Практическое задание №2

Задача №44

Студент 305 группы Менделевич Лев Владиславович

Преподаватель Митина Ирина Владимировна

Физический факультет

Московский государственный университет имени М. В. Ломоносова

1.Постановка задачи.

Используя метод переменных направлений, решите краевую задачу:

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta u, x \in \left(0, \frac{\pi}{3}\right), y \in \left(0, \frac{\pi}{2}\right), t \in (0, T), \\ \left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=\frac{\pi}{3}} = 0, \\ u|_{y=0} = u|_{y=\frac{\pi}{2}} = 0, \\ u|_{t=0} = cos(3x)sin(4y) \end{cases}$$

2.Аналитическое решение.

Будем искать решение в виде:

$$u(x, y, t) = \sum_m^\infty \sum_n^\infty T_{nm}(t) V_{nm}(x, y).$$

Рассмотрим вспомогательную задачу Штурма-Лиувилля:

$$\begin{cases} \Delta V + \lambda V = 0, 0 < x < \frac{\pi}{3}, 0 < y < \frac{\pi}{2} \\ \left.\frac{\partial V}{\partial x}\right|_{x=0} = \left.\frac{\partial V}{\partial x}\right|_{x=\frac{\pi}{3}} = 0; \\ V|_{y=0} = V|_{y=\frac{\pi}{2}} = 0 \end{cases}$$

Методом разделения переменных задача разбивается на две задачи на отрезке:

$$\begin{cases} X'' + \mu X = 0, 0 < x < \frac{\pi}{3} \\ \left.\frac{\partial X}{\partial x}\right|_{x=0} = \left.\frac{\partial X}{\partial x}\right|_{x=\frac{\pi}{3}} = 0 \end{cases} \qquad \begin{cases} Y'' + \vartheta Y = 0, 0 < y < \frac{\pi}{2} \\ Y|_{y=0} = Y|_{y=\frac{\pi}{2}} = 0 \end{cases}$$

Тогда решения имеют вид:

$$\begin{cases} X_n = cos(3nx); \mu_n = (3n)^2, n = 1,2, \ldots \\ Y_m = sin(2my); \vartheta_m = (2m)^2, m = 0,1,2 \ldots \end{cases}$$

Получаем:

$$V_{nm} = cos(3nx)sin(2my), \lambda_{nm} = \mu_n + \vartheta_m.$$

Для задачи коши:

$$\begin{cases} \frac{dT_{12}}{dt} + \lambda_{12} T_{12} = 0 \\ T_{12}(0) = 1 \end{cases}$$

(остальные T_{nm} равны 0, из-за нулевых граничных условий)

В итоге получим решение:

$$u(x, y, t) = e^{-\lambda_{12} t} cos(3x)sin(4y)$$

3.Построение разностной схемы.

Введем сетку:

$$\omega = G \oplus [0,T], G = \{(x,y) : 0 \leqslant x \leqslant \frac{\pi}{3}, 0 \leqslant y \leqslant \frac{\pi}{2}\}$$

$$\omega_h = \{(x_i, y_j) : x_i = ih_x, n = 0,1,...,N, Nh_x = \frac{\pi}{3}\, y_j = jh_y, j = 0,1,...,M, Mh_y = \frac{\pi}{2}\},$$

$$\omega_\tau = \{t_k = \tau k, k = 0,1,...,K, K\tau = T\}, \omega_{h\tau} = \omega_h \oplus \omega_\tau.$$

На введенной разностной сетке будут рассматриваться сеточные функции. Дифференциальные операторы заменяются на их разностные аналоги:

$$\Delta u \to \Lambda w^j = \Lambda_x w^k + \Lambda_y w^k,$$

$$\Lambda_x w^k = \frac{w_{i-1,j}^k - 2w_{i,j}^k + w_{i+1,j}^k}{h_x^2},$$

$$\Lambda_y w^k = \frac{w_{i,j-1}^k - 2w_{i,j}^k + w_{i,j+1}^k}{h_y^2}$$

Переход со слоя на слой осуществляется в два шага с помощью промежуточного слоя. Переход осуществляется в два этапа:

1) Решается первое уравнение – явное по направлению x и неявное по y
2) Решается второе уравнение - явное по направлению y и неявное по x.

$$\frac{w^{k+1/2} - w^k}{0.5\tau} = \Lambda_x w^{k+1/2} + \Lambda_y w^k$$

$$\frac{w^{k+1} - w^{k+1/2}}{0.5\tau} = \Lambda_x w^{k+1/2} + \Lambda_y w^{k+1}$$

Порядок аппроксимации: $O(h_x^2 + h_y^2 + \tau^2)$.

