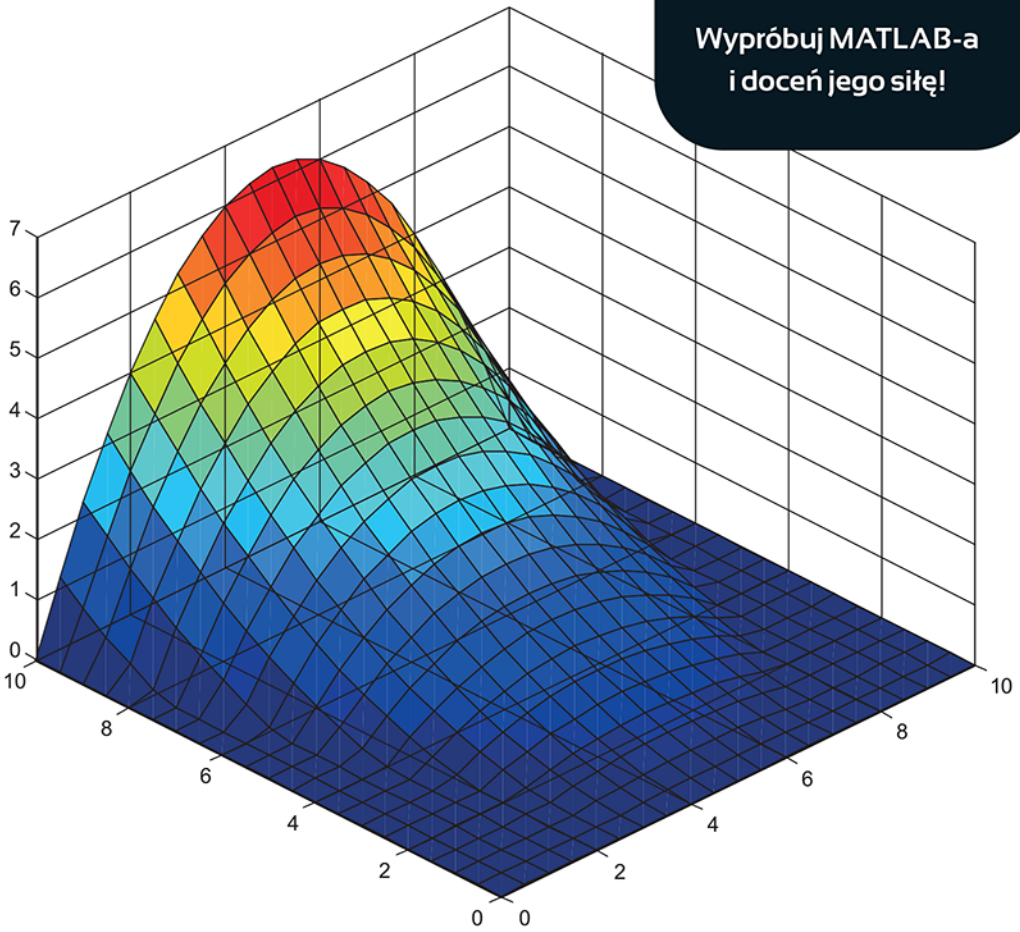


Wypróbuj MATLAB-a  
i docień jego siłę!



WALDEMAR SRADOMSKI

# MATLAB

Praktyczny podręcznik modelowania

Helion 

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione.

Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Ewelina Burska

Projekt okładki: Studio Gravite/Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/modmat>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-246-8134-1

Copyright © Helion 2015

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to!» Nasza społeczność](#)

# Spis treści

<b>Wstęp</b> .....	<b>5</b>
<b>Rozdział 1. MATLAB — szybki start</b> .....	<b>7</b>
Operator dwukropka .....	15
Dostęp do elementów macierzy .....	16
Funkcje MATLAB-a generujące tablice .....	17
Funkcje .....	19
Wykresy .....	21
<b>Rozdział 2. Skrypty i funkcje</b> .....	<b>29</b>
Uchwyt funkcji @ .....	32
Instrukcje sterujące przebiegiem programu .....	32
Warunkowe wykonywanie kodu — if .....	33
Warunek wielokrotny switch, case i otherwise .....	34
Pętle for, while, continue, break .....	35
Operatory logiczne .....	39
<b>Rozdział 3. Rozwiązywanie równań różniczkowych</b> .....	<b>41</b>
Przykłady rozwiązywania równań różniczkowych .....	42
Drapieżnik – ofiara .....	46
Wypływ cieczy ze zbiorników .....	49
Równania różniczkowe zwyczajne stopnia drugiego i wyższego rzędu .....	51
Oscylator harmoniczny .....	53
Oscylator harmoniczny z tłumieniem .....	55
Porównanie różnych algorytmów całkowania .....	58
Odbijanie się sprężystej piłki .....	61
Model zawieszenia samochodu .....	64
Elementy o parametrach rozłożonych .....	68
Przepływ ciepła .....	68
Przepływ ciepła i funkcja pdepe .....	74
Symulacja zmian temperatury w dwóch wymiarach .....	77
Drgająca struna .....	82
Drgania płyty .....	86
<b>Rozdział 4. Schematy blokowe (Simulink)</b> .....	<b>89</b>
Rysowanie schematów blokowych równań różniczkowych .....	94
Dynamika epidemii .....	97
Oscylator harmoniczny .....	102

Oscylator harmoniczny z wymuszeniem kinematycznym .....	111
Masa podnoszona na sprężystej linii .....	114
Siłownik hydrauliczny podnoszący masę .....	118
Oscylator harmoniczny z wahadłem matematycznym .....	129
Wahadło z niepełnym stopniem swobody .....	135
Sprężysta piłeczka .....	136
Animacja odbijającej się piłki .....	141
Model zderzaka hydraulicznego .....	145
Dwumasowy model zawieszenia samochodu .....	152
Rozruch przekładni hydrostatycznej .....	156
Impulsowy przetwornik elektrohydrauliczny .....	163
Rzut piłką do kosza .....	166
Katapultowanie się pilota z lecącego samolotu .....	169
Pantograf .....	173
Zderzenie dwóch wagonów .....	176
Hamowanie samolotu na pokładzie lotniskowca .....	179
<b>Rozdział 5. Przekształcenie operatorowe do rozwiązywania układów równań ....</b>	<b>183</b>
Przykład zastosowania przekształcenia operatorowego .....	186
Dobór regulatora całkującego .....	188
Przekształcenia operatorowe w Simulinku .....	191
Regulacja dwustawna temperatury w piecu .....	192
<b>Rozdział 6. Stateflow .....</b>	<b>195</b>
Warunek logiczny na przykładzie wartości bezwzględnej .....	204
Pętle realizowane za pomocą Stateflow .....	206
Regulator dwustawny .....	219
Sterowanie pompą w przepompowni .....	223
Odbicie piłki .....	233
Odpluskwanie schematów stanu .....	236
<b>Rozdział 7. Zaawansowane konfigurowanie wykresu .....</b>	<b>239</b>
Wykresy z dwiema osiami .....	244
Zapisywanie wykresu do pliku .....	246
<b>Literatura .....</b>	<b>249</b>
<b>Skorowidz .....</b>	<b>251</b>

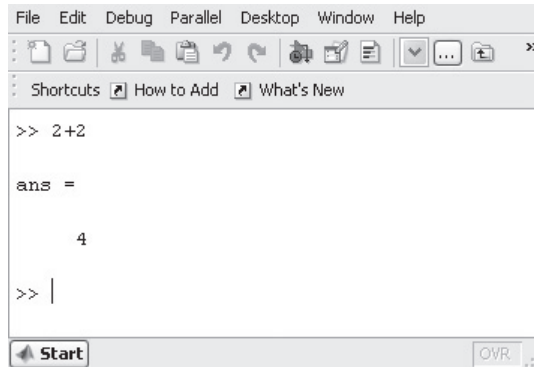
## Rozdział 1.

# MATLAB — szybki start

Interfejs MATLAB-a, oprócz typowych elementów, takich jak: menu, pasek narzędzi lub wstążka, pasek statusu, ma wyspecjalizowane okna, w których wyświetlane są poszczególne informacje. Okna można włączać, wyłączać, skalować, przenosić w prawo, w lewo, w górę, w dół, chować itp.

Najważniejszą część interfejsu MATLAB-a to okno poleceń (*Command Window*), za pomocą którego wpisując polecenia, wykonujemy obliczenia i uruchamiamy skrypty. Polecenia wpisujemy po znaku zachęty `>>` (rysunek 1.1). Spróbujmy wpisać coś łatwego, np. `2+2` ↵ (*Enter*).

