Grafică în MATLAB

MATLAB are facilităţi grafice puternice şi versatile. Se pot genera grafice şi figuri relativ uşor, iar atributele lor se pot modifica cu uşurinţă. Nu ne propunem să fim exhaustivi, ci dorim doar să introducem cititorul în facilităţile grafice MATLAB care vor fi necesare în continuare. Figurile existente pot fi modificate uşor cu utilitarul Plot Editor. Pentru utilizarea sa a se vedea `help plotedit` şi meniul Tools sau bara din ferestrele figurilor.

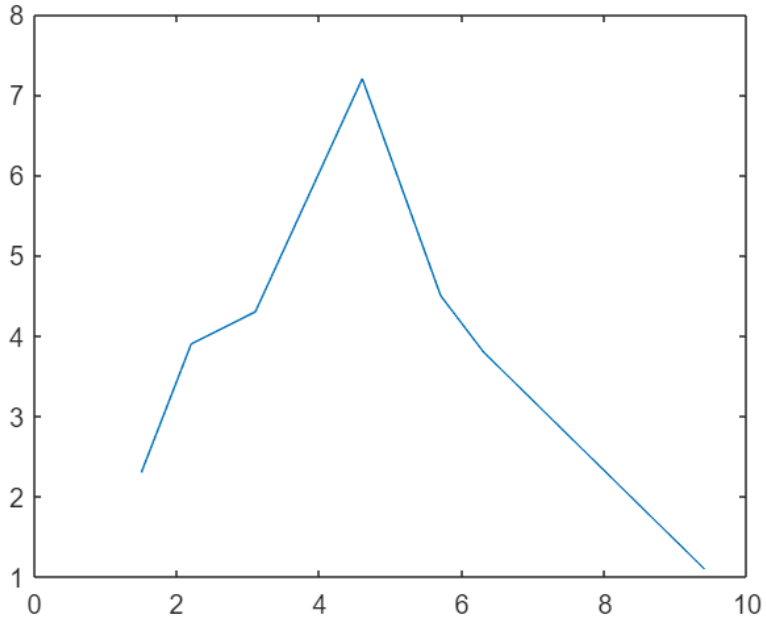
Table of Contents

Grafice 2D

Grafice de bază

Funcţia MATLAB `plot` realizează grafice bidimensionale simple unind punctele vecine.

```
x=[1.5, 2.2, 3.1, 4.6, 5.7, 6.3, 9.4];
y=[2.3, 3.9, 4.3, 7.2, 4.5, 3.8, 1.1];
plot(x,y)
```



MATLAB deschide o fereastră pentru figură (dacă una nu a fost deja deschisă ca rezultat al unei comenzi precedente) în care desenează imaginea. În acest exemplu se utilizează valori implicite ale unor facilităţi cum ar fi domeniul pentru axele x şi y, spaţiile dintre diviziunile de pe axe, culoarea şi tipul liniei.

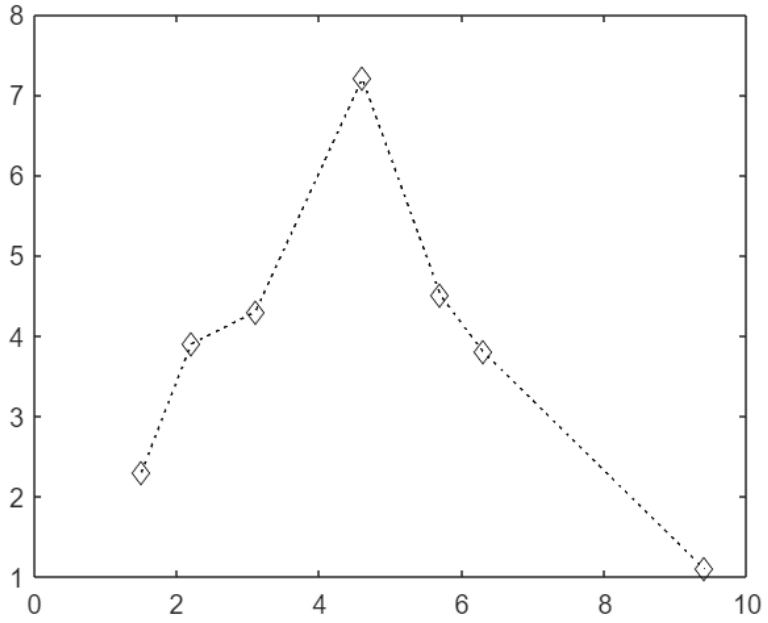
Mai general, în loc de `plot(x,y)`, putem utiliza `plot(x,y,şir)`, unde şir este un şir de caractere ce controlează culoarea, marcajul şi stilul de linie. De exemplu, `plot(x,y,'r*-')` ne spune că în fiecare punct `x(i)`, `y(i)` se va plasa un asterisc roşu, punctele fiind unite cu o linie roşie întreruptă. Cele trei elemente din şir pot apare în orice ordine; de exemplu, `plot(x,y,'ms-')` şi `plot(x,y,'s-m')` sunt echivalente.

`plot(x,y,'+y')` marchează punctele cu un plus galben, fără a le uni cu nici o linie. Tabela de mai jos dă toate opţiunile disponibile.

Culoare		Marcaj		Stil linie	
r	roşu	o	Cerc	-	continuă
g	verde	*	Asterisc	--	întreruptă
b	albastru	.	Punct	:	punctată
c	cyan	+	Plus	-.	linie-punct
m	magenta	x	Ori		
y	galben	s	Pătrat		
k	negru	d	Romb		
w	alb	^	triunghi în sus		
		v	triunghi în jos		
		>	triunghi dreapta		
		<	triunghi stânga		
		p	pentagramă (stea cu 5 colţuri)		
		h	hexagramă (stea cu 6 colţuri)		

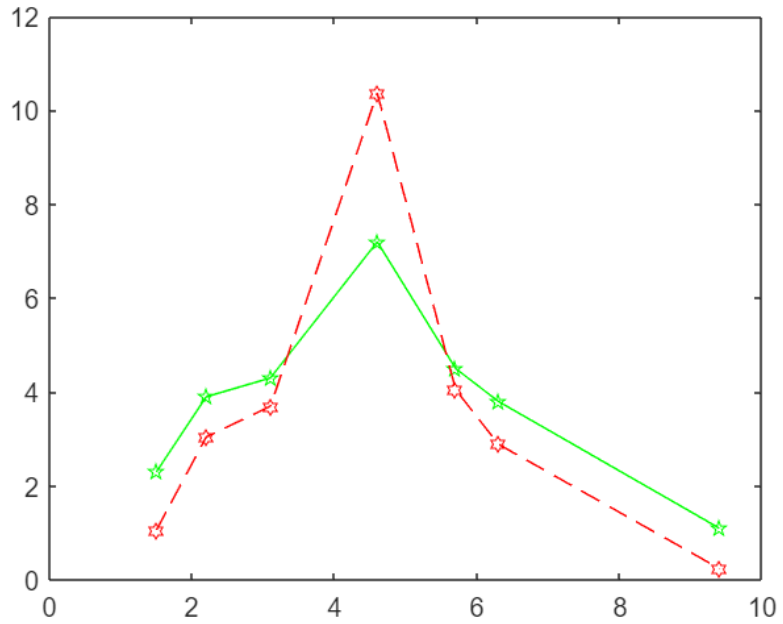
Exemplul de mai jos desenează o linie punctată neagră marcată cu romburi.

```
plot(x,y,'kd:')
```



De notat că `plot` acceptă mai multe seturi de date. De exemplu,

```
b=x; c=y.^2/5;
plot(x,y,'g-p',b,c,'r--h')
```

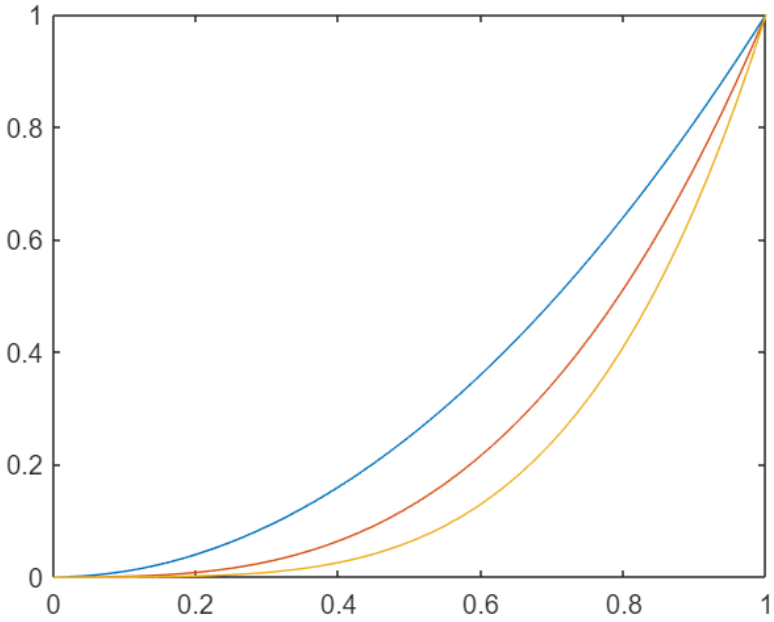


desenează în aceeaşi figură graficele pentru x(i), y(i) şi b(i), c(i) cu linie continuă verde şi respectiv cu linie întreruptă roşie, marcate cu pentagramă şi respectiv hexagramă.

Comanda `plot` acceptă şi argumente matriciale. Dacă `x` este un vector de dimensiune m şi `Y` este o matrice $m \times n$, `plot(x,Y)` suprapune graficele obţinute din `x` şi fiecare coloană a lui `Y`. Similar, dacă `X` şi `Y` sunt matrice de aceeaşi dimensiune, `plot(X,Y)` suprapune graficele obţinute din coloanele corespunzătoare ale lui `X` şi `Y`.

Exemplu: `plot` cu argument matrice

```
x=linspace(0,1,50)';
plot(x,[x.^2,x.^3,x.^4])
```

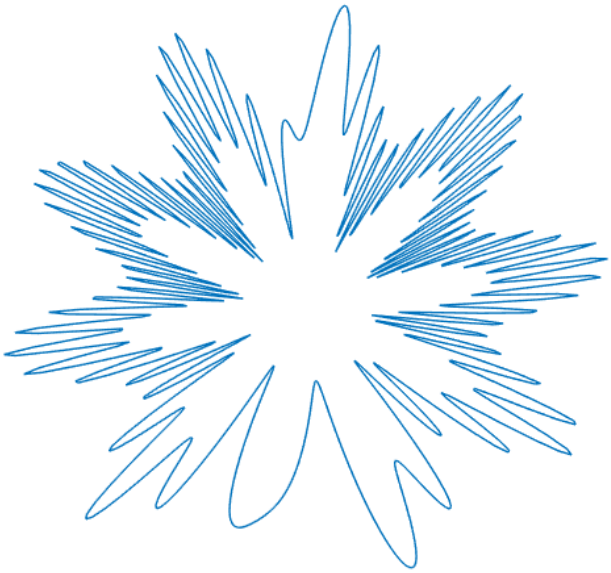


Dacă argumentele lui `plot` nu sunt reale, atunci părţile imaginare sunt în general ignorate. Singura excepţie este atunci când `plot` este apelat cu un singur argument. Dacă `Y` este complex, `plot(Y)` este echivalent cu `plot(real(Y),imag(Y))`. În cazul când `Y` este real, `plot(Y)` desenează graficul obţinut luând pe abscisă indicii punctelor şi pe ordonată Y.

