ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА»

ФИЗИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА БИОФИЗИКИ

ОТЧЕТ ПО ДИСЦИПЛИНЕ

«ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ»

Выполнил студент
303 группы
Астанкович К.А.

Преподаватель:
Белов А.А.

Москва

2024

Содержание

1. Постановка задачи

Используя метод переменных направлений, решите краевую задачу:

$$\begin{cases} \dfrac{\partial u}{\partial t} = \Delta u + \sin x \cdot \sin t, \quad 0 < x < \pi, \quad 0 < y < 3, \quad t > 0 \\ u\Big|_{x=0} = u\Big|_{x=\pi]} = 0, \\ u|_{y=0} = u|_{y=3} = 0, \\ u|_{t=0} = 0 \end{cases} \tag{1}$$

2. Аналитическое решение

Решение будем искать в виде:

$$u(x,y,t) = \sum_{m}^{\infty} \sum_{n}^{\infty} T_{nm}(t) \cdot V_{nm}(x,y) \tag{2.1}$$

Тогда в результате разделения переменных получим явный вид функций $T_{nm}(t)$ и $V_{nm}(x,y)$, который определяется из решений соответствующей задачи Штурма-Лиувилля и задачи Коши:

$$\begin{cases} \Delta V + \lambda V = 0, \ 0 < x < \pi, \ 0 < y < 3, \\ V|_{x=0} = V|_{x=\pi} = 0, \\ V|_{y=0} = V|_{y=3} = 0 \end{cases} \tag{2.2}$$

$$\begin{cases} \frac{dT}{dt} + \lambda T = f(t), \\ T(0) = 0 \end{cases} \tag{2.3}$$

Повторно разделяя переменные в задаче Штурма Лиувилля, получаем две задачи

Штурма-Лиувилля:

$$\begin{cases} X'' + \mu X = 0,\ 0 < x < \pi, \\ X\big|_{x=0} = X\big|_{x=\pi} = 0 \end{cases} \tag{2.4}$$

$$\begin{cases} Y'' + \nu Y = 0,\ 0 < y < 3, \\ Y\big|_{y=0} = Y\big|_{y=3} = 0 \end{cases} \tag{2.5}$$

Их решения имеют вид:

$$X_n = \sin{(nx)},\ \mu_n = n^2,\ n = 1, 2, ... \tag{2.6}$$

$$Y_m = \sin{\left(\frac{\pi m y}{3}\right)},\ \nu_m = \left(\frac{\pi m}{3}\right)^2,\ m = 1, 2, ... \tag{2.7}$$

Тогда $V_{nm} = \sin{(nx)}\sin{\left(\frac{\pi m y}{3}\right)},\ \lambda_{nm} = \mu_n + \nu_m = n^2 + \left(\frac{\pi m}{3}\right)^2,\ n, m = 1, 2, ...$ Теперь рассмотрим решение задачи Коши. Для начала получим явный вид функции. Его можно получить, вычислив интеграл:

$$f_{nm} = \frac{1}{||V_{nm}||}\iint\limits_D F(x, y, t)V_{nm}(x, y)dxdy \tag{2.8}$$

В нашем случае:

$$||V_{nm}|| = (V_{nm}, V_{nm}) = \int\limits_0^3 \sin{\left(\frac{\pi m y}{3}\right)^2}dy\int\limits_0^\pi \sin{(nx)^2}dx = \frac{3}{2}\cdot\frac{\pi}{2} = \frac{3\pi}{4},$$

$$\iint\limits_D F(x, y, t)V_{nm}(x, y)dxdy = \int\limits_0^\pi \sin{(x)}\sin{(nx)}dx\int\limits_0^3 \sin{(t)}\sin{\left(\frac{\pi m y}{3}\right)}dy =$$

$$= \begin{cases} 0,\ n \neq 1, \\ \frac{\pi}{2}\cdot\frac{3\cdot(1-(-1)^m)}{\pi m}\sin t \end{cases}$$

