Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет по лабораторной работе №1

По дисциплине: Имитационное моделирование робототехнических систем

Тема: Сравнительный анализ методов интегрирования

Автор: Оллилайнен Р.А., группа R4133c

Принял: Ракшин Е.А

Санкт-Петербург, 2025

Цель работы: сравнение методов интегрирования – методы явного и неявного Эйлера, метод Ранге-Кутта 4-го порядка.

Данное уравнение: $0.65\ddot{x} + 6.44\dot{x} - 9.8x = 5.49$

Аналитическое решение уравнения при нулевых начальных условиях:

$$x(t) = 0.5899e^{(1.3404\ t)} + 0.0703e^{(-11.2481\ t)} - 0.5602$$

Работа программы:

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

a, b, c, d = 0.65, 6.44, -9.8, 5.49

def ode(x):
    """
    State vector x = [x, x_dot]
    """

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

    return np.array([x[1], x_ddot])

def eq_solution(x0):
    t = np.arange(0, Tf + h, h)
    x = np.zeros((len(x0), len(t)))
    x[:, 0] = x0

    for k in range(1, len(t)):
        x[:, k] = [0.5899 * m.exp(1.3404 * t[k]) + 0.0703 * m.exp(-11.2481
* t[k]) - 0.5602, 0.5899 * 1.3404 * m.exp(1.3404 * t[k]) + 0.0703 * (-
11.2481) * m.exp(-11.2481 * t[k])]
    return x, t

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

            if error < tol:
                break

    return x_hist, t

def runge_kutta4(fun, x0, Tf, h):
    """
    4th order Runge-Kutta 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):
        k1 = fun(x_hist[:, k])
        k2 = fun(x_hist[:, k] + 0.5 * h * k1)
        k3 = fun(x_hist[:, k] + 0.5 * h * k2)
        k4 = fun(x_hist[:, k] + h * k3)

        x_hist[:, k + 1] = x_hist[:, k] + (h / 6.0) * (k1 + 2*k2 + 2*k3 +
k4)

    return x_hist, t

# Test all integrators
x0 = np.array([0.1, 0.0])  # Initial state: [angle, angular_velocity]
Tf = 10.0
h = 0.01

x_sol, t_sol = eq_solution(x0)

# Forward Euler
x_fe, t_fe = forward_euler(ode, x0, Tf, h)
```

```
# Backward Euler
x_be, t_be = backward_euler(ode, x0, Tf, h)

# Runge-Kutta 4
x_rk4, t_rk4 = runge_kutta4(ode, x0, Tf, h)

# Plot results
plt.plot(t_fe, x_fe[0, :], label='Forward Euler')
plt.plot(t_be, x_be[0, :], label='Backward Euler')
plt.plot(t_rk4, x_rk4[0, :], label='RK4', marker='*')
plt.plot(t_sol, x_sol[0, :], label='Math Solution', color='black',
linestyle='--')

plt.xlabel('Time (sec)')
plt.ylabel('Angle (rad)')
plt.legend()
plt.title('Pendulum Angle (t)')

