Matlab – Składnia + podstawy programowania

Matlab – Matrix – Laboratory – środowisko stworzone z myślą o osobach rozwiązujących problemy matematyczne, w których operuje się na danych stanowiących wielowymiarowe tabele – macierze, wektory, itp.
Sercem Matlaba jest interpreter poleceń, odpowiedzialny jest za wykonywanie poleceń/skryptów i programów. Kolejną cechą środowiska Matlab jest możliwość kompilacji kodu do postaci wykonywalnej czyli tworzenia tzw. Standlone applications, jak też pełne wsparcie dla środowisk Java VM, a ostatnio również .Net.
Charakterystyczną cechą Matlaba w porównaniu do innych podobnych środowisk jest składnia, zoptymalizowana pod kątem obliczeń macierzowych, tak zapis realizowanych operacji był maksymalnie zwięzły.

Składnia matlaba
Operator:

Przypisania	=
Tworzenia macierzy	[]
Porównania	==
Większości, większy równy	>, >=
Mniejszości, mniejszy równy	<, <=
Mnożenia macierzowego	*
Mnożenia elementowego	.*
Dodawania, odejmowania	+, -
Dzielenia macierzowego	/
Dzielenia elementowego	./
Potęgowania macierzowego	^
Potęgowania elementowego	.^
Transpozycji	‘
Operator zakresu	:

a = 1; przypisanie do zmiennej *a* określonej wartości (macierz jednowymiarowa), rozmiar 1x1
b = [1 2 3 4]; przypisanie do zmiennej *b* wektora (jednowymiarowej tablicy) o elementach 1,2,3,4. Elementy oddzielane spacją lub przecinkiem tworzą wektor poziomy. Rozmiar 1x4
c = [1;2;3;4]; elementy oddzielone średnikiem tworzą wektor pionowy. Rozmiar 4x1
d = [1 2;3 4;5 6;7 8]; tworzona jest zmienna *d* będąca macierzą dwuwymiarową o rozmiarze 4x2 (cztery wiersze, dwie kolumny)
Sprawdź wynik działa operacji:
e = [1;2 3;3 4 5];

b(2) = 10; Przypisanie drugiemu elementowi wektora b wartości 3
d(3,2) = 7; Przypisanie do elementu znajdującego się w 3 wierszu i 2 kolumnie macierzy *d* wartości 7

Ponieważ Matlab posiada możliwość wektoryzacji macierzy możliwy jest również dostęp do określonego elementu macierzy poprzez zapis
d(7) = 11;
Matlab stosuje zapis kolumnowy! Czyli elementy d(i,j) zapisywane są wg zależności d(J*(j-1)+i), przy założeniu że d ma J wierszy i K kolumn d

a == 2 w wyniku zwraca wartość 1 jeśli prawda lub 0 jeśli fałsz.

c == 2 w wyniku zwraca macierz o rozmiarze c zawierającą wartości 0 lub 1 w zależności czy dany element macierzy c był równy stałej Wynik=[0;1;0;0]

Podobne zależności występują dla operatorów >,>=,<,<=
Operatory arytmetyczne *,.*,/,./
Operator * oznacza mnożenie macierzy

```
b = [1 2 3 4];
c = [1;2;3;4];
b*c

ans =

   30
```

ale

```
>> c*b

ans =

    1    2    3    4
    2    4    6    8
    3    6    9   12
    4    8   12   16
```

MNOŻENIE MACIERZY NIE JEST PRZEMIENNE
Uwaga: Jeśli wykonywana jest dowolna operacja arytmetyczna i nie tylko bez przypisania wyniku tzn bez zapisu g = b*c; (samo b*c) wówczas wynik zapisywany jest do zmiennej ans

Podczas gdy operator mnożenia elementowego .* oznacza pomnóż element przez element

```
>> c.*b'

ans =

    1
    4
    9
   16
```

Zwróć uwagę na operator ‘ (transpozycji macierzy/wektora) Aby mogła zostać zrealizowana operacja .* rozmiar macierzy/wektorów obydwu zmiennych musi być identyczny. Powyższy zapis oznacza
Dla każdego *i* dokonaj *c(i).*b(i)* Wynik operacji jest tego samego rozmiaru co *c* i *b*
Podobne zależności zachodzą pomiędzy operatorami /,./,^,.^

