



Politechnika  
Wroclawska

# Matlab step by step

Wojciech Myszka

2024-10-10



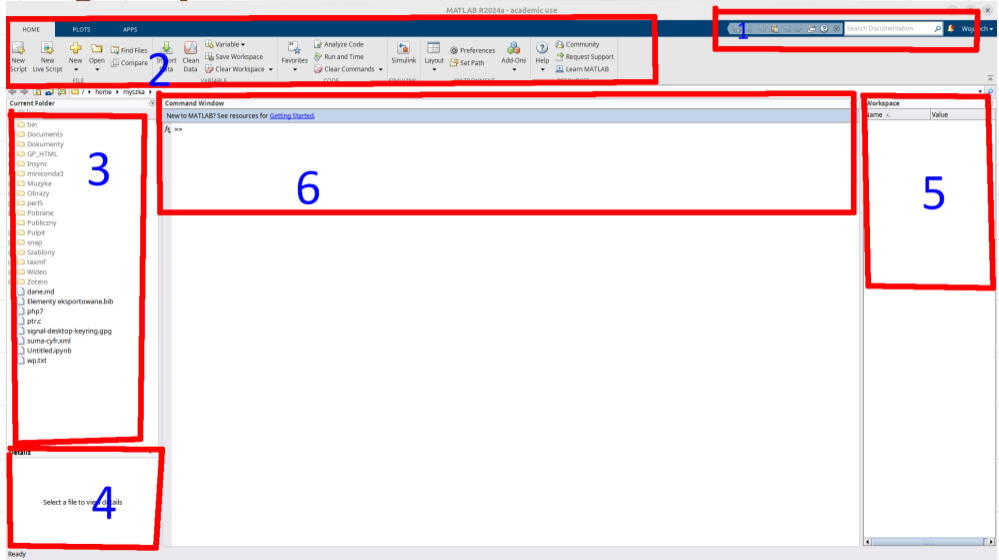
HR EXCELLENCE IN RESEARCH

- 1 Widok ogólny
- 2 Zmienne
- 3 Rozwiązywanie równań liniowych
- 4 Edytor



# Widok ogólny

# Ekran główny



Rysunek 1: image-20241010081637188

# Omówienie

1. Help
2. Menu główne (składa się z trzech „warstw”)
  - 2.1 Menu główne związane z edycją, uruchamianiem, zapisywaniem kodu,...
  - 2.2 Menu wykresów pozwalające zmieniać ich wygląd
  - 2.3 Menu pozwalające doinstalowywać dodatkowe pakiety
3. Zawartość bieżącej kartoteki
4. Informacje o wybranym elemencie z kartoteki
5. Obszar zmiennych pozwalający na ich podglądanie i zmianę wartości
6. Obszar roboczy, gdzie wpisujemy polecenia



# Polecenia MATLABa I

- ▶ Najprostszy tryb, to „tryb kalkulatora” — pozwala w trybie natychmiastowym wykonywać operacje.
- ▶ Po wpisaniu polecenie i naciśnięciu klawisza Enter pojawia się wynik, albo informacja o błędzie
- ▶ Znak % to znak komentarze — wszystko co po nim występuje jest ignorowane do końca linii (podobnie jak w C++ //)
- ▶ Przed naciśnięciem klawisza Enter można polecenie edytować używając klawiszy ←, →, Del oraz dopisując treść

## Polecenia MATLABa II

- ▶ Po naciśnięciu klawisza Enter polecenie jest wykonywane.

Edytor daje dostęp do wcześniej wykonanych poleceń po naciśnięciu klawisza ↑; historię poleceń można przeglądać: ↑ przechodzi do poleceń wydanych wcześniej, a ↓ — do późniejszych.

Polecenie wyświetla się na samym dole i można je zmodyfikować zaczynając edycję (strzałki w lewo przenoszą na wybraną pozycję; można znaki wstawiać i kasować); klawisz Enter przekazuje polecenie do wykonania.

- ▶ Wynik polecenia wypisywany jest na ekranie i zapisywany do zmiennej `ans`.



## Polecenia MATLABa III

- ▶ W obliczeniach można korzystać ze zmiennych (trzeba wcześniej nadać im jakąś wartość);

Wyniki obliczeń można zapisywać w zmiennych (do późniejszego wykorzystania)

```
x=2+2
```

wynik zostanie przypisany do zmiennej  $x$  i wyświetlony na ekranie

- ▶ Jeżeli na końcu polecenia umieścić średnik — wynik obliczeń nie pojawi się na ekranie.



