Ćwiczenie 7. Matlab – powtórzenie

Środowisko ***Matlaba*** składa się z:

- okna ***Command Window***, w którym wpisujemy pojedyncze polecenia,
- okna ***Command History*** - historia poleceń,
- okna z zakładkami:
  - ***Workspace*** - obszar roboczy - lista zainicjowanych przez użytkownika zmiennych i ich wartości,
  - ***Current Folder*** - zawartość katalogu roboczego.

Dodatkowo można otworzyć okno edytora plików tekstowych (***Editor***) do edycji plików (np. po wybraniu w menu opcji *New Script* lub kliknięciu w istniejący plik).

Domyślne rozmieszczenie okien uzyskujemy z menu: Desktop/Desktop Layout/Default.

Podstawowa praca w środowisku odbywa się sposobami:

- **interakcyjnym**, w oknie poleceń (*Command Window*) wpisujemy polecenia i na bieżąco otrzymujemy wyniki.
- **wsadowo** – wykonywanie napisanych skryptów tekstu ASCII z rozszerzeniem **.m** (**m-plików**) zawierających ciągi poleceń.

Użyteczne polecenia

help	- pomoc globalna
help elfun	- pomoc – spis funkcji elementarnych
help *rand*	- pomoc na temat wybranej funkcji (tutaj: *rand*) – albo: *doc rand*
format long	- zwiększona dokładność wyświetlanych wyników
format short	- dokładność podstawowa
clc	- czyszczenie ekranu
clear *zmienna*	- usunięcie zmiennej z obszaru roboczego (*Workspace*)
clear	- usunięcie wszystkich zmiennych z obszaru roboczego
pause	- zatrzymanie wykonywania m-pliku (ENTER - kontynuacja)

Najważniejesze polecenie: instrukcja przypisania

```
zmienna = wyrażenie
```

W nazwach zmiennych istotne są duże i małe litery (nie stosujemy spacji ani polskich znaków). **Wyrażenie** arytmetyczne budujemy w znany sposób, łącząc stałe, zmienne (uprzednio zainicjowane) oraz funkcje za pomocą operatorów arytmetycznych. Stosujemy również nawiasy okrągłe (zmiana hierarchii operatorów).

Operatory działań arytmetycznych według priorytetu wykonywania:

^	potęgowanie
+	powielenie znaku, – zmiana znaku
* /	multiplikatywne
+ -	addytywne

Uwaga: Potęgowanie wykonywane przed zmianą znaku (-2^2= -4 ale 2^-1=0.5).

Zadanie

1. Wykonać kilka przypisań do nazwanych zmiennych i kilka przykładowych operacji obliczeniowych, np:

```
a=2.3
b=-4.7
w= (a+b)*(a-c^2)
```

Instrukcje wejścia/wyjścia

Zatrzymanie m-pliku i oczekiwanie na wpisanie wartości dla zmiennej:

```
zmienna=input("tekst zachęty")
```

Wyświetlenie wartości zmiennej:

```
disp(zmienna)
```

Wyświetlenie tekstu:

```
disp('tekst')      %Uwaga: apostrofy
```

Wydruk formatowany:

fprintf('tekst', zmienne)

Wewnątrz tekstu ustalamy miejsca wypisania wartości kolejnych zmiennych w odpowiednim formacie:

%d liczba całkowita,
%e liczba w zapisie zmiennoprzecinkowym,
%f liczba w zapisie stałoprzecinkowym.

oraz

\n - kod zmiany wiersza.

Przykładowo:

fprintf('x=%f, y=%e \n', x, y);

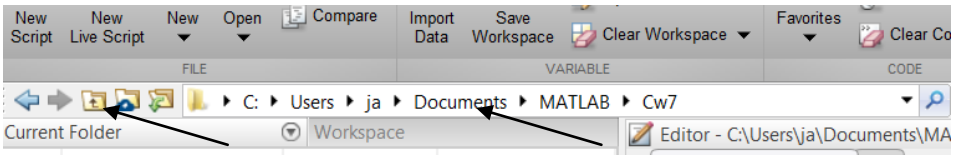
M-pliki

W *Matlab*-ie można zapisać tekst ciągu instrukcji w pliku tekstowym ASCII o rozszerzeniu m. (tzw. m-pliki), a następnie wykonać te instrukcje kolejno jedna po drugiej. *Matlab* zawiera własny edytor ASCII.

