Technologie Informacyjne i Komunikacyjne 2023/2024

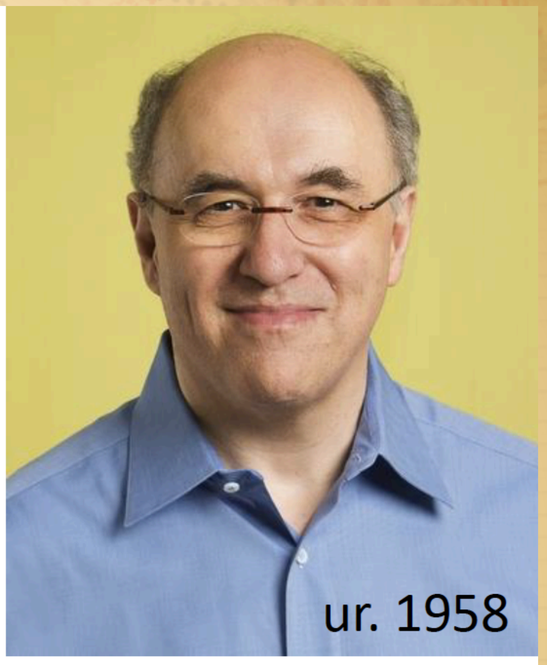
dr Magdalena Posiadała-Zezula

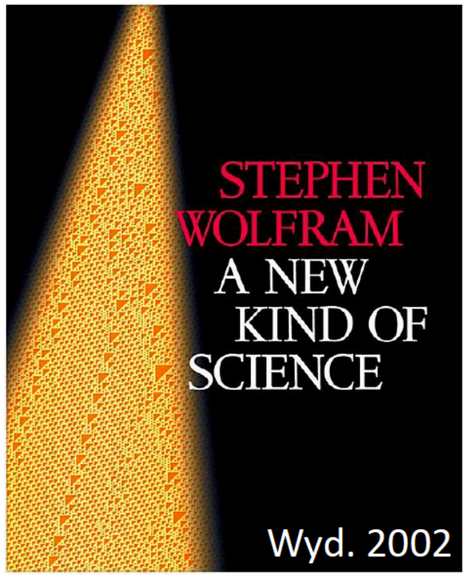
Wykład 6
Mathematica

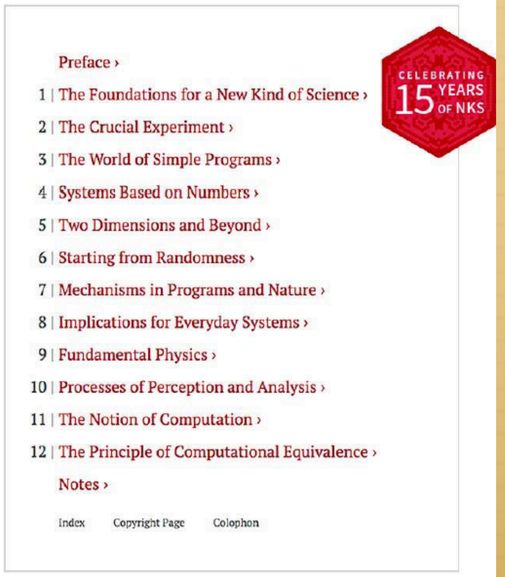
Photo Credit: Piotr Mijakowski

Twórca języka programowania Mathematica- dr Stephen Wolfram

- Pierwszy artykuł naukowy w wieku 15 lat
- Doktorat w wieku 20 lat
- Eton, Princeton, Caltech
- Firma Wolfram Research (1987)
- Mathematica (1988)
- „A New Kind of Science”
- Wolfrma Alpha (2009)







Wolfram Mathematica- materiały do samodzielnej nauki

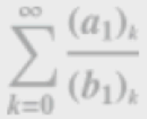
★Kurs online: FAST INTRODUCTION FOR MATH STUDENTS": https://www.wolfram.com/language/fast-introduction-for-math-students/en/

