

# «Аналитическое и численное решение ОДУ второго порядка»

## 1. Аналитическое решение

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

$$2.56\ddot{x}(t) + 2.62\dot{x}(t) - 9.57x(t) = -1.67$$

$$x(0) = 0, \quad \dot{x}(0) = 0$$

### 1.2 решение однородного уравнения

$$2.56\ddot{x}(t) + 2.62\dot{x}(t) - 9.57x(t) = 0$$

$$2.56\lambda^2 + 2.62\lambda - 9.57 = 0$$

$$\Delta = b^2 - 4ac = (2.62)^2 - 4(2.56)(-9.57) = 6.8644 + 97.9968 = 104.8612$$

$$\lambda_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{-2.62 \pm \sqrt{104.8612}}{2(2.56)} = \frac{-2.62 \pm 10.24017}{5.12}$$

$$\lambda_1 = \frac{-2.62 + 10.24017}{5.12} \approx 1.4883$$

$$\lambda_2 = \frac{-2.62 - 10.24017}{5.12} \approx -2.5117$$

Общее решение имеет вид

$$x_h(t) = C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t} = C_1 e^{1.4883t} + C_2 e^{-2.5117t}$$

### 1.3 Частное решение неоднородное уравнение

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

$$2.56(0) + 2.62(0) - 9.57(A) = -1.67$$

$$-9.57A = -1.67 \implies A = \frac{-1.67}{-9.57} \approx 0.1745$$

$$x_p(t) = 0.1745$$

### 1.4 Общее решение и нахождение констант

$$x(t) = C_1 e^{1.4883t} + C_2 e^{-2.5117t} + 0.1745$$

$$\dot{x}(t) = 1.4883C_1 e^{1.4883t} - 2.5117C_2 e^{-2.5117t}$$

$$x(0) = C_1 e^0 + C_2 e^0 + 0.1745 = C_1 + C_2 + 0.1745 = 0$$

$$\dot{x}(0) = 1.4883C_1 e^0 - 2.5117C_2 e^0 = 1.4883C_1 - 2.5117C_2 = 0$$

$$\begin{cases} C_1 + C_2 = -0.1745 \\ 1.4883C_1 - 2.5117C_2 = 0 \end{cases}$$

$$C_1 \approx -0.1095, \quad C_2 \approx -0.065$$

### 1.5 Итоговое аналитическое решение

$$x(t) = -0.1095e^{1.4883t} - 0.065e^{-2.5117t} + 0.1745$$

## 2. Численное решение

Явный метод Эйлера

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h \cdot f(t_k, \mathbf{y}_k)$$

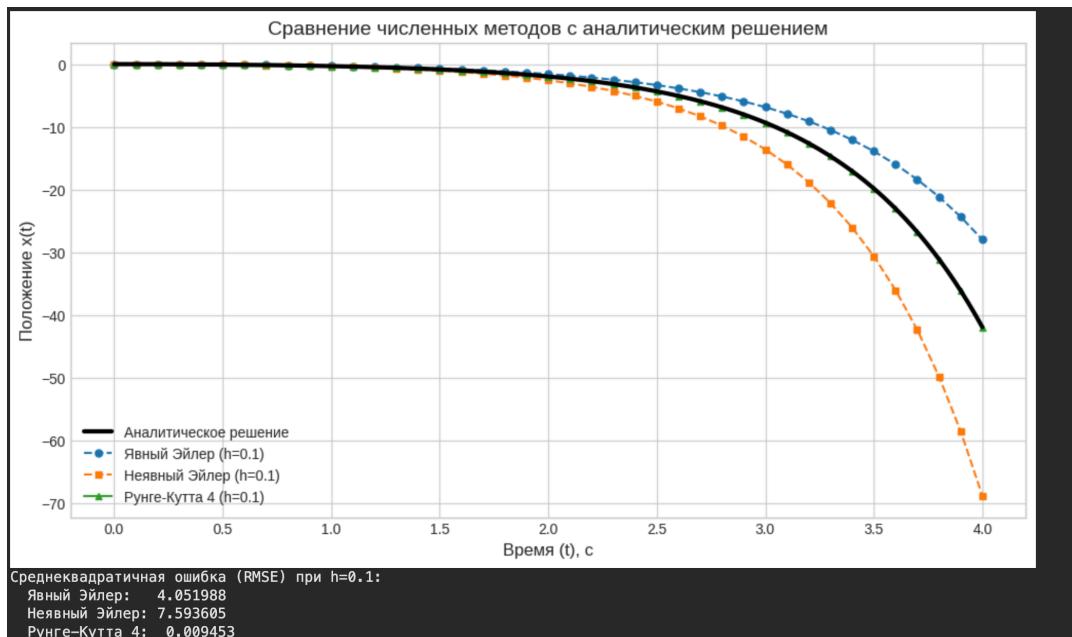
Неявный метод Эйлера

$$y_k + 1 = y_k + h \cdot f(t_k + 1, y_k + 1)$$

## Метод Рунге-Кутты 4 порядка

$$\mathbf{k}_1 = f(t_k, y_k) \quad \mathbf{k}_2 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2}\mathbf{k}_1) \quad \mathbf{k}_3 = f(t_k + \frac{h}{2}, y_k + \frac{h}{2}\mathbf{k}_2) \quad \mathbf{k}_4 = f(t_k + h, y_k + h\mathbf{k}_3) \quad y_k + 1 = y_k + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

## 3. График результата



Графики ошибки RMSE при шаге 0.1

## 4. Общие выводы по методам

В тестировании наибольшую точность продемонстрировал метод Рунге-Кутты 4 порядка.

На всем интервале моделирования от 0 до 4 секунд его график неотличим от графика аналитического решения.

