**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.

**Цель лабораторной работы:**

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

**Порядок выполнения лабораторной работы:**

1. В программе численные методы решения обыкновенных дифференциальных уравнений (ОДУ) должен быть реализован в виде отдельного класса /метода/функции;
2. Пользователь выбирает ОДУ вида (не менее трех уравнений), из тех, которые предлагает программа;
3. Предусмотреть ввод исходных данных с клавиатуры: начальные условия , интервал дифференцирования , шаг h, точность ;
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: ;
7. Для оценки точности многошаговых методов использовать точное решение задачи: ;
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.

**Рабочие формулы методов:**

Метод Рунге-Кутта:

, где

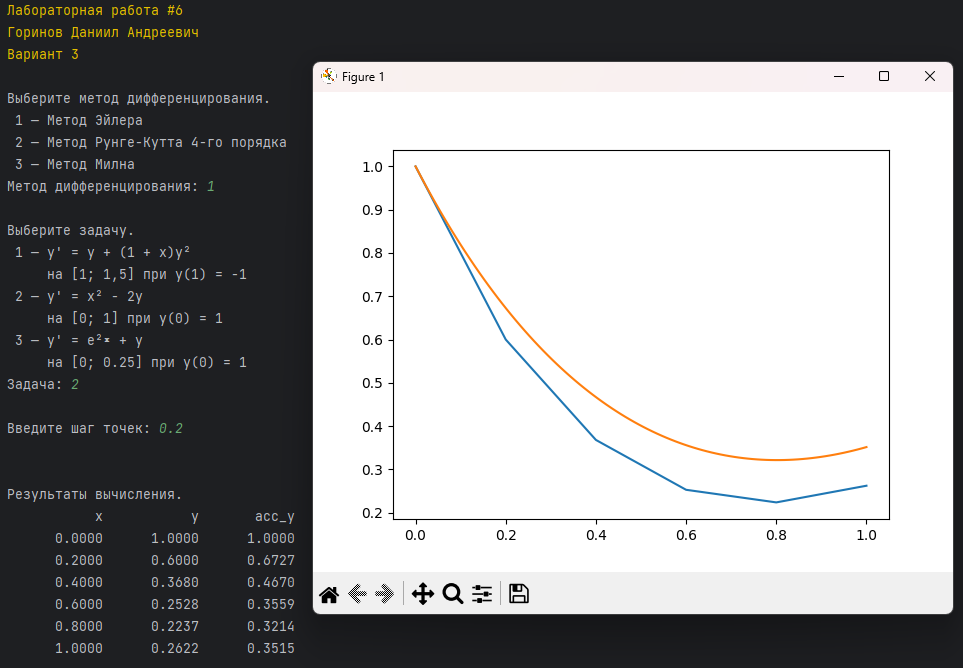
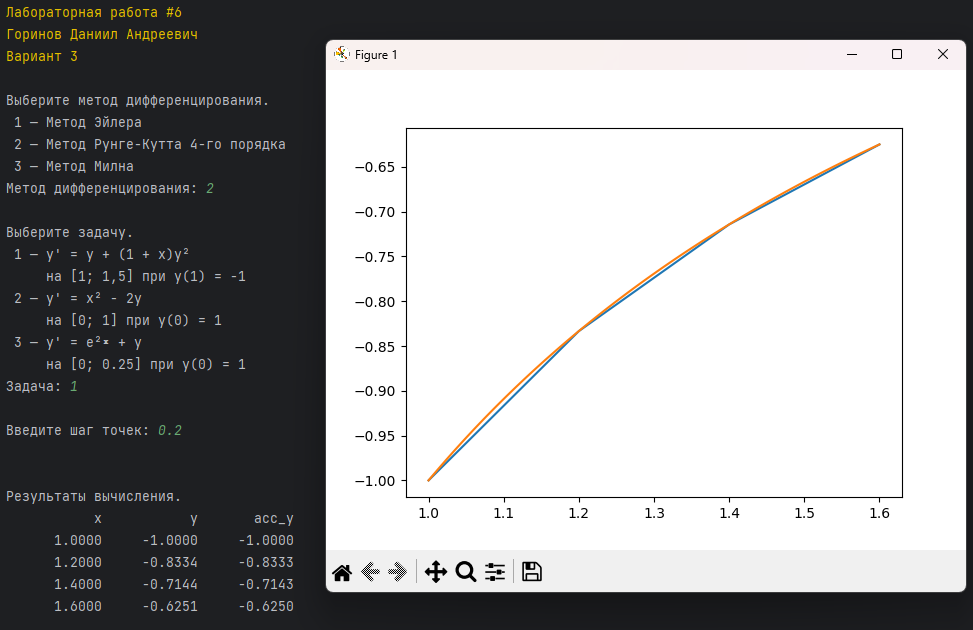
Метод Адамса:

*,*

Где

Листинг программы:

1. def euler\_method(f, a, b, y0, h):
2. dots = [(a, y0)]
3. n = int((b - a) / h)
4. for i in range(1, n + 1):
5. dots.append((dots[i - 1][0] + h, dots[i - 1][1] + h \* f(\*dots[i - 1])))
6. return dots
7. def fourth\_order\_runge\_kutta(f, a, b, y0, h):
8. dots = [(a, y0)]
9. n = int((b - a) / h) + 1
10. for i in range(1, n + 1):
11. x\_prev, y\_prev = dots[i - 1]
12. k1 = h \* f(x\_prev, y\_prev)
13. k2 = h \* f(x\_prev + h / 2, y\_prev + k1 / 2)
14. k3 = h \* f(x\_prev + h / 2, y\_prev + k2 / 2)
15. k4 = h \* f(x\_prev + h, y\_prev + k3)
16. x\_cur = x\_prev + h
17. y\_cur = y\_prev + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6
18. dots.append((x\_cur, y\_cur))
19. return dots
20. def milna\_method(f, a, b, y0, h):
21. dots = [(a, y0)]
22. fun\_t = [f(a, y0)]
23. n = int((b - a) / h) + 1
24. for i in range(1, 4):
25. x\_prev, y\_prev = dots[i - 1]
26. k1 = h \* f(x\_prev, y\_prev)
27. k2 = h \* f(x\_prev + h / 2, y\_prev + k1 / 2)
28. k3 = h \* f(x\_prev + h / 2, y\_prev + k2 / 2)
29. k4 = h \* f(x\_prev + h, y\_prev + k3)
30. x\_cur = x\_prev + h
31. y\_cur = y\_prev + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6
32. dots.append((x\_cur, y\_cur))
33. fun\_t.append(f(x\_cur, y\_cur))
34. for i in range(4, n):
35. x\_cur = dots[i - 1][0] + h
36. y\_pred = dots[i - 4][1] + 4 \* h / 3 \* (2 \* fun\_t[i - 3] - fun\_t[i - 2] + 2 \* fun\_t[i - 1])
37. fun\_t.append(f(x\_cur, y\_pred))
38. y\_cor = dots[i - 2][1] + h / 3 \* (fun\_t[i - 2] + 4 \* fun\_t[i - 1] + fun\_t[i])
39. while 0.00001 < abs(y\_cor - y\_pred) / 29:
40. y\_pred = y\_cor
41. fun\_t[i] = f(x\_cur, y\_pred)
42. y\_cor = dots[i - 2][1] + h / 3 \* (fun\_t[i - 2] + 4 \* fun\_t[i - 1] + fun\_t[i])
43. dots.append((x\_cur, y\_cor))
44. return dots

Результаты выполнения программы:

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.