

# Programowanie w zastosowaniach inżynierskich

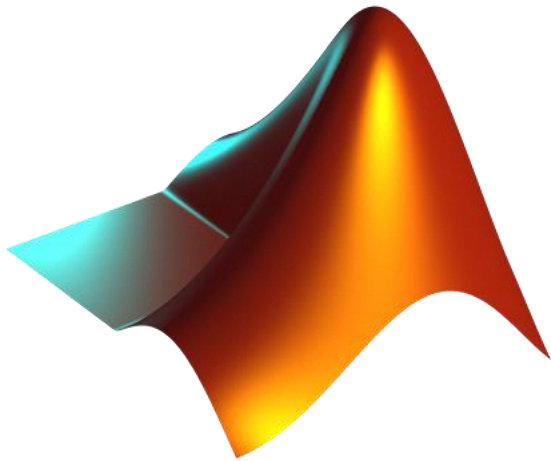
## Wprowadzenie do środowiska MATLAB

Zmienne i typy danych

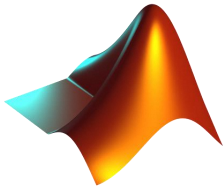
Tablice i macierze

Skrypty i funkcje

Wykresy 2D i 3D



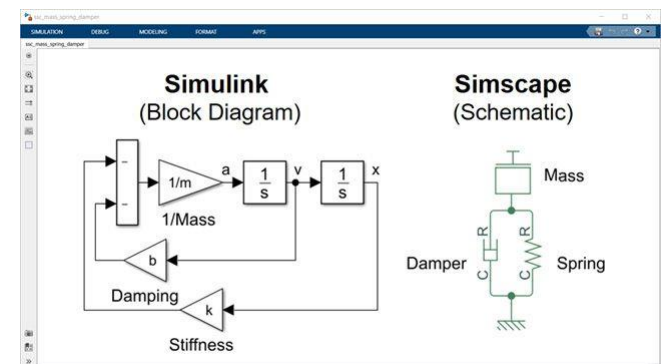
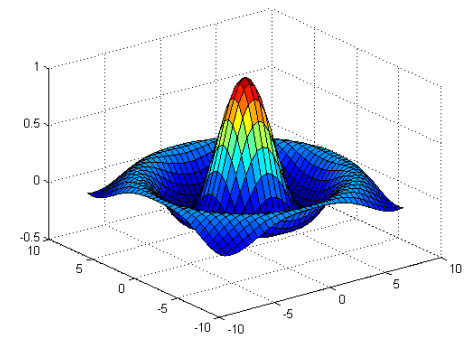
- ❑ Mrozek B., Mrozek Z., *MATLAB i Simulink. Poradnik użytkownika*, Helion, Gliwice 2017.
- ❑ Pratap R., *Matlab dla naukowców i inżynierów*, PWN, 2015
- ❑ Sradomski W., *MATLAB. Praktyczny podręcznik modelowania*, Helion, Gliwice 2015.
- ❑ Treichel W., *Matlab w działaniu. Ćwiczenia i zadania*, Witkom, 2021
- ❑ e-learning na stronie [matlab.mathworks.com](http://matlab.mathworks.com)
  - MATLAB Onramp – podstawy środowiska MATLAB, 14 mod., 2 godz.
  - MATLAB Fundamentals – wprowadzenie do analizy i wizualizacji danych, modelowania i programowania w MATLAB-ie, 18 modułów, 16.5 godz.
  - Simulink Onramp – podstawy modelowania w Simulinku, 14 mod., 2 godz.
  - Simscape Onramp – modelowanie układów fizycznych, 9 mod., 1.5 godz.



**MATLAB** – *MATrix LABoratory*, program komputerowy początkowo przeznaczony do obliczeń macierzowych, obecnie interaktywne środowisko do wykonywania obliczeń naukowych i inżynierskich oraz tworzenia zaawansowanych symulacji.

## Najważniejsze cechy programu Matlab

- Duża liczba funkcji bibliotecznych i możliwość rozbudowy przez użytkownika.
- Wbudowany język programowania wysokiego poziomu.
- Możliwość tworzenia dwu i trójwymiarowych wykresów oraz graficznej wizualizacji wyników (statyczne rysunki oraz animacje).
- Możliwość odbierania i wysyłania danych z/do urządzeń zewnętrznych.
- Dodatkowe biblioteki (toolboksy) do rozwiązywania specjalistycznych problemów z różnych dziedzin (matematyka, automatyka, elektronika, sztuczna inteligencja, itp.).
- Graficzne narzędzia do projektowania i symulacji fizycznych systemów z wykorzystaniem schematów i diagramów blokowych (Simulink, Simscape).



MATLAB Search Help Center GP

Get Help

- Documentation
- MATLAB Answers
- File Exchange
- Videos

Learn

- Online Training
- Cody
- Blogs

Open MATLAB Online Matlab w przeglądarce


Recent: Files | Folders

Name
import MATLAB Drive/import
MATLABwithPythonWorkshopExpo MATLAB Drive/MATLABwithPythonWorkshopExpo
Solution MATLAB Drive/MATLABwithPythonWorkshopExpo/Solution
mars_rover_yolov2_dl MATLAB Drive/Repositories/model_autonomous_navigation_mars_rover/mars_rover_yolov2_dl
model_autonomous_navigation_mars_rover MATLAB Drive/Repositories/model_autonomous_navigation_mars_rover

