Московский Авиационный Институт (Национальный Исследовательский Университет)

Факультет: “Информационные технологии и прикладная математика”
Кафедра: 806 “Вычислительная математика и программирование”

Отчет по лабораторной работе №8

по курсу «Численные методы»

Студент:
Полей-Добронравова
Амелия Вадимовна
Группа: М8О-407Б,
№ по списку 20
Дата: 22.12.2021
Итоговая оценка:
Подпись преподавателя:

Москва, 2021

Постановка задачи

Лабораторная работа N4

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.

10.

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial y^2} + \sin x \sin y(\mu \cos \mu t + (a+b)\sin \mu t),$$

$$u(0,y,t) = 0,$$
$$u_x(\pi,y,t) = -\sin y \sin(\mu t),$$
$$u(x,0,t) = 0,$$
$$u_y(x,\pi,t) = -\sin x \sin(\mu t),$$
$$u(x,y,0) = 0.$$

Аналитическое решение: $U(x,y,t) = \sin x \sin y \sin(\mu t)$.

1). $a = 1, b = 1, \mu = 1$.
2). $a = 2, b = 1, \mu = 1$.
3). $a = 1, b = 2, \mu = 1$.
4). $a = 1, b = 1, \mu = 2$.

Описание программы

Метод переменных направлений **MPN(...)**

На каждом дробном временном слое один из пространственных дифференциальных операторов аппроксимируется неявно (по соответствующему координатному направлению осуществляются скалярные прогонки), а остальные явно.

В двумерном случае схема метода переменных направлений для задачи:

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

(5.78)

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

Схема аппроксимации:

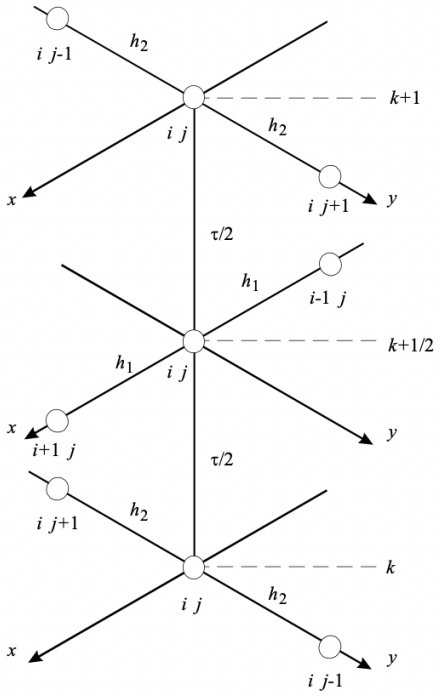


Рис. 5.7 Шаблон схемы метода переменных направлений

Метод дробных шагов **MPSH(...)**

Использует только неявные конечно-разностные операторы, что приводит к абсолютной устойчивости при любом количестве переменных.

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

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

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

Схема аппроксимации:

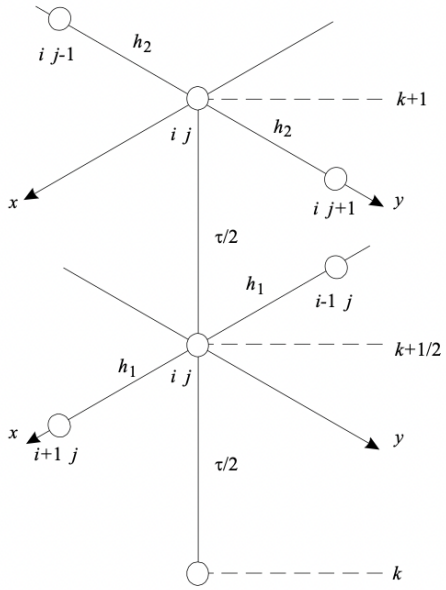
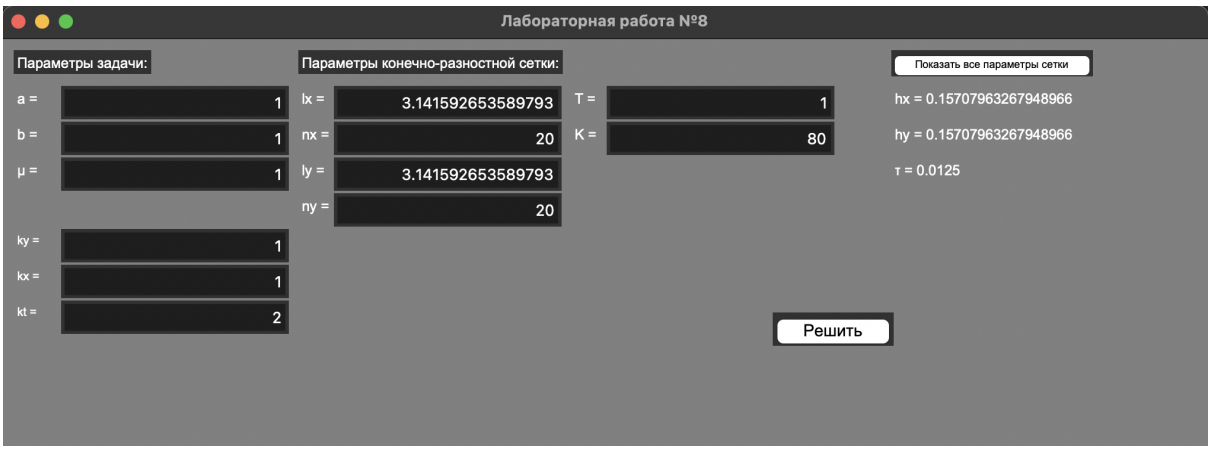
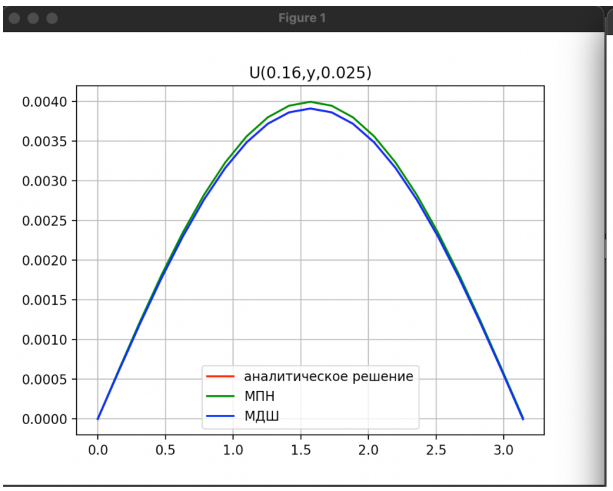
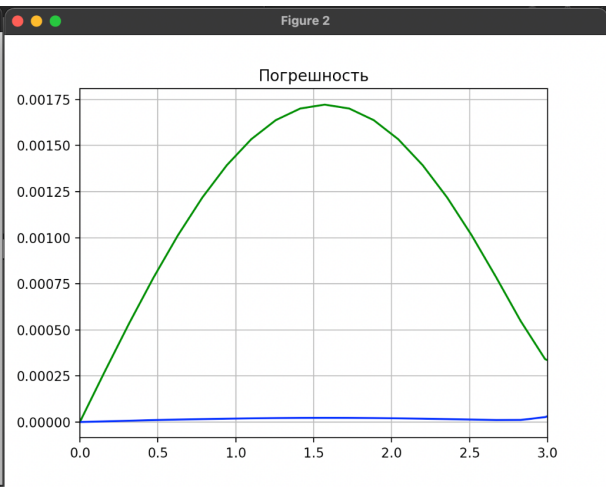


