ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет Программной Инженерии и Компьютерной Техники

Дисциплина:

«*Вычислительная математика*»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«Численное решение обыкновенных дифференциальных уравнений»

Вариант 3

Выполнил:

Студент гр. P32151

Горинов Даниил Андреевич

Проверил:

Машина Екатерина Алексеевна

Санкт-Петербург

2023г.

<u>Цель лабораторной работы:</u>

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

<u>Порядок выполнения лабораторной работы:</u>

1. В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции;
2. Пользователь выбирает ОДУ вида $y' = f(x, y)$ (не менее трех уравнений), из тех, которые предлагает программа;
3. Предусмотреть ввод исходных данных с клавиатуры: начальные условия $y_0 = y(x_0)$, интервал дифференцирования $[x_0, x_n]$, шаг h, точность ε;
4. Для исследования использовать одношаговые методы и многошаговые методы;
5. Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям, для всех методов, реализуемых в программе;
6. Для оценки точности одношаговых методов использовать правило Рунге: $R = \frac{y^h - y^{\frac{h}{2}}}{2^p - 1}$;
7. Для оценки точности многошаговых методов использовать точное решение задачи: $\varepsilon = max_{0 \le \mathrm{i} \le \mathrm{n}} |\mathrm{y}_{\mathrm{iточн}} - y_i|$;
8. Построить графики точного решения и полученного приближенного решения (разными цветами);
9. Программа должна быть протестирована при различных наборах данных, в том числе и некорректных.
10. Проанализировать результаты работы программы.

<u>Рабочие формулы методов:</u>

Метод Рунге-Кутта:

$y_{i+1} = y_i + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)$, где

$k_1 = h \cdot f(x_i, y_i)$

$k_2 = h \cdot f\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$

$k_3 = h \cdot f\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$

$k_4 = h \cdot f(x_i + h, y_i + k_3)$

Метод Адамса:

$y_{i+1} = y_i + hf_1 + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_1 + \frac{3h^4}{8}\Delta^3 f_i$,

Где

$$\Delta f_i = f_i + f_{i-1}$$
$$\Delta^2 f_i = f_i - 2f_{i-1} + f_{i-2}$$
$$\Delta^3 f_i = f_i - 3f_{i-1} + 3f_{i-3}$$

<u>Листинг программы:</u>

```
def euler_method(f, a, b, y0, h):
    dots = [(a, y0)]
    n = int((b - a) / h)
    for i in range(1, n + 1):
        dots.append((dots[i - 1][0] + h, dots[i - 1][1] + h * f(*dots[i - 1])))
    return dots


def fourth_order_runge_kutta(f, a, b, y0, h):
    dots = [(a, y0)]
    n = int((b - a) / h) + 1
    for i in range(1, n + 1):
        x_prev, y_prev = dots[i - 1]
        k1 = h * f(x_prev, y_prev)
        k2 = h * f(x_prev + h / 2, y_prev + k1 / 2)
        k3 = h * f(x_prev + h / 2, y_prev + k2 / 2)
        k4 = h * f(x_prev + h, y_prev + k3)
        x_cur = x_prev + h
        y_cur = y_prev + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        dots.append((x_cur, y_cur))
    return dots


def milna_method(f, a, b, y0, h):
    dots = [(a, y0)]
    fun_t = [f(a, y0)]
    n = int((b - a) / h) + 1
    for i in range(1, 4):
        x_prev, y_prev = dots[i - 1]
        k1 = h * f(x_prev, y_prev)
        k2 = h * f(x_prev + h / 2, y_prev + k1 / 2)
        k3 = h * f(x_prev + h / 2, y_prev + k2 / 2)
        k4 = h * f(x_prev + h, y_prev + k3)
        x_cur = x_prev + h
        y_cur = y_prev + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        dots.append((x_cur, y_cur))
        fun_t.append(f(x_cur, y_cur))
    for i in range(4, n):
        x_cur = dots[i - 1][0] + h
        y_pred = dots[i - 4][1] + 4 * h / 3 * (2 * fun_t[i - 3] - fun_t[i - 2] + 2 * fun_t[i - 1])
        fun_t.append(f(x_cur, y_pred))
        y_cor = dots[i - 2][1] + h / 3 * (fun_t[i - 2] + 4 * fun_t[i - 1] + fun_t[i])
        while 0.00001 < abs(y_cor - y_pred) / 29:
            y_pred = y_cor
```

```
            fun_t[i] = f(x_cur, y_pred)
            y_cor = dots[i - 2][1] + h / 3 * (fun_t[i - 2] + 4 * fun_t[i - 1] + fun_t[i])
        dots.append((x_cur, y_cor))
    return dots
```

Результаты выполнения программы:

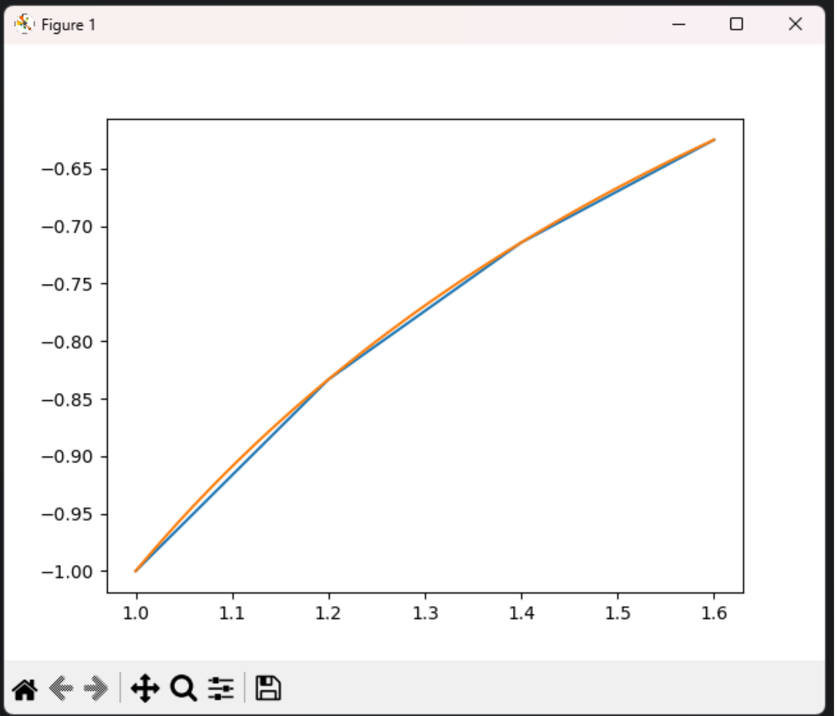
```
Лабораторная работа #6
Горинов Даниил Андреевич
Вариант 3

Выберите метод дифференцирования.
 1 – Метод Эйлера
 2 – Метод Рунге-Кутта 4-го порядка
 3 – Метод Милна
Метод дифференцирования: 2

Выберите задачу.
 1 – y' = y + (1 + x)y²
     на [1; 1,5] при y(1) = -1
 2 – y' = x² - 2y
     на [0; 1] при y(0) = 1
 3 – y' = e²ˣ + y
     на [0; 0.25] при y(0) = 1
Задача: 1

Введите шаг точек: 0.2


Результаты вычисления.
           x            y        acc_y
      1.0000      -1.0000      -1.0000
      1.2000      -0.8334      -0.8333
      1.4000      -0.7144      -0.7143
      1.6000      -0.6251      -0.6250
```



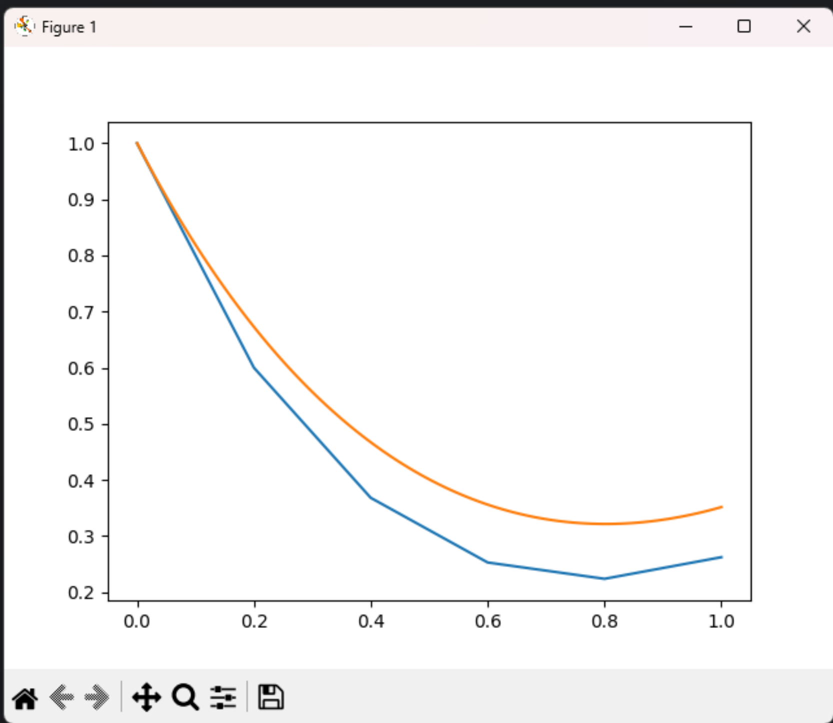

```
Лабораторная работа #6
Горинов Даниил Андреевич
Вариант 3

Выберите метод дифференцирования.
 1 – Метод Эйлера
 2 – Метод Рунге-Кутта 4-го порядка
 3 – Метод Милна
Метод дифференцирования: 1

Выберите задачу.
 1 – y' = y + (1 + x)y²
     на [1; 1,5] при y(1) = -1
 2 – y' = x² - 2y
     на [0; 1] при y(0) = 1
 3 – y' = e²ˣ + y
     на [0; 0.25] при y(0) = 1
Задача: 2

Введите шаг точек: 0.2


Результаты вычисления.
           x            y        acc_y
      0.0000       1.0000       1.0000
      0.2000       0.6000       0.6727
      0.4000       0.3680       0.4670
      0.6000       0.2528       0.3559
      0.8000       0.2237       0.3214
      1.0000       0.2622       0.3515
```



Вывод:

В процессе выполнения данной лабораторной работы я ознакомился с различными подходами к решению задачи Коши и реализовал их с использованием языка программирования Python. Оба метода обладают одинаковой точностью и на небольших интервалах показывают сопоставимые результаты. Однако, при работе с большими значениями переменных они начинают демонстрировать отличающиеся результаты. Мой опыт показывает, что метод Рунге-Кутта является более точным в таких случаях.