View more


View all files

Online Training




**Simulink Onramp**  
Unlimited Access

Start



**MATLAB Fundamentals**  
Access Expires 30.05.2024

Start



**Deep Learning Onramp**  
Unlimited Access

Start

Wideo poradniki

Lekcje online

# Interfejs Matlaba (R2022a)

**Menu**

**Editor**

```
1 close all
2 clear global; clc;
3
4 task_conf;
5
6
7 diffeqn = @diffeqn3;
8 tic;
9 [T, X] = ode45(diffeqn, [t0:0.01:25], X0, odeset('OutputFcn',
10
11 c0 = 0; t0 = 0;
12 qb = []; tcp = []);
13 for i=1:length(T)
14     x = diffeqn(T(i), X(i,:));
```

**Layout**

- SELECT LAYOUT
- Default
- Two Column
- All but Command Window Minimized
- Command Window Only
- Save Layout...
- Organize Layouts...
- SHOW
- Current Folder
- Workspace
- Panel Titles
- Toolbar
- Command History
- Quick Access Toolbar
- Current Folder Toolbar

**Current Folder**

- linfo.txt
- de1.mat
- de2.mat
- de3.mat
- de3q.txt
- de3qp.txt
- diffeqn1.m
- diffeqn2.m
- diffeqn3.m
- funout.m
- kin.m
- kin(i).m
- konf2b.mat
- m2py.m
- plot\_motion.m
- results.m

**Workspace**

Name	Value
a	5x5 double
a2	-0.6127
a3	-0.5716
b	5x7 double
c	5x7 double
c0	0.0730
Cout	1x22 double
d1	0.1807
d4	0.1742
d5	0.1199
d6	0.1166
diffeqn	@diffeqn3
eps	0.0020
epsWP	0.0500
fLV	@(ll)(ll.^0.5)*2.1
GWP	4
h	0.0050

**Command Window**

```
>> b = rand(5,7)*10-5
b =
    2.5774    2.0605    3.2346   -0.6126   -0.1024   -2.2397   -0.0164
    2.4313   -4.6817    1.9483   -1.1844   -0.5441    1.7970    4.5974
   -1.0777   -2.2308   -1.8290    2.6552    1.4631    1.5510   -1.5961
    1.5548   -4.5383    4.5022    2.9520    2.0936   -3.3739    0.8527
   -3.2881   -4.0287   -4.6555   -3.1313    2.5469   -3.8100   -2.7619

>> c = a*b
c =
   -8.6722   42.9424  -14.7724  -21.7002   -6.6729  -13.4431  -20.4443
   14.0517   30.4887   18.2956   24.8614   -5.7875   14.5588   -5.6052
  -18.4173  -53.0116  -17.0501   15.2085   24.4115  -11.0407  -11.1783
   12.2083  -43.3207   15.5031  -13.3028   14.0445  -27.6579   11.7046
   12.7452  -53.7897   20.2128    9.6235   15.9007  -12.2656   15.5399

fx >>
```

# Podstawowe elementy interfejsu

---

**Menu** – zestaw wszystkich opcji programu zebranych w postaci wstążki o zmiennej zawartości dostosowanej do aktualnie wybranego elementu interfejsu.

**Command Window** – okno umożliwiające interaktywną pracę z programem, każda wprowadzona komenda jest wykonywana, a jej wynik zostaje wyświetlony poniżej.

**Editor** – edytor tekstu umożliwiający edycję programów/skryptów użytkownika, skrypty mogą być uruchomione przez wprowadzenie ich nazwy w Command Window, lub z wykorzystaniem opcji dostępnych na wstążce menu.

**Current Folder** – zawartość aktualnie wybranego folderu (pod prawym przyciskiem dostępne podstawowe operacje na plikach).

**Workspace** – wykaz danych utworzonych w aktualnej sesji (dane są usuwane po zakończeniu pracy programu); podwójne kliknięcie pozwala na podgląd zawartości.

*Układ elementów interfejsu może być dowolnie dostosowany przez użytkownika (przeciąganie elementów myszą), układy domyślne, opcje zapisu/odczytu układu oraz widoczne elementy interfejsu mogą być ustawiane w menu Layout.*

# Zmienne i podstawowe typy danych

**Zmienna** – element języka programowania używany do przechowywania danych. Podczas pracy programu wartości zmiennej mogą ulegać modyfikacji.

## Wybrane typy danych

- numeryczne: `double`, `single` – liczby rzeczywiste (podwójna i pojedyncza precyzja), `int8`, `int16`, `int32`, `int64` – liczby całkowite (8, 16, 32 i 64 bitowe);
- tekstowe: `char` – znak (wartość podawana w apostrofach); `string` – łańcuch/ciąg znakowy (wartość podawana w cudzysłowach);
- logiczne: `logical` – wartości zapisywane jako liczbowe, interpretowane jak "prawda" (każda liczba różna od zera) i "fałsz" (wartość zero);

*Uwaga: każda wartość numeryczna domyślnie traktowana jest jako `double`.*

## Tworzenie zmiennej

`nazwa = wartość`

`nazwa` – unikalny ciąg znaków, dozwolone litery, cyfry, znak podkreślenia, zaczyna się od litery.

`wartość` – dowolna wartość dozwolona w programie Matlab, jej typ określa typ zmiennej.

## Przykłady

```
x = 123, cena = 550.27, nazwisko = "Kowalski"
```

**Tablica (array)** – kolekcja elementów tego samego typu zorganizowanych w strukturze o określonej liczbie wierszy i kolumn (możliwa większa liczba wymiarów).

**Macierz (matrix)** – tablica zawierająca wartości numeryczne; macierz o jednym wierszu lub kolumnie jest nazywana **wektorem**.

