Using ChatGPT for numerical solution of first and second order ordinary differential equations

Saso Koceski, Natasa Koceska, Limonka Koceva Lazarova, Marija Miteva, Biljana Zlatanovska

Faculty of Computer Science, University Goce Delcev Stip

FMNS, Blagoevgrad,
Bulgaria 14 - 18 June, 2023

Introduction

Differential equations play a crucial role in engineering and science as they describe a wide range of natural phenomena and physical processes.

These equations are used to model and predict the behavior of complex systems.

They are used in **various fields**:

- **In mechanics** - used to describe the motion of particles and systems of particles, to describe the behavior of fluids in motion, they are also used to describe mechanical vibrations and in many other fields.
- **In electrical engineering** - used to model and predict the behavior of electrical systems
- **In chemistry** - used to describe the rates of chemical reactions, which is important in the design of chemical processes and the production of materials.
- **In biology** - used to model the behavior of biological systems, such as the growth of populations and the spread of diseases.

Introduction

Differential equations can be classified into several types based on their order, linearity, and dependence on the independent variable.

An ordinary differential equation of the n-th order is of the form

$$F\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \frac{d^3y}{dx^3}, \ldots, \frac{d^ny}{dx^n}\right) = 0$$

Its general solution contains n arbitrary constants and is of the form

$$\varphi\left(x, y, c_1, c_2, \ldots, c_n\right) = 0$$

To obtain its particular solution, n conditions must be given so that the constants c_1, c_2, c_n can be determined. If these conditions are prescribed at one point only, then the differential equation together with the conditions constitute an initial value problem of the n-th order. If the conditions are prescribed at two or more points, then the problem is termed as boundary value problem.

For the purpose of this research, we will be focused on ordinary differential equations of first and second order.

Problem statement

Although various analytical methods for finding general solutions of ordinary differential equations of first and second order exists, it is still challenging mathematical problem that requires significant human effort.

However, the latest development of Artificial Intelligence (AI) and models capable of mimicking human behavior and improving over time, have already shown their strength and capabilities in finding creative solutions to multiple problems in various fields.

Therefore, in our research, we have conducted an empirical study to validate the performances and capabilities of OpenAI's large language models, by using the free version of its ChatGPT chat-bot application for finding numerical solution of differential equations of first and second order.

Moreover, we have evaluated the capabilities of ChatGPT for generating numerical solution for one specific problem (Newton's law of cooling states) and refining the solution to satisfy our specific requirements.

What is GPT-3 and its architecture

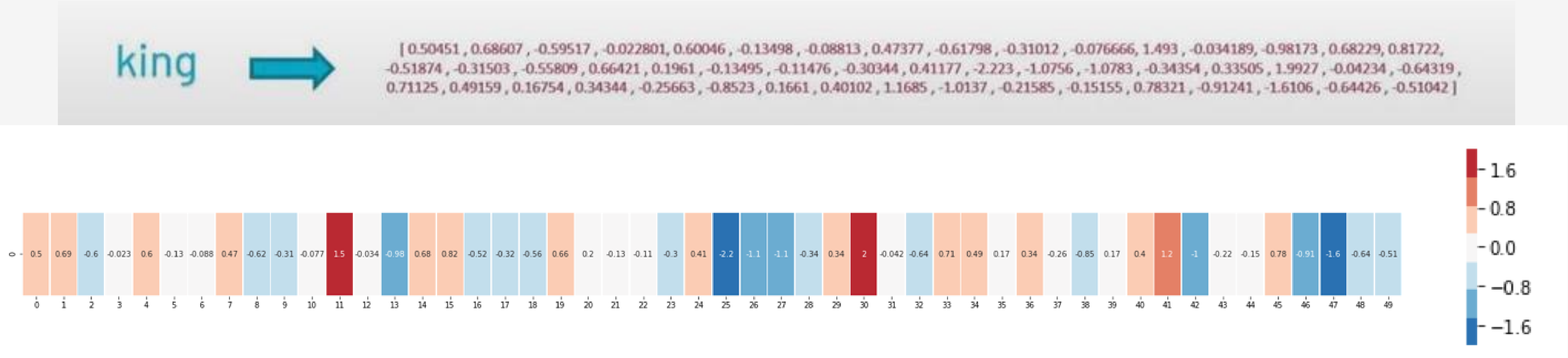
GPT-3 stands for - **G**enerative **P**re-trained **T**ransformer version 3. It is an auto-regressive language model that uses deep learning to produce human-like text.

- **G**enerative – a statistical model for creating new data based on the relationship between variables in a data set.
- **P**re-trained - a model that has been previously trained on a large data set.
- **T**ransformer - neural network architecture presented in 2017. A deep learning model adapted to work with sequential data.

What is GPT-3 and its architecture

Language model

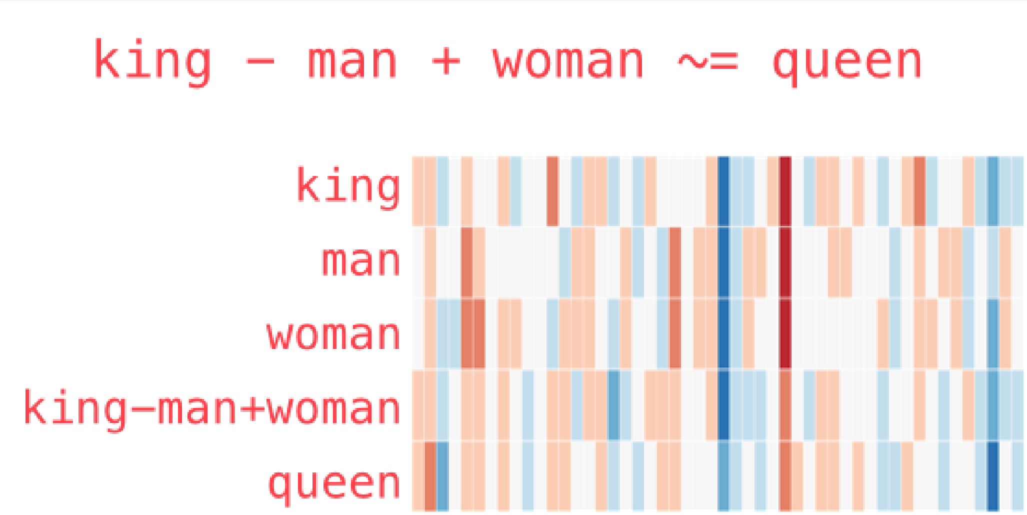
- ➢ Machine learning model - represents the language domain
- ➢ Base - representation of a word in the form of a vector of numbers
  - ➢ contain important words;
  - ➢ suitable for computers;
  - ➢ we can calculate similarity (distance).



