

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет ПИ и КТ

Лабораторная работа №6

по дисциплине: «Вычислительная математика»

«Численное решение обыкновенных дифференциальных уравнений»

Вариант 1

Выполнил:

**Болорболд Аригуун**,

группа P3211

Преподаватель:

**Малышева Татьяна Алексеевна**

Санкт-Петербург

2024



1. **Цель работы:**

Решить задачу Коши для обыкновенных дифференциальных уравнений численными методами.

1. **Рабочие формулы:**

Метод Эйлера:

Усовершенствованный метод Эйлера:

Метод Рунге-Кутта 4-го порядка:

Метод Адамса:

Метод Милна:

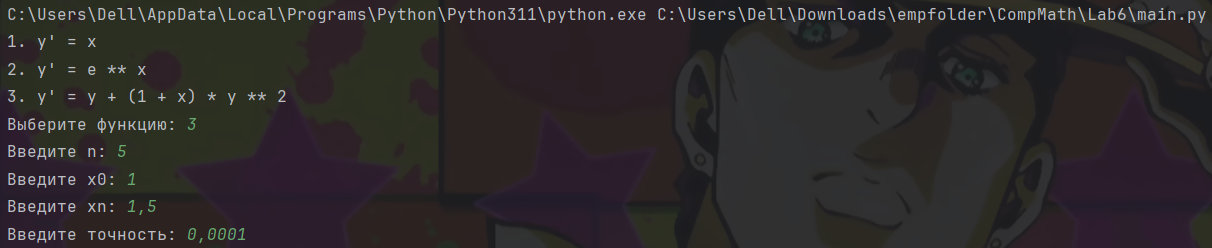
а) Этап прогноза:

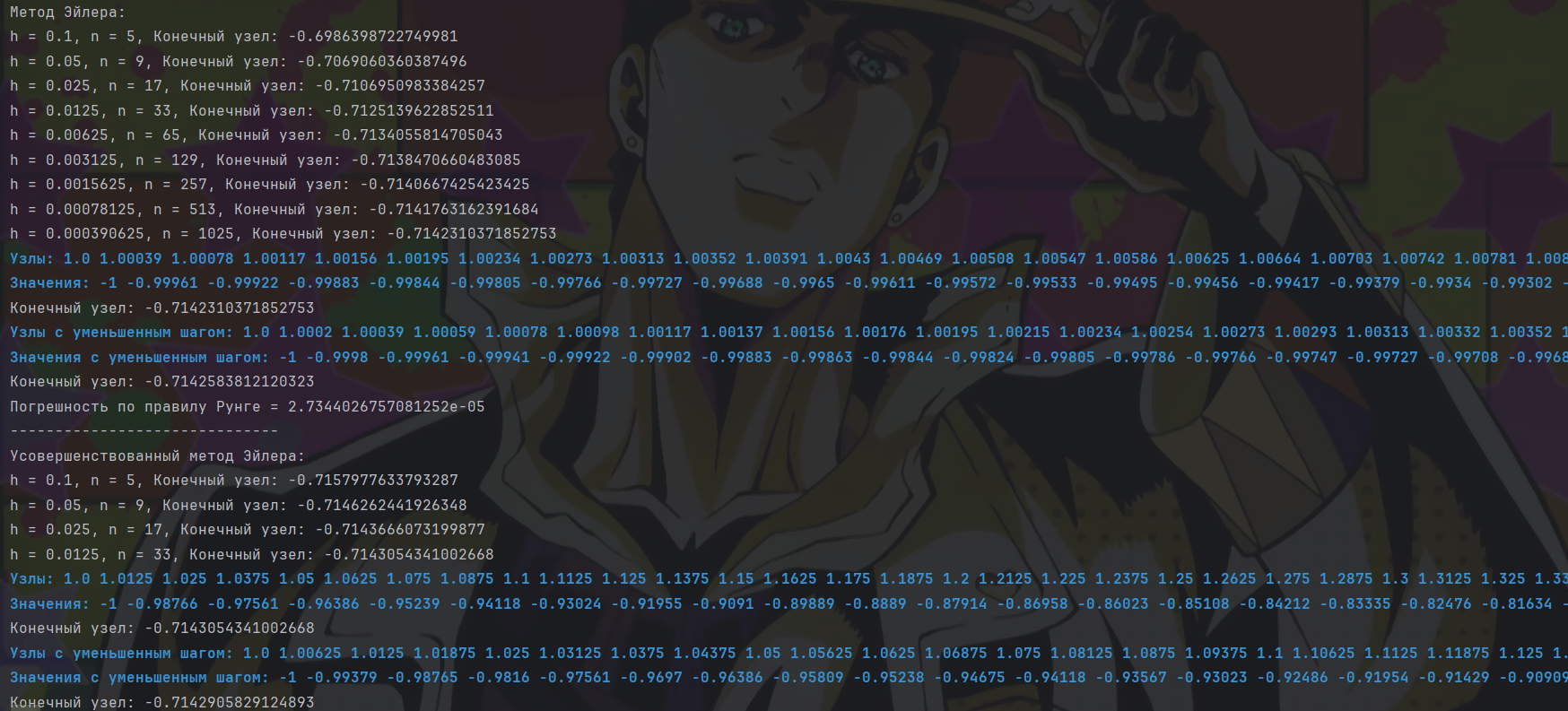
б) Этап коррекции:

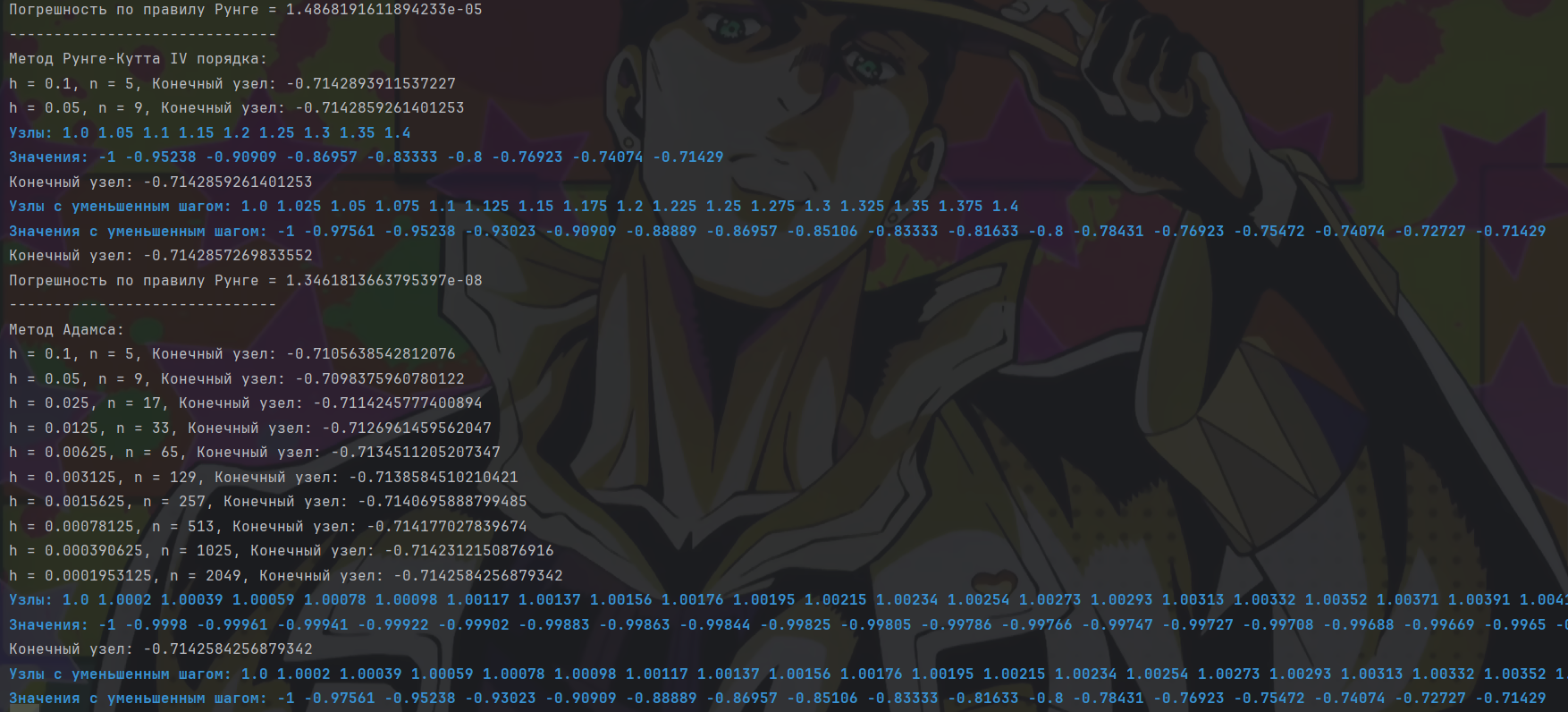
1. **Программная реализация задачи:**

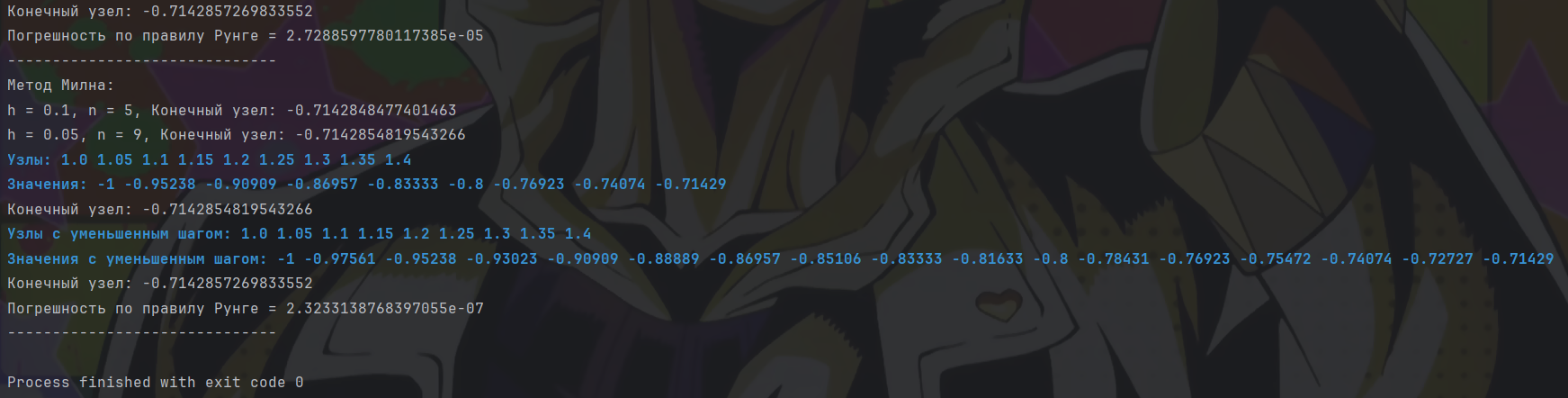
from matplotlib import pyplot as plt  
from math import exp  
  
def euler\_method(f, args, y0, h):  
 ords = [y0]  
 for i in range(1, len(args)):  
 y = ords[i - 1] + h \* f(args[i - 1], ords[i - 1])  
 ords.append(y)  
 return ords  
  
def modified\_euler\_method(f, args, y0, h):  
 ords = [y0]  
 for i in range(1, len(args)):  
 ords.append(ords[i - 1] + h \* f(args[i - 1] + h / 2, ords[i - 1] + h / 2 \* f(args[i - 1], ords[i - 1])))  
 return ords  
  
def fourth\_order\_runge\_kutta\_method(f, args, y0, h):  
 ords = [y0]  
 for i in range(1, len(args)):  
 x, y = args[i - 1], ords[i - 1]  
 k1 = h \* f(x, y)  
 k2 = h \* f(x + h / 2, y + k1 / 2)  
 k3 = h \* f(x + h / 2, y + k2 / 2)  
 k4 = h \* f(x + h, y + k3)  
 y\_next = y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
 ords.append(y\_next)  
 return ords  
  
def milne\_method(f, args, y0, h, eps=1e-7):  
 ords = fourth\_order\_runge\_kutta\_method(f, args[:4], y0, h)  
 for i in range(4, len(args)):  
 pre\_y = ords[i - 4] + 4 \* h / 3 \* \  
 (2 \* f(args[i - 3], ords[i - 3]) -  
 f(args[i - 2], ords[i - 2]) +  
 2 \* f(args[i - 1], ords[i - 1]))  
 cor\_y = ords[i - 2] + h / 3 \* \  
 (f(args[i - 2], ords[i - 2]) +  
 4 \* f(args[i - 1], ords[i - 1]) +  
 f(args[i], pre\_y))  
 while abs(pre\_y - cor\_y) > eps:  
 pre\_y = cor\_y  
 cor\_y = ords[i - 2] + h / 3 \* \  
 (f(args[i - 2], ords[i - 2]) +  
 4 \* f(args[i - 1], ords[i - 1]) +  
 f(args[i], pre\_y))  
 ords.append(cor\_y)  
 return ords  
  