## Tworzenie tablic

$$\text{nazwa} = [w_{11}, w_{12}, \dots, w_{1m}; w_{21}, w_{22}, \dots, w_{2m}; \dots, w_{n1}, w_{n2}, \dots, w_{nm}]$$

Wartości tablicy podawane są wierszami w nawiasach kwadratowych, elementy wiersza oddzielane są przecinkami lub spacjami, wiersze oddzielane są średnikami.

*Uwaga: w Matlabie każda wartość traktowana jest jako tablica, wartość prosta (pojedyncza liczba, tekst, itp.) jest tablicą o jednym wierszu i jednej kolumnie.*

## Funkcje specjalne

- `zeros(n)`, `zeros(n,m)` – tablica wypełniona zerami
- `ones(n)`, `ones(n,m)` – tablica wypełniona jedynekami
- `rand(n)`, `rand(n,m)` – tablica wypełniona wartościami losowymi
- `eye(n)`, `eye(n,m)` – macierz jednostkowa (jedyńki na głównej przekątnej)
- `a:b`, `a:b:k` – tablica jednowierszowa od `a` do `b` z domyślnym krokiem 1 lub krokiem określonym przez `k` (maksymalna wartość nie przekracza `b`).



## Indeksowanie tablic

- `nazwa_tablicy(i)` – odwołanie do *i*-tego elementu (numerowanie kolumnowe)
- `nazwa_tablicy(i, j)` – odwołanie do elementu z *i*-tego wiersza i *j*-tej kolumny

*Uwaga: indeksy mogą być tablicami w takim przypadku odwołanie dotyczy wielu elementów, wierszy lub kolumn; w przypadku tablic o liczbie wymiarów większej od 2 liczba indeksów rośnie.*

## Symbole specjalne

- dwukropek (`:`) – wszystkie elementy, wiersze, kolumny tablicy
- `end` – ostatni element, wiersz, kolumna tablicy

## Modyfikowanie elementów tablicy

`nazwa_tablicy(index) = wartość`

- jeżeli `index` wskazuje pojedynczy element zostaje on zastąpiony przez `wartość`
- jeżeli `index` jest tablicą wszystkie elementy będą zastąpione przez `wartość`
- jeżeli `wartość` jest tablicą o wymiarach zgodnych wymiarami indeksu wszystkie elementy zostaną zastąpione przez odpowiadające im elementy `wartości`

# Indeksowanie tablic – przykłady

```
>> a = [1 2 3; 4 5 6]
```

```
a =  
    1    2    3  
    4    5    6
```

```
>> b = [10 20 30]
```

```
b =  
    10    20    30
```

```
>> c = [100; 200]
```

```
c =  
    100  
    200
```

```
>> a(1,3)
```

```
ans =  
     3
```

```
>> a(3)
```

```
ans =  
     2
```

```
>> a(2,:) 
```

```
ans =  
     4     5     6
```

```
>> a(1,3) = 30
```

```
a =  
    1    2    30  
    4    5     6
```

```
>> a(1,:) = 10
```

```
a =  
    10    10    10  
     4     5     6
```

```
>> a(end,1:2) = b(2:3)
```

```
a =  
    10    10    10  
    20    30     6
```

*Uwaga: jeżeli uruchamiane wyrażenie nie będzie przypisane do zmiennej Matlab automatycznie utworzy zmienną "ans" (answer).*

# Operatory relacyjne i logiczne

## Operatory relacyjne

Operator	Opis
<code>==</code>	Równy
<code>&gt;=</code>	Większy lub równy
<code>&gt;</code>	Większy
<code>&lt;=</code>	Mniejszy lub równy
<code>&lt;</code>	Mniejszy
<code>~=</code>	Różny
<code>isequal</code>	Równość tablic

## Operatory logiczne

Operator	Opis
<code>&amp;</code>	Koniunkcja (AND)
<code> </code>	Alternatywa (OR)
<code>~</code>	Negacja (NOT)

x	y	x & y	x   y	~x
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

*Uwaga I:* wynikiem działania operatorów relacyjnych jest wartość logiczna (0 – fałsz, 1 – prawda).

*Uwaga II:* w przypadku porównywania tablic wynikiem operatorów relacyjnych jest tablica logiczna o wymiarze równym tablicy porównywanej zawierająca rezultat porównania poszczególnych elementów.

*Uwaga III:* tablice mogą być indeksowane tablicami logicznymi jako wynik zwracane są wszystkie elementy, które odpowiadają indeksom o wartości 1 (prawda).

# Indeksowanie tablic wartościami logicznymi

```
>> v = [ -2 0 4 -3 5 ]
v =
    -2     0     4    -3     5
>> v > 0
ans =
    1x5 logical array
     0     0     1     0     1
>> v(v>0)
ans =
     4     5
>> v(v>0) = 0
v =
    -2     0     0    -3     0
>> v(v<0)=-v(v<0)
v =
     2     0     0     3     0
```

```
• >> m = [ 1 -7 3 9; -6 6 0 -2 ]
• m =
     1    -7     3     9
    -6     6     0    -2
• >> idx = ( m < -5 ) | ( m > 5 )
• idx =
    2x4 logical array
     0     1     0     1
     1     1     0     0
• >> m(idx) = sign(m(idx))*5
• m =
     1    -5     3     5
    -5     5     0    -2
• >> m(~idx) = 0
• m =
     0    -5     0     5
    -5     5     0     0
```

- **sign** zwraca znak wartości (1 lub -1)

