

Metodyka i Techniki Programowania II

Katedra Telekomunikacji, EiT

dr inż. Jarosław Bułat (c)

kwant@agh.edu.pl

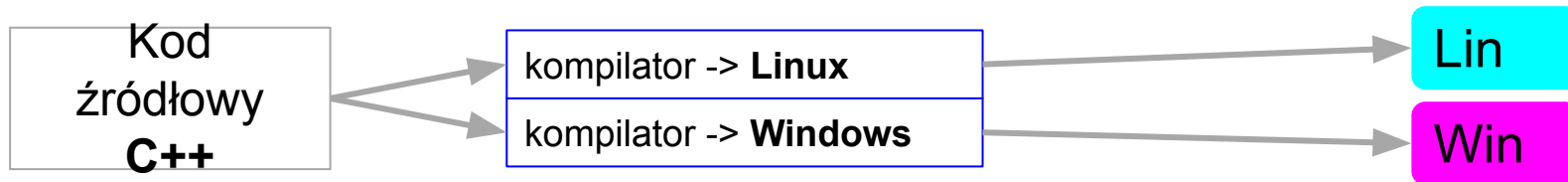
Plan prezentacji

- » Kompilator vs Interpreter
- » Matlab - główne założenia
- » Środowisko
- » Zmienna, indeksowanie macierzy
- » Przykłady prostych obliczeń

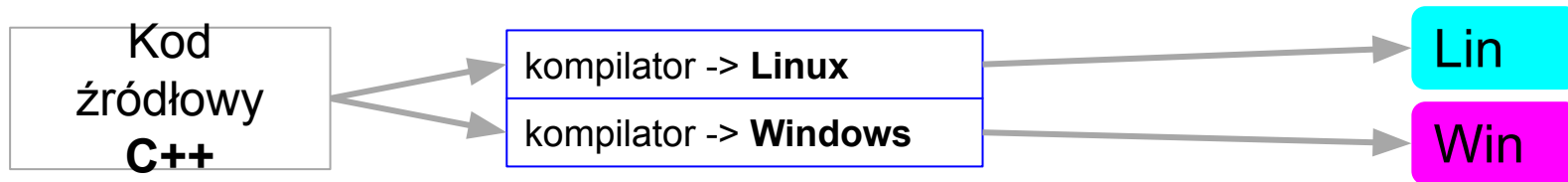
Chciałbym napisać jeden program na wiele platform

nawet na takie, które jeszcze nie istnieją

Kompilator vs Interpreter



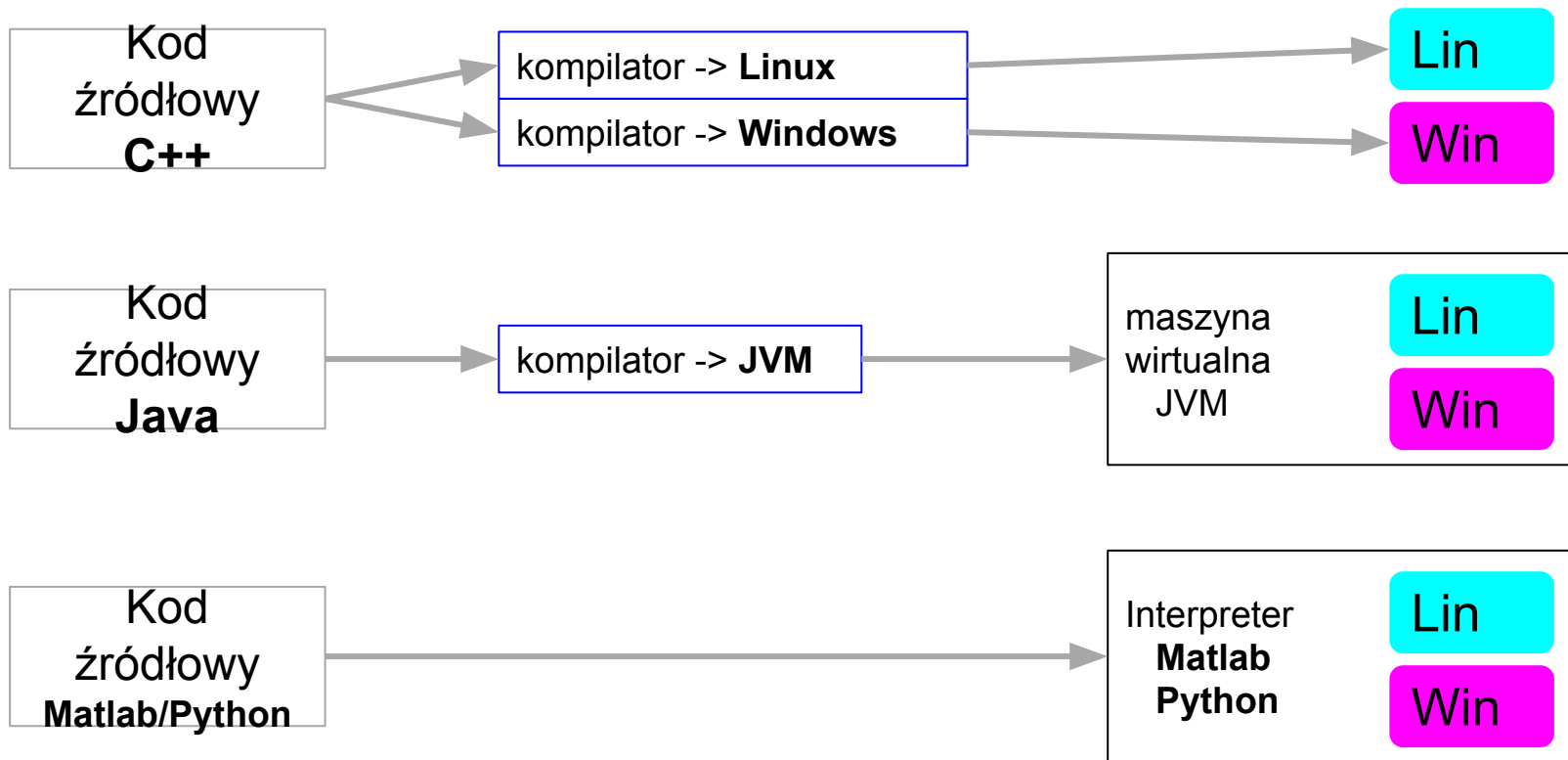
Kompilator vs Interpreter



- musi istnieć: **platforma, kompilator, biblioteki, środowisko** uruchomieniowe
- **x86-64**, ARM, PowerPC, SPARC, AVR, Alpha, IA, MISP, Motorola, VAX,
- **'-march=skylake'**

Intel Skylake CPU with 64-bit extensions, MOVBE, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, POPCNT, AVX, AVX2, AES, PCLMUL, FSGSBASE, RDRND, FMA, BMI, BMI2, F16C, RDSEED, ADCX, PREFETCHW, CLFLUSHOPT, XSAVEC and XSAVES instruction set support.

Kompilator vs Interpreter



Kompilator vs Interpreter

- musi istnieć: **platforma, biblioteki, środowisko uruchomieniowe**
- **x86-64**, ARM, PowerPC, SPARC, AVR, Alpha, IA, MISP, Motorola, VAX,
- ‘-march=skylake’

Intel Skylake CPU with 64-bit extensions, MOVBE, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, POPCNT, AVX, AVX2, AES, PCLMUL, FSGSBASE, RDRND, FMA, BMI, BMI2, F16C, RDSEED, ADCX, PREFETCHW, CLFLUSHOPT, XSAVEC and XSAVES instruction set support.

Lin

Win

maszyna
wirtualna
JVM

Lin

Win

Interpreter
Matlab
Python

Lin

Win

Kompilator vs Interpreter

Kompilator

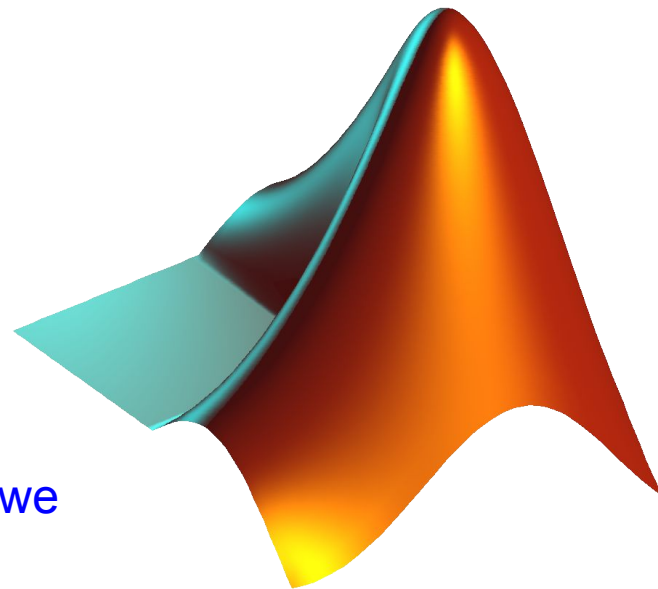
- max. szybkość działania programu (asm)
- nie jest wymagany runtime
- może działać bare metal
- kompilacja tylko raz
- nieprzenośne
- kompilacja wymaga znajomości docelowej arch.
- trudniej testować

Interpreter

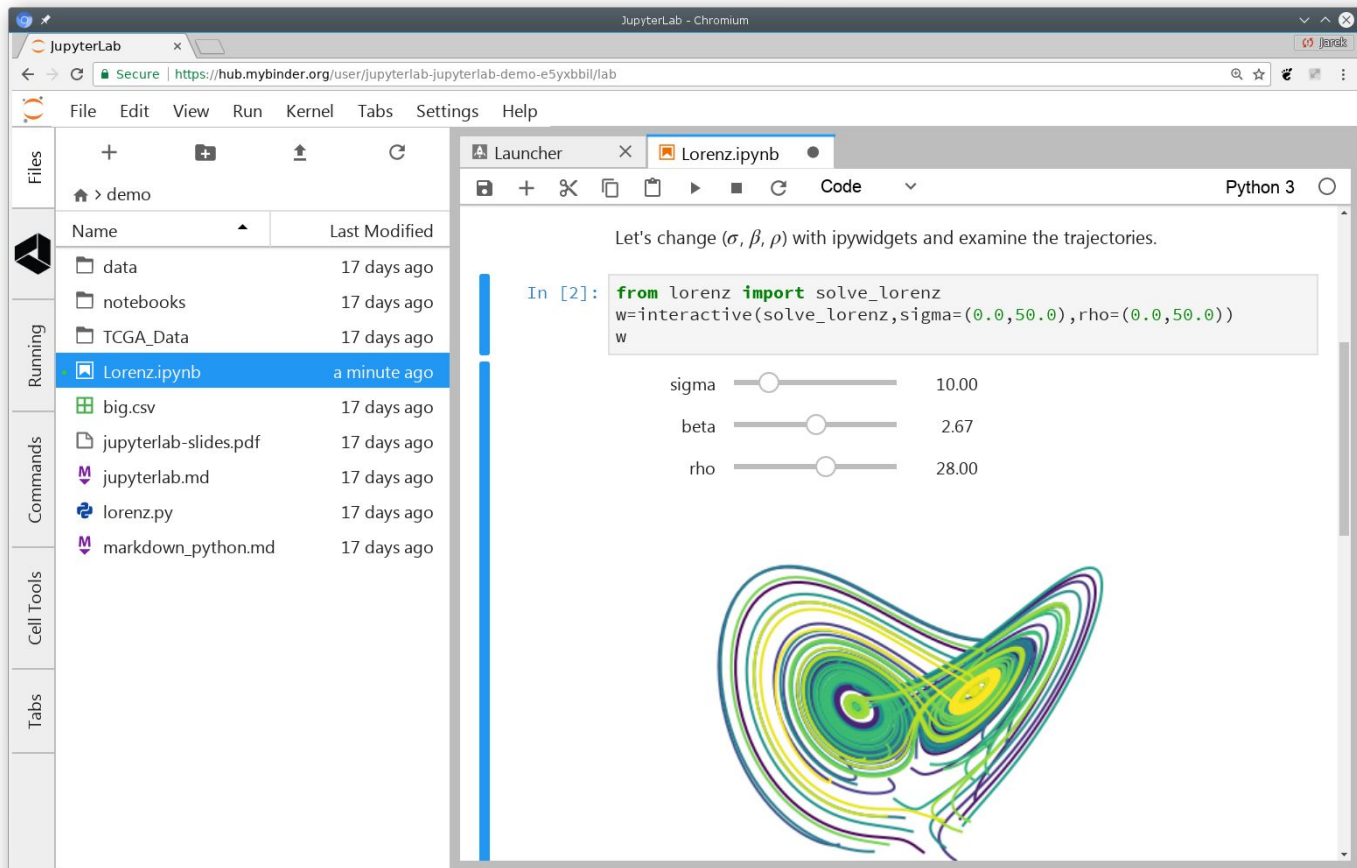
- łatwość pisania i testowania kodu
- przenośność (OS/sprzęt)
- będzie działać na przyszłych architekturach
- mniejsza wydajność
- wymagane większe zasoby (energochłonność)
- trudniej sprzedać aplikację (ukryć algoryt - knowhow)

