Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования.
«Рязанский государственный радиотехнический университет имени В. Ф. Уткина»

Кафедра вычислительной и прикладной математики

Отчет
о лабораторной работе № 5
по дисциплине
«Вычислительные алгоритмы»
на тему
“Решение обыкновенных дифференциальных уравнений”

Выполнила: ст. гр. 343 Гаджиева А.В
Проверила:
доц. Проказникова Е.Н.
ас. Щенева Ю.Б.

Рязань 2025

Дата выполнения лабораторной работы: 18.04.2025

Задание:

Решить задачи 1, 2, 3:

Задача 1: Решите на отрезке [x_0, x_{end}] задачу Коши $y' = f(x, y), y(x_0) = y_0$ методом Рунге-Кутта с постоянным шагом. Вид уравнения и начальные значения заданы в таблице. Изобразите графики решений, вычисленных с шагами h, 2h и h/2

Вариант	$f(x, y, y')=0$	Начальное условие	x_{end}
0	$(e^x + 2)dy + 2e^x dx = 0$	$y(0) = \frac{1}{9}$	1.6

Задача 2: $y_1' = f_1(x, y_1, y_2),\ y_2' = f_2(x, y_1, y_2),\ y_1(a) = y_{1,0},\ y_2(a) = y_{2,0}$ на отрезке [a, b] методом Рунге-Кутта с постоянным шагом h=0.1. Изобразите графики решений, вычисленных с шагом h, 2h и h/2.

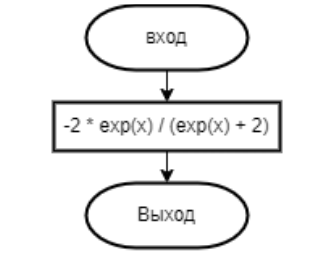
Вариант	$f_1(x, y_1, y_2)$	$f_2(x, y_1, y_2)$	$y_1(a)$	$y_2(a)$	a	b
0	$2\sqrt{3x^2 + y_1^2 + y_1}$	$\sqrt{x^2 + y_1^2 + y_2}$	1.2	1.2	0	2

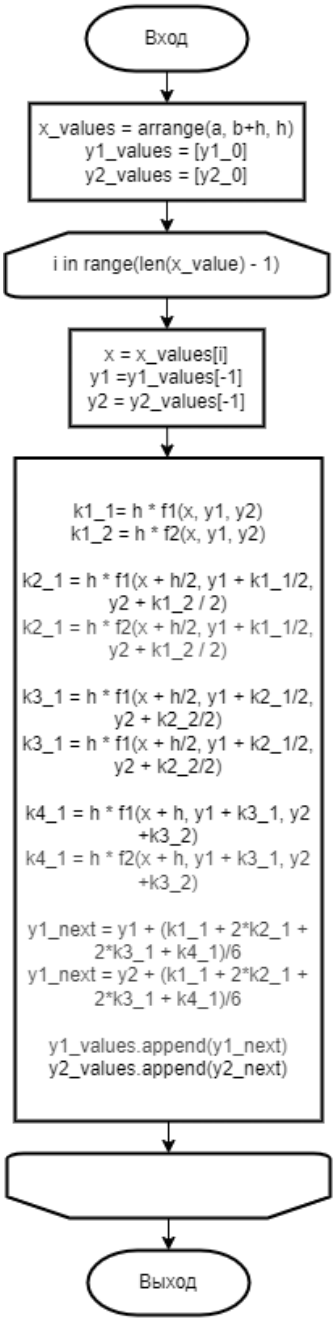
Задача 3: Найти общее решение линейного однородного уравнения второго порядка $y'' + a_1 y' + a_2 y = 0$. Решите задачу Коши, изобразите ее график.

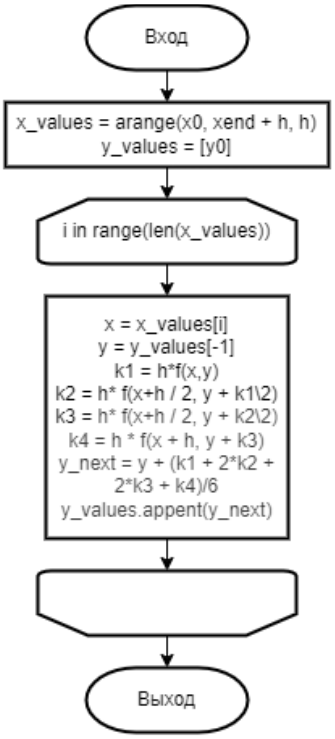
$$y'' + a_1 y' + a_2 y = 0, y(a) = y_0, y'(a) = y_1$$

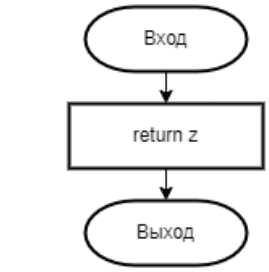
Вариант	a_1	a_2	$y(a)$	$y'(a)$	a
5	2	5	0	0	1

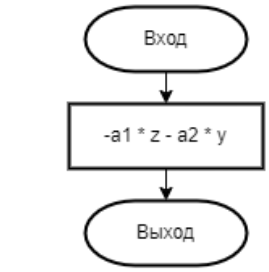
Схемы алгоритмов программы

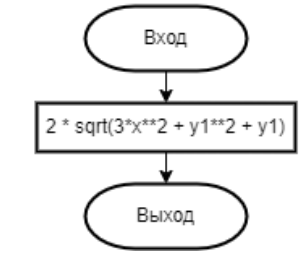


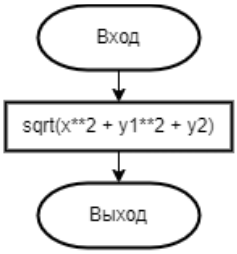


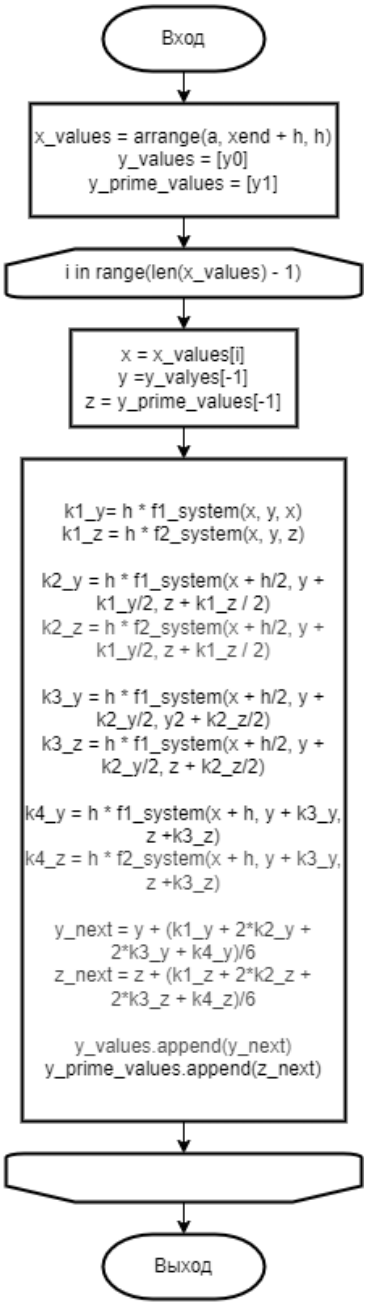












Листинг кода основных методов

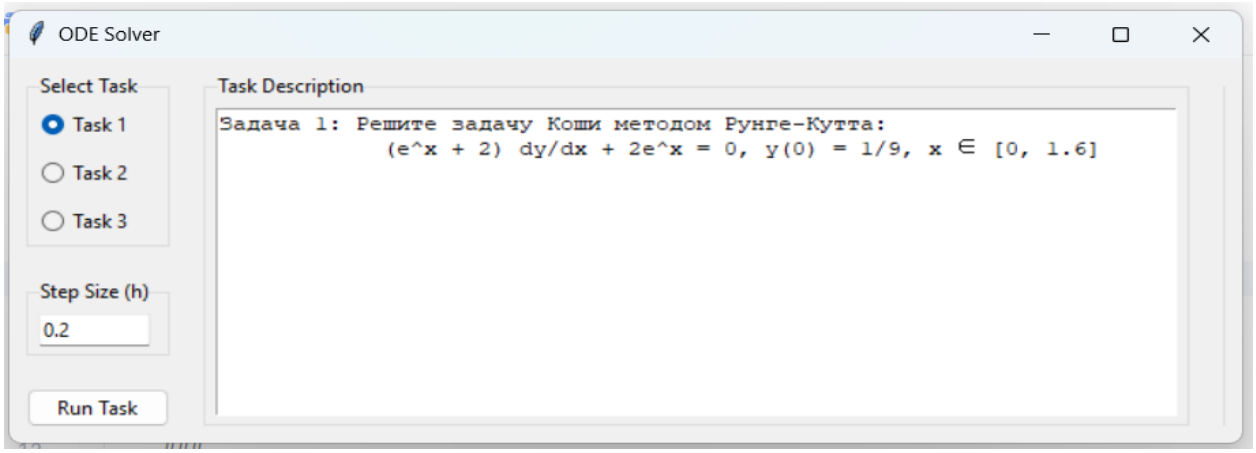
```
import numpy as np
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.figure

def f1(x, y):
    """
    Defines the ODE: (exp^x + 2)dy + (2*exp^x)dx  => dy/dx = -2*exp(x) / (exp(x) + 2)
    """
    return -2 * np.exp(x) / (np.exp(x) + 2)


def runge_kutta(f, x0, y0, xend, h):

    x_values = np.arange(x0, xend + h, h)  # Include xend in x_values
    y_values = [y0]

    for i in range(len(x_values) - 1):
        x = x_values[i]
        y = y_values[-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
        y_values.append(y_next)

    return x_values, np.array(y_values)


def task1(h=0.2):

    try:
        x0 = 0
        y0 = 1 / 9
        xend = 1.6

        fig = matplotlib.figure.Figure(figsize=(6, 4), dpi=100)  # Create a figure
        ax = fig.add_subplot(111)  # Add an axes to the figure

        # Calculate and plot for h
        x_h, y_h = runge_kutta(f1, x0, y0, xend, h)
        ax.plot(x_h, y_h, label=f'h = {h:.2f}')

        # Calculate and plot for 2h
        x_2h, y_2h = runge_kutta(f1, x0, y0, xend, 2 * h)
        ax.plot(x_2h, y_2h, label=f'2h = {2*h:.2f}')

        # Calculate and plot for h/2
        x_h_over_2, y_h_over_2 = runge_kutta(f1, x0, y0, xend, h / 2)
        ax.plot(x_h_over_2, y_h_over_2, label=f'h/2 = {h/2:.2f}')


        ax.set_xlabel('x')
        ax.set_ylabel('y')
        ax.set_title('Task 1: ODE Solution')
        ax.legend()
```

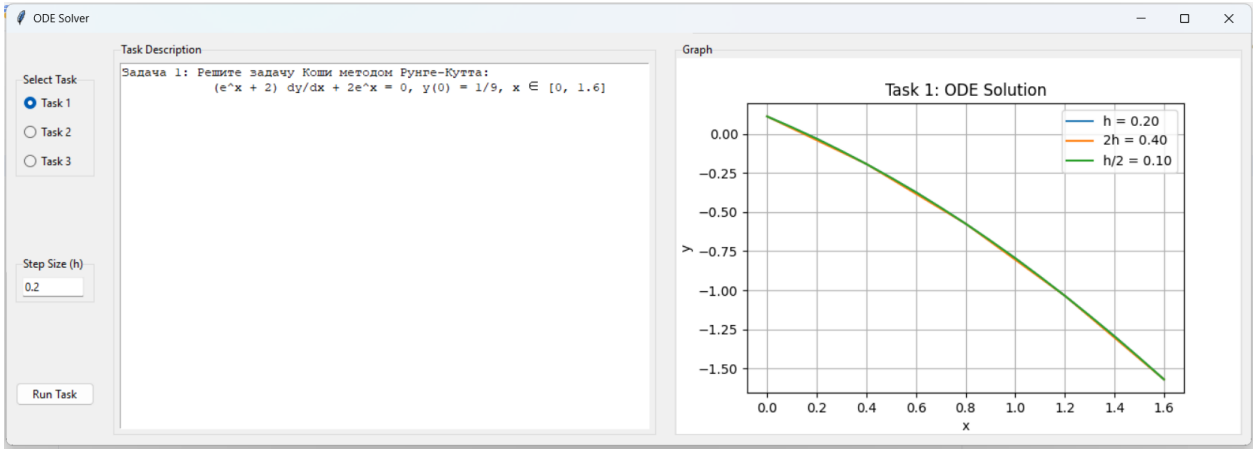
```
        ax.grid(True)
        return fig  # Return the Figure object

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred during Task 1
calculation: {e}")
        return None


# --- Task 2: System of ODEs - Runge-Kutta Method ---

def f1_system(x, y1, y2):

    return 2 * np.sqrt(3 * x**2 + y1**2 + y1)


def f2_system(x, y1, y2):

    return np.sqrt(x**2 + y1**2 + y2)


def runge_kutta_system(f1, f2, a, y1_0, y2_0, b, h):

    x_values = np.arange(a, b + h, h)
    y1_values = [y1_0]
    y2_values = [y2_0]

    for i in range(len(x_values) - 1):
        x = x_values[i]
        y1 = y1_values[-1]
        y2 = y2_values[-1]

        # Runge-Kutta steps
        k1_1 = h * f1(x, y1, y2)
        k1_2 = h * f2(x, y1, y2)

        k2_1 = h * f1(x + h / 2, y1 + k1_1 / 2, y2 + k1_2 / 2)
        k2_2 = h * f2(x + h / 2, y1 + k1_1 / 2, y2 + k1_2 / 2)

        k3_1 = h * f1(x + h / 2, y1 + k2_1 / 2, y2 + k2_2 / 2)
        k3_2 = h * f2(x + h / 2, y1 + k2_1 / 2, y2 + k2_2 / 2)

        k4_1 = h * f1(x + h, y1 + k3_1, y2 + k3_2)
        k4_2 = h * f2(x + h, y1 + k3_1, y2 + k3_2)

        y1_next = y1 + (k1_1 + 2 * k2_1 + 2 * k3_1 + k4_1) / 6
        y2_next = y2 + (k1_2 + 2 * k2_2 + 2 * k3_2 + k4_2) / 6

        y1_values.append(y1_next)
        y2_values.append(y2_next)

    return x_values, np.array(y1_values), np.array(y2_values)


def task2(h=0.1):

    try:
        a = 0
        y1_0 = 1.2
        y2_0 = 1.2
        b = 2

        fig = matplotlib.figure.Figure(figsize=(6, 4), dpi=100)  # Create a
```

