МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

ОТЧЁТ

ЛАБОРАТОРНАЯ РАБОТА №2

Вариант №14

По дисциплине**: «Иммитационное моделирование роботехнических систем»**

Выполнил:
студент гр. № R4135c
Голубева Я.Д.

Проверил:
ассистент ФСУ и Р
Ракшин Е.А.

Санкт-Петербург
2025

Задание: В соответствии со схемой, представленной на рисунке 1 составить ОДУ системы и решить его аналитически, а также используя методы прямого и обратного Эйлера и RK4. Далее, необходимо описать и сравнить полученные результаты. Сделать выводы.

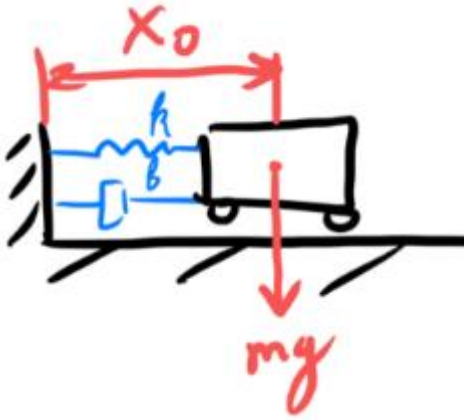


Рисунок 1 – Схема (вариант 2)

Исходные данные для варианта №14:

$m = 0{,}1$ кг;

$k = 10{,}4\ \frac{\text{н}}{\text{м}}$

$b = 0{,}01\ \frac{\text{н} \cdot \text{с}}{\text{м}}$

$x_0 = 0{,}64$ м

1. Аналитический метод

Уравнение Лагранжа в общем виде:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial x'}\right) - \frac{\partial L}{\partial x} = Q$$

Производные:

$$\frac{\partial L}{\partial x'} = m \cdot x'$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial x'}\right) = m \cdot x''$$

$$\frac{\partial L}{\partial x} = -k \cdot \text{x}$$

Таким образом, уравнение системы:

$$m \cdot x'' + b \cdot x' + k \cdot x = 0$$

Составим характеристическое уравнение, где $x(t) = e^{\lambda t}$:

$$m \cdot \lambda^2 + b \cdot \lambda + k = 0$$

Решим, как стандартное квадратное уравнение:

$$D = b^2 - 4 \cdot m \cdot k$$

$$D = 0{,}01^2 - 4 \cdot 0{,}1 \cdot 10{,}4 = -4{,}1599 < 0$$

Тк $D < 0$, то корни уравнения будут комплексными числами:

$$\lambda_{1,2} = \frac{\left[-b \pm i \cdot \sqrt{|D|}\right]}{(2 \cdot m)} = \frac{\left[-0.01 \pm i \cdot \sqrt{4.1599}\right]}{0.2}$$

$$\lambda_{1,2} = \frac{[-0.01 \pm i \cdot 2.0396]}{0.2} = -0.05 \pm i \cdot 10.198$$

Пусть, $\alpha = 0{,}05;\ \omega_d = 10.198 \frac{\text{рад}}{\text{с}}$

В общем виде, решение выглядит следующим образом:

$$x(t) = e^{-\alpha \cdot t} \cdot [C_1 \cdot \cos(\omega_d \cdot t) + C_2 \cdot \sin(\omega_d \cdot t)]$$

Определение C_1 и C_2 осуществляется, исходя из начальных условий, а именно:

$$x(0) = e^0 \cdot [C_1 \cdot \cos(0) + C_2 \cdot \sin(0)] = C_1 = 0.64$$

Скорость, $v = x'(t)$:

$$x'(t) = -\alpha \cdot e^{-\alpha \cdot t} \cdot [C_1 \cdot \cos(\omega_d \cdot t) + C_2 \cdot \sin(\omega_d \cdot t)] +$$

$$+ e^{-\alpha \cdot t} \cdot [-C_1 \cdot \omega_d \cdot \sin(\omega_d \cdot t) + C_2 \cdot \omega_d \cdot \cos(\omega_d \cdot t)]$$

При t=0 с, $v = 0 \frac{\text{м}}{\text{с}}$.

$$x'(0) = -\alpha \cdot C_1 + C_2 \cdot \omega_d = 0$$

Подставим данные в уравнение:

$$-0.05 \cdot 0.64 + C_2 \cdot 10.198 = 0$$

$$-0.032 + C_2 \cdot 10.198 = 0$$

$$C_2 = 0.032 \,/\, 10.198 \approx 0.00314$$

Таким образом, аналитическое решение:

$$x(t) = e^{-0.05 \cdot t} \cdot [0.64 \cdot \cos(10.198 \cdot t) + 0.00314 \cdot \sin(10.198 \cdot t)]$$

Далее, с использованием VS code составим решение системы разными методами: Прямой и обратный Эйлер, RK4, а также выберем предпочтительный для конкретно нашей системы на основании графика сравнения, представленного на рисунке 2.

