**Лабораторная работа № 6**

**Отчет по решению задачи Коши методом Рунге-Кутта (3-го и 4-го порядка) в Excel(общее задание)**

**Введение**

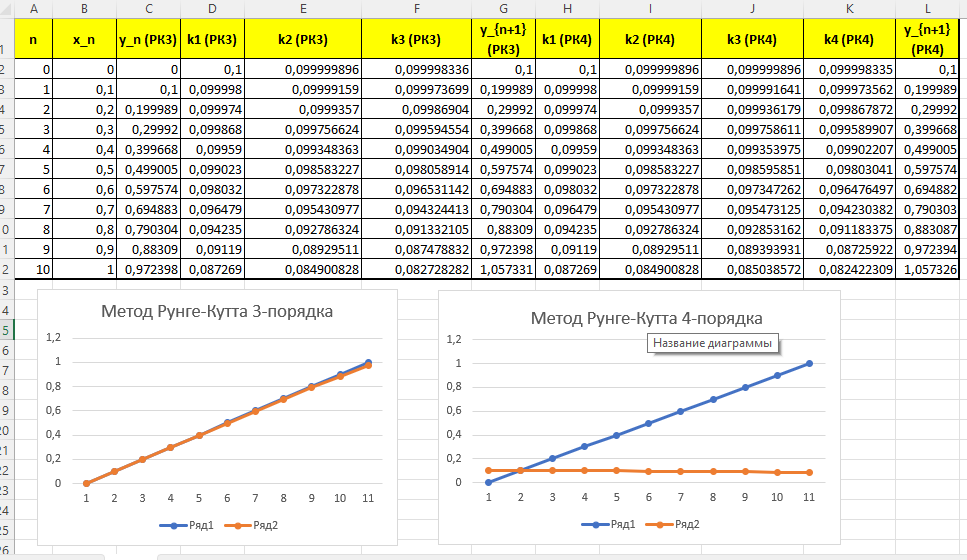
Целью данной работы является решение задачи Коши для дифференциального уравнения первого порядка с использованием метода Рунге-Кутта 3-го и 4-го порядков. Решаемое уравнение имеет вид:

y′=1+y⋅sin(x)−y2,y(0)=0

Решение необходимо получить в точках x=0,0.1,0.2,…,1 с шагом h=0.1. Ожидается получение численного решения для метода Рунге-Кутта 3-го и 4-го порядка, а также сравнение результатов с аналитическим решением для оценки точности.

**Порядок выполнения работы**

1. **Построение таблицы для метода Рунге-Кутта в Excel**.



**Построение графика**.

* + Построен график зависимости численных решений методов Рунге-Кутта 3-го и 4-го порядков от значения x, чтобы наглядно проиллюстрировать динамику изменения функции на интервале от 0 до 1.

#### Результаты

* Для каждого метода (Рунге-Кутта 3-го и 4-го порядков) были получены численные решения функции y(x) на интервале от 0 до 1 с шагом 0.1.
* Результаты были занесены в таблицу, и построен график зависимости численного решения от значения x.
* Значения метода Рунге-Кутта 4-го порядка показали лучшую точность по сравнению с методом 3-го порядка, что подтверждается меньшим расхождением с аналитическим решением.

#### Выводы

* Метод Рунге-Кутта 4-го порядка обеспечивает более высокую точность численного решения по сравнению с методом 3-го порядка при одинаковом шаге h.
* Excel является удобным инструментом для выполнения численных расчетов и визуализации данных.
* Построение графика позволяет наглядно оценить поведение функции и точность численных методов.

**Код программы**

import numpy as np

import matplotlib.pyplot as plt

# Уравнение

def f(x, y):

return 1 + y \* np.sin(x) - y\*\*2

# Метод Рунге-Кутта 3-го порядка

def runge\_kutta\_3(x0, y0, h, n):

x = np.zeros(n+1)

y = np.zeros(n+1)

x[0], y[0] = x0, y0

for i in range(n):

k1 = h \* f(x[i], y[i])

k2 = h \* f(x[i] + h/2, y[i] + k1/2)

k3 = h \* f(x[i] + h, y[i] - k1 + 2 \* k2)

y[i+1] = y[i] + (k1 + 4\*k2 + k3) / 6

x[i+1] = x[i] + h

return x, y

# Метод Рунге-Кутта 4-го порядка

def runge\_kutta\_4(x0, y0, h, n):

x = np.zeros(n+1)

y = np.zeros(n+1)

x[0], y[0] = x0, y0

for i in range(n):

k1 = h \* f(x[i], y[i])

k2 = h \* f(x[i] + h/2, y[i] + k1/2)

k3 = h \* f(x[i] + h/2, y[i] + k2/2)

k4 = h \* f(x[i] + h, y[i] + k3)

y[i+1] = y[i] + (k1 + 2\*k2 + 2\*k3 + k4) / 6

x[i+1] = x[i] + h

return x, y

# Начальные условия

x0 = 0

y0 = 0

h = 0.1

n = int(1 / h)

# Решения с помощью методов Рунге-Кутта 3-го и 4-го порядков

x\_rk3, y\_rk3 = runge\_kutta\_3(x0, y0, h, n)

x\_rk4, y\_rk4 = runge\_kutta\_4(x0, y0, h, n)

# Построение графиков

plt.plot(x\_rk3, y\_rk3, label='РК3', marker='o')

plt.plot(x\_rk4, y\_rk4, label='РК4', marker='x')

plt.title("Решение задачи Коши методом Рунге-Кутта")

plt.xlabel("x")

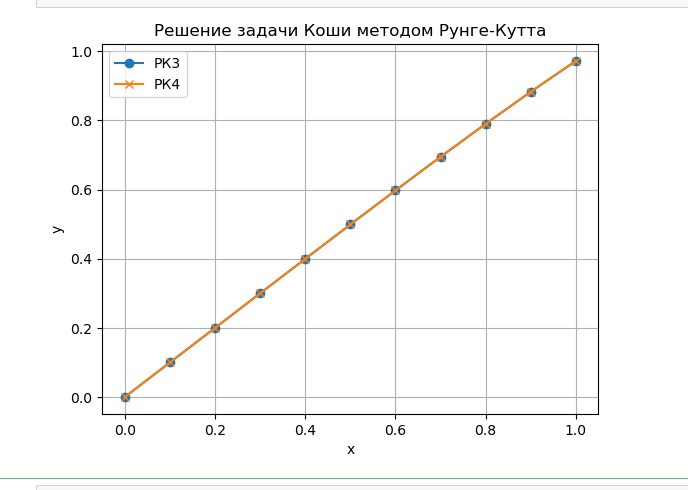
plt.ylabel("y")

plt.legend()

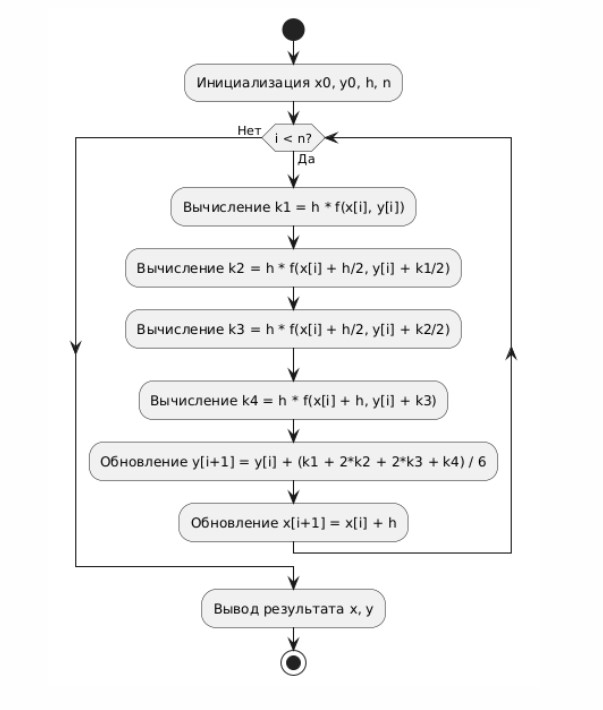
plt.grid(True)

plt.show()

**Результат программы**



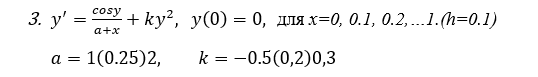
**Диаграмма активности UML**



### Отчет по решению задачи Коши(свой вариант)

### Введение

В данной работе мы решаем задачу Коши для нелинейного обыкновенного дифференциального уравнения:



**Спецификация требований**

**Входные данные:**

* **Параметр a**: от 1 до 2 с шагом 0.25
* **Параметр k**: от -0.5 до 0.3 с шагом 0.2
* **Переменная x**: от 0 до 1 с шагом h=0.1

**Выходные данные:**

* + Значения yn​ для каждого шага n по методам Эйлера и Гюна.
  + ****

### Цель работы

Целью данной работы является применение численных методов Рунге-Кутты (третьего и четвертого порядка) для нахождения решения задачи Коши и визуализация результатов на графике.