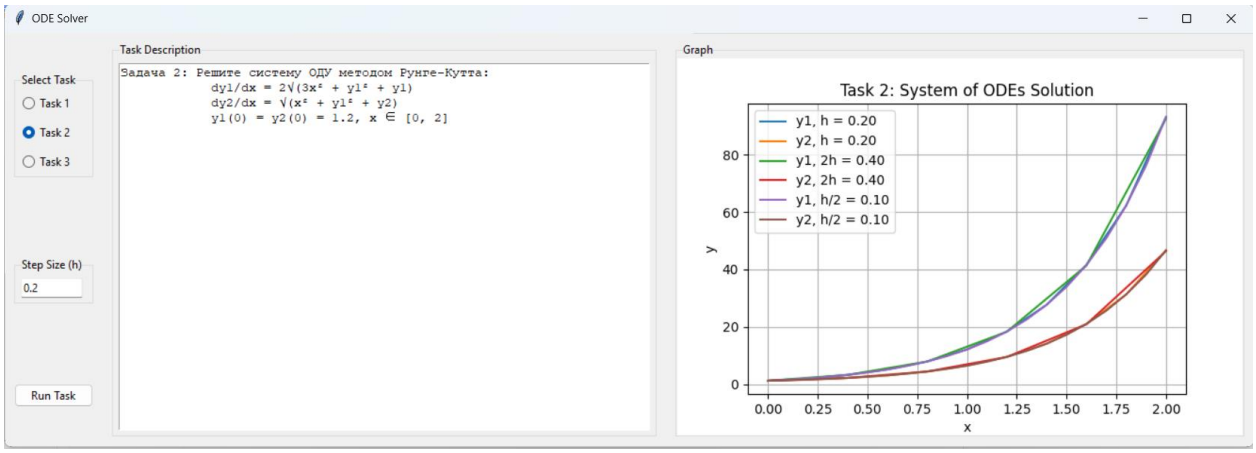
```
figure
        ax = fig.add_subplot(111)  # Add an axes to the figure


        # Calculate and plot for h
        x_h, y1_h, y2_h = runge_kutta_system(f1_system, f2_system, a, y1_0,
y2_0, b, h)
        ax.plot(x_h, y1_h, label=f'y1, h = {h:.2f}')
        ax.plot(x_h, y2_h, label=f'y2, h = {h:.2f}')

        # Calculate and plot for 2h
        x_2h, y1_2h, y2_2h = runge_kutta_system(f1_system, f2_system, a,
y1_0, y2_0, b, 2 * h)
        ax.plot(x_2h, y1_2h, label=f'y1, 2h = {2*h:.2f}')
        ax.plot(x_2h, y2_2h, label=f'y2, 2h = {2*h:.2f}')

        # Calculate and plot for h/2
        x_h_over_2, y1_h_over_2, y2_h_over_2 = runge_kutta_system(f1_system,
f2_system, a, y1_0, y2_0, b, h / 2)
        ax.plot(x_h_over_2, y1_h_over_2, label=f'y1, h/2 = {h/2:.2f}')
        ax.plot(x_h_over_2, y2_h_over_2, label=f'y2, h/2 = {h/2:.2f}')


        ax.set_xlabel('x')
        ax.set_ylabel('y')
        ax.set_title('Task 2: System of ODEs Solution')
        ax.legend()
        ax.grid(True)

        return fig # Return the Figure object

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred during Task 2
calculation: {e}")
        return None


# --- Task 3: Second-Order Linear Homogeneous ODE ---

def solve_second_order_ode(a1, a2, a, y0, y1, xend, h):

    def f1_system(x, y, z):  # z = y'
        return z  # dy/dx = y'

    def f2_system(x, y, z):
        return -a1 * z - a2 * y  # dz/dx = y'' = -a1*y' - a2*y

    x_values = np.arange(a, xend + h, h)
    y_values = [y0]
    y_prime_values = [y1]

    for i in range(len(x_values) - 1):
        x = x_values[i]
        y = y_values[-1]
        z = y_prime_values[-1]

        # Runge-Kutta steps (for a system of two equations)
        k1_y = h * f1_system(x, y, z)
        k1_z = h * f2_system(x, y, z)

        k2_y = h * f1_system(x + h / 2, y + k1_y / 2, z + k1_z / 2)
        k2_z = h * f2_system(x + h / 2, y + k1_y / 2, z + k1_z / 2)
```

