Лабораторная работа №5 «Численные методы решения задачи Коши»

Николаева Ксения, 9 группа

Содержание

1 Постановка задачи

Требуется найти численное решение задачи Коши:

$$u' = \frac{u^2 + ux}{x^2}, \quad x \in [1, 2], \quad u(1) = 0.5$$

Точное решение: $u(x) = \frac{x}{2 - \ln x}$
Необходимо:

- Применить неявный метод трапеций с использованием метода Ньютона
- Применить явный метод Рунге-Кутты 4-го порядка
- Найти решения с шагами $h = 0.1$ и $h/2 = 0.05$
- Сравнить численные решения с точным решением
- Применить правило Рунге для оценки погрешности

2 Применяемые численные методы

2.1 Неявный метод трапеций

Схема метода:

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})]$$

где $f(x, u) = \frac{u^2+ux}{x^2}$.

2.2 Явный метод Рунге-Кутты 4-го порядка

Используется следующая таблица Бутчера:

c	A			
0	0	0	0	0
1/3	1/3	0	0	0
2/3	-1/3	1	0	0
1	1	-1	1	0
	1/8	3/8	3/8	1/8

Схема метода:

$$k_i = h \cdot f\left(x_n + c_i h, y_n + \sum_{j=1}^{i-1} a_{ij} k_j\right), \quad i = 1, 2, 3, 4$$

$$y_{n+1} = y_n + \sum_{i=1}^{4} b_i k_i$$

3 Итерационный процесс метода Ньютона для неявного метода трапеций

Для решения нелинейного уравнения:

$$F(y_{n+1}) = y_{n+1} - y_n - \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] = 0$$

применяется метод Ньютона:

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} - \frac{F(y_{n+1}^{(k)})}{F'(y_{n+1}^{(k)})}$$

где:

$$F'(y_{n+1}) = 1 - \frac{h}{2}\frac{\partial f}{\partial u}(x_{n+1}, y_{n+1})$$
$$\frac{\partial f}{\partial u}(x, u) = \frac{2u + x}{x^2}$$

Начальное приближение выбирается с помощью явного метода Эйлера:

$$y_{n+1}^{(0)} = y_n + h \cdot f(x_n, y_n)$$

4 Правило Рунге оценки погрешности

Для метода порядка точности p правило Рунге дает оценку:

$$\frac{\|y^h - y^{h/2}\|_{\omega_h}}{2^p - 1} = \frac{\max_{i=\overline{0,N}} |y_i^h - y_{2i}^{h/2}|}{2^p - 1}$$

где:

- Для неявного метода трапеций: $p = 2$
- Для метода Рунге-Кутты 4-го порядка: $p = 4$

5 Результаты вычислительного эксперимента

5.1 Параметры расчета

- Интервал: $x \in [1, 2]$
- Начальное условие: $u(1) = 0.5$
- Шаги: $h = 0.1$, $h/2 = 0.05$
- Количество узлов: $N = 10$ для $h = 0.1$, $N = 20$ для $h = 0.05$

5.2 Таблица численных результатов

x	Точное решение	Трапеции ($h/2$)	Рунге-Кутты ($h/2$)	Погр. Трап.	Погр. РК
1.00	0.500000	0.500000	0.500000	0.00e+00	0.00e+00
1.10	0.577522	0.577525	0.577522	3.14e-06	2.74e-08
1.20	0.660183	0.660190	0.660183	7.42e-06	5.54e-08
1.30	0.748143	0.748156	0.748143	1.29e-05	8.47e-08
1.40	0.841585	0.841605	0.841585	1.98e-05	1.16e-07
1.50	0.940713	0.940741	0.940713	2.82e-05	1.50e-07
1.60	1.045754	1.045792	1.045754	3.82e-05	1.86e-07
1.70	1.156957	1.157007	1.156957	5.02e-05	2.26e-07
1.80	1.274595	1.274659	1.274595	6.41e-05	2.70e-07
1.90	1.398966	1.399046	1.398966	8.04e-05	3.18e-07
2.00	1.530394	1.530493	1.530394	9.93e-05	3.71e-07

5.3 Сводка результатов

- Неявный метод трапеций:
  - Максимальная погрешность: 9.927550×10^{-5}
  - Оценка по правилу Рунге: 9.956834×10^{-5}
- Явный метод Рунге-Кутты 4-го порядка:
  - Максимальная погрешность: 3.714007×10^{-7}
  - Оценка по правилу Рунге: 3.472047×10^{-7}

6 Выводы

1. Метод Рунге-Кутты 4-го порядка показал значительно более высокую точность (погрешность $\sim 10^{-7}$) по сравнению с методом трапеций (погрешность $\sim 10^{-5}$), что соответствует их порядкам точности (4 и 2 соответственно).
2. Оценки погрешности по правилу Рунге хорошо согласуются с фактическими погрешностями, подтверждая корректность реализации методов.
3. При уменьшении шага в 2 раза погрешность уменьшается примерно в 2^p раз, где p - порядок метода, что соответствует теоретическим ожиданиям.
4. Неявный метод трапеций требует решения нелинейного уравнения на каждом шаге, что увеличивает вычислительную сложность, но обеспечивает лучшую устойчивость.
5. Для задач, требующих высокой точности, рекомендуется использовать метод Рунге-Кутты 4-го порядка, а для жестких задач - неявный метод трапеций.

