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

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по дисциплине «Вычислительная математика» №6**

Вариант: 4

Преподаватель:   
Рыбаков Степан Дмитриевич

Выполнил: Васильченко Роман

Группа: Р32081

Санкт-Петербург, 2023г

# Цель работы

# Решить задачу Коши для обыкновенных

# дифференциальных уравнений численными методами.

# Листинг программы

# import numpy as np

# import matplotlib.pyplot as plt

# class ODESolver:

# def \_\_init\_\_(self, f, y0, x0, xn, h):

# self.f = f

# self.y0 = y0

# self.x0 = x0

# self.xn = xn

# self.h = h

# def euler(self):

# n = int((self.xn - self.x0) / self.h)

# x = np.linspace(self.x0, self.xn, n+1)

# y = np.zeros(n+1)

# y[0] = self.y0

# for i in range(n):

# y[i+1] = y[i] + self.h \* self.f(x[i], y[i])

# return x, y

# def improved\_euler(self):

# n = int((self.xn - self.x0) / self.h)

# x = np.linspace(self.x0, self.xn, n+1)

# y = np.zeros(n+1)

# y[0] = self.y0

# for i in range(n):

# k1 = self.f(x[i], y[i])

# k2 = self.f(x[i] + self.h, y[i] + self.h \* k1)

# y[i+1] = y[i] + self.h \* (k1 + k2) / 2

# return x, y

# def adams(self, m):

# n = int((self.xn - self.x0) / self.h)

# x = np.linspace(self.x0, self.xn, n+1)

# y = np.zeros(n+1)

# y[0] = self.y0

# for i in range(m):

# y[i+1] = y[i] + self.h \* self.f(x[i], y[i])

# for i in range(m, n):

# y[i+1] = y[i] + self.h \* (55\*self.f(x[i], y[i]) - 59\*self.f(x[i-1], y[i-1]) + 37\*self.f(x[i-2], y[i-2]) - 9\*self.f(x[i-3], y[i-3])) / 24

