Matematyka dla informatyki stosowanej i systemów pomiarowych

2024/2024

dr Remigiusz Durka

Instytut Fizyki Teoretycznej
Zakład Teorii Grawitacji i Oddziaływań Fundamentalnych

Uniwersytet Wrocławski

Wrocław

200 lat pod niemieckim,
400 lat pod polskim,
500 lat pod czeskim panowaniem

- W czasach antycznych w miejscu lub okolicach istniała miejscowość o nazwie **Budorgis**.
- Została ona odwzorowana na antycznej **mapie Ptolemeusza** z lat 142–147 n.e.
- W roku 985 na Ostrowie Tumskim powstał pierwszy gród wybudowany przez **Mieszka I**.
- W średniowiecznej Kronice Polskiej Galla Anonima spisanej w latach 1112–1116 Wrocław obok Krakowa oraz Sandomierza zaliczony został do **jednej z trzech głównych stolic Królestwa Polskiego.**
- **Mikołaj Kopernik** od 1503 r. do 1538 r. był kanonikiem kapituły wrocławskiej.

Wrocławscy nobliści

- Stąd pochodzi (czyli urodziło się, pobierało nauki albo pracowało) aż **jedenastu** laureatów Nagrody Nobla!
- Dwie nagrody literackie, cztery z fizyki, 3 z chemii oraz po jednej z medycyny i ekonomii

FIZYKA:

- **Phillip Lénárd** (1862-1947)
- **Otto Stern** (1888-1969)
- **Erwin Schrödinger** (1887-1961)
- **Max Born** (1882-1970)

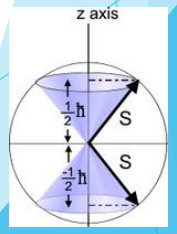
https://visitwroclaw.eu/wroclawscy-noblisci

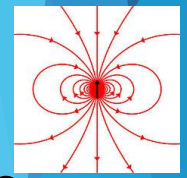
Wrocławscy nobliści

- **Erwin Schrödinger** (tak, ten od jednocześnie żywego i martwego kota i jeszcze bardziej sławnego równania będącego fundamentem fizyki kwantowej) był wykładowcą w 1921 roku.

$$[\hat{x}, \hat{p}_x] = i\hbar$$

- Tutaj studentem fizyki a później asystentem był **Maks Born**, któremu zawdzięczamy interpretację *modułu funkcji falowej* jako gęstości prawdopodobieństwa znalezienia cząstki (z czym wiążą się te wszystkie orbitale, o których uczyliście się na chemii w szkole średniej), a *także komutator operatorów położenia i pędu*
  oraz stworzenie nazwy "mechanika kwantowa“!





- Tu też wychowywał się, studiował i zrobił doktorat **Otto Stern**, który później wykazał w eksperymencie skwantowanie wewnętrznego momentu pędu, czyli spinu.

Wrocław

- **Swoje centra mają tu** Google, Nokia, IBM, Siemens, Opera, Atos, Dolby,
- Działają też światowi liderzy z branż takich jak **motoryzacja, elektronika i AGD, inżynieria mechaniczna, chemia i farmaceutyka**. To firmy, które mają mocno rozbudowane działy IT. Mocny jest też sektor finansowy.
- Ale w mieście intensywnie rozwijają się też rodzime firmy IT, podbijające europejskie i światowe rynki i wiele start-upów.
- Wiele spółek realizuje model „born global", co oznacza, że od pierwszego dnia istnienia są w stanie działać globalnie i pracować dla międzynarodowych klientów.

Globalne korporacje z centrami w Wrocławiu:

1. **Google** – posiada centrum badawczo-rozwojowe.
2. **Nokia** – specjalizuje się w telekomunikacji i technologiach sieciowych.
3. **IBM** – dostarcza rozwiązania IT oraz usługi technologiczne.
4. **Siemens** – działa w obszarze automatyzacji przemysłu i energii.
5. **Opera Software** – rozwija przeglądarkę internetową Opera.
6. **Atos** – globalna firma IT i doradcza.
7. **Dolby** – rozwija technologie dźwięku i obrazu.
8. **3M** – amerykański konglomerat, który produkuje szeroką gamę produktów.
9. **Capgemini** – międzynarodowa firma świadcząca usługi IT i konsultingowe.
10. **Credit Suisse** – duża szwajcarska firma świadcząca usługi finansowe.

Znane polskie firmy i start-upy z Wrocławia:

1. **Techland** – jeden z największych polskich producentów gier komputerowych, znany z takich tytułów jak *Dying Light* i *Call of Juarez*.
2. **LiveChat Software** – firma dostarczająca oprogramowanie do obsługi klienta i komunikacji online.
3. **PWR Racing Team** – zespół inżynierski z Politechniki Wrocławskiej, który projektuje i buduje bolidy wyścigowe, uczestniczący w międzynarodowych zawodach *Formula Student*.
4. **Objectivity** – firma IT zajmująca się tworzeniem oprogramowania na zamówienie.
5. **Synerise** – start-up specjalizujący się w sztucznej inteligencji i analizie danych.
6. **Ten Square Games** – producent gier mobilnych, specjalizujący się w grach typu free-to-play.
7. **Tooploox** – firma technologiczna, zajmująca się m.in. sztuczną inteligencją, uczeniem maszynowym oraz produktami cyfrowymi.
8. **Unit4 Polska** – firma specjalizująca się w oprogramowaniu ERP dla sektora publicznego i edukacji.

http://wroclaw.wyborcza.pl/wroclaw/7,35771,23432142,it-w-kodzie-wroclawia.html

Zakres zajęć

- Realizacja w ciągu zaledwie dwóch semestrów w wymiarze 75 godzin na semestr, z czego
- 45 godzin (3 tygodniowo) zarezerwowano na zajęcia w pracowni komputerowej
- 30 godzin (2 tygodniowo) przeznaczone są na wykłady

Zaliczenie

- Ćwiczenia/Laboratorium: 2 kolokwia
- Wykład: egzamin pisemny

Kurs dostępny online

http://users.ift.uni.wroc.pl/~zkoza/matematyka/

- autor: prof. Zbigniew Koza

http://ift.uni.wroc.pl/~rdurka/matissp/

- autor: dr Remigiusz Durka

Zagadnienia

- Narzędzia informatyczne do obliczeń inżynierskich (numerycznych i symbolicznych)
- Granice funkcji, ciągi, szeregi
- Funkcje i ich wykresy
- Narzędzia do wizualizacji funkcji/danych
- Pochodne i całki funkcji jednej i dwóch zmiennych
- Równania różniczkowe zwyczajne
- Równania różniczkowe cząstkowe

- Liczby zespolone
- Wektory i macierze
- Układy równań liniowych
- Zagadnienie na wartości i wektory własne
- Równania nieliniowe i algebraiczne
- Algebra

Semestr 1

Semestr 2

Idea stojąca za kursem

- Podstawową ideą leżącą u podstaw jego konstrukcji jest przekonanie, że licencjacki kurs matematyki powinien dać studentom **ogólne rozeznanie** w matematyce wyższej traktowanej jako ***narzędzie*** do rozwiązywania konkretnych problemów a sposób nauczania powinien nadążać za najnowszymi osiągnięciami techniki.

- Kurs oparty jest na **narzędziach informatycznych:**

  (*Octave*, *Gnuplot*, *Wolfram Alpha*)

Idea stojąca za kursem

- Kurs **ten nie ma na celu** kształcenia zawodowych matematyków i w bardzo niewielkim stopniu porusza zagadnienie dla matematyków kluczowe: dowodzenie twierdzeń w oparciu o rygorystyczny formalizm.
- Nauka matematyki **jako narzędzia** pozwala szybko pokazać jej użyteczność bez wrzucania studentów na zbyt dla nich głębokie wody matematycznej abstrakcji…
- Wadą takiego podejścia jest oczywiście **groźba powierzchownego potraktowania** matematyki, bo siłą matematyki jest jej zdolność do uogólnień, które z kolei wymagają pewnego poziomu abstrakcji.

