Współczesne Narzędzia Obliczeniowe

M. Czubenko

Politechnika Gdańska
Wydział Elektroniki Telekomunikacji i Informatyki
Katedra Systemów Decyzyjnych

2 października 2013

Przedmiot: Współczesne narzędzia obliczeniowe

Skład przedmiotu

- Wykład – 15h
- Laboratorium – 45h

Zasady zaliczenia

Skład procentowy

- Wykład – 50%
- Laboratorium – 50%

Zaliczenie wykładu

- Kolokwium (ostatnie zajęcia)
- Projekt (przed końcem wykładu)

Narzędzia obliczeniowe

Definicja

Narzędzia wspomagające wykonywanie obliczeń matematycznych.

Dawno, dawno temu w odległej galaktyce

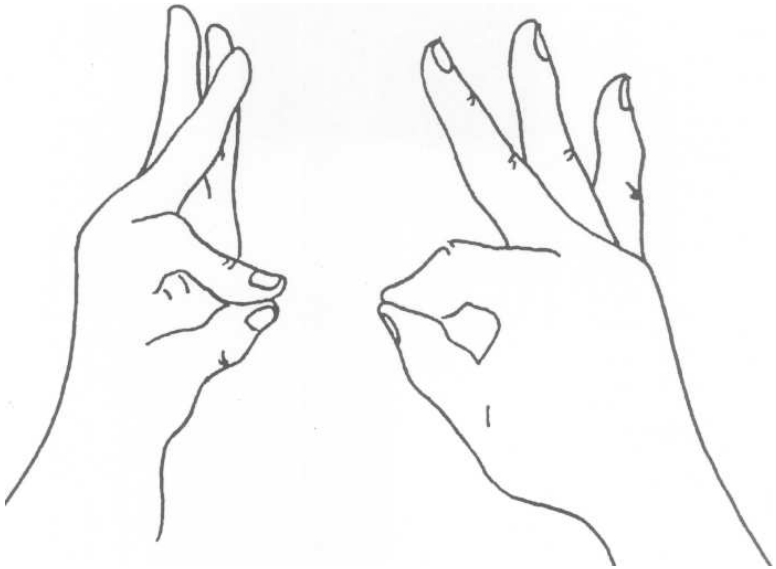
Narzędzia matematyczne

- nacięcia (35 – 20 tys. p.n.e)

Dawno, dawno temu w odległej galaktyce

Narzędzia matematyczne

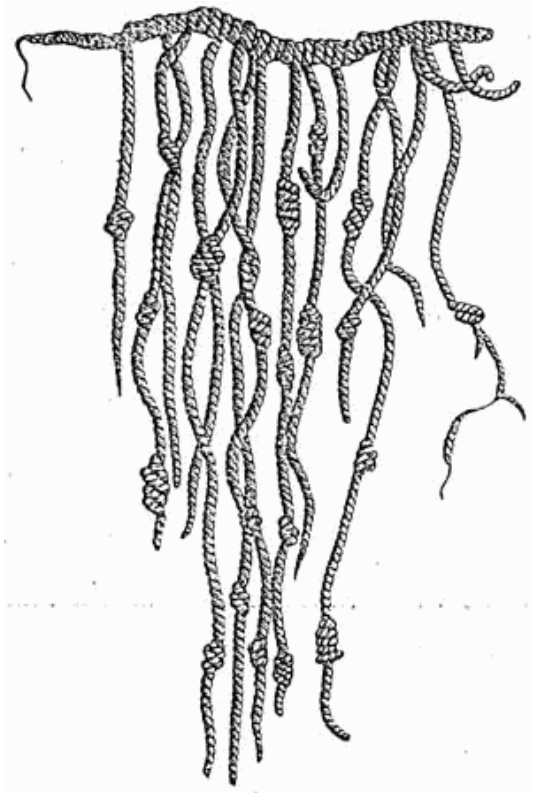
- palce u rąk i nóg oraz inne części ciała



Dawno, dawno temu w odległej galaktyce

Narzędzia matematyczne

- kipu Inków, od XII w n.e



Dawno, dawno temu w odległej galaktyce

Narzędzia matematyczne

- kamyki (abaki, tabliczki)

Antyczne maszyny liczące

Kalkulatory astronomiczne

- Antikythera ($\sim$ 150 p.n.e., Grecja)
- astrolabe ($\sim$ 100 p.n.e., Grecja)
- planisfera ($\sim$ 1000 n.e., Bliski Wschód)
- astronomiczna wieża zegarowa ($\sim$ 1090, Chiny)
- programowalny zegar zamkowy (Al–Jazari, 1206 n.e.)

Mechaniczne maszyny liczące

Kalkulatory matematyczne

- kości Napiera
- suwak logarytmiczny ($\sim$ 1620)
- zegar liczący Schickarda (1623)
- Pascalina – dodawanie i odejmowanie (1642, Pascal)
- ulepszona Pascalina – mnożenie i dzielenie (1672, Leibniz)

Kalkulatory mechaniczne produkowane masowo

- Arithmometr (1820, Charles Xavier Thomas)
- Arithmometr Yazu (1903, Ryoichi Yazu)

Epoka kart perforowanych

Programowalne maszyny

- krosno (1801, Joseph–Marie Jacquard)
- opis silnika analitycznego (1835, Charles Babbage)
- pianino (XIX wiek)
- pierwsze komputery na karty ($\sim$ 1909)
- karty jako nośnik danych (1880, Herman Hollerith) -¿ IBM

Analogowe maszyny liczące

Czysto analogowe

- wodny integrator (1928)
- MONIAC – hydrauliczny komputer

Elektryczne

- Malloc (1941)
- silnik analityczny (1835, Charles Babbage) – opis

Lata dawne

Teoria i praktyka

- ENIAC (1946)
- maszyna Turinga (1936)
- architektura von Neumana
- Z3 (Zuse 1941)
- Harvard Mark I (1944)
- Colossus Mark I (1944)

Lata dawne

Generacje

- 0 - przed pojawieniem się uniwersalnych, elektronicznych maszyn cyfrowych, np. przekaźnikowy Z3
- 1 - budowane na lampach elektronowych, np. XYZ
- 2 - budowane na tranzystorach, np. ZAM 41
- 3 - budowane na układach scalonych małej i średniej skali integracji, np. Odra 1305
- 4 - budowane na układach scalonych wielkiej skali integracji, np. komputer osobisty (PC)
- 5 - projekty o niekonwencjonalnych rozwiązaniach, np. komputer optyczny.

Współczesne programy obliczeniowe

Nazwa	Producent	Cena
LabVIEW	National Instruments	$1249 ($79.95)
Maple	Maplesoft	$1895 ($99)
Mathcad	Parametric Technology Corporation	$1195 ($99)
Mathematica	Wolfram Research	$2495 ($145)
MATLAB	MathWorks	$2450 ($99)
GNU Octave	John W. Eaton	free
Scilab	Scilab Consortium	free

Współczesne biblioteki numeryczne

Nazwa	Język	Cena	System
ALGLIB	C++, C#, Pascal, VBA	free	WL
DotNumerics	C#	free	WMLBU
GNU	C	free	WMLBU
ILNumerics.Net	C#	free	WMLBU
IMSL	C, Java, C#, F, Py	~$700.00	WLU
NAG	C, F	NK	WMLU
NMath	C#	free	W
SciPy	Py	free	WMLBU

Co to jest MATLAB?

MatLab

MATRIX LABORATORY

Zastosowanie

- obliczenia matematyczne
- tworzenie (testowanie, uruchamianie) algorytmów
- modelowanie, symulowanie
- analiza danych
- wizualizacja wyników
- tworzenie aplikacji

Historia MATLABa

Na początku był FORTRAN...

- Linpack
- Eispack
- program dla studentów w FORTRANIE (C. Moler)

Przejście do C

- GUI
- JACKPAC (C. Moler, S. Bangert i J. Little)
- MATLAB oparty na JACKPACu (1985)
- MATLAB oparty na LAPACKu (2005)

Składniki MATLABa

Moduły

- Desktop Tools and Development Environment
- Mathematical Function Library
- The Language
- Graphics
- External Interfaces

Charakterystyka MATLABa

Składnia

- uproszczone C

Zmienne

- bazują głównie na macierzach

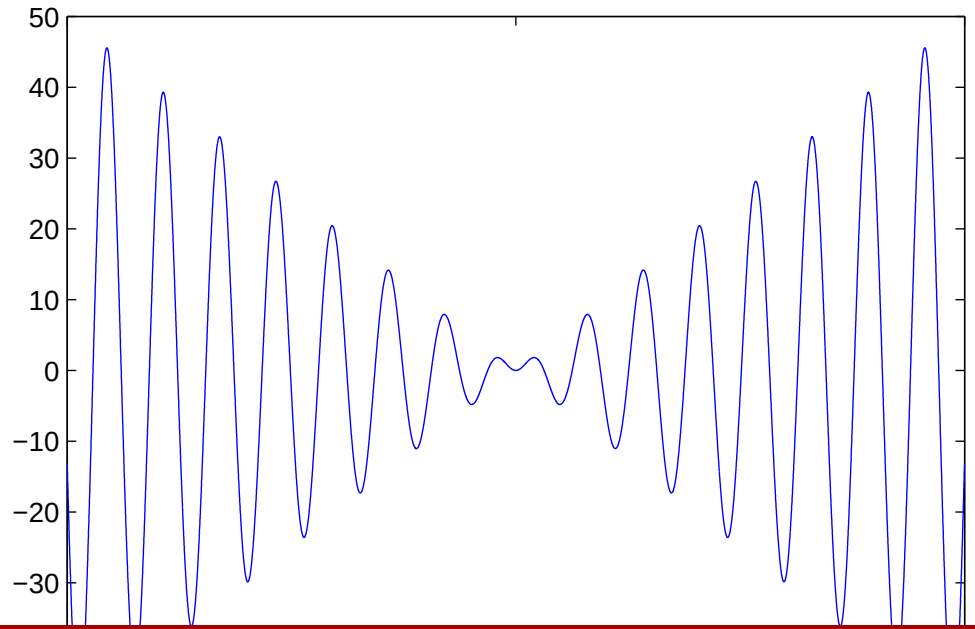
Pliki

- .mex – biblioteki C i Fortrana
- .fig – wykresy
- .mat – dane
- .m – skrypty

Grafika

Wykresy

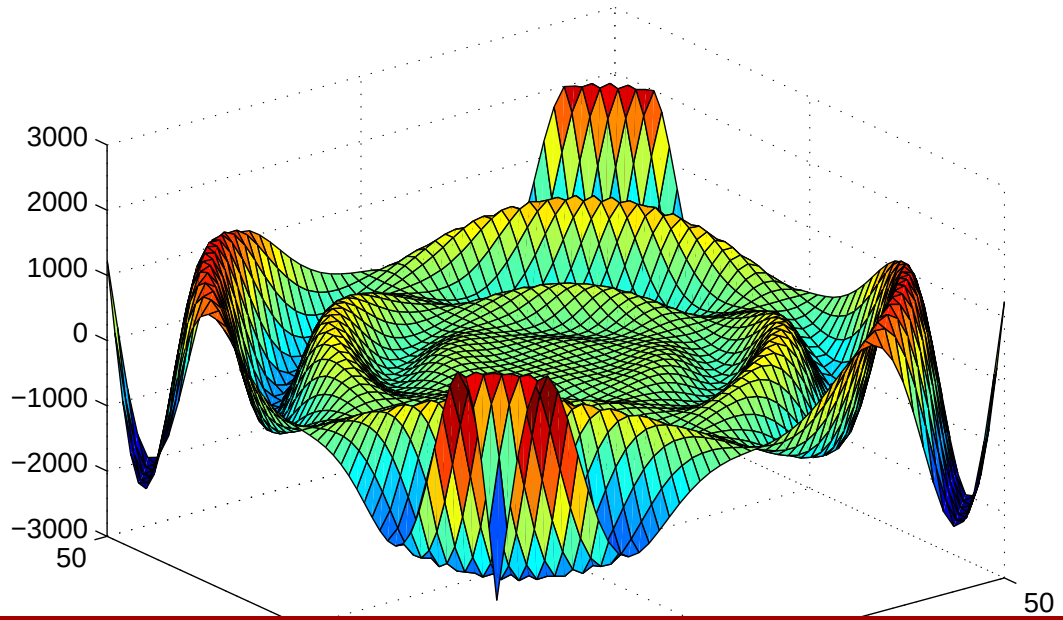
- 2D



Grafika

Wykresy

- 3D



Alternatywy

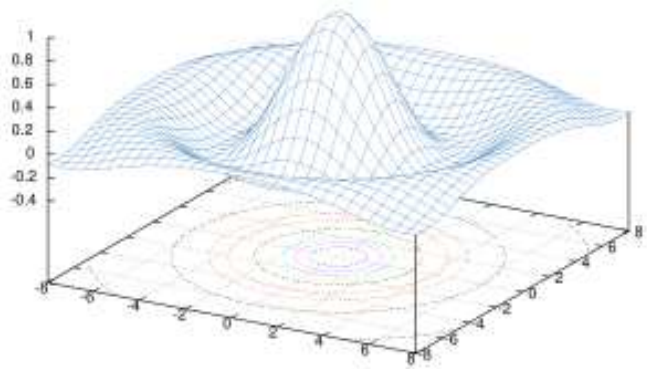
Wolny MATLAB

- SciLab

Alternatywy

Wolny MATLAB

- GNU Octave



Wersje MATLABa

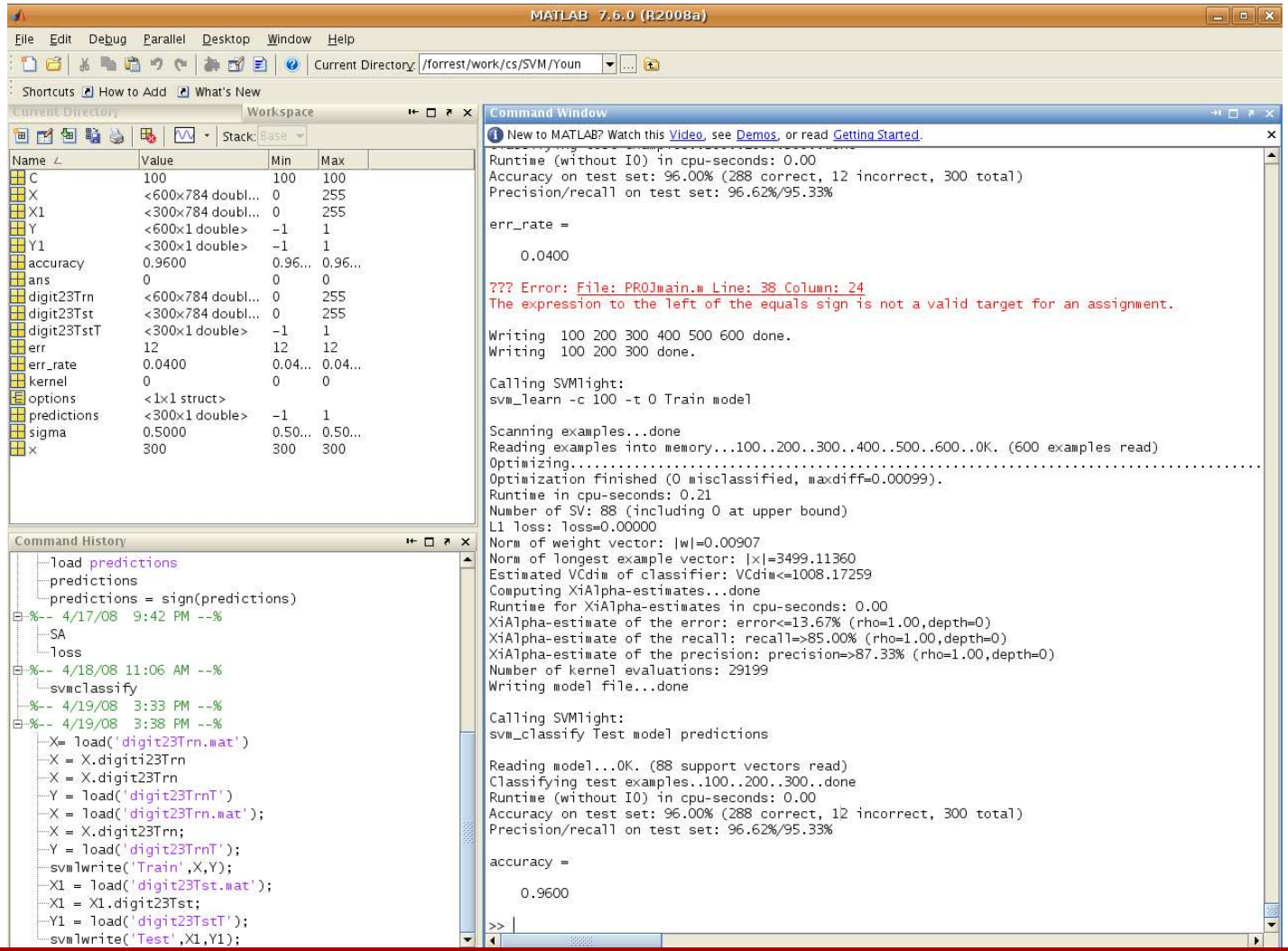
Wersja	Nazwa	Rok
MATLAB 6.5.2	R13SP2	2003
MATLAB 7	R14	2004
MATLAB 7.0.1	R14SP1	
MATLAB 7.0.4	R14SP2	2005
MATLAB 7.1	R14SP3	
MATLAB 7.2	R2006a	2006
MATLAB 7.3	R2006b	
MATLAB 7.4	R2007a	2007
MATLAB 7.5	R2007b	
MATLAB 7.6	R2008a	2008
MATLAB 7.7	R2008b	
MATLAB 7.8	R2009a	2009

Dodatki MATLABa

Toolboxy

- Controll System Toolbox
- Fuzzy Logic Toolbox
- Genetic Algorithm ...
- Image Processing Toolbox
- Mapping Toolbox
- Neural Network Toolbox
- Parallel Computing
- Symbolic Math Toolbox
- Partial Differential Equation Toolbox
- Simulink

Okienko MATLABa



Plecenia sterujące

Porządki

clc	czyszczenie widoku
clear	czyszczenie zmiennych
clear x	czyszczenie zmiennej x

Katalogi

cd x	przejście do katalogu x
dir	wyświetlenie zawartości katalogu
delete x	usunięcie pliku x

Plecenia sterujące

Pliki

save *x*	zapisz wszystkie zmienne do pliku *x*
save *x* **A B**	zapisz zmienne **A B** do pliku *x*
load *x*	załadowanie zmiennych z pliku *x*

Odpytywanie

who	wyświetlenie zmiennych
whos	wyświetlenie zmiennych i ich wymiarów
help	wywołanie pomocy
help *x*	wywołanie pomocy o poleceniu *x*

Stałe w MATLABie

Typowe stałe

i/j	jednostka urojona
pi	3.1416; sin(pi)!=0;
ans	wynik ostatniego działania
Inf	nieskończoność, np. 1/0, exp(1000)...
NaN	wynik absurdalny, nie liczba; np. Inf-Inf, 0/0...

Uwaga

Istnieje możliwość nadpisania stałych Inf i NaN!

