Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 «Программная инженерия» –

Системное и прикладное программное обеспечение

**Отчёт**

**По лабораторной работе №6**

**По вычислительной математике**

**Вариант: 3**

Выполнил:

студент 2 курса

Батманов Даниил Евгеньевич

Группа: Р3207

Принял:

Рыбаков Степан Дмитриевич

Отчёт принят «\_\_»\_\_\_\_\_2024 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

г. Санкт-Петербург, 2024

**Оглавление**

[Задание 3](#_Toc162833322)

[Рабочие формулы 3](#_Toc162833323)

[Часть I: вычислительная 4](#_Toc162833324)

[Часть II: программная 5](#_Toc162833325)

[Вывод 13](#_Toc162833326)

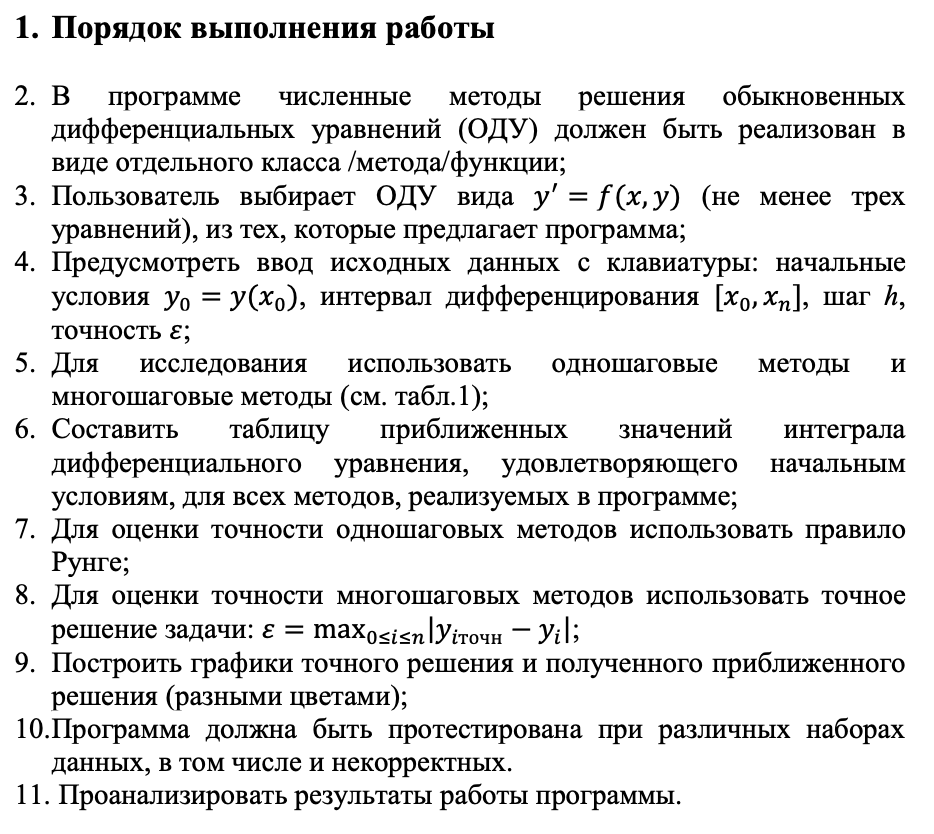
