Wstęp do środowiska Mathematica

Kontakt: mkuich@fuw.edu.pl

Mteriały: `www.fuw.edu.pl/~mkuich/tik2022/`

Program Mathematica

- Mathematica to jeden z najbardziej popularnych programów do wykonywania obliczeń symbolicznych i numerycznych
- Inne podobne programy to komercyjny MAPLE lub darmowa MAXIMA
- Każdy student ma dostęp do darmowej licencji: www.fuw.edu.pl → Dla studentów → Oprogramowanie → Mathematica
- Program uruchamiamy z Menu: "Programy" →" "Wolfram Mathematica" → "Mathematica 13" lub linii poleceń:

```
[mkuich@workstation:~]$ mathematica
```

- W oknie powitalnym tworzymy nowy "Notebook", lub otwieramy już istniejący dokument
- Otwieramy dokumentację "Help" → "Wolfram Documentation"

Podstawowe operacje matematyczne

- Wyrażenia obliczamy (ang. *evaluate*) kombinacją klawiszy **Shift+Enter**, lub z menu: "Evaluation" → "Evaluate cells"
- Trwające obliczanie wyrażenia anulujemy kombinacją **Alt+.**
- Poprzednie wyrażenie (In) przywołujemy kombinacją **Ctrl+L**
- Wynik poprzedniego obliczenia (Out) przywołujemy znakiem **%**
- Wynik poprzedniego obliczenia o numerze X (Out[X]) przywołujemy **%X**
- Przybliżoną wartość numeryczną uzyskujemy dodając **//N** na końcu wyrażenia, lub za pomocą funkcji **N[wyrażenie, precyzja]**

```
In[1]:= 3 + 4
Out[1]= 7

In[2]:= 2^3
Out[2]= 8

In[3]:= % - 6
Out[3]= 2

In[4]:= 1 / %1
Out[4]= 1/7

In[5]:= 1 / 7 // N
Out[5]= 0.142857

In[6]:= N[1 / 7, 10]
Out[6]= 0.1428571429
```

Podstawowe funkcje matematyczne

- Mnożenie zapisujemy jako **x*y** lub **x y**, ale **xy** jest już nazwą zmiennej
- Nazwy funkcji i stałych zaczynają się od wielkiej litery, np.: **Sin[x]**
- Argument funkcji jest podawany w nawiasach kwadratowych **[]**
- Dodając kropkę na końcu argumentu, jako wynik otrzymamy przybliżenie numeryczne
- Funkcje trygonometryczne domyślnie wymagają agrumentu w radianach; argument w stopniach przekazujemy jako **X Degree**

```
In[7]:= Sqrt[3]
Out[7]= √3

In[8]:= Exp[4]
Out[8]= e^4

In[9]:= Sin[2.0]
Out[9]= 0.909297

In[10]:= Sin[2 Degree]
Out[10]= Sin[2 °]

In[11]:= Pi
Out[11]= π

In[12]:= Infinity
Out[12]= ∞

In[13]:= Sqrt[-1]
Out[13]= i
```

Własne definicje - zmienne

- Zmienne definiujemy używając znaku = (zaleca się używania małych liter w nazwach własnych zmiennych)
- Możemy użyć zmiennych do przechowywania wartości liczbowych obliczeń
- Kiedy już nie potrzebujemy zmiennej, należy ją usunąć używając **nazwa=.** lub **Clear[nazwa]**

```
In[15]:= x = 5
Out[15]= 5

In[16]:= 2 * x
Out[16]= 10

In[17]:= x = Sin[Pi / 3]
Out[17]= Sqrt[3]/2

In[18]:= 3 x
Out[18]= 3 Sqrt[3]/2

In[19]:= x =.

In[20]:= x
Out[20]= x
```

Własne definicje - funkcje

- Funkcje definiujemy używając składni **nazwa[x_]:=wzór**
- Uwaga:pamiętajmy, by nie mieć zmiennych o nazwie używanej w definicji funkcji
- Uwaga:pamiętajmy,by funkcje definiować prze znak **:=** nie **=**
- Definicję funkcji własnych i wbudowanych możemy sprawdzić używając **?Nazwa**
- Kiedy już nie potrzebujemy funkcji, należy ją usunąć używając **nazwa=.** lub **Clear[nazwa]**

```
In[21]:= f[x_] := x^2

In[22]:= ?f

Out[22]= Symbol
         Global`f
         Definitions f[x_] := x^2
         Full Name Global`f

In[23]:= f[16]

Out[23]= 256

In[24]:= Clear[f]

In[25]:= ?f

Out[25]= Symbol
         Global`f
         Full Name Global`f
```

Obliczenia symboliczne - przykłady 1

`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 $\sum_{i=i_{min}}^{i_{max}} f$
`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

- Do elementów złożonego wyniku dostajemy się np. przez **%X[[1]]**
- Podstawienie liczby lub poprzedniego wyniku realizujemy przez znak /., np. **x/.%**

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

Symbol

Global`f

Definitions f[x_] := Sin[2 x]
Full Name Global`f

```
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[5]:= Solve[f[x] == 0, x]
Out[5]= {{x → ConditionalExpression[π c1, c1 ∈ ℤ]},
         {x → ConditionalExpression[1/2 (π + 2 π c1), c1 ∈ ℤ]}}
In[6]:= Simplify[f[x] /. %[[1]]]
Out[6]= ConditionalExpression[0, c1 ∈ ℤ]
In[7]:= Series[f[x], {x, 0, 4}]
Out[7]= 2 x - 4 x^3/3 + O[x]^5
In[8]:= Limit[Sin[x] / x , x → 0]
Out[8]= 1
In[9]:= g[x_] := x ^ 2
In[10]:= Minimize[g[x], x]
Out[10]= {0, {x → 0}}
```

Obliczenia symboliczne - przykłady 2

```
In[11]:= Expand[(a + b)^5]
Out[11]= a^5 + 5 a^4 b + 10 a^3 b^2 + 10 a^2 b^3 + 5 a b^4 + b^5

In[12]:= Factor[6 + 11 x + 6 x^2 + x^3]
Out[12]= (1 + x) (2 + x) (3 + x)

In[13]:= Sum[x^n / n!, {n, 0, ∞}]
Out[13]= e^x

In[17]:= Simplify[1/(3 (1 + x)) - (-1 + 2 x)/(6 (1 - x + x^2)) + 2/(3 (1 + 1/3 (-1 + 2 x)^2))]
Out[17]= 1/(1 + x^3)
```

`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.

Obliczenia numeryczne - przykłady 1

N[*expr*]	numerical value of an expression
NIntegrate[f, {x, x_{min}, x_{max}}]	numerical approximation to $\int_{x_{min}}^{x_{max}} f\,dx$
NSum[f, {i, i_{min}, Infinity}]	numerical approximation to $\sum_{i_{min}}^{\infty} f$
FindRoot[*lhs*==*rhs*, {x, x_0}]	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, x_0}]	search for a minimum of f, starting with $x = x_0$
NMinimize[f, x]	attempt to find the global minimum of f

```
In[18]:= Clear[f]

In[19]:= f[x_] := Sin[x] - Exp[x]

In[20]:= ? f

Out[20]= Symbol
         Global`f
         Definitions f[x_] := Sin[x] - Exp[x]
         Full Name Global`f

In[21]:= N[f[1]]

Out[21]= -1.87681

In[22]:= NIntegrate[f[x], {x, 0, 1}]

Out[22]= -1.25858

In[23]:= FindRoot[f[x] == -1.8, {x, 1}]

Out[23]= {x → 0.963666}

In[24]:= f[x /. %]

Out[24]= -1.8
```

- Do elementów złożonego wyniku dostajemy się np. przez **%X[[1]]**
- Podstawienie liczby lub poprzedniego wyniku realizujemy przez znak /., np. **x/.%**

Zadania 1 - obliczenia symboliczne i numeryczne

1. Wypisz wartość liczby Eulera z dokładnością do 100 cyfr
2. Oblicz $\sin(\log(-1)$, podaj wartość numeryczną
3. Oblicz pierwiastek czwartego stopnia z π
4. Wymnóż $(a+b+c)^7$
5. Przeprowadź faktoryzację wielomianu
   $2x + 4x^2 + 3x^3 + 2x^4 + 3x^5 + 3x^7 + x^9$
6. Rozwiąż równanie $x^3 + 6x^2 + 11x + 6 = 0$ i sprawdź wszystkie miejsca zerowe
7. Rozwiąż równanie $ax^2 + bx + c = 0$, sprawdź i uprość rozwiązania
8. Rozwiń w szereg potęgowy funkcje (do 5 rzędu rozwinięcia):
   $\cos(x)$ wokół $x = 0$
   $e^{\sin(\log(x))}$ wokół $x = 1$ i $x = 2$
9. Oblicz sumy: $S = 1 + 2 + ... + 100$
   $S = \frac{1}{1\cdot 2\cdot 3} + \frac{1}{2\cdot 3\cdot 4} + ... + \frac{1}{n\cdot(n+1)\cdot(n+2)}$

Zadania 2 - obliczenia symboliczne i numeryczne

10. Oblicz granice: $\lim_{n\to\infty} \sqrt[100]{n^{100}+n^{99}} - n$ oraz $\lim_{n\to\infty} \frac{1^5+2^5+\ldots+n^5}{n^6}$
11. Oblicz pochodną pierwszego stopnia po zmiennej x z funkcji: $w(x) = ax^5 + (b+1)x^3 + 7x + 1$ oraz $g(x) = x^2 \exp(-3ax^3)$
12. Oblicz $f^{10}(x)$ oraz $f^{10}(0)$ dla $f(x) = x^2 \cos(2x)$
13. Oblicz symbolicznie następujące całki i sprawdź je licząc pochodne:
    a) $\int (x^2 - 2x + 3)\exp(x)dx$ b) $\int \sqrt{x}(\log x)^2 dx$ c) $\int \frac{1}{x^3+1} dx$
14. Oblicz całki oznaczone:
    a)
    $$\int_1^2 \exp(\sin(\log x))dx$$
    b)
    $$\int_0^{\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \sin(x)\cos(y)dxdy$$

Macierze 1

+, *****, **^**, **. . .** — all automatically work element-wise

Dot (**.**) — products of matrices, automatically handling row and column vectors

Inverse — matrix inverse (use **LinearSolve** for linear systems)

Transpose — transpose (m^{T}, entered with Esc tr Esc)

ConjugateTranspose — conjugate transpose ($m^{\dagger}$, entered with Esc ct Esc)

Tr — trace

Det — determinant

KroneckerProduct — matrix direct product (outer product)

MatrixPower — powers of numeric or symbolic matrices

MatrixExp — matrix exponential

Eigenvalues, **Eigenvectors** — exact or approximate eigenvalues and eigenvectors