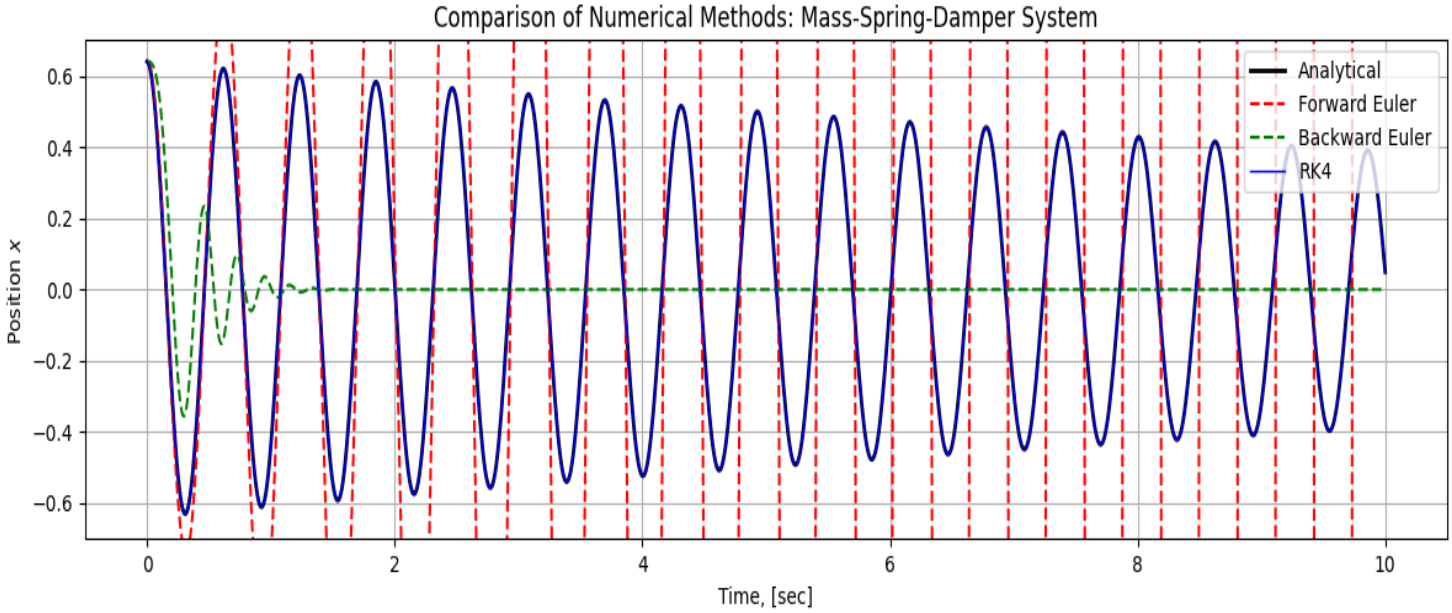


Рисунок 2 - Графики сравнения численных методов

На основании анализа графика можно сделать следующие выводы:

1. Метод RK4 является предпочтительным для численного решения данной системы, ввиду того, что метод RK4 практически полностью совпадает с аналитическим решением, в то время как методы Эйлера имеют заметные ошибки.

2. Графики показали, что метод RK4 практически полностью совпадает с аналитическим решением, в то время как методы Эйлера имеют заметные ошибки.

3. Система совершает слабозатухающие колебания с начальной амплитудой 0,64 м. Из-за малого коэффициента демпфирования колебания затухают очень медленно.

