KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
WYŻSZA SZKOŁA GOSPODARKI W BYDGOSZCZY
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY
Człowiek - najlepsza inwestycja!

Tomasz Ocetkiewicz

Zastosowanie Matlaba w mechatronice

Część I - Wprowadzenie

MATERIAŁY ZREALIZOWANE
W RAMACH PROJEKTU

**NOWE SPECJALNOŚCI W MECHATRONICE
- ROZWÓJ NOWOCZESNEGO KSZTAŁCENIA
W WYŻSZEJ SZKOLE GOSPODARKI W BYDGOSZCZY**

NUMER PROJEKTU: POKL.04.01.01-00-333/09-00

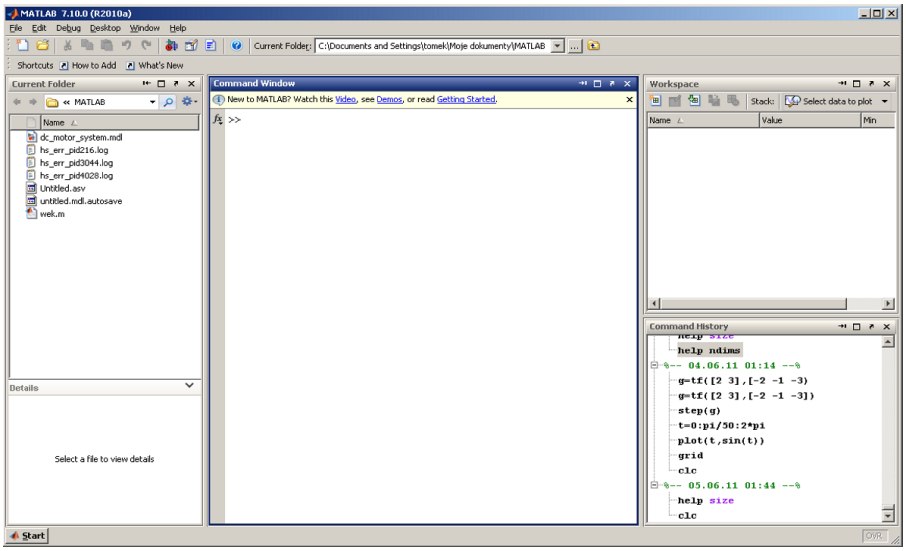
WYŻSZA SZKOŁA GOSPODARKI W BYDGOSZCZY
BYDGOSZCZ 2011

Projekt współfinansowany ze środków Unii Europejskiej w ramach
Europejskiego Funduszu Społecznego

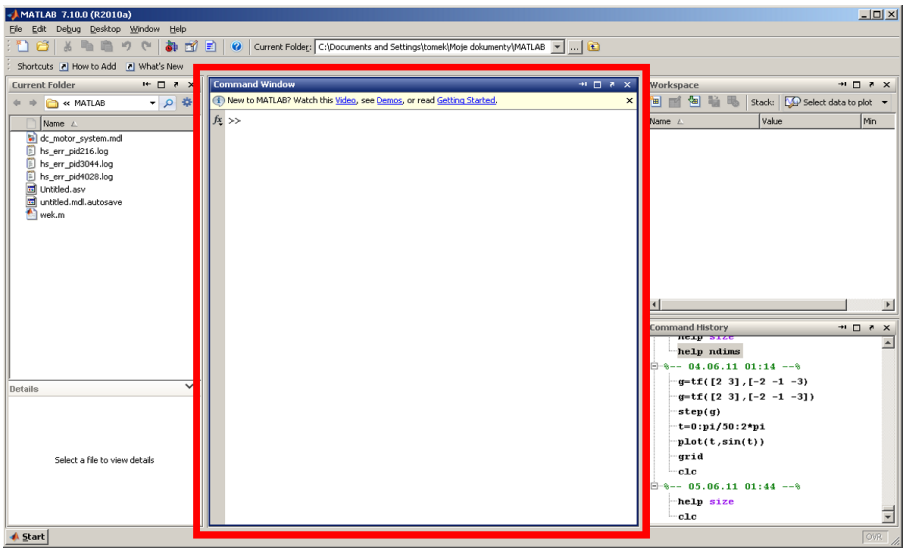
Spis treści

1. Wstęp

Matlab jest pakietem oprogramowania firmy Mathworks przeznaczonym do wykonywania obliczeń numerycznych naukowych i inżynierskich. Nazwa Matlab jest skrótem od MATrix LABoratory, czyli Laboratorium Macierzowe. Po uruchomieniu programu na ekranie monitora powinno zostać wyświetlone następujące okno główne programu.

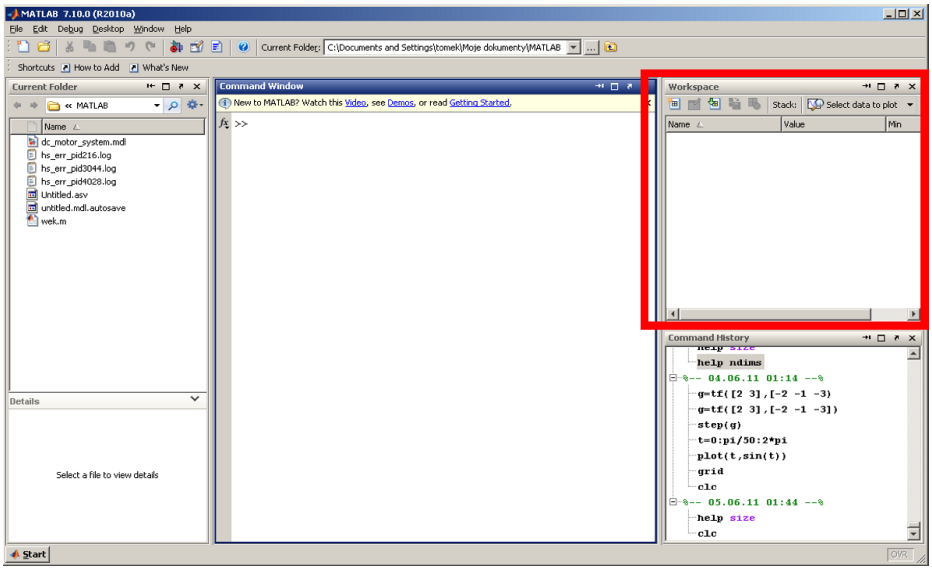


Wygląd głównego okna programu Matlab można zmieniać korzystając z menu ***Desktop***. Polecenia w nim zawarte umożliwiają wyłączenie wyświetlania pewnych elementów powyższego okna lub zastąpienie ich innymi, np. oknem edytora m-plików czy oknem przeglądarki. Jeżeli użytkownik chce powrócić do układu jak na powyższym rysunku, powinien wykorzystać polecenie ***Default*** z menu ***Desktop*** → ***Desktop layout*** → ***Default***. Najważniejszą elementem wyświetlonego okna dialogowego jest okno komend – ***Command Window***.



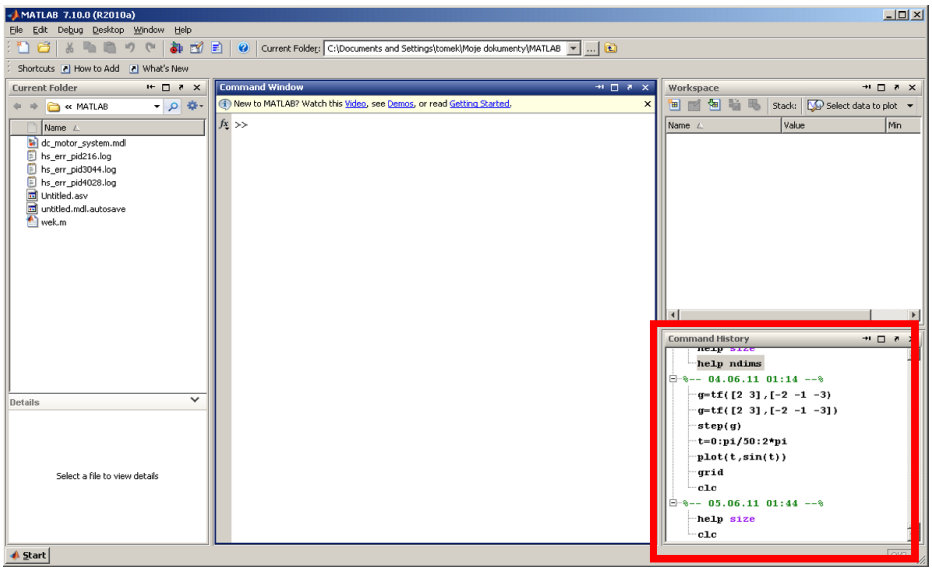
Okno komend jest przeznaczone do wprowadzania komend i instrukcji przez użytkownika oraz do wyświetlania wyników obliczeń. Program zgłasza gotowość do pracy znakiem zachęty >>. Zawartość okna można wyczyścić korzystając z polecenia ***clc***.

Okno ***Workspace*** jest określane jako przestrzeń robocza pakietu Matlab.



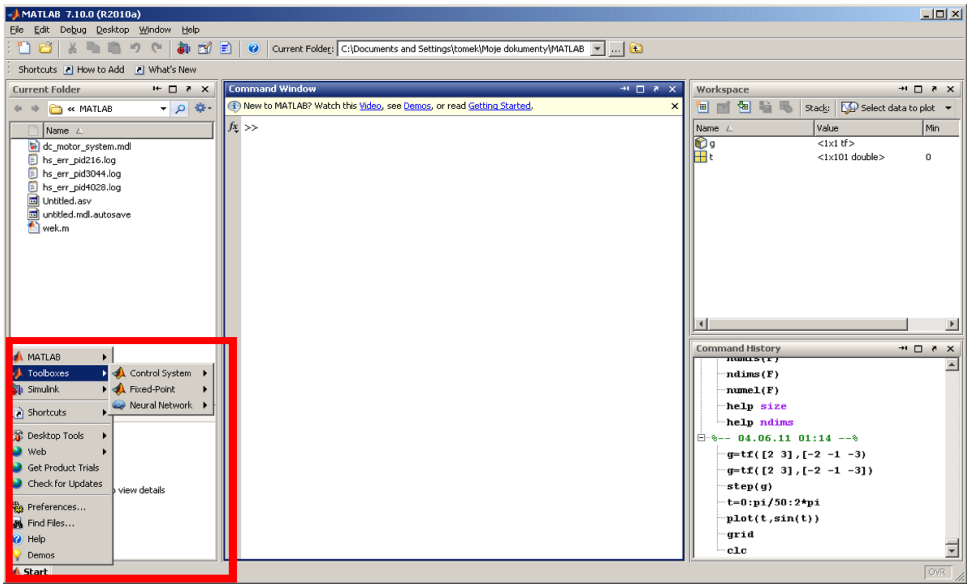
Korzystając z tego okna użytkownik może podejrzeć zawartość wszystkich zmiennych, które zostały wprowadzone w bieżącej sesji programu. Dla każdej zmiennej wyświetlany jest jej typ oraz minimalna i maksymalna wartość. Przestrzeń robocza jest wspólna dla Matlaba oraz wszystkich zainstalowanych w nim bibliotek (tzw. toolbox'ów) i pakietów symulacyjnych.

