**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

по дисциплине: «Численные методы математической физики**»**

на тему: «Разработка программ для решения уравнения теплопроводности»

Выполнил: студент гр. ИТП-22

Расшивалов Н.И.  
 Принял: доцент

Стародубцев Е.Г.

Гомель 2021

**Цель работы:** изучить основы использования метода конечных разностей для решения краевых задач.

**ЗАДАНИЕ**

Решить двумерную задачу теплопроводности методом конечных разностей.

Уравнение теплопроводности:

t = 0, T = 89, 0 < x < 60, 0 < y < 70

x = 0: T=8, t>0

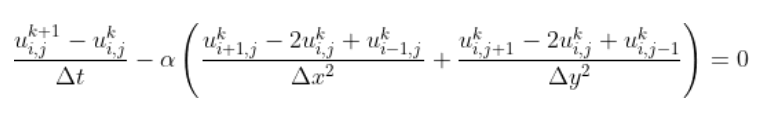
x = 60: T = 1, t>0

y = 0: T = 5, t>0

y = 70: T = 6, t>0

du/dt –alpha(d2u/dx + d2u/dy) = 0

u(x,y,t) = uki,j



Полученное распределение температуры изображены на рисунке 1.

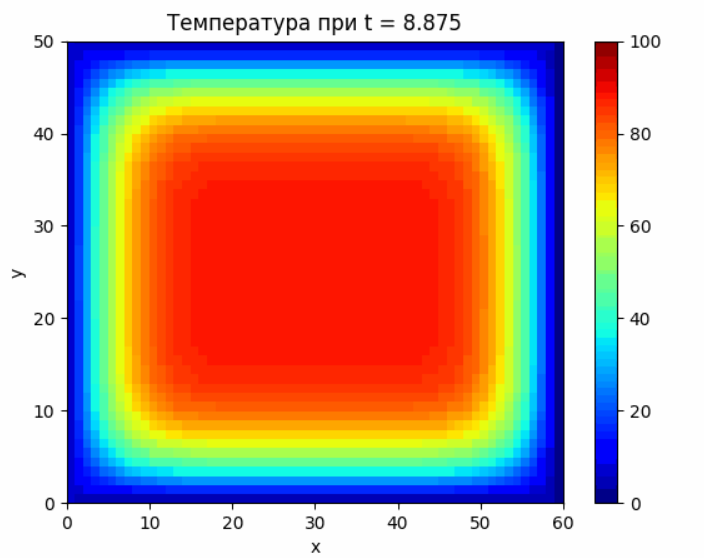


Рисунок 1 – Распределение температуры в определенный момент времени

Графические схемы алгоритмов и листинг программы представлены в приложениях А и Б.

**Вывод**: изучен метод конечных разностей для решения задачи теплопроводности.

# **ПРИЛОЖЕНИЕ А**

**Графические схемы алгоритмов**



Рисунок A.1 – Графическая схема алгоритма решения задачи теплопроводности

u – трехмерный массив значений x,y,t

iterationsNumber – количество итераций по времени

xLength – длина по x

yLength – длина по y

xDelta – шаг по x

yDelta – шаг по y

alpha – теплопроводность материала

# 

# **ПРИЛОЖЕНИЕ Б**

**Листинг программы**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.animation as animation

from matplotlib.animation import FuncAnimation

xLength = 60 #длина по x

yLength = 70 #длина по y

iterationsNumber = 400 #количество итераций

alpha = 2 #коэффицент теплопроводности материала

delta\_x = 1 #шаг по x

delta\_y = 1 #шаг по н

delta\_t = 0.125 #шаг по времени

# Инициализация трехмерного массива

u = np.empty((iterationsNumber, xLength, yLength))

# Начальное условие

u\_initial = 89

# Граничное условие

u\_top = 6.0

u\_left = 8.0

u\_bottom = 5.0

u\_right = 1.0

#Установка начальных услдвия

u.fill(u\_initial)

# Установка граничных условий

u[:, (xLength-1):, :] = u\_top

u[:, :, :1] = u\_left

u[:, :1, 1:] = u\_bottom

u[:, :, (yLength-1):] = u\_right

def calculate(u): #функция вычисляющая значения температуры

for k in range(0, iterationsNumber-1, 1):

for i in range(1, xLength-1, delta\_x):

for j in range(1, yLength-1, delta\_x):

u[k + 1][i][j] = u[k][i][j] + delta\_t \* alpha \* ((u[k][i+1][j]-2\*u[k][i][j]+u[k][i-1][j])/delta\_x\*\*2+(u[k][i][j+1]-2\*u[k][i][j]+u[k][i][j-1])/delta\_y\*\*2)

return u

def plotheatmap(u\_k, k): #создает график распределения температуры

# Clear the current plot figure

plt.clf()

plt.title(f"Температура при t = {k\*delta\_t:.3f}")

plt.xlabel("x")

plt.ylabel("y")

# This is to plot u\_k (u at time-step k)

plt.pcolormesh(u\_k, cmap=plt.cm.jet, vmin=0, vmax=100)

plt.colorbar()

return plt

# Do the calculation here

u = calculate(u)

def animate(k): #задает k и передает значения

plotheatmap(u[k], k)

anim = animation.FuncAnimation(plt.figure(), animate, interval=1, frames=iterationsNumber, repeat=False) #создает анимацию

anim.save('D:\\animation.gif', writer='PillowWriter', fps=20) #сохраняет анимацию в виде gif

print("Выполнено")