Wprowadzenie do Octave

1 Proponowana literatura:

- Zalewski A., Cegieła R. - "Matlab. Obliczenia numeryczne i ich zastosowania"
- Mrozek B., Mrozek Z. - "MATLAB 5.x SIMULINK 2.x, Poradnik użytkownika"
- Kamińska A., Pańczyk B. - "Matlab. Przykłady i zadania"
- Polecenie help w programie

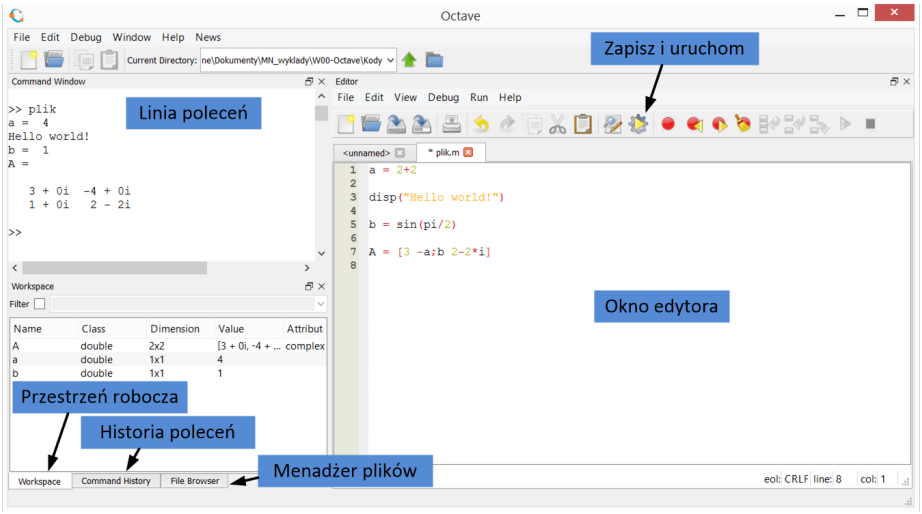
2 Czym jest Octave?

Octave jest wolnym i darmowym środowiskiem do obliczeń numerycznych oraz językiem do obsługi tego środowiska. Octave jest odpowiednikiem komercyjnego środowiska Matlab. Niektóre zaawansowane funkcje Matlaba nie istnieją w Octave'ie, a niektóre proste, podstawowe funkcje Octave'a nie występują w Matlabie. W zastosowaniach studenckich, gdzie nie jest tak istotna szybkość działania, budowanie interfejsów graficznych czy generowanie animacji 3D, bardziej odpowiedni jest Octave – zarówno język (więcej interesujących poleceń), jak i sam program (darmowy). Oba środowiska mozna zainstalować na systemach linuksowych, i na Windowsach, oraz na MAC OS.

3 Historia Octave

- lata 50 XX w. – Fortran - kompilator wysokiego poziomu do obliczeń naukowych i inżynierskich (John Backus)
- lata 70 XX w. – Linspack, Eispack – biblioteki do obliczeń macierzowych dla Fortranu
- 1985 - pierwsza wersja komercyjnego programu Matlab (C)
- 1993/94 – pierwsza wersja Octave (John W. Eaton) "darmowy Matlab"
- 2015 – Octave 4.0.0

4 Okno programu



5 Linia poleceń w Octave

Jednym ze sposobów pracy z Octave jest wykorzystanie jego lini poleceń do wykonywania obliczeń.

- znak >> oznacza gotowość do wprowadzenia polecenia.
- wykonajmy proste dodawanie.

  ```
  >> 2+4
  ```

  Polecenia pisane w lini poleceń zatwierdzamy enterem.
  Wynikiem działania jest :

  ```
  ans =  6
  ```

- wartość wyrażenia można przypisać do zmiennej (domyślna zmienna to ans)

  ```
  >> a = 2*3
  a =  6
  >> b = sqrt(2) + sin(a)
  b =  1.1348
  ```

- Zmienna ans jest traktowana jak normalna zmienna tzn. jej wartość jest w pamięci Octave'a, co umożliwia odwoływanie się do niej.
  UWAGA ! Wprowadzanie kolejnych wartości wyrażeń bez przypisanych do nich zmiennych nadpisuje zmienną ans.

- Zmienne od razu przypisujemy do wyrażenia, nie deklaruje się ich na początku programu oraz nie podaje się ich typu.
- średnik na końcu powoduje nie wyświetlanie wyniku polecenia.

```
>> b = 3 ;
```

- Jeśli polecenie nie mieści się w lini możemy je przenieść do następnej korzystając z "..."

```
>> c = 1 + 1/2 + 1/3 + ...
1/4 + 1/5
>> c = 2.2833
```

- Octave wyświetla przybliżone wartości wyników obliczeń i wartości zmiennych. Domyślnie wyświetlane jest 5 cyfr znaczących wyniku. Jeżeli chcemy zwiększyć dokładność do 15 cyfr, co z grubsza odpowiada domyślnej dokładności obliczeń, powinniśmy wydać polecenie:

```
>> format long
>> c
c =  2.28333333333333
```

Aby powrócić do domyślnego wyświetlania zmiennych (z 4 miejscami po przecinku) należy wydać polecenie

```
>> format short
```

Inne możliwe modyfikatory polecenia format są dostępne po wpisaniu w linii poleceń:

```
>> help format
```

- Wszelkie znaki wprowadzone po symbolu % są traktowane jako komentarz.
- Tabulacja – wyświetlenie "podpowiedzi"

```
>> cart2 %(Tab)
cart2pol   cart2sph
```

- Strzałki w górę/w dół - przywoływanie poprzednich poleceń.

