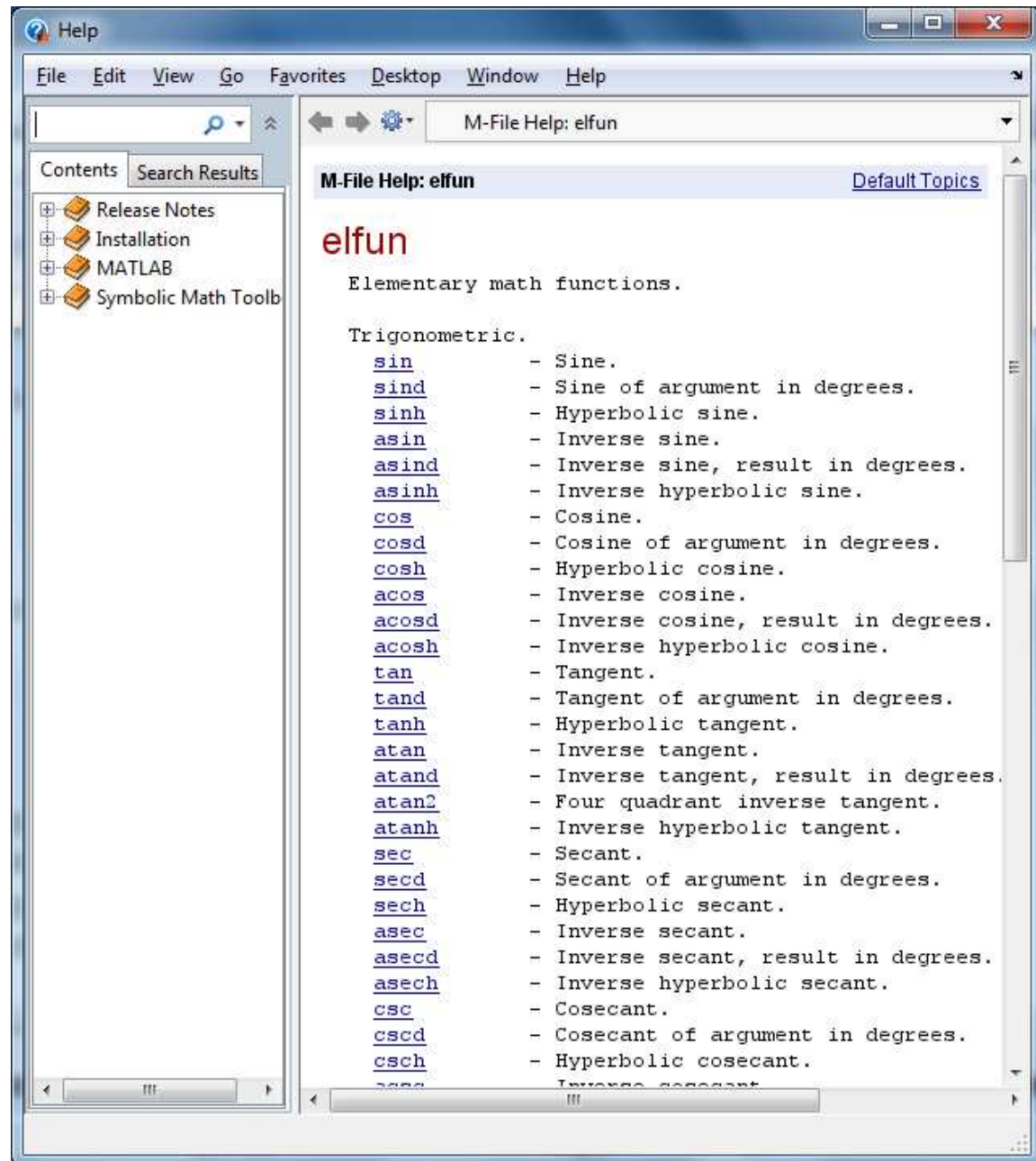
- Bemutató

```
>> X = [-6 -6 -7 0 7 6 6 -3 -3 0 0; ...  
        -7 2 1 8 1 2 -7 -7 -2 -2 -7]  
% a házikó pontjai  
>> plot(X(1,:),X(2,:))  
% pontok kirajzolása  
>> X(:,end+1) = X(:,1);  
% korrekció (összekötés)
```



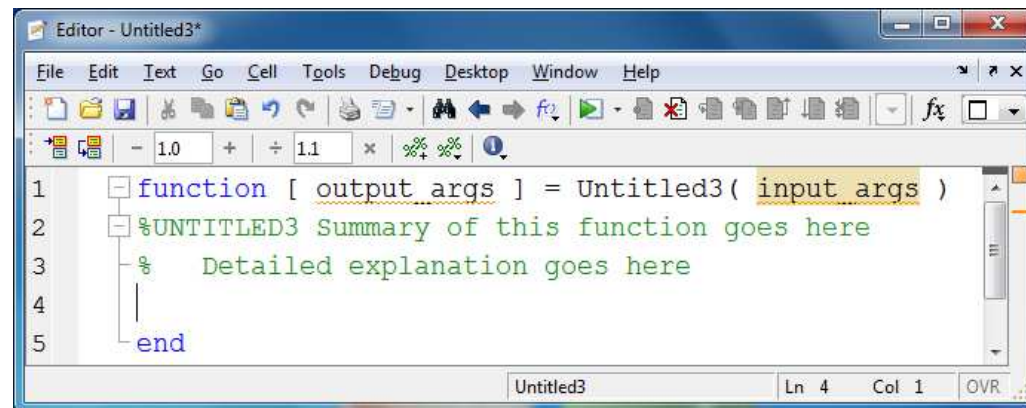
## Elemi függvények

- Elfun csoport
  - A Mathematics kategórián belül
- Alcsoportok
  - Trigonometrikus ...
  - Exponenciális ...
  - Komplex ...
  - Kerekítő ...
  - Diszkrét matematikai függvények



## Saját függvény létrehozása és hívása

- A függvények definícióit célszerűen M-fájlokban helyezzük el
- Létrehozás: File/New/Function
  - (Ugyanitt script létrehozása is kérhető – üres M-fájl)
- A függvények szintaktikája a Matlabban:  
**function** kimenő paraméter(ek) = **függvényténév**(bemenő paraméterek)  
utasítások;  
**end** % függvényténév
- A „Matlab” kottát ad a kitöltéshez
  - A felesleges sorok törlendőek
  - Figyeljünk a paraméterekre! (aktuális, formális, bemenő, kimenő)
- Fontos szabályok
  - A függvény neve kötelezően azonos a tartalmazó M-fájl nevével
  - Ha több visszatérési/kimenő paraméter van, akkor szögletes zárójelet kell használni a megadásnál
  - Paraméterek szeparálása: szóköz, ill. ,



```
1 function [ output_args ] = Untitled3( input_args )
2 %UNTITLED3 Summary of this function goes here
3 % Detailed explanation goes here
4
5 end
```

