Лабараторная Работа № 3
Численное решение смешанной задачи для уравнения теплопроводности

Царик Виталий
3-й курс 2-я группа

13 декабря 2019 г.

1 Постановка задачи

Условие

На сетке узлов $\overline{\omega}_{h\tau}$ найти численное решение смешанной задачи для одномерного уравнения теплопроводности с использованием:

- явной разностной схемы с $\tau = h = 0.1$ и $h = 0.1, \tau = \frac{h^2}{2}$
- чисто неявной разностной схемы с $\tau = h = 0.1$
- разностной схемы Кранка-Николсон с $\tau = h = 0.1$

Выписать соответствующие разностные схемы, указать их порядок аппроксимации, указать являются ли схемы абсолютно устойчивыми по начальным данным. Вычислить погрешность численного решения (т.е. найти $\max_{i,j} |y_i^j - u_i^j|$). Построить графики, демонстрирующие устойчивое и неустойчивое поведение явной разностной схемы.

Задача

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial^2 x} - \dfrac{x^2 + 2t + 3}{(t+1)^2}, \qquad 0 < x < 1, \quad 0 < t < 0.5 \\ u(x,0) = x^2 + 1, \qquad 0 \le t \le 0.5 \\ u(0,t) = \dfrac{1}{t+1}, \qquad 0 \le t \le 0.5 \\ u(1,t) = \dfrac{2}{t+1}, \qquad 0 \le t \le 0.5 \end{cases} \tag{1}$$

Точное решение

$$u(x,t) = \frac{x^2+1}{t+1} \tag{2}$$

2 Краткие теоретические сведения

Явная разностная схема

$$\begin{cases} y_t = y_{\overline{x}x} - \dfrac{x^2+2t+3}{(t+1)^2} \\ y(x,0) = x^2+1 \\ y(0,t) = \dfrac{1}{t+1} \\ y(1,t) = \dfrac{2}{t+1} \end{cases}$$

Индексная форма

$$\begin{cases} \dfrac{y_i^{j+1} - y_i^j}{\tau} = \dfrac{y_{i+1}^j + y_{i-1}^j - 2y_i^j}{h^2} - \dfrac{x_i^2 + 2t_j + 3}{(t_j+1)^2}, \qquad i = \overline{1, N_1 - 1}, \quad j = \overline{0, N_2 - 1} \\ y_i^0 = x_i^2 + 1, \qquad i = \overline{0, N_1} \\ y_0^j = \dfrac{1}{t_j+1}, \qquad j = \overline{0, N_2} \\ y_N^j = \dfrac{2}{t_j+1}, \qquad j = \overline{0, N_2} \end{cases}$$

$$y_i^{j+1} = \tau\left(\frac{y_{i+1}^j + y_{i-1}^j - 2y_i^j}{h^2} - \frac{x_i^2+2t_j+3}{(t_j+1)^2}\right) + y_i^j, \qquad i = \overline{1, N_1 - 1}, \quad j = \overline{0, N_2 - 1}$$

Порядок аппроксимации: $O(\tau + h^2)$

Устойчивость: условная (при $\tau \le \frac{h^2}{2}$)

Неявная разностная схема

$$\begin{cases} y_t = \hat{y}_{\overline{x}x} - \dfrac{x^2+2t+3}{(t+1)^2} \\ y(x,0) = x^2+1 \\ y(0,t) = \dfrac{1}{t+1} \\ y(1,t) = \dfrac{2}{t+1} \end{cases}$$

Индексная форма

$$\begin{cases} \dfrac{y_i^{j+1}-y_i^j}{\tau} = \dfrac{y_{i+1}^{j+1}+y_{i-1}^{j+1}-2y_i^{j+1}}{h^2} - \dfrac{x_i^2+2t_j+3}{(t_j+1)^2}, \qquad i=\overline{1,N_1-1}, \quad j=\overline{0,N_2-1} \\ y_i^0 = x_i^2+1, \qquad i=\overline{0,N_1} \\ y_0^j = \dfrac{1}{t_j+1}, \qquad j=\overline{0,N_2} \\ y_N^j = \dfrac{2}{t_j+1}, \qquad j=\overline{0,N_2} \end{cases}$$