6 Macierze jako podstawowy typ danych w Octave

Elementy macierzy indeksujemy od 1 !

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1q} \\ a_{21} & a_{22} & \cdots & a_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \cdots & a_{pq} \end{bmatrix}$$

6.1 Tworzenie macierzy

Macierze można zdefiniować poprzez:

- wymienienie elementów
- wygenerowanie elementów
- sklejenie innych z macierzy

6.1.1 Wymienienie elementów

A = [1 2 3; 7 8 9]

A = [1,2,3; 7,8,9]

A = [1 2 3
7,8,9]

6.1.2 Generowanie elementów

Jednym ze specyficznych operatorów w programie Matlab/Octave jest dwukropek. Umożliwia on m.in. tworzenie wektorów o równomiernie rozłożonych elementach. Polecenie:

$p : q : r$

utworzy wektor postaci:

$[p, p + q, p + 2q, \ldots, r]$

Jeśli parametr q zostanie pominięty domyślnie zostanie użyty krok $q = 1$.

Przykłady zastosowania:

- 1:7
  ans = 1 2 3 4 5 6 7
- 1:3:10
  ans = 1 4 7 10

6.1.3 Sklejanie

Często spotykanym sposobem jest również tworzenie macierzy z innych macierzy.

Przykład:

- A = 1:4 ;
- B = 1:5 ;
- C = 0 ;
- D = [A, C ; B]
  D =1 2 3 4 0
  1 2 3 4 5

Tablica 1: Funkcje wspomagające generowanie macierzy

Funkcja	Opis
eye(n) eye(n,m)	macierz jednostkowa o wymiarach $n \times n$ lub $n \times m$
ones(n) ones(n,m)	macierz wypełniona jedynkami
rand(n) rand(n,m)	macierz wypełniona liczbami pseudolosowymi o rozkładzie jednostajnym na [0,1]
zeros(n) zeros(n,m)	macierz wypełniona zerami

Należy przy tym pamiętać, aby wymiary łączonych macierzy zgadzały się. W przeciwnym przypadku próba utworzenia macierzy zakończy się komunikatem błędu.

7 Odwoływanie się do elementów macierzy

Aby odwołać się do elementu macierzy X trzeba podać wiersz oraz kolumnę w następujący sposób:
X(wiersz,kolumna)

Kilka przykładów zastosowania:

```
>> V = [ [9:-1:7; 1:3], zeros(2,1);
1 1 1 1]
V =
9    8    7    0
1    2    3    0
1    1    1    1

>> V(1)
ans =  9

>> V(1,3)
ans =  7

>> V(3,1)
ans =  1

>> V(:,2)                    %znak : oznacza wszystkie elementy
ans =                         w tym przypadku z kolumny drugiej
8
2
```

```
1

>> V(2,:)
ans =
1    2    3    0

>> V(2, 1:3)              %pobierz z drugiego wiersza elementy od 1 do 3
ans =
1    2    3

>> V(2, [1 3 4])          % z drugiego wiersza wybierz element 1 , 3 oraz 4
ans =
1    3    0
```

7.1 Usuwanie części macierzy

Polecenie [] definiuje macierz pustą. Przypisanie takiej macierzy do wiersza lub kolumny spowoduje ich usunięcie, np:

- D(:,2:3) = []
  D =1 4 0
  1 4 5
- D(1,:) = []
  D =1 3 4 5 W ten sposób możemy usuwać *całe* wiersze, kolumny bądź ich grupy. Usunięcie pojedynczego elementu macierzy nie miałoby sensu i wygeneruje błąd.

8 String - jako typ danych

Stringi to tak naprawdę również macierze!
Tekst wpisywany jest pomiędzy znaki ” ”

```
>> s1 = "laczenie"
s1 = laczenie
>> s2 = 'kilku'
s2 = kilku
>> s3 = ["stringow"]
s3 = stringow
>> s4 = ["a tu " s1 ' ' s2 ' ' s3]
s4 = a tu laczenie kilku stringow
>> disp(s4)
a tu laczenie kilku stringow
>> s1(3)
ans = c
```

Tablica 2: Operatory arytmetyczne

Operator	macierzowy	tablicowy
dodawanie	+	+
odejmowanie	−	−
mnożenie	*	.*
dzielenie prawostronne	/	./
dzielenie lewostronne	\	.\
potęgowanie	^	.^
transpozycja	'	.'

