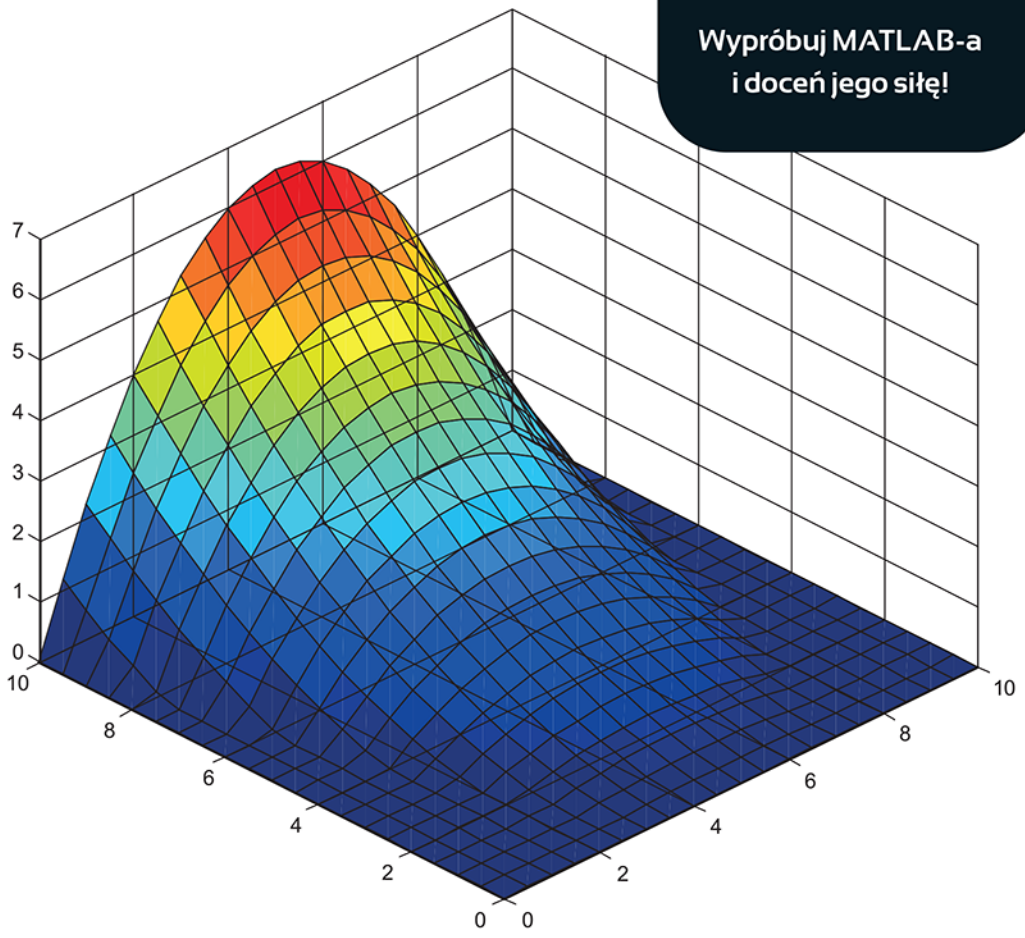
Wypróbuj MATLAB-a
i doceń jego siłę!



WALDEMAR SRADOMSKI

MATLAB

Praktyczny podręcznik modelowania

Helion

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*
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

Printed in Poland.

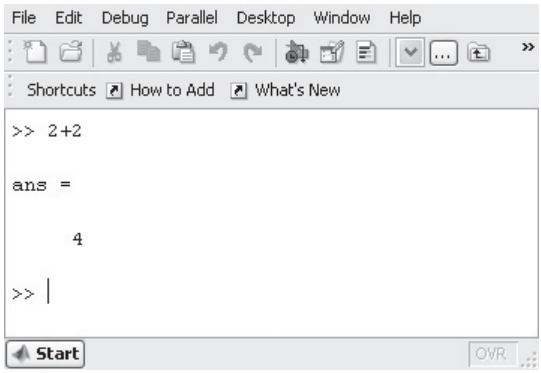
Spis treści

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.

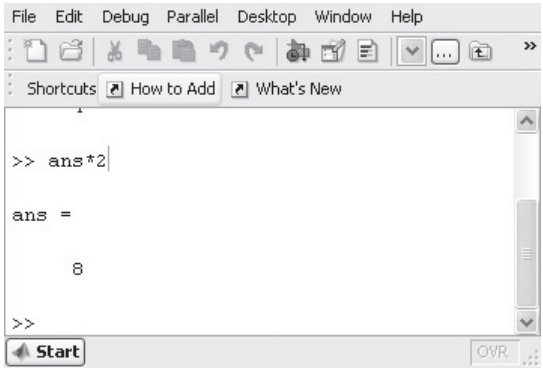
Najważniejsza część interfejsu MATLAB-a to okno poleceń (*Command Window*), za pomocą którego wpisując polecenia, wykonujemy obliczenia i uruchamiamy skrypty. Polecenia wpisujemy po znaku zachęty **>>** (rysunek 1.1). Spróbujmy wpisać coś łatwego, np. `2+2` ↵ (*Enter*).