Ćwiczenie

Utworzyć własny folder.

Zmienić katalog bieżący w *Matlabie* na utworzony katalog metodami jak na rysunku:



Utworzyć nowy **m-plik** (*menu File/New Scipte* lub z menu podręcznego *okna Current Directory*) w folderze roboczym, nadając mu nazwę z rozszerzeniem **m**, np. **test1.m**,

W oknie edytora *Matlaba* napisać kolejne instrukcje jako tekst w m-pliku:

```
clc
clear
a = input('Podaj wartość a:');
b = input('Podaj wartość b:');
c = input('Podaj wartość c:');
d = a/(b+c);
fprintf('Obliczona wartość wynosi=%e \n',d)
```

Uwagi:

- instrukcje piszemy w osobnych wierszach, jeśli w jednym wierszu to oddzielamy je przecinkami,

- średniki na końcu instrukcji powodują brak wyświetlenia echa instrukcji na ekranie,

- nazwy zmiennych muszą być różne od nazwy m-pliku!

Wykonanie m-pliku:

- przy pomocy narzędzia ***Run*** w menu,
- wpisując nazwę pliku (bez rozszerzenia!) w linii poleceń *Command Window*:

```
test1
```

Wybrane funkcje arytmetyczne

Funkcje matematyczne

sin(w) cos(w) tan(w) cot(w) *(funkcje trygonometryczne wymagają kąta w radianach!)*

sind(w) cosd(w) tand(w) cotd(w) *(funkcje trygonometryczne wymagają kąta w stopniach)*

log(w)	***logarytm naturalny,***
log10(w)	***logarytm dziesiętny***
exp(w)	e^x
sqrt(w)	***pierwiastek kwadratowy***
abs(w)	***wartość bezwzględna***
fix(w)	***zaokrąglenie do całkowitej (w kierunku zera)***
floor(w)	***zaokrąglenie do całkowitej w kierunku -∞***
ceil(w)	***zaokrąglenie do całkowitej w kierunku +∞***
round(w)	***zaokrąglenie do najbliższej całkowitej***

rand ***bezargumentowy generator liczby losowej dziesiętnej z przedziału (0, 1)***

Uwaga: rand(N) losuje macierz kwadratową NxN

rand(N,M) losuje macierz prostokątną NxM

rem(a,b) ***reszta z dzielenia***

power(a,b) a^b **alternatywnie do operatora ^**

pi – stała **π**

gdzie: w, a, b, x, y – dowolne wyrażenia obliczeniowe.

Przykład kodu w m-pliku:

```
x=input('Podaj x:');
y=(x^3-exp(-2*x))/(x^6-2);
fprintf('Wynik=%f \n', y)
```

$\rightarrow \frac{x^3-e^{-2x}}{x^6-2}$

Zadania

2. Utworzyć m-plik obliczający w Matlabie wartości wyrażeń dla x=3,678:

$$\frac{|3x-5|tg\ x}{2^x-4} \qquad \frac{log(x^2+1)-x}{cos^2 3x+3{,}5} \qquad \frac{\sqrt[3]{e^x+3}+2}{x(ctg\ x-1)}$$

3. Sprawdzić algorytm zaokrąglania z dowolną dokładnością:

```
clc
clear
format long
pi5=round(pi*1E5)/1E5          %zaokrąglenie liczby π do 5-ciu miejsc dziesiętnych
```

Zmienne zespolone

Zmienna zespolona ma postać:

$a + b\,i$

gdzie: a – liczba będąca tzw. częścią rzeczywistą,

b – liczba będąca tzw. częścią urojoną,

i – jednostka urojona: $i^2=-1$

W Matlab-ie otrzymujemy wyniki w postaci zespolonej dla niektórych działań nieposiadających rozwiązań w dziedzinie liczb rzeczywistych, np. $\sqrt{-2}$, log(-2) itp.

