MATLAB

Materiały pomocnicze do ćwiczeń z Podstaw Informatyki

Wydział Inżynierii Mechanicznej i Robotyki AGH w Krakowie

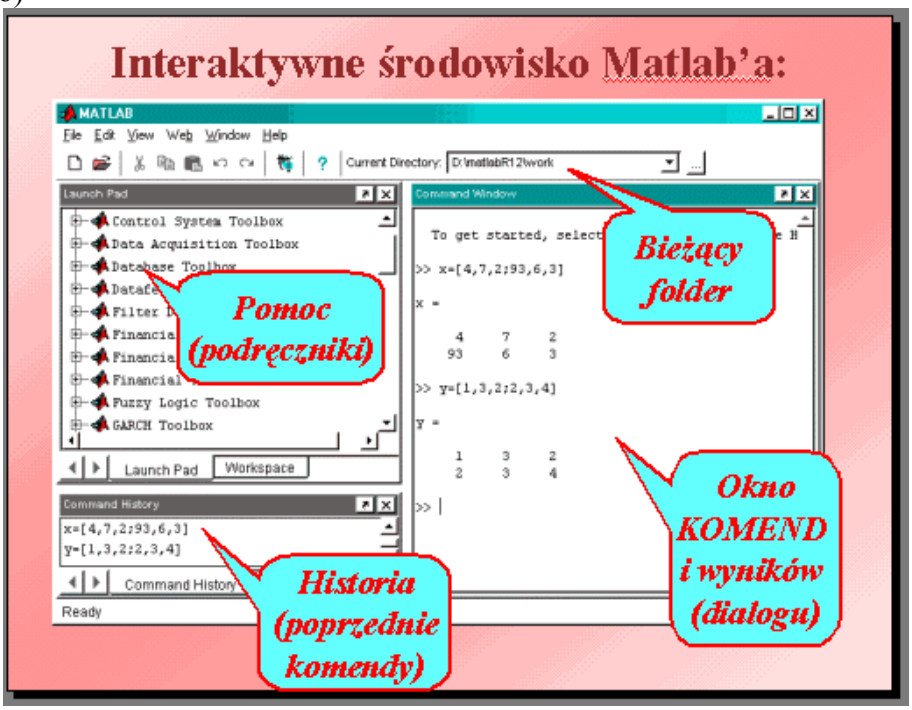
Opracował: dr inż. Zbigniew Rudnicki
(Wersja z dnia 6 maja 2004)

1. Wprowadzenie

1.1 Co to jest MATLAB

MATLAB (od angielskiej nazwy: MATrix LABoratory) to pakiet oprogramowania matematycznego firmy MathWorks Inc. (rozwijany od roku 1984)
zawiera:

- język i środowisko programowania do obliczeń naukowo-technicznych
  oraz
- obszerny zestaw tematycznych bibliotek podprogramów (toolbox'ów)
- i wiele tysięcy stron (na CD) podręczników, przykładów i aplikacji demonstracyjnych (demo)



1.2 Dlaczego warto poznać MATLABa?

bo:

- jest powszechnie nauczany na uczelniach Świata
- jest łatwy (jak BASIC a nie jak C czy C++)
- staje się najczęściej używanym narzędziem w badaniach naukowych
- posiada bardzo obszerną i przystępnie napisaną dokumentację (w j. angielskim), oraz przykłady i system pomocy.
- specjalistyczne „toolbox'y" czynią go narzędziem dostosowanym do prawie każdej dziedziny

- pozwala poznawać metody matematyczne w praktyce
- pozwala tworzyć wykresy, animacje, aplikacje, ...
- jest stale rozwijany i wzbogacany

Jest coraz więcej książek o Matlabie w języku polskim a między innymi:

1. A.Kamińska, B.Pańczyk: „Matlab - przykłady i zadania" - wyd. Mikom 2002, z serii „ćwiczenia z..." (150 stron)
2. J.Brzózka, L.Dorobczyński: „Programowane w Matlab", wyd.Mikom 1998. (314 stron)
3. B.Mrozek, Zb.Mrozek: MATLAB i Simulink. Poradnik użytkownika. wyd.Helion 2004
4. Marcin Stachurski: Metody numeryczne w programie Matlab. wyd.MIKOM 2003
5. Wiesława Regel: Statystyka matematyczna w Matlab. wyd.MIKOM 2003
6. Wiesława Regel: Wykresy i obiekty graficzne w MATLAB. wyd.MIKOM 2003
7. B.Mrozek, Zb.Mrozek: „MATLAB 5.x, Simulink 2.x"., wyd. PLJ 1998
8. B.Mrozek, Zb.Mrozek: „MATLAB uniwersalne środowisko obliczeń naukowo-technicznych". PLJ 1996
9. Z.Wróbel, R.Koprowski: „Przetwarzanie obrazu w programie MATLAB". Wyd. Uniw. Śl., K-ce 2001

1.3 Niektóre cechy MATLABa (wersja 6):

- Przyjazne dla użytkownika, interakcyjne środowisko
- Język programowania wysokiego poziomu
- Zbiór (ok.30) toolbox'ów - zestawów procedur i funkcji
- Zbiór podręczników (ok.70 x kilkaset stron, 433MB)
- MATLAB umożliwia m.in:
  - wykonywanie obliczeń naukowych i inżynierskich,
  - modelowanie i symulację,
  - analizę danych (w tym: sygnałów i obrazów)
  - graficzną wizualizację danych i wyników obliczeń.
- Podstawowym typem danych w MATLABie jest tablica (macierz) o elementach rzeczywistych lub zespolonych.

1.4 Łagodny start - kalkulator

W pierwszych spotkaniach z Matlabem możemy wypróbować jego najprostsze możliwości w działaniach podobnych jak na kalkulatorze.
W oknie komend widać znak gotowości do przyjmowania komend :
>>
Wpisanie 2+3 daje od razu wynik:

```
>> 2+3
ans =
     5
>>
```

Nie wstawiliśmy wyniku do żadnej zmiennej dlatego MATLAB użył zmiennej „ans"

Możemy wyniki obliczeń podstawiać do zmiennych np.:

```
>> x=sin(pi/2)
x =
     1
>>
```

Komenda zakończona średnikiem wykona się lecz nie będzie wyświetlony jej wynik:

```
>> x=sin(pi/2);
>>
```

Wpisanie samej nazwy zmiennej (w dowolnym momencie) wyświetli jej aktualną wartość:

```
>> x
x =
     1
```

2. Wyrażenia i ich składniki – stałe, zmienne, działania i funkcje matematyczne. Powtarzanie i poprawianie komend

2.1 Objaśnienia

Wyrażenia - podobnie jak w innych językach - mogą zawierać:

- stałe
- zmienne (nazwy zmiennych)
- operatory działań
- funkcje

Jednak inaczej niż w innych językach - wyrażenia te dotyczą tablic (macierzy), które - jak wspomniano - w szczególności mogą być skalarami (pojedynczymi liczbami).