What is GPT-3 and its architecture

Language model

- ➢ Machine learning model - represents the language domain
- ➢ Base - representation of a word in the form of a vector of numbers
    - ➢ contain important words;
    - ➢ suitable for computers;
    - ➢ we can calculate similarity (distance).

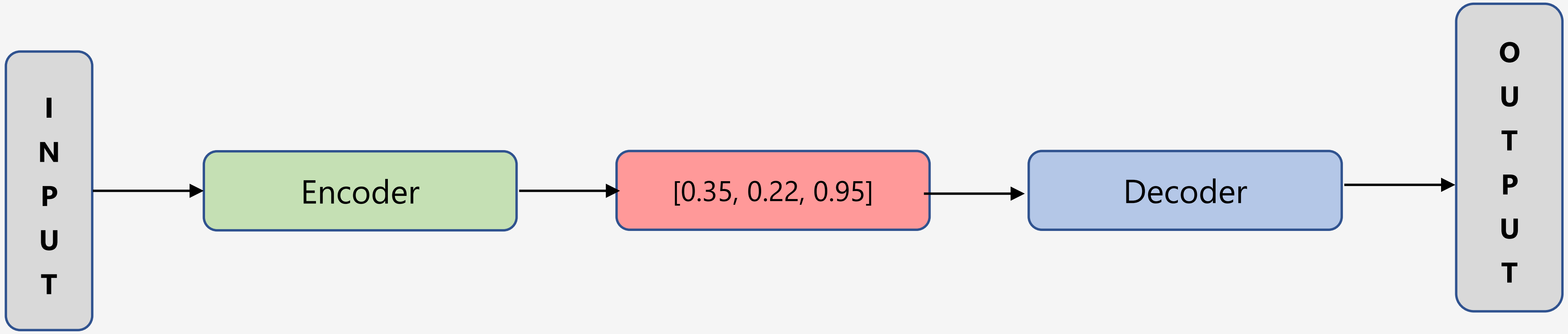



```
model.most_similar(positive=["king","woman"], negative=["man"])

[('queen', 0.8523603677749634),
 ('throne', 0.7664333581924438),
 ('prince', 0.7592144012451172),
 ('daughter', 0.7473883032798767),
 ('elizabeth', 0.7460219860076904),
 ('princess', 0.7424570322036743),
 ('kingdom', 0.7337411642074585),
 ('monarch', 0.721449077129364),
 ('eldest', 0.7184862494468689),
 ('widow', 0.7099430561065674)]
```

What is GPT-3 and its architecture

GPT-3:

- ENCODER - from the input data it generates its vector representation
- DECODER - from the vector form it generates the output sequence



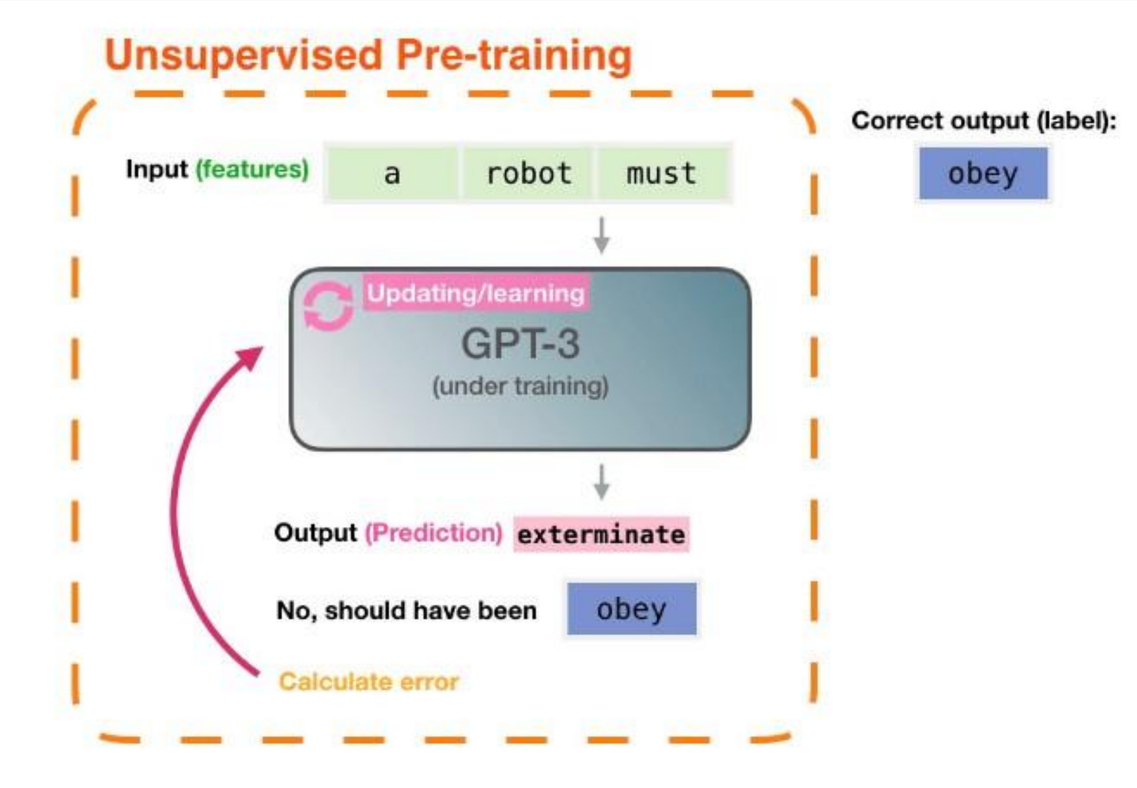
What is GPT-3 and its architecture

Training datasets:

Dataset	Quantity	Weight in training mix	Description
Common crawl (filtered)	410 billion	60%	data obtained from 8 years web crawling (PB)
WebText2	19 billion	22%	text from web pages linked by Reddit posts with 3+ votes
Books1	12 billion	8%	Book collection
Books2	55 billion	8%	Book collection
Wikipedia	3 billion	3%	Text from Wikipedia

What is GPT-3 and its architecture

Training process:



Goals of this research

In our research we were interested to discover the frequency of correct answers and explanations of ChatGPT to the problems defined in our dataset.

