Московский государственный университет
имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики

# Компьютерный практикум по курсе

# «ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

## ЗАДАНИЕ №2
## Подвариант №1

Решение задачи Коши для дифференциального уравнения первого порядка или системы дифференциальных уравнений первого порядка

**ОТЧЕТ**
**о выполненном задании**

студента 202 учебной группы факультета ВМК МГУ
Кузнецова Михаила Константиновича

Москва
2020

# Содержание

# Цель работы

Освоить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциального уравнения (или системы дифференциальных уравнений) первого порядка.

# Постановка задачи

Рассматривается обыкновенное дифференциальное уравнение первого порядка, разрешенное относительно производной и имеющее вид:

$$\frac{dy}{dx} = f(x, y), x_0 < x,$$

с дополнительным начальным условием, заданным в точке $x = x_0$:

$$y(x_0) = y_0.$$

Предполагается, что правая часть уравнения такова, что гарантирует существование и единственность решения задачи Коши. В том случае, если рассматривается не одно дифференциальное уравнение, а система обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производных неизвестных функций, то соответствующая задача Коши имеет вид (на примере двух дифференциальных уравнений):

$$\begin{cases} \dfrac{dy_1}{dx} = f_1(x, y_1, y_2), \\ \dfrac{dy_2}{dx} = f_2(x, y_1, y_2), \end{cases}, x > x_0.$$

Начальные условия задаются в точке $x = x_0$:

$$y_1(x_0) = y_1^{(0)}, y_2(x_0) = y_2^{(0)}$$

Также предполагается, что правые части уравнений заданы так, что это гарантирует существование и единственность решения задачи Коши для системы обыкновенных дифференциальных уравнений первого порядка в форме, разрешенной относительно производных неизвестных функций.

# Задачи практической работы

1. Решить задачу Коши методами Рунге-Кутта второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой (на равномерной сетке); полученное конечно-разностное уравнение (или уравнения в случае системы), просчитать численно;

2. Найти численное решение задачи и построить его график;

3. Найденное численное решение сравнить с точным решением дифференциального уравнения.

## Метод Рунге-Кутта

Разложим решение дифференциального уравнения по формуле Тейлора и получим разностное уравнение:

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i) + \frac{h}{2} * \left\{ \frac{\partial f}{\partial x}(x_i, y_i) + \frac{\partial f}{\partial y}(x_i, y_i) f(x_i, y_i) \right\}$$

Главная идея метода Рунге-Кутта заключается в том, чтобы приближенно заменить правую часть на сумму значений функции f в двух разных точках с точностью до членов порядка $h^2$. Таким образом, получим однопараметрическое семейство разностных схем Рунге-Кутта:

$$\frac{y_{i+1} - y_i}{h} = (1-\alpha) f(x_i, y_i) + \alpha f \left( x_i + \frac{h}{2\alpha}, y_i + \frac{h}{2\alpha} f(x_i, y_i) \right)$$

Наиболее удобной для вычисления разностной схемой этого семейства соответствует значение параметра $\alpha = \frac{1}{2}$. Тогда формула принимает рекурентный вид:

$$y_{i+1} = y_i + \frac{h}{2} \left\{ f(x_i, y_i) + f(x_i + h, y_i + h f(x_i, y_i)) \right\}$$

В случае решения задачи Коши для системы дифференциальных уравнений, эта формула также справедлива для любого $y^{(i)}$ из вектора решений.

Иногда второго порядка точности недостаточно. Поэтому часто используется схема Рунге-Кутта четвертого порядка точности следующего вида:

$$\frac{y_{i+1} - y_i}{h} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

где

$$k_1 = f(x_i, y_i);$$

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

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

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

## Описание программы

- При запуске программы необходимо ввести количество итераций
- выбрать размер шага h,
- номер теста,
- порядок метода Рунге-Кутта (2 или 4)

  Результатом работы программы являются пара списков $x_i, y_i$ или $x_i, y_{ji}$ соответсвенно.

