Московский Государственный Университет им. М. В. Ломоносова
Физический факультет

Практическое задание по ОММ

Задача #2

Выполнил студент 335 группы
Будакян Я. С.
Преподаватель Домбровская Ж. О.

1 Постановка задачи

Задача 2. Используя метод переменных направлений, решите краевую задачу:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u \quad 0 < x < 1, \quad 0 < y < 2, \quad t > 0 \\ u|_{x=0} = u|_{x=1} = 0, \\ u|_{y=0} = u|_{y=2} = 0, \\ u|_{t=0} = \sin 2\pi x \sin \pi y \end{cases} \tag{1}$$

2 Метод решения

2.1 Метод переменных направлений

Для численного решения задачи, введем двумерную пространственную и временную равномерные сетки:

$$\bar{\omega}_{h_1} = \{x_i = ih_1;\ i = \overline{0, N_1};\ h_1 N_1 = 1\}$$
$$\bar{\omega}_{h_2} = \{y_j = jh_2;\ j = \overline{0, N_2};\ h_2 N_2 = 2\}$$
$$\bar{\omega}_{\tau} = \{t_k = k\tau;\ k = \overline{0, S};\ \tau S = T\}$$
$$\bar{\omega}_{h_1 h_2 \tau} = \bar{\omega}_{h_1} \times \bar{\omega}_{h_2} \times \bar{\omega}_{\tau} = \{(x_i, y_j, t_k) \in \bar{D}\}$$
$$\bar{D} = \{0 < x < 1;\ 0 < y < 2;\ 0 \le t \le T\}$$

Введем сеточную функцию:

$$\omega^k_{h_1 h_2} \stackrel{\text{def}}{=} u(x_i, y_j, t_k)$$

Запишем разностную аппроксимацию оператора Лапласа:

$$L\omega \to \Lambda\omega = \Lambda_1\omega + \Lambda_2\omega,$$

где

$$\Lambda_1\omega = \frac{\omega_{i-1,j} - 2\omega_{ij} + \omega_{i+1,j}}{h_1^2}$$

$$\Lambda_2\omega = \frac{\omega_{i,j-1} - 2\omega_{ij} + \omega_{i,j+1}}{h_2^2}$$

При решении данной задачи используется схема переменных направлений, являющаяся экономичной разностной схемой. Такая схема сочетает в себе ряд преимуществ, таких как сложность $O\left(N_1 N_2\right)$ и безусловная устойчивость. В этой схеме переход с одного временного слоя на другой происходит в 2 шага, привлекая промежуточный(дробный) слой. Разностная аппроксимация примет вид:

$$\frac{\omega^{k+\frac{1}{2}} - \omega^k}{0.5\tau} = \Lambda_1\omega^{k+\frac{1}{2}} + \Lambda_2\omega^k$$

$$\frac{\omega^{k+1} - \omega^{k+\frac{1}{2}}}{0.5\tau} = \Lambda_1\omega^{k+\frac{1}{2}} + \Lambda_2\omega^{k+1}$$

Рассмотрим переход $k \to k+\frac{1}{2}$. Используя явный вид операторов Λ_1 и Λ_2, имеем:

$$\begin{cases} \frac{\gamma_1}{2}\omega_{i-1,j}^{k+\frac{1}{2}} - (1+\gamma_1)\omega_{ij}^{k+\frac{1}{2}} + \frac{\gamma_1}{2}\omega_{i+1,j}^{k+\frac{1}{2}} = -\frac{\gamma_2}{2}\omega_{i,j-1}^{k} - (1-\gamma_2)\omega_{ij}^{k} - \frac{\gamma_2}{2}\omega_{i,j+1}^{k} \\ \omega_{0j}^{k+\frac{1}{2}} = \omega_{N_1,j}^{k+\frac{1}{2}} = 0 \quad j = \overline{1, N_2-1}, \end{cases} \tag{2}$$

где $\gamma_1 = \frac{\tau}{h_1^2}$, $\gamma_2 = \frac{\tau}{h_2^2}$. Введя обозначения:

$$A^{(1)} = B^{(1)} = \frac{\gamma_1}{2}$$

$$C^{(1)} = 1 + \gamma_1$$

$$F_{ij}^k = \frac{\gamma_2}{2}\omega_{i,j-1}^k + (1-\gamma_2)\omega_{ij}^k + \frac{\gamma_2}{2}\omega_{i,j+1}^k,$$

получим систему:

$$\begin{cases} A^{(1)}\omega_{i-1,j}^{k+\frac{1}{2}} - C^{(1)}\omega_{ij}^{k+\frac{1}{2}} + B^{(1)}\omega_{i+1,j}^{k+\frac{1}{2}} = -F_{ij}^k \\ \omega_{0j}^{k+\frac{1}{2}} = \omega_{N_1,j}^{k+\frac{1}{2}} = 0 \quad j = \overline{1, N_2-1} \end{cases} \tag{3}$$