2.1.1 Stałe liczbowe

Podobnie jak w większości języków programowania **zapis liczb** w MATLABie może zawierać:

- początkowy znak plus lub minus
- **kropkę dziesiętną (NIE PRZECINEK!)** poprzedzającą część ułamkową np.: -97.6397
- może być stosowana tzw. **notacja naukowa** w której **e oznacza "dziesięć do potęgi ..."** np.: **-1.60210e-23** oznacza: -1.60210 razy 10 do potęgi -23
- w zapisie liczb urojonych i zespolonych stosuje się symbole i oraz j np.: **1i -3.14159j 3e5i**

Liczby rzeczywiste mają określony zakres (od stałej **realmin** do **realmax**), w przybliżeniu: **+-(10^{-308} do 10^{+308})** i są pamiętane z dokładnością ok. 15-16 cyfr znaczących.

2.1.2 Format liczb

Postać prezentowania liczb można zmieniać przy pomocy dyrektywy:

format *parametr*

gdzie *parametr* jest jednym ze słów:

- **short**
- **short e**
- **long**

Na przykład:

Wpisana liczba w domyślnym formacie **short**	Po zmianie na format **short e**	Po zmianie na format **long**
» 2.5 ans = 2.5000	» format short e » 2.5 ans = 2.5000e+000	» format long » 2.5 ans = 2.50000000000000

2.1.3 Typy i nazwy zmiennych

- Wszystkie zmienne w MATLABie sa traktowane jak macierze
- Wektory i skalary są uważane za szczególne przypadki macierzy
- Nazwy zmiennych rozpoczynają się od litery, a po niej mogą być litery, cyfry i znaki podkreślenia
- Pamiętanych jest 19 pierwszych znaków
- MATLAB rozróżnia duże i małe litery. **Polecenia standardowe należy pisać małymi literami** a dla nazw własnych programów i zmiennych można używać małych i dużych liter

2.1.4 Deklarowanie zmiennych

- Deklarowanie typu i wymiarów macierzy odbywa się automatycznie - przez rozpoznanie rodzaju wpisanych wartości oraz maksymalnych wskaźników

2.1.5 Operatory działań arytmetycznych

Pamiętajmy, że operatory dotyczą nie tylko skalarów ale ogólnie macierzy. Operatory poprzedzone kropką nie reprezentują działań macierzowych lecz tablicowe to znaczy mają być zastosowane do odpowiadających sobie par elementów dwu macierzy. Obie macierze muszą wówczas mieć te same wymiary chyba że jeden z nich skalarem.
Oto operatory arytmetyczne w Matlabie:

Operator	Objaśnienie
+ lub .+	dodawanie skalarów lub odpowiadających sobie elementów macierzy (o takich samych wymiarach)
- lub .-	odejmowanie (j.w.) lub zmiana znaku
*	**mnożenie** skalarów lub mnożenie macierzowe
/	**dzielenie** j.w.
\	**dzielenie lewostronne** macierzy. Zamiast odwracania: inv(A)*B lepiej użyć A\B
^	**potęgowanie**
'	transponowanie macierzy (zamiana wierszy na kolumny)
.*	mnożenie tablicowe czyli odpowiadających sobie par elementów
./	dzielenie tablicowe
.^	potęgowanie tablicowe

2.1.6 Ważniejsze funkcje elementarne MATLAB'a

Nazwa funkcji	Objaśnienie
sqrt	pierwiastek
abs	wartość bezwzględna
exp	e do x
log	logarytm naturalny
log2	logarytm o podstawie 2
log10	logarytm o podst 10
sign	znak
mod	reszta z dzielenia
sin, cos, tan, cot	funkcje trygonometryczne
sinh, cosh, tanh, coth	hiperboliczne
asin, acos, atan, acot	odwrotne do trygonometrycznych
round	zaokrągla do najbliższej całkowitej
ceil	zaokrąglenie w górę (dosłownie: sufit)
fix	zaokrągla w stronę zera
floor	zaokrągla w dół (dosłownie: podłoga)
imag	część urojona liczby zespolonej
real	część rzeczywista liczby zespolonej
gcd	największy wspólny podzielnik
lcm	najmniejsza wspólna wielokrotność

Wykaz funkcji elementarnych można uzyskać wpisując: **help elfun**

2.1.7 Powtarzanie i poprawianie komend

Wpisana komenda zakończona naciśnięciem ENTER jest natychmiast wykonywana. Jeśli chcesz powtórzyć komendę lub ją poprawić przed ponownym wykonaniem to naciśnij (raz lub kilka razy) klawisz "strzałka w górę". Przywołuje to ostatnio wprowadzone komendy.

2.2 Obliczenia kalkulatorowe

Sprawdź w Matlabie poprawność obliczeń:

$\sin\left(\frac{\pi}{6}\right) = 0{,}5$ $\qquad$ $\frac{3{,}87 \cdot 10^5 - 72{,}86^3}{2 - \sqrt{2}} = 370{,}255$ $\qquad$ oblicz: $7{,}7^{310}$

Zastosuj tzw. „notację naukową" zapisu liczb w obliczaniu: $\frac{1{,}78 \cdot 10^{14} + 9{,}76^{11}}{128000000} + 0{,}00005 =$

2.3 Obliczenia z użyciem zmiennych

Oblicz wartość zmiennej: $y = \frac{2}{\sqrt{|x| - 4}} - \frac{6}{x^2 - 5x}$

dla x = 8,167 oraz x = –8,167
(Powinno być: 0,748 i 0,924)

3. Ciągi, wektory, macierze i niektóre działania na nich. Rozwiązywanie układu równań liniowych

3.1 Objaśnienia

Wektory i macierze mogą być wprowadzane przez:

1. wpisywanie (w oknie komend lub w m-plikach)
2. generowanie przez operatory, funkcje i komendy Matlab'a:
3. wczytywanie z pliku dyskowego

Przy wprowadzaniu, wyświetlaniu i operowaniu na wektorach i macierzach używane są znaki specjalne objaśnione w tabeli poniżej.

3.1.1 Znaki specjalne

Znaki	Objaśnienie
[ ]	w nawiasach prostokątnych umieszcza się wartości elementów macierzy
{ }	nawiasy klamrowe są używane przy definiowaniu tzw. macierzy komórkowych
( )	nawiasy okrągłe używamy w wyrażeniach oraz dla wskaźników macierzy i argumentów funkcji
:	dwukropek ma kilka znaczeń: 1) w wyrażeniu złożonym z trzech elementów połączonych dwoma dwukropkami na przykład: 5:2:13 oznacza: "ciąg od 5 z przyrostem 2 do 13" 2) w wyrażeniu złożonym z dwu elementów połączonych dwukropkiem: 5:10 oznacza: "ciąg od 5 do 10 domyślnie z przyrostem 1" 3) samodzielny dwukropek zamiast wskaźnika lub wskaźników macierzy zastępuje wszystkie wartości wskaźnika lub wskaźników na przykład jeśli macierz A ma wymiary 3x5 to zamiast pisać i=1:3; j=1:5; A(i,j) można napisać: A(:,:) lub A(:)
=	przypisuje zmiennej wartość wyrażenia n.p.: x=2*sin(pi/6)
.	kropka poprzedza część ułamkową liczby (lub nazwę pola rekordu)
,	przecinek rozdziela indeksy, argumenty funkcji lub poszczególne instrukcje (zamiast zmiany linii)
;	dajemy po instrukcjach jeśli nie chcemy wyświetlania wyników ich realizacji w przeciwnym przypadku kończymy instrukcje zmianą linii lub przecinkiem.
%	znak procentu poprzedza komentarze w programach (m-plikach)