# Код программы

```
import numpy as np

def runge_kurt_2(f, x0=0, y0=0, c_it = 10, h=0.25):
    y = []
    x = []
    xi = x0
    yi = y0
    for i in range(c_it):
        k1 = h * f(xi, yi)
        k2 = h * f(xi + h, yi + k1)
        yi_1 = yi + 0.5 * (k1 + k2)
        xi += h
        x.append(xi)
        y.append(yi_1)
        yi = yi_1
    return x, y

def runge_kurt_4(f, x0=0, y0=0, c_it = 10, h=0.25):
    y = []
    x = []
    xi = x0
    yi = y0
    for i in range(c_it):
        k1 = h * f(xi, yi)
        k2 = h * f(xi + h/2, yi + k1/2)
        k3 = h * f(xi + h/2, yi + k2/2)
        k4 = h * f(xi + h, yi + k3)
        yi_1 = yi + 1/6 * (k1 + 2*k2 + 2*k3 + k4)
        xi += h
        x.append(xi)
        y.append(yi_1)
        yi = yi_1
    return x, y

def runge_kurt_s(f1, f2, x0=0, y0=0, z0=0, c_it=10, h=0.25):
    y = []
    z = []
    x = []
    xi = x0
    yi = y0
    zi = z0
    for i in range(c_it):
        k1 = h * f1(xi, yi, zi)
        l1 = h * f2(xi, yi, zi)
        k2 = h * f1(xi + h/2, yi + k1/2, zi + l1/2)
        l2 = h * f2(xi + h/2, yi + k1/2, zi + l1/2)
        k3 = h * f1(xi + h/2, yi + k2/2, zi + l2/2)
        l3 = h * f2(xi + h/2, yi + k2/2, zi + l2/2)
        k4 = h * f1(xi + h, yi + k3, zi + l3)
        l4 = h * f2(xi + h, yi + k3, zi + l3)
        yi_1 = yi + 1/6 * (k1 + 2*k2 + 2*k3 + k4)
        zi_1 = zi + 1/6 * (l1 + 2*l2 + 2*l3 + l4)
        xi += h
        x.append(xi)
        y.append(yi_1)
        z.append(zi_1)
        yi = yi_1
        zi = zi_1
    return x, y, z

#begin of the program
n = int(input())
h = float(input())
numt = int(input())
degree = int(input())
if numt == 1:
    if degree == 2:
        x, y = runge_kurt_2(f1_1, x0=0, y0=1, c_it=n, h=h)
```

```
    else:
        x, y = runge_kurt_4(f1_1, x0=0, y0=1, c_it=n, h=h)
    print((x, y))
else:
    x, y, z = runge_kurt_s(f2_1, f2_2, x0=0, y0=1, z0=0.25, c_it=n, h=h)
    print((x, y))
    print((x, z))
```

# Тестирование программы

Таблица 1. Вариант 6

$y' = (x - x^2)y$, $y(0) = 1$.

Аналитическое решение $y = e^{-1/6*x^2(-3+2x)}$

По результатам работы программы построен график, где отмеченно аналитическое решение и решение методом Рунге-Кутта двух порядков