Matlab - The MathWorks

- » **Matlab == MATrix LABoratory**
- » Program komputerowy
- » **Interaktywne środowisko** do:
 - obliczeń naukowych
 - obliczeń inżynierskich
 - symulacji komputerowych
- » **Język programowania**
- » Własne (jedyne!) **środowisko uruchomieniowe** i developerskie IDE
- » Praca w środowisku **graficznym** i tekstowym
- » Wieloplatformowość (Java)
- » Alternatywy FLOSS: Octave, Scilab, **Python + NumPy**, Julia, Jupyter-notebook (Python, Julia, Matlab, R), **Jupyter-lab**



JupyterLab



The screenshot displays the JupyterLab web interface in a Chromium browser. The address bar shows the URL: <https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-e5yxbbil/lab>. The interface is divided into several panels:

- Files Panel:** Shows a file browser for the 'demo' directory. It lists files and folders with their last modified times:

Name	Last Modified
data	17 days ago
notebooks	17 days ago
TCGA_Data	17 days ago
Lorenz.ipynb	a minute ago
big.csv	17 days ago
jupyterlab-slides.pdf	17 days ago
jupyterlab.md	17 days ago
lorenz.py	17 days ago
markdown_python.md	17 days ago
- Code Editor:** The active file is 'Lorenz.ipynb'. It contains the following Python code:


```
Let's change ( $\sigma$ ,  $\beta$ ,  $\rho$ ) with ipywidgets and examine the trajectories.

In [2]: from lorenz import solve_lorenz
w=interactive(solve_lorenz,sigma=(0.0,50.0),rho=(0.0,50.0))
w
```
- Interactive Widgets:** Below the code, there are three sliders for parameters:
 - sigma: 10.00
 - beta: 2.67
 - rho: 28.00
- Figure:** A plot of the Lorenz attractor trajectories, showing a complex, chaotic pattern with multiple overlapping loops in various colors.

Matlab

- » Wywodzi się z języka **Fortran** (konwencja indeksowania, składnia wzorowana na C)
- » Siłą Matlaba są biblioteki - **Toolbox** (biblioteki standardowe)
- » Pisząc program w Matlabie, używa się:
 - języka Matlab
 - środowiska programistycznego Matlab
 - środowiska uruchomieniowego Matlab
- » Program napisany w języku Matlab jest **interpretowany**
- » Funkcje biblioteczne (z toolbox-ów) są zazwyczaj “natywne”, tj. skompilowane dla aktualnie używanej architektury
- » Program napisany w Matlabie można skompilować (jak Java)

Matlab

» Zalety:

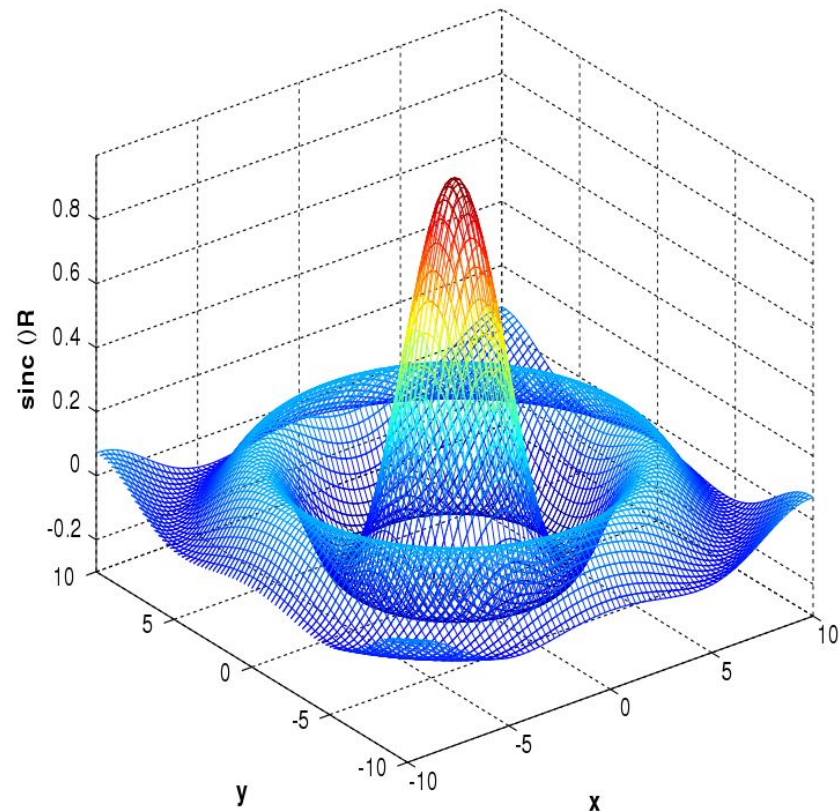
- szybkie pisanie, prototypowanie programu (algorytmu)
- dostęp do wielu bibliotek, dobrze przetestowanych
- IDE, wizualizacja wyników, **dokumentacja**
- przenośność, wieloprocusowość, standard “przemysłowy”
- **Simulink**: graficzny język (diagramy blokowe)

» Wady:

- **cena !!!**
- program wynikowy jest wolny, 10-100x wolniejszy niż C++
- “podstarzały” język: funkcje, struktury, od kilku lat forsowana obiektowość (niechętnie używana)

Matlab - cena za jakość

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)
```



Matlab - cena za jakość

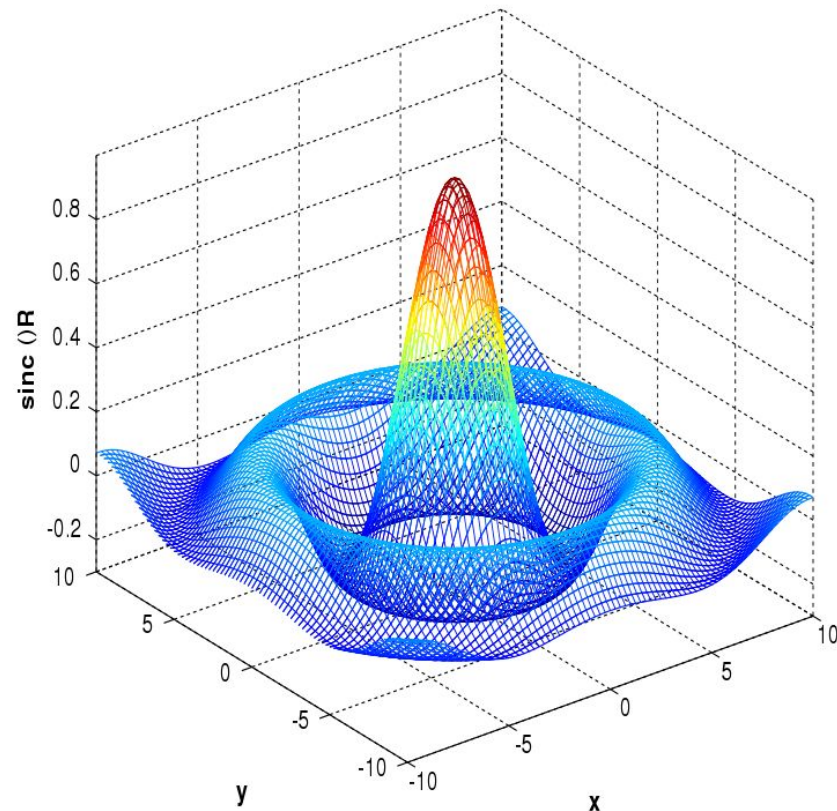
```
[X,Y] = meshgrid(-8:5:8);
```

```
R = sqrt(X.^2 + Y.^2) + eps;
```

```
Z = sin(R)./R;
```

```
mesh(X,Y,Z)
```

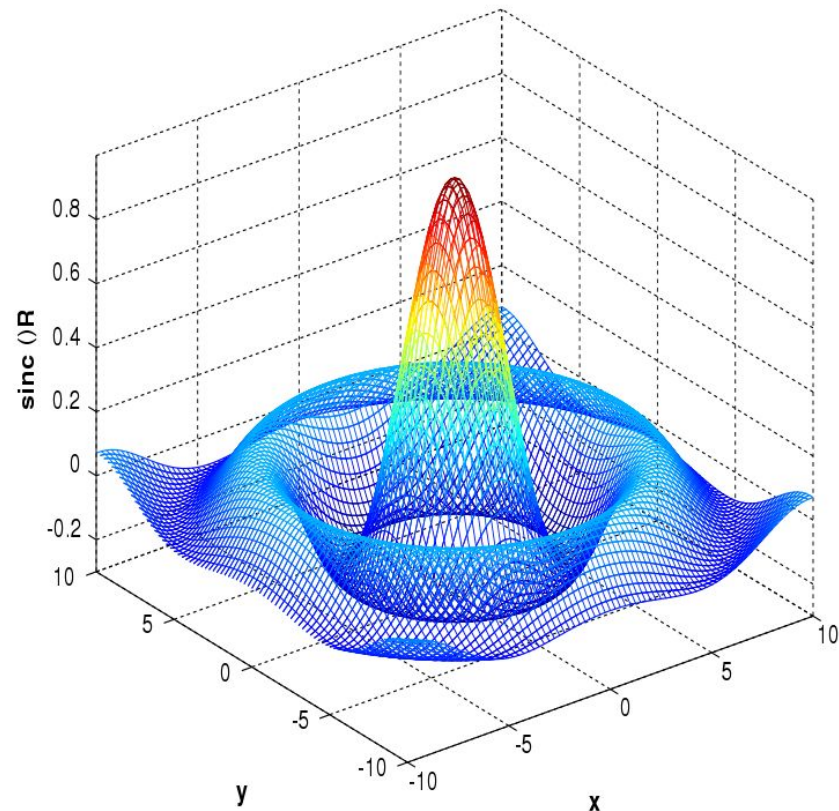
» Matlab: **EUR 2000** (800/rok)



Matlab - cena za jakość

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)
```

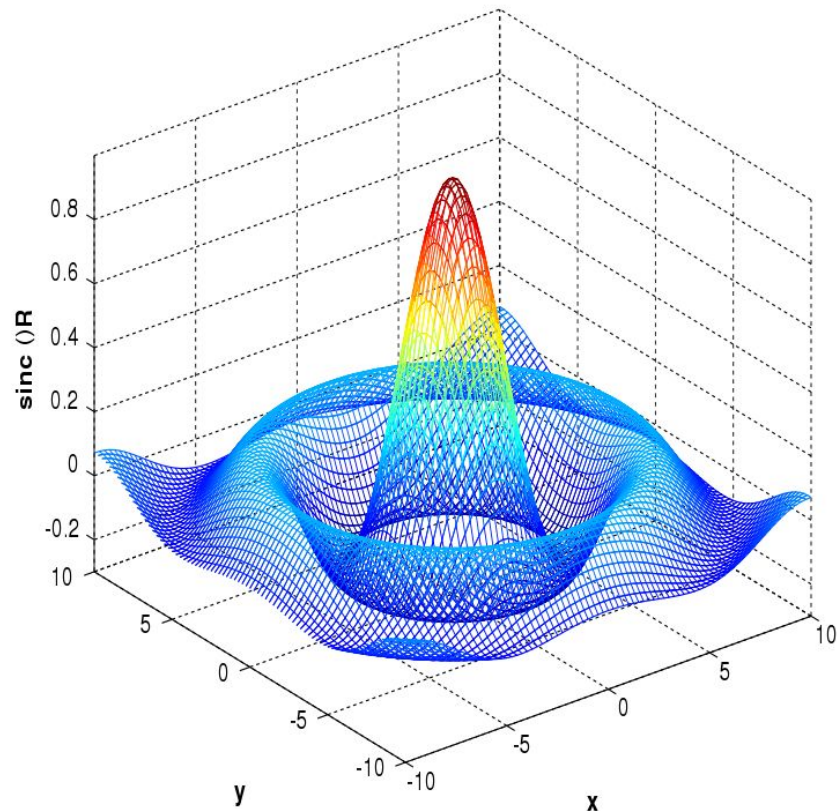
- » Matlab: **EUR 2000** (800/rok)
- » Comm: **EUR 1250** (500/rok)