Рис.5.8. Шаблон схемы метода дробных шагов

Демонстрация работы







Исходный код

```
import numpy as np
import math
import matplotlib.pyplot as plt
from math import sqrt
from tkinter import *
from PIL import Image, ImageTk

#вариант 10

def true_fval(x, y, t, mu):
    return np.sin(x) * np.sin(y) * np.sin(mu*t)

def phi11(y, t, mu):
    return 0

def phi12(y, t, mu):
    return - np.sin(y) * np.sin(mu*t)

def phi21(x, t, mu):
    return 0

def phi22(x, t, mu):
    return - np.sin(x) * np.sin(mu*t)

def psi(x, y):
    return 0

def f(x, y, t, mu, a, b):
    return np.sin(x) * np.sin(y) * (mu*np.cos(mu*t) + (a
+ b)*np.sin(mu*t))

def copy_matr(U,x,y):
    L = np.zeros((len(x),len(y)))
    for i in range(len(x)):
        for j in range(len(y)):
            L[i,j] = U[i,j]
    return L

def norma(a):
    norm = 0
    for i in range(len(a)):
        norm += a[i]**2
    return sqrt(norm)


def prog(a, b, c, d):
    n = len(d)
    x=np.zeros(n)
    p = [-c[0] / b[0]]
    q = [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[-1] = q[-1]
    for i in reversed(range(n-1)):
        x[i] = p[i] * x[i + 1] + q[i]
    return x

def MPN(a, b, c, d, e, alfa1, alfa2, beta1, beta2, gama1,
gama2, delta1, delta2, mu, 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)):
        print('Слой k = ',k)
        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]
        #print('L = ', L)
        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*alfa2 - alfa1
            bb[-1] = hx*beta2 + beta1
            cc[0] = alfa1
            aa[-1] = - beta1
            dd[0] = phi11(y[j],t2,mu)*hx
            dd[-1] = phi12(y[j],t2,mu)*hx
            for i in range(1, len(x)-1):
                aa[i] = a - hx*c/2
                bb[i] = hx**2 - 2*(hx**2)/tau - 2*a
                cc[i] = a + hx*c/2
                dd[i] = -2*(hx**2)*L[i,j]/tau -
b*(hx**2)*(L[i,j+1] - 2*L[i,j] + L[i,j-1])/(hy**2) -
d*(hx**2)*(L[i,j+1] - L[i,j-1])/(2 * hy**2) -
(hx**2)*f(x[i],y[j],t2,mu,a,b)
            xx = prog(aa, bb, cc, dd)
            for i in range(len(x)):
                U1[i,j] = xx[i]
                U1[i,0] = (phi21(x[i],t2,mu) -
gama1*U1[i,1]/hy)/(gama2 - gama1/hy)
                U1[i,-1] = (phi22(x[i],t2,mu) +
delta1*U1[i,-2]/hy)/(delta2 + delta1/hy)
        for j in range(len(y)):
            U1[0,j] = (phi11(y[j],t2,mu) -
alfa1*U1[1,j]/hx)/(alfa2 - alfa1/hx)
            U1[-1,j] = (phi12(y[j],t2,mu) +
beta1*U1[-2,j]/hx)/(beta2 + beta1/hx)
        #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*gama2 - gama1
            bb[-1] = hy*delta2 + delta1
            cc[0] = gama1
            aa[-1] = - delta1
            dd[0] = phi21(x[i],t[k],mu)*hy
            dd[-1] = phi22(x[i],t[k],mu)*hy
            for j in range(1, len(y)-1):
                aa[j] = b - hy*d/2
                bb[j] = hy**2 - 2*(hy**2)/tau - 2*b
                cc[j] = b + hy*d/2
                dd[j] = -2*(hy**2)*U1[i,j]/tau -
a*(hy**2)*(U1[i+1,j] - 2*U1[i,j] + U1[i-1,j])/(hx**2) -
c*(hy**2)*(U1[i+1,j] - U1[i-1,j])/(2 * hx**2) -
(hy**2)*f(x[i],y[j],t[k],mu,a,b)
            xx = prog(aa, bb, cc, dd)
            for j in range(len(y)):
                U2[i,j] = xx[j]
                U2[0,j] = (phi11(y[j],t[k],mu) -
alfa1*U2[1,j]/hx)/(alfa2 - alfa1/hx)
                U2[-1,j] = (phi12(y[j],t[k],mu) +
