> Выполнил: Литти Тимофей

Теоретическая справка

Рассматривается задача Коши:

$$\frac{du}{dt} = u^{2/3}, \quad u(3) = 0$$

Стандартные методы (Эйлер, Рунге-Кутта) при $u_0 = 0$ дают $u_{i+1} = u_i = 0$, так как производная в нуле равна нулю. Однако точное решение существует и не равно нулю:

$$u(t) = \left(\frac{t}{3} - 1\right)^3$$

Код

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

t_start = 3.0
t_end = 9.0  # Интервал для наглядности развития функции
h = 0.1  # Шаг интегрирования
num_steps = int((t_end - t_start) / h)
t_values = np.linspace(t_start, t_end, num_steps + 1)


# Правая часть ДУ: du/dt = u^(2/3)
def func(t, u):
    return np.abs(u) ** (2 / 3)


# Точное решение: u(t) = (t/3 - 1)^3
def exact_solution(t):
    return (t / 3 - 1) ** 3


# Метод Эйлера
def euler_method(u0, t_vals, h):
    u = np.zeros(len(t_vals))
    u[0] = u0
    for i in range(len(t_vals) - 1):
        u[i + 1] = u[i] + h * func(t_vals[i], u[i])
```

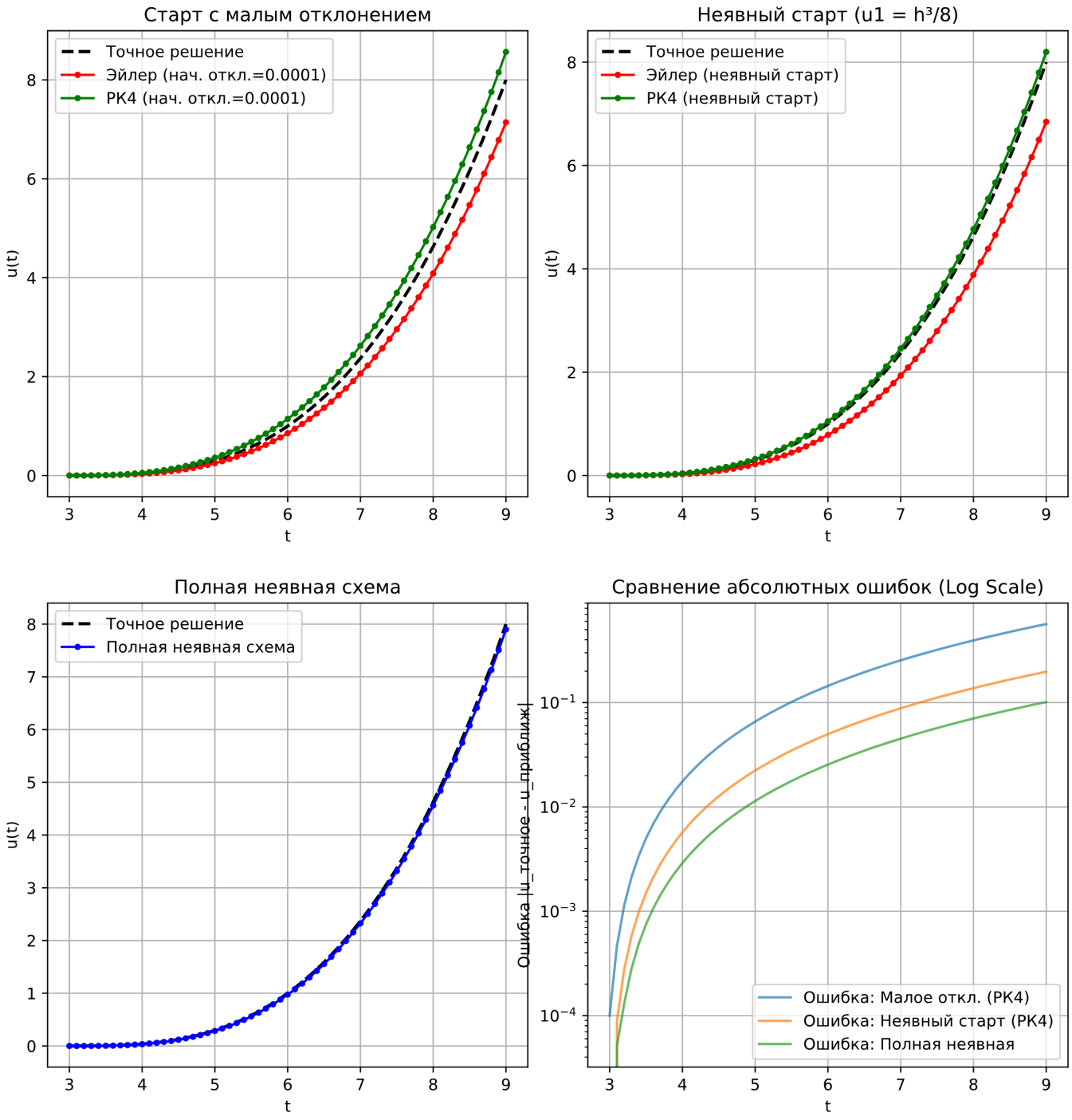
```
    return u


# Метод Рунге-Кутты 4-го порядка
def rk4_method(u0, t_vals, h):
    u = np.zeros(len(t_vals))
    u[0] = u0
    for i in range(len(t_vals) - 1):
        t_i = t_vals[i]
        u_i = u[i]
        k1 = func(t_i, u_i)
        k2 = func(t_i + 0.5 * h, u_i + 0.5 * h * k1)
        k3 = func(t_i + 0.5 * h, u_i + 0.5 * h * k2)
        k4 = func(t_i + h, u_i + h * k3)
        u[i + 1] = u_i + (h / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4)
    return u