9 Operacje na macierzach

9.1 Operatory arytmetyczne (macierzowe i tablicowe)

Matlab/Octave umożliwia wykonywanie na macierzach dwóch rodzajów operacji arytmetycznych. Operatory **macierzowe** są określone regułami algebry macierzowej. Natomiast operatory **tablicowe** traktują macierze jako tablice liczb i umożliwiają wykonywanie operacji na odpowiadających sobie elementach macierzy. Każdy operator macierzowy ma swój tablicowy odpowiednik poprzedzony kropką. Wyjątkiem są operatory dodawania i odejmowania, które z definicji operują na elementach macierzy.

Aby wykonać jakąkolwiek operację tablicową macierze *muszą* mieć takie same rozmiary bądź jedna z nich musi być skalarem. Zgodnie z definicją iloczyn (w sensie Cauchy'ego) dwóch macierzy może być określony tylko wtedy gdy liczba kolumn pierwszej macierzy jest równa liczbie wierszy drugiej macierzy. Mnożenie macierzowe nie jest przemienne!

Jak wiadomo dzielenie jest operacją odwrotną do mnożenia ($\frac{a}{b} = a \cdot b^{-1}$). Matlab umożliwia dwa rodzaje dzielenia. Dzielenie prawostronne dwóch macierzy zdefiniowane jest następująco:

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

Z kolei mnożenie lewostronne:

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

W analogiczny sposób zdefiniowane jest dzielenie tablicowe. Każdy element c_{ij} macierzy $C = A./B$ jest równy $c_{ij} = a_{ij} \cdot \frac{1}{b_{ij}}$. Z kolei dla tablicowego dzielenia prawostronnego, czyli gdy $C = A.\backslash B$ mamy: $c_{ij} = \frac{1}{a_{ij}} \cdot b_{ij}$.

9.2 Operatory porównań

W programie Matlab/Octave porównywać możemy całe macierze (tablice). Warunkiem są takie same rozmiary obu operandów (Wyjątkiem jest sytuacja, gdy jeden z nich jest skalarem. Wtedy skalar ten jest porównywany z każdym elementem drugiego operandu). Wynikiem działania operatorów porównań jest macierz (tablica) takiej samej wielkości co operandy wypełniona jedynkami gdy

Tablica 3: Operatory porównania

Operator	Relacja
==	równość
~=	różność
>	większe niż
<	mniejsze niż
>=	większe lub równe
<=	mniejsze lub równe

odpowiadające sobie elementy spełniają relację i zerami w przeciwnym przypadku.

I tak na przykład, jeśli A = [-2 3 0], B=[3 3 3], wyrażenie $A >= B$ ma wartość [0 1 0], natomiast $A \sim= B$ będzie równe [1 0 1].

9.3 Operatory logiczne

Najnowsze wersje programu udostępniają trzy rodzaje operatorów i funkcji logicznych. Są to operatory:

- elementowe"(element-wise) - wykonują operacje na elementach macierzy traktując liczbę 0 jako zero logiczne i każdą inną wartość jako jedynkę logiczną.
- bitowe (bit-wise) - operują na liczbach całkowitych lub zbudowanych z nich macierzach o takich samych rozmiarach. Wykonują operacje logiczne na wartościach bitowych odpowiednich liczb (np. bitor(2,4) = 6, ponieważ $(2)_{10} = (0010)_2$, $(4)_{10} = (0100)_2$, $(6)_{10} = (0110)_2$
- szybkie"(short-circuit) - operatory alternatywy i koniunkcji, które działają na wyrażeniach logicznych zawierających wartości skalarne. W przypadku gdy po sprawdzeniu pierwszego operandu wartość logiczna wyrażenia jest określona ewaluacja kolejnych operandów jest pomijana. (odpowiedniki operatorów || i && w C/C++)

Ponieważ na laboratorium będą używane głównie pierwsze z wymienionych operatorów logicznych – elementowe, tylko one są podane w Tab. 4. Szczegółowy opis wszystkich, zarówno logicznych jak i pozostałych, operatorów można odnaleźć na stronach internetowych programów Octave/Matlab. Dodatkowo Tab. 4 zawiera przydatne funkcje logiczne (*all*, *any*), działające na macierzach, które często okazują się bardzo przydatne podczas pracy.

Tablica 4: Operatory i funkcje logiczne

Operator	Funkcja logiczna
$\sim$	negacja
&	koniunkcja
|	alternatywa
xor	różnica symetryczna
all	zwraca 1 jeśli wszystkie elementy są niezerowe
any	zwraca 1 jeśli jakikolwiek element jest niezerowy

10 Możliwości graficzne Octave

Podstawową funkcją do tworzenia wykresów jest funkcja **plot**.

plot(X)-rysuje wektor X w funkcji indeksu, w przypadku macierzy traktuje ją jako zestaw wektorów.

plot(X,Y)-wykreśla wektor Y w funkcji wektora X. Gdy X lub Y jest macierzą to wektor jest rysowany odpowiednio w funkcji kolumn lub rzędów.

plot(X,Y,Z)-wykreśla jak funckja plot(X,Y) oraz dodatkowo pozwala wybierać kolor, rodzaj linii i symbole punktów.

