МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №6

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №25

Студент:
Хоанг Ван Куан
Группа P3266

Преподаватель:
Машина Екатерина Александровна

Санкт-Петербург, 2024

1. Цель работы

Цель лабораторной работы: решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

2. Порядок выполнения работы

Программная реализация

- В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции
- Пользователь выбирает ОДУ вида $y' = f(x, y)$ (не менее трех уравнений), из тех, которые предлагает программа
- Предусмотреть ввод исходных данных с клавиатуры: начальные условия $y_0 = y(y_0)$, интервал дифференцирования $[x_0, x_n]$, шаг h, точность ε
- Для исследования использовать одношаговые методы и многошаговые методы (см. табл.1)
- Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям, для всех методов, реализуемых в программе
- Для оценки точности одношаговых методов использовать правило Рунге
- Для оценки точности многошаговых методов использовать точное решение задачи: $|\varepsilon = max_{0\le i\le n}|y_{i\text{точн}} - y_i||$
- Построить графики точного решения и полученного приближенного решения (разными цветами)
- Программа должна быть протестирована при различных наборах данных, в том числе и некорректных.
- Проанализировать результаты работы программы.

3. Листинг программы

```
import math
# порядок точности: 1
def EulerMethod(f, a, b, y0, h, epsilon):
    check = True
    while(check):
        res1 = Euler(f, a, b, y0, h)
        res2 = Euler(f, a, b, y0, h/2)
        for i in range(len(res1) - 1, -1, -1):
            if(abs(res1[i][1] - res2[i * 2][1]) >= epsilon):
                h /= 4
                break
            else: check = False
    return Euler(f, a, b, y0, h)

def Euler(f, a, b, y0, h):
    res = [(a, y0)]
    n = int((b - a) / h)
    for i in range(1, n + 1):
```

```
        res.append((res[i - 1][0] + h, res[i - 1][1] + h * f(res[i - 1][0], res[i
- 1][1])))
    return res

# порядок точности: 4
def RungeKuttaMethod(f, a, b, y0, h, epsilon):
    check = True
    while(check):
        res1 = RungeKutta(f, a, b, y0, h)
        res2 = RungeKutta(f, a, b, y0, h/2)
        for i in range(len(res1) - 1, -1, -1):
            if(abs(res1[i][1] - res2[i * 2][1])/15 > epsilon):
                h /= 4
                break
            else: check = False
    return RungeKutta(f, a, b, y0, h)

def RungeKutta(f, a, b, y0, h):
    res = [(a, y0)]
    n = int((b - a) / h)
    for i in range(1, n + 1):
        k1 = h * f(res[i - 1][0], res[i-1][1])
        k2 = h * f(res[i - 1][0] + h/2, res[i-1][1] + k1/2)
        k3 = h * f(res[i - 1][0] + h/2, res[i-1][1] + k2/2)
        k4 = h * f(res[i - 1][0] + h, res[i-1][1] + k3)
        res.append((res[i - 1][0] + h, res[i - 1][1] + (k1 + 2*k2 + 2*k3 +
k4)/6))
    return res

# порядок точности: 4
def MilnaMethod(f, a, b, y0, h, epsilon):
    n = int((b - a) / h)
    b0 = min(b, a + 3.1*h) # 0.1 для погрешности в Python
    res = RungeKuttaMethod(f, a, b0, y0, h, epsilon)

    for i in range(4, n + 1):
        xi = res[i - 1][0] + h
        _Yprog = res[i - 4][1] + 4*h*(2*f(res[i - 3][0], res[i - 3][1]) - f(res[i
- 2][0], res[i - 2][1]) + 2*f(res[i - 1][0], res[i - 1][1])) / 3
        _Fprog = f(xi, _Yprog)
        _Ykorr = res[i - 2][1] + h*(f(res[i - 2][0], res[i - 2][1]) + 4*f(res[i -
1][0], res[i - 1][1]) + _Fprog) / 3
        res.append((xi, _Ykorr))

    return res
```