Jednoosobowa armia do rozwiązywania problemów

Analiza matematyczna w zadaniach

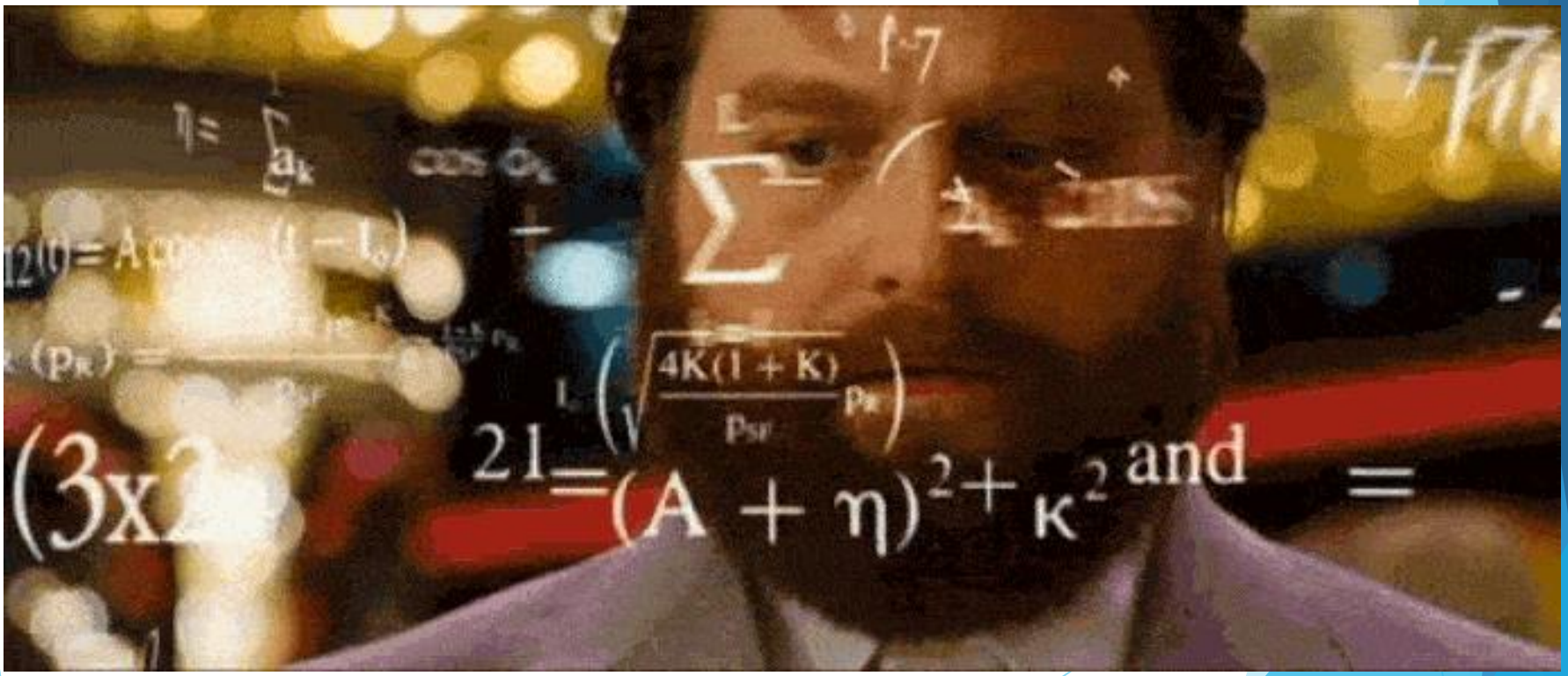
Krysicki, Włodarski

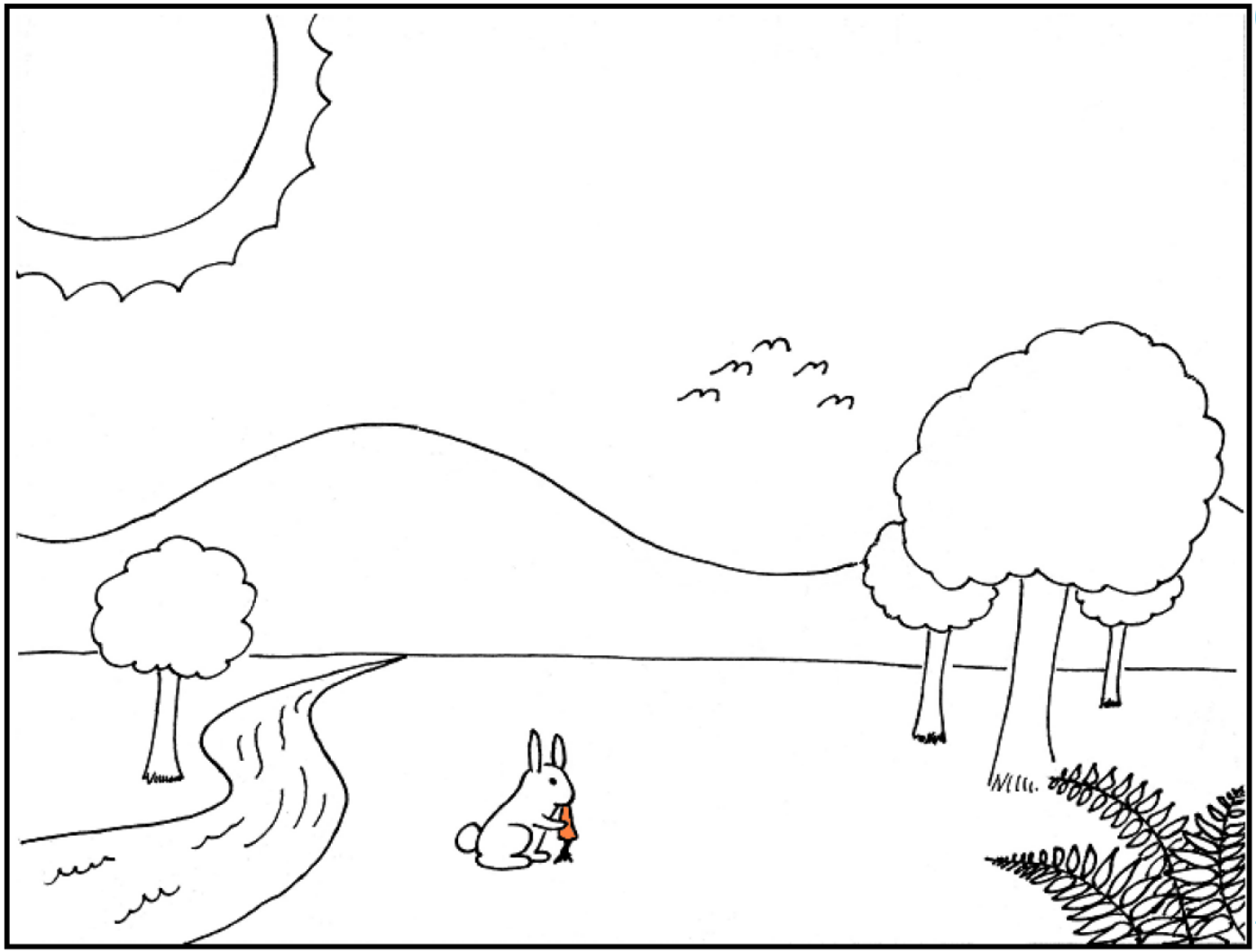
tom 1
tom 2


EVERYTHING
I SAY
WILL BE ON
THE EXAM

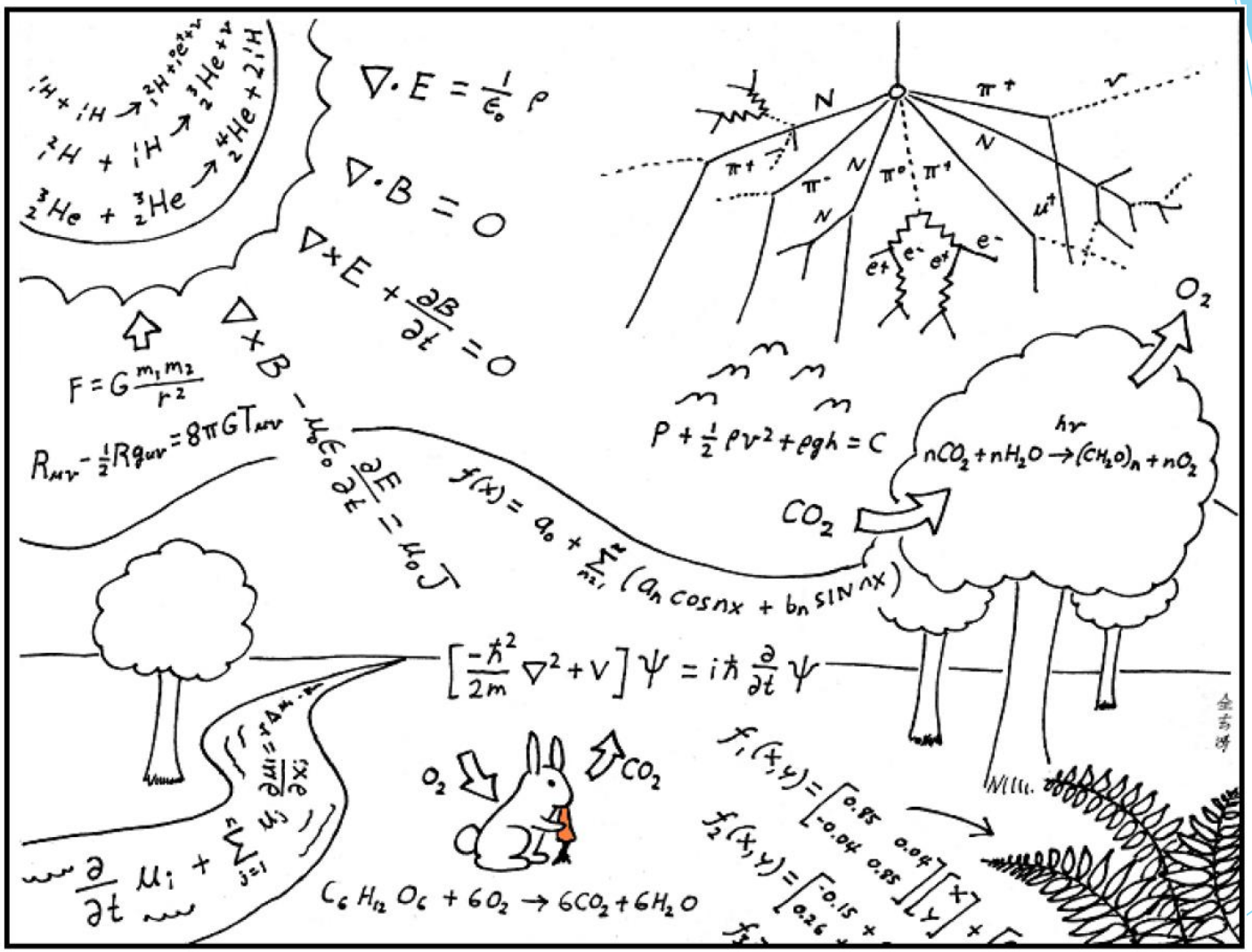

Everything
I SAY
will be on the
EXAM

- Matematyka nie sprowadza się do "*dowodzenia twierdzeń*" lub "*realizacji obliczeń na kalkulatorze*"



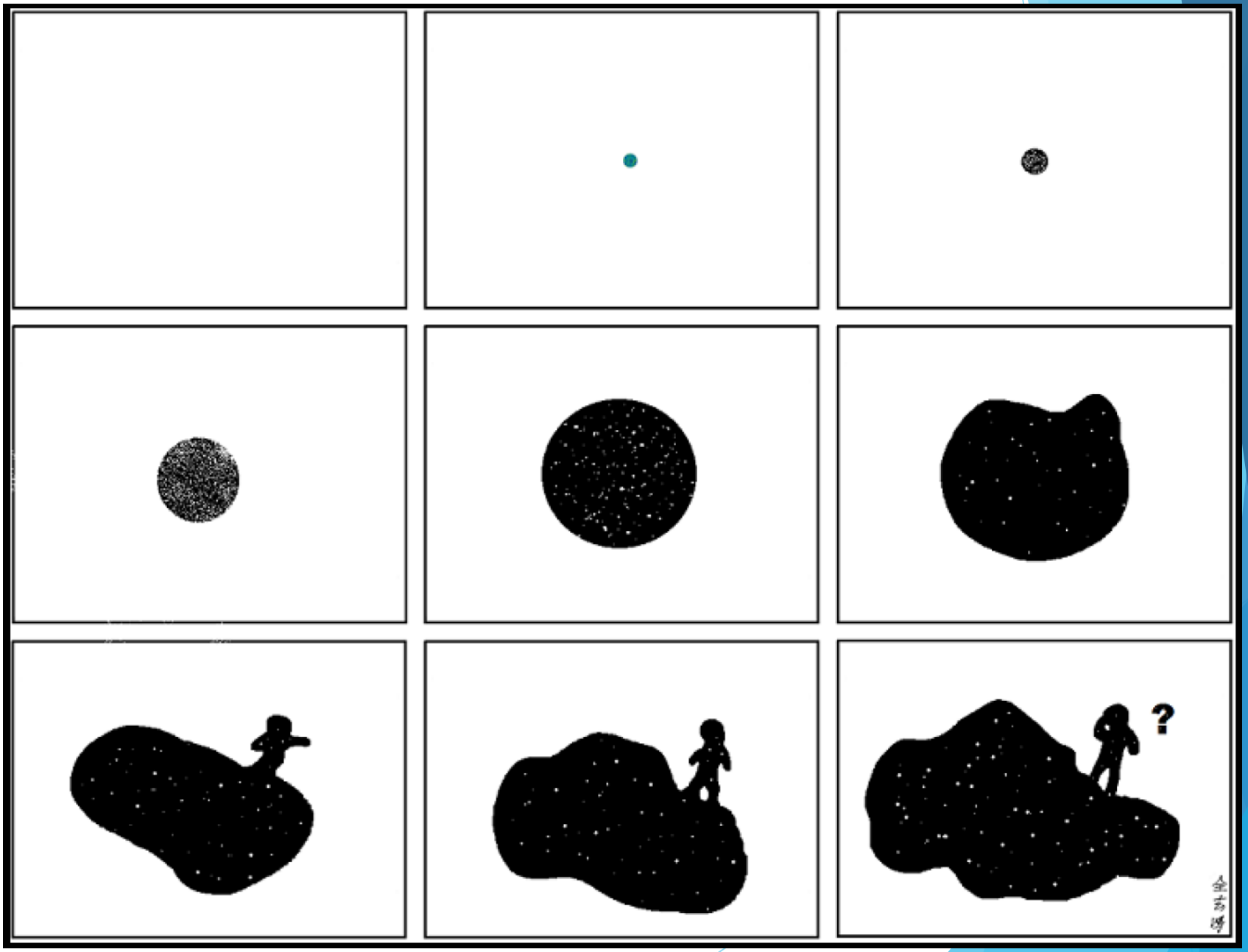


Abstroose Goose

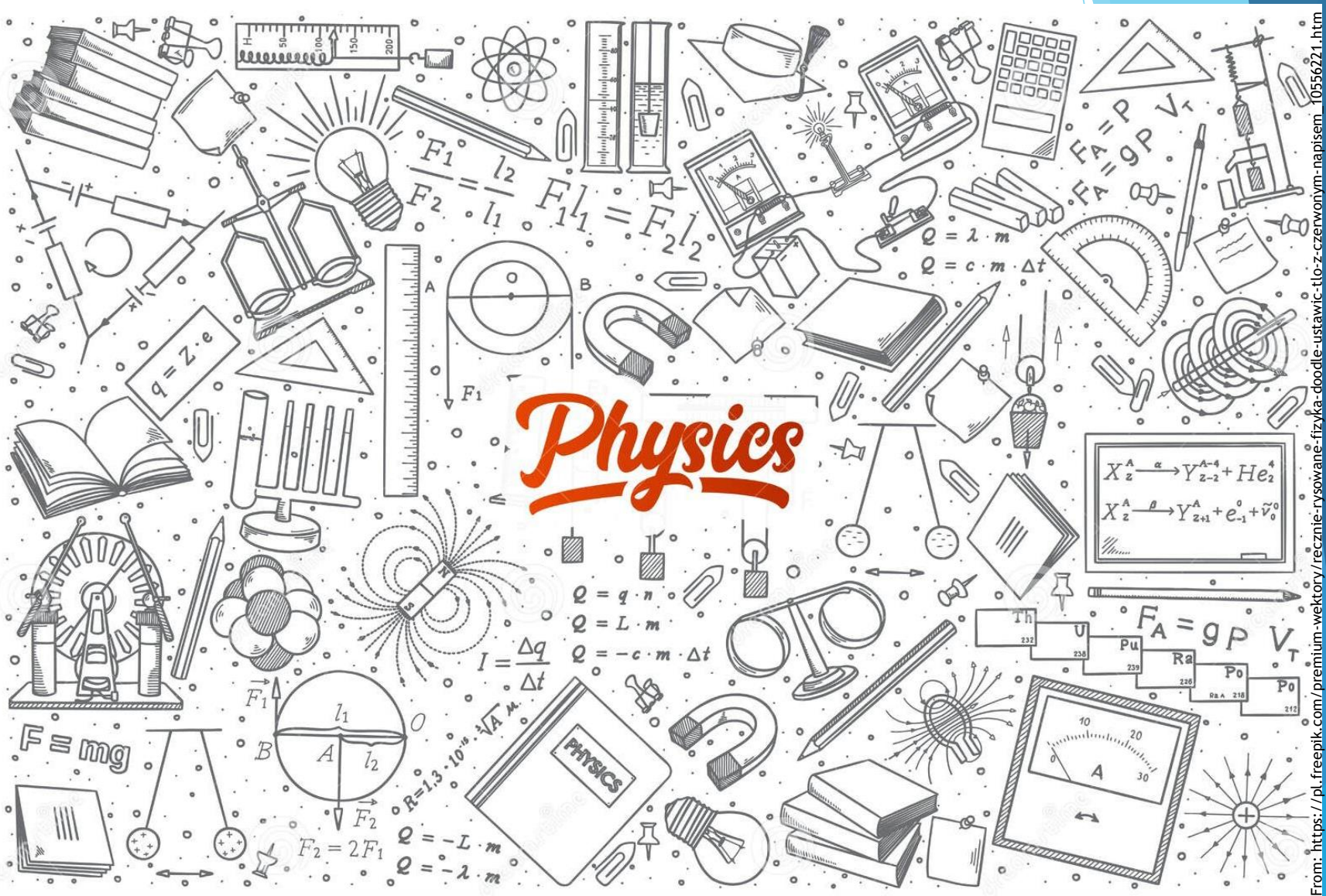


This is how scientists see the world.