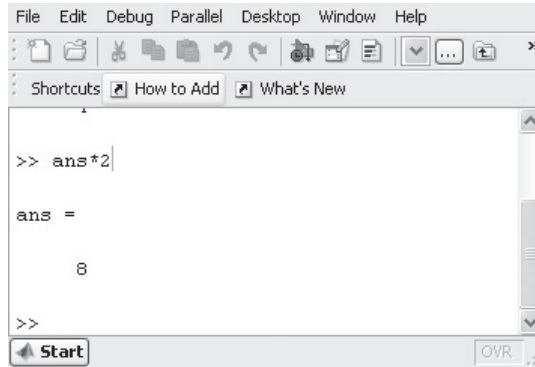
**Rysunek 1.1.**  
*Pierwsze polecenie w oknie Command Window*



Po naciśnięciu klawisza *Enter* (*Return*) MATLAB wykona polecenie, w tym przypadku sumowanie, i wypisze wynik. Widzimy, że MATLAB w odpowiedzi napisał `ans =` i wynik w wierszu niżej. Zmienna `ans` jest definiowana, gdy wynik wyrażenia nie jest przypisany do żadnej zmiennej. Możemy wykorzystać tę zmienną w następnych poleceniach, np. pomnożmy zmienną `ans` przez 2. Proszę zauważyć na rysunku 1.2, że po mnożeniu i nieprzypisaniu wyniku do zmiennej wartość przypisana do `ans` zmieniła się. Poprzednie polecenie przywołuje się za pomocą klawisza ↑ na klawiaturze, a więc szybko możemy liczyć wielokrotności liczby 4.

**Rysunek 1.2.**

*Polecenie z wykorzystaniem zmiennej*

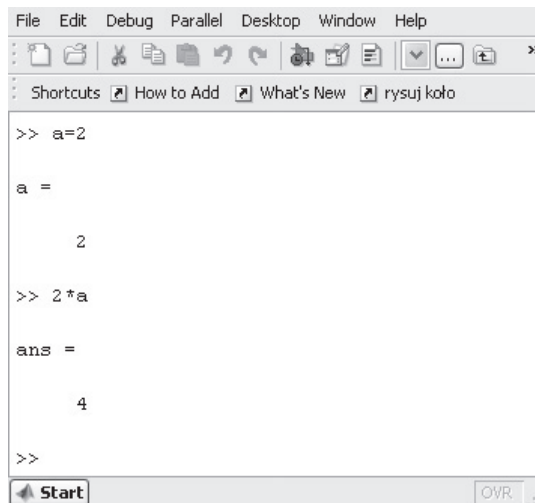


```
File Edit Debug Parallel Desktop Window Help
Shortcuts How to Add What's New
>> ans*2
ans =
     8
>>
Start OVR
```

Definiowanie własnych zmiennych i przypisywanie im wartości realizowane jest za pomocą znaku przypisania `=`. Składnia polecenia przypisania jest taka jak w większości języków programowania: `zmienna=wartość`, a więc jeżeli chcemy zdefiniować zmienną `a` i przypisać jej wartość 2, powinniśmy napisać `a=2` (rysunek 1.3). Typ zmiennej rozpoznawany jest w trakcie przypisania.

**Rysunek 1.3.**

*Definicja własnej zmiennej i mnożenie z wykorzystaniem tej zmiennej*



```
File Edit Debug Parallel Desktop Window Help
Shortcuts How to Add What's New rysuj koło
>> a=2
a =
     2
>> 2*a
ans =
     4
>>
Start OVR
```

Nazwy zmiennych możemy utworzyć z liter, cyfr, znaku podkreślenia, ale nazwa musi zaczynać się od litery. W nazwach nie mogą pojawiać się operatory, spacje, znaki narodowe.

Uwagi odnośnie do nazw:

- ◆ MATLAB **rozdziela wielkość liter w nazwach**, a więc `a` i `A` to dwie różne zmienne;
- ◆ nie są kontrolowane definiowane nazwy zmiennych, a więc nie będzie żadnego komunikatu, jeżeli utworzymy zmienną, która jest już zdefiniowana.

Przykładowe predefiniowane stałe<sup>1</sup> w MATLAB-ie:

- ♦ pi —  $\pi = 3.141592653589793$ ;
- ♦ i, j — *liczba urojona*,  $i^2 = -1$ ;
- ♦ ans — *wynik ostatniego polecenia*;
- ♦ eps — *najmniejsza liczba rozpoznawana jako nie zero*;
- ♦ realmax — *największa liczba rzeczywista* =  $1.7977e + 308$ ;
- ♦ realmin — *najmniejsza liczba rzeczywista* =  $2.2251e - 308$ .

Nic nie stoi na przeszkodzie, żeby utworzyć zmienną pi i przypisać jej wartość np. 2 (rysunek 1.4) — program nie będzie informował o błędzie, nawet jeśli utworzymy zmienną, której nazwa jest taka sama jak nazwa funkcji. Przypisanie `sin=pi` jest poprawne i nie generuje błędu. Takie liberalne podejście do nazw może skutkować dziwnymi komunikatami o błędach (łatwiejszy przypadek) lub przedziwnymi wynikami obliczeń, jeżeli uda nam się utworzyć funkcję, która wykorzystywana jest przez inne funkcje. Polecenie `edit nazwa_funkcji` otwiera edytor MATLAB-a z wczytaną funkcją.

#### Rysunek 1.4.

*Zmiana wartości predefiniowanej zmiennej pi i powrót do predefiniowanej wartości*

```

File Edit Debug Parallel Desktop Window Help
>> pi=2;sin(pi)

ans =

    0.9093

>> clear pi
>> sin(pi)

ans =

    1.2246e-016

>>
  
```

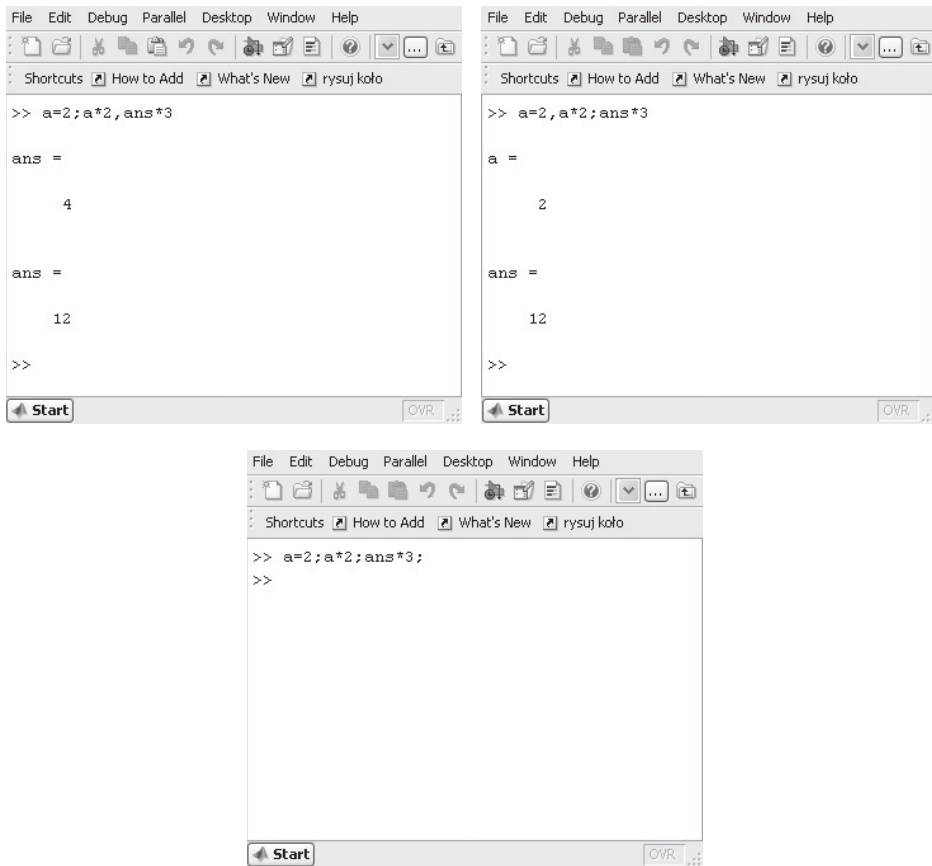
Za pomocą polecenia `clear` można usunąć zmienną lub przywrócić zmienne predefiniowane.

- ♦ `clear all` — *usuwa wszystkie zmienne z pamięci i przywraca zmienne predefiniowane*.
- ♦ `clear nazwa` — *usuwa zmienną nazwa*.

Poleceniem `who` możemy zapytać MATLAB-a o zdefiniowane zmienne, polecenie `whos` podaje zaś listę zdefiniowanych zmiennych, rozmiar i typ przypisanych wartości.

<sup>1</sup> Tak naprawdę są to funkcje, których wynikiem jest określona wartość stałej.

W jednym wierszu polecenia możemy wpisać kilka działań rozdzielonych przecinkiem lub średnikiem (rysunek 1.5). Działanie zakończone średnikiem wykonywane jest bez echa, tzn. jeżeli w poleceniu nie ma błędu i nic nie jest wypisywane w oknie poleceń, kursor tekstowy przenoszony jest do nowego wiersza. Działanie zakończone przecinkiem wykonywane jest z wypisywaniem wyniku lub komunikatu o wykonanym działaniu.