Подставим получившиеся выражения в формулу (2.8) и получим:

$$f_{nm} = \begin{cases} 0,\ n \neq 1,\ m = 1, 2, ... \\ \frac{2\cdot(1-(-1)^m)}{\pi m}\sin{(t)},\ n = 1,\ m = 1, 2, ... \end{cases} \tag{2.9}$$

Значит, можно рассмотреть два случая задачи Коши и решить их по отдельности:

При $n \neq 1$:

$$\begin{cases} \frac{dT_{nm}(t)}{dt} + \lambda_{nm}T_{nm}(t) = 0,\ t > 0, \\ T_{nm}(0) = 0 \end{cases} \tag{2.10}$$

В этом случае:

$$T_{nm}(t) = 0 \tag{2.11}$$

При n = 1 введём обозначения: $T_{1m} \equiv T_m(t)$ и $\lambda_{1m} \equiv \lambda_m$. Получаем неоднородное дифференциальное уравнение:

$$\begin{cases} \frac{dT_m(t)}{dt} + \lambda_m T_m(t) = \frac{2\cdot(1-(-1)^m)}{\pi m}\sin{(t)},\ t > 0, \\ T_m(0) = 0 \end{cases} \tag{2.12}$$

Его решением является следующее выражение:

$$T_m(t) = \int\limits_0^t e^{-\lambda_m(t-\tau)} \frac{2\cdot(1-(-1)^m)}{\pi m} \sin(\tau) d\tau =$$
$$= \frac{2\cdot(1-(-1)^m)}{\pi m} \frac{e^{-\lambda_m t} + \lambda_m \cdot \sin(t) - \cos(t)}{\lambda_m^2 + 1}$$

Учтем, что:

$$\frac{2\cdot(1-(-1)^m)}{\pi m} = \begin{cases} 0,\ m = 2k,\ k = 1, 2, ..., \\ \frac{4}{\pi m},\ m = 2k+1, k = 0, 1, 2, ... \end{cases} \tag{2.13}$$

получим итоговый ответ:

$$u(x, y, t) = \sum_{k=0}^{\infty} \frac{4}{\pi(2k+1)} \sin(x) \sin\left(\frac{\pi y(2k+1)}{3}\right) \left(\frac{e^{-\lambda_k t} + \lambda_k \cdot \sin(t) - \cos(t)}{\lambda_k^2 + 1}\right), \tag{2.14}$$

где $\lambda_k = 1 + \left(\frac{\pi(2k+1)}{3}\right)^2$

3. Метод решения

3.1 Разностная аппроксимация

Введём разностную сетку в области:

$$D = G \cap [0, T], \text{где } G = \{(x, y) : 0 \leqslant x \leqslant \pi, 0 \leqslant y \leqslant 3\}$$

$$x_{i_x} = i_x \cdot h_x,\ i_x = 0, 1, ..., N_x,\ h_x \cdot N_x = \pi$$
$$y_{i_y} = i_y \cdot h_y,\ i_y = 0, 1, ..., N_y,\ h_y \cdot N_y = 3,$$
$$t_j = j \cdot \tau,\ j = 0, 1, ..., M,\ \tau \cdot M = T = 20,$$

где N_x – число узлов по оси абсцисс, N_y – число узлов по оси ординат, M – число узлов по оси времени, h_x и h_y – шаг по соответсвующей координате, τ – шаг по времени. T возьмем заведомо большим. Разностная аппроксимация оператора Лапласа $\Lambda u = \Lambda_x u + \Lambda_y u$, где

$$\Lambda_x u = \frac{1}{h_x}\left(\frac{u_{i_x+1,i_y} - u_{i_x,i_y}}{h_x} - \frac{u_{i_x,i_y} - u_{i_x-1,i_y}}{h_x}\right) = \frac{u_{i_x+1,i_y} - 2u_{i_x,i_y} + u_{i_x-1,i_y}}{h_x^2} \tag{3.1.1}$$

$$\Lambda_y u = \frac{1}{h_y}\left(\frac{u_{i_x,i_y+1} - u_{i_x,i_y}}{h_y} - \frac{u_{i_x,i_y} - u_{i_x,i_y-1}}{h_y}\right) = \frac{u_{i_x,i_y+1} - 2u_{i_x,i_y} + u_{i_x,i_y-1}}{h_y^2} \tag{3.1.2}$$