# Operatory tablicowe i macierzowe

**Operatory tablicowe** operują na pojedynczych elementach tablicy.

**Operatory macierzowe** wykonują operacje zgodnie z regułami algebry liniowej.

Operator		Opis
+		Suma tablic/macierzy
-		Różnica tablic/macierzy
.*	*	Iloczyn tablicowy ( $A.*B=[A(i,j)*B(i,j)]$ ), mnożenie macierzy
./	/	Prawostronne dzielenie tablic ( $A./B=[A(i,j)/B(i,j)]$ ), prawostronne dzielenie macierzy (rozwiązanie układu równań $xA=B$ , A i B muszą mieć tę samą liczbę kolumn)
.\	\	Lewostronne dzielenie tablic ( $A.\B=[B(i,j)/A(i,j)]$ ), lewostronne dzielenie macierzy (rozwiązanie układu równań $Ax=B$ , A i B muszą mieć tę samą liczbę wierszy)
.^	^	Potęgowanie tablicowe ( $A.^B=A(i,j)^B(i,j)$ ), potęgowanie macierzy
.'	'	Transpozycja (wiersze są przekształcane w kolumny), transpozycja algebraiczna (różnica występuje tylko dla macierzy zespolonych)

*Uwaga: część operacji wykonywana jest tak samo w przypadku tablic i macierzy, takie operatory mają pojedynczy symbol.*

# Operatory tablicowe – przykłady

```
>> a = [1 2 3; 4 5 6];
```

```
>> b = [3 2 1; 0 -1 -2];
```

```
>> c = a + b
```

```
c =
```

```
4 4 4
4 4 4
```

```
>> c = a + 2
```

```
c =
```

```
3 4 5
6 7 8
```

```
>> c = a .* b
```

```
c =
```

```
3 4 3
0 -5 -12
```

```
>> c = a.^2
```

```
c =
```

```
1 4 9
16 25 36
```

```
>> c = b.^a
```

```
c =
```

```
3 4 1
0 -1 64
```

```
>> c = a'
```

```
c =
```

```
1 4
2 5
3 6
```

*Uwaga: jeżeli wyrażenie zostanie zakończone średnikiem, to jego wynik nie będzie wyświetlony w Command Window.*

# Operatory macierzowe – przykłady

```
a = [2 0.5; 3 1];  
b = [1; 2];  
c = [1 2 3; 4 5 6];
```

*Uwaga: mnożenie macierzy jest wykonalne, gdy liczba kolumn pierwszej jest równa liczbie wierszy drugiej*

```
>> d = c * 3
```

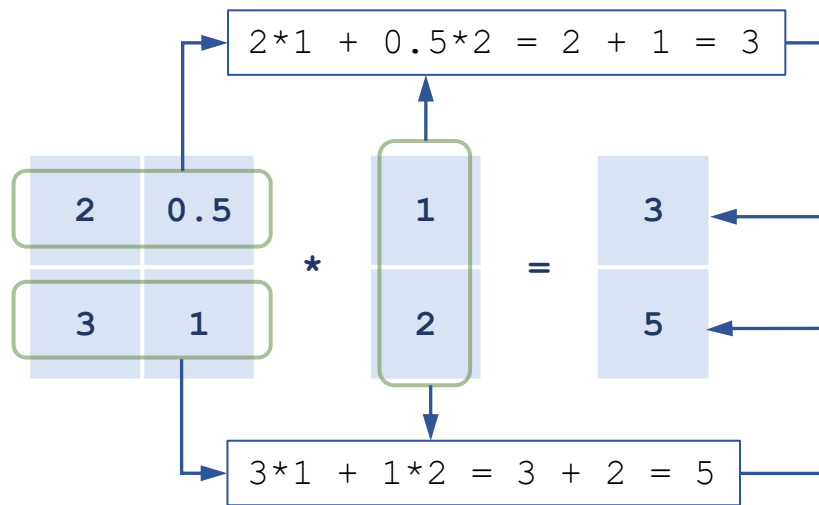
```
d =  
     3     6     9  
    12    15    18
```

```
>> d = a * b
```

```
d =  
     3  
     5
```

```
>> d = a * c
```

```
d =  
     4     6.5     9  
     7     11    15
```



## Łączenie i modyfikowanie tablic

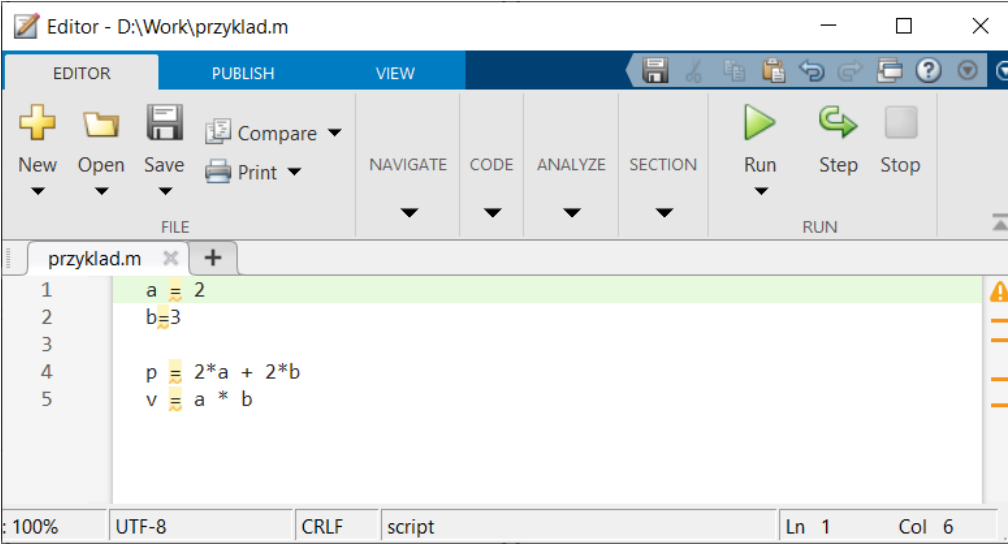
- `horzcat(A, B, ..., C)` – łączy tablice w poziomie (alternatywa `[A, B, ...C]`), liczba wierszy we wszystkich tablicach musi być równa
- `vertcat(A, B, ..., C)` – łączy tablice w pionie (alternatywa `[A; B; ...C]`), liczba kolumn we wszystkich tablicach musi być równa
- `reshape(A, d)` – przekształca `A` przestawiając elementy tak żeby utworzyły tablicę o wymiarach określonych przez wektor `d` (liczba elementów nie może ulec zmianie)
- `A(index) = []` – usuwa element określony przez `index`