На каждом шаге $j=\overline{0,N_2-1}$ решаем систему методом прогонки

$$\begin{cases} a_i = -\dfrac{1}{h^2}, \qquad i=\overline{1,N_1-1} \\ a_{N_1} = 0 \\ b_0 = 0 \\ b_i = -\dfrac{1}{h^2}, \qquad i=\overline{1,N_1-1} \\ c_0 = 1 \\ c_i = -\left(\dfrac{2}{h^2}+\dfrac{1}{\tau}\right), \qquad i=\overline{1,N_1-1} \\ c_{N_1} = 1 \\ f_0 = \dfrac{1}{t_j+1} \\ f_i = \dfrac{x_i^2+2t_j+3}{(t_j+1)^2} - \dfrac{y_i^j}{\tau}, \qquad i=\overline{1,N_1-1} \\ f_{N_1} = \dfrac{2}{t_j+1} \end{cases}$$

Порядок аппроксимации: $O(\tau+h^2)$

Устойчивость: абсолютная

Разностная схема Кранка-Николсон

$$\begin{cases} y_t = \dfrac{y_{\overline{x}x} + \hat{y}_{\overline{x}x}}{2} - \dfrac{x^2 + 2t + 3}{(t+1)^2} \\ y(x,0) = x^2 + 1 \\ y(0,t) = \dfrac{1}{t+1} \\ y(1,t) = \dfrac{2}{t+1} \end{cases}$$

Индексная форма

$$\begin{cases} \dfrac{y_i^{j+1} - y_i^j}{\tau} = \dfrac{y_{i+1}^j + y_{i-1}^j - 2y_i^j + y_{i+1}^{j+1} + y_{i-1}^{j+1} - 2y_i^{j+1}}{2h^2} - \dfrac{x_i^2 + 2t_{j+\frac{1}{2}} + 3}{(t_{j+\frac{1}{2}} + 1)^2}, \qquad i = \overline{1, N_1 - 1}, \quad j = \overline{0, N_2 - 1} \\ y_i^0 = x_i^2 + 1, \qquad i = \overline{0, N_1} \\ y_0^j = \dfrac{1}{t_j + 1}, \qquad j = \overline{0, N_2} \\ y_N^j = \dfrac{2}{t_j + 1}, \qquad j = \overline{0, N_2} \end{cases}$$

Аналогично, чисто неявной разностной схеме

$$\begin{cases} a_i = -\dfrac{1}{2h^2}, \qquad i = \overline{1, N_1 - 1} \\ a_{N_1} = 0 \\ b_0 = 0 \\ b_i = -\dfrac{1}{2h^2}, \qquad i = \overline{1, N_1 - 1} \\ c_0 = 1 \\ c_i = -\left(\dfrac{1}{h^2} + \dfrac{1}{\tau}\right), \qquad i = \overline{1, N_1 - 1} \\ c_{N_1} = 1 \\ f_0 = \dfrac{1}{t_j + 1} \\ f_i = \dfrac{x_i^2 + 2t_j + 3}{(t_j + 1)^2} - \dfrac{y_i^j}{\tau} - \dfrac{y_{i+1}^j + y_{i-1}^j - 2y_i^j}{2h^2}, \qquad i = \overline{1, N_1 - 1} \\ f_{N_1} = \dfrac{2}{t_j + 1} \end{cases}$$