# Задание 1: Малое отклонение
epsilon = 1e-4
u_exact = exact_solution(t_values)
u_euler_dev = euler_method(epsilon, t_values, h)
u_rk4_dev = rk4_method(epsilon, t_values, h)

# Задание 2: Неявный старт
u1_start = (h**3) / 8


def solve_with_custom_start(u1, t_vals, h, method_type="euler"):
    u = np.zeros(len(t_vals))
    u[0] = 0
    u[1] = u1
    for i in range(1, len(t_vals) - 1):
        if method_type == "euler":
            u[i + 1] = u[i] + h * func(t_vals[i], u[i])
        elif method_type == "rk4":
            t_i = t_vals[i]
            u_i = u[i]
            k1 = func(t_i, u_i)
            k2 = func(t_i + 0.5 * h, u_i + 0.5 * h * k1)
            k3 = func(t_i + 0.5 * h, u_i + 0.5 * h * k2)
            k4 = func(t_i + h, u_i + h * k3)
            u[i + 1] = u_i + (h / 6.0) * (k1 + 2 * k2 + 2 * k3 + k4)
    return u
```

```python
u_euler_imp_start = solve_with_custom_start(u1_start, t_values, h, "euler")
u_rk4_imp_start = solve_with_custom_start(u1_start, t_values, h, "rk4")


# Задание 3: Полная неявная схема
def full_implicit_scheme(t_vals, h):
    u = np.zeros(len(t_vals))
    u[0] = 0
    for i in range(len(t_vals) - 1):
        u_old = u[i]

        def equation(u_new):
            if u_new < 0:
                u_new = 0
            return (
                u_new
                - u_old
                - (h / 2.0) * (func(t_vals[i], u_old) + func(t_vals[i + 1],
u_new))
            )

        guess = u_old + h * func(t_vals[i], u_old)
        if guess == 0:
            guess = 1e-5
        u[i + 1] = fsolve(equation, guess)[0]
    return u


u_implicit_full = full_implicit_scheme(t_values, h)

# Визуализация с русскими подписями
plt.figure(figsize=(14, 10))

# 1. Малое отклонение
plt.subplot(2, 2, 1)
plt.plot(t_values, u_exact, "k--", label="Точное решение", linewidth=2)
plt.plot(t_values, u_euler_dev, "r.-", label=f"Эйлер (нач. откл.=
{epsilon})")
plt.plot(t_values, u_rk4_dev, "g.-", label=f"РК4 (нач. откл.={epsilon})")
plt.title("Старт с малым отклонением")
plt.xlabel("t")
plt.ylabel("u(t)")
plt.legend()
plt.grid(True)

# 2. Неявный старт
```

```
plt.subplot(2, 2, 2)
plt.plot(t_values, u_exact, "k--", label="Точное решение", linewidth=2)
plt.plot(t_values, u_euler_imp_start, "r.-", label="Эйлер (неявный старт)")
plt.plot(t_values, u_rk4_imp_start, "g.-", label="РК4 (неявный старт)")
plt.title(f"Неявный старт (u1 = h³/8)")
plt.xlabel("t")
plt.ylabel("u(t)")
plt.legend()
plt.grid(True)

# 3. Полная неявная схема
plt.subplot(2, 2, 3)
plt.plot(t_values, u_exact, "k--", label="Точное решение", linewidth=2)
plt.plot(t_values, u_implicit_full, "b.-", label="Полная неявная схема")
plt.title("Полная неявная схема")
plt.xlabel("t")
plt.ylabel("u(t)")
plt.legend()
plt.grid(True)

# 4. Сравнение ошибок
plt.subplot(2, 2, 4)
plt.semilogy(
    t_values, np.abs(u_exact - u_rk4_dev), label="Ошибка: Малое откл.
(РК4)", alpha=0.7
)
plt.semilogy(
    t_values,
    np.abs(u_exact - u_rk4_imp_start),
    label="Ошибка: Неявный старт (РК4)",
    alpha=0.7,
)
plt.semilogy(
    t_values,
    np.abs(u_exact - u_implicit_full),
    label="Ошибка: Полная неявная",
    alpha=0.7,
)
plt.title("Сравнение абсолютных ошибок (Log Scale)")
plt.xlabel("t")
plt.ylabel("Ошибка |u_точное - u_приближ|")
plt.legend()
plt.grid(True)

plt.tight_layout()
```

```
plt.show()
```

Графики



Анализ результатов и выводы

1. **Задание 1 (Малое отклонение):**
   - Введение ϵ (в коде 10^{-4}) позволяет методам "стронуться" с мертвой точки. Однако, точность решения сильно зависит от величины ϵ. Если ϵ слишком велико, мы

вносим начальную ошибку. Если слишком мало — решение долго остается близким к нулю, отставая от точного графика ("lagging").
    - Вывод: Метод рабочий, но неустойчив к выбору величины отклонения.
2. **Задание 2 (Неявный старт):**
    - Использование аналитически выведенного первого шага $u_1 = h^3/8$ дает гораздо более точный результат на начальном этапе.
    - Это позволяет методам (особенно RK4) сразу "встать" на траекторию точного решения без искусственного подбора ϵ.
3. **Задание 3 (Полная неявная схема):**
    - В данном методе на каждом шаге решается нелинейное уравнение $u_i = u_{i-1} + \frac{h}{2}(u_{i-1}^{2/3} + u_i^{2/3})$.
    - Это наиболее устойчивый метод. Как видно из графика ошибок (нижний правый), этот метод может давать высокую точность, но вычислительно он дороже, так как требует использования (`fsolve`) на каждой итерации.