Wszechświat starający sam siebie zrozumieć



Eksploracja **jak rzeczy działają** i **dlaczego**!

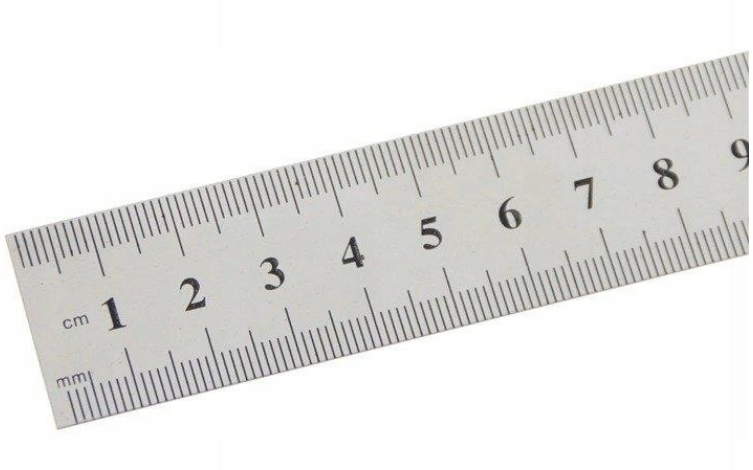


From: https://pl.freepik.com/premium-wektory/recznie-rysowane-fizyka-doodle-ustawic-tlo-z-czerwonym-napisem_10556221.htm

Od skalarów do tensorów

- **Skalar** - wielkość, do której określenia wystarczy jedna liczba rzeczywista (wraz z wymiarem wielkości fizycznej lub bezwymiarowa)

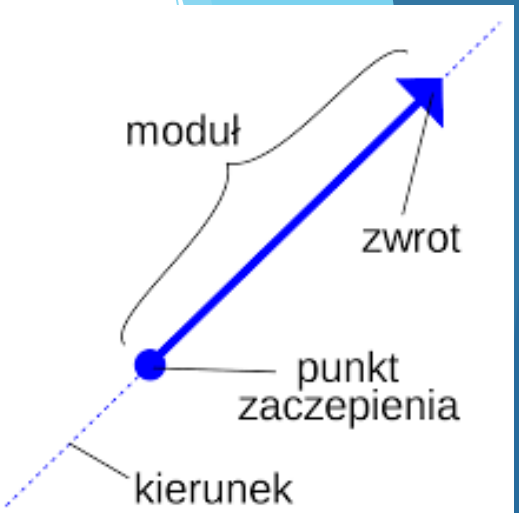
- masa M,
- długość L,
- pole powierzchni A,
- objętość V,
- temperatura T,
- gęstość,
- praca W,
- potencjał pola elektrostatycznego lub grawitacyjnego.

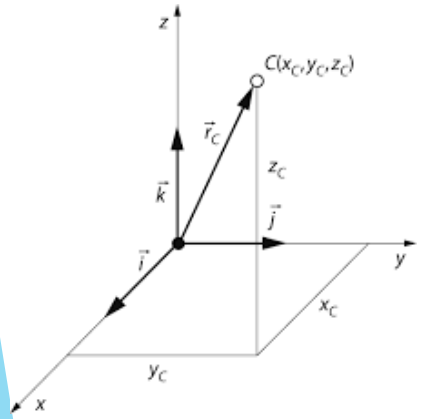


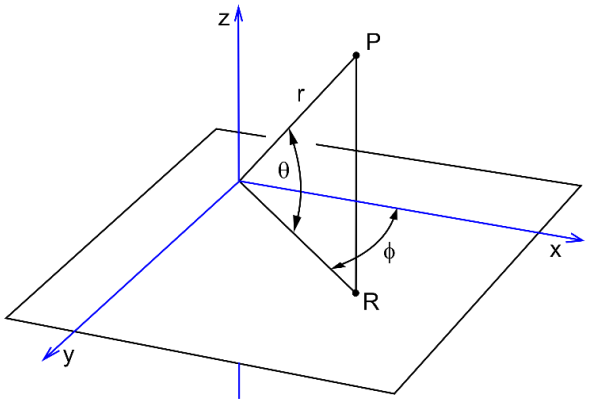
$$\phi = -G\frac{M}{r}.$$

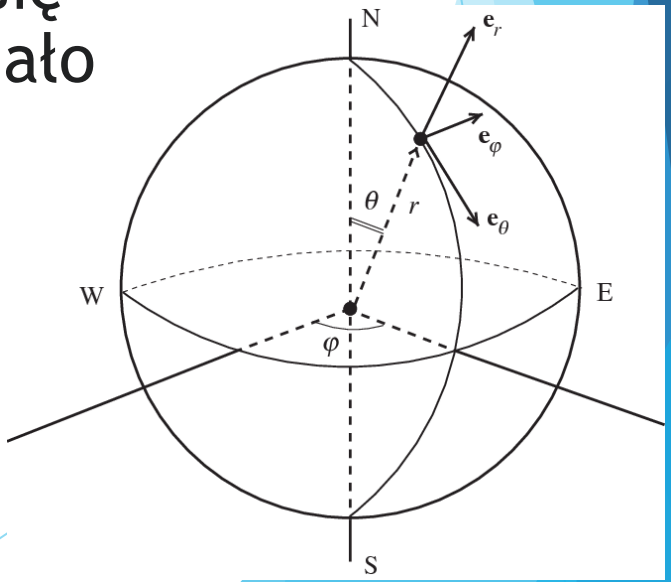
Od skalarów do tensorów

- **Wektor** jest to obiekt matematyczny opisywany za pomocą modułu (długością) oraz kierunku wraz ze zwrotem.
- Wektory odgrywają ważną rolę w fizyce opisując m.in. położenie, prędkość i przyspieszenie poruszającego się obiektu oraz siły działająca na ciało

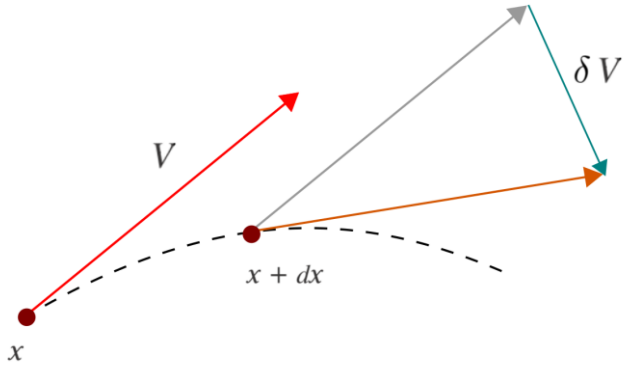


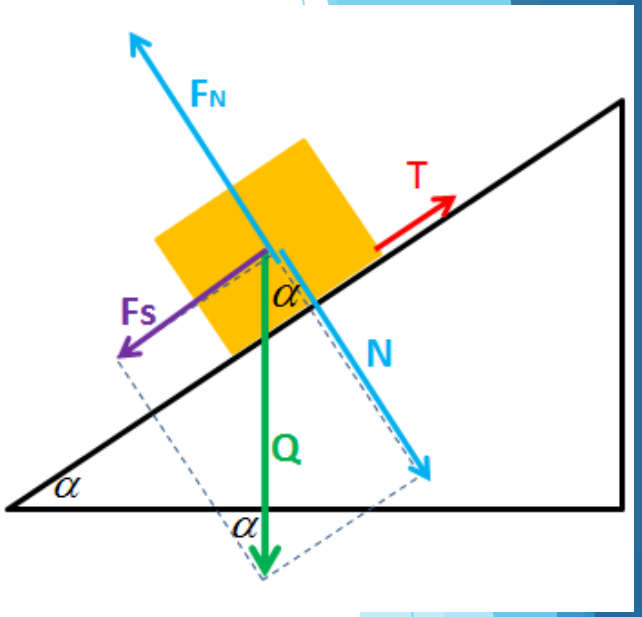




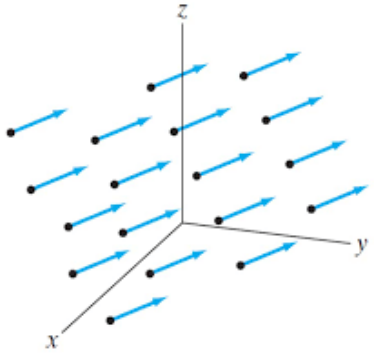


Od skalarów do tensorów





Pole wektorowe



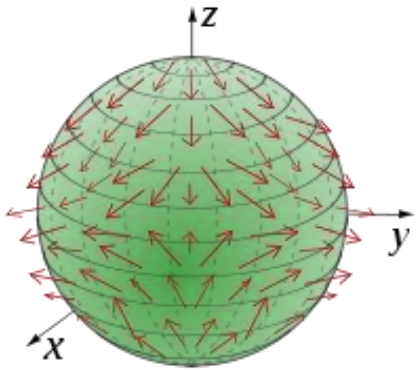
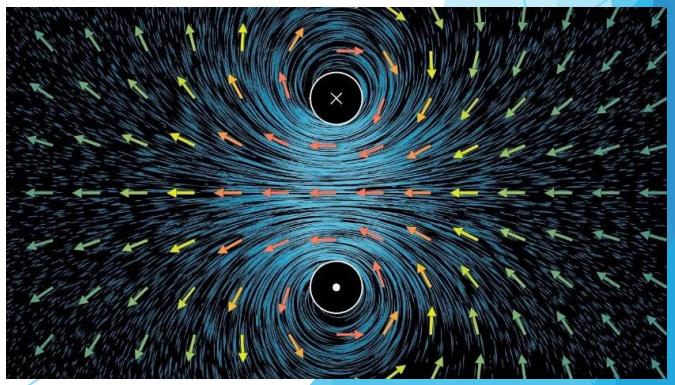


Figure 3.40 The constant vector field $\mathbf{F} = \mathbf{a}$.