Количество итераций - 10

Шаг - 0.25

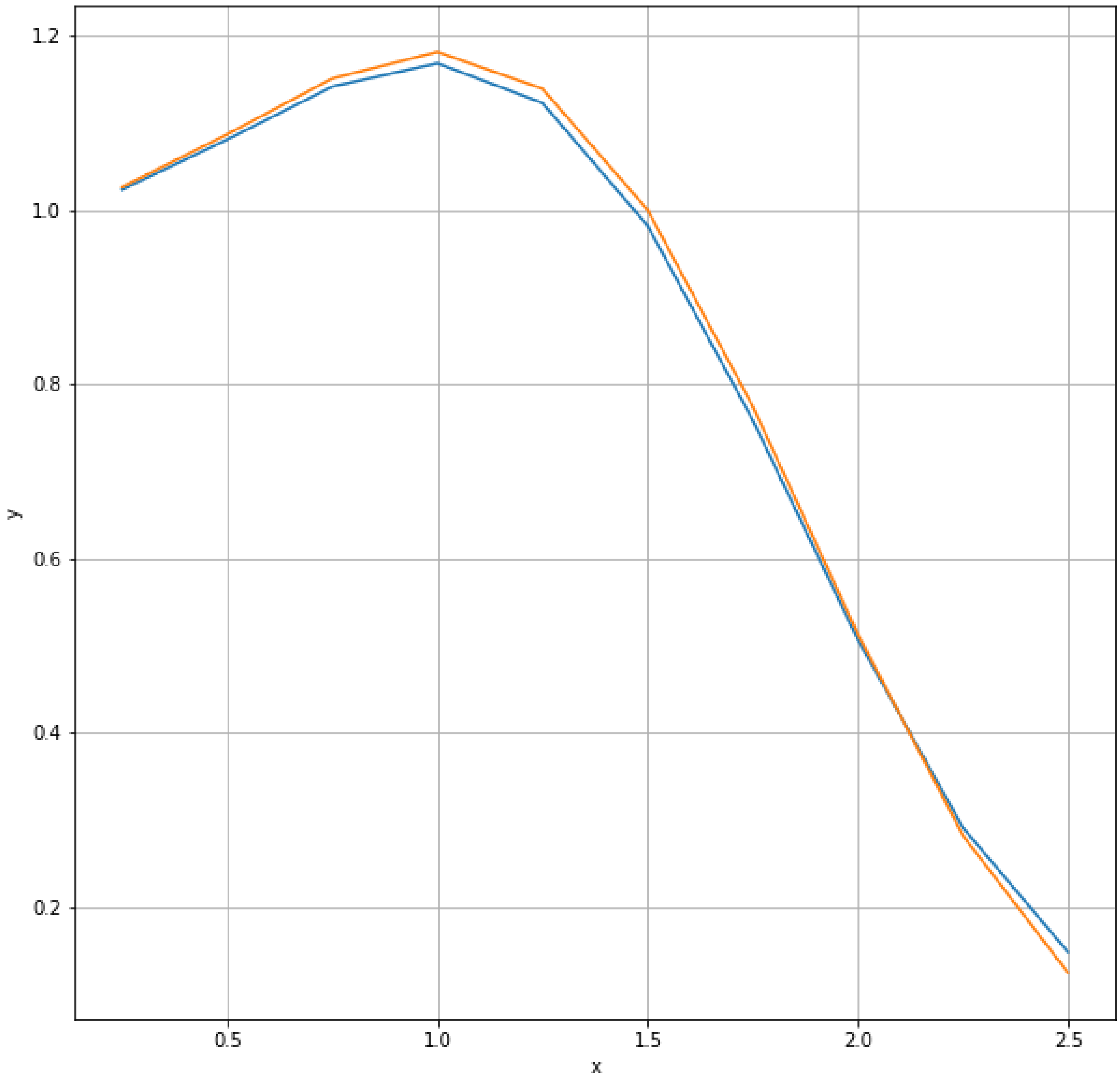


Рис. 1:

По результатам работы программы построен график, где отмеченно аналитическое решение и решение методом Рунге-Кутта четырёх порядков
Количество итераций - 10
Шаг - 0.25

