Metode Numerice
Curs 2: Grafică în Matlab

Octavia-Maria BOLOJAN

obolojan@uoradea.ro
octavia.nica@math.ubbcluj.ro

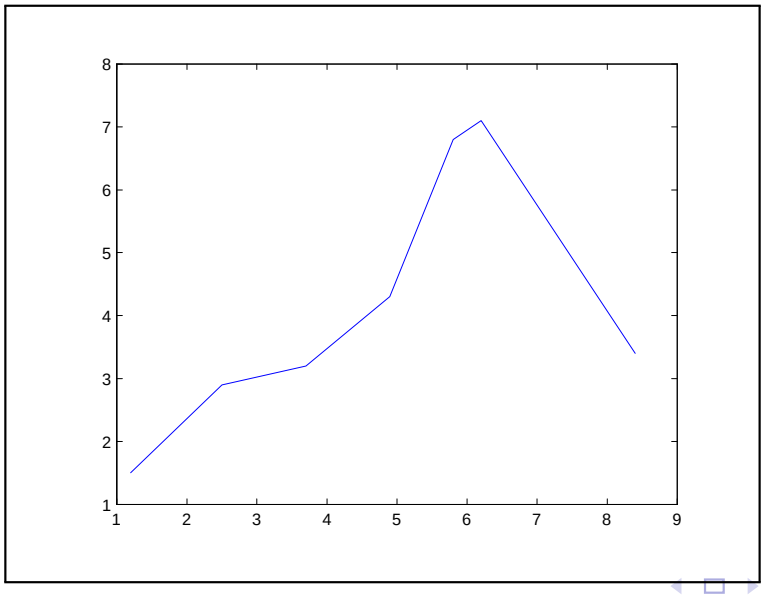
12 Octombrie 2018

1. Grafice bidimensionale

- Funcția Matlab `plot()` realizează grafice bidimensionale unind punctele vecine din două seturi de valori
- Spre exemplu, avem următoarele secvențe de cod scrise într-un script Matlab:

```
x=[1.2, 2.5, 3.7, 4.9, 5.8, 6.2, 8.4];
y=[1.5, 2.9, 3.2, 4.3, 6.8, 7.1, 3.4];
plot(x,y)
```

- Rezultatul compilării va fi o linie poligonală care unește punctele $x(i), y(i)$ din vectorii x și y, conform figurii de mai jos



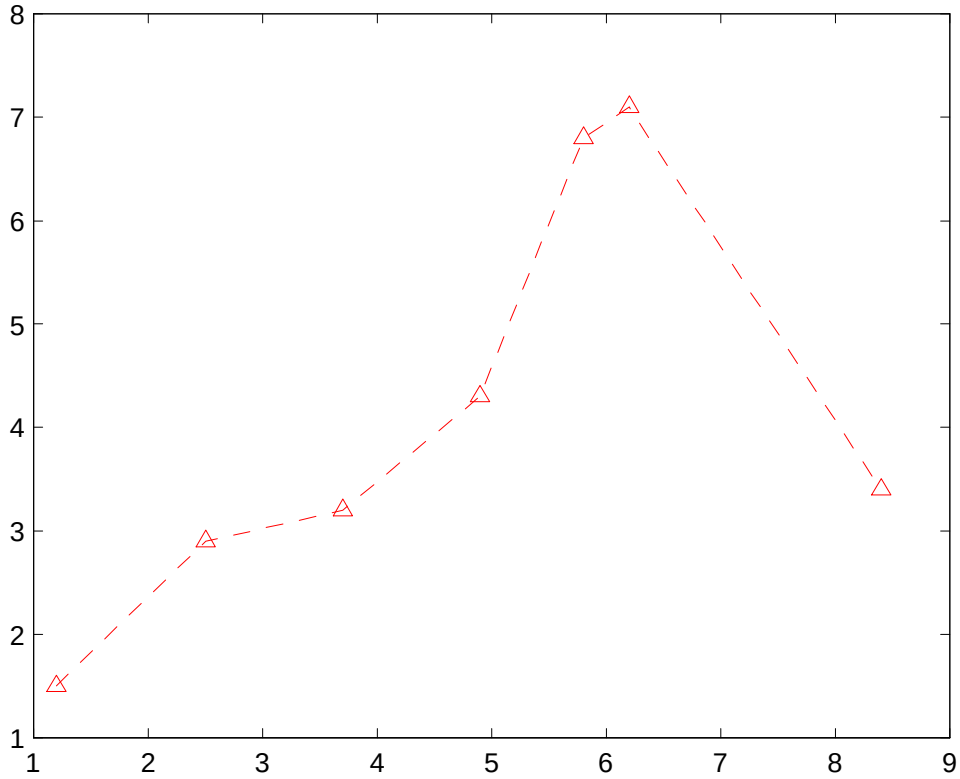
- În acest exemplu se utilizează valori implicite pentru tipul de linie al reprezentării grafice, culoare, domeniul pentru axele x și y sau spațiile dintre diviziunile pe axe
- Putem însă opta pentru reprezentări neimplicite, adăugând în funcția `plot()` un al treilea argument, reprezentând specificatori pentru culoare, marcajul punctelor din reprezentare sau stilul liniei
- Forma generală a comenzii este

plot(x,y,'specificatori reprezentare grafică').

- Astfel, avem câteva posibile implementări:

`plot(x,y,'r*- -')` marchează faptul că punctele (x(i),y(i)) vor fi reprezentare cu simbolul asterisc de culoare roșie, linia graficului fiind întreruptă, tot de culoare roșie
`plot(x,y,'+m')` marchează faptul că punctele (x(i),y(i)) vor fi reprezentare cu simbolul '+' de culoare magenta, fără a fi unite cu vreo linie

Specificațiile pentru reprezentare din cadrul funcției plot() pot fi date în orice ordine. De exemplu, `plot(x,y,'r^- -')` și `plot(x,y,'^r- -')` vor avea același efect:



Funcția `plot()` acceptă și mai multe seturi de date. Astfel, putem avea apeluri de forma `plot(x1,y1,'g- -',x2,y2,'b-')` care generează în aceeași figură graficele pentru `x1(i),y1(i)` și `x2(i),y2(i)` cu linie întreruptă verde, respectiv linie continuă de culoare albastră.

- Opțiuni de culoare, marcaj, stil de linie pentru reprezentările grafice realizate cu comanda `plot()`:

`r`	roșu
`g`	verde
`b`	albastru
`c`	cian
`m`	magenta
`y`	galben
`k`	negru
`w`	white

`o`	cerc
`*`	asterisc
`.`	punct
`s`	pătrat
`d`	romb
`^`	triunghi în sus
`v`	triunghi în jos
`>`	triunghi dreapta
`<`	triunghi stânga
`p`	pentagramă (stea cu 5 colțuri)
`h`	hexagramă (stea cu 6 colțuri)

`-`	linie continuă
`--`	linie întreruptă
`:`	linie punctată
`-.`	linie-punct

- Dacă o comandă de reprezentare grafică este urmată de alta, atunci noua imagine o va înlocui pe cea de dinainte sau va fi suprapusă peste ea, depinzând de starea `'hold'` folosită la momentul respectiv
- Comanda `hold on` permite ca toate graficele care urmează să fie suprapuse peste reprezentarea grafică curentă, iar comanda `'hold off'` va face ca fiecare nouă reprezentare grafică să înlocuiască imaginea precedentă
- În mod implicit, avem starea setată pe `'hold off'`