Rysunek 1.1. *Pierwsze polecenie w oknie Command Window*

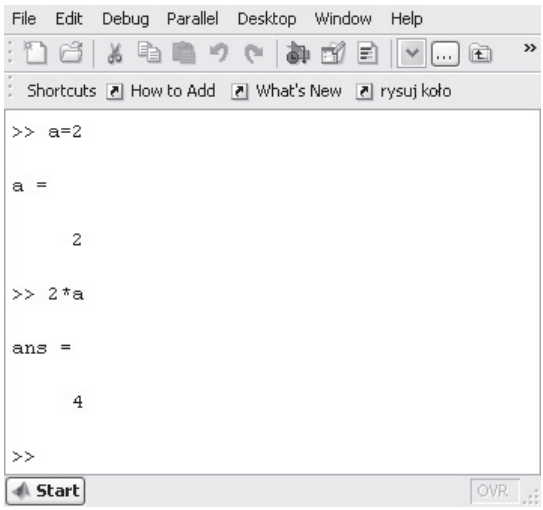
Po naciśnięciu klawisza *Enter* (*Return*) MATLAB wykona polecenie, w tym przypadku sumowanie, i wypisze wynik. Widzimy, że MATLAB w odpowiedzi napisał `ans =` i wynik w wierszu niżej. Zmienna `ans` jest definiowana, gdy wynik wyrażenia nie jest przypisany do żadnej zmiennej. Możemy wykorzystać tę zmienną w następnych poleceniach, np. pomnóżmy 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*



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*



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:

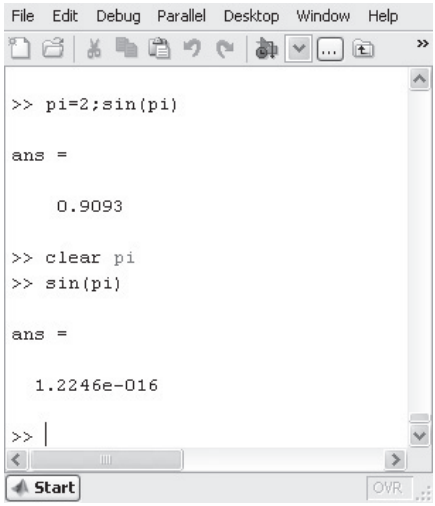
- MATLAB **rozróżnia wielkość liter w nazwach**, a więc `a` i `A` to dwie różne zmienne;
- nie są kontrolowane definiowane nazwy zmiennych, a więc nie będzie żadnego komunikatu, jeżeli utworzymy zmienną, która jest już zdefiniowana.

Przykładowe predefiniowane stałe[1] 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*



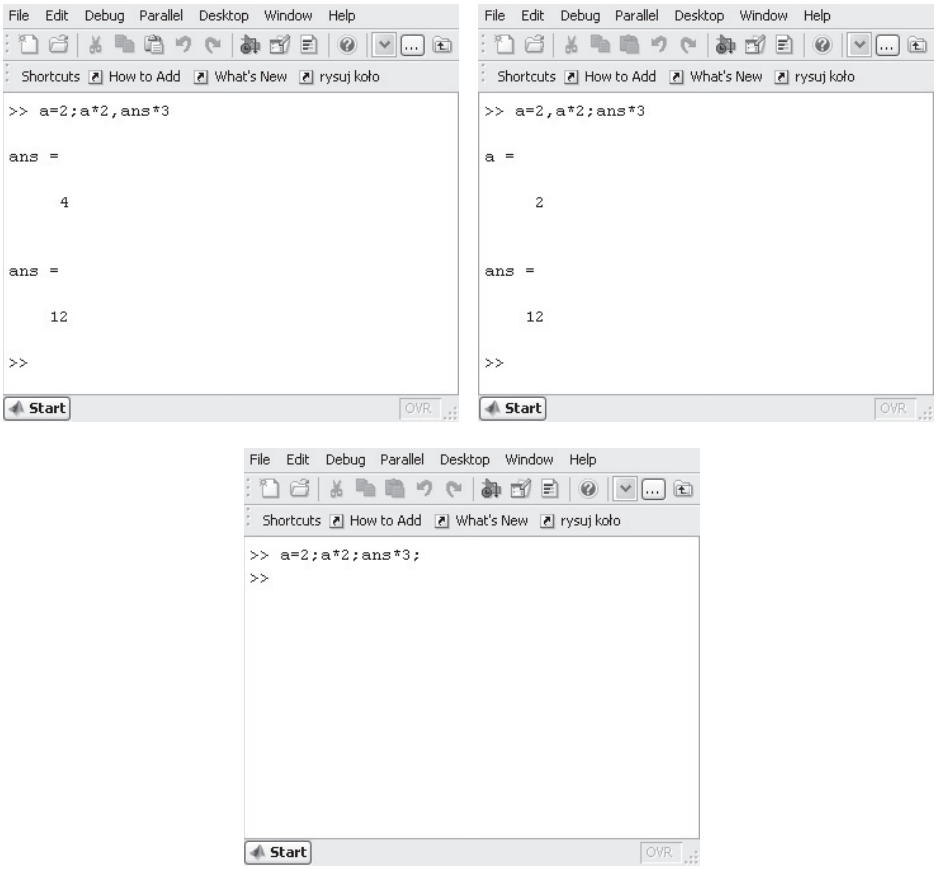
Za pomocą polecenia `clear` można usunąć zmienną lub przywrócić zmienne predefiniowane.