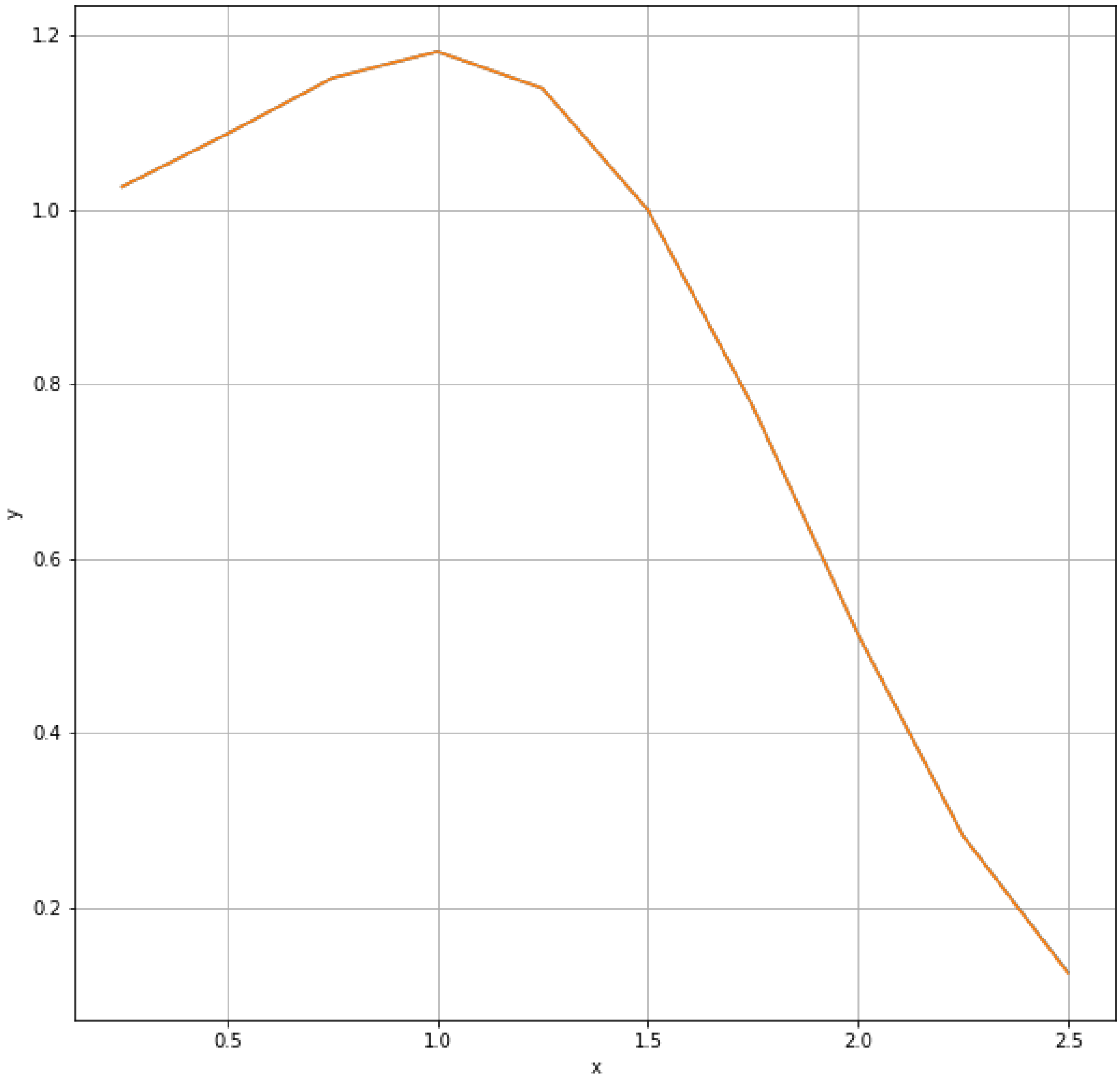


Рис. 2:

Дополнительный тест 1

$y' = sin(x) - y$, $y(0) = 10$.

Аналитическое решение $y = -0.5cos(x) + 0.5sin(x) + \frac{21}{2}e^{-x}$

По результатам работы программы построен график, где отмеченно аналитическое решение и решение методом Рунге-Кутта двух порядков
Количество итераций - 10
Шаг - 0.5