★Kurs online: Introduction for Programmers : https://www.wolfram.com/language/fast-introduction-for-programmers/en

„Not Just Numbers, Not Just Math—But Everything"

Symbolic Language

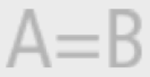
Mathematical Computation



Numerics

Visualization

Algebraic Manipulation



Number Theory

Data Analysis

Graph Computation

Interactive Computation

Image Computation

Geometric Computation

Importing & Exporting

http://www.wolfram.com/mathematica/?source=nav

Podstawowe operacje matematyczne (1)

★Wyrażenia obliczamy używając **SHIFT+ENTER** po wpisaniu wyrażenia (ang. evaluate)
★Obliczanie wyrażenia anulujemy kombinacja **ALT+**
★Poprzednie wyrażenie przywołujemy kombinacją **CTRL+L**
★Przybliżoną wartość numeryczną uzyskujemy dodając **//N** na końcu lub używając funkcji **N[wyrażenie, precyzja]**
★Wynik poprzedniego wyrażenia przywołujemy znakiem **%**
★Wynik poprzedniego obliczenia o numerze X przywołujemy używając **%X**

```
In[1]:= 2 + 2
Out[1]= 4
In[2]:= 2^2
Out[2]= 4
In[3]:= 1/5
Out[3]= 1/5
In[6]:= 1/7//N
Out[6]= 0.142857
In[7]:= N[1/7, 10]
Out[7]= 0.1428571429
In[8]:= %
Out[8]= 0.1428571429
In[9]:= %6
```

Podstawowe operacje matematyczne (2)

★Mnożenie zapisujemy jako **x*y** lub **x y ze spacja!** Bo xy to już nazwa!
★Nazwy funkcji i stałych zaczynają się od wielkiej litery, np **Sin[x]**
★Argument funkcji jest podawany w nawiasach kwadratowych []
★Dając kropkę na końcu argumentu, jako wynik otrzymamy przybliżenie numeryczne
★Funkcje trygonometryczne domyślnie wymagają argumentu w radianach
★Argument w stopniach przekazujemy jako X Degree

```
In[10]:= Sqrt[3]
Out[10]= √3
In[11]:= Exp[4]
Out[11]= e^4
In[14]:= Sin[2.0]
Out[14]= 0.909297
In[15]:= Sin[2 Degree]
Out[15]= Sin[2 °]
In[16]:= Pi
Out[16]= π
In[17]:= Infinity
Out[17]= ∞
In[18]:= Sqrt[-1]
```

Własne definicje

★Zmienne definiujemy używając znaku = (zaleca się używania małych liter w nazwach własnych zmiennych)
★Mozemy użyć zmiennych by przechowywać wartości liczbowe obliczeń
★Kiedy już nie potrzebujemy zmiennej lub funkcji należy je usunąć używając **nazwa=.** lub **Clear[nazwa]**

```
In[1]:= x = 5

Out[1]= 5

In[2]:= 2 * x

Out[2]= 10

In[3]:= 3 x

Out[3]= 15

In[5]:= Sin[181. Degree]

Out[5]= -0.0174524

In[6]:= x = %5

Out[6]= -0.0174524

In[7]:= 3 x

Out[7]= -0.0523572

In[8]:= x =.

In[9]:= x
```

Własne definicje (2)

★Funkcje definiujemy używając składni nazwa[x_]:=wzór
★UWAGA! Pamietajmy by NIE mieć zmiennych o nazwie używanej w definicji funkcji np. x
★Definicje funkcji własnych i wbudowanych możemy sprawdzić używając polecenia **?Nazwa**
★Kiedy już nie potrzebujemy zmiennej lub funkcji należy je usunąć używając **nazwa=.** lub **Clear[nazwa]**

```
f[x_] := x^2

?f
```

```
Global`f
```

```
f[x_] := x^2

f[16]

256

Clear[f]

?f
```

```
Global`f
```