# 

# 

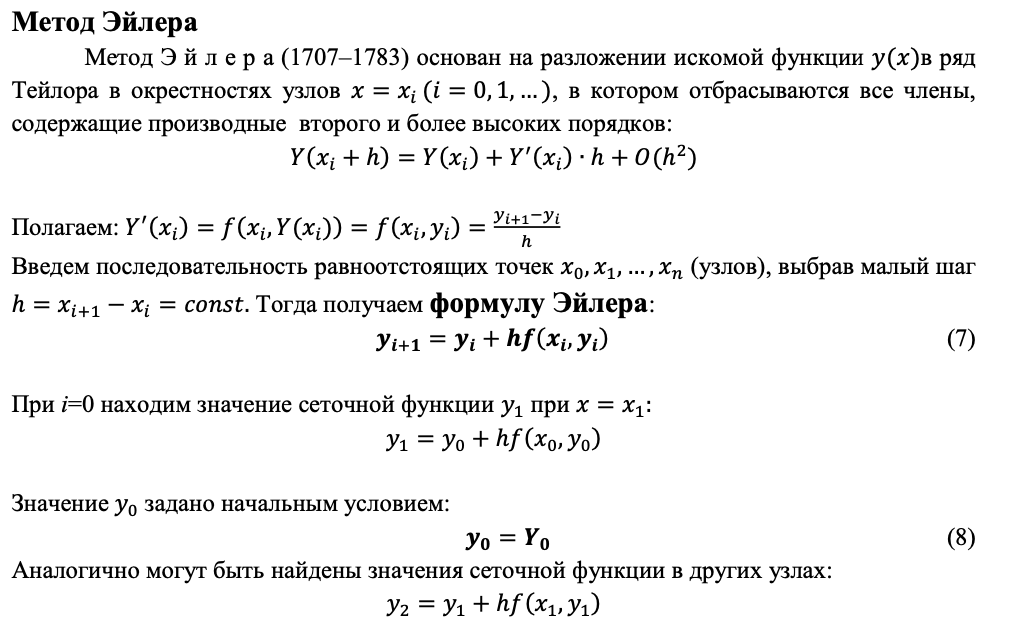
# 

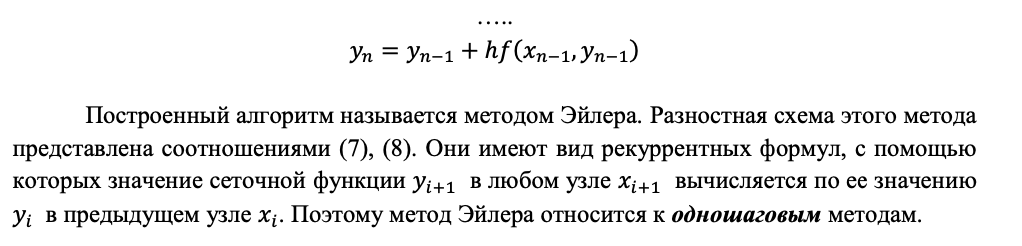
# Задание

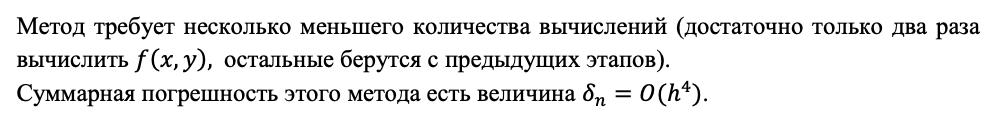
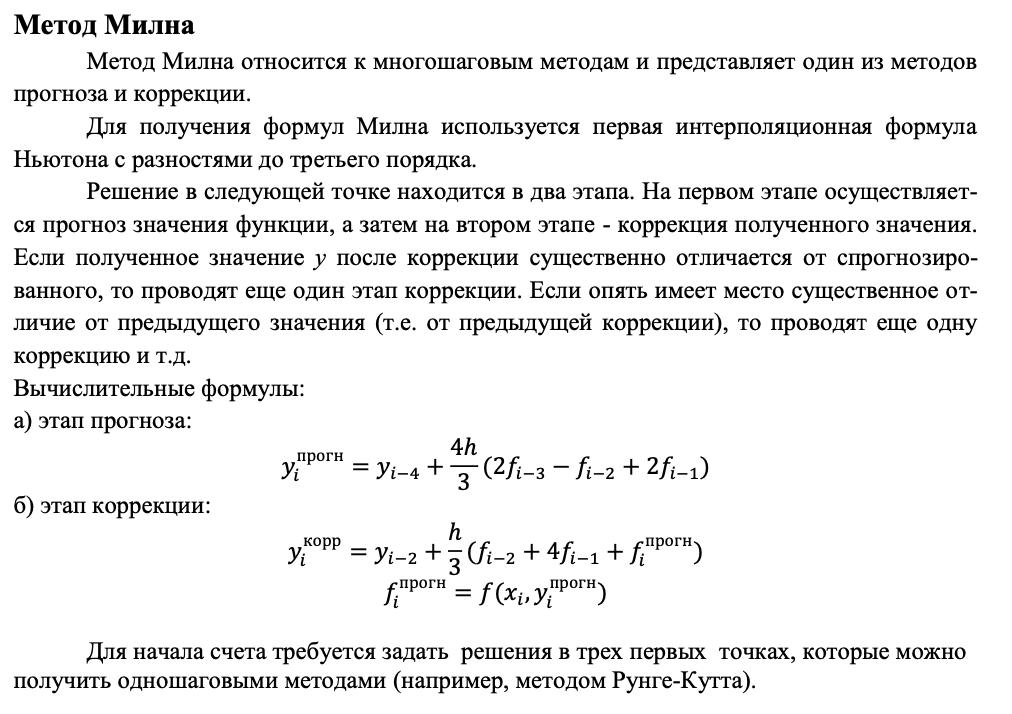
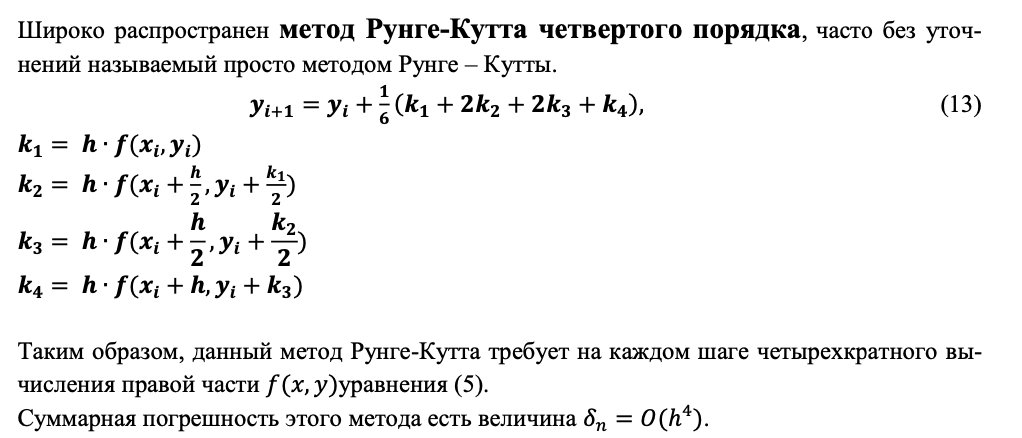
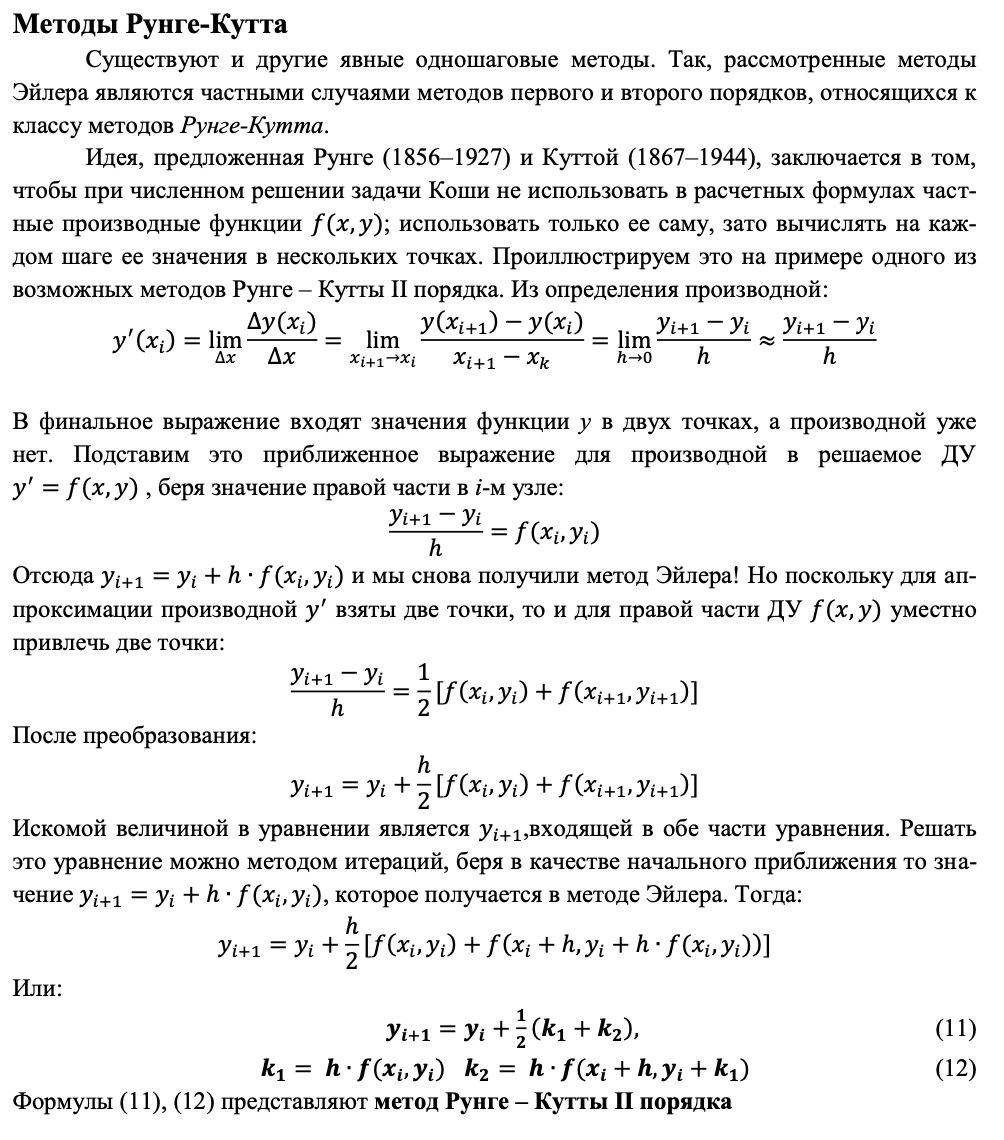
Цель лабораторной работы: решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.