Аналогично, вводя обозначения

$$A^{(2)} = B^{(2)} = \frac{\gamma_2}{2}$$

$$C^{(2)} = 1 + \gamma_2$$

$$F_{ij}^{k+\frac{1}{2}} = \frac{\gamma_1}{2}\omega_{i-1,j}^{k+\frac{1}{2}} + (1-\gamma_2)\omega_{ij}^{k+\frac{1}{2}} + \frac{\gamma_2}{2}\omega_{i+1,j}^{k+\frac{1}{2}},$$

получим систему для перехода $k + \frac{1}{2} \to k + 1:$

$$\begin{cases} A^{(2)}\omega_{i,j-1}^{k+1} - C^{(2)}\omega_{ij}^{k+1} + B^{(2)}\omega_{i,j+1}^{k+1} = -F_{ij}^{k+\frac{1}{2}} \\ \omega_{i0}^{k+1} = \omega_{i,N_2}^{k+1} = 0 \quad i = \overline{1, N_1 - 1} \end{cases} \tag{4}$$

Полученные системы (3), (4) решаются методом прогонки.

2.2 Метод прогонки

Метод прогонки используется для решения систем линейный уравнений вида $Ax = F$, где F - трехдиагональная матрица. Пусть дана система уравнений

$$\begin{cases} A_n y_{n-1} - C_n y_n + B_n y_{n+1} = -F_n, \quad n = \overline{1, N-1} \\ A_n \neq 0, \quad B_n \neq 0, \quad n = \overline{1, N-1} \\ y_0 = k_1 y_1 + \mu_1, \quad y_N = k_2 y_{N-1} + \mu_2 \end{cases} \tag{5}$$

3 Порядок аппроксимации

4 Устойчивость схемы

5 Результаты

Численный расчет с помощью описанного алгоритма дает следующий результаты:

6 Листинг программы

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 17 14:43:10 2016

@author: Yan
"""

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

#функция граничных условий
def u_t0(x,y):
    return sin(2*np.pi*x) * sin(np.pi*y)

#TMA - tridiagonal matrix algorithm - метод прогонки
def TMA(A, B, C, F, k1, k2, mu1, mu2):
    N = A.size
    w = np.zeros(N+1)
    alpha = np.zeros(N+1)
    beta = np.zeros(N+1)

    #прямой ход прогонки
    alpha[1] = k1
    beta[1] = mu1

    for n in range(1, N):
        alpha[n+1] = B[n] / (C[n] - alpha[n]*A[n])
        beta[n+1] = (A[n]*beta[n] + F[n]) / (C[n] - alpha[n]*A[n])

    #обратный ход
    w[N] = (mu2 + beta[N]*k2) / (1 - alpha[N]*k2)

    for n in range(N-1, -1, -1):
        w[n] = alpha[n+1]*w[n+1] + beta[n+1]

    return w


#шаги по времени и координатам
N1 = 20
N2 = 40
S = 100

#рассматриваемый промежуток времени
T = 0.07

h1 = 1 / N1
h2 = 2 / N2
tau = T / S

#коэффициенты гамма_1 и гамма_2
g1 = tau / h1**2
g2 = tau / h2**2

#массив для искомого решения
#индексы: время(k) - x(i) - y(j)
u = np.zeros((S+1, N1+1, N2+1))
#массив для хранения значений на промежуточном слое k+1/2
u12 = np.zeros((S+1, N1+1, N2+1))

#заполняем граничные условия
for i in range(0, N1):
    for j in range(0, N2):
        u[0][i][j] = u_t0(i*h1, j*h2)

#основной цикл расчета
for k in range(S):
    #шаг k -> k + 1/2
    A = np.zeros(N1)
    B = np.zeros(N1)
    C = np.zeros(N1)
    F = np.zeros(N1)
```

```python

    k1 = 0
    k2 = 0
    mu1 = 0
    mu2 = 0

    for j in range(1, N2):
        for i in range(1, N1):
            A[i] = g1 / 2
            B[i] = g1 / 2
            C[i] = 1 + g1
            F[i] = (g2 / 2) * (u[k][i][j-1] + u[k][i][j+1]) + (1 - g2) * u[k][i][j]

        w = TMA(A, B, C, F, k1, k2, mu1, mu2)

        for i in range(N1+1):
            u12[k][i][j] = w[i]

    #шаг k + 1/2 -> k + 1
    A = np.zeros(N2)
    B = np.zeros(N2)
    C = np.zeros(N2)
    F = np.zeros(N2)