4. Результаты выполнения программы

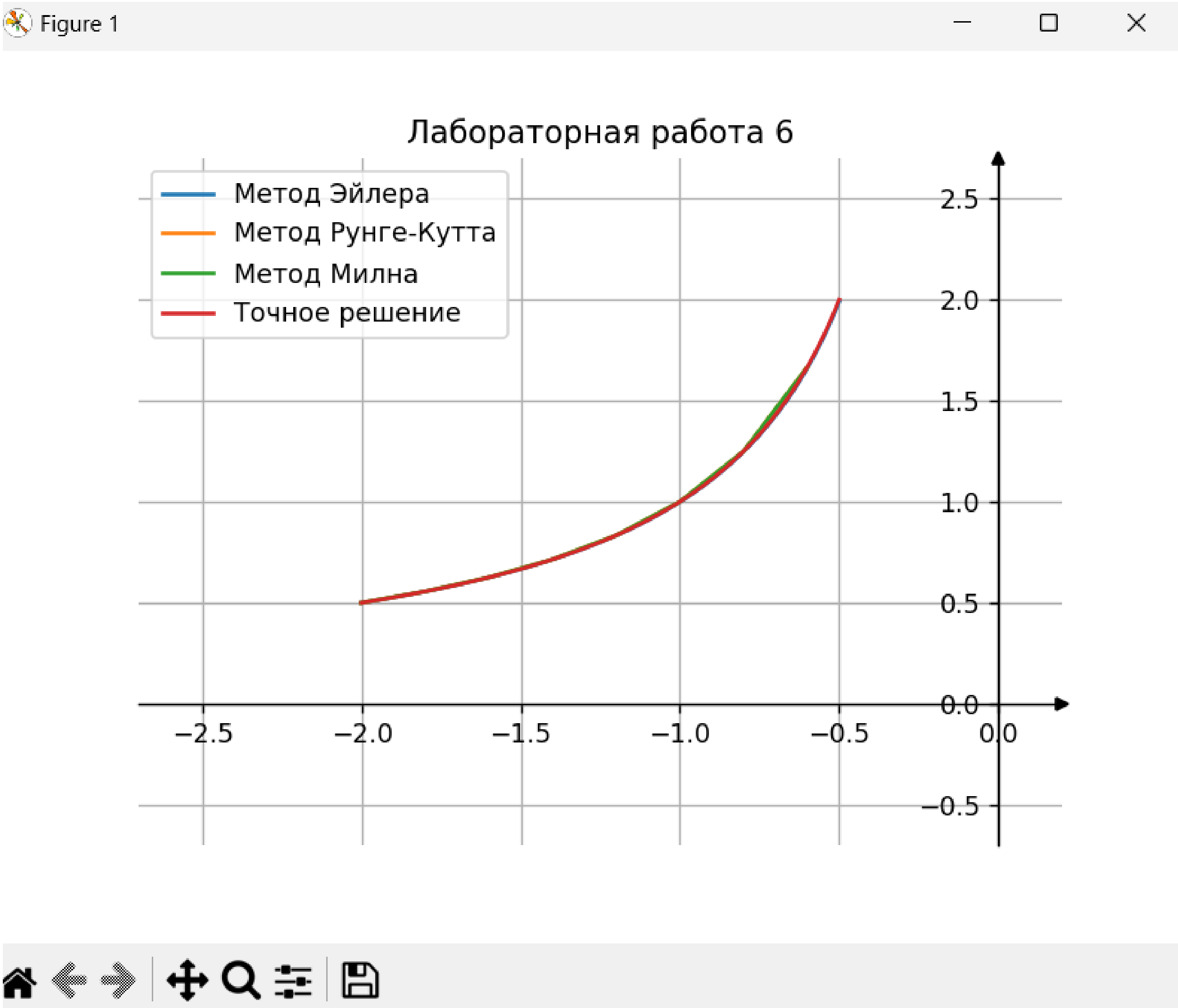
Figure 1
Лабораторная работа 6
Метод Эйлера
Метод Рунге-Кутта
Метод Милна
Точное решение
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−2.5
−2.0
−1.5
−1.0
−0.5
0.0

```
ЛАБОРАТОРНАЯ РАБОТА
--------------------------------------------
Выберите ОДУ
 1) y' = y + (1 + x)y²    на [-2 ; -0.5]    при y(-2) = 1/2
 2) y' = x² - 2y          на [0; 1]         при y(0) = 1
ОДУ: 1
Введите шаг точе h: 0.2
Введите точность: 0.01
Результаты вычисления.
+---------+-------------+---------------+
|    x    | Метод Эйлера | Точное решение |
+---------+-------------+---------------+
|   -2    |     0.5     |      0.5      |
| -1.99687 |   0.50078   |    0.50078    |
| -1.99375 |   0.50156   |    0.50157    |
| -1.99062 |   0.50235   |    0.50235    |
| -1.9875  |   0.50314   |    0.50314    |
| -1.98437 |   0.50393   |    0.50394    |
| -1.98125 |   0.50472   |    0.50473    |
| -1.97812 |   0.50552   |    0.50553    |
|  -1.975  |   0.50632   |    0.50633    |
| -1.97187 |   0.50712   |    0.50713    |
| -1.96875 |   0.50792   |    0.50794    |
| -1.96562 |   0.50873   |    0.50874    |
| -1.9625  |   0.50954   |    0.50955    |
| -1.95937 |   0.51035   |    0.51037    |
| -1.95625 |   0.51116   |    0.51118    |
| -1.28125 |   0.77987   |    0.78049    |
| -1.27812 |   0.78177   |     0.7824    |
|  -1.275  |   0.78368   |    0.78431    |
| -1.27187 |    0.7856   |    0.78624    |
| -1.26875 |   0.78753   |    0.78818    |
| -1.26562 |   0.78947   |    0.79012    |
| -1.2625  |   0.79142   |    0.79208    |
| -1.25937 |   0.79338   |    0.79404    |
| -1.25625 |   0.79535   |    0.79602    |
| -1.25312
+------+------------------+---------------+
|  x   | Метод Рунге-Кутта | Точное решение |
+------+------------------+---------------+
|  -2  |       0.5        |      0.5      |
| -1.8 |     0.55556      |    0.55556    |
| -1.6 |      0.625       |     0.625     |
| -1.4 |     0.71428      |    0.71429    |
| -1.2 |     0.83333      |    0.83333    |
| -1.0 |       1.0        |      1.0      |
| -0.8 |     1.24999      |     1.25      |
| -0.6 |     1.66664      |    1.66667    |
+------+------------------+---------------+
+------+------------+---------------+
|  x   | Метод Милна | Точное решение |
+------+------------+---------------+
|  -2  |    0.5     |      0.5      |
| -1.8 |  0.55556   |    0.55556    |
| -1.6 |   0.625    |     0.625     |
| -1.4 |  0.71428   |    0.71429    |
| -1.2 |  0.83336   |    0.83333    |
| -1.0 |  1.00004   |      1.0      |
| -0.8 |  1.25002   |     1.25      |
| -0.6 |  1.66619   |    1.66667    |
+------+------------+---------------+
```