Obliczenia symboliczne

`D[f, x]`	the (partial) derivative $\frac{\partial f}{\partial x}$
`Integrate[f, x]`	the indefinite integral $\int f\, dx$
`Sum[f, {i, imin, imax}]`	the sum $\sum_{i=imin}^{imax} f$
`Solve[lhs==rhs, x]`	solution to an equation for x
`Series[f, {x, x0, order}]`	a power series expansion of f about the point $x = x_0$
`Limit[f, x->x0]`	the limit $\lim_{x \to x_0} f$
`Minimize[f, x]`	minimization of f with respect to x

```
In[1]:= f[x_] := Sin[2 x]

In[2]:= ?f

        Global`f

        f[x_] := Sin[2 x]

In[3]:= D[f[x], x]

Out[3]= 2 Cos[2 x]

In[4]:= Integrate[f[x], x]

Out[4]= -1/2 Cos[2 x]

In[7]:= Solve[f[x] == 0, x]

        Solve::ifun :
          Inverse functions are being used by Solve,

Out[7]= {{x → 0}}

In[8]:= Series[f[x], {x, 0, 4}]

Out[11]= 2 x - 4 x^3/3 + O[x]^5

In[13]:= Limit[Sin[x] /x, x → 0]

Out[13]= 1

In[9]:= g[x_] := x^2

In[10]:= Minimize[g[x], x]

Out[10]= {0, {x → 0}}
```

Obliczenia symboliczne

`Expand[expr]`
expands out products and positive integer powers in *expr*.

`Factor[poly]`
factors a polynomial over the integers.

`Simplify[expr]`
performs a sequence of algebraic and other transformations on *expr*, and returns the simplest form it finds.

`Simplify[expr, assum]`
does simplification using assumptions.

`FullSimplify[expr]`
tries a wide range of transformations on *expr* involving elementary and special functions, and returns the simplest form it finds.

```
In[15]:= Expand[(a + b)^5]
Out[15]= a^5 + 5 a^4 b + 10 a^3 b^2 + 10 a^2 b^3 + 5 a b^4 + b^5

In[16]:= Factor[6 + 11 x + 6 x^2 + x^3]
Out[16]= (1 + x) (2 + x) (3 + x)

In[18]:= Sum[x^n / n!, {n, 0, Infinity}]
Out[18]= e^x

In[38]:= Simplify[1/(3 (1 + x)) - (-1 + 2 x)/(6 (1 - x + x^2)) + 2/(3 (1 + 1/3 (-1 + 2 x)^2))]
Out[38]= 1/(1 + x^3)
```

Obliczenia numeryczne

`N[expr]`	numerical value of an expression
`NIntegrate[f, {x, xmin, xmax}]`	numerical approximation to $\int_{x_{min}}^{x_{max}} f\, dx$
`NSum[f, {i, imin, Infinity}]`	numerical approximation to $\sum_{i_{min}}^{\infty} f$
`FindRoot[lhs==rhs, {x, x0}]`	search for a numerical solution to an equation, starting with $x = x_0$
`NSolve[lhs==rhs, x]`	numerical approximations to all solutions of an equation
`FindMinimum[f, {x, x0}]`	search for a minimum of f, starting with $x = x_0$
`NMinimize[f, x]`	attempt to find the global minimum of f