Eigensystem — eigenvalues and eigenvectors together

CharacteristicPolynomial — symbolic characteristic polynomial

```
In[25]:= {{Cos[x], Sin[x]}, {-Sin[x], Cos[x]}}
Out[25]= {{Cos[x], Sin[x]}, {-Sin[x], Cos[x]}}

In[26]:= % // MatrixForm
Out[26]//MatrixForm=
( Cos[x]   Sin[x] )
( -Sin[x]  Cos[x] )

In[27]:= A = %25
Out[27]= {{Cos[x], Sin[x]}, {-Sin[x], Cos[x]}}

In[28]:= B = Transpose[A]
Out[28]= {{Cos[x], -Sin[x]}, {Sin[x], Cos[x]}}

In[29]:= A * B
Out[29]= {{Cos[x]^2, -Sin[x]^2}, {-Sin[x]^2, Cos[x]^2}}

In[30]:= A.B
Out[30]= {{Cos[x]^2 + Sin[x]^2, 0}, {0, Cos[x]^2 + Sin[x]^2}}

In[31]:= Simplify[%] // MatrixForm
Out[31]//MatrixForm=
( 1  0 )
( 0  1 )

In[32]:= Simplify[B.A] // MatrixForm
Out[32]//MatrixForm=
( 1  0 )
( 0  1 )

In[33]:= Simplify[Inverse[A] - B // MatrixForm]
Out[33]//MatrixForm=
( 0  0 )
( 0  0 )

In[34]:= Simplify[Det[A]]
Out[34]= 1
```

Macierze 2

Eigensystem

`Eigensystem[m]`
gives a list {*values*, *vectors*} of the eigenvalues and eigenvectors of the square matrix m.

`Eigensystem[{m, a}]`
gives the generalized eigenvalues and eigenvectors of m with respect to a.

`Eigensystem[m, k]`
gives the eigenvalues and eigenvectors for the first k eigenvalues of m.

`Eigensystem[{m, a}, k]`
gives the first k generalized eigenvalues and eigenvectors.

```
In[39]:= A = {{2, 0}, {0, 2}}
Out[39]= {{2, 0}, {0, 2}}

In[40]:= Eigensystem[A] // MatrixForm
Out[40]//MatrixForm=
( 2      2      )
( {0, 1} {1, 0} )

In[41]:= %40[[1]]
Out[41]= {2, 2}

In[42]:= %40[[1, 1]]
Out[42]= 2

In[44]:= x = %40[[2, 1]]
Out[44]= {0, 1}

In[45]:= (A.x) .x / Norm[x]
Out[45]= 2
```

- Dla wektorów ($\vec{x}$) i wartości (λ) własnych macierzy A zachodzi związek:

$$A \cdot \vec{x} = \lambda \cdot \vec{x}$$

Rozwiązywanie układów równań algebraicznych

`Solve[`$lhs==rhs, x$`]`	solve an equation, giving a list of rules for x
x`/.`$solution$	use the list of rules to get values for x
$expr$`/.`$solution$	use the list of rules to get values for an expression

`Solve[{`$lhs_1==rhs_1, lhs_2==rhs_2, \ldots$`},{`$x, y, \ldots$`}]`

solve a set of simultaneous equations for x, y, …

```
In[9]:= r1[x_, y_, z_] := 2 * x + y + 3 * z - 9
        r2[x_, y_, z_] := x - 2 * y + z + 2
        r3[x_, y_, z_] := 3 * x + 2 * y + 2 * z - 7

In[12]:= Solve[{r1[x, y, z] == 0, r2[x, y, z] == 0, r3[x, y, z] == 0}, {x, y, z}]
Out[12]= {{x → -1, y → 2, z → 3}}

In[14]:= r1[x /. %12[[1]], y /. %12[[1]], z /. %12[[1]]]
Out[14]= 0

In[15]:= r2[x /. %12[[1]], y /. %12[[1]], z /. %12[[1]]]
Out[15]= 0

In[16]:= r3[x /. %12[[1]], y /. %12[[1]], z /. %12[[1]]]
Out[16]= 0

In[17]:= r1[x /. %12[[1]], y /. %12[[1]], 2]
Out[17]= -3

In[19]:= r1[x, y, z] /. %12[[1]]
Out[19]= 0
```

Rozwiązywanie równań różniczkowych

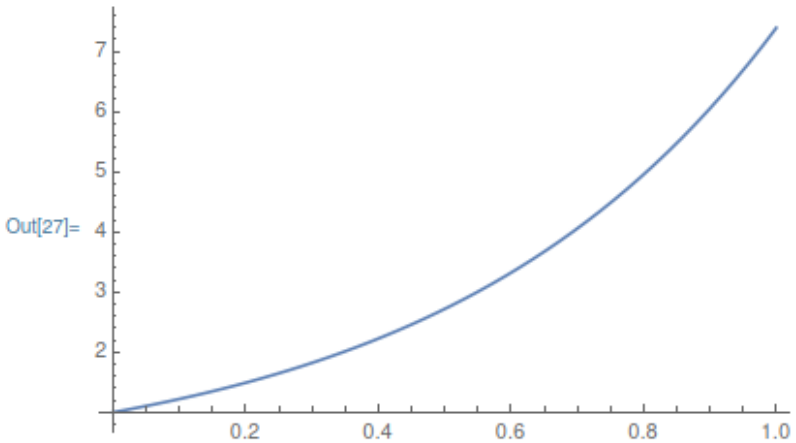
`DSolve[`eqn`, `y`, `x`]`
solves a differential equation for the function y, with independent variable x.

`DSolve[{`eqn_1`, `eqn_2`, ...}, {`y_1`, `y_2`, ...}, `x`]`
solves a list of differential equations.

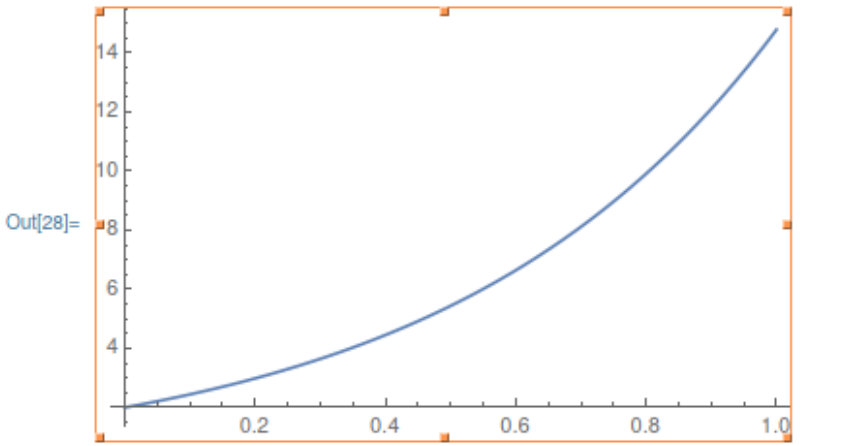
`DSolve[`eqn`, `y`, {`x_1`, `x_2`, ...}]`
solves a partial differential equation.

```
In[20]:= DSolve[y'[x] == y[x], y[x], x]
Out[20]= {{y[x] → e^x c1}}

In[22]:= DSolve[{y'[x] == a y[x], y[0] == 1}, y, x]
Out[22]= {{y → Function[{x}, e^(a x)]}}

In[24]:= y'[x] - a y[x] /. %22[[1]]
Out[24]= 0

In[25]:= y[x] /. %22[[1]]
Out[25]= e^(a x)

In[26]:= a = 2
Out[26]= 2
```

```
In[27]:= Plot[y[x] /. %22[[1]], {x, 0, 1}]
```



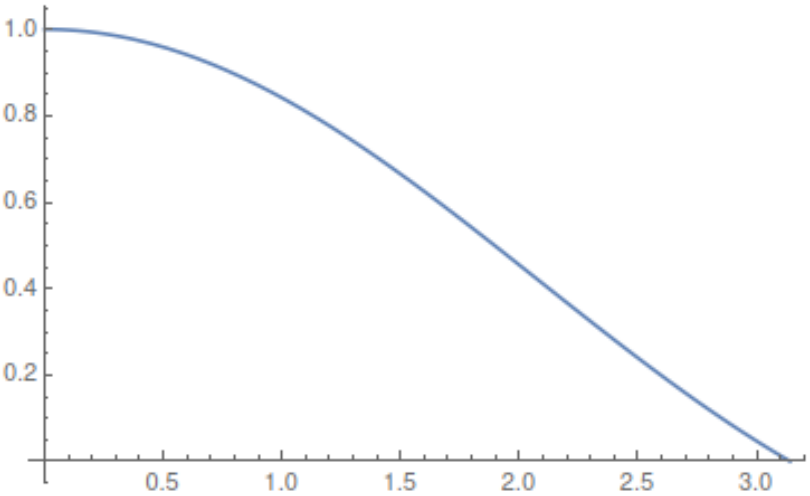

```
In[28]:= Plot[y'[x] /. %22[[1]], {x, 0, 1}]
```



Rysowanie wykresów funkcji

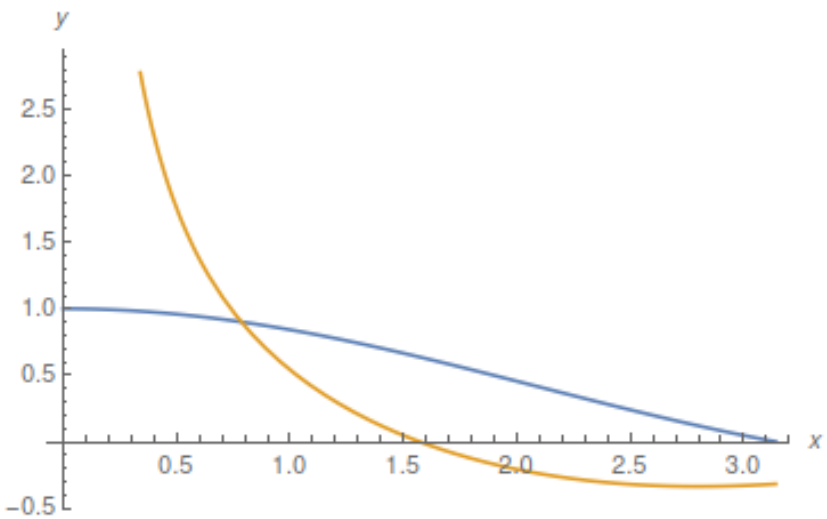
```
In[1]:= f[x_] := Sin[x] / x

In[2]:= Plot[f[x], {x, 0, Pi}]
```