beta1*U2[-2,j]/hx)/(beta2 + beta1/hx)
        for i in range(len(x)):
            U2[i,0] = (phi21(x[i],t[k],mu) -
gama1*U2[i,1]/hy)/(gama2 - gama1/hy)
            U2[i,-1] = (phi22(x[i],t[k],mu) +
delta1*U2[i,-2]/hy)/(delta2 + delta1/hy)
        #print(U2)
        for i in range(len(x)):
            for j in range(len(y)):
                UU[i,j,k] = U2[i,j]
    return UU


def MDSH(a, b, c, d, e, alfa1, alfa2, beta1, beta2, gama1,
gama2, delta1, delta2, mu, 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)):
        print('Слой k = ',k)
        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]
        #print('L = ', L)
        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*alfa2 - alfa1
            bb[-1] = hx*beta2 + beta1
            cc[0] = alfa1
            aa[-1] = - beta1
            dd[0] = phi11(y[j],t2,mu)*hx
```

```
        dd[-1] = phi12(y[j],t2,mu)*hx
        for i in range(1, len(x)-1):
            aa[i] = a
            bb[i] =  - (hx**2)/tau - 2*a
            cc[i] = a
            dd[i] = -(hx**2)*L[i,j]/tau -
(hx**2)*f(x[i],y[j],t2,mu,a,b)/2
        xx = prog(aa, bb, cc, dd)
        for i in range(len(x)):
            U1[i,j] = xx[i]
            U1[i,0] = (phi21(x[i],t2,mu) -
gama1*U1[i,1]/hy)/(gama2 - gama1/hy)
            U1[i,-1] = (phi22(x[i],t2,mu) +
delta1*U1[i,-2]/hy)/(delta2 + delta1/hy)
    for j in range(len(y)):
        U1[0,j] = (phi11(y[j],t2,mu) -
alfa1*U1[1,j]/hx)/(alfa2 - alfa1/hx)
        U1[-1,j] = (phi12(y[j],t2,mu) +
beta1*U1[-2,j]/hx)/(beta2 + beta1/hx)
    #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*gama2 - gama1
        bb[-1] = hy*delta2 + delta1
        cc[0] = gama1
        aa[-1] = - delta1
        dd[0] = phi21(x[i],t[k],mu)*hy
        dd[-1] = phi22(x[i],t[k],mu)*hy
        for j in range(1, len(y)-1):
            aa[j] = b
            bb[j] = - (hy**2)/tau - 2*b
            cc[j] = b
            dd[j] = -(hy**2)*U1[i,j]/tau -
(hy**2)*f(x[i],y[j],t[k],mu,a,b)/2
        xx = prog(aa, bb, cc, dd)
        for j in range(len(y)):
            U2[i,j] = xx[j]
            U2[0,j] = (phi11(y[j],t[k],mu) -
alfa1*U2[1,j]/hx)/(alfa2 - alfa1/hx)
            U2[-1,j] = (phi12(y[j],t[k],mu) +
beta1*U2[-2,j]/hx)/(beta2 + beta1/hx)
    for i in range(len(x)):
        U2[i,0] = (phi21(x[i],t[k],mu) -
gama1*U2[i,1]/hy)/(gama2 - gama1/hy)
        U2[i,-1] = (phi22(x[i],t[k],mu) +
delta1*U2[i,-2]/hy)/(delta2 + delta1/hy)
    #print(U2)
    for i in range(len(x)):
        for j in range(len(y)):
            UU[i,j,k] = U2[i,j]
  return UU
```

```
def plot_U(a, b, c, d, e, alfa1, alfa2, beta1, beta2, gama1,
gama2, delta1, delta2, mu, lbx, ubx, nx, lby, uby, ny, T,
K, kx, ky, kt):
  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)

  UU1 = MPN(a, b, c, d, e, alfa1, alfa2, beta1, beta2,
gama1, gama2, delta1, delta2, mu, lbx, ubx, nx, lby, uby,
ny, T, K)
  #print(UU)
  UU2 = MDSH(a, b, c, d, e, alfa1, alfa2, beta1, beta2,
gama1, gama2, delta1, delta2, mu, lbx, ubx, nx, lby, uby,
ny, T, K)

  plt.figure(1)
  tit1 = 'U('+str("%.2f"%x[kx])+',y,'+str(t[kt])+')'
  plt.title(tit1)
  plt.plot(y, true_fval(x[kx],y,t[kt], mu), color = 'red',
label = 'аналитическое решение')
  plt.plot(y, UU1[kx,:,kt], color = 'green', label = 'МПН')
  plt.plot(y, UU2[kx,:,kt], color = 'blue', label = 'МДШ')
