

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

ЛАБОРАТОРНАЯ РАБОТА №1

в рамках дисциплины
«Имитационное моделирование робототехнических систем»

Выполнил:

Студент группы R4150 Тамм А.Э.

Санкт-Петербург 2025

ОСНОВНАЯ ЧАСТЬ

1 Цель работы

Выполнить аналитическое и имитационное моделирование ОДУ системы по варианту.

2 Задачи

1. Решить аналитически
2. Составить три интегратора на основе явного Эйлера, неявного Эйлера и метода Рунге-Кутты 4 порядка.
3. Сравнить результаты симуляций, сделать выводы.
4. Составить отчёт по результатам работы

3 Исходные данные

$$a \cdot x'' + b \cdot x' + c \cdot x = d$$

```
# Коэффициенты из задания  
a = -8.57  
b = 7.85  
c = -8.03  
d = -2.16
```

Данное уравнение имеет решение общее решение:

$$x(t) = e^{0.458t} (A \cos(0.853t) + B \sin(0.853t)) + 0.269$$

С начальными условиями $x(0)=1.0$ $x'(0)=0$

$$A=0.731 \quad B=-0.392$$

```

def explicit_euler(func, x0, t_span, dt):
    """Явный метод Эйлера"""
    t = np.arange(t_span[0], t_span[1] + dt, dt)
    x = np.zeros((len(x0), len(t)))
    x[:, 0] = x0

    for i in range(len(t) - 1):
        x[:, i + 1] = x[:, i] + dt * func(t[i], x[:, i])

    return t, x

def implicit_euler(func, x0, t_span, dt, tol=1e-8, max_iter=100):
    """Неявный метод Эйлера"""
    t = np.arange(t_span[0], t_span[1] + dt, dt)
    x = np.zeros((len(x0), len(t)))
    x[:, 0] = x0

    for i in range(len(t) - 1):
        # Начальное приближение
        x_guess = x[:, i]

        # Итерации Ньютона
        for _ in range(max_iter):
            f_next = func(t[i + 1], x_guess)
            x_new = x[:, i] + dt * f_next

            if np.linalg.norm(x_new - x_guess) < tol:
                break
            x_guess = x_new

        x[:, i + 1] = x_new

    return t, x

def runge_kutta4(func, x0, t_span, dt):
    """Метод Рунге-Кутты 4-го порядка"""
    t = np.arange(t_span[0], t_span[1] + dt, dt)
    x = np.zeros((len(x0), len(t)))
    x[:, 0] = x0

    for i in range(len(t) - 1):
        k1 = func(t[i], x[:, i])
        k2 = func(t[i] + dt / 2, x[:, i] + dt / 2 * k1)
        k3 = func(t[i] + dt / 2, x[:, i] + dt / 2 * k2)
        k4 = func(t[i] + dt, x[:, i] + dt * k3)

        x[:, i + 1] = x[:, i] + dt / 6 * (k1 + 2 * k2 + 2 * k3 + k4)

    return t, x

