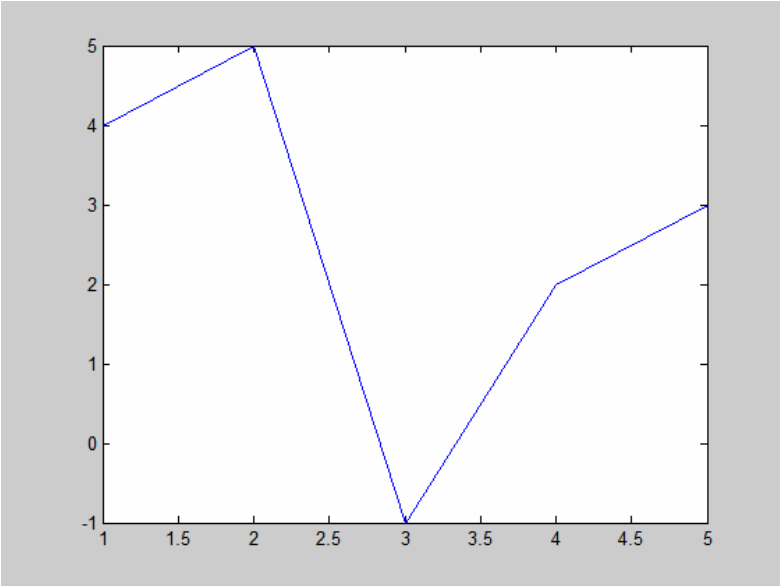
Grafika

A **`plot`** paranccsal síkbeli pontokat rajzolhatunk. Külön vektorban adjuk meg az egyes pontok x- illetve y-koordinátáit.

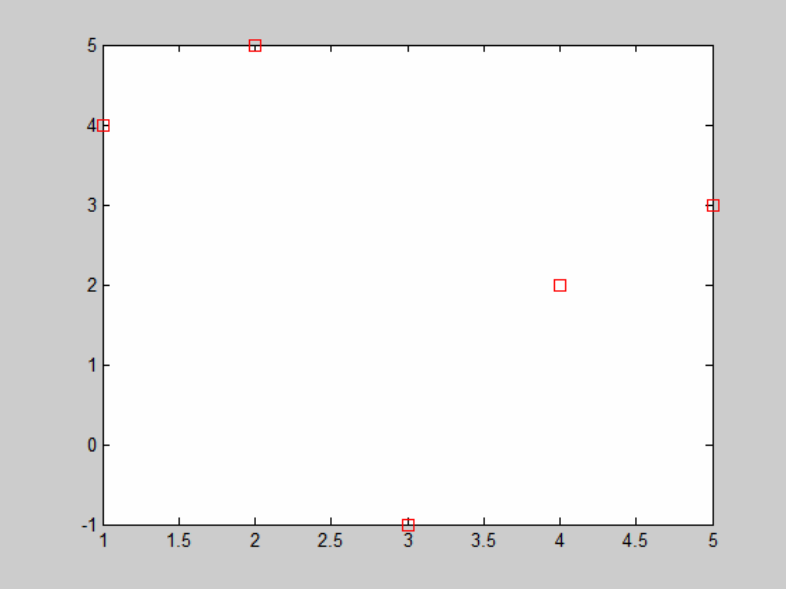
```
>> x=[1,2,3,4,5];
>> y=[4,5,-1,2,3];
>> plot(x,y)
```