Подставляя операторы и учитывая краевые условия, получаем такую краевую задачу:

$$\begin{cases} \dfrac{0.5\tau}{h_1^2}\omega_{i-1,j}^{k+1/2} - \left(1 + \dfrac{\tau}{h_1^2}\right)\omega_{i,j}^{k+1/2} + \dfrac{0.5\tau}{h_1^2}\omega_{i+1,j}^{k+1/2} = -F_{i,j}^{k+1/2} \\ \omega_{0,j}^{k+1/2} = \omega_{1,j}^{k+1/2}, \omega_{N_x,j}^{k+\frac{1}{2}} = \omega_{N_x-1,j}^{k+\frac{1}{2}} \\ F_{i,j}^{k+1/2} = \dfrac{0.5\tau}{h_2^2}\left(\omega_{i,j-1}^k + \omega_{i,j+1}^k\right) + \left(1 - \dfrac{\tau}{h_2^2}\right)\omega_{i,j}^k \\ \dfrac{0.5\tau}{h_2^2}\omega_{i,j-1}^{k+1} - \left(1 + \dfrac{\tau}{h_2^2}\right)\omega_{i,j}^{k+1} + \dfrac{0.5\tau}{h_2^2}\omega_{i,j+1}^{k+1} = -F_{i,j}^{k+1} \\ \omega_{i,0}^{k+1} = 0, \omega_{i,N_y}^{k+1} = 0 \\ F_{i,j}^{k+1} = \dfrac{0.5\tau}{h_1^2}\left(\omega_{i-1,j}^{k+1/2} + \omega_{i+1,j}^{k+1/2}\right) + \left(1 - \dfrac{\tau}{h_1^2}\right)\omega_{i,j}^{k+1/2} \end{cases}$$

4.Метод подгонки.

Все искомые значения в узлах связаны между собой рекуррентным соотношением:

$$Ay_{n+1} - Cy_n + By_{n-1} = -F_n,\ 0 < n < N$$

$$y_0 = \kappa_1 y_1 + \mu_1, \quad y_N = \kappa_2 y_{N-1} + \mu_2$$

Решение ищется в виде:

$$y_n = \alpha_n y_{n+1} + \beta_n, \quad n = N-1, N-2, \ldots, 0, \quad (1)$$

Коэффициенты определяются из рекуррентных соотношений:

$$\alpha_{n+1} = \frac{B_n}{C_n - A_n\alpha_n}, \quad \beta_{n+1} = \frac{A_n\beta_n + F_n}{C_n - A_n\alpha_n}, n = 0, 1, \ldots, N-1$$

Из краевых условий получаем:

$$\alpha_0 = \kappa_1, \quad \beta_0 = \mu_1$$

$$y_M = \frac{\mu_2 + \kappa_2\beta_M}{1 - \kappa_2\alpha_M}.$$

Зная y_M можно найти все остальные y_n из формулы 1.

5.Устойчивость.

Устойчивость схемы по начальным данным можно определить с помощью спектрального метода Неймана.

Будем искать решение в виде:

$$w^j_{n,m} = \lambda^j_x e^{i(\alpha n + \beta m)}.$$

Подстановка в разностное уравнение дает решение:

$$\sqrt{\lambda^j_x} = \frac{1 - \frac{2\tau}{h_y^2}\sin\frac{\beta}{2}^2}{1 + \frac{2\tau}{h_x^2}\sin\frac{\alpha}{2}^2}$$

Получаем что

$$\lambda_x < 1\ \forall\tau, h_x, h_y, \alpha, \beta.$$

Аналогично для y:

$$\sqrt{\lambda^j_y} = \frac{1 - \frac{2\tau}{h_x^2}\sin\frac{\alpha}{2}^2}{1 + \frac{2\tau}{h_y^2}\sin\frac{\beta}{2}^2}$$

$$\lambda_y < 1\ \forall\tau, h_x, h_y, \alpha, \beta.$$

Таким образом выполняется критерий Неймана.

