МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

Институт информационных технологий, математики и механики

Направление подготовки: «Прикладная математика и информатика»
Профиль подготовки: «Вычислительные методы и суперкомпьютерные технологии»

Отчет
по лабораторной работе №3
по курсу «Прикладная нелинейная динамика»

Выполнил:
студент группы 3823М1ПМвм
Бекетов Е.В.

Проверил:
д.ф.-м.н., доц., зав.каф. ПМ
Иванченко М.В.

Нижний Новгород
2024

1. Численное интегрирование

Необходимо решить численно дифференциальное уравнение вида $\dot{x}(t) = f(x,t)$ с начальным условием $x(0) = x_0$. Для этого можно использовать метод Эйлера, он достаточно прост и у него небольшая вычислительная сложность. Шаг вычисления нового значения выглядит следующим образом:

$$x_{i+1} = x_i + hf(x_i, h \cdot i)$$

где $h > 0$ – шаг метода, а $n = \overline{0, Nmax}$, $Nmax$ – число итераций метода.

Так же можно использовать метод Рунге-Кутты 4 порядка, он так же достаточно прост, но в сравнении с методом Эйлера у него выросла вычислительная сложность. Шаг вычисления нового значения выглядит следующим образом:

$$k_1 = hf(t_i, x_i)$$
$$k_2 = hf(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_1)$$
$$k_3 = hf(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_2)$$
$$k_4 = hf(t_i + h, x_i + k - 3)$$
$$x_{i+1} = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

где $h > 0$ – шаг метода, а $n = \overline{0, Nmax}$, $Nmax$ – число итераций метода.

Опробуем методы на 4 задачах, аналитические решения первых трех задач известны и разность между численным и аналитическим решением будет погрешностью численного решения. В последней задаче аналитическое решение не известно, погрешностью будем считать разность численного решения с шагом h и $h/2$.

$$\begin{cases} \dot{x} = -x \\ x(0) = 2 \end{cases} \quad x(t) = x(0)e^{-t} \tag{1}$$

$$\begin{cases} \dot{x} = x \\ x(0) = 2 \end{cases} \quad x(t) = x(0)e^{t} \tag{2}$$

$$\begin{cases} \ddot{x} + x = 0 \\ x(0) = 2 \\ \dot{x}(0) = 3 \end{cases} \quad x(t) = x(0)\cos(t) + \dot{x}(0)\sin(t) \tag{3}$$

$$\begin{cases} \dot{x} = -y - z \\ \dot{y} = x + 0.3y \\ \dot{z} = 0.3 + z(x - 5.7) \\ x(0) = 10,\ y(0) = 10,\ z(0) = 10 \end{cases} \tag{4}$$

2. Численное интегрирование уравнений методом Эйлера с постоянным шагом

Рассмотрим результаты интегрирования задач (1) и (2) с интервалом $t \in [0; 100]$, с разными шагами h. На Рис. 1 и Рис. 2 видно, что уточнение шага уменьшает погрешность, что достаточно логично и соответствует теории, а если посмотреть на Рис. 3 с интервалом $t \in [0; 10]$, то видно что у метода Эйлера первый порядок сходимости, т.к. при уменьшении шага в 10 раз ошибка так же падает в 10 раз.

Аналогичная ситуация наблюдается в задачах (3) и (4) Рис. 4 и Рис. 5 (время сокращено до 5, т.к. до этого момента кривые выглядят наиболее наглядно). Однако глядя на погрешность однозначно можно сказать, что для данных задач метод Эйлера не лучшее решение. Возможно метод Рунге-Кутты покажет лучше результаты.