Operator zakresu „:”
Podstawową cechą jak i atutem języka matlab jest operator zakresu „:”. „:” oznacza pewien zakres wartości. Np. zapis d(:,1) oznacza weź pierwszą kolumnę z macierzy d. d(2,:) oznacza weź drugi wiersz z macierzy d.
e = d(:,2) spowoduje utworzenie wektora *e* który będzie się składał z elementów e=[2;4;6;8]
f = d(3,:) utworzenie wektora *f* o elementach *f*=[5 6]

Operator zakresu można też używać do określenia przedziału
g = 2 : 5; oznacza to że zmienna *g* przyjmuje kolejne wartości g = [2 3 4 5]
h = 4 : 0.1 : 5; oznacza h = [4 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0]
Jeśli chcemy by kierunek przedziału był malejący to jedyną możliwą postacią jest:
i = 6 : -1 : 2; co oznacza i = [6 5 4 3 2]

Składnia języka
Instrukcja warunkowa

```
if warunek
        ciąg poleceń

end;

if warunek
        ciąg poleceń
else
        ciąg poleceń
end;

if warunek1
        ciąg poleceń
elseif warunek2
        ciąg poleceń
elseif warunek3
        ciąg poleceń
end;

switch zmienna
        case wartość
                ciąg poleceń
        case wartość
                ciąg poleceń
end;
```

Przy czym zmienna może być dowolnego typu zarówno string jak i liczba

Pętle
W matlabie występują dwa typy pętli – for oraz Chile

```
for zmienna=przedział wartości
        ciąg instrukcji
end;

np.
for i=1:10
        x(i) = i^2;
end;

while warunek
        ciąg instrukcji
end;
```

np.

```
i=1;
while i<=10
        x(i) = i^2;
        i = i+1;
end;
```

Przydatne polecenia
a = load('nazwa.pliku'); - wczytanie zawartości pliku
save('nazwa.pliku','zmienna'); - zapisanie zmiennej *zmienna* do pliku. Uwaga *zmienna* jest podawana jako napis! Jako nazwa zmiennej
s=sort(a); - funkcja zwraca posortowane wartości podane w postaci wektora *a* wynik zapisywaney jest do zmiennej *s*. Jeśli *a* jest macierzą, funkcja sortuje każdą z kolumn niezależnie
[s,o]=sort(a); - funkcja robi to co powyżej dodatkowo zwracając zmienną *o* która zawiera indeksy posortowanych wartości, tak że: (gdy a jest wektorem) s=a(o);
plot(x,y); - rysuje wykres 2D, jeżeli *x* jest macierzą, rysuje kilka wykresów, osobno dla każdej kolumny macierzy *x*. Uwaga *x* oraz *y* muszą mieć ten sam rozmiar
plot(x,y,'ab'); tworzy wykres jak wyżej ale z odpowiedniemu właściwościami gdzie *a* może przyjmować wartości [r g b c m y k] odpowiadające kolorom wykresu, zaś *b* odpowiada postaci wykresu:

-	Ciągły
:	Linia punktowa
.-	Linia kropka punkt
.	Wykres punktowy
S	Punktowy, ale punkty są reprezentowane jako kwadraty
^	Punktowy, ale punkty są reprezentowane jako trójkąty
>	Punktowy, ale punkty są reprezentowane jako trójkąty zwrócone w prawo
<	Punktowy, ale punkty są reprezentowane jako trójkąty zwrócone w lewo