Exemplu de grafic în complex: dorim să reprezentăm funcţia complexă de argument real $f : [-1, 1] \to \mathbb{C}$,

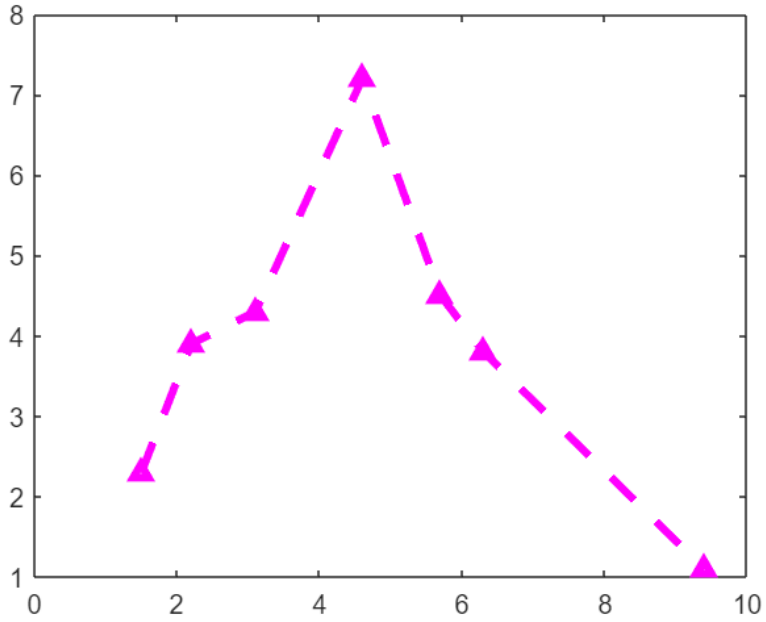
$$f(x) = \left(3 + \sin 10\pi x + \sin 61 e^{0.8 \sin \pi x + 0.7}\right) e^{\pi i\, x}$$

```
x=linspace(-1,1,650);
f=(3+sin(10*pi*x)+sin(61*exp(0.8*sin(pi*x)+0.7))).*exp(pi*1i*x);
plot(f); axis equal; axis off
```



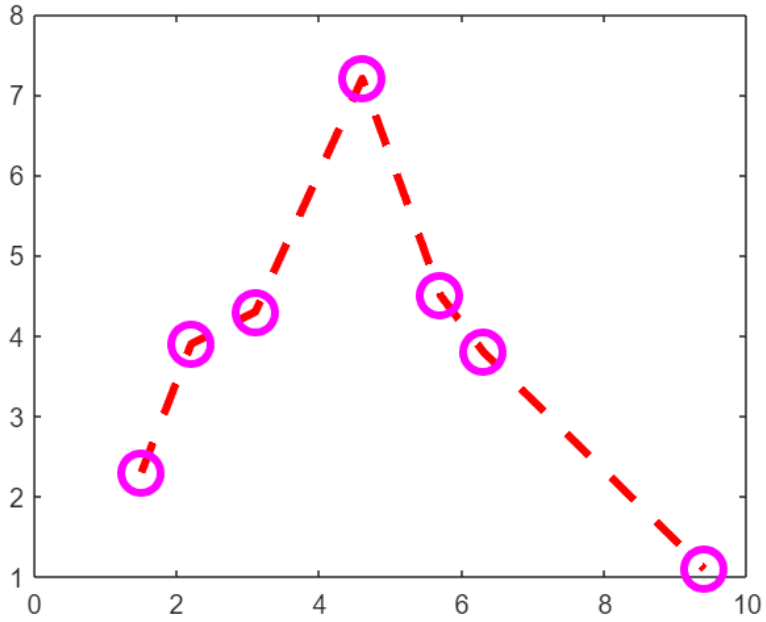
Atributele se pot controla furnizând argumente suplimentare lui `plot`. Proprietăţile `Linewidth` (implicit 0.5 puncte) şi `MarkerSize` (implicit 6 puncte) pot fi specificate în puncte, unde un punct este 1/72 inch.

```
x=[1.5, 2.2, 3.1, 4.6, 5.7, 6.3, 9.4];
y=[2.3, 3.9, 4.3, 7.2, 4.5, 3.8, 1.1];
plot(x,y,'m--^','LineWidth',3,'MarkerSize',5)
```



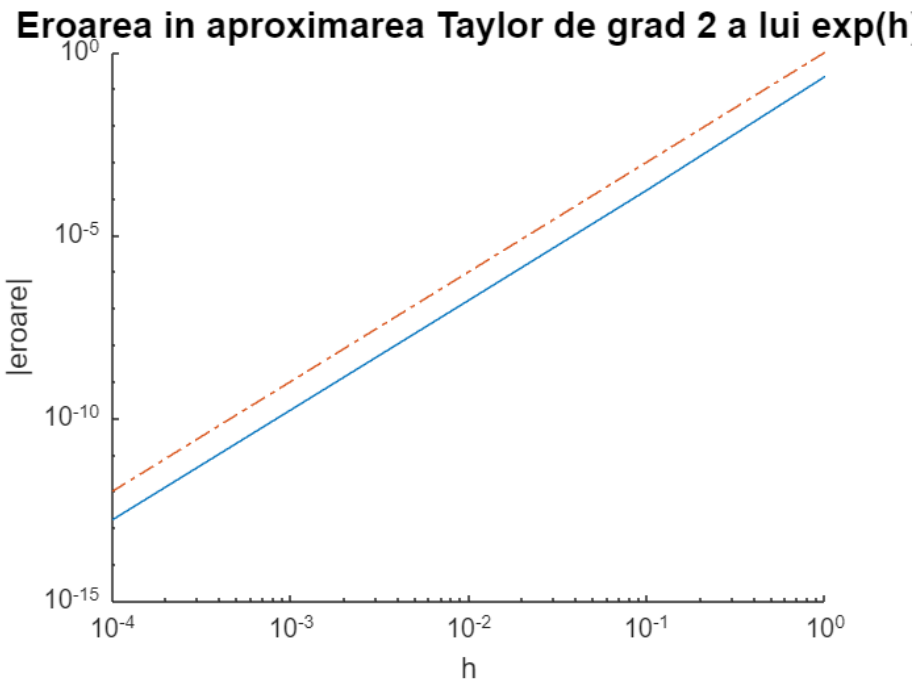
Culoarea laturilor marcajului şi a interiorului marcajului se poate seta pe una din culorile din tabela 2.1 cu proprietăţile `MarkerEdgeColor` şi `MarkerFaceColor`.

```
plot(x,y,'r--o','MarkerEdgeColor','m','LineWidth',3,'MarkerSize',15)
```



Funcţia `loglog`, spre deosebire de `plot`, scalează axele logaritmic. Această facilitate este utilă pentru a reprezenta relaţii de tip putere sub forma unei drepte. În continuare vom reprezenta graficul restului Taylor de ordinul al doilea $\left|1+h+\frac{h^2}{2}-e^h\right|$ pentru $i = 0, 1, \ldots, 4$. Când h este mic, această cantitate se comportă ca un multiplu al lui h^3 şi deci pe o scară log-log valorile vor fi situate pe o dreaptă cu panta 3. Vom verifica aceasta reprezentând restul şi dreapta de referinţă cu panta prevăzută cu linie punctată.

```
h=10.^(0:-1:-4);
taylerr=abs((1+h+h.^2/2)-exp(h));
loglog(h,taylerr,'-',h,h.^3,'-.')
xlabel('h'), ylabel('|eroare|')
title('Eroarea in aproximarea Taylor de grad 2 a lui exp(h)','Fontsize',14)
box off
```



În acest exemplu s-au utilizat comenzile `title`, `xlabel` şi `ylabel`. Aceste funcţii afişează şirul parametru de intrare deasupra imaginii, axei x şi respectiv axei y. Comanda `box off` elimină caseta de pe marginea graficului curent, lăsând doar axele de coordonate. Dacă `loglog` primeşte şi valori nepozitive, MATLAB va da un avertisment şi va afişa doar datele pozitive. Funcţiile înrudite `semilogx` şi `semilogy`, scalează doar una din axe.

Dacă o comandă de afişare este urmată de alta, atunci noua imagine o va înlocui pe cea veche sau se va suprapune peste ea, depinzând de starea `hold` curentă. Comanda `hold on` face ca toate imaginile care urmează să se suprapună peste cea curentă, în timp ce `hold off` ne spune că fiecare imagine nouă o va înlocui pe cea precedentă. Starea implicită corespunde lui `hold off`.

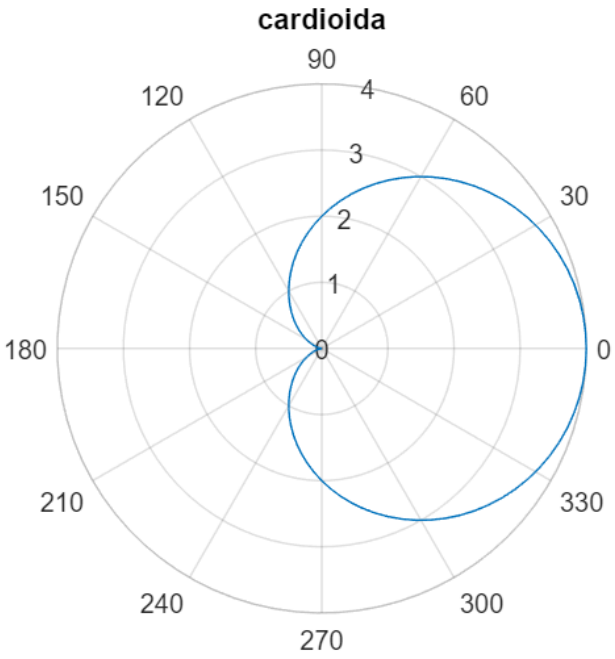
Se pot reprezenta curbe în coordonate polare cu ajutorul comenzii `polar(t,r)`, sau `polarplot(t,r)` unde `t` este unghiul polar, iar `r` este raza polară. Se poate folosi şi un parametru suplimentar `s`, cu aceeaşi semnificaţie ca la `plot`. Graficul unei curbe în coordonate polare, numită cardioidă, şi care are ecuaţia

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

unde a este o constantă reală dată, se obţine cu secvenţa:

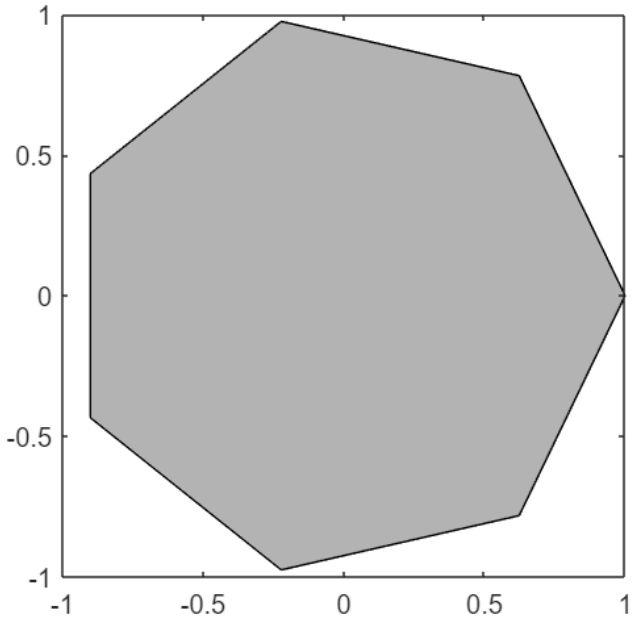
```
%cardioida
t=0:pi/50:2*pi;a=2;
r=a*(1+cos(t));
```

```
polarplot(t,r)
title('cardioida')
```



Funcţia `fill` lucrează la fel ca plot. Comanda `fill(x,y,c)` reprezintă poligonul cu vârfurile `x(i)`, `y(i)` în culoarea `c`. Punctele se iau în ordine şi ultimul se uneşte cu primul. Culoarea `c` se poate da şi sub forma unui triplet RGB, `[r g b]`. Elementele r, g şi b, care trebuie să fie scalari din [0,1], determină nivelul de roşu, verde şi albastru din culoare. Astfel, `fill(x,y,[0 1 0])`umple poligonul cu culoarea verde, iar `fill(x,y,[1 0 1])` cu magenta. Dând proporţii egale de roşu, verde şi albastru se obţin nuanţe de gri care variază de la negru ([0 0 0]) la alb ([1 1 1]). Exemplul următor desenează un heptagon regulat în gri:

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



Comanda `clf` şterge figura curentă, iar `close` o închide. Este posibil să avem mai multe ferestre figuri pe ecran. Cel mai simplu mod de a crea o nouă figură este comanda `figure`. A `n`-a fereastră figură (unde `n` apare în bara de titlu) poate fi făcută figură curentă cu comanda `figure(n)`. Comanda `close all` va închide toate ferestrele figurii. De notat că multe atribute ale unei figuri pot fi modificate interactiv, după afişarea figurii, utilizând meniul Tool al ferestrei sau bara de instrumente (toolbar). În particular, este posibil să se facă zoom pe o regiune particulară cu ajutorul mouse-ului (vezi `help zoom`).

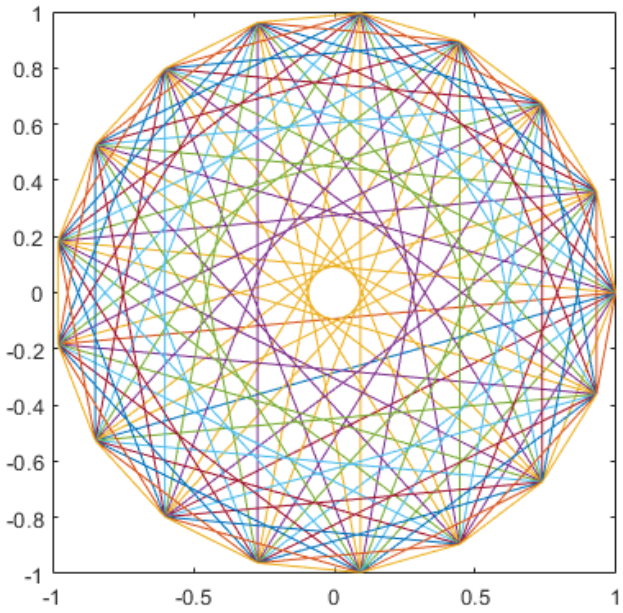
Axe și adnotarea

Diversele aspecte ale unui grafic pot fi controlate cu comanda `axis`. Unele opţiuni se dau în tabela de mai jos.