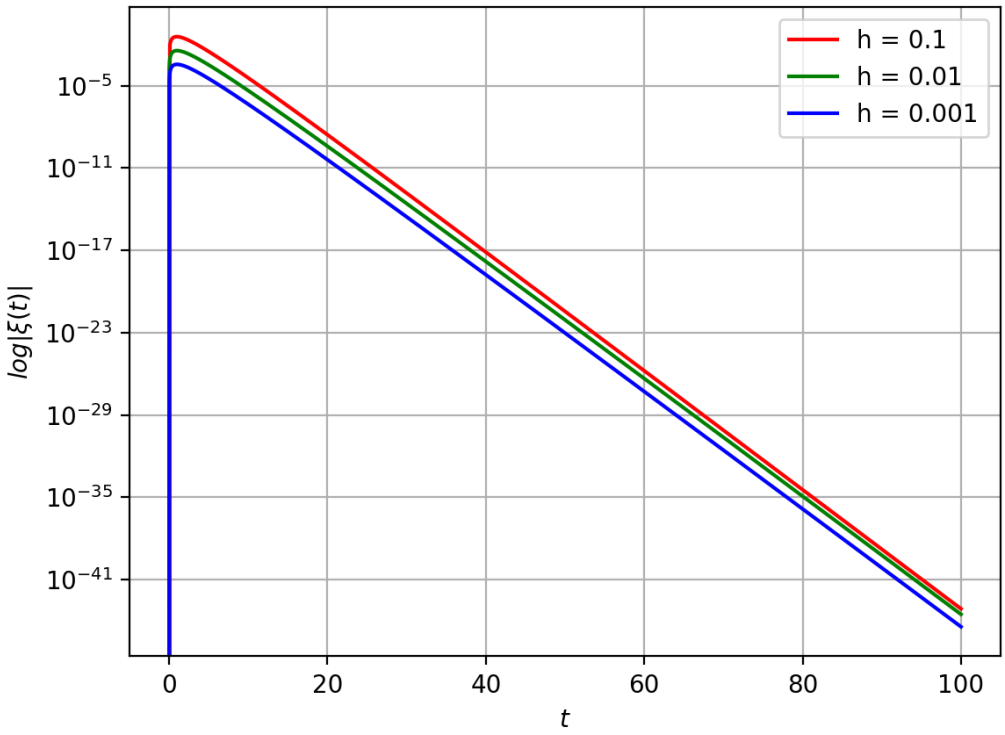

Рис. 1: Ошибка интегрирования уравнения (1) методом Эйлера при различных значениях шага.