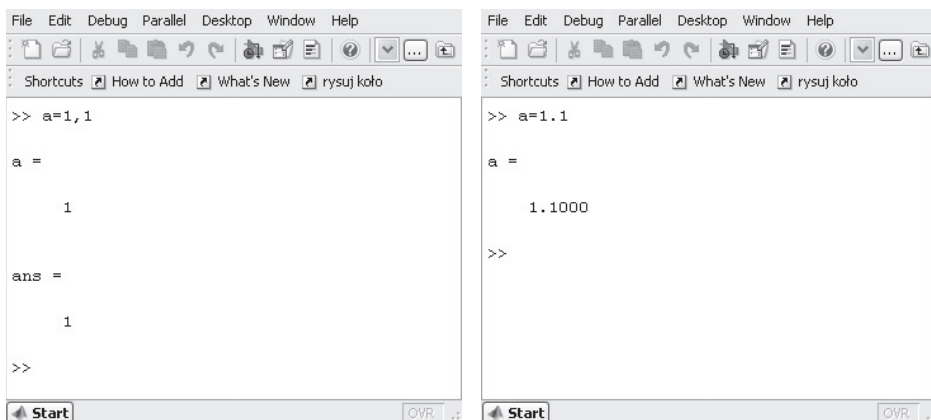
**Rysunek 1.5.** Trzy polecenia w jednym wierszu poleceń rozdzielone przecinkiem lub średnikiem

Przecinek rozdziela polecenia, liczby, ale nigdy nie rozdziela ułamka dziesiętnego od liczby całkowitej. **Liczby rzeczywiste wprowadzamy zawsze z kropką**, niezależnie od ustawień systemu.

Jak widzimy na rysunku 1.6, polecenie `a=1,1` interpretowane jest przez MATLAB-a jako dwa działania. Pierwsze to przypisanie `a=1`, drugie działanie to przypisanie do zmiennej `ans` wartości 1.

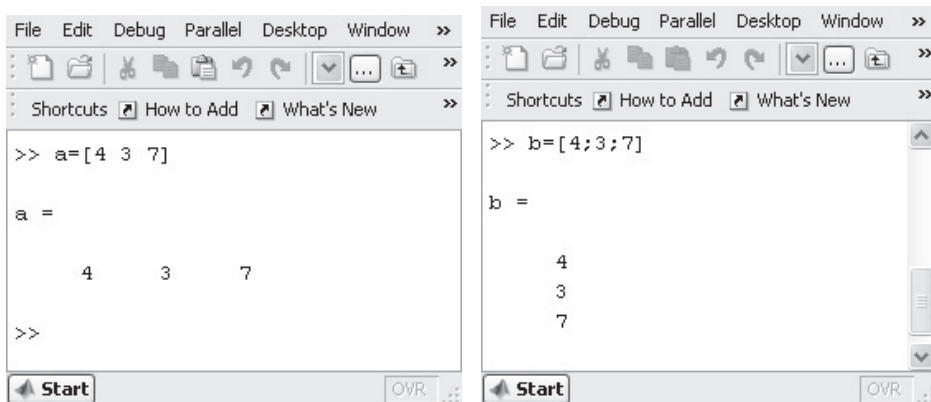
Podstawową strukturą danych w MATLAB-ie jest macierz (*MATRIX*). Nawet zmienna z jedną wartością też jest macierzą, co możemy sprawdzić, pisząc: `a(1)`.





Rysunek 1.6. Ilustracja funkcji przecinka w wierszu poleceń

Liczby (wyrażenia) macierzy wpisujemy w nawiasach kwadratowych. Kolumny rozdzielamy *Spacją* lub przecinkiem, a wiersze średnikiem lub *Enterem*. Jeżeli polecenia nie zakończymy średnikiem, po wpisaniu wektora lub macierzy MATLAB wypisze, jak zinterpretował nasze polecenie (rysunki 1.7 i 1.8).



Rysunek 1.7. Definicja macierzy jednowierszowej i jednokolumnowej

Macierz  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$  należy wprowadzić tak jak na rysunku 1.8. Jeżeli wpisalibyśmy różną ilość liczb w wierszach, odpowiedzią programu będzie informacja o błędzie.

Macierze indeksowane są od 1 i nie można tego zmienić.

Jeżeli przypisując wartość do macierzy, podamy indeks elementu macierzy większy niż dotychczasowy rozmiar, MATLAB rozszerzy macierz i uzupełni zerami brakujące elementy (rysunek 1.9).

**Rysunek 1.8.**

*Dwa sposoby  
definiowania  
macierzy*

```

File Edit Debug Parallel Desktop Window Help
Shortcuts How to Add What's New rysuj koło

>> A=[1 2 3;4 5 6]

A =

     1     2     3
     4     5     6

>> A=[1,2,3;4,5,6]

A =

     1     2     3
     4     5     6

>>
Start OVR

```

**Rysunek 1.9.**

*Ilustracja  
rozszerzania  
rozmiaru macierzy  
w trakcie przypisania*

```

File Edit Debug Parallel Desktop Window Help
a(4) = 2
a(3,1) = 3

>> a=1

a =

     1

>> a(4)=2

a =

     1     0     0     2

>> a(3,1)=3

a =

     1     0     0     2
     0     0     0     0
     3     0     0     0

>>
Start OVR

```

Próba odwołania się do nieistniejącego elementu macierzy spowoduje błąd.

Macierze możemy dodawać, odejmować, mnożyć, dzielić zgodnie z zasadami rachunku macierzowego. Dodawanie macierzy zapisujemy tak jak zmienne z jedną wartością, co ilustruje rysunek 1.10. Próba pomnożenia macierzy  $A * A$  spowoduje wypisanie komunikatu o niezgodności liczby wierszy i kolumn w drugiej macierzy.

**Rysunek 1.10.**

Operacje na macierzy  
*A* zdefiniowanej  
na rysunku 1.8

```

File Edit Debug Parallel Desktop Window Help
Shortcuts How to Add What's New rysuj koło

>> A+A

ans =

     2     4     6
     8    10    12

>> 2*A-A

ans =

     1     2     3
     4     5     6

>> |
Start OVR

```

Działania mogą być wykonywane na macierzach, na elementach macierzy i na wartościach (skalarach). Dostępne operatory różnie się zachowują w zależności od tego, na jakich danych ma być wykonana operacja. Podstawowe operatory arytmetyczne MATLAB-a, sposób realizacji operacji arytmetycznej i przykład zastosowania znajdują się w tabeli 1.1.

**Tabela 1.1.** Operatory arytmetyczne i operacje arytmetyczne

Operator	Realizacja operacji arytmetycznej
dodawanie	$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$ $A + c = \begin{bmatrix} a_{11} + c & a_{12} + c \\ a_{21} + c & a_{22} + c \end{bmatrix}$ $d = A + c$ $A = A + 2$
odejmowanie	$A - B = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} \\ a_{21} - b_{21} & a_{22} - b_{22} \end{bmatrix}$ $A - c = \begin{bmatrix} a_{11} - c & a_{12} - c \\ a_{21} - c & a_{22} - c \end{bmatrix}$ $A = A - c$
mnożenie macierzy	$A * B = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{bmatrix}$

**Tabela 1.1.** Operatory arytmetyczne i operacje arytmetyczne — ciąg dalszy

Operator	Realizacja operacji arytmetycznej
mnożenie elementów macierzy (mnożenie tablicowe)	$A .* B = \begin{bmatrix} a_{11} \cdot b_{11} & a_{12} \cdot b_{12} \\ a_{21} \cdot b_{21} & a_{22} \cdot b_{22} \end{bmatrix}$ $d = A * c$ $A = A * 2$
potęgowanie	$A ^ k = \underbrace{A \cdot A \cdot \dots \cdot A}_k$
potęgowanie tablicowe	$A . ^ k = \underbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \dots \cdot \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_k$
dzielenie prawostronne	$A / B = A \cdot B^{-1}$
dzielenie lewostronne	$A \setminus B = A^{-1} \cdot B$
dzielenie prawostronne tablicowe	$A ./ B = \begin{bmatrix} a_{11} / b_{11} & a_{12} / b_{12} \\ a_{21} / b_{21} & a_{22} / b_{22} \end{bmatrix}$
dzielenie lewostronne tablicowe	$A . \setminus B = \begin{bmatrix} b_{11} / a_{11} & b_{12} / a_{12} \\ b_{21} / a_{21} & b_{22} / a_{22} \end{bmatrix}$
transpozycja macierzy	$A' = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$

W tabeli 1.1  $A^{-1}$  jest macierzą odwrotną.

Jeżeli argumentem operatora potęgowania jest liczba ujemna i wykładnik nie jest liczbą całkowitą, wynikiem będzie liczba zespolona (rysunek 1.11).

### Rysunek 1.11.

*Ilustracja kolejności wykonania działań*

```

MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
C:\Documents and ...
>> -27^(1/3)

ans =

    -3

>> (-27)^(1/3)

ans =

    1.5000 + 2.5981i

>>
Start OVR

```

MATLAB wykonuje formułę z uwzględnieniem kolejności wykonywania działań. Proszę zwrócić uwagę na przykład na rysunek 1.11: w pierwszym wierszu minus jednoargumentowy ma niższy priorytet niż potęgowanie, a więc wynik jest zgodny z naszą intuicją ( $-3^3=-27$ ), ale gdybyśmy zrobili przypisanie np.  $a=-27$  i podnieśli do potęgi  $\frac{1}{3}$ , uzyskamy wynik taki jak w drugiej operacji — oczywiście pierwszy i drugi wynik (zmienna `ans`) podniesiony do trzeciej potęgi da wartość  $-27$ . Jeżeli wpisalibyśmy potęgę  $\frac{1}{3}$  bez nawiasów, zostałyby wykonane potęgowanie z wykładnikiem 1, a następnie wynik zostałby podzielony przez 3.