plt.show()
```

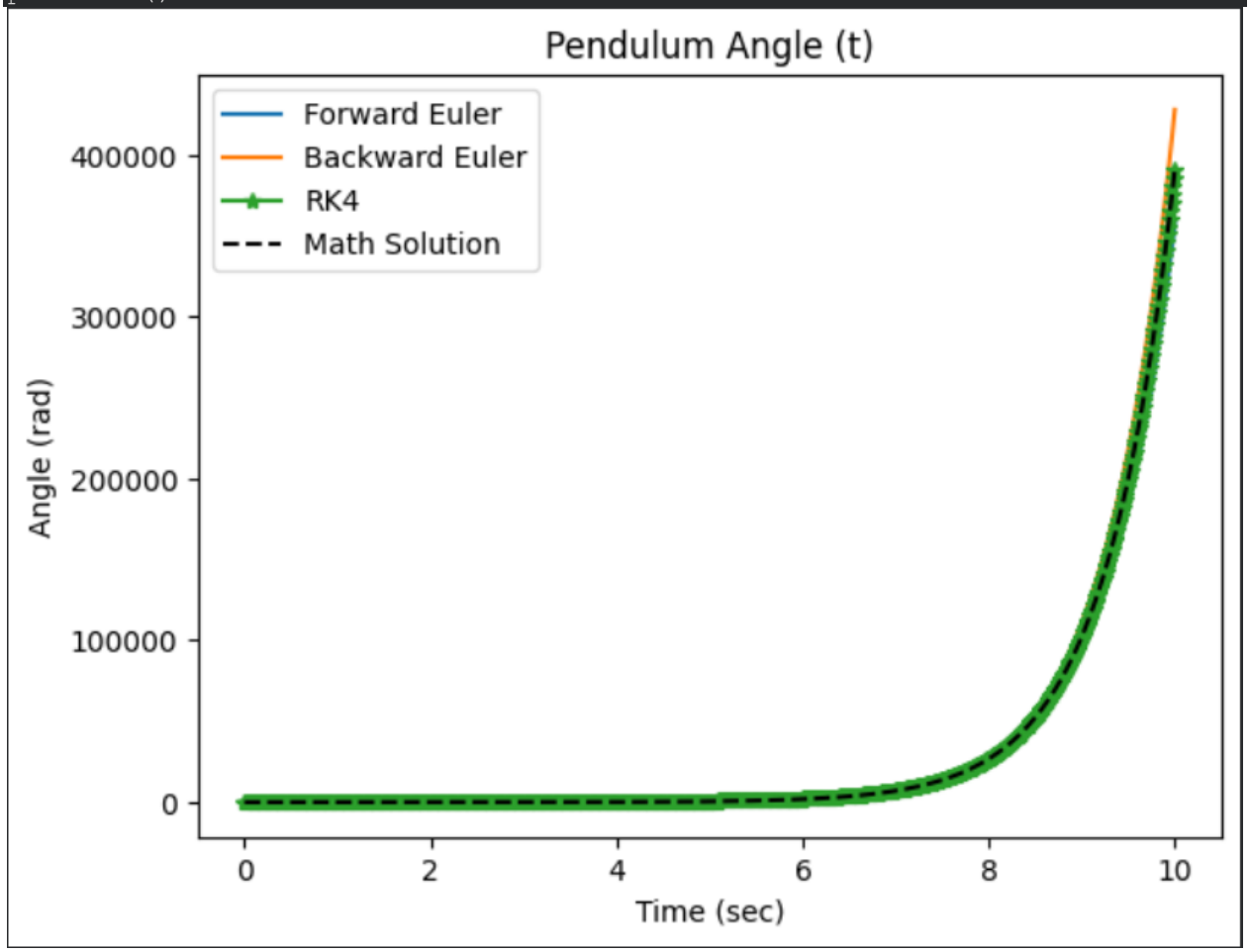

Рис. 1. Положение

```
plt.plot(t_fe, x_fe[1, :], label='Forward Euler')
plt.plot(t_be, x_be[1, :], label='Backward Euler')
plt.plot(t_rk4, x_rk4[1, :], label='RK4', marker='*')
plt.plot(t_sol, x_sol[1, :], label='Math Solution', color='black',
linestyle='--')

plt.xlabel('Time (sec)')
plt.ylabel('Angular Velocity (rad/s)')
plt.legend()
plt.title('Angular Velocity (t)')