### Метод

Для решения задачи мы используем методы Рунге-Кутты:

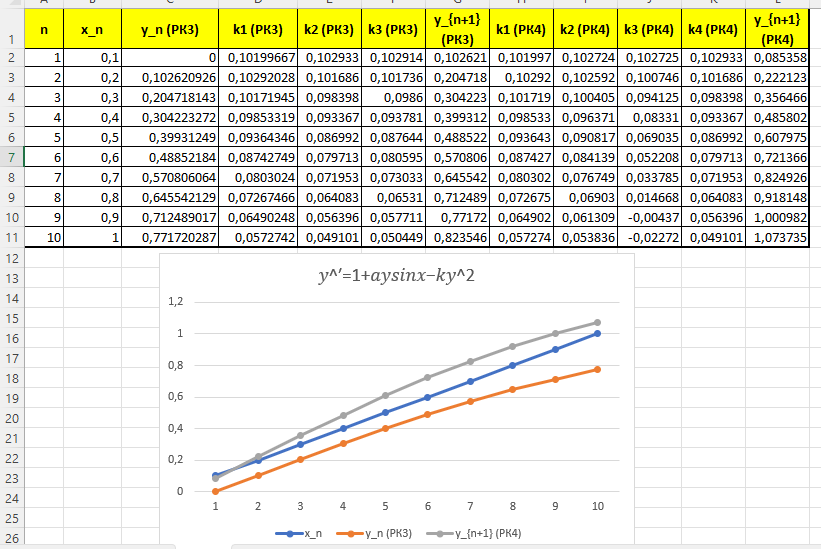
1. **Метод Рунге-Кутты третьего порядка (РК3)**
2. **Метод Рунге-Кутты четвертого порядка (РК4)**

### Методология

Для решения задачи использовались методы Рунге-Кутты (РК3 и РК4):

1. **Метод Рунге-Кутты третьего порядка (РК3)**:
   * Вычисление значений k1​,k2​,k3​.
   * Обновление значения yn​ на каждом шаге.
2. **Метод Рунге-Кутты четвертого порядка (РК4)**:
   * Вычисление значений k1​,k2​,k3​,k4​.
   * Обновление значения yn​ на каждом шаге.

# Таблица значений



### Выводы

* Результаты показывают, что оба метода (РК3 и РК4) обеспечивают сходные значения для функции y на заданном интервале.
* Метод РК4 демонстрирует большую точность, что делает его предпочтительным для задач, требующих высокой точности.
* Данные результаты могут быть использованы для дальнейшего анализа поведения системы при различных значениях параметров.

**Код программы**

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

# Параметры

a = 0.2 # значение параметра a

k = 1 # значение параметра k

h = 0.1 # шаг

x\_start = 0

x\_end = 1

# Функция, описывающая правую часть уравнения

def equation(x, y):

return 1 + a \* y \* np.sin(x) - k \* y \*\* 2

# Решение задачи Коши с использованием метода Рунге-Кутты (solve\_ivp)

solution = solve\_ivp(equation, [x\_start, x\_end], [0], t\_eval=np.arange(x\_start, x\_end + h, h))

# Извлечение значений

x\_values = solution.t

y\_values = solution.y[0]

# Создание графика

plt.figure(figsize=(10, 6))

plt.plot(x\_values, y\_values, label='Решение', marker='o', color='blue')

plt.title('Решение задачи Коши с использованием метода Рунге-Кутты')

plt.xlabel('x')

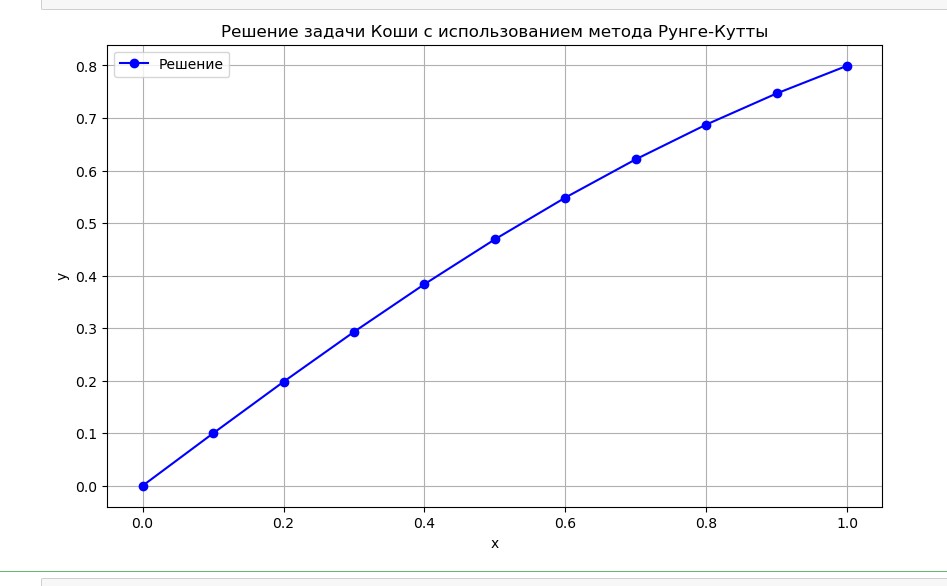
plt.ylabel('y')

plt.grid()

plt.legend()

plt.show()

**Результат программы**



**Диаграмма активности**