## Ćwiczenie

Chcemy rozwiązać układ równań liniowych:

$$\begin{cases} x_1 + 2x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$$

Układ równań możemy zapisać w postaci macierzowej:

$$A \cdot X = B$$

gdzie:

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Równanie macierzowe mnożymy lewostronnie przez macierz  $A$  odwrotną:

$$A^{-1} \cdot A \cdot X = A^{-1} \cdot B$$

Z definicji:  $A^{-1} \cdot A = I$

a więc mamy:

$$X = A^{-1} \cdot B \quad \text{lub} \quad X = A \setminus B$$

## Operator dwukropka

Program umożliwi nam szybkie generowanie wektorów za pomocą operatora *dwukropka*:

- ♦ `min:krok:max` — generuje wektor wierszowy zawierający liczby od `min` do `max`, a różnica pomiędzy wyrazami wektora wynosi `krok`. Jeżeli pominiemy `krok`, zostanie on przyjęty domyślnie jako 1.
- ♦ `min:max` — generuje wektor zawierający liczby różniące się o 1.

Krok może być większy lub mniejszy od zera; krok mniejszy od zera tworzy ciągi liczb malejących, oczywiście pod warunkiem, że będzie to możliwe. W tabeli 1.2 znajdują się dwa przykłady użycia operatora dwukropka, którego wynikiem jest macierz pusta. Jeżeli  $\min$  i krok są liczbami całkowitymi, to wynikiem jest wektor liczb całkowitych.

**Tabela 1.2.** Przykłady użycia operatora dwukropka

Przykład	Wynik
0:0.5:2	0 0.5000 1.0000 1.5000 2.0000
0:0.6:2	0 0.6000 1.2000 1.8000
1:5	1 2 3 4 5
1.1:5	1.1000 2.1000 3.1000 4.1000
5:1	Empty matrix: 1-by-0
5:-1:1	5 4 3 2 1
1:-1:5	Empty matrix: 1-by-0

Podobne zadanie pełni funkcja `linspace(d0,dk,n)`. Funkcja ta generuje wektor o  $n$  elementach w przedziale od  $d0$  do  $dk$ . Jeżeli nie podamy trzeciego argumentu funkcji,  $n$  będzie równe  $100$ . Oczywiście funkcję `linspace` możemy zastąpić `:`, co zostało zilustrowane na rysunku 1.12.

```

File Edit Debug Parallel Desktop Window Help
>> 0:0.5:4

ans =

    0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000

>> linspace(0,4,9)

ans =

    0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
  
```

**Rysunek 1.12.** Porównanie działania operatora `:` i funkcji `linspace`

Funkcja `logspace(d0,dk,n)` generuje wektor o  $n$  elementach rozmieszczonych logarytmicznie pomiędzy liczbami  $10^{d0}$  a  $10^{dk}$ . Funkcję `logspace` możemy zastąpić poleceniem: `10.^linspace(d0,dk,n)`.

## Dostęp do elementów macierzy

Do elementów macierzy odwołujemy się, pisząc *nazwę zmiennej* i w nawiasach podając indeksy elementów macierzy. Jeżeli macierz  $A$  jest wektorem, to odwołanie  $A(i)$  jest odwołaniem do  $i$ -tego elementu wektora, a jeżeli macierz  $A$  jest macierzą dwu-

wymiarową, odwołanie  $A(i,j)$  jest odwołaniem do  $i$ -tego wiersza i  $j$ -tej kolumny, odwołanie zaś do macierzy dwuwymiarowej z wykorzystaniem jednego indeksu  $A(i)$  oznacza odwołanie do wektora utworzonego z kolejnych kolumn pierwotnej macierzy.

Mając zdefiniowaną macierz  $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

```
>> A(1) → ans=1
>> A(2) → ans=4
>> A(3) → ans=2
>> A(1,1) → ans=1
>> A(1,[1 2]) → ans=[1 2]
>> A(1,[1 2 3]) → ans=[1 2 3]
```

zamiast pisać wektor  $[1 \ 2 \ 3]$ , możemy wpisać  $1:3$

```
>> A(1,1:3) → ans=[1 2 3]
>> A(1,:) → ans [1 2 3] %pierwszy wiersz
>> A(1,3:-1:1) → ans=[3 2 1] %pierwszy wiersz wypisany w kolejności odwrotnej
>> A(:,1) → ans [1;4] %pierwsza kolumna
>> A(:,[2 1 3]) → ans  $\begin{bmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \end{bmatrix}$  %zamiana drugiej i pierwszej kolumny
```

## Funkcje MATLAB-a generujące tablice

Podstawową strukturą danych w MATLAB-ie jest macierz. Zdefiniowanych jest wiele funkcji, które ułatwiają pracę z programem:

- ♦ `zeros(n)` — generuje macierz  $n \times n$  składającą się z samych zer.
- ♦ `ones(n)` — generuje macierz  $n \times n$  składającą się z samych jedynek.
- ♦ `eye(n)` — generuje macierz jednostkową, jedynki są na przekątnej macierzy  $n \times n$ .
- ♦ `rand(n)` — generuje macierz  $n \times n$  o elementach będącymi liczbami pseudolosowymi o rozkładzie równomiernym z przedziału  $(0,1)$ .
- ♦ `randn(n)` — generuje macierz  $n \times n$  o elementach będącymi liczbami pseudolosowymi o rozkładzie normalnym.

Funkcje te mogą być również wywołane z dwoma argumentami, np. poleceniem `ones(n,m)` utworzymy macierz prostokątną o  $n$  wierszach i  $m$  kolumnach. Wymiar macierzy może być dowolny. Wystarczy, że wpiszemy w funkcji odpowiednią liczbę argumentów: 3 to macierz trójwymiarowa, funkcja z 5 argumentami utworzy natomiast macierz pięciowymiarową.

- ♦ `magic(n)` — kwadrat magiczny; jest to macierz, w której sumy elementów wierszy, kolumn i przekątnych wynoszą tyle samo, ponadto elementy macierzy nie powtarzają się.

Kwadrat magiczny o wymiarze 4 utworzony przez MATLAB-a to:

$$a = \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

Jeżeli zamienimy kolumnę drugą z trzecią, kwadrat magiczny będzie taki sam jak na miedziorycie *Melancholia* Dürera nad skrzydłem anioła.

Zamianę kolumn wykonamy poleceniem:

```
a(:, [2 3])=a(:, [3 2])
```

Środkowe liczby w ostatnim wierszu (po zamianie 15 i 14) to rok, w którym powstał miedzioryt.

Macierz magiczna pełni głównie funkcję rozrywkową, ale możemy wykorzystać ją do testowania funkcji MATLAB-a lub własnych. Macierz magiczna zawsze ma wyznacznik, a układ równań ze współczynnikami macierzy magicznej z prawą stroną ma rozwiązanie. Przykład wykorzystania funkcji `magic` to test rozwiązania układu równań:

```
>> n=11; A=magic(n);B=ones(n,1);A\B
```

Inny przykład:

```
>> n=11; A=magic(n);B=(1:n)';A^-1*B
```

Sprawdź, jakie są wyniki dla  $n$  parzystego i nieparzystego.

- ◆ `reshape(A,m,n)` — zmiana rozmiaru macierzy  $A$  na macierz o  $m$  wierszach i  $n$  kolumnach.

Polecenie `reshape(a,16,1)` zamieni macierz magiczną na wektor kolumnowy, a polecenie `reshape(a,1,16)` — na wektor jednowierszowy. Jeżeli zamiast jednego z argumentów (liczba wierszy lub liczba kolumn) wpisujemy [], MATLAB sam wyliczy, ile ma być kolumn lub wierszy. Funkcja może zmieniać rozmiary macierzy, ale musi się zgadzać liczba elementów, a więc wpisanie np. liczby 5 jako liczba wierszy lub kolumn spowoduje błąd funkcji.

- ◆ `size(A,m)` — liczba wierszy i kolumn macierzy  $A$ .

Jeżeli pominiemy  $m$ , wynikiem działania funkcji `size` będzie wektor jednowierszowy o długości równej wymiarowi macierzy. Macierz jednowymiarowa (wektor) traktowana jest jako macierz o jednym wierszu lub jednej kolumnie. Drugim argumentem jest wymiar, którego rozmiar chcemy sprawdzić: 1 to wiersze, 2 to kolumny, 3 to liczba macierzy prostokątnych itd.



# Funkcje

W MATLAB-ie jest ogromne bogactwo funkcji: są funkcje trygonometryczne, logarytmiczne, statystyczne, a także wiele specjalizowanych funkcji. Drobny przykład funkcji wbudowanych przedstawiono w tabeli 1.3.