`axis([xmin xmax ymin ymax])`	Setează limitele axelor x şi y
`axis auto`	Returnează limitele implicite
`axis equal`	Egalează unităţile pe axele de coordonate
`axis off`	Elimină axele
`axis square`	Face caseta axelor pătrată (cubică)
`axis tight`	Setează limitele axelor egale cu limitele datelor
`xlim([xmin xmax])`	Setează limitele pe axa x
`ylim([ymin,ymax])`	Setează limitele pe axa y

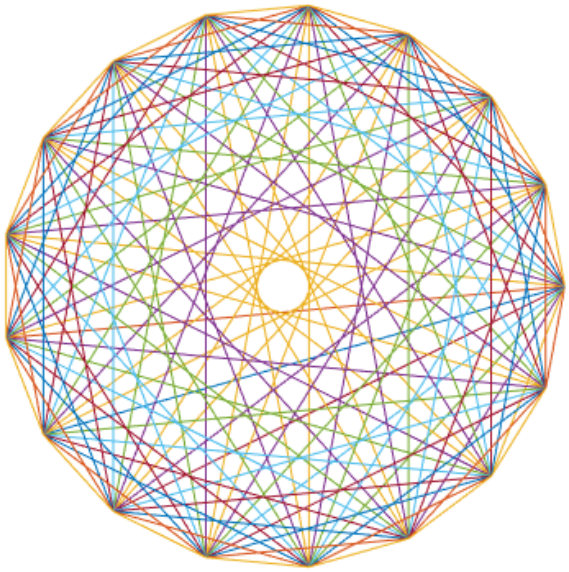
Axele pot fi eliminate cu `axis off`. Raportul dintre unitatea pe x şi cea pe y (aspect ratio) poate fi făcut egal cu unu, astfel ca cercurile să nu pară elipse, cu `axis equal`. Comanda `axis square` face caseta axelor pătrată.

```
plot(fft(eye(17))), axis equal, axis square
```



Deoarece figura este situată în interiorul cercului unitate, axele sunt foarte necesare.

```
plot(fft(eye(17))), axis equal, axis off
```



Comanda `axis([xmin xmax ymin ymax])`setează limitele pentru axa x şi respectiv y. Pentru a reveni la setările implicite, pe care MATLAB le alege automat în funcţie de datele care urmează a fi reprezentate, se utilizează `axis auto`. Dacă se doreşte ca una dintre limite să fie aleasă automat de către MATLAB, ea se ia –inf sau inf; de exemplu, `axis([-1,1,-inf,0]`. Limitele pe axa x sau y se pot seta individual cu cu `xlim([xmin xmax])` şi `ylim([ymin, ymax])`.

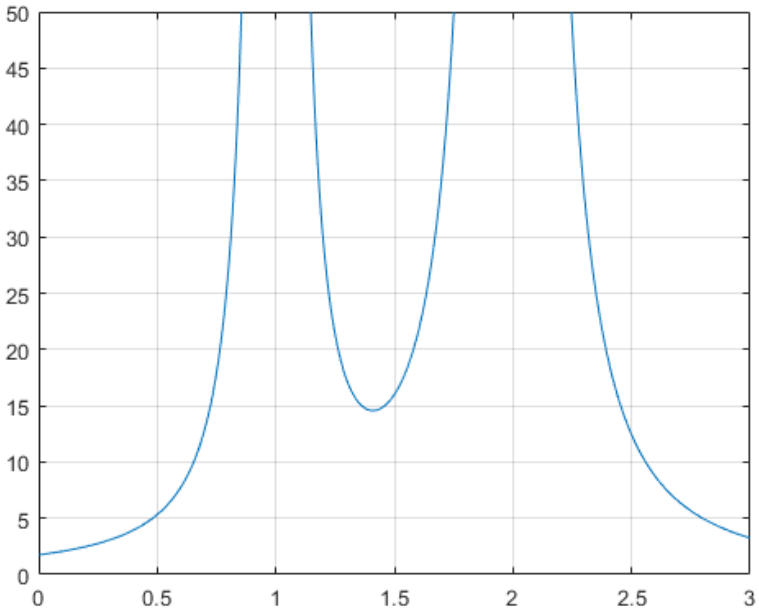
Exemplul următor reprezintă funcţia $f(x) = \frac{1}{(x-1)^2} + \frac{3}{(x-2)^2}$ pe intervalul $[0, 3]$:

```
x=linspace(0,3,500);
plot(x,1./(x-1).^2+3./(x-2).^2);
grid on
```

Comanda `grid on` produce o grilă de linii orizontale şi verticale care pornesc de la diviziunile axelor. Rezultatul se poate vedea în figura de mai sus.

Datorită singularităţilor din $x = 1, 2$ graficul nu dă prea multă informaţie. Totuşi, executând comanda

```
ylim([0,50])
```

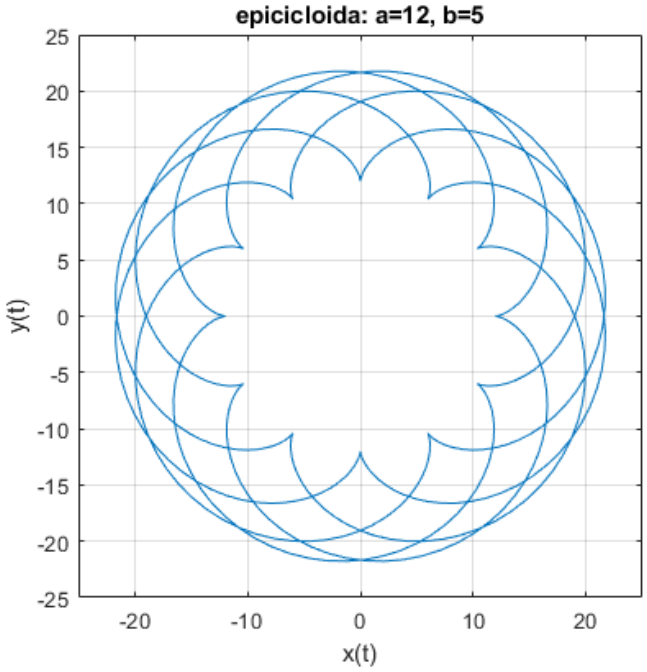


se observă părţile interesante ale graficului.
Exemplul următor reprezintă epicicloida

$$\begin{cases} x(t) = (a+b)\cos t - b\cos\left(\frac{a}{b}+1\right)t \\ y(t) = (a+b)\sin t - b\sin\left(\frac{a}{b}+1\right)t \end{cases}, \quad 0 \le t \le 10\pi$$

pentru $a = 12$ şi $b = 5$.

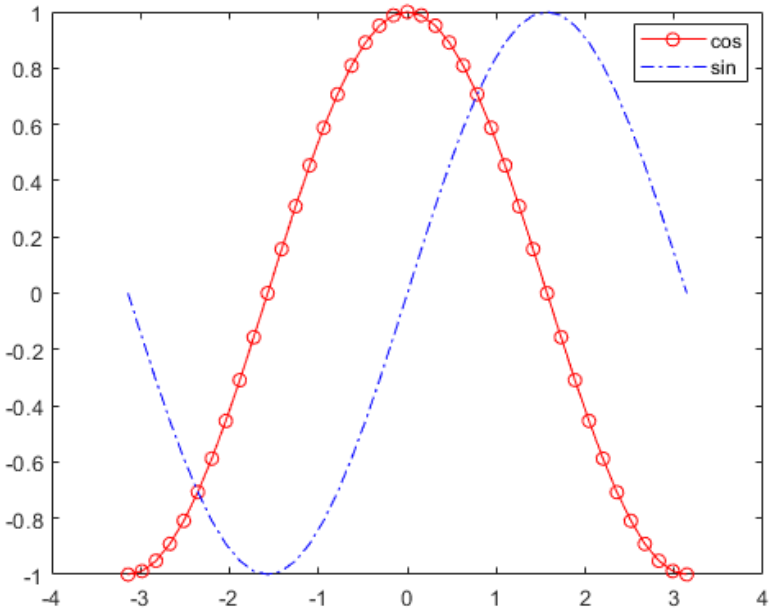
```
a = 12; b=5;
t=0:0.05:10*pi;
x = (a+b)*cos(t)-b*cos((a/b+1)*t);
y = (a+b)*sin(t)-b*sin((a/b+1)*t);
plot(x,y)
axis equal
axis([-25 25 -25 25])
grid on
title('epicicloida: a=12, b=5')
xlabel('x(t)'), ylabel('y(t)')
```



Limitele din `axis` au fost alese astfel ca să rămână un oarecare spaţiu în jurul epicicloidei.

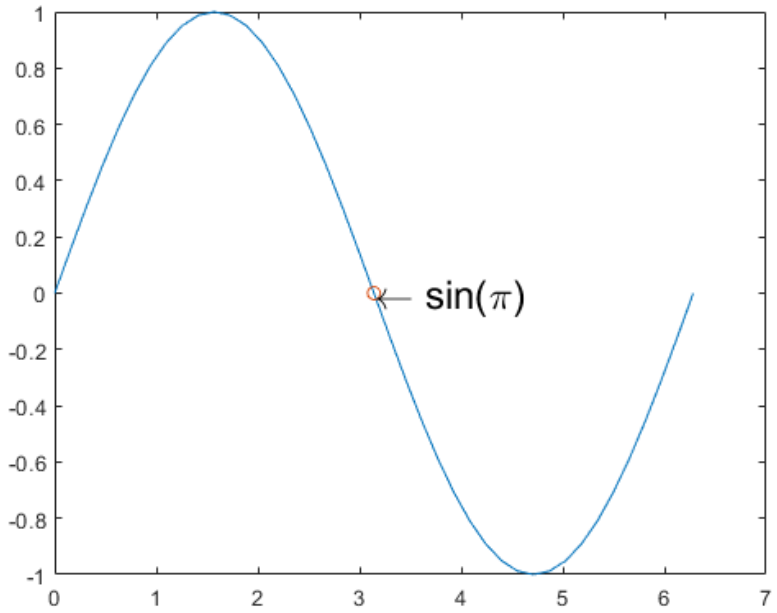
Comanda `legend('string1','string2',...,'stringn',pp)`va ataşa unui grafic o legendă care pune 'stringi' după informaţia culoare/ marcaj/stil pentru graficul corespunzător. Parametrul opţional `pp` indică poziţia legendei (vezi `help legend`). În versiunile mai noi, poziţia se specifică cu `'Location'`, `val`. Exemplul care urmează adaugă o legendă unui grafic al sinusului şi cosinusului.

```
x = -pi:pi/20:pi;
plot(x,cos(x),'-ro',x,sin(x),'-.b')
h = legend('cos','sin','Location','best');
```



Textele se pot include în grafice cu ajutorul comenzii `text(x, y, s)`, unde `x` şi `y` sunt coordonatele textului, iar `s` este un şir de caractere sau o variabilă de tip şir. Începând cu versiunea 5, MATLAB permite introducerea în interiorul parametrului `s`a unor construcţii TeX, de exemplu _ pentru indice, ^ pentru exponent, sau litere greceşti (\alpha, \beta, \gamma, etc.). De asemenea, anumite atribute ale textului, cum ar fi tipul font-ului, dimensiunea şi altele sunt selectabile începând cu versiunea 4. Comenzile

```
plot(0:pi/20:2*pi,sin(0:pi/20:2*pi),pi,0,'o')
text(pi,0,'\leftarrow sin(\pi)','FontSize',18)
```