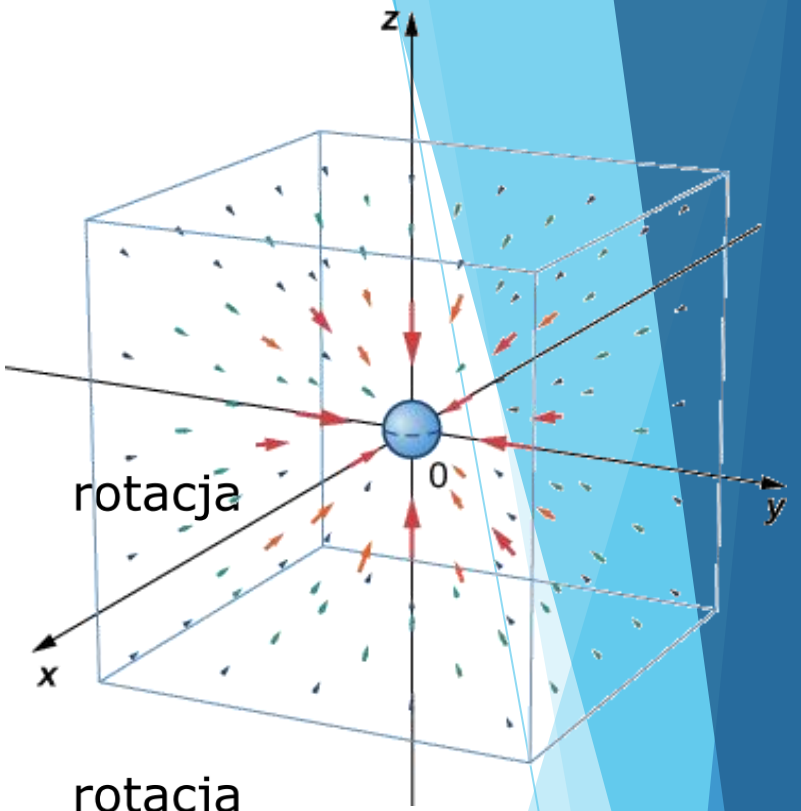
Równania Maxwella

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}$$

$$\vec{\nabla} \times \vec{B} = \mu\varepsilon\vec{j} + \frac{\partial \vec{E}}{\partial t}$$

$$\varepsilon\vec{\nabla} \cdot \vec{E} = \rho$$

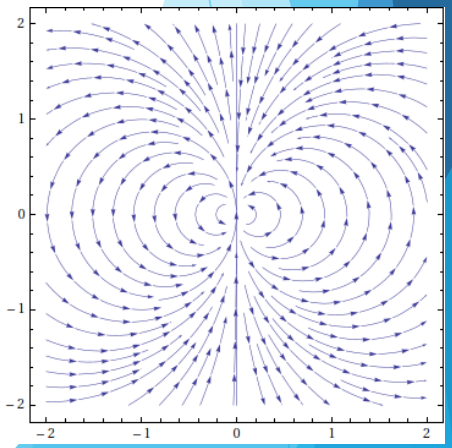
$$\vec{\nabla} \cdot \vec{B} = 0$$



rotacja

rotacja

dywergencja



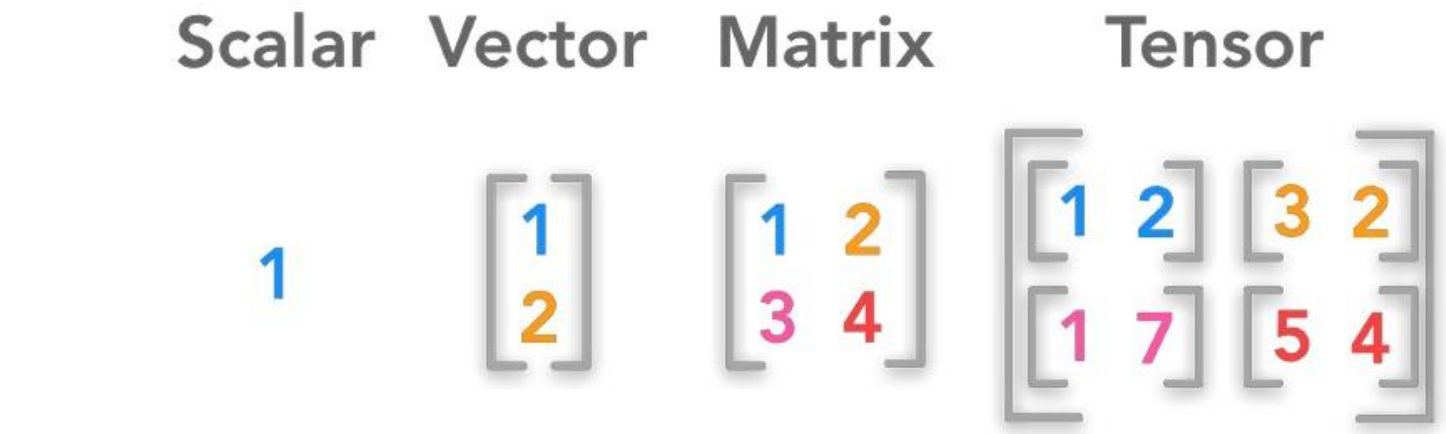
dywergencja

Od skalarów do tensorów

Macierz/Tablica

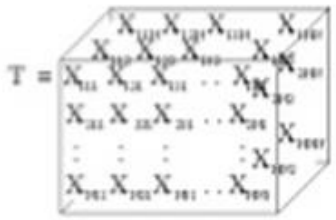
Kolumna/Wiersz

Bloki i hiperbloki



- Tensors are n dimensional arrays
  - Scalar is 0 D tensor
  - Vector is 1 D tensor
  - Matrix is 2 D tensor

rank	object
0	scalar
1	vector
2	$N \times N$ matrix
≥ 3	tensor



Od równania na jedną zmienną do równań różniczkowych

- x-1=2 => x = 3 (szukamy LICZBY)

$$\begin{cases} 2x_1 + 3x_2 - x_3 = 1 \\ x_1 - x_2 + x_3 = 2 \\ 3x_1 + x_2 - 2x_3 = 3 \end{cases}$$

(szukamy LICZB)

Jest to układ równań postaci $AX = B$, gdzie

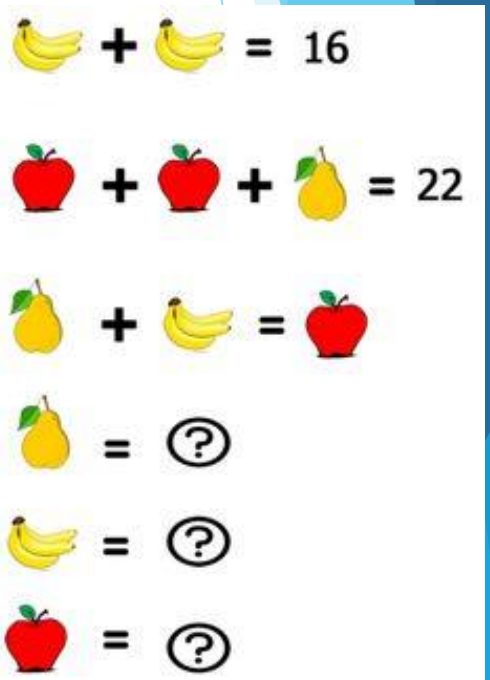
$$A = \begin{bmatrix} 2 & 3 & -1 \\ 1 & -1 & 1 \\ 3 & 1 & -2 \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Do rozwiązania układu zastosujemy wzory Cramera:

$$x_1 = \frac{detA_{x_1}}{detA}, x_2 = \frac{detA_{x_2}}{detA}, x_3 = \frac{detA_{x_3}}{detA}$$

gdzie $A_{x_1}, A_{x_2}, A_{x_3}$ to macierze powstałe przez zastąpienie odpowiednich kolumn macierzy A kolumną wyrazów wolnych B:

$$A_{x_1} = \begin{bmatrix} 1 & 3 & -1 \\ 2 & -1 & 1 \\ 3 & 1 & -2 \end{bmatrix}, A_{x_2} = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 2 & 1 \\ 3 & 3 & -2 \end{bmatrix}, A_{x_3} = \begin{bmatrix} 2 & 3 & 1 \\ 1 & -1 & 2 \\ 3 & 1 & 3 \end{bmatrix}$$



Równania różniczkowe

- Izaak Newton – “dajcie mi konfiguracje sił a powiem Wam co się stanie dalej”

$$ma = F$$

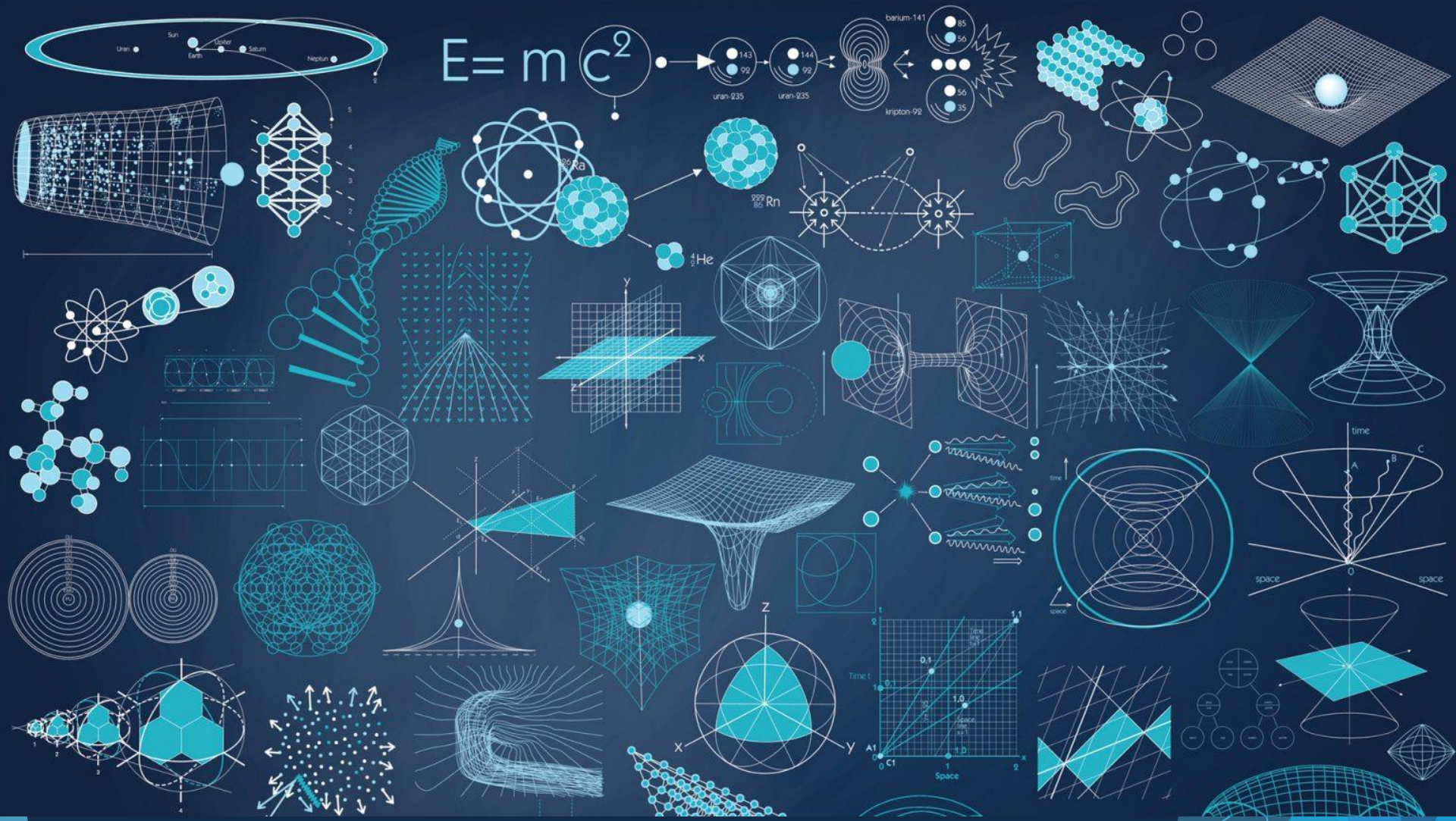
$$m\frac{d^2}{dt^2}x(t) = F$$

szukamy funkcji x(t)

Np. nie ma sił F=0 => $x(t) = C_1 \cdot t + C_2$

$x(t) = V_0 \cdot t + x_0$

Siła constans F=mg => $x(t) = \frac{g \cdot t^2}{2} + C_1 \cdot t + C_2$



Wszędzie obliczenia

ChatGPT

ChatGPT

A mathematical pendulum, also known as a simple pendulum, is an idealized model used in physics to describe the motion of a pendulum under the influence of gravity. It consists of a weightless, inextensible cord of fixed length l, with a mass m (referred to as a bob) attached to one end, while the other end is fixed at a pivot point. When displaced from its equilibrium position and released, the mass swings back and forth due to the gravitational force, undergoing periodic motion.

The motion of a mathematical pendulum is described by the differential equation:

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0$$

where:

- θ is the angular displacement from the vertical,
- g is the acceleration due to gravity,
- l is the length of the pendulum.

For small angles (θ), the $\sin(\theta)$ term can be approximated by θ (in radians), leading to a simpler form of the equation that describes simple harmonic motion:

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\theta = 0$$

The period T of the pendulum, which is the time it takes for the pendulum to complete one full swing back and forth, is given by:

$$T = 2\pi\sqrt{\frac{l}{g}}$$

This formula shows that the period T is independent of the mass m and depends only on the length l of the pendulum and the gravitational acceleration g.