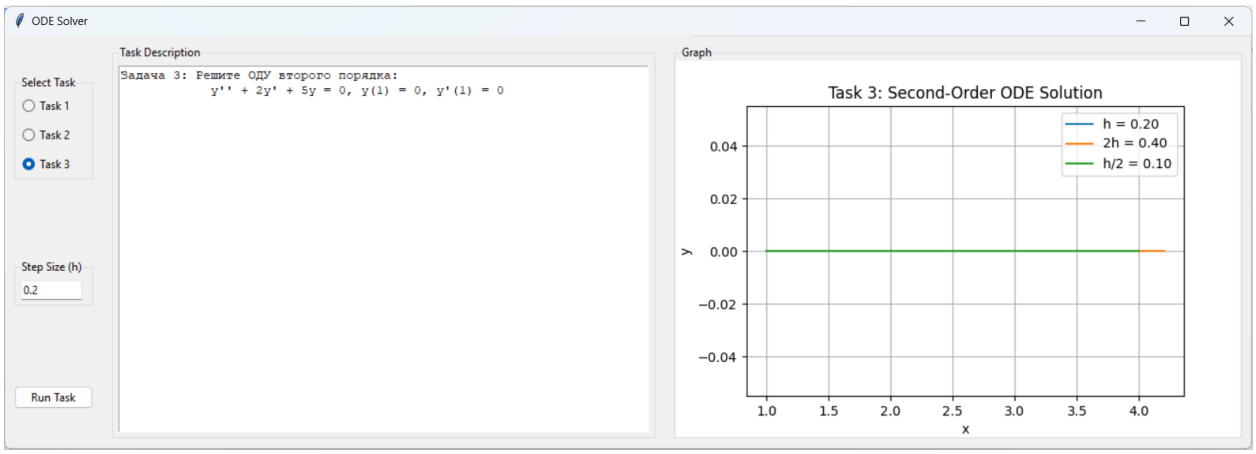
```
        k3_y = h * f1_system(x + h / 2, y + k2_y / 2, z + k2_z / 2)
        k3_z = h * f2_system(x + h / 2, y + k2_y / 2, z + k2_z / 2)

        k4_y = h * f1_system(x + h, y + k3_y, z + k3_z)
        k4_z = h * f2_system(x + h, y + k3_y, z + k3_z)

        y_next = y + (k1_y + 2 * k2_y + 2 * k3_y + k4_y) / 6
        z_next = z + (k1_z + 2 * k2_z + 2 * k3_z + k4_z) / 6

        y_values.append(y_next)
        y_prime_values.append(z_next)

    return x_values, np.array(y_values), np.array(y_prime_values)


def task3(h=0.2):

    try:
        a1 = 2
        a2 = 5
        a = 1
        y0 = 0
        y1 = 0
        xend = 4  # Adjust as needed.

        fig = matplotlib.figure.Figure(figsize=(6, 4), dpi=100)  # Create a
figure
        ax = fig.add_subplot(111)  # Add an axes to the figure

        # Calculate and plot for h
        x_h, y_h, yp_h = solve_second_order_ode(a1, a2, a, y0, y1, xend, h)
        ax.plot(x_h, y_h, label=f'h = {h:.2f}')

        # Calculate and plot for 2h
        x_2h, y_2h, yp_2h = solve_second_order_ode(a1, a2, a, y0, y1, xend, 2
* h)
        ax.plot(x_2h, y_2h, label=f'2h = {2*h:.2f}')

        # Calculate and plot for h/2
        x_h_over_2, y_h_over_2, yp_h_over_2 = solve_second_order_ode(a1, a2,
a, y0, y1, xend, h/2)
        ax.plot(x_h_over_2, y_h_over_2, label=f'h/2 = {h/2:.2f}')

        ax.set_xlabel('x')
        ax.set_ylabel('y')
        ax.set_title('Task 3: Second-Order ODE Solution')
        ax.legend()
        ax.grid(True)

        return fig  # Return the Figure object

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred during Task 3
calculation: {e}")
        return None
```

Графический интерфейс программы





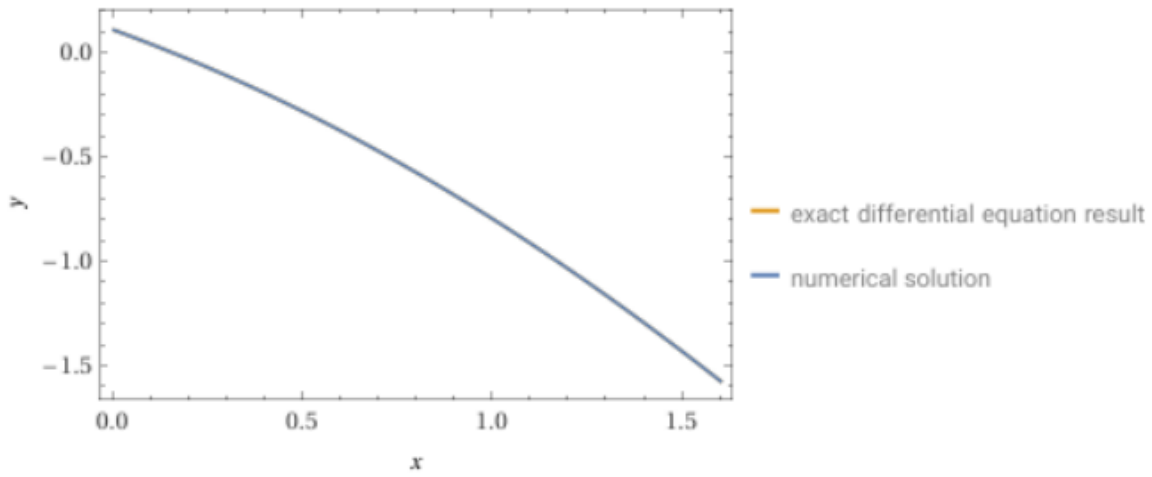




Проверка правильности работы программы

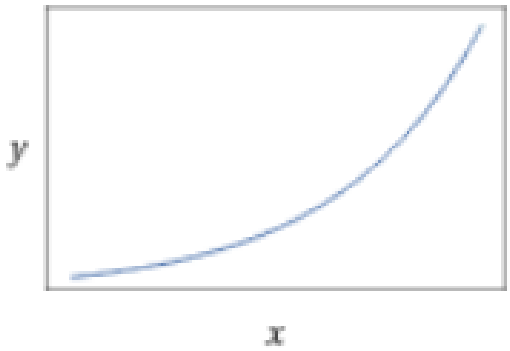
Задание 1

Solution plot



Задание 2

Plots of sample individual s



Задание 3

Differential equation solution

$$y(x) = 0$$

Differential equation infinite series expansion

$$\sum_{n=0}^{\infty} 0\, x^n$$

(unable to determine the interval of convergence)