

## 4. GYAKORLAT

# MÁTRIXFÜGGVÉNYEK, SAJÁT FÜGGVÉNYEK, GRAFIKA

## SÁVMÁTRIXOK, ALSÓ- ÉS FELSŐHÁROMSZÖG MÁTRIXOK

A *diag* parancs felhasználásával kiemelhetjük egy mátrix főátlóját vagy valamelyik mellékátlóját, ill. vektor felhasználásával diagonális mátrixot készíthetünk (az adott vektor a főátlóba vagy valamelyik mellékátlóba elhelyezhető).

**Feladat**

Legyen  $D = [1:4; 11:14; 21:24; 31:34]$ .

Vegyük ki  $D$  főátlóját, majd készítsük el azt a mátrixot, amely csak  $D$  főátlójának elemeit tartalmazza, és a többi eleme 0.

Mit eredményez a  $\text{diag}(D, -1)$  és a  $\text{diag}(\text{diag}(D, -1), -1)$  kifejezés?

Az előző mintára állítsuk elő azt a kifejezést, amelyik olyan mátrixot generál, amelyben csak a  $D$  mátrix felső mellékátlójának elemei nem 0 értékűek!

**Feladat**

Ellenőrizzük, hogy a  $\text{diag}(\text{diag}(D, -1), -1) + \text{diag}(\text{diag}(D, 1), 1) + \text{diag}(\text{diag}(D))$  kifejezés a  $D$  mátrix olyan részét emeli ki (sáv), amelyben a főátló és a két első mellékátló elemei szerepelnek. Mit állít elő a  $\text{diag}(\text{diag}(D, -1), 1) + \text{diag}(\text{diag}(D, 1), -1) + \text{diag}(\text{diag}(D))$  kifejezés?

A *triu* és a *tril* parancs felső- ill. alsóháromszög mátrixot készít az alapmátrixból.

**Feladat**

Állítsuk elő  $D$  alsó- és felsőháromszög mátrixát!

Mit eredményez a  $\text{triu}(D, 1)$  és a  $\text{tril}(D, -1)$  kifejezés?

Ez alapján állítsuk elő  $D$ -t oly módon, hogy összerakjuk (most: mátrixösszeadással!) a főátlójából, az alsó- és a felsőháromszög mátrixából!

## A FIND PARANCS

A *find* parancs adott értékű elemek (és indexeik) megkeresésére szolgál. Alapértelmezés szerint a nem nulla elemeket kereshetjük, de akár összetett logikai keresési feltételek is megadhatók.

**Feladat**

Keressük meg a  $3 \times 3$ -as mágikus mátrix 5-nél kisebb elemeit, és azok indexeit!

```
>> k = find(M<5) % mo. részlet, a jó indexek
```

**Feladat**

Adott egy  $x$  vektor és két határ,  $x = [1.9 \ 3.8 \ 3.9 \ 4.1 \ 5 \ 10.2]$ ,  $also = 3,7$  és  $felso = 4,2$ .

Logikai indexeléssel és a *find* parancs segítségével határozzuk meg a határok közé eső vektorértékeket és indexeiket.

A megoldás javasolt lépései:

- „És” kapcsolatos feltétellel állítsunk elő egy logikai igaz-hamis vektort, amely mutatja, hogy egy elem jó-e (logikai indexelés);
- A *find* paranccsal határozzuk meg a megfelelő indexeket;
- Írassuk ki a megfelelő elemeket.

```
>> indexek1 = (x > also) & (x < felso) % mo. részlet
```

## MATEMATIKAI ÉS SZÁMELMÉLETI FÜGGVÉNYEK

A trigonometrikus, exponenciális, logaritmikus függvények előállítására és megjelenítésére szolgáló parancsokat más helyen (már korábban és még majd később is) gyakoroljuk.

**Feladatok**

Határozzuk a Matlabbal, hogy hány prímszám van 1000-ig (*primes* függvény)! Írassuk is ki őket!

Bontsuk fel prímtényezősszorzatra a következő számokat: 1001, 10001, 100001.

Teszteljünk különböző véletlenül választott egészeket (ne legyenek párosak és hárommal, öttel oszthatók), hogy prímek-e (pl. 8-10 jegyű számok)! Készítsünk statisztikát, hogy a számok hány %-a lesz prím!

## 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)*. A parancs *harmadik paraméterével* a kirajzolás/megjelenítés módja (vonalszín, vonalstílus, jelölők) szabályozható (lásd előadás és *súgó*).

**Feladat**

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

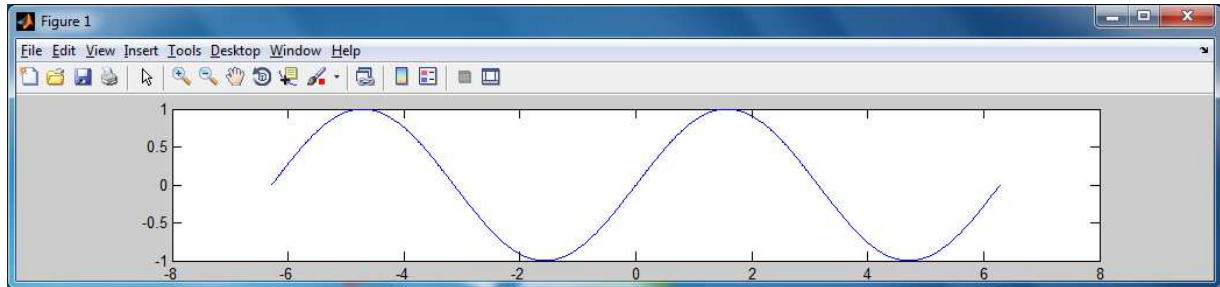
Először az alapértelmezett színt használjuk, utána legyen zöld, majd fekete a vonal.

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 szinusz 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 előállításuk kulcskérdés, 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* után minden rajz egymásra kerül addig, amíg a *hold off* parancsot ki nem adjuk (ill. be nem csukjuk az ablakot).

### 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 gyártunk le a megfelelő x vektorokat (pl. *x1* és *x2* néven), és adjuk ki a rajzoló utasításokat.

Ismételjük meg az előző kirajzoló 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.

### Feladat

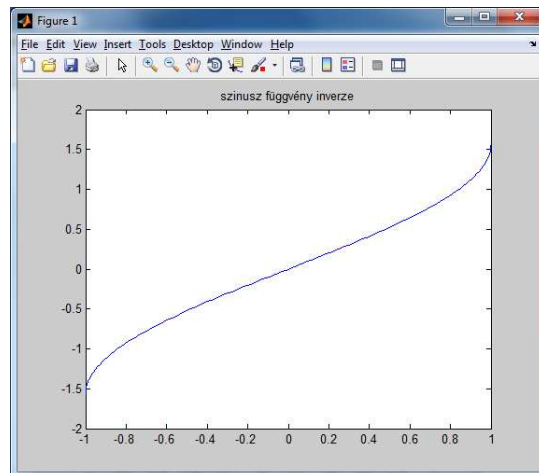
Rajzoltassuk ki a szinusz függvény inverzét a  $[-1, 1]$  intervallumban!

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!

(Hf.: Készítsünk a feladat megoldására „szép” scriptet, külön m-fájlba. A scriptet kommentáljuk, és igény szerint bővítsük kiírással.)