# Operatory I

- ▶ + (suma)
- ▶ - (różnica)
- ▶ \* (iloczyn)
- ▶ / (iloraz)
- ▶ \ „lewy” iloraz

```
>> 1/2
```

```
ans =
```

```
0.5000
```

```
>> 1\2
```

```
ans =
```

```
2
```

- ▶ ^ (potęga)

# Operatory II

Lista operatorów jest ciut szersza, zajmiemy się tym później.

## Operatory jako funkcje

Każdy z operatorów ma swój **funkcyjny** odpowiednik; zamiast pisać  $a + b$  można napisać `plus(a, b)`

# Funkcje I

W obliczeniach można korzystać z bardzo szerokiego zestawu funkcji:

- ▶ wykładnicze i logarytmiczne

$\exp(x)$ ,  $\log(x)$ ,  $\log_{10}(x)$ ,  $\text{sqrt}(x)$

- ▶ trygonometryczne i ich odwrotności

$\cos(x)$ ,  $\cot(x)$ ,  $\sin(x)$ ,  $\tan(x)$ ,...

- ▶ hiperboliczne i ich odwrotności

$\cosh(x)$ ,  $\sinh(x)$ ,  $\tanh(x)$ ,  $\coth(x)$ ,...

- ▶ liczb zespolonych

$\text{abs}(x)$ ,  $\text{angle}(x)$ ,  $\text{conj}(x)$ ,  $\text{imag}(x)$ ,  $\text{real}(x)$

# Funkcje II

- ▶ numeryczne
  - ▶ `ceil` — zaokrąglenie do najbliższej całkowitej w kierunku  $\infty$
  - ▶ `fix` — zaokrąglenie w kierunku zera
  - ▶ `floor` — zaokrąglenie w kierunku  $-\infty$
  - ▶ `sign` — funkcja signum

Nie sposób wymienić wszystkich!



# Zmienne



# Typ zespolony I

- ▶ O typach danych już wspominałem, nie mówiąc nic o liczbach zespolonych.
- ▶ Są naturalną częścią MATLABa

```
>> a=sqrt(-1)
```

```
a =
```

```
0.0000 + 1.0000i
```

```
>> a^2
```

```
ans =
```

```
-1
```

```
>> class(a)
```

```
ans =
```

```
'double'
```

Symbolem  $i$  (lub  $j$ ) oznacza się jednostkę zespoloną

## Typ zespolony II

```
>> (1*j)^2 % ta jedynka nie jest potrzebna  
ans =  
    -1
```

Funkcja `complex(a,b)` tworzy liczbę zespoloną o wartości  $a + ib$ .



# Zmienne proste I

Po „zadeklarowaniu” zmiennej poleceniem

```
a = 1;
```

a staje się zmienną skalarną;



## Odczyt wartości zmiennej

1. Sprawdzamy w *workspace*
2. wypisujemy nazwę zmiennej w linii:

```
>> a
```

```
a =
```

```
1
```

3. Używamy polecenia `display()`

```
>> display(a)
```

```
a =
```

```
1
```

## Format wyświetlania

Używany jest format default; do dyspozycji mamy:

`short, long, shortE, longE, shortG, longG, shortEng, longEng, bank, hex, rational, +`

W większości przypadków format `hex` nie będzie specjalnie przydatny.

Przygotowałem demonstrację, która pozwoli wizualizować sobie każdy z tych formatów.



# Zmienne złożone I

- ▶ Nie ma czegoś takiego jak deklaracja tablicy.
- ▶ Nie ma wielkiej różnicy między zmienną skalarną, a tablicą:

```
>> a = 1
```

```
a =
```

```
1
```

```
>> display(a);
```

```
a =
```

```
1
```

```
>> display(a(1))
```

```
1
```

```
>> display(a(1,1))
```

```
1
```



## Zmienne złożone II

```
>> display(a(1,1,1))  
1
```

► Tablica jednowymiarowa

```
>> a=[1 2 3]
```

```
a =
```

```
1 2 3
```

```
>> a=[1,2,3]
```

```
a =
```

```
1 2 3
```

```
>> a=[1;2;3]
```

```
a =
```

```
1
```

```
2
```

# Zmienne złożone III

3

- ▶ Tablica dwuwymiarowa

```
>> a=[1 2 3; 4 5 6; 7 8 9]
```

```
a =
```

```
1     2     3
4     5     6
7     8     9
```



## Rozmiar tablicy

- ▶ Ustalany jest na podstawie danych
- ▶ W szczególności zadziała coś takiego:

```
>> a(1)=1
```

```
a =
```

```
1
```

```
>> a(2)=2
```

```
a =
```

```
1
```

```
2
```

```
>> a(5)=5
```

```
a =
```

```
1
```

```
2
```

```
0
```

```
0
```

```
5
```

