Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО (НИИ ИТМО)

Факультет систем управления и робототехники

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине

«Имитационное моделирование робототехнических систем»

ПРОВЕРКА РЕШЕНИЯ ОДУ ЧИСЛЕННЫМИ МЕТОДАМИ

15.04.06 «Робототехника и ИИ»

Выполнила: Шульга И.Д. (468129)
Группа: R4133c
Преподаватель: Ракшин Е.А.
Дата выполнения: 07 ноября 2025 г.

г. Санкт-Петербург
2025 год

1 Цель работы

Решить дифференциальное уравнение второго порядка с использованием трёх численных методов интегрирования: метода явного Эйлера, метода неявного Эйлера и метода Рунге-Кутты четвертого порядка. Провести сравнительный анализ точности методов и визуализировать полученные результаты.

2 Задачи

1. Реализовать в рамках ПО Python решение дифференциального уравнения с использованием трёх методов:
   - Метод явного Эйлера
   - Метод неявного Эйлера
   - Метод Рунге-Кутты 4-го порядка
2. Построить графики решений для визуального сравнения методов:
   - График зависимости угла от времени
   - График зависимости угловой скорости от времени
   - Фазовый портрет системы
3. Провести анализ точности методов и сделать выводы об их эффективности
4. Сравнить численные решения с аналитическим решением

3 Теоретическая часть и расчёты

3.1 Постановка задачи

Рассматривается линейное обыкновенное дифференциальное уравнение второго порядка вида:

$$a \cdot \ddot{x} + b \cdot \dot{x} + c \cdot x = d \tag{1}$$

где:

- $a = -1.1$ - коэффициент при второй производной
- $b = -7.04$ - коэффициент при первой производной
- $c = 1.46$ - коэффициент при функции
- $d = -3.19$ - свободный член

3.2 Начальные условия и параметры

- Начальное положение: $x(0) = 0.1$ рад
- Начальная скорость: $\dot{x}(0) = 0.0$ рад/с
- Время моделирования: $T_f = 10.0$ с
- Шаг интегрирования: $h = 0.01$ с

3.3 Приведение к системе первого порядка

Для численного решения методом пространства состояний приведём исходное уравнение к системе дифференциальных уравнений первого порядка. Введём вектор состояния:

$$\mathbf{s} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{2}$$

Тогда система уравнений принимает вид:

$$\frac{d\mathbf{s}}{dt} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} s_2 \\ \dfrac{d - b \cdot s_2 - c \cdot s_1}{a} \end{bmatrix} \tag{3}$$

где $s_1 = x$, $s_2 = \dot{x}$.

3.4 Численные методы решения

3.4.1 Метод явного Эйлера

Метод явного Эйлера - простейший метод первого порядка точности:

$$\mathbf{s}_{n+1} = \mathbf{s}_n + h \cdot f(t_n, \mathbf{s}_n) \tag{4}$$

где $f(t, \mathbf{s})$ - правая часть системы дифференциальных уравнений.

3.4.2 Метод неявного Эйлера

Метод неявного Эйлера - метод первого порядка точности с улучшенной устойчивостью:

$$\mathbf{s}_{n+1} = \mathbf{s}_n + h \cdot f(t_{n+1}, \mathbf{s}_{n+1}) \tag{5}$$

Для решения нелинейного уравнения на каждом шаге используется метод простой итерации.

3.4.3 Метод Рунге-Кутты 4-го порядка

Метод Рунге-Кутты 4-го порядка - метод повышенной точности:

$$k_1 = f(t_n, \mathbf{s}_n) \tag{6}$$

$$k_2 = f(t_n + \frac{h}{2}, \mathbf{s}_n + \frac{h}{2}k_1) \tag{7}$$

$$k_3 = f(t_n + \frac{h}{2}, \mathbf{s}_n + \frac{h}{2}k_2) \tag{8}$$

$$k_4 = f(t_n + h, \mathbf{s}_n + hk_3) \tag{9}$$

$$\mathbf{s}_{n+1} = \mathbf{s}_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{10}$$

3.5 Аналитическое решение

Для верификации результатов численного моделирования проведём аналитическое решение исходного дифференциального уравнения.