**Tabela 1.3.** Przykłady funkcji wbudowanych

Funkcja	Opis
sqrt	pierwiastek
abs	wartość bezwzględna
exp	$e^x$
log	logarytm naturalny
log10	logarytm dziesiętny
log2	logarytm o podstawie 2
sign	znak liczby — wartość +1 lub -1
sin, cos, tan, cot	funkcje trygonometryczne (argumentem jest kąt w radianach)
sind, cosd, tand, cotd	funkcje trygonometryczne (argumentem jest kąt w stopniach)
gcd	najmniejszy wspólny dzielnik $gcd(9,12) \rightarrow 3$
lcm	najmniejsza wspólna wielokrotność $lcm(9,12) \rightarrow 36$

Liczba funkcji zależy od liczby zainstalowanych pakietów funkcji, które nazywane są toolboksami. W czasie instalacji programu MATLAB możemy zdecydować, jakie pakiety chcemy zainstalować.

Listę wbudowanych funkcji elementarnych uzyskamy, pisząc:

```
>> help elfun
```

Cechami wspólnymi wszystkich funkcji są: zapis małymi literami i sposób wywołania. Przykład wywołania funkcji z jednym argumentem przedstawiono na rysunku 1.13.

```
>> zmienna=nazwa_funkcji(argumenty, funkcji, rozdzielone, przecinkami)
```

## Rysunek 1.13.

Przykład wywołania funkcji

```
File Edit Debug Parallel Desktop Window Help
C:\Document
>> log10(realmax)
ans =
    308.2547
>> log2(realmax)
ans =
    1024
>>
```

W łatwy sposób możemy utworzyć własne funkcje — przykładem niech będzie logarytm o dowolnej podstawie:

$$\log_a x = \frac{\log_b(x)}{\log_b(a)}$$

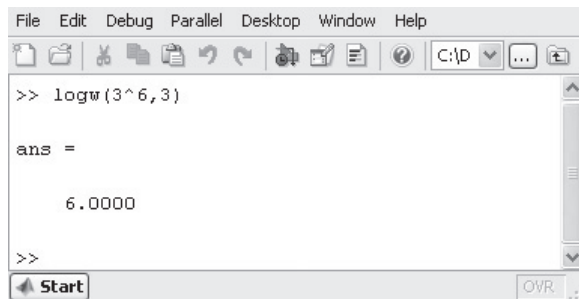
W oknie MATLAB-a z menu wybieramy *File/New/M-file*, do którego wpisujemy:

```
function c=logw(x,a)
c=log(x)/log(a);
```

Plik należy zapisać pod taką samą nazwą jak nazwa funkcji, ponadto deklaracja funkcji musi znaleźć się w pierwszym wierszu pliku. Po wywołaniu funkcji MATLAB przeszukuje ścieżki zawarte w zmiennej środowiskowej PATH. Na liście przeszukiwanych folderów jest folder *Work*, który jest tworzony w katalogu instalacyjnym MATLAB-a lub w folderze *Moje dokumenty*, i tam możemy zapisać nasz plik z funkcją. Po zapisaniu funkcji w odpowiednim katalogu wywołanie funkcji odbywa się tak jak wywołanie każdej innej (rysunek 1.14).

#### Rysunek 1.14.

*Test własnej funkcji*



```
File Edit Debug Parallel Desktop Window Help
C:\D
>> logw(3^6,3)

ans =

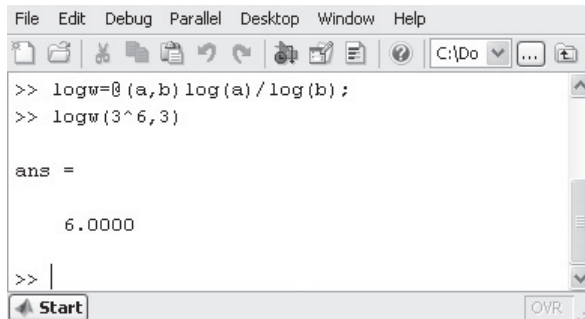
    6.0000

>>
```

Nie jest to jedyny sposób tworzenia własnych funkcji. Proste jednopoleceniowe funkcje (*Help* MATLAB-a określa je jako *Anonymous Functions*) możemy utworzyć za pomocą operatora  $@$ . Funkcję anonimową wywołuje się tak samo jak inne funkcje (rysunek 1.15).

#### Rysunek 1.15.

*Definicja i wywołanie funkcji anonimowej*



```
File Edit Debug Parallel Desktop Window Help
C:\Do
>> logw=@(a,b) log(a)/log(b);
>> logw(3^6,3)

ans =

    6.0000

>> |
```

# Wykresy

Do kreślenia wykresów służą funkcje: `plot`, `line`, `semilogx`, `semilogy`, `loglog`, `bar`, `bar3` i inne. Argumentami funkcji są wektory danych i (opcjonalnie) format wykresów.

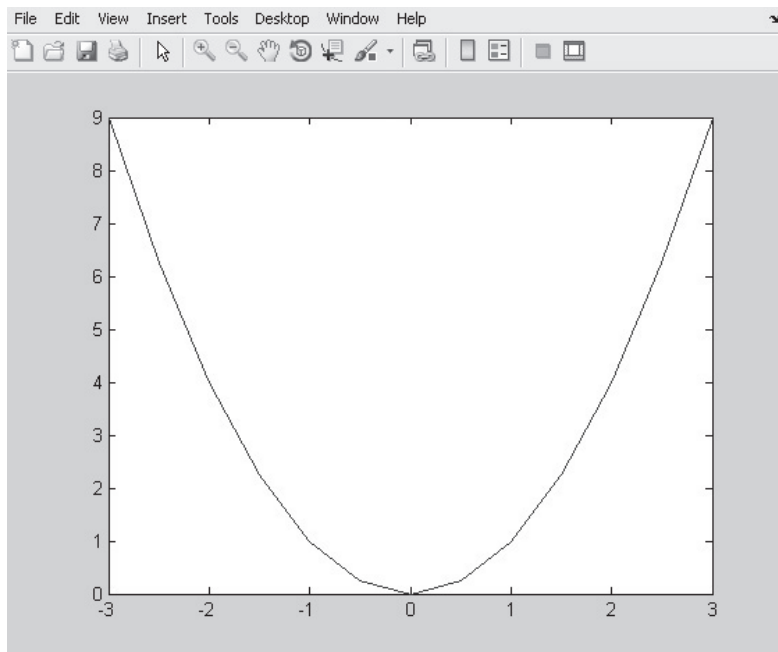
Poleceniem:

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

utworzymy wykres paraboli (rysunek 1.16).

**Rysunek 1.16.**

*Okno wykresu*



Składnia funkcji `plot`:

```
plot(Y)
plot(X1,Y1,X2,Y2...)
plot(X1,Y1,LineStyle,X2,Y2,LineStyle)
plot(X1,Y1,'PropertyName',PropertyValue)
plot(axes_handle,...)
```

gdzie:

`LineStyle` to napis określający format wykresu.

Mamy możliwość formatowania koloru linii, rodzaju linii i znacznika punktu.

Oznaczenia formatu wykresu (na podstawie pomocy MATLAB-a):

B	niebieski	.	point	-	solid
G	zielony	0	circle	:	dotted
R	czerwony	X	x-mark	-.	dashdot
C	cyan	+	plus	--	dashed
M	magenta	*	star	(none)	no line
Y	żółty	S	square		
K	czarny	D	diamond		
W	biały	V	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		P	pentagram		
		H	hexagram		

Poleceniem z rysunku 1.17 utworzymy trzy wykresy w jednym oknie (rysunek 1.18).

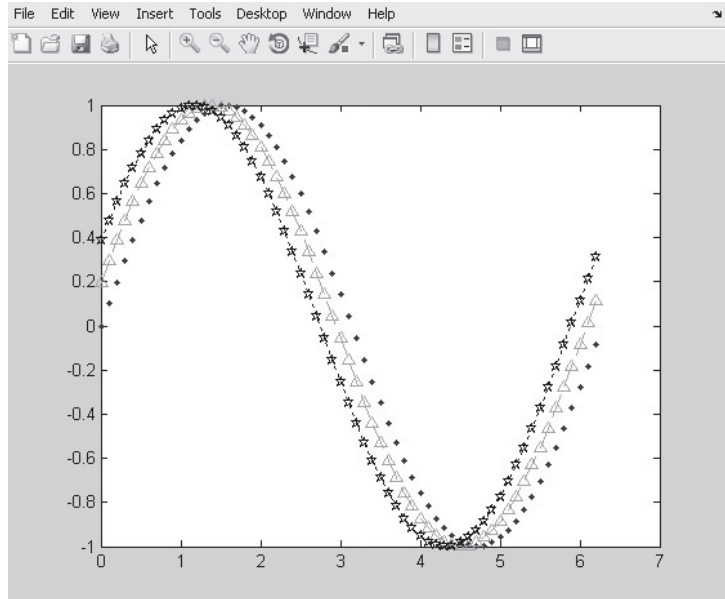
#### Rysunek 1.17.