Okno ***Command History*** zawiera zestawienie wszystkich wprowadzonych przez użytkownika poleceń.



Podświetlenie wybranej instrukcji i dwukrotne kliknięcie lewym przyciskiem myszy spowoduje przeniesienie wybranej instrukcji do okna komend i jej wykonanie. W przypadku naciśnięcia prawego przycisku myszy, po zaznaczeniu instrukcji, zostanie otwarte menu kontekstowe, które umożliwia między innymi skopiowanie wybranego polecenia do okna komend lub do nowego m-pliku. Zawartość okna ***Command History*** można przeglądać, w Oknie Komend, korzystając z przycisków ↑ i ↓.

Ważnym elementem pakietu Matlab są tzw. toolbox'y. Toolbox jest zbiorem programów (najczęściej w formie m-plików) zawierającym narzędzia przeznaczone do wspomagania obliczeniowego wybranego zagadnienia. Przykładowo, do analizy i syntezy układów sterowania w dziedzinie czasu lub częstotliwości przeznaczony jest pakiet ***Control System Toolbox***. Lista zainstalowanych toolbox'ów wraz ich plikami pomocy dostępna jest po naciśnięciu przycisku ***Start***.



Pozostałe elementy głównego okna programu zostaną omówione w dalszej części skryptu.

2. Matlab jako kalkulator

Pakiet Matlab można wykorzystać do elementarnych obliczeń:

```
>> 10+2
ans =
      12

>> 15*242
ans =
      3630

>> 5^42
ans =
      2.2737e+029
```

W przypadku korzystania z wartości zespolonych należy posłużyć się jednostką urojoną ***i*** lub ***j***:

```
>> 5+2i+4+7j
ans =
      9.0000 + 9.0000i
```

Zestawienie podstawowych funkcji matematycznych dostępnych w programie znajduje się w tabeli.

Funkcje	Opis	Przykład

sin, ***sind***
cos, ***cosd***
tan, ***tand***
cot, ***cotd***
	funkcje trygonometryczne; argumentami funkcji mogą być skalary lub macierze zawierające elementy rzeczywiste i zespolone	`sin(pi/4) %argument w rad`
		`ans =`
		`0.7071`
		`sind(30) %argument w stopniach`
		`ans =`
		`0.7071`

asin, ***asind***
acos, ***acosd***
atan, ***atand***
cot, ***acot***
	funkcje cyklometryczne; argumentami funkcji mogą być skalary lub macierze zawierające elementy rzeczywiste i zespolone	`asin(0.5) %wynik w rad`
		`ans =`
		`0.5236`
		`asind(0.5) %wynik w stopniach`
		`ans =`
		`30.0000`

sinh ***cosh*** ***tanh*** ***coth***	funkcje hiperboliczne; argumentami funkcji mogą być skalary lub macierze zawierające elementy rzeczywiste i zespolone	-
asinh ***acosh*** ***atanh*** ***acoth***	funkcje odwrotne do hiperbolicznych; argumentami funkcji mogą być skalary lub macierze zawierające elementy rzeczywiste i zespolone	-

sqrt
realsqrt
	pierwiastek kwadratowy zespolony(***sqrt***) lub rzeczywisty(***realsqrt***); argumentami funkcji mogą być macierze lub skalary	`sqrt(4+10i)`
		`ans =`
		`2.7176 + 1.8399i`
		`realsqrt(23)`
		`ans =`
		`4.7958`

exp
realpower
	funkcja ***exp***(*a*) zwraca wartość wyrażenia w postaci e^a; funkcja ***realpower***(*a,b*) operuje skalarach i macierzach rzeczywistych, i zwraca wartość a^b	`realpow(3,2)`
		`ans =`
		`9`

log
log2
log10
reallog
	logarytm; funkcje ***log***, ***log2*** i ***log10*** są określone w ciele liczb zespolonych; funkcja ***reallog*** operuje wyłącznie na liczbach rzeczywistych dodatnich	`reallog(10)`
		`ans =`
		`2.3026`
		`log(-1)`
		`ans =`
		`0 + 3.1416i`

rem
mod
	reszta z dzielenie 2 liczb; funkcje działają dla argumentów rzeczywistych	`rem(4,3)`
		`ans =`
		`1`

round
fix
floor
ceil
	zaokrąglenie liczby; część rzeczywista i urojona są zaokrąglane oddzielnie	`a = [-1.9, -0.2, 3.4]`
		`round``(a)`
		`ans =`
		`  -2  0  3`
		`fix``(a)`
		`ans =`
		`  -1  0  3`
		`floor``(a)`
		`ans =`
		`  -2  -1  3`
		`ceil``(a)`
		`ans =`
		`  -1  0  4`

abs	wartość bezwzględna dla liczb rzeczywistych; moduł liczby zespolonej równy $\sqrt{a^2+b^2}$ dla liczby $a+jb$	**`abs`**`(-10)`
		`ans =`
		`  10`
		`abs``(-10+4i)`
		`ans =`
		`  10.7703`

factorial	funkcja silnia	**`factorial`**`(4)`
		`ans =`
		`  24`

gcd	największy wspólny dzielnik	**`gcd`**`(4,16)`
		`ans =`
		`  4`

lcm	najmniejsza wspólna wielokrotność	`a = 3`
		`b = 5`
		`lcm``(a,b)`
		`ans =`
		`  15`

angle	argument, w radianach, liczby zespolonej równy $arctg\frac{b}{a}$ dla liczby $a+jb$	`a=5+10j`
		`angle``(a)`
		`ans =`
		`  1.1071`

complex	liczba zespolona na bazie 2 wartości rzeczywistych	`c=`**`complex`**`(2,4)`
		`c =`
		`  2.0000 + 4.0000i`

conj	liczba zespolona sprzężona równa $a-jb$ dla liczby $a+jb$	`c=complex(2,4)`
		`conj``(c)`
		`ans =`
		`  2.0000 - 4.0000i`

imag	część urojona liczby zespolonej	`c=complex(2,4)`
		`imag``(c)`
		`ans =`
		`  4`

isreal	funkcja zwraca wartość 1 jeżeli argumentem jest skalar lub macierz złożona z elementów rzeczywistych; w przeciwnym wypadku zwracane jest 0	`c=complex(2,4)`
		`isreal``(c)`
		`ans =`
		`  0`

real	część rzeczywista liczby zespolonej	`c=complex(2,4)`
		`real``(c)`
		`ans =`
		`  2`

3. Typy i formaty danych

Matlab dopuszcza stosowanie danych liczbowych typu rzeczywistego i zespolonego. Deklaracja zmiennej odbywa się poprzez podanie nazwy i wartości początkowej:

```
a=5
b=4+10j
```

Określenie typu i rozmiaru zmiennej nie jest wymagane. Zmienne mogą być oznaczone dowolną kombinacją liter i cyfr o długości nie przekraczającej 31 znaków, przy czym pierwszym znakiem w nazwie zmiennej musi być litera. Stosowanie znaków narodowych (ą, ę, etc.) jest zabronione. Wskazane jest niestosowanie następujących kombinacji znaków do oznaczania własnych zmiennych:

Oznaczenie	Opis
pi	przybliżenie liczby Л
i , j	jednostki urojone równe $\sqrt{-1}$
eps	precyzja liczb zmienno-przecinkowych równa $2^{-52} \approx 2.2204 \cdot 10^{-16}$
realmin	najmniejsza liczba zmienno-przecinkowa równa $2^{-1022} \approx 2.2251 \cdot 10^{-308}$
realmax	największa liczba zmienno-przecinkowa równa $2^{1022} \approx 2.2251 \cdot 10^{308}$
intmin	najmniejsza liczba całkowita 32-bitowa równa -2147483647
intmax	największa liczba całkowita 32-bitowa równa 2147483647
inf	nieskończoność
NaN	wyrażenie nieoznaczone
ans	wynik ostatniej operacji liczbowej, który nie jest przypisany do zmiennej

Przypisanie do wymienionych w tabeli symboli innych wartości może prowadzić, w pewnych przypadkach, do błędów obliczeniownych:

```
j
ans =
      0 + 1.0000i
a=5+2*j
a =
      5.0000 + 2.0000i
```

```
j=5
j =
      5
a=5+2*j
a =
      15
```

Duże i małe litery są rozróżniane, czyli polecenia `aa=2`, `aA=3`, `Aa=4` i `AA=5` oznaczają zadeklarowanie 4 różnych zmiennych. Wprowadzone w oknie komend zmienne są przechowywane w przestrzeni roboczej (***Workspace***). W celu usunięcia zmiennej należy posłużyć się poleceniem ***clear*** *nazwa_zmiennej*. Poniższy przykład spowoduje usunięcie zmiennej *a*:

```
a=3
a =
     3
clear a
a
??? Undefined function or variable 'a'.
```

Jeżeli zachodzi konieczność usnięcia więcej niż jednej zmiennej, nazwy kolejnych usuwanych zmiennych należy oddzielić spacjami:

```
clear aa ab ac
```

Zastosowanie symbolu * w poleceniu `clear a*` umożliwia skasowanie wszystkich zmiennych o nazwach zaczynających się na *a*. W przypadku wywołania `clear *ba` zostaną usunięte zmienne o nazwach zakończonych *ba*. Wywołanie `clear all` usuwa wszystkie zmienne z pamięci i przestrzeni roboczej. Wymiennie z poleceniem `clear` można stosować polecenie `clearvars`.

W Matlabie istnieje możliwość zdefiniowania zmiennych typu łańcuch znaków(*string*):

```
s = 'Matlab'
```

Tekst jest zapamiętywany w formie wektora znaków.