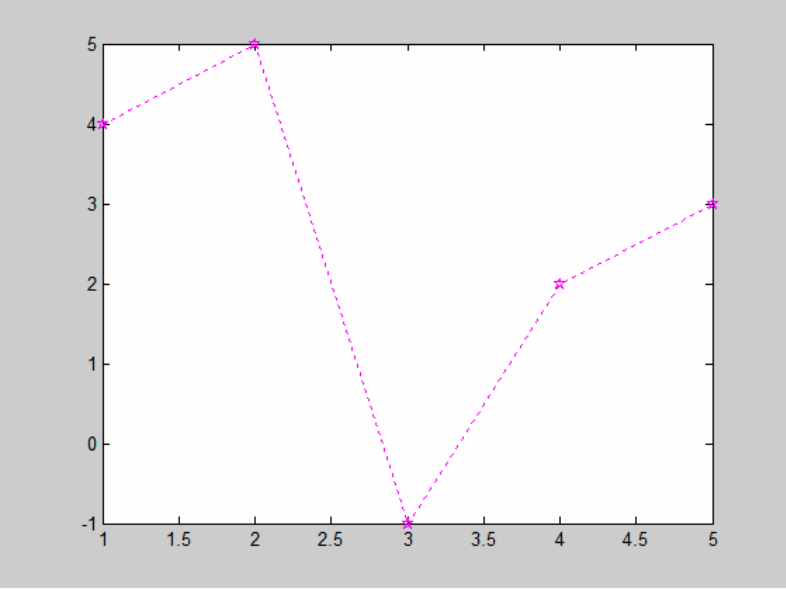
A plot parancs harmadik paraméterében opciókat adhatunk át. Három különböző tulajdonságot adhatunk meg az alábbi táblázatban felsorolt egy-két karakter hosszú kódértékekből összeállított stringben. A három tulajdonság közül bármelyik megadható együtt vagy külön is.

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

```
>> plot(x,y,'sr')
```




```
>> plot(x,y,':pm')
```



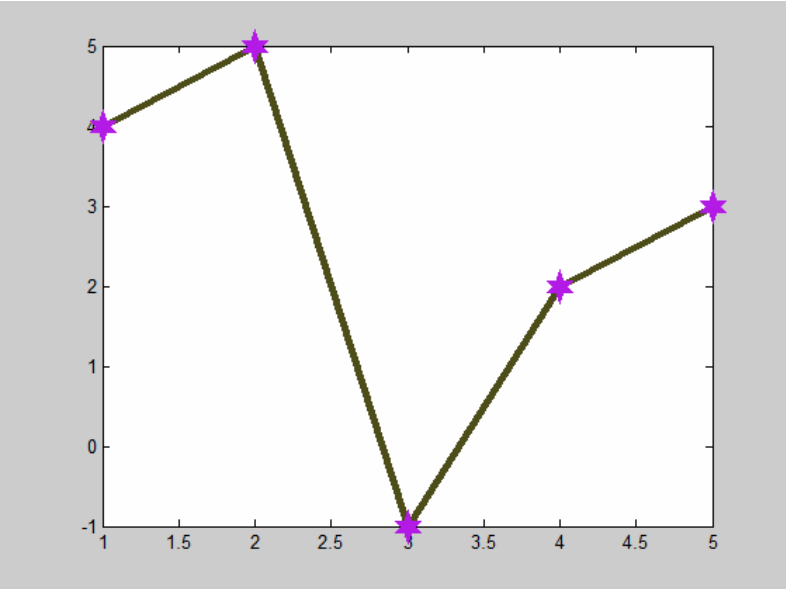
Részletesebb beálltásokat adhatunk meg, ha a plot parancsot az alábbi szintaxissal hívjuk meg: plot(x,y,tulajdonság neve, tulajdonság értéke)
Több tulajdonságot is megadhatunk név-érték párokat ismételve.

Néhány fontosabb beállítható tulajdonság

Color	a vonal színe	színkód (mint az előző táblázatban) vagy RGB kódolással: 3 hosszú vektorban 0 és 1 közötti valós számok megadásával
LineStyle	a vonal stílusa	mint a fenti táblázatban: - -- : -.