(**чёрная сплошная линия**). Это говорит о том, что накопленная ошибка метода РК4 при шаге  $h=0.1$  пренебрежимо мала.

Явный метод Эйлера (**синяя пунктирная линия**): В начальный момент времени (приблизительно до  $t=2.0$  с) метод показывает хорошую точность. Однако по мере увеличения времени моделирования начинает накапливаться ошибка. График явного Эйлера отклоняется вверх от аналитического решения, то есть метод дает завышенное значение  $x(t)$ . К моменту  $t=4.0$  с отклонение становится уже значительным - аналитическое значение составляет около -40, в то время как решение явным методом Эйлера — примерно -25.

**Неявный метод Эйлера (оранжевая пунктирная линия):** Как и явный метод он достаточно точен вначале. Однако его ошибка также растет со временем, причем даже быстрее, чем у явного метода. Траектория неявного Эйлера отклоняется вниз от аналитического решения, давая заниженное значение  $x(t)$ . К моменту  $t=4.0$  с ошибкой этого метода является наибольшей среди всех: его решение достигает значения около -60, что сильно отличается от истинного значения в - 40.

Интересным наблюдением является то, что явный и неявный методы Эйлера отклоняются от аналитического в разные стороны. Явный метод систематически завышает результат, а неявный — занижает. В нашем конкретном случае для нестабильной системы (решение уходит на минус бесконечность) неявный метод Эйлера показал даже худшую точность, чем явный. Это доказывает, что главное преимущество неявных методов — это их устойчивость на строгих задачах, а не обязательно более высокая точность на всех ОДУ.

Таким образом метод Рунге-Кутты является лучшим выбором для решения нашего дифференциального уравнения. Он обеспечивает высокую точность при умеренном шаге интегрирования 0.1 и это делает его надежным.

Методы Эйлера (явный и неявный) не подходят для точного моделирования данной системы на длительном промежутке времени — они быстро накапливают ошибку и это приводит к искажениям результата.

А выбор численного метода важен и наш пример демонстрирует как методы разного порядка точности дают кардинально отличающиеся по качеству результаты при одинаковых параметрах симуляции. Для получения достоверных результатов в инженерных расчетах следует использовать методы высокого порядка точности, такие как Рунге-Куэты.

## Листинг 1

```

import numpy as np
import matplotlib.pyplot as plt
a = 2.56
b = 2.62
c = -9.57
d = -1.67
def linear_ode_dynamics(y):
    y1 = y[0] # x
    y2 = y[1] # x'
    dy1_dt = y2

    dy2_dt = (d - c * y1 - b * y2) / a
    return np.array([dy1_dt, dy2_dt])
def analytical_solution(t):
    return -0.1095 * np.exp(1.4883 * t) - 0.065 * np.exp(-2.5117 * t) +
0.1745
def forward_euler(fun, y0, Tf, h):
    t = np.arange(0, Tf + h, h)
    y_hist = np.zeros((len(y0), len(t)))
    y_hist[:, 0] = y0
    for k in range(len(t) - 1):
        y_hist[:, k + 1] = y_hist[:, k] + h * fun(y_hist[:, k])
    return y_hist, t
def backward_euler(fun, y0, Tf, h, tol=1e-8, max_iter=100):
    t = np.arange(0, Tf + h, h)
    y_hist = np.zeros((len(y0), len(t)))
    y_hist[:, 0] = y0
    for k in range(len(t) - 1):
        y_next_guess = y_hist[:, k]
        for i in range(max_iter):
            y_next = y_hist[:, k] + h * fun(y_next_guess)
            error = np.linalg.norm(y_next - y_next_guess)
            y_next_guess = y_next
            if error < tol:
                break
        y_hist[:, k + 1] = y_next_guess
    return y_hist, t
def runge_kutta4(fun, y0, Tf, h):
    t = np.arange(0, Tf + h, h)
    y_hist = np.zeros((len(y0), len(t)))
    y_hist[:, 0] = y0
    for k in range(len(t) - 1):
        k1 = fun(y_hist[:, k])
        k2 = fun(y_hist[:, k] + 0.5 * h * k1)
        k3 = fun(y_hist[:, k] + 0.5 * h * k2)
        k4 = fun(y_hist[:, k] + h * k3)
        y_hist[:, k + 1] = y_hist[:, k] + (h / 6.0) * (k1 + 2*k2 + 2*k3
+ k4)
    return y_hist, t
y0 = np.array([0.0, 0.0])
Tf = 4.0
h = 0.1
y_fe, t_fe = forward_euler(linear_ode_dynamics, y0, Tf, h)
y_be, t_be = backward_euler(linear_ode_dynamics, y0, Tf, h)
y_rk4, t_rk4 = runge_kutta4(linear_ode_dynamics, y0, Tf, h)
t_analytical = np.linspace(0, Tf, 500)
x_analytical = analytical_solution(t_analytical)
plt.style.use('seaborn-v0_8-whitegrid')
plt.figure(figsize=(12, 6))
plt.plot(t_analytical, x_analytical, 'k-', label='Аналитическое
решение', linewidth=3, zorder=5)
plt.plot(t_fe, y_fe[0, :], 'o--', label=f'Явный Эйлер (h={h})',
markersize=5)
plt.plot(t_be, y_be[0, :], 's--', label=f'Неявный Эйлер (h={h})',
markersize=5)
plt.plot(t_rk4, y_rk4[0, :], '^-', label=f'Рунге–Кутта 4 (h={h})',
markersize=5)

plt.xlabel('Время (t), с', fontsize=12)
plt.ylabel('Положение x(t)', fontsize=12)
plt.title('Сравнение численных методов с аналитическим решением',
fontsize=14)
plt.legend(fontsize=10)
plt.grid(True)
plt.show()

```