## Rozmiar i kształt tablic

- `length(A)` – długość największego wymiaru tablicy
- `size(a)` – rozmiar tablicy (wektor dla tablic wielowymiarowych)
- `ndims(A)` – liczba wymiarów tablicy
- `numel(A)` – liczba elementów tablicy
- `isscalar(A)` – określa czy `A` jest wartością skalarną (np. pojedyncza liczba)
- `isvector(A)`, `iscolumn(A)`, `isrow(A)` – określa czy `A` jest wektorem (dowolnym, kolumnowym, wierszowym)
- `ismatrix(A)` – określa czy `A` jest macierzą



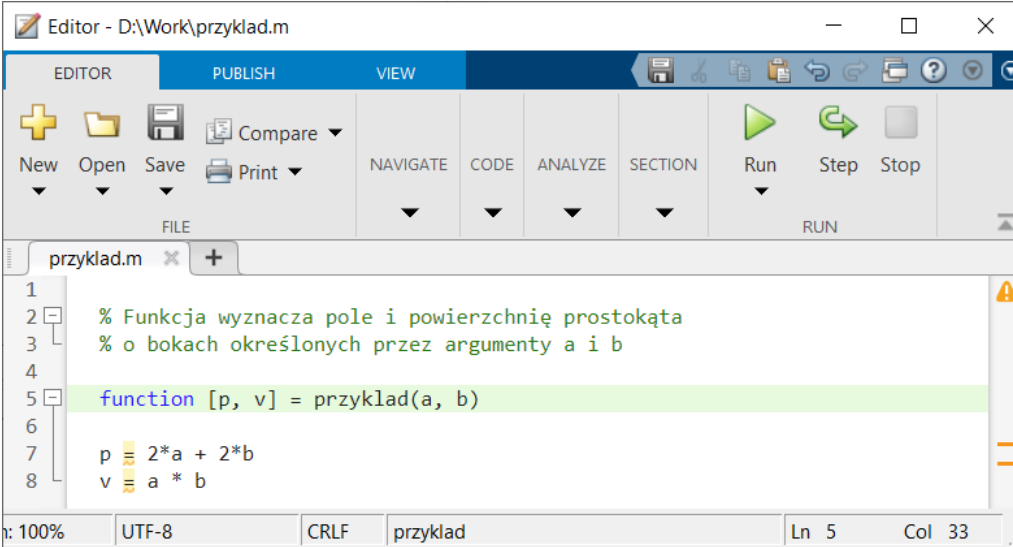
**Skrypt (m-plik)** – ciąg poleceń Matlba zapisany w pliku tekstowym \*.m, uruchomienie skryptu powoduje wykonanie poleceń zapisanych w kolejnych wierszach.



```
Editor - D:\Work\przyklad.m
EDITOR PUBLISH VIEW
New Open Save Compare Print
NAVIGATE CODE ANALYZE SECTION Run Step Stop
FILE RUN
przyklad.m
1 a = 2
2 b = 3
3
4 p = 2*a + 2*b
5 v = a * b
: 100% UTF-8 CRLF script Ln 1 Col 6
```

- nazwa pliku nie powinna zawierać polskich znaków,
- uruchomienie skryptu następuje po wprowadzeniu nazwy pliku w Command Window (bez rozszerzenia) lub po wybraniu opcji Run w edytorze,
- skrypt może być uruchomiony z innego skryptu przez wprowadzenia jego nazwy,
- wszystkie zmienne utworzone podczas działania skryptu pozostają w obszarze roboczym Matlaba (również po zakończeniu skryptu),
- tekst rozpoczynający się znakiem % jest komentarzem i będzie pomijany podczas wykonania.

**Funkcja** – rodzaj skryptu Matlaba, który pozwala na zdefiniowanie argumentów wejściowych i wyjściowych.



The screenshot shows the MATLAB Editor interface. The title bar reads "Editor - D:\Work\przyklad.m". The ribbon includes "EDITOR", "PUBLISH", and "VIEW" tabs. The "EDITOR" tab is active, showing a menu with "New", "Open", "Save", "Compare", and "Print". Below the menu are sections for "NAVIGATE", "CODE", "ANALYZE", and "SECTION", each with a dropdown arrow. To the right are "Run", "Step", and "Stop" buttons. The main editor area shows a file named "przyklad.m" with the following code:

```
1  
2 % Funkcja wyznacza pole i powierzchnię prostokąta  
3 % o bokach określonych przez argumenty a i b  
4  
5 function [p, v] = przyklad(a, b)  
6  
7 p = 2*a + 2*b  
8 v = a * b
```

The status bar at the bottom indicates "Ln 5 Col 33".

