Национальный исследовательский университет ИТМО
Факультет систем управления и робототехники

Отчет
о выполнении практического задания №2
по дисциплине «Имитационное моделирование робототехнических систем»

Выполнил
студент гр. R4134c
ИСУ 505887 А. С. Абраменко

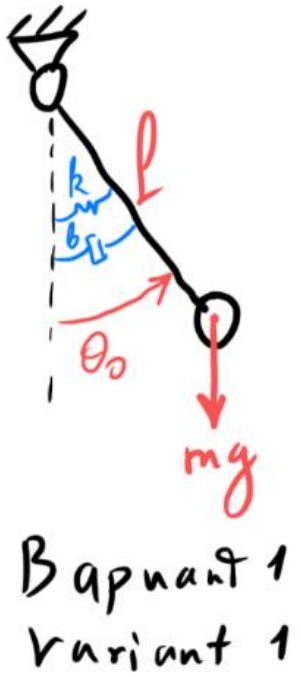
Преподаватель Е. А. Ракшин

«___» _____ 2025 г.

Санкт-Петербург
2025

Задание

1. Составить ОДУ из системы пружинно-массового демпфера, используя приведенные варианты. Вариант 1:



Данные для варианта 1:

m, kg	k, N/m, Nm/rad	b, N*s/m, Nm*s/rad	l, m	θ_0, rad
0.1	17.4	0.035	0.34	-1.303801445

2. Решить аналитически. При невозможности решить описать причину.
3. Сравнить аналитическое решение (если существует) с интегральными методами

Ход работы

1. Уравнение движения маятника

Динамику системы предложено выразить в механике Лагранжа. Кинетическая энергия маятника:

$$K = \frac{1}{2}mv^2 = \frac{1}{2}m(l \cdot \dot{\theta})^2 = \frac{1}{2}ml^2 \cdot \dot{\theta}^2$$

Потенциальная энергия маятника:

$$U = mgh + \frac{1}{2}k\theta^2,$$

где $h = l - l \cdot cos\theta = l \cdot (1 - cos\theta)$

Функция Лагранжа:

$$L = K - P$$

$$L = \frac{1}{2}ml^2 \cdot \dot{\theta}^2 - mgl \cdot (1 - cos\theta) - \frac{1}{2}k\theta^2$$

Уравнение Лагранжа:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q,$$

где $Q = -b\dot{\theta}$.

В данной задаче обобщенная координата

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} = ml^2\ddot{\theta}$$

$$\frac{\partial L}{\partial \theta} = -mglsin\theta - k\theta$$

Подставляя в уравнение Лагранжа:

$$ml^2\ddot{\theta} + mglsin\theta + k\theta = -b\dot{\theta}$$

ОДУ в стандартной форме:

$$\ddot{\theta} + \frac{b}{ml^2}\dot{\theta} + \frac{k}{ml^2}\theta + \frac{gsin\theta}{l} = 0$$

2. Аналитическое решение ОДУ

Полученное ОДУ с использованием значений параметров (см. Задание):

$$\ddot{\theta} + \frac{0{,}035}{0{,}1 \cdot 0{,}34^2}\dot{\theta} + \frac{17{,}4}{0{,}1 \cdot 0{,}34^2}\theta + \frac{9{,}8sin\theta}{0{,}34} = 0$$

Исходное нелинейное ОДУ (в данном ОДУ нелинейность обуславливается наличием слагаемого с $sin\theta$) невозможно решить аналитически методами, приводящими к точному решению в элементарных функциях. В данном случае методы решения данного ОДУ ограничиваются численными методами и решением в аналитическом виде с предварительной линеаризацией. Выполним линеаризацию с тем уточнением, что она справедлива для малых углов θ:

$$\ddot{\theta} + 3{,}02768 \cdot \dot{\theta} + (1505{,}19031 + 28{,}8235) \cdot \theta = 0$$