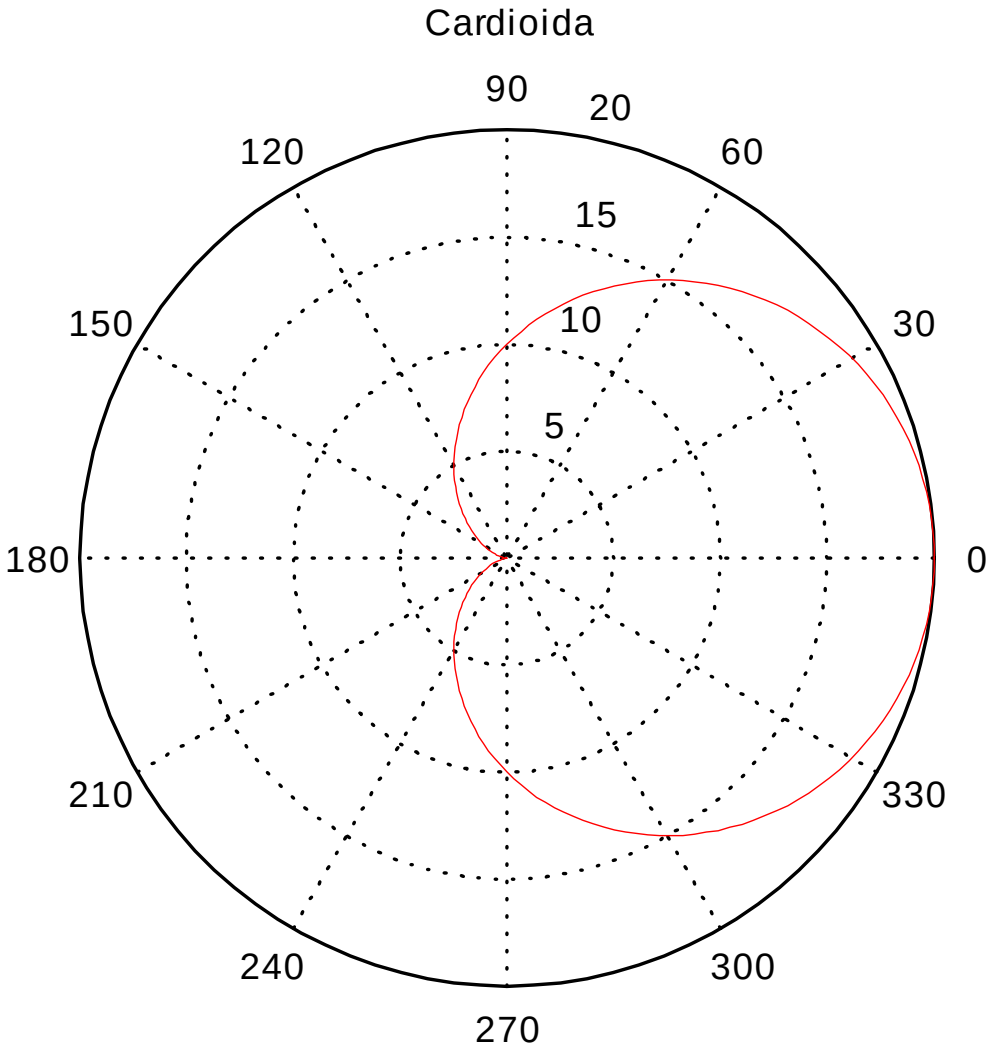
- Pentru **reprezentarea grafică a curbelor în coordonate polare**, folosim comanda `polar(t,r)`, unde `t` este unghiul polar, iar `r` este raza polară
- Se pot folosi și atribute suplimentare, la fel ca în cazul funcției `plot()`

De exemplu, dacă dorim să reprezentăm graficul curbei în coordonate polare, numită și **cardioidă**, de forma

$$r = a(1 + \cos t), \quad t \in [0, 2\pi], a \in \mathbb{R},$$

acesta poate fi obținut folosind secvențele de cod de mai jos (pentru $a = 10$):

```
t=0:pi/100:2*pi;
a=10;
r=a*(1+cos(t));
polar(t,r,'r');
title('Cardioida')
```



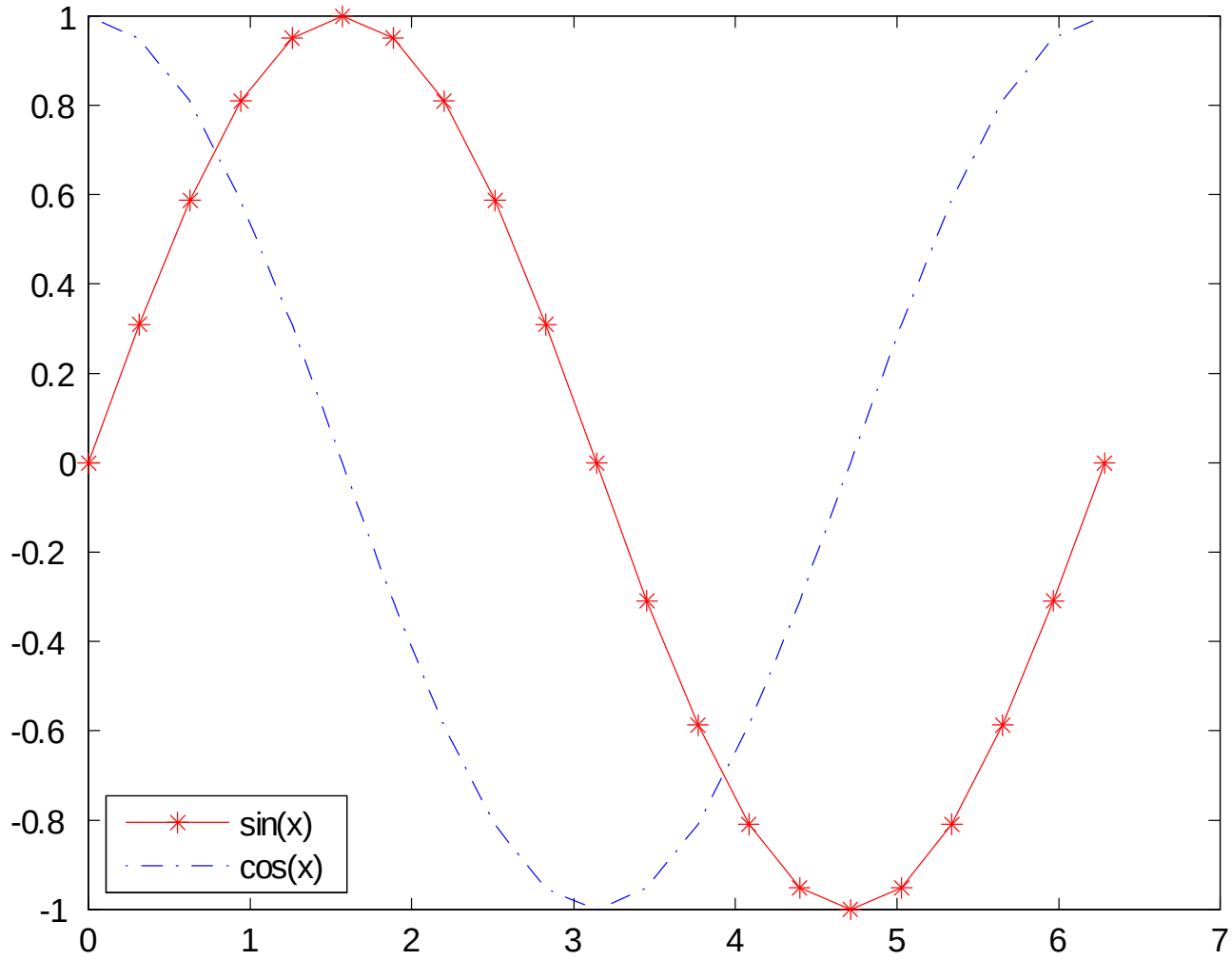
- Reprezentării grafice i s-a adăugat și titlu cu ajutorul comenzii `title('Cardioida')`.

- Putem adăuga și legendă unei reprezentări grafice cu ajutorul comenzii

  `legend('şir1','şir2',...,'şirk',p)`,

  unde ultimul parametru indică poziția legendei (a se consulta `help legend`).
- De exemplu, dacă dorim să afișăm în același grafic funcțiile $\sin x$ și $\cos x$ pe intervalul $[0, 2\pi]$ și să adăugăm o legendă corespunzătoare, folosim secvențele de cod de mai jos:

```
x=0:pi/10:2*pi;
plot(x,sin(x),'-r*',x,cos(x),'-.b')
h=legend('sin(x)','cos(x)',3);
```