def adams\_method(f, args, y0, h):  
 ords = fourth\_order\_runge\_kutta\_method(f, args[:4], y0, h)  
 for i in range(4, len(args)):  
 df = f(args[i - 1], ords[i - 1]) - f(args[i - 2], ords[i - 2])  
 d2f = f(args[i - 1], ords[i - 1]) - 2 \* f(args[i - 2], ords[i - 2]) + f(args[i - 3], ords[i - 3])  
 d3f = f(args[i - 1], ords[i - 1]) - 3 \* f(args[i - 2], ords[i - 2]) + 3 \* f(args[i - 3], ords[i - 3]) - f(args[i - 4], ords[i - 4])  
 y = ords[i - 1] + h \* f(args[i - 1], ords[i - 1]) + h \*\* 2 / 2 \* df + 5 \* h \*\* 3 / 12 \* d2f + 3 \* h \*\* 4 / 8 \* d3f  
 ords.append(y)  
 return ords  
  
  
def plot(a, b, func, dx=0.01):  
 args, ords = [], []  
 a -= dx  
 b += dx  
 x = a  
 while x <= b:  
 args.append(x)  
 ords.append(func(x))  
 x += dx  
 plt.plot(args, ords, 'g')  
def main(f, args, y0, exact\_y, h, eps):  
 methods = [euler\_method,  
 modified\_euler\_method,  
 fourth\_order\_runge\_kutta\_method,  
 adams\_method,  
 milne\_method]  
 methods\_names = {  
 euler\_method: "Метод Эйлера:",  
 modified\_euler\_method: "Усовершенствованный метод Эйлера:",  
 fourth\_order\_runge\_kutta\_method: "Метод Рунге-Кутта IV порядка:",  
 adams\_method: "Метод Адамса:",  
 milne\_method: "Метод Милна:"  
 }  
 h\_original = h  
 args\_original = args  
 for method in methods:  
 err = eps + 1  
 print(methods\_names.get(method))  
  
 if h < h\_original:  
 h = h\_original  
 if len(args) > len(args\_original):  
 args = args\_original  
 ords\_original = method(f, args\_original, y0, h)  
 while True:  
 ords = method(f, args, y0, h)  
 if method in (adams\_method, milne\_method):  
 inaccuracy = [abs(exact\_y(x) - y) for x, y in zip(args[1:], ords[1:])]  
 print(f"h = {h}, n = {len(args)}, Конечный узел: {ords[-1]}")  
 else:  
 args2 = []  
 for x1, x2 in zip(args, args[1:]):  
 args2.extend([x1, (x1 + x2) / 2])  
 args2.extend([args[-1]])  
 ords2 = method(f, args2, y0, h / 2)  
 if method is fourth\_order\_runge\_kutta\_method:  
 p = 4  
 else:  
 p = 1  
 inaccuracy = [abs(y1 - y2) / (2 \*\* p - 1) for y1, y2 in zip(ords, ords2[0::2])]  
 print(f"h = {h}, n = {len(args)}, Конечный узел: {ords[-1]}")  
 if err <= eps:  
 print("\033[1;34mУзлы:", \*map(lambda x: round(x, 5), args), "\033[0m")  
 print("\033[1;34mЗначения:", \*map(lambda x: round(x, 5), ords), "\033[0m")  
 print("Конечный узел:", ords[-1])  
 print("\033[1;34mУзлы с уменьшенным шагом:", \*map(lambda x: round(x, 5), args2), "\033[0m")  
 print("\033[1;34mЗначения с уменьшенным шагом:", \*map(lambda x: round(x, 5), ords2), "\033[0m")  
 print("Конечный узел:", ords2[-1])  
 print(f"Погрешность по правилу Рунге = {max(inaccuracy)}")  
 break  
 else:  
 err = max(inaccuracy)  
 h /= 2  
 args2.clear()  
 for x1, x2 in zip(args, args[1:]):  
 args2.extend([x1, (x1 + x2) / 2])  
 args2.extend([args[-1]])  
 args = args2.copy()  
 plt.title(methods\_names.get(method)[:-1])  
  
 plot(args\_original[0], args\_original[-1], exact\_y)  
 for i in range(len(args\_original)):  
 plt.scatter(args\_original[i], ords\_original[i], c='r')  
 plt.xlabel("X")  
 plt.ylabel("Y")  
 plt.show()  
 print('-' \* 30)  
  