To investigate this, we have created a dataset composed of 215 differential equations extracted from various academic books and textbooks [1-10], as well as our own written differential equations that could not be found in any of the previous sources.

The structure of the input dataset is given in the Table 1 bellow.

Type of equation	Number of equations in the dataset
First order differential equations	110
Second order differential equations	105

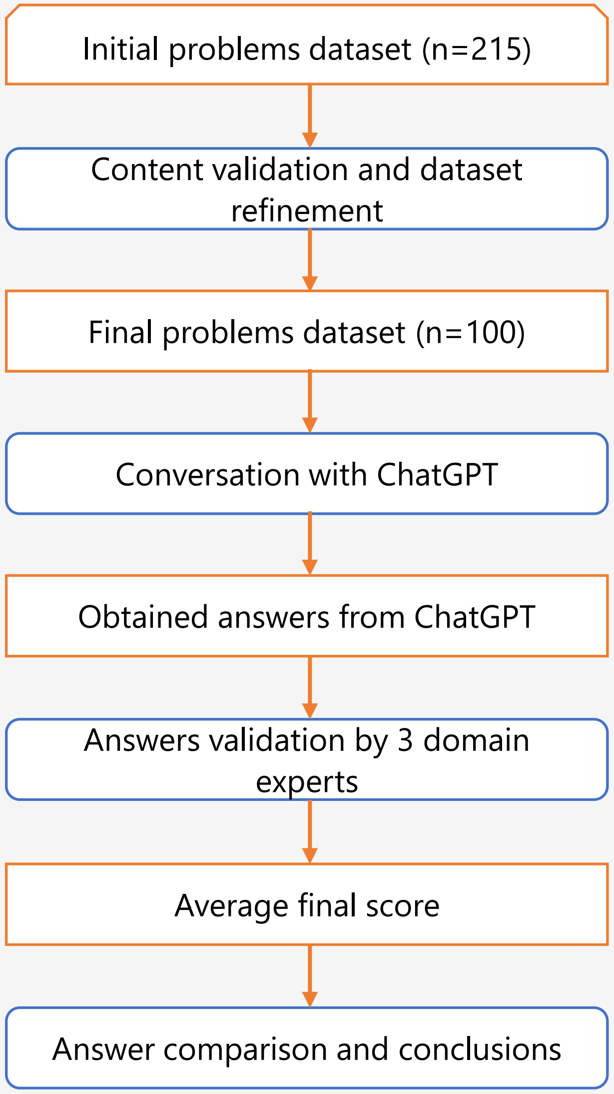
Methodology

In this research we have created an initial dataset composed of 215 ordinary differential equations extracted from various academic books and textbooks [1-10], as well as our own written differential equations that could not be found in any of the previous sources.

Then domain experts have reviewed the initial dataset and validated its content. After the revision the problems dataset was reduced to 100 equations (50 of them were first order and other 50 second order equations.

The structure of the final problems dataset is given in the Table bellow.

Type of equation	Number of equations in the dataset
First order linear homogeneous differential equations	50
Second order linear homogeneous differential equations	50



Methodology

Then the problems were given as input to ChatGPT in a single conversation.

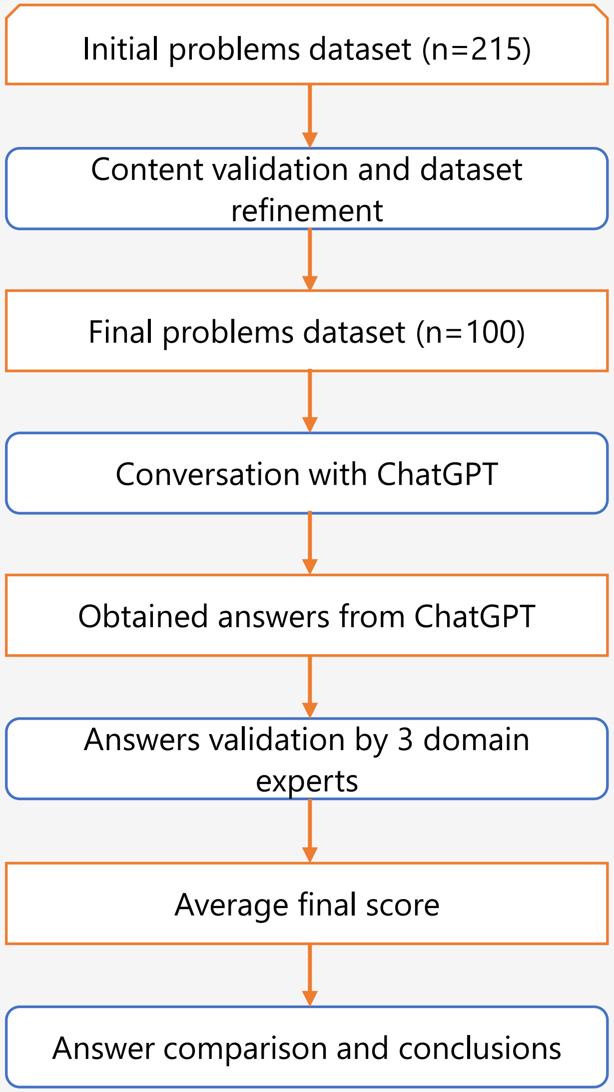
Obtained answers from ChatGPT were graded by 3 independent domain experts using grades from 1-5 Likert scale.

Final score for every problem was obtained as an average of the obtained three answers.

Each conversation with ChatGPT contained the following elements:

1) Definition of the equation, definition of the initial conditions

2) Requirement to write the numerical solution using particular method, plot the solution and find the value of y at given points.