plt.show()
```

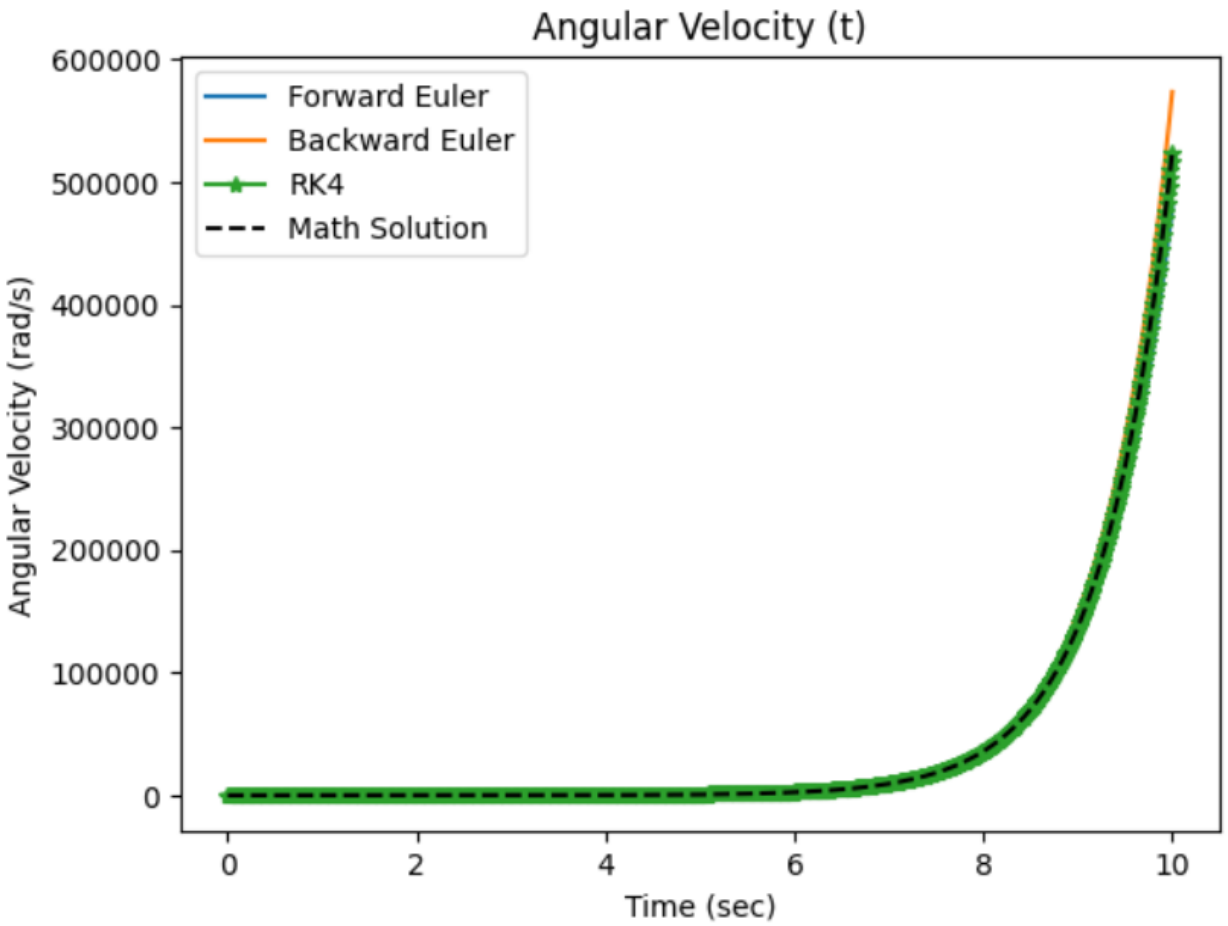

Рис. 2. Скорость

```
plt.plot(x_fe[0, :], x_fe[1, :], label='Forward Euler')
plt.plot(x_be[0, :], x_be[1, :], label='Backward Euler')
plt.plot(x_rk4[0, :], x_rk4[1, :], label='RK4', marker='*')
plt.plot(x_sol[0, :], x_sol[1, :], label='Math Solution', color='black',
linestyle='--')

plt.xlabel('Angle (rad)')
plt.ylabel('Angular Velocity (rad/s)')
plt.legend()
plt.title('Phase Portrait')