$$\ddot{\theta} + 3{,}02768 \cdot \dot{\theta} + 1534{,}0135 \cdot \theta = 0$$

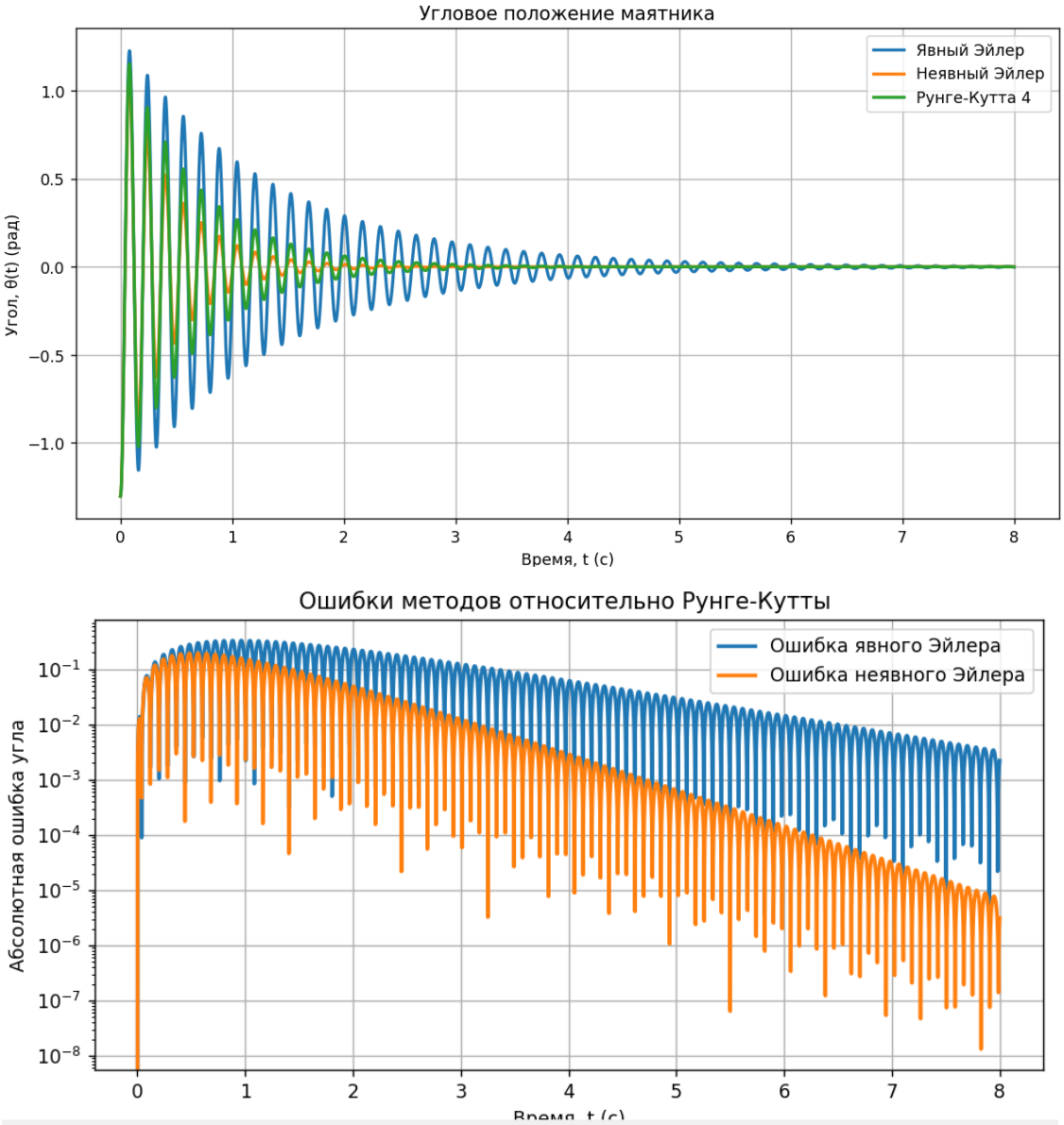
В случае данной задачи, где $\theta_0 = -1.303801445 \approx 74$ град линеаризацию использовать нельзя, для больших углов линеаризация неприменима, поэтому аналитического решения нет.

2. Реализация интеграторов и сравнительная характеристика

Реализация интегральных методов явного и неявного Эйлера, а также метода Рунге-Кутты представлена в приложении. Программа также содержит описание динамики системы с использованием найденного ранее уравнения движения маятника.