**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 alfa 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).

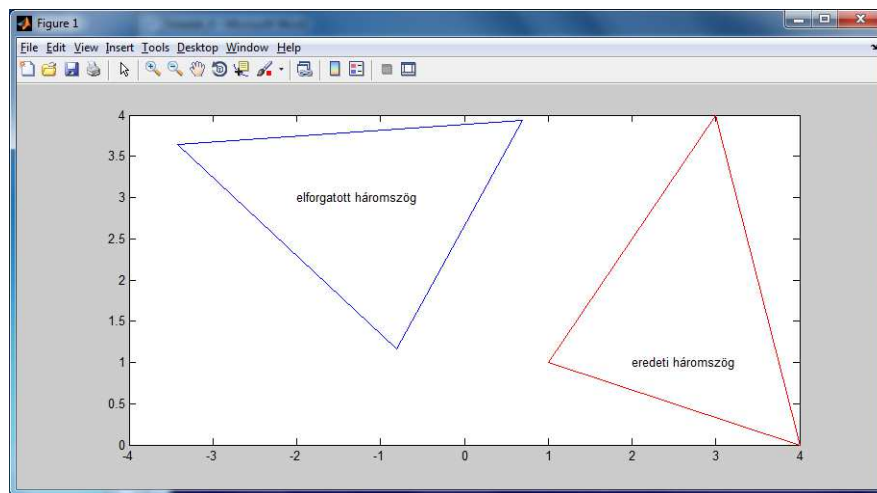
Mutassuk be megfelelő ábrán az eredeti és az elforgatott háromszöget! (Rakjunk ki feliratot is.)  
Tipp:

Az alfa fokos forgatást megvalósító mátrix alakja:  $A = [\cos(\alpha) \ -\sin(\alpha); \sin(\alpha) \ \cos(\alpha)]$ , ahol az alfa szög értéke radiánban adott. A Matlabos megvalósításban használhatjuk a *cosd* és a *sind* függvényeket is (fok).

A háromszög pontjait vegyük fel koordinátaként 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!

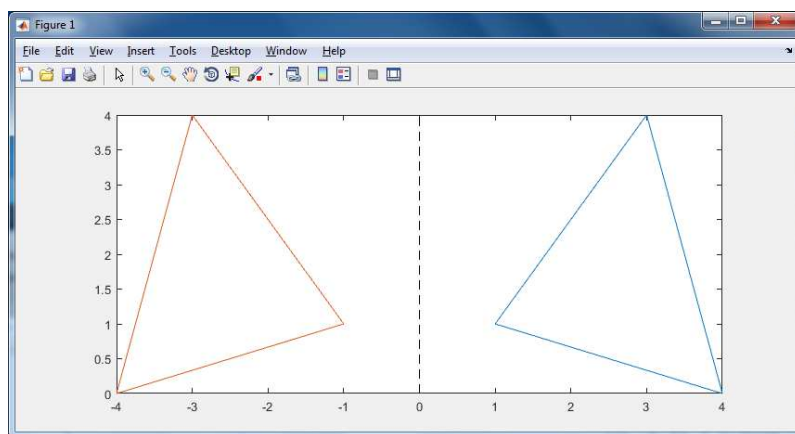
(További segítségként fel lehet idézni az előadáson szereplő *dot2dot* és *forгат* eljárást.)

**Feladat** (transzformációs mátrix, önálló gyakorlásra)

Készítsünk transzformációs mátrixot az *x*-tengelyre, ill. az *y*-tengelyre történő tükrözés megvalósításához (lásd „Lineáris algebrai alkalmazások” előadás)!

Az előző feladatban szereplő háromszögre hajtjuk végre mindkét transzformációt! Ábrázoljuk az eredeti és a tükrözött háromszögeket egy ábrán! Jelöljük be szaggatott vonallal a tengelyeket is! Használjunk különböző színeket. Ízlés szerint rakjunk az ábrára feliratokat. Végül mentjük ki a parancsokat egy script fájlba.

A megoldás részlete:

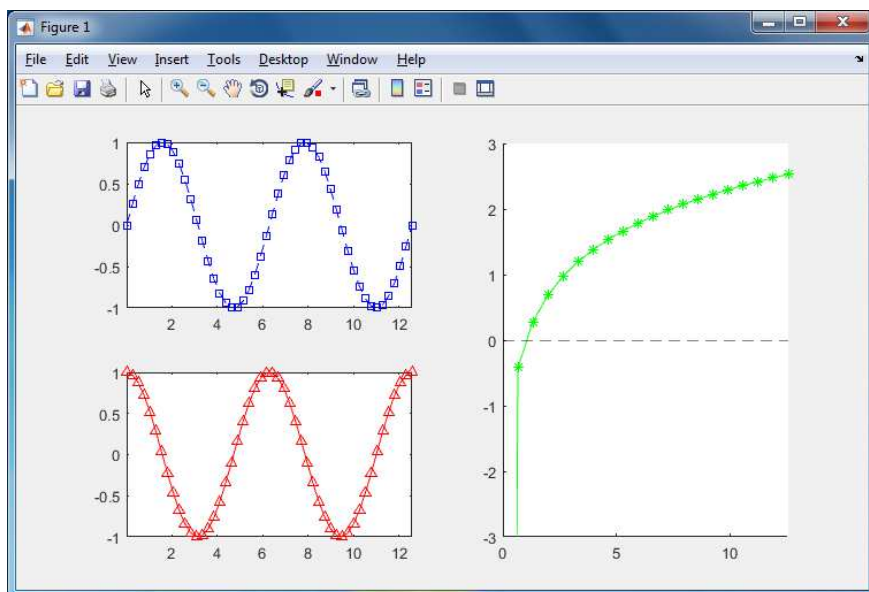


## SUBPLOT

### Feladat

Készítsünk olyan ábrát, amely a szinusz, koszinusz és a természetes alapú logaritmus függvények grafikonjait tartalmazza (a megjelenítés eps-tól induljon, a szinusz függvény két teljes periódusáig). A rajzszínek különbözőek legyenek (pl. piros, kék és zöld), és használjunk többféle jelölőt, ill. többféle vonalstílust! A rajzok rendre a bal felső és bal alsó sarokban, ill. a jobb oldalon jelenjenek meg. A logaritmus függvény  $x$ -tengelyéhez húzzunk szaggatott fekete vonalat, és itt az  $y$  skálázást külön kezeljük le! A parancsokat végül írjuk bele egy scriptbe. (Segítségként érdemes megnézni az előadás fóliákon szereplő példát.)

Egy lehetséges megoldás:



## SAJÁT FÜGGVÉNY LÉTREHOZÁSA, FPLOTT

### 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 (Home szalag) 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 (D: meghajtó).

Próbáljuk ki a függvényünket úgy, hogy kiíratunk néhány függvényértéket (pl.  $f(1)$ ,  $f(2)$  stb.).

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

### Feladatok

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

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

Fontos: az *fplot* parancs *elfogadja az olyan függvénydefiníciót is, ahol nem írunk pontozott műveleteket!* Próbáljuk ezt ki! (Az újabb Matlab verziók azonban már figyelmeztetést küldenek, hogy ez nem javasolt!)

### Feladat

Az előző feladatot oldjuk meg közvetlen függvénydefinícióval (*anonymus* megadás, `@`) is! (Anonymus megadásnál is elhagyható a pont a megf. műveletek előtt, ha *fplot*-ot használunk.)

### Mintafeladat

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

```
>> f = @(x) 1.2*x^2*exp(-.5*x), fplot(f,[0 20])  
% itt csak a függvénynév kell aposztrófok nélkül  
>> fplot('f2', [0 20])  
% a függvény máshol definiált (saját függvény), aposztrófok közé tesszük!
```

## OPTIMALIZÁCIÓS FELADATOK

A *max*, ill. *min* függvények használatával (+ függvényrajz, ha szükséges) a Matlabbal gyorsan és hatékonyan megoldhatunk sokféle olyan típusú optimalizációs feladatot, amikor valamilyen kifejezés (most: egyváltozós) szélsőértékét kell megtalálni.

**Feladat**

Egy háromszög  $a$  és  $b$  oldalai 10 és 14 egység hosszúak. A  $\gamma$  közbezárt szög mely értékénél lesz a terület maximális?

Tipp: A háromszög területe  $T = a \cdot b \cdot \sin(\gamma)/2$ . Hozzunk létre egy  $g$  sorozatot, amely 0-tól 180-ig lépdél (pl. 100 elemmel), majd ezt felhasználva egy  $t$  sorozatot, amely a területet számolja. Határozzuk meg a maximális terület értékét és a hozzá tartozó indexet ( $\max$  függvény). Végül határozzuk meg a  $g$  sorozat adott indexű elemét is. Ellenőrizzük rajzzal és értékkiírással, hogy valóban ez a legnagyobb terület!

