Московский Авиационный Институт

(национальный исследовательский университет)

Институт №8 “Информационные технологии и прикладная математика”

Кафедра 806 “ Вычислительная математика и программирование”

Лабораторная работа № 8 по курсу “Численные методы”

на тему

“ Численные методы решения многомерных задач математической физики”

Студент: Четвергов А. О.

Группа: М8О-409Б-19

Вариант: 3

Преподаватель: Пивоваров Д. Е.

Москва

1. Задание

Используя схемы переменных направлений и дробных шагов, решить двумерную начально-краевую задачу для дифференциального уравнения параболического типа. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением $U(x,t)$. Исследовать зависимость погрешности от сеточных параметров τ, h_x, h_y.

2. Вариант

3.

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + a\frac{\partial^2 u}{\partial y^2},\ a>0,$$

$$u(0,y,t) = \cosh(y)\exp(-3at),$$

$$u(\frac{\pi}{4},y,t) = 0,$$

$$u(x,0,t) = \cos(2x)\exp(-3at),$$

$$u(x,\ln 2,t) = \frac{5}{4}\cos(2x)\exp(-3at),$$

$$u(x,y,0) = \cos(2x)\cosh(y).$$

Аналитическое решение: $U(x,y,t) = \cos(2x)\cosh(y)\exp(-3at)$.

3. Теория

Рассмотрим задачу для двумерного уравнения параболического типа в прямоугольнике со сторонами l_1, l_2 и граничными условиями I-го рода.

Для пространственно-временной области

$$\bar{G}_T = \bar{G}\times[0,T], t\in[0,T], \bar{G} = G+\Gamma, G = \ell_1\times\ell_2$$

Рассмотрим следующую задачу:

$$\frac{\partial u}{\partial t} = a\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f(x,y,t),\quad x\in(0,\ell_1),\quad y\in(0,\ell_2),\quad t>0\ ; \qquad (5.71)$$

$$u(x,0,t) = \varphi_1(x,t),\ x\in[0,\ell_1],\ y=0,\ t>0\ ; \qquad (5.72)$$

$$u(x,\ell_2,t) = \varphi_2(x,t),\ x\in[0,\ell_1],\ y=\ell_2,\ t>0\ ; \qquad (5.73)$$

$$u(0,y,t) = \varphi_3(y,t),\ x=0, y\in[0,\ell_2], t>0; \qquad (5.74)$$

$$u(\ell_1,y,t) = \varphi_4(y,t),\ x=\ell_1, y\in[0,\ell_2],\ t>0; \qquad (5.75)$$

$$u(x,y,0) = \psi(x,y),\ x\in[0,\ell_1],\ y\in[0,\ell_2],\ t=0. \qquad (5.76)$$

Введем пространственно-временную сетку с шагами h_1, h_2, τ соответственно по переменным x, y, t:

$$\omega^{\tau}_{h_1h_2} = \left\{x_i = ih_1, i=\overline{0,I}; x_j = jh_2, j=\overline{0,J}; t^k = k\tau,\ k=0,1,2,\ldots\right\} \qquad (5.77)$$

И на той сетке будем аппроксимировать дифференциальную задачу (5.71)-(5.76) методом конечных разностей.

Метод переменных направлений:

В схеме МПН шаг по времени τ разбивается на число независимых пространственных переменных. На каждом дробном временном слое один из пространственных дифференциальных операторов аппроксимируется неявно (по соответствующему координатному направлению осуществляются скалярные прогонки), а остальное явно. На следующем дробном шаге следующий по порядку дифференциальный оператор аппроксимируется неявно, а остальные явно и т.д. В двумерном случае схема МПН для задачи (5.71)-(5.76) имеет вид

$$\frac{u_{ij}^{k+1/2}-u_{ij}^{k}}{\tau/2}=\frac{a}{h_1^2}\left(u_{i+1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1\,j}^{k+1/2}\right)+\frac{a}{h_2^2}\left(u_{ij+1}^{k}-2u_{ij}^{k}+u_{ij-1}^{k}\right)+f_{ij}^{k+1/2},$$