7 Листинг программы с комментариями

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

def f(x, u):
    """Правая часть дифференциального уравнения: u' = (u^2 + ux)/x^2"""
    return (u**2 + u*x) / (x**2)

def exact_solution(x):
    """Точное решение: u(x) = x/(2 - ln(x))"""
    return x / (2 - np.log(x))

def df_du(x, u):
    """Частная производная f по u для метода Ньютона"""
    return (2*u + x) / (x**2)

def implicit_trapezoidal_newton(x0, u0, h, x_end):
    """
    Неявный метод трапеций с использованием метода Ньютона
        x0: начальная точка
        u0: начальное условие
        h: шаг сетки
        x_end: конечная точка

        x_vals, u_vals: массивы значений x и u
    """
    n = int((x_end - x0) / h)
    x_vals = np.linspace(x0, x_end, n + 1)
    u_vals = np.zeros(n + 1)
    u_vals[0] = u0

    for i in range(n):
        x_curr = x_vals[i]
        x_next = x_vals[i + 1]
        u_curr = u_vals[i]

        # Функция для метода Ньютона: F(u_next) = u_next - u_curr - h/2 * (f(x_curr, u_c
        def F(u_next):
            return u_next - u_curr - h/2 * (f(x_curr, u_curr) + f(x_next, u_next))

        # Производная F по u_next для метода Ньютона
        def dF_du(u_next):
            return 1 - h/2 * df_du(x_next, u_next)

        # Начальное приближение (явный метод Эйлера)
        u_next_0 = u_curr + h * f(x_curr, u_curr)

        # Метод Ньютона
```

```
        u_next = u_next_0
        max_iter = 10
        tol = 1e-10

        for _ in range(max_iter):
            F_val = F(u_next)
            if abs(F_val) < tol:
                break
            dF_val = dF_du(u_next)
            u_next = u_next - F_val / dF_val

        u_vals[i + 1] = u_next

    return x_vals, u_vals

def runge_kutta_4_explicit(x0, u0, h, x_end):
    """
    Явный метод Рунге-Кутты 4-го порядка с заданной таблицей Бутчера

    Таблица Бутчера:
    0   | 0   0   0   0
    1/3 | 1/3 0   0   0
    2/3 |-1/3 1   0   0
    1   | 1  -1   1   0
    ----+---------------
        | 1/8 3/8 3/8 1/8
    """
    n = int((x_end - x0) / h)
    x_vals = np.linspace(x0, x_end, n + 1)
    u_vals = np.zeros(n + 1)
    u_vals[0] = u0

    # Коэффициенты таблицы Бутчера
    c = np.array([0, 1/3, 2/3, 1])
    a = np.array([[0, 0, 0, 0],
                  [1/3, 0, 0, 0],
                  [-1/3, 1, 0, 0],
                  [1, -1, 1, 0]])
    b = np.array([1/8, 3/8, 3/8, 1/8])

    for i in range(n):
        x_curr = x_vals[i]
        u_curr = u_vals[i]

        # Вычисление k_i
        k = np.zeros(4)

        for j in range(4):
            x_stage = x_curr + c[j] * h
            u_stage = u_curr
```

```
            for l in range(j):
                u_stage += a[j, l] * k[l]
            k[j] = h * f(x_stage, u_stage)

        # Вычисление следующего значения
        u_vals[i + 1] = u_curr + np.sum(b * k)

    return x_vals, u_vals

def compute_max_norm_error(u_exact, u_numerical):
    """Вычисление максимальной нормы погрешности"""
    return np.max(np.abs(u_exact - u_numerical))

def runge_rule_estimate(y_h, y_h2, p):
    """
    Правило Рунге для оценки погрешности
    Args:
        y_h: решение с шагом h
        y_h2: решение с шагом h/2
        p: порядок точности метода
    """
    # Берем значения y_h2 в точках, соответствующих сетке с шагом h
    y_h2_coarse = y_h2[::2]  # каждое второе значение

    max_diff = np.max(np.abs(y_h - y_h2_coarse))
    return max_diff / (2**p - 1)

# Параметры задачи
x0, x_end = 1, 2
u0 = 0.5
h1 = 0.1
h2 = h1 / 2

print("Задача Коши:")
print("u' = (u² + ux)/x², x  [1,2]")
print("u(1) = 0.5")
print("Точное решение: u(x) = x/(2 - ln(x))\n")

# Решение неявным методом трапеций
print("1. Неявный метод трапеций:")
print(f"Решение с шагом h = {h1}")
x_trap_h, u_trap_h = implicit_trapezoidal_newton(x0, u0, h1, x_end)

print(f"Решение с шагом h/2 = {h2}")
x_trap_h2, u_trap_h2 = implicit_trapezoidal_newton(x0, u0, h2, x_end)

# Точное решение
u_exact_h2 = exact_solution(x_trap_h2)

# Погрешность для метода трапеций
```

```
error_trap = compute_max_norm_error(u_exact_h2, u_trap_h2)
print(f"||u - y^(h/2)||_ = {error_trap:.6e}")

# Правило Рунге для метода трапеций (p = 2)
runge_trap = runge_rule_estimate(u_trap_h, u_trap_h2, 2)
print(f"Оценка по правилу Рунге: {runge_trap:.6e}\n")

# Решение методом Рунге-Кутты
print("2. Явный метод Рунге-Кутты 4-го порядка:")
print(f"Решение с шагом h = {h1}")
x_rk_h, u_rk_h = runge_kutta_4_explicit(x0, u0, h1, x_end)

print(f"Решение с шагом h/2 = {h2}")
x_rk_h2, u_rk_h2 = runge_kutta_4_explicit(x0, u0, h2, x_end)

# Погрешность для метода Рунге-Кутты
error_rk = compute_max_norm_error(u_exact_h2, u_rk_h2)
print(f"||u - y^(h/2)||_ = {error_rk:.6e}")

# Правило Рунге для метода Рунге-Кутты (p = 4)
runge_rk = runge_rule_estimate(u_rk_h, u_rk_h2, 4)
print(f"Оценка по правилу Рунге: {runge_rk:.6e}\n")

# Построение графиков
plt.figure(figsize=(15, 10))

# График для метода трапеций
plt.subplot(2, 2, 1)
plt.plot(x_trap_h2, u_exact_h2, 'r-', linewidth=2, label='Точное решение')
plt.plot(x_trap_h2, u_trap_h2, 'b--', linewidth=2, label='Неявный метод трапеций (h/2)')
plt.plot(x_trap_h, u_trap_h, 'g:', linewidth=2, label='Неявный метод трапеций (h)')
plt.xlabel('x')
plt.ylabel('u(x)')
plt.title('Неявный метод трапеций')
plt.legend()
plt.grid(True)

# График для метода Рунге-Кутты
plt.subplot(2, 2, 2)
plt.plot(x_rk_h2, u_exact_h2, 'r-', linewidth=2, label='Точное решение')
plt.plot(x_rk_h2, u_rk_h2, 'b--', linewidth=2, label='Рунге-Кутты 4 (h/2)')
plt.plot(x_rk_h, u_rk_h, 'g:', linewidth=2, label='Рунге-Кутты 4 (h)')
plt.xlabel('x')
plt.ylabel('u(x)')
plt.title('Явный метод Рунге-Кутты 4-го порядка')
plt.legend()
plt.grid(True)

# График погрешностей для метода трапеций
plt.subplot(2, 2, 3)
```

```
error_trap_vals = np.abs(u_exact_h2 - u_trap_h2)
plt.semilogy(x_trap_h2, error_trap_vals, 'b-', linewidth=2)
plt.xlabel('x')
plt.ylabel('|u(x) - y(x)|')
plt.title('Погрешность метода трапеций')
plt.grid(True)

# График погрешностей для метода Рунге-Кутты
plt.subplot(2, 2, 4)
error_rk_vals = np.abs(u_exact_h2 - u_rk_h2)
plt.semilogy(x_rk_h2, error_rk_vals, 'b-', linewidth=2)
plt.xlabel('x')
plt.ylabel('|u(x) - y(x)|')
plt.title('Погрешность метода Рунге-Кутты')
plt.grid(True)

plt.tight_layout()
plt.show()

# Таблица результатов
print("3. Таблица численных результатов:")
print("x\t\tТочное\t\tТрапеции(h/2)\tРунге-Кутты(h/2)\tПогр.Трап.\tПогр.РК")
print("-" * 80)
for i in range(0, len(x_trap_h2), 2):  # каждое второе значение для читаемости
    x_val = x_trap_h2[i]
    u_exact_val = u_exact_h2[i]
    u_trap_val = u_trap_h2[i]
    u_rk_val = u_rk_h2[i]
    err_trap = abs(u_exact_val - u_trap_val)
    err_rk = abs(u_exact_val - u_rk_val)
    print(f"{x_val:.2f}\t\t{u_exact_val:.6f}\t{u_trap_val:.6f}\t\t{u_rk_val:.6f}\t\t{err

print(f"\n4. Сводка результатов:")
print(f"Неявный метод трапеций:")
print(f"  - Максимальная погрешность: {error_trap:.6e}")
print(f"  - Оценка по правилу Рунге: {runge_trap:.6e}")
print(f"Явный метод Рунге-Кутты 4-го порядка:")
print(f"  - Максимальная погрешность: {error_rk:.6e}")
print(f"  - Оценка по правилу Рунге: {runge_rk:.6e}")
```