LABORATORIUM 3
ALGORYTMY OBLICZENIOWE W ELEKTRONICE I TELEKOMUNIKACJI

Wprowadzenie do środowiska Matlab

1. Podstawowe informacje

Przedstawione poniżej informacje maja wprowadzić i zapoznać ze środowiskiem Matlab oraz jego podstawowymi funkcjami.

1.1. Deklaracja zmiennych

Zmienne są podstawowym elementem każdego języka programowania. W języku Matlab nazwy zmiennych muszą spełniać dwa podstawowe warunki:

- mogą zawierać tylko litery a-z, cyfry 0-9 i podkreślenie '_',
- muszą rozpoczynać się literą

Zmienne są tworzone poprzez przypisanie im wartości, np.:

```
a = 22;
```

W Matlabie wszystkie zmienne są traktowane jako tablice, w związku z czym zmienna określona pojedynczą wartością (skalar) traktowana jest jako tablica 1x1.
Wartość zmiennej można wyświetlić podając jej nazwę bez zakończenia wyrażenia średnikiem lub korzystając z polecenia `disp`, np.: `disp(x)`.

1.2. Wielkość liter w nazwach

W języku Matlab wielkość liter używanych do tworzenia nazw zmiennych lub funkcji jest rozróżniana (jest to język *case sensitive*). Przykładowo zmienne `promień` i `PROMIEŃ` to dwie różne wielkości. Dlatego bardzo istotne jest zwrócenie uwagi na używanie dokładnie takich nazw jakie zostały zadeklarowane.

1.3. Przestrzeń robocza

Jednym z podstawowych zagadnień w Matlabie jest przestrzeń robocza (*workspace*). Wszystkie zmienne utworzone podczas pracy zostają zapisane w przestrzeni roboczej, aż nie zostaną z niej usunięte. Zmienne można usunąć z przestrzeni roboczej poleceniem `clear`. Listę wszystkich zmiennych zapisanych w przestrzeni roboczej można uzyskać wpisując komendę `who` lub `whos`.

2. Podstawowe operacje w Matlabie

2.1. Typy danych

W Matlabie można wyróżnić 14 podstawowych typów danych. Domyślnym typem jest `double` (double-precision floating point) – typ zmiennoprzecinkowy. Spośród innych typów danych można wyróżnić jeszcze `single` (single-precision floating point) oraz typy całkowite `int`.

2.2. Podstawowe operacje

Obliczanie wartości wprowadzanych wyrażeń odbywa się poprzez wykonywanie operacji arytmetycznych. Można wyróżnić następujące operacje wykonywane na wielkościach skalarnych:

- dodawanie: `a + b`,
- odejmowanie: `a - b`,
- mnożenie: `a * b`,
- dzielenie (prawostronne i lewostronne): `a / b`, `a \ b`,

– potęgowanie: `a ^ b`.

2.3. Kolejność wykonywania operacji

W wyrażeniach wprowadzanych w środowisku Matlab kilka operacji może być podanych jednocześnie, np.: `a * b ^ 3`. Dlatego Matlab ma bardzo ścisłe reguły dotyczące kolejności wykonywania operacji. Kolejność wykonywania działań jest następująca:

Kolejność	Operator
1	operacje w nawiasach
2	potęgowanie
3	mnożenie i dzielenie
4	dodawanie i odejmowanie

W przypadku operatorów o tej samej wadze operacje są wykonywane od lewej do prawej.

3. Wektory i macierze (tablice)

Tablice w Matlabie to listy liczb lub wyrażeń ułożonych poziomo w wierszach i pionowo w kolumnach. Tablicę o tylko jednym wierszu lub tylko jednej kolumnie nazywamy wektorem, natomiast gdy ma ona *m* wierszy i *n* kolumn nazywamy ją macierzą o wymiarach $m \times n$. Korzystając z operacji na tablicach można wykonywać określone operacje na dużych zbiorach danych, co jest jedną z największych zalet Matlaba.

3.1. Deklaracja wektorów i macierzy

Wektory lub macierze można deklarować w następujący sposób:

– podając ich elementy, np.:

```
x = [1 2 3]          %deklaracja wektora wierszowego
x = [1; 2; 3]        %deklaracja wektora kolumnowego
x = [1 2 3; 4 5 6]   %deklaracja macierzy 2×3
```

– podając wartość minimalną i maksymalną oraz krok postępu (opcjonalnie):

```
x = 1:10       %wektor o 10 elementach będących wartościami od 1 do 10 z
domyślnym krokiem 1
x = 1:0.5:10   %wektor o 20 elementach będących wartościami od 1 do 10 z
krokiem 0.5
x = [1:5; 6:10]%macierz 2×5 o elementach będących wartościami od 1 do 10
z domyślnym krokiem 1
```

– korzystając z polecenia `linspace` tworzącego wektor o równomiernie rozmieszczonych elementach:

```
x = linspace(1, 10, 5)     %wektor o 5 równomiernie rozmieszczonych
elementach od 1 do 10
```

3.2. Operacje na wektorach i macierzach

W przypadku korzystania z wektorów i macierzy możliwe jest wykonywanie takich samych operacji jak w przypadku wielkości skalarnych, jednak z uwzględnieniem zasad algebry liniowej, tj.:

– dodawane, odejmowane lub dzielone wektory/macierze muszą mieć takie same wymiary,
– mnożenie macierzy A*B jest możliwe jeśli macierz B ma tyle samo wierszy co macierz A kolumn,
– potęgowanie jest możliwe w przypadku macierzy kwadratowych.