Порядок аппроксимации: $O(\tau^2 + h^2)$

Устойчивость: абсолютная

3 Листинг программы

Листинг 1: main.py

```
import numpy as np

from matrix_solver import solve_tridiag
from utils import plot, error

h = 0.1
tau = 0.1
(x0, x1) = (0, 1)
(t0, t1) = (0, 1)


def fi(x, t):
    return (x ** 2 + 2 * t + 3) / ((t + 1) ** 2)


def exact_solution(x, t):
    return (x ** 2 + 1) / (t + 1)


def init_data(h, tau):
    x = np.arange(x0, x1 + h, h)
    t = np.arange(t0, t1 + tau, tau)
    n1 = len(x)
    n2 = len(t)

    y = np.empty((n1, n2))

    for i in range(n1):
        y[i, 0] = x[i] ** 2 + 1

    return x, t, n1, n2, y


def solve_explicit(h, tau):
    x, t, n1, n2, y = init_data(h, tau)

    for j in range(n2):
        y[0, j] = 1 / (t[j] + 1)
        y[-1, j] = 2 / (t[j] + 1)

    for j in range(n2 - 1):
        for i in range(1, n1 - 1):
            y[i, j + 1] = tau * ((y[i + 1, j] + y[i - 1, j] - 2 * y[i, j])
                                                              / (h ** 2) - fi(x[i], t[
                                                            j])) + y[i, j]

```

```python
    return x, t, y


def solve_implicit(h, tau, iteration):
    x, t, n1, n2, y = init_data(h, tau)

    for j in range(n2 - 1):
        y[:, j + 1] = iteration(h, tau, y, x, t, j, n1)

    return x, t, y


def pure_implicit_iteration(h, tau, y, x, t, j, n):
    a = np.empty(n - 1)
    av = np.vectorize(lambda x: -1 / h ** 2)
    a[:-1] = av(x[1:-1])
    a[-1] = 0

    b = np.empty(n - 1)
    b[0] = 0
    bv = np.vectorize(lambda x: -1 / h ** 2)
    b[1:] = bv(x[1:-1])

    c = np.empty(n)
    c[0] = 1
    cv = np.vectorize(lambda x: -(2 / h ** 2 + 1 / tau))
    c[1:-1] = cv(x[1:-1])
    c[-1] = 1

    f = np.empty(n)
    f[0] = 1 / (t[j] + 1)
    fv = np.vectorize(lambda x, y: fi(x, t[j]) - y / tau)
    f[1:-1] = fv(x[1:-1], y[1:-1, j])
    f[-1] = 2 / (t[j] + 1)

    return solve_tridiag(a, b, c, f)


def crank_nicolson_iteration(h, tau, y, x, t, j, n):
    a = np.empty(n - 1)
    av = np.vectorize(lambda x: -1 / (2 * h ** 2))
    a[:-1] = av(x[1:-1])
    a[-1] = 0

    b = np.empty(n - 1)
    b[0] = 0
    bv = np.vectorize(lambda x: -1 / (2 * h ** 2))
    b[1:] = bv(x[1:-1])

    c = np.empty(n)
    c[0] = 1
    cv = np.vectorize(lambda x: -(1 / h ** 2 + 1 / tau))
    c[1:-1] = cv(x[1:-1])
    c[-1] = 1
```

```python

    f = np.empty(n)
    f[0] = 1 / (t[j + 1] + 1)
    fv = np.vectorize(lambda x, i:
                      fi(x, t[j]) - y[i, j] / tau - (y[i + 1, j] + y[i - 1, j] - 2 * y[i, j]) / (2 * h ** 2))
    f[1:-1] = fv(x[1:-1], range(1, n - 1))
    f[-1] = 2 / (t[j + 1] + 1)

    return solve_tridiag(a, b, c, f)


if __name__ == '__main__':
    x, t, y = solve_explicit(h, tau)
    plot(x, t, y)
    print(error(y, x, t, exact_solution))

    x, t, y = solve_explicit(h, h ** 2 / 2)
    plot(x, t, y)
    print(error(y, x, t, exact_solution))

    x, t, y = solve_implicit(h, tau, pure_implicit_iteration)
    print(error(y, x, t, exact_solution))

    x, t, y = solve_implicit(h, tau, crank_nicolson_iteration)
    print(error(y, x, t, exact_solution))
```

Листинг 2: matrix_solver.py

```python
import numpy as np


def solve_tridiag(a, b, c, f):
    n = len(f)
    beta = np.empty(n)
    alpha = np.empty(n - 1)
    x = np.empty(n)

    alpha[0] = b[0] / c[0]
    beta[0] = f[0] / c[0]
    for i in range(1, n - 1):
        din = c[i] - a[i - 1] * alpha[i - 1]
        alpha[i] = b[i] / din
        beta[i] = (f[i] + a[i - 1] * beta[i - 1]) / din
    beta[-1] = (f[-1] + a[-1] * beta[-2]) / (c[-1] - a[-1] * alpha[-1])

    x[-1] = beta[-1]
    for i in reversed(range(n - 1)):
        x[i] = alpha[i] * x[i + 1] + beta[i]

    return x
```