3.5.1 Исходное уравнение и приведение к стандартной форме

Исходное уравнение:

$$a \cdot \ddot{x} + b \cdot \dot{x} + c \cdot x = d \tag{11}$$

С коэффициентами: $a = -1.1$, $b = -7.04$, $c = 1.46$, $d = -3.19$.
Приведём уравнение к стандартной форме, разделив на коэффициент a:

$$\ddot{x} + \frac{b}{a}\dot{x} + \frac{c}{a}x = \frac{d}{a} \tag{12}$$

Подставляя численные значения:

$$\ddot{x} + \frac{-7.04}{-1.1}\dot{x} + \frac{1.46}{-1.1}x = \frac{-3.19}{-1.1} \tag{13}$$

Упрощаем коэффициенты:

$$\ddot{x} + 6.4\dot{x} - 1.32727x = 2.9 \tag{14}$$

3.5.2 Решение однородного уравнения

Рассмотрим однородное уравнение:

$$\ddot{x} + 6.4\dot{x} - 1.32727x = 0 \tag{15}$$

Характеристическое уравнение:

$$r^2 + 6.4r - 1.32727 = 0 \tag{16}$$

Находим корни характеристического уравнения:

$$r_{1,2} = \frac{-6.4 \pm \sqrt{6.4^2 + 4 \cdot 1.32727}}{2} = \frac{-6.4 \pm \sqrt{40.96 + 5.30908}}{2} = \frac{-6.4 \pm \sqrt{46.26908}}{2} \tag{17}$$

$$r_1 \approx \frac{-6.4 + 6.802}{2} \approx 0.201, \quad r_2 \approx \frac{-6.4 - 6.802}{2} \approx -6.601 \tag{18}$$

Общее решение однородного уравнения:

$$x_h(t) = C_1 e^{0.201t} + C_2 e^{-6.601t} \tag{19}$$

3.5.3 Частное решение неоднородного уравнения

Поскольку правая часть уравнения - константа, частное решение ищем в виде:

$$x_p = A \quad \text{(константа)} \tag{20}$$

Подставляем в исходное уравнение:

$$0 + 6.4 \cdot 0 - 1.32727 \cdot A = 2.9 \tag{21}$$

$$A = \frac{2.9}{-1.32727} \approx -2.185 \tag{22}$$

3.5.4 Общее решение и определение констант

Общее решение неоднородного уравнения:

$$x(t) = x_h(t) + x_p = C_1 e^{0.201t} + C_2 e^{-6.601t} - 2.185 \tag{23}$$

Определим константы C_1 и C_2 из начальных условий: $x(0) = 0.1$, $\dot{x}(0) = 0$.
Из условия $x(0) = 0.1$:

$$C_1 + C_2 - 2.185 = 0.1 \Rightarrow C_1 + C_2 = 2.285 \tag{24}$$

Находим производную:

$$\dot{x}(t) = 0.201 C_1 e^{0.201t} - 6.601 C_2 e^{-6.601t} \tag{25}$$

Из условия $\dot{x}(0) = 0$:

$$0.201 C_1 - 6.601 C_2 = 0 \tag{26}$$

Решаем систему уравнений:

$$C_1 + C_2 = 2.285 \tag{27}$$

$$0.201 C_1 - 6.601 C_2 = 0 \tag{28}$$

Из второго уравнения: $C_1 = \frac{6.601}{0.201} C_2 \approx 32.841 C_2$. Подставляем в первое:

$$32.841 C_2 + C_2 = 2.285 \Rightarrow 33.841 C_2 = 2.285 \Rightarrow C_2 \approx 0.0675 \tag{29}$$

Тогда:

$$C_1 = 2.285 - 0.0675 \approx 2.2175 \tag{30}$$

3.5.5 Окончательное аналитическое решение

$$x(t) = 2.2175 e^{0.201t} + 0.0675 e^{-6.601t} - 2.185 \tag{31}$$

3.5.6 Анализ устойчивости решения

Наличие положительного корня $r_1 \approx 0.201$ в характеристическом уравнении указывает на неустойчивость системы. Решение содержит растущую экспоненту $e^{0.201t}$, что объясняет экспоненциальный рост амплитуды колебаний, наблюдаемый в численных экспериментах.