Pomoc dotyczącą nawiasów można uzyskać wpisując: help paren

3.1.2 Wektor (ciąg) generowany jako postęp arytmetyczny

Aby wygenerować ciąg (postęp arytmetyczny) trzeba podać informacje w następującej formie:

nazwa_wektora = pierwszy element : przyrost lub ubytek : ostatni element

Na przykład: **x=0:0.2:1**

```
x =
    0    0.2000    0.4000    0.6000    0.8000    1.0000
```

Jeśli przyrost ma być równy 1 to można go pominąć w zapisie np.:

```
i = 1:6
 i =
   1   2   3   4   5   6
```

3.1.3 Wprowadzanie wektorów i macierzy

Wprowadzanie wektora:

Przykładem wektora (ciągu) wpisanego z klawiatury mogą być wyniki pomiarów napięcia w sieci elektrycznej w ciągu doby:

U = [220, 221, 220, 218, 218, 219, 220, 221]

Wprowadzanie macierzy:

Aby wpisać macierz w oknie komend należy **przecinkiem** oddzielać elementy wiersza a **średnikiem** oddzielać wiersze, np.:

```
>> A = [1, 2, 3,4; 7, 8, 9,10]
```

daje:

```
A =
   1   2   3   4
   7   8   9   10
```

3.1.4 Rola dwukropka w wybieraniu elementów macierzy

Podane już (w tabeli) funkcje dwukropka mogą być wykorzystywane do wybierania elementów tabel. Na przykład dla wprowadzonej wcześniej macierzy A:

A(: , 2) – dwukropek oznacza wiersze wszystkie a 2 to wybrana druga kolumna:

```
>> A(:,2)
ans =
   2
   8
```

A(:,2:3) – wybieramy kolumny 2 i 3:

```
>> A(:,2:3)
ans =
   2   3
   8   9
```

A(:,1:2:4) – wybieramy kolumny od 1 co 2 do 4

```
>> A(:,1:2:4)
ans =
   1   3
   7   9
```

A(2,:) – wybieramy drugi wiersz (a kolumny wszystkie):

```
>> A(2,:)
ans =
    7    8    9   10
```

A(: , :) to w tym przypadku to samo co A czyli cała macierz:

```
>> A(:,:)
ans =
    1    2    3    4
    7    8    9   10
```

A(:) – wymusza natomiast zamianę macierzy na wektor (kolumnami)

```
>> A(:)
ans =
    1
    7
    2
    8
    3
    9
    4
   10
```

3.1.5 Generowanie macierzy

Do generowania pewnych macierzy można stosować funkcje:

zeros(w,k): macierz wypełniona zerami np.: A = zeros(2,3)

```
ans =
     0     0     0
     0     0     0
```

ones(w,k): macierz wypełniona jedynkami np.: A = ones(2,4)

```
ans =
     1     1     1     1
     1     1     1     1
```

rand(w,k): macierz liczb pseudolosowych o rozkładzie równomiernym np.: A = rand(2,5)

```
ans =
   0.9501   0.6068   0.8913   0.4565   0.8214
   0.2311   0.4860   0.7621   0.0185   0.4447
```

eye(N): macierz jednostkowa (kwadratowa N x N z jedynkami na przekątnej głównej i zerami

np.: A = eye(3)

```
ans =
     1     0     0
     0     1     0
     0     0     1
```

Wiele innych macierzy można generować programami z użyciem instrukcji FOR i innych.

3.1.6 Wczytywanie macierzy z pliku

Załóżmy, że mamy plik tekstowy o nazwie DANE1.TXT a w nim są dwie linie i w każdej po 4 liczby oddzielane odstępami:

2 4 6 8
3 6 9 12

Aby wczytać te liczby do macierzy można napisać następujące instrukcje:

```
[plik1 info] = fopen('DANE1.TXT');
A = fscanf(plik1, '%f %f %f %f', [4, 2])
close(plik1)
```

Ale **UWAGA: dane czytane są z pliku wierszami ale umieszczane w macierzy kolumnami,** dlatego po wczytaniu uzyskamy macierz:

```
A =
    2    3
    4    6
    6    9
    8   12
```

Aby uzyskać to samo co w pliku trzeba macierz transponować (jak niżej)

3.1.7 Podstawowe operacje na macierzach

Transponowanie

- to operacja polegająca na zamianie wierszy macierzy na kolumny.

Operatorem transponowania jest w Matlabie apostrof [‘].

Przykładowo jeśli:

```
A =
    2    3
    4    6
    6    9
    8   12
```

to macierz transponowana:

```
A' =
    2    4    6    8
    3    6    9   12
```

Suma i różnica

Dla sumowania oraz odejmowania macierze muszą mieć jednakowe wymiary, sumowane są elementy o tych numerach.

```
A=[4, 2, 3; 3, 6, 1 ],  B=[5, 3, 8; 4, 1, 2]
A =
    4    2    3
    3    6    1
B =
    5    3    8
    4    1    2
A+B
ans =
    9    5   11
    7    7    3
```

Mnożenie macierzy lub ich elementów

Rozróżniane jest mnożenie macierzowe (*) oraz tablicowe (.*)

W mnożeniu macierzowym każdy element macierzy wynikowej powstaje przez pomnożenie odpowiedniego **wiersza** pierwszej macierzy przez **kolumnę** drugiej i zsumowaniu iloczynów par wyrazów. Wynika z tego warunek aby liczba elementów wiersza macierzy pierwszej była równa liczbie elementów w kolumnie macierzy drugiej. Dletego dla naszych macierzy A i B zostanie zasygnalizowany błąd:

```
>> A*B
    ??? Error using ==> *
    Inner matrix dimensions must agree.
```

Natomiast wykonalne będzie to działanie gdy macierz B transponujemy:

```
>> A*B'
ans =
   50   24
   41   20
```

Bez transponowania można wykonać tak zwane mnożenie tablicowe (operator: kropka i gwiazdka). W tym mnożeniu element macierzy wynikowej C(w,k) jest po prostu iloczynem pary elementów A(w,k)*B(w,k). Na przykład:

```
>> A .* B
ans =
   20    6   24
   12    6    2
```

3.1.8 Układ równań liniowych. Odwracanie oraz dzielenie macierzy

Załóżmy że układ równań liniowych doprowadziliśmy do postaci macierzowej zapisanej (w opisie a nie w Matlabie) jako: **A*X=B**

gdzie: **A**=macierz współczynników przy niewiadomych,
X=wektor niewiadomych,
B= wektor wyrazów wolnych

Wtedy rozwiązanie czyli wektor niewiadomych **X** wyznaczamy przez lewostronne pomnożenie obu stron równania przez macierz odwrotną do A zapisywaną w Matlabie jako **inv(A)**:

inv(A)*A*X= inv(A)*B

a ponieważ iloczyn macierzy danej i odwrotnej jest macierzą jednostkową którą można pominąć więc rozwiązanie dowolnego ukladu równań liniowych otrzymamy przy pomocy jednego wzoru:

X=inv(A)*B

Jednakże Matlab nie zaleca stosowania funkcji inv(..) a zamiast niej poleca **dzielenie lewostronne macierzy** (operator „\” w odróżnieniu od dzielenia prawostronnego „/”) jako mniej pracochłonne dla komputera i mogące w większości przypadków zastąpić odwracanie macierzy. W szczególności dla naszego układu równań liniowych stosując lewostronne dzielenie mamy (w opisie): A\A*X=A\B co po uproszczeniu trzeba zapisać w Matlabie jako

X=A\B

Matlab stosuje wówczas wydajniejszą metodę eliminacji Gauss’a zamiast pracochłonnego odwracania macierzy, co skraca czas obliczeń 2 do 3 razy i poprawia dokładność.