Kolor
y-żółty
m-różowy
c-błękitny
r-czerwony
g-zielony
b-niebieski
w-biały
k-czarny

Symbole punktów
.-point
o-circle
x-x-mark
+-plus
*-star
s-kwadraty
d-romb
v-trójkąt w dół
<-trójkąt w lewo
>-trójkąt w prawo
p-pięciokąt

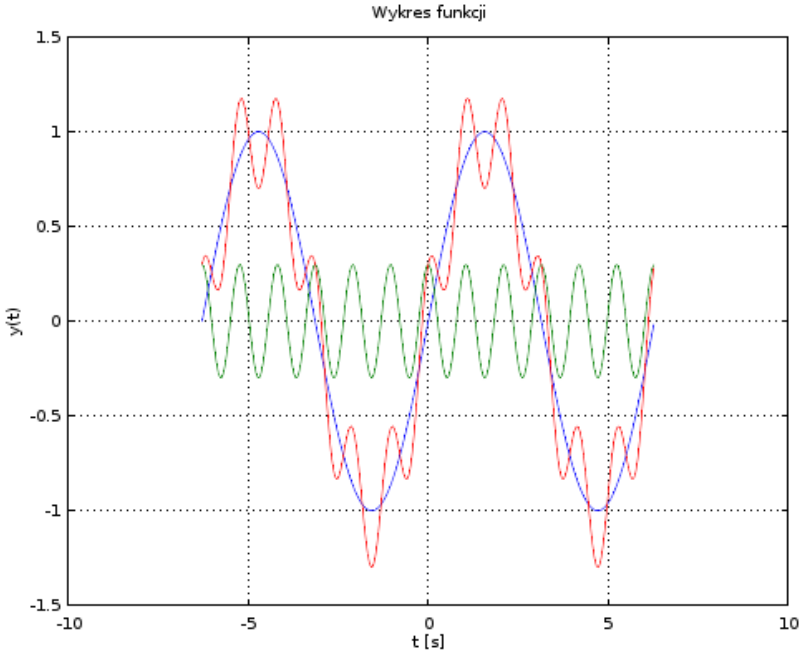
Rodzaj linii
- ciągła
- - przerywana
: kropkowana
-. przerywana-kropkowana

10.1 Przykład

W przedziale czasu $-2\pi \leqslant t \leqslant 2\pi$ przy $\Delta t = 0.05$ narysuj na jednym wykresie przebiegi trzech funkcji:

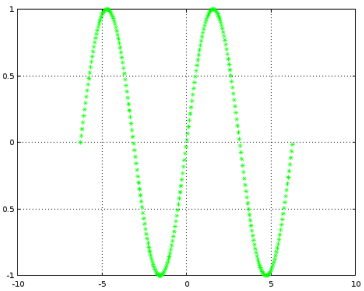
$s = \sin(t)$ $c = 0.3\cos(6t)$ $sc = s + c$

```
>> t=[-2*pi:0.05:2*pi];          %wygenerowanie wektora z przedzialem czasu
>> s=sin(t);                     %pierwsza funkcja
>> c=0.3*cos(6*t);               %druga funkcja
>> sc=s+c;                       %trzecia funkcja
>> plot(t,s,t,c,t,sc),grid;      %wyrysowanie trzech funkcji wraz z siatka
```



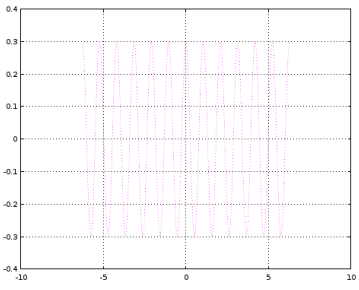
Rysunek 1: Otrzymany wykres

```
>> xlabel('t(s)');                %podspisanie osi x
>> ylabel('y(t)');                %podpisanie osi y
>> title('Wykres funkcji');       %podpisanie wykresu
```



plot(t,s, 'g*')

wykreślenie funkcji s wraz z siatką- zielone gwiazdki



plot(t,c, 'm:')

wykreślenie funkcji c wraz z siatką- różowa kropkowana linia

11 Programowanie w Octave

11.1 Tworzenie m-plików

Przy bardziej skomplikowanych obliczeniach opisany powyżej tryb interaktywny (kalkulator) jest wykorzystywany w niewielkim stopniu. Pracę można zautomatyzować i uprościć korzystając z "trybu wsadowego". Polecenia, które wydalibyśmy w linii poleceń, będziemy grupować w m-plikach (skrypty lub funkcje), które edytujemy jakimkolwiek dostępnym edytorem tekstu (np. Notatnik, Matlab-owy Meditor itp.), a następnie nazwę pliku wywołujemy z linii poleceń. Wtedy interpreter języka zacznie przetwarzać linijka po linijce polecenia zawarte w pliku. W ten właśnie sposób będziemy pracować na laboratorium.