4 Реализация кода в Python

4.1 Основной код

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

def pendulum_dynamics(s):
    """
    ODE: a*d x/dt +b*dx/dt+c*x=d
    State vector: s=[x, dx/dt]
    """
    a = -1.1
    b = -7.04
    c = 1.46
    d = -3.19

    x = s[0]
    x_dot = s[1]
    x_ddot = (d - b*x_dot - c*x)/a

    return np.array([x_dot, x_ddot])

def forward_euler(fun, x0, Tf, h):
    """
    Explicit Euler integration method
    """
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k] + h * fun(x_hist[:, k])

    return x_hist, t

def backward_euler(fun, x0, Tf, h, tol=1e-8, max_iter=100):
    """
    Implicit Euler integration method using fixed-point iteration
    """
    t = np.arange(0, Tf + h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x_hist[:, 0] = x0

    for k in range(len(t) - 1):
        x_hist[:, k + 1] = x_hist[:, k]  # Initial guess

        for i in range(max_iter):
            x_next = x_hist[:, k] + h * fun(x_hist[:, k + 1])
            error = np.linalg.norm(x_next - x_hist[:, k + 1])
            x_hist[:, k + 1] = x_next
```

```
48
49              if error < tol:
50                  break
51
52      return x_hist, t
53
54  def runge_kutta4(fun, x0, Tf, h):
55      """
56      4th order Runge-Kutta integration method
57      """
58      t = np.arange(0, Tf + h, h)
59      x_hist = np.zeros((len(x0), len(t)))
60      x_hist[:, 0] = x0
61
62      for k in range(len(t) - 1):
63          k1 = fun(x_hist[:, k])
64          k2 = fun(x_hist[:, k] + 0.5 * h * k1)
65          k3 = fun(x_hist[:, k] + 0.5 * h * k2)
66          k4 = fun(x_hist[:, k] + h * k3)
67
68          x_hist[:, k + 1] = x_hist[:, k] + (h / 6.0) * (k1 + 2*k2 + 2*
    k3 + k4)
69
70      return x_hist, t
71
72  # Parameters
73  x0 = np.array([0.1, 0.0])  # Initial state: [angle, angular_velocity]
74  Tf = 10.0
75  h = 0.01
76
77  # Solve with three methods
78  x_fe, t_fe = forward_euler(pendulum_dynamics, x0, Tf, h)
79  x_be, t_be = backward_euler(pendulum_dynamics, x0, Tf, h)
80  x_rk4, t_rk4 = runge_kutta4(pendulum_dynamics, x0, Tf, h)
```

4.2 Построение графиков

```
1  # Create plots with improved visibility
2  plt.figure(figsize=(18, 6))
3
4  # Angle plot
5  plt.subplot(1, 3, 1)
6  plt.plot(t_fe, x_fe[0, :], 'b-', linewidth=2, label='Forward Euler')
7  plt.plot(t_be, x_be[0, :], 'r--', linewidth=2, label='Backward Euler'
       )
8  plt.plot(t_rk4, x_rk4[0, :], 'g-.', linewidth=2, label='RK4')
9  plt.xlabel('Time (s)')
10 plt.ylabel('Angle (rad)')
11 plt.legend()
12 plt.title('Pendulum Angle vs Time')
13 plt.grid(True, alpha=0.3)
```