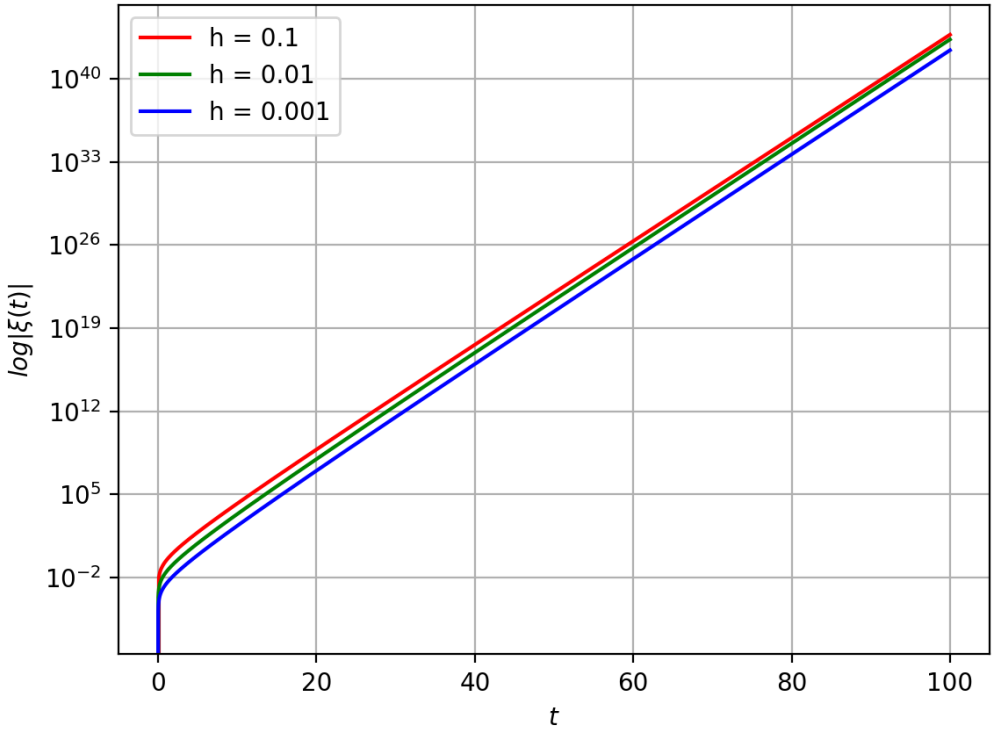

Рис. 2: Ошибка интегрирования уравнения (2) методом Эйлера при различных значениях шага.

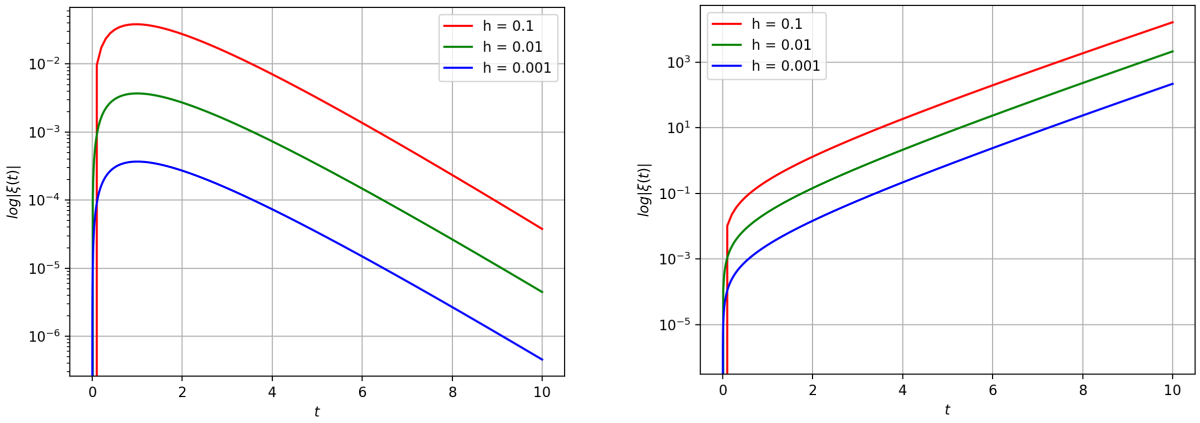

Рис. 3: Ошибка интегрирования уравнения (1) и (2) методом Эйлера при различных значениях шага.

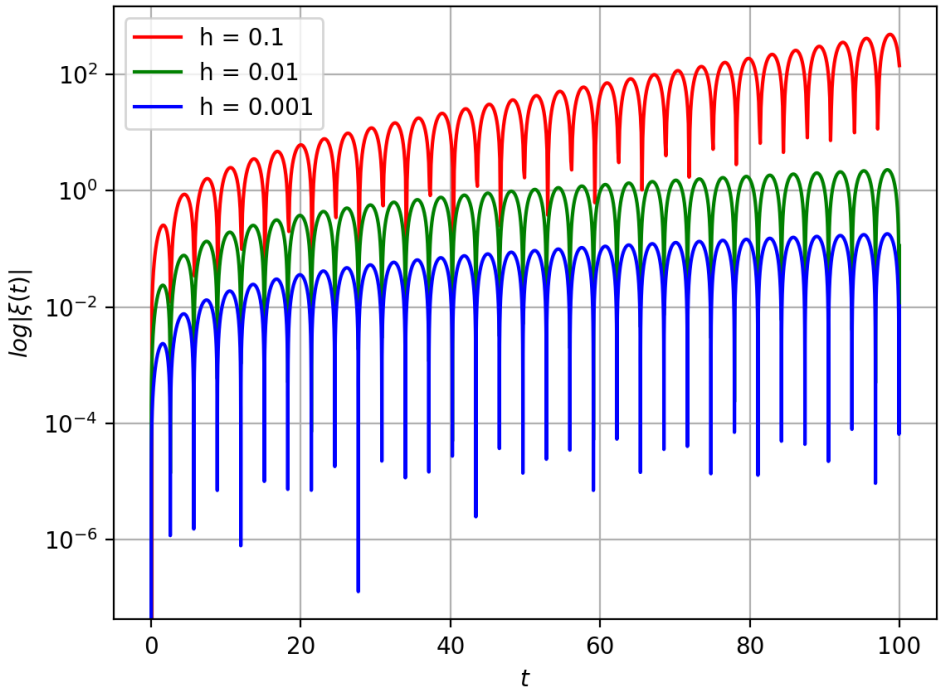

Рис. 4: Ошибка интегрирования уравнения (3) методом Эйлера при различных значениях шага.

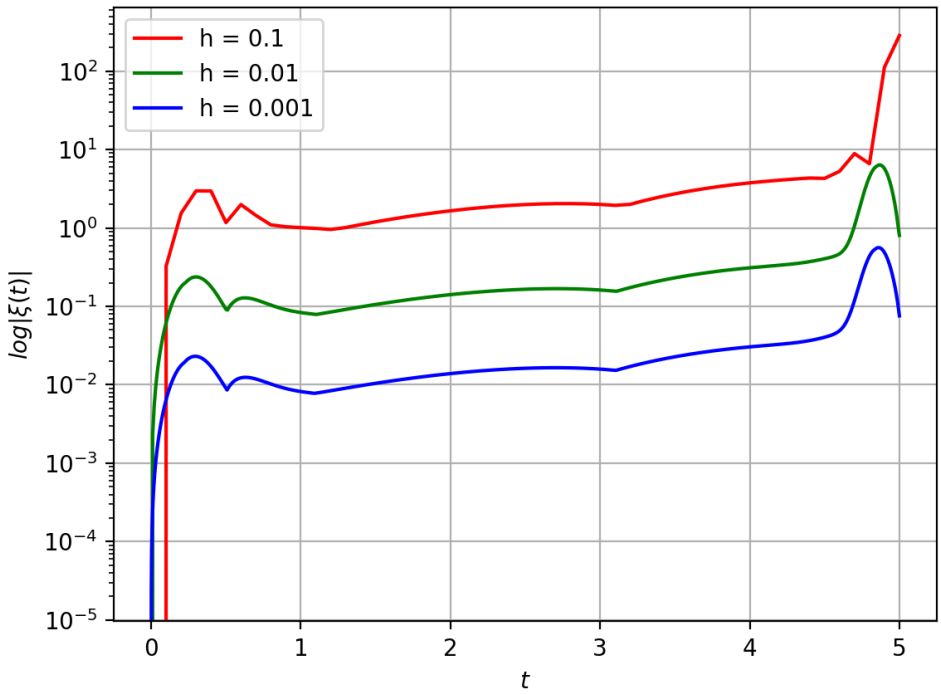

Рис. 5: Ошибка интегрирования уравнения (4) методом Эйлера при различных значениях шага.

3. Численное интегрирование уравнений методом Рунге-Кутты с постоянным шагом

Рассмотрим результаты интегрирования задач (1) и (2) с интервалом $t \in [0; 100]$, с разными шагами h. На Рис. 6 и Рис. 7 видно, что уточнение шага уменьшает погрешность, что достаточно логично и соответствует теории, а если посмотреть на Рис. 8, то видно что у метода Рунге-Кутта четвертый порядок сходимости, т.к. при уменьшении шага в 10 раз ошибка падает в 10^4 раз.

Аналогичная ситуация наблюдается в задачах (3) и (4) Рис. 9 и Рис. 10. Как и ожидалось, метод Рунге-Кутты показал меньшую ошибку в задаче (4), однако она тоже растет.

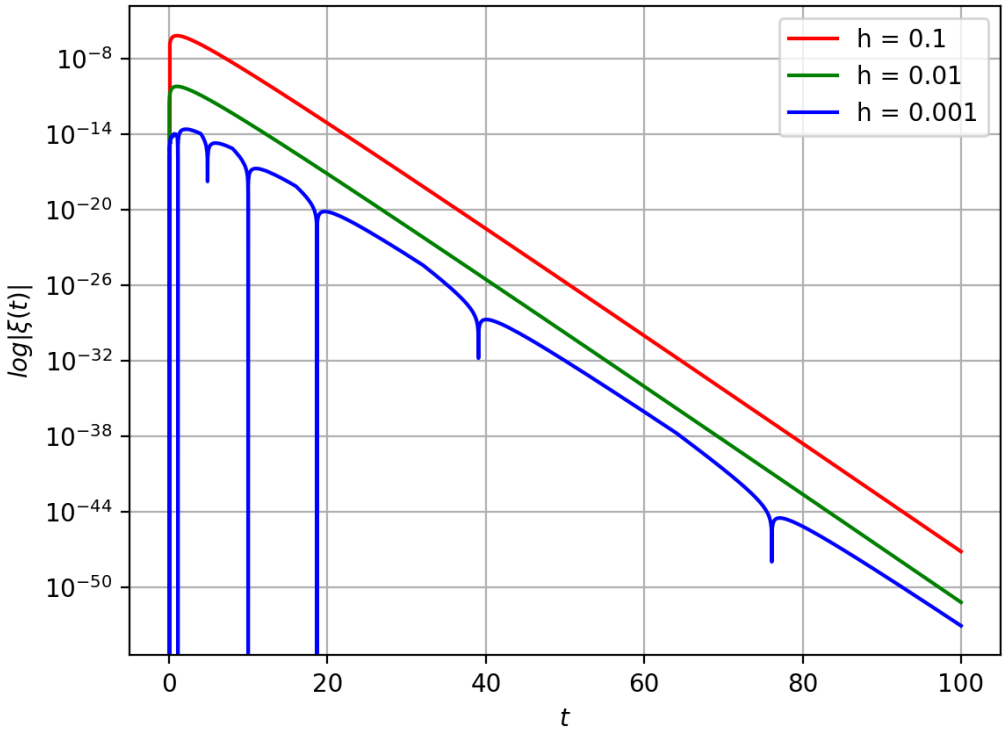

Рис. 6: Ошибка интегрирования уравнения (1) методом РК4 при различных значениях шага.

