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 №1

по дисциплине
«Имитационное моделирование робототехнических систем»

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

Преподаватель:
ассистент ФСУиР *Е.А. Ракшин*

Санкт Петербург 2025

Начальные данные (вариант 35, ису 336878)

$a = 1.84, b = 8.58, c = -9.6,\ d = -9.45$

1. Аналитическое решение ОДУ

$$a\ddot{x} + b\dot{x} + cx = d$$

$$1.84\ddot{x} + 8.58\dot{x} - 9.6x = -9.45$$

Найдём корни характеристического уравнения:

$$1.84\ddot{x} + 8.58\dot{x} - 9.6x = 0$$

$$\ddot{x} + 4.663\dot{x} - 5.217x = 0$$

$$\lambda^2 + 4.663\lambda - 5.217 = 0$$

$$\lambda = \frac{-4.663 \pm \sqrt{42.611}}{2}$$

$$\lambda_1 = 0.932$$

$$\lambda_2 = -5.595$$

Общее решение имеет вид:

$x_h(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} = C_1 e^{0.932t} + C_2 e^{-5.595t}$

Частное решение неоднородного уравнения:

$$x_p(t) = A \implies \dot{x}_p(t) = 0, \ddot{x}_p(t) = 0$$

$$1.84(0) + 8.58(0) - 9.6(A) = -9.45$$

$$-9.6(A) = -9.45 \Longrightarrow A = \frac{-9.45}{-9.6} \approx 0.9844$$

$$x_p(t) = 0.9844$$

Общее решение и нахождение констант:

$$x(t) = C_1 e^{0.932t} + C_2 e^{-5.595t} + 0.9844$$

$$\dot{x}(t) = 0.932C_1 e^{0.932t} - 5.595C_2 e^{-5.595t}$$

$$x(0) = C_1 e^0 + C_2 e^0 + 0.9844 = C_1 + C_2 + 0.9844 = 0$$

$$\dot{x}(0) = 0.932C_1 e^0 - 5.595C_2 e^0 = 0.932C_1 - 5.595C_2 = 0$$

$$\begin{cases} C_1 + C_2 = -0.9844 \\ 0.932C_1 - 5.595C_2 = 0 \end{cases}$$

$$C_1 \approx 0.854, \quad C_2 \approx -1.838$$

Итоговое аналитическое решение:

$$x(t) = 0.854e^{0.932t} - 1.838e^{-5.595t} + 0.9844$$

2. Численное решение с помощью интеграторов

- Прямой (явный) Эйлер

```
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
```

- Обратный (неявный) Эйлер

```
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
```

- Рунге-Кутты

```
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
```

3. Графики

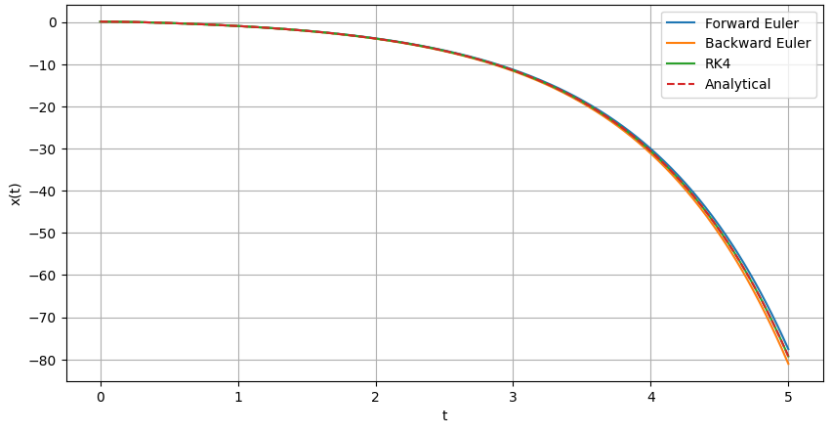

Рисунок 1 – Сравнение численных методов с аналитическим решением $x(t)$.