# return x, y

# def runge\_rule(self, h1, h2, p):

# n1 = int((self.xn - self.x0) / h1)

# n2 = int((self.xn - self.x0) / h2)

# y\_h1 = np.zeros(n1+1)

# y\_h2 = np.zeros(n2+1)

# \_, y\_h1 = self.euler()

# self.h = h2

# \_, y\_h2 = self.euler()

# return np.abs(y\_h1[-1] - y\_h2[-1]) / (2\*\*p - 1)

# def main():

# f1 = lambda x, y: x + y

# f2 = lambda x, y: np.sin(x) - y

# f3 = lambda x, y: np.exp(-x) - y\*\*2

# print("f1 = x + y")

# print("f1 = sin(x) - y")

# print("e^(-x) - y^2")

# func\_input = input("Выберите функцию f1/f2/f3 ")

# func = f1

# 

# match func\_input:

# case "f1":

# func = f1

# case "f2":

# func = f2

# case "f3":

# func = f3

# case \_:

# main()

# return

# solver = ODESolver(func, 1, 0, 1, 0.1)

# x, y = solver.euler()

# plt.plot(x, y, label='Euler')

# x, y = solver.improved\_euler()

# plt.plot(x, y, label='Improved Euler')

# x, y = solver.adams(4)

# plt.plot(x, y, label='Adams')

# epsilon = solver.runge\_rule(0.1, 0.05, 1)

# print(f"Точность метода Эйлера по правилу Рунге: {epsilon}")

# 

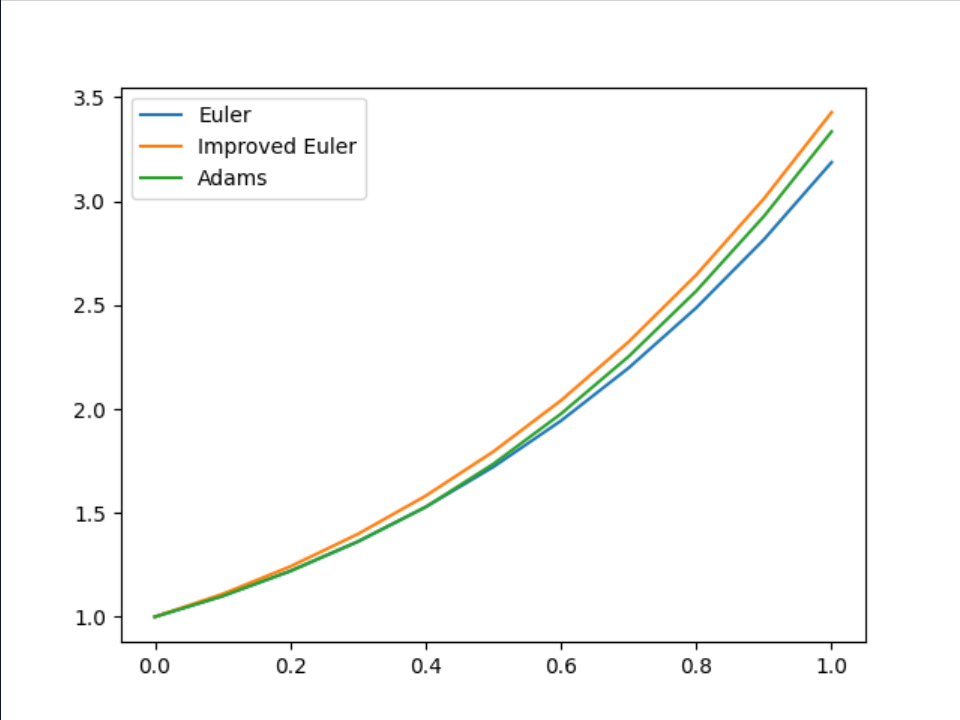
# plt.legend()