- `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.

[1] 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.

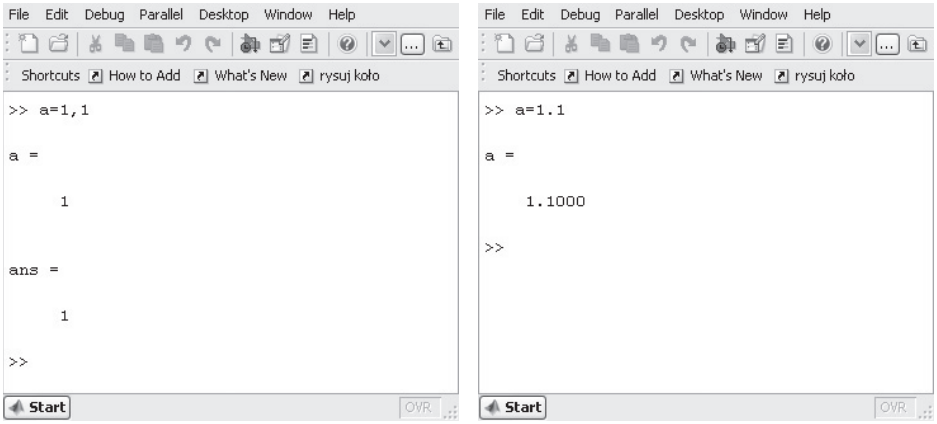


Rysunek 1.5. *Trzy polecenia w jednym wierszu poleceń rozdzielone przecinkiem lub średnikiem*

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

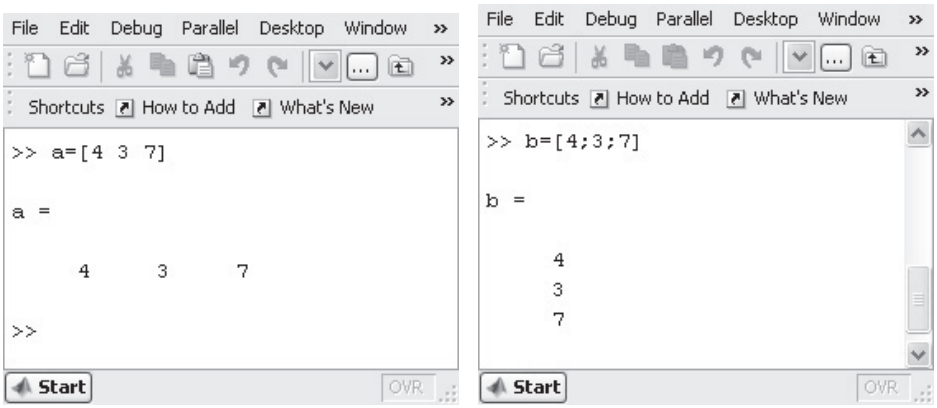
Jak widzimy na rysunku 1.6, polecenie `a=1,1` interpretowane jest przez MATLAB-a jako dwa działania. Pierwsze to przypisanie `a=1`, drugie działanie to przypisanie do zmiennej `ans` wartości 1.

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).