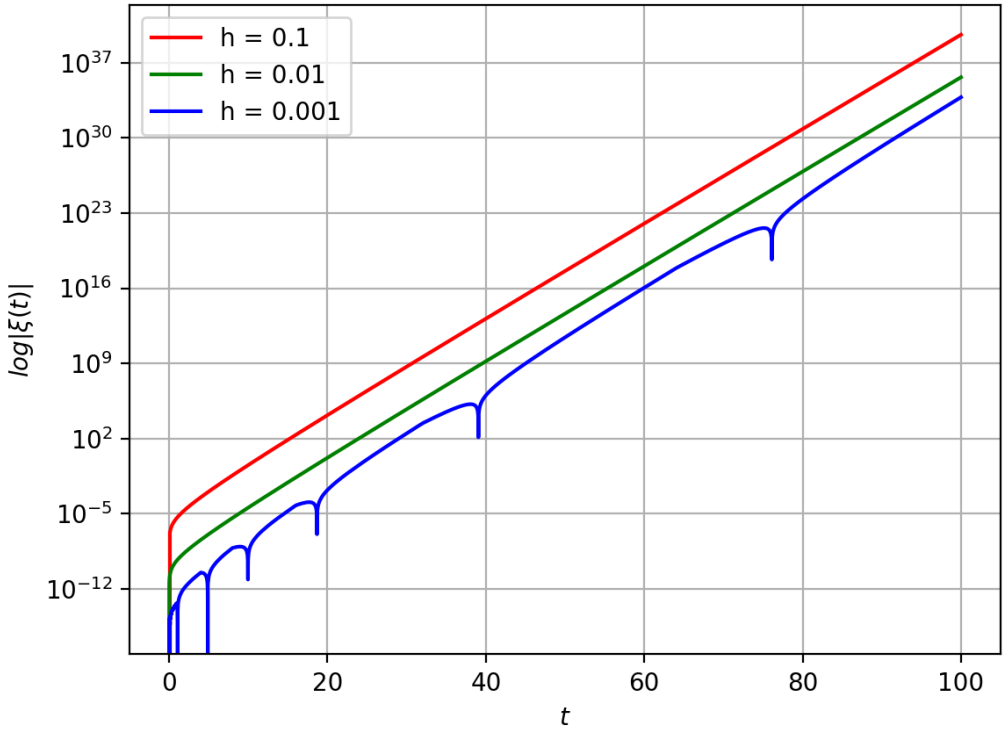

Рис. 7: Ошибка интегрирования уравнения (2) методом РК4 при различных значениях шага.

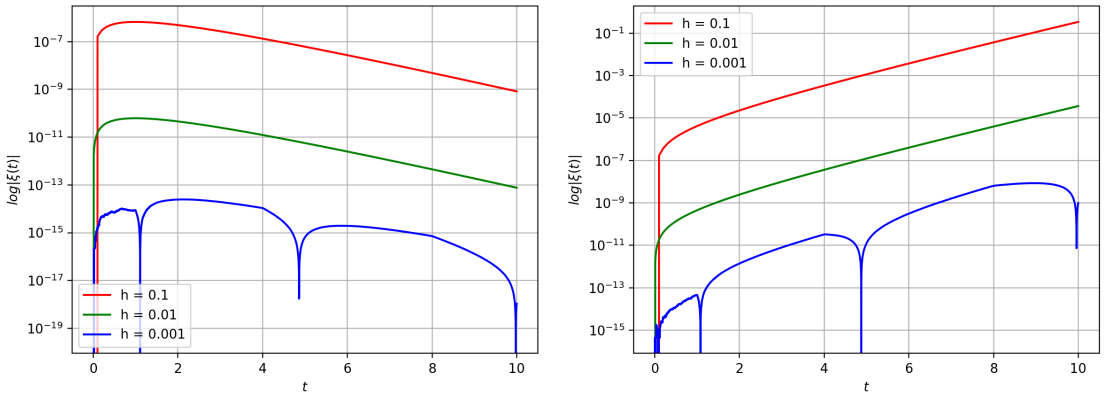

Рис. 8: Ошибка интегрирования уравнения (1) и (2) методом РК4 при различных значениях шага.