3.2 Ćwiczenia

Rozwiąż w Matlabie układ równań liniowych mając daną macierz współczynników M oraz wektor wyrazów wolnych C:

$$M := \begin{bmatrix} -2 & 0.5 & 4.2 & 8 \\ 0 & 4 & 8 & 2 \\ -5 & 7 & 3 & 1 \\ 10 & 12 & -6 & 4 \end{bmatrix} \qquad C := \begin{bmatrix} 73.5 \\ 15.2 \\ -33 \\ 5 \end{bmatrix}$$

rozwiązanie: $x = \begin{bmatrix} 3.258 \\ -4.499 \\ 1.817 \\ 9.329 \end{bmatrix}$

Dodatkowo wpisz odpowiednią komendę aby wyswietlić:
1) macierz transponowaną wzgledem M oraz C
2) wyznacznik macierzy M używając funkcji **det(macierz kwadratowa)**

3) pierwszą a potem drugią kolumnę macierzy M (używając wyrażenia z dwukropkiem)
4) elementy macierzy M z indeksami 1,1 oraz 2,2
5) liczbę elementów wektora C przy pomocy funkcji **length(wektor)**
6) rozmiary macierzy M oraz C przy pomocy funkcji **size(macierz)**
Jak numerowane są elementy wektorów i macierzy (od 0 czy od 1)?

4. Najprostsze programy (skrypty). Wprowadzanie danych i wyprowadzanie wyników. Pętla WHILE ... END

4.1 Najprostsze programy - objaśnienia

Programy dla MATLABa można pisać przy pomocy najprostszych edytorów tekstu jak NOTATNIK (w Ms Windows). Poszczególne instrukcje można pisać w oddzielnych liniach a gdy są w tej samej linii to oddzielać przecinkami, natomiast trzeba instrukcje kończyć średnikami - jeśli chcemy zablokować wyświetlanie wyniku każdej instrukcji na ekranie.

Po utworzeniu trzeba zapisać program do pliku tekstowego o rozszerzeniu nazwy: ".m" potocznie nazywanego: "m-plikiem".

Istnieją **dwa rodzaje m-plików**:

- **skrypty** czyli procedury nie wymagające parametrów, wywoływane przez wpisanie nazwy pliku (bez rozszerzenia ".m") w oknie komend (działają one na zmiennych tworzonych w tzw. przestrzeni roboczej Matlab'a)
- **funkcje** zwracające wartości (w postaci skalara lub wektora) i zazwyczaj wymagające podania parametrów czyli argumentów funkcji; wywołania ich są najczęściej używane w wyrażeniach stanowiących fragmenty instrukcji innych m-plików (zmienne używane w funkcji są lokalne tzn. niedostepne poza ciałem funkcji).

Dowolne objaśnienia czyli **komentarze** można umieszczać w m-plikach rozpoczynając od znaku procentu [%] .
Aby uruchomić napisany program w Matlabie trzeba:

1. ustawić jako **bieżący folder** - Current Directory (u góry) - ten dysk i folder w którym m-plik został zapisany, korzystając z przycisku [...],
2. wpisać nazwę m-pliku bez rozszerzenia „.m" ale uwaga:
   Matlab rozróżnia duże i małe litery

Objaśnienie instrukcji i funkcji używanych w pierwszym programie:

Składnia instrukcji lub funkcji	Objaśnienie
zmienna = **input(**'*żądanie danych*'**)**	Zostaje wyświetlony tekst *żądania danych* a następnie wpisana z klawiatury liczba zostaje podstawiona do *zmiennej*. Zamiast liczby można wpisać wyrażenie Matlaba
zmienna = **input(**'*żądanie danych*','**s**'**)**	wyświetla *żądanie danych,* oczekuje na wpisanie przez użytkownika *łańcucha znakowego* i przypisuje go *zmiennej*
disp('tekst'**)** lub **disp(**zmienna**)**	Wyświetla *tekst* lub wartość *zmiennej*
while *wyrażenie* . . . . . *instrukcje* **end**	Powtarza *instrukcje* tak długo jak *wyrażenie* ma wartość logiczną *true* (czyli *prawda*)

4.2 Pisanie pierwszego programu

Napisz w "Notatniku" program (skrypt) obliczania trzeciej potęgi podanej liczby i zapisz na dyskietkę do pliku "pr1.m" (Uwaga: aby nie dopisało się samoczynnie rozszerzenie ".txt" trzeba wybrać u dołu okna "Zapisz jako..." typ **"wszystkie pliki"**):

```
a = input('dana liczba=')
x=a^3
disp('szescian tej liczby='),disp(x)
```

Uruchom ten program w Matlabie.
Aby pozbyć się wyświetlania "echa" działania każdej instrukcji trzeba instrukcje kończyć średnikiem [;].
Linie komentarzy umieszczone na początku pliku (przed instrukcjami) wyświetlą się jako jego opis gdy wpiszemy: "help nazwa_pliku", a dodatkowo program po uruchomieniu też powinien się przedstawić.
Tak więc ulepszona wersja programu może wyglądać tak:

```
% Program oblicza sześcian podanej liczby
disp('Obliczanie sześcianu danej liczby:');
a = input;('dana liczba=')
x=a^3;
disp('szescian tej liczby=');disp(x);
```

Wpisz: **help pr1** a następnie jeszcze raz uruchom program.

Jeśli chcemy aby program działał dla wielu kolejno podawanych liczb – trzeba zastosować jeden z kilku możliwych typów pętli programowych. W tym przypadku najodpowiedniejsza będzie pętla WHILE ... END powtarzająca operacje aż do momentu gdy zechcemy ją zatrzymać.

Uzupełnij program pętlą WHILE ... END tak aby zatrzymał się po wprowadzeniu zera jako danej dla x.

5. Pętla FOR ... END. Zapisywanie wyników do pliku

5.1 Objaśnienia

5.1.1 Pętla for ... end

Pętla ta ma postać

for *zmienna = macierz*
. . . . *instrukcje*
end

i wykonuje się tyle razy ile jest kolumn w *macierzy* a wartościami *zmiennej* kontrolnej są właśnie całe kolumny tej macierzy czyli wektory.
W szczególności najczęściej *macierz* jest ciągiem i jest to wtedy bardziej podobne do pętli for w innych językach (np. w języku BASIC) a mianowicie:

for *zmienna = wart_p : krok : wart_k*
. . . . *instrukcje*
end

Pozwala więc ona powtarzać wykonywanie bloku *instrukcji* określoną liczbę razy przy czym dodatkowo *zmienna* kontrolna w tej pętli przyjmuje kolejno wartości od *wart_p* do *wart_k* z przyrostem (lub ubytkiem) *krok*. Jeśli krok=1 to można go pominąć w zapisie.

5.1.1.1 Przykład 1:

Aby otrzymać kwadraty liczb parzystych od 2 do 10 i podstawić je do kolejnych elementów wektora p:

```
for i = 1:5
        p(i) = (2*i)^2
end
```

5.1.1.2 Przykład 2:

Aby wyliczyć i wstawić do wektora y ciąg wartości funkcji sinus przyjmujemy konkretny zakres kąta (w radianach) na przykład od zera do pi/2 i przyrost na przykład 0.2:

```
k=0;
for x = 0 : 0.2 : pi/2
        k=k+1;
        y(k)=sin(x);
end
```

5.1.1.3 Przykład 3:

Program:

```
for w=1:3
  for k=1:4
        M(w,k)=w+k;
  end
end
M
```

Wygeneruje macierz o 3 wierszach i 4 kolumnach:

M =

2	3	4	5
3	4	5	6
4	5	6	7

5.1.1.4 Przykład 4. Mnożenie macierzowe macierzy

Używane w różnych modelach matematycznych układy równań o regularnej budowie mogą być wygodnie i krótko zapisywane w postaci macierzowej. Przy ich formułowaniu i rozwiązywaniu stosuje się operacje macierzowe a jedną z nich jest mnożenie.

W działaniu tym wyznaczane będą sumy iloczynów par składających się z elementu wierszy macierzy pierwszej i elementu kolumny macierzy drugiej.

Wynika stąd warunek wykonalności mnożenia macierzy:

jeśli mamy macierz A(Lwa,Lka) o Lwa wierszach i Lka kolumnach i analogicznie mamy macierz B(Lwb,Lkb) to liczba elementów w wierszu macierzy A musi być równa liczbie elementów kolumny macierzy B czyli Lka=Lwb. Elementy będą powstawać niejako na przecięciach wierszy A z kolumnami B skąd wymiary macierzy wynikowej C(Lwa,Lkb)

Konkretnie dla macierzy A(4,3) B(3,2) mnożenie jest możliwe bo Lka=Lwb=3 a macierzą wynikową będzie C(4,2)

Oto przykład obliczania macierzy C która ma powstać jako iloczyn macierzy A i B:

$$A=\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \\ a41 & a42 & a43 \end{bmatrix} \qquad B=\begin{bmatrix} b11 & b12 \\ b21 & b22 \\ b31 & b32 \end{bmatrix}$$

$$C = A \cdot B = \begin{bmatrix} a11 \cdot b11 + a12 \cdot b21 + a13 \cdot b31 & a11 \cdot b12 + a12 \cdot b22 + a13 \cdot b32 \\ a21 \cdot b11 + a22 \cdot b21 + a23 \cdot b31 & a21 \cdot b12 + a22 \cdot b22 + a23 \cdot b32 \\ a31 \cdot b11 + a32 \cdot b21 + a33 \cdot b31 & a31 \cdot b12 + a32 \cdot b22 + a33 \cdot b32 \\ a41 \cdot b11 + a42 \cdot b21 + a43 \cdot b31 & a41 \cdot b12 + a42 \cdot b22 + a43 \cdot b32 \end{bmatrix} = \begin{bmatrix} c11 & c12 \\ c21 & c22 \\ c31 & c32 \\ c41 & c42 \end{bmatrix}$$

Popatrzmy jak obliczono dowolny element macierzy C na przykład c32 a dokładniej c(3,2) bo zapiszemy tym razem wskaźniki w nawiasach i oddzielone przecinkiem:

$c(3,2) = \sum_{i=1}^{Lka} a(3,i) \cdot b(i,2)$ .Uogólniając mamy wzór na obliczanie elementu macierzy wynikowej:

$$c(w,k) = \sum_{i=1}^{Lka} a(w,i) \cdot b(i,k)$$

5.1.2 Zapisywanie wyników do pliku

Funkcje realizujące odczytywanie i zapisywanie informacji z i do plików dyskowych są zapożyczone z języka C. Poniżej objaśniono kilka najważniejszych funkcji na przykładzie:

```
% Program zapisuje do pliku wartości kąta x oraz jego funkcji sin(x), cos(x)
[id, kom] = fopen('a:\WYNIKI1.TXT','wt'),  % Tworzy plik wyników
fprintf(id,'%s\n', ' kąt x [stopnie]    sin(x)    cos(x)' );  % Nagłówek tabelki
for xs = 0 : 5 : 90
        x = xs*pi/180;    % kąt xs zamieniony na radiany
        y1=sin(x);  y2=cos(x);
        fprintf(id,'       %3d', xs);
        fprintf(id,' %15.4f   %12.4f\n', y1,y2);
end
fclose(id);
```

Wyniki w pliku 'Wynik.txt' mają następującą postać:

```
 kąt x [stopnie]     sin(x)     cos(x)
           0         0.0000      1.0000
           5         0.0872      0.9962
          10         0.1736      0.9848
          15         0.2588      0.9659
          20         0.3420      0.9397
         . . .       . . . .     . . . .
```

Objaśnienia:

Funkcja **fopen** tworzy i otwiera plik o nazwie ***'a:\Wyniki1.txt'*** bo ma podaną taką nazwę pliku jako pierwszy parametr. Drugi parametr **'wt'** określa typ dostępu:

'w' = zapis (ang. write), **'t'** – plik typu tekstowego

Funkcja ta zwraca dwie wartości, które w tym przypadku zostaną podstawione do zmiennych **[id, kom]** gdzie: **id** = identyfikator pliku, **kom** = komunikat o ewentualnej przyczynie niemożliwości otwarcia pliku.

Funkcja fprintf wyprowadza informacje do pliku tekstowego o identyfikatorze podanym jako pierwszy parametr tej funkcji (w naszym przypadku: **id**). Drugim parametrem jest łańcuch tekstowy określający format wyprowadzanej informacji. Spacje również są tu istotne.

%3d – określa **3** miejsca dla liczby **całkowitej** (o czym świadczy litera **d**)

%12.4f - to format dla liczb rzeczywistych, a w nim **12** miejsc zadeklarowano dla całej liczby (i poprzedzających ją spacji) a w tych 12-tu zarezerwowano **4** miejsca po kropce dla części ułamkowej
\n – oznacza rozkaz zmiany linii na wydruku wyprowadzanym do pliku (nie pomyl: \ a nie /)
fclose(id) – zamyka plik o identyfikatorze **id** po wyprowadzeniu wszystkich informacji.

5.2 Ćwiczenia

a) Napisz program z pętlą FOR...END, który wyświetla na ekranie pierwiastki z kolejnych liczb nieparzystych od 1 do 9

b) Napisz program z pętlą FOR...END, który dla ciągu wartości φ od 3,6 do 13 co 0,4 oblicza wartości wyrażenia: β **= (φ - 0,5)/(1,1+sin φ)**

Zadanie domowe:
Napisz programy z pętlami FOR...END wykonujące (a) mnożenie macierzowe, (b) mnożenie tablicowe dwu macierzy i sprawdź dla konkretnych macierzy A i B zgodność wyników działań twoich programów z działaniami Matlaba: **A*B** oraz **A .* B**

6. Instrukcja IF

Instrukcja ta ma najczęściej postać:

if *warunek*
. . . .*instrukcje1*
else
. . . . *instrukcje2*
end

Instrukcja IF (zupełnie podobnie jak w BASIC-u i innych językach) pozwala zależnie od spełnienia podanego *warunku* wykonać blok *instrukcje1* lub blok *instrukcje2*.
Warunek używany w instrukcji IF to dowolne wyrażenie logiczne.