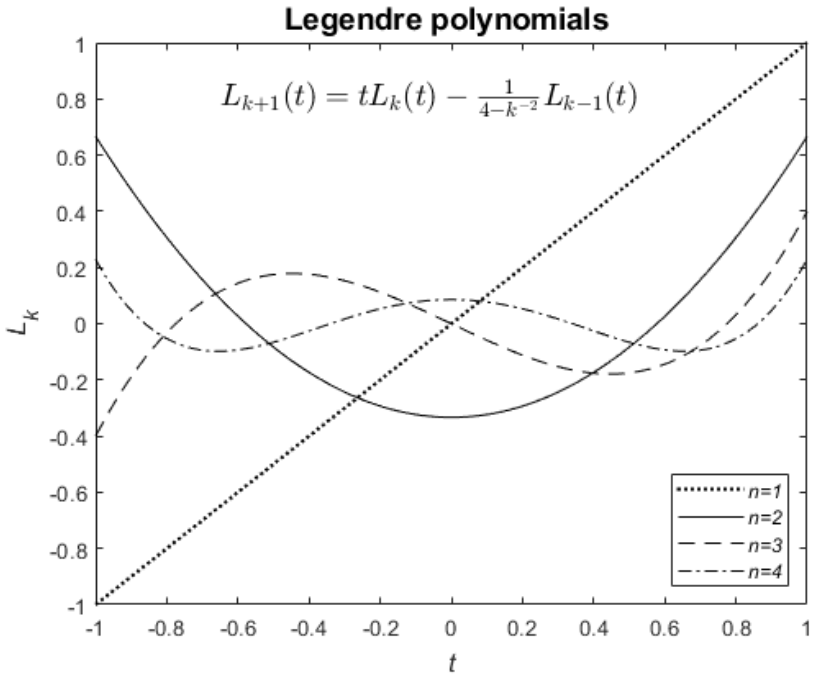


adnotează punctul de coordonate $(\pi, 0)$ cu şirul $\sin(\pi)$. Aceste facilităţi se pot utiliza şi în titluri, legende sau etichete ale axelor, care sunt obiecte de tip text. Începând cu MATLAB 7 primitivele text suportă un subset puternic LATEX. Proprietatea corespunzătoare se numeşte `Interpreter` şi poate avea valorile `TeX`, `LaTeX` sau `none`. Pentru un exemplu de utilizare a macrourilor LaTEX a se vedea script-ul `graphLegendre.m`

```
%graphs for Legendre polynomials
n=4; clf
t=(-1:0.01:1)';
s=[];
ls={':','-','--','-.'};
lw=[1.5,0.5,0.5,0.5];
for k=1:n
    y=vLegendre(t,k);
    s=[s;strcat('\itn=',int2str(k))];
    plot(t,y,'LineStyle',ls{k},'Linewidth',lw(k),'Color','k');
    hold on
end
legend(s,'Location','SouthEast')
xlabel('t','FontSize',12,'FontAngle','italic')
ylabel('L_k','FontSize',12,'FontAngle','italic')
title('Legendre polynomials ','Fontsize',14);
text(-0.65,0.8,'$L_{k+1}(t)=tL_k(t)-\frac{1}{4-k^{-2}}L_{k-1}(t)$',...
```

```
'FontSize',14,'FontAngle','italic','Interpreter','LaTeX')
```

```
graphLegendre
```



Codul pentru funcţia `vLegendre`:

```
function vl=vLegendre(x,n)
%VLEGENDRE - valorile polinomului Legendre
%apel vl=vLegendre(x,n)
%x  - puncte
%n  - grad
%vl - valori

pnm1 = ones(size(x));
if n==0, vl=pnm1; return; end
pn = x;
if n==1, vl=pn; return; end
for k=2:n
    vl=x.*pn-1/(4-(k-1)^(-2)).*pnm1;
    pnm1=pn; pn=vl;
end
```

Mai multe grafice pe aceeași figură

Funcţia `subplot` permite plasarea mai multor imagini pe o grilă în aceeaşi figură.

Format:
`subplot(mnp)`, sau echivalent, `subplot(m,n,p)`.

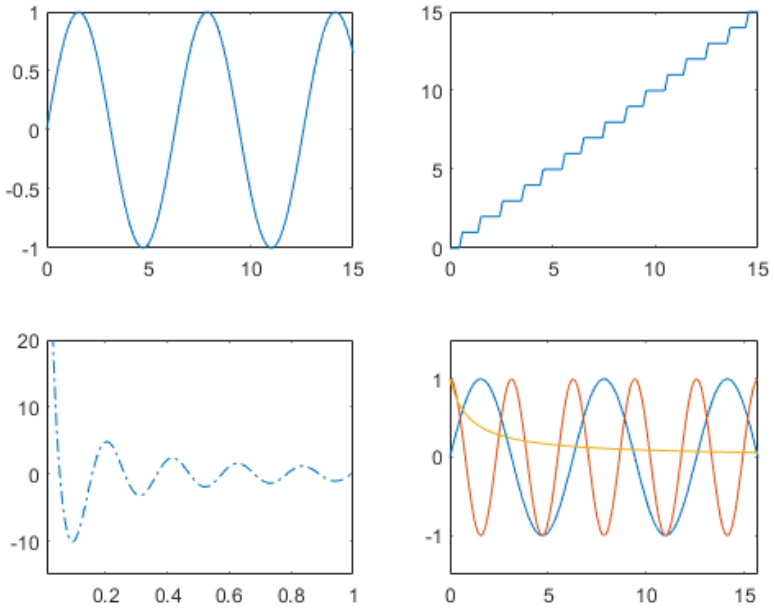
Efect: fereastra figurii se împarte într-un tablou cu `m`×n regiuni, fiecare având propriile ei axe. Comanda de desenare curentă se va aplica celei de-a `p`-a dintre aceste regiuni, unde contorul variază de-a lungul primei linii, apoi de-a lungul celei de-a doua ş.a.m.d. De exemplu, `subplot(425)`împarte fereastra figurii într-o matrice 4×2 de regiuni şi ne spune că toate comenzile de desenare se vor aplica celei de-a cincea regiuni, adică primei regiuni din al treilea rând. Dacă se execută mai târziu `subplot(427)`, atunci poziţia (4,1) devine activă. Vom da în continuare mai multe exemple.

```
subplot(221), fplot(@(x) exp(sqrt(x).*sin(12*x)),[0 2*pi])
subplot(222), fplot(@(x) sin(round(x)),[0,10],'-')
subplot(223), fplot(@(x) cos(30*x)./x,[0.01 1],'-.')
ylim([ -15 20])
subplot(224)
fplot(@(x) [sin(x),cos(2*x),1./(1+x)]);
axis([0 5*pi -1.5  1.5])
```

Pentru sintaxa generală a lui `fplot`, vezi `help fplot` sau `doc fplot`.

Este posibil să se obţină grile neregulate de imagini apelând `subplot` cu şabloane de grile diferite. Exemplu:

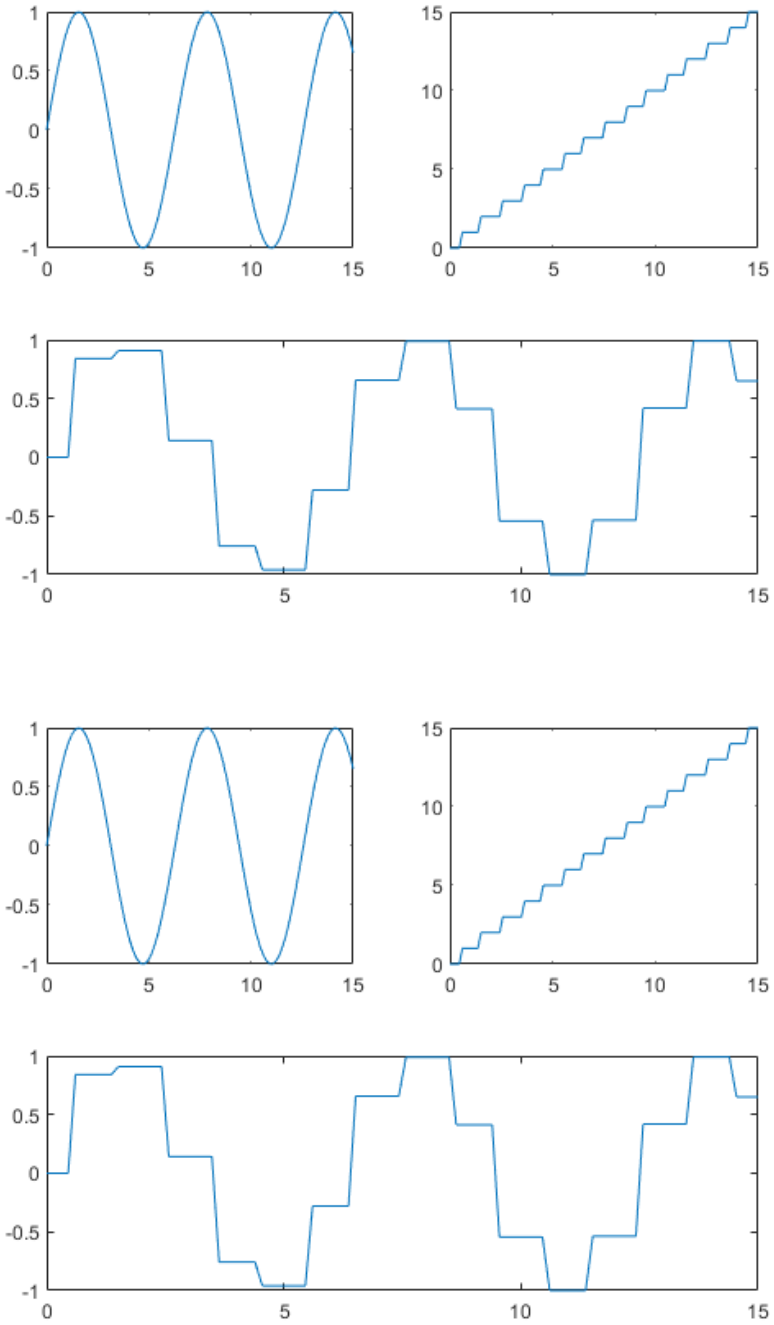
```
x = linspace(0,15,100);
subplot(2,2,1), plot(x,sin(x))
subplot(2,2,2), plot(x,round(x))
```




```
subplot(2,1,2), plot(x,sin(round(x)))
```

Al treilea argument al lui `subplot` poate fi un vector ce specifică mai multe regiuni; ultima linie se poate înlocui cu

```
subplot(2,2,3:4), plot(x,sin(round(x)))
```



Dăm o tabelă rezumativă a funcțiilor grafice bidimensionale

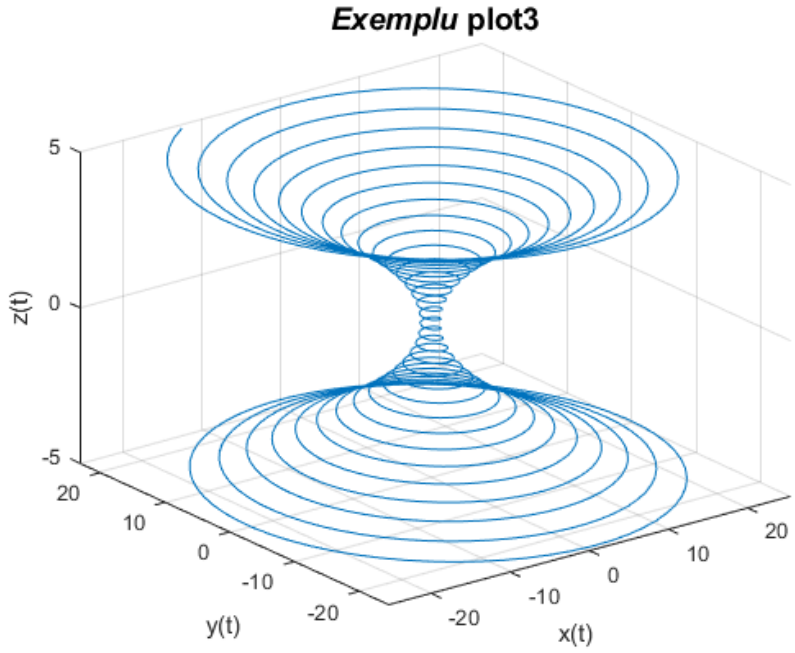
`plot`	grafic ξ-ψ simplu
`loglog`	grafic cu scară logaritmică pe ambele axe
`semilogx`	grafic cu scară logaritmică pe axa ξ
`semilogy`	grafic cu scară logaritmică pe axa ψ
`plotyy`	grafic ξ-ψ cu axe ψ şi l stânga şi la dreapta
`polar`	grafic polar
`fplot`	reprezentare grafică automată a unei funcţii
`ezplot`	versiune uşor de utilizat (easy-to-use) a lui `plot`
`ezpolar`	versiune uşor 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`	Histogramă
`pie`	grafic cu sectoare de cerc
`comet`	grafic ξ-ψ animat
`errorbar`	grafic cu bare de eroare
`quiver`	câmp de vectori bidimensional
`scatter`	Grafic dispersat (nor de puncte)

Grafice 3D

Curbe 3D

Funcţia `plot3` este un analog tridimensional al lui `plot`. Exemplu:

```
clf
t = -5:0.005:5;
x = (1+t.^2).*sin(20*t);
y = (1+t.^2).*cos(20*t);
z=t;
plot3(x,y,z)
grid on
xlabel('x(t)'), ylabel('y(t)'), zlabel('z(t)')
title('{\itExemplu} plot3','FontSize',14)
```



Limitele de axe în spaţiul tridimensional se determină automat, dar ele pot fi schimbate cu

`axis([xmin, xmax, ymin, ymax, zmin, zmax])`

Înafară de `xlim` şi `ylim`, există şi `zlim`, prin care se pot schimba limitele pe axa z.

