1 Численное решение задачи Коши

1.1 Постановка задачи

Необходимо решить задачу Коши ($x \in [0;2]$):

$$\begin{cases} y' = \cos\left(\dfrac{5}{2}x - \dfrac{1}{2}y\right) \\ y(0) = 0 \end{cases}$$

Решение выполнить тремя методами: Эйлера, улучшенный метод Эйлера и Рунге-Кутта 4-го порядка. Сравнить результаты.

1.2 Решение

1.2.1 Аналитическое решение

$$y' = \cos\left(\frac{5}{2}x - \frac{1}{2}y\right)$$

Характеристика: линейное неоднородное уравнение.

Решение:

$$\arctan\left(\frac{\sqrt{3}}{2}\tan\frac{5x-y}{4}\right) = \frac{\sqrt{6}}{2}x + C$$

Решение задачи Коши:

$$y = 5x - 4\arctan\left(\frac{2}{\sqrt{3}}\tan\frac{\sqrt{6}}{2}x\right)$$

1.2.2 Метод Эйлера

Расчетная формула метода Эйлера:

$$y_{i+1} = y_i + h * f(x_i, y_i),\ \text{где } i = 0, 1...n; y' = f(x,y) \quad (1)$$

Решение реализовано с помощью программы *Python*. Результаты программы представлены в таблице, где y_i^* – точное решение, а y_i – приближенное:

Euler

x_i	y*_i	y_i	|y*_i - y_i|
0.0000	0.0000	0.0000	0.0000
0.2000	0.1947	0.2000	0.0053
0.4000	0.3574	0.3842	0.0268
0.6000	0.4573	0.5224	0.0651
0.8000	0.4684	0.5876	0.1192
1.0000	0.3802	0.5606	0.1804
1.2000	0.2113	0.4397	0.2285
1.4000	12.5809	0.2527	12.3282
1.6000	12.4231	0.0580	12.3651
1.8000	12.3519	-0.0770	12.4289
2.0000	12.3799	-0.1116	12.4915

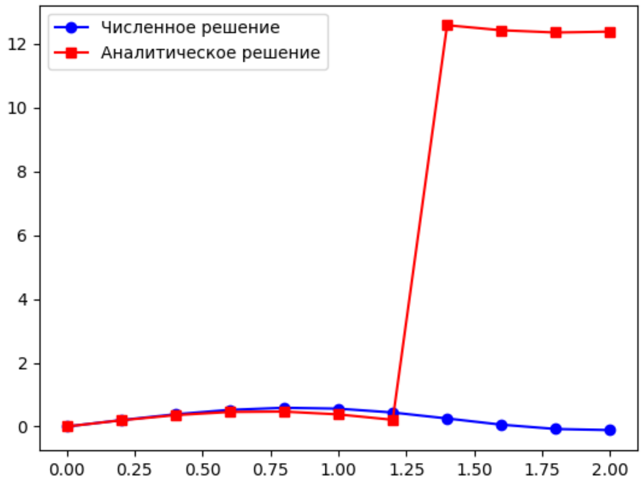


Рис. 1: График точного решения и решения методом Эйлера

1.2.3 Улучшенный метод Эйлера

Расчетная формула улучшенного метода Эйлера:

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y*_{i+1})), \text{ где } i = 0, 1...n;$$
$$y*_{i+1} \text{ – значение в классическом методе Эйлера} \quad (2)$$

Решение реализовано с помощью программы *Python*. Результаты программы представлены в таблице, где y_i^* – точное решение, а y_i – приближенное:

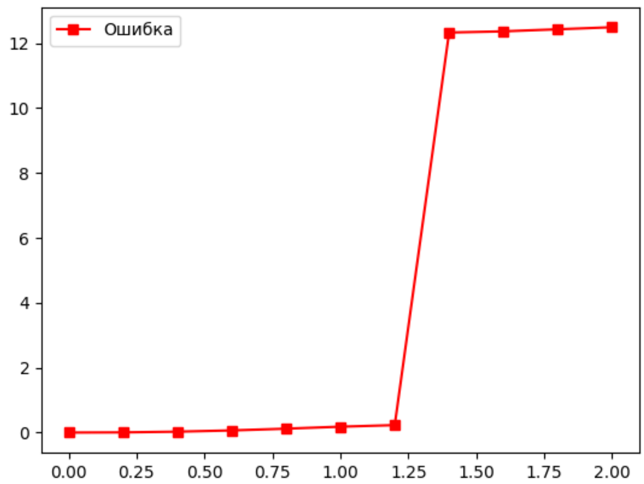


Рис. 2: График ошибки

Enhanced Euler

x_i	y*_i	y_i	|y*_i - y_i|
0.0000	0.0000	0.0000	0.0000
0.2000	0.1947	0.1921	0.0026
0.4000	0.3574	0.3532	0.0042
0.6000	0.4573	0.4537	0.0035
0.8000	0.4684	0.4696	0.0011
1.0000	0.3802	0.3898	0.0096
1.2000	0.2113	0.2293	0.0180
1.4000	12.5809	0.0352	12.5457
1.6000	12.4231	-0.1266	12.5497
1.8000	12.3519	-0.2043	12.5562
2.0000	12.3799	-0.1817	12.5615