- ▶ Jeżeli jednak znamy rozmiar danych lepiej zainicjować tablicę raz, na maksymalny rozmiar, używając jednego z niżej opisanych poleceń

# Tworzenie tablic z „jakąś” zawartością I

- ▶ `ones` — Tablica wypełniona jedynekami
  - ▶ `ones(n)` — kwadratowa o rozmiarze  $n \times n$
  - ▶ `ones(n, m)` — prostokątna o rozmiarze  $n \times m$
  - ▶ `ones(n, m, k)` — o trzech wymiarach  $n \times m \times k$

```
>> ones(2,2,2)
```

```
ans(:,:,1) =
```

```
    1    1
```

```
    1    1
```

```
ans(:,:,2) =
```

```
    1    1
```

```
    1    1
```

- ▶ `eye(n, m)` — tworzy tablicę o rozmiarach  $n \times m$  z jedynekami na „głównej” przekątnej i zerami na pozostałych pozycjach



## Tworzenie tablic z „jakąś” zawartością II

```
>> eye
```

```
ans =
```

```
1
```

```
>> eye(2)
```

```
ans =
```

```
1 0
```

```
0 1
```

```
>> eye(3,4)
```

```
ans =
```

```
1 0 0 0
```

```
0 1 0 0
```

```
0 0 1 0
```

```
>> eye(4,3)
```

```
ans =
```





## Tworzenie tablic z „jakąś” zawartością III

1	0	0
0	1	0
0	0	1
0	0	0

- ▶ `zeros(n,m,k)` — tworzy tablicę o zadanych rozmiarach wypełnioną zerami
- ▶ `rand(n,m,k)` — tworzy tablicę o zadanych rozmiarach wypełnioną liczbami losowymi o rozkładzie jednostajnym z zakresu (0, 1)
- ▶ `randn` — generuje liczby losowe o rozkładzie normalnym

Funkcja `rng()` pozwala zdefiniować parametry startowe generatora liczb pseudolosowych.

- ▶ `size()` informuje o rozmiarach tablicy



# Tworzenie tablic z „jakąś” zawartością IV

```
>> a=rand(2,2,2,2);
```

```
>> size(a)
```

```
ans =
```

```
     2     2     2     2
```



# Tworzenie tablic ze zmiennych I

- ▶ Jeżeli mamy kilka zmiennych o nazwach `var1`, `var2`, ... `varN` o tej **samej liczbie wierszy** można z nich utworzyć tablicę poleceniem `table`

```
T=table(var1,...,varN)
```

- ▶ Można też prealokować pamięć poleceniem

```
T = table('Size',sz,'VariableTypes',varTypes)
```

- ▶ `sz` to dwuelementowa tablica definiująca liczbę wierszy (`sz(1)`) i liczbę zmiennych (`sz(2)`)
- ▶ `varTypes` to tablica definiująca typy zmiennych



## Tworzenie tablic ze zmiennych II

```
>> T = table('Size',[50 3],...  
'VariableTypes',{'string','double','datetime'})
```

```
T =  
50x3 table  
      Var1      Var2      Var3  
-----  
<missing>      0      NaT  
<missing>      0      NaT  
<missing>      0      NaT
```

- W tak utworzonej tablicy każda z kolumn ma swoją nazwę i do tych danych możemy odwoływać się na różne sposoby



## Tworzenie tablic ze zmiennych III

T.Var1        % całą kolumna pierwsza tablicy

T.Var2(3)    % trzeci element w drugiej kolumnie

T(3,2)        % to samo co wyżej

- ▶ Tworząc tablicę poleceniem `table` można poszczególnym „kolumnom” nadać nazwy; można to też zrobić już po utworzeniu tablicy

```
T = table(lat,lon,'VariableNames',["Latitude","Longitude"])
```

# Operacje na tablicach

Obowiązuje „mieszanka” algebry liniowej i MATLABa.

- ▶ można dodawać tablice o tych samych rozmiarach
- ▶ można mnożyć tablice o **odpowiednich** rozmiarach
- ▶ mnożenie tablicy przez skalar — standardowo
- ▶ argumenty wszystkich (standardowych) funkcji mogą być tablicami.



# „Uogólnione” operacje na tablicach I

Popatrzmy na coś takiego:

```
>> a=[1 2; 3 4]
```

```
a =
```

```
    1    2  
    3    4
```

Co znaczy  $a^2$ ?

