1 Programowanie w matlabie - skrypty i funkcje

1.1 Skrypty

Skrypt jest plikiem tekstowym z rozszerzeniem ***.m** zawierającym listę poleceń do wykonania. Aby utworzyć skrypt w matlabie wybierz 'File → New → Script', otworzy się edytor, w którym możesz wpisać listę poleceń.

Uruchomienie skryptu:

1. Wywołanie w konsoli podając nazwę skryptu (bez rozszerzenia). Ważne aby folder zawierający skrypt dodany był do ścieżki przeszukiwania matlaba.
2. Normalne uruchomienie - klawisz F5,
3. Praca krokowa + punkty przerywania (breakpointy) - pozwalają na krokowe wykonywanie skryptu wraz z podglądem stanu zmiennych, aby postawić breakpointa należy ustawić kursor w danej lini i nacisnąć F12, w tej lini powinna po lewej stronie pojawić się czerwona kropka.
4. Uruchamianie poszczególnych komórek - komórki oddzielają pewną logiczną część naszego skryptu, tworzy się je wykorzystując podwójny znak procentu, aby uruchomić daną komórkę należy w jej ciele umieścić kursor i nacisnąć Ctr+Enter.

1.2 Ćwiczenie

Stwórz nowy skrypt i wprowadź do niego poniższe polecenia. Przetestuj sposoby uruchomienia skryptu:

```
%% W pierwszym wierszu podajemy opis skryptu

%% Obliczenia
x=-2*pi:0.01:2*pi;
y1=x.*sin(x);
y2=x.*cos(x);
%% Rysujowanie wykresu
figure(1)
hold on
plot(x,y1,'r');
plot(x,y2,'g');
% dodanie legendy
legend('xsin(x)','xcos(x)');
hold off
```

1.3 Funkcje

Funkcję tworzymy i uruchamiamy podobnie jak skrypty. Najważniejsze różnice pomiędzy skryptem i funkcją:

1. w funkcji zmienne nie są widoczne w globalnej przestrzeni roboczej (workspace)
2. funkcje posiadają parametry wywołania i zwracają wynik (wyniki)
3. nazwa funkcji i nazwa pliku muszą być takie same

4. wywołując funkcję musimy podać wartości parametrów

Przykład

```
function [y]=kwadrat(x,a,b,c)
%% [y]=kwadrat(a,b,c)
%% funkcja oblicza wartość funckji kwadratowej ax^2+bx+c

y = a*x^2+b*x+c;

end
```

1.4 Instrukcje warunkowe i pętle

Pętla for

```
a=rand(10,1);
sum=0;

for i=1:length(a)
   sum=sum+a(i);
   fprintf('Suma %g\n',sum);
end
fprintf('-----------');
fprintf('Suma ostateczna %g\n',sum);
```

Warunek if

```
if(warunek1)
   % gdy warunek1 prawdziwy
elseif(warunek2)
   % gdy warunek2 prawdziwy
else
    % w przeciwnym wypadku
end
```

Operatory logiczne

||	operator 'or' (lub)
&&	operator 'and' (i)
$\sim$	negacja
==	operator równości
$\sim$=	operator nierówności

1.5 Parametry opcjonalne funkcji

Parametry opcjonalne funkcji, można za symulować wykorzystując wbudowaną zmienną 'narig'- zwracającą ilość paramterów przekazanych do funckji

```
function [y]=kwadrat(x,a,b,c)
%% [y]=kwadrat(a,b,c)
%% funkcja oblicza wartość funckji kwadratowej ax^2+bx+c

if(nargin<2)
```

```
    %nie podano a,b,c
    a=1;
    b=0;
    c=0;
elseif(nargin<3)
    %nie podano b,c
    b=0;
    c=0;
elseif(nargin<4)
    %nie podano c
    c=0;
end

y = a*x^2+b*x+c;

end
```

1.6 Ćwiczenie

Utwórz funkcję kwadrat jak wyżej, przetestuj jej wywołanie z różną ilością parametrów:

```
kwadrat(4);
kwadrat(4,1);
kwadrat(4,1,2);
kwadrat(4,1,2,1);
```

Sprawdź czy wyniki zgadzają się z zakładanymi.