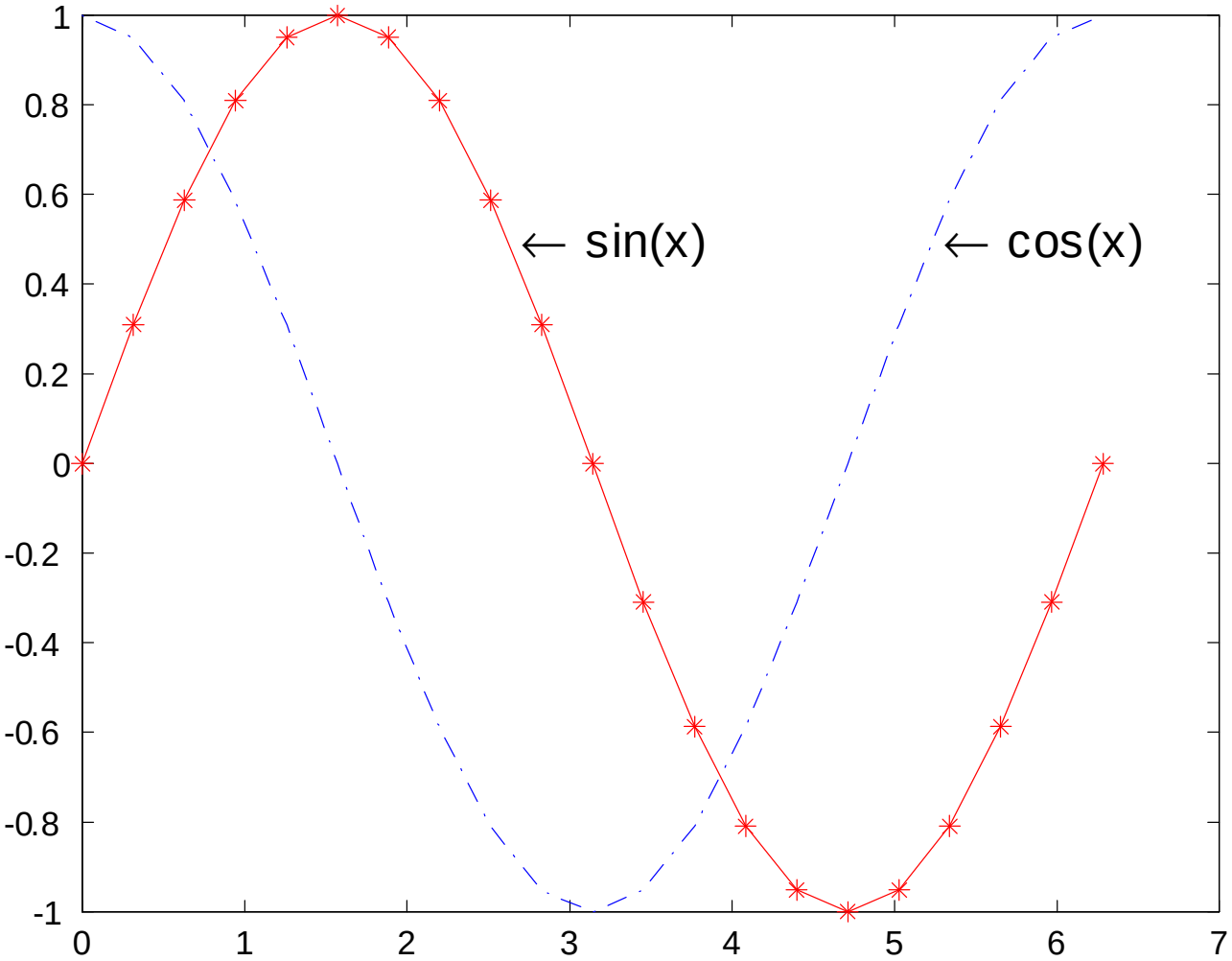


Putem adăuga de asemenea text într-un grafic cu ajutorul comenzii `text(x,y,'şir')`, unde `x,y` reprezintă coordonatele de poziționare a textului, iar `'şir'` este un șir de caractere sau o variabilă de tip șir.

Pentru exemplul, anterior, avem următoarele secvențe de cod:

```
x=0:pi/10:2*pi;
plot(x,sin(x),'-r*',x,cos(x),'-.b')
text(2.7,0.5,'\leftarrow sin(x)','FontSize',14)
hold on
text(5.3,0.5,'\leftarrow cos(x)','FontSize',14)
```

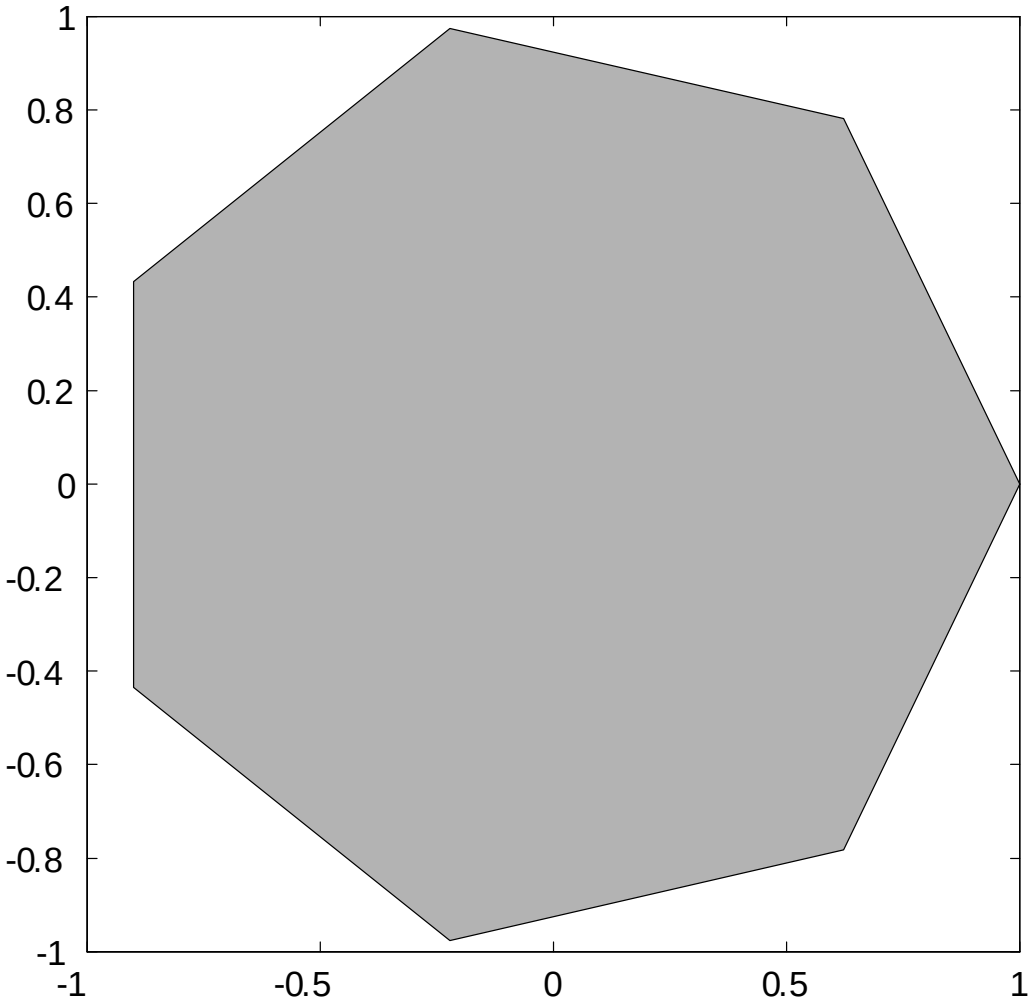
care vor genera figura:



- Funcția `fill()` este similară cu `plot()`
- Comanda `fill(x,y,c)` reprezintă poligonul cu vârfurile `x(i), y(i)` în culoarea `c`; Punctele se iau în ordine și ultimul se unește cu primul
- Culoarea `c` se poate da și sub forma unui triplet RGB, `[r g b]`
- Elementele `r,g,b`, care trebuie să fie scalari din intervalul $[0, 1]$, determină nivelul de roșu, verde și albastru din culoarea finală
- Astfel, `fill(x,y,[0 1 0])` umple poligonul cu culoarea verde, iar `fill(x,y,[1 0 1])` umple poligonul cu culoarea magenta
- Dând proporții egale de roșu, verde și albastru se obțin nuanțe de gri care variază de la negru (`[0 0 0]`) la alb (`[1 1 1]`)

Considerăm secvența (generare heptagon regulat colorat cu gri):

```
n=7;
t=2*(0:n-1)*pi/n;
fill(cos(t),sin(t),[0.7,0.7,0.7])
axis square
```



- Diversele aspecte ale unui grafic pot fi controlate cu comanda `axis`
- Axele pot fi eliminate cu comanda `axis off`
- Comenzi pentru controlul axelor:

`axis([xmin xmax ymin ymax])`	setează limitele axelor Ox și Oy
`axis auto`	returnează limitele implicite
`axis equal`	egalează unitățile pe axele de coordonate
`axis off`	elimină axele
`axis square`	face caseta axelor pătrată (cubică)
`xlim([xmin xmax])`	setează limitele pe axa Ox
`ylim([ymin,ymax])`	setează limitele pe axa Oy

- Este posibil de asemenea să reprezentăm mai multe grafice pe aceeași figură cu ajutorul funcției Matlab `subplot()`
- Astfel, apelul `subplot(mnp)` sau `subplot(m,n,p)` determină împărțirea ferestrei în care se fac reprezentările într-o matrice de $m \times n$ regiuni, fiecare regiune având propriile axe
- Al treilea parametru, p, va indica regiunea în care să fie făcută reprezentarea curentă
- De exemplu, comanda `subplot(311)` împarte fereastra sub forma unei matrice 3×1 regiuni (3 linii, o coloană) și va poziționa pe prima poziție (prima linie/regiune) graficul rezultat ca urmare a comenzii de reprezentare grafică plasate imediat după `subplot(311)`
- Dăm mai jos un exemplu de utilizare a funcției `subplot()`, alături de o altă funcție folosită pentru reprezentările grafice, și anume, `fplot()`
- Funcția `fplot()` având sintaxa