Out[2]=



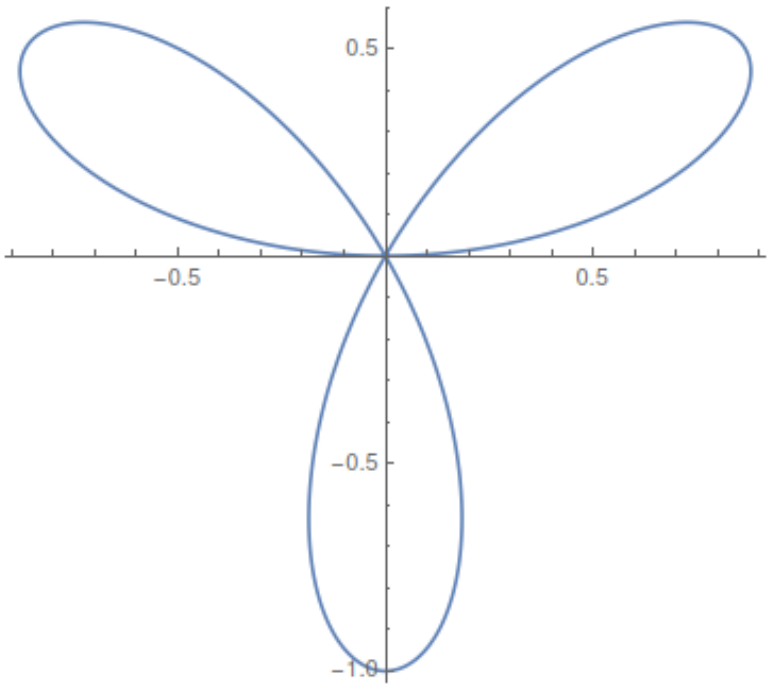

```
In[3]:= g[x_] := Cos[x] / x

In[4]:= Plot[{f[x], g[x]}, {x, 0, Pi}, AxesLabel → {x, y}]
```

Out[4]=



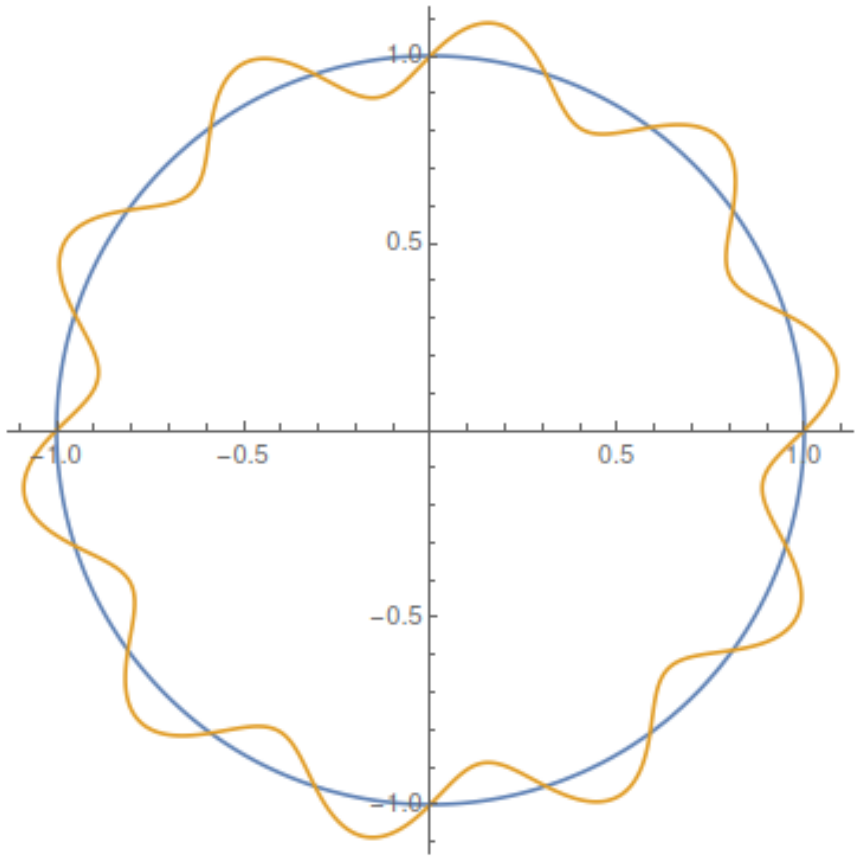
Rysowanie wykresów funkcji w układzie biegunowym

```
In[28]:= PolarPlot[Sin[3 t], {t, 0, Pi}]
```



Out[28]=

```
In[29]:= PolarPlot[{1, 1 + 1 / 10 Sin[10 t]}, {t, 0, 2 Pi}]
```

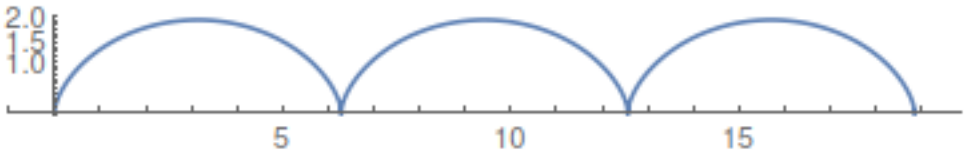


Out[29]=

Rysowanie wykresów parametrycznych

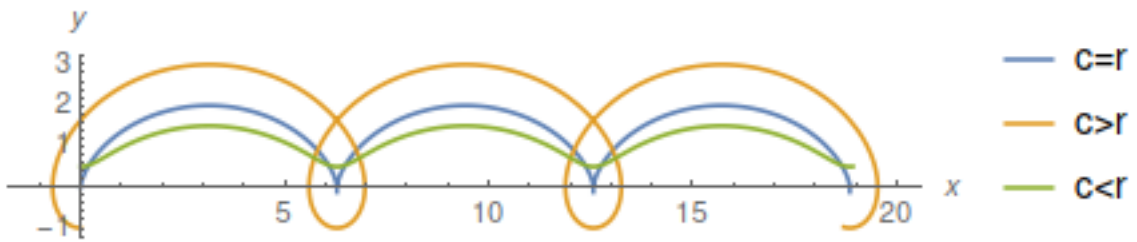
```
In[18]:= ParametricPlot[{t - Sin[t], 1 - Cos[t]}, {t, 0, 19}]
```

Out[18]=




```
In[20]:= ParametricPlot[{{t - Sin[t], 1 - Cos[t]}, {t - 2 Sin[t], 1 - 2 Cos[t]},
   {t - 0.5 Sin[t], 1 - 0.5 Cos[t]}}, {t, 0, 19}, AxesLabel → {x, y},
  PlotLegends → {"c=r", "c>r", "c<r"}]
```

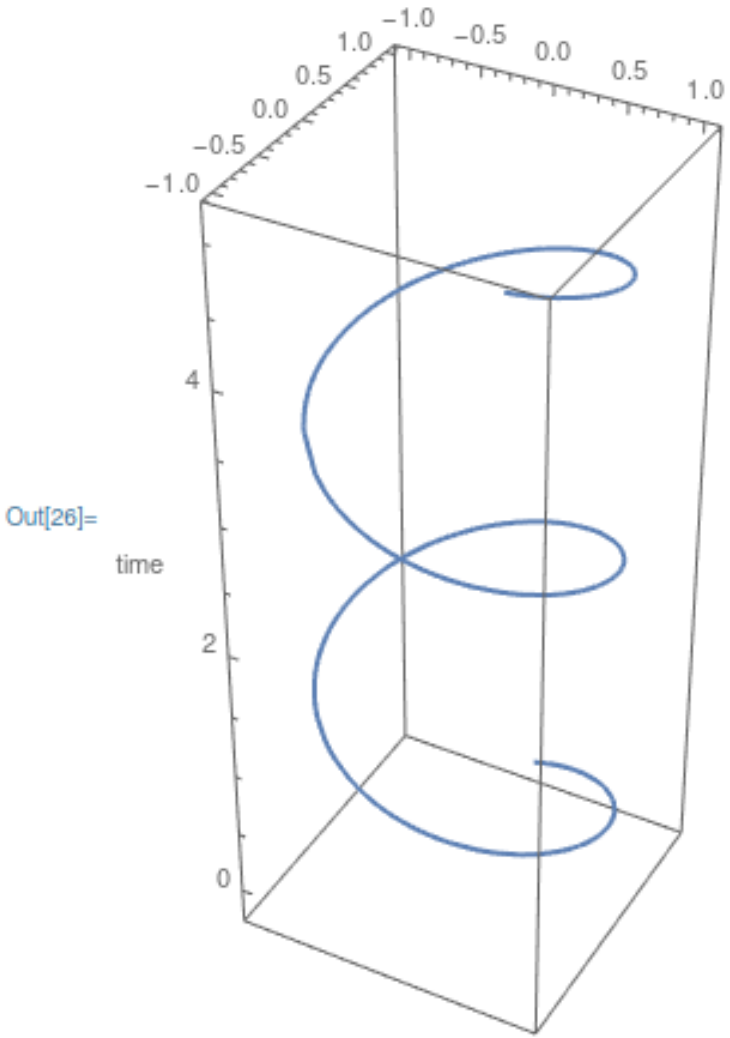
Out[20]=



Rysowanie wykresów parametrycznych 3D

In[26]:=
```
ParametricPlot3D[{Sin[t], Cos[t], t / 3}, {t, 0, 15}, AxesLabel → "time"]
```