Gemini

A mathematical pendulum is an idealized model of a real pendulum. It aims to simplify the real-world behavior by making certain assumptions, making it easier to analyze and understand its motions. Here are the key characteristics of a mathematical pendulum:

Idealizations:

- **Point mass:** The bob (the swinging object) is treated as a point with all its mass concentrated at a single location.
- **Massless string/rod:** The string or rod suspending the bob has no mass, so it doesn't contribute to the movement.
- **Frictionless:** There is no friction or air resistance acting on the bob, so its swings wouldn't slow down over time.
- **Small angles:** The pendulum swings back and forth only through small angles compared to its length. This allows us to use sine and cosine functions for calculations, which simplifies the equations.
- **Planar motion:** The bob moves only in a vertical plane determined by its initial swing.

Motion:

- **Oscillation:** Due to gravity, the bob oscillates back and forth about its equilibrium position (hanging straight down).
- **Period:** The time it takes to complete one full swing (back and forth) is called the period and is independent of the bob's mass, depending only on the length of the pendulum and the acceleration due to gravity (g).
- **Frequency:** This is the number of swings per second and is inversely proportional to the period.

ChatGPT

A mathematical pendulum, or simple pendulum, is a model consisting of a mass (the "bob") attached to a string or rod of fixed length, which swings freely under the influence of gravity. The motion is described by a second-order differential equation derived from Newton's second law and assuming small angles (linear approximation):

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin(\theta) = 0$$

where:

- θ is the angular displacement from the vertical,
- g is the acceleration due to gravity ($9.81\ \mathrm{m/s^2}$ on the surface of the Earth),
- L is the length of the pendulum.

For small angles ($\theta \ll 1$ radian), $\sin(\theta) \approx \theta$, simplifying the equation to:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0$$

This simplification leads to simple harmonic motion, which can be solved analytically. However, to solve the non-linear equation for any angle, we typically use numerical methods. Below is a Python code snippet using `scipy.integrate.solve_ivp` to solve the differential equation for a mathematical pendulum with an initial angle:

python — Copy code

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

# Constants
g = 9.81  # acceleration due to gravity, in m/s^2
L = 1.0   # length of pendulum, in meters

# Differential equation for the pendulum (theta'' + (g/L)*sin(theta) = 0)
def pendulum(t, y):
    theta, theta_dot = y
    dtheta_dt = theta_dot
    dtheta_dot_dt = -(g/L) * np.sin(theta)
    return [dtheta_dt, dtheta_dot_dt]