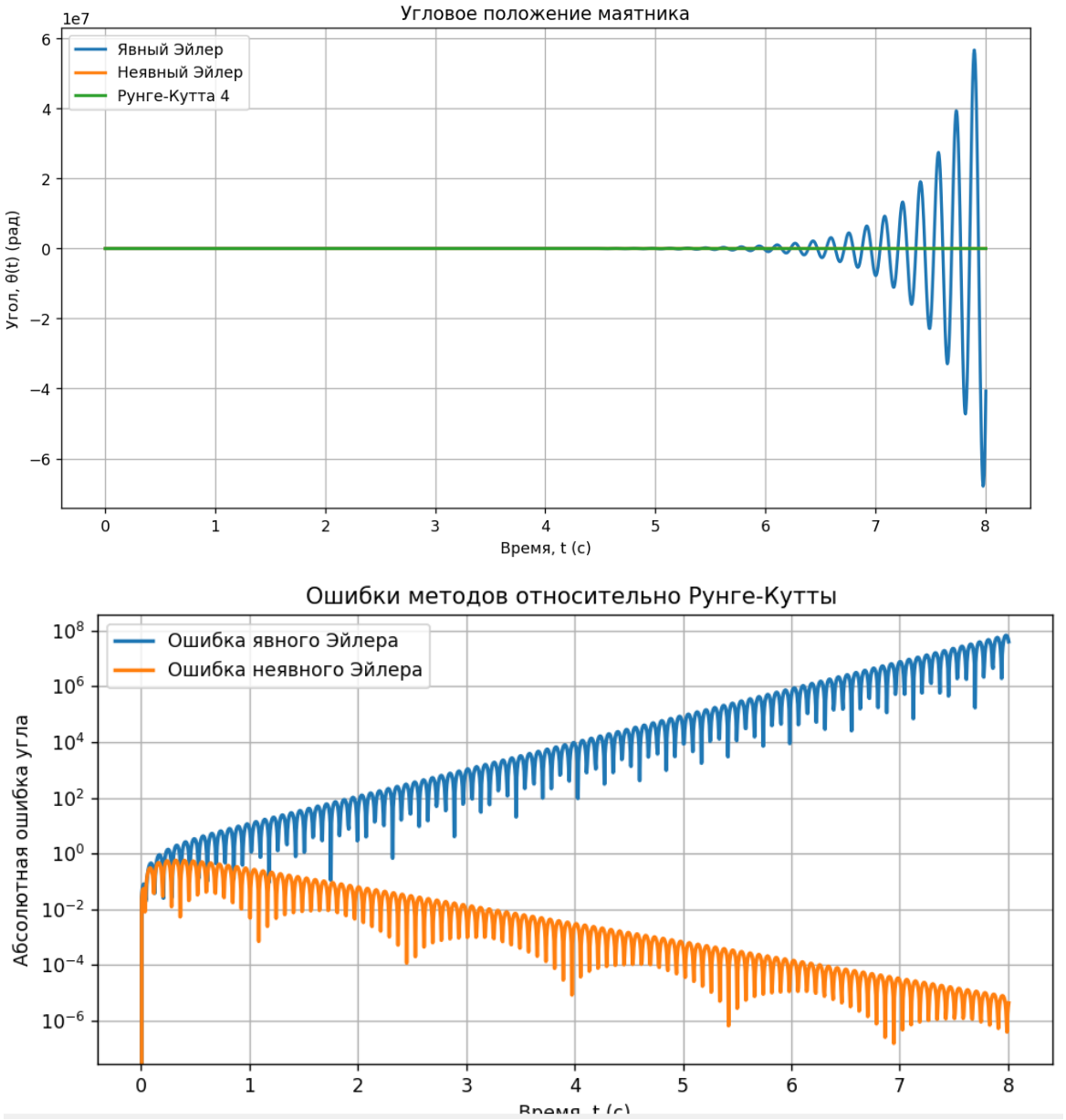
3. Результаты тестирований

1. При $T = 8$, $h = 0{,}001$:

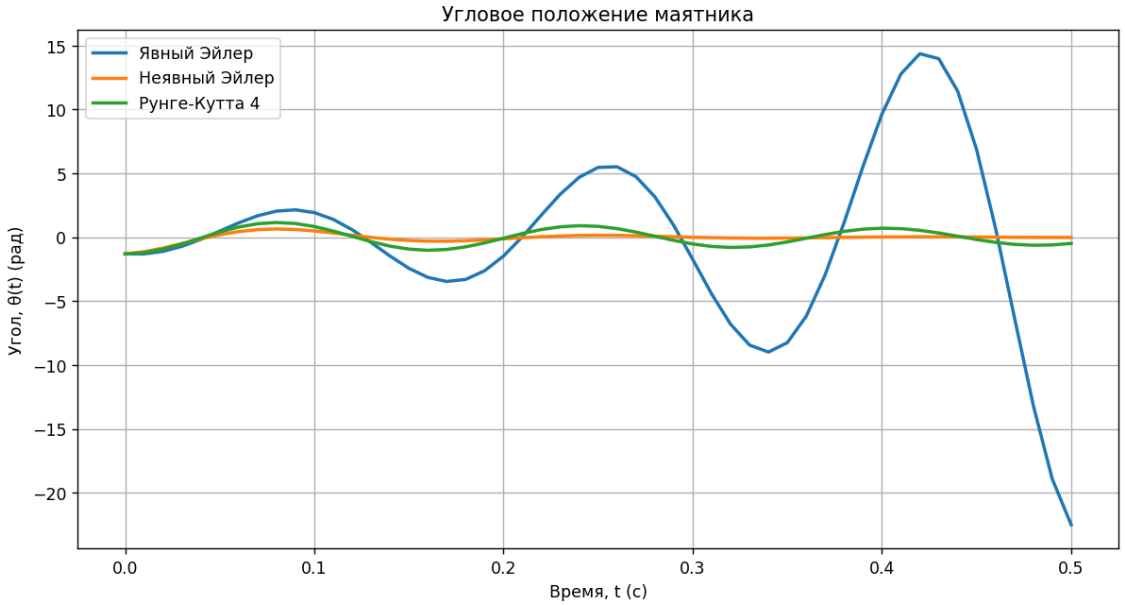


Из-за отсутствия аналитического решения для данной задачи в качестве сравнительной характеристики методов приведена СКО методов Эйлера относительно метода Рунге-Кутты. При данном низком шаге интегрирования ($h = 0{,}001$) наблюдается сходимость для всех методов, в том числе для метода явного Эйлера

