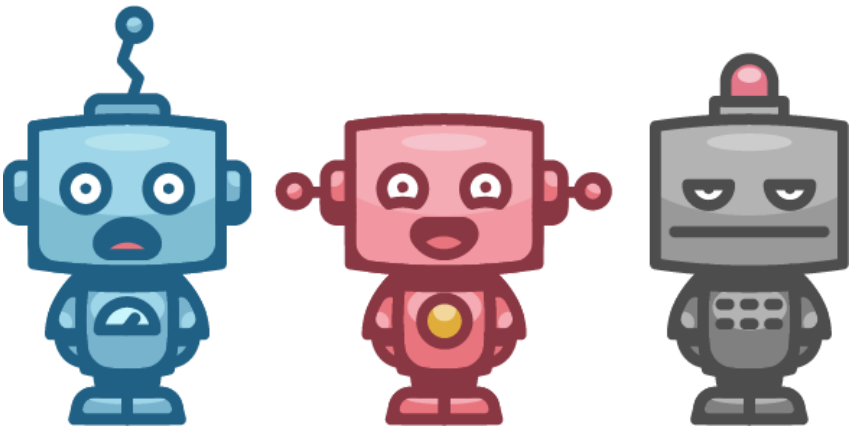
Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ–ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙУНИВЕРСИТЕТ ИТМО

ФСУиР

ЛАОРАТОРНАЯ РАБОТА №1:



Выполнил: Гусаров С.А.
Группа: R4133c

г. Санкт–Петербург, 2025

ОСНОВНАЯ ЧАСТЬ

1 Аналитическое решение

Рассмотрим уравнение вида:

$$7.7\ddot{x} - 4.52\dot{x} - 3.26x + 1.71 = 0 \tag{1.1}$$

Определим решение однородного дифференциального уравнения второго порядка и частного решения неоднородного уравнения:

$$\lambda^2 - 0.587\lambda - 0.423 = -0.222 \tag{1.2}$$

$$\begin{cases}\lambda_1 = -0.42 \\ \lambda_2 = 1\end{cases} \tag{1.3}$$

$$x(t) = c_1 e^{-t} + c_2 e^{0.42t} + x_p(t), \tag{1.4}$$

где $c_1, c_2 -$ коэффициенты, определяемые начальными условиями, $x_p(t) -$ частное решение неоднородного уравнения.

$$x_p(t) = \frac{d}{c} \approx 0.525$$

Пусть $x(0) = \dot{x}(0) = 0$, тогда:

$$\begin{cases}c_1 + c_2 = -0.525 \\ -c_1 + 0.42c_2 = 0\end{cases} \Leftrightarrow \begin{cases}c_1 = -0.15 \\ c_2 = -0.37\end{cases} \tag{1.5}$$

Итого:

$$x(t) = -0.15e^{-t} - 0.37e^{0.42t} + 0.525 \tag{1.6}$$

2 Интеграторы

Сравним полученное аналитическое решение с решениями с помощью методов явного, неявного Эйлера и Рунге-Кутта. Данные полученные в ходе сравнения приведем на графиках рисунков 1 – 2.