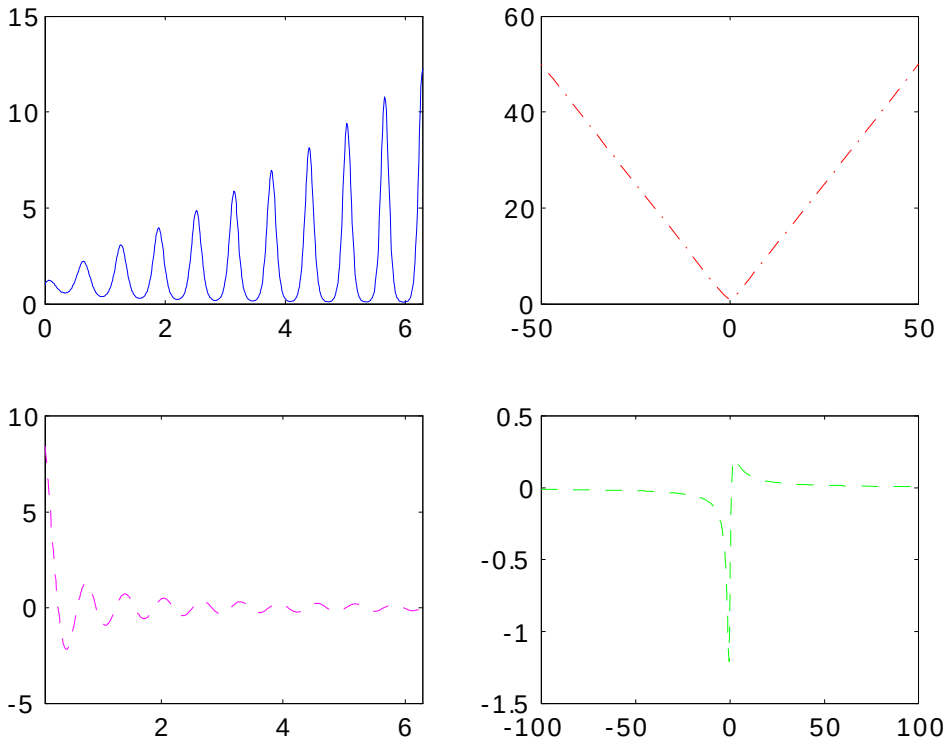
  `fplot('expresie funcţie',interval)`

  este deosebit de utilă, întrucât aceasta permite alegerea în mod implicit a numărului de puncte din intervalul pe care se dorește reprezentarea, pentru a produce un grafic suficient de exact

Astfel, următoarele secvențe de cod

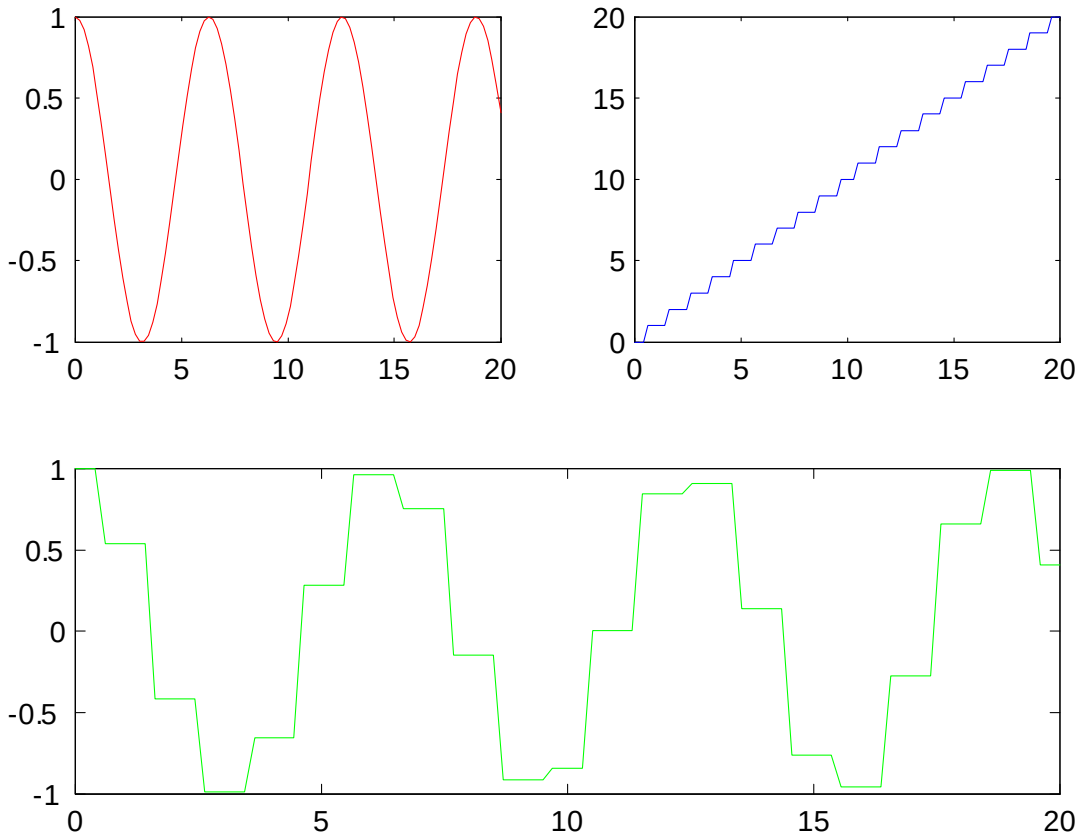
```
subplot(221),fplot('exp(sqrt(x)*cos(10*x))',[0 2*pi])
subplot(222),fplot('sqrt(x^2+1)',[-50 50],'r-.')
subplot(223),fplot('sin(10*x)/x',[0.1 2*pi],'--m')
subplot(224),fplot('(x-1)/(x^2+1)',[-100 100],'--g')
```

vor avea ca rezultat imaginea:



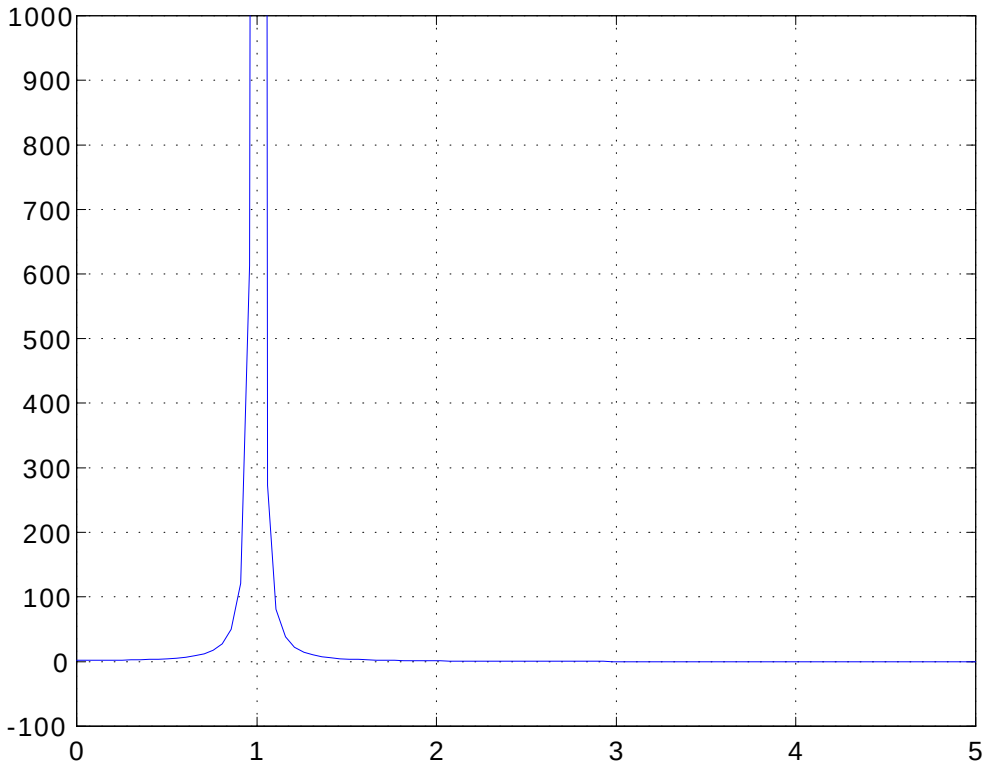
Cu ajutorul comenzii `subplot()`, putem genera și reprezentări în mai multe regiuni amplasate succesiv, prin specificații diferite:

```
x=linspace(0,20,100)
subplot(2,2,1),plot(x,cos(x),'r')
subplot(2,2,2),plot(x,round(x))
subplot(2,2,3:4),plot(x,cos(round(x)),'g')
```



În continuare, va fi reprezentată grafic funcția $f(x) = 1/(x+1)^2 + 1/(x+2)^3$ pe intervalul $[0, 5]$ și va fi adăugată o grilă de linii orizontale și verticale care pornesc de la diviziunile axelor cu ajutorul comenzii `grid on`, precum și limitele pe axa *Oy* cu ajutorul comenzii `ylim([ymin,ymax])`:

```
x=linspace(0,5,100);
plot(x,1./(x-1).^2+1./(x+1).^2)
grid on
ylim([-100,1000])
```



