Politechnika Białostocka
Wydział Elektryczny
Katedra Elektrotechniki Teoretycznej i Metrologii

Instrukcja
do pracowni specjalistycznej z przedmiotu

Informatyka 1

Kod przedmiotu: **EZ1D200 008**

(studia niestacjonarne)

MATLAB - SKRYPTY I FUNKCJE, ELEMENTY PROGRAMOWANIA

Numer ćwiczenia

INF08Z

Autor:
dr inż. Jarosław Forenc

Białystok 2017

Spis treści

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows (XP/7/10).

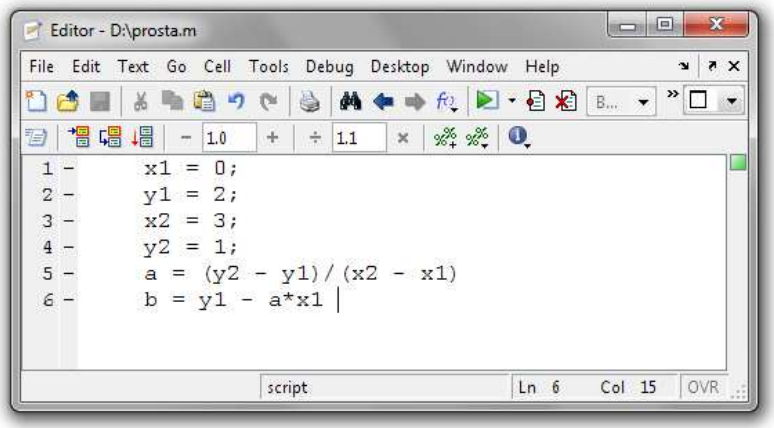
1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko Matlab R2007b (Version 7.5.0.342), classroom license.

2. Wiadomości teoretyczne

2.1. Skrypty

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 (Rys. 1) wywoływany z menu **Desktop → Editor**.



Rys. 1. Edytor m-plików

Poniżej znajduje się przykładowa zawartość skryptu obliczającego i wyświetlającego współczynniki **a** i **b** równania prostej **y = ax + b** przechodzącej przez dwa punkty o współrzędnych **(0, 2)** i **(3, 1)**.

```
x1 = 0;
y1 = 2;
x2 = 3;
y2 = 1;
a = (y2 - y1)/(x2 - x1)
b = y1 - a*x1
```

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

```
>> prosta
   a =
      -0.3333
   b =
         2
```

Wykonanie skryptu jest równoważne z wpisywaniem i zatwierdzaniem kolejnych jego poleceń w wierszu poleceń (*Command Window*).

Matlab wykona skrypt jeśli będzie on znajdował się 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 (*Current Directory*) jest to katalog, w którym zapisywane są pliki tworzone podczas pracy. Nazwa bieżącego katalogu wyświetlana jest w górnej części głównego okna programu Matlab (Rys. 2).

Rys. 2. Okno bieżącego katalogu

Informację o bieżącym katalogu można wyświetlić także poleceniem **pwd**. Katalog można zmienić wywołując polecenie **cd**.

Przy nazywaniu skryptów obowiązują takie same zasady jak w przypadku innych zmiennych w Matlabie. Nazwa skryptu nie powinna być taka sama jak nazwa zmiennej w przestrzeni roboczej lub zmiennej występującej w skrypcie. Jeśli powyższa sytuacja wystąpi, to wprowadzenie w wierszu poleceń takiej nazwy spowoduje wyświetlenie wartości zmiennej, a nie wykonanie skryptu.

2.2. Komentarze i system pomocy

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 **prosta.m** zawierającego pomoc i komentarze:

```
% PROSTA - równanie prostej y = ax + b
% Skrypt obliczający współczynniki a i b równania
% prostej y = ax + b przechodzącej przez dwa punkty
% o współrzędnych (x1,y1) i (x2,y2)
x1 = 0;
y1 = 2;
x2 = 3;
y2 = 1;
% Obliczenie wartości a i b
a = (y2 - y1)/(x2 - x1);
b = y1 - a*x1;
% Wyświetlenie wartości a i b
txt = sprintf('a = %g',a);
disp(txt);
txt = sprintf('b = %g',b);
disp(txt);
```