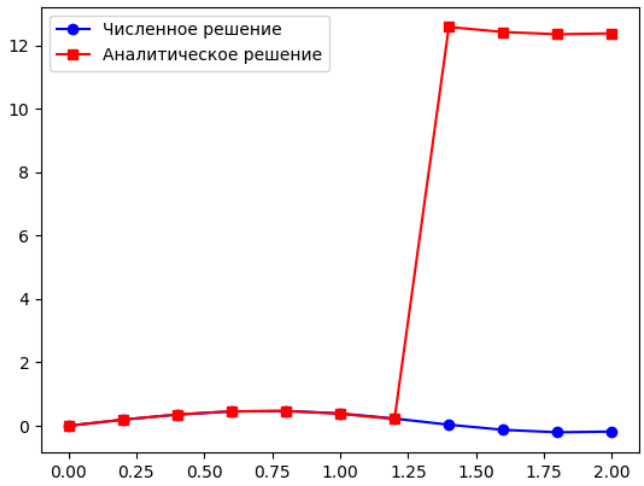


Рис. 3: График точного решения и решения улучшенным методом Эйлера

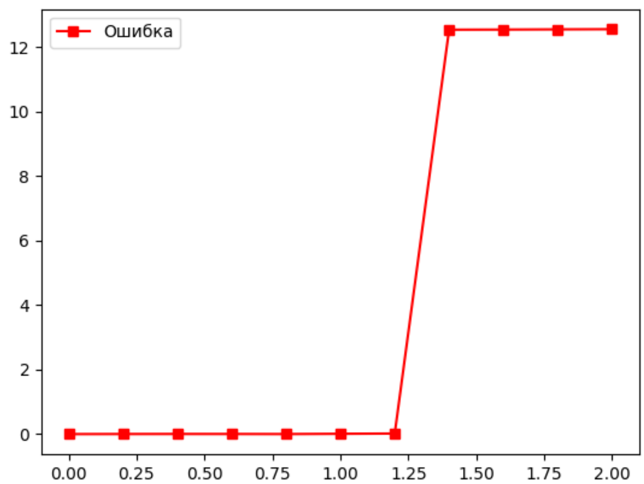


Рис. 4: График ошибки

1.2.4 Метод Рунге-Кутта 4-го порядка

Расчетная формула метода Рунге-Кутта 4-го порядка:

$$\begin{aligned}
&y_{i+1} = y_i + (k_1 + 2*k_2 + 2*k_3 + k_4)/6 \\
&k_1 = h*f(x_i, y_i) \\
&k_2 = h*f(x_i + h/2, y_i + h*k_1/2) \\
&k_3 = h*f(x_i + h/2, y_i + h*k_2/2) \\
&k_4 = h*f(x_i + h, y_i + h*k_3)
\end{aligned} \tag{3}$$

Решение реализовано с помощью программы *Python*. Результаты программы представлены в таблице, где y_i^* – точное решение, а y_i – приближенное:

```
rk_4
|   x_i    ||   y*_i   ||   y_i    ||y*_i - y_i|
|  0.0000  ||  0.0000  ||  0.0000  ||  0.0000  |
|  0.2000  ||  0.1947  ||  0.1947  ||  0.0000  |
|  0.4000  ||  0.3574  ||  0.3574  ||  0.0000  |
|  0.6000  ||  0.4573  ||  0.4573  ||  0.0000  |
|  0.8000  ||  0.4684  ||  0.4684  ||  0.0000  |
|  1.0000  ||  0.3802  ||  0.3802  ||  0.0000  |
|  1.2000  ||  0.2113  ||  0.2113  ||  0.0000  |
|  1.4000  || 12.5809  ||  0.0145  || 12.5663  |
|  1.6000  || 12.4231  || -0.1432  || 12.5663  |
|  1.8000  || 12.3519  || -0.2145  || 12.5663  |
|  2.0000  || 12.3799  || -0.1865  || 12.5663  |
```

