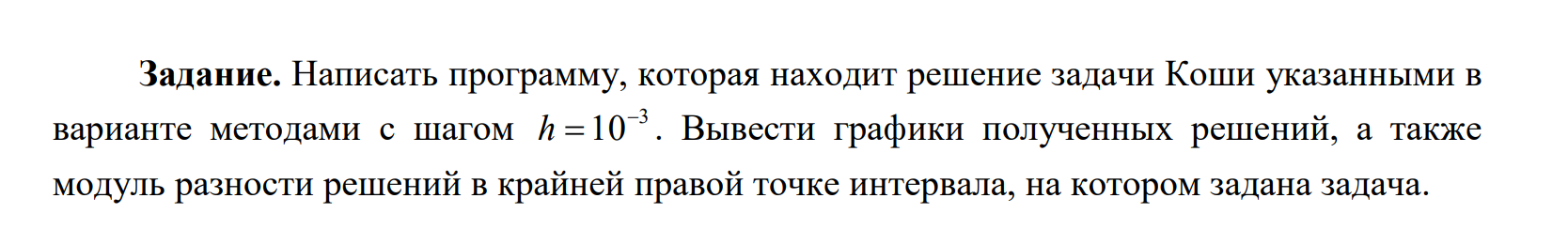
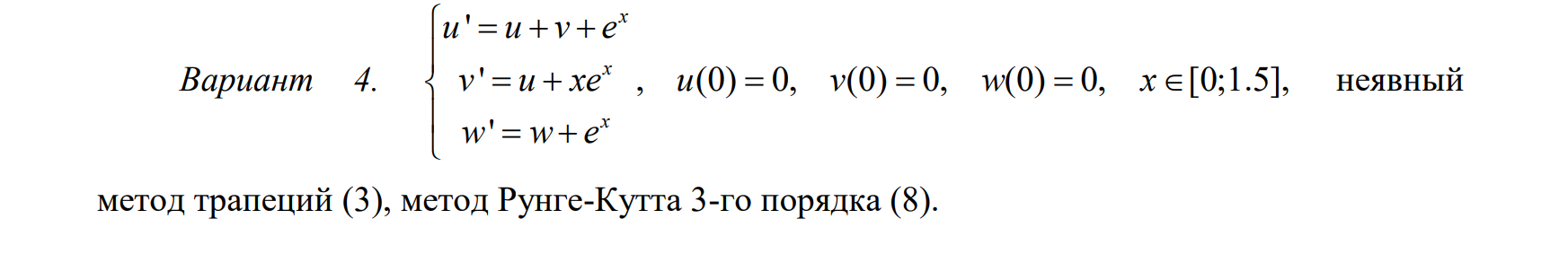
**Лабораторная работа № 3 Задача Коши для ОДУ**

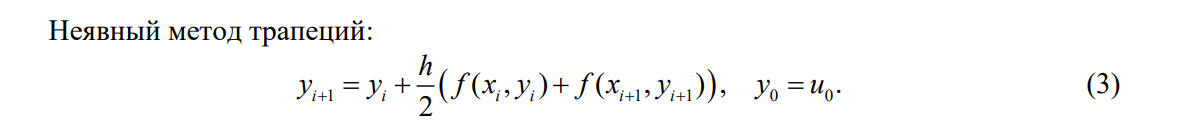
Выполнил студент 3 курса 4 группы ФПМИ БГУ Видевич Александр.

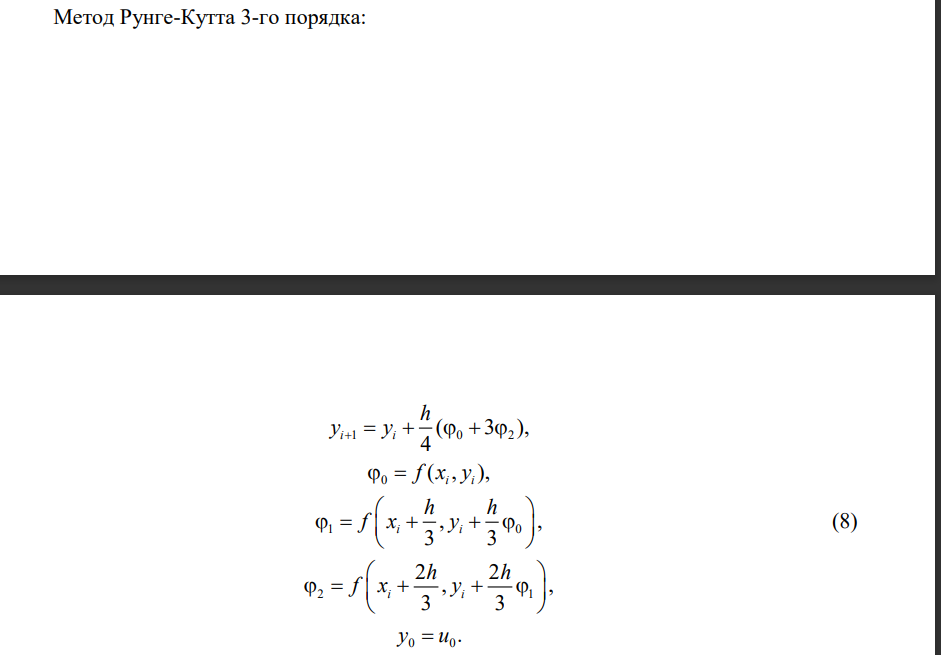
**Задача.**





**Использованная теория.**





**Листинг программы**

**main.py***from* implicit\_trapezoidal\_method *import* implicit\_trapezoidal\_method

*from* runge\_kutta\_third\_order *import* runge\_kutta\_third\_order

*from* input\_data *import* \*

*from* f *import* f

*from* display\_data *import* display\_data

x\_values, y\_values = implicit\_trapezoidal\_method(f, h, u0, x\_values)

display\_data(x\_values, y\_values,

             'Решение системы дифференциальных уравнений неявным методом трапеций')

y\_imt\_right = y\_values[-*1*]

x\_values, y\_values = runge\_kutta\_third\_order(f, h, u0, x\_values)

display\_data(x\_values, y\_values,

             'Решение системы дифференциальных уравнений методом Рунге-Кутты 3-го порядка')

y\_rk\_right = y\_values[-*1*]

print("Модуль разности решений в крайней правой точке интервала: ",

      abs(y\_imt\_right-y\_rk\_right))

**runge\_kutta\_third\_order.py**

*import* numpy *as* np

def runge\_kutta\_third\_order(f, h, y0, x\_values):

    y\_values = np.zeros((len(x\_values), len(y0)))

    y\_values[*0*] = y0

*for* i *in* *range*(*1*, len(x\_values)):

        xi = x\_values[i]

        yi\_prev = y\_values[i - *1*]

        fi0 = f(xi, yi\_prev)

        fi1 = f(xi + h / *3*, yi\_prev + h / *3* \* fi0)

        fi2 = f(xi + *2* \* h / *3*, yi\_prev + *2* \* h / *3* \* fi1)

        yi\_new = yi\_prev + h / *4* \* (fi0 + *3* \* fi2)

        y\_values[i] = yi\_new

*return* x\_values, y\_values

**implicit\_trapezoidal\_method.py**

*import* numpy *as* np

*from* scipy.optimize *import* fsolve

def implicit\_trapezoidal\_method(f, h, y0, x\_values):

    y\_values = np.zeros((len(x\_values), len(y0)))

    y\_values[*0*] = y0

*for* i *in* *range*(*1*, len(x\_values)):

        xi = x\_values[i]

        yi\_prev = y\_values[i - *1*]

        def equation(y\_next):

*return* yi\_prev + h / *2* \* (f(xi, yi\_prev) + f(xi + h, y\_next)) - y\_next

        yi\_next\_guess = yi\_prev + h \* f(xi, yi\_prev)

        yi\_next = fsolve(equation, yi\_next\_guess)

        y\_values[i] = yi\_next

*return* x\_values, y\_values

**input\_data.py**

*import* numpy *as* np

u0 = np.array([*0*, *0*, *0*])

h = *0.001*

x\_range = [*0*, *1.5*]

num\_steps = *int*((x\_range[*1*] - x\_range[*0*]) / h)

x\_values = np.linspace(x\_range[*0*], x\_range[*1*], num\_steps + *1*)

**f.py**

*import* numpy *as* np

def f(x, y):

    u, v, w = y

*return* np.array([u + v + np.exp(x), u + x \* np.exp(x), w + np.exp(x)])

**display\_data.py**

*import* matplotlib.pyplot *as* plt

def display\_data(x\_values, y\_values, title):

    u\_values = y\_values[:, *0*]

    v\_values = y\_values[:, *1*]

    w\_values = y\_values[:, *2*]

    plt.figure(figsize=(*10*, *6*))

    plt.plot(x\_values, u\_values, label='u(x)')

    plt.plot(x\_values, v\_values, label='v(x)')

    plt.plot(x\_values, w\_values, label='w(x)')

    plt.title(title)

    plt.xlabel('x')

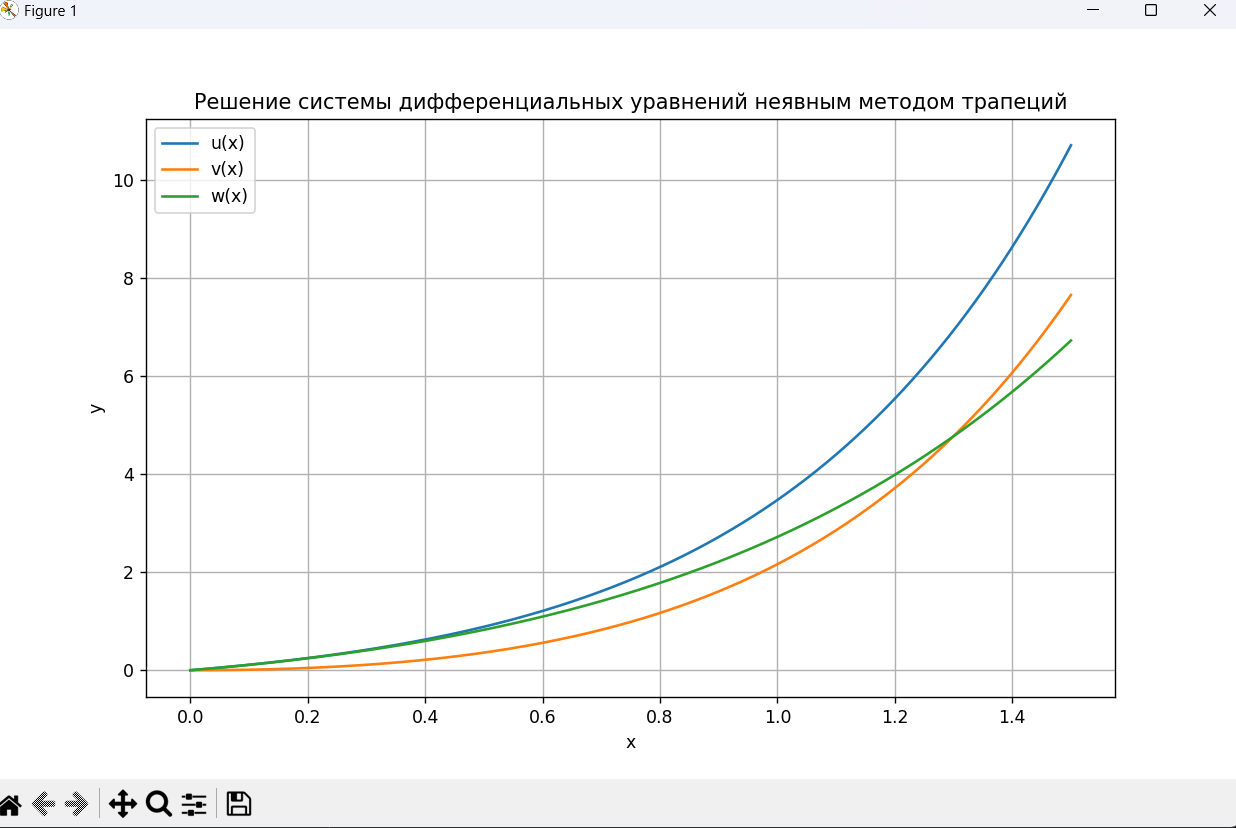
    plt.ylabel('y')

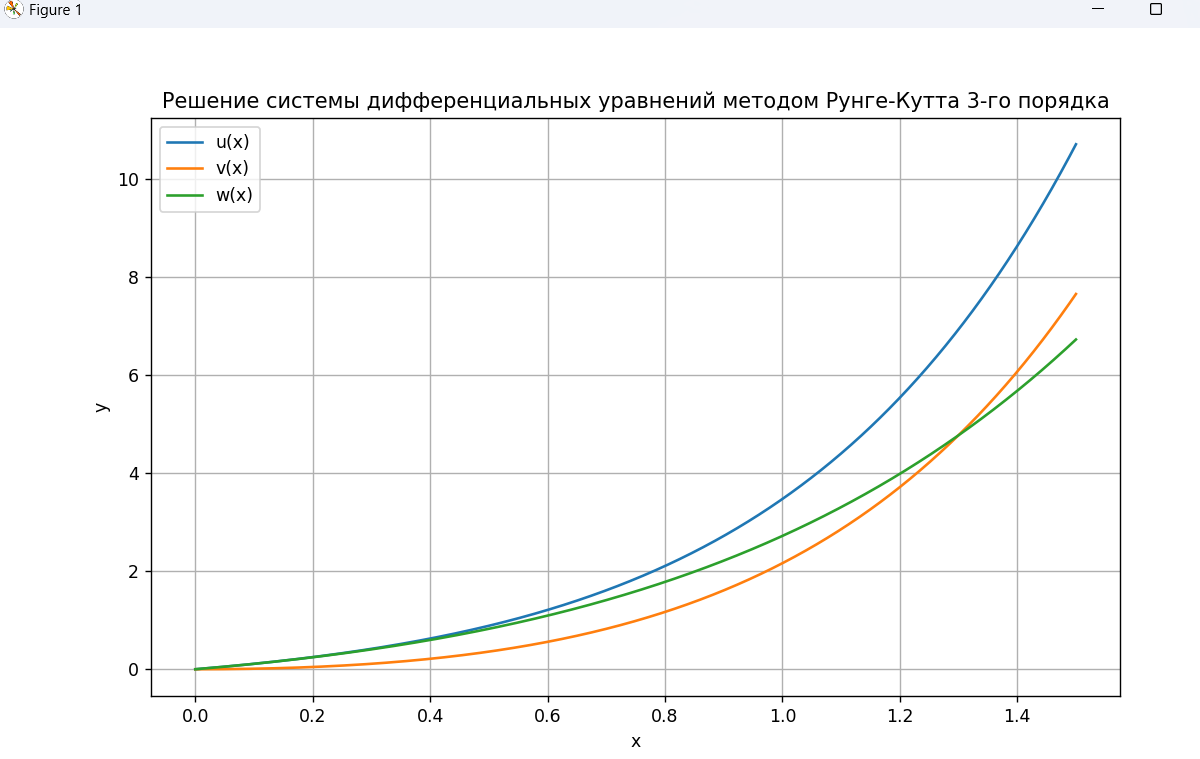
    plt.legend()

    plt.grid(*True*)

    plt.show()

**Выходные данные.**

****

****

**Модуль разности решений в крайней правой точке интервала: [1.40001139e-05 7.83841698e-06 2.52026504e-06]**