Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

Лабораторная работа № 8

По курсу «Численные методы»

«Численное решение многомерных уравнений математической физики»

**Выполнил:**

Гришин П. Ф.

**Группа:**

М8О-401Б-21

**Преподаватель:**

Ревизников Д. Л.

Москва, 2025

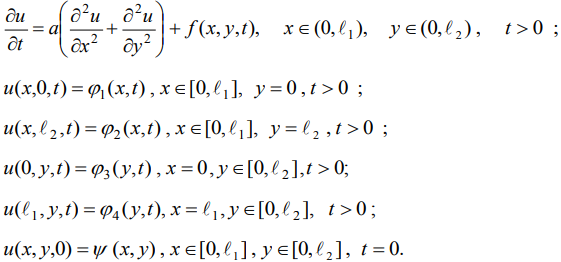
# Задача

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, y).

# Описание метода

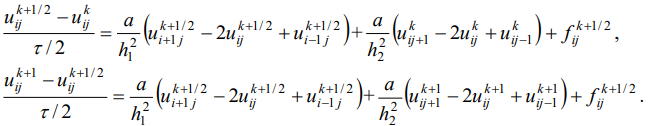
При численном решении многомерных задач математической физики очень важным является вопрос экономичности методов. Рассмотрим два метода из наиболее распространенных, основанных на расщеплении: метод переменных направлений и метод дробных шагов.

Рассмотрим задачу для двумерного уравнения параболического типа в прямоугольнике со сторонами 𝑙1, 𝑙2 и граничными условиями первого рода:



Первым шагом введем пространственно-временную сетку:

Рассмотрим метод переменных направлений. Как и в других методах расщепления, шаг по времени 𝜏 разбивается на число независимых пространственных переменных. На каждом дробном слое один из пространственных дифференциальных операторов аппроксимируется неявно, а остальные явно. На следующем дробном шаге неявно аппроксимируется следующий по порядку дифференциальный оператор, а остальные – явно и тд. Таким образом для двумерного случая мы получаем схему:



Таким образом при помощи скалярным прогонок в направлении переменной х мы

1

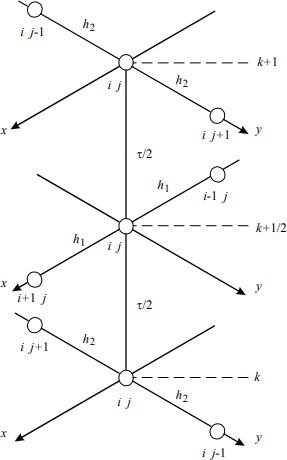
𝑘+

сначала получаем распределение сеточной функции 𝑢𝑖𝑗 2, а после второго этапа прогонок

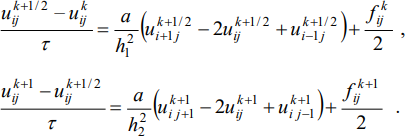
уже в направлении переменной у получаем распределение сеточной функции 𝑢𝑘+1.

𝑖𝑗

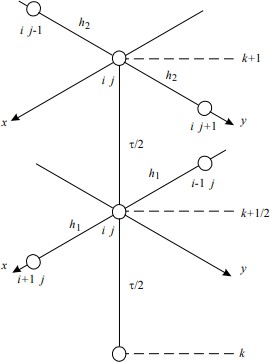
В двумерном случае схема абсолютно устойчива и имеет высокую точность. Также приведем шаблон описанной схемы:



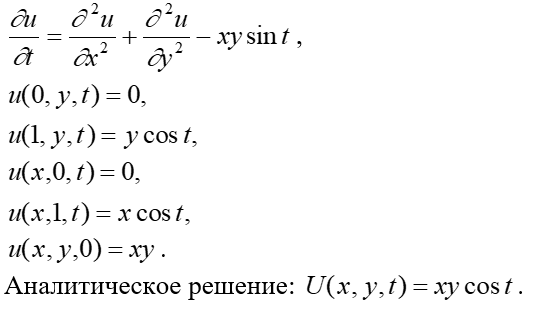
Далее рассмотрим метод дробных шагов, который использует только неявные конечноразностные операторы, что делает его абсолютно устойчивым в задачах, не содержащих смешанные производные.

Для двумерной задачи схема принимает вид:

Таким образом на первом дробном шаге осуществляются скалярные прогонки в направлении х, а на втором дробном шаге – в направлении у. схема метода дробных шагов имеет первый порядок по времени и второй – по переменным х и у.

