

Laboratorium 2

SKRYPTY

Są interpretowane, linijka po linijce – nie kompilowane w całości. A więc nie są to de facto *programy*. Skrypt to ciąg instrukcji, zapisywany z rozszerzeniem .M

Zawartość skryptu można wyświetlić w **Command Window**, używając komendy **type**. Przed stworzeniem skryptu należy się upewnić, że zapisujemy go w odpowiednim folderze, najlepiej bieżącym.

PRZYKŁAD 0.1. Pole koła ze zmienną *radius*.

```
1 % Ten skrypt wylicza pole kola
2 % Najpierw przypisujemy wartosc promienia
3 r = 5
4 % Teraz obliczamy pole na podstawie wartosci r
5 area = pi * (r^2)
```

Zapisujemy plik jako *script1.m*

Możemy teraz użyć **type**, by wyświetlić skrypt:

```
>> diary on
>> type script1
script1 is the user-defined function
  defined from: /home/paulina/MatLab/script1.m

r =5
area = pi * r^2

>> script1
r = 5
area = 78.540
>> diary off
```

Ważne, by w komentarzach opisywać, co robią poszczególne funkcje i polecenia.

Po wpisaniu *helpscript1* pojawi się pierwszy komentarz.

Komentarz może zajmować wiele linijek. Zapisujemy to następująco:

```
% krotki komentarz
kod w matlabie
%{
bardzo
bardzo
baardzo
dluugi komentarz
%}
```

INPUT / OUTPUT

Funkcja **input**

```
>> diary on
>> rad = input("Podaj promien: ")
Podaj promien: 12
rad = 12
>> diary off
```

Jeśli podajemy *char* lub *string*, dodajemy 's':

```
>> letter = input("Podaj litere: ", 's')
Podaj litere: g
letter = g
>> word = input("Podaj slowo: ", 's')
Podaj slowo: go
word = go
>> mystr = input("Podaj slowo: ", 's')
Podaj slowo:          go
mystr =              go
>> length(mystr)
ans = 11
>> mystrr = input("Podaj slowo: ", 's')
Podaj slowo: go
mystrr = go
>> length(mystrr)
ans = 11
>> diary on
>> letter = input("Podaj litere: ", 's')
Podaj litere: g
letter = g
>> word = input("Podaj slowo: ", 's')
Podaj slowo: go
word = go
>> mystr = input("Podaj slowo: ", 's')
Podaj slowo:          go
mystr =              go
>> length(mystr)
ans = 13
>> mystrr = input("Podaj slowo: ", 's')
Podaj slowo: go
mystrr = go
>> length(mystrr)
```

```
ans = 13
>> diary off
```

Powyżej widać, że spacja to też znak. Jest wliczana w długość słowa zarówno przed jak i po nim.

WYŚWIETLANIE

Funkcja **disp**.

```
>> diary on
>> disp("Hello")
Hello
>> disp(4^3)
64
>> x = input("Podaj \n wspolrzeczna x: ")
Podaj
wspolrzeczna x: 15
x = 15
>> diary off
```

Wyświetlanie wektorów i macierzy

Najlepiej również funkcją **display**. Choć, jeśli ktoś ma ochotę, może też wypróbować **fprintf**.

```
>> diary on
>> mat = [15 11 14; 7 10 13]
mat =

    15    11    14
     7    10    13

>> disp(mat)
    15    11    14
     7    10    13
>> fprintf('%d\n', mat)
15
7
11
10
14
13
>> fprintf('%d %d %d\n', mat)
15 7 11
10 14 13
```

4

```
>> fprintf('%d %d %d\n', mat')
15 11 14
7 10 13
>> vec = 2:5
vec =

     2     3     4     5

>> disp(vec)
     2     3     4     5
>> fprintf('%d ',vec)
2 3 4 5 >> fprintf('\n')

>> fprintf('%d %d %d %d\n', vec)
2 3 4 5
>> fprintf('%d %d %d\n', vec)
2 3 4
5 >>
>> diary off
```

WYKRESY

Przykładowy skrypt **plotonepoint.m** rysujący wykres jednopunktowy:

```
1 % This is a really simple plot of just one point!
2 % Create coordinate variables and plot a red '*'
3 x = 11;
4 y = 48;
5 plot(x,y,'r*')
6 % Change the axes and label them axis([9 12 35 55])
7 xlabel('Time')
8 ylabel('Temperature')
9 % Put a title on the plot
10 title('Time and Temp')
```

Do funkcji **plot** podajemy następujące argumenty: oś x argumentów, oś y wartości i sposób zaznaczenia na wykresie, tu: czerwona gwiazdka czyli red $*$.

