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 ДЛЯ СИСТЕМЫ МАССА-ПРУЖИНА-ДЕМПФЕР

15.04.06 «Робототехника и ИИ»

Выполнила: Шульга И.Д. (468129)
Группа: R4133c
Преподаватель: Ракшин Е.А.
Дата выполнения: 12 ноября 2025 г.

г. Санкт-Петербург
2025 год

Содержание

1 Цель работы

Цель данного задания — получить, аналитически и численно решить дифференциальное уравнение системы масса-пружина-демпфер, сравнить результаты различных численных методов, проанализировать их погрешности, соответствие аналитическому решению и сделать выводы.

2 Теоретическая часть

Исходная схема и обозначения приведены на рисунке 1.

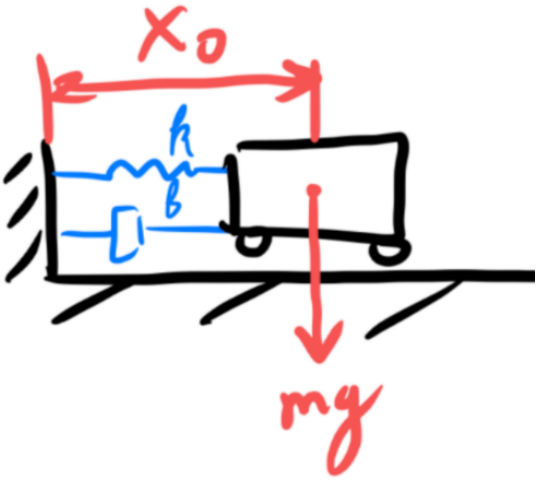

Рис. 1: Схема масс-пружина-демпфер

Уравнение движения для системы масс-пружина-демпфер:

$$m\ddot{x} + b\dot{x} + kx = 0$$

где

- $m = 0.4$ кг — масса,
- $k = 17.4$ Н/м — жёсткость пружины,
- $b = 0.045$ Н·с/м — коэффициент демпфирования,
- $x(t)$ — отклонение массы.

Подставим значения:

$$0.4\ddot{x} + 0.045\dot{x} + 17.4x = 0$$

Разделим на 0.4:

$$\ddot{x} + 0.1125\dot{x} + 43.5x = 0$$

Аналитическое решение при начальных условиях $x(0) = x_0$, $\dot{x}(0) = v_0$:

$$x(t) = e^{-0.05625t}\left[A\cos(6.601t) + B\sin(6.601t)\right]$$

где A, B — определяются из начальных условий.

3 Практическая часть

3.1 Численные методы

В качестве численных методов используются:

- Явный Эйлер,
- Неявный Эйлер,
- Рунге-Кутта 4-го порядка.

3.2 Реализация решения (Python)

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

def mass_spring_damper(s):
    m = 0.4
    b = 0.045
    k = 17.4

    x = s[0]
    x_dot = s[1]
    x_ddot = -(b/m)*x_dot - (k/m)*x
    return np.array([x_dot, x_ddot])

def forward_euler(fun, x0, Tf, h):

    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):

    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]
        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):
    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

x0 = np.array([0.27, 0.0])
Tf = 10.0
h = 0.01

x_fe, t_fe = forward_euler(mass_spring_damper, x0, Tf, h)
x_be, t_be = backward_euler(mass_spring_damper, x0, Tf, h)
x_rk4, t_rk4 = runge_kutta4(mass_spring_damper, x0, Tf, h)

plt.figure(figsize=(18, 6))

plt.subplot(1, 3, 1)
plt.plot(t_fe, x_fe[0, :], 'b-', linewidth=2, label='Forward␣Euler')
plt.plot(t_be, x_be[0, :], 'r--', linewidth=2, label='Backward␣Euler')
plt.plot(t_rk4, x_rk4[0, :], 'g-.', linewidth=2, label='RK4')
plt.xlabel('          ␣(  )')
plt.ylabel('                ␣x␣(  )')
plt.legend()
plt.title('                ␣vs␣          ')
plt.grid(True, alpha=0.3)

plt.subplot(1, 3, 2)
plt.plot(t_fe, x_fe[1, :], 'b-', linewidth=2, label='Forward␣Euler')
plt.plot(t_be, x_be[1, :], 'r--', linewidth=2, label='Backward␣Euler')
plt.plot(t_rk4, x_rk4[1, :], 'g-.', linewidth=2, label='RK4')
plt.xlabel('          ␣(  )')
plt.ylabel('                ␣x_dot␣(  /  )')
plt.legend()
plt.title('                ␣vs␣          ')
plt.grid(True, alpha=0.3)