Do określania **sposobu** wyświetlania wartości na ekranie komputera służy polecenie ***format***. Składania tego polecenia jest następująca:

format *nazwa_formatu*

Nazwa formatu	Przykładowa reprezentacja
short	0.6667 ; -5.2345e-10
short e	6.6667e-001 ; -5.2345e-010
long	0.666670000000000 ; -5.234500000000000e-010
long e	6.666700000000000e-001 ; -5.234500000000000e-010
hex	zawartość komórek pamięci przechowujących dane liczby 3fe5555c52e72da1; be01fc509ccb10ed
+	dla liczb dodatnich wyświetlany jest znak +
bank	format walutowy; do 2 miejsc po przecinku
compact	wyłącza dodawanie dodatkowych pustych wierszy
loose	włącza dodawanie dodatkowych pustych wierszy
rat	przybliżenie liczb za pomocą ułamków

Wprowadzone do przestrzeni roboczej Matlaba zmienne można przed zakończeniem pracy zapisać na dysku przy pomocy polecenia `save`. Wywołanie tego polecenia bez parametrów spowoduje zapisanie do pliku *matlab.mat* wszystkich zmiennych z *Workspace*. Zmienne można zapisać do pliku o nazwie określonej przez użytkownika. W takim przypadku polecenie ***save*** należy wywołać z parametrem określającym nazwę pliku z rozszerzeniem. Zapisanie wybranych zmiennych jest możliwe poprzez podanie ich nazw rozdzielonych znakiem spacji. W poniższym przykładzie zapisano w pliku *test.mat* zmienne *a*, *b* i *c*:

```
save test.mat a b c
```

Dane są zapisywane w pliku w specjalnym formacie Matlaba. Można je zapisać w ASCII dodając do polecenia ***save*** parametr *-ascii*:

```
save test.mat a b c -ascii
```

Zmienne można ponownie wprowadzić do Workspace przy pomocy polecenia ***load***:

```
load test.mat
```

4. Definiowanie wektorów i macierzy

Wektor można wprowadzić do Matlaba poprzez wpisanie wszystkich jego elementów oddzielonych znakiem spacji lub przecinkiem, otoczonych nawiasami kwadratowymi:

```
A = [0 2 4 8]
A =
      0     2     4     8

A = [0,2,4,8]
A =
      0     2     4     8
```

Ilość odstępów pomiędzy elementami nie ma znaczenia:

```
A = [0  2      4                    8]
A =
      0     2     4     8
```

W przypadku definiowania macierzy kolejne wiersze należy rozdzielić średnikami(**;**):

```
B = [0 2 4 8;1 3 5 7;0 14 2 4]
B =
      0     2     4     8
      1     3     5     7
      0    14     2     4
```

Definicja macierzy pustej jest następująca:

```
C = []
```

Wartość elementu macierzy można wyświetlić poprzez podanie nazwy macierzy oraz numeru wiersza i kolumny:

```
B(1,3)            % element z 1 wiersza i 3 kolumny
ans =
      4
```

Wiersze i kolumny są numerowanie od 1.

Jeżeli kolejne elementy wektora zmieniają się o stałą wartość to do zdefiniowania takiego wektora można użyć następującego wyrażenia:

C = *wartość_początkowa*:*przyrost_wartości*:*wartość końcowa*

gdzie:

wartość_początkowa – pierwszy element wektora;
przyrost_wartosci – różnica pomiędzy 2 kolejnymi elementami wektora;
wartość_końcowa – ostatni element wektora nie może być większy niż ta wartość.

```
C = 0:2:10
C =
      0     2     4     6     8    10
```

Jeżeli przyrost wartości wynosi 1 to wyrażenie można uprościć:

C = wartość_początkowa:wartość_końcowa

Inną metodą definiowania wektorów jest skorzystanie z funkcji ***linspace*** i ***logspace***. Składnie tych poleceń są następujące:

D = ***linspace****(d1,d2,ilosc_przedzialow)*
D = ***logpsace****(d1,d2,ilosc_przedzialow)*

W przypadku polecenia ***linspace*** *d1* i *d2* oznaczają odpowiednio pierwszy i ostatni element budowanego wektora. Wektor zbudowany przy pomocy funkcji ***logspace*** będzie posiadał pierwszy element równy 10^{d1}, a ostatni 10^{d2}. Parametr *ilosc_przedzialow*, w obu przypadkach, oznacza ilość elementów wektora (lub ilość odcinków na jaki zostanie podzielony przedział <*d1*,*d2*>).

```
D = linspace(0,100,10)
D =
      Columns 1 through 6
      0   11.1111   22.2222   33.3333   44.4444   55.5556
      Columns 7 through 10
      66.6667   77.7778   88.8889  100.0000
```

```
D = logspace(0,2,10)
D =
      Columns 1 through 6
      1.0000    1.6681    2.7826    4.6416    7.7426   12.9155
      Columns 7 through 10
      21.5443   35.9381   59.9484  100.0000
```

Oba polecenia są często wykorzystywane do budowania wektora zmiennej niezależnej przy konstruowaniu wykresów.

Kolejną metodą definiowania wektorów i macierzy jest korzystanie z innych macierzy. W poniższym przykładzie z macierzy *A*, *B* i *C* zostanie zbudowana macierz *D*.

```
A=[1 2;3 4]            B = [5 ; 6]            C=[7 8 9]
A =
                       B =                    C =
     1     2
     3     4                5                      7     8     9
                            6
D = [A B;C]
D =
     1     2     5
     3     4     6
     7     8     9
```

Wszystkie wymienione techniki definiowania macierzy można łączyć. Korzystając z macierzy C z poprzedniego przykładu:

```
E = [C ; 1 2 5 ; 3 4 7 ; 11:13]
E =
     7     8     9
     1     2     5
     3     4     7
    11    12    13
```

Zestawienie dodatkowych funkcji wspomagających konstruowanie macierzy zawiera tabela:

Funkcja	Rodzaj utworzonej macierzy	Przykłady

***zeros**(n,m)*	macierz o n-wierszach i m-kolumnach, której wszystkie elementy są zerowe; jeżeli funkcja zostanie wywołana z jednym argumentem zwracana jest macierz kwadratowa	`A=`**`zeros`**`(3,2)`
		`A =`
		`0 0`
		`0 0`
		`0 0`

***ones**(n,m)*	macierz o n-wierszach i m-kolumnach, której wszystkie elementy mają wartość równa 1; jeżeli funkcja zostanie wywołana z jednym argumentem zwracana jest macierz kwadratowa	`A=`**`ones`**`(2)`
		`A =`
		`1 1`
		`1 1`

***eye**(n,m)*	macierz jednostkowa o n-wierszach i m-	`A=`**`eye`**`(2,3)`

	kolumnach; jeżeli funkcja zostanie wywołana z jednym argumentem zwracana jest macierz kwadratowa	`A =`
		`1 0 0`
		`0 1 0`

***rand**(n,m)*	macierz o n-wierszach i m-kolumnach, której elementy są liczbami pseudolosowymi o rozkładzie równomiernym na przedziale (0,1); jeżeli funkcja zostanie wywołana z jednym argumentem zwracana jest macierz kwadratowa	`A = rand(4,2)`
		`A =`
		`0.4822 0.5274`
		`0.0141 0.7250`
		`0.6229 0.6074`
		`0.2311 0.5884`

***randi**([max,min],n,m)*	macierz o n-wierszach i m-kolumnach, której elementy są pseudolosowymi liczbami całkowitymi o rozkładzie równomiernym na przedziale (min,max); w przypadku braku parametru m zwracana będzie macierz kwadratowa	`A = randi([-3,3],3)`
		`A =`
		`3 1 1`
		`-3 1 0`
		`-1 1 -3`

***randn**(n,m)*	macierz o n-wierszach i m-kolumnach, której elementy są liczbami pseudolosowymi o rozkładzie normalnym na przedziale (0,1); jeżeli funkcja zostanie wywołana z jednym argumentem zwracana jest macierz kwadratowa	`A = randn(4,2)`
		`A =`
		`-0.2146 1.0905`
		`0.4863 -0.9465`
		`0.3309 -0.4385`
		`1.2679 0.3432`

gallery	przeznaczona do tworzenia specyficznych macierzy, np. Cauchy'ego, Toeplitz'a, etc; szczegółowy opis znajduje się w dokumentacji Matlaba (można skorzystać z polecenia ***help gallery***)	-

***magic**(n)*	macierz kwadratową o rozmiarach n x n, w której sumy wartości elementów w wierszach, kolumnach i na głównej przekątnej są jednakowe	`magic(3)`
		`ans =`
		`8 1 6`
		`3 5 7`
		`4 9 2`

***meshgrid**(x,y)*	regularna siatka na płaszczyźnie xy, oparta na wektorze poziomym x i pionowym y	`meshgrid(-1:0.5:1,-1:0.5:1)`

***diag**(a)*	macierz diagonalna; elementy na głównej przekątnej są parametrem funkcji	`A = diag([2 4 6 8])`
		`A =`
		`2 0 0 0`
		`0 4 0 0`
		`0 0 6 0`
		`0 0 0 8`

Polecenie ***randn*** zwraca wartości pseudolosowe o rozkładzie normalnym, wartości oczekiwanej 0 i odchyleniu standardowym równym 1. W celu uzyskania innych parametrów rozkładu normalnego należy posłużyć się operacjami arytmetycznymi. Przykładowo, rozkład o wartości oczekiwanej 1 i odchyleniu standardowym równym 2 można uzyskać w następujący sposób:

```
A = 1 + 2.*randn(100,1)
```

Do tworzenia dużej macierzy, która składa się z wielu powtórzeń mniejszej macierzy, można wykorzystać polecenie ***repmat***:

```
A=[1:3;2:4]
A =
      1     2     3
      2     3     4
B = repmat(A,2,2)
B =
      1     2     3     1     2     3
      2     3     4     2     3     4
      1     2     3     1     2     3
      2     3     4     2     3     4
```

Macierz *B* będzie n-krotnym powtórzeniem macierzy *A* w poziomie i m-krotnym w pionie.

5. Odwołanie do elementów wektorów i macierzy

Zdefiniowane wektory i macierze mogą podlegać modyfikacjom. Zmiany wartości elementu wektora lub macierzy dokonujemy poprzez wywołanie tego elementu i przypisanie mu nowej wartości:

```
F = [1:4;10:13;4:7]
F =
      1     2     3     4
      10    11    12    13
      4     5     6     7

F(2,4)
ans =
      13
```

Jeżeli modyfikacji ma podlegać kilka kolejnych elementów macierzy to można posłużyć się znakiem dwukropka(:). Zapis *F(1:2,:)* oznacza, że odwołujemy się do wszystkich elementów z wierszy od 1 do 2 macierzy *F:*

```
F(1:2,:)
ans =
      1     2     3     4
      10    11    12    13
```

Znak dwukropka można wykorzystać do budowania podmacierzy z danej macierzy:

```
F1=F(:,1)
F1 =
      1
      10
      4
```

Notacja *F2 = F([1 3],[2 3])* oznacza, że macierz *F2* zostanie utworzona z elementów 1 i 3 wiersza oraz 2 i 3 kolumny:

```
F2 = F([1 3],[2 3])
F2 =
      2     3
      5     6

F3 = F([1 3],:)
F3 =
      1     2     3     4
      4     5     6     7
```

Matlab pozwala na rozbudowanie już istniejącej macierzy poprzez dopisanie kolumn lub wierszy. Warunkiem wykonania takiej operacji są poprawne rozmiary dopisywanych elementów.

```
G=([1 2;3 5])
G =
      1     2
      3     5

G1=[G;7 8]
G1 =
      1     2
      3     5
      7     8

G2=[9 3;G]
G2 =
      9     3
      1     2
      3     5

G3=[G [2;3]]
G3 =
      1     2     2
      3     5     3

G4=[[2;3] G]
G4 =
      2     1     2
      3     3     5
```

W pewnych zastosowaniach przydatne jest traktowanie macierzy jako wektora kolumnowego, w którym kolejne kolumny będą umieszczone jedna za drugą.

```
F = [1:4;10:13;4:7]
F =
      1     2     3     4
      10    11    12    1
      4     5     6     7
F(:)'
ans =
     1    10     4     2    11     5     3    12     6     4    13     7
```

Polecenie ***F(2)*** zwróci wartość 10, a ***F(6)*** wartość 5.