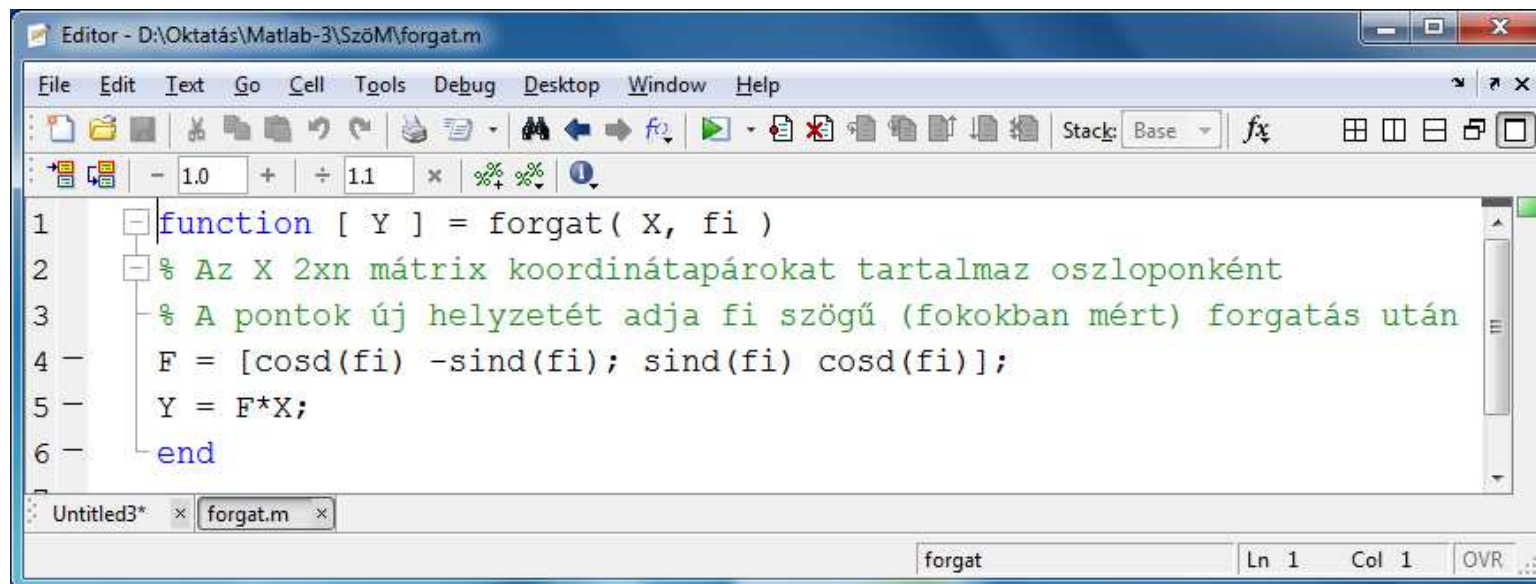
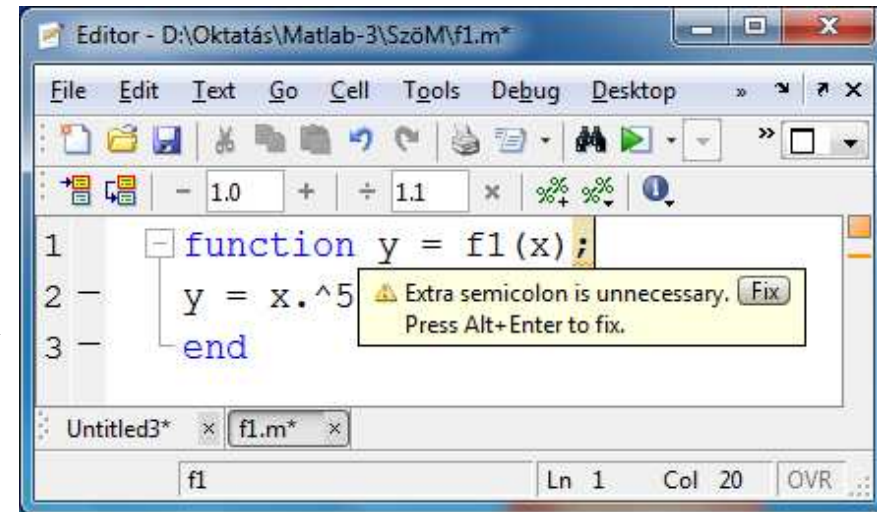


## Saját függvény létrehozása és hívása

(M-fájlos definíció, folyt.)

- Kezelőfelület
  - Egyszerű, barátságos szerkesztő
- A Matlab a beírás közben jelzi a hibákat, és segítséget ad a javításhoz
  - Tipikus hiba a pontozott műveletek nem-alkalmazása (amikor szükségesek lennének)
  - Használjunk kommenteket!
- Példa: forgatófüggvény
  - Hívás:

```
>> for i=0:360 dot2dot(forgat(X,i)), pause(0.1), end;  
% függvények egymásba is ágyazhatók
```





## Saját függvény létrehozása és hívása

A Matlab három lehetőségének összehasonlítása

- Függvénydefiníció M-fájlban definiált függvénnyel
  - Példafüggvény:  $fv(x) = 1/x + 2x^2$
  - Tudjuk: az osztást és a négyzetre emelést a ./ illetve .^ művelettel végezzük
  - `function y = fv(x)`  
    `y=1./x+2*x.^2;     % kell a pontosvessző, főleg rajzolásnál`  
    `end`
  - Hívás:  
    `>> x = 0.01:0.01:5; plot(x, fv(x))`
- Közvetlen függvénydefiníció (parancssorban, ún. anonim megadás)
  - A @ jel használatával
  - `>> fw = @(x) x.^2.*sin(x) + 1`
  - Hívás:  
    `>> s = -3:0.1:3; plot(s, fw(s))`
- Inline megadás
  - A függvény egy sztringben megadott legális kifejezés
  - `>> f = inline('3*sin(2*x^2)', argnames(f), fplot(f, [0 pi]))`  
    % az argnames itt biztonsági ellenőrzésre szolgál  
    % fplot esetében a . elhagyható a megfelelő műveletek előtt

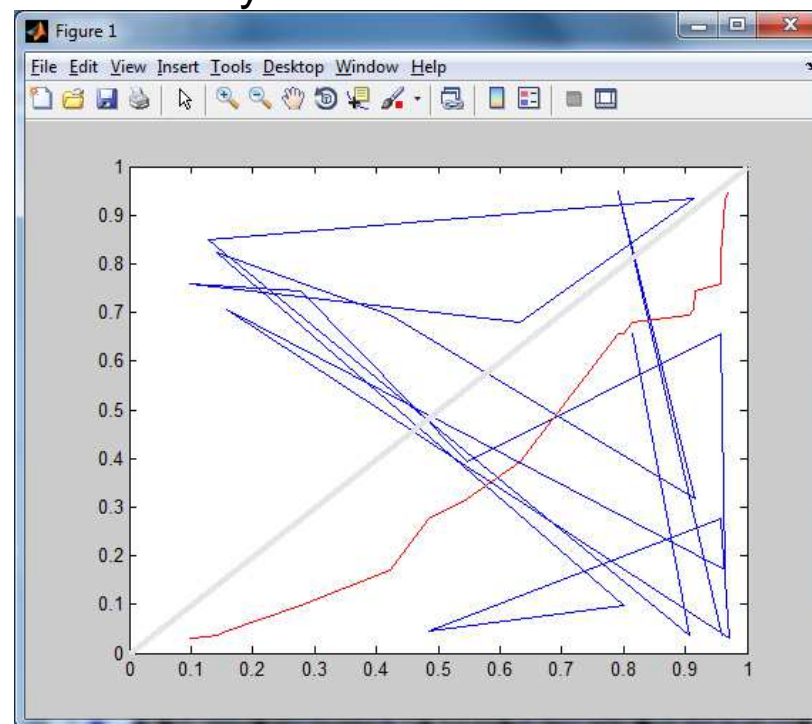






## Grafika – áttekintés

- A Matlab a rajzokat grafikus ablakban készíti el
  - A `figure(n)` paranccsal válthatunk a már létező, vagy még nem létező ablakok között (ekkor új ablak jön létre)
- Egy grafikus ablakban sok objektum helyezhető el, amelyek mindegyike hangolható
- A **plot** paranccsal síkbeli pontokat rajzolhatunk
  - Külön vektorban adjuk meg az egyes pontok x ill. y koordinátáit
  - Alapértelmezés szerint a pontok folytonos vonallal összekötöttek
  - A plot parancs harmadik paraméterében opciókat adhatunk át
    - Három különböző tulajdonságot adhatunk meg egy-két karakter hosszú kódértékekből összeállított sztringben
    - A tulajdonságok együtt vagy külön is megadhatók
- Példa (kód: lásd jegyzet) →







## Grafika – áttekintés

- A plot parancs lehetséges harmadik paraméterei

- Példa

```
>> x = rand(1,3), y = rand(1,3), x(1,4) = x(1,1),  
y(1,4) = y(1,1), plot(x,y, '*-.b')
```

kód	vonall stílusa	kód	szimbólum	kód	szín
-	folytonos (alapértelmezés)	+	+	r	piros
--	szaggatott	o	kör	g	zöld
:	pontosított	*	*	b	kék
-.	pont-vonal	.	pont	c	cián
		x	x	m	magenta
		s	négyszög	y	sárga
		d	rombusz	k	fekete
		^	háromszög fel	w	fehér
		v	háromszög le		
		>	háromszög jobbra		
		<	háromszög balra		
		p	ötágú csillag		
		h	hatágú csillag		





## Grafika – áttekintés

- Részletesebb beállításokat végezhetünk el, ha a plot parancsot az alábbi szintaxissal hívjuk meg:  
`plot(x, y, tulajdonság neve, tulajdonság értéke)`
- Több tulajdonságot is megadhatunk név–érték párokat ismételve
- Néhány fontosabb:

Color	A vonal színe	Színkód (lásd előző táblázat), vagy RGB kódolással: 3 számérték (vektorban), 0 és 1 közötti valós számok
LineStyle	A vonal stílusa	Mint a fenti táblázatban: - - : - .
LineWidth	A vonal vastagsága	Az érték megadható

- További tulajdonságok (pl. jelölőre vonatkozó beállítások, színek):  
lásd súgó, ill.  
`h = plot(x, y, ...), get(h), set(h, 'tulajdonság neve', tulajdonság értéke)`
- Példa  

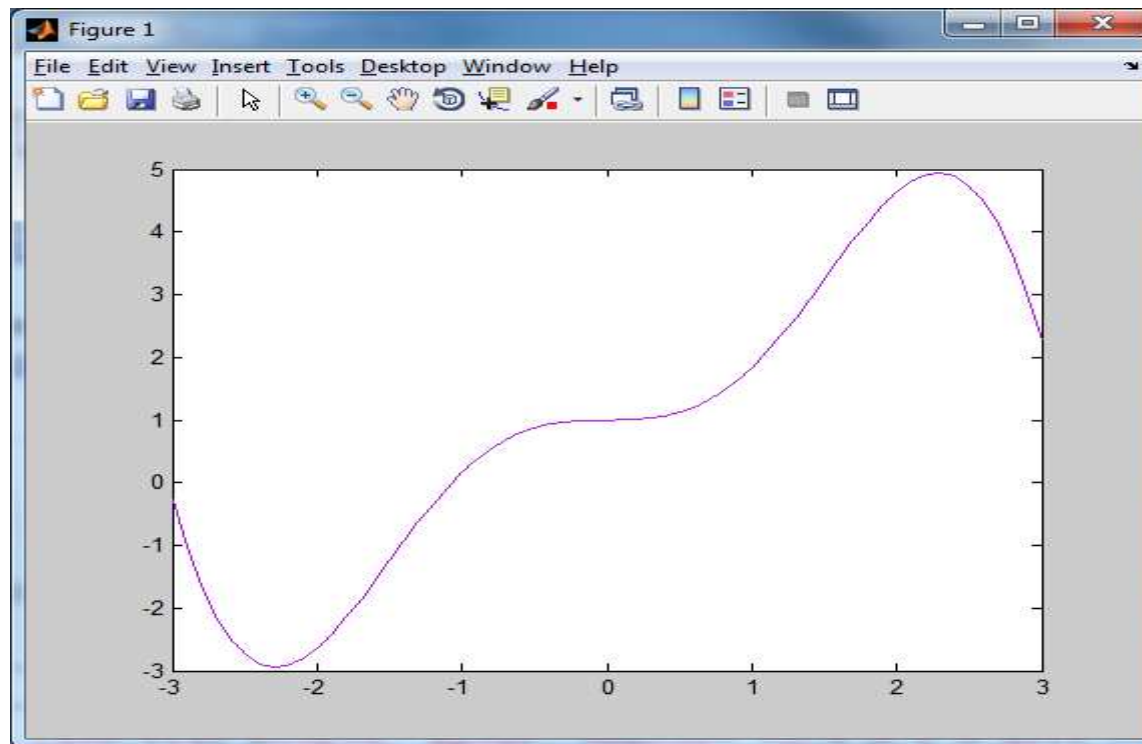
```
>> x1 = linspace(-2*pi, 2*pi, 101);  
plot(x1, sin(x1), 'LineWidth', 2, 'Color', [.8 .3 0]);
```



## Függvényábrázolás (plot)

- Az előzőek szerint:
  - Elkészítjük az alappontokat (vektor) és a függvénydefiníciót (valamelyik megismert módon)
  - Megadjuk a rajzoló utasítást az esetleges opciókkal
  - Ha több rajzot szeretnénk egy ábrára helyezni, akkor ezt is beállítjuk (hold on)
- Példa (az előző anonimusan megadott függvénye)  

```
>> s = -3:0.1:3; plot(s, fw(s), 'Color', [0.7 0.1 0.9])
```

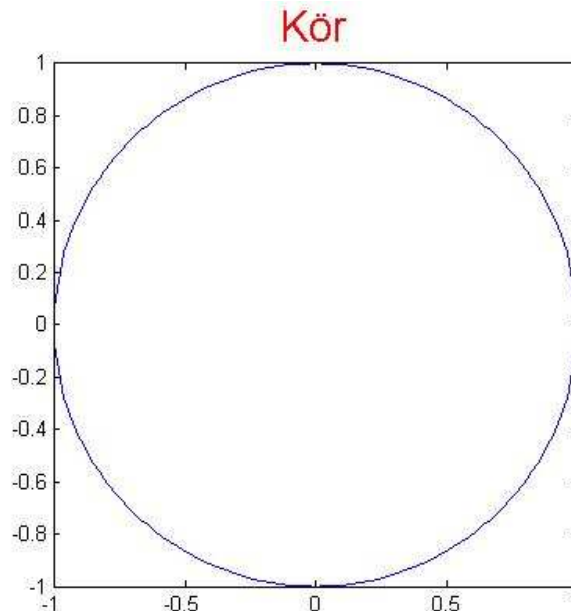




## Függvényábrázolás (plot)

- A koordinátpárokat komplex számsorral is megadhatjuk (paraméteres ábrázolás)
- Példa

```
z = exp(i*linspace(0, 2*pi, 181)); plot(z);  
axis square;  
% az ábra méretezését négyzetesre alakítjuk  
h = title('Kör', 'Color', 'R');  
% feliratot teszünk az aktív rajzra  
% az objektum mutatót feljegyezzük  
get(h);  
% megjelenítjük az objektum hangolható tulajdonságait  
set(h, 'FontSize', 20) % beállítjuk a betűméretet
```





## Függvényábrázolás (plot)

A rajzok készítéséhez használható fontosabb parancsok

- **clf**: grafikus ablak tartalmának törlése
- **axis([xmin xmax ymin ymax])**: a koordinátarendszer határainak beállítása a megadott értékekre
- **axis auto**: a határok visszaállítása az automatikus beállításra
- **axis equal**: az x és y irányú egység azonos hosszúságú
- **axis normal**: az x és y irányban automatikus nyújtás/összenyomás megengedett
- **axis off/on**: a koordinátarendszert elrejt, bekapcsolja
- **grid on/off**: rács be- és kikapcsolása
- **title('cím')**: grafika címének magadása
- **xlabel('felirat'), ylabel('felirat')**: x/y-tengely felirata
- **text(x,y,'string')**: adott pozícióra kiír egy szöveget
- **legend('string', ..., poz.)**: jelmagyarázat megadása
- **hold on/off**: a meglevő ábrába helyezi el a következő függvényrajzot, vagy újba
- **figure(n)**: új ablak létrehozása, illetve váltás egy létező ablakra, ahol n az ablak mutatója (mindig pozitív egész)
- **gcf**: az aktuális ablak mutatóját adja vissza