В таком случае уравнение для сеточной функции берем в виде:

$$\frac{u^{j+1} - u^j}{\tau} = \Lambda(\sigma u^{j+1} + (1-\sigma)u^j) + f^{j+\frac{1}{2}}, \tag{3.1.3}$$

где $f^{j+\frac{1}{2}} = \sin x \sin t_{j+\frac{1}{2}}$, а σ – некоторое число в промежутке от нуля до единицы. Начальное условие:

$$u^0_{i_x,i_y} = 0,\ \forall i_x = 0, ..., N_x,\ \forall i_y = 0, ..., N_y \tag{3.1.4}$$

Граничные условия:

$$u^j_{0,i_y} = u^j_{N_x,i_y} = 0,\ \forall i_y = 0, ..., N_y,\ \forall j = 0, ..., M; \tag{3.1.5}$$

$$u^j_{i_x,0} = u^j_{i_x,N_y} = 0,\ \forall i_x = 0, ..., N_x,\ \forall j = 0, ..., M. \tag{3.1.6}$$

Явная ($\sigma = 0$) и неявная ($\sigma = 1$) схемы имеют одинаковый порядок точности, но различаются в числе операций: $Q_{\text{explicit}} = O\left(\frac{1}{h_x h_y}\right)$, $Q_{\text{implicit}} = O\left(\frac{1}{(h_x h_y)^2}\right)$. Помимо прочего, явная схема является условно устойчивой, а неявная – безусловно. Для оптимального решения задачи нам нужна схема, которая будет безусловна устойчива и при этом иметь минимальное число операций, по этим причинам мы будем использовать альтернативную схему, которая является безусловно устойчивой и имеет объём работ: $Q_{\text{ПН}} = O\left(\frac{1}{h_x h_y}\right)$. Ее название - схема переменных направлений.

3.2 Схема переменных направлений

Разностная аппроксимация нашего уравнения в схеме переменных направлений имеет вид:

$$\frac{u^{j+\frac{1}{2}} - u^j}{0.5\tau} = \Lambda_x u^{j+\frac{1}{2}} + \Lambda_y u^j + f^{j+\frac{1}{2}}, \tag{3.2.1}$$

$$\frac{u^{j+1} - u^{j+\frac{1}{2}}}{0.5\tau} = \Lambda_x u^{j+\frac{1}{2}} + \Lambda_y u^{j+1} + f^{j+\frac{1}{2}} \tag{3.2.2}$$

Переход от слоя j к слою j + 1 осуществляется в два этапа с шагами $0.5 \cdot \tau$: сначала решается уравнение (3.2.1), неявное по направлению x и явное по направлению y, а затем уравнение (3.2.2), явное по направлению x и неявное по направлению y. Значение $u^{j+\frac{1}{2}}$ является промежуточным и играет вспомогательную роль. Схема переменных направлений безусловно устойчива при любых шагах h_x, h_y и τ.