Можно заметить, что:

$$|\lambda_x\lambda_y| < 1\ \forall\tau, h_x, h_y, \alpha, \beta$$

Тогда критерий Неймана будет выполнятся при переходе с j на j+1 слой.

6.Программа.

Компьютерная программа написана на языке С

Библиотеки:

```
#include <stdio.h>
#include <malloc.h>
#include <math.h>
#include <string.h>
#include <process.h>
#include <time.h>
```

Глобальные переменные:

```
double hx,hy,tau,*A,*B,*C,*F;
int i, j, NX = 50, NY = 50, NT = 50, NN = 1000;
const double PI = 3.141592653589793;
```

Реализация метода прогонки для нижнего слоя:

```
double** progonX(int k,double **U0)
 {
 double g1 = tau / (hx*hx);
 double g2 = tau / (hy*hy);
 double *d, *sigma,*U,**U1;
 U1 = (double**)malloc(NX * sizeof(double*));
 for (i = 0; i < NX; i++)  // цикл по строкам
        U1[i] = (double*)malloc(NY * sizeof(double));
 A = (double*)malloc(NN * sizeof(double));
 B = (double*)malloc(NN * sizeof(double));
 C = (double*)malloc(NN * sizeof(double));
 U = (double*)malloc(NN * sizeof(double));
 F = (double*)malloc(NN * sizeof(double));
 d = (double*)malloc(NN * sizeof(double));
 sigma = (double*)malloc(NN * sizeof(double));
 for (int i = 0; i < NX; i++)
 {
        U1[i][0] = U0[i][0];
        U1[i][NY-1] = U0[i][NY-1];
 }
        for (int j = 1; j < NY - 1; j++)
              {
               for (int i = 1; i < NX - 1; i++)
                      {
                             A[i] = 0.5 * g1;
                             C[i] = 1 + g1;
                             B[i] = 0.5 * g1;
              F[i] = 0.5*g2*U0[i][j - 1] + (1 - g2) * U0[i][j] +0.5*g2*U0[i][j + 1];
                      }
                      sigma[0] = 0;
                      d[0] = 1;
                       for (int i = 1; i < NX - 1; i++)
                                    {
                                           d[i] = B[i] / (C[i] - d[i-1] * A[i]);
                      sigma[i] = (A[i] * sigma[i-1] + F[i]) / (C[i] - d[i-1]* A[i]);
                                    }
                       U[NX-1] = sigma[NX-2] / (1 - d[NX-2]);
                       for (int i = NX - 2; i >= 0; i--)
```

```
                        {
                        U[i] = d[i]* U [i + 1] + sigma[i];
                        }
                for (int i = 0; i < NX; i++)
                        U1[i][j] = U[i];
                }
        return (U1);
        free(U1), free(U), free(A); free(B); free(C); free(F); free(d); free(sigma);
}
```

Реализация метода прогонки для верхнего слоя:

```
double** progonY(int k, double **U0)
{
        double g1 = tau / (hx*hx);
        double g2 = tau / (hy*hy);
        double *d, *sigma, *U, **U1;
        U1 = (double**)malloc(NX * sizeof(double*));
        for (i = 0; i < NX; i++)  // цикл по строкам
                U1[i] = (double*)malloc(NY * sizeof(double));
        A = (double*)malloc(NN * sizeof(double));
        B = (double*)malloc(NN * sizeof(double));
        C = (double*)malloc(NN * sizeof(double));
        U = (double*)malloc(NN * sizeof(double));
        F = (double*)malloc(NN * sizeof(double));
        d = (double*)malloc(NN * sizeof(double));
        sigma = (double*)malloc(NN * sizeof(double));
        for (int j = 0; j < NY; j++)
        {
                U1[0][j] = U0[0][j];
                U1[NX-1][j] = U0[NX-1][j];
        }
        for (int i = 1; i < NX - 1; i++)
        {
                for (int j = 1; j < NY - 1; j++)
                {
                        A[j] = 0.5 * g2;
                        C[j] = 1 + g2;
                        B[j] = 0.5 * g2;
                        F[j] = 0.5*g1*U0[i-1][j] + (1 - g1) * U0[i][j] + 0.5*g1*U0[i+1][j];
                }
                sigma[0] = 0;
                d[0] = 0;
                for (int j = 1 ; j < NY - 1; j++)
                {
                        d[j] = B[j] / (C[j] - d[j-1] * A[j]);
                        sigma[j] = (A[j] * sigma[j-1] + F[j]) / (C[j] - d[j-1] * A[j]);
                }
                U[NY-1] = 0;
                for (int j = NY - 2; j >= 0; j--)
                {
                        U[j] = d[j] * U[j + 1] + sigma[j];
                }
                for (int j = 0; j < NY; j++)
                        U1[i][j] = U[j];
        }
        return (U1);
        free(U1), free(U), free(A); free(B); free(C); free(F); free(d); free(sigma);
}
```