Sprawdźmy

```
>> a^2
```

```
ans =
```

```
    7    10  
   15    22
```



## „Uogólnione” operacje na tablicach II

Najprawdopodobniej jest to wynik operacji  $a * a$  — macierzowe mnożenie.

$b =$

3      4

Chcielibyśmy **każdy** z elementów tablicy podnieść do kwadratu.

$b^2$  nie *zadziała* bo nie da się macierzy  $b$  pomnożyć przez macierz  $b$  — nie zgadzają się rozmiary.

Trzeba wymyślić dodatkowe operatory macierzowe

- ▶  $\wedge$  to potęgowanie macierzy
- ▶  $\cdot^{\wedge}$  to potęgowanie elementów macierzy
- ▶  $/$  to operator „dzielenia” ( $A/B$  to prawie to samo, co  $A * B^{-1}$ )





## „Uogólnione” operacje na tablicach III

Po dodaniu kropki do operatora wiele z nich traci znane z algebry znaczenie i wykonuje operację na **elementach tablicy**.

### „Kompatybilne” rozmiary

Przy omawianiu „operatorów z kropką” pojawia się pojęcie **kompatybilności** rozmiarów gdy po obu stronach operatora są tablice

$$a = [a_1 a_2] \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b^3 \end{bmatrix} \quad a. \wedge b = \begin{bmatrix} a_1^{b_1} & a_2^{b_1} \\ a_1^{b_2} & a_2^{b_2} \\ a_1^{b_3} & a_2^{b_2} \end{bmatrix}$$

Dosyć to dziwne. Odsyłam do dokumentacji.

Operatory z kropką to `.*`, `./`, `.\`, `.^`, `.'`



## „Uogólnione” operacje na tablicach IV

Ten ostatni symbol to symbol transpozycji macierzy. W przypadku gdy macierz jest zespolona oprócz transpozycji dokonywane jest operacja sprzężenia (zmiany znaku części urojonej)

```
>> a=[sqrt(-1) sqrt(-3)]
```

```
a =
```

```
0.0000 + 1.0000i    0.0000 + 1.7321i
```

```
>> a'
```

```
ans =
```

```
0.0000 - 1.0000i
```

```
0.0000 - 1.7321i
```

```
>> a*a'
```

```
ans =
```

```
4.0000
```



## „Uogólnione” operacje na tablicach V

```
>> a.'  
ans =  
    0.0000 + 1.0000i  
    0.0000 + 1.7321i
```

W przypadku użycia operatora `.'` wykonywana jest **tylko** transpozycja.

# Dostęp do wiersza/kolumny macierzy I

Bardzo często trzeba z macierzy wyciągnąć jeden wiersz lub kolumnę.  
Pomaga w tym operator : użyty zamiast numeru kolumny/wiersza:

```
>> a=[1 2 3; 4 5 6; 7 8 9]
```

```
a =
```

```
     1     2     3  
     4     5     6  
     7     8     9
```

```
>> a(:,1)
```

```
ans =
```

```
     1  
     4
```



## Dostęp do wiersza/kolumny macierzy II

```
    7
>> a(2, :)
ans =
    4    5    6
```

A jak z tablicy wyciągnąć pierwszy i trzeci wiersz?

```
>> a([1 3], :)
ans =
    1    2    3
    7    8    9
```

Można też tak:



## Dostęp do wiersza/kolumny macierzy III

```
>> a(:, [3 2])
```

```
ans =
```

```
3 2
```

```
6 5
```

```
9 8
```

zmieniliśmy kolejność kolumn.



# Rozwiązywanie równań liniowych

1. Popatrzmy na takie równanie:

$$7 \cdot x = 21$$

Rozwiązanie jest proste: dzielimy obie strony przez 7 i dostajemy

$$x = \frac{21}{7} = 3$$



## Ogólnie II

2. Aby rozwiązać równanie postaci

$$Ax = b$$

postępujemy analogicznie (mnożymy lewostronnie przez  $A^{-1}$ )

$$A^{-1}Ax = A^{-1}b$$

$$x = A^{-1}b$$

3. Funkcja `inv()` służy do wyliczania macierzy odwrotnej.
4. Funkcja `det()` wyznacza wartość wyznacznika macierzy.

## Ogólnie III

5. Przypominam nieoczywisty operator „lewego” dzielenia  $\backslash$ :

```
>> 2\1  
ans =  
    0.5000
```

Można go zastosować i do tablic

$$A \backslash b$$

Żeby dostać rozwiązanie równania  $Ax = b$ .