Przykład:

Dla danej wartości x obliczyć y dane wzorem:

$$y = \begin{cases} 1 - x^2 & dla \quad x < 1 \\ x & dla \quad -1 \le x \le 1 \\ 1 + x^2 & dla \quad x > 1 \end{cases}$$

```
% program w Matlabie:
x = input('x=');
if x<1
     y = 1-x²
else
     if x>1
          y = 1+x²
     else
          y = x
     end
end
```

Składniki wyrażeń logicznych opisano poniżej.

6.1 Relacje i wyrażenia logiczne – objaśnienia

Prostymi wyrażeniami logicznymi są relacje. Relacja to dwa wyrażenia arytmetyczne połączone operatorem relacji. Są następujące operatory relacji:

Operator	Opis
<	"mniejsze"
<=	"mniejsze lub równe"
>	"większe"
>=	"większe lub równe"
==	równe
~=	nierówne

Operatory logiczne to:

Operator	Znaczenie
&	**i**
|	**lub**
~	**nie**

Zamiast nich można stosować funkcje:

Funkcja	Znaczenie
and(A,B)	A i B
or(A,B)	A lub B
not(A)	nie A

6.2 Przykład programu (skryptu) z instrukcją IF

Rozwiązywanie równania kwadratowego jako przykład m-pliku skryptowego z zastosowaniem instrukcji IF:

```
% po znaku procentu można umieszczać dowolne komentarze
% Program rozwiazywania równania kwadratowego
a=input('a=');   b=input('b=');    c=input('c=');
delta = b*b-4*a*c;
if delta<0
        disp('Brak pierwiastkow rzeczywistych');
else
        x1=(-b-sqrt(delta))/(2*a); x2=(-b+sqrt(delta))/(2*a);
        disp('x1='); disp(x1); disp('x2='); disp(x2);
end
```

Taki skrypt jest samodzielnym programem a więc nie tylko oblicza ale także żąda danych z klawiatury i wyświetla na ekranie wyniki.

6.3 Ćwiczenia

Napisz program, który dla dowolnej liczby x wprowadzonej z klawiatury - sprawdza przy pomocy funkcji mod(x,2) wyznaczającej resztę z dzielenia przez 2 – czy jest to liczba (a) parzysta, (b) nieparzysta, (c) niecałkowita. Po sprawdzeniu ma wyświetlić odpowiedni komunikat.

7. Pisanie własnych funkcji

7.1 Objaśnienia

Zaletą Matlaba jest ogromne bogactwo gotowych funkcji z wszelakich niemal dziedzin zastosowań komputerów, jednak w razie potrzeby użytkownik może pisać własne funkcje i używać je identycznie jak funkcje Matlab'a. Własne funkcje pisze się gdy:

1. chcemy aby program składał się z bloków funkcjonalnych o ściśle określonym działaniu – co czyni program lepiej zrozumiałym i jest szczególnie zalecane w przypadku długich programów
2. funkcja będzie przynajmniej kilkukrotnie wykorzystywana w danym programie

Definicja funkcji w Matlabie musi rozpoczynać się od linii o następującej strukturze:

function *wektor_zmiennych_wynikowych* = ***nazwa_funkcji****(parametry_wejściowe)*

Na przykład: **function** [y1, y2, y3] = fun1(x1, x2, x3, x4)

7.1.1 Przykład 1. Funkcja "silnia"

```
% Funkcja silnia wyznacza watosc n!
function [wynik]= silnia(n)
wynik=1;
for i=1:n
     wynik=wynik*i;
end
```

Po zapisaniu tej funkcji do pliku, który musi w tym przypadku mieć nazwę **silnia.m** można wywoływać funkcję z konkretnymi wartościami argumentu **n**:

```
>> silnia(5)
ans =
      120

>> silnia(9)
ans =
      362880
```

7.1.2 Przykład 2. Funkcja "pierwiastek z sumy kwadratów"

W przypadku gdy w wyrażeniach powtarzają się pewne działania lecz dotyczą one różnych danych, może być opłacalne napisanie własnej funkcji. Na przykład jeśli dla danego x mamy obliczyć wyrażenie:

$p = \dfrac{1-\sqrt{9+x^2}}{\sqrt{4x^4+16}}$ i być może następne wyrażenia w których występuje pierwiastek z sumy kwadratów to można zdefiniować funkcję:

```
function c = pwsk(a,b)
% dla dwu liczb danych jako argumenty oblicza pierwiastek z sumy ich kwadratów
c = sqrt(a^2+b^2)
```

Trzeba zapisać ją do pliku o nazwie **pwsk.m** a następnie można wykorzystać ją na przykład w takim programie:

```
x=1
while x ~ = 0
  x = input('x=');
  p = (1 – pwsk(3,x))/(pwsk(2*x^2, 4)
  disp('p='); disp(p)
end
```

7.1.3 Przykład 3. Funkcja z instrukcją IF (rozwiązywanie równania kwadratowego)

Podany wcześniej przykład skryptu (procedury) do rozwiązywania równania kwadratowego pokazano poniżej przerobiony na funkcję:

```
function [x1, x2] = prkw(a, b, c)
% ta funkcja oblicza pierwiastki x1, x2
% rownania: a*x^2 + b*x + c = 0
delta = b*b-4*a*c;
if delta<0
% dla delta<0 podstawimy NaN = "nieokreslone"
        x1=NaN;    x2=NaN
else
        x1=(-b-sqrt(delta))/(2*a);
        x2=(-b+sqrt(delta))/(2*a);
end
```

Funkcję należy zapisać do pliku o takiej samej nazwie jak nazwa funkcji, a więc: **prkw.m**

Rola tej funkcji jest taka sama jak funkcji standardowych (na przykład sinus) to znaczy nie zawiera ona instrukcji wejścia/wyjścia bo wprowadzenie do niej danych nastepuje przez parametry (a,b,c) a wynik zostaje jak to się mówi "zwrócony" przy pomocy nazwy funkcji (prkw) pełniącej rolę parametru wyjściowego. Ponieważ w Matlabie macierze pełnią rolę zmiennych więc i tutaj wynik może być macierzą a w szczególności wektorem lub skalarem.

Funkcja może być wywołana samodzielnie (z konkretnymi parametrami) ale najczęściej opłaca się ją napisać gdy będzie używana jako cegiełka większego programu.