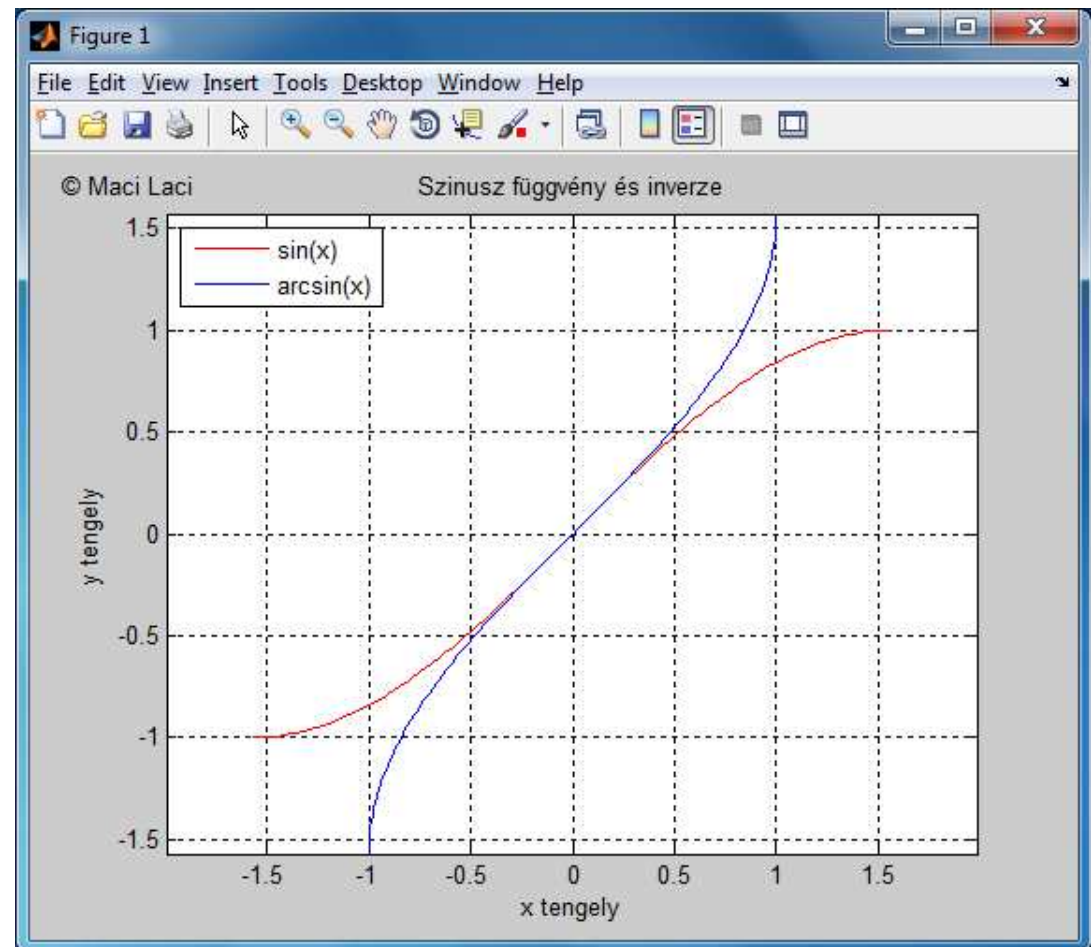


## Függvényábrázolás (plot)

- Összetett plot példa (függvény és inverze)

```
>> x = linspace(-pi/2, pi/2, 101); y = sin(x);  
>> plot(x, y, 'r'); hold on; plot(y, x, 'b'); axis auto  
>> xlabel('x tengely'), ylabel('y tengely')  
>> title('Szinusz  
függvény és inverze')  
>> text(-2.5, 1.7,  
    '@ Maci Laci')  
% szöveg, koordinátával  
>> legend('sin(x)',  
    'arcsin(x)', 2);  
% pozíció megadása, 1-4  
>> grid on,  
axis equal, hold off;
```

- A legend színezése a plotok sorrendjét követi







## Függvényábrázolás (plot)

- Animáció lehetősége: a rajzot mindig újrarajzoljuk és közben szüneteket tartunk
- Egyszerű példa – egységvektor körbeforgatása

```
>> x = [0 1; 0 0];  
% 1 hosszú vektor végpontjai az x tengelyen  
>> fok = 2;  
>> T = [cosd(fok) -sind(fok); sind(fok) cosd(fok)];  
% forgatás mátrixa  
>> for fi = 0:fok:360  
    plot(x(1,:),x(2,:),'.-'), axis([-1 1 -1 1]); % rajz  
    pause(0.1); % várakozás  
    x = T*x;  
    % az x vektor új pozíciója  
end  
% ciklus vége
```
- Most nem kell *hold on* parancs, mert minden lépésben új vásznat rajzolunk, a régit pedig eldobjuk
- Az ábrára cím, feliratok, tengelyfeliratok stb. is tehetők

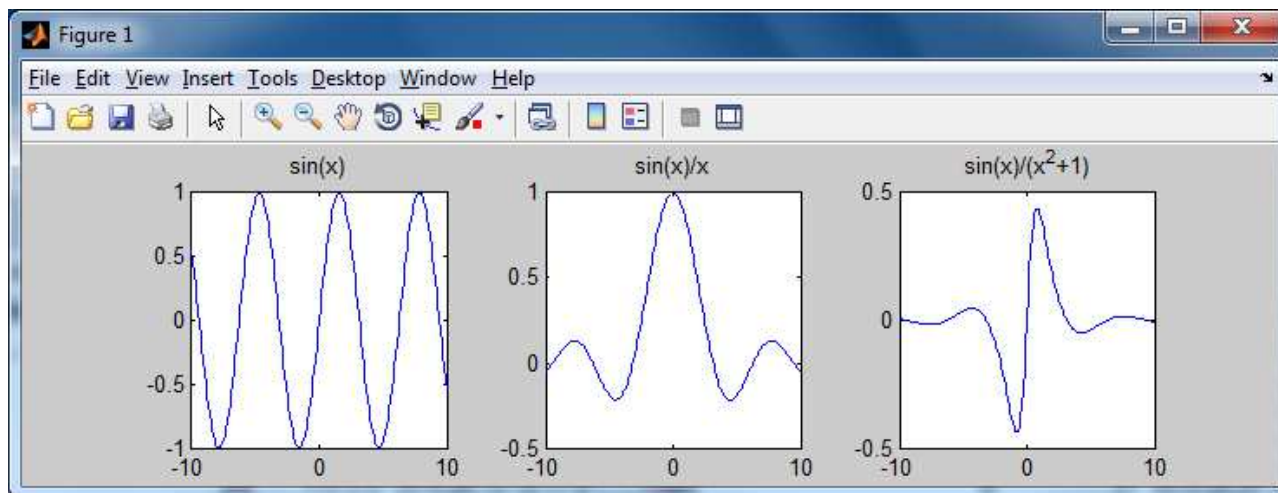


## Függvényábrázolás (subplot)

- A **subplot** utasítással egy képtáblázat megfelelő cellájába illesztjük be a következő plot utasítással elkészülő képet
- A subplot(n, m, p) utasítás az n sorú és m oszlopú képtáblázat p-edik, sorfolytonosan számozott celláját teszi aktívvá. A plot utasítás itt helyezi el a következő képet.

- Most egy 1×3 méretű képtáblázatba három függvénydiagramot helyezünk el:

```
x = -10:0.1:10; figure(1),  
y = sin(x); subplot(1,3,1); plot(x,y); axis square,  
title('sin(x)');  
y = sin(x)./x; subplot(1,3,2); plot(x,y); axis square,  
title('sin(x)/x');  
y = sin(x)./(x.^2+1); subplot(1,3,3); plot(x,y); axis  
square, title('sin(x)/(x^2+1)');
```





## Függvényábrázolás (fplot)

- Az **fplot** utasítás/függvény segítségével ismert/megadott egyváltozós függvények grafikonját rajzoltathatjuk ki
- Eltérés a plot utasítástól: a függvény változó, automatikus osztással kerül kiértékelésre
  - A gyorsabb függvényérték-változásoknál sűrűbben, máshol ritkábban
  - x alappontokat így külön nem kell készítenünk
  - Ezáltal a grafikus kép is lehet jóval kifejezőbb (eml.: „rossz” plot)
- Példa

```
function y = fw(t);  
    y = t*exp(-t^2)*sin(4*t);  
end
```
- Hívás:

```
>> fplot('fw', [0 pi])  
% a függvény nevét aposztrófok határolják!
```
- Tudjuk:
  - fplot használata esetén nem kell pontozott műveleteket használni
  - Anonimus és inline megadásnál is használhatunk fplot rajzolást