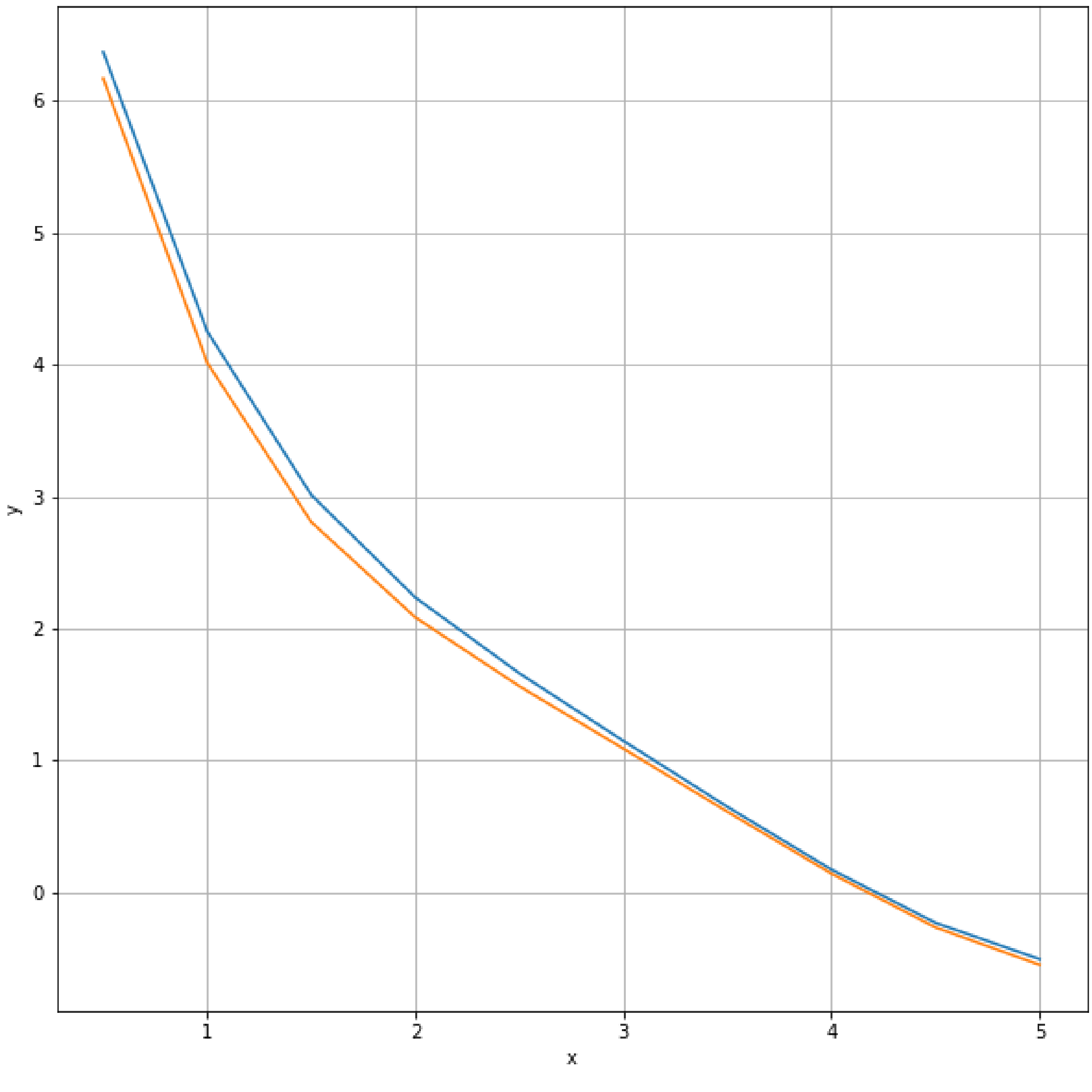


Рис. 3:

По результатам работы программы построен график, где отмеченно аналитическое решение и решение методом Рунге-Кутта четырёх порядков
Количество итераций - 10
Шаг - 0.5

