МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет систем управления и робототехники

Отчёт по практической работе №2

по дисциплине
«Имитационное моделирование робототехнических систем»

Студент:
Группа № R4134c *Д.С. Обухова*

Преподаватель:
ассистент ФСУиР *Е.А. Ракшин*

Санкт Петербург 2025

Начальные данные (вариант 1, #35, ису 336878)

$$m = 0.1\ kg,\ \ k = 12.4\frac{N}{m},\ \ b = 0.005\frac{N*s}{m},$$

$$l = 0.48\ m, \qquad \theta_0 = 1.005340873\ \text{rad}, \qquad x_0 = 0.25\ m$$

1. Дифференциальное уравнение

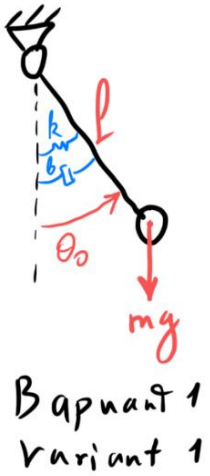

Рисунок 1 – Система «масса-пружина-демпфер» согласно 1 варианту

Кинетическая энергия	$K = \frac{1}{2}ml^2\dot{\theta}^2$
Потенциальная энергия (при малых углах)	$P_{mg} = mgl(1 - cos\theta) = \frac{1}{2}mgl\theta^2$ $P_k = \frac{1}{2}k\theta^2$ $P = P_{mg} + P_k = mgl\theta^2 + \frac{1}{2}k\theta^2 = \frac{1}{2}\theta^2(mgl + k)$
Демпфирующий момент	$Q = -b\dot{\theta}$
Лагранжиан	$L = K - P = \frac{1}{2}ml^2\theta^2 - \frac{1}{2}\theta^2(mgl + k)$
Уравнение Лагранжа	$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right) - \frac{\partial L}{\partial\theta} = Q$ $\frac{d}{dt}\left(ml^2\dot{\theta}\right) + (k + mgl)\theta = -b\dot{\theta}$ $ml^2\ddot{\theta} + b\dot{\theta} + (k + mgl)\theta = 0$ $\ddot{\theta} + \frac{b}{ml^2}\dot{\theta} + \frac{k + mgl}{ml^2}\theta = 0$ $\omega = \frac{k+mgl}{ml^2},\ \ c = \frac{b}{ml^2},\ \ \zeta = \frac{c}{2\omega} = \frac{b}{2ml^2\omega}$

Аналитическое решение при начальных условиях $\theta(0) = \theta_0, \dot{\theta}(0) = \dot{\theta_0}$

- Колебание

$$\zeta = \frac{b}{2ml^2\omega} < 1$$

Корни характеристического уравнения:

$$C_1 = \theta_0$$

$$C_2 = \frac{\dot{\theta}_0 + \zeta\omega\theta_0}{\omega\sqrt{1-\zeta^2}}$$

$$\theta(t) = e^{-\omega t}\left(\theta_0 + t\left(\dot{\theta}_0 + \omega\theta_0\right)\right)$$

$$\theta(t) = e^{-\zeta\omega t}\left(\theta_0 \cos\left(\omega t\sqrt{1-\zeta^2}\right) + \frac{\dot{\theta}_0 + \zeta\omega\theta_0}{\omega\sqrt{1-\zeta^2}} \sin\left(\omega t\sqrt{1-\zeta^2}\right)\right)$$

- Критическое затухание

$$\zeta = \frac{b}{2ml^2\omega} = 1$$

Корни характеристического уравнения

$$\lambda_{1,2} = -\omega$$

$$C_1 = \theta_0$$

$$C_2 = \theta_0 + \dot{\omega}\theta_0$$

$$\theta(t) = e^{-\omega t}\left(\theta_0 + t\left(\dot{\theta_o} + \omega\theta_0\right)\right)$$

- Перезатухание

$$\zeta = \frac{b}{2ml^2\omega} > 1$$

Корни характеристического уравнения

$$\lambda_1 = -\omega(\zeta + \sqrt{\zeta^2 - 1})$$

$$\lambda_1 = -\omega(\zeta - \sqrt{\zeta^2 - 1})$$

$$C_1 = \theta_0 - \frac{\lambda_1\theta_0 - \dot{\theta_0}}{\lambda_1 - \lambda_2}$$

$$C_2 = \frac{\lambda_1\theta_0 - \dot{\theta_0}}{\lambda_1 - \lambda_2}$$

$$\theta(t) = e^{-\lambda_1 t}\left(\theta_0 - \frac{\lambda_1\theta_0 - \dot{\theta_0}}{\lambda_1 - \lambda_2}\right) + e^{-\lambda_2 t}\frac{\lambda_1\theta_0 - \dot{\theta_0}}{\lambda_1 - \lambda_2}$$

2. Методы Эйлера и Рунге-Кутты

```python
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
```