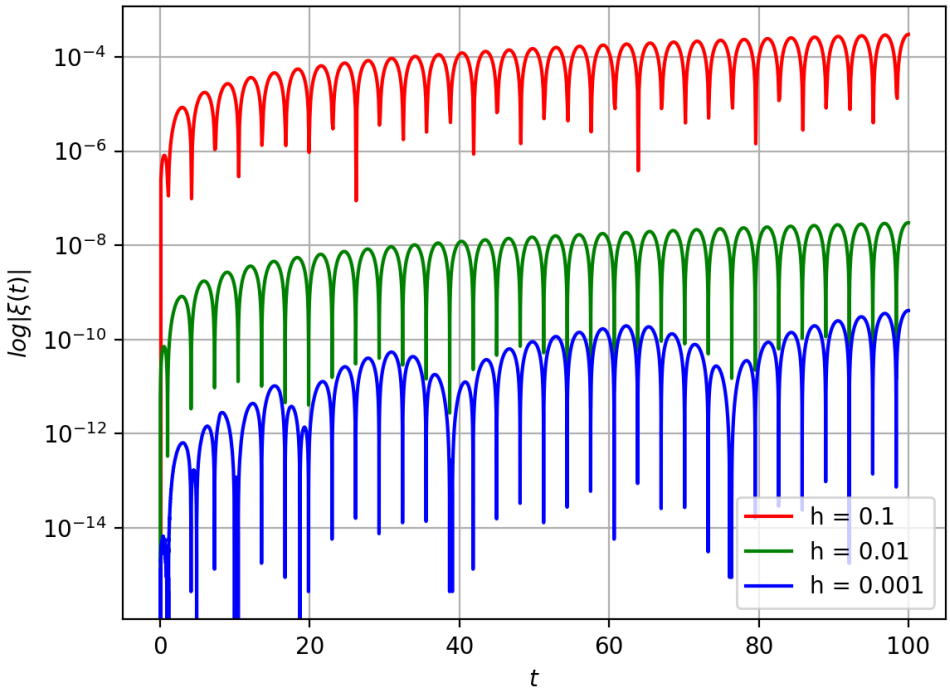

Рис. 9: Ошибка интегрирования уравнения (3) методом РК4 при различных значениях шага.

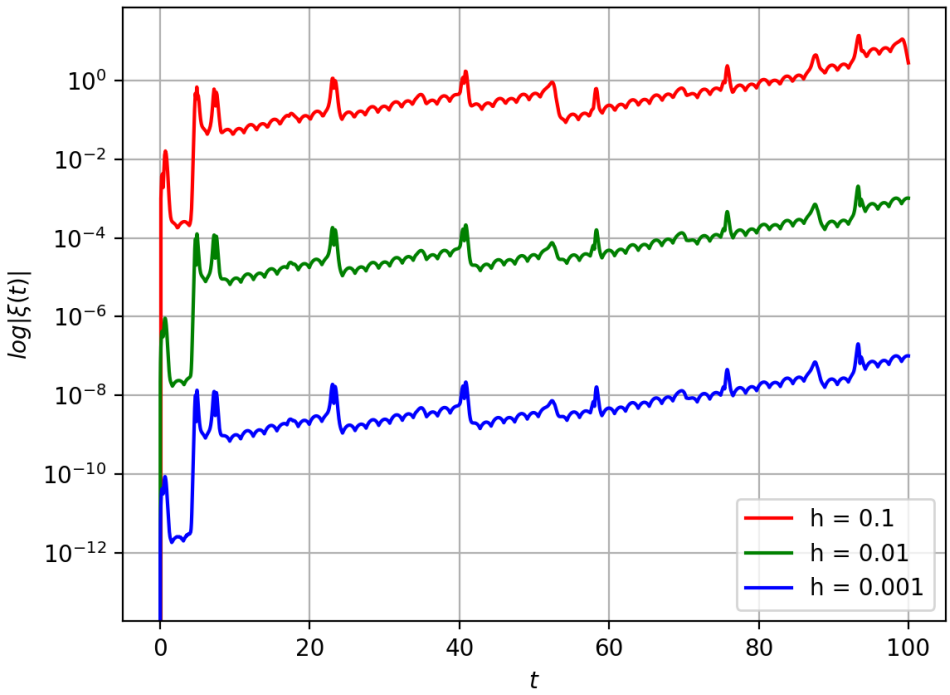

Рис. 10: Ошибка интегрирования уравнения (4) методом РК4 при различных значениях шага.

4. Вывод

В конечном итоге был написан скрипт на языке Python, который наглядно показал порядок ошибки метода Эйлера и Рунге-Кутты, на 4 задачах.