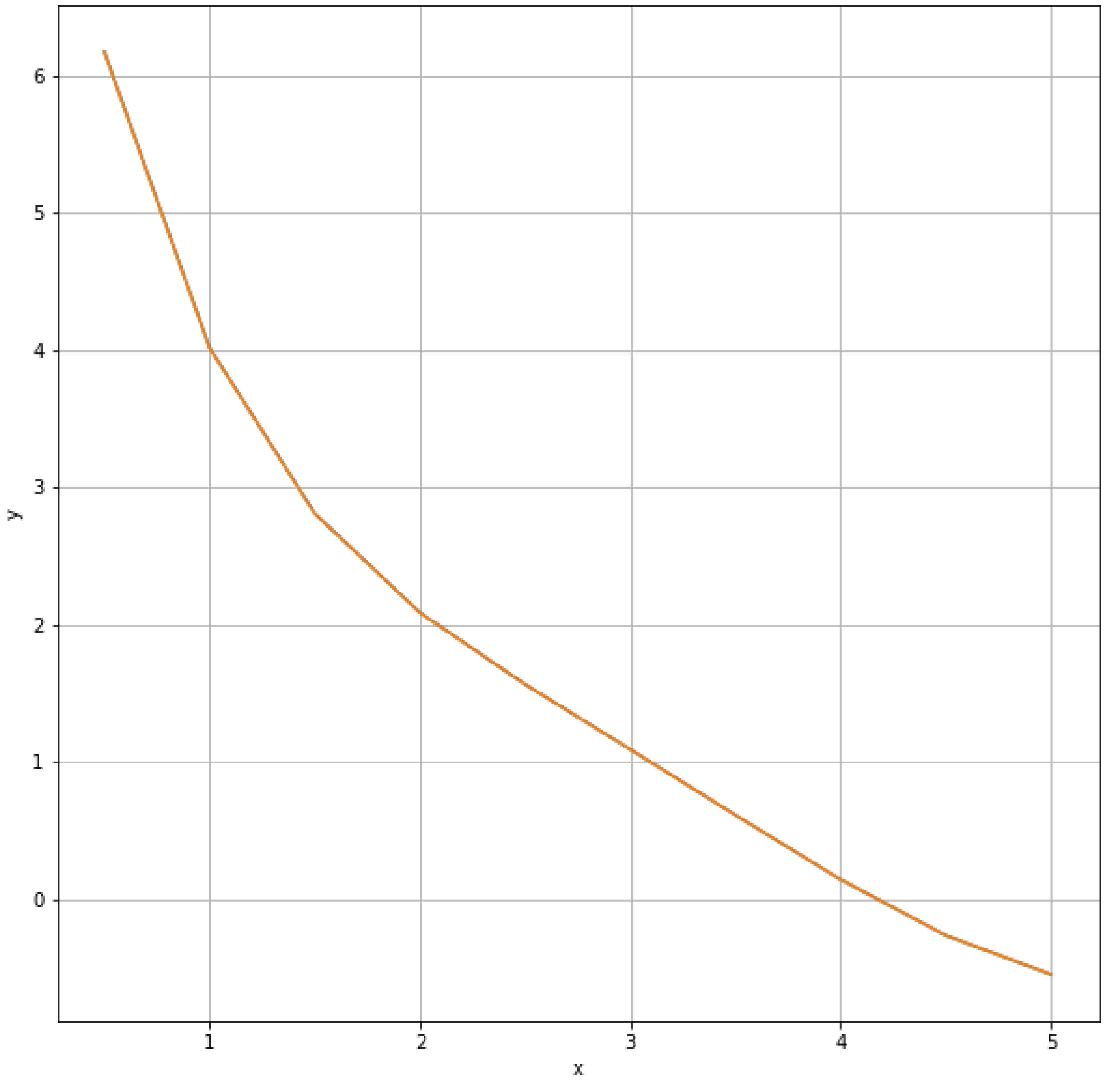


Рис. 4:

Таблица 2. Вариант 21

$$\begin{cases} y_1' = 2.4 * y_2 - y_1 \\ y_2' = e^{-y_1} - x + 2.2 * y_2, \end{cases}$$

$$y_1(0) = 1, y_2(0) = 0.25$$

По результатам работы программы построен график , где отмеченны решения методом Рунге-Кутта четырёх порядков
Количество итераций - 10
Шаг - 0.25