# Рабочие формулы





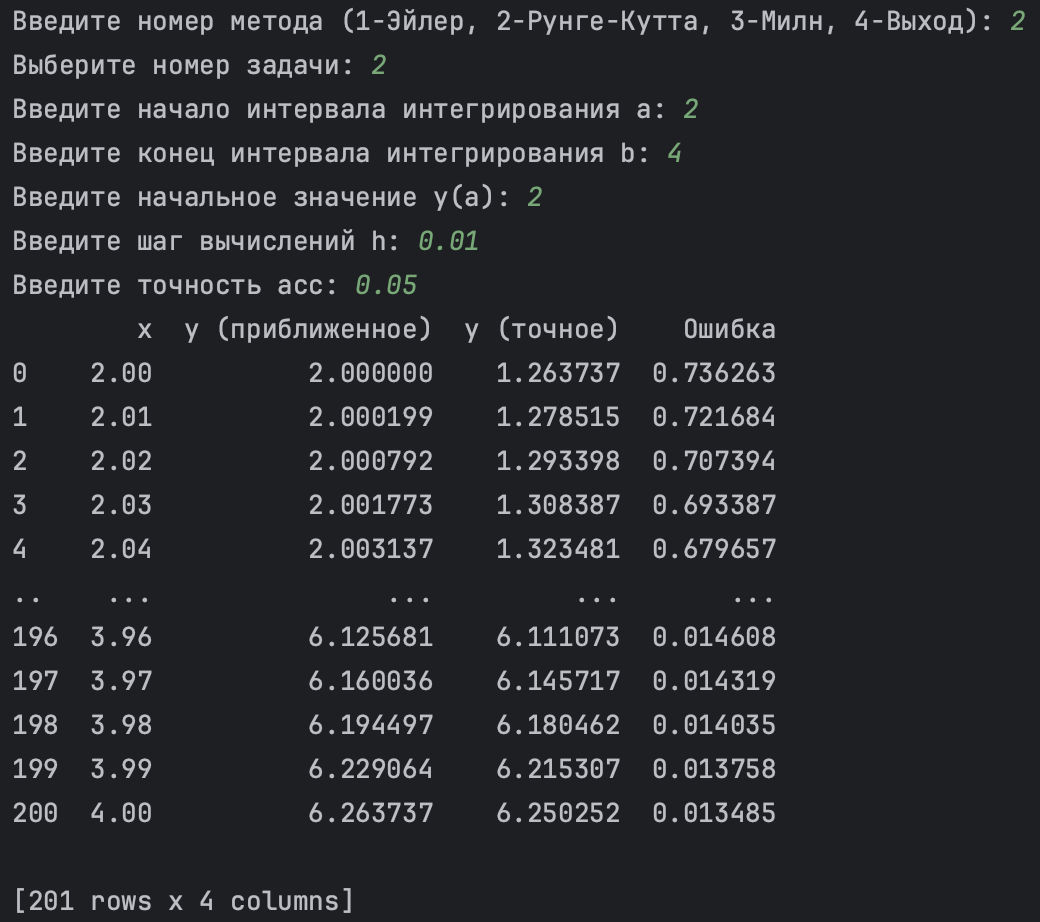


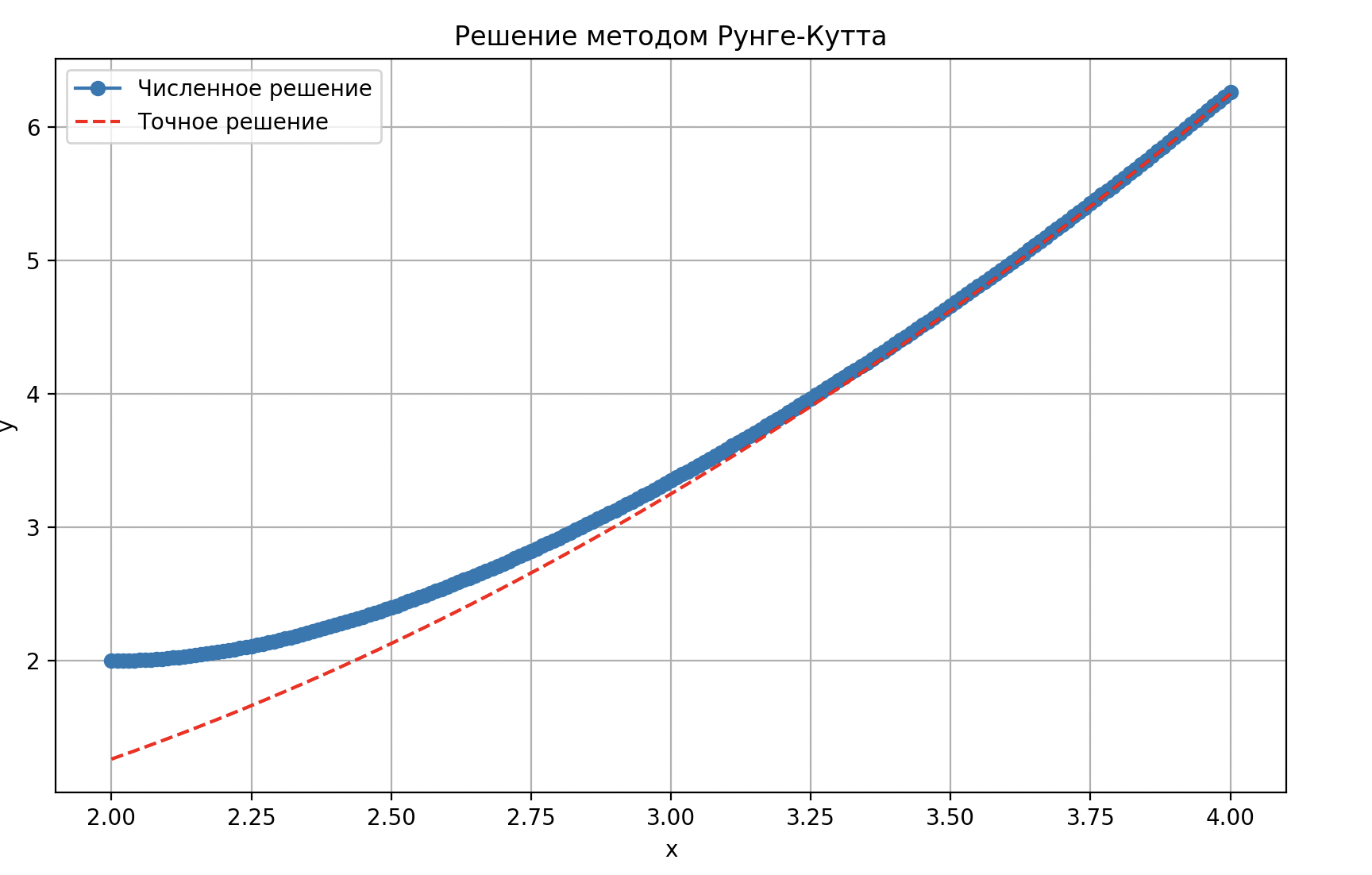
# Часть I: программная

***Исходный код:***

import math  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
  
  
def get\_task(task\_id=None):  
 tasks = {  
 1: {'fun': lambda x, y: y + (1 + x) \* y \*\* 2, 'acc\_fun': lambda x: -1 / x, 'description': "y' = y + (1 + x) \* y^2"},  
 2: {'fun': lambda x, y: x \*\* 2 - 2 \* y, 'acc\_fun': lambda x: 0.25 \* (2 \* x \*\* 2 - 2 \* x + 3 \* math.exp(-2 \* x) + 1), 'description': "y' = x^2 - 2y"},  
 3: {'fun': lambda x, y: math.exp(2 \* x) + y, 'acc\_fun': lambda x: math.exp(2 \* x), 'description': "y' = e^(2x) + y"}  
 }  
 if task\_id is not None:  
 return tasks.get(task\_id)  
 return tasks  
  
  
def list\_tasks():  
 print("Доступные задачи:")  
 tasks = get\_task()  
 for id, task in tasks.items():  
 print(f"{id}: {task['description']}")  
  
class Solver:  
 def \_\_init\_\_(self, method, data, h, acc):  
 self.method = method  
 self.data = data  
 self.h = h  
 self.acc = acc  
 self.n = int((data['b'] - data['a']) / h) + 1  
  