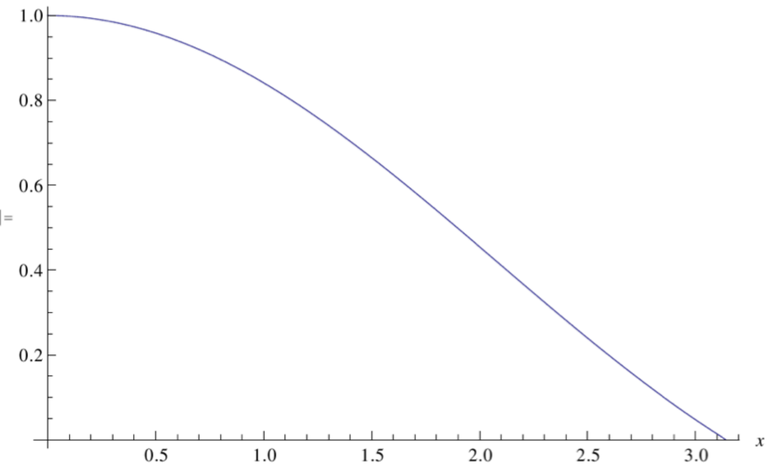
```
In[14]:= f[x_] := Sin[x] - Exp[x]

In[16]:= N[f[1]]
Out[16]= -1.87681

In[17]:= NIntegrate[f[x], {x, 0, 1}]
Out[17]= -1.25858

In[19]:= FindRoot[f[x] == -1.8, {x, 1}]
Out[19]= {1 → 0.963666}

In[27]:= f[x /. %19]
Out[27]= -1.8
```