Matlab - cena za jakość

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)
```

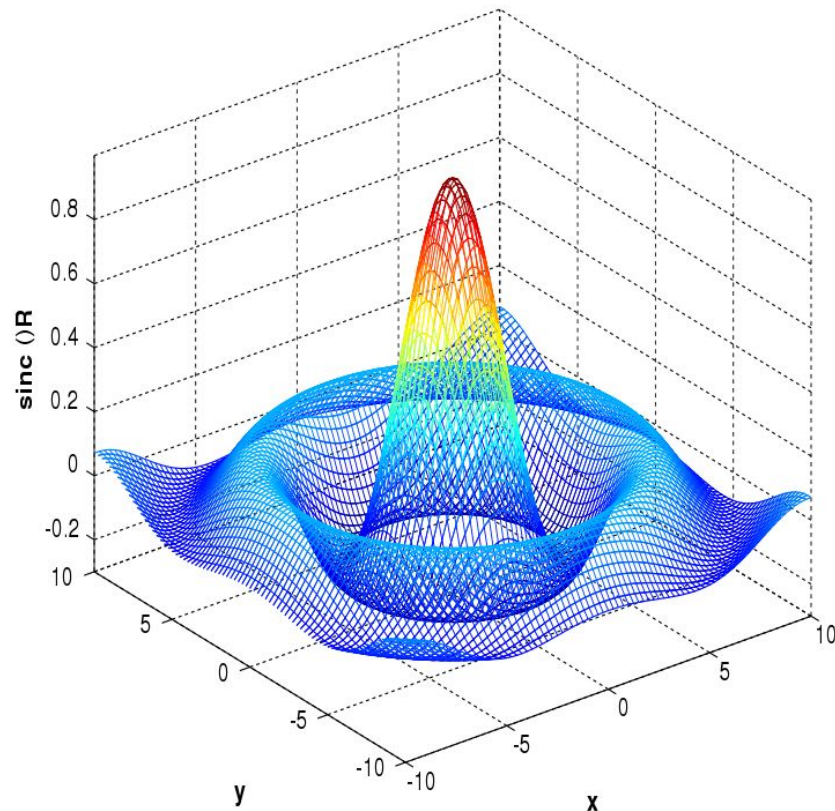
- » Matlab: **EUR 2000** (800/rok)
- » Comm: **EUR 1250** (500/rok)
 - DSP: **EUR 1250** (500)
 - SP: **EUR 1000** (400)



Matlab - cena za jakość

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)
```

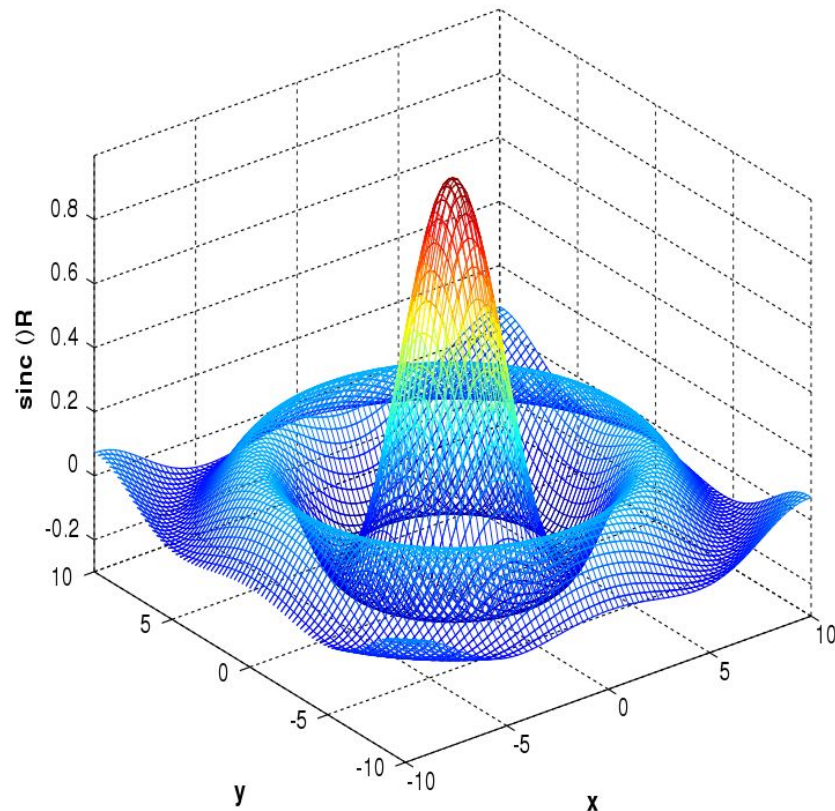
- » Matlab: **EUR 2000** (800/rok)
- » Comm: **EUR 1250** (500/rok)
 - DSP: **EUR 1250** (500)
 - SP: **EUR 1000** (400)
- » Antenna: **EUR 2000**



Matlab - cena za jakość

```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)
```

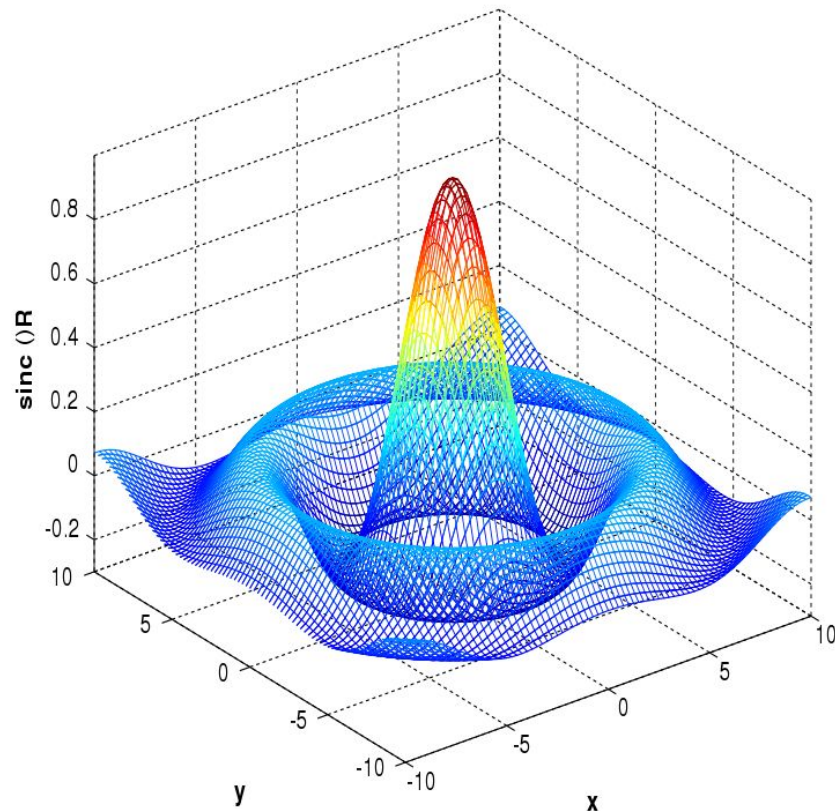
- » Matlab: **EUR 2000** (800/rok)
- » Comm: **EUR 1250** (500/rok)
 - DSP: **EUR 1250** (500)
 - SP: **EUR 1000** (400)
- » Antenna: **EUR 2000**
- » Data acquisition: **EUR 1000**



Matlab - cena za jakość

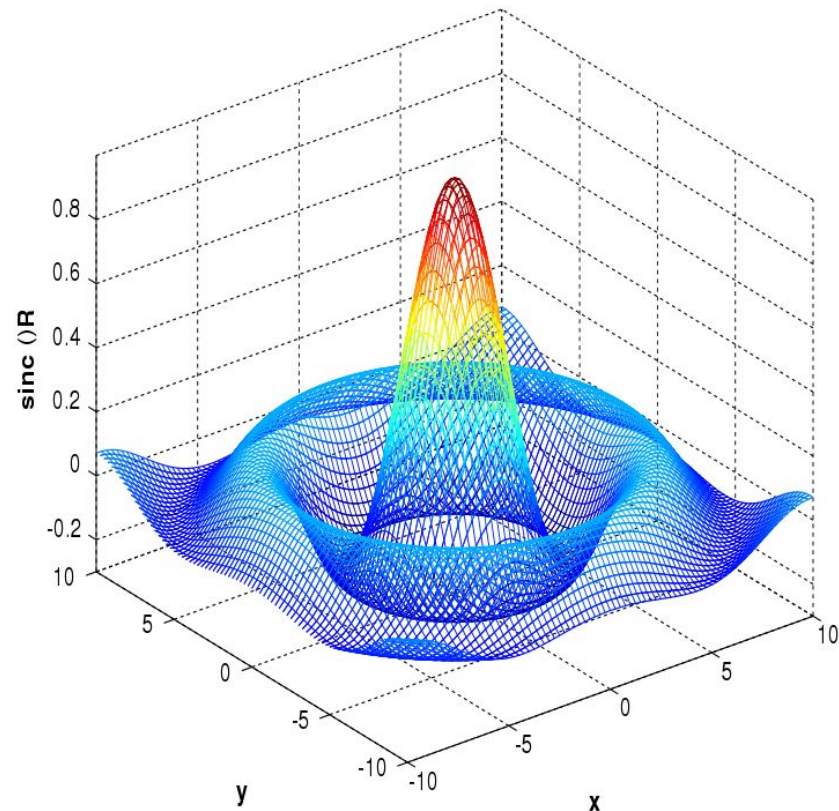
```
[X,Y] = meshgrid(-8:5:8);  
R = sqrt(X.^2 + Y.^2) + eps;  
Z = sin(R)./R;  
mesh(X,Y,Z)
```

- » Matlab: **EUR 2000** (800/rok)
- » Comm: **EUR 1250** (500/rok)
 - DSP: **EUR 1250** (500)
 - SP: **EUR 1000** (400)
- » Antenna: **EUR 2000**
- » Data acquisition: **EUR 1000**
- » Image Processing: **1000**



Matlab - cena za jakość

- » **Education:** 4-5x taniej
- » **Home:** EUR 119
 - DSP: EUR 35
 - Comm: EUR 35
- » **Student:** EUR 69 (suite)



Składnia - prosty program

Matlab vs C++

Hello world x 10

```
% comment
for x = 1:10
    disp('Hello world '+string(x));
end
```

```
#include<iostream>
using namespace std;

// comment
int main(){
    for (size_t x = 0; x < 10; ++x){
        cout << "Hello world " << x << endl;
    }
}
```

- » Prostota (częsta cecha języków interpretowanych)
- » Składnia, zasady, etc... podobne do innych języków imperatywnych (proceduralnych)