- nazwa funkcji powinna być zgodna z nazwą pliku, w którym jest zapisana,
- funkcja ma własny, lokalny obszar roboczy, zmienne utworzone podczas jej działania nie są dostępne w globalnym obszarze roboczym Matlaba (są usuwane po zakończeniu funkcji),
- opcjonalny komentarz umieszczony przed deklaracją funkcji będzie wyświetlany w przypadku użycia komendy `help`.

## Deklaracja funkcji

```
function [out1,out2,...,outN] = nazwa(in1,in2,...,inM)
```

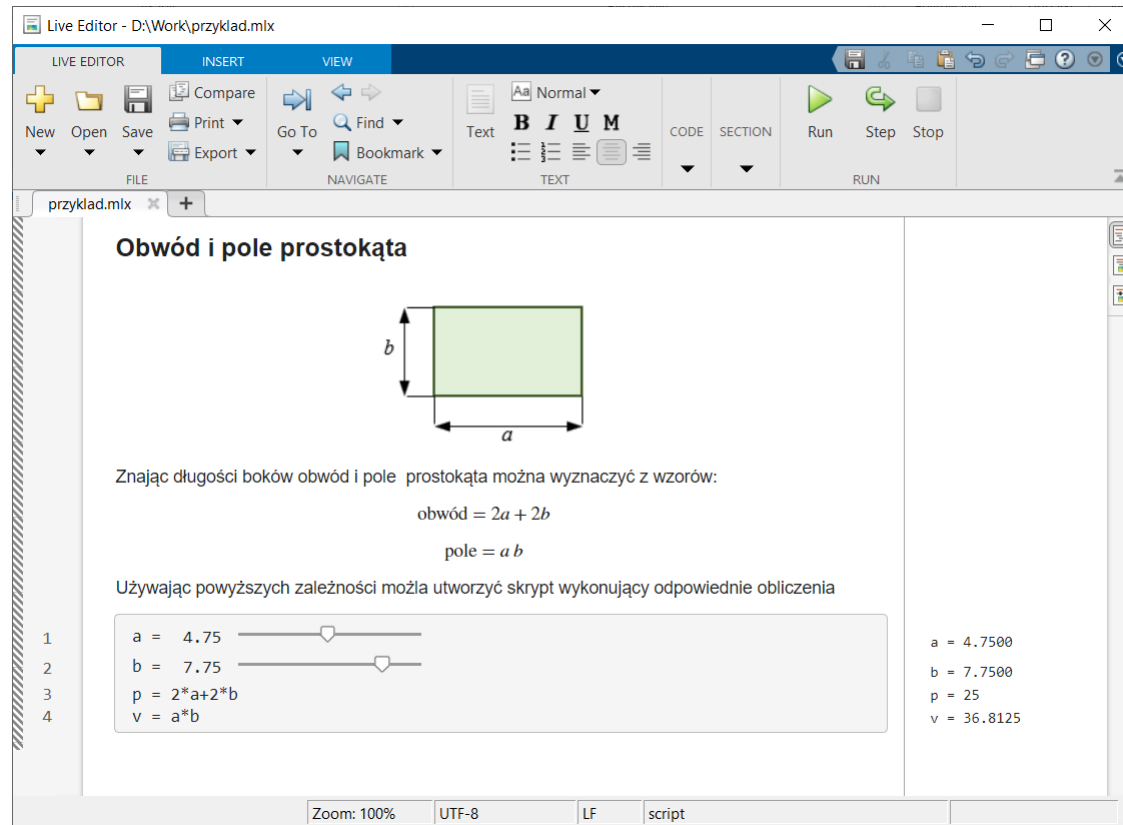
- `nazwa` – nazwa funkcji, dozwolone litery, cyfry, znak podkreślenia, zaczyna się od litery, zgodna z nazwą pliku, w którym funkcja jest zapisana,
- `out1, ..., outN` – argumenty wyjściowe (opcjonalne, zasady nazewnictwa jak zmienne),
- `in1, ..., inM` – argumenty wejściowe (opcjonalne, zasady nazewnictwa jak zmienne),
- w jednym pliku można umieścić deklarację kilku funkcji, pierwsza jest funkcją główną, pozostałe są lokalne, dostępne tylko w pliku, w którym zostały zadeklarowane,
- deklaracja funkcji kończy się słowem `end` (opcjonalne gdy plik zawiera jedną funkcję),
- nagłówek funkcji głównej powinien być pierwszym wykonywalnym wierszem skryptu.

## Wywołanie funkcji

```
[out1,out2,...,outN] = nazwa(in1,in2,...,inN)
```

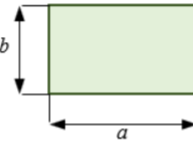
- liczba argumentów wejściowych i wyjściowych nie może być większa od określonej w deklaracji (nagłówku) funkcji,
- w przypadku mniejszej liczby parametrów wyjściowych pozostałe wartości zwracane przez funkcję zostaną zignorowane.

**Live script** – interaktywny dokument zapisany w pliku **\*.mlx**, który łączy kod Matlabu ze sformatowanym tekstem, elementami graficznymi, multimedialnymi, sterującymi, itp.