  plt.legend()
  plt.grid()
  plt.figure(2)
  plt.title('Погрешность')
  plt.grid()
  eps = []
  U1 = UU1[:,:,kt]
  for j in range(len(y)):
    a = true_fval(x, y[j], t[kt],mu) - U1[:,j]
    eps = np.append(eps, norma(a))
  plt.plot(y, eps, color = 'green')

  eps = []
  U2 = UU2[:,:,kt]
  for j in range(len(y)):
    a = true_fval(x, y[j], t[kt],mu) - U2[:,j]
    eps = np.append(eps, norma(a))
  plt.plot(y, eps, color = 'blue')
  plt.xlim((0,3))
  plt.show()
  return

root = Tk()
root.title("Лабораторная работа №8")
root["bg"] = "grey"
w = root.winfo_screenwidth() # ширина экрана
h = root.winfo_screenheight() # высота экрана
ww = str(int(w/2))
hh = str(int(h-70))
a = ww + 'x' + hh
w = w//2 # середина экрана
```

```
h = h//2
w = w - 200 # смещение от середины
h = h - 200
root.geometry(a + '+0+0'.format(w, h))

label4 = Label(text = "Параметры задачи:", justify =
LEFT, font = "Arial 12")
label4.place(x = 10, y = 10)

labela = Label(text = "a = ", justify = LEFT, font = "Arial
12", bg = "grey")
labela.place(x = 10, y = 40)
labelb = Label(text = "b = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelb.place(x = 10, y = 70)
labelc = Label(text = "μ = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelc.place(x = 10, y = 100)

entrya = Entry(root, justify = RIGHT)
entrya.place(x = 50, y = 40)

entryb = Entry(root, justify = RIGHT)
entryb.place(x = 50, y = 70)

entrymu = Entry(root, justify = RIGHT)
entrymu.place(x = 50, y = 100)

entrya.insert(0, 1)
entryb.insert(0, 1)
entrymu.insert(0, 1)

label5 = Label(text = "Параметры конечно-разностной
сетки:", justify = LEFT, font = "Arial 12")
label5.place(x = 250, y = 10)
labellx = Label(text = "lx = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labellx.place(x = 250, y = 40)
labelnx = Label(text = "nx = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelnx.place(x = 250, y = 70)
labelly = Label(text = "ly = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelly.place(x = 250, y = 100)
labelny = Label(text = "ny = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelny.place(x = 250, y = 130)

labelT = Label(text = "T = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelT.place(x = 480, y = 40)
labelK = Label(text = "K = ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelK.place(x = 480, y = 70)

entrylx = Entry(root, justify = RIGHT)
entrylx.place(x = 280, y = 40)

entrynx = Entry(root, justify = RIGHT)
entrynx.place(x = 280, y = 70)

entryly = Entry(root, justify = RIGHT)
entryly.place(x = 280, y = 100)

entryny = Entry(root, justify = RIGHT)
entryny.place(x = 280, y = 130)

entryT = Entry(root, justify = RIGHT)
entryT.place(x = 510, y = 40)

entryK = Entry(root, justify = RIGHT)
entryK.place(x = 510, y = 70)

entrylx.insert(0, np.pi)
entrynx.insert(0, 20)
entryly.insert(0, np.pi)
entryny.insert(0, 20)
entryT.insert(0, 1)
entryK.insert(0, 80)


labelhx = Label(text = " ", justify = LEFT, font = "Arial
12", bg = "grey")
labelhx.place(x = 750, y = 40)
labelhy = Label(text = " ", justify = LEFT, font = "Arial
12", bg = "grey")
labelhy.place(x = 750, y = 70)
labeltau = Label(text = " ", justify = LEFT, font = "Arial
12", bg = "grey")
labeltau.place(x = 750, y = 100)
labelerror = Label(text = " ", justify = LEFT, font =
"Arial 12", bg = "grey")
labelerror.place(x = 750, y = 130)


def params():
