1. OPIS STANOWISKA

1.1. Stosowana aparatura

- komputer klasy PC z systemem operacyjnym Microsoft Windows (XP/Vista/7).

1.2. Oprogramowanie

- środowisko Matlab R2007b (Version 7.5.0.342), classroom license.

2. WSTĘP TEORETYCZNY

2.1. Skrypty w Matlabie

Skrypt jest to plik tekstowy o rozszerzeniu **.m** zawierający polecenia i instrukcje Matlaba. Skrypt można utworzyć w dowolnym edytorze zapisującym niesformatowane pliki tekstowe. Środowisko Matlab zawiera własny edytorów m-plików (**Desktop → Editor**).

Przykład skryptu generującego pseudolosowo i wyświetlającego elementy macierzy 2×3:

```
clc;

A = rand(2,3);

disp(A);
```

Skrypt uruchamia się podając jego nazwę (bez rozszerzenia) w wierszu poleceń Matlaba. Jeśli powyższy skrypt będzie zapisany pod nazwą **s1.m**, to jego wykonanie wymaga wpisania polecenia:

```
>> s1

    0.9218    0.1763    0.9355

    0.7382    0.4057    0.9169
```

Matlab wykona skrypt jeśli będzie się on znajdował w bieżącym katalogu lub w katalogu udostępnionym poleceniem **path**.

`path`	wyświetla aktualną listę ścieżek (katalogów)
`path(path,kat1)`	dodaje do listy ścieżek katalog o podanej nazwie (**kat1**)

Bieżący katalog jest to katalog, w którym zapisywane są pliki tworzone podczas pracy. Informację o bieżącym katalogu wyświetla polecenie **pwd**. Katalog można zmienić poleceniem **cd**.

W skryptach można wstawiać komentarze. Komentarz rozpoczyna się znakiem procentu (**%**). Matlab zignoruje wszystko co znajdzie się między znakiem **%** a końcem wiersza. Jeśli pierwsze linie skryptu zaczynają się od znaków **%**, to stanowią pomoc wyświetlaną na ekranie po wywołaniu polecenia:

```
>> help skrypt
```

gdzie słowo **skrypt** jest nazwą skryptu (bez rozszerzenia **.m**).

Przykładowa zawartość skryptu **s1.m** zawierającego pomoc i komentarze:

```
% S1 Generowanie macierzy 2x3
%    S1 generuje macierz o rozmiarze 2x3 a następnie
%    wyświetla jej zawartość w postaci wektora wierszowego
clc;
A = rand(2,3);
disp(A);
% wyświetlenie macierzy w postaci wektora wierszowego
disp(A(:)');
```

Polecenie:

```
>> help s1
```

spowoduje wyświetlenie informacji o skrypcie:

```
S1 Generowanie macierzy 2x3
   S1 generuje macierz o rozmiarze 2x3 a następnie
   wyświetla jej zawartość w postaci wektora wierszowego
```

Natomiast wpisanie:

```
>> s1
```

spowoduje wykonanie skryptu.

```
0.9218    0.1763    0.9355
0.7382    0.4057    0.9169
0.9218    0.7382    0.1763    0.4057    0.9355    0.9169
```

Skrypty nie pobierają żadnych argumentów wejściowych, ani nie zwracają żadnych argumentów wyjściowych. Operują tylko na zmiennych dostępnych w przestrzeni roboczej Matlaba.

2.2. Wczytywanie danych do skryptu

Podczas wykonywania skryptu można wczytywać do niego dane z klawiatury wykorzystując funkcję **input**:

`x=input(napis)`	w tej postaci wyświetlana jest zawartość łańcucha znaków **napis**, a następnie Matlab czeka na wprowadzenie liczby, która przypisywana jest zmiennej **x**
`x=input(napis,'s')`	działa j.w., ale służy do wczytania łańcucha znaków

Przykładowy skrypt wykorzystujący wczytywanie danych:

```
% Skrypt generuje macierz prostokatna
% o rozmiarze wczytanym z klawiatury
n = input('Podaj liczbe wierszy: ');
m = input('Podaj liczbe kolumn:  ');
A = rand(n,m);
disp(A);
```

Wykonanie powyższego skryptu:

```
>> s1
Podaj liczbe wierszy: 2
```

```
Podaj liczbe kolumn:  4

    0.8147    0.1270    0.6324    0.2785

    0.9058    0.9134    0.0975    0.5469
```

Wykonywanie skryptu może być zatrzymane przy użyciu polecenia **pause.**

`pause`	zatrzymuje wykonywanie skryptu do naciśnięcia przez użytkownika dowolnego klawisza
`pause(x)`	zatrzymuje wykonywanie skryptu na **x** sekund

2.3. Funkcje w Matlabie

Funkcja różni się tym od skryptu, że można do niej przekazywać argumenty podczas wywołania oraz może ona zwracać wartości.