 def euler\_method(self):  
 x, y = self.data['a'], self.data['y0']  
 dots = [(x, y)]  
 for i in range(1, self.n):  
 x\_prev, y\_prev = dots[-1]  
 y += self.h \* self.data['fun'](x\_prev, y\_prev)  
 x += self.h  
 dots.append((x, y))  
 return dots  
  
 def runge\_kutta\_4(self):  
 x, y = self.data['a'], self.data['y0']  
 dots = [(x, y)]  
 for i in range(1, self.n):  
 x\_prev, y\_prev = dots[-1]  
 k1 = self.h \* self.data['fun'](x\_prev, y\_prev)  
 k2 = self.h \* self.data['fun'](x\_prev + self.h / 2, y\_prev + k1 / 2)  
 k3 = self.h \* self.data['fun'](x\_prev + self.h / 2, y\_prev + k2 / 2)  
 k4 = self.h \* self.data['fun'](x\_prev + self.h, y\_prev + k3)  
 y += (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
 x += self.h  
 dots.append((x, y))  
 return dots  
  
 def milne\_method(self):  
 dots = self.runge\_kutta\_4()  
 for i in range(4, self.n):  
 x\_prev, y\_prev = dots[i - 1]  
 y\_pred = dots[i - 4][1] + 4 \* self.h / 3 \* (2 \* self.data['fun'](\*dots[i - 3]) - self.data['fun'](\*dots[i - 2]) + 2 \* self.data['fun'](\*dots[i - 1]))  