Hello world x 10

```
% comment
for x = 1:10
    disp('Hello world '+string(x));
end
```

```
#include<iostream>
using namespace std;

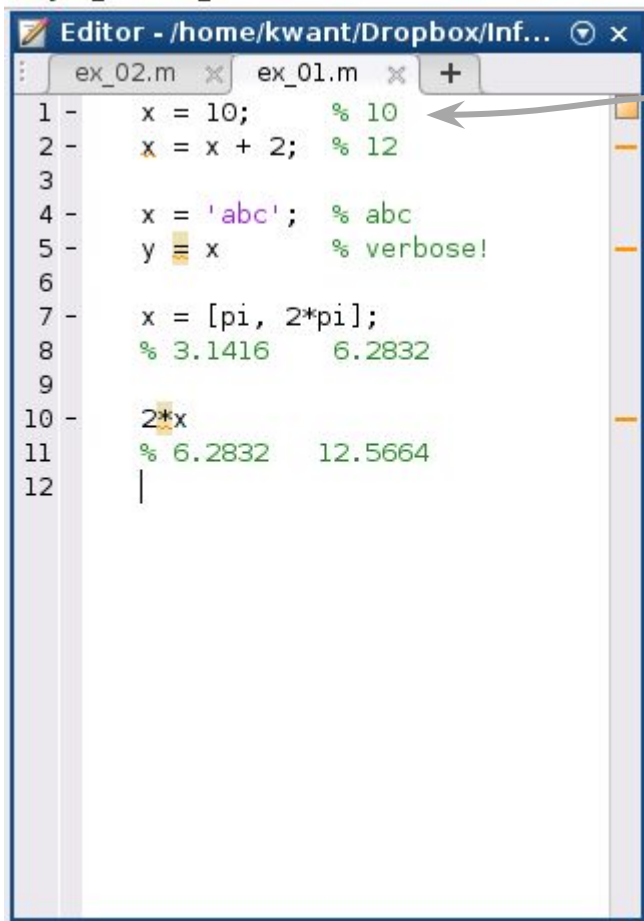
// comment
int main(){
    for (size_t x = 0; x < 10; ++x){
        cout << "Hello world " << x << endl;
    }
}
```

- » Prostota (częsta cecha języków interpretowanych)
- » Składnia, zasady, etc... podobne do innych języków imperatywnych (proceduralnych)
- » Znajdź 10 różnic ;-)

Matlab - język

- » Język wysokiego poziomu: **funkcje**, **struktury**, **programowanie obiektowe**
- » Standardowe instrukcje sterujące: **if**, **for**, **while**, **switch**
- » Brak konieczności deklaracji zmiennych
- » Język dynamicznie typowany
- » Podstawowy (defaultowy) typ zmiennej to **complex double**
- » Zmienne mogą być: **skalarem**, **wektorem**, **macierzą**
- » Podstawowe operacje arytmetyczne są zdefiniowane na macierzach: **$a=x*b$** to mnożenie macierzowe macierzy **x** oraz macierzy **b** o odpowiednich rozmiarach.
- » Typy danych: **double**, **char**, **sparse**, **struct**, **cell**, **uint8**
- » Średnik na końcu wyrażenia zapobiega wypisaniu zawartości na konsoli

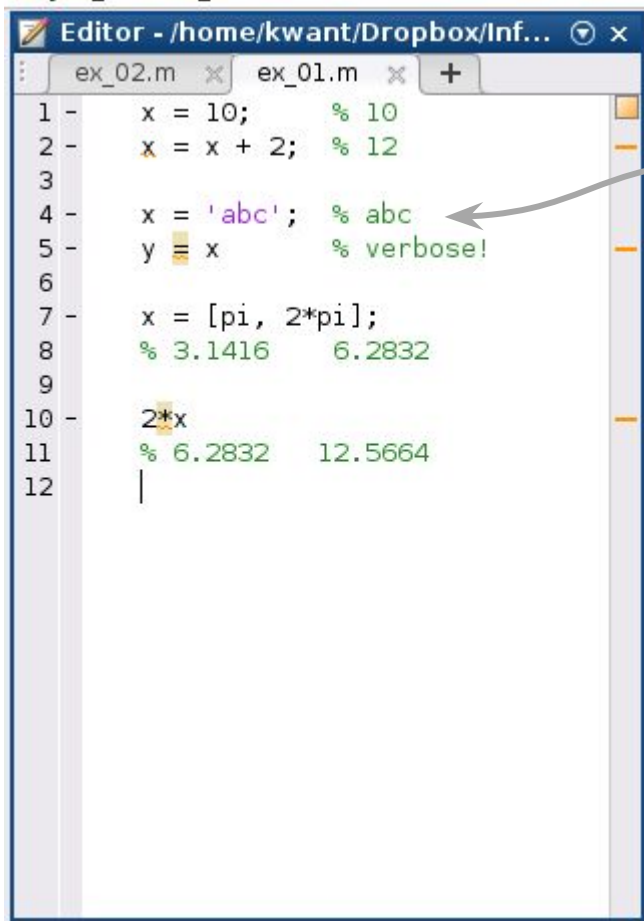
Matlab - język



```
Editor - /home/kwant/Dropbox/Inf...
ex_02.m ex_01.m +
1 - x = 10;      % 10
2 - x = x + 2;  % 12
3
4 - x = 'abc';  % abc
5 - y = x      % verbose!
6
7 - x = [pi, 2*pi];
8 - % 3.1416    6.2832
9
10 - 2*x
11 - % 6.2832    12.5664
12
```

» Brak deklaracji zmiennych

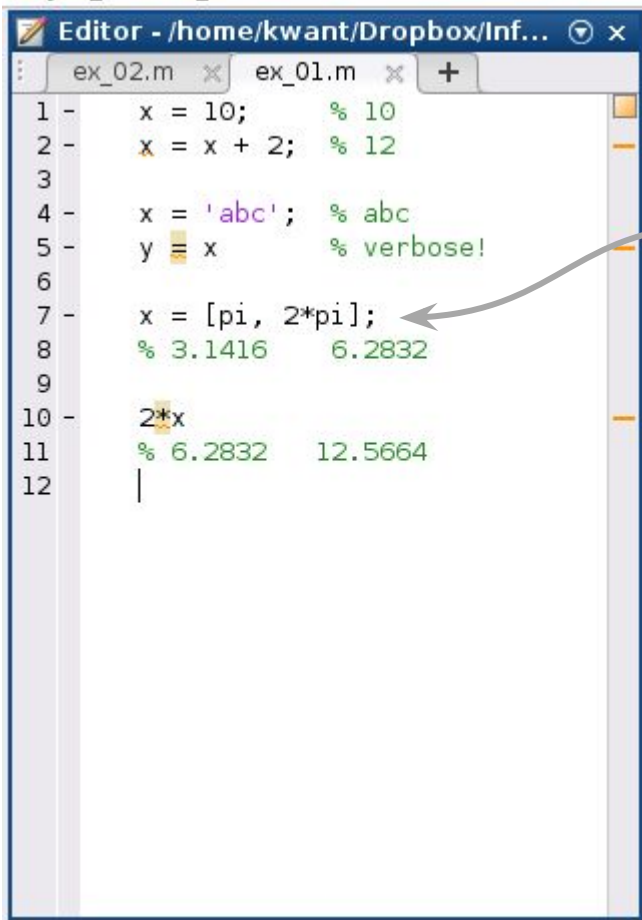
Matlab - język



```
Editor - /home/kwant/Dropbox/Inf...
ex_02.m ex_01.m +
1 - x = 10;      % 10
2 - x = x + 2;  % 12
3
4 - x = 'abc';  % abc
5 - y = x      % verbose!
6
7 - x = [pi, 2*pi];
8 - % 3.1416    6.2832
9
10 - 2*x
11 - % 6.2832    12.5664
12
```

- » Brak deklaracji zmiennych
- » Zmienne zmieniają typ

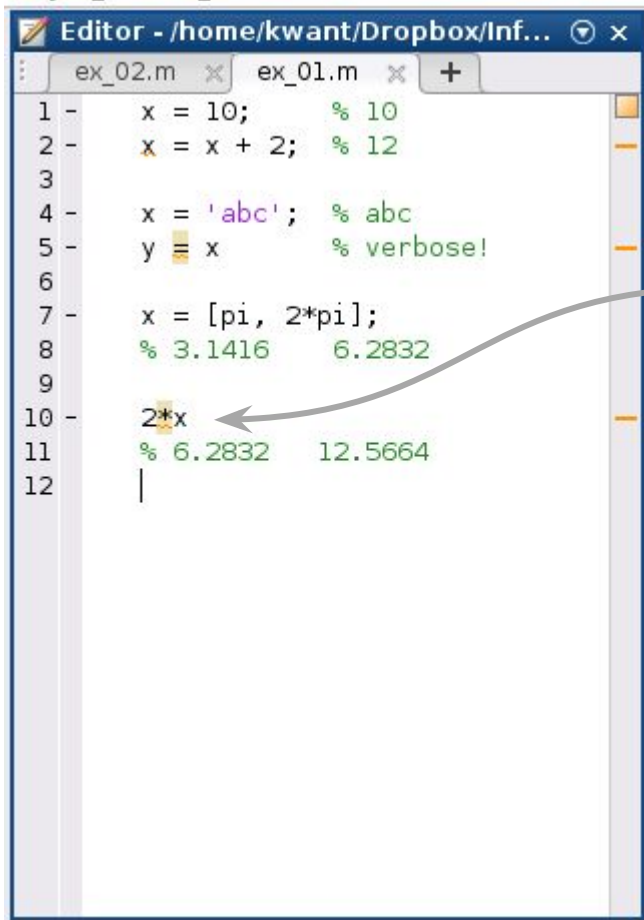
Matlab - język



```
Editor - /home/kwant/Dropbox/Inf...
ex_02.m ex_01.m +
1 - x = 10; % 10
2 - x = x + 2; % 12
3
4 - x = 'abc'; % abc
5 - y = x % verbose!
6
7 - x = [pi, 2*pi];
8 % 3.1416 6.2832
9
10 - 2*x
11 % 6.2832 12.5664
12
```

- » Brak deklaracji zmiennych
- » Zmienne zmieniają typ
- » Wektor, Macierz N-D (tablica)

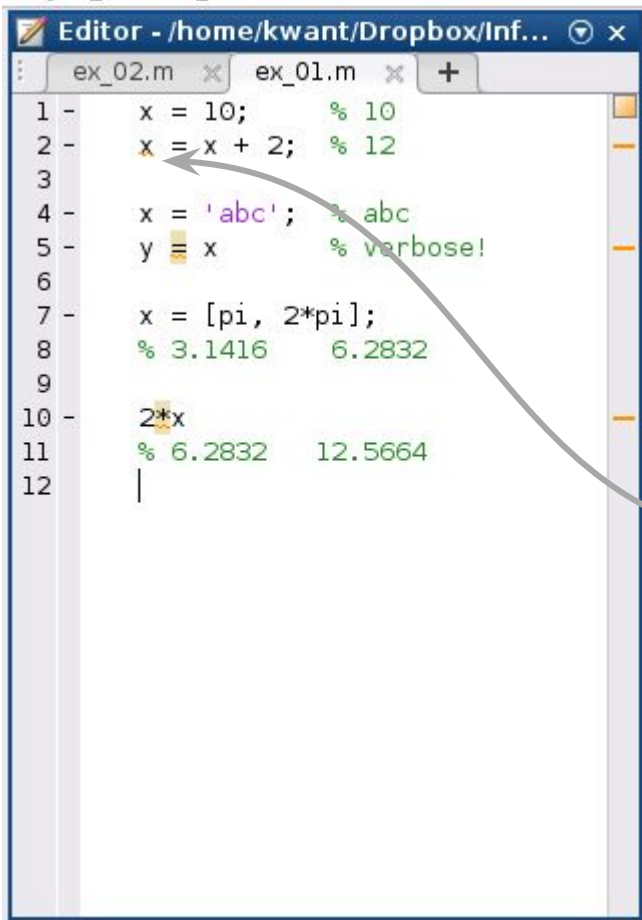
Matlab - język



```
Editor - /home/kwant/Dropbox/Inf...
ex_02.m ex_01.m +
1 - x = 10; % 10
2 - x = x + 2; % 12
3
4 - x = 'abc'; % abc
5 - y = x % verbose!
6
7 - x = [pi, 2*pi];
8 - % 3.1416 6.2832
9
10 - 2*x
11 - % 6.2832 12.5664
12
```

- » Brak deklaracji zmiennych
- » Zmienne zmieniają typ
- » Wektor, Macierz N-D (tablica)
- » Mnożenie **skalar * wektor**
 - +, -, *, /, ^, inv(), ...