Suprafețe

O funcţie de două variabile $z = f(x, y)$ se reprezintă cu ajutorul valorilor ei pe o mulţime discretă $z_{i,j} = z(x_i, y_j)$, unde $x = \{x_i : i = 1, \ldots, m\}$ şi $y = \{y_j : j = 1, \ldots, n\}$ sunt puncte de pe axele x şi y luate în ordine crescătoare. Produsul cartezian $x \times y$ ne dă o grilă rectangulară carteziană.

Pentru ilustrare, să considerăm grila definită prin

$$x_{i,j} = x_i = -2 + 0.2(i - 10, \quad 1 \leq i \leq 21$$

$$y_{i,j} = y_j = -2 + 0.2(j - 1) \quad 1 \leq j \leq 21$$

şi valorile funcţiei date de

$$z_{i,j} = x_i \exp(-x_i^2 - y_j^2)$$

Funcţia `meshgrid` este utilă la obţinerea grilelor. Ea are forma

`[X,Y] = meshgrid(x,y)`

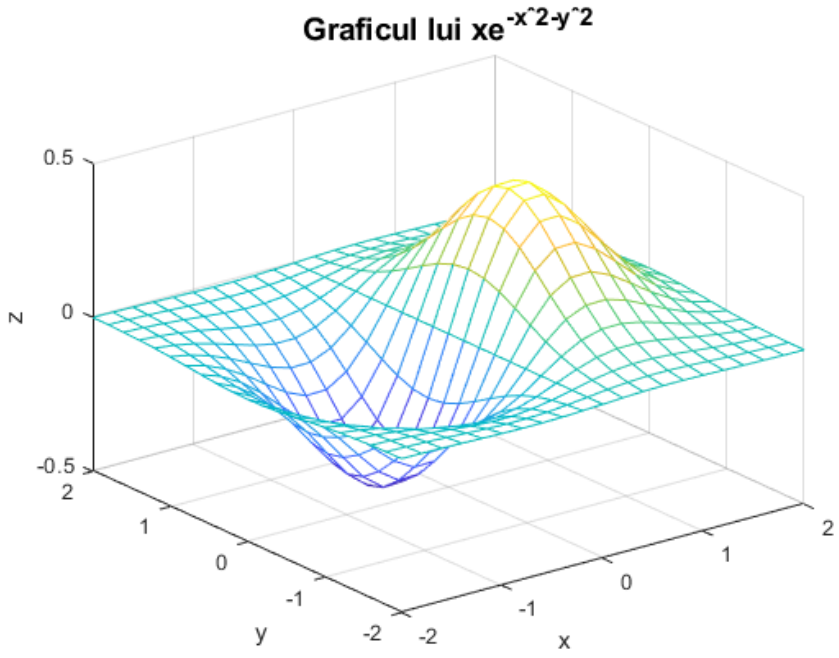
`[X,Y] = meshgrid(x)`
`[X,Y,Z] = meshgrid(x,y,z)`

Prima formă transformă domeniul specificat de vectorii `x` şi `y` în tablourile bidimensionale `X` şi `Y`, care pot fi utilizate la evaluarea funcţiilor de două variabile şi la obţinerea graficelor tridimensionale cu ajutorul funcţiilor `mesh` şi `surf` şi a variantelor lor. Liniile tabloului `X` sunt copii ale lui `x`, iar coloanele lui `Y` sunt copii ale lui `y`. Această regulă trebuie respectată şi dacă elementele lui `z` se obţin prin cicluri `for/end`. Forma a doua are acelaşi efect ca `[X,Y]=meshgrid(x,x)`, iar cea de-a treia este utilizată pentru producerea unor tablori tridimensionale, folosite la calculul valorilor unor funcţii tridimensionale sau la obţinerea unor grafice volumetrice.

Funcţia `mesh(x,y,z)`, unde `x`, `y` şi `z` sunt tablouri bidimensionale de aceeaşi dimensiune, generează o reprezentare de tip cadru de sârmă a suprafeţei parametrice specificate de `x`, `y` şi `z`.

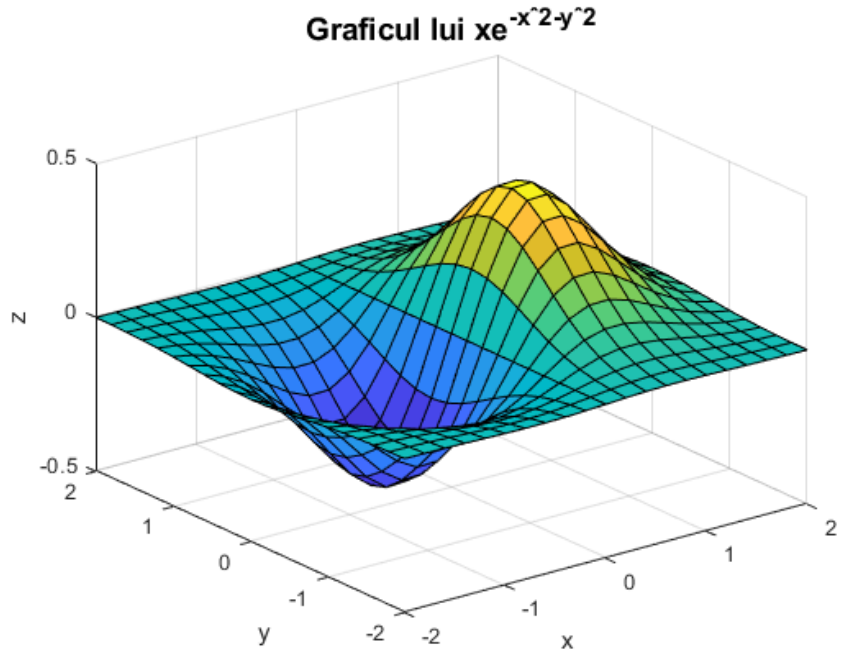
Exemplu:

```
clf
xa=-2:0.2:2;
ya=-2:0.2:2;
[x,y]=meshgrid(xa,ya);
z=x.*exp(-x.^2-y.^2);
mesh(x,y,z)
title('Graficul lui xe^{-x^2-y^2}','FontSize',14)
xlabel('x'),ylabel('y'),zlabel('z')
```



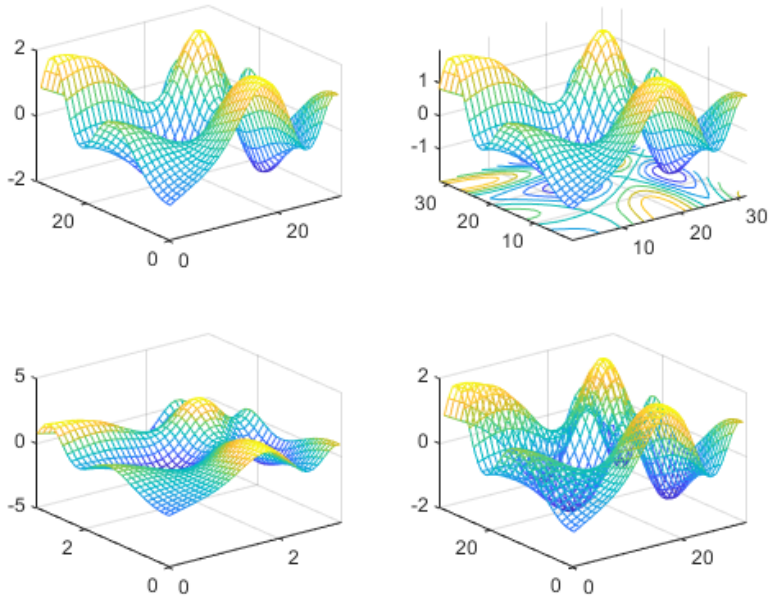
Funcţia `surf` produce grafice asemănătoare cu cele obţinute cu ajutorul funcţiei `mesh`, dar cu deosebirea că celulele suprafeţei sunt colorate. Dacă în exemplul precedent înlocuim `mesh` cu `surf` se obţine graficul:

```
clear,clf
xa=-2:0.2:2;
ya=-2:0.2:2;
[x,y]=meshgrid(xa,ya);
z=x.*exp(-x.^2-y.^2);
surf(x,y,z)
title('Graficul lui xe^{-x^2-y^2}','FontSize',14)
xlabel('x'),ylabel('y'),zlabel('z')
```

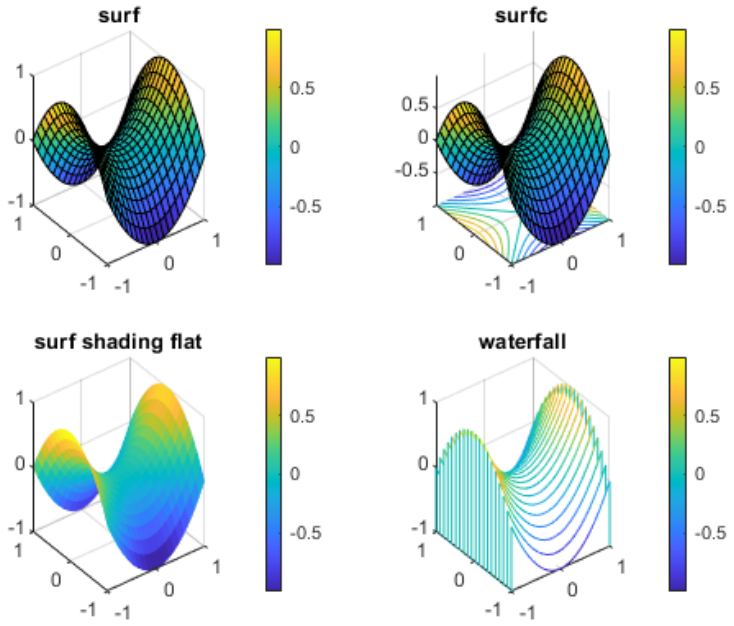


Alte exemple:

```
x = 0:0.1:pi; y=0:0.1:pi;
[X,Y]=meshgrid(x,y);
Z=sin(Y.^2+X)-cos(Y-X.^2);
subplot(221)
mesh(Z)
subplot(222)
meshc(Z)
subplot(223)
mesh(x,y,Z)
axis([0 pi 0 pi -5 5])
subplot(2,2,4)
mesh(Z)
hidden off
```



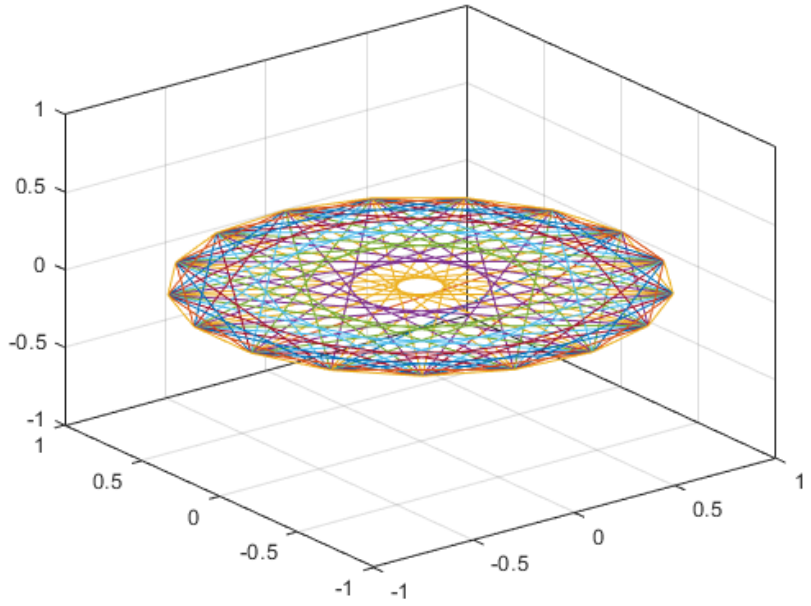

```
clf
[X,Y]=meshgrid(linspace(-1,1,20)); Z=X.^2-Y.^2;
FS = 'FontSize';
subplot(2,2,1), surf(X,Y,Z),
title('\bf{surf}',FS,14), colorbar
subplot(2,2,2), surfc(X,Y,Z),
title('\bf{surfc}',FS,14), colorbar
subplot(2,2,3), surf(X,Y,Z), shading flat
title('\bf{surf} shading flat',FS,14), colorbar
subplot(2,2,4), waterfall(X,Y,Z)
title('\bf{waterfall}',FS,14), colorbar
```



Graficele tridimensionale exemplificate utilizează unghiurile de vizualizare implicite ale MATLAB. Acestea pot fi modificate cu `view`. Apelul `view(a,b)` alege unghiul de rotaţie în sens invers acelor de ceasornic în jurul axei z (azimutul) de `a` grade şi unghiul faţă de planul xOy (elevaţia) de `b` grade. Implicit este `view(-37.5,30)`. Instrumentul `rotate 3D` de pe bara de instrumente a ferestrei figurii permite utilizarea mouse-ului pentru schimbarea unghiurilor de vedere.