plt.show()
```

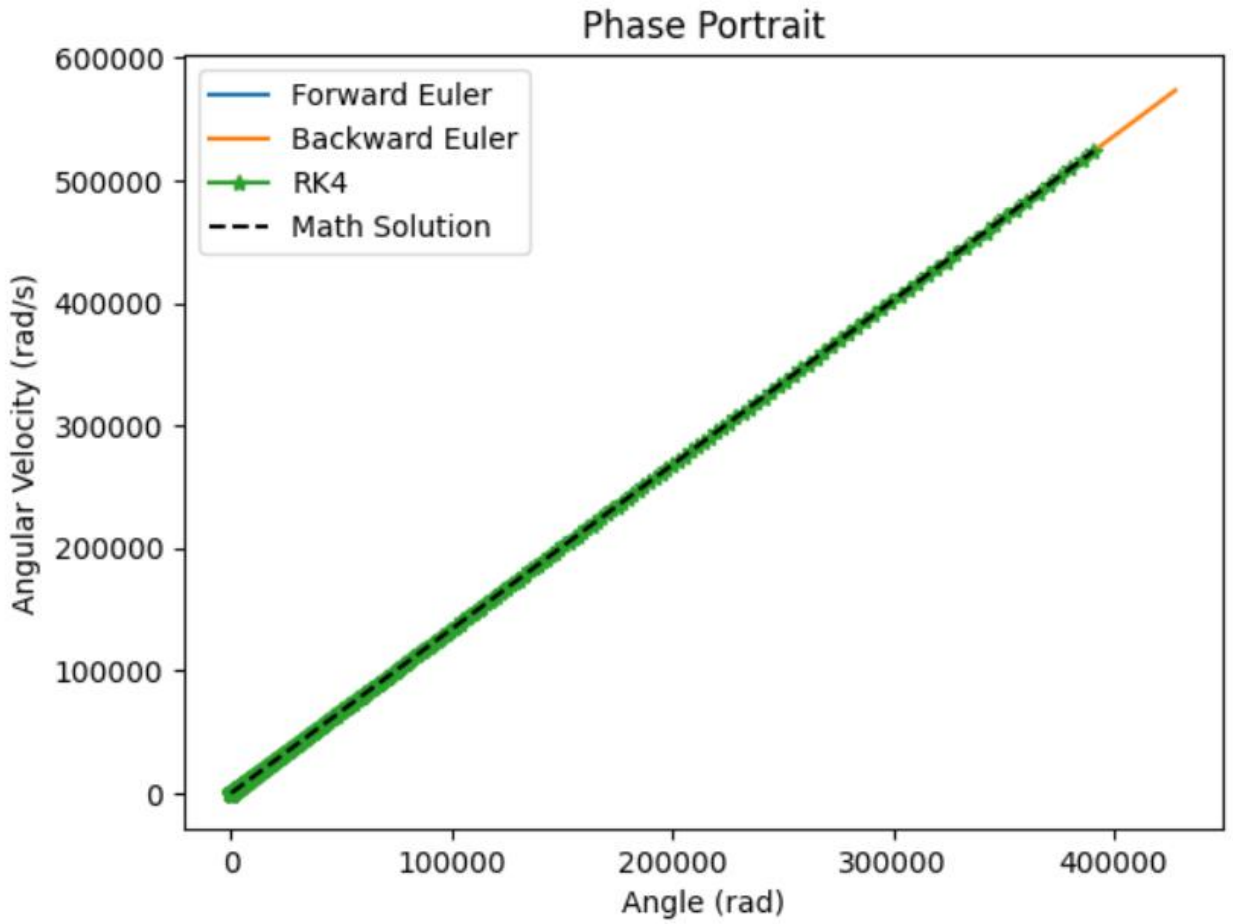

Рис. 3. Фазовый портрет

Вывод: в ходе данной работы было выполнено три метода интегрирования, каждый из которых показал хорошую точность в сравнении с аналитическим решением, на графиках положения, скорости и фазового портрета видно, что ошибки между данными методами и аналитическим решением практически нулевые, что говорит о применимости каждого из методов.