Date: 3rdMay-2025

SOFTWARE ANALYSIS OF NUMERICAL METHODS FOR SOLVING DIFFERENTIAL EQUATIONS (USING PYTHON)

Rizayeva Bahoroy Jahongir qizi
Karshi State University,Applied Mathematics Department, 2nd year student
rizayevabahoroy@gmail.com

Anotatsiya:Ushbu maqolada differensial tenglamalarni yechishning asosiy numerik usullari — Eyler usuli, Runge-Kutta usuli va ularning dasturiy realizatsiyasi tahlil qilinadi. Maqolada Python dasturlash tili yordamida oddiy differensial tenglamalarni yechish algoritmlari ishlab chiqilib, natijalar graflar yordamida vizual tarzda tahlil qilinadi. Shuningdek, har bir usulning afzalliklari va kamchiliklari solishtiriladi.

Аннотация:В данной статье рассматриваются основные численные методы решения дифференциальных уравнений — метод Эйлера, метод Рунге-Кутты и их программная реализация. С использованием языка программирования Python разрабатываются алгоритмы решения обыкновенных дифференциальных уравнений и проводится визуальный анализ результатов с помощью графиков. Также сравниваются достоинства и недостатки каждого метода.

Annotation:This article analyzes the main numerical methods for solving differential equations — the Euler method, the Runge-Kutta method, and their software implementation. Using the Python programming language, algorithms for solving ordinary differential equations are developed and results are analyzed visually through graphs. The advantages and disadvantages of each method are also compared.

Kalit so‘zlar:differensial tenglama, numerik usullar, Eyler usuli, Runge-Kutta usuli, Python, algoritm, vizualizatsiya.

Ключевые слова:дифференциальное уравнение, численные методы, метод Эйлера, метод Рунге-Кутты, Python, алгоритм, визуализация.

Keywords:differential equation, numerical methods, Euler method, Runge-Kutta method, Python, algorithm, visualization.

In many areas of modern science and technology—such as physics, biology, economics, and engineering—numerous problems are modeled using differential equations. Unfortunately, not all of these equations have exact analytical solutions. Therefore, in applied mathematics, it is common to use numerical methods to find approximate solutions to such equations. This paper discusses the theoretical foundations of the Euler and Runge-Kutta methods and provides practical implementations using the Python programming language.

A differential equation is an equation that involves an unknown function and its derivatives. It is widely used to mathematically model various physical, biological, economic, or engineering processes. For example, heat diffusion, substance exchange, population growth, and changes in financial interest rates can all be represented using

differential equations. An ordinary differential equation (ODE) involves only one independent variable and its derivatives. Its general form is as follows:

$\frac{dy}{dx} = f(x, y), \ y(x_0) = y_0$ Where:

- y is the unknown function (the output),
- x is the independent variable (usually time or space),
- f(x;y) is a given functional expression,
- x_0 ,y_0 are the initial conditions used to determine a unique solution.

Differential equations are divided into two main types:

1. Ordinary Differential Equations (ODEs) – involve only one independent variable.
2. Partial Differential Equations (PDEs) – involve multiple independent variables and are used to model multi-dimensional processes (e.g., wave propagation, fluid flow).

Euler Method

The Euler method is one of the simplest and oldest numerical methods for solving ordinary differential equations. It approximates the solution of the differential equation step-by-step in discrete form.Given the equation:

$\frac{dy}{dx} = f(x, y), \ y(x_0) = y_0$

The Euler method calculates the value at the next step using the formula:

$y_{n+1} = y_n + h*f(x_n, y_n)$ Where:

- h is the step size,
- x_n, y_n are the current values,
- y_{n+1} is the approximate value at the next point.

The Euler method is typically used at the learning stage or for initial estimations due to its simplicity, although it lacks precision and accumulates error quickly.

Runge-Kutta Method (4th Order)

The Runge-Kutta method is more accurate and stable than the Euler method. The fourth-order Runge-Kutta method, in particular, is the most widely used in practice. This method provides high-precision results by performing several intermediate calculations at each step.