Przykłady bezpośredniego użycia zdefiniowanej przed chwilą funkcji o nazwie **prkw**:

>> [x1, x2]=prkw(1,1,1)
x1 = NaN
x2 = NaN
W tym przypadku brak było pierwiastków rzeczywistych.
>> [x1, x2]=prkw(-1,1,1)
x1 = 1.6180
x2 = -0.6180

8. Wykresy dwuwymiarowe: liniowy i słupkowy

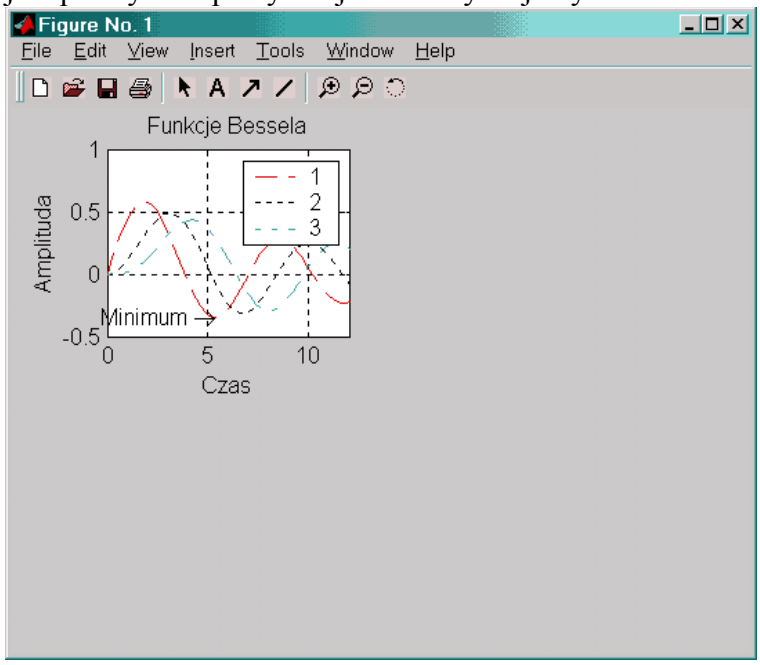
Grafika (a więc wykresy 2D i 3D oraz obrazy) są w Matlabie wyświetlane w osobnych oknach graficznych określanych angielskim terminem "Figure". Funkcje graficzne wyświetlają wyniki swych działań w **aktywnym** (ostatnio otwartym lub używanym) oknie a jeśli żadne okno "Figure" nie było otwarte to automatycznie tworzą nowe.
Przy sporządzaniu wykresów dwuwymiarowych mogą wystąpić etapy opisane w tabeli poniżej:

Etap:	Przykład:	Objaśnienie przykładu:

Przygotowanie danych	x = 0:0.2:12;
	y1 = Bessel(1,x);
	y2 = Bessel(2,x); y3 = Bessel(3,x);
		ciąg wartości dla osi x;
		wybranie z macierzy B trzech wierszy dla wykresów

Otwarcie lub wybranie okna graficznego i ewentualnie pozycji w tym oknie	figure(1) subplot(2,2,1)	okno graficzne o numerze 1 podzielone na 2 wiersze i 2 kolumny i wybrana część nr 1
wywołanie funkcji realizującej wykres	h = plot(x,y1,x,y2,x,y3);	wyświetli 3 wykresy liniowe i przypisze identyfikator h
określenie parametrów linii wykresu i znaczników punktów	set(h,'LineWidth',2,{'LineStyle'}, {'--';':';'-.'}) set(h,{'Color'},{'r';'g';'b'})	grubość linii =2, linie ciągła, przerywana i "osiowa", kolory: czerwony, zielony, niebieski
określenie parametrów osi i pokazanie siatki	axis([0 12 –0.5 1]) grid on	zdefiniowane osie współrz. włączona siatka
zdefiniowanie tekstów opisów wykresu, osi i legendy	xlabel('Czas'); ylabel('Amplituda') legend(h,'1','2','3') title('Funkcje Bessela') [y,ix] = min(y1); text(x(ix),y,'Minimum \rightarrow',... 'HorizontalAlignment','right')	opisy osi x i y, teksty legendy, tytuł wykresu dodatkowy tekst ze strzałką wyrównany w prawo
wydrukowanie lub eksport wykresu do pliku	print –dwinc -r200 wykres1	wysyła do pliku *wykres1* kolorowy (*-dwinc*) obraz o rozdzielczości 200 dpi (*-r200*)

W wyniku instrukcji zapisanych w powyższej tabeli uzyskujemy na ekranie następujący obraz:



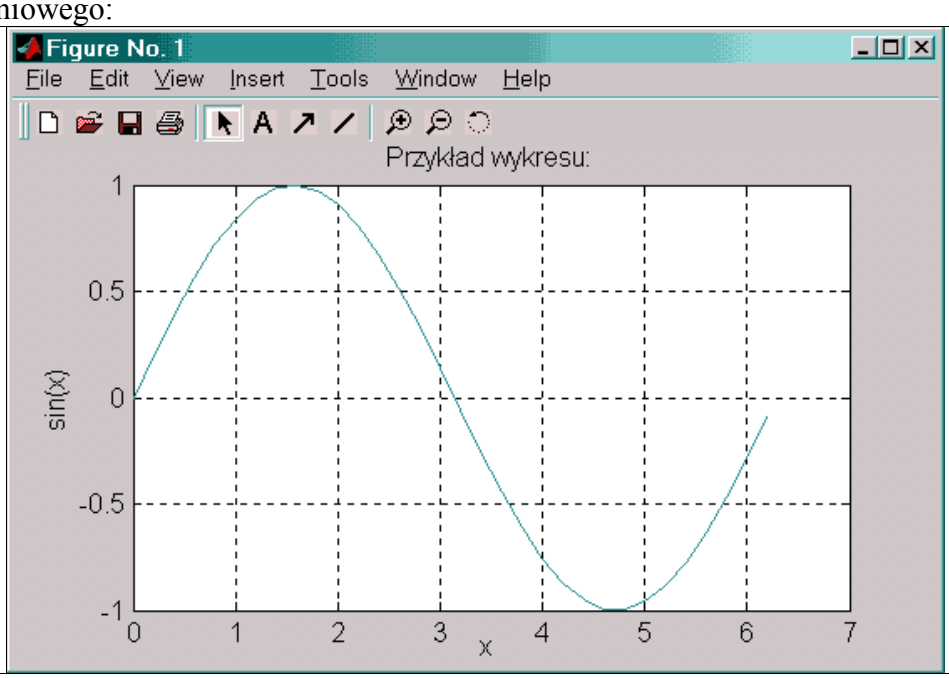
a dodatkowo powstanie plik *wynik1.ps* z zapisem obrazu w języku postscript.

8.1 Wybrane narzędzia dla wykresów dwuwymiarowych

Instrukcja lub funkcja Matlab'a	Opis
nr = figure	Otwiera nowe okno graficzne o numerze **nr**. Może być pominięta jeśli wystarcza nam tylko jedno okno graficzne.
figure(nr)	Uaktywnia okno o numerze **nr** jeśli takie istnieje a jeśli nie istnieje to tworzy nowe okno i nadaje mu numer **nr**.
plot(x,y)	Dla danych wektorów x,y rysuje wykres liniowy
plot(y)	Wykres liniowy wartości y a na osi x są ich numery
plot(x1,y1, x2,y2, ...)	umożliwia rysowanie kilku wykresów w jednym oknie
plot(x1,y1,s1, x2,y2,s2, ...)	umożliwia rysowanie kilku wykresów przy czym: **s1, s2** to opisane dalej łańcuchy znaków określające typ linii, kolor linii oraz rodzaj znacznika punktów
bar(x,y,s)	Wykres słupkowy y(x), s= stosunek szerokości słupka do odstepu między słupkami
bar(y)	Wykres słupkowy wartości y a na osi x są ich numery
grid on	Włącza siatkę wykresu
title('Tytuł wykresu')	Definiuje tytuł wykresu
xlabel('opis x'); ylabel('opis y')	Definiują opisy osi x i y