5. Приложение – Код программы

```
import numpy as np
import matplotlib.pyplot as plt

#! Глобальные переменные
eps = 0.0001
a = 0.2
b = 0.2
c = 5.7
T = 100
N = 1000
steps = [10e-2, 10e-3, 10e-4]
colors = ['r', 'g', 'b']

def equation1(x, t):
    return -x

def equation2(x, t):
    return x

def equation3(x, t):
    return np.array([x[1], -x[0]])

def equation4(x, t):
    return np.array([-x[1] - x[2], x[0] + a*x[1], b + x[2]*(x[0] - c)])

def euler(func, start_t, end_t, init, h):
    num_steps = int((end_t - start_t) / h)
    x_path = [0]*(num_steps + 1)
    t_path = [0]*(num_steps + 1)
    x_path[0] = init
    t_path[0] = start_t
    for i in range(len(t_path) - 1):
        x_path[i + 1] = x_path[i] + h*func(x_path[i], t_path[i])
        t_path[i + 1] = t_path[i] + h
    t_path[-1] = end_t
    x_path[-1] = x_path[-2] + h*func(x_path[-2], end_t)
    return np.array(t_path), np.array(x_path)

def RK4(func, start_t, end_t, init, h):
    num_steps = int((end_t - start_t) / h)
    x_path = [0.]*(num_steps + 1)
    t_path = [0.]*(num_steps + 1)
    x_path[0] = init
    t_path[0] = start_t
    for i in range(len(t_path) - 1):
        k1 = func(x_path[i], t_path[i])
        k2 = func(x_path[i] + 0.5*h*k1, t_path[i] + 0.5*h)
        k3 = func(x_path[i] + 0.5*h*k2, t_path[i] + 0.5*h)
```

```
        k4 = func(x_path[i] + h*k3, t_path[i] + h)
        x_path[i + 1] = x_path[i] + h*(k1 + 2*k2 + 2*k3 + k4) / 6
        t_path[i + 1] = t_path[i] + h
    return np.array(t_path), np.array(x_path)

def error_plot(t, err, name):
    plt.yscale('log')
    for i, step in enumerate(steps):
        plt.plot(t[i], err[i], colors[i], label = 'h = ' + str(step))
    plt.xlabel('$t$')
    plt.ylabel('$log|\\xi(t)|$')
    plt.grid()
    plt.legend(loc = 'best', fontsize = 10)
    plt.savefig(name, dpi = 200)
    plt.cla()
    #plt.show()

def main():
    t_start = 0
    t_stop = 100

    #! Система 1
    err_values = []
    t_values = []
    x0 = 2
    for h in steps:
        t, x = RK4(equation1, t_start, t_stop, x0, h)
        err_values.append(np.abs(x - x0*np.exp(-t)))
        t_values.append(t)
    error_plot(t_values, err_values, 'lab3_equation1_rk.png')

    #! Система 2
    err_values = []
    t_values = []
    x0 = 2
    for h in steps:
        t, x = RK4(equation2, t_start, t_stop, x0, h)
        err_values.append(np.abs(x - x0*np.exp(t)))
        t_values.append(t)
    error_plot(t_values, err_values, 'lab3_equation2_rk.png')

    #! Система 3
    err_values = []
    t_values = []
    x0 = np.array([2, 3])
    for h in steps:
        t, x = RK4(equation3, t_start, t_stop, x0, h)
        err_values.append(np.abs(x[:, 0] - (x0[0]*np.cos(t) + \
        x0[1]*np.sin(t))))
        t_values.append(t)
```

```
    error_plot(t_values, err_values, 'lab3_equation3_rk.png')

    #! Система 4
    err_values = []
    t_values = []
    x0 = np.array([10, 10, 10])
    for h in steps:
        t1, x1 = RK4(equation4, t_start, t_stop, x0, h)
        t2, x2 = RK4(equation4, t_start, t_stop, x0, h/2)
        err_values.append(np.sum(np.abs(x1 - x2[::2]), axis = 1))
        t_values.append(t1)
    error_plot(t_values, err_values, 'lab3_equation4_rk.png')

if __name__ == '__main__':
    main()
```