ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.ЛОМОНОСОВА»

Физический факультет

КАФЕДРА ФИЗИЧЕСКОЙ ЭЛЕКТРОНИКИ

Основы математического моделирования

«**Отчет по практическому заданию №2, Вариант 6**»

Выполнила студентка

323 группы

Мелкозерова Юлия Алексеевна

Москва

2018

ОГЛАВЛЕНИЕ

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc10789554)

[АНАЛИТИЧЕСКОЕ РЕШЕНИЕ 3](#_Toc10789555)

[ПОСТРОЕНИЕ РАЗНОСТНОЙ СХЕМЫ 4](#_Toc10789556)

[МЕТОД ПРОГОНКИ 5](#_Toc10789557)

[РЕАЛИЗАЦИЯ ПРОГРАММЫ 6](#_Toc10789558)

## ПОСТАНОВКА ЗАДАЧИ

Используя метод переменных направлений решить краевую задачу:

# АНАЛИТИЧЕСКОЕ РЕШЕНИЕ

Решение этой задачи строится в виде ряда по собственным функциям задачи Неймана для прямоугольника, которые имеют вид:

Выделена собственная функция, равная 1, соответствующая нулевому собственному значению. Остальные собственные значения равны:

Решение задачи имеет вид:

где есть решение задачи Коши.

Следовательно n=1, m=2

# ПОСТРОЕНИЕ РАЗНОСТНОЙ СХЕМЫ

Метод конечных разностей заключается в замене искомой функции сеточной функцией, определённой на дискретном множестве узлов. При этом дифференциальные соотношения заменяются соответствующими им конечно-разностными соотношениями.

Для решения нашей задачи введём равномерные сетки:

На введенной сетке будем рассматривать сеточные функции . Далее аппроксимируем оператор Лапласа разностным оператором Λ𝑤=Λ1𝑤+Λ2𝑤,

Тогда уравнение для сеточной функции будет иметь вид

(при – явная схема, при - неявная)

Начальные условия для функции записываются в виде

Граничные условия по 𝑥 аппроксимируются следующим образом:

По y

Явная (𝜎=0) и неявная (𝜎=1) схемы имеют одинаковый порядок точности. При использовании явной схемы число действий, необходимых для вычисления 𝑤𝑘+1 пропорционально числу узлов сетки.

Но явная схема лишь условно устойчива. Для неявной схемы чтобы определить 𝑤𝑘+1 нужно решать систему уравнений, число которых пропорционально числу узлов сетки

Но неявная схема безусловна устойчива. Экономичные схемы сочетают в себе достоинства явных и неявных схем (являются безусловно устойчивыми и требуют при переходе со слоя на слой числа арифметических операций, пропорционального числу узлов сетки). Схема переменных направлений является экономичной разностной схемой.

В схеме переменных направлений переход со слоя на слой осуществляется в два шага, с помощью промежуточного (дробного) слоя. Разностная аппроксимация уравнения будет иметь вид:

Для определения получаем разностную задачу, которая решается методом прогонки :

Где

# МЕТОД ПРОГОНКИ

Решение задачи будем искать в виде: 𝑦

Тогда

Подставим

Приравняв коэффициенты при одинаковых степенях и выразив через

Из граничных условий:

Прямой ход прогонки: вычисление коэффициентов 𝛼𝑛+1, 𝛽𝑛+1 по известным 𝛼1,𝛽1

Из вторых граничных условий:

Найдем :

**Обратный ход прогонки:** по известному и коэффициентам 𝛼𝑛,𝛽𝑛 вычисляем 𝑦𝑛 по формуле :

Достаточные условия устойчивости полученного решения:

# РЕАЛИЗАЦИЯ ПРОГРАММЫ

import math  
from mpl\_toolkits.mplot3d import Axes3D  
import matplotlib.pyplot as plt  
import numpy as np  
import imageio  
  
Nx, Ny, Nt = 100, 100, 1000  
  
x1, x2 = 0.0, math.pi  
y1, y2 = 0.0, 2.0  
t1, t2 = 0.0, 1.0  
  
hx = abs(x2 - x1) / Nx  
hy = abs(y2 - y1) / Ny  
ht = abs(t2 - t1) / Nt  
  
alphax = [[0.0 for i in range(Nx)] for j in range(Ny)]  
betax = [[0.0 for i in range(Nx)] for j in range(Ny)]  
pointx = [[0.0 for i in range(Nx)] for j in range(Ny)]  
fx = [[0.0 for i in range(Nx)] for j in range(Ny)]  
  
alphay = [[0.0 for i in range(Nx)] for j in range(Ny)]  
betay = [[0.0 for i in range(Nx)] for j in range(Ny)]  
pointy = [[0.0 for i in range(Nx)] for j in range(Ny)]  
fy = [[0.0 for i in range(Nx)] for j in range(Ny)]  
  