Przed rozpoczęciem pracy z programem musimy utworzyć skrypt, w którym będziemy umieszczać nasze polecenia. W wybranym katalogu tworzymy nowy plik tekstowy i zapisujemy go z rozszerzeniem .m (w przeciwnym przypadku program nie będzie potrafił zinterpretować pliku). Załóżmy, że utworzyliśmy plik C:\mojkatalog\Lab0.m

Matlab/Octave widzi tylko skrypty, które znajdują się w aktualnym katalogu oraz skrypty określone zmienną matlabpath/path. Dlatego po uruchomieniu programu zmieniamy aktualny katalog poleceniem:
cd 'C:\mojkatalog'

Teraz będziemy mogli uruchamiać komendy zawarte w pliku Lab0.m

wpisując w linii poleceń jego nazwę bez rozszerzenia, czyli:

>> Lab0

11.2 Instrukcja warunkowa if

```
if wyrazenie_warunkowe1
instrukcje1 ;
elseif wyrazenie_warunkowe2
instrukcje2 ;
elseif
...
else
instrukcje_koncowe ;
end                %endif
```

Przykład zastosowania

```
a = 5 ;
if ( a < 0 )
b = -1 ;
elseif a == 0
b = 0 ;
else
b = 1 ;
end
```

11.3 Instrukcja wyboru switch

```
switch wyrazenie
case wartosc1
instrukcje1 ;
case wartosc2
instrukcje2 ;
...
otherwise
instrukcje;
end                % endswitch
```

Przykład zastosowania

```
c = 2 ;
switch c
case 0
a = 8 ;
case 1
```

```
a = 9 ;
disp('Zmienna c ma wartosc 1') ;
otherwise
disp(['c = ' num2str(c)]) ;
end
```

Uwaga! Nie ma dwukropków i nie potrzeba break'ów!

11.4 Pętla for

```
for zmienna = wektor_wartosci
instrukcje
end              % endfor
```

Przykład zastosowania

```
suma = 0 ;
w = 1:10 ;
for k = w
suma = suma + k ;
end
```

11.5 Pętla while

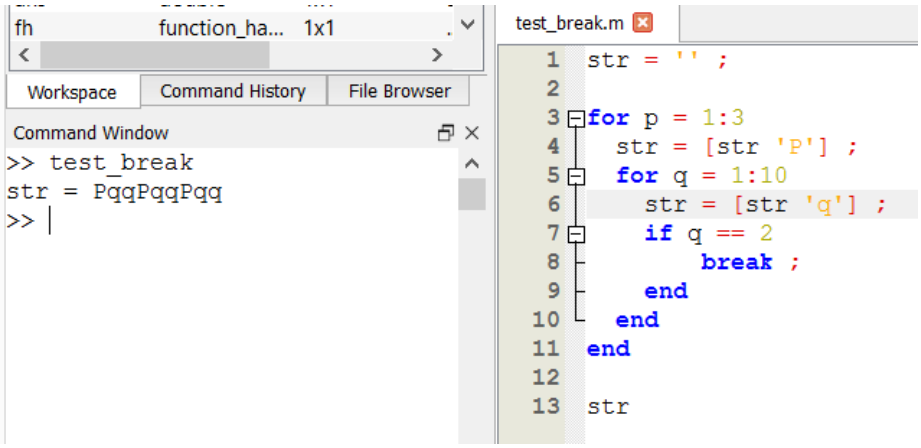
```
while wyrazenie
instrukcje
end              %endwhile
```

Przykład zastosowania

```
suma = 0 ;
k = 1 ;
while k <= 10
suma = suma + k ;
k = k + 1 ;
end
```

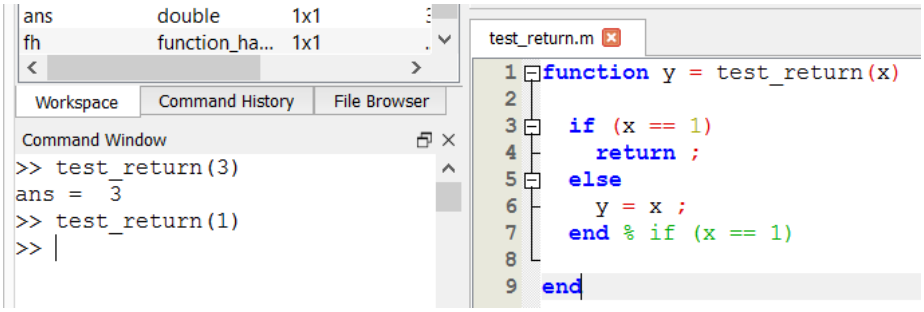
11.6 Instrukcja break

Powoduje przerwanie wykonywania danej pętli. Opuszczany jest tylko jeden poziom zagłębienia.



11.7 Instrukcja return

Powoduje bezwarunkowe przerwanie wykonywania danego skryptu (funkcji) i powrót do miejsca jego (jej) wywołania.



12 Funkcje

W programie Matlab/Octave istnieje wielu rodzajów funkcji takich jak: funkcje w m-plikach, funkcje inline, funkcje anonimowe czy funkcje zagnieżdżone. Jednak na zajęciach nie będzie istniała potrzeba korzystania z tych wszystkich możliwości. Dlatego celem uniknięcia podawania zbędnych informacji poniższy opis jest ograniczony jedynie do dwóch rodzajów, których znajmość uznano za użyteczną.

