

## 5. GYAKORLAT

# SAJÁT FÜGGVÉNYEK, GRAFIKA, FÜGGVÉNYVIZSGÁLAT

## A PLOT UTASÍTÁS

A plot utasítás a legegyszerűbb esetben  $(x, y)$  pontpárok összekötött megjelenítésére szolgál (a pontok koordinátáit vektorok tartalmazzák). A szintaktika: `plot(x, y)`.

**Feladat**

Ábrázoljuk a  $[0, 0]$  és  $[1, 1]$  pontok által meghatározott szakaszt!

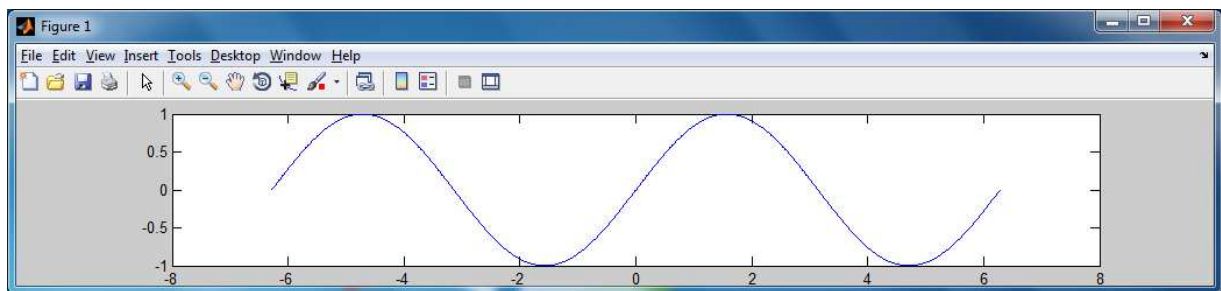
Először az alapértelmezett színt használjuk, utána legyen zöld, majd fekete a vonal.

Függvényábra készítésénél úgy indulunk el, hogy egy vektorba legyártjuk az alappontokat (linspace parancs vagy `:` operátor), majd erre húzzuk rá a függvényt.

**Példa**

Rajzoljuk ki a  $\sin(x)$  függvény grafikonjának pontjait a  $[-2\pi, 2\pi]$  intervallumban 1001 pont segítségével!

```
>> x = linspace(-2*pi, 2*pi, 1001); plot(x,sin(x))
```



Az alappontok megfelelően sűrű előállítása kulcs lépés, anélkül a grafikonunk nem lesz korrekt.  
**F:** Nézzük meg, hogy mi történik, ha az  $x$  sorozat csak 11 elemű!

Több rajz egy ábrán a hold on/off parancsokkal jeleníthető meg. A hold on kiadása után minden ábra egymásra kerül mindaddig, amíg a hold off parancsot ki nem adjuk.

**Feladat**

Ismételjük meg az előző két grafikon kirajzolását, de most már egy közös ábrán! A vonalak színe legyen különböző (pl. piros és kék)! (Próbáljuk ki a vonalstílus megváltoztatását is.)

Tipp: hold on és hold off között gyártsuk le a megfelelő x vektorokat (pl. x1 és x2 néven), és adjuk ki a rajzoló utasításokat.

Ismételjük meg az előző kirajzoltató utasításokat úgy, hogy a vonalvastagságot is változtatjuk, és a vonalszínt az RGB skálán állítjuk be.

```
>> x1 = linspace(...); plot(x1,sin(x1),'LineWidth',1,'Color',[1 0 0]);
```

Ha a plot parancs megadásánál az x és az y sorozatot felcseréljük, akkor így módon az inverz függvényt tudjuk direkt módon kirajzoltatni. A szintaktika ekkor tehát plot(y, x).

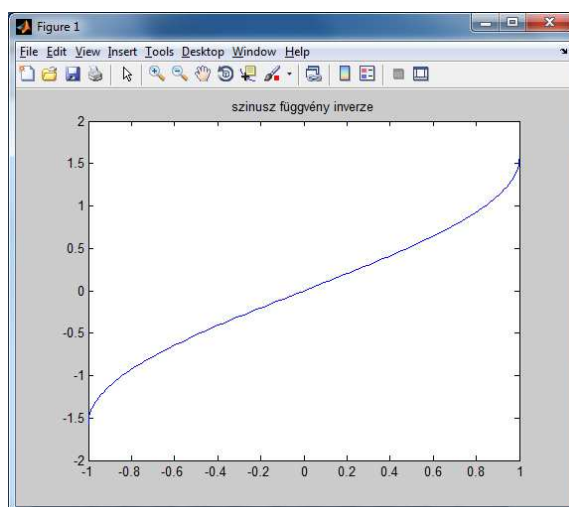
### Feladat

Rajzoltassuk ki a  $\sin(x)$  függvény inverzét a  $[-1, 1]$  intervallumban!