# plt.show()

# if \_\_name\_\_ == "\_\_main\_\_":

# main()

# Результаты выполнения программы



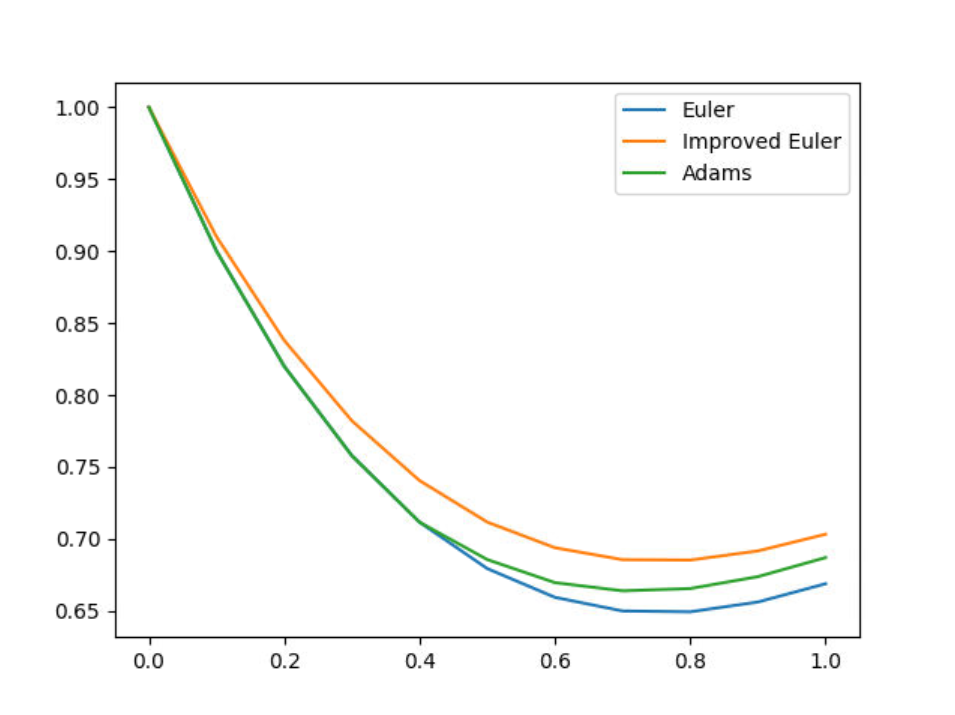
# f1 = x + y

# f2 = sin(x) - y

# f3 = e^(-x) - y^2

# Выберите функцию f1/f2/f3 f1

# Точность метода Эйлера по правилу Рунге: 0.11911049008884023



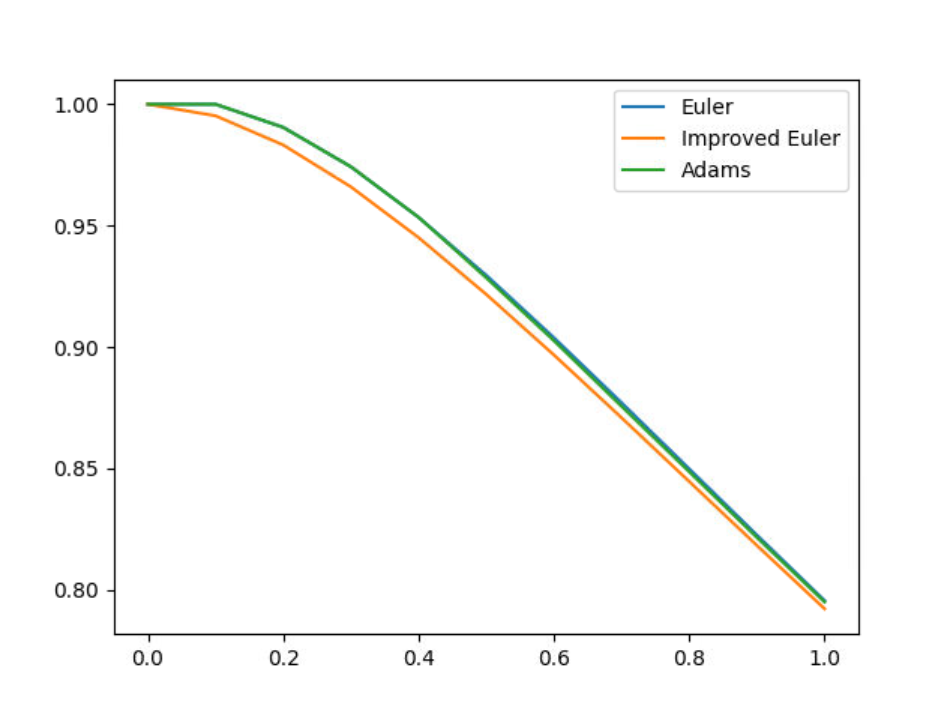
# f1 = x + y

# f2 = sin(x) - y

# f3 = e^(-x) - y^2

# Выберите функцию f1/f2/f3 f2

# Точность метода Эйлера по правилу Рунге: 0.017184509533635484



# f1 = x + y

# f2 = sin(x) - y

# f3 = e^(-x) - y^2

# Выберите функцию f1/f2/f3 f3

# Точность метода Эйлера по правилу Рунге: 0.0016874540522936465

# Выводы

В ходе выполнения данной лабораторной работы я рассмотрел и реализовал численные методы решения обыкновенных дифференциальных уравнений: метод Эйлера, усовершенствованный метод Эйлера и метод Адамса.

Реализация этих методов была произведена на языке Python. Я также реализовал правило Рунге для оценки точности одношагового метода (метода Эйлера). Визуализация результатов позволила продемонстрировать эффективность каждого из методов. Во время работы я поработал с численными методами в решении обыкновенных дифференциальных уравнений.