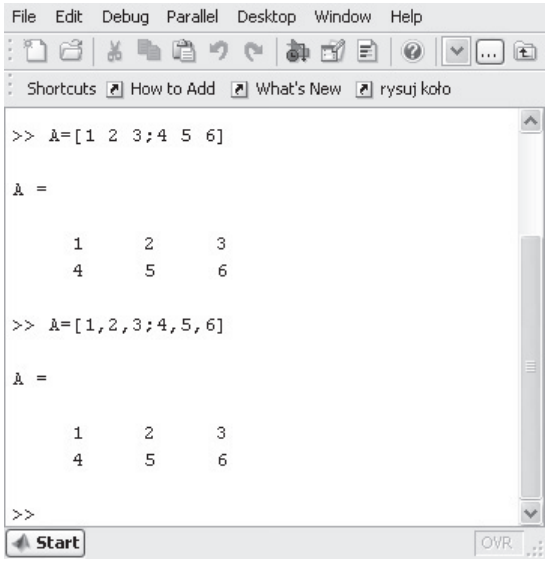


Rysunek 1.7. *Definicja macierzy jednowierszowej i jednokolumnowej*

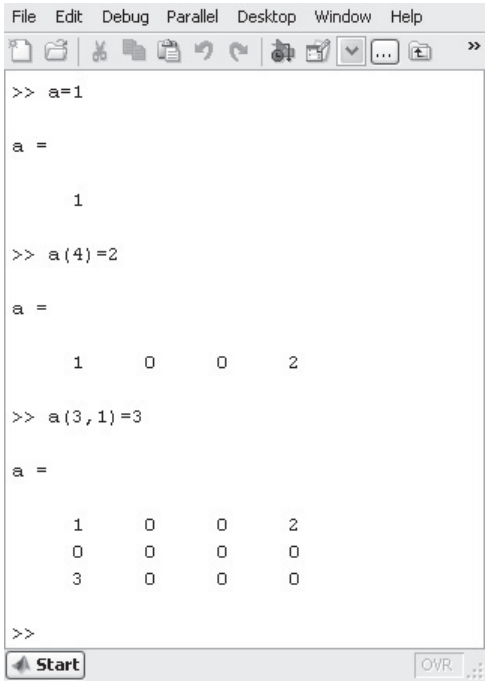
Macierz $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ należy wprowadzić tak jak na rysunku 1.8. Jeżeli wpisalibyśmy różną ilość liczb w wierszach, odpowiedzią programu będzie informacja o błędzie.

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).



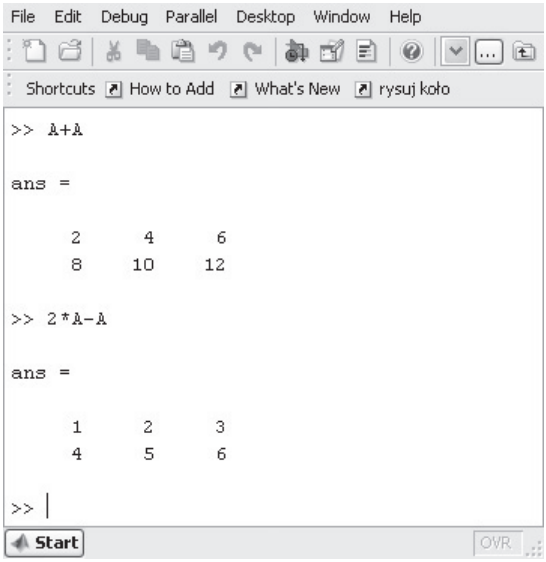
Rysunek 1.8. *Dwa sposoby definiowania macierzy*



Rysunek 1.9. *Ilustracja rozszerzania rozmiaru macierzy w trakcie przypisania*

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*

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*

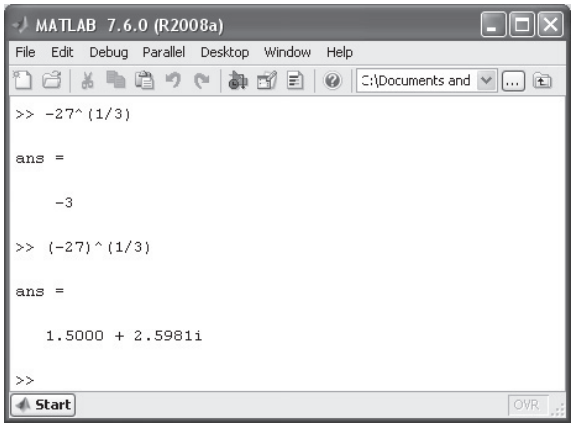
Operator	Realizacja operacji arytmetycznej
dodawanie	$A+B=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \end{bmatrix}$ $A+c=\begin{bmatrix} a_{11}+c & a_{12}+c \\ a_{21}+c & a_{22}+c \end{bmatrix}$ $d=A+c$ $A=A+2$
odejmowanie	$A-B=\begin{bmatrix} a_{11}-b_{11} & a_{12}-b_{12} \\ a_{21}-b_{21} & a_{22}-b_{22} \end{bmatrix}$ $A-c=\begin{bmatrix} a_{11}-c & a_{12}-c \\ a_{21}-c & a_{22}-c \end{bmatrix}$ $A=A-c$
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	$A\wedge k=\underbrace{A\cdot A\cdot\ldots\cdot A}_{k}$
potęgowanie tablicowe	$A.\wedge k=\underbrace{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.\ldots.\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}}_{k}$
dzielenie prawostronne	$A/B=A\cdot B^{-1}$
dzielenie lewostronne	$A\backslash B=A^{-1}\cdot B$
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	$A.\backslash B=\begin{bmatrix} b_{11}/a_{11} & b_{12}/a_{12} \\ b_{21}/a_{21} & b_{22}/a_{22} \end{bmatrix}$
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 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, zostałoby 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:

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

Równanie macierzowe mnożymy lewostronnie przez macierz A odwrotną:

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

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

a więc mamy:

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

Operator dwukropka

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

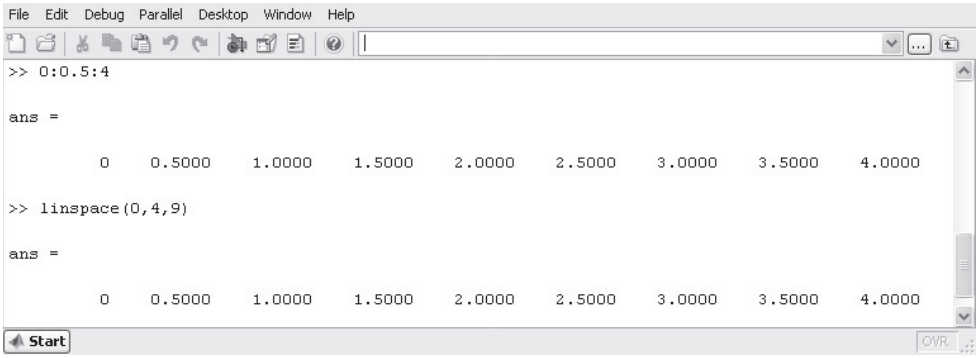
- `min:krok:max` — *generuje wektor wierszowy zawierający liczby od* min *do* max, *a różnica pomiędzy wyrazami wektora wynosi* krok. *Jeżeli pominiemy* krok, *zostanie on przyjęty domyślnie jako* 1.
- `min:max` — *generuje wektor zawierający liczby różniące się o* 1.

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.