Zmienne – podstawy

Typy zmiennych

double, single typy zmiennoprzecinkowe; domyślny typ MATLABa;

int*x*, uint*x* typy całkowite; $x \in (8, 16, 32, 64)$;

char typ znakowy (ASCII albo UNICODE);

logical typ logiczny;

function_handle wskaźnik na funkcję;

struct struktura;

cell przechowuje tablice różnych typów; ułatwia manipulację;

Macierze – podstawy

Przykłady

wektor » vector = [0, 1, 4, 5]

kolumna » column = [0; 1; 4; 5]

macierz » array = [0, 1; 4, 5]

Uwaga

Macierz jest domyślnym typem tablicowym w MATLABie (nie można stworzyć sth = [2, 0, 1; 4, 5])

Macierze – tworzenie

Przykłady

wpisanie wartości » array = [0, 1; 4, 5]

wygenerowanie wartości » array = [1:2; 4:5]

złożenie wektorów » array = [A; B]

eye(*a*,*b*)) macierz jednostkowa o rozmiarze *a*x*b*

zeros(*a*,*b*)) macierz zerowa o rozmiarze *a*x*b*

ones(*a*,*b*)) macierz jedynkowa o rozmiarze *a*x*b*

rand(*a*,*b*)) macierz losowa o rozmiarze *a*x*b*

magic(*a*)) kwadrat magiczny o rozmiarze *a*

Macierze – operacje

Typy zmiennych

- » A(1,:) pobranie 1-go wiersza macierzy
- » A(1,2:3) pobranie kolumn od 2 do 3 z 1-go wiersza macierzy
- » A(:,1) pobranie 1-ej kolumny macierzy
- » A(1,1) pobranie elementu macierzy

Macierze – operacje

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Przykłady

» A(1,:) **ans** $= \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$

» A(1,2:3) **ans** $= \begin{bmatrix} 2 & 3 \end{bmatrix}$

» A(:,1) **ans** $= \begin{bmatrix} 1 \\ 4 \\ 7 \end{bmatrix}$

» A(1,1) **ans** $= [1]$

» A(3) **ans** $= [7]$

Macierze – operacje skalarne

Typy zmiennych

- » A+*a* dodanie do każdego elementu *a*
- » A**a* mnożenie każdego elementu przez*a*
- » A(*war*) pobranie elementów odpowiadających zapytaniu logicznemu *war*
- » find(*war*) pobranie indeksów elementów odpowiadających zapytaniu logicznemu *war*

Macierze – operacje skalarne

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{a=2}$$

Przykłady

» A+a $\quad ans = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 11 \end{bmatrix}$

» A*a $\quad ans = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \\ 14 & 16 & 18 \end{bmatrix}$

» A(A>2 & A<5) $\quad ans = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$

Macierze – operacje macierzowe

Typy zmiennych

- » A+B dodawanie macierzy
- » A*B mnożenie macierzy
- » A/B mnożenie przez odwrotność ($A * B^{-1}$)
- » A^(a) potęgowanie macierzy

Macierze – operacje skalarne

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 4 & 3 \\ 2 & 1 \end{bmatrix}$$

Przykłady

» A+B $\quad ans = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$

» A*B $\quad ans = \begin{bmatrix} 8 & 5 \\ 20 & 13 \end{bmatrix}$

» A/B $\quad ans = \begin{bmatrix} 1.5 & -2.5 \\ 2.5 & -3.5 \end{bmatrix}$

» A^-1 $\quad ans = \begin{bmatrix} -2 & 1 \\ 1.5 & -1.5 \end{bmatrix}$

Macierze – przydatne polecenia

Typy zmiennych

- » diag(A) pobieranie przekątnej macierzy
- » size(A) pobieranie wymiarów macierzy
- » length(A) pobieranie większego wymiaru macierzy
- » inv(A) odwracanie macierzy
- » det(A) wyznacznik macierzy
- » eig(A) wartości własne macierzy
- » poly(A) współczynniki wielomianu charakterystycznego macierzy
- » rank(A) rząd macierzy
- » min(A) wiersz złożony z minimów kolumn macierzy

Macierze – przydatne polecenia

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Przykłady

» diag(A) $ans = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$

» size(A) $ans = \begin{bmatrix} 2 & 2 \end{bmatrix}$

» length(A) $ans = 2$

» inv(A) $ans = \begin{bmatrix} -2 & 1 \\ 1.5 & -1.5 \end{bmatrix}$

» det(A) $ans = -2$

» eig(A) $ans = \begin{bmatrix} -0.3723 \end{bmatrix}$

Macierze – przydatne polecenia

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Przykłady

» poly(A) $ans = \begin{bmatrix} 1 & -5 & -2 \end{bmatrix}$

» rank(A) $ans = 2$

» min(A) $ans = \begin{bmatrix} 1 & 2 \end{bmatrix}$

» max(A) $ans = \begin{bmatrix} 3 & 4 \end{bmatrix}$

Macierze – transpozycja

Typy zmiennych

- » A' transpozycja klasyczna
- » A.' sprzężenie hermitowskie

Macierze – transpozycja

$$\mathbf{A} = \begin{bmatrix} 1+j & 2+j \\ 3+j & 4+j \end{bmatrix}$$

Przykłady

» A' $ans = \begin{bmatrix} 1-j & 3-j \\ 2-j & 4-j \end{bmatrix}$

» A.' $ans = \begin{bmatrix} 1+j & 3+j \\ 2+j & 4+j \end{bmatrix}$

Zmienne tekstowe – Stringi

Definicja

Zmienna tekstowa jest wektorem znaków. Istnieje możliwość konwersji na kod ASCII double(s) i odwrotnej char(a).

s = 'sin(pi)';

Ciekawostki

» eval(s) 1.2246e-016

Uwaga

sin(π) w MATLABie jest bliski zeru, ze względu na niedokładność stałej pi.

Struktury

Tworzenie

» a = struct('liczba', 0, 'tekst', 'test'); stworzenie struktury zawierającej pola **liczba** i **tekst**

Odwoływanie się

» a.liczba 0

» a.tekst 'test'

Zaawansowane struktury – cell arrays

Różnice

zwykła macierz » array $= [0, 1; 4, 5]$

macierz komórkowa » cellarray $= \{0, 1; 4, 5\}$

Przykład

cellarray = { 1, 2, 'a', 'abc'; rand(3, 2), magic(3), eye(3), 'śmieć'}

$$\begin{bmatrix} [1] & [2] & \text{'a'} & \text{'abc'} \\ [3x2double] & [3x3double] & [3x3double] & \text{'śmieć'} \end{bmatrix}$$

Zalety

- każdy element może być innego typu

Ciekawostki

Wprowadzanie wartości z linii poleceń

liczby `x=input('Podaj wartość x: ')`

tekst `s=input('Podaj wyrażenie x: ','s')`

Zmienne analityczne

`syms x` utworzenie zmiennej

`% tekst` komentarz

Operatory

Lista operatorów

==	równe
~=	różne
<	mniejsze
>	większe
<=	mniejsze równe
>=	większe równe
&	i
|	lub

Instrukcje warunkowe

if

```
if warunek
    instrukcje;
  end;
```

Instrukcje warunkowe

if

```
if warunek
    instrukcje;
 else
    instrukcje;
end;
```

Instrukcje warunkowe

if

```
if warunek
    instrukcje;
elseif
    instrukcje;
 end;
```

Instrukcje warunkowe

if

```
if warunek
    instrukcje;
elseif
    instrukcje;
else
    instrukcje;
end;
```

Instrukcje warunkowe

case

```
switch wyrażenie
case wartość 1
    instrukcje;
case wartość 2
    instrukcje;
...
case (wartość 8, wartość 9)
    instrukcje;
otherwise
    instrukcje;
end;
```

Pętle

for

```
for zmienna = macierz
    instrukcje;
end;
```

while

```
while warunek;
    instrukcje;
end;
```

Pętle

try – catch

```
try
    instrukcje;
catch
    instrukcje;
end;
```

Uwaga

We wszystkich petlach dozwolone jest używanie instrukcji break i continue

Przegląd funkcji

Matematyczne – trygonometria

sin(x)	sinus
cos(x)	cosinus
tan(x)	tangens
cot(x)	cotangens
a*trg*(x)	arcus
*trg*h(x)	funkcja hiperboliczna
a*trg*h(x)	arcus hiperboliczny

Przegląd funkcji

Matematyczne – podstawy

sqrt(x)	pierwiastek kwadratowy
exp(x)	eksponenta
log(x)	logarytm naturalny
log2(x)	logarytm o podstawie 2
log10(x)	logarytm o podstawie 10

Przegląd funkcji

Matematyczne – zsepolone

abs(x)	macierz modułów elementów macierzy x
angle(x)	macierz argumentów elementów macierzy x
real(x)	macierz części rzeczywistych elementów macierzy x
imag(x)	macierz części urojonych elementów macierzy x
conj(x)	macierz o elementach sprzężonych z elementami macierzy x

Przegląd funkcji

Matematyczne]- dodatkowe

abs(x)	wartość bezwzględna liczby
ceil(x)	zaokrąglanie w górę
floor(x)	zaokrąglanie w dół
fix(x)	zaokrąglanie zbliżające do zera
round(x)	zaokrągla elementy macierzy x do najbliższej liczby całkowitej
rand(n)	macierz o wymiarze n wypełniona liczbami losowymi od 0 do 1
rem(x,y)	reszta z dzielenia odpowiadających sobie elementów macierzy x i y

Tworzenie funkcji

Skrypt

Skrypt matlabowy jest zbiorem poleceń zapisanych w m-pliku.

Funkcja

Funkcja własna użytkownika to skrypt o takiej samej nazwie, jak nazwa funkcji.

Funkcja

```
function[wyn1, wyn1, ...] = nazwa(arg1, arg2, ...)
    instrukcje; return;
```

Tworzenie funkcji

Dlaczego warto używać funkcji?

- funkcja ma własny workspace
- umożliwia powielanie wywołań
- umożliwia zmienną ilość argumentów i wyjść
- umożliwia debugowanie

Tworzenie funkcji

Przykład – Silnia

```
function[wyn] = silnia(n);
    wyn = 1;
    for i=1:n
        wyn=wyn*i;
    end;
```

Tworzenie funkcji

Przykład – Silnia rekurencyjna

```
functionwyn = silniaR(n);
    if (n¡0)
        error('Niepoprawny argument');
    elseif (n==0)
        wyn = 1;
    else
        wyn = n*silniaR(n-1);
    end;
```

Rysowanie wykresów klasycznych

Narzędzie – plot

plot(x)	rysowanie wektora **x** w zależności od indeksów
plot(x,y)	rysowanie wektora **x** w zależności od wektora **y**
plot(x,y,str)	rysowanie wektora **x** w zależności od wektora **y** z ustawieniami **str**

Plot – opcje

Kolory

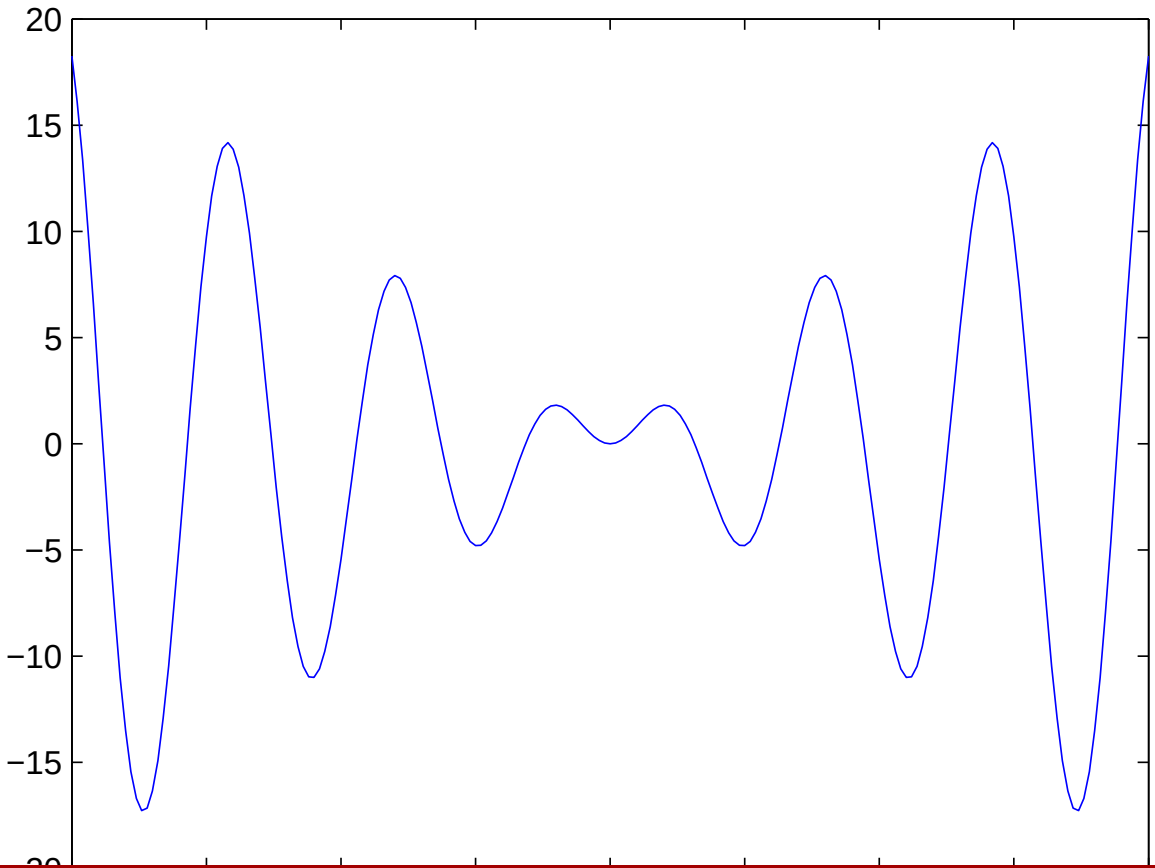
- y yellow
- m magenta
- c cyan
- r red
- g green
- b blue
- w white
- k black

Style

- . punkt
- x krzyżyk
- o kółko
- + plusik
- - myślnik
- * gwiazdeczka
- : kropkowany
- -. kropka–kreska
- - - kreskowany

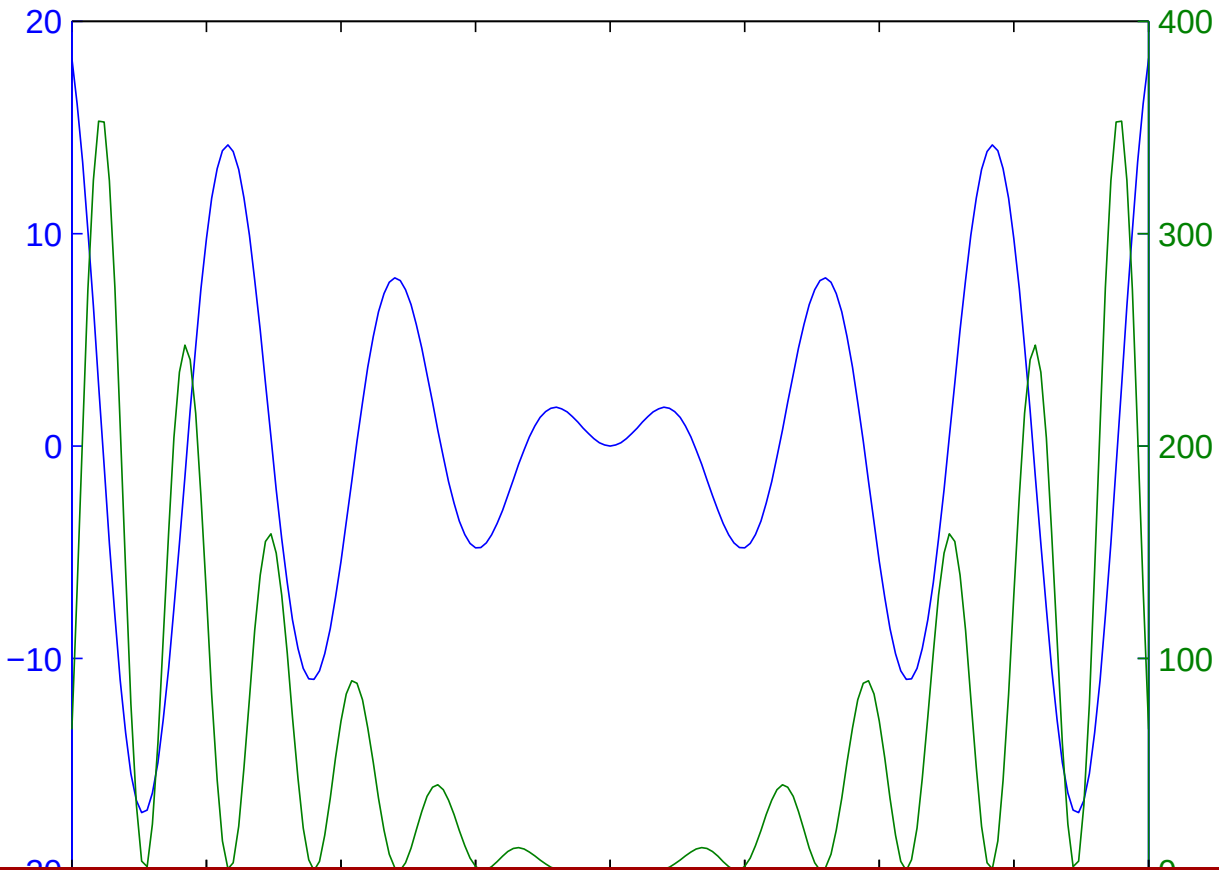
Wykres liniowy

- plot(x,y)



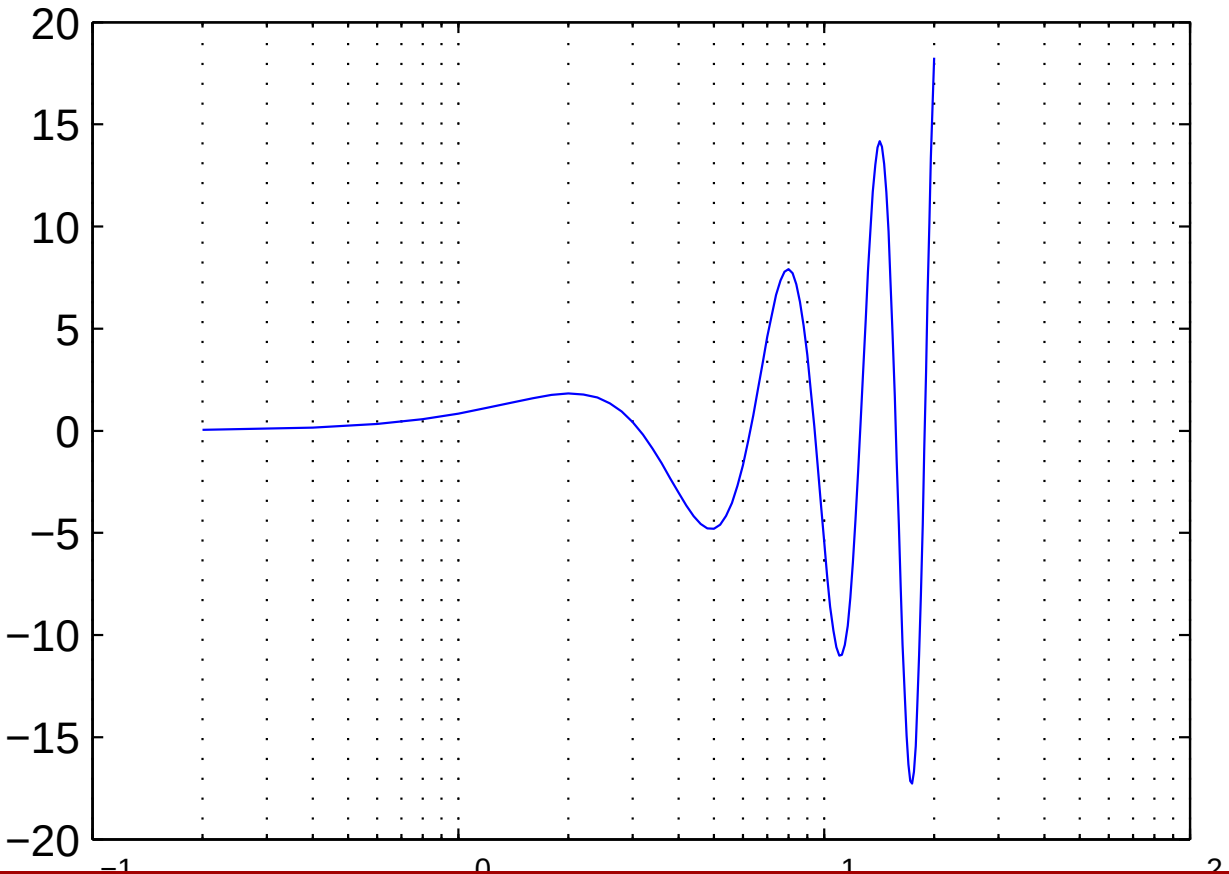
Wykres liniowy – dwupodziałkowy

- plotyy(x1,y1,x2,y2)