(5.78)

$$\frac{u_{ij}^{k+1}-u_{ij}^{k+1/2}}{\tau/2}=\frac{a}{h_1^2}\left(u_{i+1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1\,j}^{k+1/2}\right)+\frac{a}{h_2^2}\left(u_{ij+1}^{k+1}-2u_{ij}^{k+1}+u_{ij-1}^{k+1}\right)+f_{ij}^{k+1/2}. \quad (5.79)$$

Метод дробных шагов:

В отличие от МПН метод дробных шагов (МДШ) использует только неявные конечно-разностные операторы, что делает его абсолютно устойчивым в задачах, не содержащих смешанные производные. Он обладает довольно значительным запасом устойчивости и в задачах со смешанными производными.

Для задачи (5.71)-(5.76) схема МДШ имеет вид

$$\frac{u_{ij}^{k+1/2}-u_{ij}^{k}}{\tau}=\frac{a}{h_1^2}\left(u_{i+1\,j}^{k+1/2}-2u_{ij}^{k+1/2}+u_{i-1\,j}^{k+1/2}\right)+\frac{f_{ij}^{k}}{2}, \quad (5.80)$$

$$\frac{u_{ij}^{k+1}-u_{ij}^{k+1/2}}{\tau}=\frac{a}{h_2^2}\left(u_{i\,j+1}^{k+1}-2u_{ij}^{k+1}+u_{i\,j-1}^{k+1}\right)+\frac{f_{ij}^{k+1}}{2}. \quad (5.81)$$

С помощью чисто неявной подсхемы (5.80) осуществляются скалярные прогонки в направлении оси x в количестве, равном J-1, в результате чего получаем сеточную функцию $u_{i,j}^{k+1/2}$. На втором дробном шаге по времени с помощью подсхемы (5.81) осуществляются скалярные прогонки в направлении оси y в количестве, равном I-1, в результате чего получаем сеточную функцию $u_{i,j}^{k+1}$.

4. Программа

Параметры:

```
a = 1
l1 = np.pi/4
l2 = mt.log(2)
l3 = 1
nx = 10
ny = 10
nt = 100
h1 = l1/nx
h2 = l2/ny
tau = l3/nt
```

Дано:

```
def phi1(y,t):
    return np.cosh(y)*np.exp(-3*a*t)
def phi2(y,t):
    return 0
def phi3(x,t):
    return np.cos(2*x)*np.exp(-3*a*t)
def phi4(x,t):
    return 5/4*np.cos(2*x)*np.exp(-3*a*t)
def psi(x,y):
    return np.cos(2*x)*np.cosh(y)
def exact(x,y,t):
    return np.cos(2*x)*np.cosh(y)*np.exp(-3*a*t)
```

Сетка:

```
x = np.arange(0, l1+h1, h1)
y = np.arange(0, l2+h2, h2)
t = np.arange(0, l3+tau, tau)
```

Аналитическое решение:

```
def analytic(x, y, t):
    ext = np.zeros((len(x), len(y), len(t)))
    for i in range(0, len(x)):
        for j in range(0, len(y)):
            for k in range(0, len(t)):
                ext[i,j,k] = exact(x[i], y[j], t[k])
    return ext
```

Метод Прогонки:

```
def swp(a,b,c,d):
    i = np.shape(d)[0]
    def searchP():
        P = np.zeros(i)
        P[0] = -c[0] / b[0]
        for j in range(1, len(P)):
            P[j] = -c[j] / (b[j] + a[j] * P[j - 1])
        return P
    def searchQ():
        Q = np.zeros(i)
        Q[0] = d[0] / b[0]
        for j in range(1, len(Q)):
            Q[j] = (d[j] - a[j] * Q[j - 1]) / (b[j] + a[j] * P[j - 1])
        return Q
    def searchX():
        X = np.zeros(i)
        X[i - 1] = Q[i - 1]
        for j in range(len(X) - 2, -1, -1):
            X[j] = P[j] * X[j + 1] + Q[j]
```

```
        return X
    P = searchP()
    Q = searchQ()
    X = searchX()
    return X
```