Natomiast do funkcji **axis** podajemy wektor liczb: pierwsze dwie to minimum i maximum zakresu wartości dla osi x , a kolejne to min i max dla osi y .

Możemy też zadać to w ten sposób, bardziej ogólnie dla dowolnych wartości x, y : `axis([x-2 x+2 y-10 y+10])`.

Aby przedstawić na wykresie więcej jak jeden punkt, podajemy wektor wartości. Na ten przykład, by wyrysować następujące punkty:

(1, 1), (2, 5), (3, 3), (4, 9), (5, 11), (6, 8),

wydzielamy wektor x -ów i y -ów.

Czyli:

```
>> diary on
>> x = 1:6;
>> y = [1 5 3 9 11 8];
>> plot(x,y)
>> plot(x,y, 'r*')
>> diary off
```

Tutaj widzimy, że bez podania trzeciego argumentu wykres został uciąglony. Ponadto, osie dostosowały się do współrzędnych punktów.

W tym przykładzie x -y po prostu numerują y -ki, tzn. ich współrzędne to $1, 2, \dots, 6$. Jeśli tak jest, nie trzeba ich podawać do funkcji **plot**.

```
>> diary on
>> plot(y)
>> diary off
```

Komenda **help plot** pokaże nam różne opcje modyfikacji wykresu, na przykład kolor, styl.

Możliwe kolory to: b blue

c cyan

g green

k black

m magenta

r red

w white

y yellow.

Style wypunktowywanej krzywej to:

o kółko

d diamond

h hexagram

p pentagram

$+$ plus

$.$ punkt

s kwadrat – square

$*$ gwiazdka

v trójkąt w dół

$<$ trójkąt w lewo

> trójkąt w prawo
 ^ trójkąt w górę
 x znak x

I typy linii:

- przerywana
- . kreska-kropka
- : kropkowana
- ciągła

FUNKCJE POWIĄZANE Z WYKRESAMI

clf czyści okno Figure Window

figure tworzy nowe, puste Figure Window, gdy nie podamy argumentu, natomiast **figure(n)** pozwala na operowanie kilkoma oknami

hold przytrzymuje wykres tak, by można było nań nałożyć inne, aby go "puścić" wpisujemy znów **hold**; można też użyć **hold on** i **hold off**

legend wyświetla napisy do wykresów w kolejności ich stworzenia

grid wyświetla siatkę na wykresie, wywołana ponownie wyłącza się, można też użyć **grid on**, **grid off**.

Przykład użycia: skrypt *plot2figs.m*

```

1 % This creates 2 different plots, in 2 different
2 % Figure Windows, to demonstrate some plot features
3 clf
4 x = 1:5; % Not necessary
5 y1 = [2 11 6 9 3];
6 y2 = [4 5 8 6 2];
7 % Put a bar chart in Figure 1
8 figure(1)
9 bar(x,y1)
10 % Put plots using different y values on one plot
11 % with a legend
12 figure(2)
13 plot(x,y1,'k')
14 hold on
15 plot(x,y2,'ko')
16 grid on
17 legend('y1','y2')
```

Kolejny skrypt pokazuje graficznie różnicę między sin i cos.

```

1 % This script plots sin(x) and cos(x) in the same Figure Window
2 % for values of x ranging from 0 to 2*pi
3 clf
```

```

4 x = 0: 2*pi/40: 2*pi;
5 y = sin(x);
6 plot(x,y,'ro')
7 hold on
8 y = cos(x);
9 plot(x,y,'bp')
10 legend('sin', 'cos')
11 xlabel('x')
12 ylabel('sin(x) or cos(x)')
13 title('sin and cos on one graph')

```

Tworzy wektor x , iteruje się po przedziale $[0, 2\pi]$ co $\frac{2\pi}{40}$. Znajduje $\sin x$ i zaznacza na wykresie czerwonymi kółkami.

Zamrażamy wykres i tworzymy kolejny z niebieskimi plusami.

Można to też zrobić jedną komendą - bez używania **hold on**.

ZAPISYWANIE DO PLIKU I WCZYTYWANIE Z PLIKU

Komenda **save** służy do zapisywania danych z macierzy do pliku. Formuła jest następująca:

```

1 save nazwa_pliku nazwa_zmiennej_macierzy -ascii
2
3 %ascii używamy, gdy mamy do czynienia z tekstem

```

Na przykład:

```

>> diary on
>> mymat = rand(2,3)
mymat =

    0.136845    0.312139    0.948741
    0.038264    0.075954    0.726409

```

```

>> save testfile.dat mymat -ascii
>> diary off

```

tworzy plik przechowujący liczby w macierzy. Proszę go otworzyć komendą **type testfile.dat**

Aby dopisać coś do pliku, używamy - **append**.