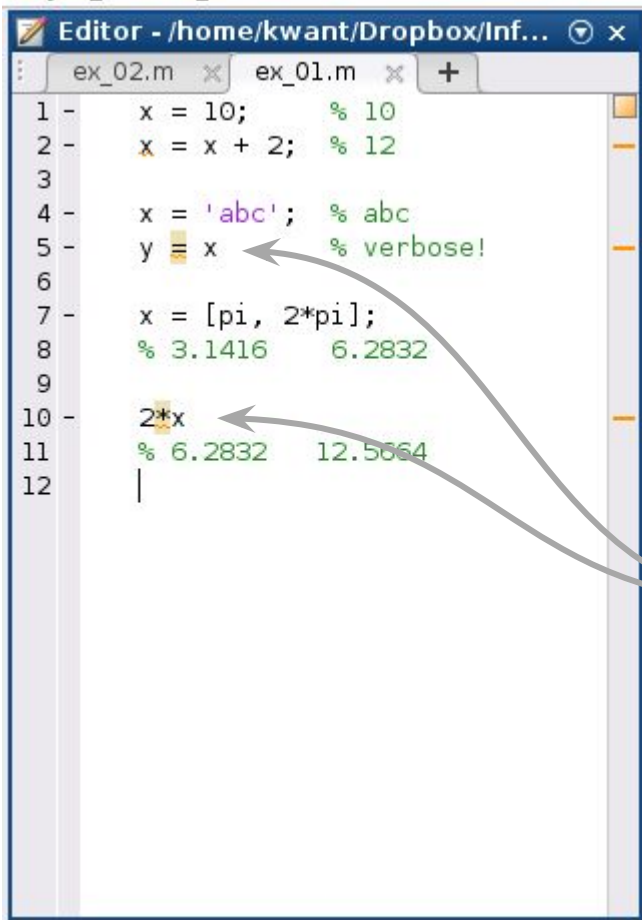
Matlab - język



```
Editor - /home/kwant/Dropbox/Inf...
ex_02.m  ex_01.m  +
1 - x = 10;           % 10
2 - x = x + 2;       % 12
3
4 - x = 'abc';       % abc
5 - y = x            % verbose!
6
7 - x = [pi, 2*pi];
8 - % 3.1416         6.2832
9
10 - 2*x
11 - % 6.2832        12.5664
12 - |
```

- » Brak deklaracji zmiennych
- » Zmienne zmieniają typ
- » Wektor, Macierz N-D (tablica)
- » Mnożenie skalar * wektor
 - +, -, *, /, ^, inv(), ...
- » Środowisko podpowiada usterki w kodzie:
 - 'x' might be **unused**

Matlab - język



```

Editor - /home/kwant/Dropbox/Inf...
ex_02.m  ex_01.m  +
1 - x = 10;      % 10
2 - x = x + 2;  % 12
3
4 - x = 'abc';  % abc
5 - y = x      % verbose!
6
7 - x = [pi, 2*pi];
8 - % 3.1416    6.2832
9
10 - 2*x
11 - % 6.2832    12.5664
12 -
  
```

- » Brak deklaracji zmiennych
- » Zmienne zmieniają typ
- » Wektor, Macierz N-D (tablica)
- » Mnożenie skalar * wektor
 - +, -, *, /, ^, inv(), ...
- » Środowisko podpowiada usterki w kodzie:
 - 'x' might be unused
 - terminate statement

Operacje macierzowe

”każda zmienna jest macierzą”

Matlab - macierze

```
x = [1,2,3; 4,5,6]
```

```
% x =
```

```
%    1    2    3
```

```
%    4    5    6
```

```
y = [4,5; 2,3; 1,1]
```

```
% y =
```

```
%    4    5
```

```
%    2    3
```

```
%    1    1
```

```
z=x*y
```

```
% z =
```

```
%    11    14
```

```
%    32    41
```

```
z(1,1)
```

```
% ans =
```

```
%    11
```

» Macierze **inicjalizowane** w nawiasach kwadratowych

Matlab - macierze

```
x = [1,2,3; 4,5,6]
```

```
% x =
```

```
%   1   2   3
```

```
%   4   5   6
```

```
y = [4,5; 2,3; 1,1]
```

```
% y =
```

```
%   4   5
```

```
%   2   3
```

```
%   1   1
```

```
z=x*y
```

```
% z =
```

```
%  11  14
```

```
%  32  41
```

```
z(1,1)
```

```
% ans =
```

```
%   11
```

» Macierze inicjalizowane w nawiasach kwadratowych

“ , ” - kolejne liczby w wierszu

“ ; ” - koniec wiersza

Matlab - macierze

```
x = [1,2,3; 4,5,6]
```

```
% x =
```

```
%   1   2   3
```

```
%   4   5   6
```

```
y = [4,5; 2,3; 1,1]
```

```
% y =
```

```
%   4   5
```

```
%   2   3
```

```
%   1   1
```

```
z=x*y
```

```
% z =
```

```
%  11  14
```

```
%  32  41
```

```
z(1,1)
```

```
% ans =
```

```
%   11
```

» Macierze inicjalizowane w nawiasach kwadratowych

» Mnożenie macierzowe

Matlab - macierze

```
x = [1,2,3; 4,5,6]
```

```
% x =
```

```
%   1   2   3
```

```
%   4   5   6
```

```
y = [4,5; 2,3; 1,1]
```

```
% y =
```

```
%   4   5
```

```
%   2   3
```

```
%   1   1
```

```
z=x*y
```

```
% z =
```

```
%  11  14
```

```
%  32  41
```

```
z(1,1)
```

```
% ans =
```

```
%    11
```

- » Macierze inicjalizowane w nawiasach kwadratowych
- » Mnożenie macierzowe
- » Rozmiary muszą się zgadzać, inaczej błąd:

Error using *
Inner matrix dimensions must agree.

Matlab - macierze

```
x = [1,2,3; 4,5,6]
```

```
% x =
```

```
%   1   2   3
```

```
%   4   5   6
```

```
y = [4,5; 2,3; 1,1]
```

```
% y =
```

```
%   4   5
```

```
%   2   3
```

```
%   1   1
```

```
z=x*y
```

```
% z =
```

```
%  11  14
```

```
%  32  41
```

```
z(1,1)
```

```
% ans =
```

```
%   11
```

- » Macierze inicjalizowane w nawiasach kwadratowych
- » Mnożenie macierzowe
- » Rozmiary muszą się zgadzać, inaczej błąd:

Error using *

Inner matrix dimensions must agree.

- » Indeksowanie macierzy nawiasami okrągłymi

Matlab - macierze

```
x = [1,2,3; 4,5,6]
```

```
% x =
```

```
%   1   2   3
```

```
%   4   5   6
```

```
y = [4,5; 2,3; 1,1]
```

```
% y =
```

```
%   4   5
```

```
%   2   3
```

```
%   1   1
```

```
z=x*y
```

```
% z =
```

```
%  11  14
```

```
%  32  41
```

```
z(1,1)
```

```
% ans =
```

```
%   11
```

- » Macierze inicjalizowane w nawiasach kwadratowych
- » Mnożenie macierzowe
- » Rozmiary muszą się zgadzać, inaczej błąd:

Error using *

Inner matrix dimensions must agree.

- » Indeksowanie macierzy nawiasami okrągłymi
- » Indeksowanie od "1" (jak w Pascalu, Fortranie) a nie od "0" jak w C/C++ !!!

Matlab - macierze

```
x = [1,2; 4,5];
```

```
% x =
```

```
%    1    2
```

```
%    4    5
```

```
y = [4,5; 2,3];
```

```
% y =
```

```
%    4    5
```

```
%    2    3
```

```
z=x*y;
```

```
% z =
```

```
%    8    11
```

```
%   26    35
```

```
z=x.*y;
```

```
% z =
```

```
%    4    10
```

```
%    8    15
```

- » Dwie macierze kwadratowe
- » Mnożenie macierzowe
- » Mnożenie poszczególnych elementów macierzy

Matlab - macierze

```
x = [1,2; 4,5];
```

```
% x =
```

```
%   1   2
```

```
%   4   5
```

```
y = [4,5; 2,3];
```

```
% y =
```

```
%   4   5
```

```
%   2   3
```

```
z=x*y;
```

```
% z =
```

```
%   8  11
```

```
%  26  35
```

```
z=x.*y;
```

```
% z =
```

```
%   4  10
```

```
%   8  15
```

- » Dwie macierze kwadratowe
- » Mnożenie macierzowe
- » Mnożenie poszczególnych elementów macierzy
- » Zawsze kończyć wyrażenie średnikiem

Matlab - zakres

clear

close **all**

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

% x =

% 1 2 3

% 4 5 6

% 7 8 9

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

» Na początku zawsze **skasuj**
wszystkie zmienne z workspace

Matlab - zakres

```
clear
```

```
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » **Zamknij** wszystkie otwarte okienka z wykresami

Matlab - zakres

```
clear  
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » Zamknij wszystkie otwarte okienka z wykresami
- » Można wybrać zakres elementów z macierzy:

Matlab - zakres

```
clear  
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » Zamknij wszystkie otwarte okienka z wykresami
- » Można wybrać zakres elementów z macierzy:
 - pierwszy wiersz

Matlab - zakres

```
clear  
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » Zamknij wszystkie otwarte okienka z wykresami
- » Można wybrać zakres elementów z macierzy:
 - pierwszy wiersz
 - druga kolumna

Matlab - zakres

```
clear  
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » Zamknij wszystkie otwarte okienka z wykresami
- » Można wybrać zakres elementów z macierzy:
 - pierwszy wiersz
 - druga kolumna
 - z trzeciej kolumny wiersze od 1 do 2 (włącznie)

Matlab - zakres

```
clear  
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » Zamknij wszystkie otwarte okienka z wykresami
- » Można wybrać zakres elementów z macierzy:
 - pierwszy wiersz
 - druga kolumna
 - z trzeciej kolumny wiersze od 1 do 2 (włącznie)
 - z pierwszego wiersza, kolumny od 2 do końca

Matlab - zakres

```
clear  
close all
```

```
x = [1,2,3;  
     4,5,6;  
     7,8,9];
```

```
% x =  
%   1   2   3  
%   4   5   6  
%   7   8   9
```

```
x(1, :);  
x(:, 2);  
x(1:2, 3);  
x(1, 2:end);  
x(:, end);
```

- » Na początku zawsze skasuj wszystkie zmienne z workspace
- » Zamknij wszystkie otwarte okienka z wykresami
- » Można wybrać zakres elementów z macierzy:
 - pierwszy wiersz
 - druga kolumna
 - z trzeciej kolumny wiersze od 1 do 2 (włącznie)
 - z pierwszego wiersza, kolumny od 2 do końca
 - ostatnia kolumna

Instrukcje sterujące

if, for, while, switch

Matlab - if

clear
close all

if expression
 statements
elseif expression
 statements
else
 statements
end

- » Działanie tożsame z C/C++, składnia trochę inna
- » Nie ma klamerek wydzielających blok instrukcji
- » Nie ma nawiasów w warunku (są opcjonalne)
- » elseif, else - opcjonalne
- » elseif - dowolna ilość
- » end - kończy warunek (zamiast klamerek)

Matlab - if

```
clear
```

```
x = 3;
```

```
if x == 1
```

```
    disp('1');
```

```
elseif x <= 2
```

```
    disp('<=3');
```

```
elseif abs(x) > 2.4
```

```
    disp('abs(x) > 2.4');
```

```
elseif x > 5 && x < 7
```

```
    disp('5 < x < 7');
```

```
else
```

```
    disp('unsupported');
```

```
end
```

» Wyrażenie logiczne:

- ==, <, >, <=, >=
- ~ (negacja)
- ||, &&

» Jaki będzie wynik???