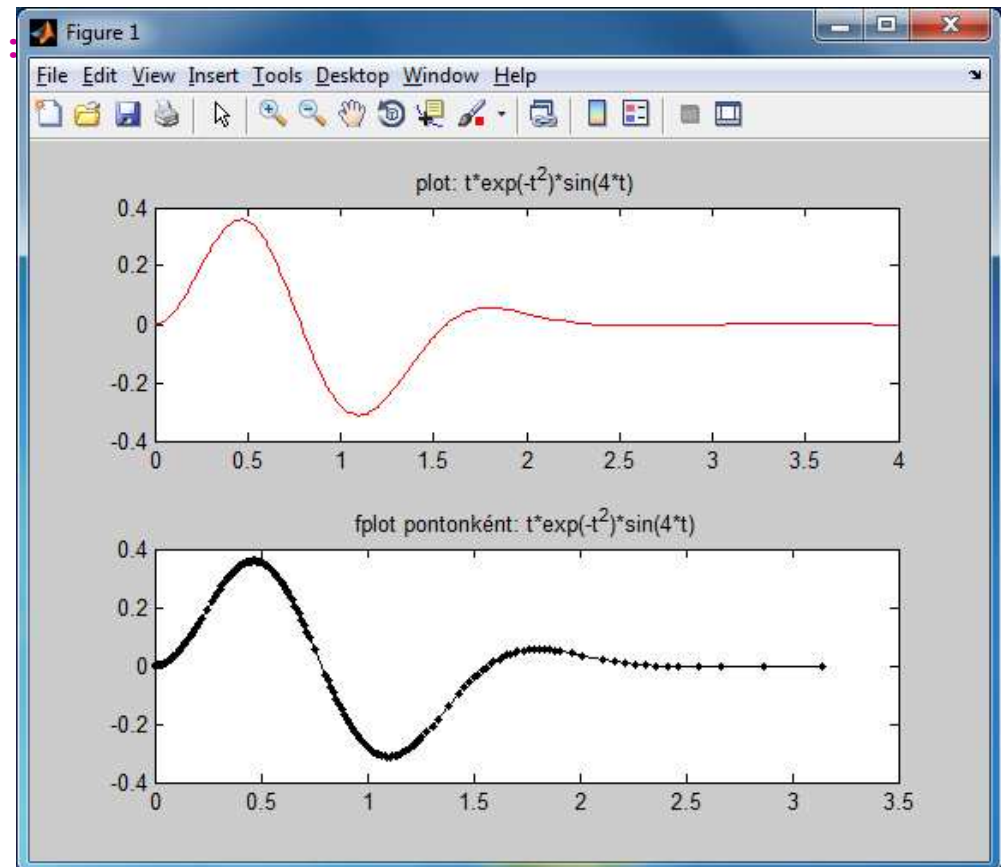
## Függvényábrázolás (fplot)

Érdekesség: az fplot belső működése

- ```
>> subplot(2,1,1); fplot('fw',[0,4], 'r');  
% sima vonalas függvényrajz  
>> title('plot: t*exp(-t^2)*sin(4*t)')  
>> [x,y] = fplot('fw', [0, pi]); subplot(2,1,2);  
plot(x,y,'k.-');  
>> title('fplot pontonként: t*exp(-t^2)*sin(4*t)')
```

- **Magyarázat**

- Ha az fplot hívás [x y] visszaadott értékeit eltároljuk, akkor a grafikon nem jelenik meg
- Ezt felhasználva, az fplothoz szükséges x, y koordinátpárokat feljegyezve, egy plot hívással csak az x, y koord.párokkal megadott pontsorozatot rajzoltatjuk ki
- + összehasonlító ábra: sima fplot



## Függvényábrázolás (ezplot)

- Az **ezplot** („easy-plot”) egyszerű segédeszköz a közvetlen megadású egyváltozós, az implicit megadású és a paraméteres megadású függvények ábrázolásához

- Ha nem adjuk meg az ábrázolási intervallumhatárokat, akkor azokat a Matlab automatikusan generálja
  - Eml.: az fplotnál nem volt elhagyható az ábr. intervallum!
- A függvény automatikusan ábracímet is készít

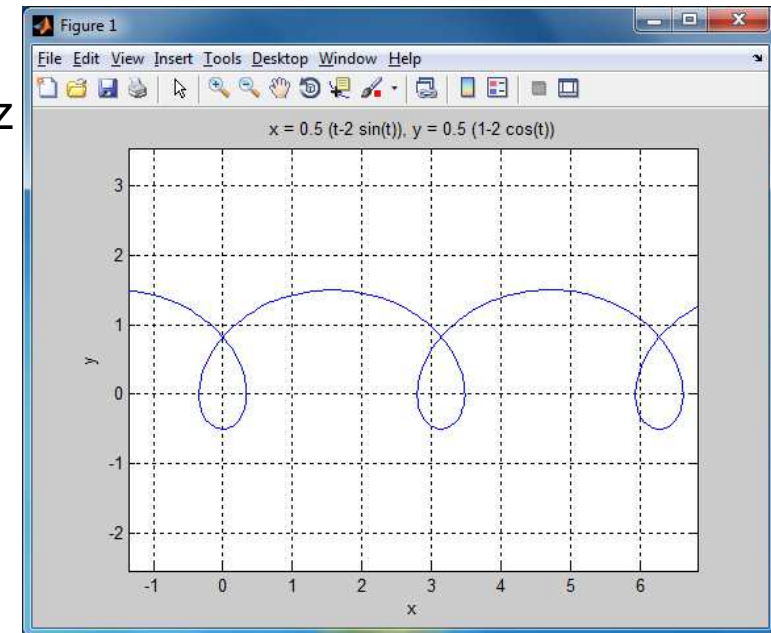
### Példák

- Egyváltozós függvény
 

```
>> ezplot('fw')
>> ezplot('sin(x)/x', [-4*pi 4*pi])
```
- Implicit megadású függvény:
 

```
>> ezplot('(x-1)^2/3^2 + (y-2)^2/2^2 = 1', [-3 5, -1 5]),
grid on
```
- Paraméteresen adott függvény:
 

```
>> ezplot('0.5*(t-2*sin(t))','0.5*(1-2*cos(t))',
[-3 15]), grid on
% ciklois
>> ezplot('t*cos(t)','t*sin(t)',[0 4*pi]), grid on
% archimédeszi spirális
```





## Függvényábrázolás (ezplot)

- Szimbolikusan adott függvényeket az **ez** kezdetű parancsokkal lehet kirajzoltatni (ezplot, ezcontour, ezsurf, ezplot3, stb.)
- A szimbolikus megadás nagy előnye: a függvények formálisan deriválhatók, integrálhatók, stb.
- Most csak az ezplot ilyen típusú használatát nézzük meg

- Példa

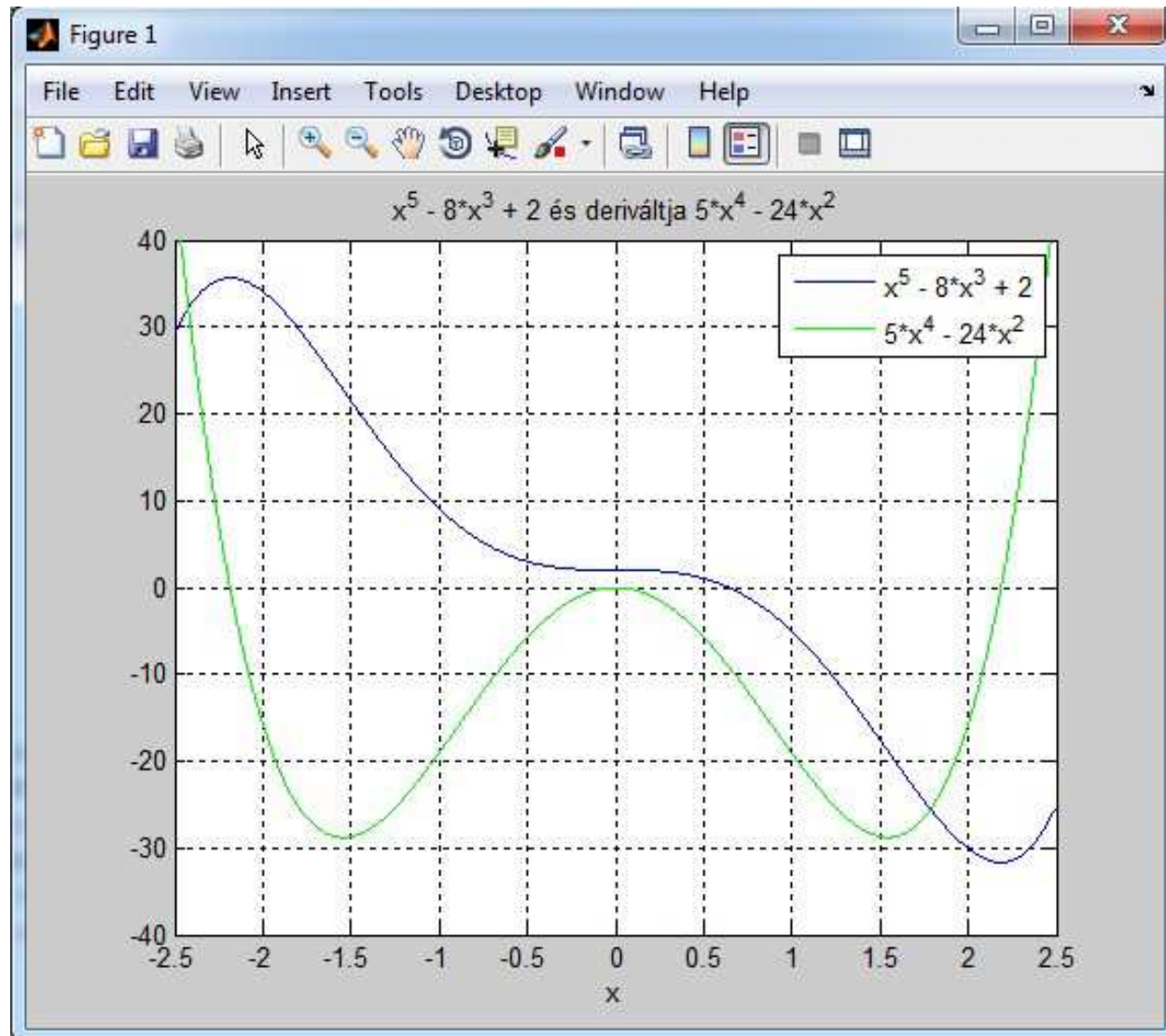
```
syms x; % x szimbolikus változó lesz
fp = x^5-8*x^3+2 % fp függvény (szimbolikusan)
fp_diff = diff(fp, x)
% fp függvény deriváltja (szimbolikusan)
ezplot(fp, [-2.5 2.5]), grid on, hold on % fp rajz
h = ezplot(fp_diff, [-2.5 2.5]);
% fp_diff rajz, objektumkezelő feljegyzés
set(h, 'Color', 'g') % objektum színének változtatása
axis([-2.5 2.5 -40 40]) % intervallumhatárok beállítása
title('x^5 - 8*x^3 + 2 és deriváltja 5*x^4 - 24*x^2')
legend('x^5 - 8*x^3 + 2', '5*x^4 - 24*x^2', 1)
% cím, jelmagyarázat (jobb felső sarok)
```