Kétféle módon is oldjuk meg a feladatot: a plot(y, x) szintaktikával, és az asin inverz függvény felhasználásával!

Tegyük az ábrára feliratot!

Ezután oldjuk meg a feladatot úgy is, hogy egy ábrán helyezzük el a szinusz függvényt (megfelelő szakasz) és inverzét! (Készítsünk ehhez scriptet – script M-fájlt.)



### Megoldás (részlet)

```
>> x = linspace(-pi/2, pi/2, 101); y = sin(x); plot(y, x);  
title('szinusz függvény inverze') % plot(y, x) szintaktika
```

vagy

```
>> x = linspace(-1, 1, 101); plot(x, asin(x)) % inverz függvény
```

### Feladat (forgatómátrix)

Készítsük el az origó körüli  $\alpha$  fokos forgatást megvalósító mátrixot, és a felhasználásával forgassunk el egy adott háromszöget! Legyen például  $\alpha = 80^\circ$ , a háromszög pontjai pedig rendre A(1, 1), B(4, 0) és C(3, 4).

Mutassuk be megfelelő ábrán az eredeti és az elforgatott háromszöget! (Rakjunk ki feliratot is.)

Tipp:

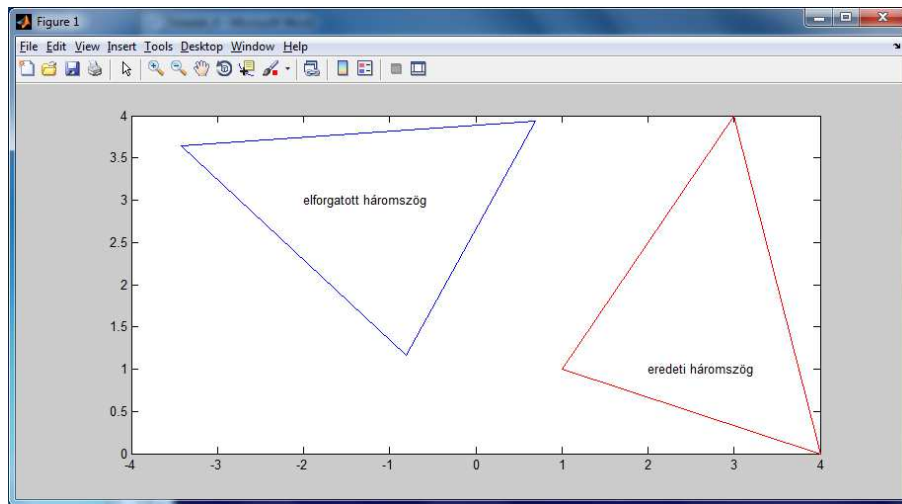
Az  $\alpha$  fokos forgatást megvalósító mátrix alakja:  $A = [\cos(\alpha) \ -\sin(\alpha); \sin(\alpha) \ \cos(\alpha)]$ , ahol az  $\alpha$  szög értéke radiánban adott.

Ennek megfelelően a Matlabos megvalósításban a cosd és a sind függvényeket használjuk.

A háromszög pontjait vegyük fel koordinátáinként egy megfelelő vektorban, majd szorozzuk össze a transzformációs mátrixot a vektorkoordinátákkal.

Az ábrázolásnál figyeljünk arra, hogy az alakzat záródjon, azaz a harmadik pontot is kössük össze az elsővel!

(További segítségként érdemes felidézni az előadáson szereplő dot2dot és forgat eljárást.)



## SAJÁT FÜGGVÉNY LÉTREHOZÁSA, FPLOT

### Feladat

Ábrázoljuk az  $f: x \rightarrow \sin(x)/(x^2 + 1)$  függvényt a  $[-10, 10]$  intervallumon úgy, hogy ehhez saját függvényt (function M-fájlt) hozunk létre!

Tipp: A File/New menüponttal hozzunk létre egy új function m-fájlt, ennek neve legyen f.m. Alakítsuk ki a fejléct és függvény törzsét megfelelő módon (pl.  $y = \dots$ ).

Vigyázzunk arra, hogy egyes helyeken pontozott műveleteket kell használni!