Matlab - switch

clear

» Jak w C++

```
switch switch_expression
case case_expression
    statements
case case_expression
    statements
...
otherwise
    statements
end
```

Matlab - pętla for

clear

```
for index = values  
    statements  
end
```

- » Prawie jak w C++
- » Nie ma wyrażenia trójargumentowego
- » Pętla oparta o wyrażenie dwuargumentowe
- » **Iterator**
- » Dla wszystkich wartości **ze zbioru**
- » Instrukcje wykonają się tyle razy ile było wartości w zmiennej **values**, wartość **index** będzie przyjmować kolejne wartości ze zmiennej **values**

Matlab - pętla for, przykład

```
clear;
```

```
for v = [3 2 0 -7]  
    disp(v)  
end
```

```
% 3
```

```
% 2
```

```
% 0
```

```
% -7
```

- » Prosta pętla
- » W kolejnej iteracji, zmienna **v** przybierze kolejne wartości z wektora **v**

Matlab - pętla for, przykład

```
clear;
```

```
w = 5;
```

```
h = 6;
```

```
y = zeros(w, h);
```

```
for r = 1:w
```

```
    for c = 1:h
```

```
        y(r,c) = r+c;
```

```
    end
```

```
end
```

```
% y =
```

```
%   2   3   4   5   6   7
```

```
%   3   4   5   6   7   8
```

```
%   4   5   6   7   8   9
```

```
%   5   6   7   8   9  10
```

```
%   6   7   8   9  10  11
```

» Pętla w pętli

» Wyrażenie **1:w** wygeneruje wszystkie liczby od **1** do **w** włącznie: [1, 2, 3, 4, 5];

Matlab - pętla for, przykład

```
clear;
```

```
w = 5;
```

```
h = 6;
```

```
y = zeros(w, h);
```

```
% y = [];
```

```
for r = 1:w
```

```
    for c = 1:h
```

```
        y(r,c) = r+c;
```

```
    end
```

```
end
```

```
% y =
```

```
%    2    3    4    5    6    7
```

```
%    3    4    5    6    7    8
```

```
%    4    5    6    7    8    9
```

```
%    5    6    7    8    9   10
```

```
%    6    7    8    9   10   11
```

» Pętla w pętli

» Wyrażenie `1:w` wygeneruje wszystkie liczby od 1 do w włącznie: [1, 2, 3, 4, 5];

» Prealokowanie miejsca

» Można utworzyć pustą macierz i potem ją **rozszerzać** ale będzie to **znacznie wolniejsze**

Matlab - pętla for, przykład

```
clear;
```

```
w = 5;
```

```
h = 6;
```

```
y = zeros(w, h);
```

```
% y = [];
```

```
for r = 1:w
```

```
    for c = 1:h
```

```
        y(r,c) = r+c;
```

```
    end
```

```
end
```

```
% y =
```

```
%   2   3   4   5   6   7
%   3   4   5   6   7   8
%   4   5   6   7   8   9
%   5   6   7   8   9  10
%   6   7   8   9  10  11
```

» Pętla w pętli

» Wyrażenie `1:w` wygeneruje wszystkie liczby od 1 do w włącznie: [1, 2, 3, 4, 5];

» Prealokowanie miejsca

» Można utworzyć pustą macierz i potem ją rozszerzać ale będzie to **znacznie wolniejsze**

» **Indeksowanie** iteratorami wynikowej macierz i obliczanie wyniku (**wzór matematyczny**),
uwaga pułapka !!!!

Matlab - pętla parfor

```
n = 200;  
A = 500;  
a = zeros(n);  
tic  
for i = 1:n  
    a(i) = max(abs(rand(1,A)));  
end  
toc  
  
tic  
parfor i = 1:n  
    a(i) = max(abs(rand(1,A)));  
end  
toc
```

- » **parfor** zrównoleglenie obliczeń na poziomie języka !!!
- » **Warunki:**
 - iterator musi być liczbą całkowitą, zmieniać się o +1
 - każda iteracja musi być niezależna (dane we/wy)
- » Para instrukcji **tic-toc** służy do liczenia czasu
- » parfor jest w toolboksie **Parallel Computing Toolbox**
- » Zagadka, ile kosztuje PCT?

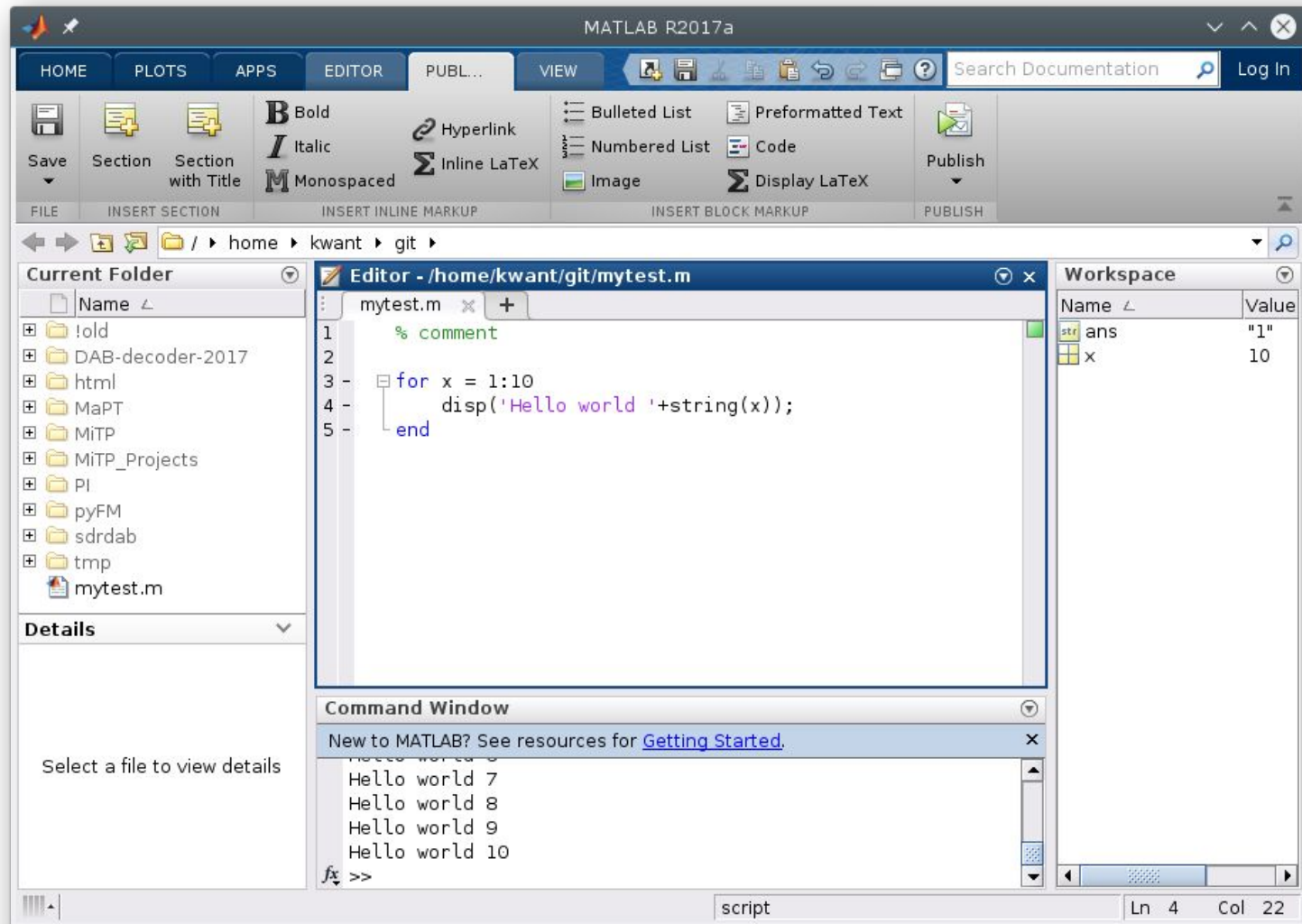
Matlab - pętla parfor

```
n = 200;  
A = 500;  
a = zeros(n);  
tic  
for i = 1:n  
    a(i) = max(abs(rand(1,A)));  
end  
toc  
  
tic  
parfor i = 1:n  
    a(i) = max(abs(rand(1,A)));  
end  
toc
```

- » **parfor** zrównoleglenie obliczeń na poziomie języka !!!
- » **Warunki:**
 - iterator musi być liczbą całkowitą, zmieniać się o +1
 - każda iteracja musi być niezależna (dane we/wy)
- » Para instrukcji **tic-toc** służy do liczenia czasu
- » parfor jest w toolboksie **Parallel Computing Toolbox**
- » Zagadka, ile kosztuje PCT?
EUR 1000 (400/rok)

Jak pisać program

m-pliki + Command Window



MATLAB R2017a

HOME PLOTS APPS EDITOR PUBL... VIEW

Save Section Section with Title Bold Italic Monospaced Hyperlink Inline LaTeX Image Bulleted List Numbered List Image Preformatted Text Code Display LaTeX Publish

FILE INSERT SECTION INSERT INLINE MARKUP INSERT BLOCK MARKUP PUBLISH

Search Documentation Log In

Current Folder: /home/kwant/git

Editor - /home/kwant/git/mytest.m

```

1 % comment
2
3 for x = 1:10
4     disp('Hello world '+string(x));
5 end
  
```

Workspace

Name	Value
ans	"1"
x	10

Command Window

New to MATLAB? See resources for [Getting Started.](#)

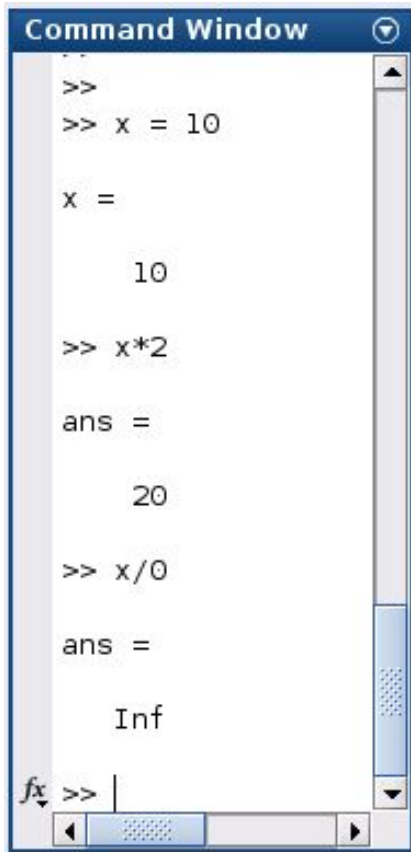
```

Hello world 7
Hello world 8
Hello world 9
Hello world 10
fx >>
  
```

script Ln 4 Col 22

Matlab - interpreter

- » Program można pisać w “Command Window” (nawet pętle i warunki)

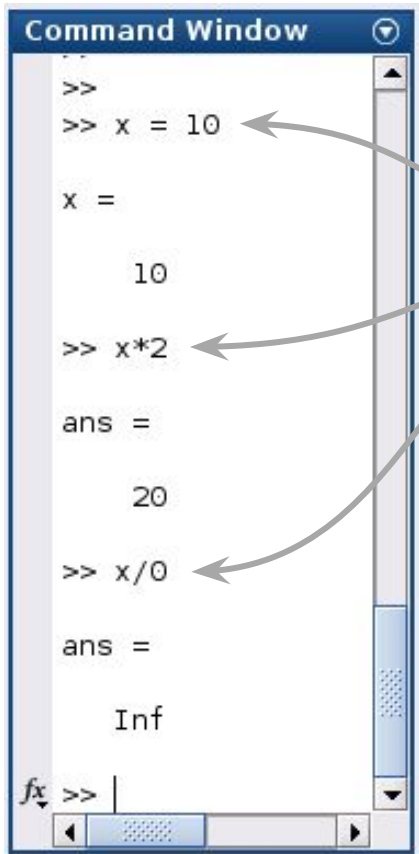


The screenshot shows the MATLAB Command Window interface. The title bar reads "Command Window". The window contains the following text:

```
>>  
>> x = 10  
  
x =  
  
    10  
  
>> x*2  
  
ans =  
  
    20  
  
>> x/0  
  
ans =  
  
    Inf
```

At the bottom, there is a prompt `fx >> |` followed by a vertical cursor. The window has a scroll bar on the right and a status bar at the bottom with navigation buttons.