  try:
    lbx = 0
    ubx = float(entrylx.get())
    lby = 0
    uby = float(entryly.get())
    nx = float(entrynx.get())
    ny = float(entryny.get())
    T = float(entryT.get())
    K = float(entryK.get())
    labelhx.config(text = "hx = {}".format((ubx -
lbx)/nx))
    labelhy.config(text = "hy = {}".format((uby -
lby)/ny))
    labeltau.config(text = "τ = {}".format(T/K))
  except ValueError:
    labelerror.config(text = "Заполните все поля", fg =
"red")

but = Button(root, text = "Показать все параметры
сетки", command = params, font = "Arial 9")
```

```
but.place(x = 750, y = 10)

labelky = Label(text = "ky = ", justify = LEFT, font =
"Arial 10", bg = "grey")
labelky.place(x = 10, y = 160)

entryky = Entry(root, justify = RIGHT)
entryky.place(x = 50, y = 160)
entryky.insert(0, 1)

labelkx = Label(text = "kx = ", justify = LEFT, font =
"Arial 10", bg = "grey")
labelkx.place(x = 10, y = 190)

entrykx = Entry(root, justify = RIGHT)
entrykx.place(x = 50, y = 190)
entrykx.insert(0, 1)

labelkt = Label(text = "kt = ", justify = LEFT, font =
"Arial 10", bg = "grey")
labelkt.place(x = 10, y = 220)

entrykt = Entry(root, justify = RIGHT)
entrykt.place(x = 50, y = 220)
entrykt.insert(0, 2)

ky = float(entryky.get())
kx = float(entryky.get())
kt = float(entrykt.get())

a = float(entrya.get())
b = float(entryb.get())
mu = float(entrymu.get())

c = 0
d = 0
e = 0

alfa1 = 0
alfa2 = 1
beta1 = 1
beta2 = 0
gama1 = 0
gama2 = 1
delta1 = 1
delta2 = 0

lbx = 0
ubx = float(entrylx.get())
nx = float(entrynx.get())
hx = (ubx - lbx)/nx


lby = 0
uby = float(entryly.get())
ny = float(entryny.get())
hy = (uby - lby)/ny

T = float(entryT.get())
K = float(entryK.get())
tau = T/K

def solvv():
  try:
    ky = int(entryky.get())
    kx = int(entrykx.get())
    kt = int(entrykt.get())
    a = float(entrya.get())
    b = float(entryb.get())
    mu = float(entrymu.get())

    c = 0
    d = 0
    e = 0

    alfa1 = 0
    alfa2 = 1
    beta1 = 1
    beta2 = 0
    gama1 = 0
    gama2 = 1
    delta1 = 1
    delta2 = 0

    lbx = 0
    ubx = float(entrylx.get())
    nx = float(entrynx.get())
    hx = (ubx - lbx)/nx

    lby = 0
    uby = float(entryly.get())
    ny = float(entryny.get())
    hy = (uby - lby)/ny

    T = float(entryT.get())
    K = float(entryK.get())
    tau = T/K

    plot_U(a, b, c, d, e, alfa1, alfa2, beta1, beta2,
gama1, gama2, delta1, delta2, mu, lbx, ubx, nx, lby, uby,
ny, T, K, kx, ky, kt)
  except ValueError:
    labelerror.config(text = "Заполните все поля", fg =
"red")

solv = Button(root, text = "  Решить  ", font = "Arial 14",
command = solvv)
solv.place(x = 650, y = 230)

root.mainloop()
```

Выводы

1) При численном решении многомерных задач математической физики исключительно важным является вопрос об *экономичности* используемых методов. (число выполняемых операций пропорционально числу узлов сетки.
2) В двумерном случае схема МПН абсолютна устойчива.
3) К *достоинствам* МНП можно отнести высокую точность. *Недостатки* МНП - условная устойчивость при количестве переменных больше двух.
4) *Достоинства* МДШ: простота в алгоритмизации, запас устойчивости для задач имеющих смешанные производные. *Недостатки*: на дробном шаге достигается дробная аппроксимация, полная только на последнем шаге (суммарная аппроксимация). Первый порядок точности по времени.