Wykres logarytmiczny – oś-x

- semilogx(x,y)

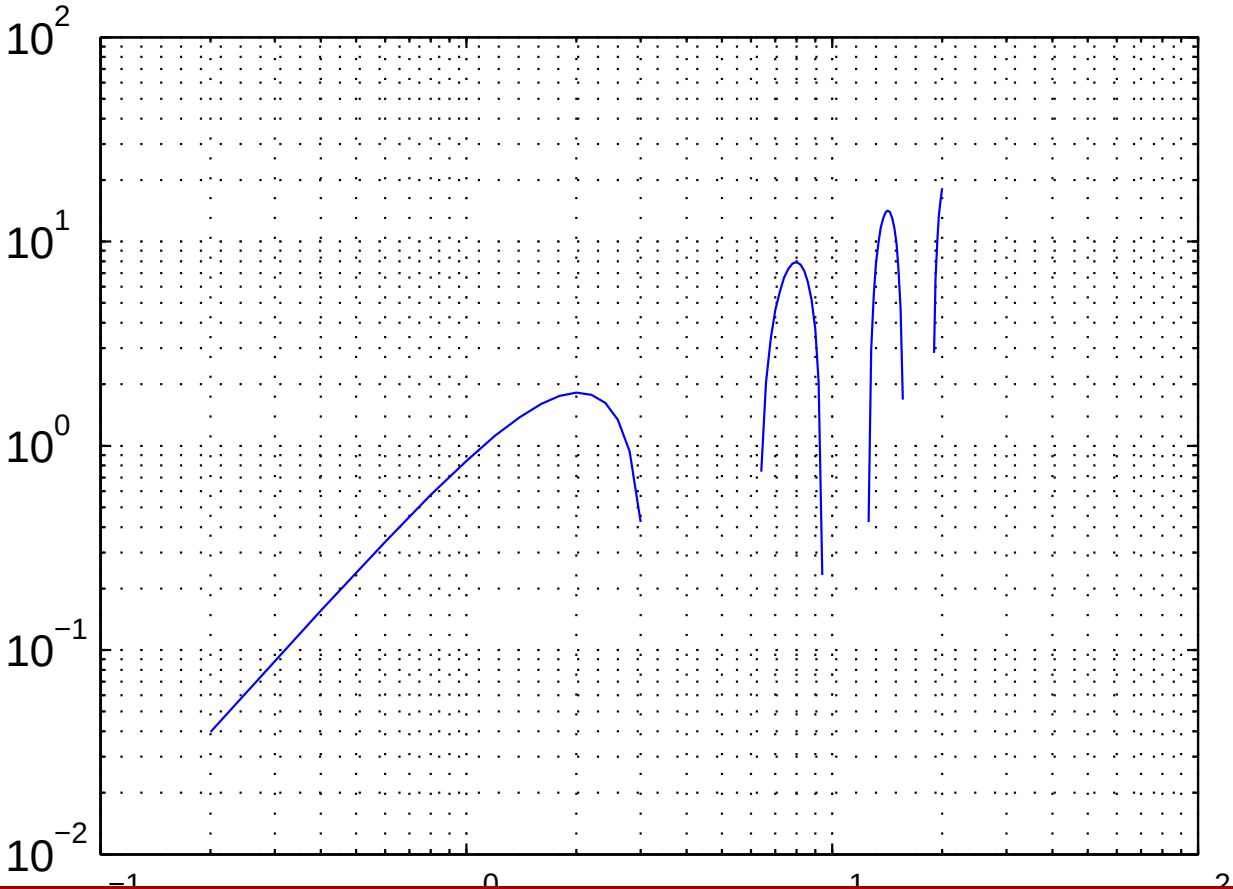


Wykres logarytmiczny – oś-y

- semilogy(x,y)

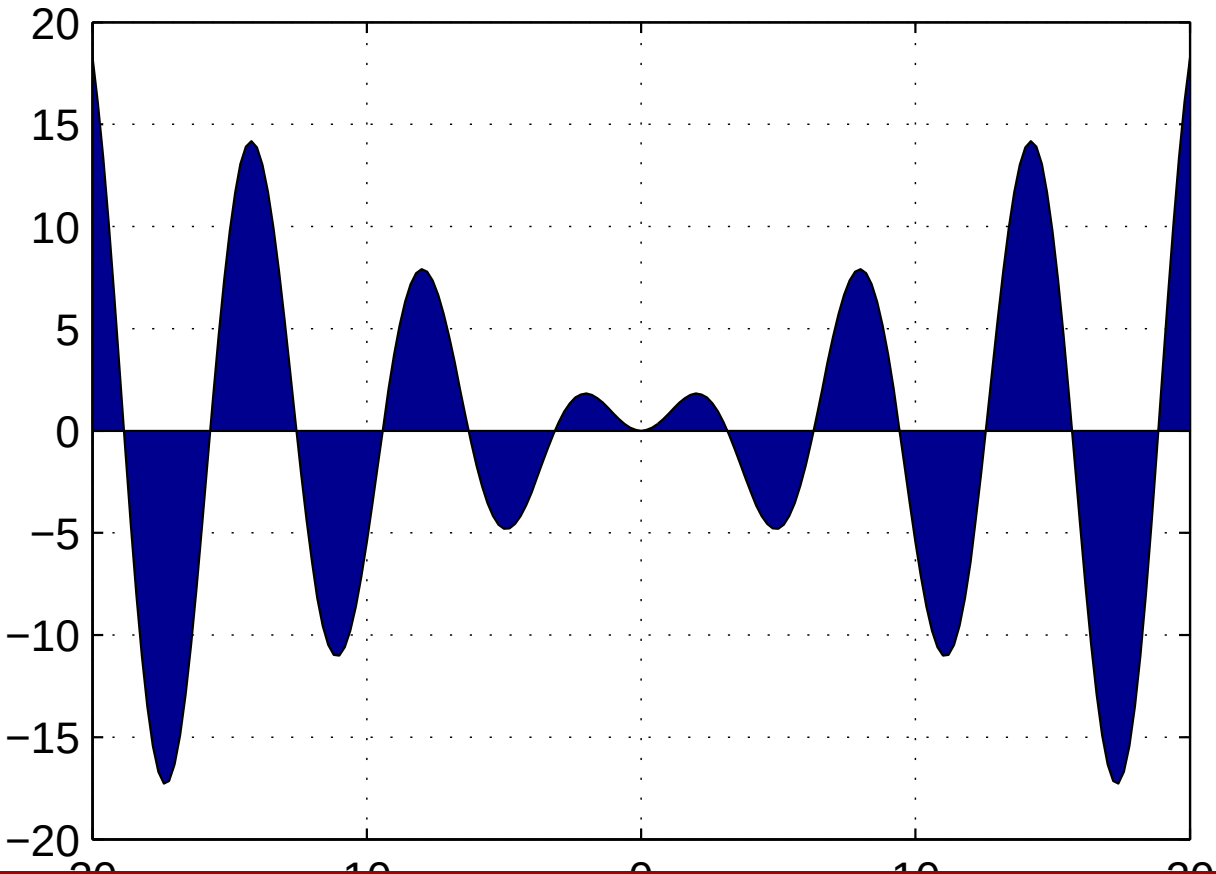
Wykres logarytmiczny

- loglog(x,y)



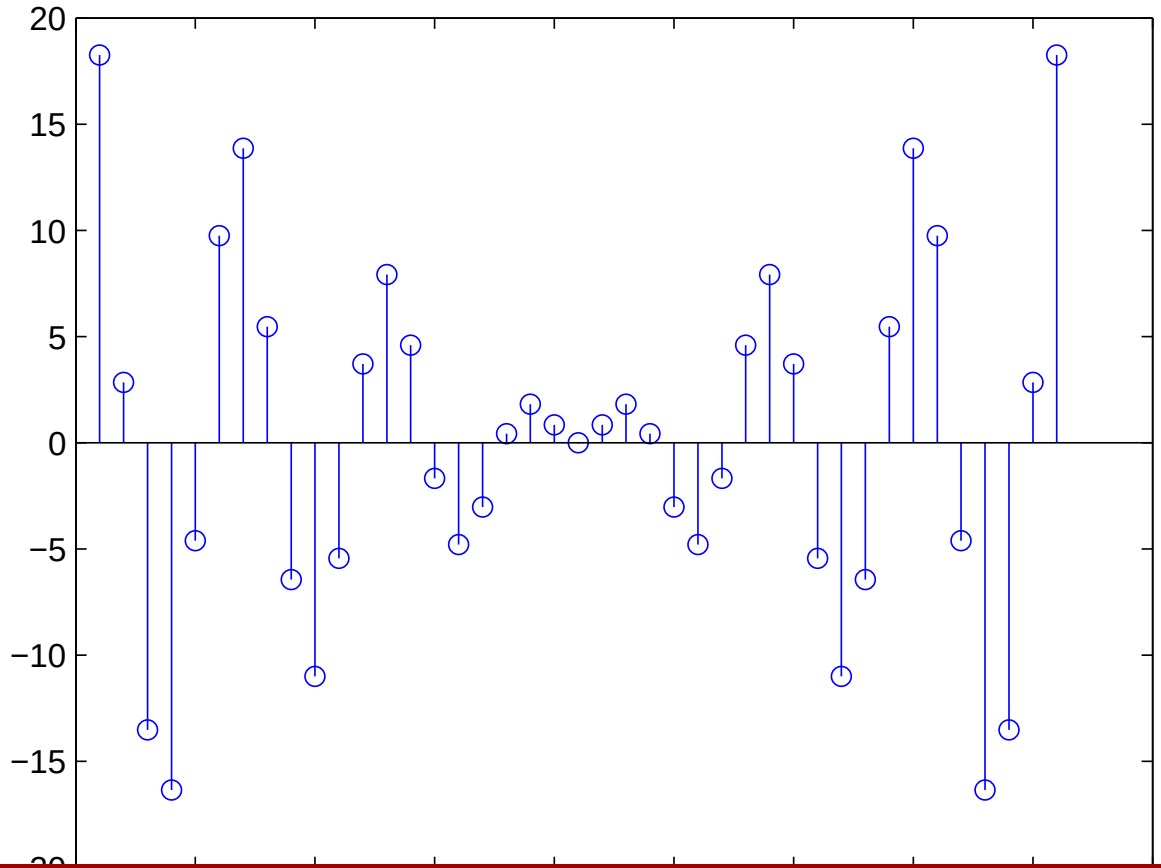
Wykres płaszczyzn

- area(x,y)



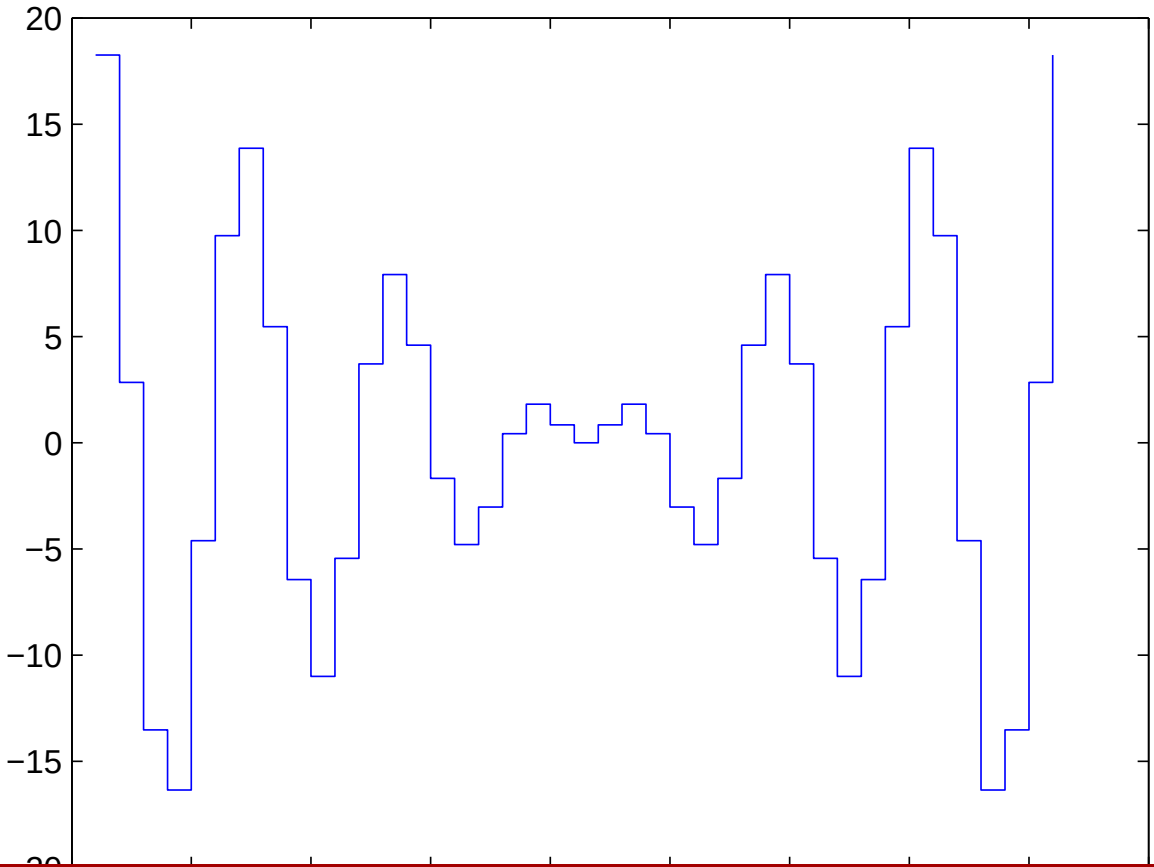
Wykres punktowy – impulsowy

- stem(x,y)



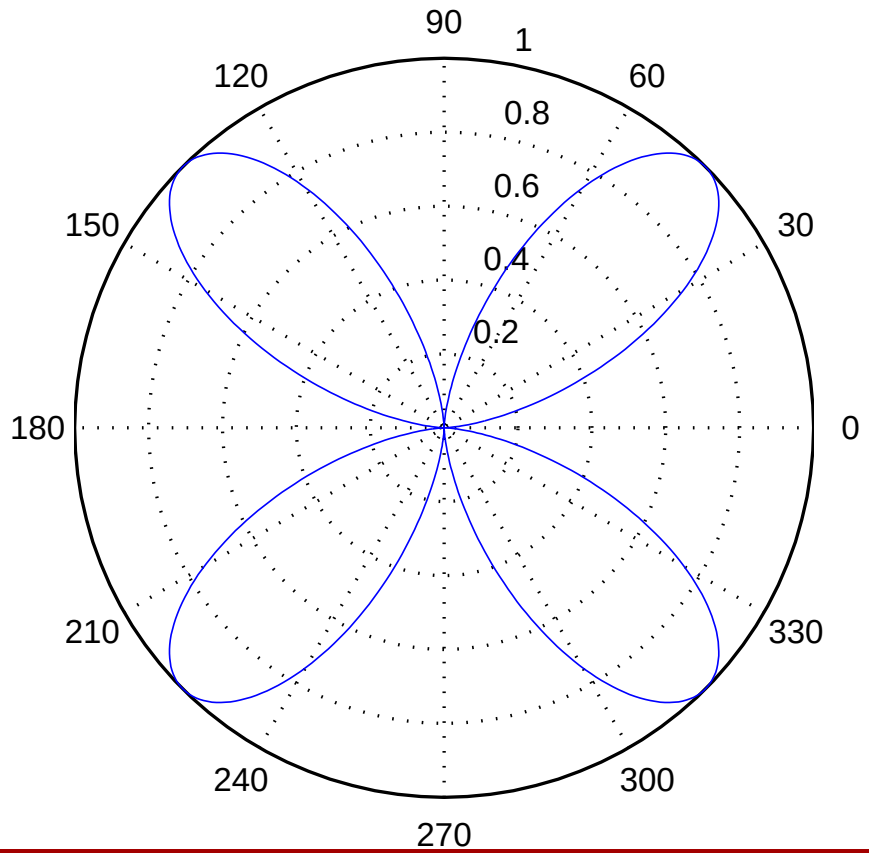
Wykres schodkowy

- stairs(x,y)



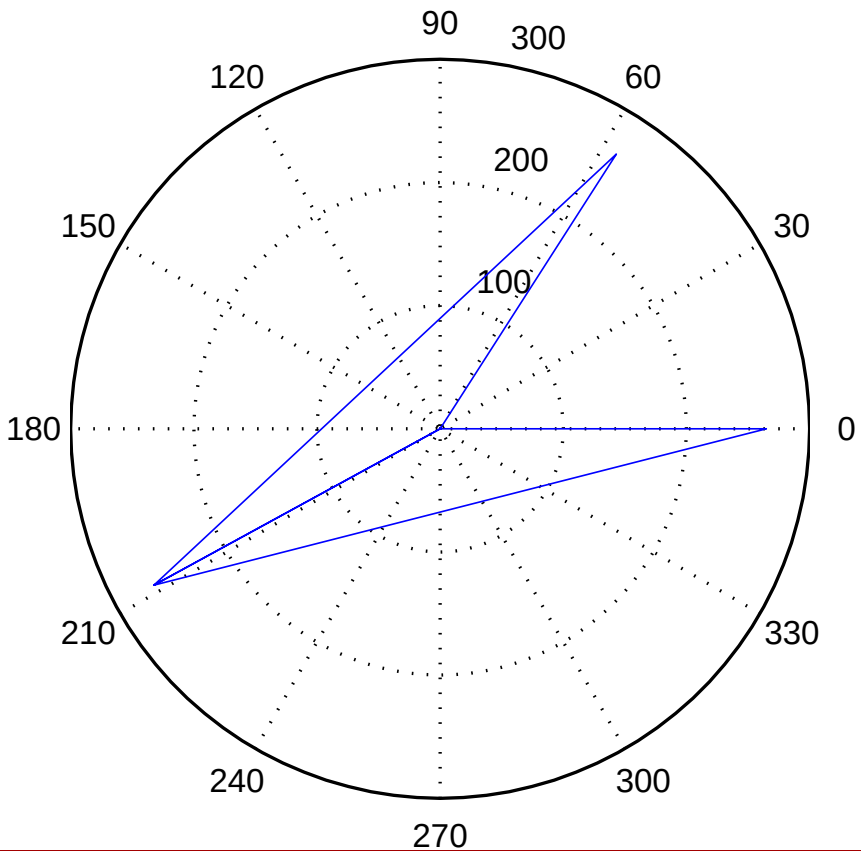
Wykres biegunowy

- polar(x,y)