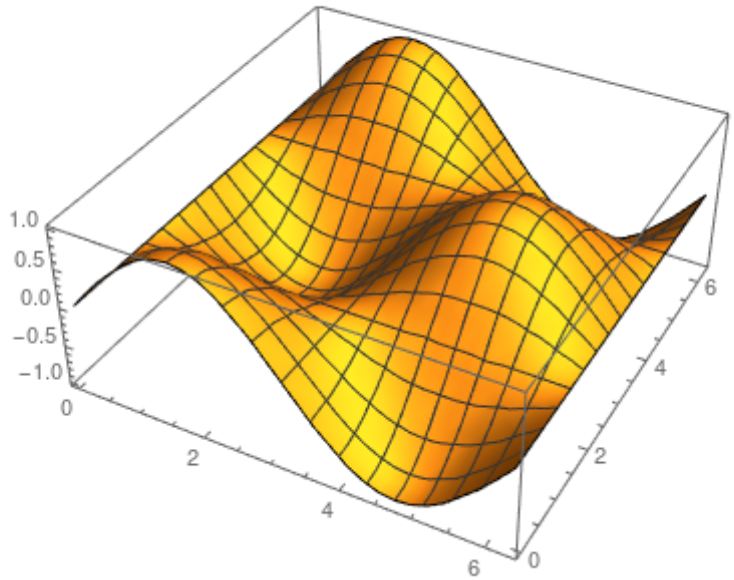
Out[26]=



Rysowanie wykresów 3D

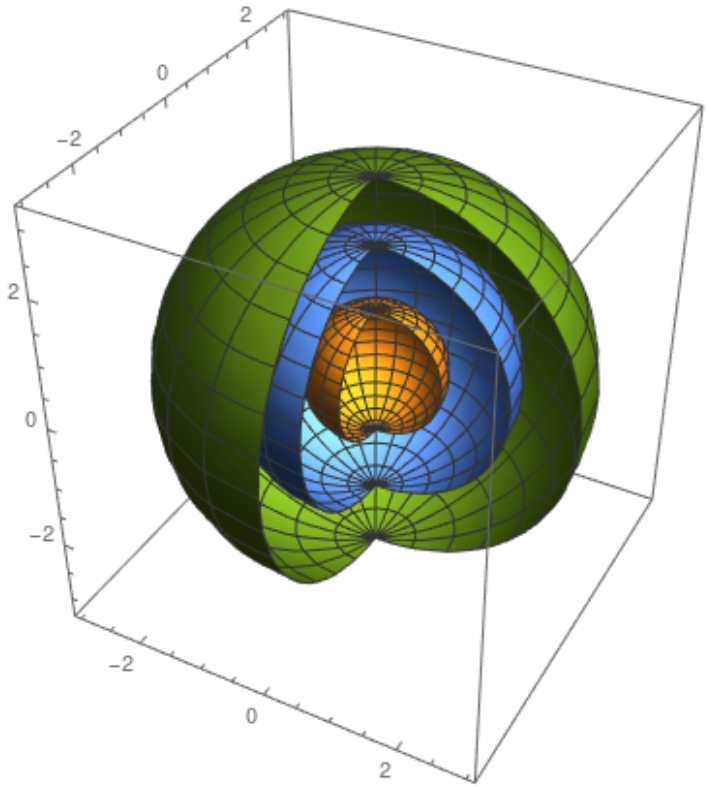
```
In[38]:= Plot3D[Sin[x] Cos[y], {x, 0, 2 Pi}, {y, 0, 2 Pi}]
```

Out[38]=




```
In[21]:= SphericalPlot3D[{1, 2, 3}, {θ, 0, Pi}, {ϕ, 0, 3 Pi / 2}]
```

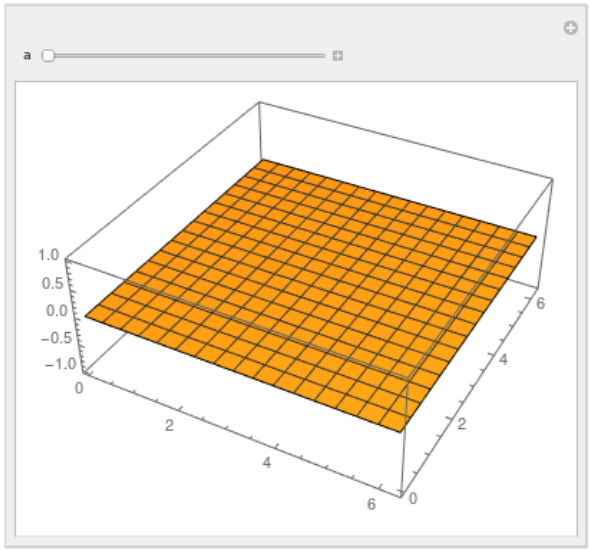
Out[21]=

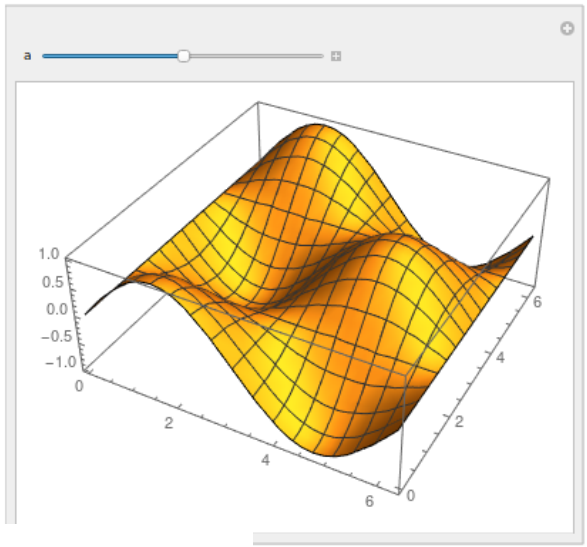


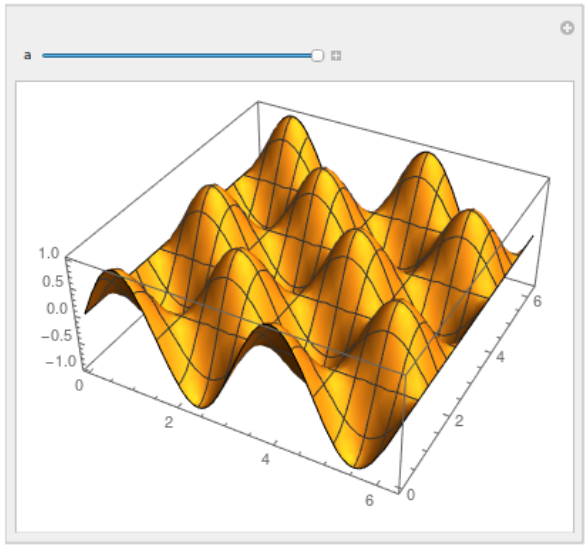
Manipulowanie współczynnikami i animowanie

```
In[39]:= Manipulate[Plot3D[Sin[a x] Cos[a y], {x, 0, 2 Pi}, {y, 0, 2 Pi}], {a, 0, 2}]
```

Out[39]=

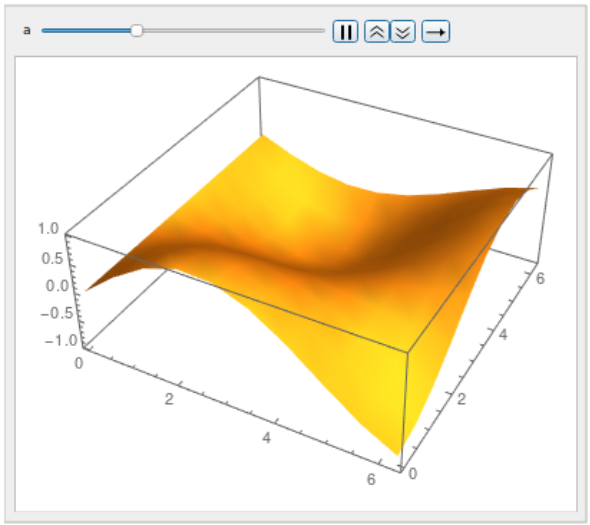







```
In[40]:= Animate[Plot3D[Sin[a x] Cos[a y], {x, 0, 2 Pi}, {y, 0, 2 Pi}], {a, 0, 2}]
```

Out[40]=



Zadania 3 - macierze

15. Oblicz A^{-1} oraz AA^{-1}, gdy:

$$A = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$$

16. Oblicz logarytm macierzy B element po elemencie oraz poprzez funkcję macierzową, gdy

$$B = \begin{pmatrix} 3 & 1 \\ -4 & -1 \end{pmatrix}$$

17. Znajdź wyznacznik macierzy M oraz jej wektory i odpowiadające im wartości własne. Następnie, sprawdź otrzymane wyniki.

$$M = \begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 2 \\ -1 & -1 & 4 \end{pmatrix}$$

Zadania 4 - układy równań

18. Rozwiąż układ równań:

$$\begin{cases} x - y + 2z = 1 \\ x - 2y - z = 2 \\ 3x - y + 5z = 3 \\ -2x + 2y + 3z = -4 \end{cases}$$

19. Rozwiąż układ równań, zakładając, że zmienne z i u mogą mieć dowolne wartości:

$$\begin{cases} x - y + 5z - u = 0 \\ x + y - 2z + 3u = 0 \\ 3x - y + 8z + u = 0 \\ x + 3y - 9z + 7u = 0 \end{cases}$$

Zadania 5 - układy równań różniczkowych

20. Znajdź rozwiązanie równania różniczkowego bez zakładania warunków brzegowych:

$$y''(x) + \omega_0^2 y(x) = 0$$

21. Znajdź rozwiązanie równania różniczkowego:

$$y''(x) + 2\beta y'(x) + \omega_0^2 y(x) = 0$$

zakładają warunki brzegowe: $y(0) = a$ i $y'(0) = 0$

Zadania 6 - rysowanie wykresów

22. Narysuj wykresy funkcji:
    a) $y(x) = \sqrt{(x+2) - \sin x}$ dla x od 0 do 10; podpisz osie x i y (AxesLabel)
    b) $y(x) = \sqrt{(x+2) - \sin x}$ oraz $\sin(\log(x))$ dla x od 0 do 10 na jednym wykresie; zacień obszar między tymi funkcjami (Filling)
    c) $y(x) = \Gamma(x)$ (funkcja Gamma Eulera) w zakresie od 0 do 5; dodaj legendę na wykresie (PlotLegends)
    d) $y(x) = \mathrm{tg}(1 + \log(x))$ w zakresie od 0 do 3; dodatnie wartości funkcji zaznacz na zielono, ujemne - na czerwono (ColorFunction)
    e) $r(\theta) = \exp(\theta/10)$ w układzie biegunowym, gdy θ zmienia się od 0 do 30 radianów; podpisz wykres definicją funkcji (PlotLabel)
    f) $z(x, y) = x^2 + y^2$ oraz $z(x, y) = x^2 - y^2$ dla x i y w zakresie od -1 do 1; zadbaj o proporcjonalność osi z względem osi x i y (BoxRatios)

Zadania 7 - manipulowanie wykresami

23. Utwórz interaktywne okno pozwalające na obserwowanie wykresów dwóch funkcji opisanych wzorami:

$$y(x) = \sin(x + \phi)$$

$$y(x) = \cos(x + \phi)$$

dla x w zakresie od 0 do 2π i manipulowanie parametrem ϕ w zakresie od 0 do π.

24. Utwórz interaktywne okno pozwalające na obserwowanie wykresu funkcji danej w układzie biegunowym wzorem:

$$r(\theta) = \log(1 + \theta) + a \sin(b\theta)$$

i manipulowanie parametrami a w zakresie od 0 do 2 i b w zakresie od 0 do 5. Zaanimuj otrzymany wykres.