The formulas are as follows:

$k_1 = f(x_n, y_n)$

$k_2 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$

$k_3 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$

$k_4 = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_3)$

$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Where:

- k_1, k_2, k_3, k_4 are intermediate calculations,
- h is the step size,
- y_{n+1} is the approximate value at the next step.

Although this method requires more computations than Euler's method, it provides significantly greater accuracy and is widely used in scientific and engineering applications.

In solving differential equations numerically, modeling them programmatically under real-world conditions is of great importance. Below is the Python implementation of the Euler and fourth-order Runge-Kutta (RK4) methods for solving the differential equation

$\frac{dy}{dx}$= x + y, y(0) = 1

Euler's Method

Euler's method is the simplest and most intuitive numerical approach to solving differential equations. It estimates the solution step by step by approximating the function's curve using a tangent line at each point.

Python implementation:

```
def f(x, y):
return x + y
def euler(x0, y0, h, n):
x, y = x0, y0
for i in range(n):
y = y + h * f(x, y)
x = x + h
print(f"Euler: x = {x:.2f}, y = {y:.4f}")
```

This function generates an approximate solution to the given differential equation in a sequential manner. At each iteration, the point is updated and the result is printed.

Fourth-Order Runge-Kutta Method

The fourth-order Runge-Kutta method is significantly more accurate and stable compared to Euler's method. It computes the next value of using four intermediate evaluations , which leads to a more precise estimation of the curve.

Python implementation:

```
def rk4(x0, y0, h, n):
x, y = x0, y0
for i in range(n):
k1 = f(x, y)
k2 = f(x + h/2, y + h/2 * k1)
k3 = f(x + h/2, y + h/2 * k2)
k4 = f(x + h, y + h * k3)
y = y + h/6 * (k1 + 2*k2 + 2*k3 + k4)
x = x + h
print(f"RK4: x = {x:.2f}, y = {y:.4f}")
```

This code ensures high accuracy at each iteration. Although the RK4 algorithm involves more computations than Euler's method, it yields results that are significantly closer to the exact solution.

Program Execution

The functions can be executed with the following parameters:

```
x0 = 0    # Initial x-value
y0 = 1    # Initial y-value
h = 0.1   # Step size
n = 10    # Number of iterations
euler(x0, y0, h, n)
print("\n")
rk4(x0, y0, h, n)
```

The output shows that the Euler method accumulates error more rapidly, while the RK4 method maintains high accuracy and closely approximates the true solution.

In many cases, analytical methods for solving differential equations are not applicable, especially when modeling complex or real-world systems. In such situations, numerical methods—particularly the Euler and fourth-order Runge-Kutta (RK4) methods—play a crucial role.

This article has presented the theoretical foundations of these methods and demonstrated their practical implementation using the Python programming language. The Euler method is characterized by its simplicity but suffers from low accuracy. In contrast, the Runge-Kutta method, though more computationally intensive, provides significantly higher precision and stability.

Therefore, in practical computations or scientific simulations, the RK4 method is generally preferred for its accuracy, while the Euler method serves as a useful tool for educational purposes and initial approximations.

REFERENCES:

1. Boyce, W. E., & DiPrima, R. C. (2017). Elementary Differential Equations and Boundary Value Problems. Wiley.
2. Burden, R. L., & Faires, J. D. (2011). Numerical Analysis (9th ed.). Brooks/Cole.
3. Atkinson, K. E. (1989). An Introduction to Numerical Analysis (2nd ed.). Wiley.
4. Chapra, S. C., & Canale, R. P. (2015). Numerical Methods for Engineers (7th ed.).McGraw-Hill Education.
5. Butcher, J. C. (2016). Numerical Methods for Ordinary Differential Equations (3rd ed.). Wiley.
6. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). Numerical Recipes: The Art of Scientific Computing (3rd ed.). Cambridge University Press.
7. Rao, S. S. (2011). Engineering Optimization: Theory and Practice (4th ed.). Wiley.