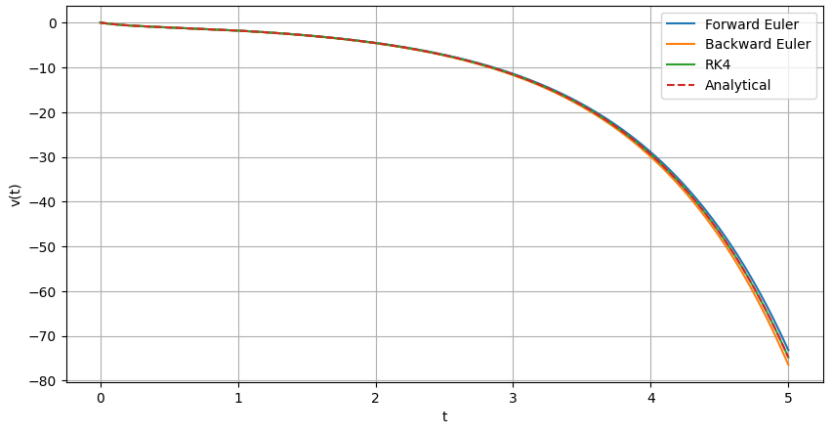

Рисунок 2 – Сравнение численных методов с аналитическим решением $v(t)$.

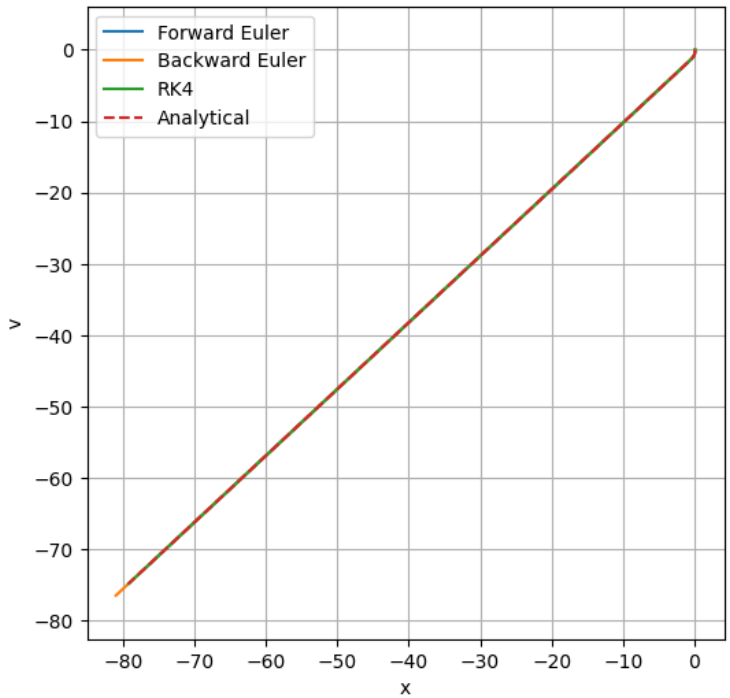

Рисунок 3 – Сравнение численных методов с аналитическим решением $x(v)$.

	max|x-x*|	max|v-v*|	L2(x)	L2(v)
Forward Euler	1.714938e+00	1.599061e+00	5.078566e-01	4.735415e-01
Backward Euler	1.774240e+00	1.654355e+00	5.244605e-01	4.890231e-01
RK4	2.338521e-08	2.226445e-08	6.992396e-09	8.623901e-09

Рисунок 4 – Таблица подсчёта ошибок.

Заключение

В ходе выполнения данной практической работы были рассмотрены аналитические и численные решения линейного дифференциального уравнения 2-го порядка. К чсиленным решениям относятся такие методы как явный и неявный Эйлер, а также Рунге-Кутта 4-го порядка.

Для визуального сравнения аналитического решения с численными методами были построены 3 графика. Также были подсчитаны ошибки. По этим данным можно сделать вывод, что наиболее приближенным к аналитическому решению оказался метод Рунге-Кутта, наиболее отдалённым – Явный Эйлер.

Листинги

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

a, b, c, d = 1.84, 8.58, -9.6, -9.45

# --- Определение системы ---
def ode_linear_rhs(state):
    x, v = state
    return np.array([v, (d - b*v - c*x)/a])

# --- Аналитическое решение ---
def analytical_solution_lin(t, x0, v0, a, b, c, d):
    t = np.asarray(t)
    xp = d / c
    disc = b*b - 4*a*c

    if disc > 1e-12:
        r1 = (-b - np.sqrt(disc)) / (2*a)
        r2 = (-b + np.sqrt(disc)) / (2*a)
        y0 = x0 - xp
        C1 = (y0*r2 - v0) / (r2 - r1)
        C2 = (v0 - y0*r1) / (r2 - r1)
        e1 = np.exp(r1*t); e2 = np.exp(r2*t)
        x = xp + C1*e1 + C2*e2
        v = C1*r1*e1 + C2*r2*e2
        return x, v

    elif abs(disc) <= 1e-12:
        r = -b / (2*a)
        y0 = x0 - xp
        C1 = y0
        C2 = v0 - r*C1
        e = np.exp(r*t)
        x = xp + (C1 + C2*t)*e
        v = (C2 + r*(C1 + C2*t))*e
        return x, v

    else:
        alpha = -b/(2*a)
        beta = np.sqrt(-disc)/(2*a)
        y0 = x0 - xp
        A = y0
        B = (v0 - alpha*A)/beta
        e = np.exp(alpha*t)
        cb = np.cos(beta*t); sb = np.sin(beta*t)
        x = xp + e*(A*cb + B*sb)
        v = e*((alpha*A + beta*B)*cb + (alpha*B - beta*A)*sb)
        return x, v
```

```
# --- Численные методы ---
def forward_euler(f, x0, Tf, h):
    t = np.arange(0, Tf+h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x = x0.copy()
    for i, ti in enumerate(t):
        x_hist[:, i] = x
        x = x + h * f(x)
    return x_hist, t

def backward_euler(f, x0, Tf, h):
    t = np.arange(0, Tf+h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x = x0.copy()
    for i, ti in enumerate(t):
        x_hist[:, i] = x
        for _ in range(3):
            fx = f(x)
            x = x0 + h * fx
        x0 = x
    return x_hist, t

def runge_kutta4(f, x0, Tf, h):
    t = np.arange(0, Tf+h, h)
    x_hist = np.zeros((len(x0), len(t)))
    x = x0.copy()
    for i, ti in enumerate(t):
        x_hist[:, i] = x
        k1 = f(x)
        k2 = f(x + h*k1/2)
        k3 = f(x + h*k2/2)
        k4 = f(x + h*k3)
        x = x + h*(k1 + 2*k2 + 2*k3 + k4)/6
    return x_hist, t

# --- Основная часть ---
x0 = np.array([0.1, 0.0])
Tf, h = 5.0, 0.01

x_fe, t = forward_euler(ode_linear_rhs, x0, Tf, h)
x_be, _  = backward_euler(ode_linear_rhs, x0, Tf, h)
x_rk4, _ = runge_kutta4(ode_linear_rhs, x0, Tf, h)

x_a, v_a = analytical_solution_lin(t, x0[0], x0[1], a, b, c, d)

# --- Подсчёт ошибок ---
def err_stats(x_hist, x_a, v_a):
    ex = np.abs(x_hist[0,:] - x_a)
    ev = np.abs(x_hist[1,:] - v_a)
```

```
    return {
        "max|x-x*|": np.max(ex),
        "max|v-v*|": np.max(ev),
        "L2(x)": np.sqrt(np.mean(ex**2)),
        "L2(v)": np.sqrt(np.mean(ev**2)),
    }

errs = {
    "Forward Euler": err_stats(x_fe, x_a, v_a),
    "Backward Euler": err_stats(x_be, x_a, v_a),
    "RK4":           err_stats(x_rk4, x_a, v_a),
}

display(pd.DataFrame(errs).T)

# --- Графики ---
plt.figure(figsize=(10,5))
plt.plot(t, x_fe[0,:], label="Forward Euler")
plt.plot(t, x_be[0,:], label="Backward Euler")
plt.plot(t, x_rk4[0,:], label="RK4")
plt.plot(t, x_a, "--", label="Analytical")
plt.xlabel("t"); plt.ylabel("x(t)")
plt.legend(); plt.grid(True); plt.show()

plt.figure(figsize=(10,5))
plt.plot(t, x_fe[1,:], label="Forward Euler")
plt.plot(t, x_be[1,:], label="Backward Euler")
plt.plot(t, x_rk4[1,:], label="RK4")
plt.plot(t, v_a, "--", label="Analytical")
plt.xlabel("t"); plt.ylabel("v(t)")
plt.legend(); plt.grid(True); plt.show()

plt.figure(figsize=(6,6))
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")
plt.plot(x_a, v_a, "--", label="Analytical")
plt.xlabel("x"); plt.ylabel("v")
plt.legend(); plt.axis("equal"); plt.grid(True)
plt.show()
```