Traktowanie macierzy jako wektora umożliwia usunięcie wszystkich wartości z macierzy *F* większych od 10:

```
F=F(F<10)
```

Innym przykładem może być usunięcie wszystkich, wierszy w których 2 element jest większy od 3:

```
F=F(F(:,2)<3,:)
F =
      1     2     3     4
```

Wiersze lub kolumny można usuwać, z macierzy, korzystając z macierzy pustej:

```
G=([1 2;3 5])
G =
      1     2
      3     5

G(1,:)=[]
G =
      3     5
```

6. Operacje na wektorach i macierzach

Zestawienie podstawowych operacji macierzowych, zdefiniowanych w Matlabie, znajduje się w poniższej tabeli. Działanie zostanie wykonane wyłącznie w przypadku, gdy wymiary poszczególnych czynników będą zgodne z zasadami rachunku macierzowego.

Operacja	Opis	Przykład

A'	transpozycja macierzy *A*, z zamianą elementów zespolonych na sprzężone	`A = [1 2;1+j 2-2*j]`
		`A =`
		`1 2`
		`1 + 1i 2 - 2i`
		`A'`
		`ans =`
		`1 1 - 1i`
		`2 2 + 2i`

A.'	transpozycja macierzy *A*	`A = [1 2;1+j 2-2*j]`
		`A =`
		`1 2`
		`1 + 1i 2 - 2i`
		`A.'`
		`ans =`
		`1 1 + 1i`
		`2 2 - 2i`

$A + B$	dodawanie macierzy	-
$A - B$	odejmowanie macierzy	-
*A*B*	mnożenie macierzy w sensie Cauchy'ego	-
A/B	prawostronne dzielenie macierzy; odpowiada rozwiązaniu równania algebraicznego $x \cdot B = A$	-
A\B	lewostronne dzielenie macierzy; odpowiada rozwiązaniu równania algebraicznego $A \cdot x = B$	-
inv	inwersja macierzy	`A = [1 2;1+j 2-2*j]`

		A = [1 2;1+j 2-2*j] A = 1 2 1 + 1i 2 - 2i B=inv(A) B = 0.50 + 0.50i 0 - 0.50i 0.25 - 0.25i 0 + 0.25i
A^p	potęgowanie macierzy; jeżeli *p* jest liczbą całkowitą to operacja odbywa się poprzez wielokrotne mnożenie macierzy; dla *p* rzeczywistych wykonywany jest rozkład według wektorów i wartości własnych	A = 1 3 3 4 **A^2** ans = 10 15 15 25
eig	wyznaczenie wartości własnych	A=[1 3;3 4] **eig**(A) ans = -0.8541 5.8541
svd	rozkład macierzy względem wartości szczególnych	A=[1 3;3 4] **svd**(A) ans = 5.8541 0.8541
diff	„różniczkowanie wektora”; funkcja oblicza różnice pomiędzy kolejnymi elementami	A=[5 2 0 -2 3] **diff**(A) ans = -3 -2 -2 5
cumsum	sumowanie kumulacyjne kolejnych elementów wektora	A=[5 2 0 -2 3] **cumsum**(A) ans = 5 7 7 5 8
cumprod	mnożenie kumulacyjne kolejnych elementów wektora	A=[5 2 0 -2 3] **cumprod**(A) ans = 5 10 0 0 0
sort ***sortrows***	sortowanie elementów według niemalejących wartości	A=[5 2 0 -2 3] **sort**(A) ans = -2 0 2 3 5

7. Wyznaczenie rozmiaru macierzy i wektora

W poniższej tabli znajduje się zestawienia funkcji przeznaczonych do wyznaczenia rozmiarów macierzy. W przykładach posłużono się następującymi macierzami:

```
F = [1:4;10:13;4:7]
F =
     1     2     3     4
    10    11    12    13
     4     5     6     7

G = [1 3 5 6 7 8 9 0]
G =
     1     3     5     6     7     8     9     0
```

Funkcja	Opis	Przykład

size	funkcja zwraca 2-elementowy wektor zawierający ilość wierszy i kolumn argumentu	`size(F)`
		`ans =`
		`3 4`
		`[n,m] = size(F)`
		`n =`
		`3`

ndmis	funkcja zwraca ilość wymiarów macierzy	`ndims(F)`
		`ans =`
		`2`

numel	funkcja zwraca liczbę elementów macierzy	`numel(F)`
		`ans =`
		`12`

length	funkcja zwraca długość wektora	`length(G)`
		`ans =`
		`8`

8. Operacje tablicowe

Operacje tablicowe są skalarnymi operacjami matematycznymi zdefiniowanymi na poszczególnych elementach macierzy. Składnia operacji tablicowej różni się od składni odpowiadającej jej operacji macierzowej znakiem kropki przed symbolem operacji. Macierz wynikowa ma takie same wymiary jak argumenty.

Operacja	Opis	Przykład
A **.+** B A **.-** B	dodawanie/odejmowanie poszczególnych elementów macierzy A i B; operacja tożsama z $A+B$ lub $A-B$	-

*A.*B*	mnożenie elementu a_{ij} macierzy *A* przez element b_{ij} macierzy *B*	A=[1 2;3 4] B=[0.5 1;4 -2] **A.*B** ans = 0.5000 2.0000 12.0000 -8.0000
A./B	dzielenie elementu a_{ij} macierzy *A* przez element b_{ij} macierzy *B*	A=[1 2;3 4] B=[0.5 1;4 -2] **A./B** ans = 2.0000 2.0000 0.7500 -2.0000
A.\B	dzielenie elementu b_{ij} macierzy *B* przez element a_{ij} macierzy *A*	A=[1 2;3 4] B=[0.5 1;4 -2] **A.\B** ans = 0.5000 0.5000 1.3333 -0.5000
A.^p	każdy element a_{ij} macierzy *A* jest podnoszony do potęgi *p*	A=[1 2;3 4] **A.^2** ans = 1 4 9 16

9. Wielomiany

Wielomiany są wprowadzane do Matlaba przez podanie wektora współczynników stojących przy zmiennej niezależnej, w kolejnych potęgach, w porządku malejącym. Wielomian *W(s)* w postaci:

$$W(s) = 4s^5 + 5s^3 - 2s^2 + 7s + 1$$

jest w Matlabie reprezentowany przez wektor:

```
W = [4 0 5 -2 7 1]
```

Innym sposobem zdefiniowania wielomianu jest skorzystanie z funkcji ***poly(A)***, która oblicza współczynniki wielomianu charakterystycznego stowarzyszonego z macierzą *A*:

```
A=[5 12;3 4]
poly(A)
ans =
      1.0000   -9.0000  -16.0000
```

Zestawienie funkcji wspomagających obliczenia na wielomianach znajduje się w tabeli:

Operacja	Opis	Przykład

conv	splot (mnożenie) 2 wielomianów	`A=[1 4 7]`
		`B=[2 3 5]`
		`C=`**`conv`**`(A,B)`
		`C =`
		`2 11 31 41 35`

deconv	rozplot (dzielenie) wielomianów; wynikiem dzielenia wielomianu *B* przez wielomian *A* jest część całkowita (*Q*) i wielomian reszty(*R*)	`[Q,R]=`**`deconv`**`(B,A)`
		`Q =`
		`2`
		`R =`
		`0 -5 -9`

polyder	pochodna wielomianiu lub pochodna iloczynu wielomianów	`A=[1 4 7]`
		
		`polyder``(A)`
		`ans =`
		`2 4`
		
		`B=[2 3 5]`
		
		`polyder``(A,B)`
		`ans =`
		`8 33 62 41`

polyval	obliczenie wartości wielomianu w punkcie lub w punktach	`A=[1 4 7]`
		
		`polyval``(A,3)`
		`ans =`
		`28`
		
		`b=[3 5]`
		
		`polyval``(A,b)`
		`ans =`
		`28 52`

roots	wyznaczenie pierwiastków wielomianu	`A=[1 4 7]`
		
		`roots``(A)`
		`ans =`
		`-2.0000 + 1.7321i`
		`-2.0000 - 1.7321i`

10. Operacje i funkcje logiczne

W Matlabie logiczna prawda odpowiada macierzy zawierającej wyłącznie niezerowe elementy. Logiczny fałsz odpowiada macierzy, w której co najmniej jeden element jest równy zero lub macierz jest pusta.

Operacja	Opis
a == *b*	test, czy *a* jest równe *b*
a ~= *b*	test, czy *a* jest różne od *b*
a < *b*	test, czy *a* mniejsze od *b*
a > *b*	test, czy *a* większe od *b*
a <= *b*	test, czy *a* mniejsze równe *b*
a >=*b*	test, czy *a* większe równe *b*
a | *b*	alternatywa *a* lub *b*
a **&** *b*	koniunkcja *a* i *b*
***xor**(a,b)*	alternatywa wyłączająca *a* i *b* (funkcja logiczna Ex-OR)
~*a*	negacja *a*
***any**(a)*	zwraca wartość TRUE jeżeli co najmniej jeden z elementów macierzy jest różny od zera
***all**(b)*	zwraca wartość TRUE jeżeli wszystkie elementy macierzy są różne od zera

```
A=[1 0 -5 10]
B=[0 2 -6 2]

A>B
ans =
      1     0     1     1

A&B
ans =
      0     0     1     1

xor(A,B)
ans =
      1     1     0     0

any(A)
ans =
      1

all(A)
ans =
      0
```

11. Instrukcje sterujące

Lista instrukcji sterujących w Matlabie jest następująca:

- instrukcje iteracyjne (pętle) ***for*** i ***while***
- instrukcje ***break***, ***return***, ***error, try, catch***
- instrukcję warunkową ***if-else***
- instrukcja ***switch-case***

Ogólna postać pętli ***for*** jest następująca:

for *i = wektor*
  podprogram pętli;
end

Jeżeli wyrażenie *wektor* będzie zawierało jedną wartość to *podprogram pętli* zostanie wykonany określą ilość razy równą tej wartości. W przypadku, gdy wyrażenie *wektor* będzie miało postać wektora wierszowego to *podprogram pętli* zostanie wykonany dla kolejnych elemetów tego wektora, a ilość wykonań pętli będzie równa ilości elementó wektora. Pętle ***for*** mogą być zagnieżdżane dowolną ilość razy. Każda pętla ***for*** musi być zakończona instrukcją ***end***.