# Initial conditions: theta = 0.2 radians, theta_dot = 0.0 radians/s
y0 = [0.2, 0.0]

# Time span for the solution
t_span = (0, 10)  # 10 seconds
t_eval = np.linspace(t_span[0], t_span[1], 1000)

# Solve the differential equation
sol = solve_ivp(pendulum, t_span, y0, t_eval=t_eval, method='RK45')

# Plot the solution
plt.plot(sol.t, sol.y[0])
plt.xlabel('Time (s)')
plt.ylabel('Angular displacement (radians)')
plt.title('Pendulum Motion')
plt.grid(True)
plt.show()
```

This code defines the differential equation for the pendulum's motion, solves it over a time span of 10 seconds for a pendulum of length 1 meter with an initial displacement of 0.2 radians, and plots the angular displacement over time.

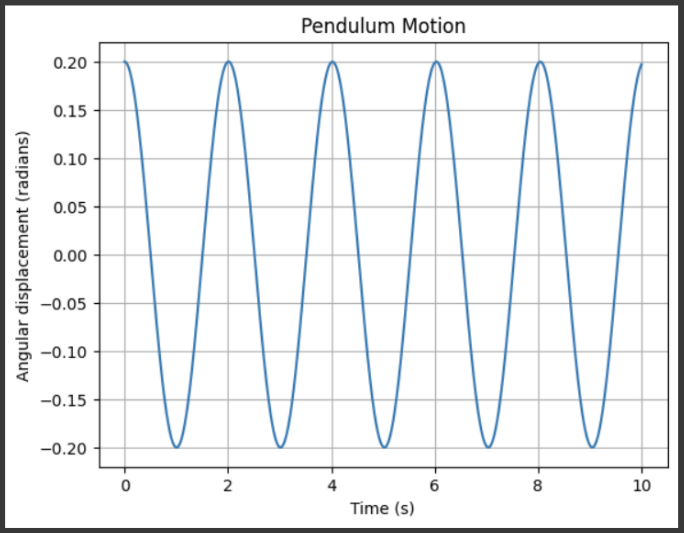
GPT + Colab

MP.ipynb

File Edit View Insert Runtime Tools Help

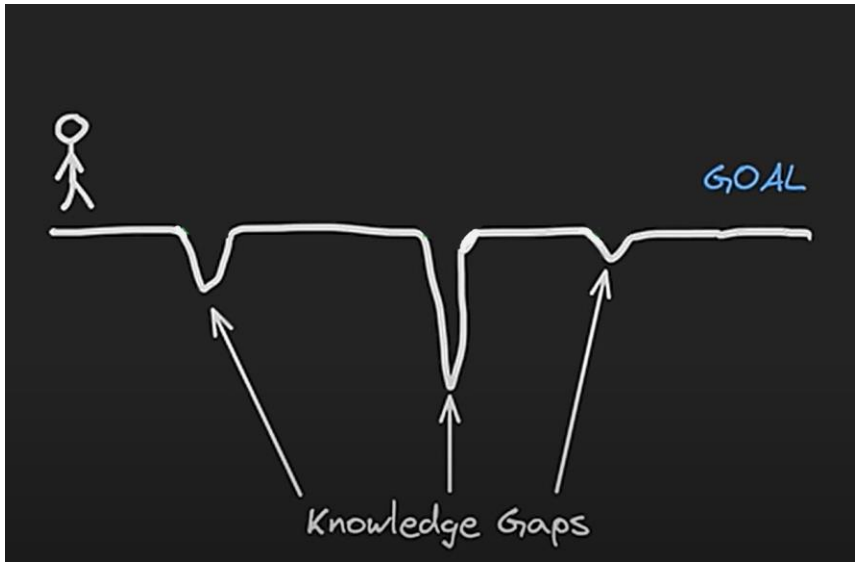
+ Code + Text

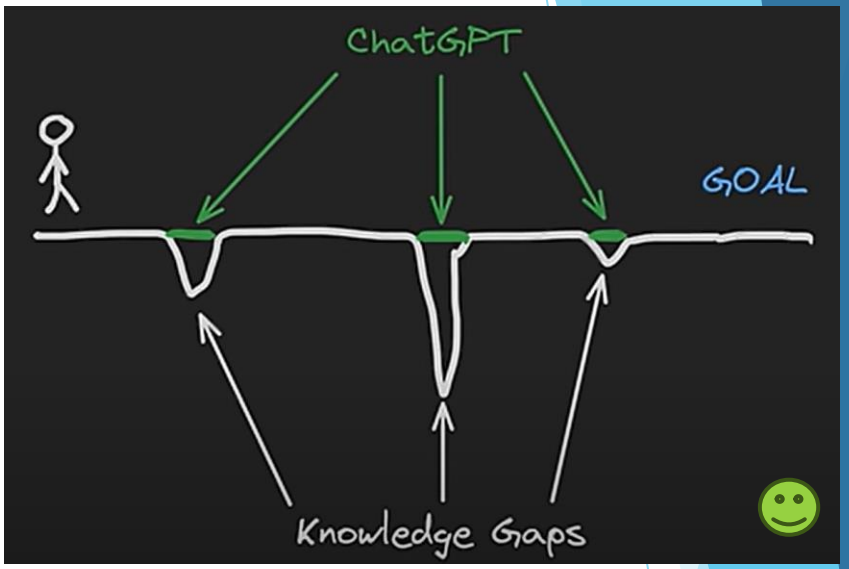
```
[1]  1 import numpy as np
     2 from scipy.integrate import solve_ivp
     3 import matplotlib.pyplot as plt
     4
     5 # Constants
     6 g = 9.81  # acceleration due to gravity, in m/s^2
     7 L = 1.0   # length of pendulum, in meters
     8
     9 # Differential equation for the pendulum (theta'' + (g/L)*sin(theta) = 0)
    10 def pendulum(t, y):
    11     theta, theta_dot = y
    12     dtheta_dt = theta_dot
    13     dtheta_dot_dt = -(g/L) * np.sin(theta)
    14     return [dtheta_dt, dtheta_dot_dt]
    15
    16 # Initial conditions: theta = 0.2 radians, theta_dot = 0.0 radians/s
    17 y0 = [0.2, 0.0]
    18
    19 # Time span for the solution
    20 t_span = (0, 10)  # 10 seconds
    21 t_eval = np.linspace(t_span[0], t_span[1], 1000)
    22
    23 # Solve the differential equation
    24 sol = solve_ivp(pendulum, t_span, y0, t_eval=t_eval, method='RK45')
    25
    26 # Plot the solution
    27 plt.plot(sol.t, sol.y[0])
    28 plt.xlabel('Time (s)')
    29 plt.ylabel('Angular displacement (radians)')
    30 plt.title('Pendulum Motion')
    31 plt.grid(True)
    32 plt.show()
    33
```

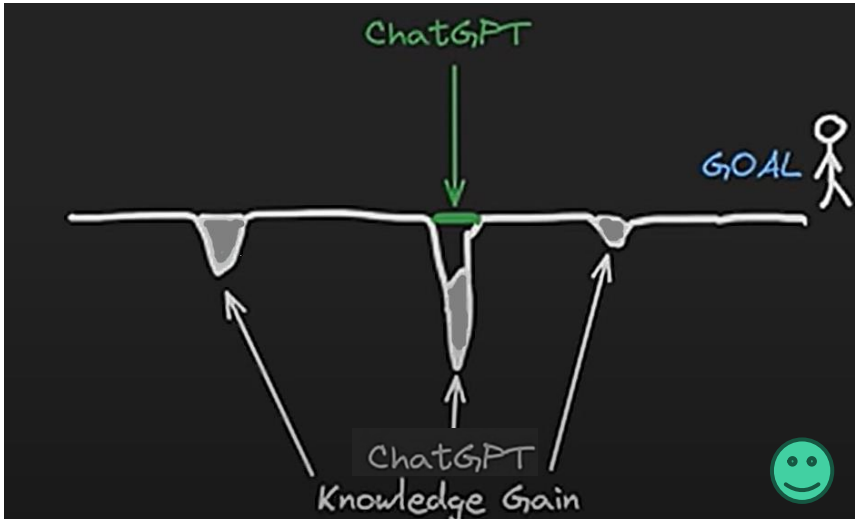


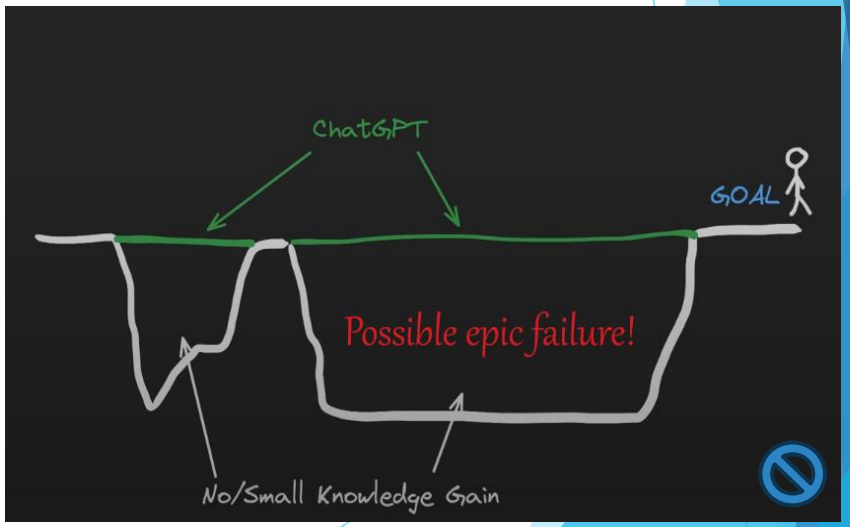
Dosłownie rzucamy zaklęcia!

Jak powinno się używać i nieużywać ChataGPT









From https://www.youtube.com/watch?v=6CGtwF_5kzY

ChatGPT jako rower dla umysłu!

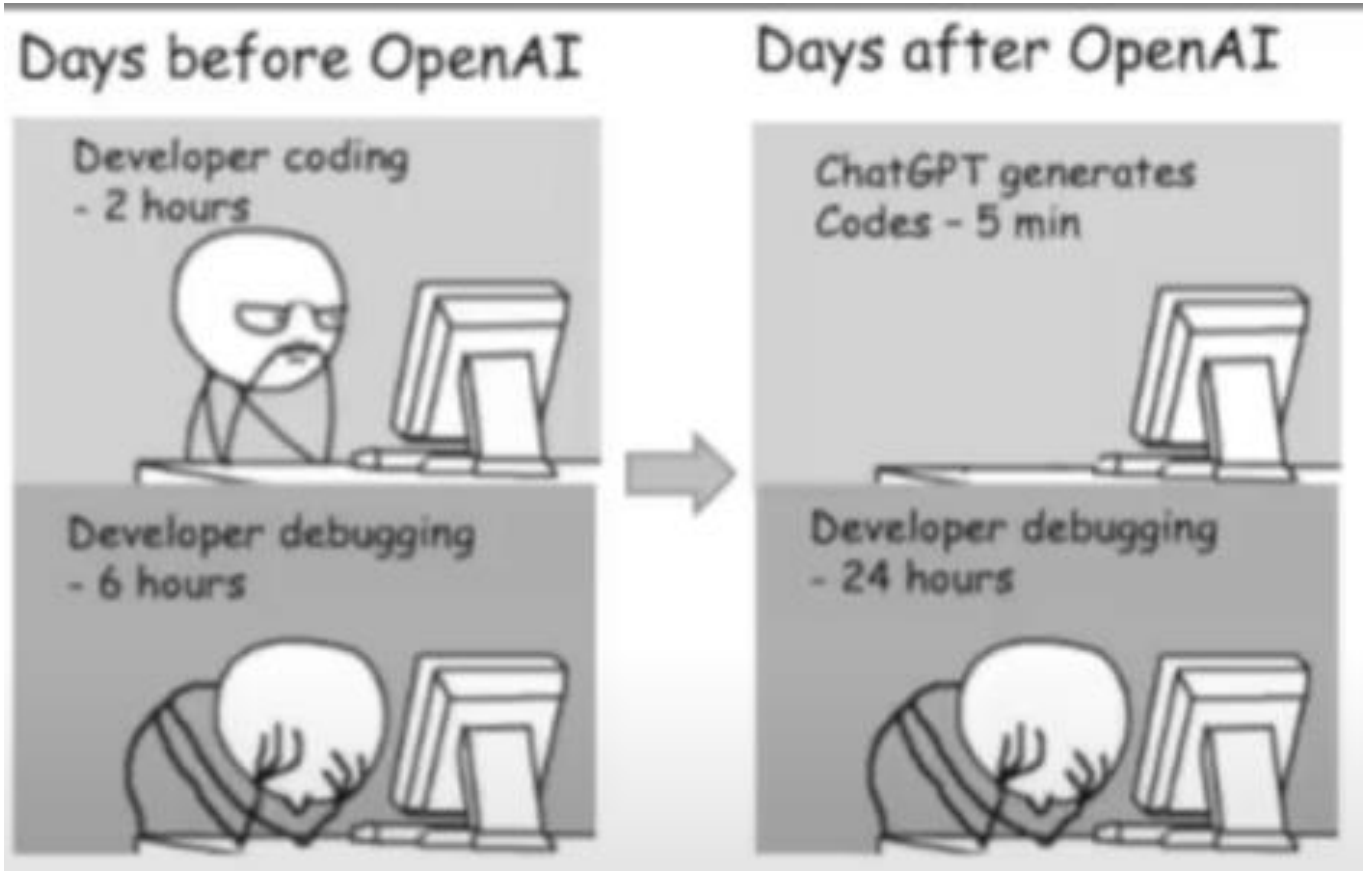
Inne narzędzia

- Geogebra - https://www.geogebra.org/
- Colab - https://colab.research.google.com/

- ChatGPT - https://chat.openai.com/
- Gemini Google - https://gemini.google.com/app
- Claude - https://claude.ai/

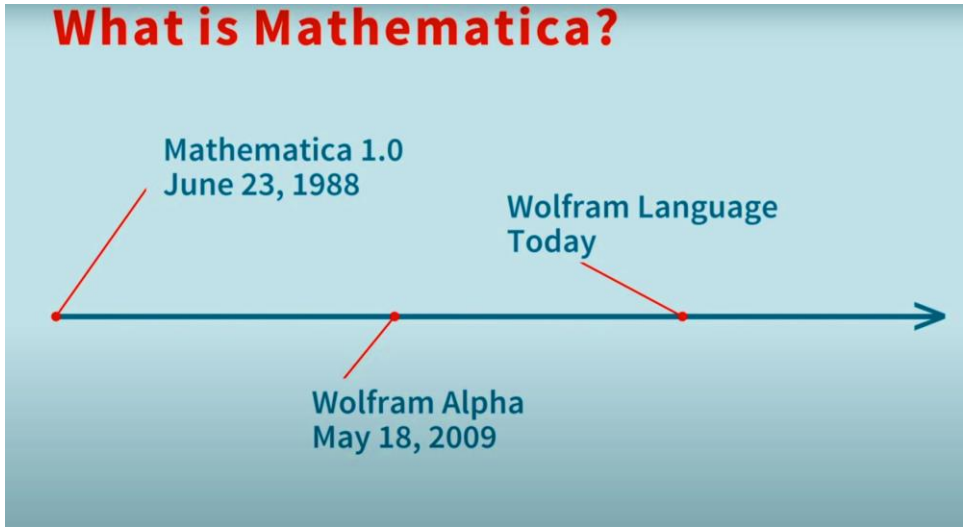
Ty musisz być
za kierownicą
tych narzędzi!

Workflow


Days before OpenAI
Developer coding
- 2 hours
Developer debugging
- 6 hours
Days after OpenAI
ChatGPT generates
Codes - 5 min
Developer debugging
- 24 hours

Mathematica

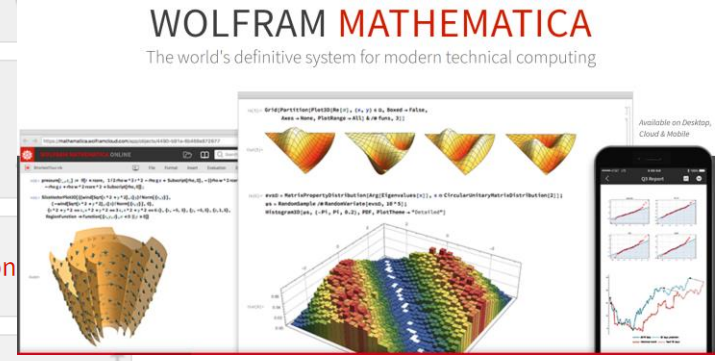
[Stephen Wolfram]



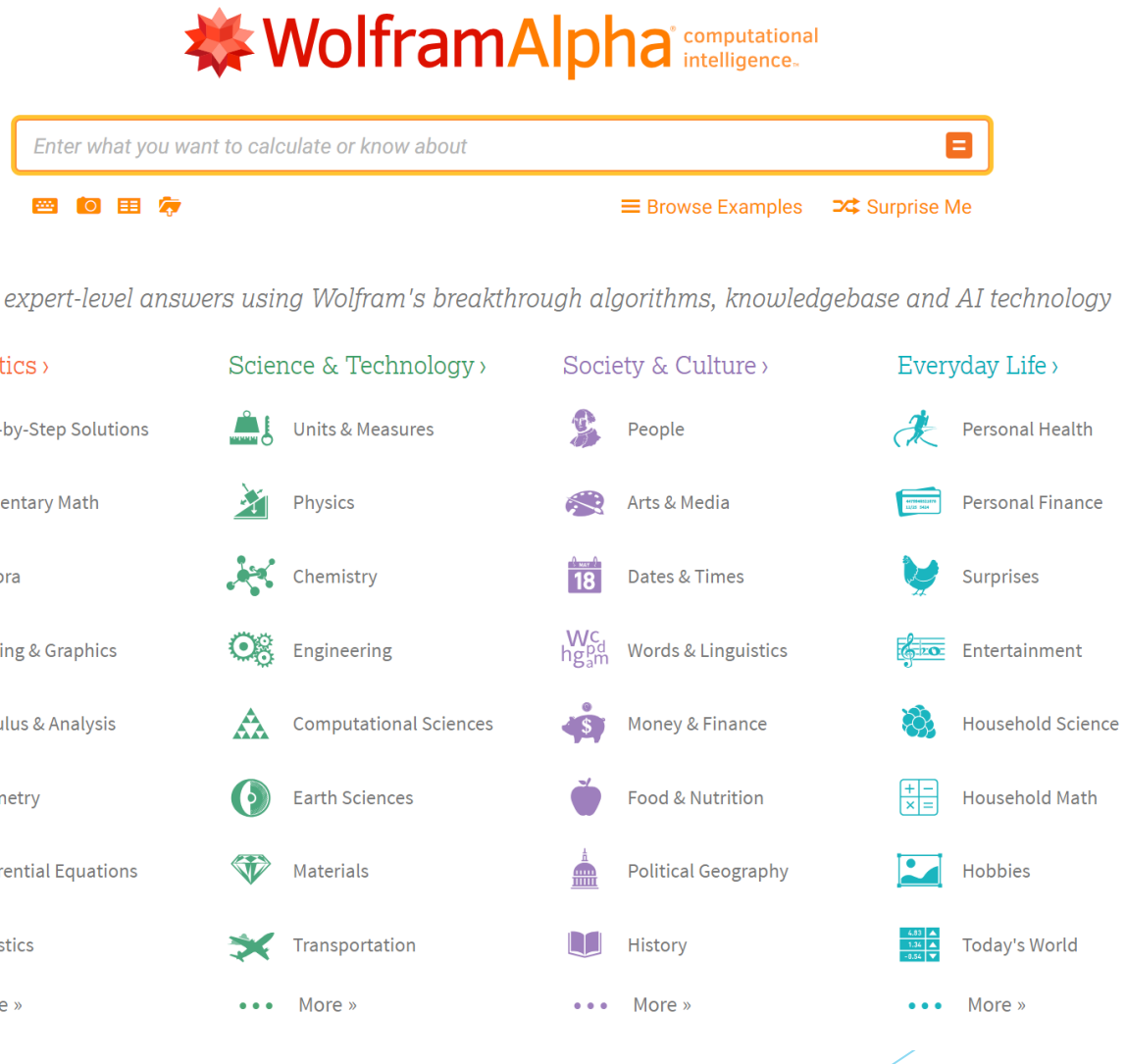
- https://www.youtube.com/watch?v=QMAjAQg2Grc
- https://www.youtube.com/watch?v=ysVdWiBVKHc

Symbolic Language f[x]	Mathematical Computation $\sum_{k=0}^{\infty} \frac{(a_1)_k}{(b_1)_k}$
Numerics	Visualization
Algebraic Manipulation A=B	Number Theory

Data Analysis	Graph Computation	Interactive Computation
Image Computation	Geometric Computation	Importing & Exporting

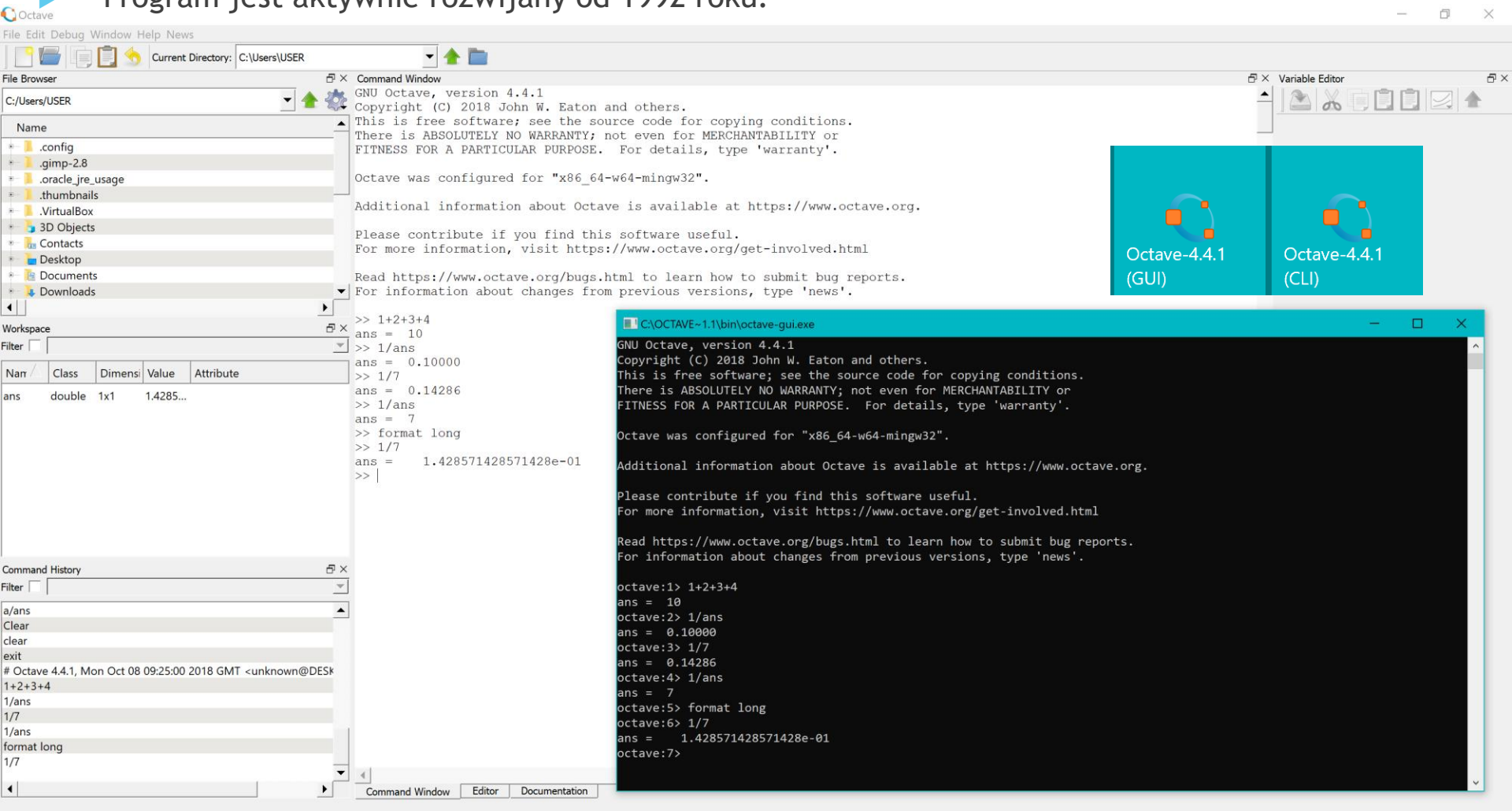


http://www.wolframalpha.com/



Octave Command Line and GUI

- Octave jest to program komputerowy oraz skryptowy język programowania przeznaczony do wykonywania obliczeń numerycznych, wolny odpowiednik komercyjnego programu Matlab.
- Program jest aktywnie rozwijany od 1992 roku.



Octave służy do rozwiązywania problemów numerycznych, m.in.:

- obliczanie wartości wyrażeń (w tym wyrażeń zawierających zaawansowane funkcje matematyczne, np. funkcje zespolone)
- znajdowania wartości sum i iloczynów ciągów liczb o bardzo dużej liczbie elementów
- znajdowania rozwiązań układów równań liniowych (mogących mieć nawet tysiące niewiadomych)
- rozwiązywania równań i układów równań nieliniowych
- znajdowania wartości całek (oznaczonych)
- rozwiązywania układów równań różniczkowych zwyczajnych
- rozwiązywania układów równań różniczkowych cząstkowych
- rozwiązywanie standardowych problemów algebry liniowej, m.in. wyznaczania wartości i wektorów własnych
- prezentacji rozwiązań w postaci wykresów
- Octave to także skryptowy język programowania posiadający mechanizmy włączania do obliczeń wysokowydajnych funkcji napisanych w kompilowanych językach programowania, np. C/C++.

Obliczenia numeryczne i symboliczne

- Są dwa różne podejścia do rozwiązywania problemów matematycznych i fizycznych, często wykorzystywane w naukach ścisłych i inżynierii.

1. **Obliczenia symboliczne**:
    1. Polegają na manipulacji wyrażeń matematycznych w formie symbolicznej, np. znajdowaniu dokładnych rozwiązań równań.
    2. Wykorzystują reguły algebry, rachunku różniczkowego i całkowego do przekształcania wyrażeń w formie dokładnej.
    3. Typowe narzędzia: Mathematica, Maple, SymPy (Python).
    4. Przykłady zastosowań: dowody matematyczne, analiza funkcji, symboliczne rozwiązywanie równań.
2. **Obliczenia numeryczne**:
    1. Polegają na przybliżonym rozwiązywaniu równań matematycznych za pomocą algorytmów numerycznych.
    2. Używane, gdy analityczne (symboliczne) rozwiązania są trudne lub niemożliwe do uzyskania.
    3. Wymagają zastosowania metod takich jak różniczkowanie numeryczne, całkowanie, metody iteracyjne (np. metoda Newtona).
    4. Typowe narzędzia: MATLAB, NumPy (Python), Fortran, C++.
    5. Przykłady zastosowań: symulacje fizyczne, analiza danych, rozwiązywanie równań różniczkowych.

Obliczenia symboliczne

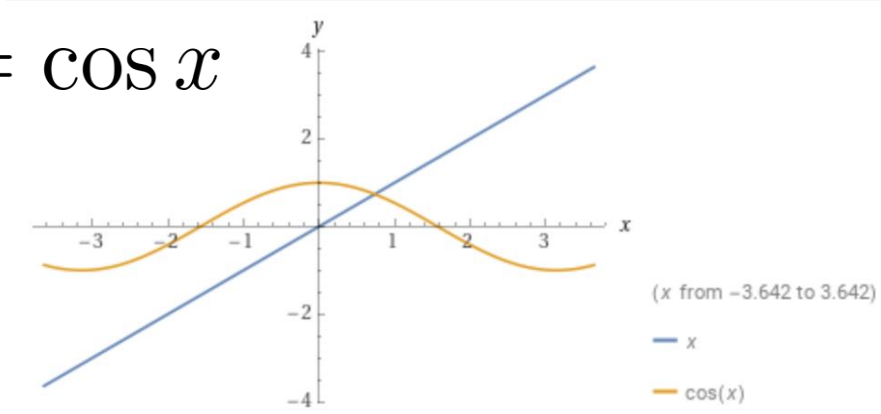
$$2^x = 8 \qquad 2^x = 2^3 \qquad x = 3$$

$$e^x = 10 \qquad x = \ln 10 \qquad x = 2,3025\ldots$$

$$\sin x = \cos x \qquad \tan x = 1 \qquad x = \arctan(1) = \frac{\pi}{4}$$

Obliczenia numeryczne

$$x = \cos x$$



$$x \approx 0.739085\ldots$$

Liczenie

Operatory arytmetyczne

Octave umożliwia wykonywanie wszystkich podstawowych operacji matematycznych. Wykaz najważniejszych operatorów rozpoznawanych przez Octave znajduje się w Tabeli 1.

Tabela 1. Podstawowe operatory arytmetyczne Octave

Operacja	Operator	Przykłady
potęgowanie	** lub ^	`2**3, 2^3`
mnożenie	*	`2 * 3`
dzielenie	/ lub \	`1/7 lub 7\1`
dodawanie	+	`2 + 2`
odejmowanie	–	`3 – 2`

Tabela 2. Najważniejsze stałe w Octave

Stała	Zapis w Octave	Znaczenie
π	pi	iloraz obwodu koła do jego średnicy
e	e	podstawa logarytmów naturalnych
i	i lub j	jednostka urojona

Dokładne znaczenie dwóch ostatnich stałych zostanie omówione w dalszej części kursu. Teraz wystarczy zapamiętać, że `e` jest pewną liczbą,