12.1 Funkcje w m-plikach

Podstawowym sposobem tworzenia własnej funkcji jest umieszczenie jej treści wraz z definicją w pliku z rozszerzeniem .m. Ogólna postać definicji funkcji wygląda następująco:

`function` $[wartosc1, wartosc2, \ldots] = nazwa\_funkcji(parametr1, parametr2, \ldots)$
instrukcje
`endfunction`

A oto przykładowa funkcja zapisana w pliku *porownaj.m*:

```
% Tutaj mozemy dac opis funkcji, ktory bedzie
% wyswietlany przez program gdy wpiszemy w linii
% polecen: help porownaj

% A ta linia wyswietlona nie zostanie

function k = porownaj(a,b)

    if (a<b)
        k = -1 ;
    elseif (a>b)
        k = 1 ;
    else
        k = 0 ;
    end
end % endfunction
```

Zarówno parametry jak i wartości funkcji są opcjonalne. Zmienne utworzone wewnątrz takiej funkcji są lokalne, czyli nie są widoczne w skrypcie, z którego została ona wywołana. Podobnie zmienne utworzone poza funkcją (znajdujące się w przestrzeni roboczej) nie są dostępne w jej wnętrzu. Aby zmienne istniejące we wspólnej pamięci programu były dostępne wewnątrz funkcji muszą być zadeklarowane jako globalne(polecenie `global`).

Charakterystyczną cechą Matlab'a/Octave jest to, że argumenty funkcji są kopiowane jedynie wtedy gdy są modyfikowane w jej ciele.

Argumenty funkcji są przekazywane przez wartość, jednak ich kopiowanie następuje jedynie wtedy gdy są modyfikowane w jej ciele. (W przykładowej funkcji *porownaj* argumenty funkcji nie były kopiowane)

UWAGA: Nazwa funkcji z definicji nie musi być taka sama jak nazwa pliku. Co więcej, w pliku definiującym funkcję może być zdefiniowanych więcej niż jedna funkcja. Jednak odwołując się do takiego pliku interpreter jako funkcję główną traktuje pierwszą zdefiniowaną funkcję. Celem uniknięcia niepotrzebnych pomyłek zaleca się w związku z tym, aby pierwszą funkcję w pliku nazywać identycznie jak sam plik wewnątrz którego jest ona zdefiniowana.

12.2 Funkcje anonimowe

Bardzo wygodnym sposobem tworzenia prostych/krótkich funkcji jest możliwość ich definiowania bezpośrednio w kodzie (lub w linii poleceń) bez konieczności tworzenia m-pliku. Składnia takiego polecenia wygląda następująco:

nazwa_funkcji = @(lista_argumentow) cialo_funkcji ;

Niech za przykład posłuży prosta funkcja, która oblicza wartość trójmianu kwadratowego przy czym parametry tego wielomianu są również podawane jako argumenty funkcji:

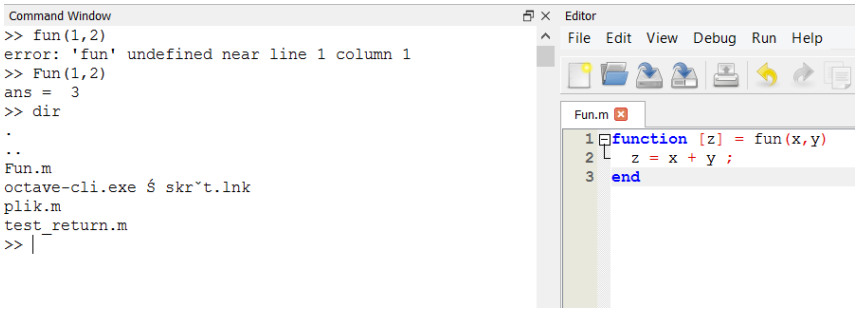
w = @(x,a,b,c) a*x.^2 + b*x + c ;

Wykorzystując funkcję zdefiniowaną w ten sposób możemy następnie obliczyć np. wartość wielomianu $2x^2 - 3x + 1$ dla punktów z przedziału $< -1, 1 >$ następująco:

y = w(-1:0.01:1, 2, -3, 1) ;

Kilka zasad dobrego postępowania z funkcjami:

- nazywaj główną funkcję wewnątrz pliku tak jak sam plik
- case sensitive – wielkość ma znaczenie!



- nie używaj polskich liter, spacji, znaków operatorów arytmetycznych/logicznych w nazwie funkcji/pliku
- a jeśli miałbyś ochotę nazwać funkcję ”coś” to też jej nie nazywaj ”cos” bo potem nie będziesz mógł obliczyć cosinusa!