*Polecenia tworzące wykresy na rysunku 1.18*

```
File Edit Debug Parallel Desktop Window Help
C:\Documents and Settings\...
>> t=0:0.1:2*pi;y=sin(t);y2=sin(t+0.2);y3=sin(t+0.4);
>> plot(t,y,'.',t,y2,'--^g',t,y3,'pk');
>>
```

#### Rysunek 1.18.

*Wykresy utworzone poleceniami z rysunku 1.17*



## Ćwiczenie

Chcemy narysować okrąg o promieniu 2 i początku w układzie współrzędnych.

### Rozwiązanie:

Równanie okręgu we współrzędnych kartezjańskich można zapisać jako:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

lub jako równanie parametryczne:

$$\begin{cases} x = x_0 + r \cdot \cos \alpha \\ y = y_0 + r \cdot \sin \alpha \end{cases}$$

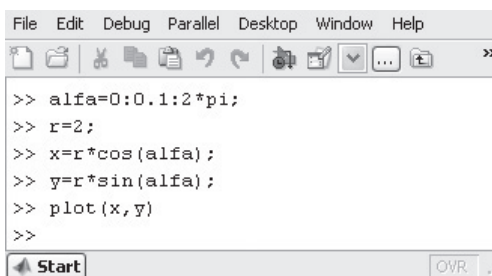
gdzie:

$$\alpha \in (0, 2\pi)$$

Wygodniej narysować okrąg, korzystając z równania parametrycznego. Lista poleceń, którymi narysujemy okrąg, może być taka jak na rysunku 1.19.

### Rysunek 1.19.

*Ciąg poleceń, którymi narysujemy okrąg*



```
File Edit Debug Parallel Desktop Window Help
>> alfa=0:0.1:2*pi;
>> r=2;
>> x=r*cos(alfa);
>> y=r*sin(alfa);
>> plot(x,y)
>>
```

Na rysunku 1.20 widać, że okrąg bardziej przypomina elipsę, a to dlatego, że MATLAB próbuje przeskalować wykres tak, żeby zajmował całą przestrzeń okna wykresu. Poleceniem `axis equal` spowodujemy, że osie  $x$  i  $y$  będą w tej samej skali, w wyniku czego uzyskamy śliczny okrąg (rysunek 1.21). Okrąg na rysunku 1.21 jest niepełny, przy kącie zerowym jest przerwa. Zaproponuj, jak zmienić ciąg instrukcji 1.19, żeby pozbyć się tej dziwnej przerwy.

Oczywiście możemy narysować okrąg, wykorzystując równanie:

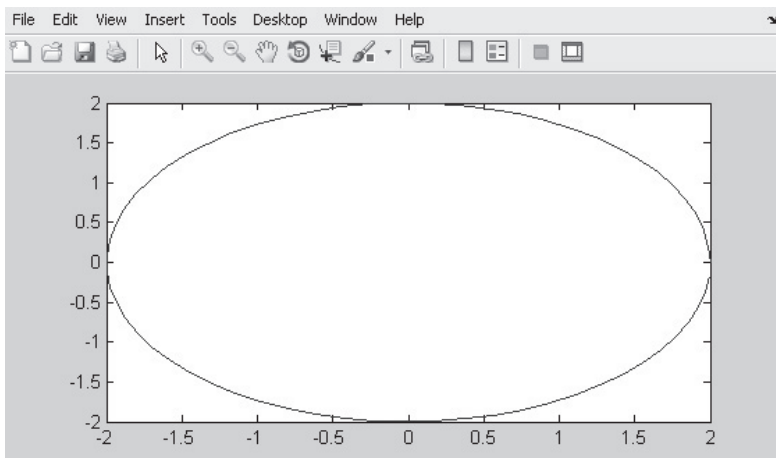
$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

Przyjmując, że rysujemy okrąg o środku w punkcie  $(0,0)$ , uzyskamy prostsze równanie:

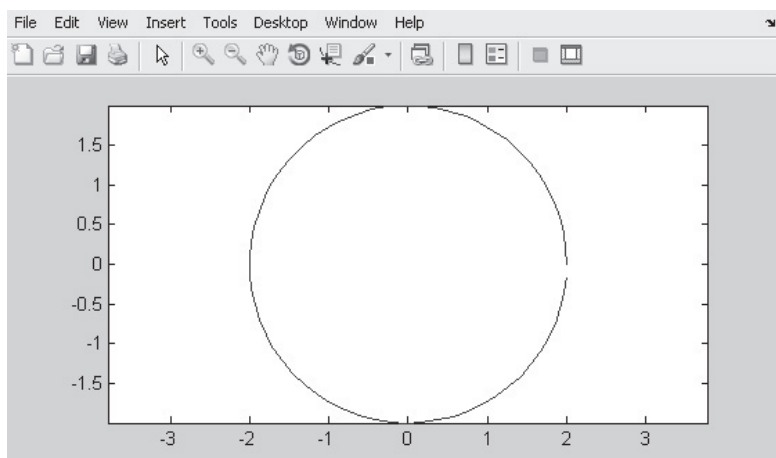
$$x^2 + y^2 = r^2$$

**Rysunek 1.20.**

Okno wykresu z narysowanym okręgiem

**Rysunek 1.21.**

Okno wykresu z narysowanym okręgiem i z równą skalą osi



Promień  $r$  jest zadany i jest wartością stałą. Przyjmijmy, że  $x$  jest zmienną niezależną, i wyznaczmy z równania  $y$ .

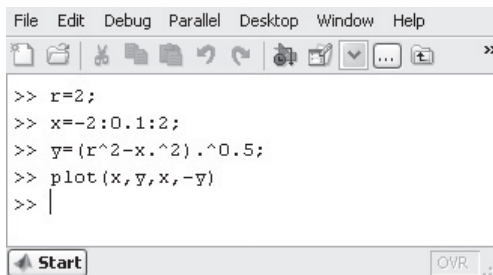
$$y = \pm\sqrt{r^2 - x^2}$$

Jak widzimy, tak naprawdę są to dwie funkcje: górna połowa okręgu  $y = \sqrt{r^2 - x^2}$  i dolna połowa okręgu  $y = -\sqrt{r^2 - x^2}$ .

W poleceniach z rysunku 1.22 należy zwrócić szczególną uwagę na potęgowanie do kwadratu i pierwiastka. Proszę zauważyć, że w tym przypadku rysujemy dwa wykresy: górną i dolną połowę okręgu (rysunek 1.23).

**Rysunek 1.22.**

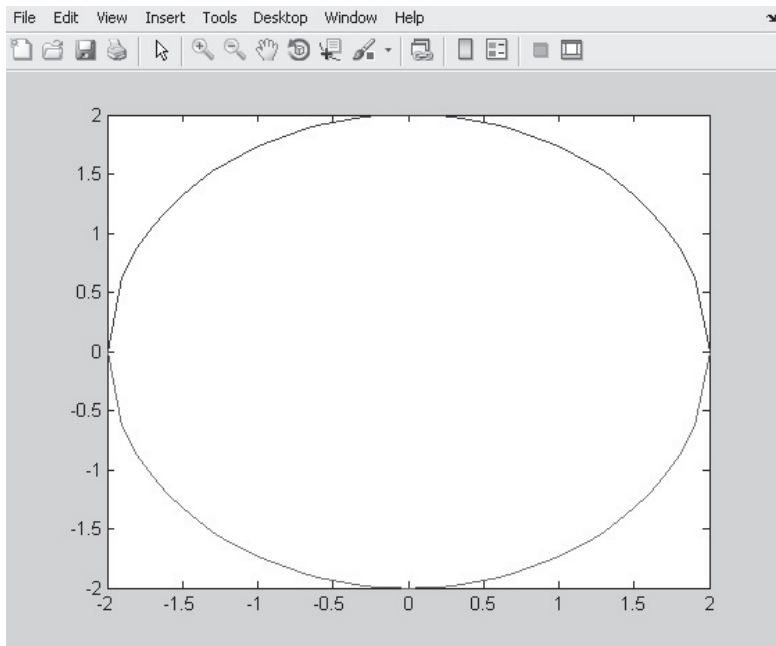
*Ciąg instrukcji rysujących okrąg*



```
File Edit Debug Parallel Desktop Window Help
>> r=2;
>> x=-2:0.1:2;
>> y=(r^2-x.^2).^0.5;
>> plot(x,y,x,-y)
>> |
Start OVR
```

**Rysunek 1.23.**

*Efekt wykonania ciągu instrukcji z rysunku 1.22*

**Zadania dla czytelnika**

Sprawdź, czy rozumiałeś rysowanie wykresów, rysując:

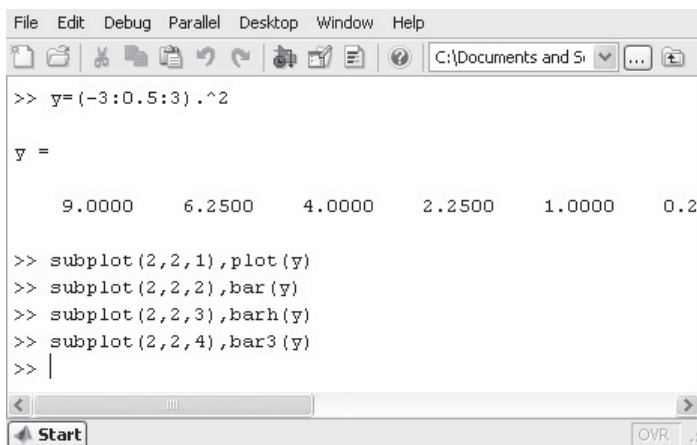
- ♦ elipsę,
- ♦ spiralę Archimedesa,
- ♦ hipotrochoidę.