Polecenie:

```
>> help prosta
```

spowoduje wyświetlenie informacji o skrypcie:

```
PROSTA - równanie prostej y = ax + b
Skrypt obliczający współczynniki a i b równania
prostej y = ax + b przechodzącej przez dwa punkty
o współrzędnych (x1,y1) i (x2,y2)
```

Natomiast wpisanie:

```
>> prosta
```

spowoduje wykonanie skryptu.

```
a = -0.333333
b = 2
```

W powyższym skrypcie do sformatowania wyświetlanych wyników zastosowano funkcję **sprintf**, która ma identyczną składnię jak funkcja **printf** w języku C. Funkcja **sprintf** zwraca łańcuch znaków. Łańcuch ten wyświetlany jest poleceniem **disp**. Wywołanie funkcji **sprintf** można wstawić bezpośrednio do polecenia **disp**:

```
% Wyświetlenie wartości a i b
disp(sprintf('a = %g',a));
disp(sprintf('b = %g',b));
```

Skrypty nie pobierają żadnych argumentów wejściowych, ani nie zwracają żadnych argumentów wyjściowych. Operują tylko na zmiennych dostępnych w przestrzeni roboczej Matlaba. Wyniki wykonania skryptu (np. utworzone zmienne) pozostają w przestrzeni roboczej.

2.3. Wczytywanie danych do skryptu

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

`x=input(napis)`	wyświetla **napis**, a następnie 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:

```
% PROSTA – równanie prostej y = ax + b
% Skrypt obliczający współczynniki a i b równania
% prostej y = ax + b przechodzącej przez dwa punkty
% o współrzędnych (x1,y1) i (x2,y2)
disp('Współrzędne punktu nr 1:');
x1 = input('x1: ');
y1 = input('y1: ');
disp('Współrzędne punktu nr 2:');
x2 = input('x2: ');
y2 = input('y2: ');
% Obliczenie wartości a i b
a = (y2 – y1)/(x2 – x1);
b = y1 – a*x1;
% Wyświetlenie wartości a i b
disp(sprintf('a = %g',a));
disp(sprintf('b = %g',b));
```

Wykonanie powyższego skryptu:

```
>> prosta
   Współrzędne punktu nr 1:
   x1: 0
   y1: 2
   Współrzędne punktu nr 2:
   x2: 3
   y2: 1
   a = –0.333333
   b = 2
```

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

2.4. Funkcje

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 m-pliku, w którym się znajduje);
- listy parametrów funkcji.

Definicja funkcji ma następującą postać:

```
function [wart1, wart2,...]=nazwa(par1,par2,...)
% opis funkcji – jako komentarz
instrukcje
```

Wśród instrukcji funkcji muszą znajdować się przypisania o postaci:

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

Przykładowa funkcja obliczająca i zwracająca wartość współczynników **a** i **b** równania prostej **y = ax + b** przechodzącej przez dwa punkty o współrzędnych **(x1,y1)** i **(x2,y2)**:

```
function [a,b] = prosta(x1,y1,x2,y2)
% PROSTA – równanie prostej y = ax + b
% Funkcja obliczająca współczynniki a i b równania
% prostej y = ax + b przechodzącej przez dwa punkty
% o współrzędnych (x1,y1) i (x2,y2)
a = (y2 – y1)/(x2 – x1);
b = y1 – a*x1;
```

W wywołaniu funkcji należy podać wartości jej argumentów oraz nazwy zmiennych pod które zostaną podstawione zwrócone wartości:

```
>> [wa,wb] = prosta(0,2,3,1)
   wa =
       -0.3333
   wb =
        2
```

Jeśli nie zostaną podane nazwy zmiennych wynikowych, to wyświetlona zostanie tylko jedna zwracana wartość:

```
>> prosta(0,2,3,1)
   ans =
      -0.3333
```

W przypadku, gdy funkcja zwraca tylko jedną wartość, to nie trzeba stosować nawiasów kwadratowych w pierwszym wierszu jej definicji. Poniższa funkcja oblicza pole koła o promieniu **r**.

```
function p = pole(r)
% POLE - pole koła o promieniu r
p = pi*r.^2;
```

Przykładowe wywołanie funkcji **pole**:

```
>> pole(2.5)
   ans =
      19.6350
```

Argumentami funkcji mogą być także zmienne, wyrażenia arytmetyczne lub wywołania innych funkcji.