The screenshot displays the MATLAB Live Editor window titled "Live Editor - D:\Work\przyklad.mlx". The interface includes a menu bar with "LIVE EDITOR", "INSERT", and "VIEW" tabs, and a toolbar with icons for file operations (New, Open, Save, Print, Export), navigation (Go To, Find, Bookmark), text formatting (Normal, Bold, Italic, Underline, Monospace), and execution (Run, Step, Stop). The main workspace contains the following content:

### Obwód i pole prostokąta



Znając długości boków obwód i pole prostokąta można wyznaczyć z wzorów:

$$\text{obwód} = 2a + 2b$$
$$\text{pole} = a b$$

Używając powyższych zależności można utworzyć skrypt wykonujący odpowiednie obliczenia

```
1 a = 4.75
2 b = 7.75
3 p = 2*a+2*b
4 v = a*b
```

Output results:

```
a = 4.7500
b = 7.7500
p = 25
v = 36.8125
```

The status bar at the bottom shows "Zoom: 100%", "UTF-8", "LF", and "script".

```
function [v] = pole(a, b)
```

```
v = a*b;
```

```
end
```

```
>> pole(2,3)
```

```
ans =
```

```
6
```

```
>> x = pole(2,3)
```

```
x =
```

```
6
```

```
>> x = pole(4)
```

```
Not enough input arguments.
```

```
function [v,p] = pole_obwod(a, b)
```

```
v = a*b;
```

```
p = 2*a + 2*b
```

```
end
```

```
>> pole_obwod(2,3)
```

```
ans =
```

```
6
```

```
>> x = pole_obwod(2,3)
```

```
x =
```

```
6
```

```
>> [x,y] = pole_obwod(2,3)
```

```
x =
```

```
6
```

```
y =
```

```
10
```

Zadanie: należy wyznaczyć minimum funkcji

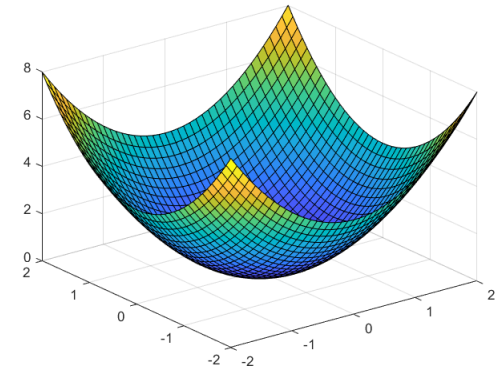
$$z = x^2 + y^2$$

Używana funkcja Matlab

```
[x,fval] = fminsearch(f, x0)
```

- `x` to znalezione minimum,
- `fval` to wartość funkcji w minimum
- `f` to funkcja, której minimum jest poszukiwane (zadeklarowana jako funkcja Matlab),
- `x0` to punkt początkowy (punkt od którego rozpoczyna się poszukiwanie minimum).

*Uwaga: `fminsearch` operuje na funkcjach jednoargumentowych, jeżeli minimalizowana funkcja jest funkcją wielu zmiennych to należy przekazywać je w postaci wektora.*



```
function y = fun(x)
y = x(1)^2 + x(2)^2;
End
```

```
>> [x,v] = fminsearch(@fun, [1,1])
x =
    1.0e-04 *
   -0.2102    0.2548
v =
    1.0915e-09
```

Uwaga:

$1.0e-04 = 1 * 10^{-4} = 0.0001$

$1.0e+04 = 1 * 10^4 = 10000$

Funkcje rysujące wykresy 2D wymagają przygotowania danych w postaci dwóch wektorów zawierających punkty z dziedziny i odpowiadające im wartości.

## Wybrane funkcje wykreślające wykresy 2D

- `plot(X, Y, LineSpec)` – wykres liniowy
- `stairs(X, Y, LineSpec)` – wykres schodkowy
- `bar(X, Y), barh(X, Y)` – wykres słupkowy (pionowy i poziomy)
- `histogram(X, n)` – wyświetla histogram danych zgromadzonych w `X` z zastosowaniem podziału na `n` grup (ilość elementów w każdej grupie)
- `bubblechart(X, Y, s)` – wykres bąbelkowy (w punktach określonych przez `X` i `Y` rysowane jest kółko o rozmiarze `s`, argument `s` może być wektorem)

### Uwagi:

- funkcja wymagające określenia dziedziny i zbioru wartości akceptują jeden argument, który traktowany jest jako zbiór wartości, dziedzinę stanowią wówczas numery próbek,
- w funkcji `plot` argument `Y` może być macierzą wielokolumnową, każda kolumna zostanie potraktowana jako osobna seria danych (funkcja wykreśli kilka wykresów).
- w funkcji `plot` dziedzina i zbiór wartości mogą być podawane wielokrotnie (jako kolejne argumenty), każdy zbiór danych zostanie wykreślony jako kolejny wykres.

# Wykresy 2D – parametry linii

Parametr `LineStyle` pozwala określić styl i kolor linii używanej do wykreślenia wykresu oraz rodzaj markerów używanych do oznaczenia punktów. Parametr podawany jest jako łańcuch znakowy (tekst w cudzysłowach) zawierający zestaw symboli.

<b>Styl linii</b>	
<b>Symbol</b>	<b>Opis</b>
-	Linia ciągła
--	Linia przerywana
:	Linia kropkowa
-.	Linia kreska-kropka

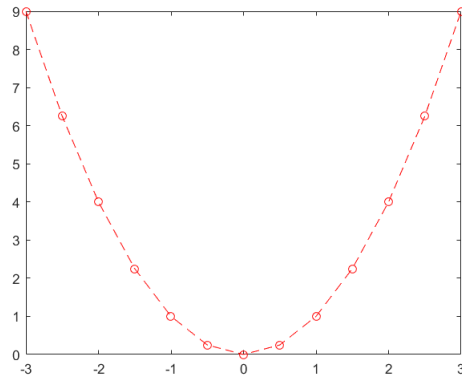
<b>Markery</b>	
<b>Symbol</b>	<b>Opis</b>
o	Kółko
+	Plus
*	Gwiazdka
.	Kropka
x	Krzyż