Dla równania  $7 \cdot x = 21$  będzie to tak

## Ogólnie IV

```
>> x=7\21
```

```
x =
```

```
3
```

6. Żeby rozwiązać równanie  $xA = b$  użyjemy „zwykłego” operatora dzielenia:  $x = b/A$ , czyli dla równania  $x \cdot 7 = 21$  będzie to tak

```
>> x=21/7
```

```
x =
```

```
3
```

Co w przypadku skalarnym wydaje się być *masłem maślanym*.

7. „Lewe” dzielenie ( $\backslash$ ) można zastąpić wywołaniem funkcji `mldivide()`, a prawe — `mrdivide()`.

## Uwaga

Dokumentacja MATLABa podpowiada, żeby podczas rozwiązywania układów równań liniowych nie korzystać z funkcji `inv()` gdyż operatory dzielenia używają zoptymalizowanych algorytmów znakomicie sprawdzających się w tych właśnie przypadkach.



# Wyznacznik różny od zera I

```
>> A=pascal(3);
```

```
>> b=[3;1;4];
```

```
>> x=A\b
```

```
x =
```

```
    10
```

```
   -12
```

```
     5
```

```
>> A*x-b
```

```
ans =
```

```
     0
```

```
     0
```

```
     0
```

## Wyznacznik różny od zera II

(`pascal()` to funkcja generująca **trójkąt Pascala**)

Jeżeli teraz `b` będzie macierzą kwadratową współczynników, czyli mamy zestaw układów równa o tej samej macierzy

```
>> b=magic(3);
```

```
>> X=A\b
```

```
X =
```

```
    19    -3    -1  
   -17     4    13  
     6     0    -6
```

```
>> A*X-b
```

```
ans =
```

```
     0     0     0
```

# Wyznacznik różny od zera III

$$\begin{vmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$



# Macierz osobliwa I

- ▶ Sytuacja jest trudniejsza i albo rozwiązanie nie istnieje albo jest niejednoznaczne.
- ▶ Funkcja `rank()` pozwala sprawdzić jaki jest rząd macierzy

```
>> A=[1 3 7; -1 4 4;1 10 18]
```

```
A =
```

```
     1     3     7
    -1     4     4
     1    10    18
```

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

```
ans =
```

```
     2
```





## Macierz osobliwa II

- Możemy użyć funkcji `pinv()` aby wyliczyć macierz pseudo-odwrotną. Funkcja ta tworzy taką macierz aby albo znaleźć **jakieś** rozwiązanie dokładne, albo rozwiązanie najlepsze w sensie średniokwadratowym

```
>> b=[5;2;12];
```

```
>> x=pinv(A)*b
```

```
x =
```

```
    0.3850
```

```
   -0.1103
```

```
    0.7066
```

```
>> A*x-b
```

```
ans =
```

```
    1.0e-14 *  
    0.0888
```



## Macierz osobliwa III

-0.7550

-0.5329

W przypadku gdy nie istnieje rozwiązanie dokładne:

```
>> b=[3;6;0];
```

```
>> A*pinv(A)*b
```

```
ans =
```

-1.0000

4.0000

2.0000



# System nadokreślony

Czyli więcej równań niż niewiadomych

Demonstracja!

# System niedookreślony

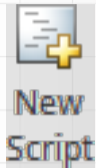
Czyli mniej równań niż niewiadomych

W takim przypadku nie ma jednoznacznego rozwiązania



Edytor

# Nowy skrypt I



- ▶ Najprostsza metoda pracy to wpisywanie poleceń i oglądanie wyników.
- ▶ Gdy obliczenia polegają na wykonaniu kilku poleceń — można je wpisać do pliku tworząc „nowe polecenie”
- ▶ MATLAB wyposażony jest w edytor ułatwiający tworzenie i modyfikowanie takich plików z poleceniami (zwanymi tradycyjnie **skryptami**)
- ▶ Edytor pozwala na pracę z wieloma plikami, ma też możliwość edytowania jednego pliku w dwu oknach/miejscach



## Nowy skrypt II

- ▶ Dodatkowo ma możliwość odpluskwiania programów (debugowania — będzie o tym mowa później)
- ▶ Polecenia zapisuje się do pliku (o rozszerzeniu `.m`) i mogą być one uruchamiane na dwa sposoby:
  - ▶ naciskając klawisz Run
  - ▶ wpisując nazwę pliku jako polecenie MATLABa
- ▶ Zapisując plik MATLAB może ostrzegać, że nazwa pliku pokrywa się z nazwą jakiejś jego funkcji

# New Live script



New  
Live Script