Wykres biegunowy – histogram

- rose(x,y)



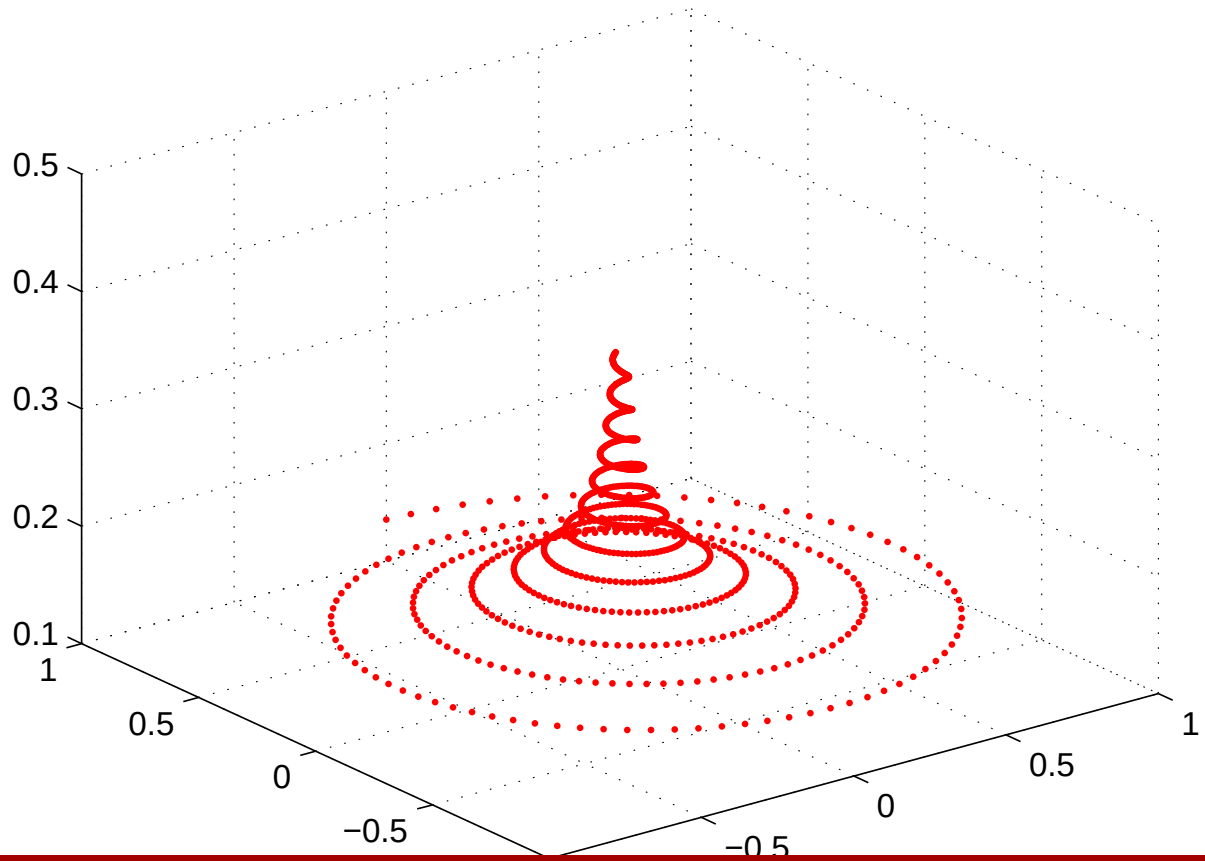
Wykresy

Przydatne polecenia

grid on/off	dodanie siatki na wykresie
title('Funkcja')	dodanie tytułu wykresu
xlabel('Kierunek poziomy')	dodanie opisu osi poziomej
ylabel('Kierunek pionowy')	dodanie opisu osi pionowej
hold	zatrzymanie aktualnego wykresu
clf	czyszczenie okna z wykresem
legend('sin(x)')	ustawienie legendy
get(gcf)	pobranie właściwości rysunku
set(gcf,par,val)	ustawienie właściwości rysunku
subplot	zagnieżdżanie rysunków

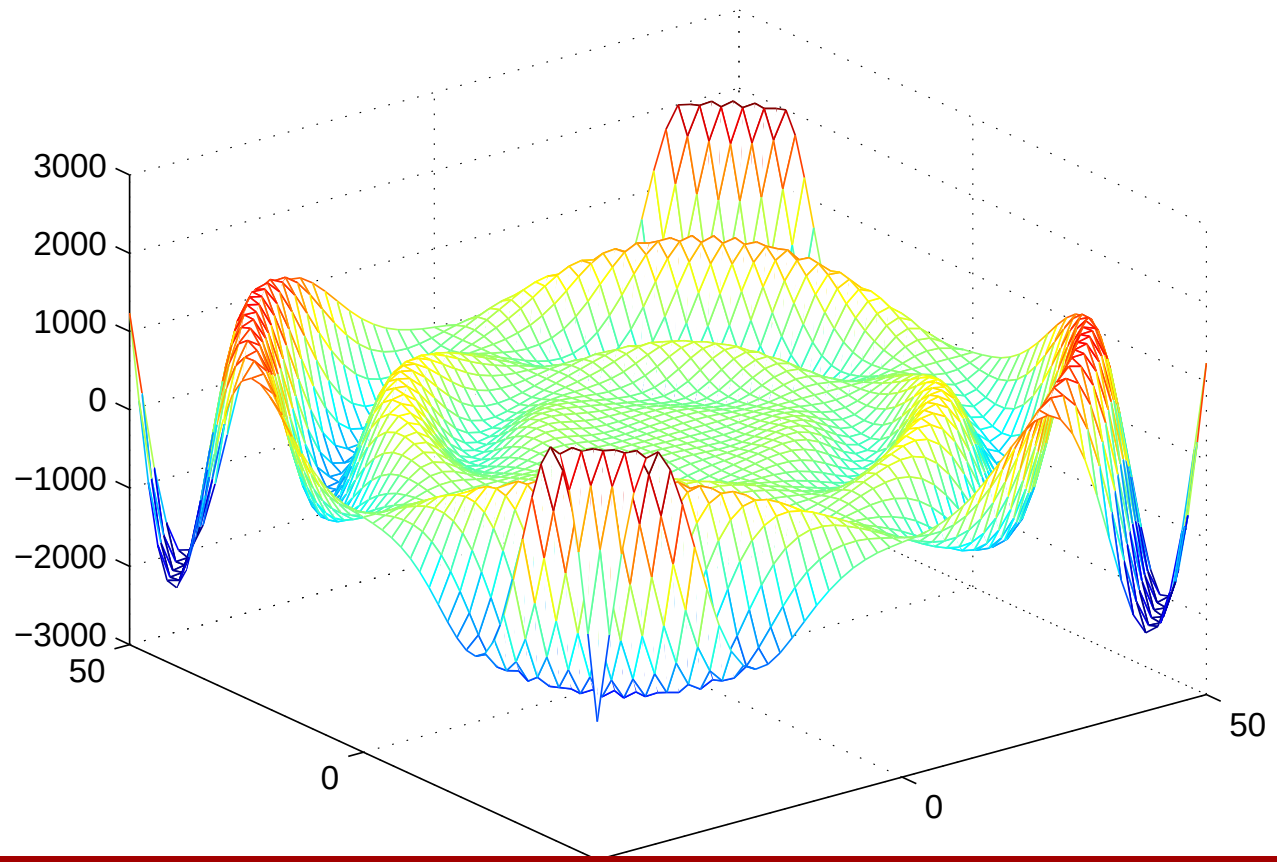
Wykres 3d – liniowy

- plot3(x,y,z)



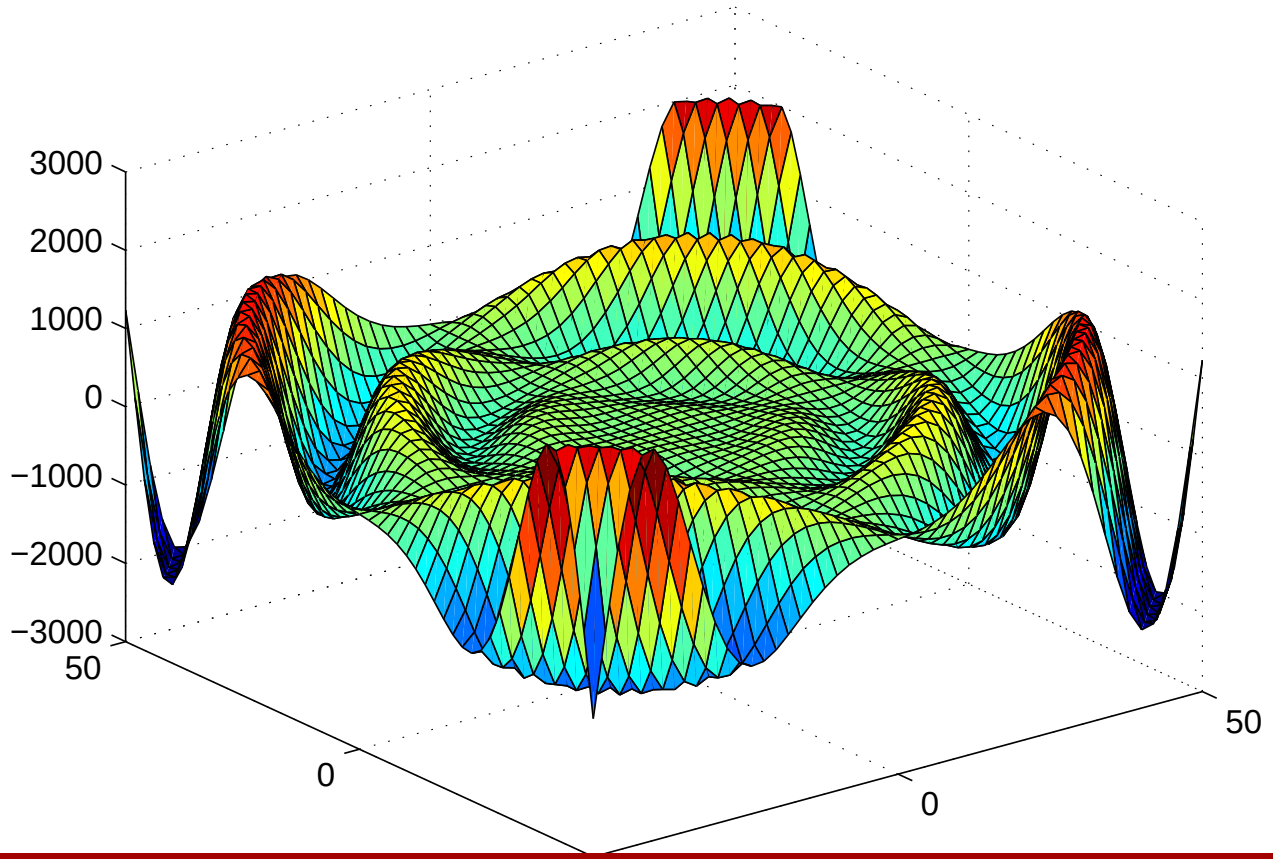
Wykres 3d – płaszczyzna przezroczysta

mesh(x,y,z) – zmienne x i y muszą tworzyć siatkę



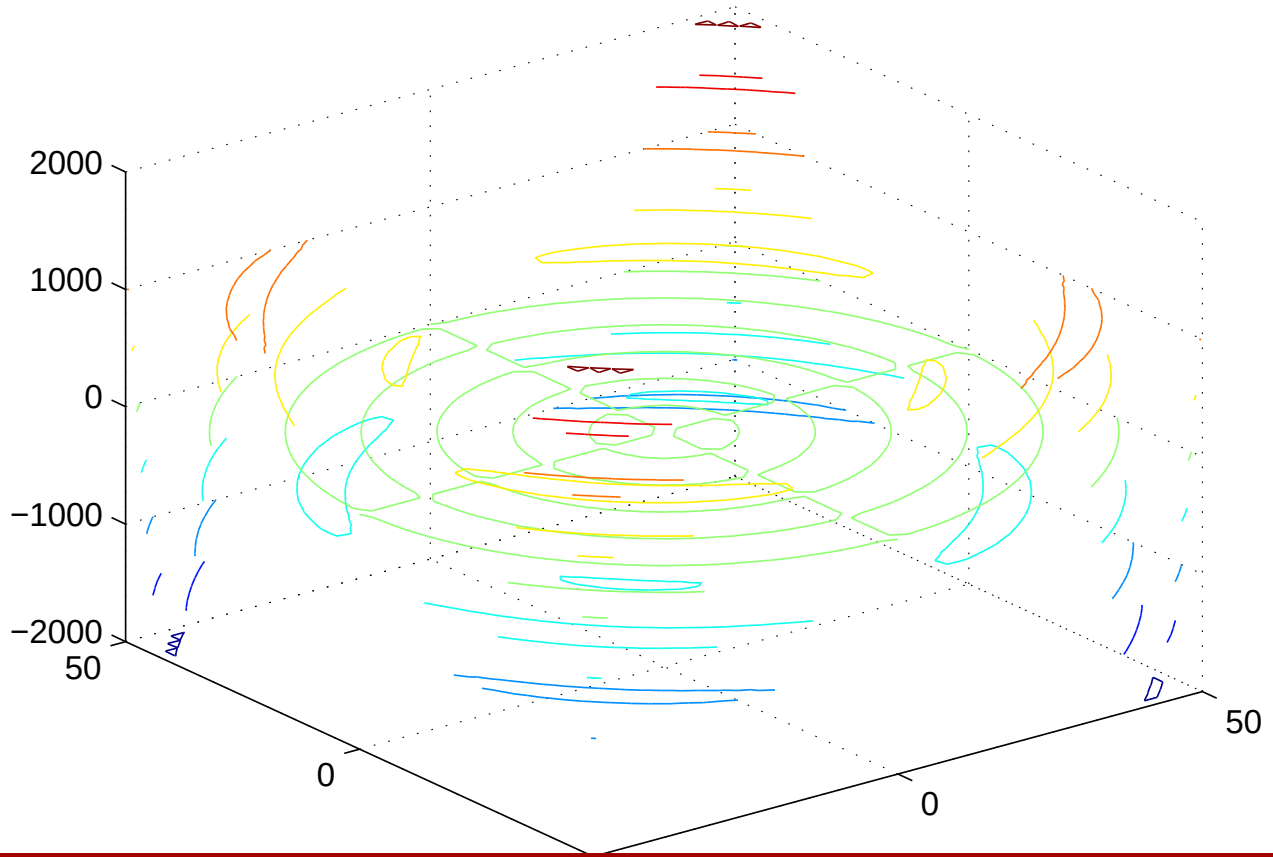
Wykres 3d – płaszczyzna

surf(x,y,z) – zmienne x i y muszą tworzyć siatkę



Wykres 3d – kontur

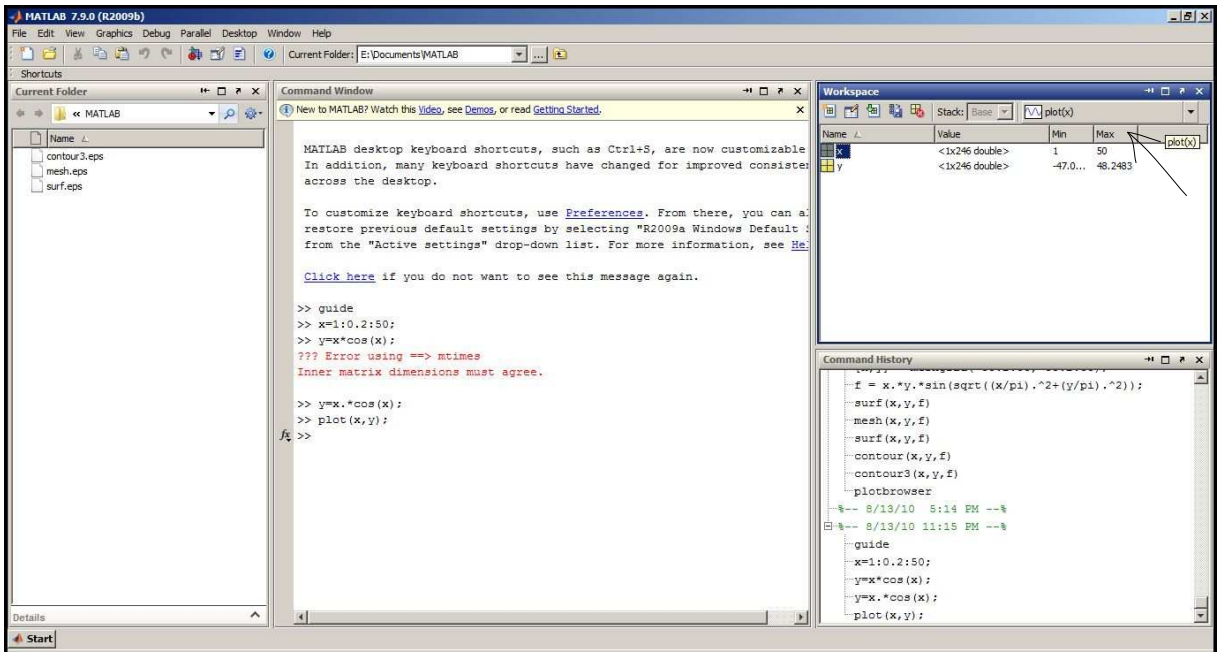
- contour3(x,y,z)



Wykresy – prosty sposób

Algorytm

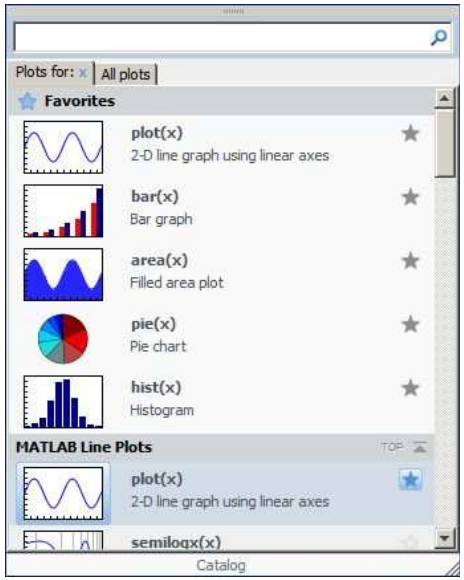
- Zaznaczyć zmienną
- Wybrać rodzaj wykresu