def read\_number(s: str):  
 while True:  
 try:  
 return float(input(s).replace(",", "."))  
 except Exception:  
 continue  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("1. y' = x")  
 print("2. y' = e \*\* x")  
 print("3. y' = y + (1 + x) \* y \*\* 2")  
 mode = read\_number("Выберите функцию: ")  
 n = read\_number("Введите n: ")  
 x0 = read\_number("Введите x0: ")  
 xn = read\_number("Введите xn: ")  
 eps = read\_number("Введите точность: ")  
 h = (xn - x0) / n  
 xs = [x0 + h \* i for i in range(int(n))]  
 try:  
 if mode == 1:  
 f = lambda x, y: x  
 y0 = 1  
 exact\_y = lambda x: x \*\* 2 / 2 + 1  
 elif mode == 2:  
 f = lambda x, y: exp(x)  
 y0 = 0  
 exact\_y = lambda x: exp(x) - 1  
 elif mode == 3:  
 f = lambda x, y: y + (1 + x) \* y \*\* 2  
 y0 = -1  
 exact\_y = lambda x: - (1 / x)  
 except (ZeroDivisionError, ArithmeticError) as e:  
 print("Функция не определена")  
 main(f, xs, y0, exact\_y, h, eps)

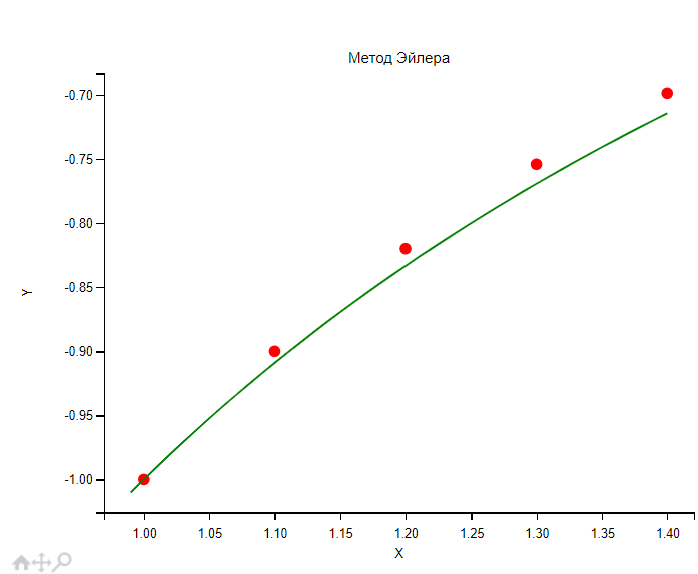
1. **Тестовые данные:**

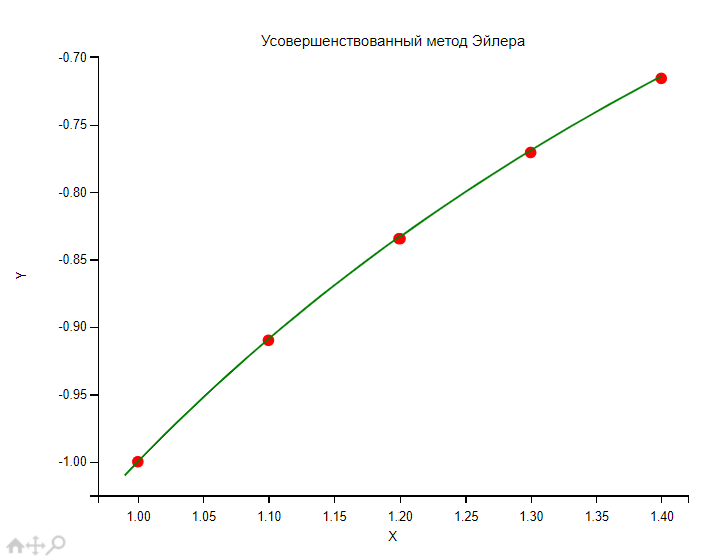


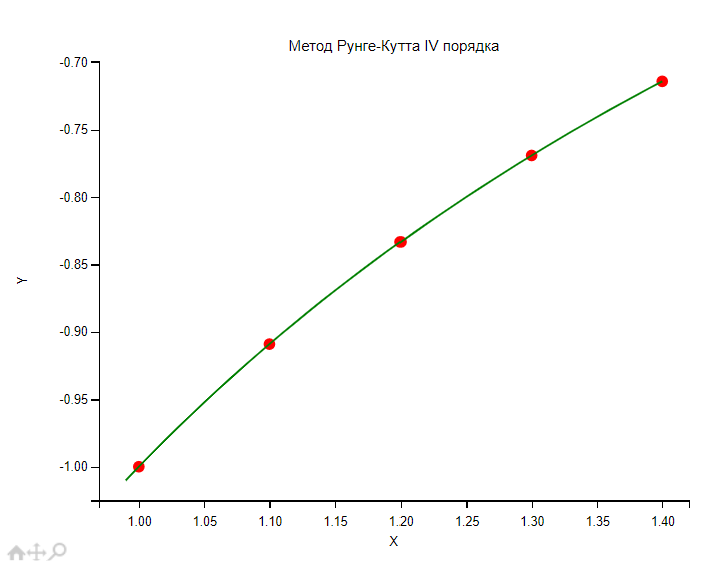


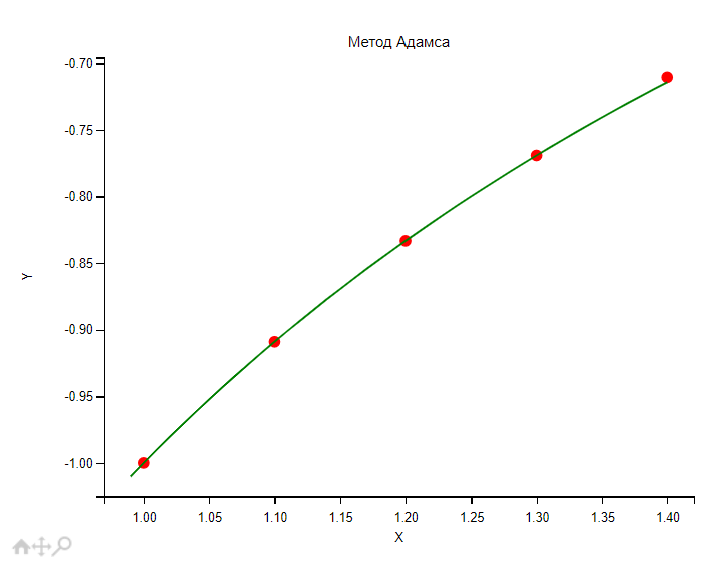


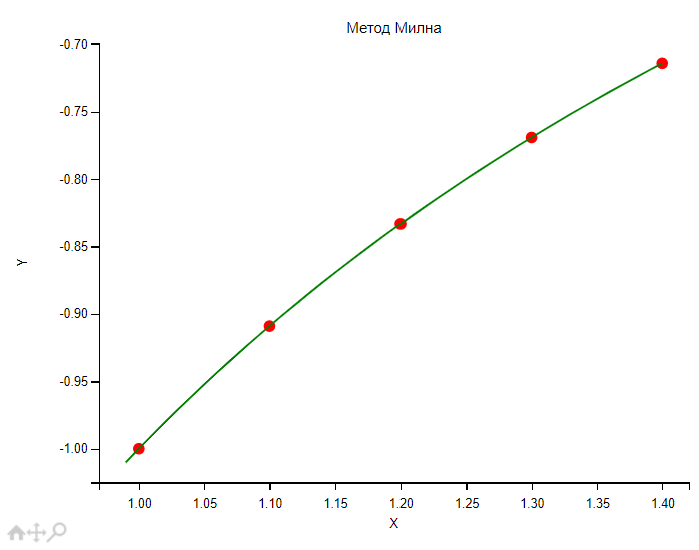












1. **Выводы:**

В ходе данной лабораторной работы я познакомился с численными решением ОДУ.

Метод Эйлера — простой, но довольно грубый метод, даёт удовлетворительную точность лишь при малом шаге .

Модифицированные метод Эйлера — более точный по сравнению с оригинальным.

Методы Рунге-Кутта — хороший метод, но требует довольно много вычислений. Все вышеперечисленные методы являются одношаговыми.

Метод Адамса — точный многошаговый метод. Использует на каждом шаге результаты предыдущих 4 шагов и конечные разности.

Метод Милна — многошаговый метод прогноза и коррекции. Коррекция проводится до тех пор, пока она не будет приближаться к прогнозу. Также использует результаты предыдущих 4 шагов.