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 i powrót do miejsca wywołania.

2.5. 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)

`xor(x,y)`	`x` xor `y` (XOR)

Jeśli porównywane są skalary i wyrażenie logiczne jest prawdziwe to zwracana jest wartość **1**, jeśli fałszywe - wartość **0**.

```
>> x = 3; y = 4;

>> x > y      >> x < y      >> x == y     >> x ~= y
   ans =         ans =         ans =         ans =
        0             1             0             1
```

Jeśli porównywane są wektory lub macierze o tych samych rozmiarach, to porównywanie wykonywane jest element po elemencie i zwracany jest wektor lub macierz zawierająca wartości **1** lub **0** na odpowiednich pozycjach (zależnie od wyniku porównania).

```
>> x = [3 6 2 4 5];
>> y = [4 3 2 7 1];

>> x > y
   ans =
        0   1   0   0   1

>> x ~= y
   ans =
        1   1   0   1   1
```

W przypadku operatorów logicznych wszystkie ich argumenty o wartościach różnych od **0** są traktowane jako prawda, zaś równych **0** - jako fałsz. Jeśli argumentami operatorów logicznych są wektory lub macierze o tych samych rozmiarach, to operacje wykonywane są element po elemencie i zwracany jest wektor lub macierz zawierająca wartości **1** lub **0** na odpowiednich pozycjach.

```
>> x = [0 2 1 0 3 0];
>> y = [1 3 0 2 1 0];
>> x & y
   ans =
        0   1   0   0   1   0
>> x | y
   ans =
        1   1   1   1   1   0
>> ~x
   ans =
        1   0   0   1   0   1
>> xor(x,y)
   ans =
        1   0   1   1   0   0
```

Do operacji na wektorach i macierzach można zastosować także specjalne funkcje logiczne przedstawione poniżej.

`all(A)`	zwraca **1** jeśli wszystkie elementy wektora **A** są różne od zera, natomiast jeśli przynajmniej jeden element wektora **A** jest równy zero, to zwraca **0**; jeśli **A** jest macierzą, to sprawdzenie odbywa się oddzielnie dla każdej kolumny macierzy, a wynikiem jest wektor wierszowy zawierający zera i jedynki
`any(A)`	zwraca **1** jeśli przynajmniej jeden element wektora **A** jest różny od zera, natomiast jeśli wszystkie elementy wektora **A** są równe zero, to zwraca **0**; jeśli **A** jest macierzą, to sprawdzenie odbywa się oddzielnie dla każdej kolumny macierzy, a wynikiem jest wektor wierszowy zawierający zera i jedynki

`isequal(A,B,…)`	zwraca **1** jeśli macierze będące argumentami funkcji mają taki sam rozmiar i taką samą zawartość; w przeciwnym przypadku zwraca **0**
`isempty(A)`	zwraca **1** jeśli macierz **A** jest pusta (nie ma żadnych elementów) lub zwraca **0** jeśli macierz nie jest pusta

2.6. 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`
`else`
`  instrukcje3`
`end`
	- 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 wykonywane **instrukcje1**, sprawdzane jest natomiast **wyrażenie2**, jeśli jest ono prawdziwe, to wykonywane są **instrukcje2**, w przeciwnym wypadku - **instrukcje3**

W poniższym skrypcie instrukcja **if** została zastosowana do sprawdzenia czy osoba o podanym roku urodzenia jest pełnoletnia.

```
% TEST - skrypt sprawdzający czy osoba jest pełnoletnia
rok = input('Podaj rok urodzenia: ');
wiek = 2019 - rok;
if wiek >= 18
    disp('Osoba pełnoletnia');
else
    disp('Osoba niepełnoletnia');
end
```

Przykładowe wywołania powyższego skryptu:

```
>> test
   Podaj rok urodzenia: 2010
   Osoba niepełnoletnia
>> test
   Podaj rok urodzenia: 1998
   Osoba pełnoletnia
```

Poniższy skrypt rozwiązuje równanie kwadratowe i zawiera najbardziej rozbudowaną postać instrukcji warunkowej **if**.

```
% ROW_KW - 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
```

Przykładowe wywołanie powyższego skryptu:

```
>> row_kw
   Podaj a: 2
   Podaj b: 8
   Podaj c: 2
   x1 =-3.7321
   x2 =-0.26795
```

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**). Do sformatowania wyświetlanego wyniku można zastosować także funkcję **sprintf**.