 y\_cor = dots[i - 2][1] + self.h / 3 \* (self.data['fun'](\*dots[i - 2]) + 4 \* self.data['fun'](x\_prev, y\_pred) + self.data['fun'](\*dots[i - 1]))  
 dots[i] = (x\_prev + self.h, y\_cor)  
  
 return dots  
  
 def display\_results(self, x\_vals, y\_vals, acc\_y\_vals):  
 data = {"x": x\_vals, "y (приближенное)": y\_vals, "y (точное)": acc\_y\_vals, "Ошибка": [abs(y - y\_true) for y, y\_true in zip(y\_vals, acc\_y\_vals)]}  
 df = pd.DataFrame(data)  
 print(df)  
  
 def plot\_results(self):  
 x\_vals, y\_vals = zip(\*self.dots)  
 acc\_x\_vals = np.linspace(self.data['a'], self.data['b'], len(x\_vals))  
 acc\_y\_vals = [self.data['acc\_fun'](x) for x in acc\_x\_vals]  
 plt.figure(figsize=(10, 6))  
 plt.plot(x\_vals, y\_vals, '-o', label="Численное решение")  
 plt.plot(acc\_x\_vals, acc\_y\_vals, 'r--', label="Точное решение")  
 plt.title(f"Решение методом {'Эйлера' if self.method == 1 else 'Рунге-Кутта' if self.method == 2 else 'Милна'}")  
 plt.xlabel("x")  
 plt.ylabel("y")  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
 self.display\_results(x\_vals, y\_vals, acc\_y\_vals)  
  