Wykresy – prosty sposób

Algorytm

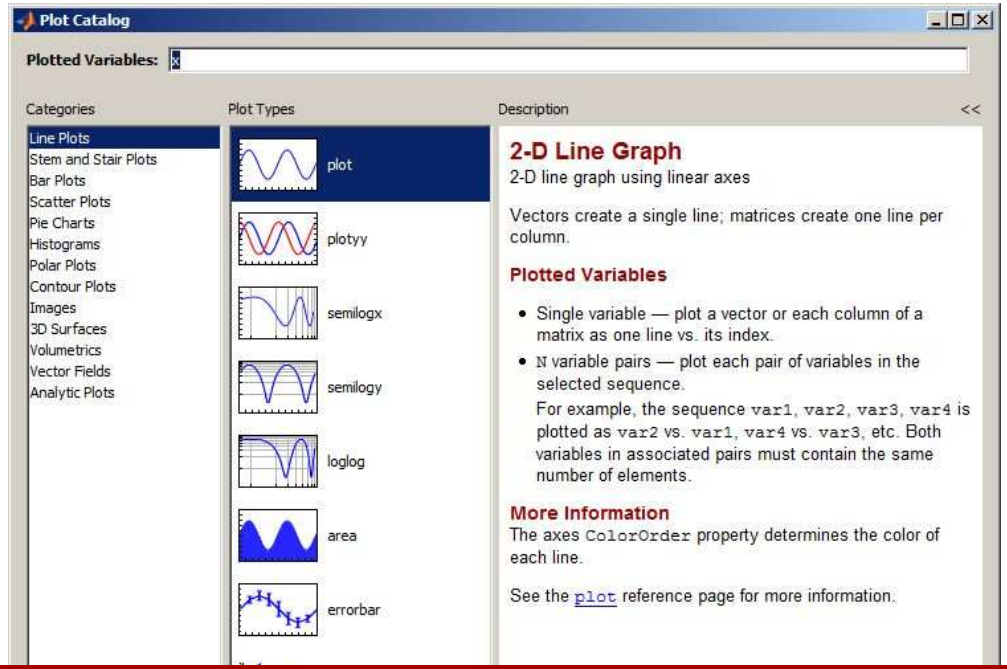
- Zaznaczyć zmienną
- Wybrać rodzaj wykresu



Wykresy – prosty sposób

Algorytm

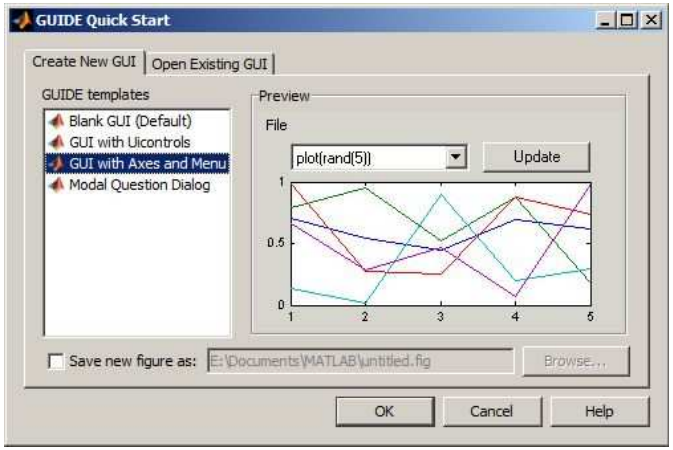
- Zaznaczyć zmienną
- Wybrać rodzaj wykresu



Tworzenie Layoutu

Wywołanie okna

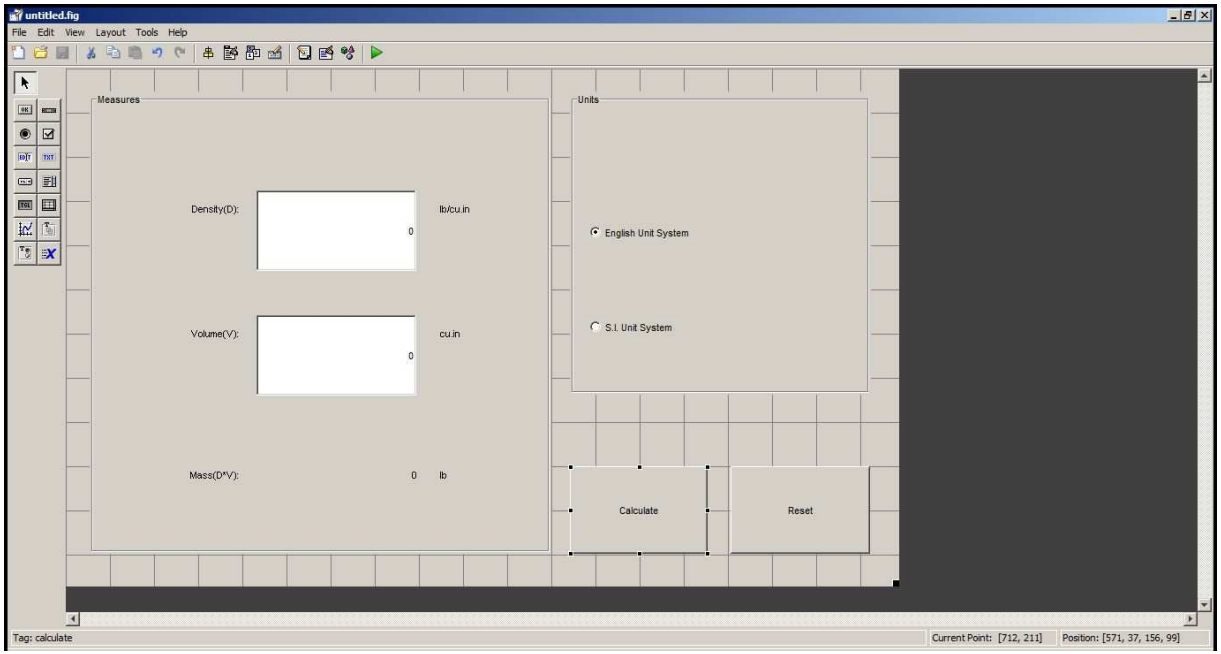
- guid



Tworzenie Layoutu

Wywołanie okna

- guid



Tworzenie funkcji obsługujących

Aktywne kontrolki – sposób działania

- przypisanie funkcji do odpowiedniej akcji, np. *ButtonDownFcn*
- pobranie właściwości kontrolki poprzez funkcję *get*, np. *get(handles.button , 'value')*
- ustawienie właściwości kontrolki poprzez funkcję *set*, np. *set(handles.button , 'value',15)*

Simulink

Do czego służy?

- edytor schematów blokowych
  - różne typy elementów
  - biblioteka elementów
  - tworzenie własnych elementów
- symulator schematów blokowych
  - pięć tryby symulacji
  - pełen dostęp do MATLABa
  - przegląd wszystkich parametrów i sygnałów
- analiza modelu

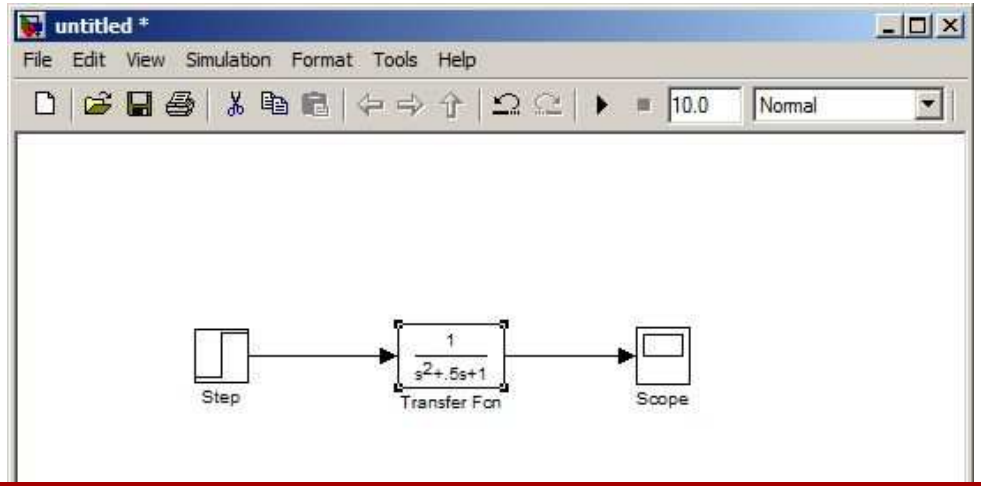
Simulink – przykład

Przykładowe zadanie

Wyznaczyć odpowiedź skokową układu o transmitancji:

$$G(s) = \frac{1}{s^2 + 2 * s + 1} \tag{1}$$

Rozwiązanie



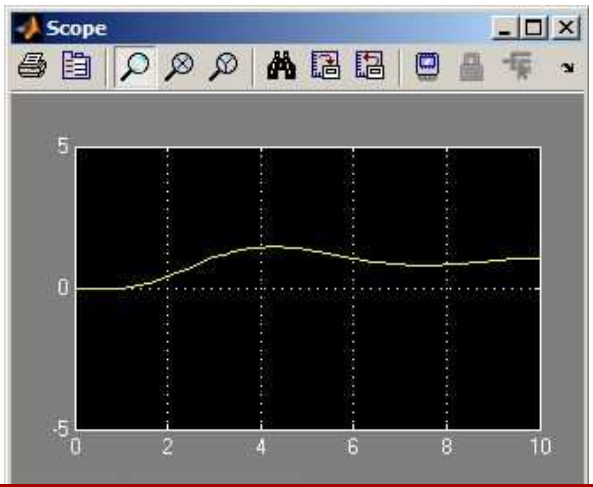
Simulink – przykład

Przykładowe zadanie

Wyznaczyć odpowiedź skokową układu o transmitancji:

$$G(s) = \frac{1}{s^2 + 2 * s + 1} \tag{1}$$

Odpowiedź

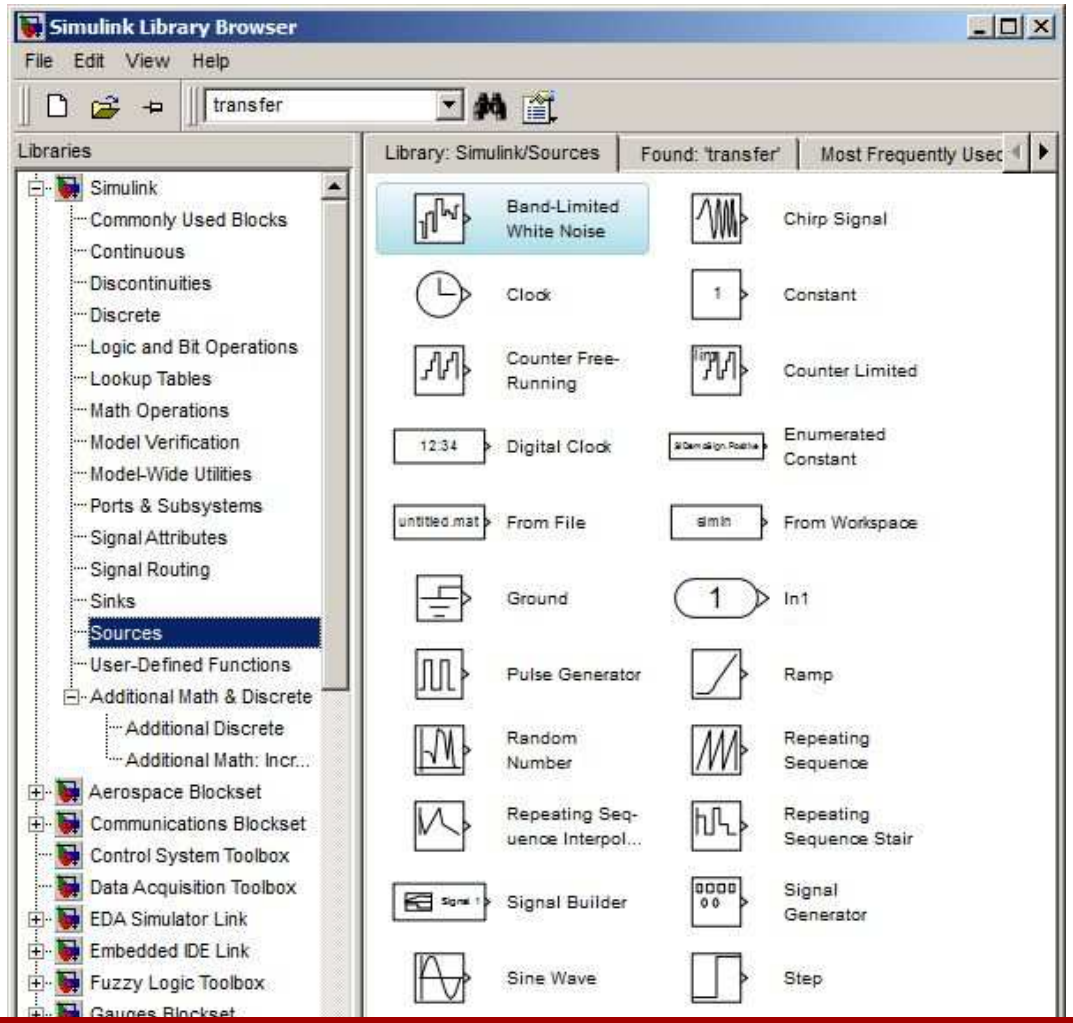


Zasada działania

Jak działa Simulink

- model graficzny jako równania stanu
- rozwiązywanie równań stanu metodami:
  - stałokrokowe Eulera
  - stałokrokowe Rungego-Kutty
  - **zmiennokrokowe RK**
  - Adamsa-Bashfortha-Moultona
  - NDF
  - inne

Biblioteka elementów



Biblioteka elementów

Źródła (**Sources**)

Band-Limited White Noise szum biały

Chirp Signal świergot – sinusoida ze zmienną f

Clock czas symulacji

Constant stała

From File pobranie z pliku

From Workspace pobranie z MATLABa

Inport port wejściowy podsystemu

Pulse Generator generator impulsów

Ramp sygnał rampy

Random Number losowa liczba

Biblioteka elementów

Wyjścia (**Sinks**)

- **Display** wyświetlacz
- **Outport** port wyjściowy podsystemu
- **Scope** wyświetlanie sygnałów podczas symulacji
- **Stop Simulation** zatrzymanie
- **To File** do pliku
- **To Workspace** do MATLABa
- **XY Graph** plot

Biblioteka elementów

Elementy czasu ciągłego (**Continuous**)

Derivative	człon całkujący
Integrator	człon różniczkujący
PID	kontroler PID
PID (2DOF)	PID dla dwóch stopni swobody
State Space	model przestrzeni stanów
Transfer Fcn	transmitancja
Transport Delay	opóźnienie przesyłowe
Zero-Pole	transmitancja w postaci zer i biegunów

Biblioteka elementów

Elementy czasu dyskretnego (**Discrete**)

Difference różnica

Discrete Derivative dyskretny człon całkujący

Discrete Integrator dyskretny człon różniczkujący

Zero-Order Hold ekstrapolator pierwszego rzedu

First-Order Hold ekstrapolator pierwszego rzedu

Integer Delay opóźnienie

PID kontroler PID

PID (2DOF) PID dla dwóch stopni swobody

Discrete State Space dyskretny model przestrzeni stanów

Transfer Fcn First Order transmitancja pierwszego rzędu

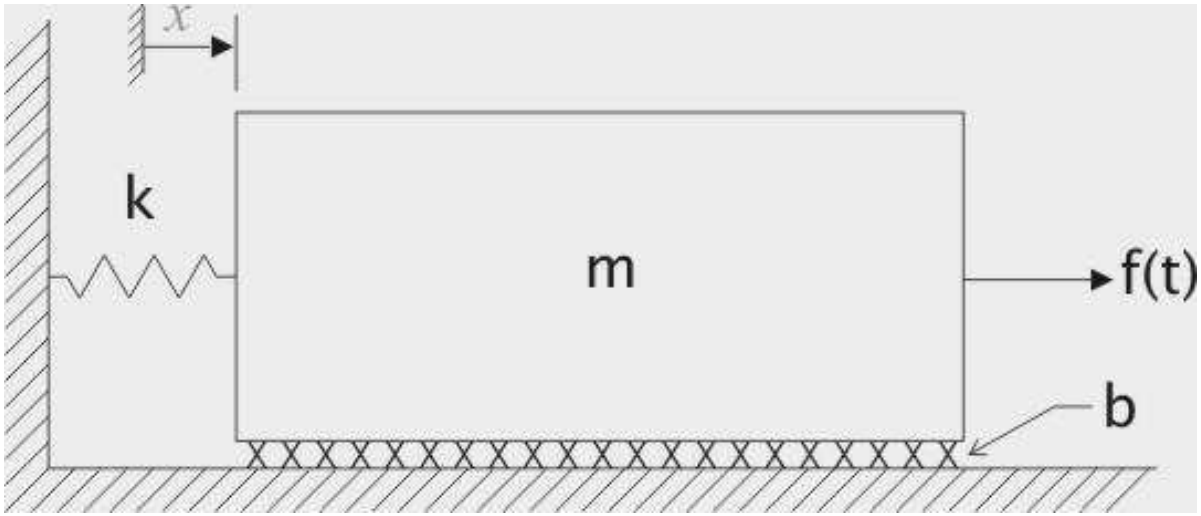
Simulink s-funkcje

Funkcje użytkownika (**User–defined Functions**))

- możliwość wykonania dowolnej funkcji
- funkcje pisane w różnych językach (C, FORTRAN, MATLAB)