Zadanie

4. Napisać m-plik, w którym:
   - Zainicjujemy dwie zmienne zespolone o wartościach:

     4.5+4.7i –2.5 –5.6i
   - Wykonamy i wyświetlimy wynik dodawania i mnożenia obu liczb.

Instrukcja warunkowa

Instrukcja służy do **sprawdzenia warunków** i alternatywnego wykonywania różnych grup instrukcji gdy dany warunek będzie prawdziwy (true). Postać ogólna:

```
if warunek1
  instrukcje  (wykonywane gdy jest spełniony warunek1)
elseif warunek2
  instrukcje (wykonywane gdy jest spełniony warunek2)
elseif warunek3
  instrukcje (wykonywane gdy jest spełniony warunek3)
…itd
else
  instrukcje (wykonywane gdy niespełnione oba warunki)
end
```

Bloki **else** i **elseif** mogą zostać pominięte – wówczas gdy warunek1 nie jest spełniony wykonywana jest kolejna instrukcja po instrukcji **if**. Gdy pierwszy lub kolejny warunek jest prawdziwy, pozostałe warunki nie są już sprawdzane.

Warunek to połączenie dwóch wyrażeń arytmetycznych znakami:

> < >= <= == (równe) ~= (nie równe)

Dwa warunki można związać:

- koniunkcją warunków, łącząc je operatorem logicznym **&&** (można też jeden znak **&**)
- alternatywą warunków, łącząc je operatorem logicznym **||** (można też jeden znak **|**)

Przykłady warunków:

```
a == 0          (czy jest równe, UWAGA: 2 znaki ==)
b<c
2*a >= 5
x ~= 5          (różne od)
x>0 && x<100
```

Rezultaty warunków mogą być też przypisywane zmiennym:

```
warunek=sind(40)<cosd(20);
disp(warunek)
```

1 to logiczna prawda, a 0 to logiczny fałsz

Ćwiczenie

Przeanalizować poniższy przykład, tworząc i wykonując odpowiedni m-plik:

```
clc
clear
a = input('Podaj a:');
b = input('Podaj b:');
c = input('Podaj c:');
delta =  b^2-4*a*c;
if delta<0
   disp ('delta jest ujemne')
else
  disp('delta=');
  disp(delta);
end;
```

Zadanie

5. Skorygować powyższy przykład o:
   a. obliczanie niewiadomych x1 i x2 dla dodatniej wartości *delta*,
   b. obliczanie tylko jednego pierwiastka gdy *delta* będzie równe 0,
   c. wyświetlenie komunikatu o braku rozwiązań rzeczywistych przy ujemnej wartości *delta*,
   d. sprawdzenie rozwiązań - podstawić obliczone pierwiastki do równania i przekonać się czy da to wynik 0.

Instrukcja iteracyjna („pętla liczona”)

Pętla pozwala na wielokrotne powtarzanie bloku instrukcji. Liczba powtórzeń wynika z definicji modyfikowanej wartości licznika pętli.

Postać ogólna:

```
for licznik = wartość_pocz:krok:wartość_końcowa,
 instrukcja,
 instrukcja,
...
end
```

Pominięcie kroku oznacza krok=1.

Przykład 1:

```
clc, clear
for k=1:10  % komentarz: krok=1
        disp('Witaj')
end
```

Przykład 2:

```
clc,clear
for k=100:-10:0
    disp(k)
end
```

Przykład 3:

```
clc, clear
for k=0:5:90
    y=sind(k);
    fprintf('sin(%d)=%f\n', k, y)
end
```

Przykład 4: Badanie warunku w pętli.

```
clc, clear
for k=0:360
    y=cosd(k);
    if y>0.8
        fprintf('cosd(%d)=%f\n', k, y)
    end
end
```

Zadanie

6. Wykonać m-pliki dla powyższych przykładów.
7. Napisać m-plik losujący 6 liczb ***lotto***.
8. Napisać m-plik wypisujący wartości funkcji ***tg(x)*** dla kątów z przedziału [0, 2π] z krokiem 0.1.