Wymienione wyżej operacje mają charakter macierzowy. Oprócz nich możliwe jest również wykonywanie operacji element po elemencie, takich jak:

– mnożenie: `A .* B`,
– dzielenie: `A ./ B`,

– potęgowanie: `A .^ 2`,

Operacje element po elemencie oznaczane są przez użycie przedrostka w postaci '.', np. `.*`. Kolejną operacją wykonywaną na wektorach i macierzach jest transpozycja, oznaczana symbolem „'", np.: `x'`. Transpozycja pozwala na transformację wektora wierszowego w kolumnowy i odwrotnie, a w przypadku macierzy powoduję zamianę wierszy z kolumnami. Transpozycja ma drugą co do wielkości wagę w kolejności wykonywania operacji (po nawiasach).

3.3. Indeksowanie

Położenie poszczególnych elementów wektora lub macierzy jest określane przez tzw. indeksy. W związku z tym do wartości z określonej pozycji wektora/macierzy można się odwołać podając wartość jej indeksu, np.: `x(3)`. W Matlabie indeksowanie zawsze rozpoczyna się od wartości 1.

4. Instrukcje warunkowe

Instrukcje warunkowe służą do wykonywania określonych operacji (fragmentów kodu programu) w przypadku gdy został spełniony określony warunek. W matalbie dostępne są instrukcje warunkowe `if` oraz `switch`.

4.1. Instrukcja if

Instrukcja `if` jest podstawową instrukcją warunkową i ma postać:

`if` warunek wyrażenie, `end`

Przykład:

```
If x > 0.5 x = x / 2, end
```

Warunek jest pewnym wyrażeniem logicznym, a więc zawierającym jeden z operatorów relacji:

Operator	Znaczenie
<	mniejszy
<=	mniejszy lub równy
==	równy
~=	różny
>	większy
>=	większy lub równy

Konstrukcja `if` może być rozszerzona o wyrażenie wykonywane w przypadku gdy określony warunek nie jest spełniony przy pomocy instrukcji `else`:

```
if warunek
      wyrażenie1
else
      wyrażenie2
end
```

4.2. Instrukcja elseif

Konstrukcja `elseif` pozwala na rozwinięcie instrukcji `if` z uwzględnieniem kolejnego sprawdzania kilku warunków. Warunek określony przy instrukcji `elseif` jest sprawdzany

tylko i wyłącznie wtedy, gdy poprzedzające warunki nie zostały spełnione. Konstrukcja instrukcji warunkowych z wykorzystaniem `elseif` ma postać:

```
if warunek1
     wyrażenie1
elseif warunek2
     wyrażenie2
elseif warunek3
     wyrażenie3
...
else
     wyrażenieN
end
```

Przykład:

```
if x < 0
     disp('Liczba ujemna')
elseif x > 0
     disp('Liczba dodatnia')
else
     disp('Zero')
end
```

4.3. Instrukcja switch

Instrukcja warunkowa `switch` pozwala na wykonanie określonych operacji w zależności od wartości określonej zmiennej lub wyrażenia. Ma ona postać:

```
switch (wyrażenie)
     case wartość1
      wyrażenie1
     case wartość2
      wyrażenie2
...
     otherwise
      wyrażenieN
end
```

Przykład:

```
switch (sign(x))
     case -1
      disp('Liczba ujemna')
     case 1
      disp('Liczba dodatnia')
     otherwise
      disp('Zero')
end
```

5. Pętle

Pętle umożliwiają cykliczne wykonywanie ciągu określonych instrukcji (wyrażeń) aż do momentu zajścia warunku zakończenia pętli. W Matlabie dostępne są dwa rodzaje pętli: `for` i `while`.

5.1. Pętla for

Pętla `for` służy do wykonania określonego ciągu instrukcji założoną uprzednio ilość razy. Liczba powtórzeń wykonania pętli zależna jest od liczby wartości indeksu pętli. Konstrukcja `for` ma postać:

```
for indeks = v
      wyrażenia
end
```

Wielkość v jest wektorem możliwych wartości indeksu. Stąd przykładowa konstrukcja pęli for może mieć postać:

```
for i = 1:0.5:10
      wyrażenia
end
```

lub

```
for i = [1 4 6 7 9]
      wyrażenia
end
```

5.2. Pętla while

Pętla `while` służy do wykonywania określonego ciągu instrukcji tak długo, aż nie zostanie spełniony warunek zakończenia pętli. Konstrukcja `while` ma postać:

```
while warunek_zakończenia
      wyrażenia
end
```

Przykład:

```
i = 0;
while i<10
      i = i + 1;
      disp(i)
end
```