## Függvényábrázolás (ezplot)

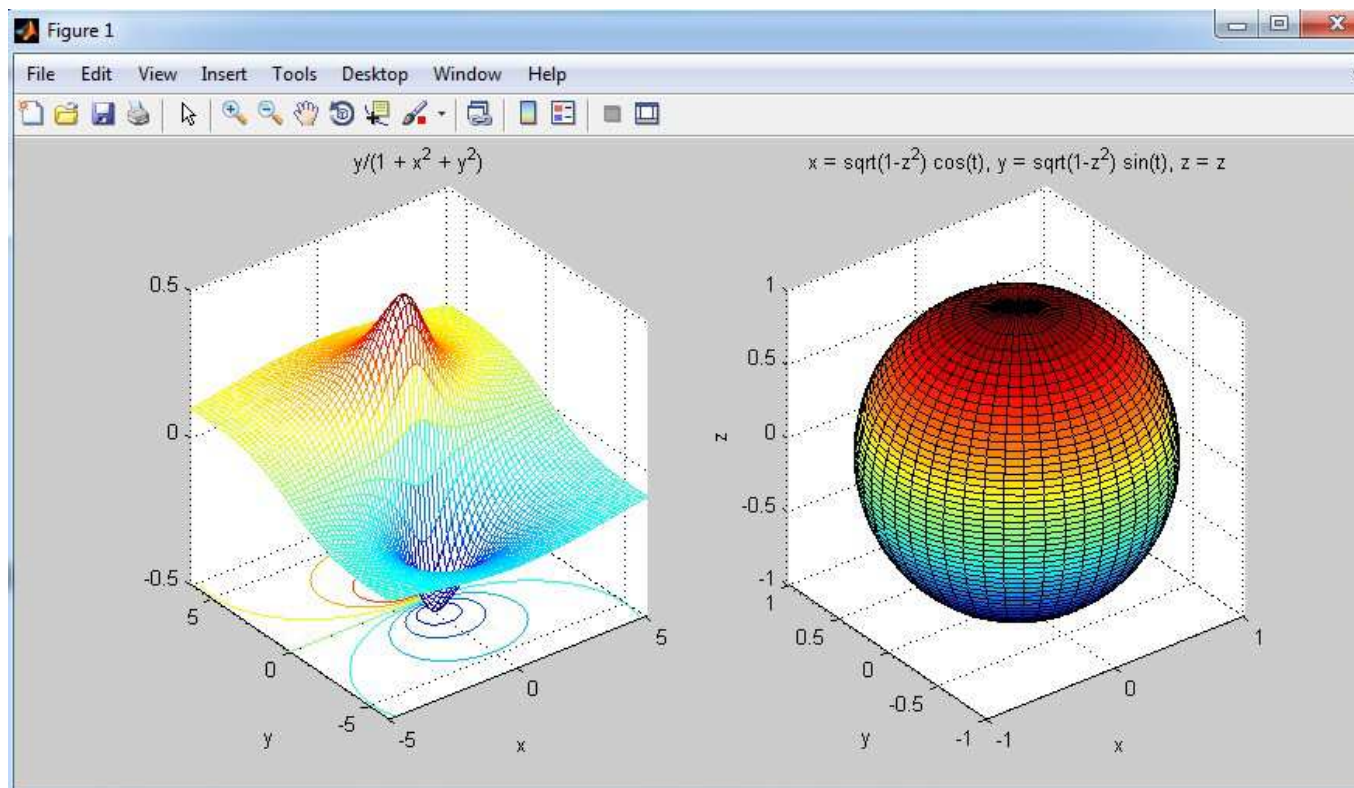
- Az eredmény:



## Függvényábrázolás (többdimenziós)

- Az ezplothoz hasonlóan igen egyszerűen lehet ábrázolni kétváltozós függvényeket és térbeli felületeket a megfelelő **ez** kezdetű függvénnyel
- Ábrázoljuk az  $f(x, y) = y/(1 + x^2 + y^2)$  függvényt a  $-5 < x < 5$ ,  $-2\pi < y < 2\pi$  tartomány felett, majd ábrázoljuk az egységsugarú gömböt is!  

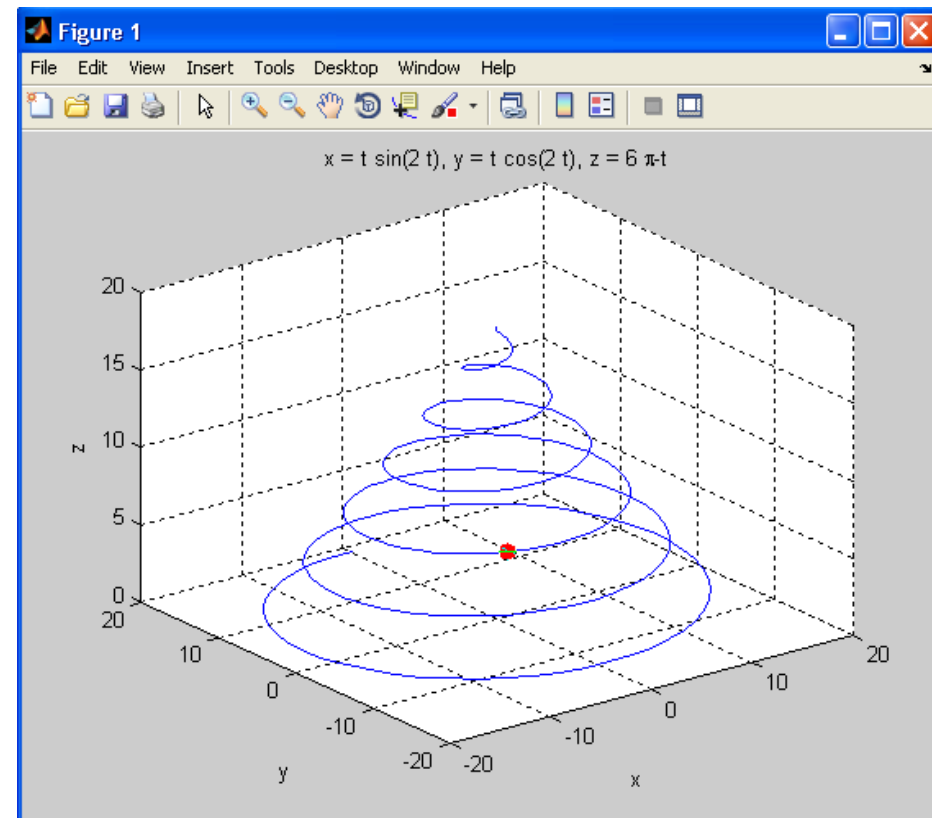
```
>> subplot(1,2,1), ezmeshc('y/(1 + x^2 + y^2)', [-5,5,-2*pi,2*pi])  
>> subplot(1,2,2), s='sqrt(1-z^2)*'; h=ezsurf([s  
'cos(t)'], [s 'sin(t)'], 'z', [0 2*pi, -1 1])
```