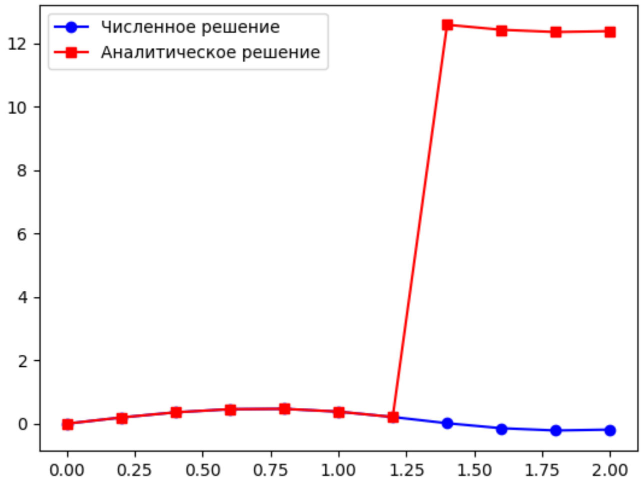


Рис. 5: График точного решения и решения методом Рунге-Кутта 4-го порядка

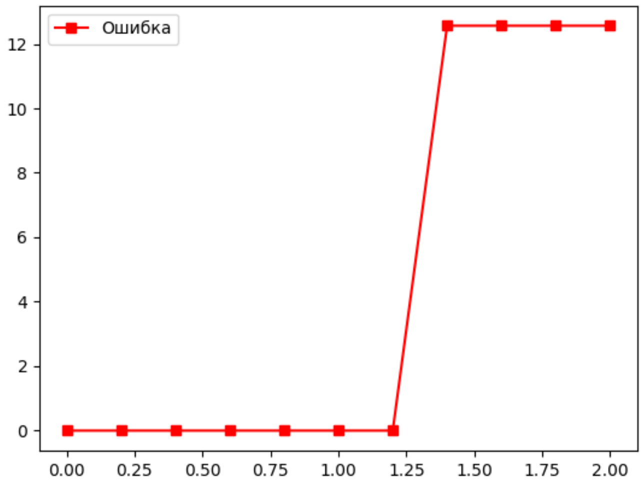
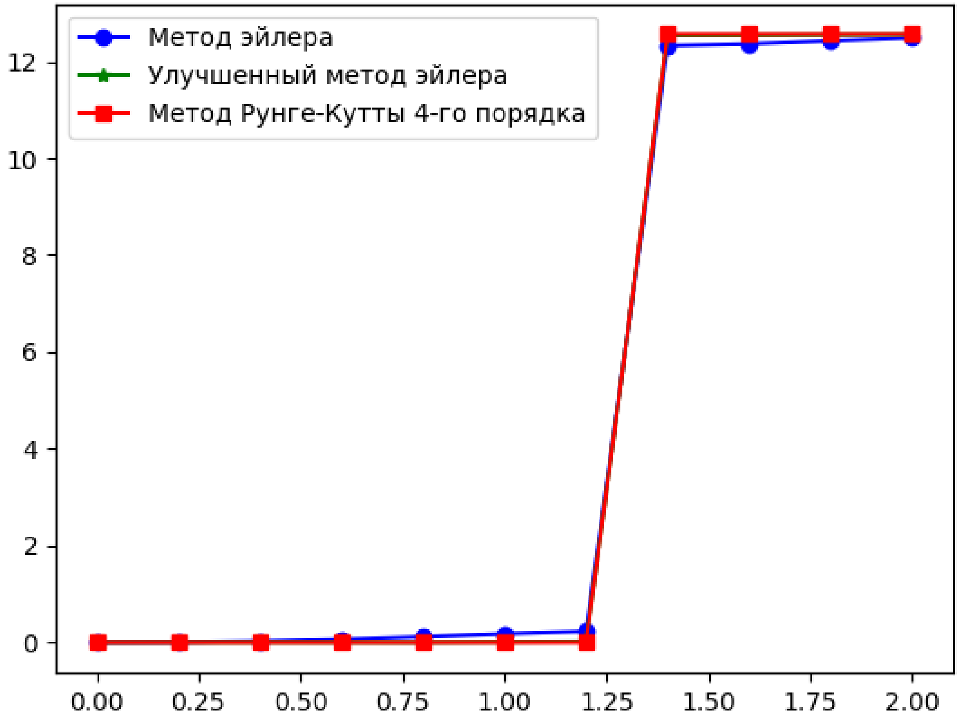


Рис. 6: График ошибки

2 Заключение



На интервале [0, 1] метод Рунге-Кутты 4-го порядка показал самую лучшую точность. Далее идет модифицированный метод Эйлера. На последнем месте по точности оказался метод Эйлера. На интервале [1, 2] аналитическое решение имеет складку, ни один из методов не смог ее воспроизвести.