W poniższym przykładzie pętla ***for*** została wykorzystana do wygenerowania macierzy kwadratowej, której każdy element (*n*,*m*) ma wartość równą iloczynowi *n* i *m*.

```
for n=1:3
      for m=1:3
            A(n,m) = n*m;
      end
end
A =
      1     2     3
      2     4     6
      3     6     9
```

W celu uniknięcia wyświetlania wyników obliczeń dla każdego powtórzenia pętli ***for***, kolejne linie *podprogramu pętli* powinny być zakończone średnikami(**;**).

Struktura instrukcji sterującej ***while*** ma następującą postać:

while *warunek*
      podprogram pętli;
end

Podprogram zawarty pomiędzy słowami kluczowymi ***while*** i ***end*** jest wykonywany dopóki jest

spełniony *warunek* określony po słowie kluczowym ***while***.

```
a=3
while(a>0)
      cumsum(A(a,:));
      a=a-1;
end
```

Stosując instrukcję ***while*** należy pamiętać, że nie wszystkie liczby rzeczywiste mają dokładną reprezentację zmiennoprzecinkową. Dlatego korzystając z operatorów logicznych == i ~= trzeba zwracać uwagę, czy nie została zbudowana nieskończona pętla, tak jak w poniższym przykładzie:

```
a=0
while a~=10
      a = a+0.01;
end
```

Alternatywą dla stosowania pętli ***for*** i ***while*** są operacje wektorowe. Czas obliczeń potrzebnych na wykonanie obliczeń z wykorzystaniem wektorów zwykle jest wielokrotnie krótszy w stosunku do obliczeń w zagnieżdżonych pętlach. W poniższym przykładzie zaprezentowano wyliczanie pierwiastka kwadratowego z 20000 liczb naturalnych. Przy pomocy instrukcji ***tic*** i ***toc*** zmierzono czasu wykonania operacji:

```
tic;
a=1:20000;
p=sqrt(a);
czas_wykonania_wektor = toc;

tic;
for i=a
      p2(i)=sqrt(a(i));
end
czas_wykonania_for = toc;
```

Czas wykonania operacji wektorowej (`czas_wykonania_wektor`) wyniósł 0.0235s. W przypadku pętli ***for*** było to 0.3708s (`czas_wykonania_for`).

Składnia wyrażenia warunkowego ***if-elseif-else*** jest następująca:

if *warunek*
    linie podprogramu
elseif *warunek*
    linie podprogramu
else
    linie podprogramu
end

Ze względu na typową konstrukcję wyrażenia ***if-elseif-else*** przykład został pominięty.

Instrukcja ***break*** powoduje przerwanie wykonywania pętli ***for*** lub ***while*** i opuszczenie pętli.

```
k = 1;
while k
  k = (k+1)^2;
    if k>50
      break
    end
end
```

Instrukcja ***try*** jest wykorzystywana do sprawdzenia, czy określona komenda lub fragment kodu źródłowego nie posiadają błędów. Ciąg instrukcji pomiędzy ***try*** a ***catch*** jest wykonywany do momentu wystąpienia błędu. Składania wyrażenia ***try-catch*** jest następująca:

try
    podprogram
catch
    podprogram
end

Jeżeli błąd wystąpi jest wykonywany blok ***catch*** – fragment kodu zawarty pomiędzy ***catch*** a ***end***.

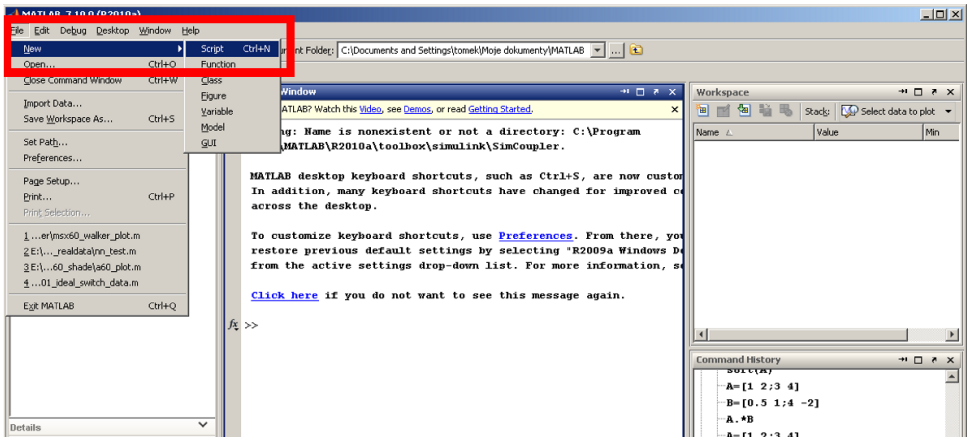
Instrukcja ***return*** powoduje bezwarunkowe opuszczenie danej funkcji lub skryptu i powrót do miejsca jej wywołania.

Wyrażenie ***switch-case*** działa na zasadzie przełącznika wybierającego do realizacji fragment kodu występujący bezpośrednio po dobranym do wartości zmiennej występującej po słowie kluczowym ***switch*** wyrażeniu poprzedzonym słowem kluczowym ***case***. Znalezienie pierwszego dopasowania powoduje wykonanie kodu źródełowego znajdującego się pomiędzy dwoma słowami kluczowymi ***case***, a następnie przejście programu do instrukcji ***end***. W Matlabie wyrażenia ***case*** nie są kończone słowem kluczowym ***break***. Należy zwrócić uwagę, że wyrażenie „domyślne” nie jest poprzedzane w Matlabie słowem kluczowym „default” (jak w języku C) tylko ***otherwise***.

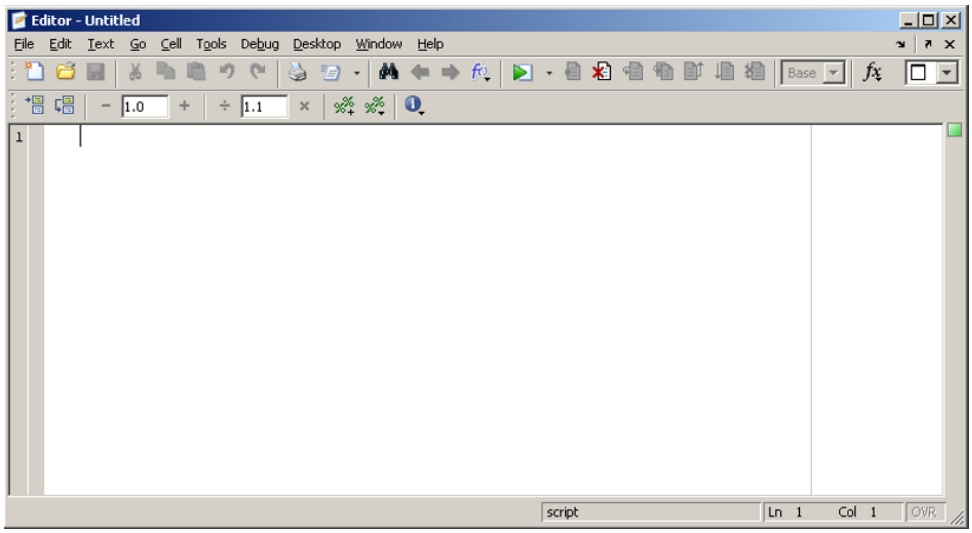
```
wynik = 10;

switch wynik
case {5,15}
      disp('Wynik: 5 lub 15') % disp wyświetla łańcuch znaków na ekranie
case 10
      disp('Wynik: 10')
case 25
      disp('Wynik: 25')
otherwise
      disp('Wynik: nieprawidlowy')
end
```

12. Skrypty

Z praktyki programowania wynika, że pewne sekwencje instrukcji mogą być wykorzywanie w różnych programach. Dlatego po ich wprowadzeniu i sprawdzeniu warto jest je zapisać, aby móc się później do nich w razie potrzeby odwołać. W Matlabie użytkownik może wykorzystać do tego celu dwa narzędzia: skrypty i funkcje. Skrypt jest plikiem tekstowym zawierającym sekwencje poleceń, które są tożsame z poleceniami wprowadzanymi w oknie komend (***Command Window***). Nie musi on spełniać żadnych wymogów formalnych poza poprawnościa składniową i semantyczną znajdujących się w nim instrukcji. W Matlabie przyjęto dla skryptów rozszerzenie ***.m***. Skrypt można utworzyć w dowolnym edytorze tekstowym, ale takim, który nie wprowadza dodatkowych znaków formatujących i sterujących. Można także skorzystać z wbudowanego edytora skryptów (zwanego w starszych wersjach Matlaba edytorem m-plików). Edytor skryptów jest uruchamiany kombinacją klawiszy ***Ctrl+n*** lub przy pomocy polecenia ***New*** → ***Script*** (***New*** → ***Blank m-file*** w straszych wersjach Matlaba) z menu ***File***.



Okno edytora skryptów wygląda następująco:



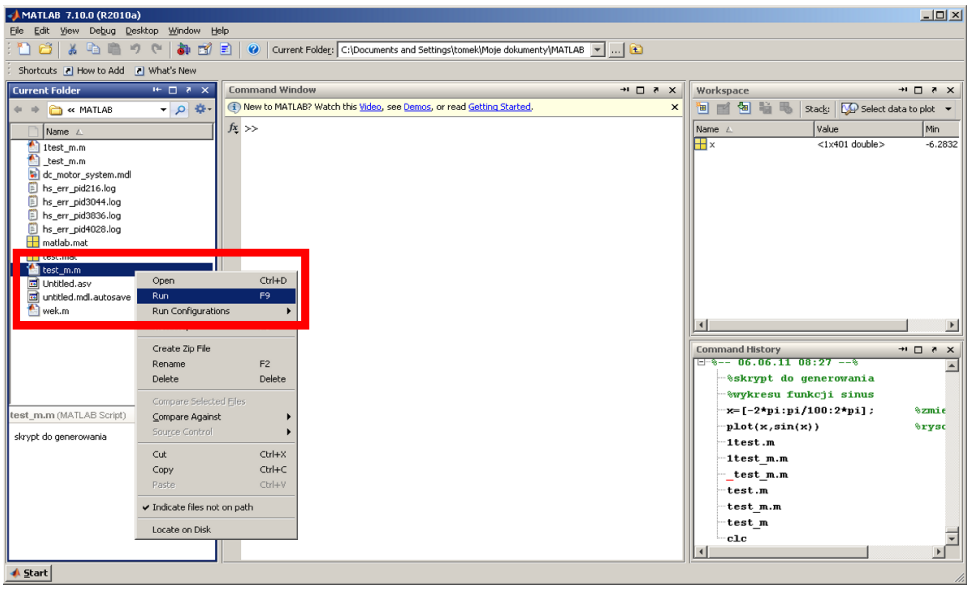
Przykładowy skrypt, którego zadaniem będzie rysowanie wykresu funkcji *sin(x)* w przedziale od *-2π* do *2π* może mieć następującą postać:

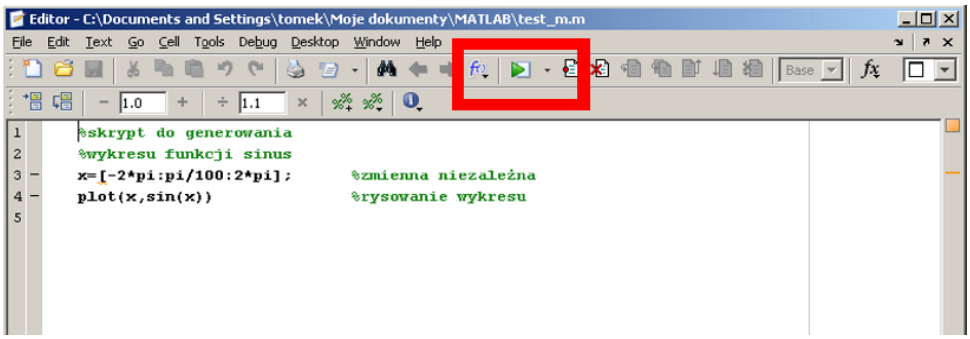
```
%skrypt do generowania
%wykresu funkcji sinus
x=[-2*pi:pi/100:2*pi];        %zmienna niezalezna
plot(x,sin(x))                %rysowanie wykresu
```

Znakiem **%** poprzedza się w skrypcie komentarze. Jeżeli dana linia skryptu będzie zakończona średnikiem(**;**) to przy wywołaniu skryptu nie zostanie wyświetlona w oknie komend Matlaba. Pierwszy blok komentarzy jest wyświetlany na ekranie jako pomoc związana z danym skryptem (po wywołaniu polecenia ***help*** *nazwa_skryptu*). Komentarz umieszczony w wierszu oddzielonym odstępem nie będzie wyświetlony na ekranie:

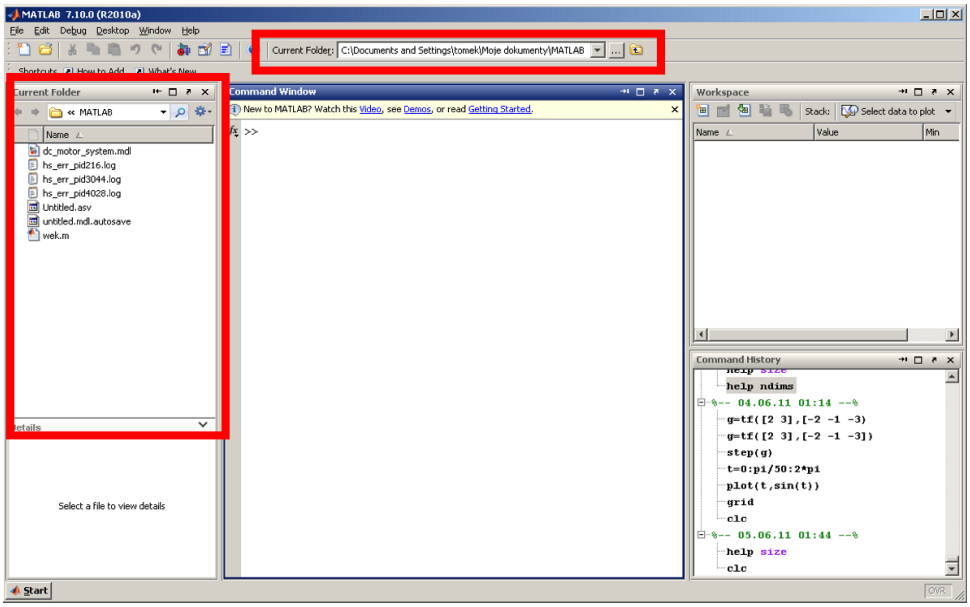
```
%Ten komentarz bedzie
%widoczny na ekranie przy
%wywolaniu polecenia
%help nazwa_pliku

%a ten fragment juz NIE!!
```

Po utworzeniu skryptu należy go zapisać przy pomocy polecenia ***File*** → ***Save As...***. Skrypt można uruchomić wpisując jego nazwę (bez roszerzenia .m) w oknie komend, podświetlając plik ze skryptem w oknie ***Current Folder*** i wybierając z menu kontekstowego opcję ***Run*** lub korzystając z przycisku w edytorze.





Przy nadawaniu nazw skryptom dobrze jest stosować te same zasady jak przy nadawaniu nazw zmiennym. Ponadto należy uważać, żeby w ścieżkach dostępu do plików ze skryptami oraz w nazwach skryptów nie występowały znaki narodowe. Skrypty są zapisywane w katalogu określonym jako ***Current folder***:



13. Funkcje

W przeciwieństwie do skryptów struktura pliku funkcyjnego jest ściśle zdefioniowana i przedstawia się następująco:

> ***function****[y1,y2,y3,....] = nazwa_funkcji(x1,x2,x3,.....)*
> *% komentarz wyświetlany po wywołaniu help nazwa_funkcji*
> *instrukcje*

Plik funkcyjny rozpoczyna się słowem kluczowym ***function*** po którym występuje zbiór zmiennych wyjściowych *[y1,y2,....]*. W przypadku, gdy dana funkcja ma jedną zmienna wyjściową (np.

oznaczoną jako y) struktura funkcji wygląda następująco:

function *y = nazwa_funkcji(x1,x2,x3,.....)*
% komentarz wyświetlany po wywołaniu help nazwa_funkcji
instrukcje

lub

function*[y] = nazwa_funkcji(x1,x2,x3,.....)*
% komentarz wyświetlany po wywołaniu help nazwa_funkcji
instrukcje

Dobrym zwyczajem jest przyjmowanie nazwy funkcji (*nazwa_funkcji*) identycznej z nazwą pliku, w którym funkcja jest umieszczona(zapisana). Po nazwie funkcji występuje zbiór danych wejściowych – *(x1,x2,x3,....)*. Komentarz umieszczony bezpośrednio po linii zawierającej słowo kluczowe ***function*** jest wyświetlany w przypadku wywołania instrukcji ***help*** której parametrem będzie *nazwa_funkcji*:

```
help nazwa_funkcji
```

Linie pomocy (***help***) kończy dowolna linia rozpoczynająca się znakiem innym niż ***%***.

Blok *instrukcje* jest zasadniczą treścią pliku funkcyjnego. Zawartość tego bloku jest podobna do zawartości skryptu. Wszystkie występujące w tym bloku komendy i wyrażenia mogą używać zmiennych wejściowych przekazywanych funkcji w momencie jej wywołania. Przykład funkcji obliczającej pole trójkąta na podstawie długości jego boków może wyglądać następująco:

```
function P = area(a,b,c)
% Obliczanie pola trójkata
% Parametry wejsciowe:
%      a,b,c - dlugoœci bokow
% Parametr wyjsciowe:
%      P - pole trojkata
t = (a + b + c)/2;
P = sqrt(s*(s - a)*(s - b)*(s - c));
```

Wywołanie funkcji odbywa się poprzez podanie nazwy funkcji oraz argumentów wejściowych:

```
area(5,6,7)             % obliczenie pola powierzchni
ans =
      14.6969

pole = area(5,6,7)      % przypisanie pola powierzchni do zmiennej pole
pole =
      14.6969
```

Rezultat polecenia ***help*** będzie następujący:

```
help area

Obliczanie pola trojkata
Parametry wejsciowe:
```

```
      a,b,c - dlugosci bokow
Parametr wyjsciowe:
      P - pole trojkata
```

Przykład funkcji obliczającej pole i obwód trójkąta, a więc zwracającej wartość dwóch parametrów wyjściowych:

```
function [P,O] = pole_i_obwod(a,b,c)
%Parametry wejsciowe:
%     a,b,c - dlugosci bokow
%Parametr wyjsciowe:
%     P - pole trojkata
%     O - obwod trojkata
t = (a + b + c)/2;
O = 2*t;
P = sqrt(t*(t - a)*(t - b)*(t - c));
```

Wywołanie funkcji:

```
[pole,obwod] = pole_i_obwod(5,6,7)

pole =
      14.6969
obwod =
       18
```

W przypadku nieprzypisania wyników obliczeń uzyskanych poprzez wywołanie funkcji do odpowiednich zmiennych zwrócona zostanie wartość pierwszego z parametrów wyjściowych

```
pole_i_obwod(5,6,7)

ans =
   14.6969
```

Podobnie będzie w przypadku przypisania:

```
P = pole_i_obwod(5,6,7)

P =
   14.6969
```

Wszystkie zmienne występujące wewnątrz pliku funkcyjnego mają charakter zmiennych lokalnych. W związku z tym są niedostępne poza daną funkcją. Dostęp do tych zmiennych można uzyskać w innych plikach jeżeli zostaną zadeklarowane jako zmienne globalne przy pomocy komendy ***global***.

14. Wykresy funkcji jednej zmiennej

Wszystkie operacje graficzne w Matlabie są wykonywane w oddzielnych oknach graficznych, które noszą nazwę *figure*. Wyświetlenie takiego okna następuje w sposób automatyczny, w przypadku wywołania funkcji przeznaczonej do tworzenia grafiki. Inną metodą utworzenia pustego okna graficznego jest użycie funkcji ***figure*** z parametrem:

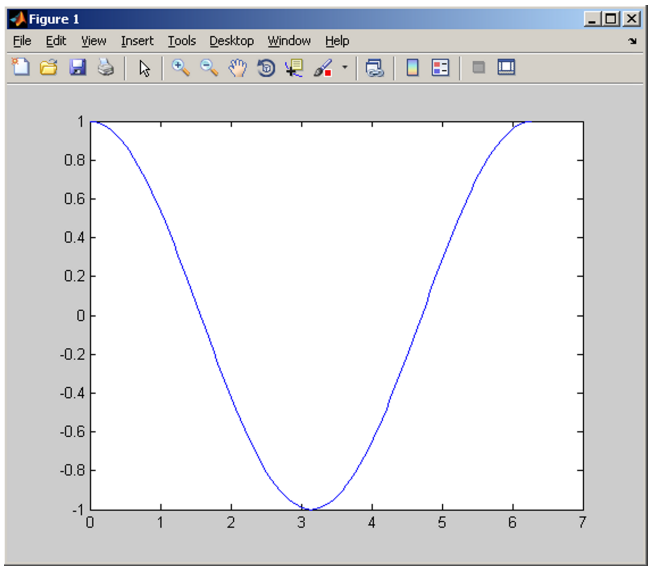
```
figure(5)
```

W przypadku, gdy okno o danym numerze istnieje, po wywołaniu funkcji ***figure*** staje się oknem aktywnym. Oznacza to, że w nim będą wyświetlanie kolejne rysunki lub wykresy. Przełączania pomiędzy aktywnymi oknami dokonujemy korzystając z polecenia ***figure***, którego parameterm jest numer nowego aktywnego okna. Numeracja kolejnych okien graficznych nie musi być ciągła.