## Függvényábrázolás (többdimenziós)

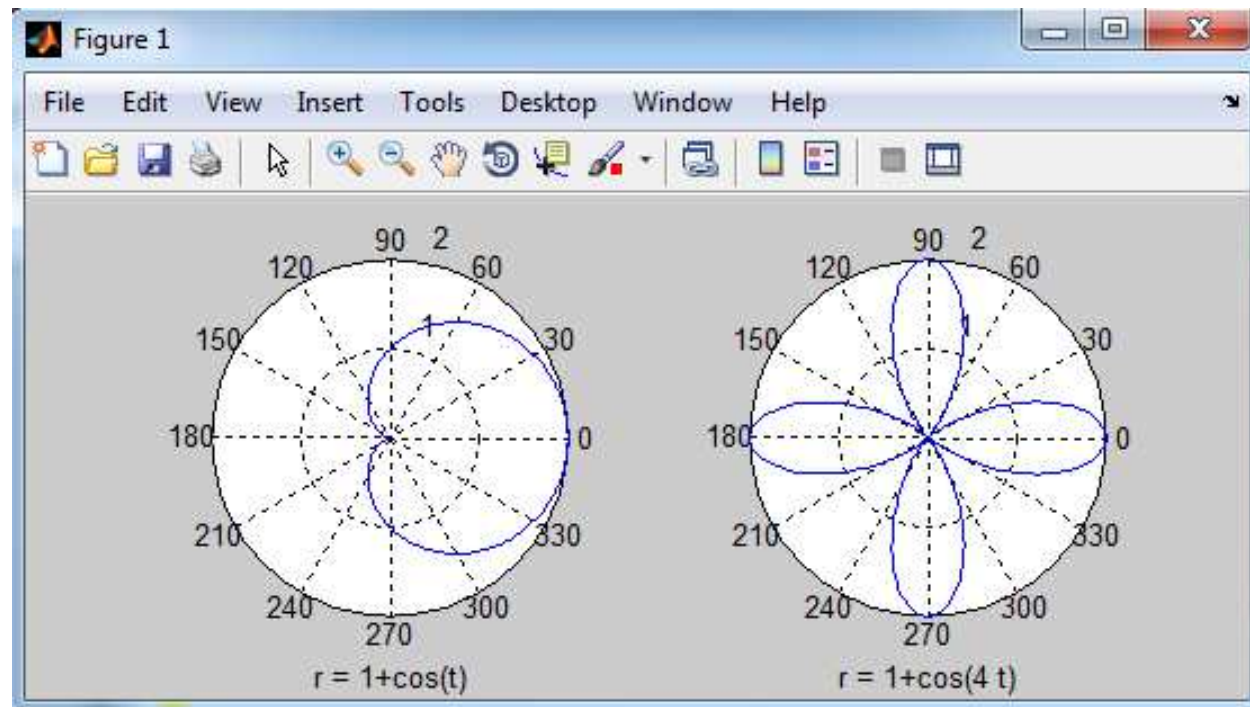
- A paraméteresen adott térgörbék rajzolásához az `ezplot3(funx,funy,funz,[tmin,tmax])` függvény használható
- Ha a paraméterlistát kiegészítjük az `animate` opcióval, akkor a térgörbén végigfutó pontot animálhatjuk is
- Az  $x = t \sin(2t)$ ,  $y = t \cos(2t)$ ,  $z = 6\pi - t$  egyenletű térgörbe egy álló kúp felületén körkörösleg legördülő pont pályáját írja le
- Ábrázoljuk és animáljuk!  

```
>> ezplot3('t*sin(2*t)', 't*cos(2*t)', '6*pi-t', [0,6*pi],  
          'animate')
```
- (További többdim. példák: érdemes a sűgöt megnézni!)



## Függvényábrázolás (polárkoordinátás)

- Az  $r = r(\varphi)$  polárkoordinátás megadás gyakran hasznos lehet
- Ilyenkor a síkgörbe paraméteres alakja:  
 $x(\varphi) = r(\varphi) \cdot \cos(\varphi), \quad y(\varphi) = r(\varphi) \cdot \sin(\varphi)$
- Ezek megjelenítésére az `ezpolar(r(t), [a, b])` parancs a legalkalmasabb
- Jelenítsük meg az  $r = 1 + \cos(t)$  és az  $r = 1 + \cos(4t)$  függvényeket!
  - Ha az intervallumot nem adjuk meg, akkor  $0 \leq t \leq 2\pi$  lesz
- ```
>> subplot(1,2,1), ezpolar('1+cos(t)'), subplot(1,2,2),  
ezpolar('1+cos(4*t)')
```





## Függvényvizsgálat (zérushelyek, metszéspontok)

- Ha két függvény metszéspontját, vagy egy függvény zérushelyeit kell meghatározni, akkor a lépéseink a következők:
  - Ábrázolás megfelelő méretű intervallumon
  - Metszéspont (zérushely) keresés egy alkalmas közeli pontból
  - Ábrázolás
- Határozzuk meg a  $\log_{10}x$  és  $\cos(x)$  függvények metszéspontjait a  $[0, 2\pi]$  intervallumban!
- ```
>> fplot('log10', [0.1 2*pi]), hold on,
fplot('cos', [0.1 2*pi], 'k'),
grid on, title('log10(x) és cos(x) metszéspontjai:')
>> legend('log10(x)', 'cos(x)', 4)
% jobb alsóba jelmagyarázat
>> fd = @(x) log10(x)-cos(x) % különbségfüggvény
>> x = [fzero(fd, 2) fzero(@(x) log10(x)-cos(x), 6)]
% zérushelyek keresése
>> plot(x, fd(x), '.b')
% kék pontok: a különbségfüggvény zérushelyei
>> plot(x, cos(x), 'or') % piros körök: metszéspontok
>> text(x(1)-0.5, -0.1, sprintf('%f', x(1))),
text(x(2)-0.5, -0.1, num2str(x(2)))
>> xd = x; % zérushelyek feljegyzése az integráláshoz
```