plt.subplot(1, 3, 3)
plt.plot(x_fe[0, :], x_fe[1, :], 'b-', linewidth=2, label='Forward␣Euler')
plt.plot(x_be[0, :], x_be[1, :], 'r--', linewidth=2, label='Backward␣Euler')
plt.plot(x_rk4[0, :], x_rk4[1, :], 'g-.', linewidth=2, label='RK4')
plt.xlabel('x␣(  )')
plt.ylabel('                ␣(  /  )')
plt.legend()
plt.title('              ␣              ')
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('comparison_plots.png', dpi=300, bbox_inches='tight')
plt.show()
```

Графики решений

На рисунках 2-4 представлены результаты численного моделирования тремя методами.

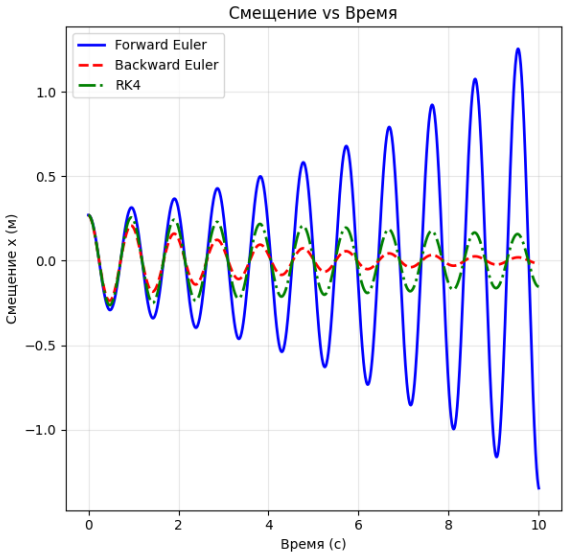

Рис. 2: График зависимости смещения от времени

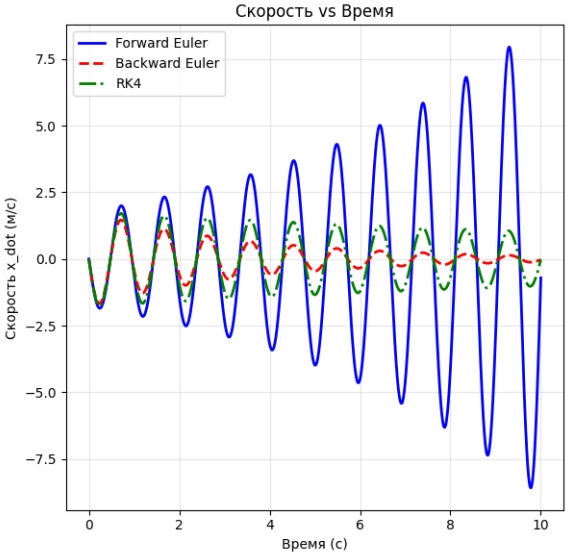

Рис. 3: График зависимости скорости от времени

4 Выводы

В данной работе было составлено ОДУ для системы "масса-пружина-демпфер получено аналитическое решение и реализованы численные методы (явный/неявный Эйлер и Рунге-Кутта 4-го порядка). Сравнение результатов показало, что аналитический и численный подходы с малыми шагами времени совпадают (РК4 наиболее точен). "Явный Эйлер"может незначительно расходиться при крупном шаге, "неявный Эйлер"устойчив.

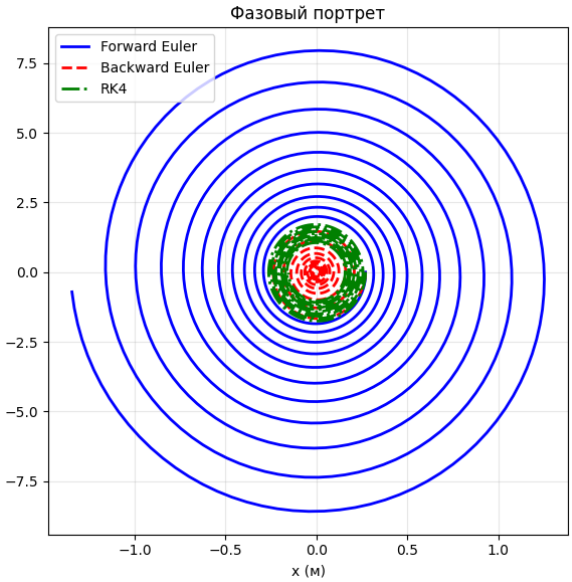

Рис. 4: Фазовый портрет системы