Przykład:

```

>> diary on
>> mat2 = rand(3,3)
mat2 =

    0.106886    0.638477    0.092339

```

```

0.509229    0.709127    0.905990
0.285780    0.296610    0.166733

```

```

>> save testfile.dat mat2 -ascii -append
>> diary off

```

Podobnie, proszę sprawdzić, jak zmienił się plik.
Czytanie z pliku:

Funkcja **load**.

```

>> diary on
>> load testfile.dat
>> who
Variables in the current scope:

```

```
testfile
```

```

>> testfile
testfile =

```

```

0.136845    0.312139    0.948741
0.038264    0.075954    0.726409
0.106886    0.638477    0.092339
0.509229    0.709127    0.905990
0.285780    0.296610    0.166733

```

```
>> diary off
```

load działa tylko wtedy, gdy w wyświetlanym pliku zgadzają się liczby kolumni i wierszy, tak że ma postać macierzy. Podobnie **save** zapisuje tylko macierz.

Przykład: Wczytanie pliku i naniesienie danych na wykres:

```

1  % This reads time and temperature data for an afternoon
2  % from a file and plots the data
3  load timetemp.dat
4  % The times are in the first row, temps in the second row
5  time = timetemp(1,:);
6  temp = timetemp(2,:);
7  % Plot the data and label the plot
8  plot(time,temp,'k*')
9  xlabel('Time')
10 ylabel('Temperature')
11 title('Temperatures one afternoon')

```


Zadania

Proszę stworzyć *diary* o nazwie *ImieNazwisko* i zapisać w nim komendy wpisywane w Command Window podczas rozwiązywania zadań.

- (1) Masa molowa cząsteczki to masa mola atomów w pierwiastku chemicznym. Na przykład, masa molowa tlenu to 15.9994, a wodoru 1.0079. Stwórz skrypt, który wyliczy masę molową nadtlenu wodoru, który składa się z 2 atomów wodoru i 2 atomów tlenu. Dołącz komentarze. Użyj **help**, by je wyświetlić.
- (2) Użyj **input**, by poprosić użytkownika o nazwę pierwiastka. Znajdź długość podanego napisu.
- (3) Poproś użytkownika o podanie wektora.
- (4) Użyj **fprintf**, by wyświetlić liczbę 12345.67890 z 4 miejscami po przecinku, z 2 miejscami po przecinku.
- (5) Twierdzenie cosinusów: $c^2 = a^2 + b^2 - 2ab \cos \alpha$.
Napisz skrypt, w którym poprosisz o długości a, b i kąt α , a następnie wyliczysz c . Sprawdź w Command Window, jak działa (by pojawiło się w *diary*).
- (6) Napisz skrypt, który z wektora w wymiarze 3 tworzy wektor jednostkowy. Potrzebny wzór to $\frac{[x,y,z]}{\sqrt{x^2+y^2+z^2}}$.
- (7) Wiele matematycznych modeli w inżynierii używa funkcji wykładniczej. Ogólna postać spadku wykładniczego to $y(t) = Ae^{-st}$, gdzie A to wartość początkowa w czasie 0, a s to skalowanie czasu. Napisz skrypt, który przetestuje wpływ tej stałej. Dla uproszczenia ustaw $A = 1$. Poproś użytkownika o dwie różne wartości s i o początkowe i końcowe wartości dla osi czasu. Następnie wylicz odpowiednie y i przedstaw obie funkcje na wykresie (różne kolory, oznacz wykresy i osie).
Jak zmienia się prędkość spadku, gdy s rośnie?
- (8) W pliku *sales.dat* znajdują się koszty i wpływy ze sprzedaży w ciągu ostatnich kilku kwartałów. Koszty widać w pierwszej kolumnie, wpływy w drugiej. Napisz skrypt o nazwie *salescosts.m*, który wczyta dane z tego pliku do macierzy. Następnie wypisze, ile kwartałów przedstawiono w tym zestawieniu. Ponadto, narysuje wykres kosztów (kółka) i wpływów (gwiazdki). Na koniec, napisz w skrypcie funkcję, która prześle dane z macierzy do pliku o nazwie *newfile.dat*, ale w innym porządku – powinna być to macierz o 2 wierszach, w pierwszym wyniki sprzedaży, a w drugim koszty.