Tablica 5: Skróty klawiaturowe przydatne podczas pracy z edytorem tekstu

Operator	Opis
F5	uruchomienie skryptu
F9	wykonanie zaznaczonej części kodu
Ctrl+C	przerwanie wykonywania skryptu/funkcji (przerywanie nieskończonych pętli)
Ctrl+Z	cofanie zmian
Ctrl+Shift+Z	przywróć cofniętą zmianę
Ctrl+R	zakomentowanie zaznaczonych linii
Ctrl+Shift+R	zakomentowanie zaznaczonych linii
Tab	blokowe wcięcie
Ctrl+Shift+Tab	cofanie blokowych wcięć
↑	poruszanie się ...
↓	... po historii poleceń

Tablica 6: Przydatne funkcje

Funkcja	Opis
help nazwa_polecenia	wyświetlenie informacji dotyczących polecenia (np. help format, help plot)
clc	czyszczenie okna poleceń
clear A B	usunięcie z pamięci podręcznej zmiennej A, B, itd.
clear all	usunięcie całej pamięci podręcznej
who	wyświetla nazwy zmiennych lokalnych przechowywanych w pamięci podręcznej
whos	to samo co powyżej + dodatkowe informacje na temat każdej zmiennej
tic	uruchomienie "stopera"
toc	wyłączenie "stopera" i wyświetlenie czasu od jego uruchomienia (w sekundach)
pwd	wyświetlenie aktualnego katalogu roboczego
cd	przejście do katalogu, który staje się katalogiem roboczym
dir	wyświetlenie zawartości aktualnego katalogu roboczego
ls	wyświetlenie zawartości aktualnego katalogu roboczego
quit	wyjście z programu
exit	wyjście z programu
save('plik1')	zapis całej pamięci podręcznej do pliku plik1
save('plik1', 'A', 'B')	zapis do pliku plik1 wybranych zmiennych
load('plik1')	odczyt zmiennych zapisanych w pliku plik1 i umieszczenie ich w pamięci podręcznej
more on	"stronicowanie" tekstów wysyłanych na ekran (do okna linii poleceń)
more off	wyłączenie "stronicowania"

13 Ćwiczenia do zrealizowania na laboratorium

1. Utwórz własny skrypt z rozszerzeniem .m.
2. Sprawdź wyniki poniższych poleceń wpisując je do skryptu i wykonując go

```
% TWORZENIE MACIERZY
% Wymienianie elementów
A = [1 2 ...
3; 7 8 9]

B = [1, 2, 3; 7, 8, 9]

C = [1 2 3
7,8,9]

% Generowanie
%  Operator :
w1 = 1 : 0.31 : 3
w2 = -3:7
w3 = 10:-2:0

% Wykorzystanie funkcji
X = ones(2)
Y = eye(3)
Z = rand(2,5) % round(10*rand(3))

% Zbudowanie z innych macierzy
U = [A, X ; Z]
%  Mieszanie powyzszych sposobow
V = [ [9:-1:7; 1:3], zeros(2,1); 1 1 1 1]

%-----------------------------------------------------
% ODWOLYWANIE SIE DO ELEMENTOW MACIERZY
V(1,3)
V(3,1)
V(:,2)
V(2,:)
V(2, 1:3)
V(2, [1 3 4])
V([1 3], [1 3 4])
V(7)
V(:)

%-----------------------------------------------------
```

```
% USUWANIE FRAGMENTOW MACIERZY
V(:,4) = []
V(2,:) = []
% V(1,1) = []   % !!!

%-----------------------------------------------------
% OPERACJE NA MACIERZACH
A = [1 0 0;
2 3 -1;
0 7 2]

B = [1 0 3;
-1 5 2;
2 2 2]

A+B
A-B
A+2


2*A
2.*A

disp('Mnozenie macierzowe')
MM1 = A*B
MM2 = B*A

disp('Mnozenie tablicowe')
MT1 = A.*B
MT2 = B.*A

disp(['Dzielenie',' ','macierzowe'])
DM1 = A/B
DM2 = A\B
DM3 = B/A
DM4 = B\A

disp('Dzielenie tablicowe')
DT1 = A./B
DT2 = A.\B
DT3 = B./A
DT4 = B.\A

disp(['Potego' 'wanie'])
PM = A^2
A*A
```

```
PT = A.^2
A.*A

disp(['Transpozycja'])
A.' % transpozycja
A'

C = [1-i 2i    3;
-1   5+3i  2+3i;
3+i 2     2]
C.'
C' % hermitowskie sprzezenie macierzy (dla m. zespolonych)

% Operacje porownan i funkcje logiczne
A == B
A ~= B
2 < A
% A > B
% A < B
% A >= B
% A <= B

~A
A & B
% A | B
% xor(A,B)

A = [2 0 7;
0 0 1]

all(A)
any(A)

%-------------------------------------------------
%-------------- INSTRUKCJE STERUJACE ---------------%

% INSTRUKCJA "if"
a = 2
b = 4

if (a<b)
k = -1
elseif (a>b)
k = 1
else
k = 0
```

```
endif

% INSTRUKCJA "switch"
c = 2 ;
switch c
case 0
zeros(2)
case 1
disp('wpisano 1')
otherwise
disp(['wpisano ' num2str(c) ' tym razem'])
endswitch

% PĘTLA "for"
suma=0;
wekt = 1:10 ;
for k = wekt
suma=suma+k;
endfor
S=suma;

% PĘTLA "while"
suma=0;
k=1;
while k<=10
suma=suma+k;
k=k+1;
endwhile
S=suma;

% INSTRUKCJA "break"
% zatrzymuje wykonywanie pętli "for" lub "while" ...
%   ... i powoduje wyjscie poza petle
suma=0;
i=1;
while i<=10
suma=suma+i; break
i=i+1;
endwhile
S=suma;
```