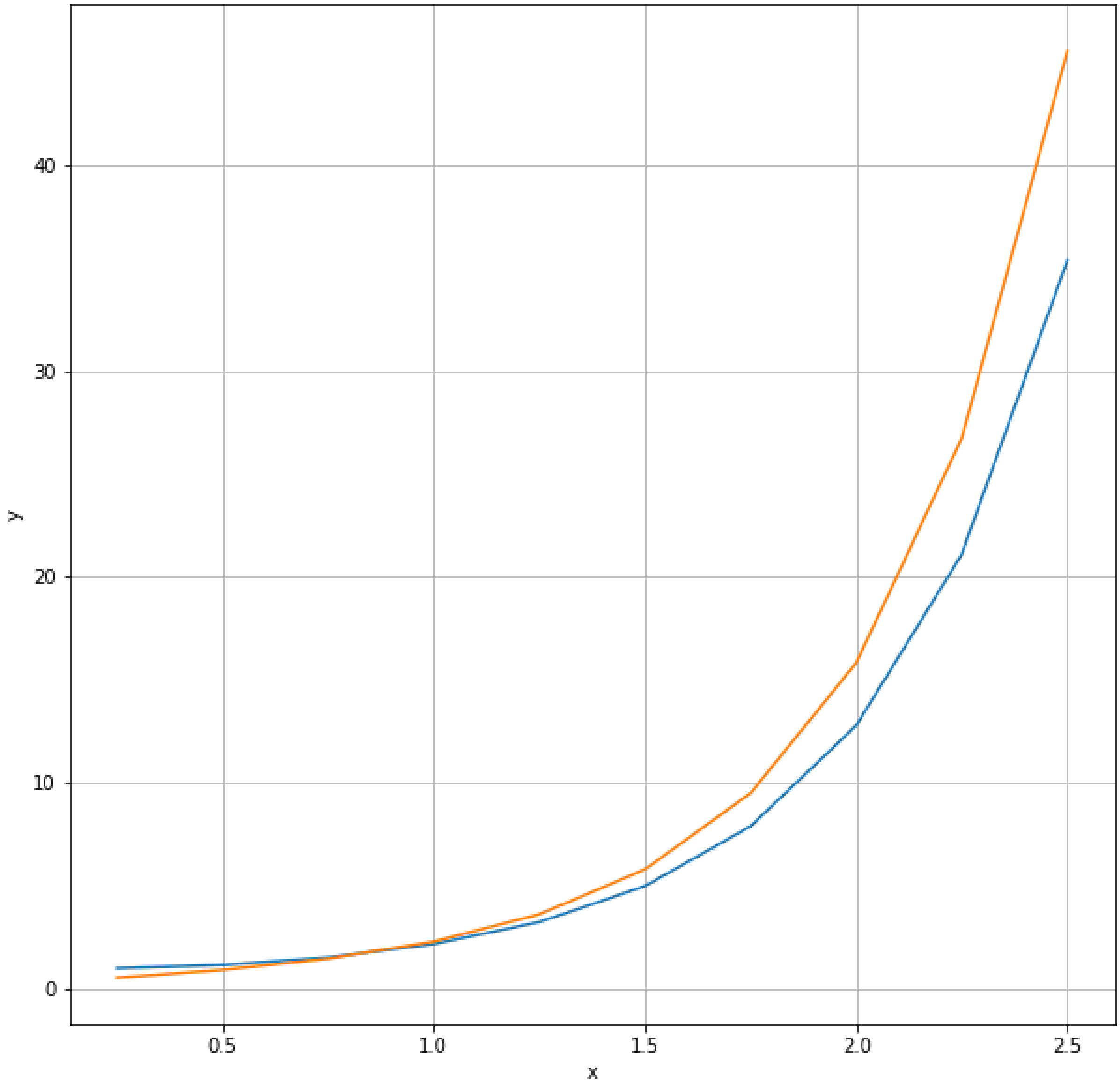


Рис. 5:

Дополнительный тест

$$\begin{cases} y_1' = ln(2x + \sqrt{4x^2 + y_2^2}) \\ y_2' = \sqrt{4x^2 + y_1^2} \end{cases}$$

$$y_1(0) = 0.5, y_2(0) = 1$$

По результатам работы программы построен график , где отмеченны решения методом Рунге-Кутта четырёх порядков
Количество итераций - 10
Шаг - 0.25