Приведем шаблон описанной схемы:

# Вариант

****

**Решение и код**

import numpy as np

import matplotlib.pyplot as plt

заданные в варианте функции и метод прогонки

def f(x, y, t):

return -x\*y\*np.sin(t)

def phi1(y, t):

## return 0

def phi2(y, t):

return y\*np.cos(t)

def phi3(x, t):

## return 0

def phi4(x, t):

return x\*np.cos(t)

def psi(x, y):

return x\*y

def solution(x, y, t):

return x\*y\*np.cos(t)

l = 1

N = 10

K = 100

Tk = 2

h = l/N tau = Tk/K

x = np.linspace(0, l, N) y = np.linspace(0, l, N) t = np.linspace(0, Tk, K)

X, Y, T = np.meshgrid(x, y, t)

def tma(a, b, c, d): n = len(a)

p, q = [], []

p.append(-c[0] / b[0])

q.append(d[0] / b[0])

for i in range(1, n):

p.append(-c[i] / (b[i] + a[i] \* p[i - 1]))

q.append((d[i] - a[i] \* q[i - 1]) / (b[i] + a[i] \* p[i - 1])) x = [0 for \_ in range(n)]

x[n - 1] = q[n - 1]

for i in range(n-2, -1, -1):

x[i] = p[i] \* x[i+1] + q[i]

return x

аналитическое решение

def analitic\_solve(N, l, K, T): u = np.zeros((N, N, K)) for i in range(N):

for j in range(N):

for k in range(K):

u[i][j][k] = solution(i\*h, j\*h, k\*tau)

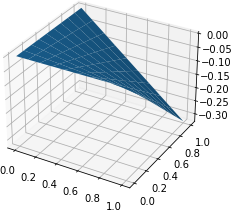
## return u

analitic = analitic\_solve(N, l, K, Tk) x\_plt, y\_plt = np.meshgrid(x, y)

fig = plt.figure()

ax = plt.axes(projection ='3d')

ax.plot\_surface(x\_plt, y\_plt, np.array(analitic[:, :, -1])) plt.show()



метод переменных направлений

def MPN(lbx, ubx, nx, lby, uby, ny, T, K): hx = (ubx - lbx)/nx

x = np.arange(lbx, ubx + hx, hx) hy = (uby - lby)/ny

y = np.arange(lby, uby + hy, hy) tau = T/K

t = np.arange(0, T + tau, tau)

UU = np.zeros((len(x),len(y),len(t)))

for i in range(len(x)):

for j in range(len(y)): UU[i,j,0] = psi(x[i], y[j])

for k in range(1,len(t)):

U1 = np.zeros((len(x),len(y))) t2 = t[k] - tau/2

*#первый дробный шаг*

L = np.zeros((len(x),len(y))) L = UU[:,:,k-1]

for j in range(len(y)-1):

aa = np.zeros(len(x))

bb = np.zeros(len(x))

cc = np.zeros(len(x))

dd = np.zeros(len(x)) bb[0] = hx

bb[-1] = hx cc[0] = 0

aa[-1] = 0

dd[0] = phi1(y[j],t2)\*hx

dd[-1] = phi2(y[j],t2)\*hx

for i in range(1, len(x)-1): aa[i] = 1

bb[i] = hx\*\*2 - 2\*(hx\*\*2)/tau - 2 cc[i] = 1

dd[i] = -2\*(hx\*\*2)\*L[i,j]/tau - 1\*(hx\*\*2)\*(L[i,j+1] - 2\*L[i,j

] + L[i,j-1])/(hy\*\*2) - (hx\*\*2)\*f(x[i],y[j],t2) xx = tma(aa, bb, cc, dd)

for i in range(len(x)):

U1[i,j] = xx[i]

U1[i,0] = (phi3(x[i],t2))

U1[i,-1] = (phi4(x[i],t2))

for j in range(len(y)): U1[0,j] = (phi1(y[j],t2))

U1[-1,j] = (phi2(y[j],t2))

*#второй дробный шаг*

U2 = np.zeros((len(x),len(y)))

for i in range(len(x)-1):

aa = np.zeros(len(x))

bb = np.zeros(len(x))

cc = np.zeros(len(x))

dd = np.zeros(len(x)) bb[0] = hy

bb[-1] = hy cc[0] = 0

aa[-1] = 0

dd[0] = phi3(x[i],t[k])\*hy

dd[-1] = phi4(x[i],t[k])\*hy

for j in range(1, len(y)-1): aa[j] = 1

bb[j] = hy\*\*2 - 2\*(hy\*\*2)/tau - 2 cc[j] = 1

dd[j] = -2\*(hy\*\*2)\*U1[i,j]/tau - 1\*(hy\*\*2)\*(U1[i+1,j] - 2\*U1[

i,j] + U1[i-1,j])/(hx\*\*2) - (hy\*\*2)\*f(x[i],y[j],t[k]) xx = tma(aa, bb, cc, dd)

for j in range(len(y)):

U2[i,j] = xx[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]))