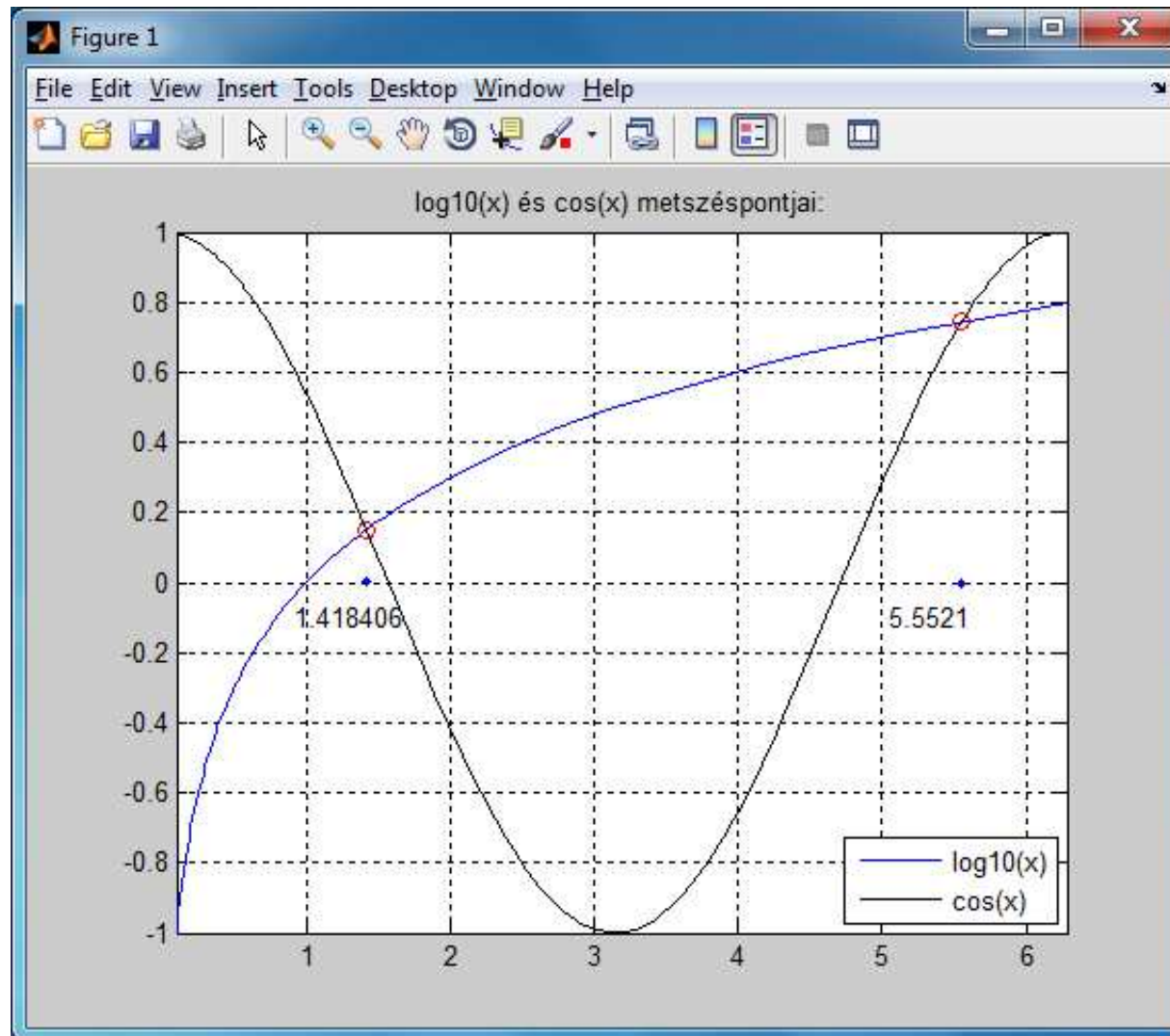






## Függvényvizsgálat (zérushelyek, metszéspontok)

- Az eredmény:







## Függvényvizsgálat (szélsőértékek)

- Egyváltozós függvények minimumhelyét az **fminbnd** függvény segítségével határozhatjuk meg
- Legyen  $f_1(x)$  definíciója M-fájlban:

```
function y = f1(x)
    y = x.^5-8*x.^3+2; % a függvényt már láttuk
% fplot-hoz nem kellene pontozott művelet, de ...
end %f1
```
- Először rajzolunk:

```
>> fplot('f1', [-2.5 2.5]), grid on, hold on
```
- Az **fminbnd** első paramétereként meg kell adni a vizsgált függvény nevét, a második és a harmadik paraméter pedig az intervallum alsó és felső határa
  - A zérushely közelítő helye letapogatható a grafikonról (lásd még: gyak.)
- A negyedik paraméter az opciók beállítására szolgál, és az **fminbnd** valójában három értéket képes visszaadni
- ```
>> [mnx mny kod] = fminbnd('f1', 2, 2.5, optimset('TolX',  
eps)), plot(mnx,mny,'ob')
```

  
% az **optimset** beállítás az **fzero**-nál is használható





## Függvényvizsgálat (szélsőértékek)

- fminbnd parancs (folyt.), a kód lehetséges értékei
  - 1: az fminbnd konvergált az érvényes opciók mellett (minden OK)
  - 0: elértük a lépésszámkorlátozást
  - -1: a függvény miatt terminálódás (pl. negatívból gyökvonás)
  - -2: inkonzisztens határok
- Függvény maximumának meghatározására *nincs külön parancs (!)*, ezt a -1-szeresének minimumaként keressük:

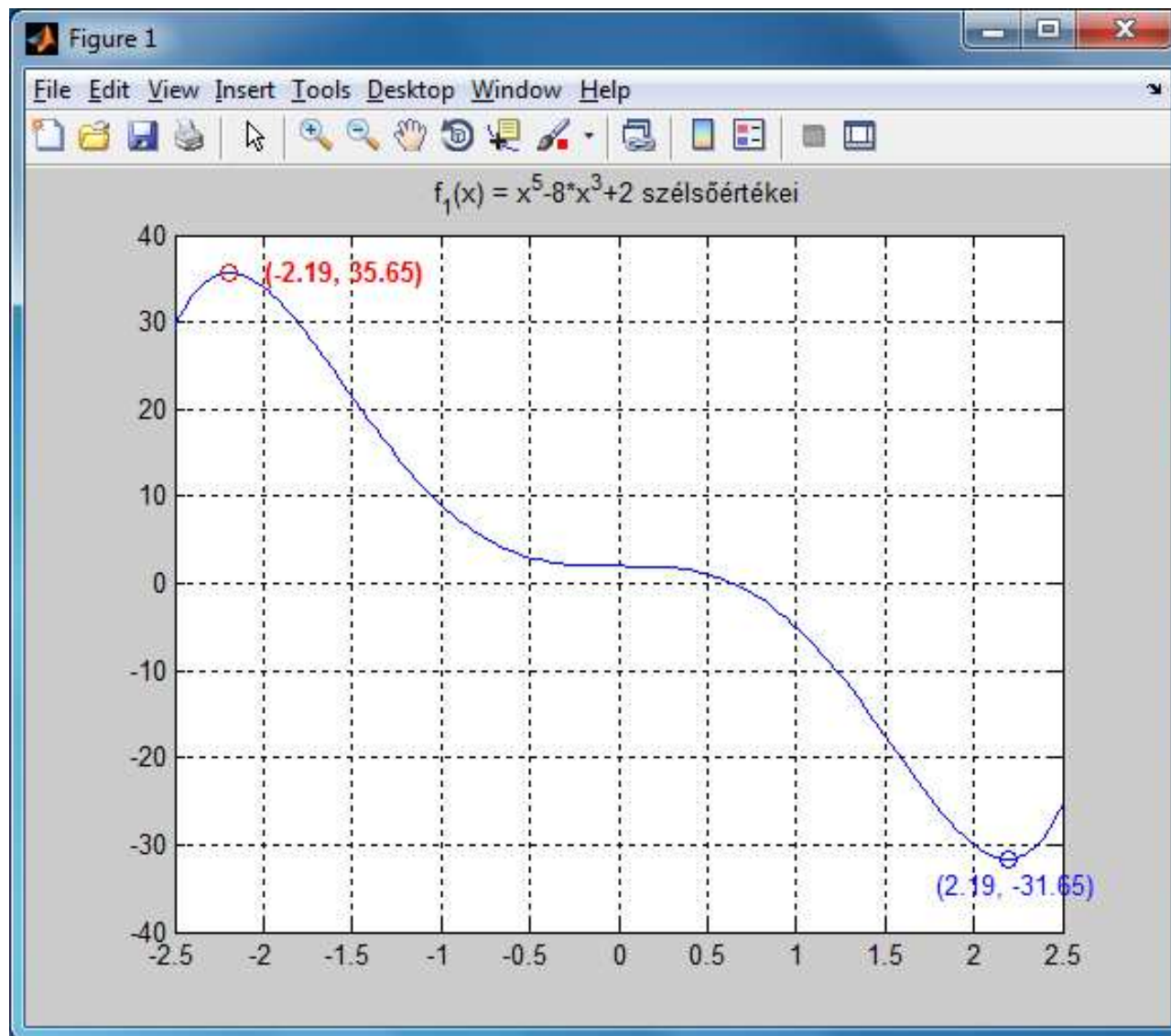
```
>> mxx = fminbnd(@(x) -f1(x), -3, -2), mxy=f1(mxx),  
plot(mxx, mxy, 'or')  
% anonimusan megadás ismert függvényekre hivatkozva!  
% de jó '-f1(x)' is  
>> hx = text(mxx+.2, mxy, sprintf('(% .2f, % .2f)', mxx,  
mxy));  
>> set(hx, 'fontweight', 'b', 'color', 'r')  
% piros és bold lesz a koordinátpár  
>> hn = text(mnx-.4, mny-3, sprintf('(% .2f, % .2f)', mnx,  
mny));  
>> set(hn, 'color', 'b') % kék koordináták  
>> title('f_1(x) = x^5-8*x^3+2 szélsőértékei')
```





## Függvényvizsgálat (szélsőértékek)

- A kész ábra:



# Határozott integrál

- A továbbiakat lásd: Súlyó