Используя явный вид разностных операторов Λ_x и Λ_y, для перехода со слоя j на промежуточный слой $j + \frac{1}{2}$ получаем следующую краевую задачу:

$$\begin{cases} \frac{1}{2}\gamma_x u^{j+\frac{1}{2}}_{i_x-1,i_y} - (1-\gamma_x) u^{j+\frac{1}{2}}_{i_x,i_y} + \frac{1}{2}\gamma_x u^{j+\frac{1}{2}}_{i_x+1,i_y} = -F^{j+\frac{1}{2}}_{i_x,i_y}, \\ u^{j+\frac{1}{2}}_{0,i_y} = u^{j+\frac{1}{2}}_{N_x,i_y} = 0, \end{cases} \tag{3.2.3}$$

где $\gamma_\alpha = \frac{\tau}{h_\alpha^2}$, $\alpha = x, y$ и

$$F^{j+\frac{1}{2}}_{i_x,i_y} = \frac{1}{2}\gamma_y \left(u^j_{i_x,i_y-1} - u^j_{i_x,i_y+1}\right) + (1-\gamma_y) u^j_{i_x,i_y} + \frac{1}{2}\tau f^{j+\frac{1}{2}}$$

Эта задача решается с помощью метода прогонки при каждом фиксированном $i_y = 1, ..., N_y - 1$. В результате получаем значения $u^{j+\frac{1}{2}}$ во всех узлах сетки G_x. Аналогично для перехода со слоя $j + \frac{1}{2}$ на слой j получаем:

$$\begin{cases} \frac{1}{2}\gamma_y u^{j+1}_{i_x,i_y-1} - (1-\gamma_y) u^{j+1}_{i_x,i_y} + \frac{1}{2}\gamma_y u^{j+1}_{i_x,i_y+1} = -F^{j+1}_{i_x,i_y}, \\ u^{j+1}_{i_x,0} = u^{j+1}_{i_x,N_y} = 0, \end{cases} \tag{3.2.4}$$

где

$$F^{j+1}_{i_x,i_y} = \frac{1}{2}\gamma_x \left(u^{j+\frac{1}{2}}_{i_x-1,i_y} - u^{j+\frac{1}{2}}_{i_x+1,i_y}\right) + (1-\gamma_x) u^{j+\frac{1}{2}}_{i_x,i_y} + \frac{1}{2}\tau f^{j+\frac{1}{2}}$$

Данная задача так же решается методом прогонки при каждом фиксированном $i_x = 1, ..., N_x - 1$. В результате получаем значение u^{j+1} на новом слое. При переходе от слоя j_k на слой $j_k + 1$ процедура повторяется аналогично.

3.3 Метод прогонки

Метод прогонки применяется к решению системы алгебраических уравнений:

$$\begin{cases} A_m u_{m-1} - B_m u_m + C_m u_{m+1} = F_m, \quad m = 1, ..., N-1, \\ u_0 = \alpha_1 u_1 + \beta_1, \; u_N = \alpha_2 u_{N-1} + \beta_2 \end{cases} \tag{3.3.1}$$

где либо $B_m > A_m + C_m$, $0 \leqslant \alpha_{1,2} \leqslant 1$, либо $B_m \geqslant A_m + C_m$, $0 \leqslant \alpha_{1,2} < 1$. В нашем случае эти требования, очевидно, выполняются: $1 + \gamma_\alpha > \frac{1}{2}\gamma_\alpha + \frac{1}{2}\gamma_\alpha = \gamma_\alpha$.

Для решения этой системы предположим, что значения искомой функции в двух любых соседних точках связаны линейным соотношением:

$$u_m = d_{m+1} u_{m+1} + \sigma_{m+1}, \tag{3.3.2}$$

где d_m и σ_m – прогоночные коэффициенты. Тогда, сдвинув индекс на единицу, получим:

$$u_{m-1} = d_m u_m + \sigma_m \tag{3.3.3}$$

Подставим (3.3.3) в уравнение (3.3.1), исключим таким образом u_{m-1}:

$$(A_m d_m - B_m) u_m = -C_m u_{m+1} + F_m - A_m \sigma_m \tag{3.3.4}$$

Используя (3.3.2), исключаем u_m:

$$u_{m+1}[(A_m d_m - B_m) d_{m+1} - C_m] = F_m - A_m d_m - \sigma_{m+1}(A_m d_m - B_m) \tag{3.3.5}$$

Для того, чтобы это соотношение было верно для любых u_{m+1}, нужно, чтобы выражение в квадратных скобках и правая часть были равны нулю. Приравнивая их к нулю, получаем рекуррентные формулы для определения прогоночных коэффициентов:

$$d_{m+1} = \frac{C_m}{B_m - A_m d_m}, \quad \sigma_{m+1} = \frac{F_m - A_m \sigma_m}{A_m d_m - B_m}. \tag{3.3.6}$$

Используя граничные условия, найдём: $d_1 = \alpha_1$, $\sigma_1 = \beta_1$. Далее совершаем прогонку в направлении возрастания индекса, последовательно определяя значения коэффициентов d_m и σ_m для $m = 1, ..., N$. На правом конце имеем два соотношения, связывающие u_{N-1} и u_N: $u_{N-1} = d_N u_N + \sigma_N$ и $u_N = \alpha_2 u_{N-1} + \beta_2$. Из этих уравнений находим:

$$n_N = \frac{\alpha_2 \sigma_N + \beta_2}{1 - \alpha_2 d_N} \tag{3.3.7}$$

При $d_1 = \alpha_1$ и всех условиях, наложенных при постановке задачи, получаем из рекуррентных формул, что $d_m < 1$ для $m = 2, ..., N$. Учитывая, что $\alpha_2 \leqslant 1$, получаем знаменатель в выражении для u_N положительным. Следовательно, и значение u_N определено.

Используя найденное u_N , делаем обратную прогонку в сторону уменьшающихся значений индекса, последовательно определяя из рекуррентных формул значения u_m. Число операций при поиске решения задачи (3.3.1) пропорционально числу узлов в слое.

4. Результаты

Программа была написана на языке программирования Python. Количество узлов для временной оси: $M = 10$, $\tau = \frac{T}{M} = \frac{10}{10} = 1$; количество узлов по оси x: $N_x = 20$,

$h_x = 1.58 \cdot 10^{-1}$; количество узлов по оси y: $N_y = 20$, $h_y = 1.5 \cdot 10^{-1}$. Результатом работы программы является набор графиков численного и аналитического решений u(x,y,t), а также ошибки численного решения в разные моменты времени.

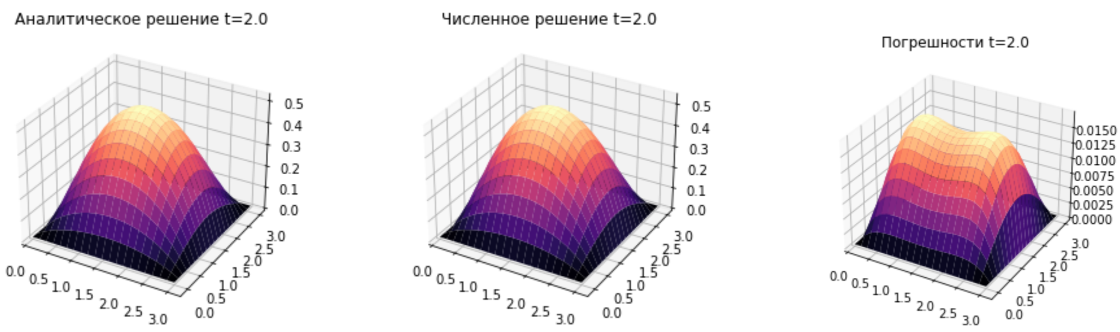

Рис. 1: Решения и погрешности для t= 2 сек

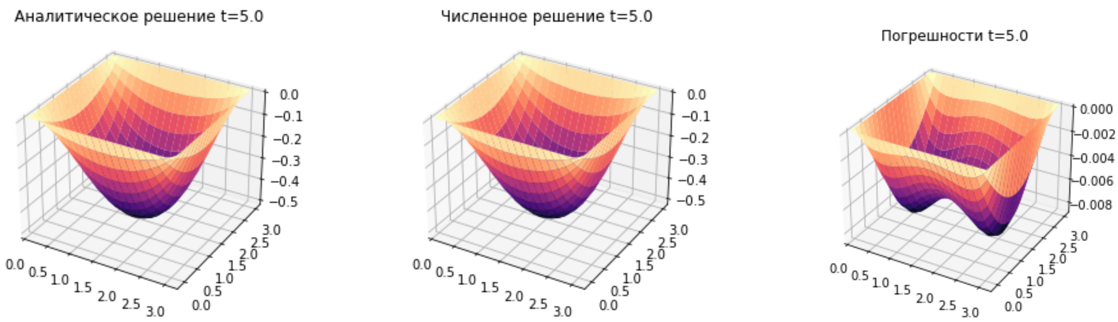

Рис. 2: Решения и погрешности для t= 5 сек

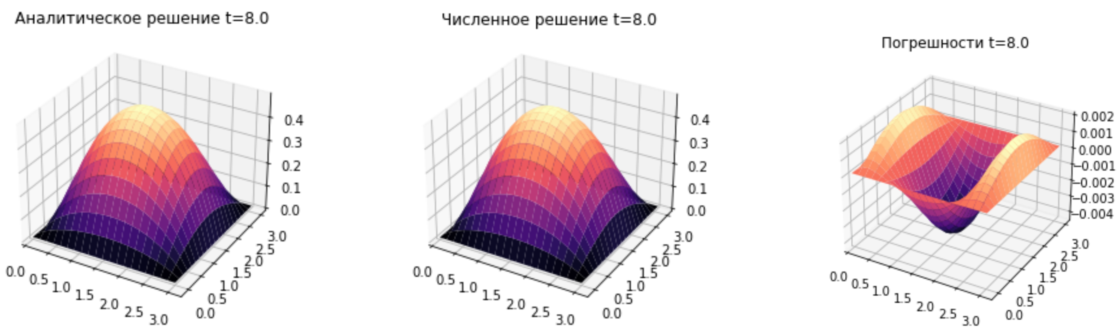

Рис. 3: Решения и погрешности для t= 8 сек

5. Приложение. Программная реализация

```
import matplotlib.pyplot as plt
import math
from math import pi as pi
from numpy import zeros,exp,sin, linspace, meshgrid,empty,cos,arctan,tan,
    max,min
from numpy.linalg import norm
from matplotlib.pyplot import plot, figure, axes, show, subplot
from sympy import symbols,diff,exp,atan,tan
#------------------------------------------------Input
x_l=0;x_r=pi
y_l=0;y_r=3
t_begin=0;t_end=10
J=10;N=20;M=20
Time_show=[2, 5, 8]
a_c=1
Type_boundary_condition_X=[0,0] #                                    == 0
Type_boundary_condition_Y=[0,0] #                                    == 1
def cz(i):
    f=(i*pi)**2/9+1
    return f
def norm_xy(i):
    f=2*(1-cos(i*pi))/i/pi
    return f
def fourier_t(t,i):
    f=1/(cz(i)**2+1)*(exp(-cz(i)*t)+cz(i)*sin(t)-cos(t))
    return f
def Function_analysis(t,x,y):
    l=50
    sum=0 # i=0
    for i in range(1,l,1):
        sum=(sum+fourier_t(t,i)*norm_xy(i)*
             sin(x)*sin(i*pi*y/3))
    return sum
def Function_begin(x,y):
    f=0
    return f
def Function_right(t,x,y):
    f=sin(x)*sin(t)
    return f
#------------------------------------------------Def
def Choic_type_boundary_condition_X_for_scheme(type_l,type_r):
    x=zeros(N+1)
    if type_l==0 and type_r==0:
        h=(x_r-x_l)/N
        for n in range(N+1):
            x[n]=x_l+n*h
    if type_l==1 and type_r==0:
        h=(x_r-x_l)/(N-1/2)
        for n in range(N+1):
            x[n]=x_l-h/2+n*h
    if type_l==0 and type_r==1:
        h=(x_r-x_l)/(N-1/2)
        for n in range(N+1):
```

```
            x[n]=x_l+n*h
    if type_l==1 and type_r==1:
        h=(x_r-x_l)/(N-1)
        for n in range(N+1):
            x[n]=x_l-h/2+n*h
    return x,h
def Choic_type_boundary_condition_Y_for_scheme(type_l,type_r):
    y=zeros(M+1)
    if type_l==0 and type_r==0:
        l=(y_r-y_l)/M
        for n in range(M+1):
            y[n]=y_l+n*l
    if type_l==1 and type_r==0:
        l=(y_r-y_l)/(M-1/2)
        for n in range(M+1):
            y[n]=y_l-l/2+n*l
    if type_l==0 and type_r==1:
        l=(y_r-y_l)/(M-1/2)
        for n in range(M+1):
            y[n]=y_l+n*l