LineWidth	vonal vastagsága	egész szám (a vastagság mértéke point-ban)
Marker	a szimbólum	mint a fenti táblázatban
MarkerEdgeColor	a szimbólum színe	színkód (mint az előző táblázatban) vagy RGB kódolással: 3 hosszú vektorban 0 és 1 közötti valós számok megadásával
MarkerFaceColor	a szimbólum kitöltésének színe	színkód (mint az előző táblázatban) vagy RGB kódolással: 3 hosszú vektorban 0 és 1 közötti valós számok megadásával
MarkerSize	a szimbólum mérete	egész szám

```
>> plot(x,y,'LineWidth',4,'Marker','h','MarkerSize',8,'Color',
[0.3 0.3 0.1],'MarkerEdgeColor',[0.7 0.1 0.9])
```

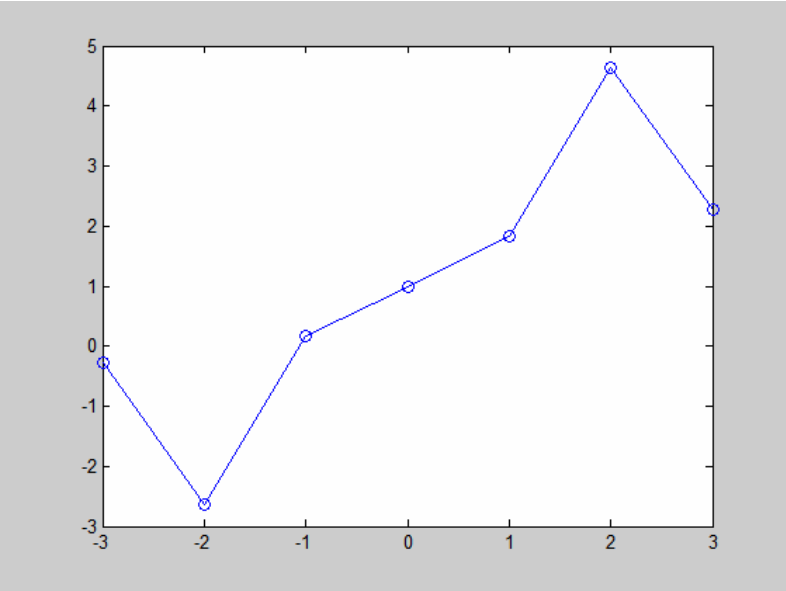


Függvénygrafikon rajzolása

Rajzoljuk ki az $f(x)=x^2 sin(x)+1$ függvény grafikonját a [-3,3] intervallumon.
Ehhez veszünk egy beosztást a [-3,3] intervallumon, és le kell generálnunk a beosztás pontjaiban a függvényértékeket. Ezt legegyszerűbben úgy kapjuk meg, ha „vektoriálisan” írjuk a függvény képletét, azaz használjuk a koordinátánkénti hatványozás, szorzás műveletét:

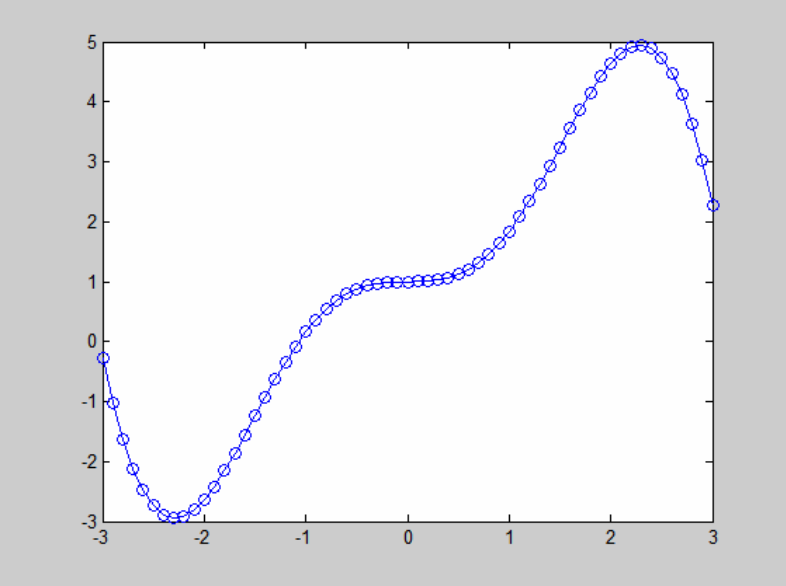
```
>> x=-3:3

x =

    -3    -2    -1     0     1     2     3
```

```
>> y=x.^2.*sin(x)+1

y =

   -0.2701   -2.6372    0.1585    1.0000    1.8415    4.6372
2.2701

>> plot(x,y,'o')
```



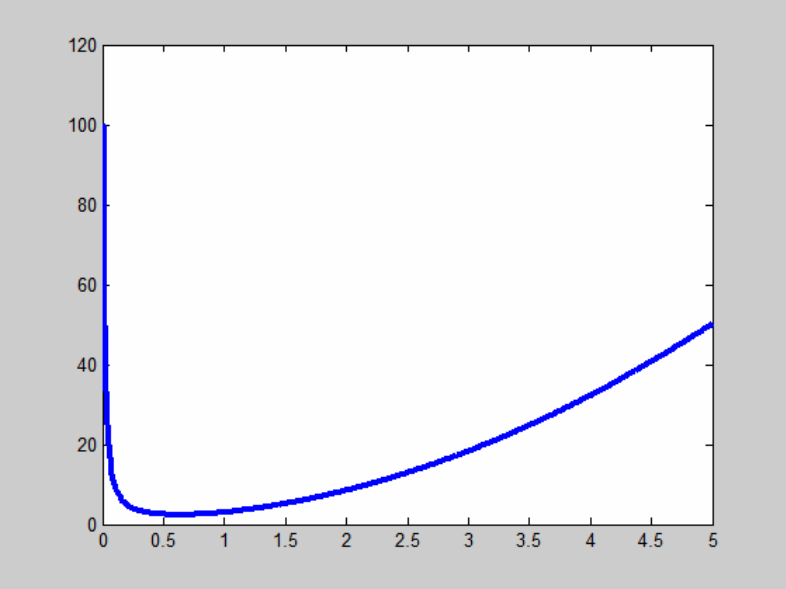
Ugyanez finomabb beosztást használva:

```
>> x=-3:0.1:3;
>> y=x.^2.*sin(x)+1;
>> plot(x,y,'-o')
```



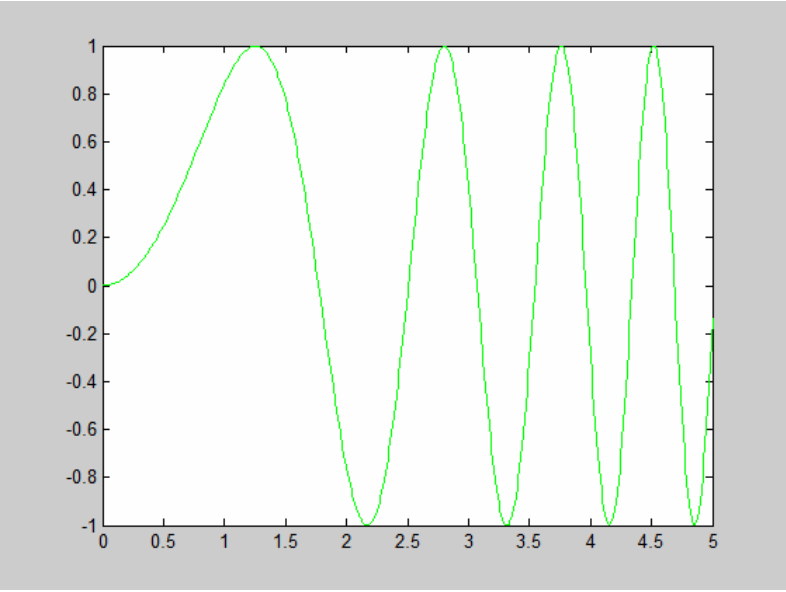
Egy másik példa, $f(x)=1/x+2x^2$ Az osztást és a négyzetre emelést a ./ illetve .^ művelettel végezzük, de a 2-vel való szorzást a * művelettel jelöljük, hiszen ez koordinátánként definiált.

```
>> x=0.01:0.01:5;
>> y=1./x+2*x.^2;
>> plot(x,y ,'LineWidth',3)
```



Egyváltozós függvény grafikonját az **`fplot`** paranccsal is megkaphatjuk. Az első paraméter egy string, amely egy vektoriálisan kiértékelhető képletet vagy függvény nevét tartalmazza, a második paraméter az intervallum, 2 hosszú vektorként. Harmadik és további paraméterben opciókat adhatunk át, mint a `plot` parancsnál.

```
> fplot('sin(x.^2)',[0,5],'g')
```



Néhány hasznos parancs:

`clf` : grafikus ablak tartalmának törlése
`cla` : koordinátarendszer tartalmának törlése

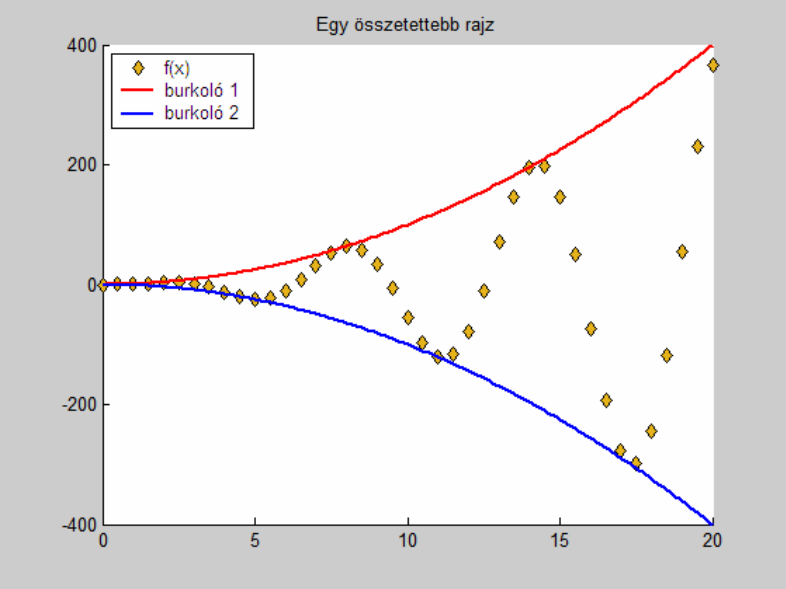
`axis([xmin xmax ymin ymax])`
: a koordinátarendszer határai beállítása a megadott értékekre
`axis auto` : a határok visszaállítása az automatikus beállításra
`axis equal` : az x és y irányú egység azonos hosszúságú

`axis normal`	: az x és y irányban automatikus nyújtás/összenyomás megengedett
`axis off`	: a koordinátarendszert elrejti
`axis on`	: bekapcsolja a koordinátarendszert
`title('cím')`	: grafika címe
`xlabel('felirat')`	: x-tengely felirata
`ylabel('felirat')`	: y-tengely felirata
`text(x,y,'string')`	: adott pozícióra kiír egy szöveget
`hold on`	: ugyanabba a koordinátarendszerbe rajzolja a következő ábrát
`hold off`	: a következő rajzoló parancs letörli a koordinátarendszert rajzolás előtt
`figure(n)`	: új ablak létrehozása, illetve váltás egy létező ablakra, ahol `n` az ablak mutatója (mindig pozitív egész)
`axes(h)`	: Egy üres koordinátarendszert hoz létre, ha paraméter nélkül hívjuk meg a függvényt, illetve átvált a `h` mutató által definiált koordinátarendszerre. A parancs outputja egy mutató a koordinátarendszer objektumára.
`gca`	: Az aktuális koordinátarendszer mutatóját adja vissza
`gcf`	: Az aktuális ablak mutatóját adja vissza

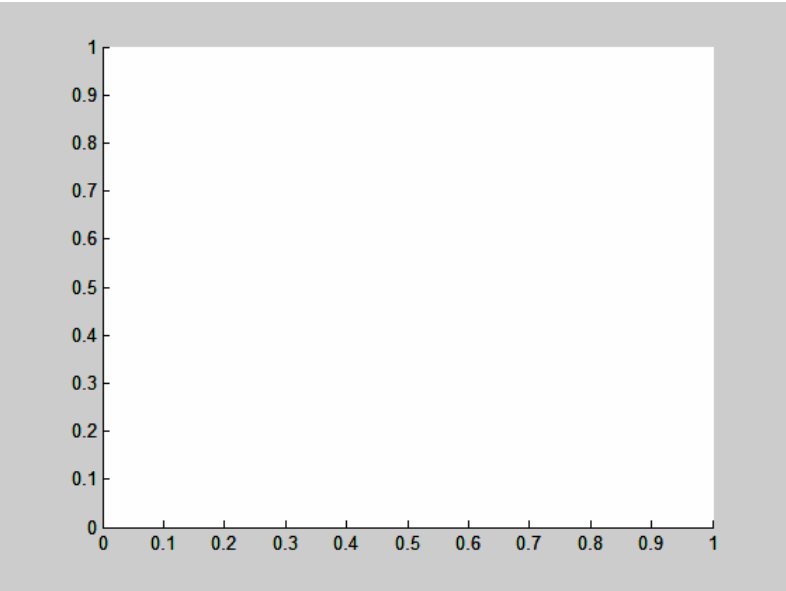
Egy script egy összetettebb grafika rajzolására:

```
% grafikai beállítások : pl2.m

x=0:0.1:20;
t=0:0.5:20;
y=t.^2.*sin(t);
z=x.^2;

hold on
h1=plot(t,y,'d')
h2=plot(x,z)
h3=plot(x,-z)
set(h1,'Color','k','MarkerFaceColor',[0.9 0.7 0.1])
set(h2,'LineWidth',2,'Color','r')
set(h3,'LineWidth',2,'Color','b')
set(gca,'XTick',[0 5 10 15 20],'YTick',[-400 -200 0 200 400])
title('Egy összetettebb rajz')
legend('f(x)','burkoló 1','burkoló 2',2)
% ------------------------------------------------------------------
```

```
>> pl2
```




```
>> h=axes

h =

  104.0087
```



A `get(h)` parancs a `h` mutató által definiált objektum összes tulajdonságát adja vissza és írja ki a képernyőre:

```
>> get(h)
      ALim = [0 1]
      ALimMode = auto
      AmbientLightColor = [1 1 1]
      Box = off
      CameraPosition = [0.5 0.5 9.16025]
      CameraPositionMode = auto
      CameraTarget = [0.5 0.5 0.5]
      CameraTargetMode = auto
```

```
CameraUpVector = [0 1 0]
CameraUpVectorMode = auto
CameraViewAngle = [6.60861]
CameraViewAngleMode = auto
CLim = [0 1]
CLimMode = auto
Color = [1 1 1]
CurrentPoint = [ (2 by 3) double array]
ColorOrder = [ (7 by 3) double array]
DataAspectRatio = [1 1 1]
DataAspectRatioMode = auto
DrawMode = normal
FontAngle = normal
FontName = Helvetica
FontSize = [10]
FontUnits = points
FontWeight = normal
GridLineStyle = :
Layer = bottom
LineStyleOrder = -
LineWidth = [0.5]
NextPlot = replace
PlotBoxAspectRatio = [1 1 1]
PlotBoxAspectRatioMode = auto
Projection = orthographic
Position = [0.13 0.11 0.775 0.815]
TickLength = [0.01 0.025]
TickDir = in
TickDirMode = auto
Title = [105.004]
Units = normalized
View = [0 90]
XColor = [0 0 0]
XDir = normal
XGrid = off
XLabel = [106.003]
XAxisLocation = bottom
XLim = [0 1]
XLimMode = auto
XScale = linear
XTick = [ (1 by 11) double array]
XTickLabel = [ (11 by 3) char array]
XTickLabelMode = auto
XTickMode = auto
YColor = [0 0 0]
YDir = normal
YGrid = off
YLabel = [107.003]
YAxisLocation = left
YLim = [0 1]
YLimMode = auto
YScale = linear
YTick = [ (1 by 11) double array]
```

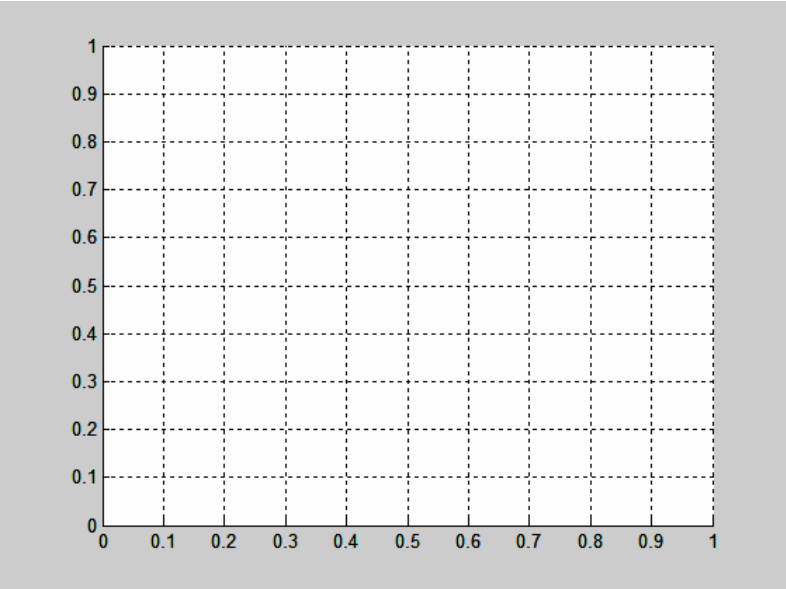
```
    YTickLabel = [ (11 by 3) char array]
    YTickLabelMode = auto
    YTickMode = auto
    ZColor = [0 0 0]
    ZDir = normal
    ZGrid = off
    ZLabel = [108.003]
    ZLim = [0 1]
    ZLimMode = auto
    ZScale = linear
    ZTick = [0 0.5 1]
    ZTickLabel =
    ZTickLabelMode = auto
    ZTickMode = auto

    BeingDeleted = off
    ButtonDownFcn =
    Children = []
    Clipping = on
    CreateFcn =
    DeleteFcn =
    BusyAction = queue
    HandleVisibility = on
    HitTest = on
    Interruptible = on
    Parent = [1]
    Selected = off
    SelectionHighlight = on
    Tag =
    Type = axes
    UIContextMenu = []
    UserData = []
    Visible = on
```

A `get` parancs második paraméterében megadhatjuk, hogy az adott objektum melyik tulajdonságát kérdezzük le:

```
>> get(h,'XGrid')

ans =

off
```

A `set` paranccsal az egyes tulajdonságok módosíthatók:

```
>> set(h,'XGrid','on','YGrid','on')
```

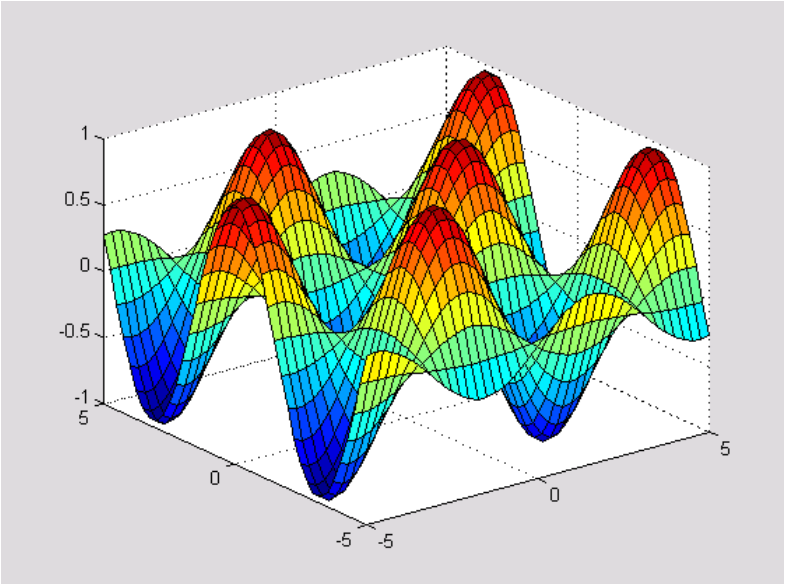


Kétváltozós függvény grafikonjának rajzolása

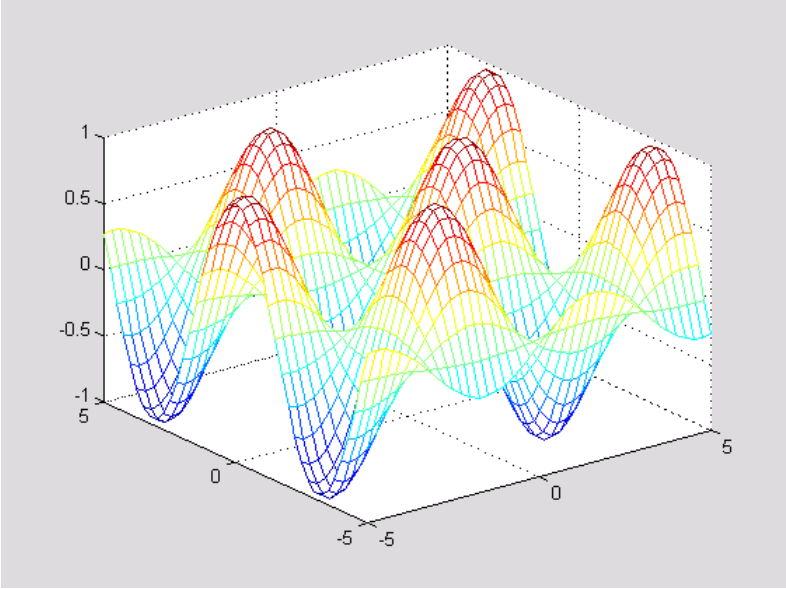
Legyen *f* egy téglalapon értelmezett kétváltozós függvény, *x* és *y* a téglalap beosztása a két oldal mentén (legyen *x* egy *n* hosszú, *y* pedig egy *m* hosszú vektor). Ezután az így kapott rácspontokban ki kell értékelni a megadott függvényt, a kapott függvényértékeket pedig egy *z*, $n \times m$ dimenziós mátrixban tároljuk. Ezzel a 3 változóval hívhatjuk meg a különböző rajzoló parancsokat.
A függvény kiértékelését vektoriálisan a következő módon tehetjük meg a legkönnyebben: A rácspontok x illetve y koordinátáit felsoroljuk *nm* hosszú vektorokban a `meshgrid` parancs segítségével, és ezekkel a vektorokkal hívjuk meg a vektoriálisan felírt képletet.
A különböző stílusú rajzoló parancsoknak a meghívása azonos.

Rajzoljuk ki az $f(x,y)=\sin(x)\cos(y)$, $(x,y)\in[-5,5]\times[-5,5]$ függvény grafikonját:

```
>> x=-5:0.25:5;
>> y=x;
>> [xx,yy]=meshgrid(x,y);
>> z=sin(xx).*cos(yy);
>> surf(x,y,z)
```



```
>> mesh(x,y,z)
```



További, különböző stílusú rajzoló parancsok:

```
surfc(x,y,z)
meshc(x,y,z)
waterfall(x,y,z)
contour(x,y,z)
```

Néhány előre definiált színtérképet használhatunk a felületek színezéséhez. Színtérkép beállítása:

```
colormap('default')
colormap(hot)
```

Hasonló előre definiált színtérképek például:
`autumn, bone, cool, copper, gray, pink`

Grafika elforgatása:

`view(a,b)` : az adott irányból nézhetünk rá a grafikára, `a` és `b` két szög, `a` az xz-síkhoz képesti elforgatás szöge, `b` pedig az xy-síkhoz képesti szög.