Mentsük el az f.m fájlt a d: meghajtó megfelelő könyvtárába.

A kirajzoláshoz hozzunk létre egy x vektort  $-10$ -tól  $+10$ -ig,  $0,1$ -es lépésközzel. Az ábrázolás ezek után egyszerűen a `plot(x, f(x))` paranccsal történhet.

A saját (és a beépített) függvények ábrázolása az **fplot** utasítással is végrehajtható. Ekkor a Matlab automatikusan generál osztáspontokat, az x vektort tehát nem kell nekünk létrehozni.

Az utasítás szintaktikája:

`fplot('függvényképlet', [intervallum határok], 'megjelenés-vezérlő')`

### Feladat

Ábrázoljuk az `fplot` paranccsal a szinusz függvényt a  $[0, 2\pi]$  intervallumban!

### Feladat

Ábrázoljuk az előző  $f: x \rightarrow \sin(x)/(x^2 + 1)$  függvényt a  $[-10, 10]$  intervallumon az `fplot` utasítással!

Fontos: az `fplot` parancs *elfogadja az olyan függvénydefiníciót is, ahol nem pontozott műveleteket írtunk!* (Próbáljuk ki!)

**F:** Az előző feladatot oldjuk meg közvetlen függvénydefinícióval (anonymus megadás) és inline megadással is!

Próbáljuk ki, hogy az inline és az anonymous megadásnál is elhagyható a pont a megfelelő műveletek előtt, ha az fplot utasítást használjuk.

### Mintafeladat

Próbáljuk ki az  $1,2x^2 \cdot e^{-0,5x}$  függvény különböző megadásait is a következők szerint.

```
>> f = @(x) 1.2*x^2*exp(-.5*x), fplot(f,[0 20])  
% itt csak a függvénynév kell aposztrófok nélkül  
>> fp = inline('1.2*x^2*exp(-.5*x)'); fplot(fp, [0 20])  
% ekkor sem kell fp-t aposztrófok közé tenni!  
>> fplot('f2', [0 20])  
% a függvény máshol definiált (saját függvény), aposztrófok  
közé!
```

### Feladat

Írjuk át az előző feladatban szereplő saját függvényt úgy, hogy az 1,2, a 2 és a  $-0,5$  érték paraméterként legyen megadható! Ábrázoljuk így is a függvényt!

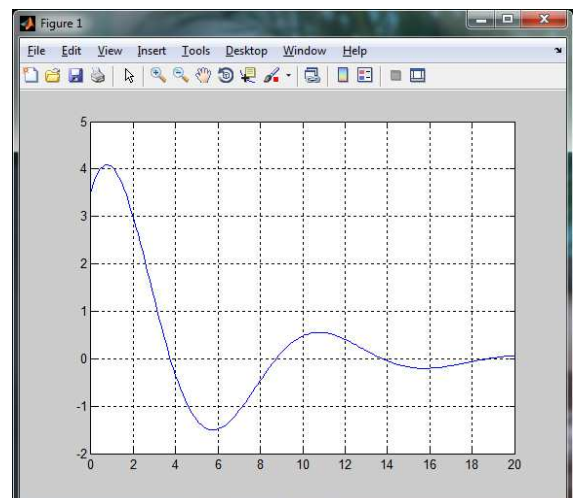
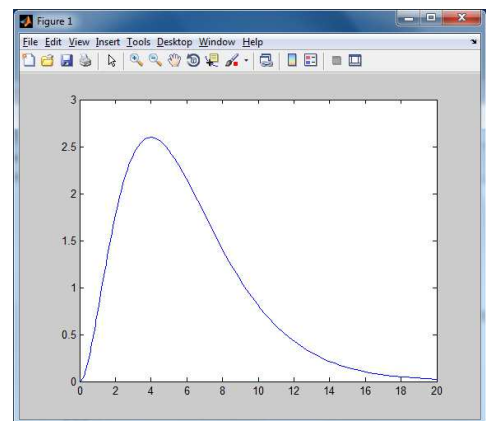
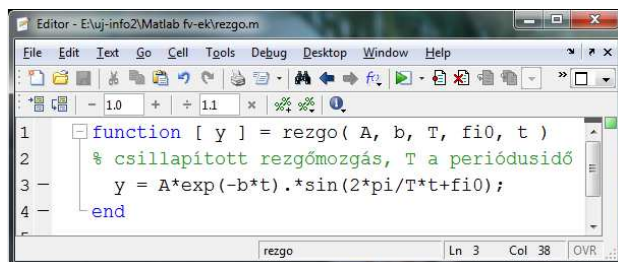
Pl.  $x=0:0.1:20$ ;  $\text{plot}(x, f1(1.2,2,-.5, x))$ .