```
>> e
ans =  2.7183
```

natomiast `i` jest „magicznym obiektem", który podniesiony do kwadratu daje -1:

```
>> i*i
ans =  -1
```

Liczenie w Octave

```
>> pi^2
ans =  9.8696
>> log10(1000)
ans =  3
>> log10(1e3)
ans =  3
>> sin(cos(sin(cos(1))))
ans =  0.76471
>> factorial(6)
ans =  720
>> factorial(50)
ans =   3.0414e+064
>> log10(factorial(50))
ans =  64.483
```

```
> cos([0, 0.1, 0.2])
ans =
   1.00000   0.99500   0.98007
```

```
>> sin(30 *  pi / 180) # sinus 30 stopni
ans =  0.50000
```

help and doc

```
>> help factorial
'factorial' is a function from the file C:\OCTAVE~1.1\share\octave\4.4.1\m\specfun\factorial.m

 -- factorial (N)
     Return the factorial of N where N is a real non-negative integer.

     If N is a scalar, this is equivalent to 'prod (1:N)'.  For vector
     or matrix arguments, return the factorial of each element in the
     array.

     For non-integers see the generalized factorial function 'gamma'.
     Note that the factorial function grows large quite quickly, and
     even with double precision values overflow will occur if N > 171.
     For such cases consider 'gammaln'.

     See also: prod, gamma, gammaln.

Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.
```

```
>> doc factorial
>>
```

factorial *(n)*

Return the factorial of *n* where *n* is a real non-negative integer.

If *n* is a scalar, this is equivalent to *prod (1:n)*. For vector or matrix arguments, return the factorial of each element in the array.

For non-integers see the generalized factorial function *gamma*. Note that the factorial function grows large quite quickly, and even with double precision values overflow will occur if *n* > 171. For such cases consider *gammaln*.

See also: prod, gamma, gammaln.

Skrypty i funcje

Załóżmy, że skrypt nazywa się `programik.m`, a jego zawartość wygląda następująco:

```
# W pliku programik.m znajduje się trywialny program testowy Octave
1/7
2/7
3/7
```

Jeżeli wydamy polecenie `programik`:

```
> programik
```

Program odpowie następująco:

```
ans =  0.14286
ans =  0.28571
ans =  0.42857
```

co świadczy o wykonaniu wszystkich instrukcji z pliku `programik.m`.

Załóżmy, że naszym celem jest zdefiniowanie funkcji o nazwie `sin5`, która dla argumentu x obliczać będzie wartość wyrażenia $x - x^3/6 + x^5/120$. Aby zdefiniować taką funkcję:

1. Tworzymy (w katalogu bieżącym) plik o nazwie `sin5.m`
2. W pliku tym wpisujemy:

```
function y = sin5(x)
   y = x - x**3/6 + x**5/120;
endfunction
```

Uwaga! Powyższa definicja nie jest optymalna i wkrótce ją poprawimy.

Funkcje tę wywołujemy w Octave w naturalny sposób:

```
>> sin5 (0.1)
ans =  0.099833
```

```
# Funkcja sin5 przybliża wartość funkcji sin(x) za pomocą wielomianu stopnia 5.
#
# Przybliżenie to jest całkiem dobre dla małych wartości x,
# np. dla |x| < 1.
#

function y = sin5(x)
   y = x - x.**3/6 + x.**5/120;
endfunction
```

Wielomiany

Wielomiany [edytuj]

definicja wielomianu

```
octave:3> c=[1 1 1]
c =   1  1  1
```

wyświetlenie wielomianu

```
octave:4> polyout(c, 'z')
1*z^2 + 1*z^1 + 1
```

Miejsca zerowe wielomianu czyli rozwiązania równania $1 * z^2 + 1 * z^1 + 1 = 0$

```
octave:5> roots(c)
ans =
-0.50000 + 0.86603i
-0.50000 - 0.86603i
```

Obliczyć wartość wielomianu $7x^3 - x^2 + 5$ w punkcie 11 .

```
P=[7 -1 0 5];
polyval(P, 11)
```

Wynikiem jest

```
9201
```

Wielomiany

Rozwiązanie układu równań liniowych

Rozwiązać układ równań $\begin{cases} 2x - y = 1 \\ x + 3y = 11 \end{cases}$