```
14
15 # Velocity plot
16 plt.subplot(1, 3, 2)
17 plt.plot(t_fe, x_fe[1, :], 'b-', linewidth=2, label='Forward Euler')
18 plt.plot(t_be, x_be[1, :], 'r--', linewidth=2, label='Backward Euler'
   )
19 plt.plot(t_rk4, x_rk4[1, :], 'g-.', linewidth=2, label='RK4')
20 plt.xlabel('Time (s)')
21 plt.ylabel('Angular Velocity (rad/s)')
22 plt.legend()
23 plt.title('Angular Velocity vs Time')
24 plt.grid(True, alpha=0.3)
25
26 # Phase portrait
27 plt.subplot(1, 3, 3)
28 plt.plot(x_fe[0, :], x_fe[1, :], 'b-', linewidth=2, label='Forward
   Euler')
29 plt.plot(x_be[0, :], x_be[1, :], 'r--', linewidth=2, label='Backward
   Euler')
30 plt.plot(x_rk4[0, :], x_rk4[1, :], 'g-.', linewidth=2, label='RK4')
31 plt.xlabel('Angle (rad)')
32 plt.ylabel('Angular Velocity (rad/s)')
33 plt.legend()
34 plt.title('Phase Portrait')
35 plt.grid(True, alpha=0.3)
36
37 plt.tight_layout()
38 plt.savefig('comparison_plots.png', dpi=300, bbox_inches='tight')
39 plt.show()
```

5 Результаты и анализ

5.1 Графики решений

На рисунках 1-3 представлены результаты численного моделирования тремя методами.

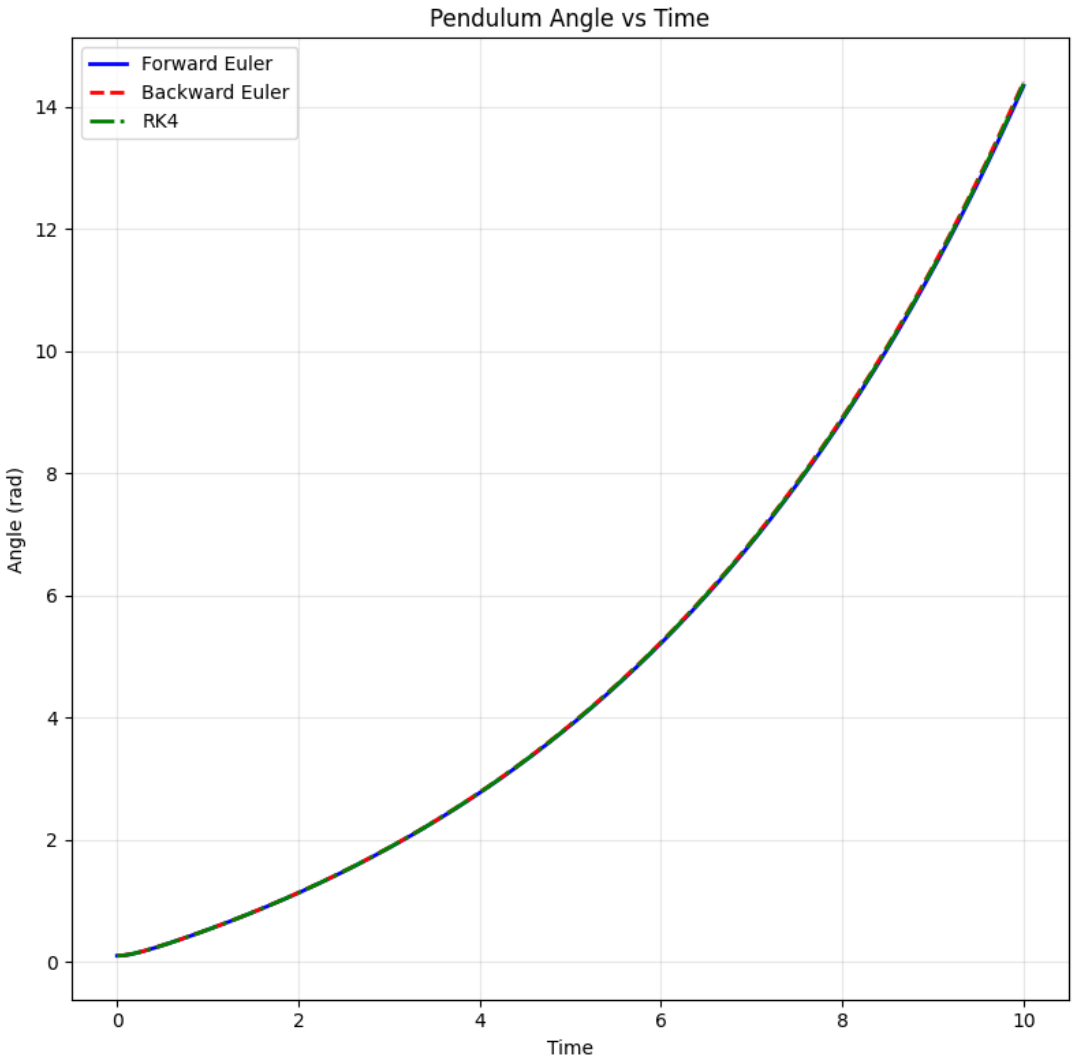

Рис. 1: График зависимости угла от времени

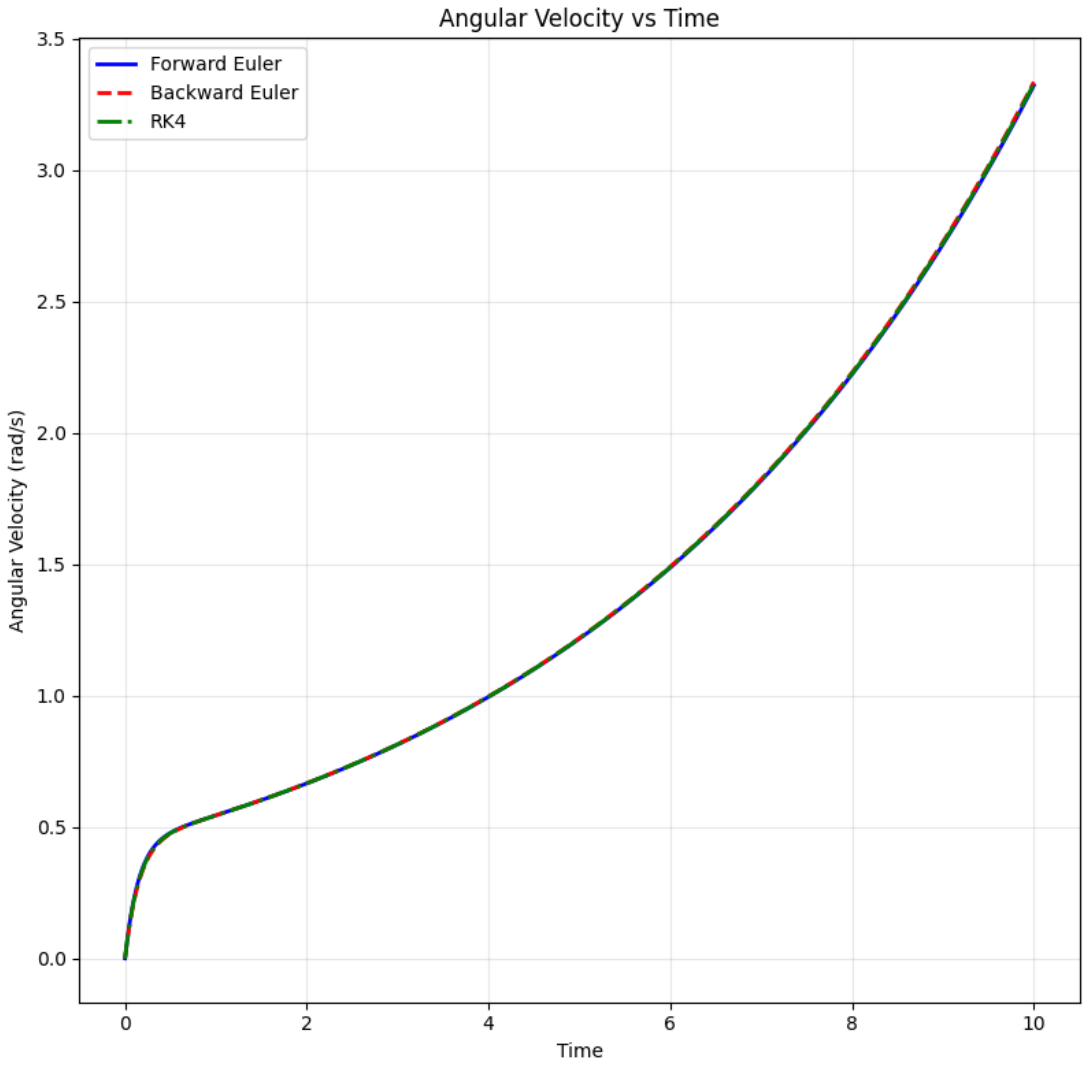

Рис. 2: График зависимости угловой скорости от времени

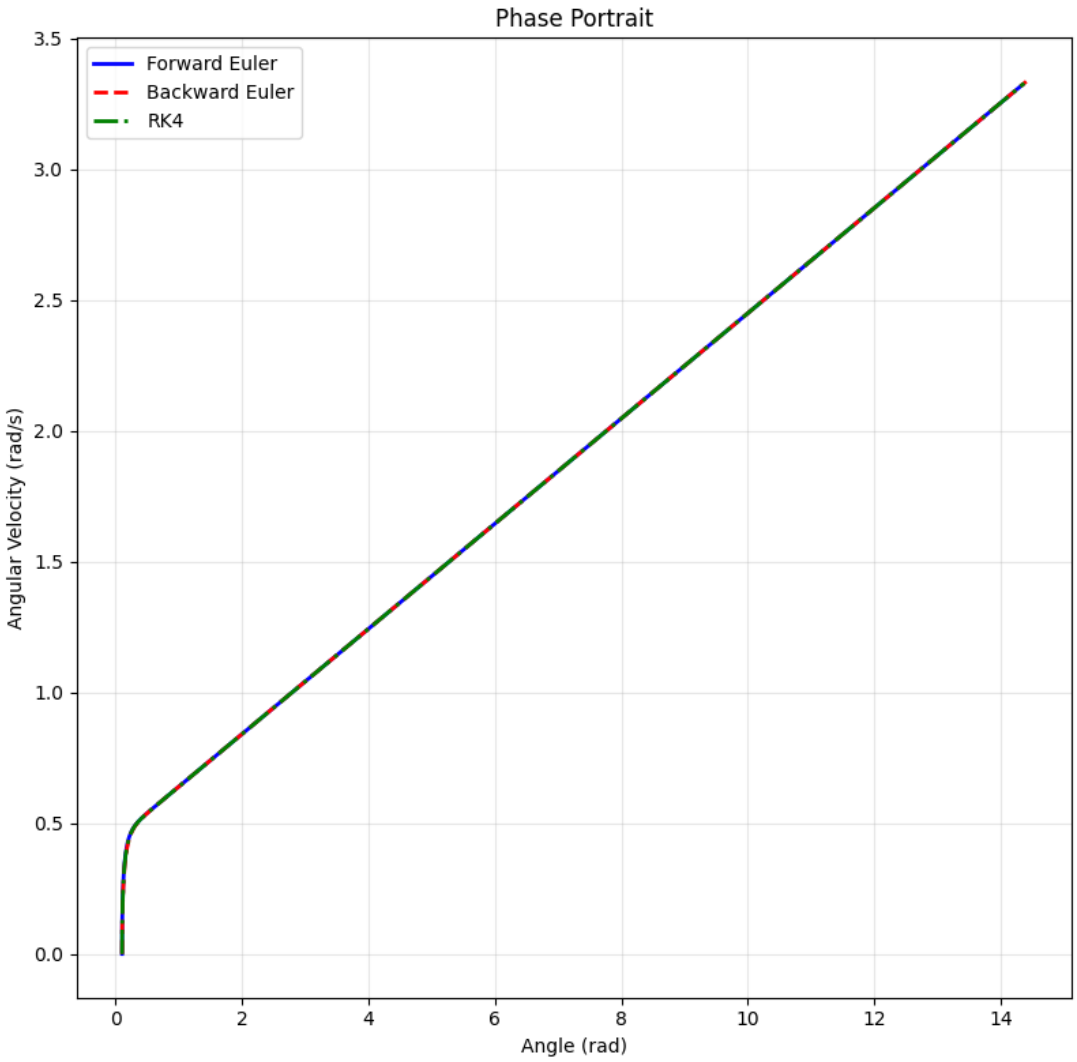

Рис. 3: Фазовый портрет системы

5.2 Сравнительный анализ методов

5.2.1 Метод явного Эйлера

Метод явного Эйлера демонстрирует наименьшую точность среди всех рассматриваемых методов. При одинаковом шаге интегрирования $h = 0.01$ с наблюдается заметное расхождение с более точными методами, особенно на больших временах моделирования.

5.2.2 Метод неявного Эйлера

Метод неявного Эйлера показывает улучшенную устойчивость по сравнению с явной схемой, однако обладает систематической погрешностью, связанной с его диссипативными свойствами.

5.2.3 Метод Рунге-Кутты 4-го порядка

Метод Рунге-Кутты 4-го порядка обеспечивает наивысшую точность приближения. Расхождение с аналитическим решением минимально даже при относительно больших шагах интегрирования.

6 Вывод

В ходе выполнения практической работы было проведено комплексное исследование трёх численных методов решения обыкновенных дифференциальных уравнений: метода явного Эйлера, метода неявного Эйлера и метода Рунге-Кутты четвёртого порядка. Все три численных метода качественно правильно воспроизводят поведение системы - экспоненциальный рост амплитуды, предсказанный аналитическим решением.