hold on/off; - wywołanie powoduje *hold on* powoduje że podczas tworzenia nowego wykresu stara zawartość wykresu nie będzie usuwana. Domyślnie jest *hold off;*
figure(i); tworzy obiekt graficzny o numerze *i* na którym będzie rysowany dany wykres.
z = sum(x); sumuje wartości *x*, jeśli *x* jest macierzą o wymiarach (n x m) sumowanie następuje w kolumnach, tak iż po realizacji uzyskujemy wynik w postaci wektora o rozmiarze (1 x m)
z = sum(x,dim); Podobnie jak wyżej z tą różnicą iż możemy wskazać wymiar macierzy po którym ma nastąpić sumowanie.
mx = max(x); funkcja znajduje maksymalną wartość *x*, jeżeli *x* jest macierzą (n x m) funkcja *max* zwraca wektor wartości maksymalnych liczony osobno dla każdej kolumny
mi = min(x); analogicznie jak wyżej dla szukania minimum.
si = size(x); funkcja zwracająca rozmiar tablicy *x*. Domyślnie wartość zwracana przez funkcję jest dwuargumentowa zwracając ilość wierszy i ilość kolumn tablicy *x*. si(1) – liczba wierszy, si(2) – liczba kolumn
si = size(x,n); podobnie jak wyżej z tą różnicą iż zmienna *n* określa wymiar który jest zwracany. Odpowiada to zależności si = size(x); si = si(n);
fi = find(warunek); polecenie zwraca listę indeksów tabeli, które spełniają określony warunek np. x = [-1 1 2 -3 4 -2]; fi = find(x<0); wynikiem powyższego programu będzie fi = [1 4 6]

Zadania:

1. Stwórz nowy skrypt o nazwie lab1.m
2. Narysuj wykres funkcji $y=x^2$ dla x z przedziału -5 do 10 z krokiem 0.01
3. Znajdź największą i najmniejszą wartość zmiennej y z zadania 2 (wykorzystaj odpowiednie funkcje opisane powyżej)
4. Na nowym rysunku narysuj wykres funkcji y1=sin(x) oraz y2=cos(x) dla x z przedział -2pi do 2pi z krokiem 0.01. Jako pi możesz użyć predefiniowanej stałej o nazwie pi (- zmienna o nazwie pi=3.141592). Spraw aby obydwie krzywe były narysowane na jednym wykresie
5. Stwórz zmienną yy składającą się z dwóch kolumn odpowiednio y1, y2
6. Znajdź wartość średnią dla obydwu kolumn zmiennej yy (zadanie rozwiąż z wykorzystaniem pętli i wypisz wynik na ekranie)
7. Powtórz zadanie 6 wykorzystując odpowiednią funkcję
8. Wczytaj dane z pliku iris.data (plik dostępny pod adresem: http://mblachnik.pl/doku.php/dydaktyka/zajecia/ai_io/materialy) i dokonaj jego normalizacji (normalizacja oznacza sprowadzenie wartości każdej z kolumn do przedziału 0 – 1 tj. tak aby największa liczba w danej kolumnie była równa 1, a najmniejsza 0. Pozostałe liczby powinny przyjmować proporcjonalnie kolejne wartości). Innymi słowy normalizacja powoduje że wartości danej zmiennej powinny mieścić się w przedziale od 0 do 1. Aby to zrealizować posłuż się zależnością
   $$y = \frac{x - \min(x)}{\max(x) - \min(x)}$$
   Gdzie:
   - x to wartości pojedynczej kolumny którą chcemy znormalizować,
   - min(x) to minimalna wartość kolumny x,
   - max(x) to maksymalna wartość kolumny x

   Do realizacji tego zadania możesz wykorzystać funkcje wykorzystane w poprzednich zadaniach np. min, max, mean
   Pamiętaj że zbiór Iris.data składa się z 5 kolumn i należy dokonać normalizacji każdej z nich
9. Programik realizujący normalizację zapisz w postaci osobnej funkcji i nazwij ją „normalizacja”
10. Po normalizacji narysuj każdą z kolumn tak iż na osi x są numery poszczególnych wektorów, a na osi y ich wartości.
11. Narysuj wykres przedstawiający wymiar 3 i 4 (3 i 4 to numery kolumn) tak iż każdy wiersz (wektor) reprezentował będzie pojedynczy punkt na wykresie, przy czym wiersze(wektory) z różnych gatunków narysuj innym kolorem i zaznacz innym symbolem.
12. Stwórz funkcję która odnajduje max i min dla każdej z kolumn, a następnie zwraca wartość środkową, tj. (max+min)/2
13. Napisz algorytm sortowania liczb i posortuj każdą z kolumn tak, aby miała wartości poukładane w kolejności od największej do najmniejszej
    Całość zaimplementuj w oparciu o algorytm sortowania bąbelkowego https://pl.wikipedia.org/wiki/Sortowanie_b%C4%85belkowe