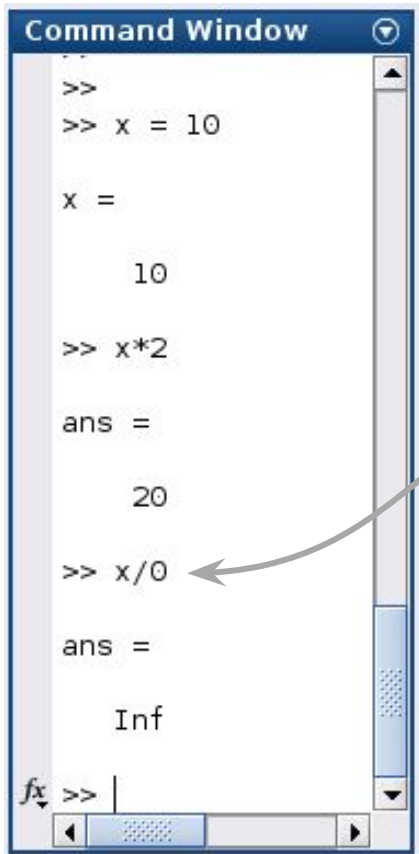
Matlab - interpreter



```
Command Window
>>
>> x = 10
x =
    10
>> x*2
ans =
    20
>> x/0
ans =
    Inf
fx >> |
```

- » Program można pisać w “Command Window” (nawet pętle i warunki)
- » Interpreter: oblicza wynik po wciśnięciu “enter”

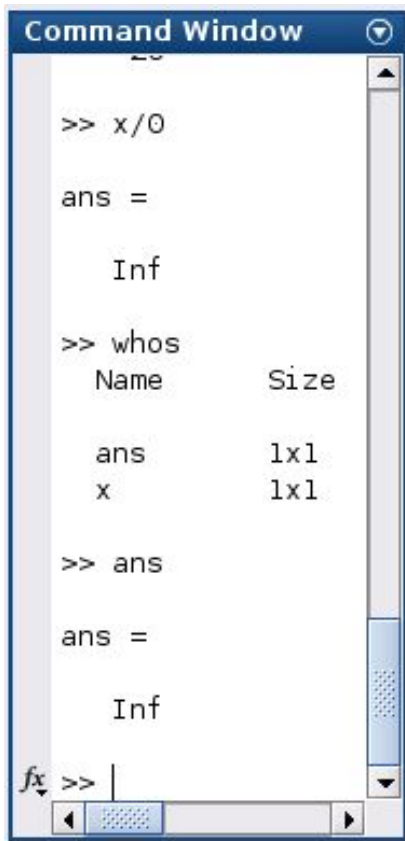
Matlab - interpreter



```
Command Window
>>
>> x = 10
x =
    10
>> x*2
ans =
    20
>> x/0
ans =
    Inf
fx >> |
```

- » Program można pisać w “Command Window” (nawet pętle i warunki)
- » Interpreter: oblicza wynik po wciśnięciu “enter”
- » Matlab jest kulturalny ;-)

Matlab - interpreter



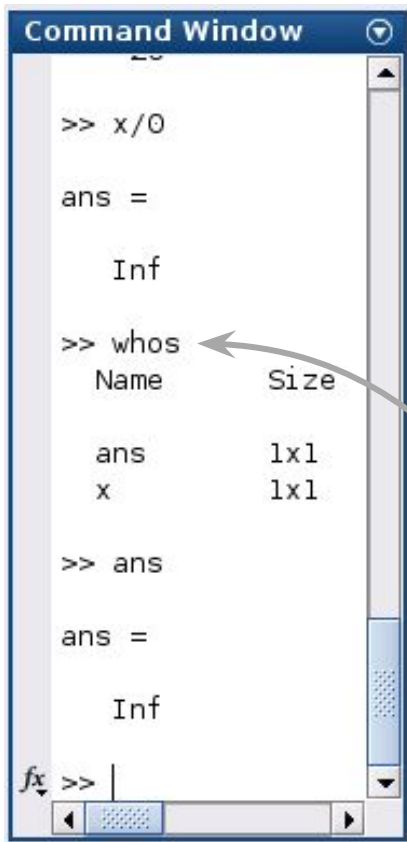
The screenshot shows the MATLAB Command Window interface. It contains the following text:

```
>> x/0  
  
ans =  
  
    Inf  
  
>> whos  
Name      Size  
  
ans       1x1  
x         1x1  
  
>> ans  
  
ans =  
  
    Inf
```

At the bottom, there is a prompt `fx >> |` followed by a cursor.

- » Program można pisać w “Command Window” (nawet pętle i warunki)
- » Interpreter: oblicza wynik po wciśnięciu “enter”
- » Stan obliczeń (zmiennych) jest przechowywany w “workspace”

Matlab - interpreter



```
>> x/0

ans =

    Inf

>> whos
Name      Size

ans       1x1
x          1x1

>> ans

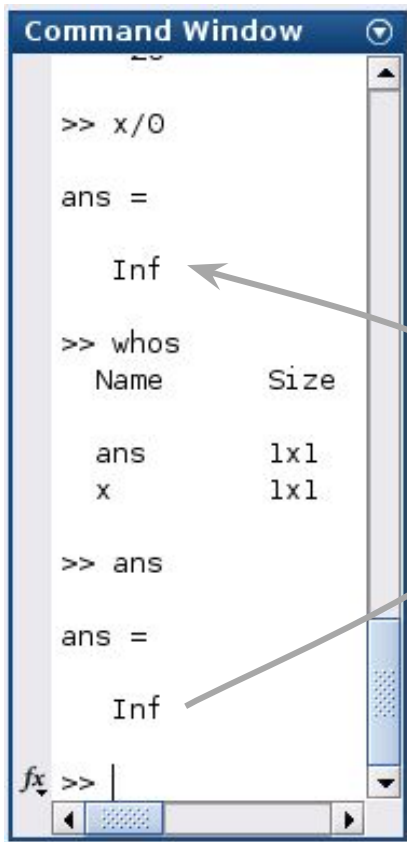
ans =

    Inf
```

The screenshot shows the MATLAB Command Window. It displays the result of a division by zero (`x/0`) as `Inf`. Then, the `whos` command is executed, showing the current workspace variables: `ans` (1x1) and `x` (1x1). The `ans` variable is highlighted in blue. A grey arrow points from the text "status znajduje się w okienku Workspace" in the list to the highlighted `ans` variable.

- » Program można pisać w “Command Window” (nawet pętle i warunki)
- » Interpreter: oblicza wynik po wciśnięciu “enter”
- » Stan obliczeń (zmiennych) jest przechowywany w “workspace”
 - status znajduje się w okienku Workspace
 - można sprawdzić w Command Window

Matlab - interpreter



```
>> x/0

ans =

    Inf

>> whos
Name      Size
ans       1x1
x          1x1

>> ans

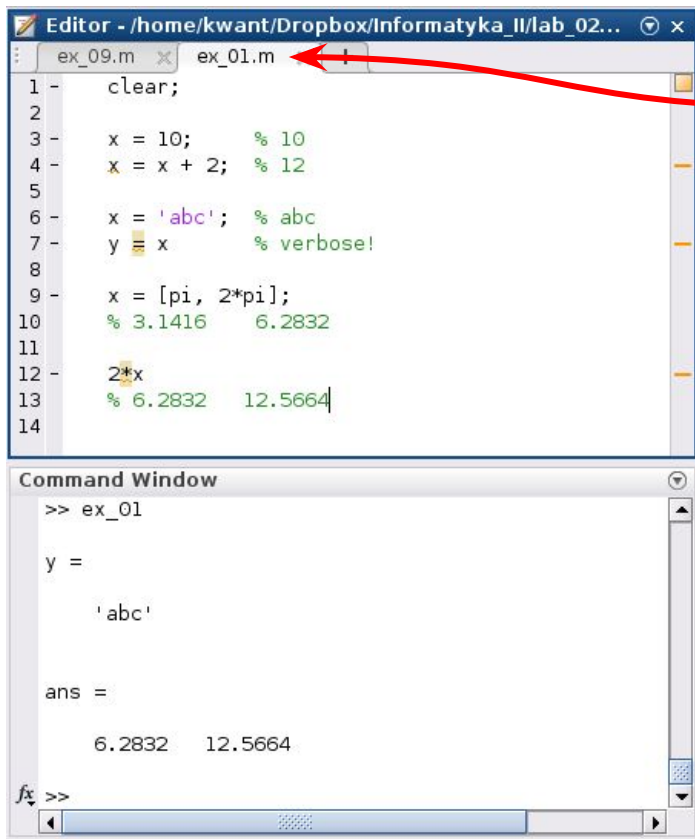
ans =

    Inf
```

The screenshot shows the MATLAB Command Window. It displays the result of a division by zero operation, which is 'Inf'. The 'whos' command is used to check the workspace, showing that 'ans' and 'x' are both 1x1 double arrays. The 'ans' variable is also displayed as 'Inf'. A curved arrow points from the 'Inf' result in the 'whos' output to the 'Inf' result in the Command Window output.

- » Program można pisać w “Command Window” (nawet pętle i warunki)
- » Interpreter: oblicza wynik po wciśnięciu “enter”
- » Stan obliczeń (zmiennych) jest przechowywany w “workspace”
 - status znajduje się w okienku Workspace
 - można sprawdzić w Command Window
- » Zmienna **ans** jest ostatnim wynikiem

Matlab - program



```
Editor - /home/kwant/Dropbox/Informatyka_II/lab_02...
ex_09.m  ex_01.m
1 - clear;
2
3 - x = 10;      % 10
4 - x = x + 2;   % 12
5
6 - x = 'abc';   % abc
7 - y = x        % verbose!
8
9 - x = [pi, 2*pi];
10 % 3.1416      6.2832
11
12 - 2*x
13 % 6.2832      12.5664
14

Command Window
>> ex_01

y =

    'abc'

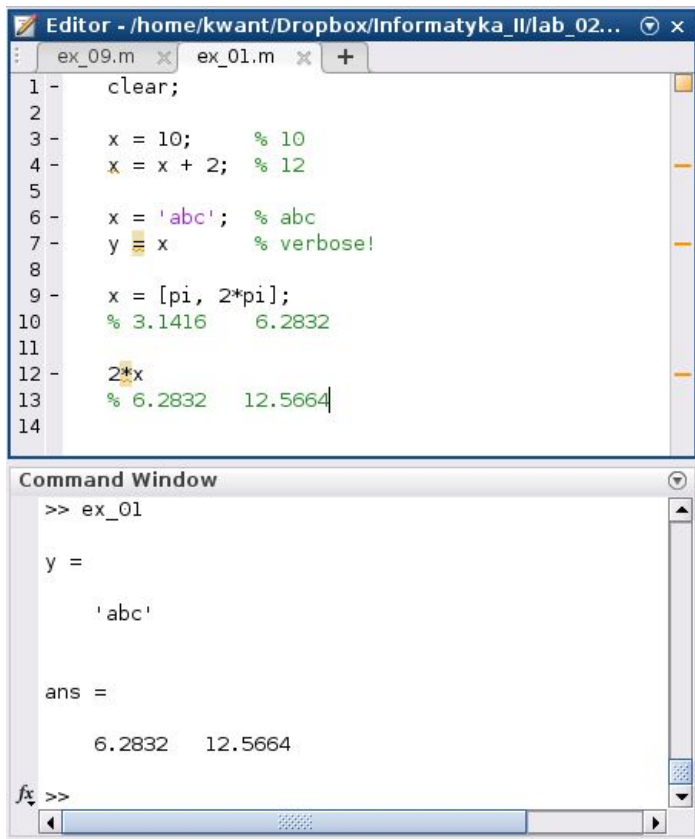
ans =

    6.2832    12.5664

fx >>
```

- » Nietrywialny program tworzony jest w **m-pliku**
- » W pliku *.m zapisywany jest kod źródłowy programu
 - skrypt
 - funkcja

Matlab - program



```
1 - clear;
2
3 - x = 10;      % 10
4 - x = x + 2;   % 12
5
6 - x = 'abc';   % abc
7 - y = x        % verbose!
8
9 - x = [pi, 2*pi];
10 % 3.1416      6.2832
11
12 - 2*x
13 % 6.2832      12.5664
14
```