Este posibil sa vedem un grafic 2D ca pe unul 3D, utilizând comanda `view` pentru a da unghiurile de vedere, sau mai simplu utilizând `view(3)`. Exemplu

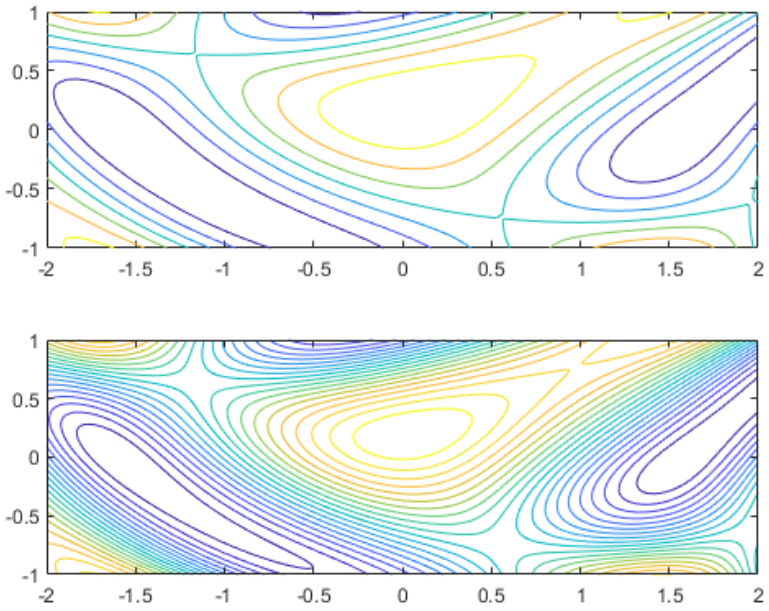
```
clf, plot(fft(eye(17))); view(3); grid
```



Contururi

O facilitate uşor de utilizat de desenare a contururilor este oferită de `fcontour`. Apelul lui `fcontour` în exemplul următor produce contururi pentru funcţia $\sin(3y - x^2 + 1) + \cos(2y^2 - 2x)$ pe domeniul dat de $-2 \leq x \leq 2$ şi $-1 \leq y \leq 1$; rezultatul se poate vedea în jumătatea de sus a figurii.

```
subplot(211)
fcontour(@(x,y) sin(3*y-x.^2+1)+cos(2*y.^2-2*x),...
    [-2,2,-1,1]);
%
x=-2:.01:2; y=-1:0.01:1;
[X,Y] = meshgrid(x,y);
Z =sin(3*Y-X.^2+1)+cos(2*Y.^2-2*X);
subplot(212)
contour(x,y,Z,20)
```



De notat că nivelurile de contur au fost alese automat. Pentru jumătatea de jos a figurii s-a utilizat funcţia `contour`. Întâi se fac iniţializările `x = -2:.01:2` şi `y = -1:.01:1` pentru a obţine puncte mai apropiate în domeniul respectiv. Apoi se execută `[X,Y] = meshgrid(x,y)`, care obţine matricele `X` şi `Y` astfel încât fiecare linie a lui `X` să fie o copie a lui `x` şi fiecare coloană a lui `Y` să fie o copie a vectorului `y`. Matricea `Z` este apoi generată prin operaţii de tip tablou din `X` şi `Y`; `Z(i,j)` memorează valoarea funcţiei corespunzând lui `x(j)`şi `y(i)`. Aceasta este forma cerută de `contour`. Apelul `contour(x,y,Z,20)` spune MATLAB să privească `Z` ca fiind formată din cote deasupra planului xOy cu spaţierea dată de `x` şi `y`. Ultimul argument de intrare spune că se vor utiliza 20 de niveluri de contur; dacă acest argument este omis, MATLAB va alege automat numărul de niveluri de contur.

Funcţia `contour` se poate utiliza şi la reprezentarea funcţiilor implicite cum ar fi

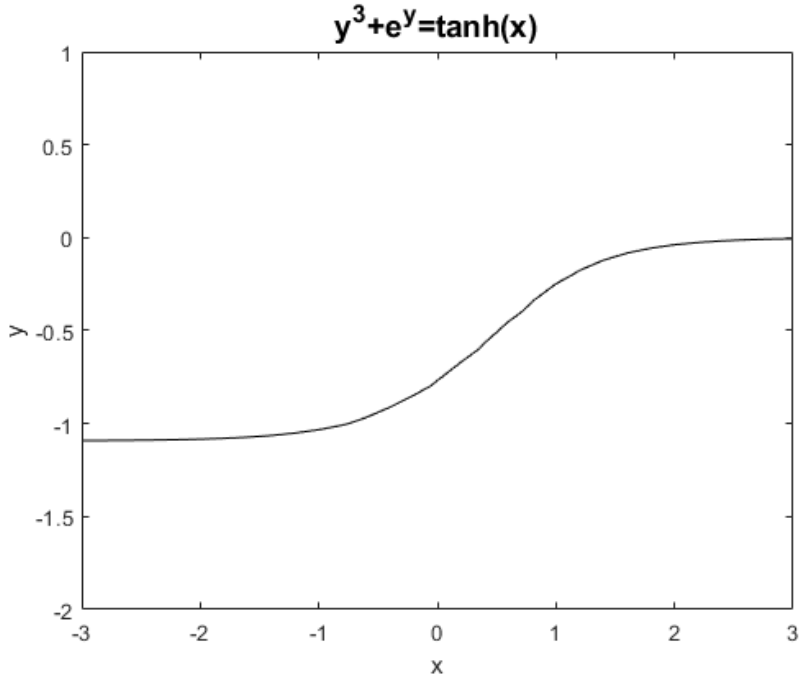
$$y^3 + e^y = \tanh x.$$

Pentru a o reprezenta grafic, rescriem ecuaţia sub forma

$$f(x, y) = y^3 + e^y - \tanh x$$

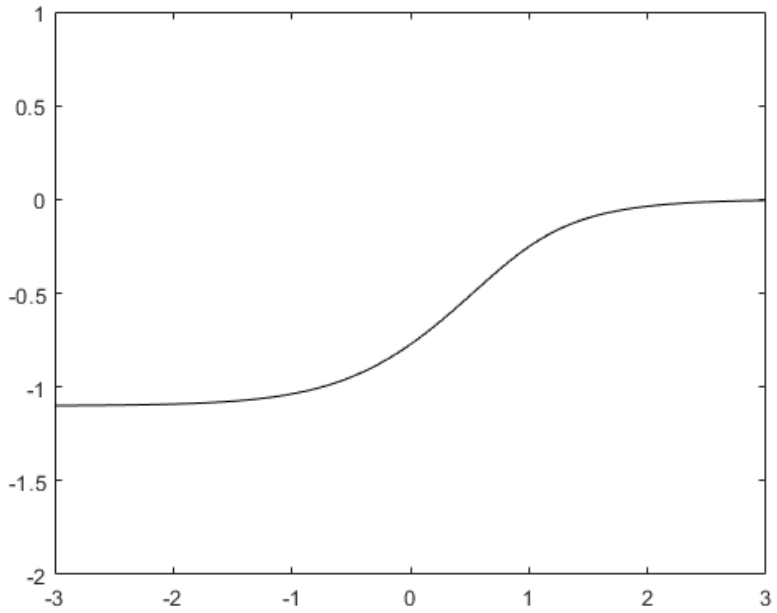
şi desenăm conturul pentru

```
clf; xm=-3:0.2:3; ym=-2:0.2:1;
[x,y]=meshgrid(xm,ym);
f=y.^3+exp(y)-tanh(x);
contour(x,y,f,[0,0],'k-')
xlabel('x'); ylabel('y');
title('y^3+e^y=tanh(x)','FontSize',14)
```



În versiunile mai noi avem funcția `fimplicit`. Graficul anterior se poate obține cu

```
fimplicit(@(x,y) y.^3+exp(y)-tanh(x),[-3,3,-2,1],...
```

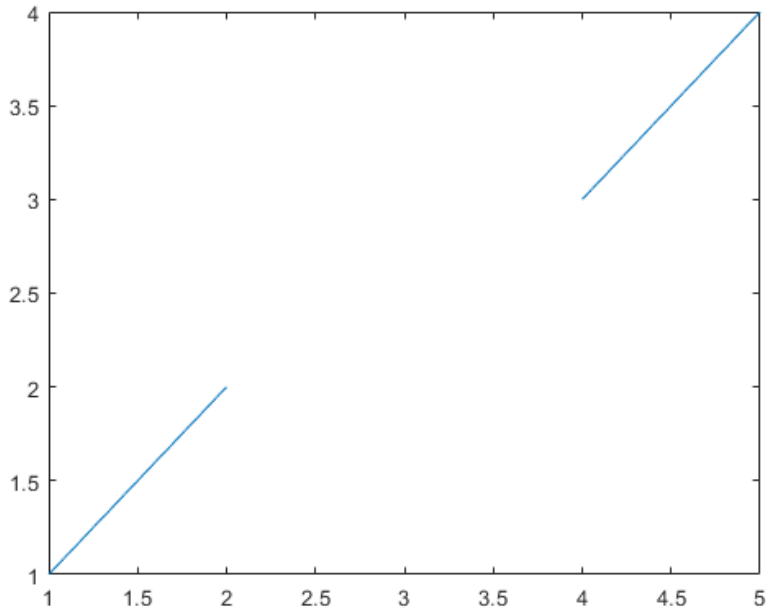



```
    'MeshDensity',121,'Color','k')
```

NaN în funcții grafice

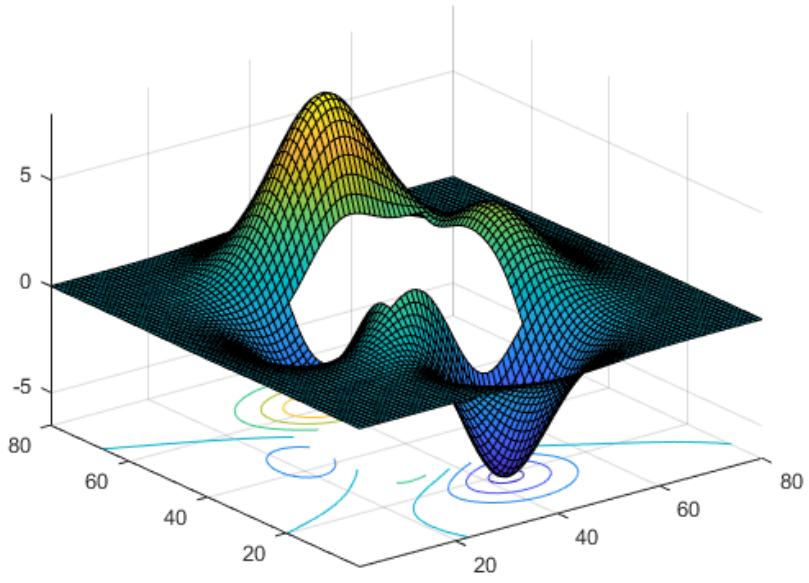
O trăsătură comună tuturor funcţiilor grafice este aceea că valorile NaN sunt interpretate ca „date lipsă” şi nu sunt reprezentate. De exemplu,

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



desenează două linii disjuncte şi nu uneşte punctele 2 şi 3, în timp ce

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



produce graficul `surfc` cu gaură din figură. (Funcţia `peaks` din MATLAB are expresia

```
z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) ...
- 10*(x/5 - x.^3 - y.^5).*exp(-x.^2-y.^2) ...
- 1/3*exp(-(x+1).^2 - y.^2)
```

şi generează o matrice de cote utilă pentru a testa şi demonstra facilităţile grafice 3D.)

Salvarea și imprimarea graficelor

Comanda `print` permite listarea unui grafic la imprimantă sau salvarea lui pe disc într-un format grafic sau sub formă de fişier M. Formatul ei este:

```
print -dperiferic -optiuni numefisier
```

Ea are mai multe opţiuni, care pot fi vizualizate cu `help print`. Dintre tipurile de periferice admise amintim:

- dps - Postscript pentru imprimante alb-negru;
- dpsc - Postscript pentru imprimante color;
- dps2 - Postscript nivelul 2 pentru imprimante alb-negru;
- dpsc2 - PostScript nivelul 2 pentru imprimante color;
- deps - Encapsulated PostScript pentru imprimante alb-negru;
- depsc - Encapsulated PostScript pentru imprimante color;
- deps2 - Encapsulated PostScript nivelul 2 pentru imprimante alb-negru;
- depsc2 - Encapsulated PostScript nivelul 2 pentru imprimante color;

- djpeg - <nn> - imagine JPEG la nivelul de calitate nn (implicit nn=75).