Przykład wykresu liniowego:

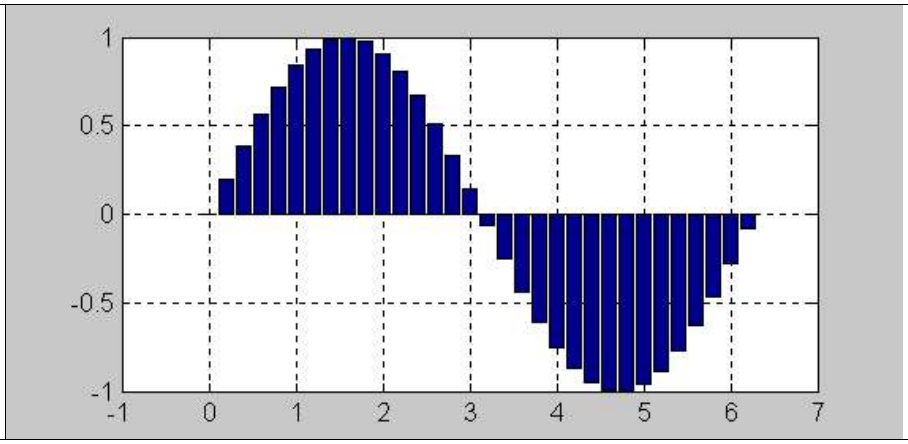
```
x=0 : 0.2 : 2*pi;
plot(x,sin(x));
grid on
title('Przykład
wykresu:');
xlabel('x');
ylabel('sin(x)');
```



Wciśnięcie przycisku ze strzałką umożliwia wybieranie (myszką) i modyfikowanie m.in. opisów.

```
% Przykład
% wykresu
% słupkowego:

x=0 : 0.2 : 2*pi;
bar(x,sin(x));
grid on
```



Opis znaków definiujących parametry wykresu liniowego (typu PLOT)

Znak	Rodzaj linii
-	
--	
:	
-.	
	Kolor linii
y	yellow – żółty
m	magenta – karmazynowy
c	cyan – turkusowy
r	red – czerwony
g	green – zielony
b	blue – niebieski
w	white – biały
k	black – czarny

Znak	Znacznik punktu
+	+
*****	*
.	kropka
o	o
x	x
s	kwadrat
d	romb
p	gwiazdka pięcioramienna
h	gwiazdka sześcioramienna
v	trójkąt z wierzchołkiem w dół
^	trójkąt z wierzchołkiem w górę
<	trójkąt z wierzchołkiem w lewo
>	trójkąt z wierzchołkiem w prawo

9. Wykresy trójwymiarowe

Rysowanie wykresów trójwymiarowych w najprostszym przypadku przbiega dwuetapowo:

1) przygotowanie siatki par współrzędnych (x,y) dla funkcji z=f(x,y) przy pomocy funkcji meshgrid
2) użycie jednej z wielu funkcji dla wykresów trójwymiarowych

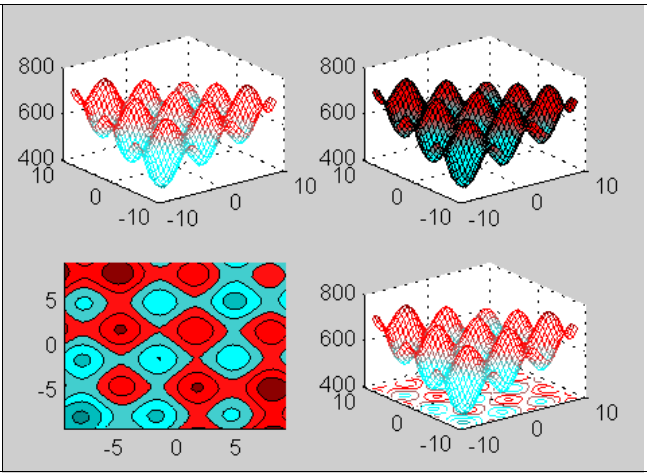
Funkcji meshgrid podajemy jako argumenty ciągi (wektory) wartości x oraz y a w wyniku uzyskujemy dwie macierze zawierające łącznie wszystkie pary współrzędnych dla któych mają być wyznaczane wartości funkcji zmiennych x,y.
Na przykład:

```
>> [x y] = meshgrid(0:0.1:0.3,  1:3)
x =
         0    0.1000    0.2000    0.3000
         0    0.1000    0.2000    0.3000
         0    0.1000    0.2000    0.3000
y =
     1     1     1     1
     2     2     2     2
     3     3     3     3
```

Tak więc funkcja wyliczana będzie dla (0, 1); (0.1, 1); (0.2, 1), ... i tak dalej.

Matlab posiada bardzo wiele funkcji dla wizualizacji linii i powierzchni trójwymiarowych. Niektóre z nich pokazano na przykładzie poniżej:

```
% najpierw siatka punktów (x,y) dla wykresu 3D:
[x,y]=meshgrid(-3*pi : 0.5 : 3*pi, -3*pi : 0.5 : 3*pi);
% następnie definiujemy funkcję z(x,y):
z = 600 – x .* y + 50 * sin(x) + 50 * sin(y);
% przy pomocy funkcji subplot
% wybieramy ćwiartki okna graficznego
% i wyświetlamy w nich wykresy 3D:
% 1) wykres siatkowy:
subplot(2,2,1); mesh(x,y,z);
% 2) wykres powierzchniowy:
subplot(2,2,2); surf(x,y,z);
% 3)wykres warstwicowy:
subplot(2,2,3); contourf(x,y,z);
% 4)wykres siatkowy z warstwicami:
subplot(2,2,4); meshc(x,y,z);
```



Ogólnie przy sporządzaniu wykresów trójwymiarowych mogą wystąpić etapy i funkcje przedstawione w tabeli poniżej – Nie podano ich opisu traktując te informacje jako zachętę do dalszego studiowania Matlab'a:

Etap:	**Przykład:**
Przygotowanie danych	Z = peaks(20);
Otwarcie lub wybranie okna graficznego i ewentualnie pozycji w tym oknie	figure(1) subplot(2,2,1)
Wywołanie funkcji wykresu	h = surf(Z);
Wybranie palety (mapy) kolorów i sposobu cieniowania	colormap hot shading interp set(h,'EdgeColor','k')
Zdefiniowanie źródła światła	light('Position',[-2,2,20]) lighting phong material([0.4,0.6,0.5,30]) set(h,'FaceColor',[0.7 0.7 0],... 'BackFaceLighting','lit')
Ustalenie punktu widzenia	view([30,25]) set(gca,'CameraViewAngleMode','Manual')
określenie parametrów osi	axis([5 15 5 15 –8 8]) set(gca'ZTickLabel','Negative||Positive')
	set(gca,'PlotBoxAspectRatio',[2.5 2.5 1])
zdefiniowanie tekstów opisów wykresu, osi i legendy	xlabel('X Axis') ylabel('Y Axis') zlabel('Function Value') title('Peaks')
wydrukowanie lub eksport wykresu do pliku	set(gcf,'PaperPositionMode','auto') print –dps2

10.Przeglądanie wybranych programów demonstracyjnych

Aby oglądać programy demonstracyjne wystarczy wpisać komendę:

demo

a następnie wybrać przykład z wykazu.