Wykresy funkcji

In[31]:= $f[x\_] := \frac{Sin[x]}{x}$

In[36]:= `Plot[f[x], {x, 0, Pi}, AxesLabel → Automatic]`

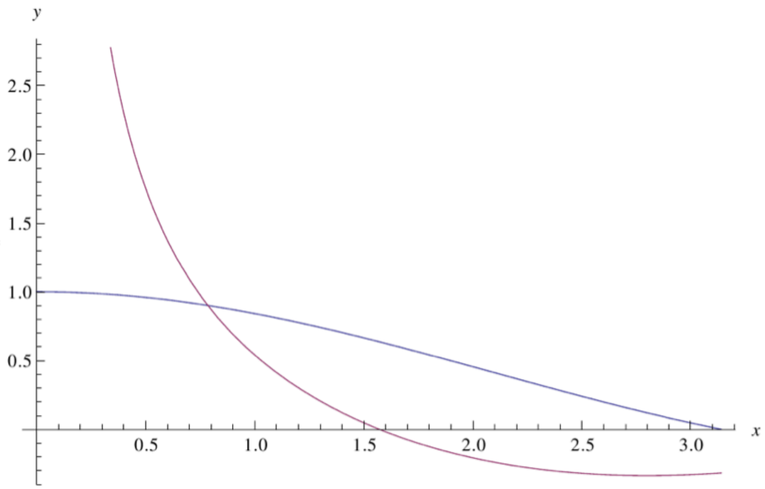
Out[36]=



In[34]:= $g[x\_] := \frac{Cos[x]}{x}$

In[37]:= `Plot[{f[x], g[x]}, {x, 0, Pi}, AxesLabel → {x, y}]`

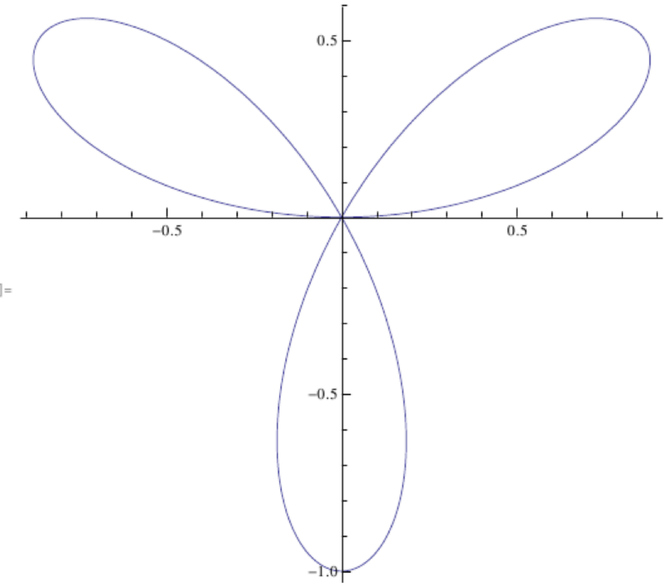
Out[37]=



Wykresy funkcji

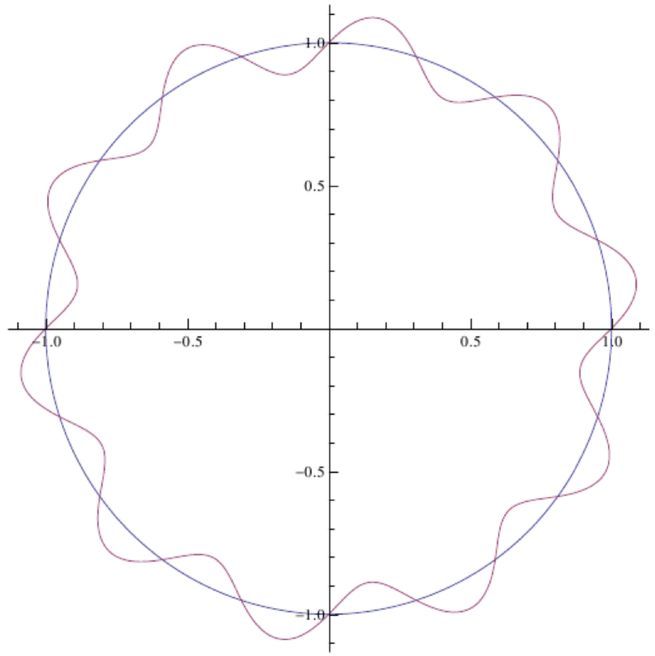
Zmienne biegunowe

In[47]:= **PolarPlot[Sin[3 t], {t, 0, Pi}]**



Out[47]=

In[48]:= **PolarPlot[{1, 1 + 1/10 Sin[10 t]}, {t, 0, 2 Pi}]**



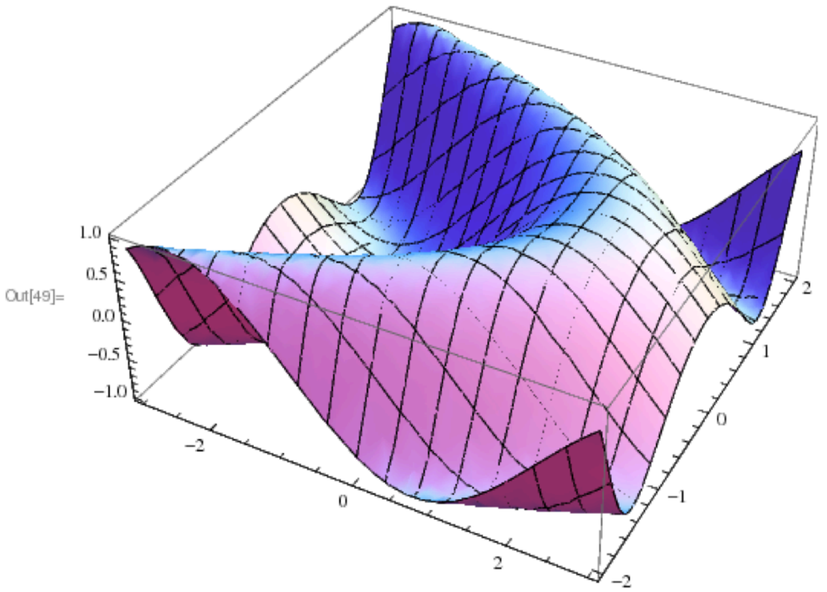
Out[48]=

Wykresy funkcji

Wykresy typu 3D

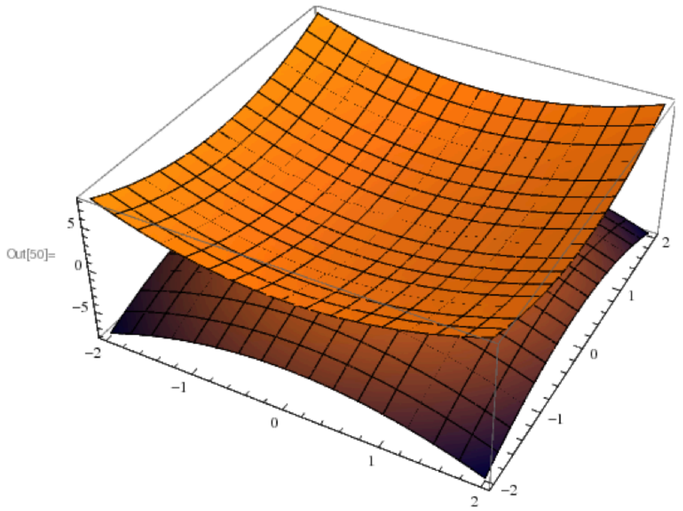
In[49]:= `Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}]`

Out[49]=



In[50]:= `Plot3D[{x^2 + y^2, -x^2 - y^2}, {x, -2, 2}, {y, -2, 2}, ColorFunction → "RustTones"]`

Out[50]=



Dopasowanie funkcji do danych doświadczalnych

- Mathematica wykorzystuje dwie podstawowe metody do dopasowań funkcji do danych doświadczalnych:
  - Metoda LinearModelFit[] oraz NonLinearModelFit[]
  - Obie metody są wyjaśnione np. tutaj: https://www.youtube.com/watch?v=KolZZm8If9Q

Dopasowanie funkcji do danych doświadczalnych - przykład

★Posługując się poleceniem SetDirectory oraz Import wczytaj dane z pliku do macierzy Dane