### Feladat

Írjunk saját függvényt az  
 $y = \text{amplitúdó} \cdot e^{-\text{csillapítás} \cdot t} \cdot \sin(2\pi/T \cdot t + \text{fázisszög})$   
szabállyal adott csillapított rezgőmozgást leíró  
képlet megvalósítására, majd ábrázoljuk a függvényünket!

A paramétereket a következők szerint válasszuk:  
amplitúdó – A (értéke: 5), csillapítás – b (értéke: 0,2),  
t – idő (változó vektor, előállítását lásd lent),  
T – periódusidő (értéke: 10), fázisszög – fi0 (értéke:  $\pi/4$ ).  
A függvény és az m-fájl neve legyen „rezgo”.  
A hívás eszerint a következő:

```
>> A=5; b=0.2; T=10; fi0=pi/4;  
>> t = linspace(0, 2*T, 181);  
>> plot(t,rezgo(A,b,T,fi0,t));  
>> grid on
```



## FÜGGVÉNYVIZSGÁLAT, NEVEZETES PONTOK, HATÁROZOTT INTEGRÁL

### Mintafeladat

Ábrázoljuk az  $f: x \rightarrow 12 \cdot \cos(0,07x) \cdot \sin(1,2x) + 1$  függvényt a  $[20, 27]$  intervallumban, majd határozzuk meg a nevezetes pontjait! (Zérushelyek, minimum- és maximumhelyek.)

Először a d:\munka könyvtárban hozzuk létre a megszokott módon az f3.m szöveges fájlt (saját függvény), majd ábrázoljuk a függvényünket.

```
function y = f3(x)
    y=12*cos(0.07*x).*sin(1.2*x) + 1;
end
```

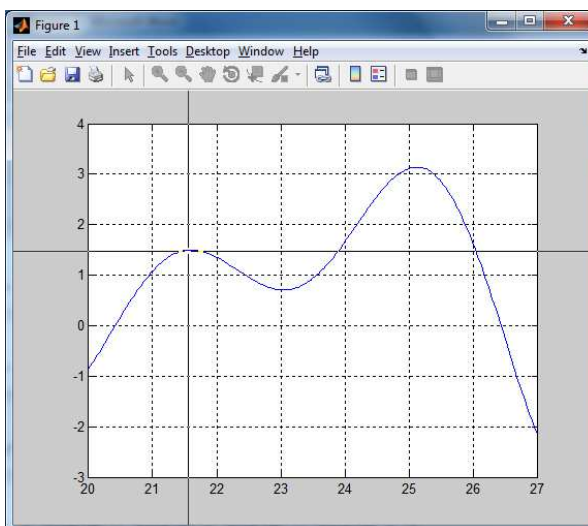
```
>> fplot('f3', [20 27]), grid on
```

A következő lépés az adatok egérrel történő leolvasása, majd a kapott közelítő értékek pontosítása a megfelelő függvénnyel (fzero, fminbnd).

Létező grafikon esetén a

[p q] = ginput

utasítás után az ábránkon egérrel egy pontra kattinthatunk, majd ezt a pozíciót Enterrel nyugtázzhatjuk. A két koordináta a p és q változókhoz rendelődik. Ha a ginput(n) alakot használjuk, akkor a p és q vektorok n eleműek lesznek, azaz folyamatosan n darab pont érzékelését végezhetjük el.



Az előbbi ábráról olvassassuk be a következő pontok közelítő értékeit ilyen sorrendben:

- a két zérushely
- a két lokális maximumpont
- az egy lokális minimumpont!

Ellenőrizzük, hogy a p és q vektorok (nagyjából) megfelelő értékeket kaptak-e!

A letapogatott pontok koordinátáit a következők szerint pontosítjuk.