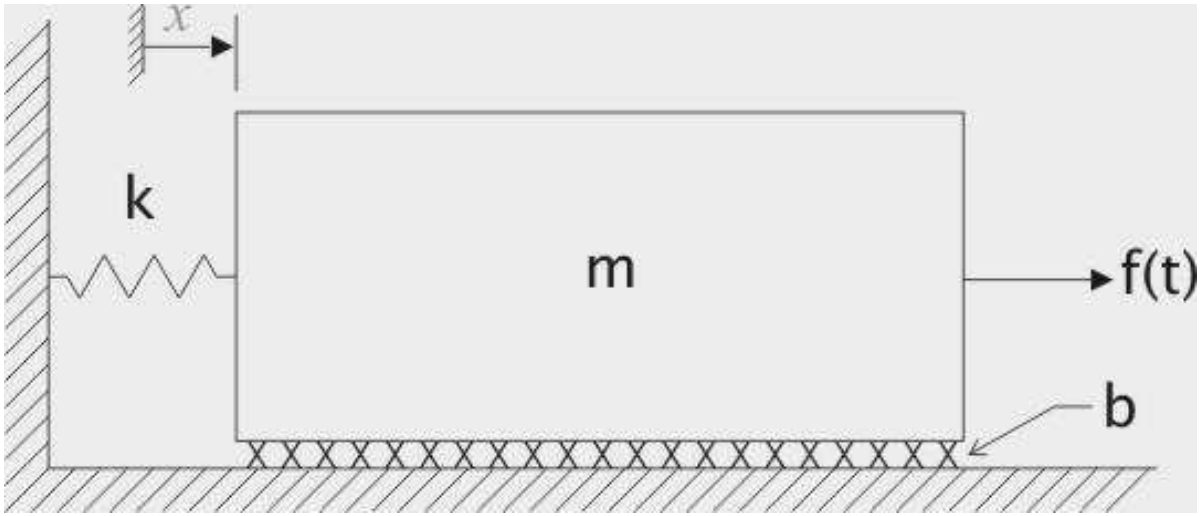
Przykład 1

Drgania harmoniczne tłumione



Przykład 1

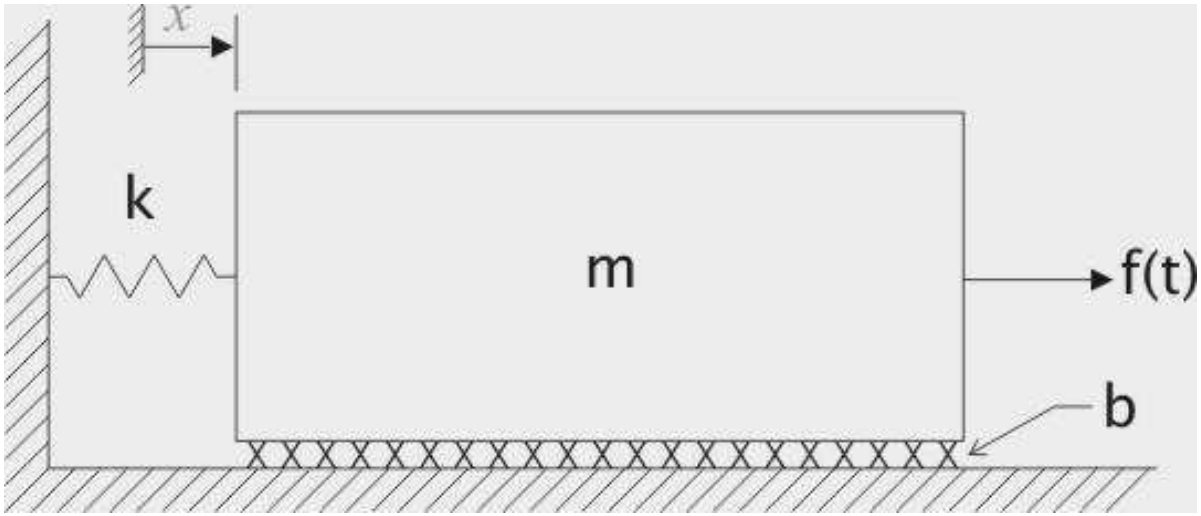
Drgania harmoniczne tłumione



$k \times x$

Przykład 1

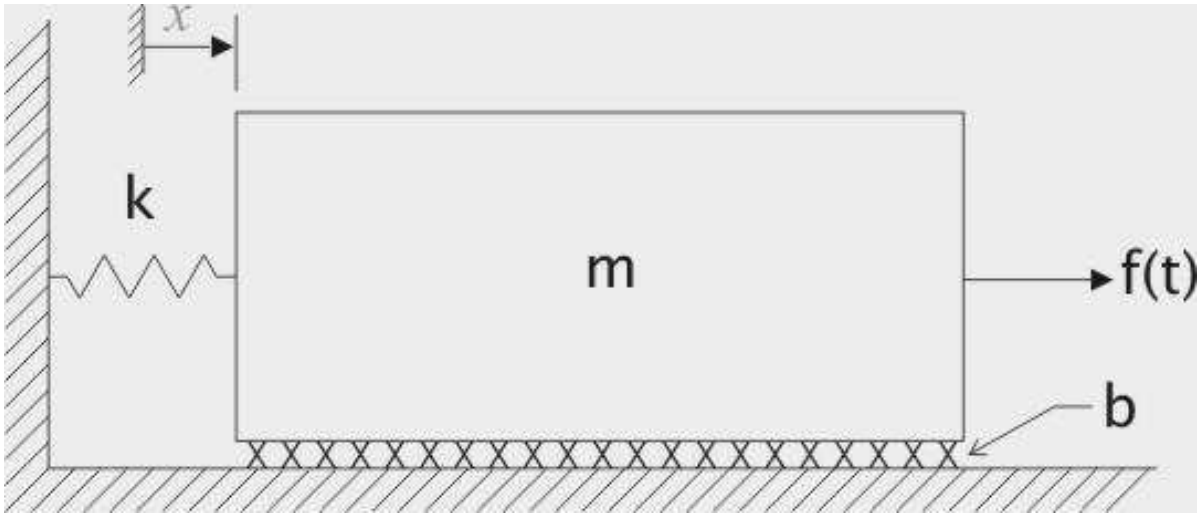
Drgania harmoniczne tłumione



$$k \times x + b \times \dot{x}$$

Przykład 1

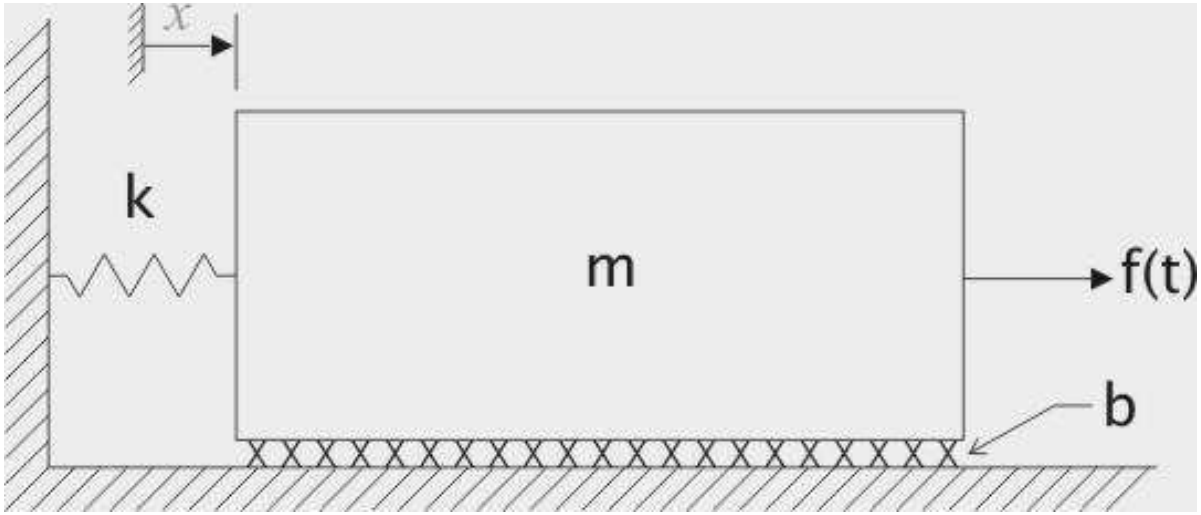
Drgania harmoniczne tłumione



$$k \times x + b \times \dot{x} + m \times \ddot{x}$$

Przykład 1

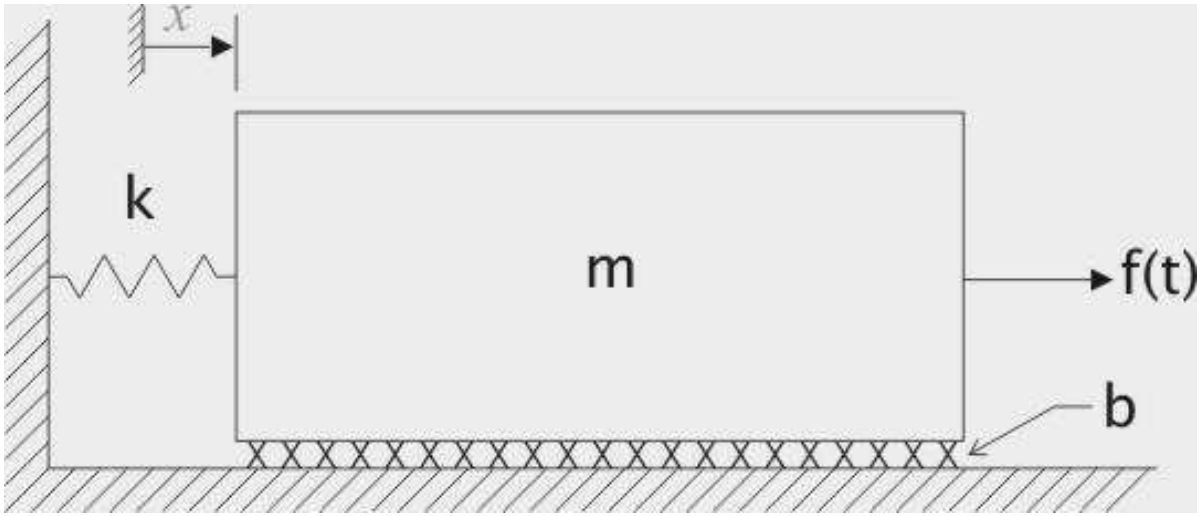
Drgania harmoniczne tłumione



$$k \times x + b \times \dot{x} + m \times \ddot{x} = f(t)$$

Przykład 1

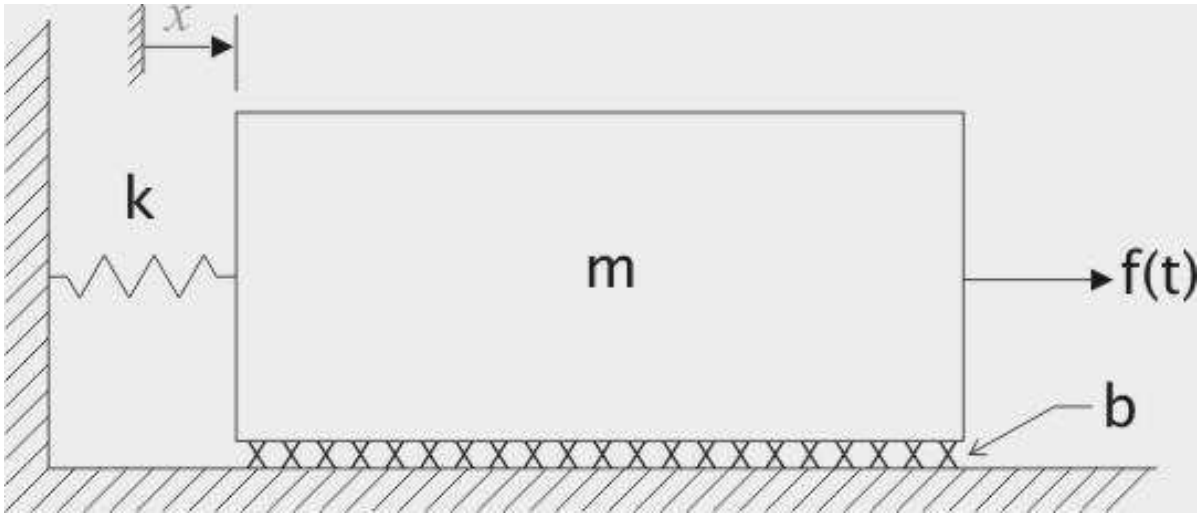
Drgania harmoniczne tłumione



$$f(t) - m \times \ddot{x} - b \times \dot{x} - k \times x = 0$$

Przykład 1

Drgania harmoniczne tłumione

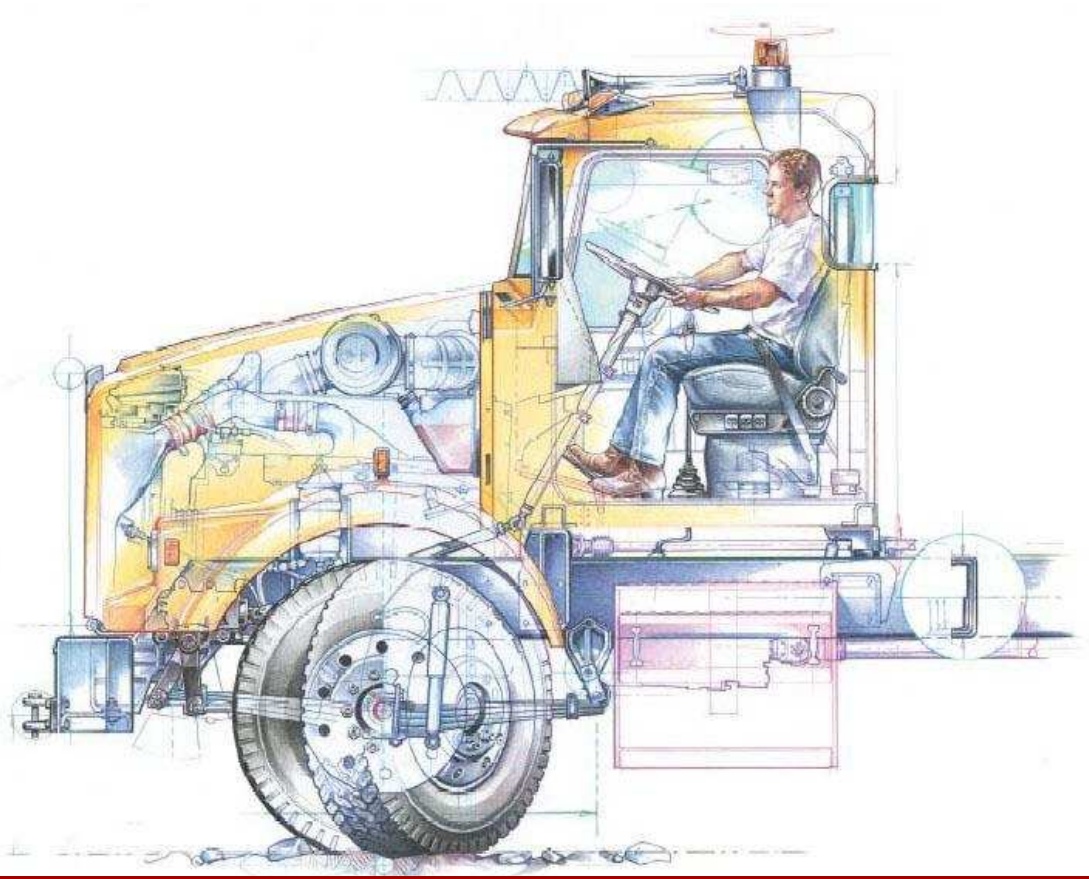


$$\ddot{x} = 1m\left(f\left(t\right) - b \times \dot{x} - k \times x\right)$$

$$\mathrm{m} = 2;\ \mathrm{b} = 5;\ \mathrm{k} = 3;$$

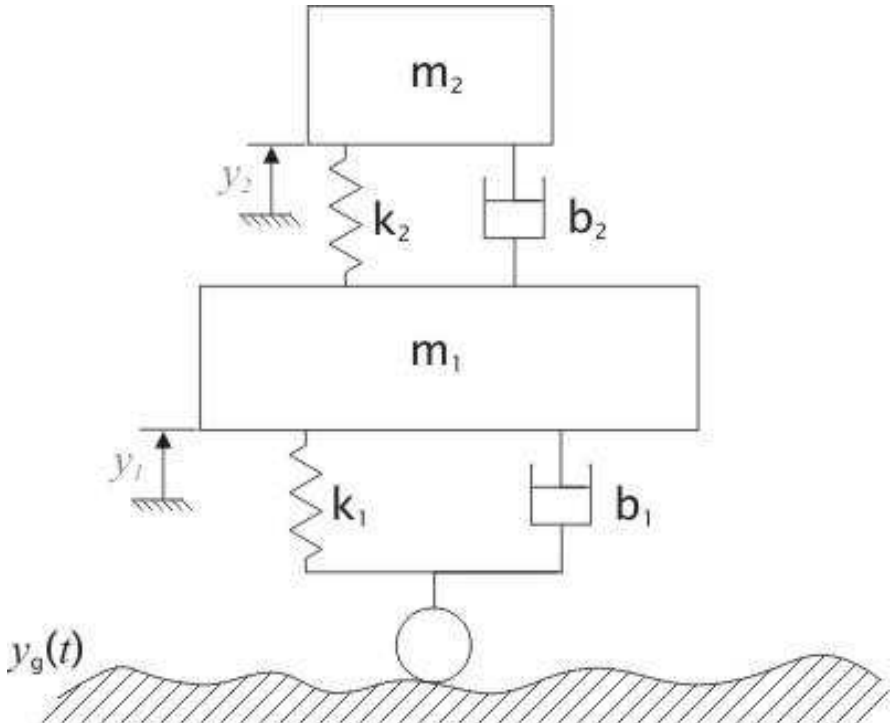
Przykład 2

Drgania – model wysokiego rzędu



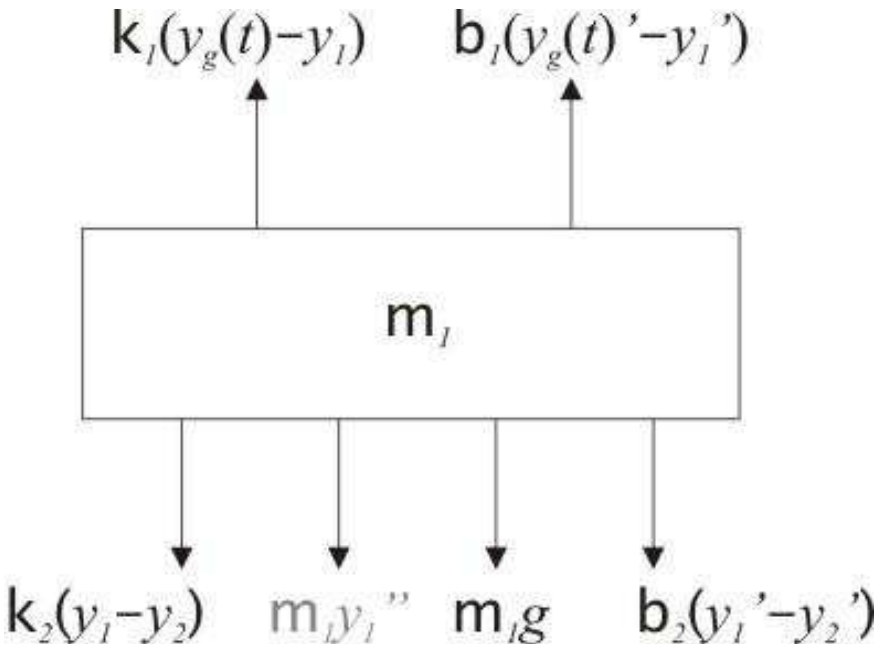
Przykład 2

Drgania – model wysokiego rzędu



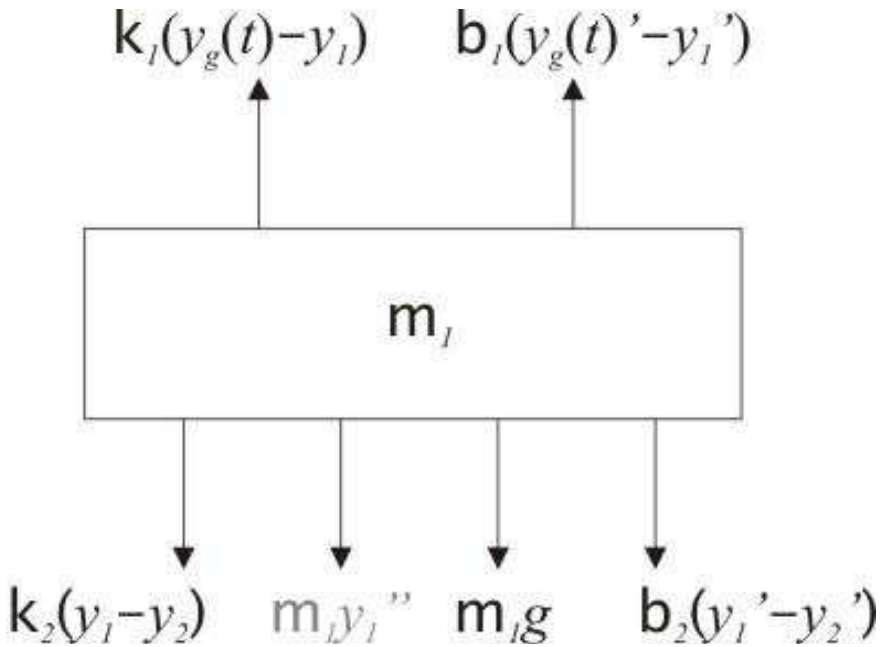
Przykład 2

Drgania – model wysokiego rzędu



Przykład 2

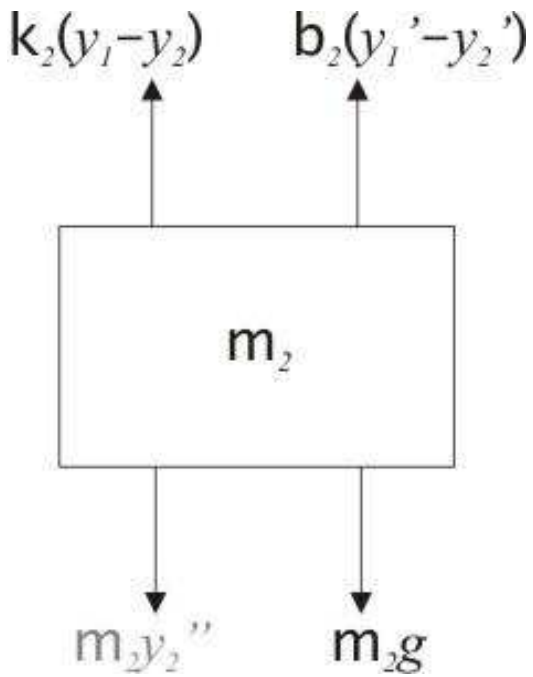
Drgania – model wysokiego rzędu



$$0 = -m_1\ddot{y}_1 + b_1\left(\dot{yg}(t) - \dot{y}_1\right) + k_1\left(yg(t) - y_1\right) - b_2\left(\dot{y}_1 - \dot{y}_2\right) - k_2\left(y_1 - y_2\right) - m_1 g$$