1.7 Ćwiczenie

Napisz skrypt generujący wykres 3D dla podanych funkcji, wybór funkcji do narysowania ma odbywać się na podstawie bieżącego czasu, jeżeli liczba minut jest podzielna przez 2 to wyświetlamy wykres dla funkcji 1, w przeciwnym wypadku rysujemy wykres 2. Ustaww tytułu wykresu, legendy, podpisów osi, zakresów wartości dla osi $x, y \in [-3, 3]$.

1. $z = (x^2 + y^2)/(1 + y^2)$
2. $z = (x^2 + y^2)/\sin(1 + y^2)$

Do pobrania bieżącego czasu użyj funkcji clock.

1.8 Ćwiczenie

Dla $x \in [-2\pi, 2pi]$ utwórz macierz wartości sin(x), sin(2x), sin(4x), sin(8x) co 0.01π. Wartości sinusów umieść w każdym kolejnym wierszu macierzy. Następnie napisz skrypt, który narysuje wszystkie wykresy na jednym obrazie, każdy wykres innym kolorem. W celu automatyzacji pracy użyj pętli 'for' do wygenerowania macierzy danych oraz wykresów.

$$\begin{bmatrix} \sin(-2\pi) & \sin(-1.99\pi) & \dots & \sin(2\pi) \\ \sin(2*(-2\pi)) & \sin(2*(-1.99\pi)) & \dots & \sin(2*2\pi) \\ \vdots & & & \\ \sin(8*(-2\pi)) & \sin(8*(-1.99\pi)) & \dots & \sin(8*2\pi) \end{bmatrix} \tag{1}$$

1.9 Ćwiczenie

Napisz funkcję obliczającą kapitał końcowy przy danym wkładzie początkowym i podanej stopie oprocentowania wkładu.

$$V = V_0(1 + \frac{r}{m})^{m*n}$$

gdzie V_0 - wkład początkowy, r - nominalna stopa procentowa (podana jako ułamek), n - liczba lat, m- liczba kapitalizacji odsetek w roku.

Parametr m powinien być parametrem opcjonalnym, domyślnie równym 1.

1.10 Ćwiczenie

Wykorzystując funkcję z ćwiczenia 1.9 napisz skrypt, który narysuje wykres przyrostu kapitału w latach, poeksperymentuj z parametrami. Wybierz wartości parametrów, które oferuje twój bank i zobacz ile potrzebujesz lat, aby stać się milionerem przy wkładzie początkowym 1000zł.

2 Wczytywanie i zapisywanie danych

W celu zapisu i odczytywania danych matlab oferuje pliki '*.mat'. Są to binarne pliki po zawalające zapisywać do nich stan poszczególnych zmiennych.

```
save('nazwa_pliku.mat','nazwa_zmiennej');
load('naza_pliku.mat');
```

2.1 Ćwiczenie

Utwórz skrypt tworzący macierz odpowiadającą tabliczce mnożenia od 1 do 20 i zapisz ją do pliku tabliczka.mat. Następnie wyczyść wszystkie zmienne w workspace i załaduj plik .mat (klikając na niego dwukrotnie w oknie Current Folder) Sprawdź stan workspace'a.

2.2 Ćwiczenie

Zmodyfikuj ćwiczenie 1.8 w taki sposób aby tablica sinusów z danymi była wczytywana z pliku *.mat a jeżeli taki plik nie istniej to niech skrypt wygeneruje odpowiednie dane.

3 Funkcje anonimowe

Pozwalają na utworzenie funkcji bez tworzenia odpowiadającego im m-pliku. Szczególnie przydatne gdy funkcja jest mała.

Składnia

```
funcHandle = @(argumenty) funcBody;
```

funcHandle - zmienna przechowywująca uchwyt do funkcji
Przykład

```
poleKola = @(r) pi*r*r;
poleKola(3);
poleKola(0:1:5);
```

operator @ pobiera uchwyt funkcji, można go także zastosować do funkcji już zdefiniowanych w matlabie np.

```
sinHandle = @sin;
```

Funkcje anonimowe i uchwyty do funkcji można przekazać jako parametr do innej funkcji.

3.1 Ćwiczenie

Napisz funkcję 'drawPlot(x,fncHandle)', która jako parametry wejściowe dostaje wektor x oraz uchwyt do innej funkcji. Zadaniem funkcji drawPlot jest narysowanie wykresu funkcji przekazanej jako parametr. Przetestuj działnie funkcji drawPlot dla argumentu fncHandle równym:

1. funkcji anonimowej x^2
2. funkcji sin
3. funkcji cos
4. funkcji log2