Politechnika Wrocławska, Wydział Informatyki i Zarządzania

Podstawy środowiska MATLAB

1 Organizacja pracy z MATLAB-em

a)

```
>> zmienna = wyrażenie
```

na przykład

```
>> a = 2
```

b)

```
>> zmienna = wyrażenie;
```

na przykład

```
>> a = 2;
```

```
>> napis = 'Nie ma takiego numeru'
```

W MATLAB-ie pierwsze użycie zmiennej jest również jej definicją.

c)

```
>> clc
```

```
>> help abs
```

d)

```
>> pwd

>> dir

>> ls

>> cd ..

>> cd nazwa_katalogu
```

2 Struktury danych

Zmienna w MATLABie traktowana jest domyślnie jako macierz.

a)

```
>> A = [1 2 1; 4 5 6; -1 0 1]

>> size(A)
```

b)

```
>> y = [1 2 3 4]

>> length(y)

>> size(y)
```

c)

```
>> A = zeros(2,3)

>> A = ones(4,1)

>> A = eye(4,3)

>> A = rand(2,8)
```

— rozkład równomierny na przedziale $[0, 1]$

```
>> A = randn(2,8)
```

— rozkład normalny $N(0, 1)$

d) Ważne, często używane:

```
>> i = 1:4
```

```
>> i = 1:0.5:4
```

e)

```
>> pi
```

```
>> omega = -2*pi:0.1:2*pi
```

f)

```
>> who
```

— lista zmiennych znajdujących się w pamięci

```
>> clear nazwa_zmiennej
```

— czyszczenie zmiennej z pamięci

```
>> clear all
```

3 Operacje na macierzach

a)

```
>> A = [1 -1 4; 2 -1 0; 0 0 1]
```

```
>> A(2,3)
```

— odczytanie elementu macierzy

```
>> A(2,3) = 5
```

— przypisanie wartości elementowi macierzy

b)

```
>> A(:,1)
```

— pierwsza kolumna macierzy

```
>> A(2,:)
```

— drugi wiersz macierzy

```
>> A(2,:) = [2 -1 -5]
```

```
>> A(1,2:3)
```

— elementy od 2 do 3 z pierwszego wiersza macierzy

```
>> A(:,2:3)
```

```
>> A(:)
```

c)

```
>> det(A)
```

— wyznacznik macierzy

```
>> rank(A)
```

— rząd macierzy

```
>> eig(A)
```

— wartości własne macierzy

```
>> A'
```

— transpozycja macierzy

```
>> A = A'
```

```
>> inv(A)
```

— macierz odwrotna

d) Czym różnią się poniższe dwa polecenia?

```
>> [1 2 3 4]' * [1 2 3 4]
>> [1 2 3 4] .* [1 2 3 4]
```

Porównaj z następującym:

```
>> [1 2 3 4].^2
```

e)

```
>> B = [-1 2 1; 3 1 -2; 4 2 -1]
```

```
>> A + B
```

```
>> A * B
```

```
>> A .* B
```

```
>> A' * inv(B)
```

4 Elementy statystyki

a)

```
>> x = [5:-1:1]
>> mean(x)
```

— średnia arytmetyczna

```
>> median(x)
```

— mediana

```
>> std(x)
```

— odchylenie standardowe

```
>> sum(x)
```

— suma elementów

```
>> prod(x)
```

— iloczyn elementów

```
>> cumsum(x)
```

— suma kumulatywna

```
>> cumprod(x)
```

— iloczyn kumulatywny

```
>> min(x)
```

— wartość najmniejszego elementu z `x`.

```
>> max(x)
```

— wartość największego elementu z `x`.

```
>>[wartosc indeks]=min(x)
```

— zwraca również informację o położeniu w `x`.

```
>> [wartosc indeks] = max(x)
```

b)

```
>> sort(x)
```

```
>> diff(x)
```

— różnice par kolejnych elementów

c)

```
>> x = randn(1,1000);
```

```
>> hist(x)
```

— rysuje histogram

d)

```
>> roots([1 2 -1])
```

-– pierwiastki wielomianu $x^2 - 1 = x^2 + 2x - 1$

5 Wykresy jednowymiarowe

a)

```
>> x = -4*pi : 0.1 : 4*pi;
>> y = sin(x);
>> plot(x,y)
```

```
>> plot(y)
```

b) Dwa wykresy na jednym obrazku:

```
>> z = cos(x).^2;
```

Pierwszy sposób:

```
>> plot(x,y,x,z)
```

Drugi sposób:

```
>> plot(y)
>> hold on
>> plot(z)
```

Trzeci sposób (różne kolory):

```
>> plot(y)
>> hold on
>> plot(z,'r')
```

c)

```
>> z = randn(2,1000);
>> plot(z(1,:), z(2,:), 'm*')
```

— oś X to `z(1,:)`, oś Y to `z(2,:)`.

6 Wykresy dwuwymiarowe

a)

```
>> [x,y] = meshgrid(-2*pi:0.2:2*pi, -2*pi:0.2:2*pi);
>> z = sin(x).*sin(y);
>> surf(x,y,z)
>> colormap jet(32)
>> alpha(0.5)
```

b)

```
>> surf(x,y,z)
>> imagesc(z)
>> contour(z)
>> imshow(z)
```

c) Narysuj wykres funkcji: $z = \dfrac{\sin(x^2 + y^2)}{1 + x^2 + y^2}$

7 Funkcje (M-pliki)

W pliku o nazwie `test.m` zapisz kod funkcji, która dla zadanego argumentu `x` zwraca wartość o `1` większą:

```
% to jest komentarz
function o1wieksza = test(x)
    o1wieksza = x + 1;
end
```

Uruchom program następującymi poleceniami:

```
>> test(1)
>> test([2 -3])
```

8 Instrukcje sterujące

a) Instrukcja warunkowa `if` :

```
if wyrażenie
   polecenia
elseif wyrażenie
   polecenia
elseif wyrażenie
   polecenia
   .........
else
   polecenia
end
```

Przykład: Funkcja sprawdzająca, czy podaną macierz można odwrócić:

```
function odwracalna(x)
   if det(x) ~= 0
      disp('tak');
   else
      disp('nie');
   end
end
```

b) Pętla `while`:

```
while wyrażenie    % { dopóki wyrażenie ma niezerową wartość, pętla się wykonuje
   polecenia
end
```

c) Pętla `for`:

```
for licznik = zakres
   polecenia
end
```

Przykład. Pojedyncza pętla `for`:

```
for i = 1:4
   i
end
```

Przykład. Podwójna pętla `for`:

```
for i = 3:5
   i
   for j = 1:4
      j
   end
end
```