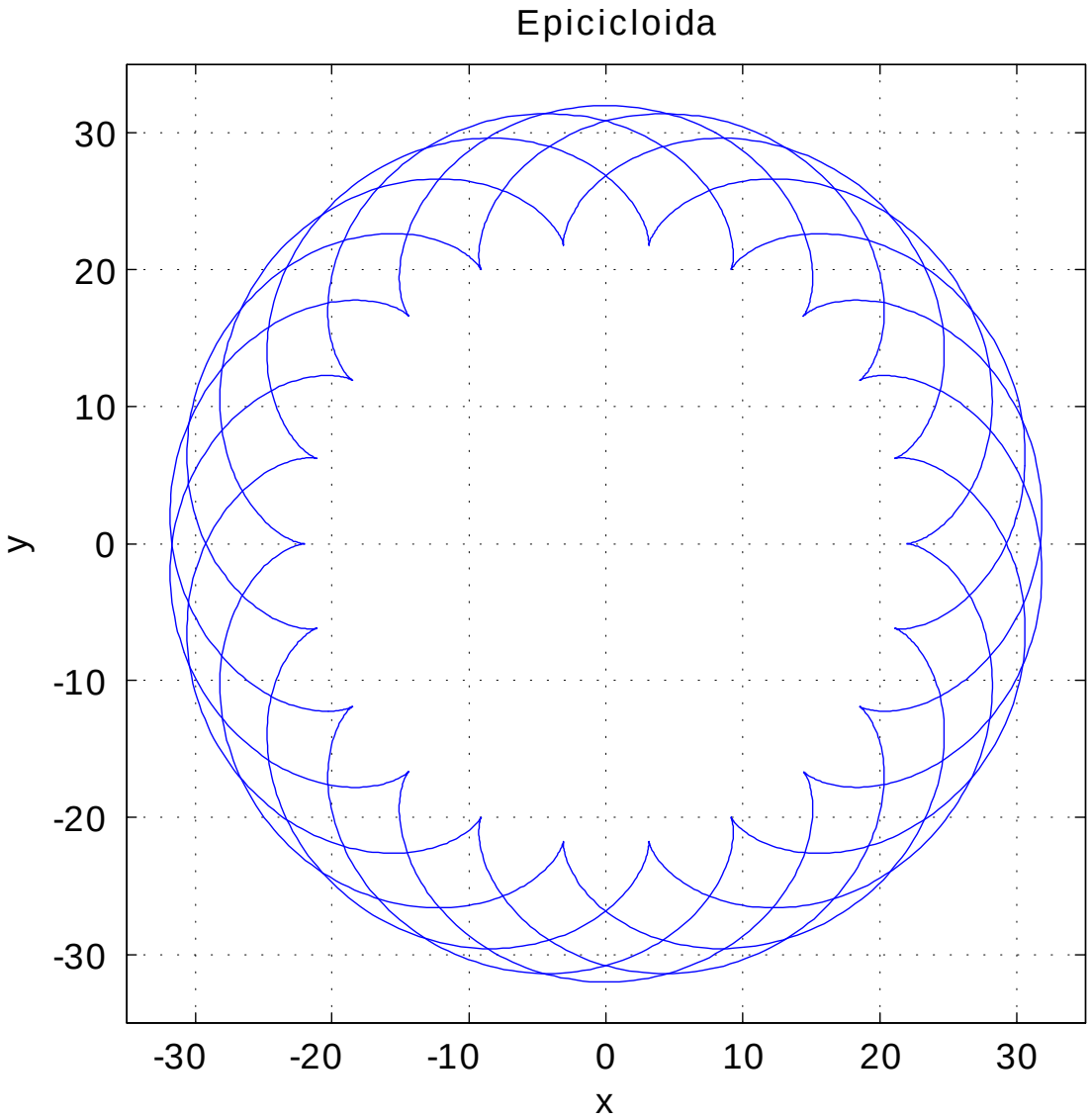
În exemplul care urmează, se va reprezenta grafic **epicicloida**

$$\begin{cases} x(t) = (a+b)\cos(t) - b\cos((a/b+1)t) \\ y(t) = (a+b)\sin(t) - b\sin((a/b+1)t) \end{cases}, t \in [0, 10\pi]$$

pentru $a = 22, b = 5$.

Utilizăm următoarele secvențe de cod:

```
a=22; b=5;
t=0:0.01:10*pi;
x=(a+b)*cos(t)-b*cos((a/b+1)*t);
y=(a+b)*sin(t)-b*sin((a/b+1)*t);
plot(x,y)
axis equal
axis([-35 35 -35 35])
grid on
title('Epicicloida')
xlabel('x'),ylabel('y')
```


Epicicloida
30
20
10
0
-10
-20
-30
y
-30
-20
-10
0
10
20
30
x

- Funcții pentru grafice 2D:

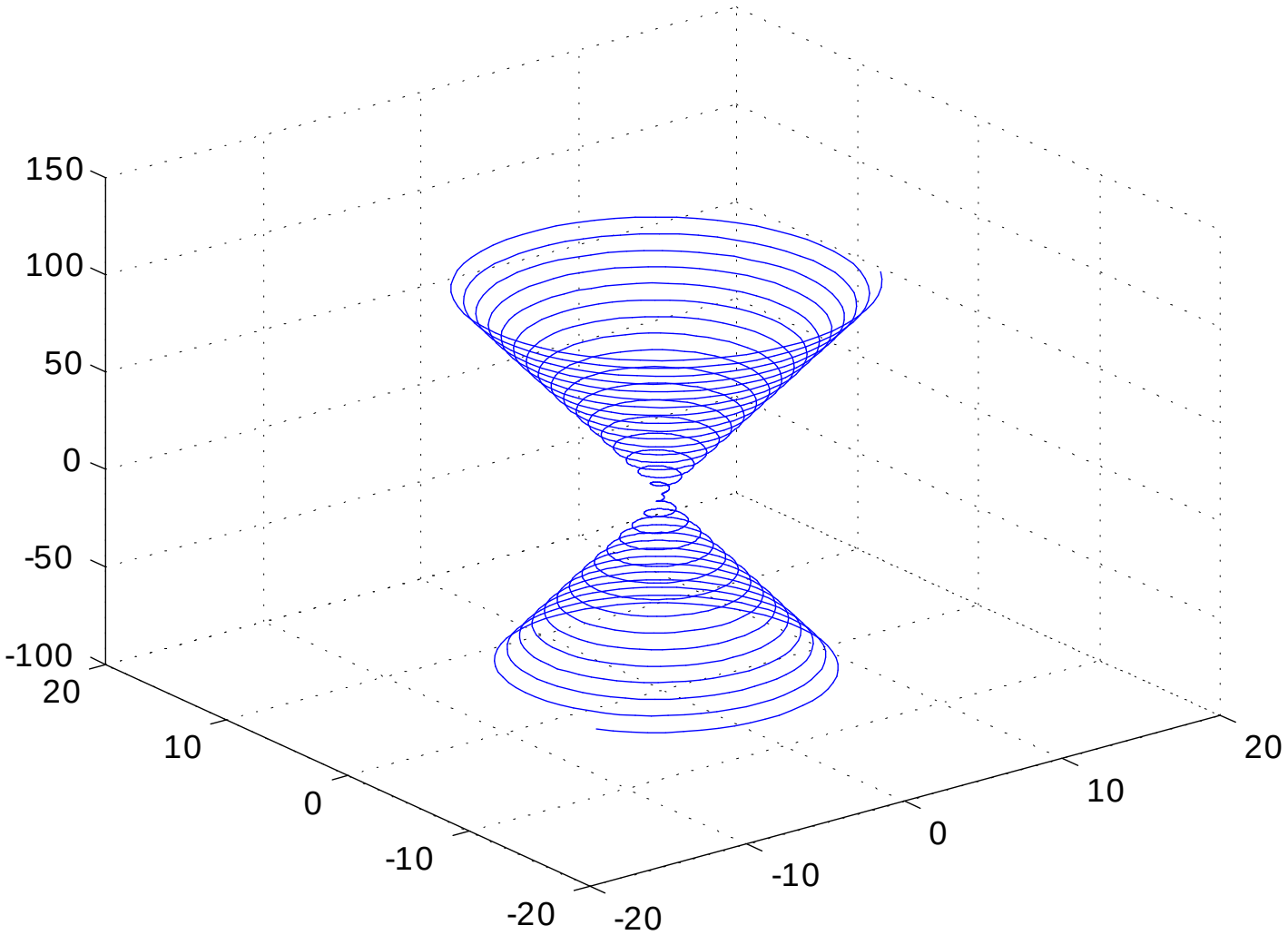
`plot`	grafic simplu 2D
`loglog`	grafic cu scară logaritmică pe ambele axe
`semilogx`	grafic cu scară logaritmică pe axa Ox
`semilogy`	grafic cu scară logaritmică pe axa Oy
`polar`	grafic polar
`fplot`	reprezentare grafică automată a unei funcții
`ezplot`	versiune ușor de utilizat (easy-to-use) a lui `fplot/plot`
`ezpolar`	versiune ușor de utilizat (easy-to-use) a lui `polar`
`fill`	umplere poligon
`area`	grafic de tip arie plină
`bar`	grafic de tip bară
`barh`	grafic de tip bară orizontală
`hist`	grafic de tip histogramă
`pie`	grafic cu sectoare de cerc
`comet`	grafic animat
`errorbar`	grafic cu bare de eroare
`quiver`	câmp cu vectori bidimensional
`scatter`	grafic dispersat (nor de puncte)