3. Utwórz plik wielomian1.m i zaimplementuj w nim funkcję, która będzie liczyć wartość wielomianu $x^3 + x^2 - 3x - 3$. Narysuj wykres tej funkcji dla $x \in < -5, 5 >$.

```
x = -5:0.01:5 ;
```

```
y = wielomian1(x) ;
plot(x,y)
```

Uwaga: Jako parametr funkcja powinna przyjmować nie tylko wartości skalarne, ale również macierze (wektory). Wykorzystaj w tym celu pętle.

4. Utwórz plik wielomian2.m i zaimplementuj w nim funkcję, która będzie liczyć wartość wielomianu x^3+x^2-3x-3 przy wykorzystaniu tablicowych operatorów arytmetycznych.

5. Porównaj czas działania obu funkcji w przypadku, gdy argument jest wektorem o małej/dużej ilości elementów. Test można przeprowadzić przy pomocy następującego kodu:

```
x = linspace(-5,5,1001) ; % x = linspace(-5,5,10001) ;
tic
y = wielomian1(x) ;
toc

tic
y = wielomian2(x) ;
toc
```

6. Zdefiniuj anonimową funkcję wielomian3 obliczającą wartość wielomianu z zadania 3 i wywołaj ją dla tych samych argumentów. (Należy oczywiście wykorzystać tablicowe operatory arytmetyczne).

7. Napisz funkcję, która w wyniku zwróci macierz kwadratową o losowych elementach będących liczbami całkowitymi z zakresu $< 0, 10 >$ oraz jej ślad (Ślad macierzy to suma elementów leżących na jej przekątnej głównej).

8. Napisz funkcję silnia korzystając z instrukcji pętli.

9. Napisz funkcję silnia wykorzystując rekurencję.

10. Zdefiniuj w pliku funkcję implementującą poniższy wzór:

$$y = \begin{cases} \sin(x) & \text{dla} \quad x < 0 \\ x^2 & \text{dla} \quad 0 \leqslant x \leqslant 1 \\ 1 & \text{dla} \quad x > 1 \end{cases}$$

Narysuj wykres tej funkcji w przedziale $< -3, 3 >$ dla argumentów równoodległych z krokiem 0.1.

11. (*) Korzystając z operatorów porównań oraz operatorów tablicowych funkcję z zadania 10 można zapisać jako pojedyńczą instrukcję. Zdefiniuj tę funkcję jako funkcję anonimową.

12. Napisz funkcję która będzie obliczać pierwiastki równania kwadaratowego, danymi wejściowymi są liczby a , b oraz c reprezentujące współczynniki w równaniu $ax^2 + bx + c$. Dodaj możliwość wyrysowania paraboli.

13. Napisz funkcję znajdującą najmiejszy element na przekątnej macierzy.

14. Napisz funkcję anonimową, która dla danego argumentu x zwraca wartość równą $sin(x) + \sqrt{x}$

15. Dana jest funkcja $f_1(x) = \frac{e^x - 1}{x}$ oraz jej rozwinięcie w szereg Taylora ograniczone do czwartego rzędu dane jako $f_2(x) = 1 + \frac{1}{2!}x + \frac{1}{3!}x^2 + \frac{1}{4!}x^3$

    - Napisz program wykonujący obliczenia wartości obu funkcji dla x dążącego do zera tj. x = 1, x = 0.1, x = 0.001 itd.
    - Przedstaw obliczone wartości funkcji f_1 i f_2 w formie wykresu.

16. W przedziale $-10 \leqslant x \leqslant 10$ przy $\Delta x = 0.1$ narysuj na wykresie przebieg funkcji:

    $a = sin(2x)$ $\qquad$ $b = 4cos(6x) + 1$ $\qquad$ $c = a + \frac{1}{2}b$

    Wykres:

    - a czerwoną przerywaną linią
    - b żółtymi plusami
    - c niebieskimi gwiazdami

17. Napisz funkcję która pozwoli na wyznaczenie dowolnego n-tego wyrazu ciągu Fibonacciego.

    Ogólny wzór na wyznaczenie wyrazu ciągu Fibonacciego :

$$F_n = \begin{cases} 0 & dla \quad n = 0 \\ 1 & dla \quad n = 1 \\ F_{n-1} + F_{n-2} & dla \quad n > 1 \end{cases}$$

18. Napisz funkcję obliczającą wartość symbolu Newtona

$$\binom{n}{k}$$

    dla zadanych wartości n oraz k , $n \geqslant k$.

19. Korzystając z pętli FOR, stworzyć tablicę o wymiarach 10x10. Wyrazy tablicy mają utworzyć tabliczkę mnożenia.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100