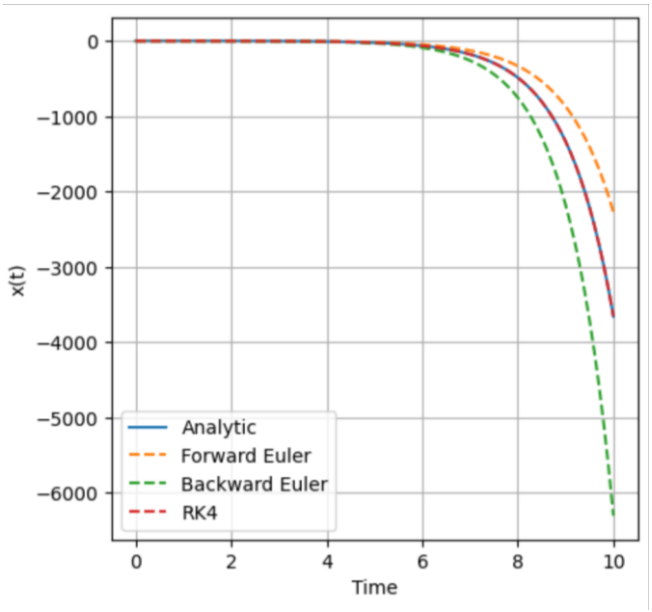


Рисунок 1 – Сравнительное моделирование при $h = 0.1$

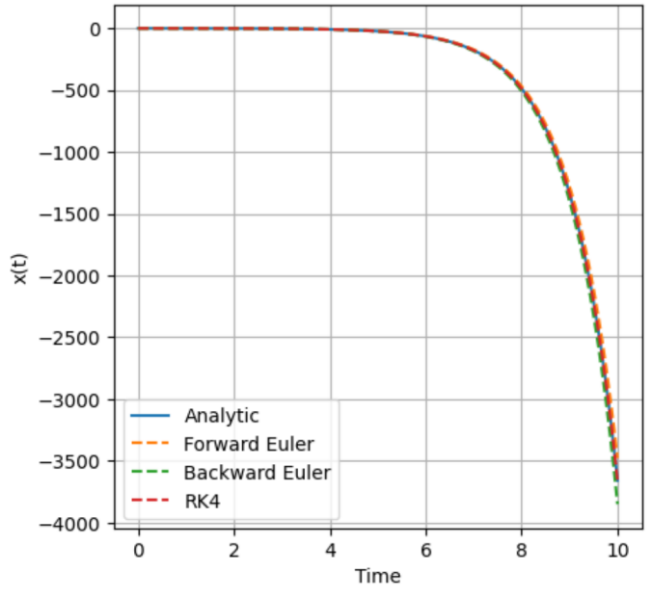


Рисунок 1 – Сравнительное моделирование при $h = 0.01$

ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы мы рассмотрели аналитическое решение дифференциального уравнения, описывающего динамику системы, а также сравнили его с решениями, полученными с помощью методов явного и неявного Эйлера, а также Рунге-Кутта.

Стоит отметить, метод Рунге-Кутта получился наиболее приближенным к аналитическому решению, что видно по графикам рисунков 1 – 2. А также уменьшение шага h решения интеграторов становятся более точными и приближаются к аналитически полученному.

Лабораторную работу считаю выполненной полностью.

Библиотеки

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

Коэффициенты уравнения

```
a = 1
b = -4.52/7.7
c = -3.26/7.7
d = -1.71/7.7
xp = d/c
```

Решение однородного уравнения:

```
D = b**2 - 4*a*c
x1 = (-b + np.sqrt(D))/(2*a)
x2 = (-b - np.sqrt(D))/(2*a)
print(x1,x2)
```

```
1.0073150708625662 -0.42030208384957934
```

Определение c1 и c2:

```
c1 = (x2 * xp)/(x1 - x2)
c2 = -c1 - xp
print(c1,c2)
```

```
-0.15442879960076097 -0.3701110776998525
```

Аналитическое решение

```
def analytic(t):
  return c1*np.exp(x1*t) + c2*np.exp(x2*t) + xp
```

```
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]  # 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

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
```

Визуализация:

```
def dynamic(x_vec):
  x1, x2 = x_vec
  return np.array([x2, (d- b*x2 - c*x1)/a])

# Test all integrators
x0 = np.array([0.0, 0.0])
Tf = 10.0
h = 0.1

# Forward Euler
x_fe, t_fe = forward_euler(dynamic, x0, Tf, h)
# Backward Euler
x_be, t_be = backward_euler(dynamic, x0, Tf, h)
# Runge-Kutta 4
x_rk4, t_rk4 = runge_kutta4(dynamic, x0, Tf, h)

# Plot results
plt.figure(figsize=(5, 5))
plt.plot(t_fe, analytic(t_fe), 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')
plt.ylabel('x(t)')
plt.legend()
plt.grid(True)
plt.show()
```