*#print(U2)*

for i in range(len(x)):

for j in range(len(y)): UU[i,j,k] = U2[i,j]

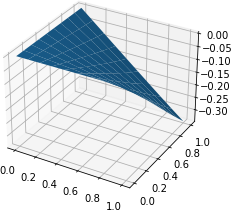
return UU

pn = MPN(0, 1, N, 0, 1, N, Tk, K)

fig = plt.figure()

ax = plt.axes(projection ='3d')

ax.plot\_surface(x\_plt, y\_plt,np.array(pn[:10, :10, -1])) plt.show()



метод дробных шагов

def MDSH(lbx, ubx, nx, lby, uby, ny, T, K): hx = (ubx - lbx)/nx

x = np.arange(lbx, ubx + hx, hx)

hy = (uby - lby)/ny

y = np.arange(lby, uby + hy, hy) tau = T/K

t = np.arange(0, T + tau, tau)

UU = np.zeros((len(x),len(y),len(t)))

for i in range(len(x)):

for j in range(len(y)): UU[i,j,0] = psi(x[i], y[j])

for k in range(1,len(t)):

U1 = np.zeros((len(x),len(y))) t2 = t[k] - tau/2

*#первый дробный шаг*

L = np.zeros((len(x),len(y))) L = UU[:,:,k-1]

for j in range(len(y)-1):

aa = np.zeros(len(x))

bb = np.zeros(len(x))

cc = np.zeros(len(x))

dd = np.zeros(len(x)) bb[0] = hx

bb[-1] = hx cc[0] = 0

aa[-1] = 0

dd[0] = phi1(y[j],t2)\*hx

dd[-1] = phi2(y[j],t2)\*hx

for i in range(1, len(x)-1): aa[i] = 1

bb[i] = - (hx\*\*2)/tau - 2 cc[i] = 1

dd[i] = -(hx\*\*2)\*L[i,j]/tau - (hx\*\*2)\*f(x[i],y[j],t2)/2 xx = tma(aa, bb, cc, dd)

for i in range(len(x)): U1[i,j] = xx[i]

U1[i,0] = (phi3(x[i],t2))

U1[i,-1] = (phi4(x[i],t2))

for j in range(len(y)): U1[0,j] = (phi1(y[j],t2))

U1[-1,j] = (phi2(y[j],t2))

*#второй дробный шаг*

U2 = np.zeros((len(x),len(y)))

for i in range(len(x)-1):

aa = np.zeros(len(x))

bb = np.zeros(len(x))

cc = np.zeros(len(x))

dd = np.zeros(len(x)) bb[0] = hy

bb[-1] = hy

cc[0] = 0

aa[-1] = 0

dd[0] = phi3(x[i],t[k])\*hy

dd[-1] = phi4(x[i],t[k])\*hy

for j in range(1, len(y)-1): aa[j] = 1

bb[j] = - (hy\*\*2)/tau - 2 cc[j] = 1

dd[j] = -(hy\*\*2)\*U1[i,j]/tau - (hy\*\*2)\*f(x[i],y[j],t[k])/2 xx = tma(aa, bb, cc, dd)

for j in range(len(y)):

U2[i,j] = xx[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]))

*#print(U2)*

for i in range(len(x)):

for j in range(len(y)): UU[i,j,k] = U2[i,j]

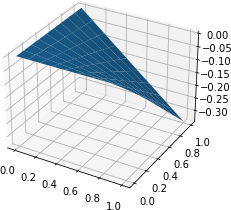
return UU

fs = MDSH(0, 1, N, 0, 1, N, Tk, K)

fig = plt.figure()

ax = plt.axes(projection ='3d')

ax.plot\_surface(x\_plt, y\_plt,np.array(fs[:10, :10, -1])) plt.show()



погрешности для каждого метода в различные моменты времени

def pogr(res, u, t):

return np.sqrt(sum([sum([(u[i][j][t]-res[i][j][t])\*\*2 for j in range(len( x))]) for i in range(len(y))]))

pogr(pn, analitic, 4)

0.08848371356825555

pogr(fs, analitic, 4)

0.00042177518087825605

pogr(pn, analitic, 58)

0.04905504747953388

pogr(fs, analitic, 58)

0.00023965383207673926

pogr(pn, analitic, 94)

0.02901032197841562

pogr(fs, analitic, 94)

0.00013742875416250075

# Выводы

В ходе выполнения этой работы была решена двумерная начально-краевая задача для дифференциального уравнения параболического типа. Для этого использовались две схемы: схема переменных направлений и схема дробных шагов. В различные моменты времени была вычислена погрешность полученных решений путем сравнения с приведенным в задании аналитическим решением, по результатам вычисления

погрешностей можно сделать вывод, что использование метода дробных шагов приводит к более точному решению.