Command Window

```
>> ex_01

y =

    'abc'

ans =

    6.2832    12.5664

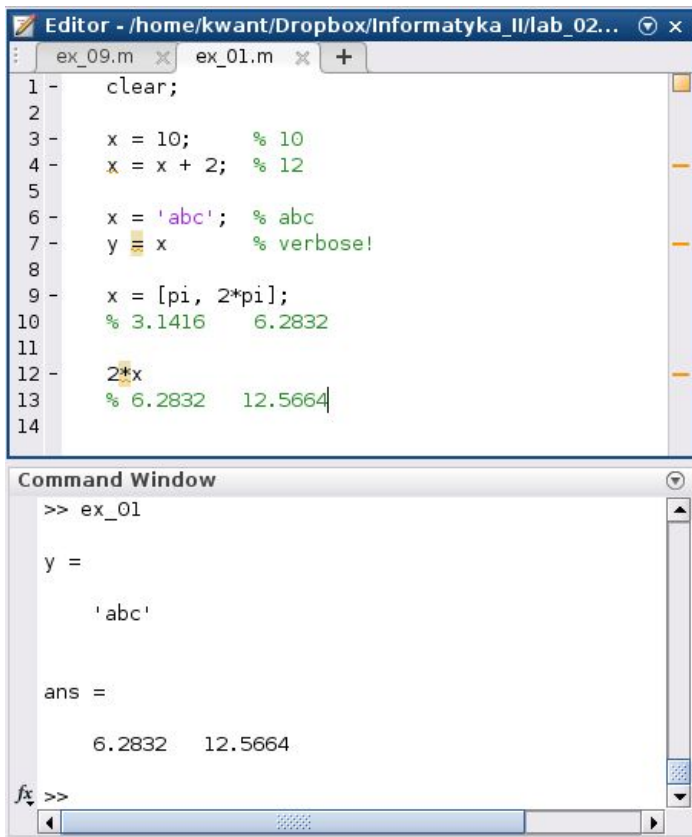
fx >>
```

- » Nietrywialny program tworzony jest w **m-pliku**
- » W pliku *.m zapisywany jest kod źródłowy programu
 - skrypt
 - funkcja
- » Można go uruchamiać w całości, lub częściowo

F5 - w polu edytora

F9 - zaznaczonego fragmentu kodu

Matlab - program



The screenshot shows the MATLAB environment. The Editor window displays a script named 'ex_01.m' with the following code:

```
1 - clear;
2
3 - x = 10;      % 10
4 - x = x + 2;   % 12
5
6 - x = 'abc';   % abc
7 - y = x        % verbose!
8
9 - x = [pi, 2*pi];
10  % 3.1416     6.2832
11
12 - 2*x
13  % 6.2832     12.5664
14
```

The Command Window shows the execution of 'ex_01' with the following output:

```
>> ex_01

y =

    'abc'

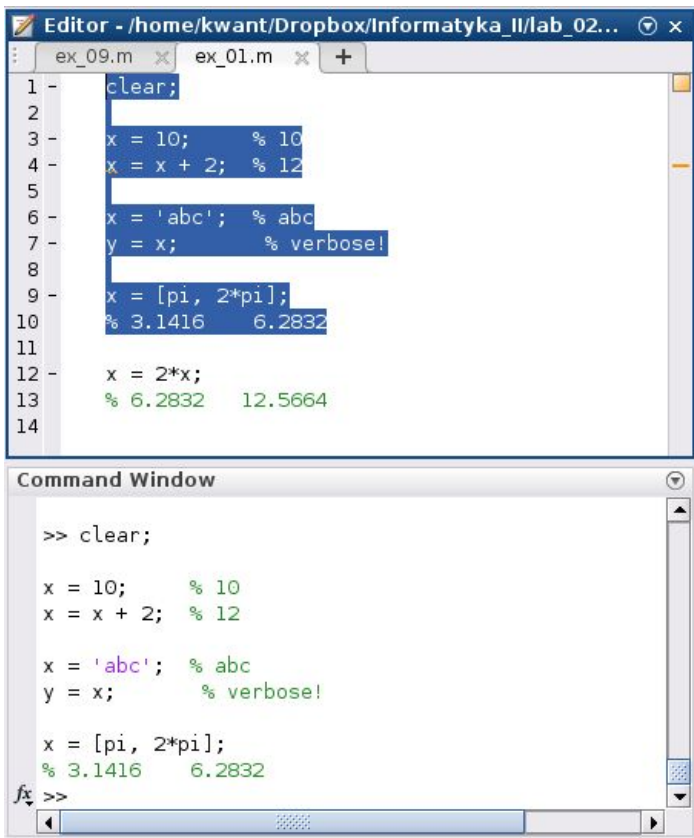
ans =

    6.2832    12.5664
```

The Command Window prompt is 'fx >>'.

- » Nietrywialny program tworzony jest w **m-pliku**
- » W pliku *.m zapisywany jest kod źródłowy programu
 - skrypt
 - funkcja
- » Można go uruchamiać w całości, lub częściowo
- » Można wywoływać:
 - funkcje/skrypt ze skryptu
 - funkcje/skrypt z poziomu Command Window
- » Aktualny stan w “Workspace”

Matlab - program



The screenshot shows the MATLAB environment. The Editor window displays a script file 'ex_01.m' with the following code:

```
1 - clear;
2
3 - x = 10; % 10
4 - x = x + 2; % 12
5
6 - x = 'abc'; % abc
7 - y = x; % verbose!
8
9 - x = [pi, 2*pi];
10 % 3.1416 6.2832
11
12 - x = 2*x;
13 % 6.2832 12.5664
14
```

The Command Window shows the execution of the script, with output displayed for each line of code:

```
>> clear;

x = 10; % 10
x = x + 2; % 12

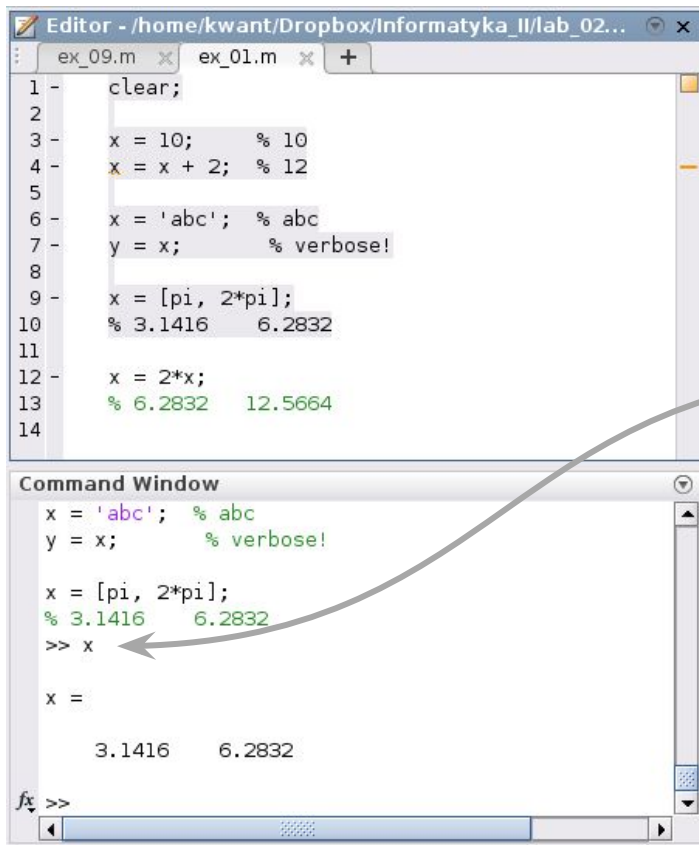
x = 'abc'; % abc
y = x; % verbose!

x = [pi, 2*pi];
% 3.1416 6.2832

fx >>
```

- » Przykład użycia:
 - uruchamiam program (lub część programu - **F9**)

Matlab - program



The screenshot shows the MATLAB environment. The Editor window displays a script with the following code:

```
1 - clear;
2
3 - x = 10; % 10
4 - x = x + 2; % 12
5
6 - x = 'abc'; % abc
7 - y = x; % verbose!
8
9 - x = [pi, 2*pi];
10 % 3.1416 6.2832
11
12 - x = 2*x;
13 % 6.2832 12.5664
14
```

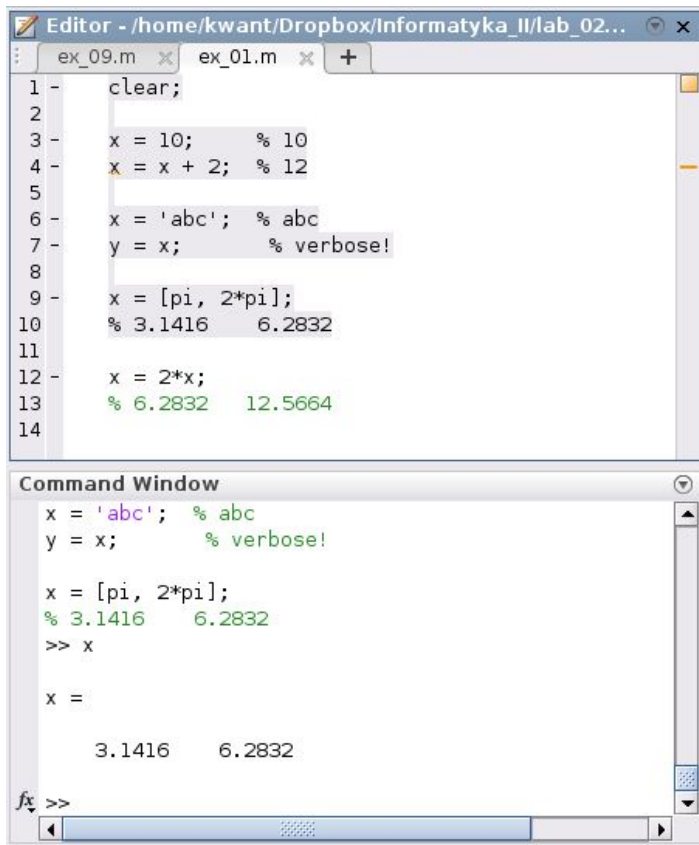
The Command Window shows the execution of the script, with the output of the last command displayed:

```
x =
    3.1416    6.2832
```

An arrow points from the text "np: plot(...), podgląd zmiennych, etc..." to the Command Window output.

- » Przykład użycia:
- uruchamiam program (lub część programu - **F9**)
 - przechodzę do “Command Window” i **kontynuuję pracę** np: **plot(...)**, **podgląd zmiennych**, etc...

Matlab - program



The screenshot shows the MATLAB Editor window with a script named 'ex_09.m'. The script contains the following code:

```

1 - clear;
2
3 - x = 10; % 10
4 - x = x + 2; % 12
5
6 - x = 'abc'; % abc
7 - y = x; % verbose!
8
9 - x = [pi, 2*pi];
10 % 3.1416 6.2832
11
12 - x = 2*x;
13 % 6.2832 12.5664
14

```

Below the editor is the Command Window, which shows the execution of the script:

```

x = 'abc'; % abc
y = x; % verbose!

x = [pi, 2*pi];
% 3.1416 6.2832
>> x

x =

    3.1416    6.2832

```

- » Przykład użycia:
- uruchamiam program (lub część programu - **F9**)
 - przechodzę do “Command Window” i **kontynuuję pracę** np: **plot(...)**, **podgląd zmiennych**, etc...
 - zapisanie “workspace” na dysk, zmiana zmiennych, utworzenie nowych, etc...
 - nie zawsze musisz uruchamiać symulację od początku !!!

Matlab - tips & tricks

- » **F5** - uruchamia cały skrypt (nie funkcje, nie ma arg.)
- » **F9** - uruchamia zaznaczony fragment kodu źródłowego !!!
- » **Ctrl+Enter** - uruchamia sekcję kodu źródłowego
- » **RBM** na wektorze -> plot(...)
- » **Double-click** na wektorze uruchamia podgląd wektora
- » **ctrl-r, ctrl-t** comment, uncomment
- » **ctrl-i**, smart-indent (ctrl-a, ctrl-i)
- » **Debugger**
- » **pause, ctrl-c**

Dziękuję