```
SetDirectory["/dmj/2000/mzpos/_work_/mathematica"]
/work/2000/mzpos/mathematica

FileNames["*", "/work/2000/mzpos/mathematica"]
{/work/2000/mzpos/mathematica/prackomp.html, /work/2000/mzpos/mathematica/Z3_data.dat}

"/work/2000/mzpos/mathematica/Z3_data.dat"
/work/2000/mzpos/mathematica/Z3_data.dat

Dane = Import["Z3_data.dat", "Table"]
{{-0.09, -0.06608}, {-0.08, -0.05885}, {-0.07, 0.11019}, {-0.06, 0.11501},
 {-0.05, 0.1199}, {-0.04, -0.02969}, {-0.03, -0.02232}, {-0.02, -0.01492},
 {-0.01, -0.00748}, {0, 0}, {0.01, 0.00752}, {0.02, 0.01508}, {0.03, 0.02269},
```

Dopasowanie funkcji do danych doświadczalnych - przykład

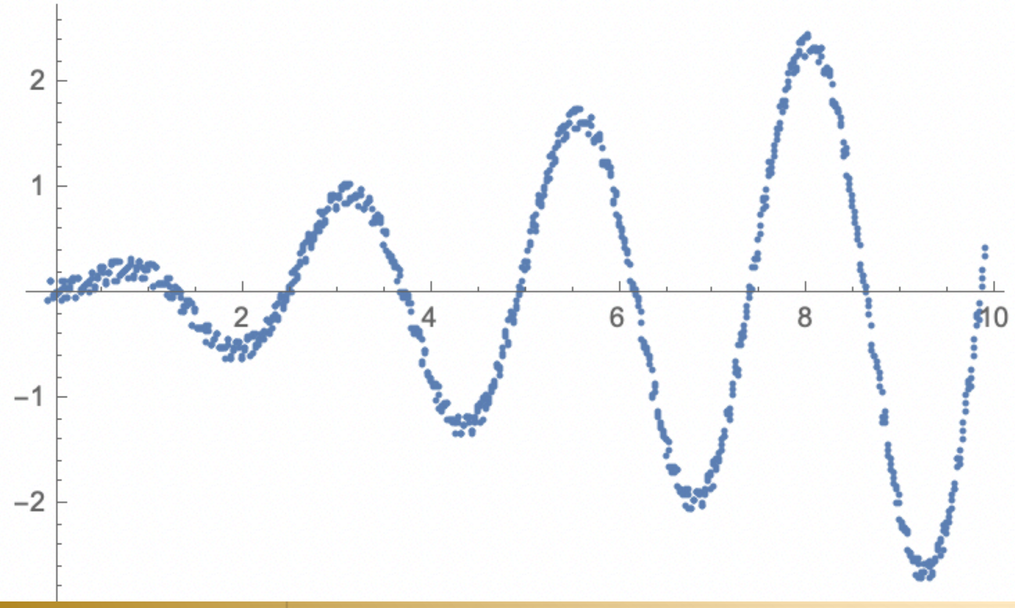
★Sprawdź rozmiar macierzy Dane.
★Wyświetl na ekranie : pierwsza kolumnę macierzy, pierwszy wiersz, element trzeciego wiersza drugiej kolumny

```
Dimensions[Dane]
{1000, 2}

Dane[[All, 1]]
{-0.09, -0.08, -0.07, -0.06, -0.05, -0.04, -0.03, -0.02, -0.01, 0, 0.01, 0.02, 0.03, 0.04,
 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18,
 0.19, 0.2, 0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3, 0.31, 0.32,
 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4, 0.41, 0.42, 0.43, 0.44, 0.45, 0.46,

Dane[[1, All]]
{-0.09, -0.06608}

Dane[[3, 2]]
0.11019
```