2. При $T = 8$, $h = 0{,}01$:



При большом шаге ($h = 0{,}01$) наблюдается «раскачка» и рост ошибки при использовании метода явного Эйлера. Для наглядности также приведено тестирование на маленьком промежутке времени $T = 0{,}5$:



На данном временном промежутке заметно, что метод неявного Эйлера затухает быстрее, чем метод Рунге-Кутты, что связано с более точным сохранением амплитуды колебаний у второго метода. При этом метод явного Эйлера показывает незатухающие колебания с возрастанием амплитуды.

Выводы

По результатам выполненной работы были сделаны следующие выводы:

1. Возможность получить аналитическое решение ОДУ зависит от линейности системы. Для нелинейных систем получить точное аналитическое решение невозможно. Для решения данной проблемы существуют методы линеаризации, которые дают возможность получить приближенное аналитическое решение. При этом для проведения линеаризация необходимо соответствие конкретным условиям, в данной работе – малый угол отклонения маятника. По начальным условиям задачи данный угол не является малым, поэтому для решения ОДУ есть возможность использовать только численные методы.

2. Рассмотренные методы интегрирования показали ожидаемые результаты: методы неявного Эйлера и Рунге-Кутты обеспечили сходимость решения, при этом второй метод обеспечил лучшее сохранение амплитуды колебаний. Метод явного Эйлера при увеличении шага интегрирования показал несходимость решения и увеличение амплитуды колебаний.

ПРИЛОЖЕНИЕ

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

# Параметры маятника с пружиной и демпфером
m = 0.1      # масса, kg
k = 17.4     # коэффициент жесткости пружины, N·m/rad
b = 0.035    # коэффициент демпфирования, N·m·s/rad
l = 0.34     # длина маятника, m
g = 9.81     # ускорение свободного падения, m/s²

def system_dynamics(x):

    x1 = x[0]  # угол θ (положение)
    x2 = x[1]  # угловая скорость θ'

    # Преобразуем уравнение к виду: θ'' = - (b/(m*l²)) * θ' - (k/(m*l²)) * θ -
(g/l) * sin(θ)
    x1_dot = x2  # x1' = x2
    x2_dot = - (b/(m*l**2)) * x2 - (k/(m*l**2)) * x1 - (g/l) * np.sin(x1)

    return np.array([x1_dot, x2_dot])

def analytical_solution(t, x0):

    return np.zeros_like(t)

# Функции интеграторов (оставляем без изменений)
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([-1.303801445, 0])
Tf = 8
h = 0.005


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

x_reference = x_rk4[0, :]

# Визуализация результатов
plt.figure(figsize=(20, 12))

# График углового положения
plt.subplot(2, 2, 1)
plt.plot(t_fe, x_fe[0, :], label='Явный Эйлер', linewidth=2)
plt.plot(t_be, x_be[0, :], label='Неявный Эйлер', linewidth=2)
plt.plot(t_rk4, x_rk4[0, :], label='Рунге-Кутта 4', linewidth=2)
plt.xlabel('Время, t (с)')
plt.ylabel('Угол, θ(t) (рад)')
plt.legend()
plt.title('Угловое положение маятника')
plt.grid(True)


# Ошибки методов относительно метода Рунге-Кутты 4-го порядка
```

```
plt.subplot(2, 2, 4)
error_fe = np.abs(x_fe[0, :] - x_reference)
error_be = np.abs(x_be[0, :len(t_fe)] - x_reference[:len(t_be)])

plt.semilogy(t_fe, error_fe, label='Ошибка явного Эйлера', linewidth=2)
plt.semilogy(t_be, error_be, label='Ошибка неявного Эйлера', linewidth=2)
plt.xlabel('Время, t (с)')
plt.ylabel('Абсолютная ошибка угла')
plt.legend()
plt.title('Ошибки методов относительно Рунге-Кутты ')
plt.grid(True)

plt.tight_layout()
plt.show()

# Вывод среднеквадратичных ошибок
print("Среднеквадратичные ошибки (относительно Рунге-Кутты 4):")
print(f"Явный Эйлер: {np.sqrt(np.mean(error_fe**2)):.6e}")
print(f"Неявный Эйлер: {np.sqrt(np.mean(error_be**2)):.6e}")
```