Dacă imprimanta dumneavoastră este setată corespunzător, comanda `print` va trimite conţinutul figurii curente spre ea. Comanda

```
print -deps2 myfig.eps
```

crează un fişier Postscript încapsulat alb şi negru, nivelul 2, numit `myfig.eps`, care poate fi listat pe o imprimantă PostScript sau inclus într-un document. Acest fişier poate fi încorporat într-un document LATEX, aşa cum se schiţează mai jos:

```
\documentclass{article}
\usepackage[dvips]{graphics}
...
\begin{document}
...
\begin{figure}
\begin{center}
\includegraphics[width=8cm]{myfig.eps}
\end{center}
\caption{...}
\end{figure}
...
\end{document}
```

Comanda `print` se poate utiliza şi în formă funcţională . Pentru a ilustra utilitatea formei funcţionale, exemplul următor generează o secvenţă de cinci figuri şi le salvează în fişierele `fig1.eps`, ..., `fig5.eps`:

```
x = linspace(0,2*pi,50);
for i=1:5
plot(x,sin(i*x))
print(-deps2',['fig',int2str(i),'.eps'])
end
```

Al doilea argument al comenzii `print` este format prin concatenare, utilizând funcţia `int2str`, care converteşte un întreg în şir. Astfel, de exemplu, pentru `i=1`, instrucţiunea `print` este echivalentă cu `print('-deps2', 'fig1.eps')`.

Comanda `saveas` salvează o figură ^ıntr-un fişier care apoi poate fi încărcat de către MATLAB. De exemplu,

```
saveas(gcf,'myfig','fig')
```

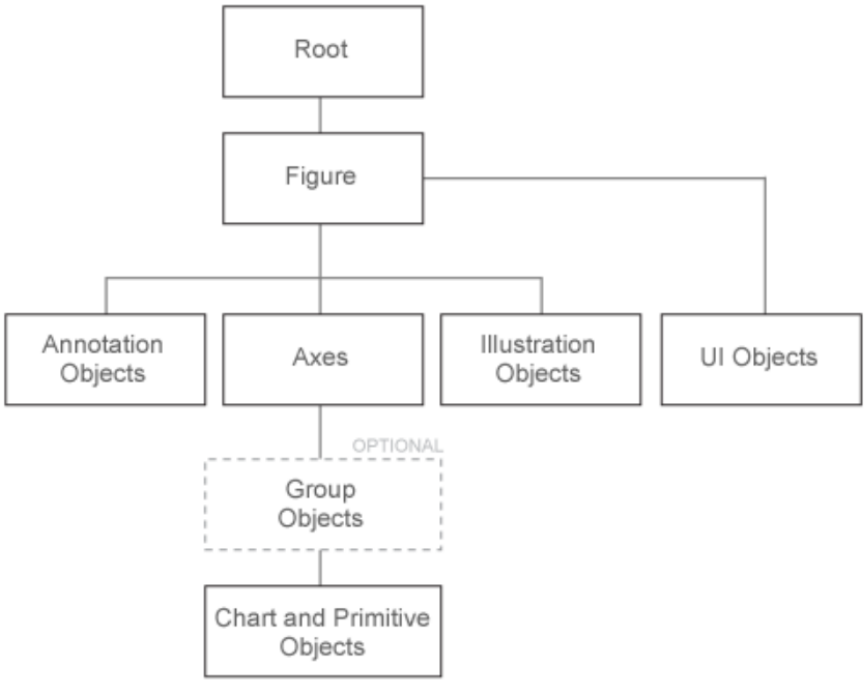
salvează figura curentă în format binar FIG, care poate fi încărcat în MATLAB cu comanda `open('myfig.fig')`. Se pot salva şi imprima figuri din meniul `File` al ferestrei figurii.

Figurile se pot salva şi din ferestrele grafice, dând click pe **File->Save as**

Handles și proprietăți

Ierarhia grafică

Obiectele grafice sunt organizate într-o ierarhie, așă cum se arată în figura de mai jos



Natura ierarhică a obiectelor grafice reflectă incluziunile obiectelor unul fața de altul. Fiecare obiect joacă un rol specific în ierarhie.

Proprietăți și handles

Orice obiect redat are un identificator (handle). Funcţiile `gcf`, `gca`, şi `gco` returnează handle-uri pe figura, axa sau obiectul activ (de obicei cel mai recent desenat sau pe care s-a dat click). Handle-ul era un număr pana la versiunile 2014; acum este un obiect mai complex. Proprietăţile pot fi accesate şi schimbate la nivel de comandă prin funcţiile `get` şi `set`, sau grafic (vezi Plot Edit Toolbar, Plot Browser, şi Property Editor în meniul View al figurii). Iată un exemplu care dă doar o idee despre ce se poate face:

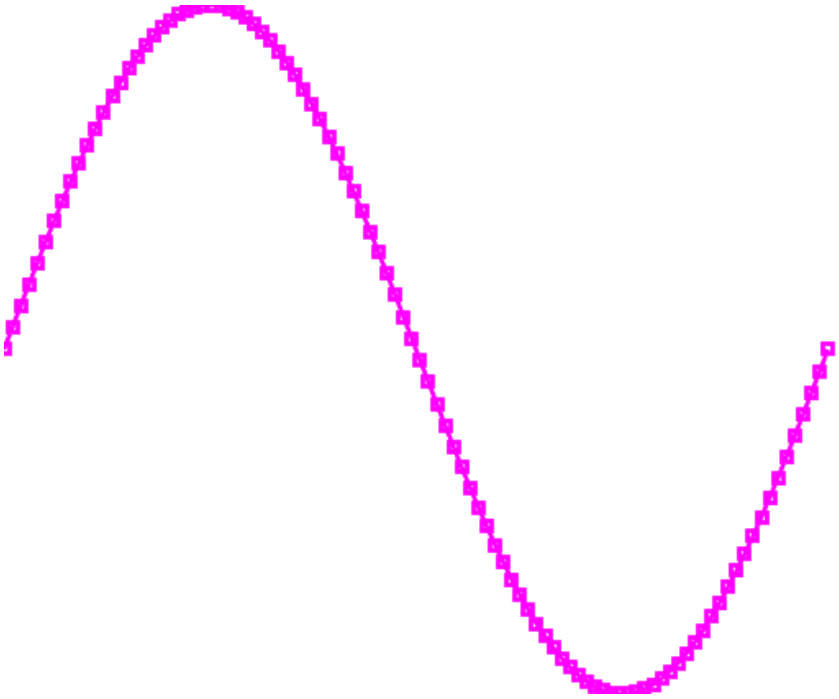
```
t=linspace(0,2*pi,100);
h = plot(t,sin(t))
```

```
h =
  Line with properties:
```

```
            Color: [0 0.4470 0.7410]
        LineStyle: '-'
        LineWidth: 0.5000
           Marker: 'none'
       MarkerSize: 6
  MarkerFaceColor: 'none'
            XData: [1x100 double]
            YData: [1x100 double]
            ZData: [1x0 double]

  Show all properties
```

```
set(h,'color','m','linewidth',2,'marker','s')
set(gca,'pos',[0 0 1 1],'visible','off')
```



În versiunile noi atributele grafice pot fi accesate direct.

```
h = plot(t,sin(t))
```

```
h =
  Line with properties:

              Color: [0 0.4470 0.7410]
          LineStyle: '-'
          LineWidth: 0.5000
             Marker: 'none'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1x100 double]
              YData: [1x100 double]
              ZData: [1x0 double]

  Show all properties
```

```
h.Color='m'; h.LineWidth=2; h.Marker='s';
ax=gca; ax.Position=[0 0 1 1]; ax.Visible='off';
```

Handle-urile permit schimbarea uşoară a unui întreg lot de obiecte la un moment dat. De exemplu, secvenţa de mai jos schimbă toate liniile albastre din figura curentă şi le face lăţimea egală cu 3:

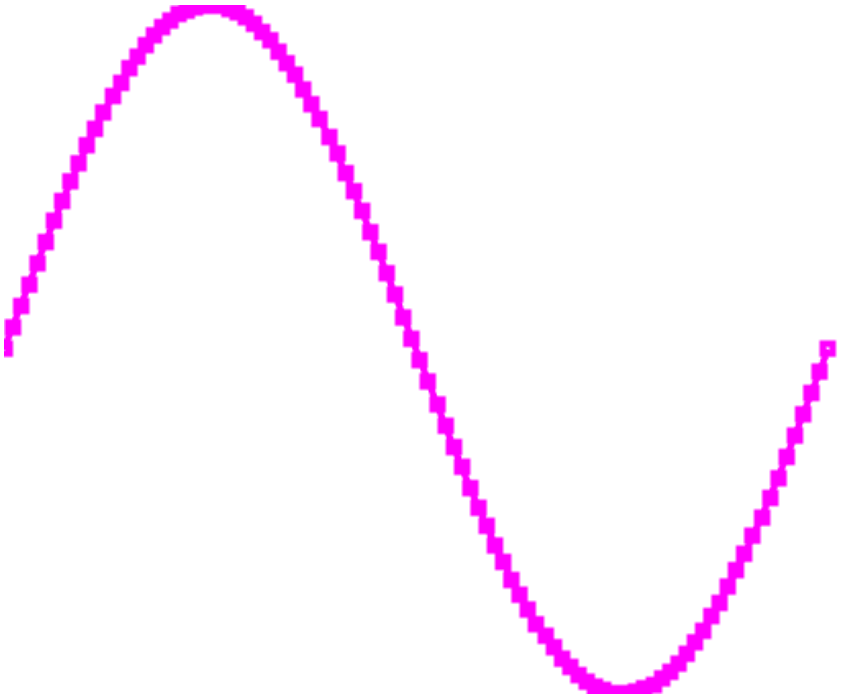
```
h = findobj(gcf,'type','line','color','m')
```

```
h =
  Line with properties:

              Color: [1 0 1]
          LineStyle: '-'
          LineWidth: 2
             Marker: 'square'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [1x100 double]
              YData: [1x100 double]
              ZData: [1x0 double]

  Show all properties
```

```
h.LineWidth=3;
```



Datorită handle-urilor, graficele în MATLAB pot fi create într-o formă de bază şi apoi modificate după dorinţă. Uneori este util să schimbăm valorile implicite ale proprietăţilor utilizate la redarea iniţială a unui obiect. Aceasta se poate face resetând valorile implicite la orice nivel din ierarhia de obiecte grafice din Figura 5.1. De exemplu, pentru a ne asigura că toate obiectele de tip text vor avea dimensiunea fontului de 10, se introduce

```
set(gcf,'defaulttextfontsize',10)
```

Toate figurile sunt considerate copilul obiectului virtual `root`, proprietăţile acestui obiect creând valori globale implicite.

Culoarea

Colorarea liniilor şi a textelor este uşor de înteles. Fiecare obiect are o proprietate Color, căreia i se poate asocia un vector RGB (red, green, blue) cu componentele luând valori între 0 şi 1.

Culorile primare şi secundare apar în tabela 5.3, împreună cu abrevierile lor de o literă.

Culoarea verde nu se vede bine pe fond alb; MATLAB utilizează un verde închis, `[0 0.5 0]`.

Suprafeţele se colorează diferit. Sunt două aspecte diferite, ambele modificabile: colorarea punctelor ce reprezintă date şi colorarea petecelor de suprafaţă şi

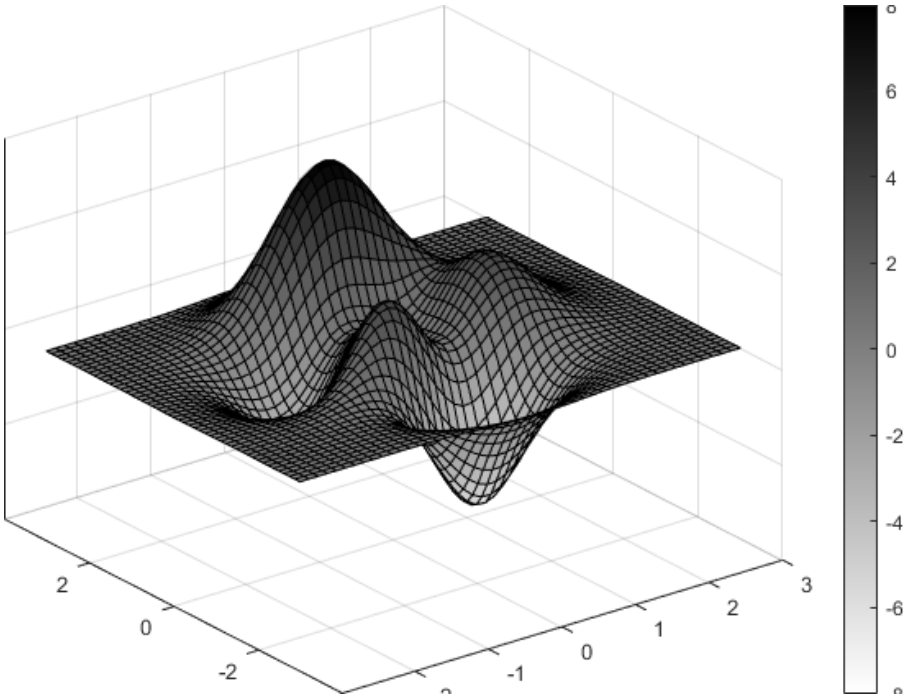
liniilor cadru dintre ele. Culorile punctelor se specifică prin proprietatea `CData` a obiectului suprafaţă. Aceasta se poate specifica prin valorile RGB din fiecare punct, modalitate numită **truecolor model** . Este cea mai buna pentru fotografii și imagini bitmap.