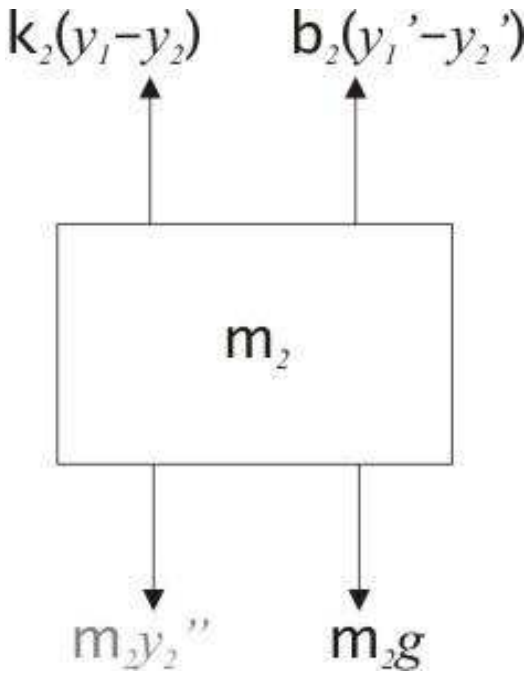
Przykład 2

Drgania – model wysokiego rzędu



Przykład 2

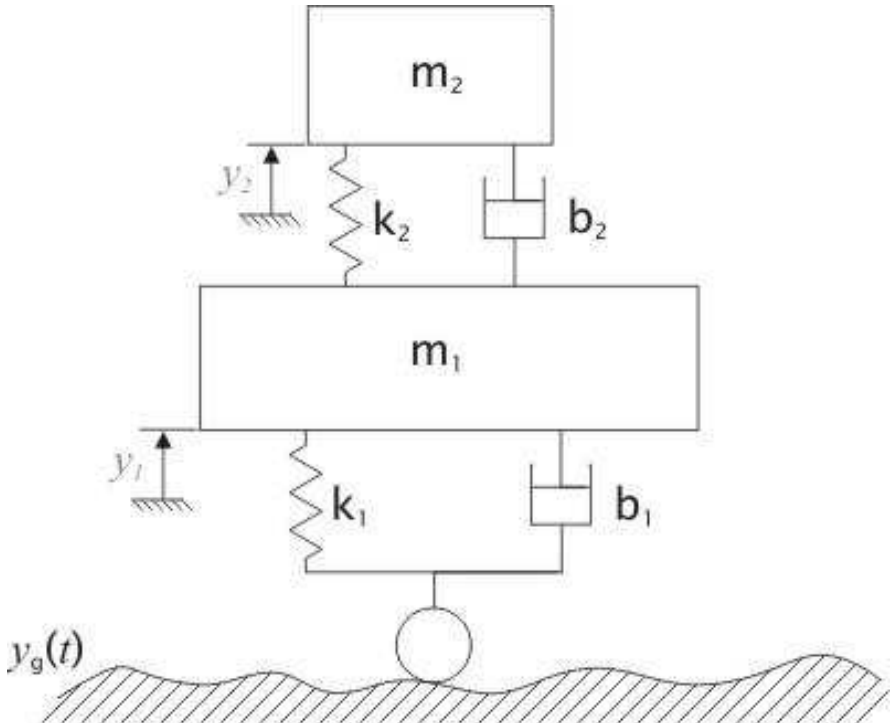
Drgania – model wysokiego rzędu



$$0 = -m_2\ddot{y}_2 + b_2\left(\dot{y}_1 - \dot{y}_2\right) + k_2\left(y_1 - y_2\right) - m_2 g$$

Przykład 2

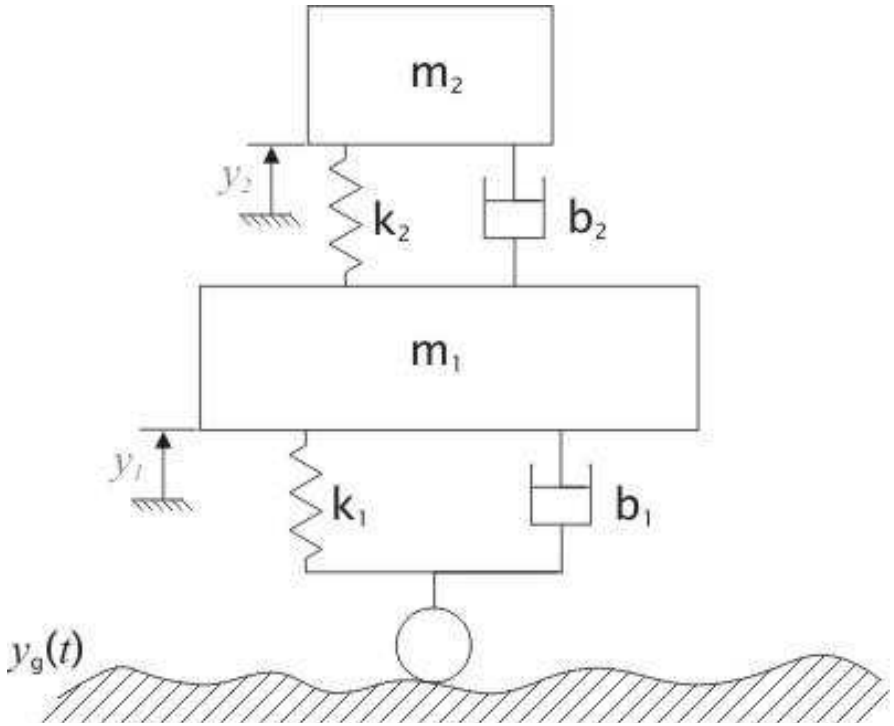
Drgania – model wysokiego rzędu



$$0 = -m_1\ddot{z}_1 + b_1\left(\dot{yg}(t) - \dot{z}_1\right) + k_1\left(yg(t) - z_1\right) - b_2\left(\dot{z}_1 - \dot{z}_2\right) -$$
$$k_2\left(z_1 - z_2\right)$$

Przykład 2

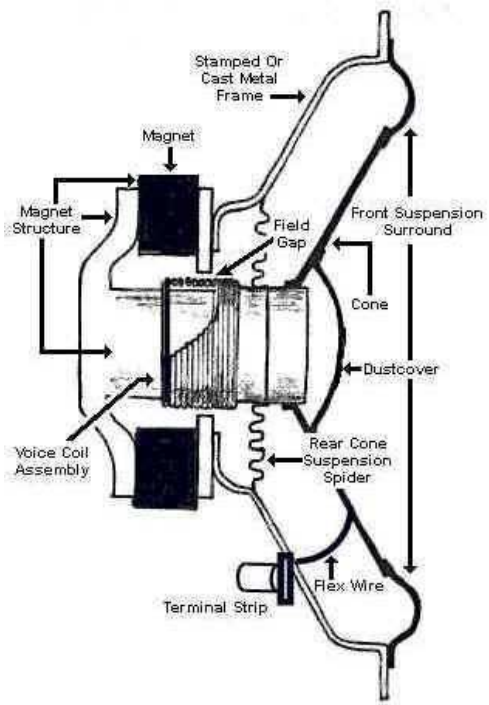
Drgania – model wysokiego rzędu



m1 = 10000; m2 = 150; b1 = 300000; b2 =1200; k1 = 100000; k2 = 11000;

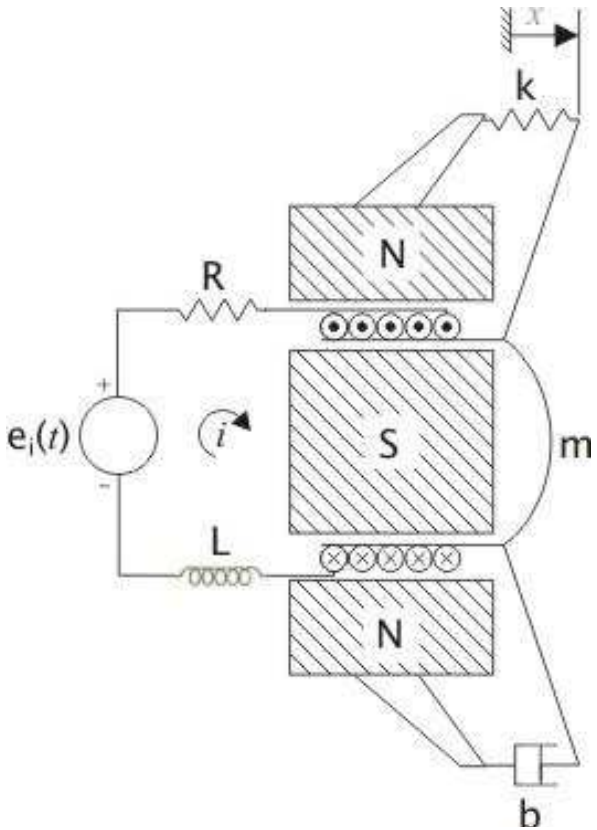
Przykład 3

Model głośnika



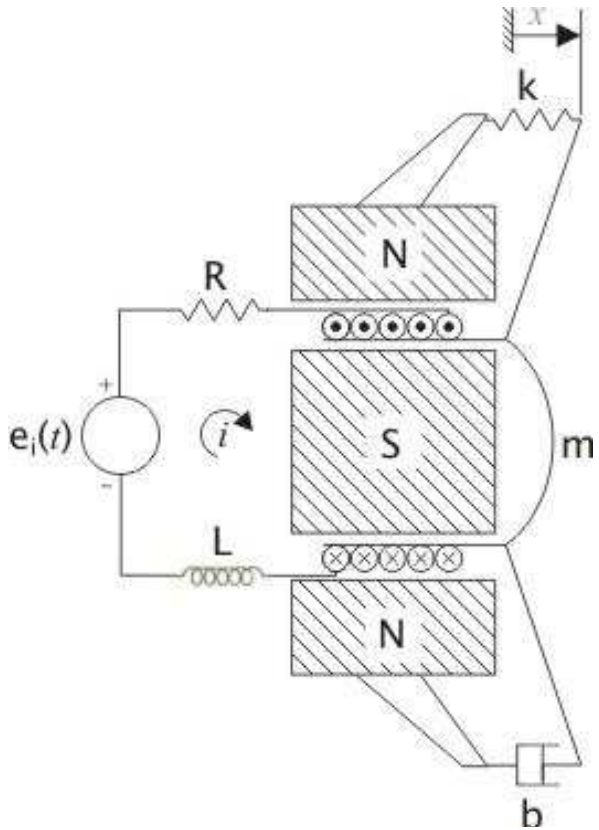
Przykład 3

Model głośnika



Przykład 3

Model głośnika



Co to jest Python?

Krótka historia

- następca języka ABC (1989)
- nazwa pochodzi od latającego cyrku
- do wersji 1.2 wydawany przez CWI (Centrum Matematyki i Informatyki w Amsterdamie)
- na licencji GPL 1.6.1 i od 2.0
- Python Software Foundation

Pythona aktualnie

Wersje:

- 2.7.3 - wsparcie wersji 2.x
- 3.2.3 - aktualna rozwojowa
- IronPython (.Net)
- Jython (Java)
- PyPy (JIT)
- Stackless Python (c)

Jak instalować Pythona?

Systemy

- Windows - wininstaller
- Linux - repozytoria
- MacOS - installer

Pythona IDE

- Anjuta (unix)
- eric python IDE (niezależny)
- geany (niezależny)
- Boa Constructor
- Komodo IDE (cross)
- MonoDevelop (cross)
- PIDA (cross)
- Spyder (Win/Linux/OS X)
- PyDev (eclipse plugin)
- PsVisual (win)
- SPE (niezależny)
- Wing (Win/Linux/OS X)

Co to jest Python?

Filozofia

- wielo-paradygmatowy
- obiektowy
- funkcyjny
- strukturalny
- typy dynamiczne
- garbage collector
- przenośność - interpreter
- brak enkapsulacji
- **prawie** wszystko jest obiektem

Ogólnie o programowaniu...

Paradygmat strukturalny

- hierarchiczne dzielenie na bloki
- omijanie skoków (*goto*, *break*, *continue*, *switch*)
- większość języków
- język stricte strukturalny - Pascal

Ogólnie o programowaniu...

Paradygmat funkcyjny

- funkcje to wartości podstawowe
- nacisk na wartościowanie
- dla danych wejść funkcja zwraca to samo wyjście
- oparte o rachunek lambda
- np. List, Scheme, F#

Ogólnie o programowaniu...

Paradygmat obiektowy

- programy to obiekty
- abstrakcyjność
- hermetyzacja
- polimorfizm
- dziedziczenie

Przykładowy prosty kod

```
def nwd(a, b):
    while b:
        a, b = b, a%b
    return a
```

Typy logiczne

- bool : True lub False
- null : None
- wszystko jest prawdą oprócz:
  - 0, 0.0, 0j;
  - " ", (), [], { };

Typy liczbowe

- całkowite: int (long w Python 2.x) *4591*
- zmiennoprzecinkowe: float *4.591*
- zespolone: complex *45.91+4.89j*

Uwaga

- typy są automatycznie przedłużane - nie występuje przepełnienie typu;
- complex zawsze w formacie *a+bj* np. *2+1j*, a nie *2+j*;

Operacje na typach liczbowych

- zapytanie typu: `type()`
- arytmetyczne proste: `+; -; *; /`
- arytmetyczne złożone: `%; //; **`
- logiczne : `==; !=; <; >; <=; >=`
- konwersje : `int(); complex(); float()`

Napisy - łańcuchy znaków

- Unicode: `str` *'Napis'* (Py 3.x) `unicode` *u'Napis'* (Py 2.x)
- ASCII: `bytes` *b'Napis'* (Py 3.x) `str` *'Napis'* (Py 2.x)

Napisy - łańcuchy znaków

- Unicode:
  `str` (Py 3.x) `'Napis'` `"Napis"`
  `unicode` (Py 2.x) `u'Napis'` `u"Napis"`
- ASCII:
  `bytes` (Py 3.x) `b'Napis'` `b"Napis"`
  `str` (Py 2.x) `'Napis'` `"Napis"`

Uwaga

- wielowierszowe w potrójnych ' lub "
- znaku specjalne Unicode działają `\n` do momentu `r'napis'`
- wyświetlanie liczb: `str(7)` `repr(9.4)`

Kolekcje

- lista (zmienna): `[4.0, 'string', True]`
- krotka (niezmienna): `(4.0, 'string', True)`
- zbiór (zmienny): `{4.0, 'string', True}`
- słownik (zmienny): `{'key1': 1.0, 3: False}`

If

```
if warunek:
    blok instrukcji
elif warunek:
    blok instrukcji
else:
    blok instrukcji
```

While

```
while warunek:
    blok instrukcji (1)
    continue
else:
    blok instrukcji (2)
```

For

```
for zmienna in lista:
    blok instrukcji (1)
else:
    blok instrukcji (2)
```

Funkcje

```
def silnia(n):
    print 'n=', n
    if n > 1:
        return n * silnia(n - 1)
    else:
        print 'koniec'
        return 1
```

String - przydatne funkcje

```
napis = 'Napis ktory leci sobie dalej'
tablica = napis.split()
napis2 = 'Poczatek napisu '
     ' '.join(i)
```

String - formatowanie

Formatowanie krotki napisów

```
napis = 'Napis ktory leci '
napis2 = 'sobie dalej'
"%s=%s" % (napis, napis2)
```

Standardowe wypisanie

```
napis = 'Napis ktory leci '
napis2 = 'sobie dalej'
i = 16
    print (napis + napis2 + str(i))
    print("2+1=",2+1)
    print("ih"*5)
```

Standardowe wejście

```
print("Halt!")
user_input = input("Who Goes there? ")
print("You may pass, " + user_input)
```

Operacje na listach

```
lista = ['ble', 667, 'fuj']
lista.index('ble')
lista[2]
lista[1:2]
lista.instert(1,'abc')
lista.append('sth')
lista.extend(['cde',556])
lista.reverse()
lista.sort()
len(lista)
```

Operacje na listach cd...

```
lista = ['ble', 667, 'fuj',778,'psik']
lista[-1]
lista[0:4]
lista[0:-1]
lista[1:]
lista[:-2]
lista.append(['cde',556])
lista.extend(['cde',556])
```

Operacje na listach cd...

```
lista = ['ble', 667, 'fuj',778,'psik']
'ble' in lista
lista.remove('ble')
lista.pop()
lista2 = ['poszedl', 'sobie', 554]
lista3 = lista+lista2
lista3+= lista2
lista3 = ['que', 'qua']*2
```

Operacje na krotkach

```
krotka = ('ble', 667, 'fuj',778,'psik')
krotka.index('fuj')
'fuj' in krotka
```

Po co są krotki?