A zérushelyek keresését az fzero('függvény', hol) paranccsal végezhetjük el.

```
>> zhx = [fzero('f3', p(1)) fzero('f3', p(2))], zhy = [0 0]
```

A minimumhely kereséséhez az fminbnd('függvény', alsó határ, felső h.) parancsot használjuk.

```
>> minx = fminbnd('f3', p(5)-0.5, p(5)+0.5), miny = f3(minx)
```

Maximumhely kereséshez az fminbnd keresést a -f(x) függvényre alkalmazzuk.

```
>> maxx(1)=fminbnd('-f3(x)',p(3)-0.5,p(3)+0.5), maxy(1)=f3(maxx(1))
>> maxx(2)=fminbnd('-f3(x)',p(4)-0.5,p(4)+0.5), maxy(2)=f3(maxx(2))
```

A pontokat feltesszük fekete, piros és zöld körökkel a grafikonra.

```
>> hold on
>> plot(zhx,zhy,'ko'), plot(maxx, maxy,'ro'), plot(minx, miny,'go')
```

Tegyük fel az ábrára jelmagyarázatot! (A megadási sorrend az ábraelemek sorrendje.)

Adjunk az ábrának címet is!

```
>> title('f(x) = 12*cos(0.07*x)*sin(1.2*x) + 1')
```

A határozott integrál (függvény alatti terület) kiszámítására – numerikusan – a `quad('függvény', alsó határ, felső határ, pontosság)` parancs szolgál.

```
>> integral = quad('f3', zhx(1), zhx(2), eps)
```

Ezt is kiírjuk az ábrára a `text(x, y, 'szöveg')` parancs segítségével.

```
>> text(20.5, -2.5, ['Zérushelyek közötti határozott integrál: '
num2str(integral)])
```

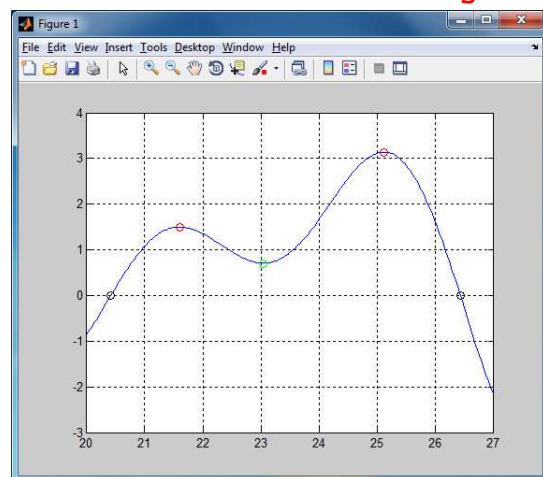
% Itt a karakterláncot egy  
sorvektorban adtuk meg  
szeletenként.

Végül mentjük el az ábránkat `abra1.jpg` néven.

```
>> print -djpeg90 -r300 abra1
% az aktuális könyvtárba
```

A `print` utasítás paraméterei (lásd `help print`):

- `-d` után a típus jön (ps, psc, eps, epsc, jpeg<nn>, tiff, png, ...)
- `-r` után a dot/inch



## IMPLICIT MEGADÁSÚ FÜGGVÉNYEK

### Mintafeladat

Rajzoljuk ki az  $x^2 + y^2 = 1$  síkegyenlettel adott kört az `ezplot` utasítással úgy, hogy a tengelyeket  $-1,1$ -től  $+1,1$ -ig skálázzuk!

```
>> f = inline('x^2+y^2=1'), argnames(f)
>> k=1.1; ezplot(f, [-k k -k k]), axis square
% inline megadás
>> k=1.1; ezplot('x^2+y^2=1', [-k k -k k]), axis square
% sztringes megadás
```

**Mintapélda**

Rajzoljuk ki az  $x^3 + y^3 = 3a \cdot x \cdot y$  implicit egyenlettel adott Descartes-levelet!

```
>> a = 2; % ekkor 3*a értéke 6 lesz  
>> f = inline('x^3+y^3=6*x*y'), argnames(f), ezplot(f, [-2*a 3*a  
-2*a 3*a]), axis square
```

**OTTHONI MUNKA****Feladat**

Próbáljuk ki, hogy a plot parancsnál megismert megjelenést módosító vezérlők (szín, vonalstílus) az fplotnál is működnek.

**Feladat**

A hold utasítás nélkül is lehet több grafikont egy ábrára tenni. A szintaktika: plot(x1, y1, string1, x2, y2, string2, ...), ahol a string1 pl. 'r' lehet. Próbáljuk ki!

**Feladat**

A csillapított rezgőmozgás függvényére is végezzük el a nevezetes pontok meghatározását, és rakjuk fel a pontokat az ábrára!

**Feladat**

Keressünk a sűgóban/interneten példákat a kétváltozós függvények ábrázolására (többdimenziós grafika)! Próbáljuk ki őket, és módosítsunk egyes paramétereket!  
(Pl.  $fk = @(x,y) x.^2 + y.^2$ , ezcontourf(fk), axis square)