```
A = [2 -1; 1  3];
f = [1 11];
u = A\f'
```

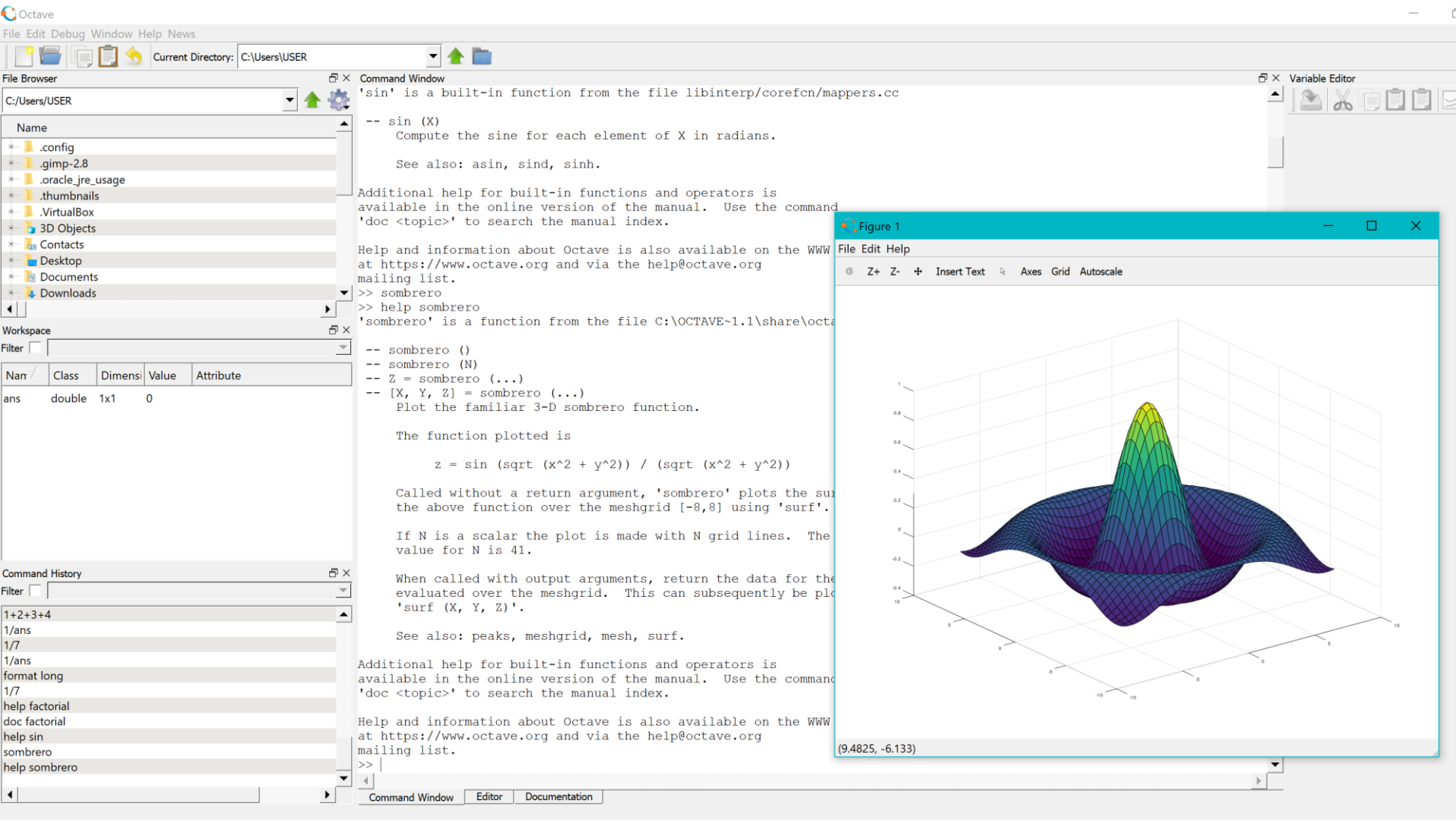
lub alternatywnie:

```
u=inv(A)*f'
```

Wynikiem jest:

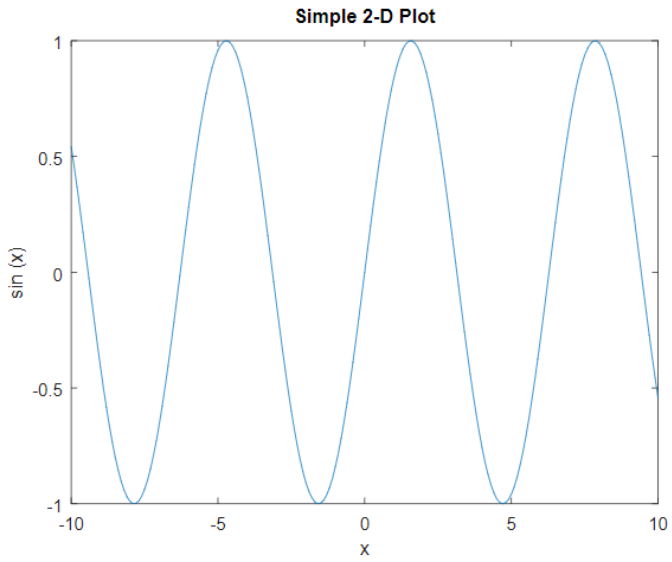
```
u = [2 3]
```

Wykresy w Oktawie



Wykresy 2D

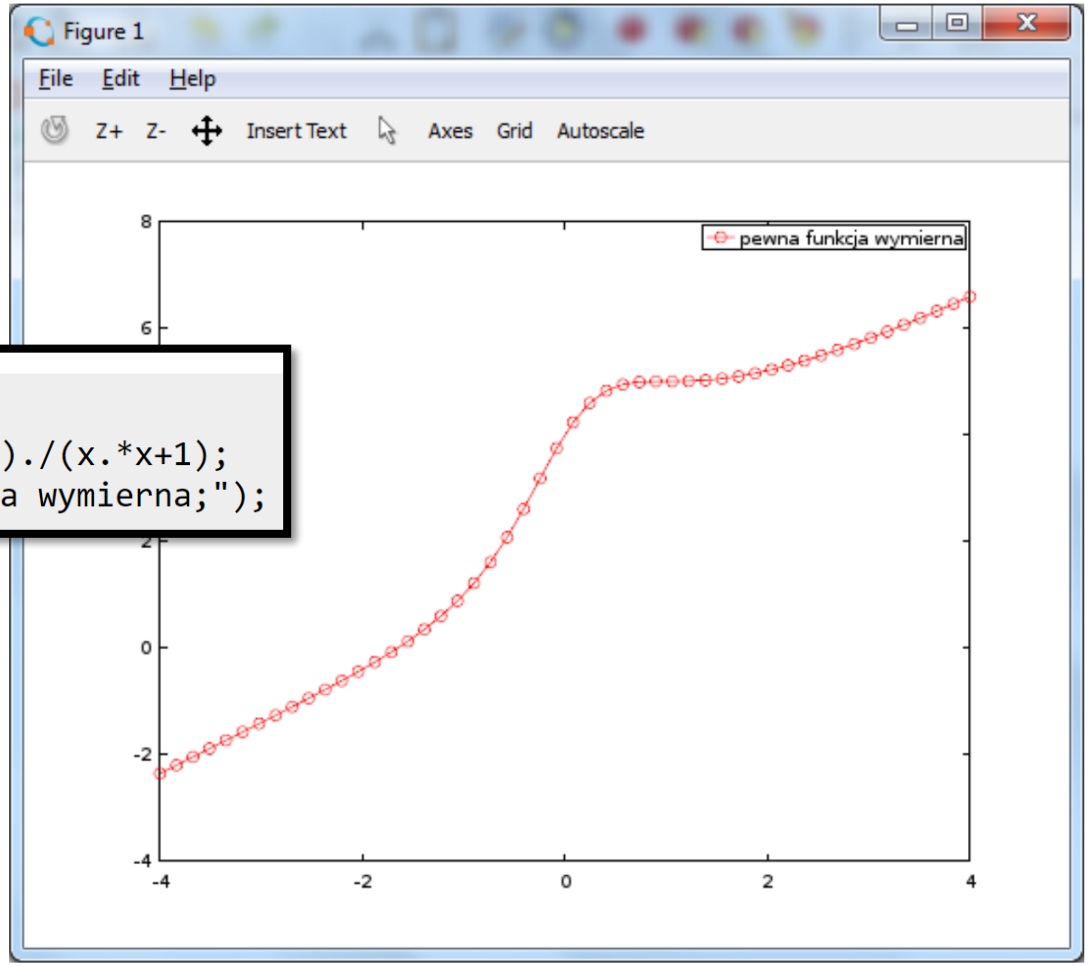
```
x = -10:0.1:10; # Create an evenly-spaced vector from -10..10
y = sin (x);    # y is also a vector
plot (x, y);
title ("Simple 2-D Plot");
xlabel ("x");
ylabel ("sin (x)");
```



Octave to tylko i aż kalkulator na sterydach

Wykresy 2D

```
x = linspace(-4,4,50);
y = (x.^3 + 2*x.^2 + 3*x + 4)./(x.*x+1);
plot(x, y, "ro-;pewna funkcja wymierna;");
```



Zmianie uległa definicja wektora x. Funkcja **linspace (a,b, N)** generuje ciąg N równoodległych punktów, z których pierwszy równy jest a, a ostatni – b. Zaletą tego podejścia jest to, że od razu widać, z ilu elementów składa się x.

Druga zmiana zaszła w wywołaniu funkcji plot: doszedł trzeci argument, napis "**ro-;pewna funkcja wymierna**;".

Ten trzeci argument definiuje format wykresu i składa się z liter i symboli umieszczonych w napisie w dowolnej kolejności

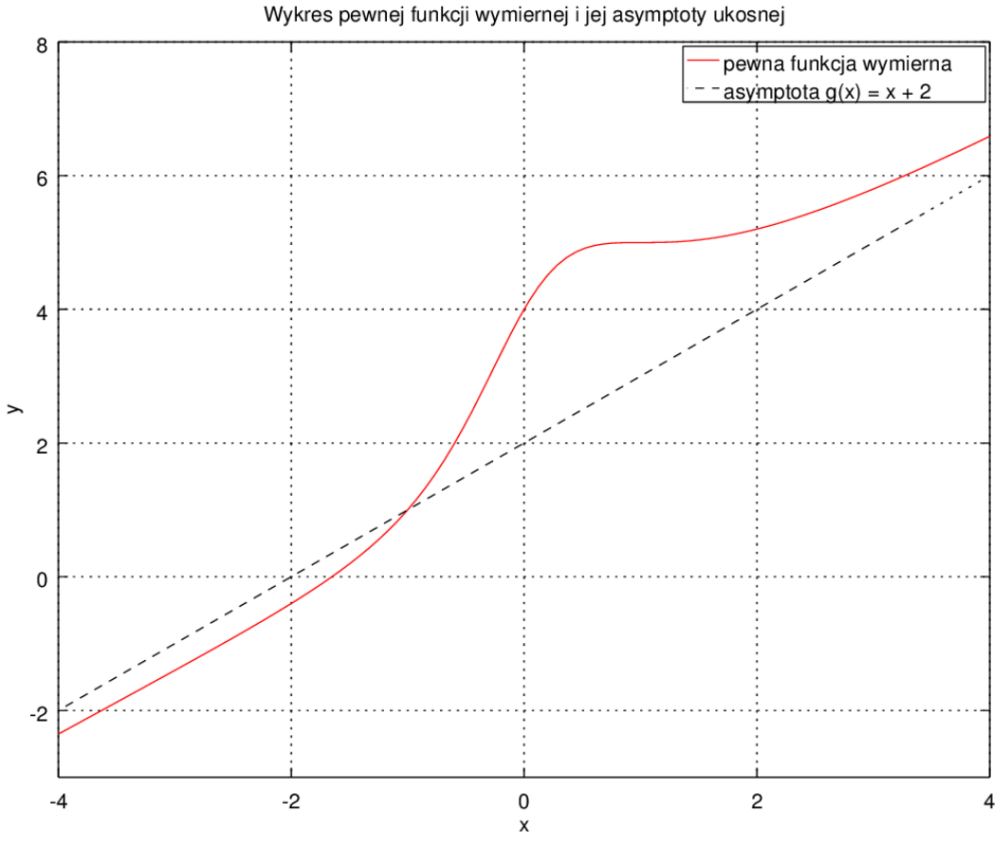
Opcje stylu

- Styl linii:
  - `-` Linia ciągła (opcja domyślna)
  - `--` Linia przerywana
  - `:` Linia kropkowana
  - `-.` Linia przerywana z kropkami
- Styl symbolu reprezentującego punkt danych:
  - `+` plusik
  - `o` kółko
  - `*` gwiazdka
  - `.` punkt
  - `x` krzyżyk
  - `s` kwadrat
  - `d` romb (ang. *diamond*)
  - `^` trójkąt z ostrzem w górę
  - `v` trójkąt w dól
  - `>` trójkąt z ostrzem w prawo
  - `<` trójkąt z ostrzem w lewo
  - `p` pięciokąt
  - `h` sześciokąt (ang. ***h**exagram*)
- Kolor linii i/lub symbolu:
  - `k` czarny (ang. *blac**K***)
  - `r` czerwony
  - `g` zielony
  - `b` niebieski
  - `m` *magenta*
  - `c` *cyan*
  - `w` biały
- Opis wykresu (legenda)
  - `;legenda;`

Dodajmy jeszcze kilka wodotrysków, jak podpisy osi, tytuł całego rysunku (niestety, bez polskich liter), siatkę, ograniczenie zakresu osi „y" do `[-3:8]` i zapiszmy wykres na dysku w formacie eps i png:

```
x = linspace(-4,4,100);
y = (x.^3 + 2*x.^2 + 3*x + 4)./(x.*x+1);
y2 = x + 2;
plot(x, y, "r-;pewna funkcja wymierna;", x, y2, "k--;asymptota g(x) = x + 2;");
xlabel("x");
ylabel("y");
title("Wykres pewnej funkcji wymiernej i jej asymptoty ukosnej");
ylim([-3,8]);
grid on;
print "rys.eps"
print "rys.png"
```

Oto wynik działania powyższego skryptu:



Po paru wodotryskach...