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 $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$, 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  %zamiana drugiej i pierwszej kolumny
```

$$\begin{bmatrix} 2 & 1 & 3 \\ 5 & 4 & 6 \end{bmatrix}$$

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×n *składającą się z samych zer*.
- `ones(n)` — *generuje macierz* n×n *składającą się z samych jedynek*.
- `eye(n)` — *generuje macierz jednostkową, jedynki są na przekątnej macierzy* n×n.
- `rand(n)` — *generuje macierz* n×n *o elementach będącymi liczbami pseudolosowymi o rozkładzie równomiernym z przedziału* (0,1).
- `randn(n)` — *generuje macierz* n×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) wpiszemy [], 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 podzielnik $gcd(9,12) \rightarrow 3$
lcm	najmniejsza wspólna wielokrotność $lcm(9,12) \rightarrow 36$

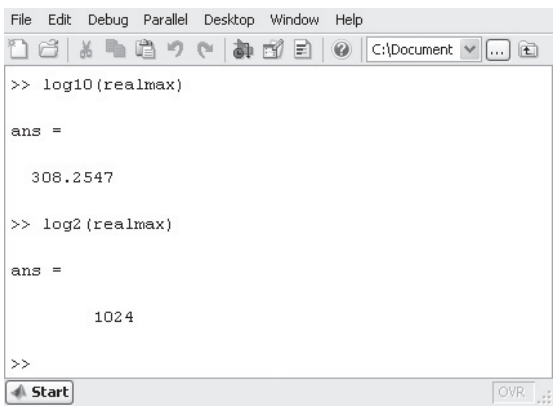
Liczba funkcji zależy od liczby zainstalowanych pakietów funkcji, które nazywane są toolboksami. W czasie instalacji programu MATLAB możemy zdecydować, jakie pakiety chcemy zainstalować.

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*

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

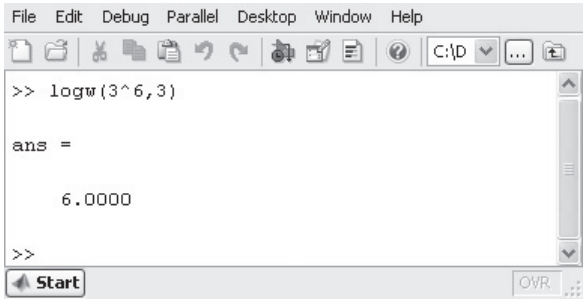
$$\log_a x = \frac{\log_b(x)}{\log_b(a)}$$

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

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

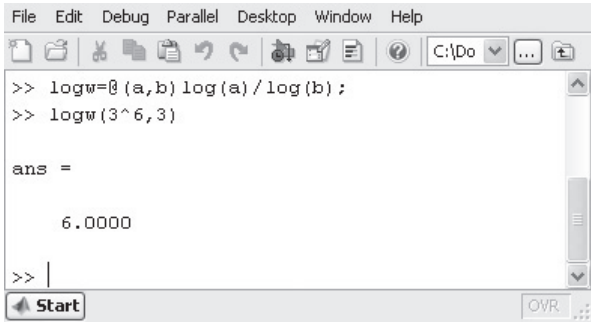