Wykres funkcji jednej zmiennej można wykreślić korzystając z polecenie ***plot***:

```
x=0:pi/50:2*pi;
plot(x,cos(x))
```

Argumentami funkcji ***plot*** są 2 wektory – wektor zmiennej niezależnej oraz wektor wartości funkcji w punktach określonych przez wektor zmiennej niezależnej. Poniższy rysunek zawiera wygląd typowego okna graficznego.



Ponowne wywołanie funkcji ***plot*** (z innymi argumentami) spowoduje skasowanie zawartości okna graficznego, a następnie wyświetlenie nowego wykresu. Aktywne okno graficzne można wyczyścić korzystając z funkcji ***clg***. Do zamknięcia okna graficznego służy opcja ***close***, której argumentem

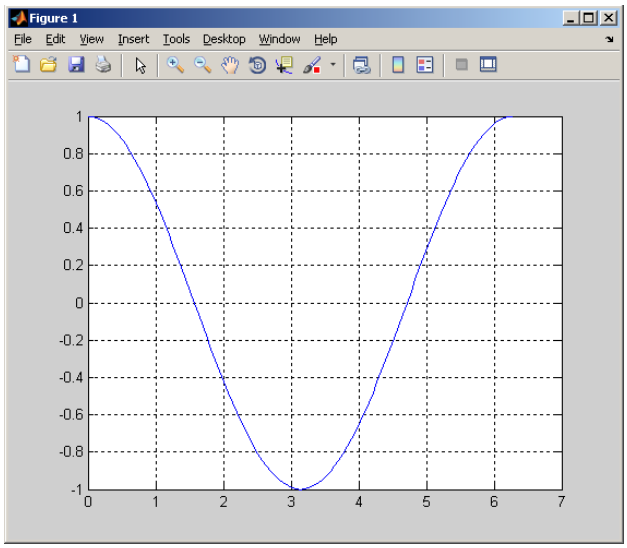
jest numer okna:

```
close(1)
```

Wywołanie funkcji ***close*** z argumentem *all* wymusi zamknięcie wszystkich okien graficznych.

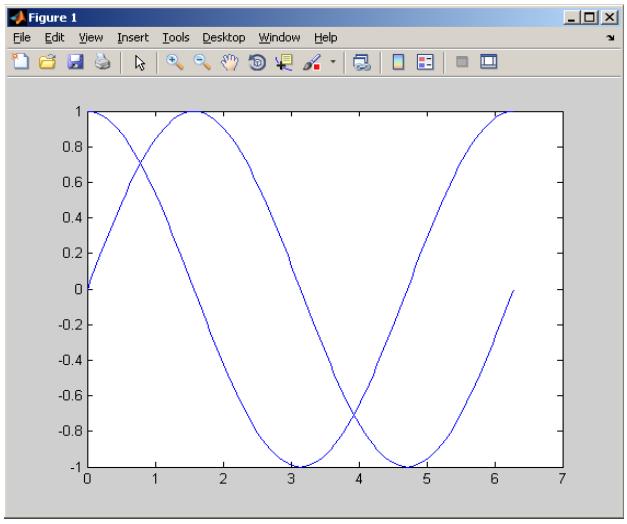
```
close all
```

Do poprawy czytelności danych często stosuje się pomocniczą siatkę danych. Do jej włączenia służy polecenie ***grid on***. Polecenie ***grid off*** wyłącza widok siatki.



Jak wspomniano ponowne wywołanie instrukcji ***plot*** spowoduje skasowanie zawartości aktywnego okna i narysowanie w nim kolejnego wykresu. Do rysowania wielu wykresów we wspólnym układzie współrzędnych można wykorzystać funkcję ***hold***. Funkcja ***hold*** wywołana z parametrem *on* spowoduje zatrzymanie bieżącego wykresu w oknie graficznym, a ***hold off*** wyłączenie zatrzymywania starych wykresów w aktywnym oknie graficznym.

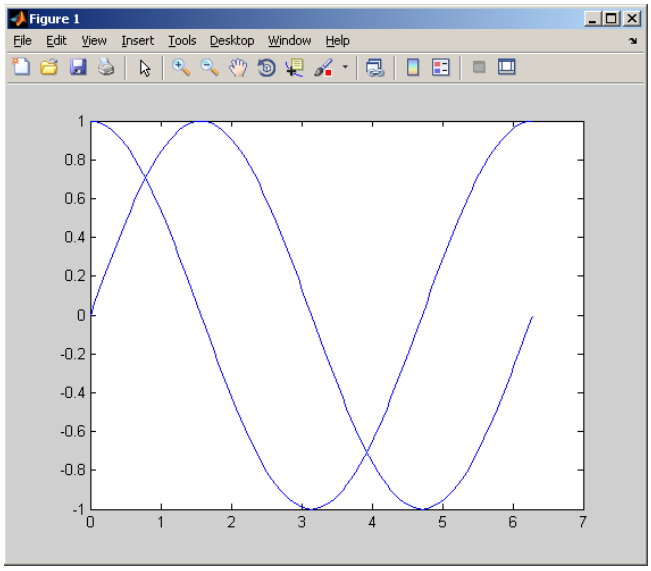
```
=0:pi/50:2*pi;
plot(x,cos(x))
hold on
plot(x,sin(x))
```



Inną metodą wykreślenia dwóch lub więcej zależności we wspólnym układzie współrzędnych jest wywołanie funkcji ***plot*** w formie:

plot*(x1,y1,x2,y2,....)*

```
x1 = 0:pi/50:2*pi;
y1 = cos(x1);
x2 = 0:pi/50:2*pi;
y2 = sin(x2);
plot(x1,y1,x2,y2)
```



Użytkownik posiada wpływ na rodzaj linii jaką będzie rysowany wykres funkcji poprzez parametr *rodzaj_linii*.

*plot(x1,y1,'**rodzaj_linii**',x2,y2,'**rodzaj_linii**',....)*

Paramer *rodzaj_linii* jest ciągiem znaków składającym się z jednego, dwóch lub trzech znaków, które opisują kolor linii, marker danych oraz rodzaj linii. Zestawienie dopuszczalnych wartości znajduje się w poniższej tabeli:

kolor linii		**marker danych**		**rodzaj linii**	
znak	**kolor**	**znak**	**wygląd punktu**	**znak**	**linia**
y	żółty	.	punkt	-	ciągła
m	fioletowy	o	okrąg	:	kropkowana
c	turkusowy	x	znak x	-.	kreska-kropka
r	czerwony	+	plus	--	przerywana
g	zielony	*	gwiazdka	„spacja”	brak linii
b	niebieski	s	kwadrat		
w	biały	d	rąb		

k	czarny	v	trójkąt		
		^	trójkąt		
		<	trójkąt		
		>	trójkąt		
		p	pięciokąt		
		h	sześciokąt		

```
plot(x1,y1,'r:')  % wykres zostanie narysowany linią czerwoną kropkowaną

plot(x2,y2,'k+')  % w punktach danych zostaną umieszczone czarne znaki +

plot(x,sin(x),'k*-')    % wykres zostanie narysowany linią czarną
                        % przerywaną, punkty danych zostaną oznaczone
                        % czarnym znakiem *
```

Pewną niedogodnością stosowania kombinacji pleceń ***plot*** i ***hold*** jest wykreślanie wszystkich zależności we wspólnym układzie współrzędnych. W przypadku dużych rozbieżności w wartościach jakie przyjmują poszczególne funkcje może to prowadzić do trudności w interpretacji uzyskanych wyników graficznych. Możliwość wykreślenia kliku zależności w różnych układach współrzędnych, w jednych oknie graficznym oferuje funkcja ***subplot***. Funkcja ***subplot*** jest wywoływana z trzema parametrami:

subplot*(n,m,p)*

gdzie:

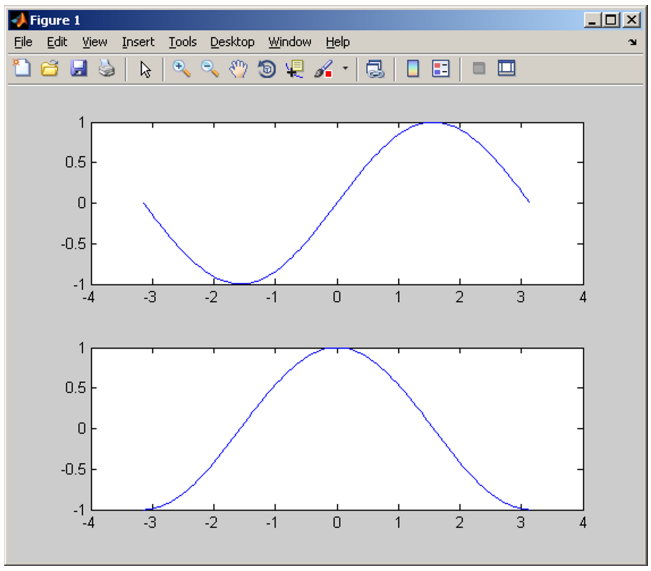
n – ilość wykresów w poziomie (ilość wierszy),

m – ilość wykresów w danym poziomie (wierszu),

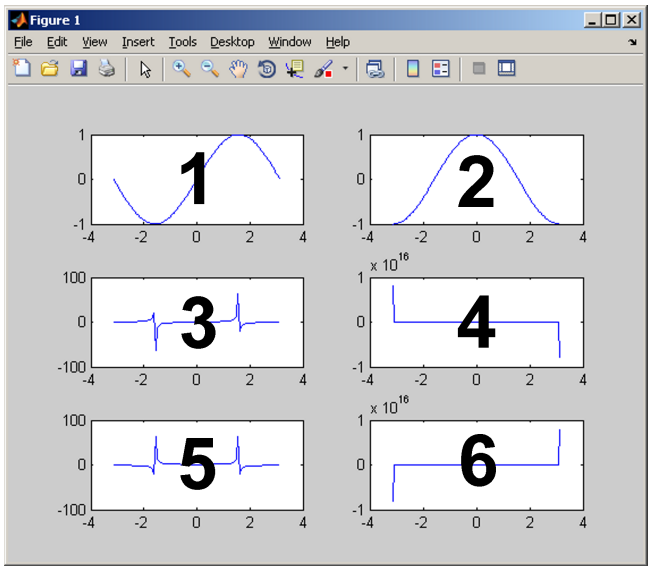
p – numer aktywnego wykresu na danym poziomie (wieszu).