Modalitatea comună implicită când nu se specifică nici o informaţie de culoare este modelul indexat de culori. Acesta presupune o interacţiune între proprietatea `CData` a suprafeţei, proprietatea `CLim` a axelor părinte şi proprietatea `Colormap` a figurii părinte a axelor. Proprietatea `Colormap` a figurii este un tablou $m \times 3$, în care fiecare linie este interpretată ca un triplet RGB. Proprietatea `CLim` este un vector `[a b]` care defineşte o transformare afină de la intervalul $[a, b]$ la intervalul $[1, m]$. Astfel, fiecare valoare din proprietatea `CData` este modificată conform transformării afine, apoi rotunjită la cel mai apropiat întreg din $\{1, \ldots, m\}$, care serveşte ca index de linie în harta de culori pentru a determina culoarea. La fel ca toate proprietăţile obiectelor grafice, cele amintite mai sus pot fi modificate cu comanda `set`. Totuşi există şi alte alternative. Implicit, proprietatea `CData` a suprafeţei este egală cu tabloul de valori ale coordonatelor z (`ZData`) ale suprafeţei, dar poate fi setată apelând funcţiile `surf` şi `mesh` cu un al patrulea argument. Proprietatea `CLim` este setată implicit pentru a include toate valorile datelor, şi se poate modifica ulterior utilizând `caxis`. Harta de culori `Colormap` implicită începând cu MATLAB 2014b este numită `parula`. Orice schimbare a acestor proprietăţi, indiferent cum este facută, are efect imediat asupra culorilor:

```
[X,Y,Z] = peaks; % some built-in data
surf(X,Y,Z)
colorbar % show data->color mapping
caxis
```

```
ans = 1x2
   -6.5466    8.0752
```

```
caxis([-8 8]), colorbar % zero-level symmetry
colormap pink           % change figure colormap
colormap gray           % change figure colormap
colormap(flipud(gray))  % reverse ordering
```



Se poate specifica explicit `CData` pentru o suprafaţă în modelul indexat. O utilizare naturală este prin intermediul unei funcţii de variabilă complexă:

```
[T,R] = meshgrid(2*pi*(0:0.02:1),0:0.05:1);
[X,Y] = pol2cart(T,R);
Z = X + 1i*Y;
W = Z.^2;
surf(X,Y,abs(W),angle(W)/pi) % arg(W) for coloring
axis equal;  colorbar
colormap hsv                 % periodic colormap
```

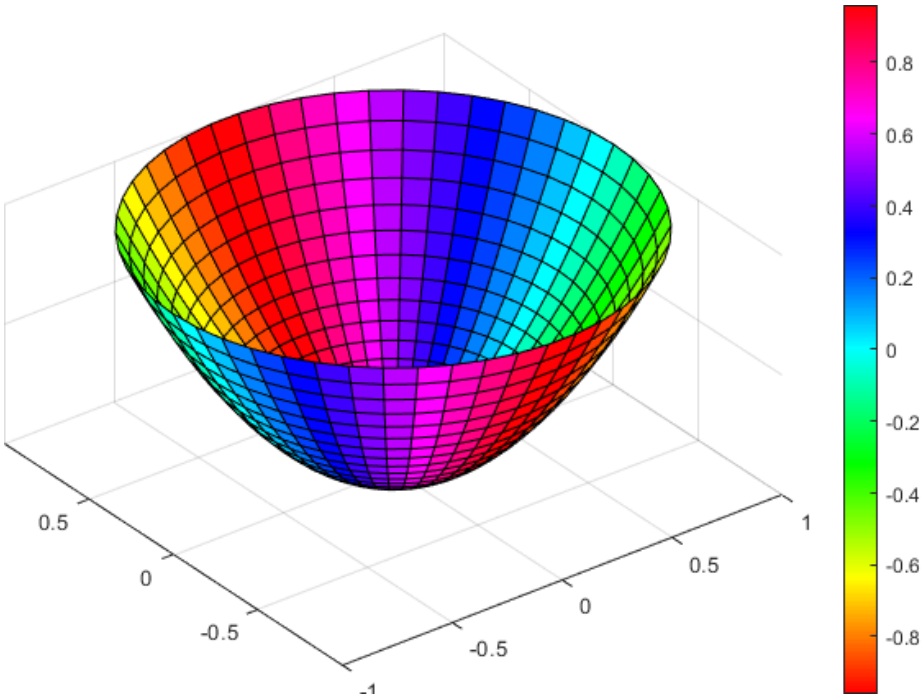

0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0.5
0
-0.5
-1
-0.5
0
0.5
1

Tabela 3. Culori RGB şi prescurtarea lor

Culoarea	Vectorul RGB	Prescurtarea
Negru	[0 0 0]	'k'
Roşu	[1 0 0]	'r'
Verde	[0 1 0]	'g'
Albastru	[0 0 1]	'b'
Galben	[1 1 0]	'y'
Magenta	[1 0 1]	'm'
Cian	[0 1 1]	'c'
Alb	[1 1 1]	'w'

Între punctele grilei, culoarea se determină prin umbrire (**shading**). Aceasta se poate modifica apelând comanda `shading` după crearea suprafeţei.

Tabela 4. Modele de umbrire pentru suprafețe

`shading flat`	Fiecare faţă sau segment are culoare constantă, determinate de un punct de pe frontieră
`shading faceted`	Umbrire constantă pentru feţe şi negru pentru muchii
`Shading interp`	Culoarea fiecărei feţe sau segment se determină prin interpolare liniară

Deşi umbrirea prin interpolare `shading interp` face culorile mai netede şi graficele mai frumoase, este lentă, în particular la imprimare. De fapt, de multe ori este mai rapid să interpolaţi dumneavoastră pe o grilă mai fină şi să imprimaţi cu `shading flat`. A se vedea funcţia `interp2` .

Animație

Se pot utiliza trei metode pentru a crea animații în MATLAB®:

- Actualizarea proprietăților unui obiect grafic și apoi afișarea actualizărilor pe ecran. Tehnica este utilă la animații la care cea mai mare parte a graficului rămâne neschimbată. De exemplu, se pot modifica proprietățile `XData` și `YData` repetat pentru a muta un obiect.
- Aplicarea de transformări obiectelor. Această tehnică este utilă dacă vrem să operăm asupra poziției și orientării unui grup de obiecte. Obiectele se pot grupa ca fiind cop'ii ai unui obiect transformare. Obiectele transformare se crează cu `hgtransform`. Setarea proprietății `Matrix` a unui astfel de obiect actualizează pozițiile tuturor copiilor săi.
- Crearea unui film (movie). Utilă la animații complexe, care nu trebuie să se schimbe în timp real, sau dacă vrem să memorăm animația pentru a o relua sau reda ulterior. Se utilizează funcția `getframe` pentru cadre și `movie` pentru creare și redare.

În unele cazuri, MATLAB nu actualizează ecranul până când nu se termină execuția codului. Utilizați una din variantele comenzii `drawnow` pentru a actualiza ecranul.

Exemplul 1. Actualizare atribute

Acest exemplu ne arată cum putem desena un marker mobil pe o curbă, actualizând proprietațile de poziție (data properties) ale marker-ului. Se deseneaza o sinusoidă și un marker roșu la începutul curbei. Se setează limitele de axe pe manual pentru a evita recalcularea limitelor în timpul animației.

```
x = linspace(0,10,1000);
y = sin(x);
plot(x,y)
hold on
p = plot(x(1),y(1),'o','MarkerFaceColor','red');
hold off
axis manual
```

Mutarea marker-ului se face actualizând proprietățile `XData` și `YData` într-un ciclu. Utilizați comanda `drawnow` sau `drawnow limitrate` pentru actualizarea ecranului. Pentru a seta proprietăți utilizați notația cu punct.

```
for k = 2:length(x)
    p.XData = x(k);
    p.YData = y(k);
    drawnow
end
```

Vezi fișierul `exanima1v2.m`.

Exemplul 2. Transformare

Se crează o stea 3-D cu un grup de obiecte suprafață care au ca părinte un singur obiect transformare. Obiectul transformare se rotește în jurul z în timp ce i se scalează dimensiunea.

Crează axele și ajustează parametri de vedere. Setează limitele axelor pentru a evita selecția automată a limitelor în timpul scalării.

```
ax = axes('XLim',[-1.5 1.5],'YLim',[-1.5 1.5],'ZLim',[-1.5 1.5]);
view(3)
grid on
```

Crează obiectele copil al căror părinte va fi transformat

```
[x,y,z] = cylinder([.2 0]);
h(1) = surface(x,y,z,'FaceColor','red');
h(2) = surface(x,y,-z,'FaceColor','green');
h(3) = surface(z,x,y,'FaceColor','blue');
h(4) = surface(-z,x,y,'FaceColor','cyan');
h(5) = surface(y,z,x,'FaceColor','magenta');
h(6) = surface(y,-z,x,'FaceColor','yellow');
```

Crează obiectul transformare și părintele obiectului suprafață. Inițializează matricea de transformare (I_3).

```
t = hgtransform('Parent',ax);
set(h,'Parent',t)

Rz = eye(4);
Sxy = Rz;
```

Construiește matricea de rotație în jurul lui z și matricea de scalare. Rotește grupul și îl scalează actualizând valoarea lui `r`.

```
for r = 1:.1:2*pi
    % matricea de rotatie în jurul axei Z
    Rz = makehgtform('zrotate',r);
    % matricea de scalare
    Sxy = makehgtform('scale',r/4);
    % Coompune transformările și setează proprietatea Matrix a
    % transformării
    set(t,'Matrix',Rz*Sxy)
    drawnow
end
pause(1)
```

Resetează orientarea originară

```
set(t,'Matrix',eye(4))
```

Vezi fișierul `exanima2v2.m`.

Exemplul 3. Movie

Utilizează `getframe` și `movie`

Crează o serie de grafice într-un ciclu și capturează fiecare grafic într-un cadru. Se asigură că limitele de axe rămân constante setându-le de fiecare dată în interiorul ciclului. Memorează cadrele în `M` .

```
close all
for k = 1:16
plot(fft(eye(k+16)))
axis([-1 1 -1 1])
M(k) = getframe;
end
```

Redă animația de 5 ori utilizând funcția `movie`.

```
figure
movie(M,5)
```

Vezi script-ul `exanima3v2.m`.

Exemplul 4. Utilizare `animatedline`

Crează un obiect inițial animated line. Adaugă 1000 de puncte la curbă într-un ciclu. Dupa adăugarea unui punct actualizează cu `drawnow`.

```
close all
h = animatedline;
axis([0,4*pi,-1,1])

x = linspace(0,4*pi,1000);
y = sin(x);
for k = 1:length(x)
    addpoints(h,x(k),y(k));
    drawnow
end
```

Pentru redare mai rapidă, se poate adauga mai mult de un punct la un pas și se poate utiliza `drawnow limitrate`.

Interoghează punctele liniei.

```
[xdata,ydata] = getpoints(h);
```

Șterge punctele de pe curbă

```
clearpoints(h)
drawnow
```

Vezi script-ul `exanima4v2.m`.

Variantă. Se utilizează un ciclu pentru a adăuga 100000 de puncte la un obiect `animatedline`. Deoarece numărul de puncte este mare și adăugarea unui singur punct în ciclu punct este lentă, se vor adăuga 100 de puncte la un moment dat, pentru o animație mai rapidă.

```
close all
h = animatedline;
axis([0,4*pi,-1,1])

numpoints = 100000;
x = linspace(0,4*pi,numpoints);
y = sin(x);
for k = 1:100:numpoints-99
    xvec = x(k:k+99);
    yvec = y(k:k+99);
    addpoints(h,xvec,yvec)
    drawnow
end
```

Vezi script-ul `exanima4bv2.m`.

Bibliografie

Higham, N. H. (2009). *MATLAB Guide.* Philadelphia: SIAM.

Moler, C. (2004). *Numerical Computing in MATLAB.*Philadelphia: SIAM

Driscoll, T. (2009). *Learning MATLAB.* Philadelphia: SIAM