Funkcje definiowane przez użytkownika przechowywane są w m-plikach (plik tekstowy o rozszerzeniu **.m**). Pierwszy wiersz funkcji musi zawierać **definicję funkcji** składającą się z:

- słowa kluczowego **function**,
- listy wartości zwracanych przez funkcję,
- nazwy funkcji (musi być taka sama jak nazwa pliku, w którym się znajduje),
- listy parametrów funkcji.

Definicja funkcji ma następującą postać:

```
function [wart1, wart2,...]=nazwa(parametr1,parametr2,...)
% opis funkcji – jako komentarz
instrukcje
```

Wśród instrukcji funkcji muszą znajdować się przypisania o postaci:

```
wart1 = ...;
wart2 = ...;
```

Podczas wykonywania funkcji nie są wyświetlane wyniki działania poleceń, które kończą się średnikiem. Zmienne oraz argumenty wejściowe występujące w funkcjach są lokalne w ciele

funkcji i nie wchodzą w skład przestrzeni roboczej Matlaba. Z poziomu funkcji nie ma dostępu do zmiennych występujących w przestrzeni roboczej Matlaba. Pojawienie się w funkcji instrukcji **return** powoduje natychmiastowe przerwanie jej wykonywania.

Przykładowa funkcja oraz jej wywołanie w programie Matlab:

```
function wynik = distxy(x1,y1,x2,y2)
% funkcja obliczajaca odleglosc dwoch punktow
wynik = sqrt((x2-x1).^2+(y2-y1).^2);
>> distxy(2,2,4,4)
ans =
    2.8284
```

2.3. Wyrażenia logiczne

Wyrażenia logiczne służą do porównania wartości zmiennych o tych samych rozmiarach. W wyrażeniach logicznych mogą występować operatory relacyjne i logiczne.

Operatory relacyjne	
operator	**znaczenie**
`x == y`	`x = y`
`x ~= y`	`x ≠ y`
`x < y`	`x < y`
`x > y`	`x > y`
`x <= y`	`x ≤ y`
`x >= y`	`x ≥ y`

Operatory logiczne	
operator	**znaczenie**
`x | y`	`x` lub `y` (OR)
`x & y`	`x` i `y` (AND)
`~x`	nie `x` (NOT)

Jeśli porównywane są skalary i wyrażenie logiczne jest prawdziwe to zwracana jest wartość **1**, jeśli fałszywe - wartość **0**. Jeśli porównywane są macierze lub wektory o tych samych rozmiarach, to porównywanie wykonywane jest element po elemencie i zwracana jest

macierz zawierająca wartości **1** lub **0** na odpowiednich pozycjach (zależnie od wyniku porównania).

2.4. Instrukcja warunkowa if

Instrukcja warunkowa **if** pozwala wykonywać różne **instrukcje** w zależności od tego czy **wyrażenie logiczne** jest prawdziwe lub fałszywe. Instrukcja ta może występować w jednej z czterech poniższych postaci. **Wyrażenie** jest to wyrażenie logiczne, natomiast **instrukcje** jest to jedna lub kilka instrukcji.

`if wyrażenie`
`  instrukcje`
`end`
	- jeśli **wyrażenie** jest prawdziwe to wykonywane są wszystkie **instrukcje** znajdujące się pomiędzy wierszem zawierającym **if**, a wierszem zawierającym **end**
	- jeśli **wyrażenie** nie jest prawdziwe, to **instrukcje** nie są wykonywane

`if wyrażenie`
`  instrukcje1`
`else`
`  instrukcje2`
`end`
	- jeśli **wyrażenie** jest prawdziwe to wykonywane są **instrukcje1**, natomiast **instrukcje2** nie są wykonywane
	- jeśli **wyrażenie** nie jest prawdziwe to wykonywane są **instrukcje2**, natomiast **instrukcje1** nie są wykonywane

`if wyrażenie1`
`  instrukcje1`
`elseif wyrażenie2`
`  instrukcje2`
`end`
	- jeśli **wyrażenie1** jest prawdziwe to wykonywane są **instrukcje1**, natomiast nie jest sprawdzana prawdziwość **wyrażenia2** oraz nie są wykonywane **instrukcje2**
	- jeśli **wyrażenie1** nie jest prawdziwe to nie są wykonywane **instrukcje1**, sprawdzane jest natomiast **wyrażenie2**, jeśli jest ono prawdziwe, to wykonywane są **instrukcje2**

`if wyrażenie1`
`  instrukcje1`
`elseif wyrażenie2`
`  instrukcje2`
	- jeśli **wyrażenie1** jest prawdziwe to wykonywane są **instrukcje1**, natomiast nie jest sprawdzana prawdziwość **wyrażenia2** oraz nie są wykonywane **instrukcje2** i **instrukcje3**
	- jeśli **wyrażenie1** nie jest prawdziwe to nie są

`else`
`instrukcje3`
`end`
	wykonywane **instrukcje1**, sprawdzane jest natomiast **wyrażenie2**, jeśli jest ono prawdziwe, to wykonywane są **instrukcje2**, w przeciwnym wypadku - **instrukcje3**

Zastosowanie instrukcji warunkowej **if** zostało przedstawione w postaci skryptu obliczającego rzeczywiste pierwiastki równania kwadratowego.

```
% Rozwiązanie równania kwadratowego
a = input('Podaj a: ');
b = input('Podaj b: ');
c = input('Podaj c: ');
if a == 0
    disp('a = 0: to nie jest równanie kwadratowe')
else
    delta = b.^2-4*a.*c;
    if delta > 0
        x1 = (-b-sqrt(delta)) / (2*a);
        x2 = (-b+sqrt(delta)) / (2*a);
        disp(strcat('x1 = ',num2str(x1)))
        disp(strcat('x2 = ',num2str(x2)))
    elseif delta == 0
        x = -b / (2*a);
        disp(strcat('x1 = x2 = ',num2str(x)))
    else
        disp('Brak pierwiastków rzeczywistych')
    end
end
```

W powyższym skrypcie wyniki obliczeń wyświetlane są przy zastosowaniu funkcji **disp**. Funkcja ta umożliwia wyświetlenie tekstu lub wartości tylko jednej zmiennej. Dodatkowo automatycznie przechodzi do nowego wiersza. Aby wyświetlić w jednym wierszu nazwę pierwiastka i jego wartość należy zamienić liczbę na tekst (funkcja **num2str**), a następnie połączyć dwa teksty w jeden (funkcja **strcat**).

2.5. Instrukcja wyboru wielowariantowego switch

Instrukcja **switch** służy do wyboru jednego z kilku wariantów:

```
switch switch_expr
   case case_expr1
        instrukcje
   case case_expr2
        instrukcje
   ...
   otherwise
         instrukcje
end
```

switch_expr może być liczbą lub łańcuchem znakowym. Wartość **switch_expr** jest porównywana z kolejnymi wartościami **case_expr**. Jeśli wartość **switch_expr** jest równa jednej z wartości **case_expr**, to wykonywane są odpowiednie instrukcje, a następnie następuje opuszczenie bloku **switch**. Jeśli żadna z wartości **case_expr** nie jest równa **switch_expr**, to wykonywane są instrukcje po opcjonalnym identyfikatorze **otherwise**.

2.6. Pętla for

Pętla for umożliwia cykliczne wykonywanie wybranych instrukcji określoną liczbę razy. Ogólna postać instrukcji **for** jest następująca:

```
for zmienna = macierz_wartości
   instrukcje
end
```

Działanie pętli **for** polega na przypisywaniu **zmiennej** kolejnych kolumn **macierzy_wartości**.

Macierz_wartości ma najczęściej jedną z dwóch postaci:

- **`min:max`** - zmiennej przypisywane są kolejne wartości od **min** do **max**, np.

  `for i = 1:4` - zmiennej **i** zostaną przypisane wartości: **1**, **2**, **3**, **4**

- **`min:krok:max`** - zmiennej przypisywane są kolejne wartości od **min** do **max** różniące się o **krok**, np.

  `for i = 1:0.5:4` - zmiennej **i** zostaną przypisane wartości: **1**, **1.5**, **2**, **2.5**, **3**, **3.5**, **4**

Poniższa funkcja **suman** oblicza sumę liczb od **1** do **n**.

```
function wynik = suman(n)
% funkcja obliczająca sumę n kolejnych liczb całkowitych
wynik = 0;
for i = 1:n
    wynik = wynik + i;
end
```

Przykładowe wywołanie funkcji **suman**:

```
>> x = suman(1234)
x =
      761995
```

2.7. Pętla while

Ogólna postać instrukcji **while**:

```
while wyrażenie
   instrukcje
end
```

Instrukcje w pętli **while** wykonywane są dopóki część rzeczywista **wyrażenia** ma wszystkie elementy różne od zera. W pętlach **for** i **while** można zastosować instrukcję **break**. Powoduje ona opuszczenie pętli i przejście do wykonywania następnej instrukcji za pętlą.

Skrypt sumujący liczby wprowadzane przez użytkownika tak długo, aż użytkownik poda liczbę zero.