2.7. 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**. W programie Matlab, w przeciwieństwie do języka C, na końcu każdego bloku **case** nie trzeba umieszczać instrukcji **break**.

Kolejny skrypt wyświetla słownie ocenę wczytaną z klawiatury.

```
% OCENA - skrypt wyświetlający słownie ocenę
x = input('Podaj ocene: ');
switch x
    case 5
        disp('Twoja ocena: bardzo dobry');
        disp('Brawo!');
    case 4
        disp('Twoja ocena: dobry');
    case 3
        disp('Twoja ocena: dostateczny');
    case 2
        disp('Twoja ocena: niedostateczny');
        disp('Musisz poprawić się!');
    otherwise
        disp('Błędna ocena');
end
```


Przykładowe wywołania powyższego skryptu:

```
>> skrypt
   Podaj ocene: 5
   Twoja ocena: bardzo dobry
   Brawo!
>> skrypt
   Podaj ocene: 4
   Twoja ocena: dobry
>> skrypt
   Podaj ocene: 2
   Twoja ocena: niedostateczny
   Musisz poprawić się!
>> skrypt
   Podaj ocene: 0
   Błędna ocena
```

W instrukcji **switch** można umieścić instrukcję **break**. Spowoduje ona przerwanie wykonywania instrukcji **switch**.

2.8. 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)
% SUMAN - suma n kolejnych liczb całkowitych
wynik = 0;
for i = 1:n
    wynik = wynik + i;
end
```

Przykładowe wywołanie funkcji **suman**:

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

W pętli **for** można umieścić instrukcję **break**. Spowoduje ona przerwanie wykonywania pętli i przejście do wykonywania następnej instrukcji za pętlą.

Pętle **for** można zagnieżdżać. Do poniższej funkcji przekazywana jest macierz **A** oraz liczba **x**. Funkcja sprawdza ile razy **x** występuje w macierzy.

```
function ile = policz(A,x)
% POLICZ - funkcja sprawdzająca ile razy x
%          występuje w macierzy A
rows = size(A,1);    % liczba wierszy
cols = size(A,2);    % liczba kolumn
ile = 0;
for i = 1:rows
    for j = 1:cols
        if A(i,j) == x
            ile = ile + 1;
        end
    end
end
```

Przykładowe utworzenie macierzy **A** zawierającej pseudolosowe liczby całkowite z zakresu ⟨**0,10**⟩ i wywołanie funkcji:


```
>> A = round(rand(4,6)*10)
   A =
      10    10     9     8     3     6
       1     0     1     4     1     5
       4     8     4     9     1     1
       1     8     3     2     9     9
>> ile = policz(A,1)
   ile =
       6
```

Do elementu macierzy **A** znajdującego się w wierszu o indeksie **i** oraz kolumnie o indeksie **j** odwołujemy się poprzez **A(i,j)**. Elementem takim można posługiwać się jak każdą inną zmienną. Indeksy wierszy i kolumn rozpoczynają się od wartości **1**.

```
>> A = [3 7 6; 4 2 1]
   A =
      3     7     6
      4     2     1
```

```
>> A(1,1)
   ans =
       3
```

```
>> A(2,3)
   ans =
       1
```

Do elementów macierzy można odwoływać się także przy użyciu jednego indeksu:

- jeśli **A** jest wektorem, to odwołanie **A(i)** oznacza odwołanie do i-tego elementu wektora;
- jeśli **A** jest macierzą dwuwymiarową, to odwołanie **A(i)** oznacza odwołanie do wektora kolumnowego uformowanego z kolejnych kolumn oryginalnej macierzy, umieszczonych jedna pod druga, np.

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(2,3)
   ans =
       6
```

```
>> A(6)
   ans =
       8
```

Wykorzystując dwukropek można odwoływać się do wybranych fragmentów macierzy.