Methodology

1. Notes on Diffy Qs - Differential Equations for Engineers, by Jiří Lebl, 2022, Amazon KDP edition ISBN-13:978-1-70623-023-6
2. Differential Equations, Jeffrey R.Chasnov, 2021, Hong Kong University of Science and Technology
3. ELEMENTARY DIFFERENTIAL EQUATIONS, William F. Trench, 2013, Brooks/Cole Thomson Learning
4. DIFFERENTIAL EQUATIONS AND LINEAR ALGEBRA, Gilbert Strang, Wellesley - Cambridge Press, MIT, 2022
5. Ordinary Differential Equations, Gabriel Nagy, Michigan State University Press, 2021
6. Fundamentals of Differential Equations, R. Kent Nagle, Edward B. Saff, Arthur David Snider, Pearson Education, Feb 28, 2012
7. ORDINARY DIFFERENTIAL EQUATIONS - An Elementary Textbook for Students of Mathematics, Engineering, and the Sciences, Morris Tenenbaum, Harry Pollard, Dover Publications, 1963
8. Numerical Methods for Scientists and Engineers, R. W. Hamming, Dover Publications; 2nd Revised ed. Edition, 1987
9. Python Programming And Numerical Methods: A Guide For Engineers And Scientists, Alexandre Bayen, Qingkai Kong, and Timmy Siauw, Elsevier Science, Dec 15, 2020
10. Numerical Python - A Practical Techniques Approach for Industry, Robert Johansson, Apress, 2015

Methodology

For numerical solution of the ordinary differential equation of first and second order the following methods were applied:

- Euler's method
- Taylor's series method
- Picard's method
- Runge-Kutta method
- Milne's method
- Adams-Bashforth method

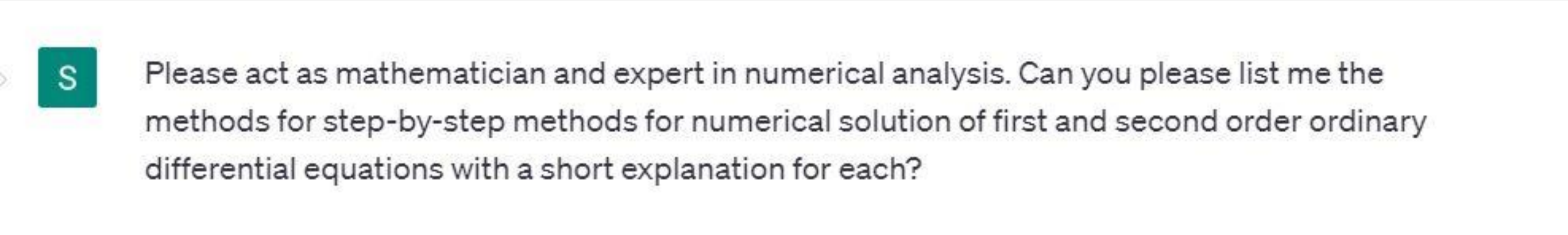
Regardless of the method used, ChatGPT was asked to generate programming code in Python programming language. Once the solution was generated, the programming code was simply copied and pasted in Google Collaboratory where it was executed (and eventually debugged).

Methodology

ChatGPT is trained with reinforcement learning through human feedback and reward models that rank the best responses. This feedback helps augment ChatGPT with machine learning to improve future responses.

One way to get the best results from ChatGPT is to assign it a role. This is a fantastic method of getting more appropriate responses to your prompts. Asking ChatGPT a question will always produce a response of some sort, but its relevance might not be suited to your requirements.

Therefore, before starting of our experimental interactive session we have assigned it a role and tried to test its initial theoretical knowledge about one specific subject.



Methodology

And here's the result we obtained:

Certainly! Here are some common step-by-step methods for the numerical solution of first and second-order ordinary differential equations:

1. Euler's Method: Euler's method is a simple and straightforward method for solving first-order ordinary differential equations. It approximates the derivative using a forward difference and updates the solution step by step. While it is easy to implement, it has limited accuracy and can exhibit significant error accumulation.
2. Improved Euler's Method (Heun's Method): This method improves upon Euler's method by using a midpoint estimate of the derivative to update the solution. It reduces the error compared to Euler's method but is still a first-order method.
3. Runge-Kutta Methods: Runge-Kutta methods are a family of numerical methods that iteratively approximate the solution of ordinary differential equations. The most commonly used is the fourth-order Runge-Kutta method (RK4). It involves calculating several intermediate slopes to estimate the derivative and update the solution. RK4 offers higher accuracy compared to Euler's and improved Euler's methods.
4. Adams-Bashforth Methods: Adams-Bashforth methods are explicit multi-step methods for solving first-order ordinary differential equations. They use previous solution values to estimate the derivative and predict future solution values. The accuracy of these methods increases with the number of steps used.

6. Finite Difference Methods: Finite difference methods approximate derivatives using difference equations. They discretize the domain and replace the derivatives with difference approximations. Finite difference methods can be applied to both first and second-order ordinary differential equations and are commonly used in numerical analysis.
7. Taylor Series Methods: Taylor series methods express the solution of an ordinary differential equation as a power series expansion around a given point. By truncating the series and using a finite number of terms, a numerical approximation of the solution can be obtained. Taylor series methods provide high accuracy but can be computationally demanding.

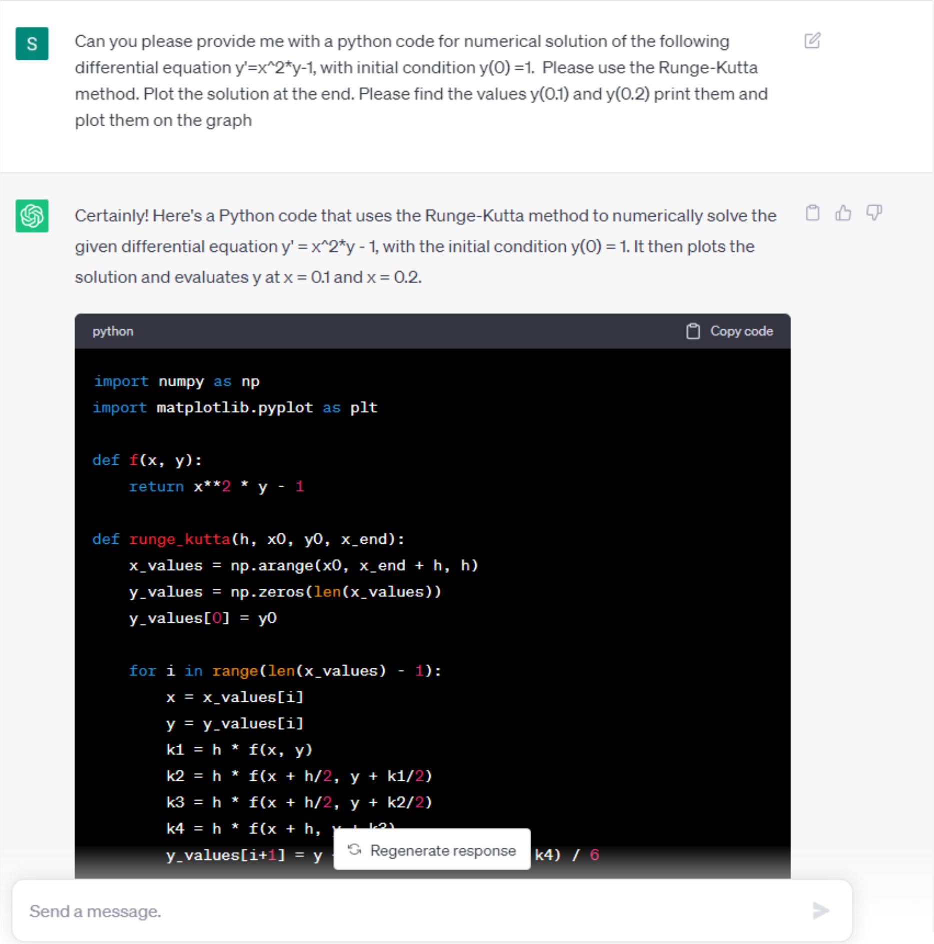
These methods vary in terms of accuracy, stability, and computational complexity. The choice of method depends on the specific characteristics of the differential equation, such as order, linearity, stiffness, and any specific requirements for accuracy and efficiency.

Regenerate response

Send a message.

Evaluation

All equations were introduced to ChatGPT in the same session (one after another), and for each equation numerical solution using different method was generated. One such conversation is presented bellow:



S

Can you please provide me with a python code for numerical solution of the following differential equation y'=x^2*y-1, with initial condition y(0) =1. Please use the Runge-Kutta method. Plot the solution at the end. Please find the values y(0.1) and y(0.2) print them and plot them on the graph

Certainly! Here's a Python code that uses the Runge-Kutta method to numerically solve the given differential equation y' = x^2*y - 1, with the initial condition y(0) = 1. It then plots the solution and evaluates y at x = 0.1 and x = 0.2.

python   Copy code

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

def f(x, y):
    return x**2 * y - 1

def runge_kutta(h, x0, y0, x_end):
    x_values = np.arange(x0, x_end + h, h)
    y_values = np.zeros(len(x_values))
    y_values[0] = y0

    for i in range(len(x_values) - 1):
        x = x_values[i]
        y = y_values[i]
        k1 = h * f(x, y)
        k2 = h * f(x + h/2, y + k1/2)
        k3 = h * f(x + h/2, y + k2/2)
        k4 = h * f(x + h,
        y_values[i+1] = y                         k4) / 6
```

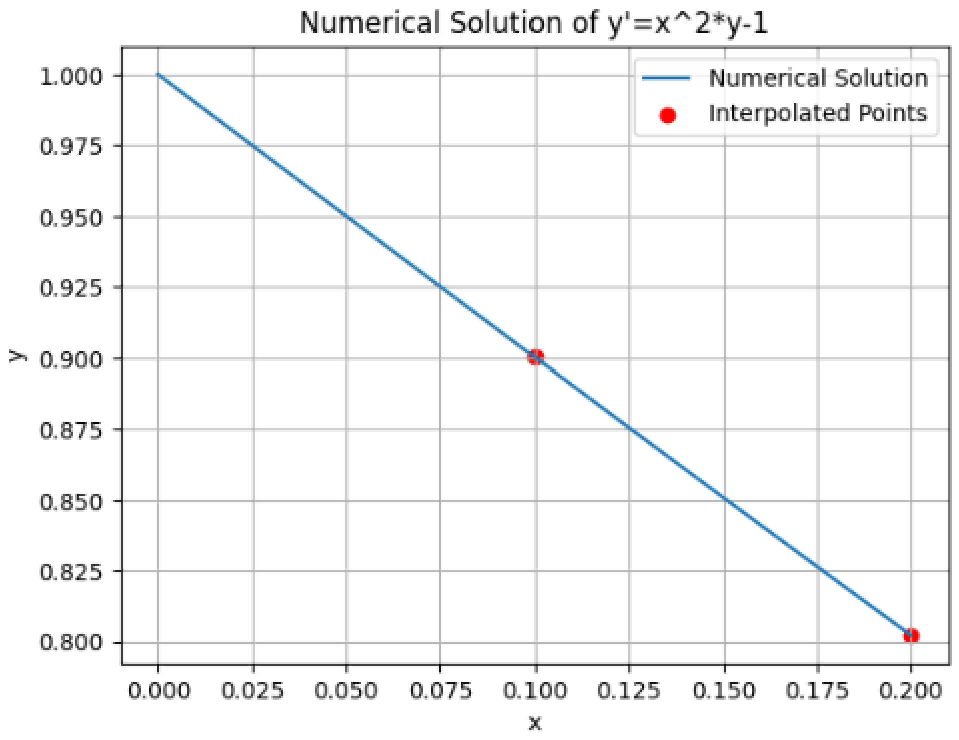
Regenerate response

Send a message.

Evaluation

The result was evaluated in Google Colaboratory

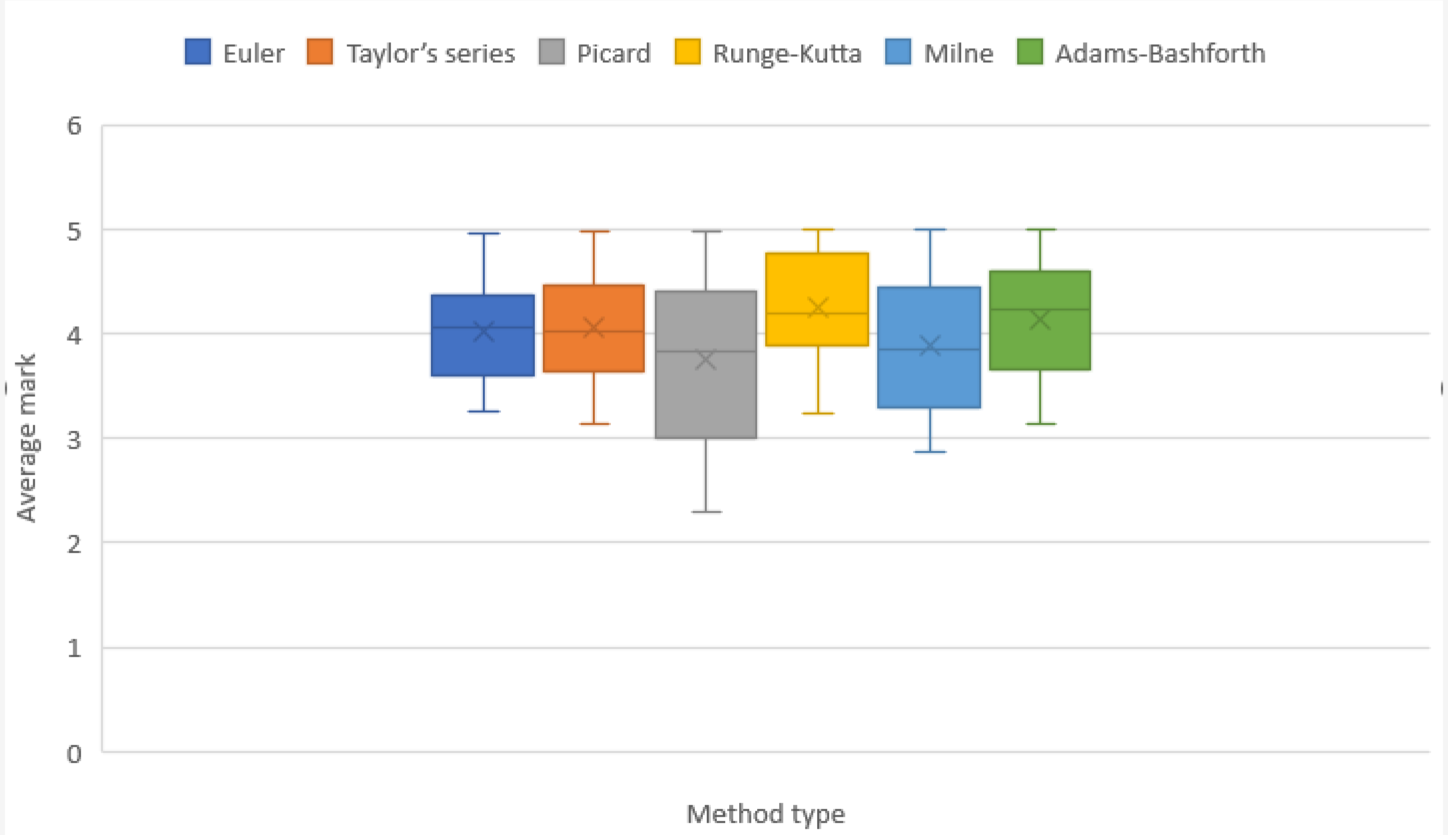
```
y(0.1) = 0.90031
y(0.2) = 0.80227
```




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

def f(x, y):
    return x**2 * y - 1

def runge_kutta(h, x0, y0, x_end):
    x_values = np.arange(x0, x_end + h, h)
    y_values = np.zeros(len(x_values))
    y_values[0] = y0

    for i in range(len(x_values) - 1):
        x = x_values[i]
        y = y_values[i]
        k1 = h * f(x, y)
        k2 = h * f(x + h/2, y + k1/2)
        k3 = h * f(x + h/2, y + k2/2)
        k4 = h * f(x + h, y + k3)
        y_values[i+1] = y + (k1 + 2*k2 + 2*k3 + k4) / 6

    return x_values, y_values

# Parameters
h = 0.01
x0 = 0
y0 = 1
x_end = 0.2

# Solve the differential equation
x_values, y_values = runge_kutta(h, x0, y0, x_end)

# Evaluate y at x = 0.1 and x = 0.2
y_at_01 = np.interp(0.1, x_values, y_values)
y_at_02 = np.interp(0.2, x_values, y_values)
print(f"y(0.1) = {y_at_01:.5f}")
print(f"y(0.2) = {y_at_02:.5f}")

# Plot the solution
plt.plot(x_values, y_values, label='Numerical Solution')
plt.scatter([0.1, 0.2], [y_at_01, y_at_02], color='red', label='Interpolated Points')
plt.xlabel('x')
plt.ylabel('y')
plt.title("Numerical Solution of y'=x^2*y-1")
plt.legend()
plt.grid(True)
plt.show()
```

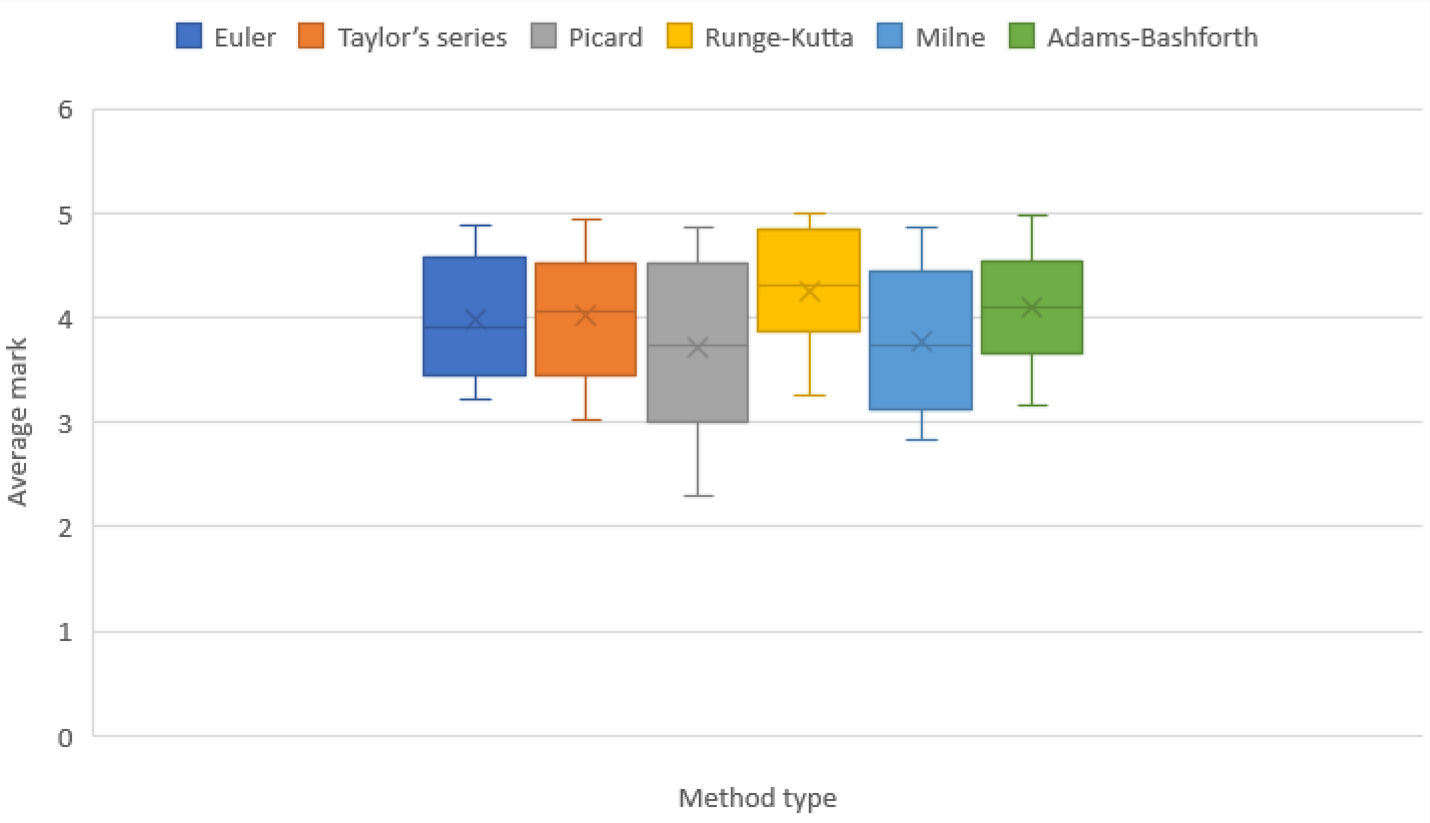
Results

Evaluation results for first order ODE



Results

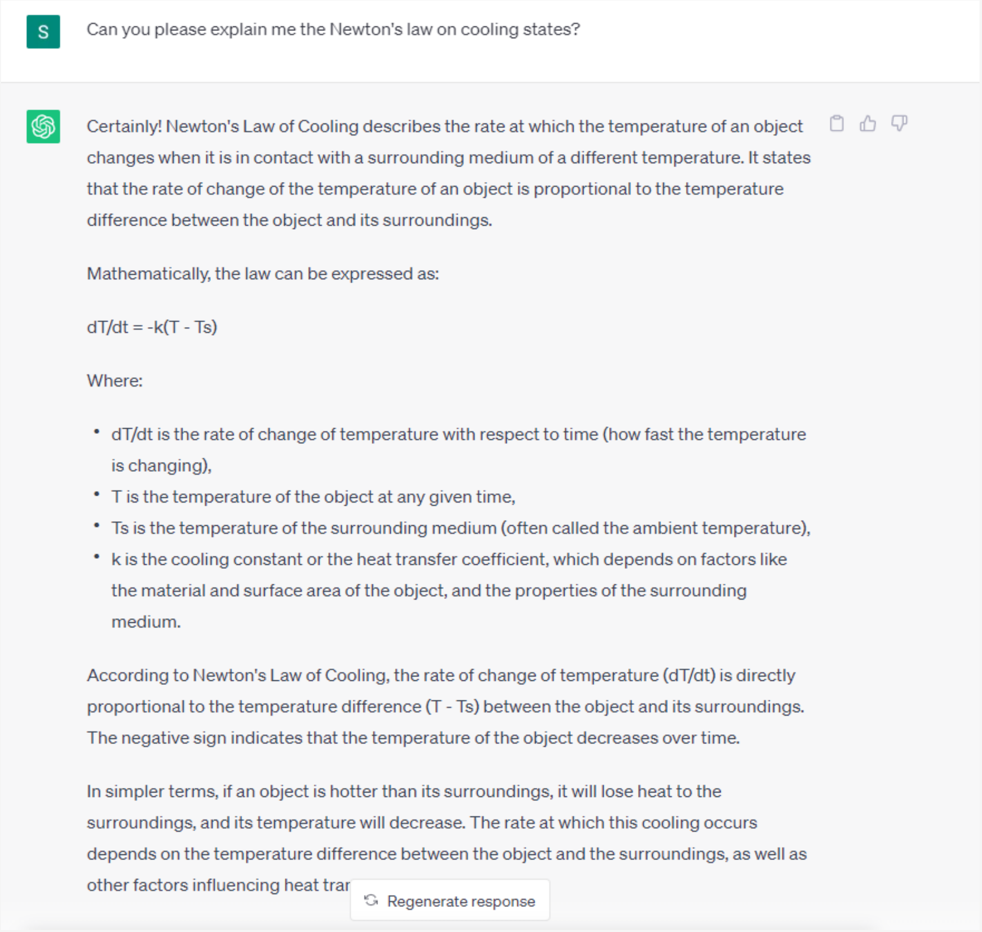
Evaluation results for second order ODE



Results

We were also interested in evaluating the ChatGPT capabilities for analytical and numerical solution of some well-known engineering problems. We took Newton's Law on cooling states as an example.

The conversation is given in the following:



S Can you please explain me the Newton's law on cooling states?

Certainly! Newton's Law of Cooling describes the rate at which the temperature of an object changes when it is in contact with a surrounding medium of a different temperature. It states that the rate of change of the temperature of an object is proportional to the temperature difference between the object and its surroundings.

Mathematically, the law can be expressed as:

dT/dt = -k(T - Ts)

Where:

- dT/dt is the rate of change of temperature with respect to time (how fast the temperature is changing),
- T is the temperature of the object at any given time,
- Ts is the temperature of the surrounding medium (often called the ambient temperature),
- k is the cooling constant or the heat transfer coefficient, which depends on factors like the material and surface area of the object, and the properties of the surrounding medium.

According to Newton's Law of Cooling, the rate of change of temperature (dT/dt) is directly proportional to the temperature difference (T - Ts) between the object and its surroundings. The negative sign indicates that the temperature of the object decreases over time.

In simpler terms, if an object is hotter than its surroundings, it will lose heat to the surroundings, and its temperature will decrease. The rate at which this cooling occurs depends on the temperature difference between the object and the surroundings, as well as other factors influencing heat trar

Results

The bot was asked for analytical solution of the ODE first:

S

Can you please find me the generic solution of the differential equation describing Newton's law of cooling states dT/dt = -k(T - Ts)?

Certainly! The differential equation describing Newton's Law of Cooling is:

$dT/dt = -k(T - Ts)$

To find the generic solution of this differential equation, we can rearrange it and separate the variables:

$dT/(T - Ts) = -k\, dt$

Now, we integrate both sides of the equation:

$\int dT/(T - Ts) = -k \int dt$

The integral on the left side can be solved using natural logarithms:

$\ln|T - Ts| = -kt + C$

Here, C is the constant of integration. Now, we can exponentiate both sides to eliminate the logarithm:

$|T - Ts| = e^{(-kt + C)}$

Regenerate response

Using the absolute value notation, we can split this equation into two cases:

1. $(T - Ts) = e^{(-kt + C)}$ if $(T > Ts)$
2. $-(T - Ts) = e^{(-kt + C)}$ if $(T < Ts)$

Next, we can simplify these equations by absorbing the constant C into another constant. Let's call this new constant A:

1. $(T - Ts) = Ae^{(-kt)}$ if $(T > Ts)$
2. $-(T - Ts) = Ae^{(-kt)}$ if $(T < Ts)$

We can further rearrange these equations to solve for T:

1. $T = Ae^{(-kt)} + Ts$ if $(T > Ts)$
2. $T = Ts - Ae^{(-kt)}$ if $(T < Ts)$

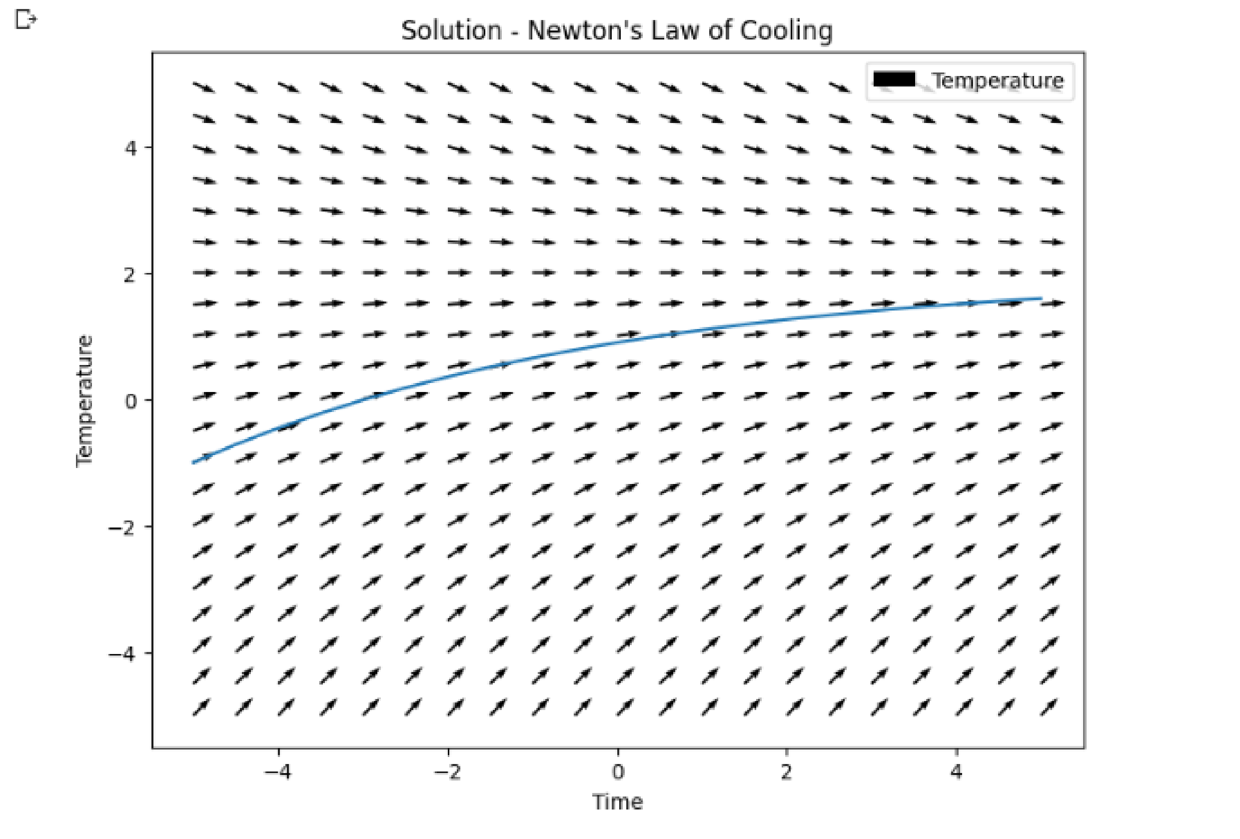
These are the generic solutions to the differential equation describing Newton's Law of Cooling. The constant A represents the initial condition or the difference between the initial temperature of the object and the ambient temperature. It can be determined based on the specific problem or initial conditions provided.

Results

And then the numerical solution, for specific initial condition, ambient temperature and cooling constant, was generated and plotted.

```
[ ] import numpy as np
    import matplotlib.pyplot as plt
    from scipy.integrate import odeint

    def newtons_law_cooling(T, t, k, Ts):
        return -k * (T - Ts)

    # Constants and initial conditions
    k = 0.2
    Ts = 2.0
    T0 = -1.0

    # Time points for evaluation
    t = np.linspace(-5, 5, 21)

    # Solve the differential equation
    solution = odeint(newtons_law_cooling, T0, t, args=(k, Ts))

    # Plotting the slope field
    T, tt = np.meshgrid(np.linspace(-5, 5, 21), np.linspace(-5, 5, 21))
    dy = newtons_law_cooling(T, tt, k, Ts)
    dx = np.ones(dTdt.shape)
    dyu = 4*dy/np.sqrt(dx**2+dy**2)
    dxu = 4*dx/np.sqrt(dx**2+dy**2)
    plt.figure(figsize=(8, 6))
    plt.quiver(tt, T, dxu, dyu, angles='xy')
    plt.xlabel('Time')
    plt.ylabel('Temperature')
    plt.title("Slope Field - Newton's Law of Cooling")

    # Plotting the solution
    plt.plot(t, solution)
    plt.xlabel('Time')
    plt.ylabel('Temperature')
    plt.title("Solution - Newton's Law of Cooling")
    plt.legend(['Temperature'])
    plt.show()
```

Just a single intervention was made on the proposed solution in order to normalize the potential field vectors



Conclusion

- ❑ Based on our findings, ChatGPT is capable of providing numerical solution to both first and second order differential equations with an average mark of around 4 and accuracy of about 80%

- ❑ There was no significant difference in the application's capability to solve both types of ODE’s.

- ❑ ChatGPT showed strong conceptual understanding of the subject and capability to provide even analytical solution of first and second order ODE

- ❑ In the future, with further improvement, ChatGPT-like programs may be used for academic purposes.

Thank you for your attention

Q&A?