Równania możesz znaleźć na stronach Wikipedii.

Za pomocą funkcji `subplot` mamy możliwość podzielenia okna i narysowania wykresów obok siebie. Poleceniami z rysunku 1.24 utworzymy cztery wykresy w jednym oknie (rysunek 1.25).

**Rysunek 1.24.**

Polecenia tworzące  
wykresy w jednym oknie



```
File Edit Debug Parallel Desktop Window Help
C:\Documents and S...
>> y = (-3:0.5:3) .^2

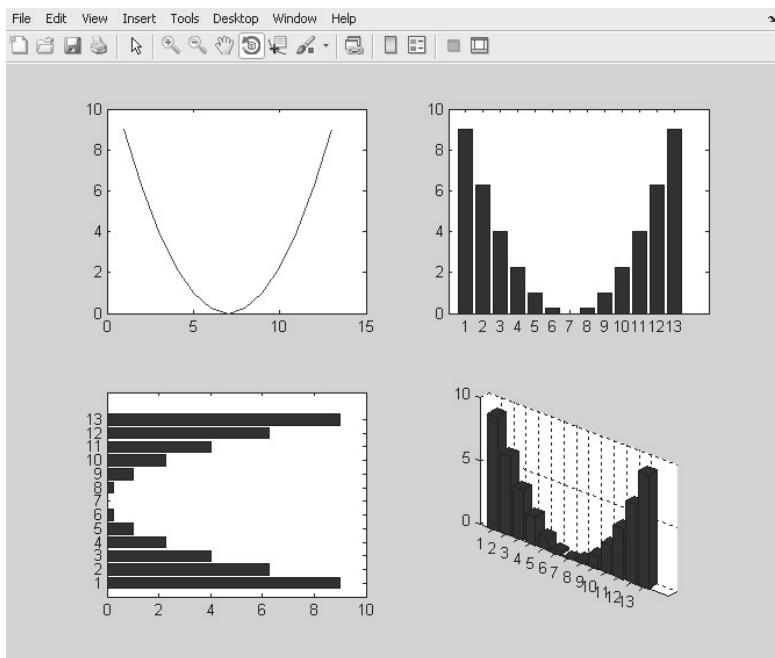
y =

    9.0000    6.2500    4.0000    2.2500    1.0000    0.2500

>> subplot(2,2,1),plot(y)
>> subplot(2,2,2),bar(y)
>> subplot(2,2,3),barh(y)
>> subplot(2,2,4),bar3(y)
>> |
```

**Rysunek 1.25.**

Wykresy paraboli  
narysowane czterema  
różnymi funkcjami



Funkcja `subplot` wymaga podania trzech argumentów `subplot(m,n,p)` lub `subplot(mnp)`. Znaczenie tych argumentów jest następujące:

$m$  — liczba wykresów w poziomie,

$n$  — liczba wykresów w pionie,

$p$  — numer wykresu.



Widzimy, że na osi poziomej jest numer punktu. Możemy to zmienić, podając dwa wektory: pierwszy z wartościami punktów na osi poziomej, drugi z wartościami punktów na osi pionowej.

Standardowe funkcje, za pomocą których możemy dodawać opisy wykresów i modyfikować osie, zawiera tabela 1.4. Opis ciekawszych efektów, które możemy uzyskać, znajduje się w rozdziale 7.

**Tabela 1.4.** *Funkcje wspomagające rysowanie wykresów*

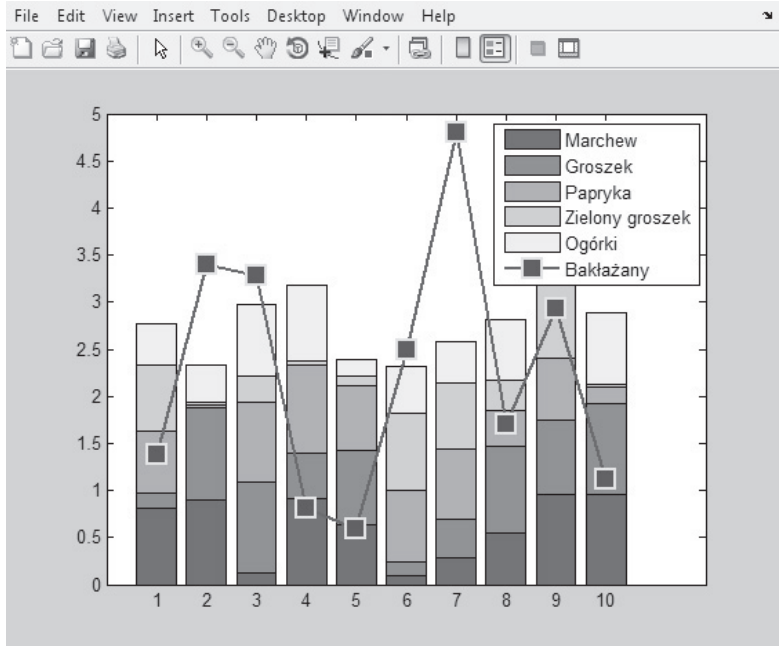
<b>Funkcja</b>	<b>Opis</b>
<code>xlabel('tekst')</code>	dodaje opis osi poziomej
<code>xlabel('tekst','Property1',PropertyValue1,...)</code>	
<code>ylabel('tekst')</code>	dodaje opis osi pionowej
<code>ylabel('tekst','Property1',PropertyValue1,...)</code>	
<code>title('tekst')</code>	dodaje tytuł wykresu
<code>title('tekst','Property1',PropertyValue1,...)</code>	
<code>text(X,Y,'napis')</code>	dodaje tekst z początkiem w punkcie $X,Y$ .
<code>text(X,Y,'napis','Property1',PropertyValue1,...)</code>	Funkcja <code>text</code> do formatowania napisu używa składni TeX.
<code>axis</code>	kontrola osi; umożliwia kontrolę osi
<code>axis on</code>	włącza osie
<code>axis off</code>	wyłącza osie
<code>axis([xmin xmax ymin ymax])</code>	określa wartości ekstremalne osi
<code>axis equal</code>	powoduje, że osie są w tej samej skali
<code>axis auto</code>	przywołuje domyślny sposób zachowania osi, wartości ekstremalne na podstawie danych itd.
<code>legend('napis1','napis2',...)</code>	dodaje legendę do wykresu (napisów nie może być więcej niż wykresów)
<code>hold on</code>	włącza/wyłącza czyszczenie wykresu; <code>hold off</code> (wartość domyślna) — wywołanie dowolnej funkcji rysującej wykres powoduje wyczyszczenie obszaru roboczego i narysowanie nowego wykresu; <code>hold on</code> powoduje, że obszar wykresu nie jest czyszczony i wykresy są nakładane na siebie
<code>hold off</code>	
<code>ishold</code>	informacja o stanie przełącznika <code>hold</code>
<code>grid on off</code>	włącza/wyłącza siatkę wykresu
<code>clf</code>	czyści obszar wykresu i ustawia <code>hold</code> na <code>off</code>
<code>figure(n)</code>	aktywowanie $n$ -tego okna wykresu; ta funkcja jest potrzebna, gdy chcemy mieć więcej niż jedno okno wykresu

Wykresy możemy nakładać na siebie, oczywiście jeżeli MATLAB będzie w stanie to zrobić. Przykład wykresów różnych typów z *Helpa* MATLAB-a (przetłumaczono nazwy warzyw — rysunek 1.26):

```
b = bar(rand(10,5),'stacked'); colormap(summer); hold on
x = plot(1:10,5*rand(10,1),'marker','square',...
        'markersize',12,'markeredgecolor','y',...
        'markerfacecolor',[.6 0 .6],'linestyle',...
        '-','color','r','linewidth',2);
hold off
legend([b,x],'Marchew','Groszek','Papryka','Zielony groszek',...
        'Ogórki','Bakłażany')
```

**Rysunek 1.26.**

*Okno wykresu z nałożonymi różnymi wykresami w jednym oknie*



W przykładzie znacznik został określony za pomocą opcji tekstowych 'marker' i 'square'. Jest to sposób alternatywny do specyfikacji rodzaju linii. Wypisując poszczególne opcje i wartości, mamy więcej pisania kosztem większej elastyczności: możemy zmienić nie tylko kształt znacznika, ale również kolor i wielkość.

Wszystkie opcje elementów wykresu są dostępne przy użyciu polecenia `plottools`. Opis tego narzędzia znajduje się w rozdziale 7.

# Skorowidz

## A

aerofiniszery, 179  
animacje, 141–145  
ans, 9  
argumenty typu fixdt, 202, 203

## B

blok  
  Chart, 195  
  Constant, 95, 96  
  Gain, 94, 201  
  Integrator, 94, 96  
  Mux, 101  
  Relay, 193  
  Scope, 89–94  
  stanu, 214  
  Stop Simulation, 150  
  Switch, 137  
  Transfer Fcn, 191  
  VR Sink, 142–145  
  XY Graph, 169  
błąd całkowania, 63

## C

ciąg Fibonacciego  
  schemat Stateflow, 207, 208, 210  
  wyliczenie z wykorzystaniem stanów, 217, 219  
Command Window, 7

## D

debugger Stateflow, 236, 237  
definiowanie  
  macierzy, 11, 12

  sygnału wejściowego i wyjściowego  
  w Stateflow, 197–200  
  transformaty, 185  
  zmiennej, 8  
deklaracja funkcji, 30  
dwukropek, 16

## E

elementy macierzy, 16  
eps, 9

## F

format  
  metafile, 240  
  wykresu, 22  
funkcje, 30, 31  
  abs, 19  
  addpath, 29  
  anonimowe, 20  
  axis auto, 27  
  axis equal, 27  
  axis off, 27  
  axis on, 27  
  axis, 27  
  cd, 29  
  clf, 27  
  exp, 19  
  eye(n), 17  
  factorial, 38  
  figure(n), 27  
  gcd, 19  
  getframe, 247  
  grid on|off, 27  
  hold off, 27  
  hold on, 27  
  imwrite, 246, 247

## funkcje

ishold, 27  
 jednowierszowe, 32  
 lcm, 19  
 legend, 27  
 linspace, 16  
 log, 19  
 log10, 19  
 log2, 19  
 logspace, 16  
 magic(n), 17  
 margin, 191  
 MATLAB-a generujące tablice, 17, 18  
 mod, 33  
 movie2avi, 85  
 odefun, 41  
 ones(n), 17  
 pdepe, 74, 75, 77  
 plot, 21  
 rand(n), 17  
 randn(n), 17  
 roots, 35  
 saveas, 246  
 savepath, 29  
 set, 242, 243  
 sign, 19  
 size(A,m), 18  
 sqrt, 19  
 step, 185  
 subplot, 25–27  
 switch, 148  
 text, 27  
 tf, 184, 185  
 tic, 59  
 title, 27  
 toc, 59  
 trygonometryczne, 19  
 wbudowane w MATLAB-ie, 19, 20  
 wspomagające rysowanie wykresów, 27  
 xlabel, 27  
 ylabel, 27  
 zeros(n), 17  
 funname, 30

**G**

generowanie wektorów, 15

**I**

ikony na pasku narzędziowym okna Stateflow, 196  
 impulsowy przetwornik elektrohydrauliczny  
 schematy blokowe, 163–166

instrukcja if, 33  
 instrukcje sterujące przebiegiem programu, 32–39

**K**

kolejność wykonania działań, 14  
 kryterium Nyquista, 191  
 kwadrat magiczny, 17, 18

**L**

liczby  
 w Matlabie, 10  
 zespolone, 14

**M**

macierz, 10–12, 14, 17  
 jednowymiarowa, 18  
 magiczna, 18  
 odwrotna, 14  
 MATrix, 10  
 model  
 Lotki-Volterry, 46  
 zawieszenia samochodu, 64  
 dwumasowy (schematy blokowe), 152–156

**N**

nakładanie wykresów na siebie, 28  
 nazwy  
 Stateflow, 197  
 zmiennych, 8

**O**

okno poleceń, 7  
 operacje  
 arytmetyczne, 13, 14  
 na macierzy, 13  
 operatory  
 arytmetyczne, 13, 14  
 dwukropka, 15, 16  
 logiczne, 39  
 łączenia warunków, 40  
 porównania, 39  
 oscylator harmoniczny, 53–55  
 a przekształcenia operatorowe, 186–188  
 schematy blokowe, 102–113  
 oscylator harmoniczny z tłumieniem, 55, 56  
 oscylator harmoniczny z wahadłem  
 matematycznym  
 schematy blokowe, 129–134

**P**

pantograf, 173, 175  
 Parallel Computing Toolbox, 37  
 Partial Differential Equation Toolbox, 77  
 PDETool, 77  
 PDEtoolbox, 77  
 pętla  
   for, 35, 37  
   parfor, 37  
   w Stateflow, 206–219  
   while, 38, 39  
 pi, 9  
 plottool, 240–242  
 podnośnik hydrauliczny  
   schematy blokowe, 118–128  
 polecenie  
   axis equal, 23  
   break, 39  
   clear, 9  
   continue, 39  
   if, 33  
   plottools, 28  
   reshape, 18  
   ver, 29  
   who, 9  
   whos, 9  
 przekładnia hydrostatyczna  
   schematy blokowe, 156–162  
 przekształcenie operatorowe do rozwiązywania  
   równań, 183–194  
 przepompownia  
   model w Stateflow, 223–233

**R**

realmax, 9  
 realmin, 9  
 regulator  
   całkujący, 188–191  
   dwustawny, 192  
     Stateflow, 219–223  
 rozdzielanie działań, 10  
 rozwiązywanie równań różniczkowych, 41–88  
 równania  
   różniczkowe, 41–88  
   van der Pola, 59, 61

**S**

schematy blokowe, 89–134, 135–182  
 Simulink, 89–134, 135–182  
   przekształcenia operatorowe, 191

skalowanie wykresu w Simulink, 92  
 składnia polecenia, 8  
 skrypt z wartościami przypisanymi  
   do zmiennych, 101  
 skrypty, 29, 30  
   a funkcje, 32  
 słowo kluczowe  
   else, 33, 34  
   function, 30  
   global, 47  
 stałe, 9  
 Stateflow, 195–237  
   pętle, 206–219  
   warunek logiczny na przykładzie wartości  
     bezwzględnej, 204–206  
 sterowanie zdarzeniami, 215

**T**

tablica prawdy, 211–213  
   sterująca włączaniem i wyłączaniem pomp  
     w pompowni, 232  
 toolboksy, 29  
 transformata Laplace'a, 183–186  
 truthtable, *Patrz* tablica prawdy  
 tworzenie  
   animacji, 141–145  
   własnych funkcji, 20  
 typ  
   fixdt, 200–202  
   int, 203  
   uint, 203

**U**

uchwyt funkcji, 32  
 układy z połówkowymi stopniami swobody, 135

**V**

Virtual Reality Modelling Language, 142  
 VRML, 142

**W**

warunek, 33, 34  
   wielokrotny, 34, 35  
 warunki brzegowe  
   drugiego rodzaju (Neumanna), 71  
   pierwszego rodzaju (Dirichleta), 71  
   trzeciego rodzaju, 71  
 wektory, 15

wykresy, 21–25  
  konfiguracja zaawansowana, 239–247  
  z dwiema osiami, 244–246  
wymuszenie kinematyczne, 111  
wyzwalacze, 216, 217  
wzmocnienie krytyczne, 189, 190

**Z**

zapisywanie wykresu do pliku, 246, 247  
zderzak hydrauliczny  
  schematy blokowe, 145–152  
zmiennie, 8, 9  
  path, 29  
znak przypisania, 8

# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

# MATLAB

to wielofunkcyjny program do zastosowań naukowych i inżynierskich, wykorzystywany przy zaawansowanych obliczeniach, rozwiązywaniu problemów technicznych i tworzeniu symulacji. Jest świetnym narzędziem, które od kilku dekad pomaga tysiącom matematyków, fizyków i inżynierów. Pozwala w mgnieniu oka rozwiązać skomplikowane równania, prześledzić różne warianty w obrębie jednego schematu czy obliczyć wzajemne zależności pomiędzy elementami projektowanego urządzenia i sprawdzić, jak zmiana jednego z nich wpływa na pozostałe.

Sięgnij po tę książkę, a szybko oswoisz się z MATLAB-em. Dzięki niemu już nigdy nie będziesz musiał obliczać równań „na piechotę” ani doświadczalnie sprawdzać skutków dokonania zmiany w projekcie. Wystarczy, że wprowadzisz do programu właściwe dane, a natychmiast zobaczysz, czy Twój pomysł jest strzałem w dziesiątkę, czy wymaga wielu poprawek. Z tej książki dowiesz się, jak zacząć pracę z programem, jak działają skrypty i funkcje. Odkryjesz, jak tworzyć modele matematyczne i fizyczne urządzeń z użyciem matematyki. Zadania zamieszczone na końcu każdego rozdziału pomogą Ci utrwalić zdobytą wiedzę.

- **MATLAB — szybki start**
- **Skrypty i funkcje**
- **Rozwiązywanie równań różniczkowych**
- **Schematy blokowe (Simulink)**
- **Przekształcenie operatorowe do rozwiązywania układów równań**
- **Stateflow**
- **Zaawansowane konfigurowanie wykresu**

**Projektuj i licz z MATLAB-em!**

sięgnij po **WIĘCEJ**



**KOD KORZYŚCI**

**Helion**

19275 numer katalogowy

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



**0 801 339900**



**0 601 339900**

Sprawdź najnowsze promocje:

🔗 <http://helion.pl/promocje>

Książki najchętniej czytane:

🔗 <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

🔗 <http://helion.pl/nowosci>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

ISBN 978-83-246-8134-1



9 788324 681341

Informatyka w najlepszym wydaniu

cena: 44,90 zł