    if type_l==1 and type_r==1:
        l=(y_r-y_l)/(M-1)
        for n in range(M+1):
            y[n]=y_l-l/2+n*l
    return y,l
def Layer_Time():
    t = zeros(J + 1)
    tau = (t_end - t_begin) / J
    for j in range(J + 1):
        t[j] = t_begin + j * tau
    return t,tau
def graph_function_analysis(u,ax_zmax,ax_zmin,time_show):
    fig = figure()
    ax = axes(projection='3d')
    X, Y = meshgrid(y, x)
    ax.plot_surface(X, Y, u, rstride=1, cstride=1, cmap='magma')
    ax.set_title('                                        t={0}'.format(
        time_show*tau))
    ax.set_ylim(x_l-0.2,x_r)
    ax.set_xlim(y_l,y_r+0.2)
    show()
    return
def Slove_progonka(N,a,b,c,parameter,f):
#   parameter[0)f0 1)fN 2)b0 3)aN 4)c0 5)cN]   ; a=a_i b=b_i c=c_i f=f_i
    Y=zeros(N+1)
    alpha=zeros(N)
    beta=zeros(N)
    alpha[0]=parameter[2]/(-parameter[4])
    beta[0]=parameter[0]/parameter[4]
#                                                :
    for n in range(0,N-1,1):
        alpha[n+1]=b/(-c-a*alpha[n])
        beta[n+1]=(a*beta[n]-f[n+1])/(-c-a*alpha[n])
#                                                         :
```

```
    Y[N]=(-parameter[1]+parameter[3]*beta[N-1])/(-parameter[5]-parameter[3]*alpha[N-1])
    for n in range(N-1,-1,-1):
        Y[n]=alpha[n]*Y[n+1]+beta[n]
    return Y
def Choic_type_boundary_condition_for_progonka_parameter(type_l,type_r):
#   parameter[0)f0 1)fN 2)b0 3)aN 4)c0 5)cN]   ; a=a_i b=b_i c=c_i f=f_i
    parameter=zeros(6)
    if type_l==0 and type_r==0:
        parameter=[0,0,0,0,1,1]
    if type_l==1 and type_r==0:
        parameter=[0,0,1,0,-1,1]
    if type_l==0 and type_r==1:
        parameter = [0, 0, 0,-1, 1, 1]
    if type_l==1 and type_r==1:
        parameter = [0, 0, 1, -1, -1, 1]
    return parameter
def graph_function_numberals(u,ax_zmax,ax_zmin,time_show):
    fig = figure()
    ax = axes(projection='3d')
    X, Y = meshgrid(y, x)
    ax.plot_surface(X, Y, u, rstride=1, cstride=1, cmap='magma')
    ax.set_title('                                   t={0}'.format(time_show*tau))
    ax.set_ylim(x_l-0.2,x_r)
    ax.set_xlim(y_l,y_r+0.2)
    show()
    return
def graph_function_errors(u,time_show):
    fig = figure()
    ax = axes(projection='3d')
    X, Y = meshgrid(y, x)
    ax.plot_surface(X, Y, u, rstride=1, cstride=1, cmap='magma')
    ax.set_title('                         t={0}'.format(time_show*tau))
    ax.set_ylim(x_l-0.2,x_r)
    ax.set_xlim(y_l,y_r+0.2)
    show()
    return

#---------------------------------------------------------
x,h=Choic_type_boundary_condition_X_for_scheme(Type_boundary_condition_X[0],Type_boundary_condition_X[1])
y,l=Choic_type_boundary_condition_Y_for_scheme(Type_boundary_condition_Y[0],Type_boundary_condition_Y[1])
t,tau=Layer_Time()
#---------------------------------------------------------

U_analysis=zeros(((J+1,N+1,M+1)))
for j in range(J+1):
    for n in range(N + 1):
        for m in range(M + 1):
            U_analysis[j,n,m]=Function_analysis(t[j],x[n],y[m])
ax_zmax=max(U_analysis) ; ax_zmin=min(U_analysis)

#---------------------------------------------------------
```

```
W_integer=zeros(((J+1,N+1,M+1)))
W_half=zeros(((J+1,N+1,M+1)))
f_right=zeros(((J+1,N+1,M+1)))
for n in range(N + 1):
    for m in range(M + 1):
        W_integer[0,n,m]=Function_begin(x[n],y[m])
paramater_X=Choic_type_boundary_condition_for_progonka_parameter(Type_boundary_condition_X[0],Type_boundary_condition_X[1])
paramater_Y=Choic_type_boundary_condition_for_progonka_parameter(Type_boundary_condition_Y[0],Type_boundary_condition_Y[1])

for j in range(J):
    for n in range(N+1):
        for m in range(M+1):
            f_right[j,n,m]=tau/2*Function_right(t[j]+0.5*tau,x[n],y[m])
    for m in range(1,M,1):
        fi_x=zeros(N+1)
        fi_x[:]=-(W_integer[j,:,m]+0.5*a_c*tau/l**2*(W_integer[j,:,m+1]-2*W_integer[j,:,m]+W_integer[j,:,m-1])+f_right[j,:,m])
        W_half[j,:,m]=Slove_progonka(N,0.5*a_c*tau/h**2,0.5*a_c*tau/h**2,-(a_c*tau/h**2+1),paramater_X,fi_x)[:]
    for n in range(1,N,1):
        fi_y=zeros(M+1)
        fi_y[:]=-(W_half[j,n,:]+0.5*a_c*tau/h**2*(W_half[j,n+1,:]-2*W_half[j,n,:]+W_half[j,n-1,:])+f_right[j,n,:])
        W_integer[j+1,n,:]=Slove_progonka(M,0.5*a_c*tau/l**2,0.5*a_c*tau/l**2,-(a_c*tau/l**2+1),paramater_Y,fi_y)[:]
for j in range(1,J+1,1):
    for m in range(0,M+1,1):
        W_integer[j,0,m]=Type_boundary_condition_X[0]*W_integer[j,1,m]
        W_integer[j,N,m]=Type_boundary_condition_X[1]*W_integer[j,N-1,m]
U_numberals=W_integer
Errors=U_numberals-U_analysis
for i in range(3):
    graph_function_analysis(U_analysis[Time_show[i]],ax_zmax,ax_zmin,Time_show[i])
    graph_function_numberals(U_numberals[Time_show[i]],ax_zmax,ax_zmin,Time_show[i])
    graph_function_errors(Errors[Time_show[i]],Time_show[i])
```