2. Grafice tridimensionale

- Analogul comenzii `plot()` pentru reprezentările grafice tridimensionale îl reprezintă comanda `plot3()`
- Spre exemplu, dacă avem următoarele secvențe de cod:

```
t=-10:0.01:10;
x=(1+t).*cos(10*t);
y=(1+t).*sin(10*t);
z=10*(1+t);
plot3(x,y,z)
grid on
```

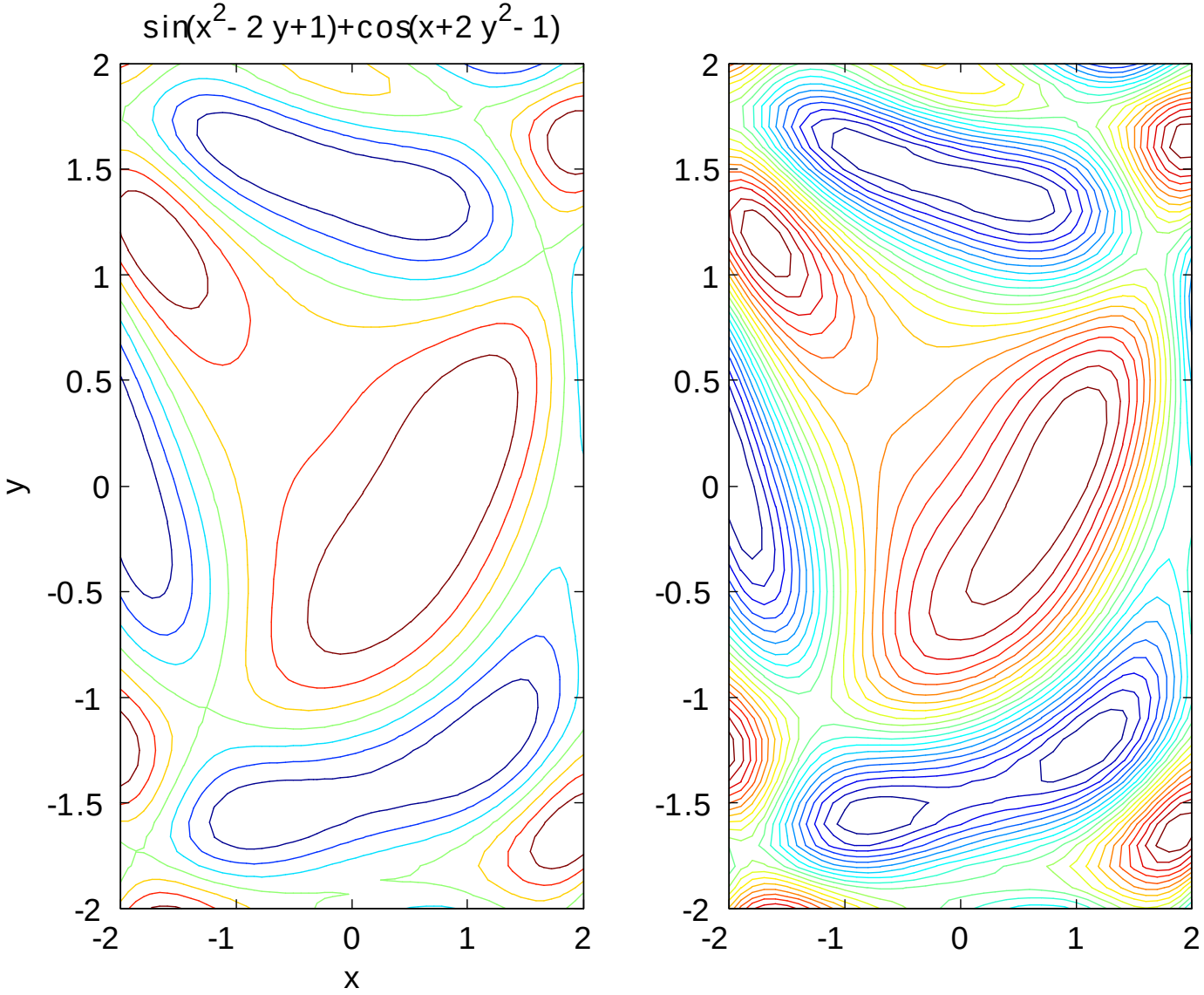
Acestea vor genera figura:



- Pentru desenarea contururilor unui grafic 3D putem folosi funcția `ezcontour()` și `contour()`
- Spre exemplu, vom dori să reprezentăm contururi pentru funcția $f(x,y) = \sin(x^2 - 2y + 1) + \cos(x + 2y^2 - 1)$ pentru $x \in [-2,2], y \in [-2,2]$
- Într-un script Matlab, considerăm următoarele secvențe de cod:

```
subplot(121)
ezcontour('sin(x^2-2*y+1)+cos(x+2*y^2-1)',[-2,2,-2,2]);
subplot(122)
x=-2:0.1:2;y=-2:0.1:2;
[X,Y]=meshgrid(x,y);
f=sin(X.^2-2*Y+1)+cos(X+2*Y.^2-1);
contour(x,y,f,20)
```

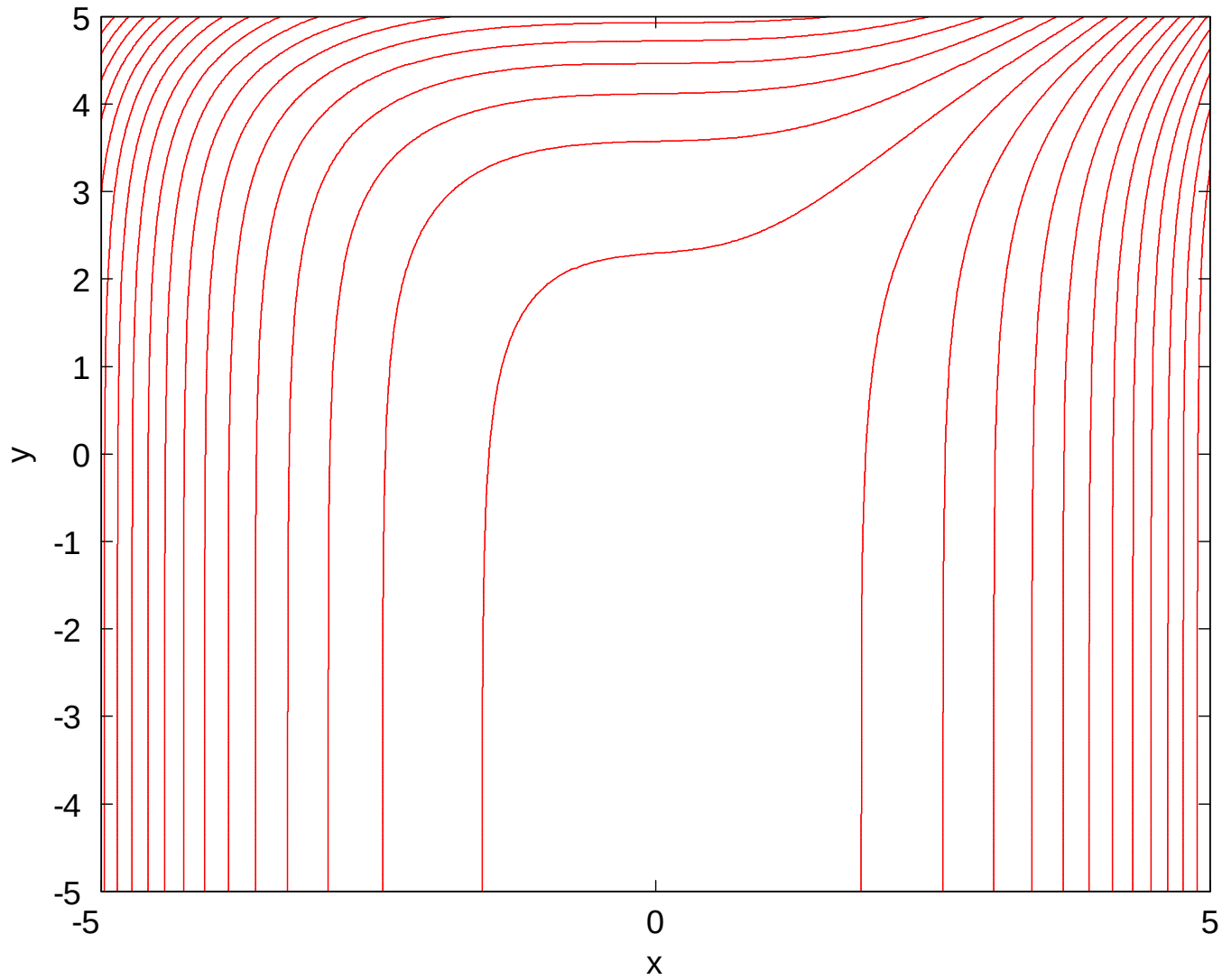
Obținem imaginea:



- În primul caz, nivelurile de contur sunt alese în mod implicit
- În al doilea caz, după ce am dat valorile din vectorii x, y, folosim comanda `[X,Y]=meshgrid(x,y)`, prin intermediul căreia obținem matricele `X` și `Y` astfel încât fiecare linie a lui `X` să fie o copie a vectorului `x` și fiecare coloană a lui `Y` să fie o copie a vectorului `y`
- În variabila `f` reținem matricea ce este generată prin operații de tip tablou din `X` și `Y`, în care fiecare element `f(i,j)` reține valoarea funcției corespunzătoare valorilor `x(j)` și `y(i)`
- Astfel, prin apelul `contour(x,y,f,20)`, `f` va fi formată din cote situate deasupra planului xOy cu spațierea dată de `x` și `y`
- Ultimul parametru din apelul `contour()` precizează faptul că se vor utiliza 20 de niveluri de contur
- Dacă acesta din urmă nu este precizat, Matlab va alege în mod implicit numărul de niveluri de contur de pe grafic

- Putem folosi funcția `contour()` și pentru **reprezentarea grafică a funcțiilor implicite**.
- Spre exemplu, putem reprezenta funcția $f(x,y) = 2x^3 - e^y + \sinh(x), x \in [-5,5], y \in [-5,5]$ folosind următoarele secvențe de cod:

```
x1=-5:0.01:5;y1=-5:0.01:5;
[x,y]=meshgrid(x1,y1);
f=2*x.^3-exp(y)+sinh(x);
contour(x,y,f,30,'r')
xlabel('x');ylabel('y');
```


5
4
3
2
1
0
-1
-2
-3
-4
-5
y
-5
0
5
x

- Alte funcții uzuale folosite în cazul reprezentărilor grafice 3D sunt:

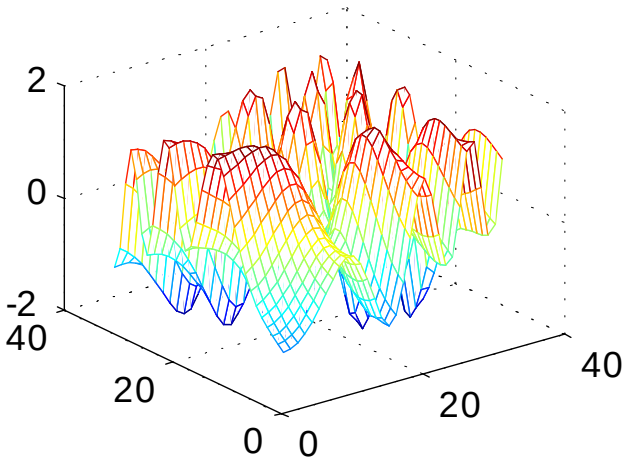
  `mesh()` și `meshc()`

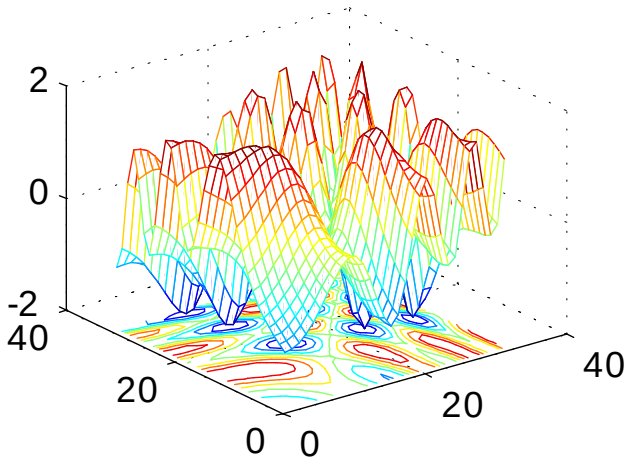
- Acestea acceptă date în aceeași formă ca și `contour()`, dar generează o reprezentare de suprafață de tip "wire-frame" (cadru de sârmă)
- Spre deosebire însă de `mesh()`, funcția `meshc()` va adăuga și un grafic de tip contur dedesubtul suprafeței reprezentate

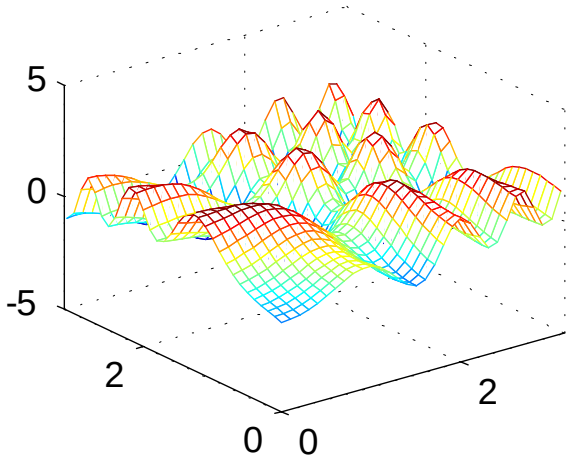
Implementăm în cele ce urmează mai multe variante de reprezentare a suprafeței definită de funcția $f(x,y) = \sin(2x^2 + y) - \cos(x + 2y^2),\ x, y \in [0, \pi]$. Într-un script Matlab, scriem următoarele comenzi:

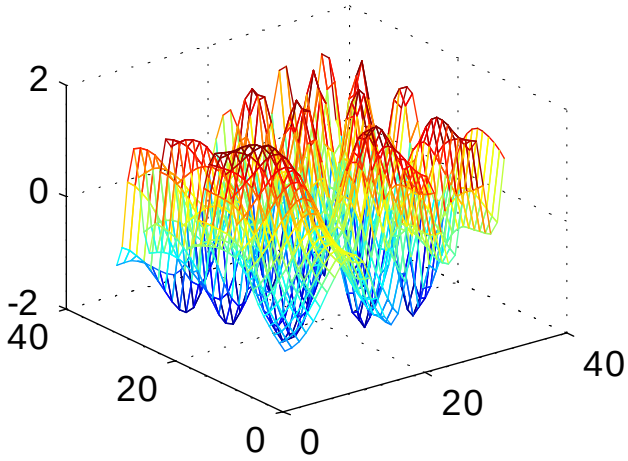
```
x1=0:0.1:pi;y1=0:0.1:pi;
[x,y]=meshgrid(x1,y1);
f=sin(2*x.^2+y)-cos(x+2*y.^2);
subplot(221),mesh(f)
subplot(222),meshc(f)
subplot(223),mesh(x,y,f),axis([0 pi 0 pi -5 5])
subplot(224),mesh(f)
hidden off
```

Vom obține imaginea:









- În cazul primului grafic, în care am folosit `mesh(f)`, întrucât nu sunt precizate valori pentru abscisă și ordonată, sunt utilizați indicii de linie și coloană
- A doua reprezentare prezintă rezultatul apelului funcției `meshc()`
- Pentru cea de-a treia reprezentare, `mesh(x,y,f)` indică faptul că gradațiile de pe axele Ox și Oy corespund valorilor `x` și `y`
- Au fost specificate de asemenea limitele pe axe cu