```

Листинг 1- Интеграторы

4 Результаты моделирования

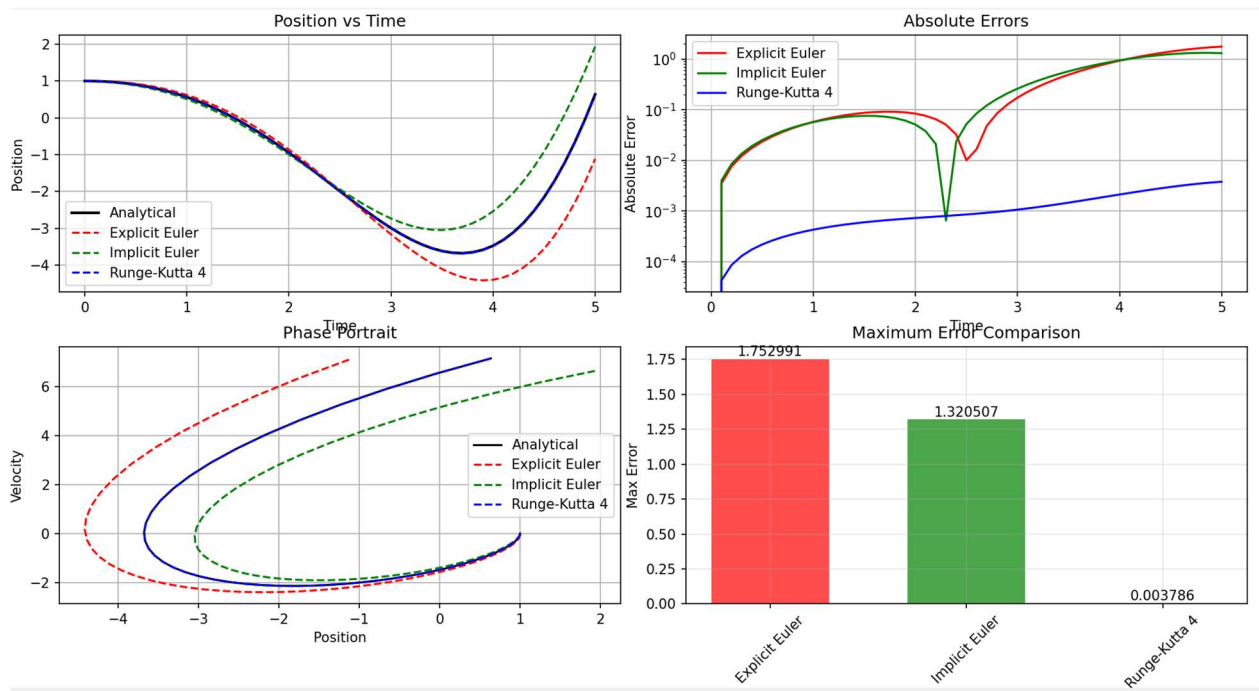


Рисунок 2 – Моделирование при $dt = 0.1$

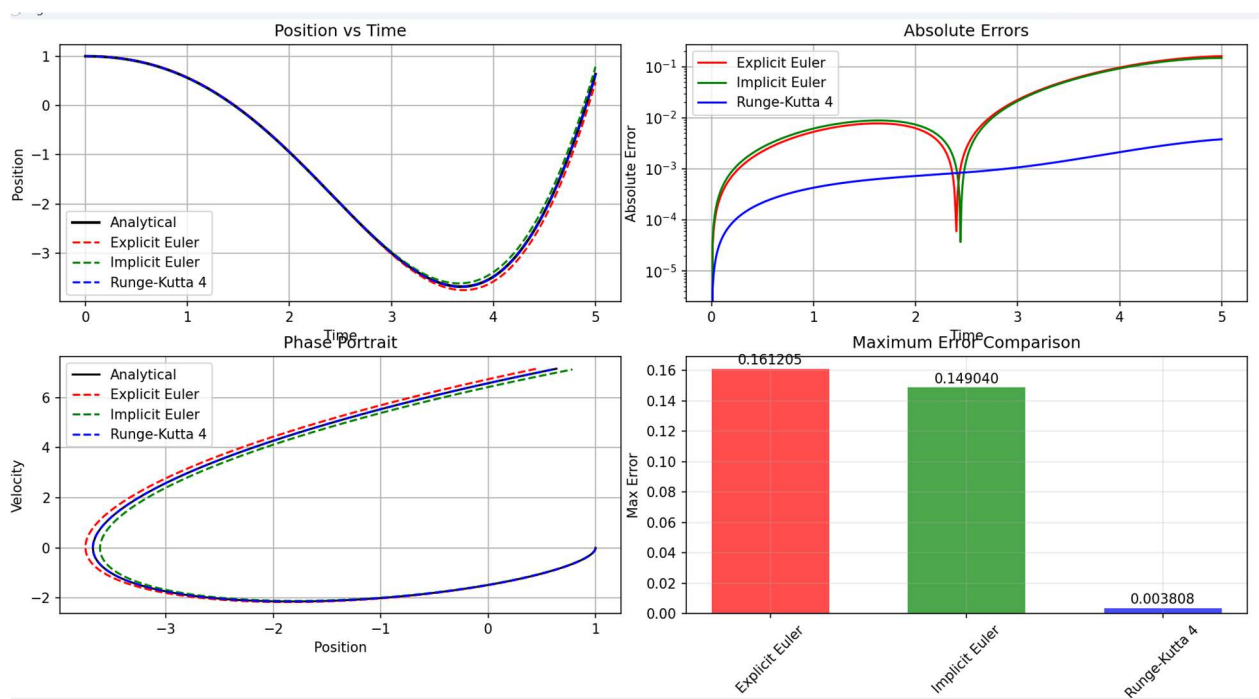


Рисунок 3 – Моделирование при $dt = 0.01$

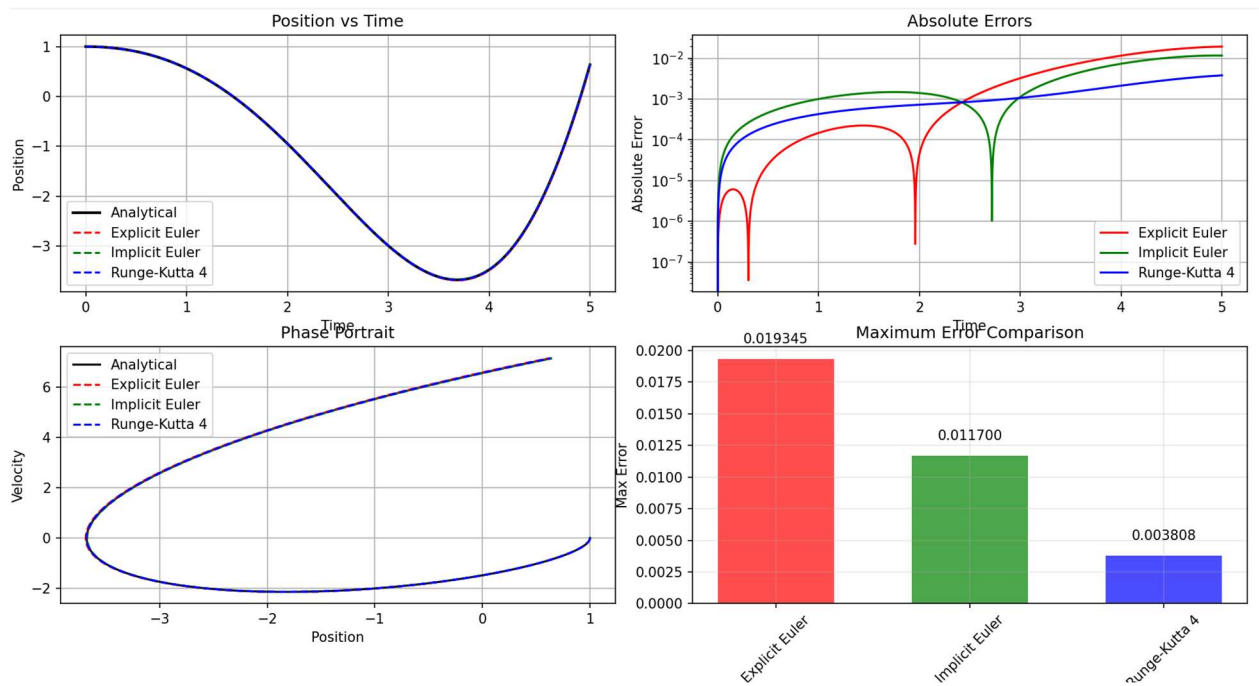


Рисунок 3 – Моделирование при $dt = 0.001$

5 Выводы

В данной работе было получено уравнение и осуществлено решение ОДУ при помощи трёх различных интеграторов. Согласно результатам моделирования, наилучшим образом показал себя интегратор Рунге Кутты. Также можно заметить что этот интегратор не почти зависит от шага интегрирования.