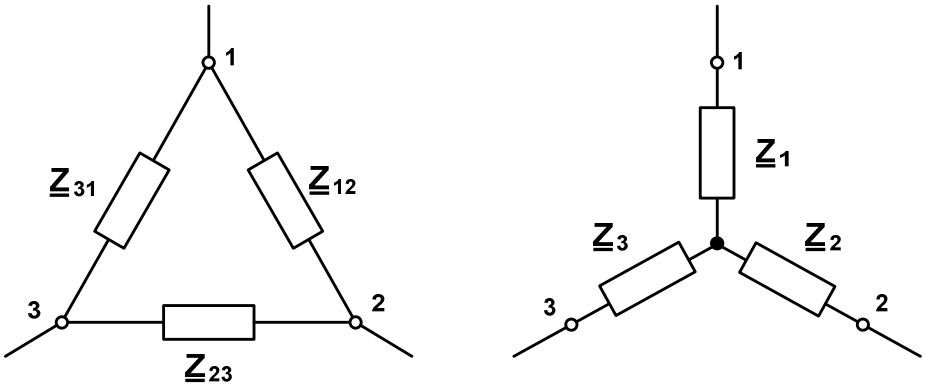
```
suma = 0;
x = input('Podaj liczbe: ');
while x ~= 0
    suma = suma + x;
    x = input('Podaj liczbe: ');
end
disp(strcat('Suma liczb: ',num2str(suma)))
```

3. PRZEBIEG ĆWICZENIA

Wykonaj podane poniżej zadania.

Zadanie 1

Napisz dwa skrypty: **trgw** - dokonującego przekształcenia trójkąta impedancji w gwiazdę oraz **gwtr** - dokonującego przekształcenia gwiazdy impedancji w trójkąt. Dodaj pomoc do skryptów.



Zadanie 2

Zamień skrypty z Zadania 1 na funkcje Matlaba.

Zadanie 3

Napisz funkcję Matlaba obliczającą częstotliwość rezonansową **fr** układu o rezystancji **R**, indukcyjności **L** i pojemności **C**.

$$f_r = \frac{1}{2\pi\sqrt{LC - (RC)^2}}$$

Sprawdź poprawność skryptu dla danych:

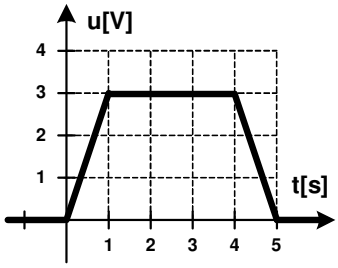
$$R = 10\,\Omega, \quad L = 0{,}1\,H, \quad C = 1\,\mu F \quad \rightarrow \quad f_r = 503{,}54398\,\mathrm{Hz}$$

Zadanie 4

Napisz skrypt **silnia** obliczający silnię liczby **n** wprowadzanej z klawiatury przez użytkownika. Dodaj pomoc do skryptu. Zabezpiecz skrypt przed obliczaniem silni liczby ujemnej lub zbyt dużej.

Zadanie 5

Na rysunku przedstawiony jest przebieg impulsu prostokątnego. Napisz funkcję Matlaba, która dla argumentu będącego czasem (**t**) zwraca wartość napięcia (**u**).



Sprawdź poprawność funkcji dla wektora zawierającego liczby od **-1** do **6** z krokiem **0,5**.

4. LITERATURA

[1] Mrozek B., Mrozek Z.: MATLAB i Simulink. Poradnik użytkownika. Wydanie III. Helion, Gliwice, 2010.

[2] Stachurski M. Treichel W.: Matlab dla studentów. Ćwiczenia, zadania, rozwiązania. Witkom, Warszawa, 2009.

[3] Pratap R.: MATLAB 7 dla naukowców i inżynierów. Wydawnictwo Naukowe PWN, Warszawa, 2010.

[4] Brzóska J., Dorobczyński L.: Matlab: środowisko obliczeń naukowo-technicznych. „Mikom", Wydawnictwo Naukowe PWN, Warszawa, 2008.

[5] Kamińska A., Pańczyk B.: Ćwiczenia z Matlab. Przykłady i zadania. Wydawnictwo MIKOM, Warszawa, 2002.

[6] Sobierajski M., Łabuzek M.: Programowanie w Matlabie dla elektryków. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2005.

[7] Dyka E., Markiewicz P., Sikora R.: Modelowanie w elektrotechnice z wykorzystaniem środowiska MATLAB. Wydawnictwa Politechniki Łódzkiej, Łódź, 2006.

[8] Czajka M.: MATLAB. Ćwiczenia. Helion, Gliwice, 2005.

5. ZAGADNIENIA NA ZALICZENIE

1. W jaki sposób definiuje się pomoc do skryptów w Matlabie?
2. Jaka jest struktura definicji funkcji w Matlabie?
3. Jakie są różnice pomiędzy skryptami a funkcjami w Matlabie?
4. Omów składnię i zastosowanie instrukcji warunkowej **if** oraz instrukcji wyboru wielowariantowego **switch**.
5. Omów składnię i zastosowanie pętli **for** i **while**.