Sekwencja poleceń, która umożliwi wyświetlenie przebiegu 2 funkcji trygonometrycznych przedziale od -π do π w jednym oknie graficznym będzie następująca:

```
x=linspace(-pi,pi,100)
subplot(2,1,1)
plot(x,sin(x))
subplot(2,1,2)
plot(x,cos(x))
```

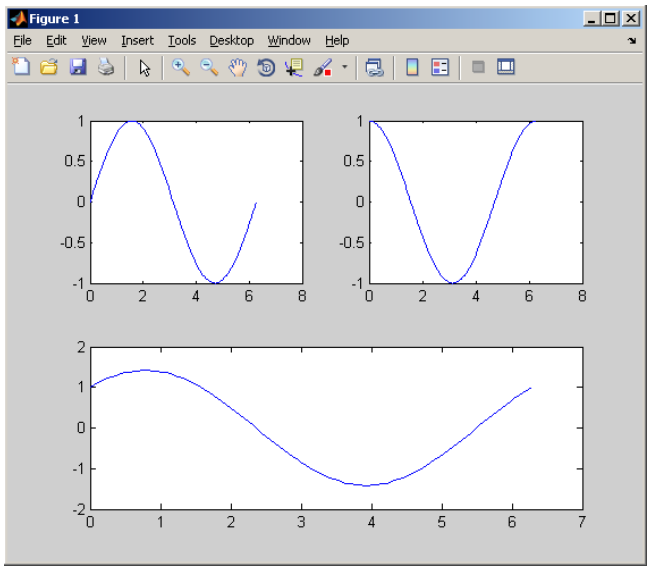


Kolejność numerowania wykresów została przedstawiona na rysunku:



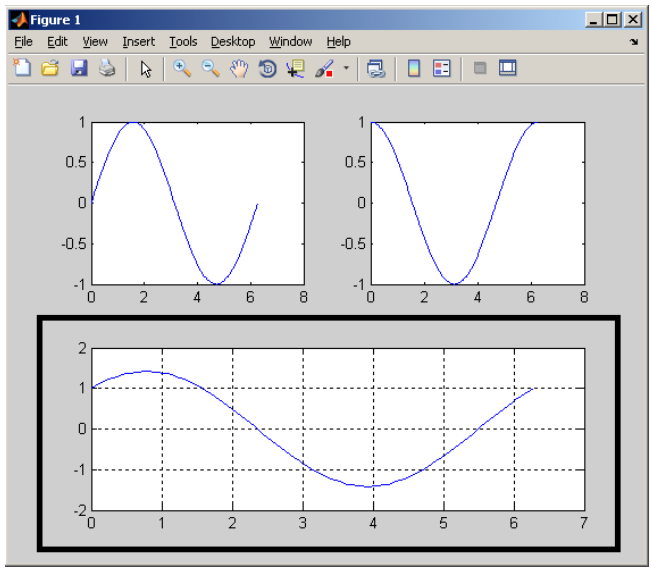
Ilość wykresów w wierszach nie musi być jednakowa:

```
x=0:pi/50:2*pi;
subplot(2,2,1);          % podział aktywnego okna graficznego (2 x 2)
plot(x,sin(x));
subplot(2,2,2);
plot(x,cos(x));
subplot(2,1,2);          % nowy podzial aktywnego okna graficznego (2 x 1)
plot(x,sin(x)+cos(x));
```



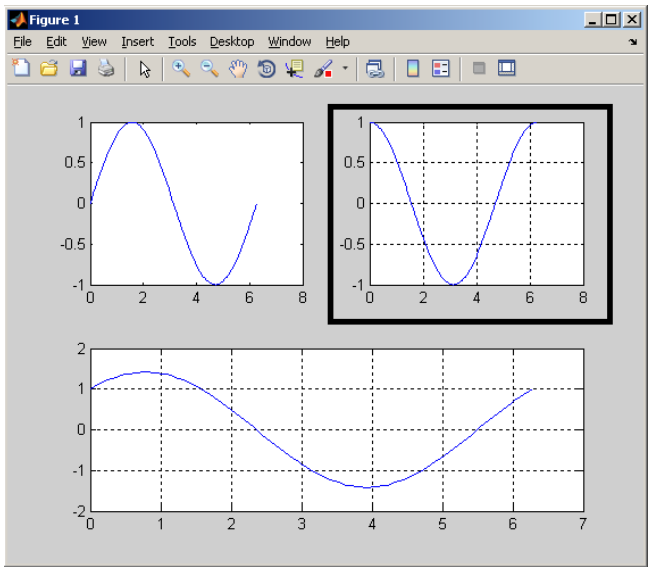
Polecenie ***grid*** sterujące wyświetlanie pomocniczej siatki odnosi się do układu współrzędnych wskazanego poprzedzającym je poleceniem subplot. W rozpatrywanym przykładzie siatka zostanie naniesiona na wykres funkcji *sin(x) + cos(x)*:

```
subplot(2,1,2);          % nowy podzial aktywnego okna graficznego (2 x 1)
plot(x,sin(x)+cos(x));
grid
```



W celu naniesienia pomocniczej siatki na inny z wykresów, należy go wskazać poleceniem subplot:

```
subplot(2,2,2);
grid
```



Oprócz funkcji ***plot*** do rysowania wykresów dwuwymiarowych można wykrzystać następujące funkcje:

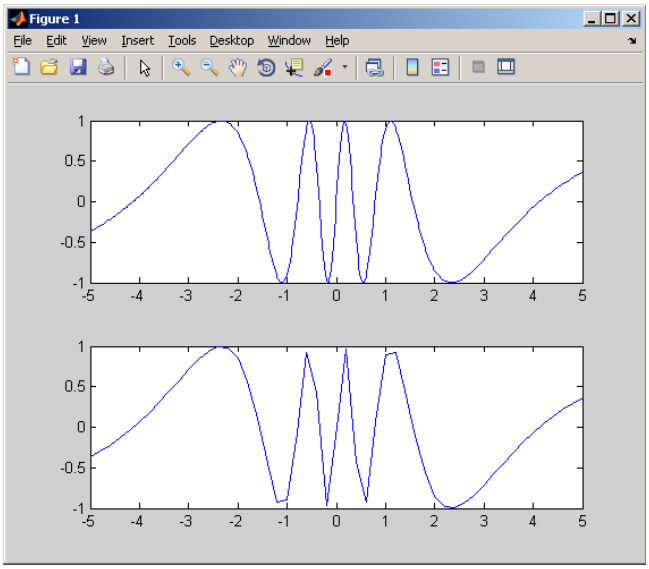
Funkcja	**Opis**
loglog	Skala na obydwu osiach jest logarytmiczna. Argumenty tak jak *plot*.
semilogx	Skala na osi odciętych (x) jest logarytmiczna. Argumenty tak jak *plot*.
semilogy	Skala na osi rzędnych (y) jest logarytmiczna. Argumenty tak jak *plot*.
plotyy	Oddzielne osie rzędnych (y) dla dwóch wykresów.
fplot	Funkcja rysująca wykres funkcji o nazwie określonej parametrem *f*. Punkty, w których dokonywane są obliczenia wartości rysowanej funkcji obliczane są automatyczne, tak aby wykres odzwierciedlał dynamikę zmian wartości funkcji. Funkcja fplot posiada 5 parametrów – ***fplot****(f,zakres,n,kat,podprzedzialy)*: • *f* – łańcuch znaków określający nazwę pliku zawierającego rysowaną funkcję; • *zakres* – wektor dwuelementowy zawierający granice przedziału w jakim ma być narysowany wykres; • *n* – parametr opcjonalny określający minimalną ilość punktów jaka zostawnie wzięta pod uwagę przy rysowaniu wykresu; • *kat* – parametr opcjonalny określający wartość kąta (w stopniach) pomiędzy sąsiednimi odcinkami wykresu powyżej której zwiększania jest liczba punktów próbkowania; • *podprzedzialy* – parametr opcjonalny – maksymalna liczba punktów próbkowania jaka może zostać dodana w przypadku dużej dynamiki zmian wartości funkcji.

Przykład pliku zawierającego funkcję f – *funkcja.m*:

```
function y = funkcja(x)
y = sin(3 * pi * atan(x));
```

Porównanie działania funkcji *plot* i *fplot*:

```
subplot(2,1,1);
fplot('funkcja',[-5,5]);
subplot(2,1,2);
x = -5:0.2:5;
plot(x,funkcja(x));
```



W poniższej tabeli zamieszczono listę poleceń wspomagających tworzenie i opisywanie wykresów:

Funkcja	Opis	Przykład
title	dodatnie tytułu wykresu	`title('Ch-ka mocy')`

axis	funkcja umożliwiająca modyfikację układu współrzędnych; argumentem funkcji może być 4-elementowy wektor o postaci [*xmin xmax ymin ymax*], którego elementy określają zakresy skal poszczególnych osi lub łańcuch znaków zmieniający sposób skalowania wykresu; szczegółowy opis stałych skalujących znajduje się w dokumentacji do pakietu Matlab oraz na stronie www firmy Mathworks; funkcja axis wywołana bez parametrów zwraca 4-elementowy wektor zawierający zakresy skal dla poszczególnych osi układu współrzędnych	`axis([-4 4 0 10])`
		`axis('auto')`
		`axis('square')`

	• axis auto skalowanie automatyczne • axis off wyłączenie rysowania osi • axis on włączenie rysowania osi	
xlabel	funkcja wyświetla łańcuch znaków, pod poziomą osią aktywnego układu współrzędnych	`xlabel('voltage [mv]')`
ylabel	funkcja wyświetla łańcuch znaków obok pionowej osi aktywnego układu współrzędnych	`ylalel('current [mA]')`

text	funkcja wyświetla łańcuch znaków na aktywnym wykresie, w miejscu określonym współrzędnymi, będącymi 2 i 3 parametrem funckji; jeżeli współrzędne dane są w postaci wektora to napis będzie wyświetlony we wszystkich punktach wskazanych przez elementy wektorów	`x=5;`
		`y=35;`
		`text('Ch-ka I-V',x,y)`
		
		`x=[15 50];`
		`y=[50 125];`
		`text('Ch-ka I-V',x,y)`

textlabel	dodanie opisu tekstowego w formacie TeX lub LaTex	`texlabel('sin(sqrt(x^2))/sqrt(x ^2 + y^2)')`
grid on	naniesienie na wykres pomocnicznej siatki punktów;	
grid off	wyłączenie pomocnieczej siatki punktów	
grid	cykliczne przełączanie pomiędzy poleceniami *grid on* i *grid off*	
legend	dodanie opisów poszczególnych wykresów	`plot(x,sin(x),'-',x,cos(x),':')` `legend('sin(x)','cos(x)')`

14. Pomoc

Do wyświetlenia instrukcji do danej funkcji służy polecenie ***help***. Składa tego polecenia jest nastąpująca:

help *nazwa_funkcji*

```
help sin

SIN    Sine of argument in radians.
   SIN(X) is the sine of the elements of X.

   See also asin, sind.

   Reference page in Help browser
     doc sin
```

Dostęp do plików pomocy można także uzyskać poprzez przycisk *Start* → *Matlab* → *Help*.

15. Literatura

1. A.Zalewski, R.Cegieła; *Matlab – obliczenie numeryczne i zastosowanie*; Wydawnictwo Nakom;
2. S.Osowski, A.Cichocki, K.Siwek; *MATLAB w zastosowaniu do obliczeń obwodowych i przetwarzaniu sygnałów*; Oficyna wydawnicza Politechniki Warszawskiej
3. *Mathworks MATLAB Product Documentation – Function Reference*