Листинг 3: matrix_solver.py

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def error(y, x, t, solution):
    x, t = np.meshgrid(x, t, indexing='ij')
    u = solution(x, t)

    return np.max(np.abs(y - u))


def plot(x, t, y):
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')

    x, t = np.meshgrid(x, t, indexing='ij')

    ax.plot_surface(x, t, y, cmap='viridis')

    ax.set_xlabel('x')
    ax.set_ylabel('t')
    ax.set_zlabel('u')
    ax.view_init(35, 65)

    plt.show()
```

4 Результаты

Графики

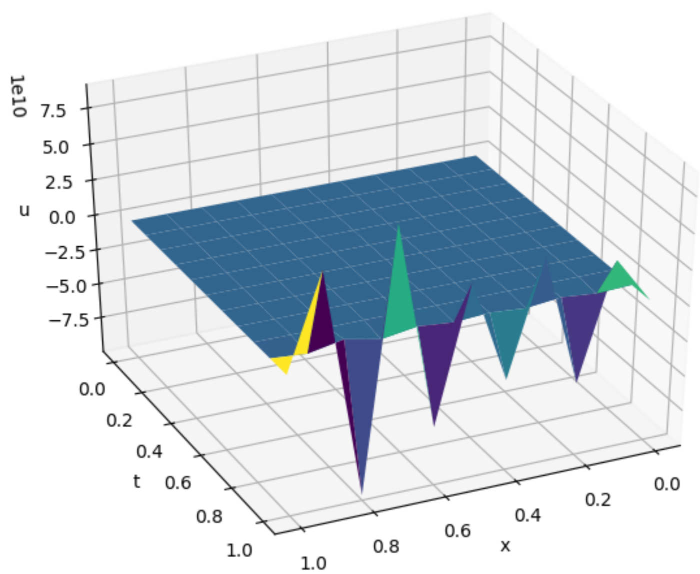

Рис. 1: Численное решение при $h = \tau = 0.1$

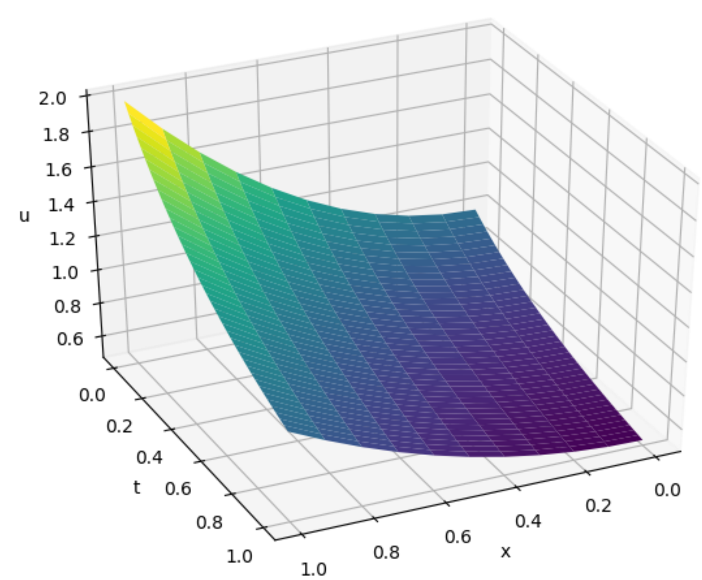

Рис. 2: Численное решение при $h = 0.1, \tau = \frac{h^2}{2}$

Погрешность

Явная разностная схема с $\tau = h = 0.1$:	97047817175.68475
Явная разностная схема с $h = 0.1, \tau = \frac{h^2}{2}$:	0.00049
Чисто неявной разностной схемы с $\tau = h = 0.1$:	0.18181
Разностной схемы Кранка-Николсон с $\tau = h = 0.1$:	0.00063