Метод переменных направлений и метод дробных шагов:

```
def MPN_MDS(f):
    u = np.zeros((len(x), len(y), len(t)))
    for i in range(len(x)):
        for j in range(len(y)):
            u[i,j,0] = psi(x[i],y[j])
    ap = np.zeros(len(x))
    bp = np.zeros(len(x))
    cp = np.zeros(len(x))
    dp = np.zeros(len(x))
    for k in range(1, len(t)):
        u1 = np.zeros((len(x), len(y)))
        th = t[k-1] + tau/2
        v = u[:,:,k-1]
        for j in range(len(y)-1):
            ap[0] = 0
            bp[0] = 1
            cp[0] = 0
            dp[0] = phi1(y[j], th)
            for i in range(1,len(x)-1):
                if (f==1):
                    ap[i] = -(a*tau) / (2*h1*h1)
                    bp[i] = (a*tau) / (h1*h1) + 1
                    cp[i] = -(a*tau) / (2*h1*h1)
                    dp[i] = (a*tau) / (2*h2*h2) * v[i][j-1] + v[i][j] * (1 -
(a*tau) / (h2*h2)) + (a*tau) / (2*h2*h2) * v[i][j+1]
                if (f==2):
                    ap[i] = -(a * tau) / (h1 * h1)
                    bp[i] = (2 * a * tau) / (h1 * h1) + 1
                    cp[i] = -(a * tau) / (h1 * h1)
                    dp[i] = v[i][j]
            ap[-1] = 0
            bp[-1] = 1
            cp[-1] = 0
            dp[-1] = phi2(y[j], th)
            prg = swp(ap,bp,cp,dp)
            for i in range(len(x)):
                u1[i,j] = prg[i]
                u1[i,0] = phi3(x[i],th)
                u1[i,-1] = phi4(x[i],th)
        for j in range(len(y)):
            u1[0,j] = phi1(y[j],th)
            u1[-1, j] = phi2(y[j], th)

        u2 = np.zeros((len(x), len(y)))
        for i in range(len(x)-1):
            ap[0] = 0
            bp[0] = 1
            cp[0] = 0
            dp[0] = phi3(x[i], t[k])
            for j in range(1,len(y)-1):
                if (f==1):
                    ap[j] = -(a*tau) / (2*h2*h2)
                    bp[j] = 1 + (a*tau) / (h2*h2)
                    cp[j] = -(a*tau) / (2*h2*h2)
                    dp[j] = (a*tau) / (2*h1*h1) * u1[i-1][j] + u1[i][j] * (1
- (a*tau) / (h1*h1)) + (a*tau) / (2*h1*h1) * u1[i+1][j]
                if (f==2):
```

```
                    ap[j] = -(a * tau) / (h2 * h2)
                    bp[j] = 1 + (2 * a * tau) / (h2 * h2)
                    cp[j] = -(a * tau) / (h2 * h2)
                    dp[j] = u1[i][j]
            ap[-1] = 0
            bp[-1] = 1
            cp[-1] = 0
            dp[-1] = phi4(x[i], t[k])
            prg = swp(ap,bp,cp,dp)
            for j in range(len(y)):
                u2[i,j] = prg[j]
                u2[0,j] = phi1(y[j],t[k])
                u2[-1,j] = phi2(y[j],t[k])
        for i in range(len(x)):
            u2[i,0] = phi3(x[i],t[k])
            u2[i, -1] = phi4(x[i], t[k])
        for i in range(len(x)):
            for j in range(len(y)):
                u[i,j,k] = u2[i,j]
    return u
```

График:

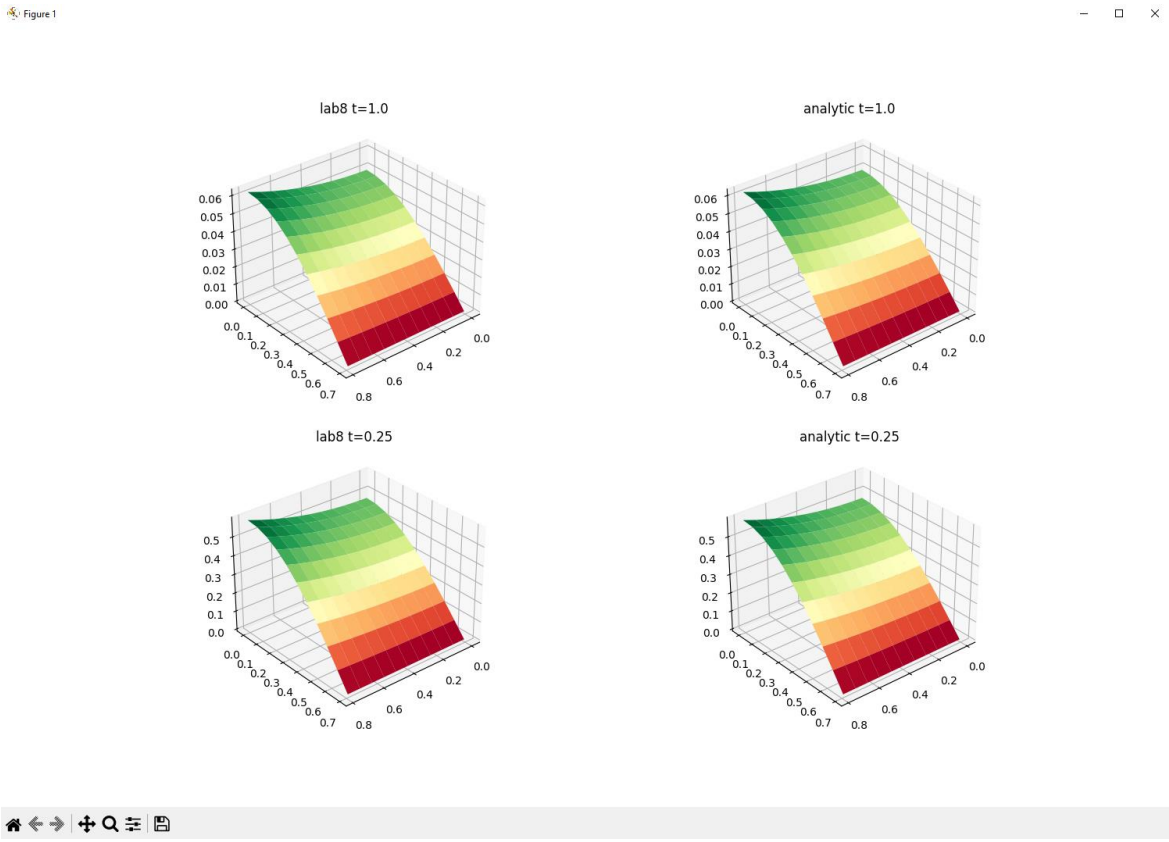
```
def plot(u,analytic,x,y,t1,t2):
    fig = plt.figure(figsize=(15,10))
    x1,y2 = np.meshgrid(x, y)

    pnt = fig.add_subplot(2, 2, 1, projection='3d')
    plt.title(f'lab8 t={t[t1]}')
    pnt.plot_surface(x1,y2,u[:, :, t1],cmap=cm.RdYlGn)
    pnt.view_init(30,50)
    pnt = fig.add_subplot(2, 2, 2, projection='3d')
    plt.title(f'analytic t={t[t1]}')
    pnt.plot_surface(x1,y2,analytic[:, :, t1],cmap=cm.RdYlGn)
    pnt.view_init(30,50)

    pnt = fig.add_subplot(2, 2, 3, projection='3d')
    plt.title(f'lab8 t={t[t2]}')
    pnt.plot_surface(x1,y2,u[:, :, t2],cmap=cm.RdYlGn)
    pnt.view_init(30,50)
    pnt = fig.add_subplot(2, 2, 4, projection='3d')
    plt.title(f'analytic t={t[t2]}')
    pnt.plot_surface(x1,y2,analytic[:, :, t2],cmap=cm.RdYlGn)
    pnt.view_init(30,50)

    plt.show()
```

Main:

```
u1 = MPN_MDS(1)
u2 = MPN_MDS(2)
ext = analytic(x,y,t)
plot(u1,ext,x,y,-1,25)
plot(u2,ext,x,y,-1,25)
```

5. Результат

Метод переменных направлений:



Метод дробных шагов: