

MATLAB JEST:

- programem przeznaczonym do wykonywania różnorodnych obliczeń numerycznych,
- uniwersalnym językiem programowania wysokiego poziomu, przeznaczonym do wykonywania obliczeń naukowo-inżynierskich w takich dziedzinach jak: automatyka, elektronika,
- interakcyjnym przyjaznym środowiskiem integrującym metody numeryczne z zakresu: algebry liniowej, macierzy rzadkich, interpolacji, przekształceń Fouriera, równań różniczkowych zwyczajnych, z prostym i efektywnym językiem programowania.

W środowisku tym stawiane problemy i otrzymywane rozwiązania wyrażane są w dobrze znanej notacji matematycznej.

MATLAB umożliwia testowanie algorytmów, modelowanie i symulację, analizę i wizualizację danych, sygnałów oraz wyników obliczeń. Jest również używany do tworzenia specjalistycznych bibliotek oprogramowania (*toolbox*'ów).

MATLAB jest wykorzystywany nie tylko w zastosowaniach związanych z techniką, ale też w biologii, medycynie, ekonomii, meteorologii i innych dziedzinach.

Pakiet MATLAB składa się z kilku elementów:

- język MATLAB (interpreter)
- środowisko pracy w MATLABie
- biblioteka funkcji graficznych (obiektowo zorientowana – wersje > 5.0)
- biblioteka funkcji matematycznych:
 - podstawowe przekształcenia macierzy
(odwracanie, rozkłady macierzy, wartości własne itp.)
 - obliczanie wartości funkcji elementarnych i specjalnych,
 - całkowanie numeryczne,
 - rozwiązywanie układów równań różniczkowych zwyczajnych,
 - podstawowe obliczenia statystyczne

- biblioteki dodatkowe (*toolbox'y*)
nabywane oddzielnie w zależności od potrzeb użytkownika,
ponad 20 wyspecjalizowanych pakietów oprogramowania:
 - *Signal Processing Toolbox* - CPS,
 - *Identification Toolbox* – metody identyfikacji,
 - *Control Toolbox* – projektowanie układów sterowania,
 - *Optimization Toolbox* – metody optymalizacji,
 - *Neural Networks*
 - nakładki – dodatkowe programy napisane w języku MATLAB,
ułatwiające realizacje obliczeń określonego typu,
np. SIMULINK – interaktywny pakiet do modelowania i symulacji ciągłych i
dyskretnych modeli dynamicznych,
umożliwia tworzenie wielopoziomowych systemów (układów
sterowania) w postaci schematów blokowych metodą Drag & Drop,
i wygodną ich symulację
 - interfejs do komunikacji z językiem C++ lub Fortran.
-

ZALETY MATLABa:

- otwartość: - możliwość modyfikacji istniejących i dodawania własnych m-plików,
 - dołączania do m-plików programów zewnętrznych napisanych np. w C++,
 - wykonywania w środowisku C++ programów napisanych w MATLABie,
- konwersja m-funkcji MATLABa na kod C++ w celu przyspieszenia ich wykonywania,
MATLAB jest zoptymalizowany do operacji macierzowych,
inne są ok. 20× szybsze w C++
- eksport / import plików z / do MATLABa (np. z / do Excel'a),
- różne platformy sprzętowe i systemowe.

Praca z MATLABem

- znak zachęty: `>>`
- zestaw programów ilustrujących zastosowanie MATLABa do rozwiązywania
wybranych problemów: `demo`
- inne polecenia pomocy: `whatsnew info ver intro doc help`
- określenie szybkości komputera: `bench`

MATLAB jest interpreterem języka. Praca w jego środowisku przypomina pracę w DOSie – polega na wydawaniu poleceń, które po zatwierdzeniu są wykonywane przez interpreter.

Przykłady poleceń:

- `>> a=3.1415;` - utworzenie zmiennej *a* o wartości 3.1415
(średnik = wynik nie zostanie wyświetlony na ekranie)
od tej chwili w przestrzeni roboczej dostępna jest zmienna *a*
o wartości (aktualnie) 3.1415
- `>> sin(a)` - obliczenie sinusa z *a*
w tym przypadku wartość wyrażenia przypisywana jest do
standardowej zmiennej *ans*
- `>> atan(log10(ln(4*(3+2))))` - wyrażenie: $\arctg(\log(\ln(4(3+2))))$
- błąd: ↑ i poprawić: $\ln \rightarrow \log$

MACIERZE

Podstawowym typem danych w MATLABie jest tablica (macierz). Jej elementami mogą być liczby rzeczywiste lub zespolone, znaki albo inne tablice. Pojedyncza wartość liczbowa jest traktowana jako macierz 1×1. Wektor to macierz składająca się z jednego wiersza lub kolumny.

MATLAB dopuszcza dwa sposoby przechowywania macierzy w pamięci:

- macierze pełne – w pamięci rezerwowane jest miejsce na całą macierz $m \times n$
i pamiętane są wartości wszystkich elementów macierzy

- macierze rzadkie – zapamiętywane są tylko wartości niezerowe oraz ich indeksy
- tablice znakowe, wielowymiarowe, blokowe, struktury

```
przykłady: >> A=[0 0 5; 1 0 3; 0 7 0]
           >> Aspr=sparse(A)
```

Definiowanie macierzy:

- przez wyliczenie elementów:

- w nawiasach kwadratowych podaje się elementy macierzy oddzielając je spacją i / lub przecinkiem, a wiersze oddzielając średnikiem, np.:

```
A=[2 2 2 1; 1 2 3 1]
```

- jeśli nie możemy w wierszu zmieścić wszystkich elementów wiersza macierzy, możemy go zakończyć trzema lub > kropkami i kontynuować w next wierszu, np. : B=[1 2 3 4 ...

5 6 7 8] jest równoznaczne z: B=[1 2 3 4 5 6 7 8]

pominięcie tych trzech kropek to jak wprowadzenie dwóch wierszy:

```
B=[1 2 3 4; 5 6 7 8]
```

- macierze zespolone na dwa sposoby: C=[1+5j 2+6j; 3+7j 4+8j]

```
C=[1 2; 3 4]+i*[5 6; 7 8]
```

- przez wygenerowanie elementów:

polecenie o postaci ogólnej: min:krok:max

generuje wektor wierszowy: [min,min+krok,min+2*krok,...,max]

```
min:max
```

```
[min,min+1,min+2,...,max]
```

przykład:

```
D=[1:10;1:2:20]
```

- przez łączenie mniejszych macierzy:

przykład 1: $A = \begin{bmatrix} 1 & 4 & 1 \\ 2 & 0 & 1 \end{bmatrix}$ $B = \begin{bmatrix} 3 & 1 \\ 4 & 1 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 2 & 2 & 0 & 1 \\ 2 & 4 & 7 & 1 & 0 \end{bmatrix}$ $D = \begin{bmatrix} A & B \\ C & \end{bmatrix}$

```
>> A=[1 4 1; 2 0 1]; B=[3 1; 4 1];
```

```
>> C=[1 2 2 0 1; 2 4 7 1 0];
```

```
>> D=[A B; C]
```

przykład 2: `>> C=[1 2; 3 4]+j*[5 6; 7 8]`

`>> D=[C imag(C); C-i*[3 6; 5 8] real(C)]`

- łączenie powyższych technik, np.: `>> A=[1 2 3; 4 5 6];`

`>> B=[A,[1;2]; 1:4]`

- wczytanie z pliku,

- użycie funkcji macierzowych:

- podstawowe macierze: **eye(n)** – jednostkowa $n \times n$

eye(m,n) lub **eye([m n])**

ones(n) – wszystkie elementy 1

ones(m,n) lub **ones([m n])**

zeros(n) lub **zeros(m,n)** lub **zeros([m n])**

- liczby pseudolosowe: **rand(n)** lub **rand(n,m)** lub **rand([n m])**

- rozkład jednostajny $\langle 0,1 \rangle$

randn(n) lub **randn(n,m)** lub **rand([n m])**

- rozkład normalny (śr.: 0 war.: 1)

- ciągi liczb: **linspace(x1,x2)** - wektor wierszowy 100 liczb

równomiernie rozmieszczonych między x_1 a x_2

linspace(x1,x2,N) - N liczb

np.: **linspace(0,20,5)** = [0 5 10 15 20]

logspace(x1,x2) - 50 liczb logarytmicznie równo rozmieszczonych między 10^{x_1} a 10^{x_2}

logspace(x1,x2,N) - np.: **logspace(0,3,4)** = [1 10 100 1000]

meshgrid(x,y) - wektory do wykresów

Dostęp do elementów macierzy:

- >> **A=[1 2 3 4 5 6; 0 9 8 7 6 5; 1 1 0 0 2 2]**

- >> **A(2,4)**

- **A(i, :)** - *i*-ty wiersz macierzy A

- **A(: , j)** - *j*-ta kolumna macierzy A

- **A(: , j:k)** - kolumny: *j*-ta, *j*+1-sza, ... , *k*-ta kolumna macierzy A

w wyniku to będą odpowiednio 1, 2, *k-j* -ta kolumna

- **A(:)** - wszystkie elementy w kolumnie

- **A(j:k)** - j.w. od elementu nr *j* do *k*

przykłady:

A(1:2,1:2)

A(2,1:6) = A(2, :)

A([1 3], :) – wiersze 1 i 3

A(: , [1:3 5]) – kolumny 1 do 3 oraz 5

A([1 3],1:2:5) – macierz z elementów na przecięciu wierszy 1 i 3 z kolumnami 1, 3 i 5

A(: 2:-1:1) – 2-ga potem 1-sza kolumna

A(: , 2:1) – wektor pusty

A(4:8)

A(end, :) – ostatni wiersz macierzy A (MATLAB 5.x)

X=A; X(2, :)=[] – usunięcie 2-go wiersza

Y=A; Y(1:2:8)=[] – usunięcie sekwencji elementów

B=[.1 .2 .3 .4; .5 .6 .7 .8;.9 1 1.1 1.2]

A(: , [2 3])=B(: , 3:4)

A(3,3)=A(1,1)+A(2,1)

Działania na macierzach:

+ - - dodawanie i odejmowanie macierzy i skalarów

***** - mnożenie macierzy (liczba kolumn A = liczbie wierszy B) lub skalarów

Odpowiednie wymiary muszą się zgadzać, do macierzy dowolnego wymiaru można dodawać / odejmować skalar, mnożyć przez skalar. W razie niezgodności wymiarów pojawi się komunikat o błędzie.

/ - dzielenie prawostronne: $B/A=B*\text{inv}(A)$ $B/A=(A'\backslash B')$

.* - mnożenie tablicowe (mnożenie odpowiadających sobie elementów)

- `./` - dzielenie tablicowe (prawostronne)
- `.\` - dzielenie tablicowe (lewostronne)
- `^` - potęgowanie macierzy - dla macierzy kwadratowych
 - wykładnik całkowity > 1 : A^b
- `.^` - potęgowanie tablicowe (podniesienie każdego elementu do danej potęgi)
- `'` - transpozycja macierzy

operator kropka: `.*` itp. powoduje zmianę operacji macierzowej na tablicową,
wykonuje daną operację na odpowiadających sobie elementach

Podobny efekt można uzyskać stosując pętlę **for**, wykonanie tej operacji wymaga wtedy wielokrotnego odczytu każdego polecenia wewnątrz pętli i jest znacznie wolniejsze od polecenia z kropką.

przykłady: `A=[1 2;3 4]; B=[5 6;7 8]`
`A+B` itd.
`A=[1 2 3;4 5 6;7 8 9]; B=[1;2;3];`
`A*B`
`B*A`
`A=[9 8;19 49]; B=[3 4;1 2]`
`A.*B`
`A=[1 2;3 4]`
`A^3`
`A*A*A`
`A=[4 3;2 1]; B=[1 2;3 4]`
`A.^B`
`A'`

Funkcje macierzowe:

- `[n,m]=size(A)` - rozmiar macierzy
- `rank(A)` - rząd macierzy
- `diag(A)` - macierz diagonalna,
- lub `diag(A,k)` główna przekątna dla $k=0$; nad przekątną dla $k>0$
- `trace(A)` - suma elementów diagonalnych
- `inv(A)` - macierz odwrotna
- `det(A)` - wyznacznik macierzy

lu(A) - rozkład na macierze trójkątne LU

norm(A) - normy wektora i macierzy

poly(A) - współczynniki wielomianu charakterystycznego

roots(A) - wyznacza pierwiastki wielomianu

sin(Z)	cos(Z)	tan(Z)
asin(Z)	acos(Z)	atan(Z)
sinh(Z)	cosh(Z)	tanh(Z)
asinh(Z)	acosh(Z)	atanh(Z)

Y=sqrt(Z) - macierz pierwiastków kwadratowych poszczególnych elementów macierzy Z

Y=exp(Z)

Y=pow2(Z)

Y=log(Z) - logarytm naturalny

Y=log2(Z)

Y=log10(Z)

abs(Z) - macierz modułów elementów macierzy Z

angle(Z) - i argumentów

real(Z) - części rzeczywistych

imag(Z) - i urojonych

conj(Z) - no i sprzężonych

ceil(Z) - zaokrągla elementy macierzy Z w górę

floor(Z) - i w dół

fix(Z) - zaokrągla dodatnie elementy w dół, ujemne w górę (czyli w kierunku zera)
(dla urojonych zaokrąglana część rzeczywista i urojona)

round(Z) - do najbliższej liczby całkowitej

sign(Z) - signum: -1 0 1 (dla zespolonych: Z./abs(Z))

rem(X,Y) - reszta z dzielenia odpowiadających sobie elementów X i Y

gcd(a,b) - naj> wspólny dzielnik liczb a i b

lcm(z,b) - naj< wspólna wielokrotność

kron(a,b) - iloczyn Kroneckera macierzy

Formaty liczb:

format short	5 cyfr	reprezentacja stałoprzecinkowa
format long	15 cyfr	reprezentacja stałoprzecinkowa
format short e	5 cyfr	reprezentacja zmiennoprzecinkowa
format long e	15 cyfr	reprezentacja zmiennoprzecinkowa
format short g	5 cyfr	najlepsza reprezentacja stało- lub zmiennoprzecinkowa
format long g	15 cyfr	j.w.
format rat	postać ułamkowa (przybliża liczby ułamkami małych liczb całkowitych)	
format compact	pomija puste linie przy wyświetlaniu	
format loose	nie pomija	
format +	wypisuje symbol + dla dodatnich, - dla ujemnych	
format bank	format walutowy – pełna część całkowita, dwa po przecinku	
format hex	format szesnastkowy	

clear nazwa_funkcji usuwa z przestrzeni roboczej

clear functions

clear all

save zapisuje binarnie wszystkie zmienne w pliku `matlab.mat`

save nazwa_pliku

save nazwa_pliku nazwy_zmiennych

save nazwa_pliku nazwy_zmiennych -ascii -double -tabs

Przykłady

1. Rozwiązać układ równań liniowych: $2x_1 + 5x_2 = 7$
 $3x_1 - x_2 = 2$
 $4x_1 + 6x_2 = 10$

```
>> A=[ 2 5; 3 -1; 4 6]
>> b=[ 7; 2; 10]
>> [x]=A\b
```

2. Rozwiązać układ równań liniowych: $3x_1 + 2x_2 + x_3 = 5$
 $2x_1 + 3x_2 + x_3 = 1$
 $2x_1 + x_2 + 3x_3 = 11$

```
>> ...
>> [x]=A\b
>> [x]=inv(A)*b
```

3. Obliczyć wyrażenie: $w = \frac{0.963^{1.5} - \sqrt[3]{361}}{2.65^{1.6}}$

```
>> w=(0.963^1.5-361^(1/7))/(2.65^1.6)
```

4. Obliczyć wyrażenie: $w = \ln(\sqrt[3]{1.03} + \sqrt[4]{0.98} - 1)$

```
>> w=log((1.03^(1/3)+(0.98)^(1/4)-1))
```

5. Obliczyć wyrażenie: $w = \cos \frac{\pi}{3} + \sin 21^\circ + \arctg^2 0.3$

```
>> w=cos(pi/3)+sin(pi*21/180)+(atan(0.3))^2
```

Grafika

MATLAB oferuje użytkownikowi całą gamę funkcji pozwalających szybko i przy małym nakładzie pracy przedstawić praktycznie dowolne dane matematyczne.

- wykresy jednej zmiennej

Podstawową funkcją służącą do rysowania takich wykresów jest **plot**.

Działanie jej sprowadza się do łączenia odcinkami punktów o podanych współrzędnych, za jej pomocą można wykreślić praktycznie dowolną krzywą.

Zapewnia ona automatyczne skalowanie obu osi.

Kolor linii wybierany jest automatycznie.

Wywołanie tej funkcji powoduje otwarcie okna graficznego, w którym narysowany zostanie wykres. Funkcja **plot** może przyjmować różną ilość argumentów.

```
plot(y)
plot(x,y)
plot(x,y,'typ linii')
plot(x1,y1,'typ linii 1',x2,y2,'typ linii 2',...)
```

```
przykład: y=[-3 -2 2 2 0 7 9 2]; plot(y)
           x=-pi:0.1:pi; z=sin(x); plot(x,z)
```

Rodzaje linii: **help plot**

Inne funkcje rysujące wykresy 2D:	loglog	skale log
	semilogx	skala log na x
	semilogy	skala log na y
	bar, barh	słupkowy (pion,poziom)
	bar3,bar3h	słupkowy 3D
	hist	histogram
	stairs	schodkowy
	rose	histogram kołowy
	polar	kołowy
	errorbar	wykres błędów
	area	wypełnia pole wykresu
	stem	wykres dyskretny

Parametry tych funkcji są identyczne jak **plot**

przykłady: `x=-pi:0.1:pi; polar([x;x]',[sin(3.*x);cos(3.*x)]', '-')`

```
%piłka na schodach
x=1:0.005:10;
r=rem(x,floor(x));
y1=5-floor(x);
y2=y1+abs(sin(8*pi*r)).*exp(-5*r);
comet(x,y2);

%próba przybliżenia kosinusa funkcją kwadratową
% wykres błędu 20%
x=-pi/2:0.1:pi/2;
y1=cos(x); % funkcja
y2=1-x.^2; % przybliżenie
ee=0.2*y1; % dopuszczalny błąd
errorbar(x,y1,ee); % wykres funkcji i błędu
hold on % zablokowanie wymazywania
axis(axis) % zamrożenie skal wykresu
plot(x,y2,'-'); % wykres przybliżenia
hold off

hist(rand(1,1000))
hist(rand(1,1000),50)
hist(randn(1,1000))
hist(randn(1,1000),50)
hist(randn(1,5000),100)
hist(randn(1,1000),[0.1 0.4 0.8 0.9 0.95 0.99])
```

Funkcja **hist** wywołana z jednym argumentem – wektorem, dzieli przedział wyznaczony przez największy i najmniejszy element tego wektora na 10 podprzedziałów, zalicza elementy należące do każdego z tych przedziałów i rysuje histogram.

Drugi argument, jeśli jest skalar, to określa liczbę podprzedziałów, jeśli jest wektorem, to jego elementy wyznaczają granice podprzedziałów.

- wykresy funkcji

Do szybkiego, wygodnego i precyzyjnego rysowania zależności funkcyjnych została stworzona funkcja **fplot**.

```
fplot(f,granice)
fplot(f,granice,n)
fplot(f,granice,n,kąt)
fplot(f,granice,n,kąt,podprzedziały)
[x,y]=fplot(...)
```

Funkcja rysuje wykres funkcji o nazwie określonej przez parametr *f*. Punkty, w których należy obliczyć wartość rysowanej funkcji są dobierane automatycznie

tak, aby wykres był dokładny, uwzględniał dynamikę zmian wartości funkcji, ale by nie wymagał nadmiernej ilości obliczeń.

Parametry:

`f` łańcuch znaków zawierający nazwę funkcji lub wyrażenie

`granice` dwuelementowy wektor opisujący granice przedziału, w jakim rysować

opcjonalne:

`n` minimalna liczba punktów wykresu (domyślnie: 25)

`kąt` kąt w stopniach, między sąsiednimi odcinkami wykresu powyżej
jakiego zwiększana jest liczba punktów próbkowania (10)

`podprzedziały` maksymalna liczba punktów próbkowania, jaką można dodać
w gwałtownie zmieniających się miejscach wykresu

przykłady: w pliku `fff.m`: `function y=fff(x)`

```

    y=sin(3*pi*atan(x));
    [xx,yy]=fplot('fff',[-10 10]);
    x=-10:0.5:10; y=fff(x);
    subplot(2,1,1)
    plot(x,y)
    title('plot')
    subplot(2,1,2)
    plot(xx,yy)
    title('fplot')

```

```

fplot('sin(x*x)/x',[0 4*pi])
fplot('sin(x*x)/x',[0 20*pi])

```

```

x1=[1.5 2.13 2.6 3.09 3.6 4.15 4.52 4.78 ...
    5.2 5.59 6.77 7.74 8.55 9.13 10.58]
ym1=[80 100 120 140 160 180 190 200 212 220 ...
    240 252 260 264 276]
ym2=239.984*atan(0.222*x1);
plot(x1,ym1,'o',x1,ym2,'g-')
axis([0,11.2,0,286])
ylabel('napiecie [V]')
xlabel('pradmagnesowania [A]')
title('Aproxymacja charakterystyki
    biegu jalowego silnika')
opis1='o punkty pomiarowe';
opis2='---- us=239.984*arctg(0.222*im)';
text(3,60,opis1)
text(3,40,opis2)

```

- funkcje opisujące wykresy funkcji

```

xlabel(tekst)
ylabel(tekst)
title(tekst)
text(x,y,tekst)
grid      grid on      grid off

```

- zarządzanie wieloma rysunkami

MATLAB pozwala posługiwać się kilkoma oknami graficznymi jednocześnie. Pierwsze powstaje automatycznie po wywołaniu dowolnej funkcji graficznej. Następne okna można tworzyć wywołując funkcję **figure**.

Jedno z okien jest zawsze oknem aktywnym (domyślnym), wywołanie dowolnej funkcji graficznej będzie wpływało na zawartość tego właśnie okna. Oknem aktywnym jest zwykle to, które zostało utworzone ostatnie. Aby zmienić okno aktywne należy wywołać funkcję **figure** podając jako parametr numer okna, które ma się stać aktywne. Nr okna pojawia się w jego nazwie, a ta – w pasku tytułowym. Okno graficzne można zamknąć funkcją **close** (bez parametrów – zamyka okno aktywne, z nrem lub wektorem nrów okien do zamknięcia).

Czasem wygodnie jest umieścić kilka wykresów obok siebie w jednym oknie. MATLAB pozwala umieścić w jednym rysunku (oknie) więcej niż jeden układ współrzędnych. Posługiwanie się wieloma układami współrzędnych umożliwia funkcja **subplot**.

```

subplot(m,n,p)   m – liczba wykresów, które mają się zmieścić w pionie
                  n – liczba wykresów, które mają się zmieścić w poziomie
                  p – nr wykresu, który ma być narysowany najbliższym plotem
                   wykresy są numerowane od lewej do prawej,
                   wiersze od góry do dołu

```

przykład: %wzmocnienie i przesunięcie fazy układu RC

```

% R=2k C=220nF
f=2.*logspace(1,4); R=1e3; C=220e-9;
out=1./(f.*i*R*C+1);
subplot(2,1,1);
plot(f,abs(out));
subplot(2,1,2);
plot(f,imag(out));

```

hold **hold on** **hold off**

Zwykle wywołanie funkcji **plot** powoduje usunięcie poprzedniego rysunku, co nie pozwala narysować nowego na tle starego. Zapobiega temu **hold on**.

- skalowanie rysunków

Funkcja **plot** automatycznie ustala skalę, w jakiej rysowany jest wykres.

Skalę wykresu można zmienić używając funkcji **axis**.

```
axis([xmin xmax ymin ymax])
axis('auto')
axis('manual')    - wyłącza auto i zamraża osie
axis('ij')        - układ współrzędnych macierzowy (l-g)
axis('xy')        - układ kartezjański
axis('equal')     - okrąg jest okrąg, a nie elipsa
axis('square')    - tyle samo jednostek na obu osiach
axis('off')
axis('on')
```

- wykresy 3D

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

Funkcja **meshgrid** tworzy macierze opisujące położenie węzłów prostokątnej siatki.

przykład: wartości funkcji $f(x,y) = \sin(x)\exp(-x^2-y^2)$
 obliczamy na siatce: $(-\pi+0.2k, -\pi+0.2l)$
 za pomocą konstrukcji: `[X,Y]=meshgrid(-pi:0.2:pi,-pi:0.2:pi)`
 `Z=sin(X).*sin(Y).*exp(-X.^2-Y.^2)`

- `mesh(x,y,z)`
`mesh(x,y,z,c)`
`mesh(z)`
`mesh(z,c)`

Funkcja **mesh** rysuje powierzchnię opisaną przez macierze x,y,z w postaci kolorowej siatki o oczkach wypełnionych kolorem tła. Elementy macierzy c określają kolory obwódek poszczególnych oczek.

przykład: `close all` % usunięcie starych rysunków
`[x,y]=meshgrid((-1:.1:2)*pi,(-1:.2:3)*pi);`
`z=sin(x).*cos(y)+4*exp(-(x-0.5).^2-(y-0.5).^2);`
`mesh(x,y,z)`
`colormap(flipud(gray))` % zmiana kolorów na szare

- **meshc**

Jak **mesh** tyle, że dodaje kontur pod wykresem (wykres poziomicowy).

- meshz

Jak **mesh** tyle, że dodaje zasłony pod wykresem.

- `surf(x,y,z)`
`surf(x,y,z,c)`
`surf(z)`
`surf(z,c)`

Funkcja **surf** rysuje różnokolorową powierzchnię opisaną przez macierze x, y, z . Elementy macierzy c określają kolory wypełnienia oczek.

```
przykład: close all % usunięcie starych rysunków
[x,y]=meshgrid(-25:0.7:25,-25:0.7:25);
r=sqrt(x.^2+y.^2); % promień - odległości od (0,0)
a=atan(x./y);      % kąty odchylenia punktów
                    % od dodatniej półosi x
d=max(max(a))-min(min(a)); % usuń uskoki kąta
z=cos(r-a*2*pi/d)*0.1;
surf(x,y,z)
axis([-25 25 -25 25 -0.2 0.5])
colormap(gray)
```

- **surfc**

Jak **surf** z konturem.

```
przykład: % wir wodny
close all % usunięcie starych rysunków
[x,y]=meshgrid(-10:0.7:10,-10:0.7:10);
r=sqrt(x.^2+y.^2); % promienie - odległości od (0,0)
a=atan(x./y);      % kąty odchylenia punktów
                  % od dodatniej półosi x
d=max(max(a))-min(min(a)); % usuń uskoki kąta
z=cos(r-a*2*pi/d)*0.1+0.02*r;
surfc(x,y,z)
axis([-10 10 -10 10 -0.2 0.5])
colormap(gray)
```

- `surfl(x,y,z)` `surfl(z)`
 `surfl(x,y,z,s)` `surfl(z,s)`
 `surfl(x,y,z,s,k)`

Jak **surf** z uwzględnieniem odbić światła. Wektor s określa kierunek, z którego pada światło (we współrzędnych kartezjańskich – wektor 3elementowy lub biegunowych – 2elementowy), wektor k – parametry światła padającego i odbijającego się.

```
przykład:  surfl(x,y,z)
           colormap(gray)
```


- ```
przykład: % wir wodny
close all % usunięcie starych rysunków
[x,y]=meshgrid(-10:0.7:10,-10:0.7:10);
r=sqrt(x.^2+y.^2); % promienie - odległości od (0,0)
a=atan(x./y); % kąty odchylenia punktów
 % od dodatniej półosi x
d=max(max(a))-min(min(a)); % usuń uskoki kąta
z=cos(r-a*2*pi/d)*0.1+0.02*r;
surf(x,y,z)
axis([-10 10 -10 10 -0.2 0.5])
colormap(gray)
shading interp
```

- **plot3(x,y,z)**

Trójwymiarowy odpowiednik funkcji **plot**.

- **contour(z)**

**contour(z,n)**             $n$  – liczba poziomicy

**contour(z,v)**             $v$  – wektor wysokości kolejnych poziomicy

**contour(x,y,z)**

**contour(x,y,z,n)**

**contour(x,y,z,v)**

Rysuje wykres poziomicy elementów macierzy traktując ich wartości jako wysokość nad płaszczyzną odniesienia.

przykład: 

```
[x,y]=meshgrid(-3:0.3:3);
z=exp(-(x-1).^2-y.^2)+exp(-(x+1).^2-y.^2);
contour(x,y,z)
```

- inne funkcje graficzne:

- **drawnow**

Funkcja **drawnow** powoduje uaktualnienie rysunków. Standardowo rysunki są uaktualniane w następujących przypadkach:

- powrót do znaku zachęty okna poleceń,
- wykonanie polecenia **pause**,
- wywołanie funkcji **getframe**.

- **whitebg**

**whitebg(f)**

**whitebg(f,c)**

**whitebg(c)**

Funkcja **whitebg** zmienia tło aktywnego rysunku na białe i modyfikuje odpowiednio własności tak, aby rysunek wyglądał dobrze.

Podając kolumnowy wektor identyfikatorów rysunków  $f$  można zmienić tło kilku rysunków jednocześnie.

Parametr  $c$  służy do ustalenia innego koloru niż biały. Może on mieć postać trójelementowego wektora wierszowego opisującego kolor w standardzie RGB lub ciągu znaków zawierającego pełną nazwę lub pierwszą literę nazwy koloru – tak jak w przypadku funkcji **plot**.

- **graymon**

Funkcja **graymon** zmienia rysunki tak, aby były czytelne na monitorze czarno-białym.

przykład: 

```
x=0:pi/15:6*pi;
subplot(2,1,1)
wykres1=plot(x,cos(2.*x)./sqrt(x+1));
title('Wykres pierwszy')
subplot(2,1,2)
wykres2=plot(x,cos(2.*x)./sqrt(x+1),':',x,sin(x.*2));
axis([-1,30,-1.5,1.5])
text(20,1,'Dwie funkcje na jednym')
ylabel('ymin=-1.5 ymax=1.5')
xlabel('xmin=-1 xmax=30')
```