 def solve(self):  
 if self.method == 1:  
 self.dots = self.euler\_method()  
 elif self.method == 2:  
 self.dots = self.runge\_kutta\_4()  
 elif self.method == 3:  
 self.dots = self.milne\_method()  
 self.plot\_results()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("Численное дифференцирование")  
 list\_tasks()  
 while True:  
 try:  
 method = int(input("Введите номер метода (1-Эйлер, 2-Рунге-Кутта, 3-Милн, 4-Выход): "))  
 if method == 4:  
 print('ПОКА')  
 break  
 elif method in [1, 2, 3]:  
 task\_id = int(input("Выберите номер задачи: "))  
 a = float(input("Введите начало интервала интегрирования a: "))  
 b = float(input("Введите конец интервала интегрирования b: "))  
 y0 = float(input("Введите начальное значение y(a): "))  
 h = float(input("Введите шаг вычислений h: "))  
 acc = float(input("Введите точность acc: "))  
 task = get\_task(task\_id)  
 if task is None:  
 print("Неверный номер задачи.")  
 continue  
 solver = Solver(method, {'fun': task['fun'], 'acc\_fun': task['acc\_fun'], 'a': a, 'b': b, 'y0': y0}, h, acc)  
 solver.solve()  
 else:  
 print("Неправильный ввод!")  
 except ValueError:  
 print("Неправильный ввод!")

***Вывод программы:***

******



# Вывод

В ходе выполнения данной лабораторной работы я решил задачу Коши для обыкновенных дифференциальных уравнений численными методами.

░░█▀░░░░░░░░░░░▀▀███████░░░░

░░█▌░░░░░░░░░░░░░░░▀██████░░░

░█▌░░░░░░░░░░░░░░░░███████▌░░

░█░░░░░░░░░░░░░░░░░████████░░

▐▌░░░░░░░░░░░░░░░░░▀██████▌░░

░▌▄███▌░░░░▀████▄░░░░▀████▌░░

▐▀▀▄█▄░▌░░░▄██▄▄▄▀░░░░████▄▄░

▐░▀░░═▐░░░░░░══░░▀░░░░▐▀░▄▀▌▌

▐░░░░░▌░░░░░░░░░░░░░░░▀░▀░░▌▌

▐░░░▄▀░░░▀░▌░░░░░░░░░░░░▌█░▌▌

░▌░░▀▀▄▄▀▀▄▌▌░░░░░░░░░░▐░▀▐▐░

░▌░░▌░▄▄▄▄░░░▌░░░░░░░░▐░░▀▐░░

░█░▐▄██████▄░▐░░░░░░░░█▀▄▄▀░░

░▐░▌▌░░░░░░▀▀▄▐░░░░░░█▌░░░░░░

░░█░░▄▀▀▀▀▄░▄═╝▄░░░▄▀░▌░░░░░░

░░░▌▐░░░░░░▌░▀▀░░▄▀░░▐░░░░░░░

░░░▀▄░░░░░░░░░▄▀▀░░░░█░░░░░░░

░░░▄█▄▄▄▄▄▄▄▀▀░░░░░░░▌▌░░░░░░

░░▄▀▌▀▌░░░░░░░░░░░░░▄▀▀▄░░░░░

▄▀░░▌░▀▄░░░░░░░░░░▄▀░░▌░▀▄░░░

░░░░▌█▄▄▀▄░░░░░░▄▀░░░░▌░░░▌▄▄

░░░▄▐██████▄▄░▄▀░░▄▄▄▄▌░░░░▄░

░░▄▌████████▄▄▄███████▌░░░░░▄

░▄▀░██████████████████▌▀▄░░░░

▀░░░█████▀▀░░░▀███████░░░▀▄░░

░░░░▐█▀░░░▐░░░░░▀████▌░░░░▀▄░

░░░░░░▌░░░▐░░░░▐░░▀▀█░░░░░░░▀

░░░░░░▐░░░░▌░░░▐░░░░░▌░░░░░░░

░╔╗║░╔═╗░═╦═░░░░░╔╗░░╔═╗░╦═╗░

░║║║░║░║░░║░░░░░░╠╩╗░╠═╣░║░║░

░║╚╝░╚═╝░░║░░░░░░╚═╝░║░║░╩═╝░

NOT BAD (НЕПЛОХО)