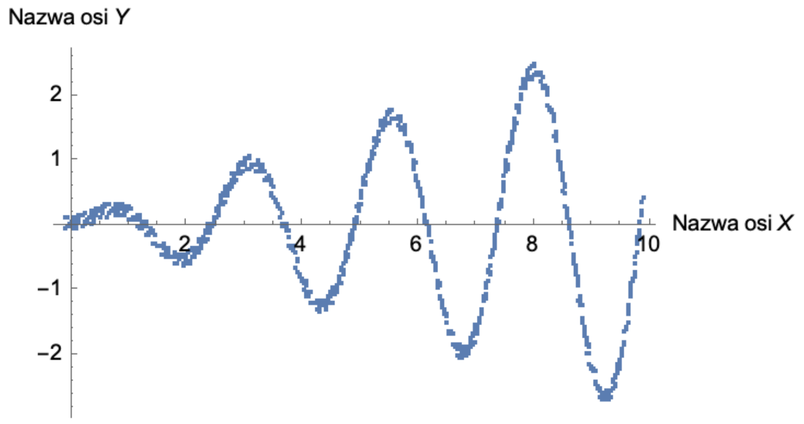
Dopasowanie funkcji do danych doświadczalnych - przykład

★Wyświetl na ekranie wykres przedstawiający pobrane dane przyjmując, ze zmienna niezależna znajduje się w kolumnie 1 a zależna w 2- skorzystaj z polecenia ListPlot

```
WykresDane = ListPlot[Dane]
```




```
Show[WykresDane, AxesLabel → {HoldForm[Nazwa osi X], HoldForm[Nazwa osi Y]},
 PlotLabel → None, LabelStyle → {GrayLevel[0]}]
```



Dopasowanie funkcji do danych doświadczalnych - przykład

★Posługując się poleceniem NonlinearModelFit:

★Dopasuj do danych funkcję $f(x) = ax\sin(bx + c) + d$

★Wyświetl na ekranie tabele z otrzymanymi parametrami dopasowania i ich błędami

★Narysuj wykres przedstawiający dopasowana funkcje

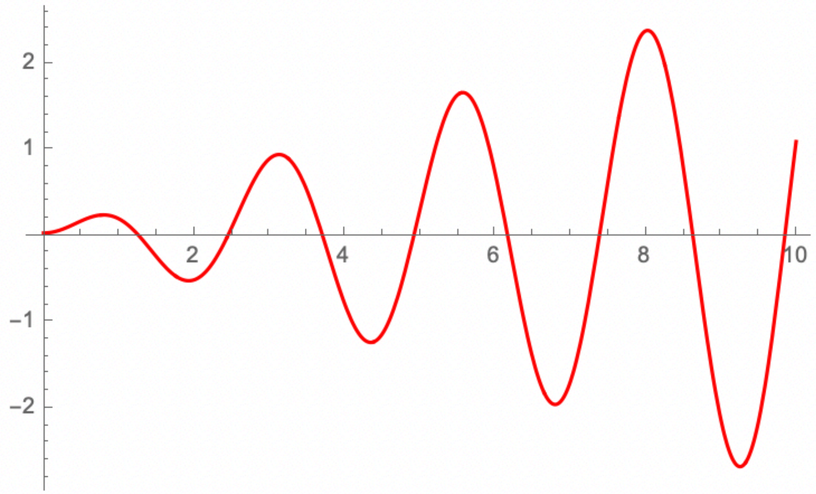
```
nlm = NonlinearModelFit[Dane, a * x * Sin[b * x + c] + d, {a, b, c, d}, x]
```

```
FittedModel[ 0.0360626 – 0.29296 x Sin[12.573 – 2.55094 x] ]
```

```
nlm["ParameterTable"]
```

	Estimate	Standard Error	t-Statistic	P-Value
a	0.29296	0.00052144	561.829	$1.775846986902\times10^{-1248}$
b	2.55094	0.000896797	2844.5	$2.083350964188\times10^{-1949}$
c	–12.573	0.00692377	–1815.91	$2.580789488702\times10^{-1755}$
d	0.0360626	0.00207517	17.3781	2.75755×10^{-59}

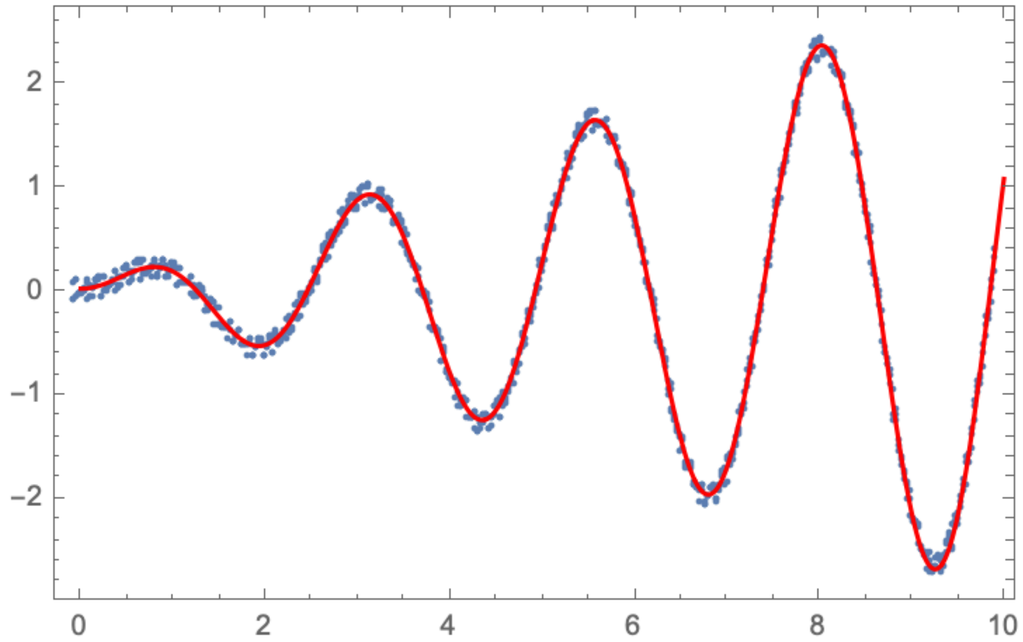
```
WykresFitu = Plot[nlm["BestFit"], {x, 0, 10}, PlotStyle → Red]
```



Dopasowanie funkcji do danych doświadczalnych - przykład

★Na jednym wykresie narysuj wykres z danymi oraz funkcje otrzymana z dopasowania używając polecenia Show

```
Show[WykresDane, WykresFitu, Frame → True, Axes → None]
```




Thank you!
Photo Credit: Piotr Mijakowski
ν_e
ν_μ
ν_τ