Wczytywanie danych z pliku

- Wskazujemy położenie pliku z danymi → **SetDirectory[“...“]**
  - w miejsce ... wpisujemy ścieżkę do katalogu, w którym są dane
  - nazwa katalogu musi być zawarta między znakami ” i “
  - nie jest to niezbędne, ale wygodne
- Importujemy dane i przypisujemy je do nowej zmiennej w formie tabeli → np.: **Dane = Import[“line.txt“, ”Table“]**
  - źródłem importowania może być plik lokalny, obiekt z chmury/url, wynik działania jakiegoś programu
  - działamy na najprostszej wersji - lokalnie;
    jeśli nie zdefiniowaliśmy katalogu roboczego, w miejsce *line.txt* wpisujemy pełną ścieżkę
  - możemy importować dane w wielu formach i formatach (pliki tekstowe, rysunki, dźwięki, ...)
  - Mathematica próbuje rozpoznać typ danych po rozszerzeniu pliku
  - użytkownik może podać typ danych, jakie importuje, jako drugi argument funkcji Import, np.: **”Table“**

Wczytywanie danych z pliku - przykład

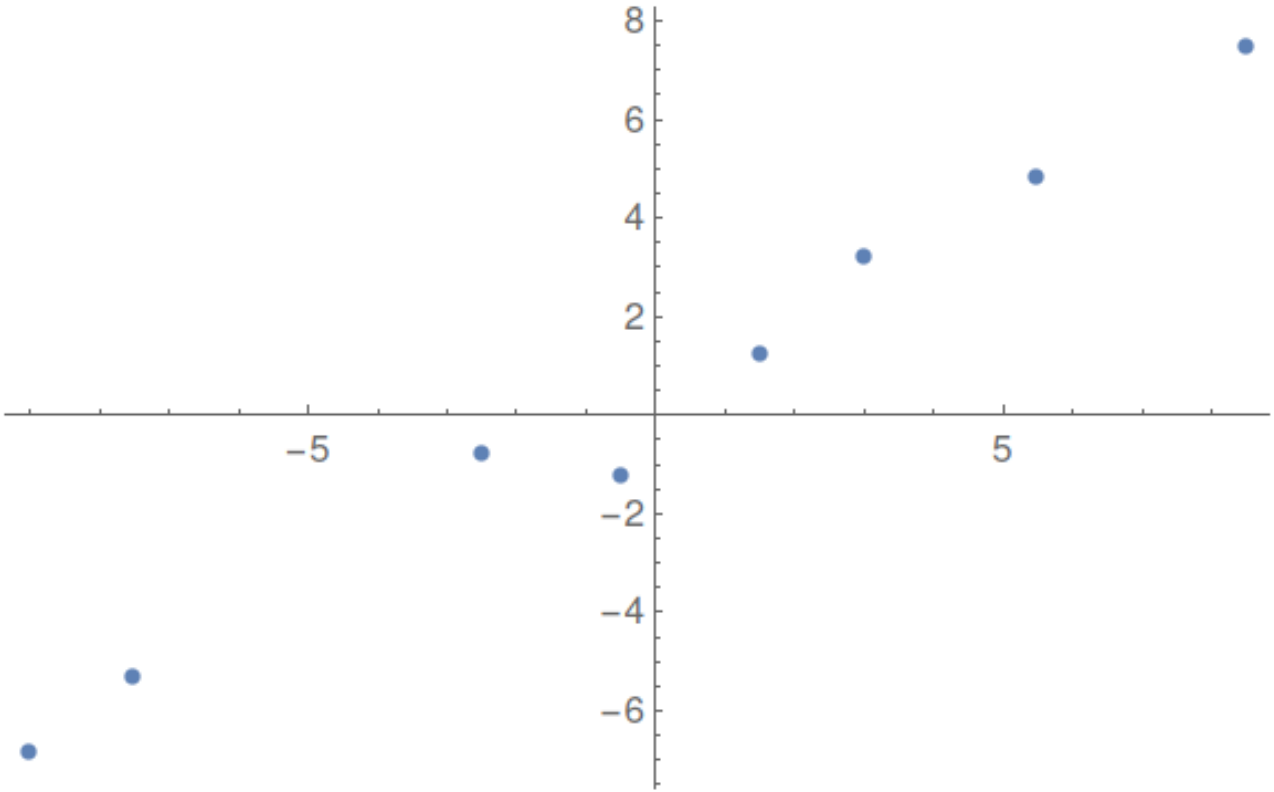
```
In[1]:= SetDirectory["/dane/Dydaktyka/TIK_2021/mathematica"]
Out[1]= /dane/Dydaktyka/TIK_2021/mathematica

In[2]:= Data = Import["line.txt", "Table"]
Out[2]= {{-9, -6.84307, 1}, {-7.5, -5.34808, 0.2}, {-2.5, -0.811542, 0.7}, {-0.5, -1.22456, 0.5},
         {1.5, 1.24481, 1.8}, {3, 3.19375, 1.1}, {5.5, 4.8006, 0.3}, {8.5, 7.47098, 1.2}}

In[4]:= DataX = Data[[All, 1]]
Out[4]= {-9, -7.5, -2.5, -0.5, 1.5, 3, 5.5, 8.5}

In[5]:= DataY = Data[[All, 2]]
Out[5]= {-6.84307, -5.34808, -0.811542, -1.22456, 1.24481, 3.19375, 4.8006, 7.47098}

In[6]:= DataYerr = Data[[All, 3]]
Out[6]= {1, 0.2, 0.7, 0.5, 1.8, 1.1, 0.3, 1.2}

In[7]:= Dimensions[Data]
Out[7]= {8, 3}

In[8]:= Length[Data]
Out[8]= 8
```

Wykres danych bez niepewności pomiarowych

- Do wizualizacji korzystamy z funkcji **ListPlot**
  - jako argument podajemy tablicę zawierającą x i y
  - formatowanie wykresu działa tak samo, jak w przypadku zwykłych wykresów (formatowanie podajemy, jako kolejne argumenty funkcji)

```
In[14]:= Wykres1 = ListPlot[Data[[All, {1, 2}]]]
```

Out[14]=

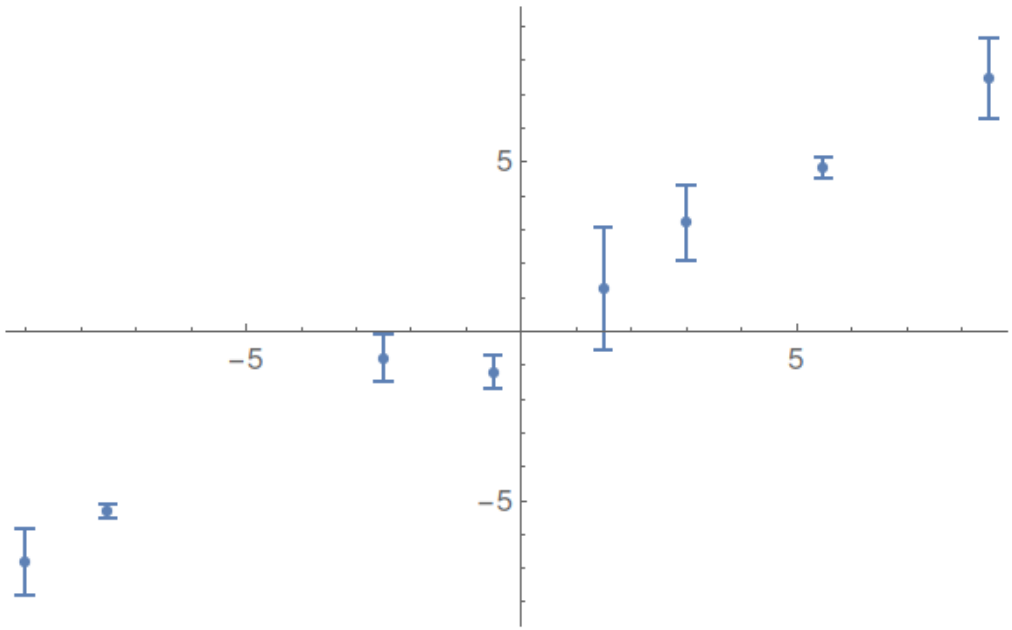


Wykres danych wraz niepewnościami pomiarowymi

- Od wersji 12 w górę korzystamy z funkcji **ListPlot** (w starszych wersjach korzystamy z funkcji ErrorListPlot)
  - niepewności pomiarowe przypisujemy do wartości za pomocą funkcji **Around**

In[15]:=
```
Wykres2 =
 ListPlot[{{-9, Around[-6.84307, 1]}, {-7.5, Around[-5.34808, 0.2]},
   {-2.5, Around[-0.811542, 0.7]}, {-0.5, Around[-1.22456, 0.5]},
   {1.5, Around[1.24481, 1.8]}, {3, Around[3.19375, 1.1]},
   {5.5, Around[4.8006, 0.3]}, {8.5, Around[7.47098, 1.2]}}]
```

Out[15]=

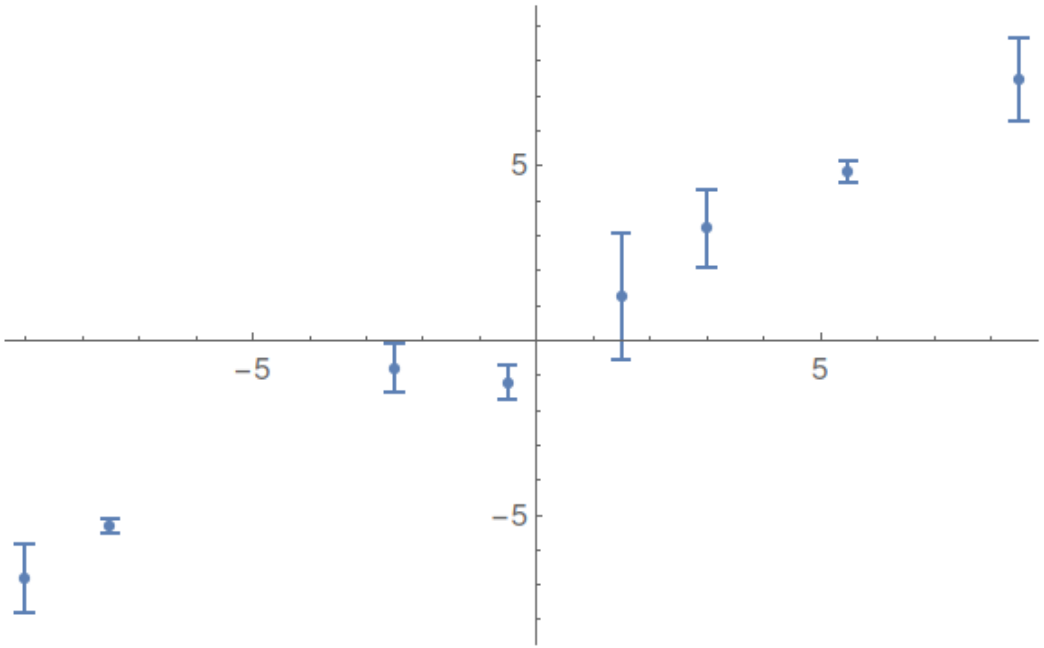


Wykres danych wraz niepewnościami pomiarowymi

- Aby przypisać niepewności punktom automatycznie, możemy skorzystać z funkcji **Table**

```
In[19]:= Wykres3 =
  ListPlot[Table[{Data[[i, 1]], Around[Data[[i, 2]], Data[[i, 3]]]},
    {i, 1, Length[Data]}]]
```

Out[19]=



Dopasowanie modelu liniowego do danych

- Model liniowy dopasowujemy za pomocą funkcji **LinearModelFit**
  - jako argumenty podajemy kolejno: dane wejściowe, model oraz zmienną-argument modelu (x)
  - **LinearModelFit** posiada wiele atrybutów (*property*), które pozwalają odczytać właściwości dopasowanego modelu, szczegóły w "Details and Options" w dokumentacji tej funkcji

```
In[21]:= lm = LinearModelFit[Data[[All, {1, 2}]], {1, x}, x]
```

Out[21]=

FittedModel[0.410412 + 0.800407 x]

```
In[22]:= lm["ParameterTable"]
```

Out[22]=

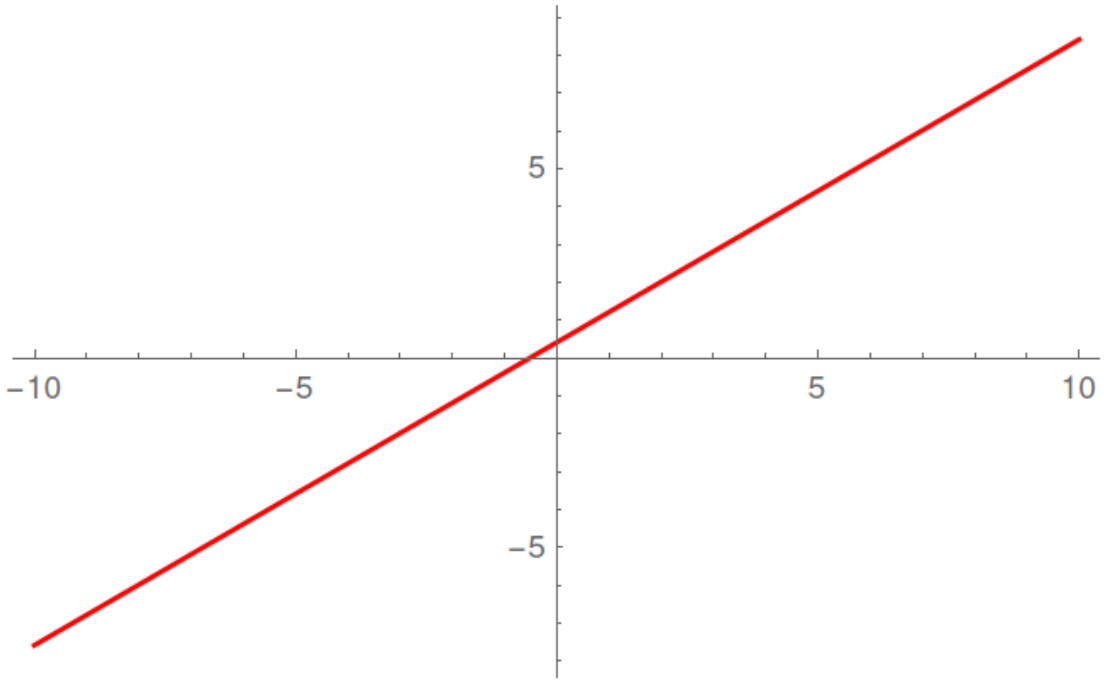
	Estimate	Standard Error	t-Statistic	P-Value
1	0.410412	0.230104	1.78359	0.124754
x	0.800407	0.0405583	19.7347	1.09771×10^{-6}

Wizualizacja dopasowanego modelu liniowego

- Z atrybutów funkcji **LinearModelFit** wybieramy "BestFit" (zawiera dopasowany model)
- Rysujemy model, korzystając z funkcji **Plot**

```
In[23]:= lmFitPlot = Plot[lm["BestFit"], {x, -10, 10}, PlotStyle → Red]
```

Out[23]=

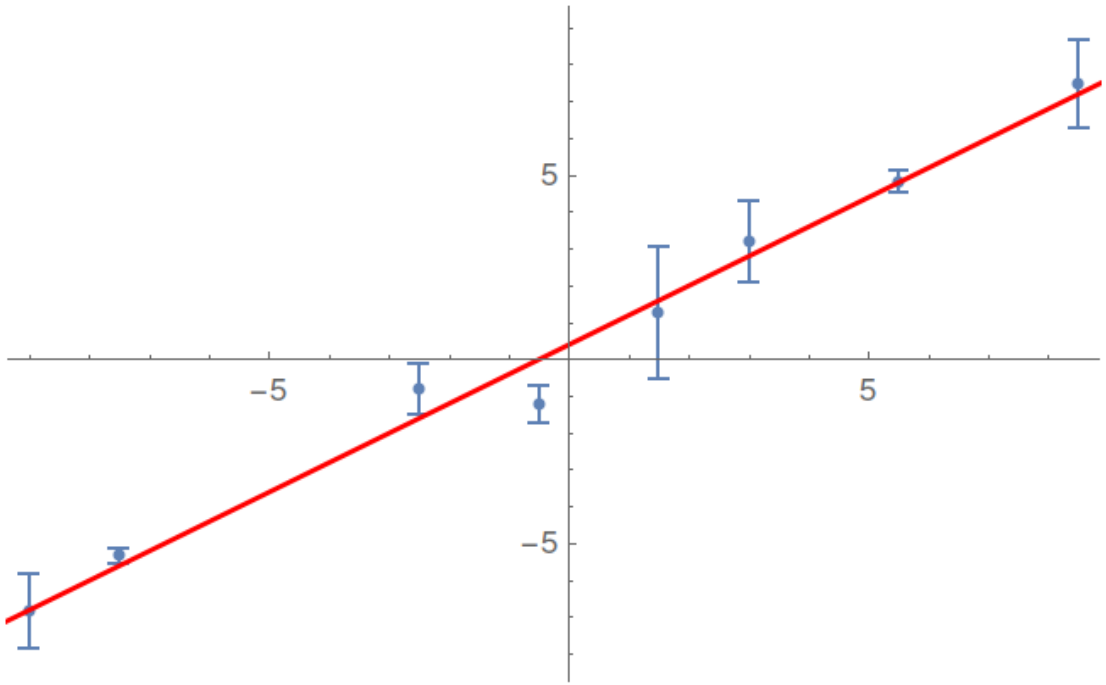


Wizualizacja dopasowanego modelu i danych 1

- Aby umieścić dane i dopasowanie na jednym panelu, korzystamy z funkcji **Show**
  - jako argumenty podajemy narysowane wykresy
  - **Show** wyświetla formatowanie (opis osi, ...) pierwszego z wykresów

```
In[25]:= Show[Wykres3, lmFitPlot]
```

Out[25]=



Zapisywanie wyników dopasowania 1

- Do zapisywania wyników dopasowania korzystamy z funkcji **Export**
  - możemy zapisać zarówno grafikę, jak i parametry dopasowania w pliku tekstowym
  - pierwszym argumentem **Export** jest nazwa pliku wynikowego; drugim - obiekt, który chcemy zapisać

```
In[26]:= Export["Wykres_z_dopasowaniem.png", Show[Wykres3, lmFitPlot]]
Out[26]=
         Wykres_z_dopasowaniem.png

In[27]:= Export["lmFitPlotParameters.dat",
          {lm["BestFitParameters"], lm["ParameterErrors"]}, "Table"]
Out[27]=
         lmFitPlotParameters.dat
```

Dopasowanie modelu nieliniowego do danych

- Model nieliniowy dopasowujemy za pomocą funkcji **NonlinearModelFit**
  - jako argumenty podajemy kolejno: dane wejściowe, model oraz zmienną-argument modelu (x)
  - opcje **NonlinearModelFit** są zbliżone do tych z **LinearModelFit**

In[30]:=
```
nlm = NonlinearModelFit[Data[[All, {1, 2}]],
  a*x^4 + b*x^3 + c*x^2 + d*x + e, {a, b, c, d, e}, x]
```

Out[30]=
```
FittedModel[ 0.0869205 + <<5>> ]
```

In[31]:=
```
Normal[nlm]
```

Out[31]=

$$0.0869205 + 0.739938\,x + 0.0362334\,x^2 + 0.000918213\,x^3 - 0.00040512\,x^4$$

In[32]:=
```
nlm["ParameterTable"]
```

Out[32]=

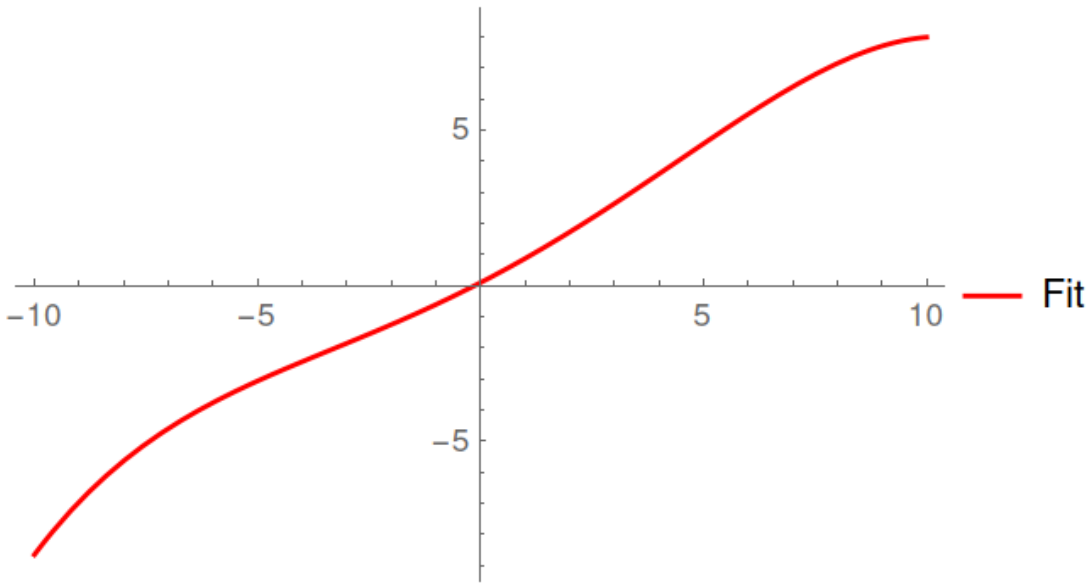
	Estimate	Standard Error	t-Statistic	P-Value
a	−0.00040512	0.000543711	−0.745101	0.510273
b	0.000918213	0.00240428	0.381908	0.727976
c	0.0362334	0.0429419	0.843778	0.460759
d	0.739938	0.157934	4.68512	0.0183781
e	0.0869205	0.493851	0.176005	0.8715

Wizualizacja dopasowanego modelu nieliniowego

- Z atrybutów funkcji **NonlinearModelFit** wybieramy "BestFit"
- Rysujemy model, korzystając z funkcji **Plot**
- Warto dodać formatowanie wykresu; w szczególności - legendę

```
In[95]:= nlmFitPlot = Plot[nlm["BestFit"], {x, -10, 10}, PlotStyle → Red,
           PlotLegends → Placed[{"Fit"}, {1, 0.5}]]
```

Out[95]=

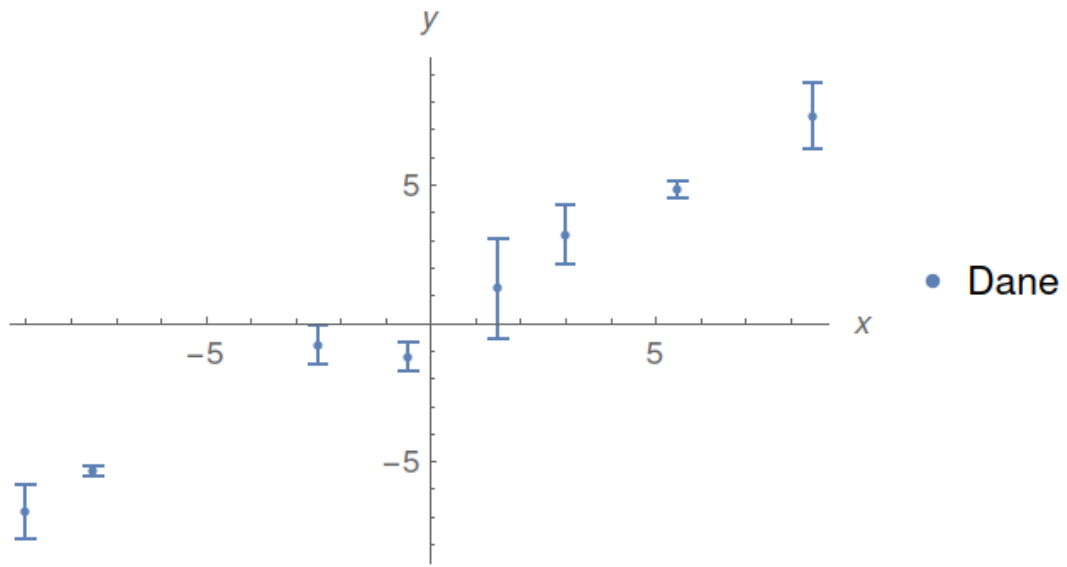


Antrakt na formatowanie wykresu z danymi

- Pamiętamy, że **Show** wyświetla formatowanie pierwszego wykresu
- Należy zadbać, żeby było ono zadowalające
  - w minimalnej wersji opis osi i legenda

In[94]:=
```
Wykres3 =
  ListPlot[Table[{Data[[i, 1]], Around[Data[[i, 2]], Data[[i, 3]]]},
    {i, 1, Length[Data]}], PlotLegends → Placed[{"Dane"}, {1, 0.5}],
   AxesLabel → {x, y}]
```

Out[94]=

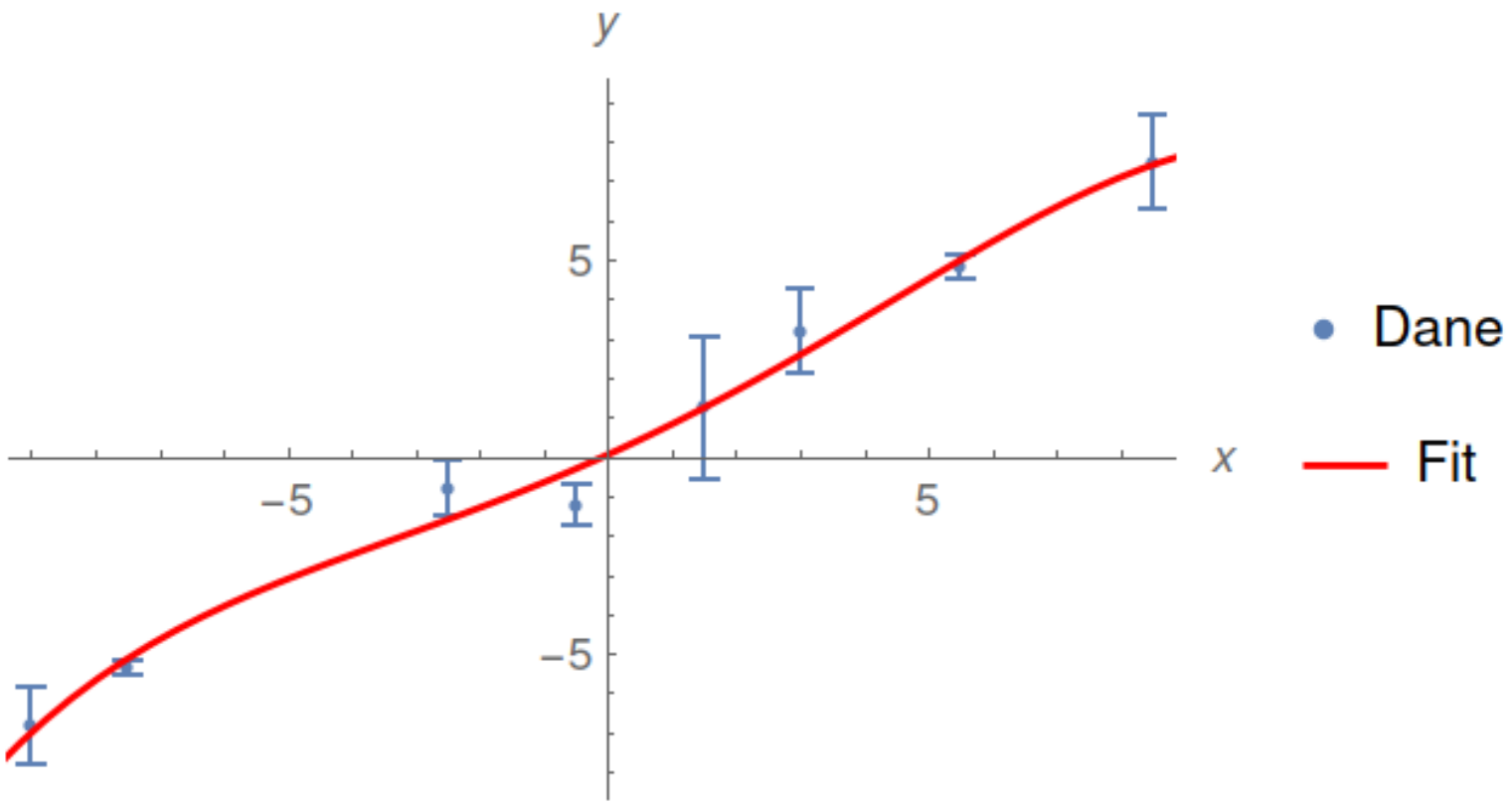


Wizualizacja dopasowanego modelu i danych 2

- Ponownie korzystamy z funkcji **Show**

```
In[93]:= Show[Wykres4, nlmFitPlot2]
```

Out[93]=



Zapisywanie wyników dopasowania 2

- Ponownie korzystamy z funkcji **Export**
  - pierwszym argumentem **Export** jest nazwa pliku wynikowego; drugim - obiekt, który chcemy zapisać

```
In[35]:= Export["Wykres_z_dopasowaniem2.png", Show[Wykres3, nlmFitPlot]]
Out[35]=
         Wykres_z_dopasowaniem2.png

In[36]:= Export["nlmFitPlotParameters.dat",
          {nlm["BestFitParameters"][[All, 2]], nlm["ParameterErrors"]},
          "Table"]
Out[36]=
         nlmFitPlotParameters.dat
```

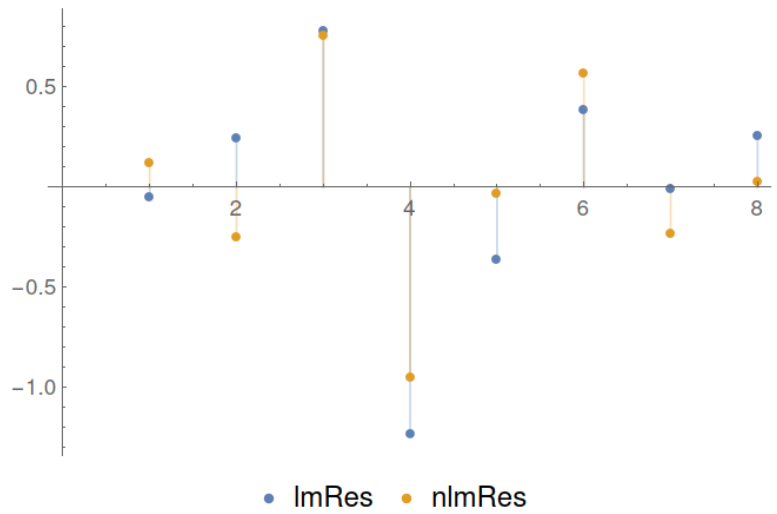
- Warto zwrócić uwagę na inny sposób eksportowania parametrów dopasowania związany z formatem tabeli parametrów
- W przykładzie eksportuję drugą kolumnę tabeli, która zawiera wartości parametrów; pierwsza zawiera nazwy parametrów

Który model wybrać? - Residua

- Do tych samych danych dopasowaliśmy 2 różne modele, warto sprawdzić residua obu modeli

```
In[40]:= lmRes = lm["FitResiduals"]
Out[40]=
{-0.0498165, 0.244563, 0.779064, -1.23477,
 -0.366213, 0.382116, -0.0120519, 0.257106}

In[41]:= nlmRes = nlm["FitResiduals"]
Out[41]=
{0.121913, -0.254399, 0.755096, -0.95043,
 -0.034591, 0.568937, -0.234099, 0.0275731}
```

```
In[99]:= ListPlot[{lmRes, nlmRes}, Filling → Axis,
          PlotLegends → Placed[{"lmRes", "nlmRes"}, Below]]
Out[99]=
```



- Residuum w tym ujęciu jest różnicą między wartością danych, a wartością dopasowanego modelu w danym punkcie

Zadania 8 - podstawowe działanie na danych

25. Wczytaj i przeanalizuj dane z pliku `M3_data.dat`:
    - wczytaj dane z pliku do macierzy o nazwie Dane
    - sprawdź rozmiar macierzy Dane (powinno wyjść 1000 x 2)
    - wyświetl na ekranie (w postaci tekstowej):
      - pierwszą kolumnę macierzy Dane
      - pierwszy wiersz macierzy Dane
      - element z trzeciego wiersza drugiej kolumny macierzy Dane
    - wyświetl na ekranie wykres przedstawiający pobrane dane przyjmując, że zmienna niezależna znajduje się w pierwszej kolumnie, a zmienna zależna w drugiej kolumnie macierzy Dane
    - sformatuj wykres wedle własnego uznania (min. opis osi i legenda)
    - dopasuj do danych funkcję, która najlepiej je opisuje; np.: $ax \sin(bx + c) + d$
    - wyświetl na ekranie tabelę z otrzymanymi parametrami dopasowania i ich błędami
    - wyświetl wykres przedstawiający dopasowaną funkcję
    - skombinuj wykres z danymi oraz wykres z dopasowaniem w jeden
    - wyeksportuj tabelę zawierającą (wyłącznie) otrzymane parametry dopasowania i ich błędy do pliku `M3_fit.dat`.

Zadania 9 - działanie na funkcji Table

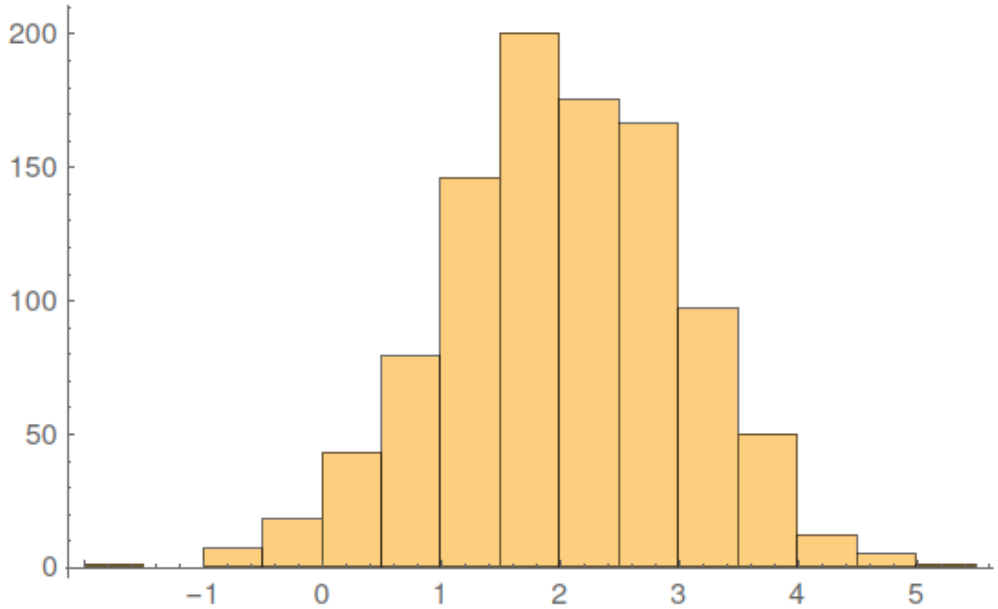
25. Bazując na danych z pliku `M3_data.dat` wykonaj następujące operacje:
    - wczytaj dane z pliku do macierzy o nazwie Dane
    - utwórz nową, jednokolumnową macierz (wektor) o nazwie NoweDaneY, o wartościach $1/(3+y)$ gdzie y to kolumna 2 w macierzy Dane
    - posługując się poleceniem Table wytwórz jednokolumnową macierz (wektor) OX, która dla kolejnych wierszy będzie przyjmowała wartości od 0 do 49.95, co 0.05
    - korzystając z kombinacji poleceń Listplot + Transpose narysuje wykres NoweDaneY w funkcji OX. Skorzystaj z opcji PlotRange, aby wykres pokazywał pełen przedział zmienności OX i NoweDaneY
    - powstały wykres zapisz do pliku `NoweDane.png`.

Histogramy i liczby losowe

- Mathematica ma rozbudowany silnik generowania liczb pseudolosowych (warto poznać **RandomInteger**, **RandomReal**, **RandomVariate**, ...)
- Histogram tworzymy za pomocą funkcji **Histogram**, gdzie jako argument podajemy zbiór danych, i ewentualnie specyfikację binów

```
In[107]:= dane1 = RandomVariate[NormalDistribution[2, 1], 1000];

In[108]:= histogram1 = Histogram[dane1, {0.5}]
```

Out[108]=



Histogramy i ich parametry

- Aby dostać się do parametrów opisujących histogram należy skorzystać z funkcji **HistogramList**, gdzie jako argument ponownie podajemy zbiór danych i specyfikację binów
- **HistogramList** zwraca nam krawędzie binów i liczbę zliczeń w każdym z nich (podobnie, jak funkcja *Histogram* w Pythonie)
- Możemy zatem wyznaczyć środki naszych binów za pomocą funkcji **MovingAverage**

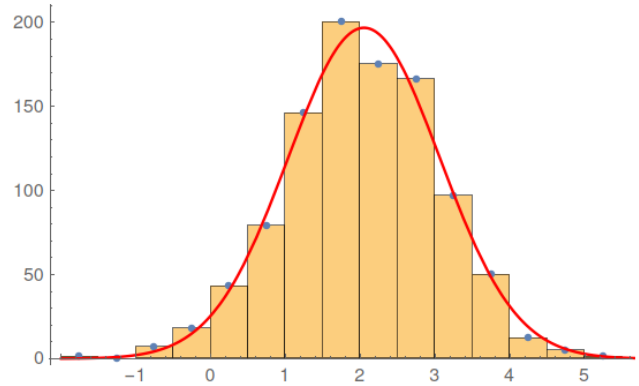
```
In[112]:= {bins, counts} = HistogramList[dane1, {0.5}]

Out[112]=

{{-2., -1.5, -1., -0.5, 0., 0.5, 1., 1.5, 2.,
  2.5, 3., 3.5, 4., 4.5, 5., 5.5}, {1, 0, 7, 18,
  43, 79, 146, 200, 175, 166, 97, 50, 12, 5, 1}}

In[113]:= centers = MovingAverage[bins, 2]

Out[113]=

{-1.75, -1.25, -0.75, -0.25, 0.25, 0.75, 1.25,
 1.75, 2.25, 2.75, 3.25, 3.75, 4.25, 4.75, 5.25}
```

Histogramy i ich analiza

- Po wyznaczeniu środków binów, możemy przeprowadzić dopasowanie modelu teoretycznego

```
In[114]:= dane = Transpose[{centers, counts}]
Out[114]=
{{-1.75, 1}, {-1.25, 0}, {-0.75, 7},
 {-0.25, 18}, {0.25, 43}, {0.75, 79}, {1.25, 146},
 {1.75, 200}, {2.25, 175}, {2.75, 166}, {3.25, 97},
 {3.75, 50}, {4.25, 12}, {4.75, 5}, {5.25, 1}}

In[115]:= model = NonlinearModelFit[dane, (a * Exp[-(x - b)^2 / c]),
            {a, b, c}, x]
Out[115]=
FittedModel[196.809 e^(-<<20>> <<1>>)]
```

```
In[118]:= Show[histogram1, ListPlot[dane],
            Plot[model["BestFit"], {x, -2, 6}, PlotStyle → Red]]
Out[118]=
```



Zadania 10 - histogram

25. Bazując na danych z pliku `wahadlo.txt` wykonaj następujące operacje:
    - wczytaj dane z pliku do macierzy o nazwie wahadlo
    - narysuj histogram przedstawiający rozkład okresu drgań wahadła, dobierz odpowiednie binowanie
    - sformatuj histogram wedle własnego uznania (min. opis osi i legenda)
    - wyznacz krawędzie binów histogramu i liczbę zliczeń w tych minach
    - wyznacz środki binów i przypisz im odpowiadającą liczbę zliczeń
    - dopasuj do danych funkcję, która najlepiej je opisuje; np.: $a \exp(\frac{-(x-b)^2}{c})$
    - wyświetl na ekranie tabelę z otrzymanymi parametrami dopasowania i ich błędami
    - wyświetl wykres przedstawiający dopasowaną funkcję
    - skombinuj histogram z danymi oraz wykres z dopasowaniem w jeden

Zadania 11 - generowanie liczb pseudolosowych

26. Posiłkując się dokumentacją oprogramowania Mathematica, wygeneruj zbiory liczb pseudolosowych podane:
    - 10, 100 oraz 1000 pseudolosowych liczb całkowitych z przedziału od 0 do 10 (**RandomInteger**)
    - 10, 100 oraz 1000 pseudolosowych liczb rzeczywistych z przedziału od 0 do 1 (**RandomReal**)
    - 10, 100 oraz 1000 pseudolosowych liczb z rozkładu Poissona z maksimum w 5 (**RandomVariate**, **PoissonDistribution**)
    - 10, 100 oraz 1000 pseudolosowych liczb z rozkładu opisanego funkcją: $f(x) = \frac{x^2}{4} - 2x + 5$ z przedziału od 0 do 10 (**ProbabilityDistribution**, **RandomVariate**).

    Dla każdego zbioru utwórz histogram (z domyślnym binowaniem) i zaobserwuj wpływ krotności generowanych liczb na kształt rozkładu.
27. Korzystając z funkcji **RandomInteger**, wygeneruj macierz binarną (zero-jedynkową) o wymiarach 40×40 i ją zwizualizuj za pomocą funkcji **ArrayPlot**.

Zadania 12 - badanie przebiegu funkcji

28. Zbadaj przebieg zadanej funkcji:

$$f(x) = \frac{x^3}{1 - x^2}$$

- wyznacz dziedzinę funkcji
- narysuj wykres funkcji
- wyznacz jej miejsca zerowe
- punkt przecięcia z osią Y
- wyznacz granice na krańcach dziedziny (pamiętaj o kierunkach)
- wyznacz asymptotę ukośną funkcji $y = ax + b$, gdzie

$$a = \lim_{x \to \infty} \frac{f(x)}{x}$$

$$b = \lim_{x \to \infty} f(x) - ax$$

- wyznacz przedziały monotoniczności funkcji ($f'(x) > 0$ oraz $f'(x) < 0$)
- wyznacz punkty przegięcia funkcji ($f''(x) = 0$) i ich wartość w tych punktach