Заполнение сетки, а также граничные условия:

```
int main(void)
{

	FILE *fp1;
	fp1 = fopen("result.out", "w");
	double **a, *x, *t,*y;
	double X = PI/3, Y = PI/2, T = 0.2 ;
	a = (double**)malloc(NX * sizeof(double*));
	for (i = 0; i < NX; i++)  // цикл по строкам
		a[i] = (double*)malloc(NY * sizeof(double));
	x = (double*)malloc(NN * sizeof(double));
	y = (double*)malloc(NN * sizeof(double));
	t = (double*)malloc(NN * sizeof(double));
	//заполняем сетку
	hx = X / (NX-1);
	hy = Y / (NY-1);
	tau = T / (NT-1);
	for (int j = 0; j < NT; j++)
	{
		t[j] = tau* j ;
	}

	for (int k = 0; k < NX; k++)
	{
		x[k] = hx* k;
	}

	for (int k = 0; k < NT; k++)
	{
		y[k] = hy*k;
	}
	// гр.условия


	for (int i = 0; i < NX; i++)
	{
		for (int j = 0; j < NY; j++)
			a[i][j] = cos(3 * x[i]) * sin(4 * y[j]);
	}
```

Нахождение a[i][j] в каждый момент времени, вывод в файл значений массива, в скобках можно указать номер временной точки, для которого нужно получить массив.

```
for (int t = 1; t < NT; t++)
	 {
	   a = progonX(t, a);

	   a = progonY(t, a);
	   if (t == 42)
	   {
		for (int i = 0; i < NX; i++)
		{
			for (int j = 0; j < NY; j++)
```

```
                        {
                                fprintf(fp1, "%lf %lf %lf\n", x[i], y[j], a[i][j]);
                        }
                }
        }
    }
  fclose(fp1);
}
```

7.Графики.

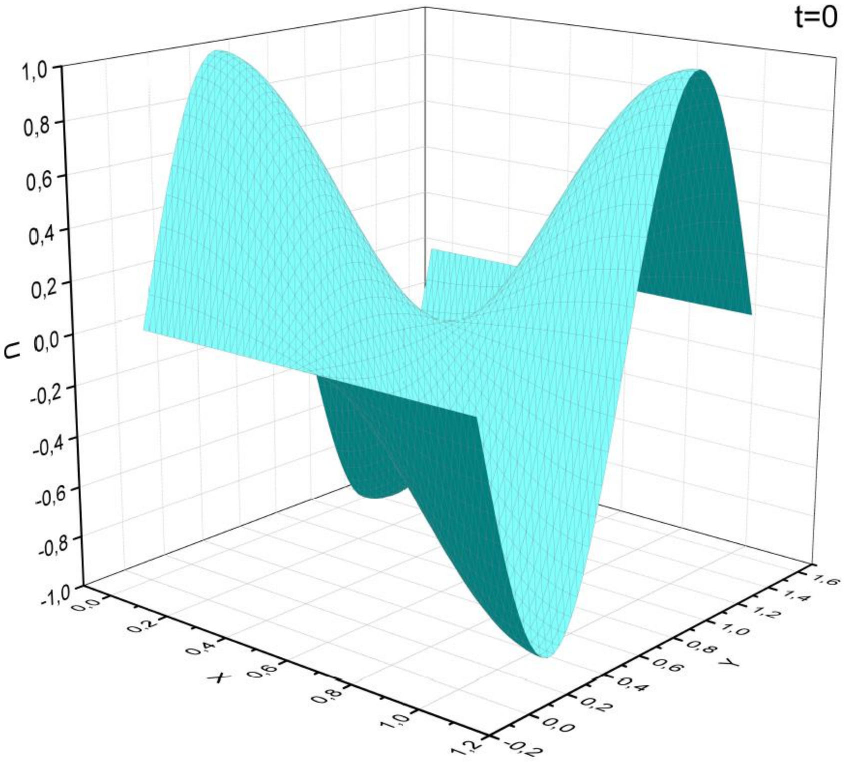

Рис.1 График зависимости функции U от координат x и y в момент времени t = 0

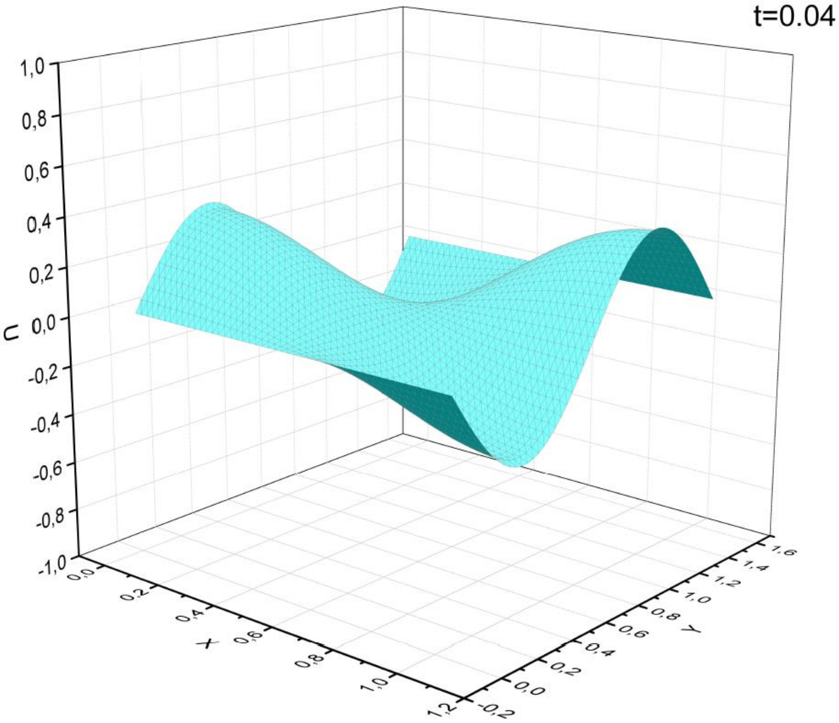

Рис2. График зависимости функции U от координат x и y в момент t=0.04

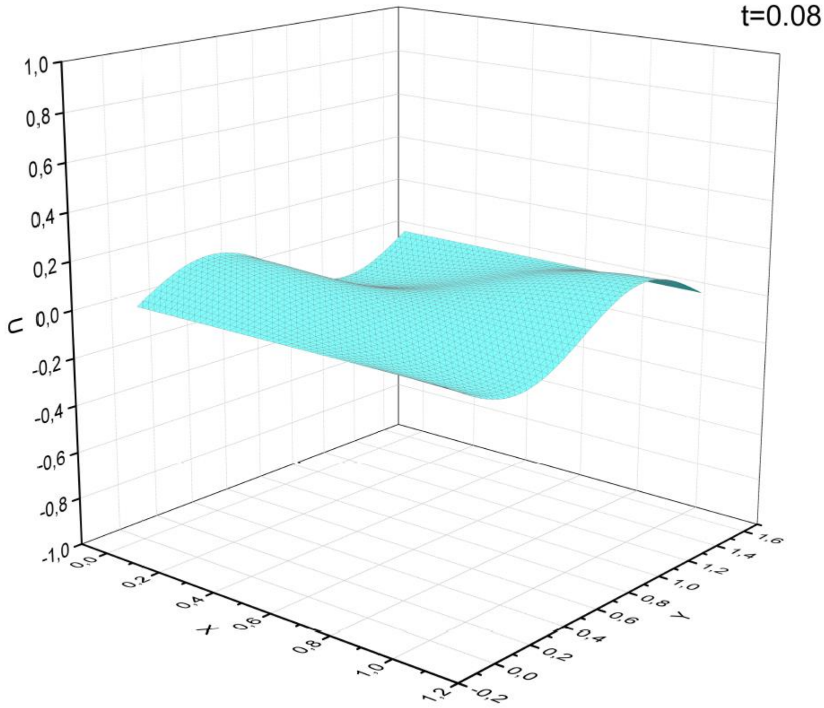

Рис.3 График зависимости функции U от координат x и y в момент t=0.08