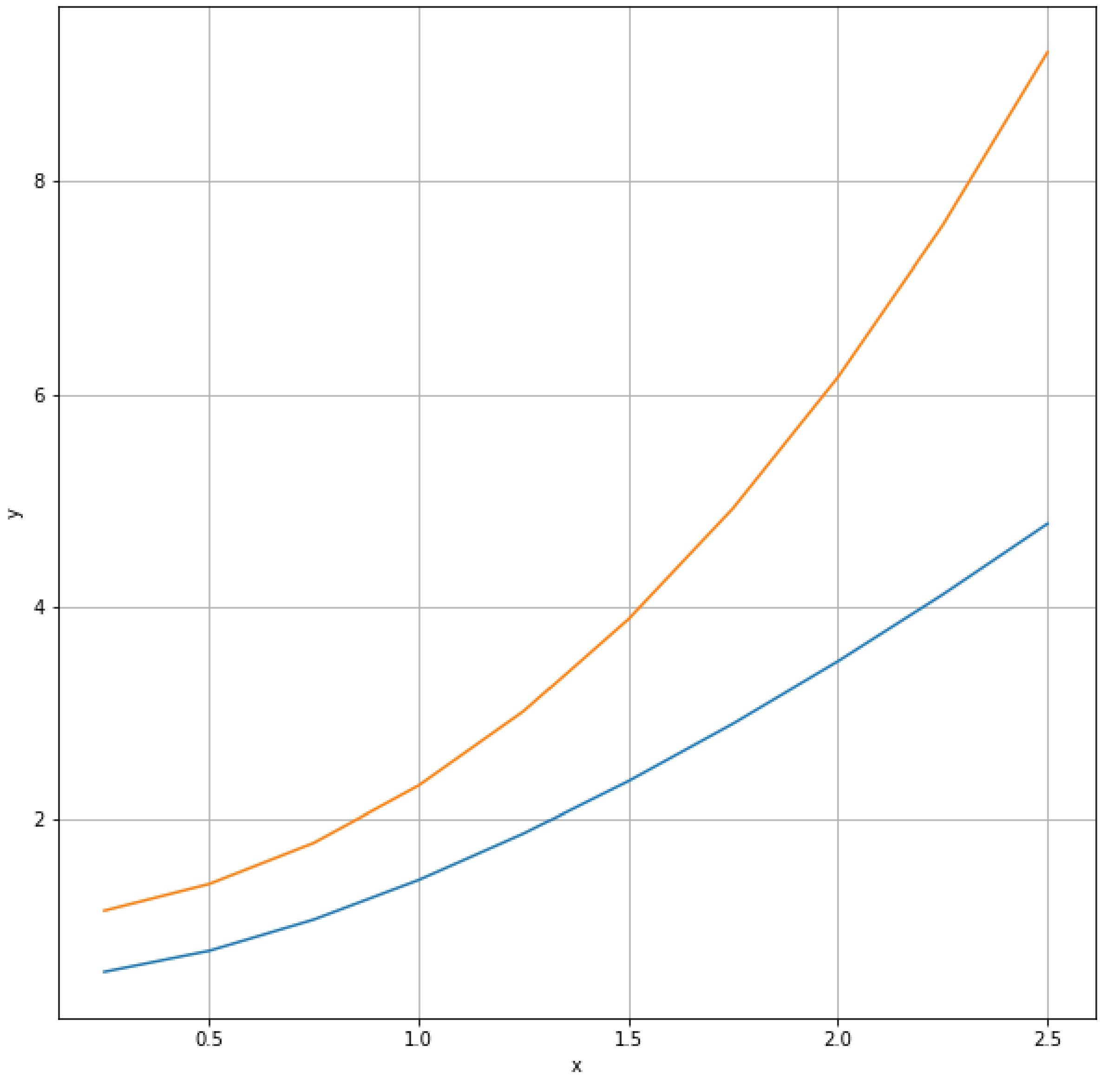


Рис. 6:

# Выводы

В ходе работы рассмотрен метод Рунге-Кутта второго и четвертого порядков для решения задачи Коши дифференциального уравнения первого порядка и системы дифференциальных уравнений. Было показано на конкретных примерах, что метод Рунге-Кутта четвертого порядка точнее, чем второго, но реализуется сложнее (в схеме второго порядка точности на каждом шаге функцию $f(x, y)$ приходилось вычислять два раза, здесь - четыре раза).