<b>Kolory</b>	
<b>Symbol</b>	<b>Opis (R,G,B)</b>
r	Czerwony (1,0,0)
g	Zielony (0,1,0)
b	Niebieski (0,0,1)
c	Cyjan (0,1,1)
m	Magenta (1,0,1)
y	Żółty (1,1,0)
k	Czarny (0,0,0)
w	Biały (1,1,1)

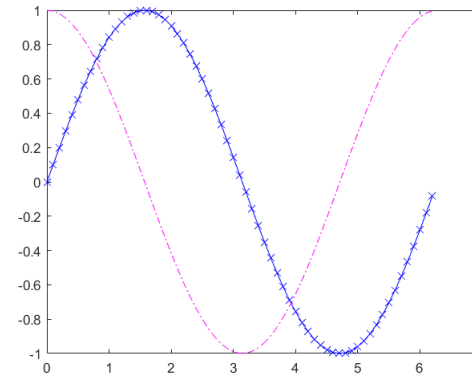


# Wykresy 2D – przykłady

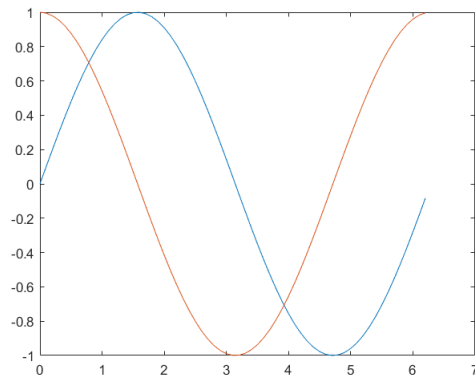
```
>> x = -3:0.5:3;  
>> y = x.^2;  
>> plot(x,y,"--or")
```



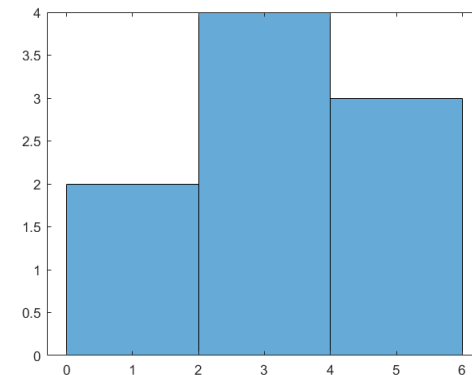
```
>> y=[sin(x); cos(x)];  
>> y1=sin(x); y2=cos(x);  
>> plot(x,y1,"-xb", x,y2,"-.m")
```



```
>> x=0:0.1:2*pi;  
>> y=[sin(x); cos(x)]
```



```
>> x = [ 2.5, 4, 2.2, 3, 6, 5, 0, 1, 3.5];  
>> histogram(x,3)
```



Funkcje rysujące wykresy 3D wymagają przygotowania danych w postaci trzech macierzy z których każda zawiera odpowiednią współrzędną kreślonego wykresu.

## Wybrane funkcje wykreślające wykresy 3D

- `surf(X, Y, Z)` – wykres powierzchniowy (oczka wypełnione)
- `mesh(X, Y, Z)` – wykres powierzchniowy (oczka niewypełnione)
- `contour(X, Y, Z)` – wykres poziomicowy

## Funkcje przygotowujące dziedzinę wykresu 3D

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

generuje siatkę punktów opisujących dziedzinę wykresu 3D,  $x$ ,  $y$  to wektory zawierające wartości współrzędnych siatki a  $X$ ,  $Y$  to wygenerowane na ich podstawie macierze ze współrzędnymi punktów dziedziny, funkcja `meshgrid` może być wywoływana z jednym parametrem jeśli  $y$  jest równe  $x$

- `[X1, X2, ...]=ndgrid(x1, x2, ...)`

generuje siatkę punktów opisujących dziedzinę wykresu w przestrzeni  $N$  wymiarowej, znaczenie parametrów  $x1, x2, \dots$  oraz  $X1, X2, \dots$  analogiczne jak w `meshgrid`

```
>> [X, Y] = meshgrid(-2:2)
```

```
X =
```

```

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

```
Y =
```

```

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

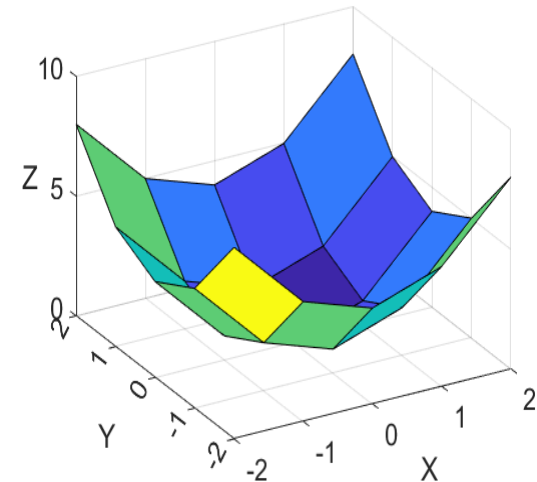
```
>> Z = X.^2 + Y.^2
```

```
Z =
```

```

 8     5     4     5     8
 5     2     1     2     5
 4     1     0     1     4
 5     2     1     2     5
 8     5     4     5     8
```

```
>> surf(X, Y, Z)
```



Dziedzina (złożenie X i Y)

```

(-2,-2) (-1,-2) (0,-2) (1,-2) (2,-2)
(-2,-1) (-1,-1) (0,-1) (1,-1) (2,-1)
(-2, 0) (-1, 0) (0, 0) (1, 0) (2, 0)
(-2, 1) (-1, 1) (0, 1) (1, 1) (2, 1)
(-2, 2) (-1, 2) (0, 2) (1, 2) (2, 2)
```