`A(i,:)`	**i**-ty wiersz macierzy **A**
`A(:,j)`	**j**-ta kolumna macierzy **A**
`A(:)`	cała macierz w postaci wektora kolumnowego
`A(:,:)`	cała macierz (dwuwymiarowa)
`A(i,j:k)`	elementy **i**-tego wiersza macierzy **A** o numerach od **j** do **k**
`A(i:j,k:l)`	elementy od **i**-tego do **j**-tego wiersza oraz od **k**-tej do **l**-tej kolumny
`A(X,i:j)`	wszystkie elementy w kolumnach od **i** do **j** i wierszach macierzy **A** o numerach określonych przez elementy wektora **X**

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(2,:)
   ans =
         4   5   6

>> A(2:3,2:3)
   ans =
         5   6
         8   9

>> A(:)
   ans =
         1
         4
         7
         2
         5
         8
         3
         6
         9
```

```
>> A(2,1:2)
   ans =
         4   5

>> A(:,[1 3])
   ans =
         1   3
         4   6
         7   9
```

Można usunąć wybrane elementy macierzy przypisując im wartość w postaci macierzy pustej symbolizowanej przez puste nawiasy kwadratowe.


```
>> A = [1 2 3; 4 5 6; 7 8 9];
   >> A(:,1)=[]
   A =
        2 3
        5 6
        8 9
```

2.9. 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 pętli **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:

```
% SUMA0 - suma liczb wprowadzanych z klawiatury
suma = 0;
x = input('Podaj liczbe: ');
while x ~= 0
    suma = suma + x;
    x = input('Podaj liczbe: ');
end
disp(sprintf('Suma liczb: %d',suma))
```

Przykładowe wywołanie skryptu:

```
>> suma0
   Podaj liczbe: 7
   Podaj liczbe: 4
   Podaj liczbe: 0
   Suma liczb: 11
```

3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Przez opornik o rezystancji **R** płynie stały prąd **I**. Napisz skrypt, który obliczy i wyświetli napięcie na oporniku **U** oraz wydzielającą się w nim moc **P**. Wartości rezystancji i prądu wczytaj z klawiatury w skrypcie. Dodaj pomoc do skryptu. Wywołanie skryptu i wyświetlenie wyników powinno mieć następującą postać:

```
Podaj R [Om]:     470
Podaj I [A]:      0.25
------------------------
Napiecie U [V]:  117.5
Moc P [W]:       29.375
```

2. Przez opornik o rezystancji **R** płynie stały prąd **I**. Napisz funkcję, która obliczy i zwróci napięcie na oporniku **U** oraz wydzielającą się w nim moc **P**. Wartości rezystancji i prądu powinny być argumentami funkcji. Dodaj pomoc do funkcji. Wywołaj napisaną funkcję.

3. Napisz funkcję obliczającą częstotliwość rezonansową **fr** układu o rezystancji **R**, indukcyjności **L** i pojemności **C**. Dodaj pomoc do funkcji.

$$f_r = \frac{1}{2\pi\sqrt{LC}}\sqrt{1-\frac{L}{R^2C}} \qquad (1)$$

Sprawdź poprawność funkcji dla danych:

$$R = 100\ \Omega, \quad L = 0{,}05\ H, \quad C = 5\ mF \quad \rightarrow \quad f_r = 10{,}0608\ Hz$$

4. Admitancja zastępcza obwodu elektrycznego określona jest wzorem (2). Napisz skrypt, który wyświetli wykres konduktancji **G** i susceptancji **B** w funkcji pulsacji ω. Pulsacja powinna zmieniać się w zakresie od **100** do **500 rad/s**. Podziel okno graficzne na dwie części (umieść wykresy obok siebie). Opisz osie, dodaj tytuły wykresów, włącz wyświetlanie pomocniczej siatki.

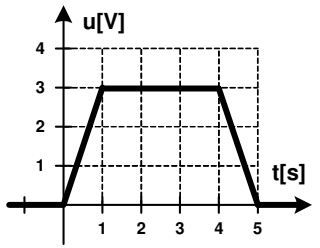
$$\underline{Y} = G + jB = \frac{R}{R^2 + X_C^2} + j\left(\frac{X_C}{R^2 + X_C^2} - \frac{1}{X_L}\right) \qquad (2)$$

Przyjmij następujące parametry obwodu:

$$R = 15\ \Omega,\ L = 0{,}125\ H,\ C = 0{,}254\ mF$$

5. Napisz skrypt, w którym użytkownik wprowadza z klawiatury liczbę wierszy i liczbę kolumn dwóch macierzy **A** i **B** (obie macierze powinny mieć takie same wymiary). Zapisz do macierzy pseudolosowe liczby całkowite z zakresu ⟨**0,5**⟩. Wyświetl elementy obu macierzy. Stosując pętlę **for** i instrukcję **if** sprawdź ile jest w obu macierzach powtarzających się elementów (te same wartości w tych samych miejscach macierzy). Zastanów się, czy powyższą operację można wykonać bez stosowania pętli **for** i instrukcji warunkowej **if**. Jeśli tak, to podaj odpowiednie instrukcje.

6. Na rysunku przedstawiony jest przebieg impulsu trapezowego. Napisz funkcję, **impuls** która dla argumentu będącego czasem (**t**) zwraca wartość napięcia (**u**). Następnie napisz skrypt, który stosując powyższą funkcję, narysuje wykres impulsu trapezowego dla czasu **t** od **-1 s** do **7 s**. Opisz osie i dodaj tytuł wykresu.



7. Napisz skrypt, który utworzy macierz zawierającą:
   - wartości czasu **t** z przedziału od **0** do **0,02 s** (100 wartości) zapisane w pierwszym wierszu macierzy;

- wartości chwilowe napięcia na dwójniku RLC (drugi wiersz macierzy) obliczone według wzoru:

$$u(t) = 10 \cdot \sin((5000 \cdot t + 10)/15) \quad (3)$$

- wartości chwilowe prądu na dwójniku RLC (trzeci wiersz macierzy) obliczone według wzoru:

$$i(t) = 5 \cdot \sin(5000 \cdot t/15) \quad (4)$$

Następnie skrypt powinien wykonać następujące operacje:

- zapisać do czwartego wiersza macierzy wartości chwilowe mocy obliczone według wzoru:

$$p(t) = u(t) \cdot i(t) \quad (5)$$

- wyświetlić na jednym wykresie wartości chwilowe napięcia, prądu i mocy w funkcji czasu (oznacz przebiegi różnymi kolorami, opisz osie, dodaj tytuł i legendę, włącz wyświetlanie siatki);
- obliczyć i wyświetlić wartości średnie napięcia, prądu i mocy;
- obliczyć i wyświetlić liczbę dodatnich i liczbę ujemnych wartości mocy chwilowej.

8. Wskaźniki zadziałania wkładek bezpiecznikowych oznacza się odpowiednimi kolorami zależnie od ich prądu znamionowego (Tabela 1).

Tabela 1. Wybrane kolory wskaźników zadziałania wkładek bezpiecznikowych

Barwa wskaźnika	Prąd znamionowy wkładki
zielona	6
czerwona	10
szara	16
niebieska	20

Napisz skrypt, w którym po wprowadzeniu przez użytkownika prądu znamionowego wkładki, wyświetlana jest barwa odpowiadającego jej wskaźnika zadziałania. W przypadku błędnej wartości prądu wyświetl odpowiedni komunikat. Zastosuj instrukcję **switch**.

4. Literatura

[1] Mrozek B., Mrozek Z.: MATLAB i Simulink. Poradnik użytkownika. Wydanie IV. Helion, Gliwice, 2018.

[2] Pratap R.: MATLAB dla naukowców i inżynierów. Wydanie 2. Wydawnictwo Naukowe PWN, Warszawa, 2015.

[3] Banasiak K.: Algorytmizacja i programowanie w Matlabie. Wydawnictwo BTC, Legionowo, 2017.

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

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

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

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

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

[9] Sradomski W.: Matlab. Praktyczny podręcznik modelowania. Helion, Gliwice, 2015.

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

5. Pytania kontrolne

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**.

5. Omów składnię i zastosowanie instrukcji wyboru wielowariantowego **switch**.
6. Omów składnię i zastosowanie pętli **for** i **while**.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.

- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.