3 Код

```
import numpy as np
from typing import Callable, Tuple
import matplotlib.pyplot as plt
from math import cos


def f(x, y):
    return cos(5*x/2 - y/2)


def orig(x):
    return 5*x - 4*np.arctan(np.sqrt(2/3)*np.tan(np.sqrt(3/2)*x))


def euler(f: Callable, point: Tuple[int, int], start: int, finish: int, n: int = 10):
    h = (finish-start)/n
    res = [point]
    for i in range(n):
        res.append((res[i][0]+h, res[i][1]+h*f(res[i][0], res[i][1])))
    return res


def enhanced_euler(f: Callable, point: Tuple[int, int], start: int, finish: int, n: int = 10):
    res = euler(f, point, start, finish, n)
    h = (finish-start)/n
    for i in range(1, len(res)):
        res[i] = (res[i][0], res[i-1][1] + h*(f(res[i-1][0], res[i-1][1])+f(res[i][0], res[i][1]))/2)
    return res


def rk4(f: Callable, point: Tuple[int, int], start: int, finish: int, n: int = 10):
    res = [point]
    h = (finish - start) / n
    for i in range(n):
        k1 = f(res[i][0], res[i][1])
        k2 = f(res[i][0] + h/2, res[i][1] + h*k1/2)
        k3 = f(res[i][0] + h/2, res[i][1] + h*k2/2)
        k4 = f(res[i][0] + h, res[i][1] + h*k3)
        res.append((res[i][0] + h, res[i][1] + h*(k1+2*k2+2*k3+k4)/6))
    return res


def print_table(*arrays):
    str_f = "|{value:^10}|"
```

```
    num_f = "|{value:^10.4f}|"
    print(str_f.format(value="x_i") + str_f.format(value="y*_i")
       +
          str_f.format(value="y_i") + str_f.format(value="y*_i -
             y_i"))
    for i in zip(*arrays):
        for j in i:
            print(num_f.format(value=j), end='')
        print()


if __name__ == '__main__':
    euler_res = euler(f, (0, 0), 0, 2)
    x = [x[0] for x in euler_res]
    y_euler = [y[1] for y in euler_res]
    y_anal = [orig(i) for i in x]
    euler_error = [np.abs(y_anal[i]-y_euler[i]) for i in range(
       len(x))]
    plt.plot(x, y_euler, '-o', label='
                     ', color='blue')
    plt.plot(x, y_anal, '-s', label='
                     ', color='red')
    plt.legend()
    plt.savefig("euler.png")
    plt.cla()
    plt.plot(x, euler_error, '-s', label='            ', color='
       red')
    plt.legend()
    plt.savefig("euler-error.png")
    plt.cla()
    print("Euler")
    print_table(x, y_anal, y_euler, euler_error)
    print()

    enhanced_euler_res = enhanced_euler(f, (0, 0), 0, 2)
    y_enhanced_euler = [y[1] for y in enhanced_euler_res]
    enhanced_euler_error = [np.abs(y_anal[i] - y_enhanced_euler[i
       ]) for i in range(len(x))]
    plt.plot(x, y_enhanced_euler, '-o', label='
                      ', color='blue')
    plt.plot(x, y_anal, '-s', label='
                     ', color='red')
    plt.legend()
    plt.savefig("euler-enhanced.png")
    plt.cla()
    plt.plot(x, enhanced_euler_error, '-s', label='            ',
        color='red')
    plt.legend()
    plt.savefig("euler-enhanced-error.png")
    plt.cla()
    print("Enhanced Euler")
```

```
    print_table(x, y_anal, y_enhanced_euler, enhanced_euler_error
        )
    print()

    rk4_res = rk4(f, (0, 0), 0, 2)
    y_rk4 = [y[1] for y in rk4_res]
    rk4_error = [np.abs(y_anal[i] - y_rk4[i]) for i in range(len(
        x))]
    plt.plot(x, y_rk4, '-o', label='
                             ', color='blue')
    plt.plot(x, y_anal, '-s', label='
                             ', color='red')
    plt.legend()
    plt.savefig("rk4.png")
    plt.cla()
    plt.plot(x, rk4_error, '-s', label='             ', color='red
        ')
    plt.legend()
    plt.savefig("rk4-error.png")
    plt.cla()

    plt.plot(x, euler_error, '-o', label='
        ', color='blue')
    plt.plot(x, enhanced_euler_error, '-*', label='
                                                              ',
             color='green')
    plt.plot(x, rk4_error, '-s', label='                          -
                  4-                     ',
             color='red')
    plt.legend()
    plt.savefig("error-union.png")
    plt.cla()
    print("rk_4")
    print_table(x, y_anal, y_rk4, rk4_error)
    print()
```