    #значения k1, k2, mu1, mu2 не поменялись
    for i in range(1, N1):
        for j in range(1, N2):
            A[j] = g2 / 2
            B[j] = g2 / 2
            C[j] = 1 + g2
            F[j] = (g1 / 2) * (u12[k][i-1][j] + u12[k][i+1][j]) + (1 - g1) * u12[k][i][j]

        w = TMA(A, B, C, F, k1, k2, mu1, mu2)

        for j in range(N2+1):
            u[k+1][i][j] = w[j]

#визуализация
ax = Axes3D(plt.figure())
X = np.arange(0, 1 + h1, h1)
Y = np.arange(0, 2 + h2, h2)
X, Y = np.meshgrid(X, Y)
Z = np.transpose(np.array(u[S]))

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, color='1')
ax.set_zlim3d(-1, 1)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u(x, y, {time})'.format(time=str(T)))
ax.view_init(elev=30, azim=230)

plt.show()
```

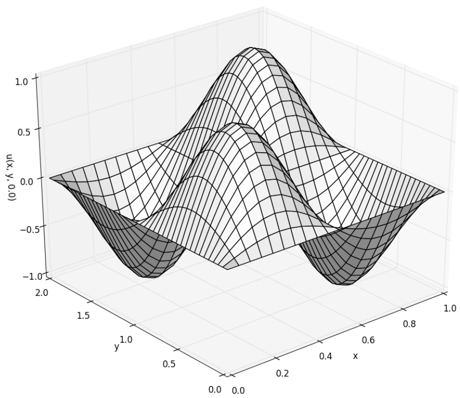

Рис. 1: T = 0

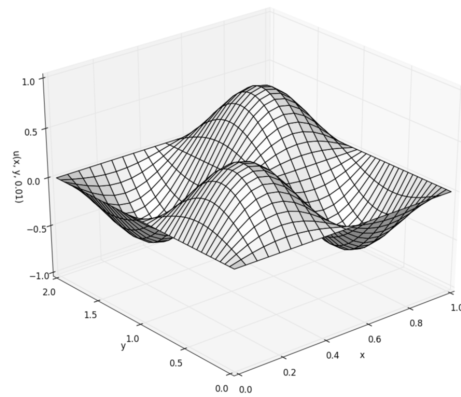

Рис. 2: T = 0.01

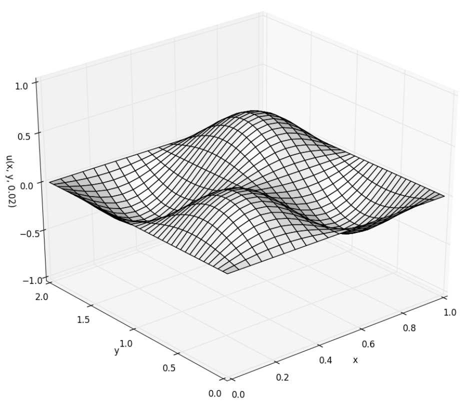

Рис. 3: T = 0.02

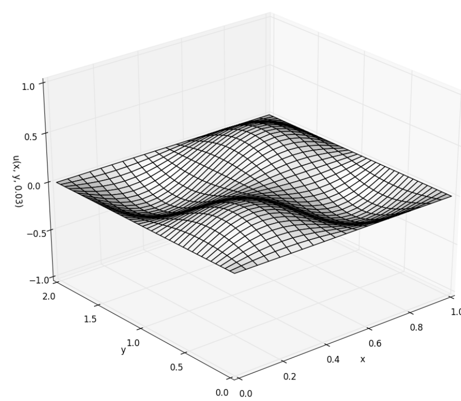

Рис. 4: T = 0.03

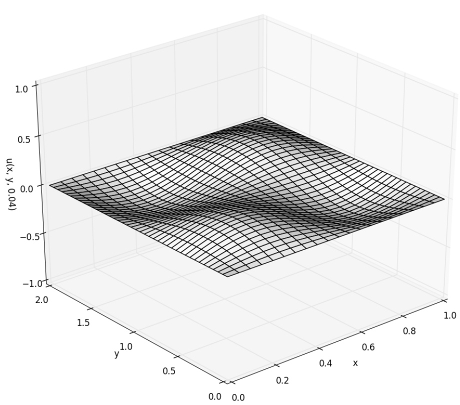

Рис. 5: T = 0.04

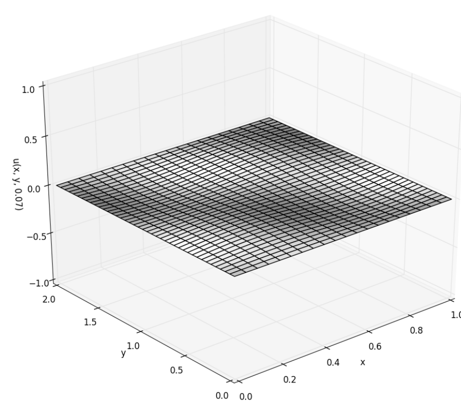

Рис. 6: T = 0.07