Также рассмотрим вариант моделирования данной системы в Simulink:

Схема которой выглядит следующим образом:

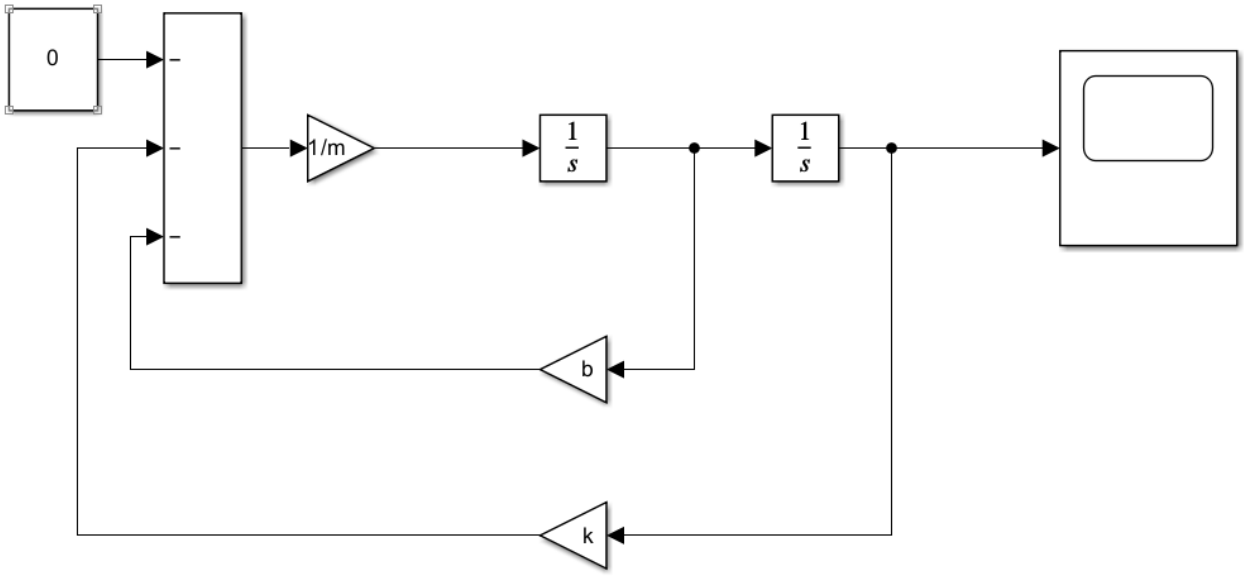
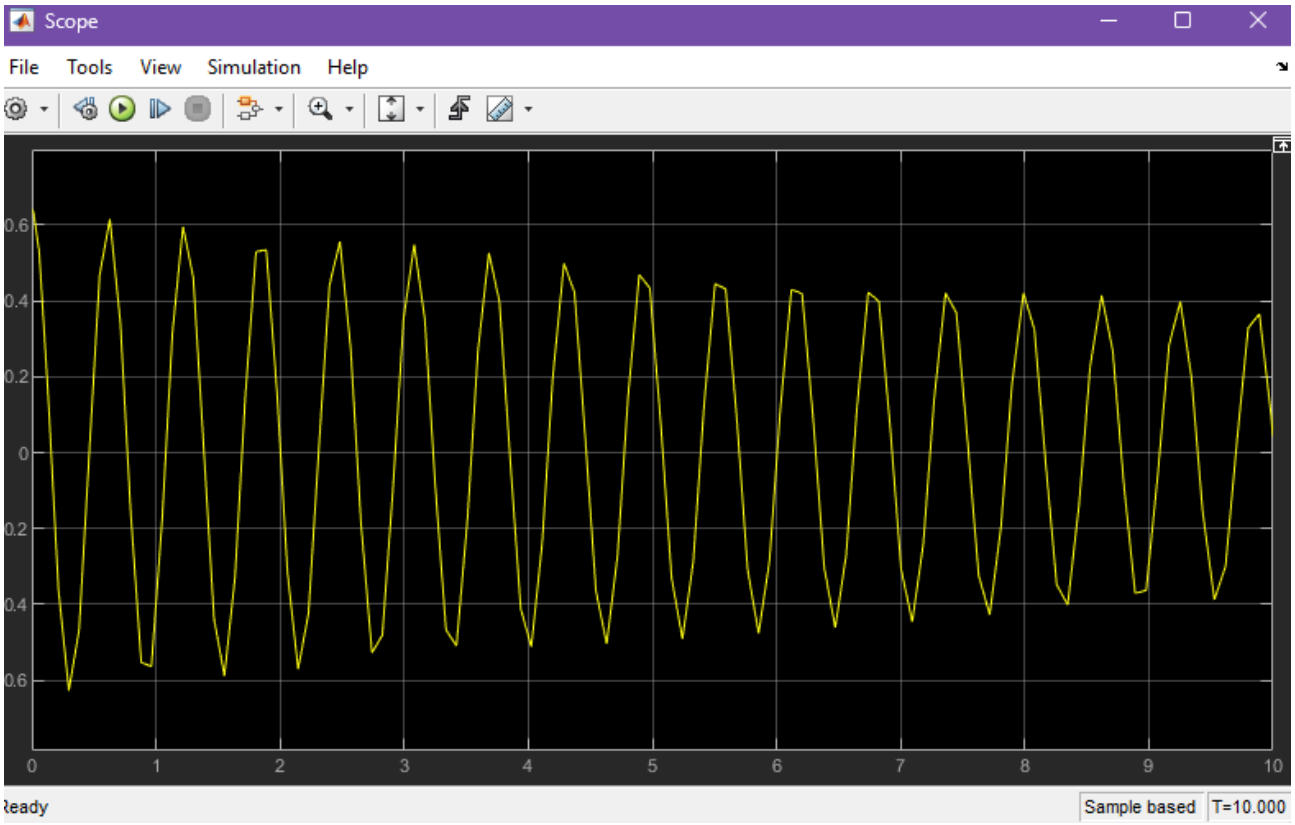