Plik należy zapisać pod taką samą nazwą jak nazwa funkcji, ponadto deklaracja funkcji musi znaleźć się w pierwszym wierszu pliku. Po wywołaniu funkcji MATLAB przeszukuje ścieżki zawarte w zmiennej środowiskowej `PATH`. Na liście przeszukiwanych folderów jest folder *Work*, który jest tworzony w katalogu 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*



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*



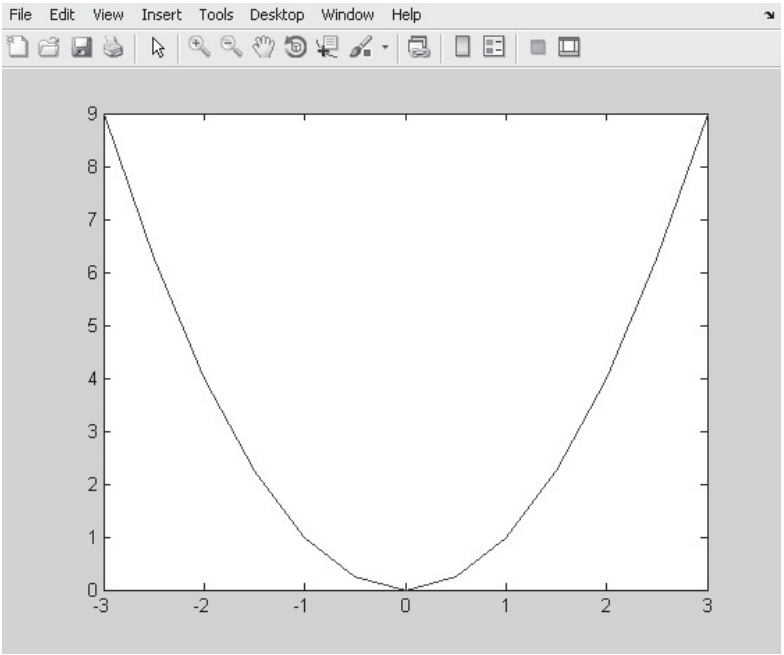
Wykresy

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

gdzie:

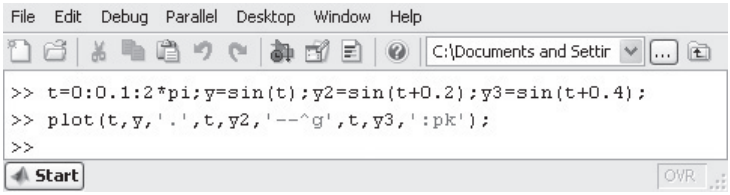
`LineSpec` to napis określający format wykresu.

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

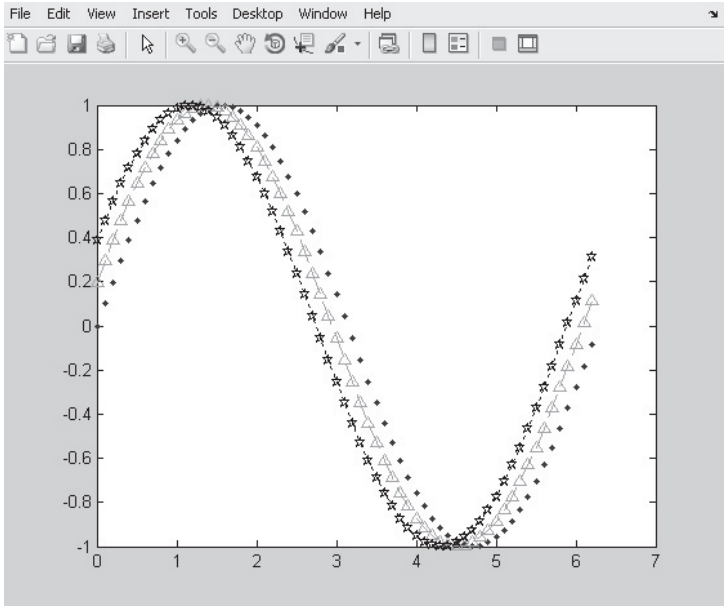
Oznaczenia formatu wykresu (na podstawie pomocy MATLAB-a):

B	niebieski	.	point	-	solid
G	zielony	O	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*



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:

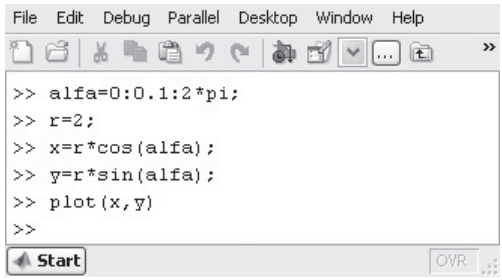
$$\begin{cases} x = x_0 + r\cdot\cos\alpha \\ y = y_0 + r\cdot\sin\alpha \end{cases}$$

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*



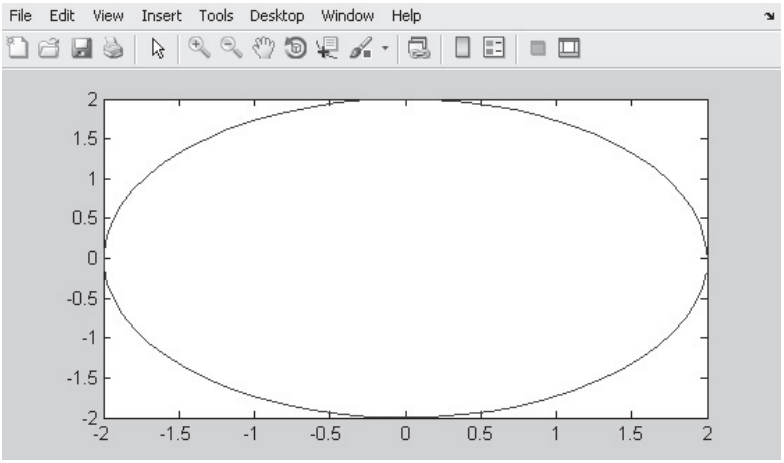
Na rysunku 1.20 widać, że okrąg bardziej przypomina elipsę, a to dlatego, że MATLAB próbuje przeskalować wykres tak, żeby zajmował całą przestrzeń okna wykresu. Poleceniem `axis equal` spowodujemy, że osie *x* i *y* będą w tej samej skali, w wyniku czego uzyskamy śliczny okrąg (rysunek 1.21). Okrąg na 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:

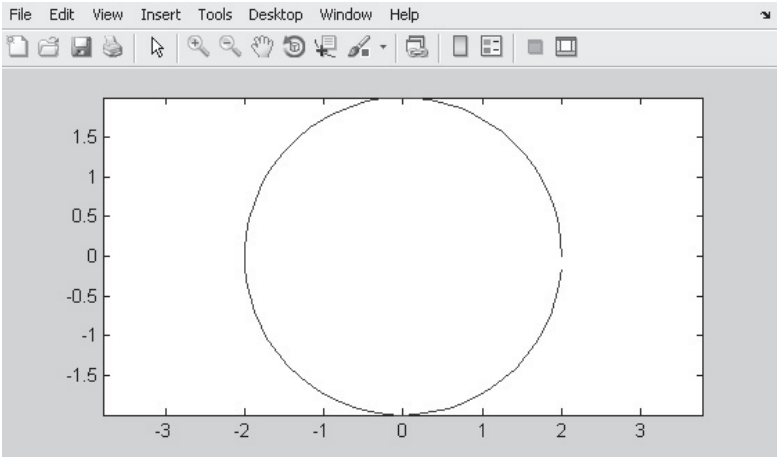
$$(x-x_0)^2+(y-y_0)^2=r^2$$

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*



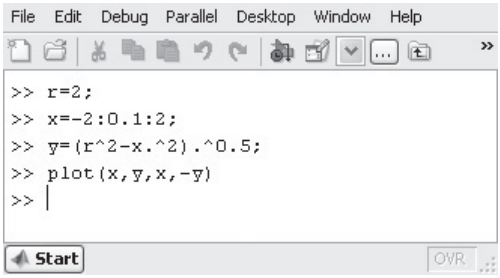
Rysunek 1.21. *Okno wykresu z narysowanym okręgiem i z równą skalą osi*

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

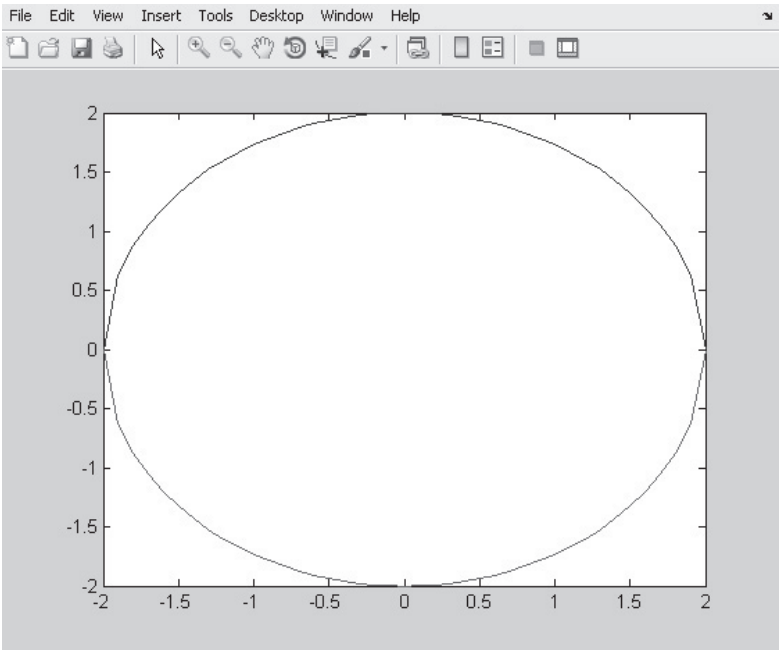
$$y = \pm\sqrt{r^2 - x^2}$$

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*



Rysunek 1.23. *Efekt wykonania ciągu instrukcji z rysunku 1.22*

Zadania dla czytelnika

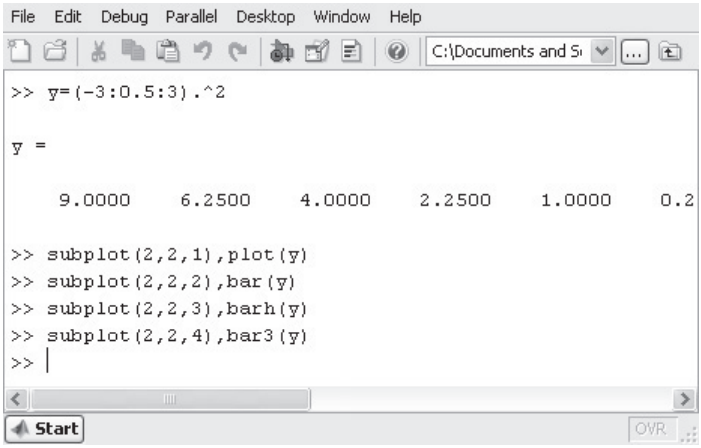
Sprawdź, czy zrozumiałeś rysowanie wykresów, rysując:

- elipsę,
- spiralę Archimedesa,
- hipotrochoidę.

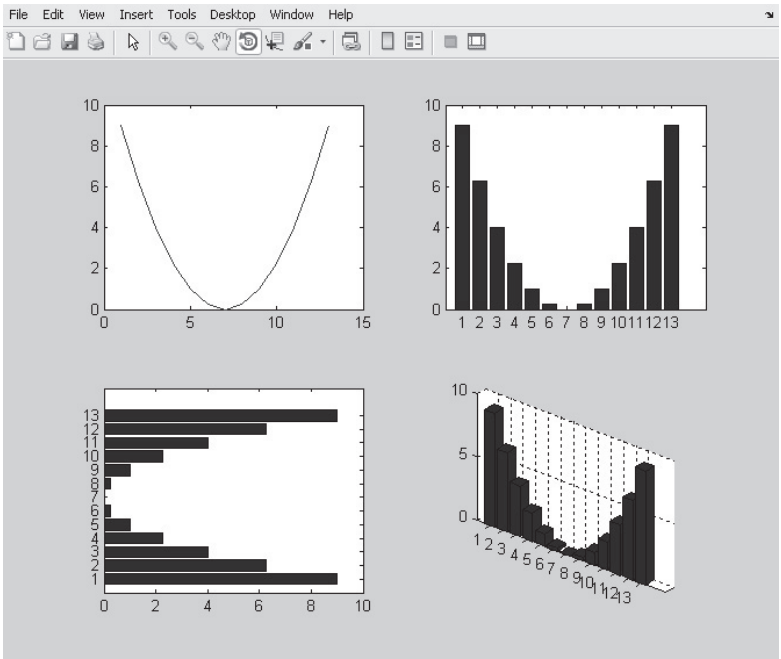
Równania możesz znaleźć na stronach Wikipedii.

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*



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.

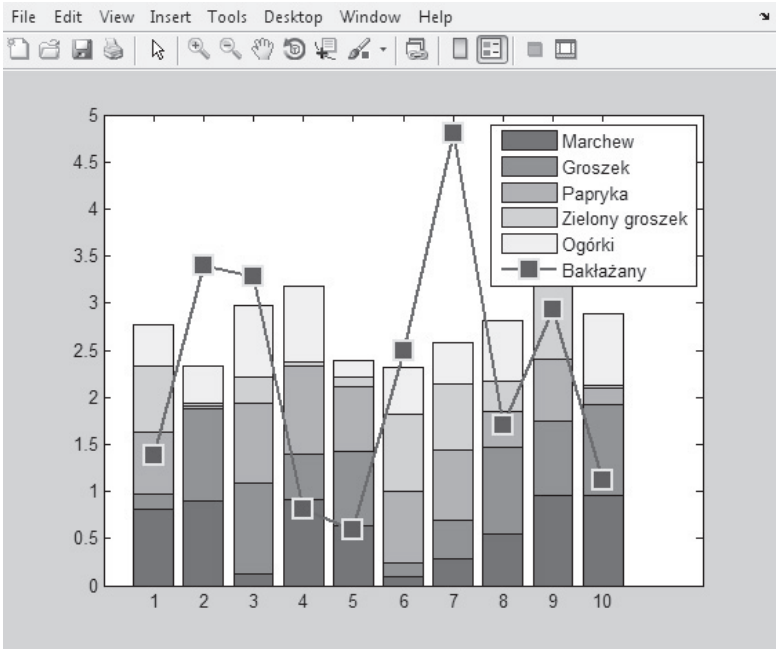
Standardowe funkcje, za pomocą których możemy dodawać opisy wykresów i modyfikować osie, zawiera tabela 1.4. Opis ciekawszych efektów, które możemy uzyskać, znajduje się w rozdziale 7.

Tabela 1.4. *Funkcje wspomagające rysowanie wykresów*

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

G

I

K

L

M

N

O

P

R

S

T

U

V

W

Z

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!

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
http://helion.pl

ISBN 978-83-246-8134-1

9 788324 681341

Informatyka w najlepszym wydaniu

cena: 44,90 zł