**Megoldás (részlet)**

```
>> t=10*14*sind(gam)/2;  
>> [max_ertek max_index] = max(t)
```

**OTTHONI MUNKA****Feladat** (üres mátrix/vektor)

Érdekes, hogy a Matlabban többféle méretű üres vektor/mátrix is létrehozható.

Milyen méretű lesz az  $A = [\text{eye}(3); 6:3]$  mátrix?

Próbáljuk ki a következő előállításokat, és hasonlítsuk össze a létrejövő objektumok méreteit!

```
>> c = [10:0], isempty(c), c_meret = size(c), c_hossz = length(c)  
>> d = [1:0]', isempty(d), d_meret = size(d), d_hossz = length(d)  
>> e = [], isempty(e), e_meret = size(e), e_hossz = length(e)  
% programozásnál az utóbbi fontos!
```

**Feladat** (sávmátrix)

Állítsunk elő a *diag* és a *ones* függvények használatával egy olyan  $6 \times 6$ -os mátrixot, amelynek főátlója csupa 8-as, felső mellékátlójában minden elem 3-as és az alsó mellékátlóban 1-esek vannak! (Vigyázat: az első mellékátlókban csak 5-5 elem van!)

**Feladat** (sávmátrix)

Tekintsük a jegyzetben szereplő, ill. a gyakorlat könyvtárában adott *tridiag.m* fájlt, amely egy speciális sávmátrixot állít elő (főátló + első mellékátlók). Az *m*-fájl segítségével adjuk meg egyetlen Matlab-utasítással azt a  $20 \times 20$ -as mátrixot, amelynek

- bal felső 10-szer 10-es blokkja olyan tridiagonális mátrix, amelynek főátlójában  $-2$ -esek, két mellékátlójában pedig 1-esek állnak (*tridiag*[1,  $-2$ , 1] típusú);
- jobb alsó 10-szer 10-es blokkja *tridiag*[2,  $-4$ , 2] típusú;
- a többi eleme pedig 0.

**Feladat** (plot)

A *hold* utasítás nélkül is lehet több grafikont egy ábrára tenni. A szintaktika: *plot*( $x_1, y_1, \text{string}_1, x_2, y_2, \text{string}_2, \dots$ ), ahol a *string* pl. 'r' lehet. Próbáljuk ki!



**Feladat (plot, fplot)**

Próbáljuk ki, hogy a *plot* parancsnál megismert megjelenést módosító vezérlők (szín, vonalstílus) az *fplot*-nál is működnek.

**Feladat (egyszerű saját függvény, több visszatérési értékkel)**

Készítsünk saját függvényt, amely paraméterként megkapja a kör sugarát, és kiírja a kerület és a terület értékét.

Módosítsuk a függvényünket úgy, hogy paraméterként egy  $\phi$  szög is megadható legyen, és az adott szögű körcikkkel dolgozzon.

**Feladat (saját függvény és rajz)**

Írjuk át az órai anyagban szereplő saját függvényt (*fplot*) úgy, hogy az 1,2, a 2 és a -0,5 érték paraméterként legyen megadható! Ábrázoljuk így is a függvényt!

Pl.  $x=0:0.1:20$ ; `plot(x, f1(1.2,2,-.5, x))`.

**Feladat (inline megadás és rajz)**

(Az *inline* megadás az újabb Matlaboktól nem támogatott, ill. nem javasolt, ezért a feladatot csak a 2015 előtti verziókban gyakoroljuk.)

Oldjuk meg az órai  $f: x \rightarrow \sin(x)/(x^2 + 1)$  függvény kirajzoltatását *inline* megadással is!

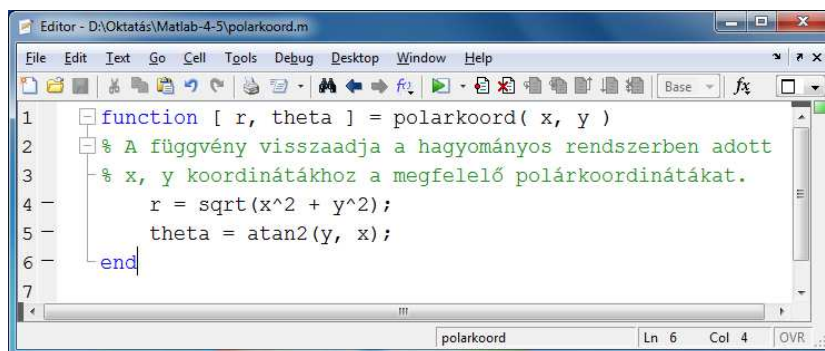
Próbáljuk ki, hogy az *inline* megadásnál is elhagyható a pont a megfelelő műveletek előtt, ha az *fplot* utasítást használjuk.

Próbáljuk ki az  $1,2x^2 \cdot e^{-0,5x}$  függvény *inline* megadását is (ábrázolással) az alábbi szerint.

```
>> fp = inline('1.2*x^2*exp(-.5*x)'); fplot(fp, [0 20])  
% ekkor sem kell fp-t aposztrófok közé tenni!
```

**Feladat (saját függvény)**

Tekintsük a következő példát.



Hívás:

```
>> [r, theta] = polarkoord(3, 4)
```

Ezen függvény mintájára készítsük el azt a hasonló függvényt, amelyik polárkoordinátákból készíti el az  $x$  és  $y$  koordinátákat.



**Feladat (saját függvény)**

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!

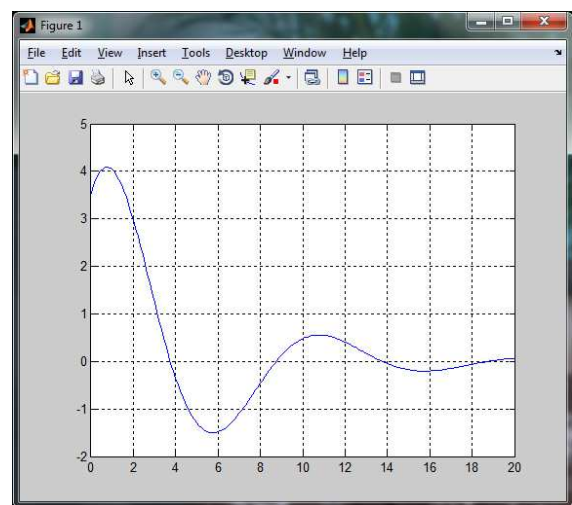
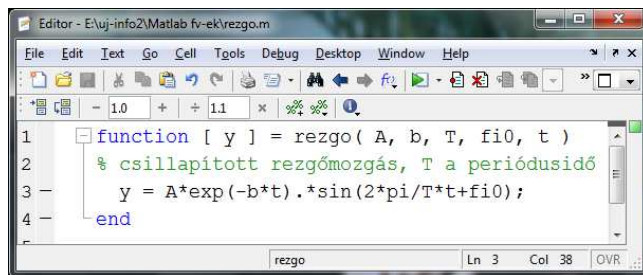
A paramétereket a következők szerint válasszuk:

amplitúdó – A (értéke: 5), csillapítás – b (értéke: 0,2),  $t$  – idő (változó vektor, előállítását lásd lent),  $T$  – periódusidő (értéke: 10), fázisszög –  $\text{fi0}$  (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
```

**Feladat (saját függvény, forgatómátrix)**

A forgatómátrixos feladatot oldjuk meg úgy is, hogy saját függvényt készítünk a forgatás végrehajtására. Paraméterként legyen megadható a forgatandó pont  $x$  és  $y$  koordinátája, eredményként kapjuk meg az új koordinátákat! Az első sikeres megoldás után módosítsuk az eljárástunkat még azzal is, hogy az alfa érték is paraméterként megadható!