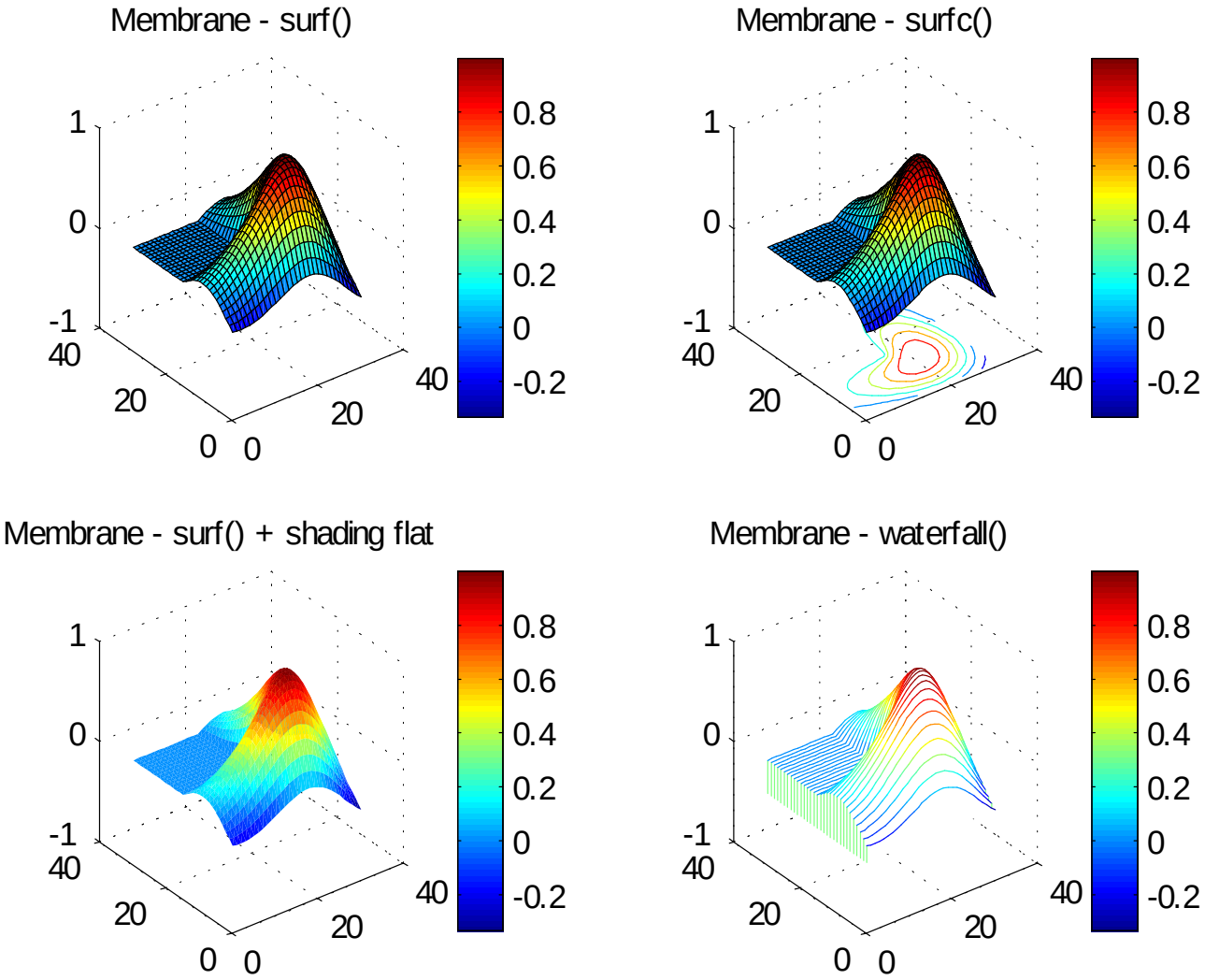
```
axis([0 pi 0 pi -5 5])
```

- Ultima imagine este reprezentată din nou cu ajutorul comenzii `mesh(f)`, urmată de comanda `hidden off`, care nu permite afișarea liniilor ascunse

- Tot pentru reprezentări grafice 3D, putem folosi și funcțiile `surf()`, respectiv `surfc()`
- Acestea se deosebesc de funcția `mesh()` prin faptul că generează un grafic de suprafață cu celulele umplute cu culoare. În plus, `surfc()` va adăuga și contururi dedesubtul suprafeței reprezentate
- De asemenea, o altă funcție similară lui `mesh()` este `waterfall()`, dar care nu mai prezintă wire-frame-ul pe direcția coloanelor
- În exemplul următor, vom reprezenta funcția Matlab `membrane`, care returnează funcția proprie a unei membrane sub forma literei L
- Vom folosi diferite specificații pentru reprezentările realizate

Considerăm astfel următoarele secvențe de cod:

```
f=membrane;
subplot(221),surf(f),title('Membrane - surf()'),colorbar
subplot(222),surfc(f),title('Membrane - surfc()'),colorbar
subplot(223),surf(f),shading flat,title('Membrane - surf() +
shading flat')
colorbar
subplot(224),waterfall(f),title('Membrane - waterfall()')
colorbar
```



- Reprezentărilor de mai sus le-a fost adăugată o scară de culori cu ajutorul specificației colorbar
- Funcția shading cu opțiunea flat elimină liniile grilei de pe suprafața reprezentată

- Comenzi pentru reprezentări grafice 3D:

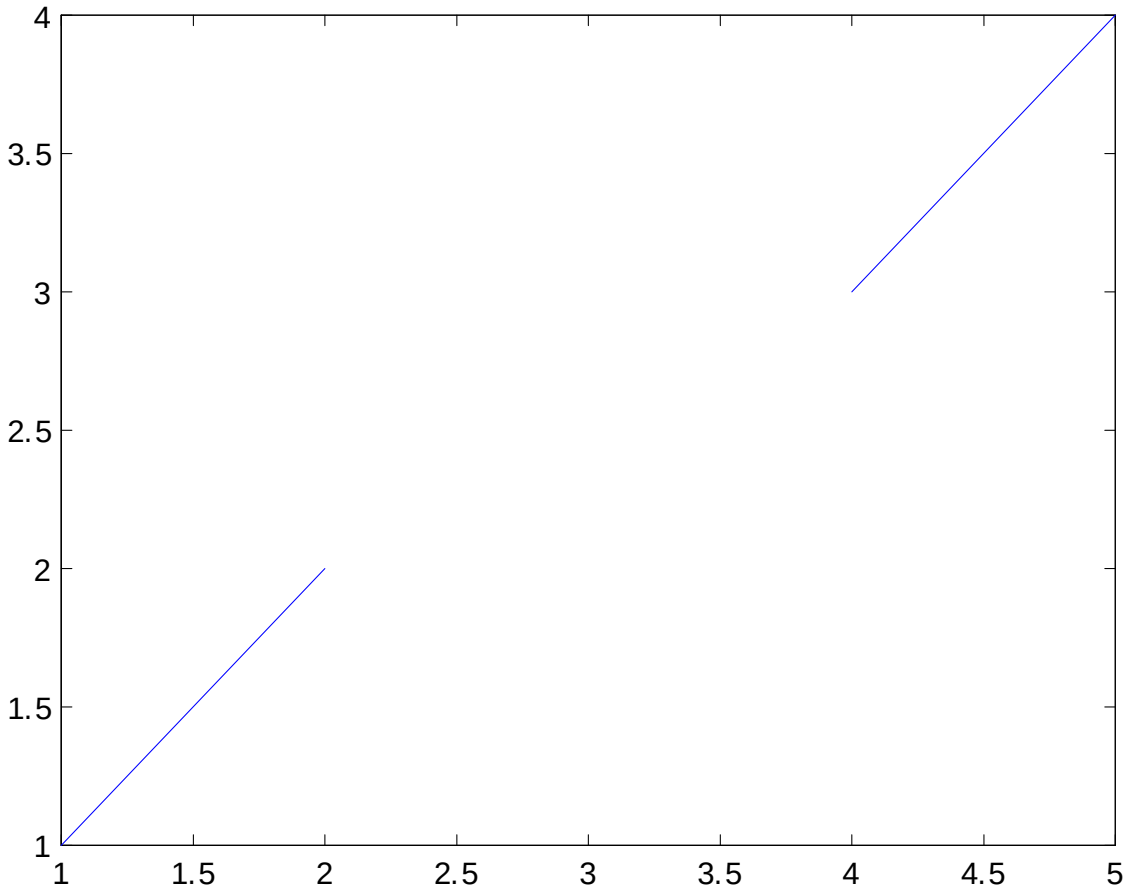
`plot3`	grafic simplu 3D
`contour`	grafic de tip contur
`contourf`	contur plin
`contour3`	contur 3D
`mesh`	reprezentare wire-frame
`meshc`	reprezentare wire-frame plus contururi
`meshz`	reprezentare wire-frame cu cortină
`surf`	grafic suprafață plină
`surfc`	suprafață plină plus contururi
`waterfall`	wire-frame unidirecțional
`bar3`	grafic de tip bară 3D
`bar3h`	bare 3D orizontale
`pie3`	grafice sector 3D
`fill3`	poligon umplut tridimensional
`comet3`	curbă 3D animatedă
`scatter3`	grafic dispersat 3D(nor de puncte 3D)

- Unele funcții au și corespondentul "easy-to-use", al căror nume începe cu ez (pentru funcția `fun` avem corespondentul `ezfun`)

- O trăsătură comună tuturor funcțiilor grafice este aceea că valorile NaN sunt interpretate ca "date lipsă" și nu sunt reprezentate
- De exemplu,

```
plot([1 2 NaN 3 4])
```

desenează două linii disjuncte și nu unește punctele 2 și 3



- În același mod,

```
A=peaks(80); A(28:52,28:52)=NaN; surfc(A)
```

produce graficul `surfc` cu spațiu vid

- Funcția `peaks` din Matlab are expresia

$$\begin{aligned} z &= 3*(1-x).\hat{}2.*exp(-(x.\hat{}2)-(y+1).\hat{}2)... \\ &\quad -10*(x/5-x.\hat{}3-y.\hat{}5).*exp(-x.\hat{}2-y.\hat{}2)... \\ &\quad -1/3*exp(-(x+1).\hat{}2-y.\hat{}2) \end{aligned}$$

și generează o matrice de cote utilă pentru a testa și demonstra facilitățile grafice 3D

- Grafic `surfc` al unei matrice care conține NaN-uri