- do iteracji (szybsze niż listy)
- zabezpieczenie przed zapisem
- jako klucz w słowniku (w przeciwieństwie do list)
- do formatowania tekstu

Operacje na słownikach

```
slownik = {'ble':'fuj', 'psik':667, 'fuj':'ble'}
slownik['ble']
slownik['psik']='poszedl'
del slownik['psik']
slownik.clear()
slownik.keys()
slownik.values()
slownik.items()
```

Zmienne, trochę ciekawostek

```
(x,y,z)=(1,2,3)
range(13)
(NIE,MOZE,TAK)=range(3)
```

Wyrażenia listowe

```
lista= [1,2,5,9]
[el*2 for el in lista]
```

Filtracja list

```
list = ['a','parszywiec','dobry','zly','brzydki','dolar','wi
[el for el in list if len(el)>0]
[el for el in list if el!='brzydki']
[el for el in list if list.count(el)==0]
```

Wyjątki

```
try:
    k=7
    print(k)
    k+'cos'
except:
    print('wyjatek')
else:
finally:
```

Wyjątki - łapanie

```
except T:
except (T1,T2):
except T as v:

raise
```

Wyjątki - typy

AssertionError
EOFError
IOError
SyntaxError
ImportError
KeyError
FloatinPointError
IndexError
TypeError
ValueError
ZeroDivisionError

Wejście - wyjście

```
input()
sys.stdin
print()
sys.stdout
```

Moduły

```
from MODUL import KLASA

from sys import stdin
```

Pliki

```
open()
read()
readline()
readlines()
write()
close()
```

Przydatne polecenia

Kodowanie Pythona

```
#-*- coding: utf-8 -*-


ord('a')
chr(97)
ord(u'a')
unichr(378)
type(234)
list = []
dir(list)
getattr(list,"pop")
```

Przydatne polecenia cd...

```
enumerate(lista)
zip(list1,list2,...)
```

Definicja funkcji

```
def nazwa(args):
"Opis funkcji w pliku nazwa.__doc__"
cialo;
```

Troszkę o funkcjach

Funkcja w Pythonie

- brak typów argumentów
- niemożliwość przeciążania funkcji
- standardowe określanie wartości `nazwa=wartosc`
- * - rozszerzenie ostatniego argumentu do postaci tablicy
- **- rozszerzenie ostatniego argumentu do postaci słownika
- `return x`

Funkcje anonimowe

```
g = lambda x: x**2
g(3)
```

Moduł

```
import m
m.funkcja
from m import *;
funkcja
```

Charakterystyka

- ponad 200 modułów
- pełna lista:
  `http://docs.python.org/py3k/py-modindex.html`

Charakterystyka

Najpopularniejsze moduły:

- time
- sys
- os
- math
- random
- pickle
- urllib
- re
- cgi
- socket

Moduł 'time'

```
gmtime(0)
localtime()  # jako krotka
time() # w sekundach

clock()  # zwraca czas procesora
ctime()  # konwersja sekund na stringa
mktime() # krotka na sekundy
sleep()

strptime("30 Nov 00", "%d %b %y")
strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())
```

Moduł 'sys'

```
argv # argumenty z linii polecen
exit(result)  # konczy program
path      # lista katalogow podczepionych do Py
version
stdin
stdout
stderr

getsizeof(object) # w bajtach
platform
getwindowsversion() # tylko w Windzie
modules # zaladowane moduly
```

Moduł 'locale'

```
getlocale()
getpreffered encoding(locale)
setlocale()
resetlocale()
```

Moduł 'os' – identyfikacja

```
environ # zmienne systemowe
environ['HOME']
getgid()
getgroups()
geteuid()
getlogin()
getpid() # analogicznie set
uname()
```

Moduł 'os' – pliki

```
open(plik) # otwiera plik
close(plik) # zamyka plik -- rozne od polecen os.*
# Polecenia os.*
open(plik, flagi)  # zwraca file descriptor
close(fd)
closerange(fd_min,fd_max)
device_encoding(fd)
dup(fd)
fchmod(fd,mode)
fchown(fd,uid,gid)
fstat(fd)
fsync(fd) # pisanie po pliku
read(fd,bytes)
write(fd,str)
```

Moduł 'os' – katalogi

```
chdir(path)
fchdir(fd)
getcwd() # current working directory
chmod(paty, mode)
chown(path,uid,gid)
mkdev(major,minor)
mkdir()
remove(path)
rename(src,dst)
rmdir(path)
stat(path)
```

Moduł 'os' – procesy

```
execl(path,arg0,...)
execv(path,args) # proces w miejscu obecnego
spawnl(mode,path,arg0,...)
spawnv(mode,path,artgs) # nowy procesu
_exit(n)
fork()
kill(pid,sig)
system(cammand)
times()  # zwraca piec roznych czasow
wait()
```

Moduł 'array'

- lista zajmująca mniej pamięci
- dopuszczalne typy: 'b', 'B' (char), 'u', 'h', 'H' (short), 'i', 'I', 'l', 'L', 'f', 'd'

```
array('l',[1,2,3,4])
itemsize
append(x)   extend(ietarble)
count(x)   # zlicza ilosc wystapien
index(x) insert(i,x)
pop([i])   remove(x)
reverse()
to*(); from*()   # bytes, file, list, string, unicode
```

Moduł 'math'

```
ceil(x)
copysign(x,y) # from y to x
fabs(x)
flood(x)
fmod(x,y) # modulo z biblioteki C
factorial(x)
frexp(x) # zamiana na format wykladniczy
fsum(iterable) suma liczb
## dla rozroznienia od sum(it) co zwraca polaczona
isinfinite(x)
isnan(x)
ldexp(x,i) zwraca x*2^i
```

Moduł 'math'

```
exp(x)
expm1(x) # zwraca (e^x)-1
log(x,base)
log1p(x)  # ln(x+1)
log10(x)
pow(x,y)
sqrt(x)
```

Moduł 'math'

```
degrees(x)
radians(x)
gamma(x)
lgamma(x) # zwraca ln z gamma(x)
```

Moduł 'random'

```
seed([x])
randrange(start,stop,step)
randint(a,b)
choice(seq)
shuffle(seq)
sample(population,k)
random()
batavariate(alpha,betha)
expovariate(lambd)
gammavariate(alpha, beta)
gauss(mu, sigma) #szybsze
normalvariate(mu,sigma)
paretovariate(alpha)
```

Moduł 'cmath'

```
phase(com)
polar(com)
rect(r, phi)
exp(com)
log(com)
log10(com)
sqrt(com)
```

Moduł 'fractions'

```
Fraction(num,denum)
Fraction(float)
Fraction(dec)
Fraction(str)
limit_denominator(max)
```

Parsery plików

- xml
- html
- configparser
- argparser

Bazy danych

- dbm
- sqlite3

Formatowanie stringów

- string
- re

Moduł 'string' - constants

```
ascii_letters
ascii_lowercase
ascii_uppercase
digits
hexdigits
whitespace
```

Moduł 'string' - Formatter

```
format(string,*args,**kwargs)
replacement_field ::=  "{" [field_name] ["!" conversion]
                          [":" format_spec] "}"
field_name        ::=  arg_name ("." attribute_name |
                          "[" element_index "]")*
arg_name          ::=  [identifier | integer]
attribute_name    ::=  identifier
element_index     ::=  integer | index_string
index_string      ::=  <any source character except "]"> +
conversion        ::=  "r" | "s" | "a"
format_spec       ::=  <described in the next section>
```

Moduł 'string' - Formatter mini language

```
format_spec ::=  [[fill]align][sign][#][0][width][,]...
                                    [.precision][type]
fill        ::=  <a character other than '}'>
align       ::=  "<" | ">" | "=" | "^"
sign        ::=  "+" | "-" | " "
width       ::=  integer
precision   ::=  integer
type        ::=  "b" | "c" | "d" | "e" | "E" | "f" | "F" | ...
                 "g" | "G" | "n" | "o" | "s" | "x" | "X" | "%"
```

Moduł 'string' - Formatter przykłady

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{}, {}, {}'.format('a', 'b', 'c')  # 3.1+ only
'a, b, c'
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{2}, {1}, {0}'.format(*'abc')
'c, b, a'
>>> '{0}{1}{0}'.format('abra', 'cad')
'abracadabra'
>>> "repr() shows quotes: {!r}; str() doesn't: {!s}" ...
.format('test1', 'test2')
"repr() shows quotes: 'test1'; str() doesn't: test2"
```

Moduł 'string' - Formatter przykłady

```
>>> width = 5
>>> for num in range(5,12):
...     for base in 'dXob':
...         print('{0:{width}{base}}'.format(num, base=base),
...     print()
```

Moduły sieciowe

- ebbrowser
- http
- urllib
- telnetlib
- ssl
- socket
- smtpd, smtplib, poplib, mailbox, imaplib, email
- ftplib
- cgi, cgitb

Moduły narzędziowe

Pakery

- zlib, gzip
- bz2
- zipfile
- tarfile

Moduły narzędziowe

Programowanie współbieżne

- threading
- multiprocessing
- subprocess

Moduły narzędziowe

Różne

- wave
- unittest
- timeit *pomiar czasu wykonywania programów*
- curses, tkinter *gui*

Okienka

Typy

- Tkinter
- PyGTK
- PyQt
- wxPython
- Dabo

Tk

```
from Tkinter import *
class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.grid()
        self.createWidgets()
    def createWidgets(self):
        self.quitButton = Button ( self, text='Quit',
            command=self.quit )
        self.quitButton.grid()

app = Application()
app.master.title("Sample application")
app.mainloop()
```

PyQt

```
from PyQt4.QtCore import *
from PyQt4.QtGui import *

class App(QApplication):
    def __init__(self, argv):
        super(App, self).__init__(argv)
        self.msg = QLabel("Hello, World!")
        self.msg.show()

if __name__ == "__main__":
    import sys
    app = App(sys.argv)
    sys.exit(app.exec_)
```

wxPython

```
import wx
class test(wx.App):
    def __init__(self):
        wx.App.__init__(self, redirect=False)
    def OnInit(self):
        frame = wx.Frame(None, -1,
                         "Test",
                         pos=(50,50), size=(100,40),
                         style=wx.DEFAULT_FRAME_STYLE)
        button = wx.Button(frame, -1, "Hello World!", (20, 20
        self.frame = frame
        self.frame.Show()
        return True
if __name__ == '__main__':
        app = test()
        app.MainLoop()
```

wxPython

```
import dabo
dabo.ui.loadUI("wx")
class TestForm(dabo.ui.dForm):
        def afterInit(self):
                self.Caption = "Test"
                self.Position = (50, 50)
                self.Size = (100, 40)
                self.btn = dabo.ui.dButton(self, Caption="He
                        OnHit=self.onButtonClick)
                self.Sizer.append(self.btn, halign="center",
        def onButtonClick(self, evt):
                dabo.ui.info("Hello World!")
if __name__ == '__main__':
        app = dabo.ui.dApp()
        app.MainFormClass = TestForm
        app.start()
```

NymPy

Co to jest?

- n-wymiarowe macierze
- zaawansowane metody macierzowe
- metody kształtowania macierzy
- podstawy algebry liniowej
- tranformaty Fouriera
- wyrafinowany random

SciPy

Co to jest?

- statystyka
- optimizacja
- numeryczne całkowanie
- algebra liniowa
- transformata Fouriera
- przetwarzanie sygnałw
- przetwarzanie obrazów
- solvery równań różnicowych

matplotlib

Co to jest?

- wykresy 2d
- wykresy 3d

```
sudo apt-get install python3-dev libpng12-dev libpng3
git clone https://github.com/matplotlib/matplotlib
cd matplotlib
python3 setup.py build
sudo python3 setup.py install
```

numpy - arra

```
>>> from numpy  import *
>>> a = arange(10).reshape(2,5)
>>> a
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
>>> a.shape
(2, 5)
>>> a.ndim
2
>>> a.dtype.name
'int32'
>>> a.itemsize
4
>>> a.size
10
>>> type(a)
numpy.ndarray
```

numpy

```
>>> b = array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
numpy.ndarray
>>> c = array( [ [1,2], [3,4] ], dtype=complex )
>>> c
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
```

numpy

```
>>> zeros( (3,4) )
array([[0.,  0.,  0.,  0.],
       [0.,  0.,  0.,  0.],
       [0.,  0.,  0.,  0.]])
>>> ones( (2,3,4), dtype=int16 )                # dtype can
array([[[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]],
       [[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]]], dtype=int16)
>>> empty( (2,3) )
array([[  3.73603959e-262,   6.02658058e-154,   6.55490914e-
        [  5.30498948e-313,   3.14673309e-307,   1.00000000e+
```

numpy

```
>>> arange( 10, 30, 5 )
array([10, 15, 20, 25])
>>> arange( 0, 2, 0.3 )                 # it accepts float ar
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])
>>> linspace( 0, 2, 9 )                 # 9 numbers from 0 to
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ,  1.25,  1.5 ,
1.75,  2.  ])
>>> x = linspace( 0, 2*pi, 100 )        # useful to evaluate
>>> f = sin(x)
```

numpy

```
>>> a = arange(6)                          # 1d array
>>> print a
[0 1 2 3 4 5]
>>> b = arange(12).reshape(4,3)            # 2d array
>>> print b
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
>>> c = arange(24).reshape(2,3,4)          # 3d array
>>> print c
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]
 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

numpy

```
>>> a = array( [20,30,40,50] )
>>> b = arange( 4 )
>>> b
array([0, 1, 2, 3])
>>> c = a-b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10*sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a<35
array([True, True, False, False], dtype=bool)
```

numpy

```
>>> A = array( [[1,1],
...             [0,1]] )
>>> B = array( [[2,0],
...             [3,4]] )
>>> A*B                         # elementwise product
array([[2, 0],
       [0, 4]])
>>> dot(A,B)                    # matrix product
array([[5, 4],
       [3, 4]])
>>> a = random.random((2,3))
>>> a
array([[ 0.6903007 ,  0.39168346,  0.16524769],
       [ 0.48819875,  0.77188505,  0.94792155]])
>>> a.sum()
3.455237210052148 5
>>> a.min()
```

numpy

```
>>> a = arange(10)**3
>>> a
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729])
>>> a[2]
8
>>> a[2:5]
array([ 8, 27, 64])
>>> a[:6:2] = -1000    # equivalent to a[0:6:2] = -1000;
>>> a
array([-1000,     1, -1000,    27, -1000,   125,   216,
343,   512,   729])
>>> a[ : :-1]                                    # reversed a
array([  729,   512,   343,   216,   125, -1000,    27, -100
1, -1000])
>>> for i in a:
...         print i**(1/3.),
...
```

numpy

```
>>> a.ravel() # flatten the array
array([ 7.,  5.,  9.,  3.,  7.,  2.,  7.,  8.,  6.,  8.,
3.,  2.])
>>> a.shape = (6, 2)
>>> a.transpose()
array([[ 7.,  9.,  7.,  7.,  6.,  3.],
       [ 5.,  3.,  2.,  8.,  8.,  2.]])
```

numpy

```
>>> a = floor(10*random.random((2,2)))
>>> a
array([[ 1.,  1.],
       [ 5.,  8.]])
>>> b = floor(10*random.random((2,2)))
>>> b
array([[ 3.,  3.],
       [ 6.,  0.]])
>>> vstack((a,b))
array([[ 1.,  1.],
       [ 5.,  8.],
       [ 3.,  3.],
       [ 6.,  0.]])
>>> hstack((a,b))
array([[ 1.,  1.,  3.,  3.],
       [ 5.,  8.,  6.,  0.]])
```

numpy

```
>>> a = floor(10*random.random((2,12)))
>>> a
array([[ 8.,  8.,  3.,  9.,  0.,  4.,  3.,  0.,  0.,  6.,
4.,  4.],
       [ 0.,  3.,  2.,  9.,  6.,  0.,  4.,  5.,  7.,  5.,
1.,  4.]])
>>> hsplit(a,3)   # Split a into 3
[array([[ 8.,  8.,  3.,  9.],
       [ 0.,  3.,  2.,  9.]]), array([[ 0.,  4.,  3.,  0.],
       [ 6.,  0.,  4.,  5.]]), array([[ 0.,  6.,  4.,  4.],
       [ 7.,  5.,  1.,  4.]])]
>>> hsplit(a,(3,4))   # Split a after the third and the fourt
[array([[ 8.,  8.,  3.],
       [ 0.,  3.,  2.]]), array([[ 9.],
       [ 9.]]), array([[ 0.,  4.,  3.,  0.,  0.,  6.,  4.,
4.],
       [ 6.,  0.,  4.,  5.,  7.,  5.,  1.,  4.]])]
```

numpy

```
>>> from numpy import *
>>> from numpy.linalg import *
>>> a = array([[1.0, 2.0], [3.0, 4.0]])
>>> print a
[[ 1.  2.]
 [ 3.  4.]]
>>> a.transpose()
array([[ 1.,  3.],
       [ 2.,  4.]])
>>> inv(a)
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
>>> u = eye(2) # unit 2x2 matrix; "eye" represents "I"
>>> u
array([[ 1.,  0.],
       [ 0.,  1.]])
>>> j = array([[0.0, -1.0], [1.0, 0.0]])
```

numpy

```
>>> dot (j, j) # matrix product
array([[-1.,  0.],
       [ 0., -1.]])

>>> trace(u)  # trace
2.0

>>> y = array([[5.], [7.]])
>>> solve(a, y)
array([[-3.],
       [ 4.]])

>>> eig(j)
(array([ 0.+1.j,  0.-1.j]),
array([[ 0.70710678+0.j,  0.70710678+0.j],
       [ 0.00000000-0.70710678j,  0.00000000+0.70710678j]]))
```

numpy - Matrix

```
>>> A = matrix('1.0 2.0; 3.0 4.0')
>>> A
[[ 1.  2.]
 [ 3.  4.]]
>>> type(A)  # file where class is defined
<class 'numpy.matrixlib.defmatrix.matrix'>
>>> A.T  # transpose
[[ 1.  3.]
 [ 2.  4.]]
>>> X = matrix('5.0 7.0')
>>> Y = X.T
>>> Y
[[5.]
 [7.]]
>>> print A*Y  # matrix multiplication
[[19.]
 [43.]]
```

numpy - Matrix

```
>>> print A.I  # inverse
[[-2.   1. ]
 [ 1.5 -0.5]]
>>> solve(A, Y)  # solving linear equation
matrix([[-3.],
        [ 4.]])
```

scipy - matrix

```
from numpy import matrix
from scipy.linalg import inv, det, eig

A=matrix([[1,1,1],[4,4,3],[7,8,5]]) # 3 lines 3 rows
b = matrix([1,2,1]).transpose()     # 3 lines 1 rows.

print det(A)      # We can check, whether the matrix is regula
print inv(A)*b    # Now we can print the solution of the Ax=b
print eig(A)
```

scipy - linalg

```
from numpy import allclose, arange, eye,  ones
from scipy import linalg, sparse

A = eye(1000)
Asp = sparse.lil_matrix(A)

Asp = sparse.lil_matrix((1000,1000))
Asp.setdiag(ones(1000))
b = arange(1,1001)

x = linalg.solve(A,b)

A = mat('[1 3 2; 1 4 5; 2 3 6]')
linalg.lu(A)
linalg.cholesky(A)
linalg.qr(A)
```

scipy

```
>>> from scipy.integrate import quad
>>> quad(lambda x: x**2, 0, 1)
(0.3333333333333331, 3.700743415417879e-15)

>>> from scipy.integrate import inf
>>> quad(lambda x: x**3/(exp(x)-1), 0.1, inf)
(6.493618402286668, 2.3462101939733651e-09)
```

scipy ODES

```
>>> from scipy.integrate import odeint
>>> odeint(lambda y, t: y, 1, [0, 1])
array([[ 1.        ],
       [ 2.71828193]])

>>> def deriv(Y, t):
...     y, yp = Y
...     return yp, -y
...
>>> odeint(deriv, [1,0], [0,N.pi,2*N.pi])
array([[  1.00000000e+00,   0.00000000e+00],
       [ -9.99999975e-01,  -1.26160610e-07],
       [  1.00000003e+00,   1.47819759e-07]])
```

matplotlib

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g',
 alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
```

matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
             arrowprops=dict(facecolor='black', shrink=0.05),
             )

plt.ylim(-2,2)
plt.show()
```