filenames = []  
  
xn = np.arange(x1, x2, hx)  
ym = np.arange(y1, y2, hy)  
  
for i in range(Nx):  
 for j in range(Ny):  
 pointy[i][j] = math.cos(xn[i]) \* math.sin(math.pi \* ym[j])  
  
fig = plt.figure()  
axes = Axes3D(fig)  
X, Y = np.meshgrid(xn, ym)  
  
surf = axes.plot\_surface(np.array(X), np.array(Y), np.array(pointy), cmap='viridis')  
axes.set\_zlim3d(-1.0, 1.0)  
plt.title('Solution')  
plt.xlabel('y')  
plt.ylabel('x')  
fname = 'dump' + str(0) + '.png'  
plt.savefig(fname, bbox\_inches='tight')  
plt.close()  
  
for k in range(Nt + 1):  
  
 ax = ht / (2.0 \* hx \*\* 2)  
 bx = ht / (2.0 \* hx \*\* 2)  
 cx = 1.0 + ht / hx \*\* 2  
  
 for i in range(Nx):  
 for j in range(Ny):  
 jm1 = j - 1 if j - 1 >= 0 else j  
 jp1 = j + 1 if j + 1 < Ny else j  
 fx[i][j] = 0.5 \* ht / (hy \*\* 2) \* (pointy[i][jm1] + pointy[i][jp1]) + (1.0 - ht / (hy \*\* 2)) \* pointy[i][j]  
  
 for j in range(Ny):  
 alphax[0][j] = 0.0  
 betax[0][j] = 0.0  
 for i in range(Nx - 1):  
 alphax[i + 1][j] = bx / (cx - alphax[i][j] \* ax)  
 betax[i + 1][j] = (ax \* betax[i][j] + fx[i][j]) / (cx - alphax[i][j] \* ax)  
  
 for j in range(Ny):  
 pointx[Nx - 1][j] = betax[Nx - 1][j] / (1.0 - alphax[Nx - 1][j])  
 for i in range(Nx)[Nx - 1:0:-1]:  
 pointx[i - 1][j] = alphax[i][j] \* pointx[i][j] + betax[i][j]  
  
 for j in range(Ny):  
 pointx[0][j] = 0.0  
 pointx[Nx - 1][j] = 0.0  
 for i in range(Nx):  
 pointx[i][0] = pointx[i][1]  
 pointx[i][Nx - 1] = pointx[i][Nx - 2]  
  
 ay = ht / (2.0 \* hy \*\* 2)  
 by = ht / (2.0 \* hy \*\* 2)  
 cy = 1.0 + ht / hy \*\* 2  
  
 for j in range(Ny):  
 for i in range(Nx):  
 im1 = i - 1 if i - 1 >= 0 else i  
 ip1 = i + 1 if i + 1 < Nx else i  
 fy[i][j] = 0.5 \* ht / (hx \*\* 2) \* (pointx[im1][j] + pointx[ip1][j]) + (1.0 - ht / (hx \*\* 2)) \* pointx[i][j]  
  
 for i in range(Nx):  
 alphay[i][0] = 1.0  
 betay[i][0] = 0.0  
 for j in range(Ny - 1):  
 alphay[i][j + 1] = by / (cy - alphay[i][j] \* ay)  
 betay[i][j + 1] = (ay \* betay[i][j] + fy[i][j]) / (cy - alphay[i][j] \* ay)  
  
 for i in range(Nx):  
 pointy[i][Ny - 1] = betay[i][Ny - 1] / (1.0 - alphay[i][Ny - 1])  
 for j in range(Ny)[Ny - 1:0:-1]:  
 pointy[i][j - 1] = alphay[i][j] \* pointy[i][j] + betay[i][j]  
  
 for j in range(Ny):  
 pointy[0][j] = 0.0  
 pointy[Nx - 1][j] = 0.0  
 for i in range(Nx):  
 pointy[i][0] = pointy[i][1]  
 pointy[i][Nx - 1] = pointy[i][Nx - 2]  
  
 if k % 5 == 0:  
  
 X, Y = np.meshgrid(xn, ym)  
 fig = plt.figure()  
 ax = fig.add\_subplot(111, projection='3d')  
 ax.set\_zlim3d(-1.0, 1.0)  
 surf = ax.plot\_surface(np.array(X), np.array(Y), np.array(pointy), cmap='viridis')  
 plt.title('Solution')  
 plt.xlabel('y')  
 plt.ylabel('x')  
 fname = 'dump' + str(k) + '.png'  
 filenames += [fname]  
 plt.savefig(fname, bbox\_inches='tight')  
 plt.close()  
  
images = []  
for filename in filenames:  
 images.append(imageio.imread(filename))  
imageio.mimsave('movie.gif', images)