3. График

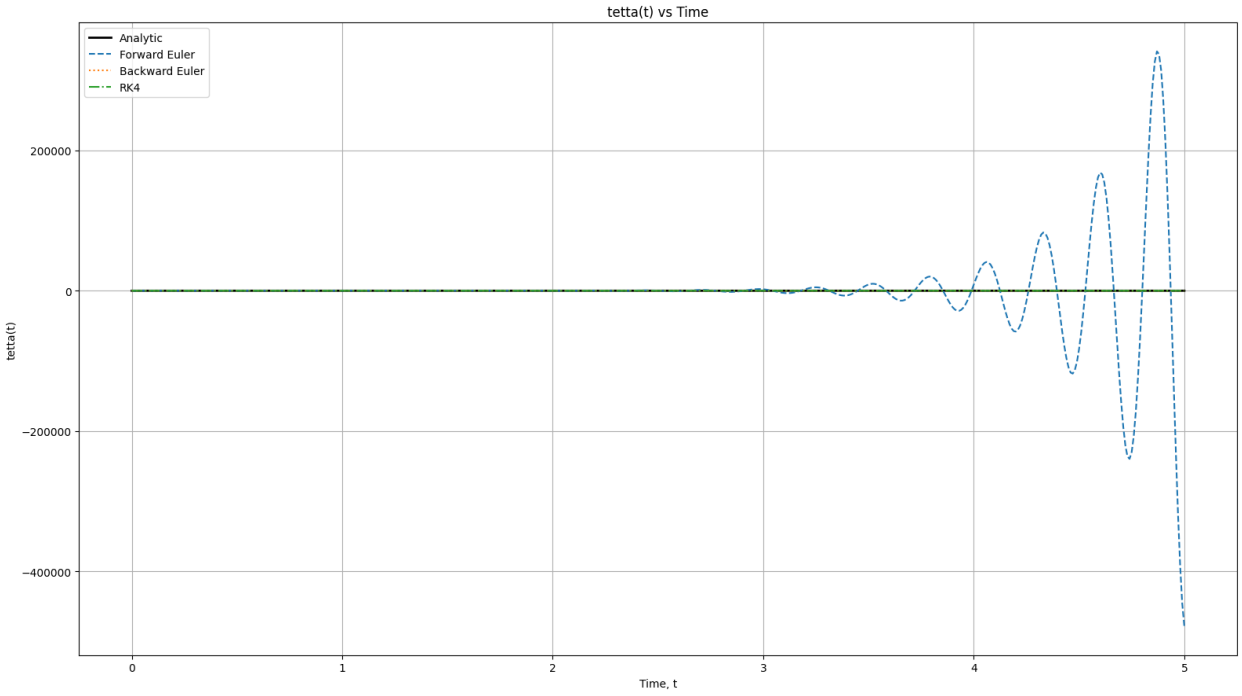

Рисунок 2 – График $\theta(t)$ для аналитического решения и численных методов: Forward Euler, Backward Euler, метода Рунге-Кутта 4-го порядка.

Заключение

В ходе выполнения данной практической работы было выведено уравнение системы «масса-пружина-демпфер». Произведено сравнение аналитического и численных методов интегрирования дифференциальных уравнений – Forward Euler, Backward Euler, метода Рунге-Кутта 4-го порядка (рисунок 2). Худший результат показал неявный Эйлер, который начал расходиться после 3 секунды. Остальные 2 численных метода показали себя достаточно корректно, совпали с графиком аналитического решения.

Листинги

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

m = 0.1
k = 12.4
b = 0.005
l = 0.48
g = 9.81

def tetta_sol(tetta, omega1, t):
  omega = np.sqrt((k + m *g*l) / (m*l**2))
  dzetta = b / (m*l**2)/(2*omega)
  if dzetta < 1:
    C1 = tetta
    C2 = (omega1 + dzetta*omega*tetta) / omega * np.sqrt(1 - dzetta**2)
    return np.exp(-dzetta * omega * t) * (C1 * np.cos(omega*np.sqrt(1-
dzetta**2)*t)+C2*np.sin(omega*np.sqrt(1-dzetta**2)*t))
  elif dzetta == 1:
    C1 = tetta
    C2 = omega1 + omega * tetta
    return (C1+ C2 * t) * np.exp(-omega * t)
  else:
    l1 = -omega * (dzetta + np.sqrt(dzetta **2 -1))
    l2 = -omega * (dzetta - np.sqrt(dzetta **2 -1))
    C2 = (l2 * tetta - omega1)/(l1-l2)
    C1 = tetta - C2
    return C1 * np.exp(l1*t) + C2 * np.exp (l2*t)


def pendulum_system(state):
  tetta = state[0]
  omega = state[1]

  dtetta = omega
  domega = -b / (m * l**2) * omega - ((k+m*g*l)/(m*l**2))*tetta

  return np.array([dtetta, domega])

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


def equation (t):
  return tetta_sol(tetta0, omega0, t_fe)

tetta0 = 1.005340873
omega0 = 0
x0 = np.array([tetta0, omega0])

Tf = 5
h = 0.01

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

x_an = equation(t_fe)

plt.figure(figsize=(16,9))

plt.plot(t_fe, x_an, 'k', lw=2, label='Analytic')
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')
plt.xlabel('Time, t')
plt.ylabel('tetta(t)')
plt.title('tetta(t) vs Time')
plt.legend()
```

```
plt.grid(True)

plt.tight_layout()
plt.show()
```