Рисунок 3 - Схема в среде Simulink

При задании параметров в соответствии с исходными данными и интерпретации результатов с помощью Scope, график выглядит следующим образом:



Мы также видим затухающие колебания с заданной амплитудой, что говорит о том, что моделирование было выполнено верно.

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

def mass_spring_system(state:list):
    x0, dx0 = state
    b = 0.01
    m = 0.1
    k = 10.4
    ddx0 = -1/m*(b*dx0+k*x0)
    return np.array([dx0, ddx0])

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

    # Параметры системы для аналитического решения системы
    m = 0.1
    k = 10.4
    b = 0.01

    for k in range(len(t)-1):
        A = np.array([[1, -h],
                      [k*h/m, 1 + b*h/m]])
        b_vec = np.array([x_hist[0, k], x_hist[1, k]])
        x_next = np.linalg.solve(A, b_vec)
        x_hist[:, k+1] = x_next

    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) * (k1 + 2*k2 + 2*k3 + k4)
```

```
    return x_hist, t

def analytical_solution(t, x0=0.64, v0=0.0):
    m = 0.1
    k = 10.4
    b = 0.01

    omega_n = np.sqrt(k/m)
    zeta = b/(2*np.sqrt(m*k))
    omega_d = omega_n * np.sqrt(1 - zeta**2)
    alpha = zeta * omega_n

    A = x0
    B = (v0 + zeta * omega_n * x0) / omega_d

    x = np.exp(-alpha * t) * (A * np.cos(omega_d * t) + B * np.sin(omega_d * t))

    return x

x0 = np.array([0.64, 0])  # начальное положение и скорость
Tf = 10
h = 0.01

# Numerical solutions
x_hist1, t_hist1 = forward_euler(mass_spring_system, x0, Tf, h)
x_hist2, t_hist2 = backward_euler(mass_spring_system, x0, Tf, h)
x_hist3, t_hist3 = runge_kutta4(mass_spring_system, x0, Tf, h)

# Analytical solution
x_analytical = analytical_solution(t_hist1)

# Plot comparison
plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)
plt.plot(t_hist1, x_analytical, 'k-', linewidth=2, label='Analytical')
plt.plot(t_hist1, x_hist1[0,:], 'r--', linewidth=1.5, label='Forward Euler')
plt.plot(t_hist2, x_hist2[0,:], 'g--', linewidth=1.5, label='Backward Euler')
plt.plot(t_hist3, x_hist3[0,:], 'b-', linewidth=1, label='RK4')
plt.xlabel('Time, [sec]')
plt.ylabel('Position $x$')
plt.title('Comparison of Numerical Methods: Mass-Spring-Damper System')
plt.legend()
plt.grid()
plt.ylim(-0.7, 0.7)  # Устанавливаем пределы по y
plt.tight_layout()
plt.show()
```