Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы численного анализа

**ОТЧЁТ**

к лабораторной работе

на тему

Метод сеток решения задачи Дирихле для уравнения Пуассона

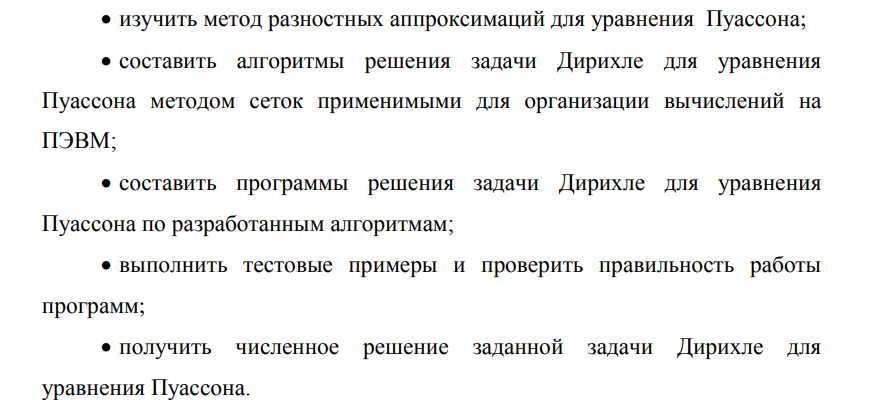
Выполнил: студент группы 153501

Тимофеев Кирилл Андреевич

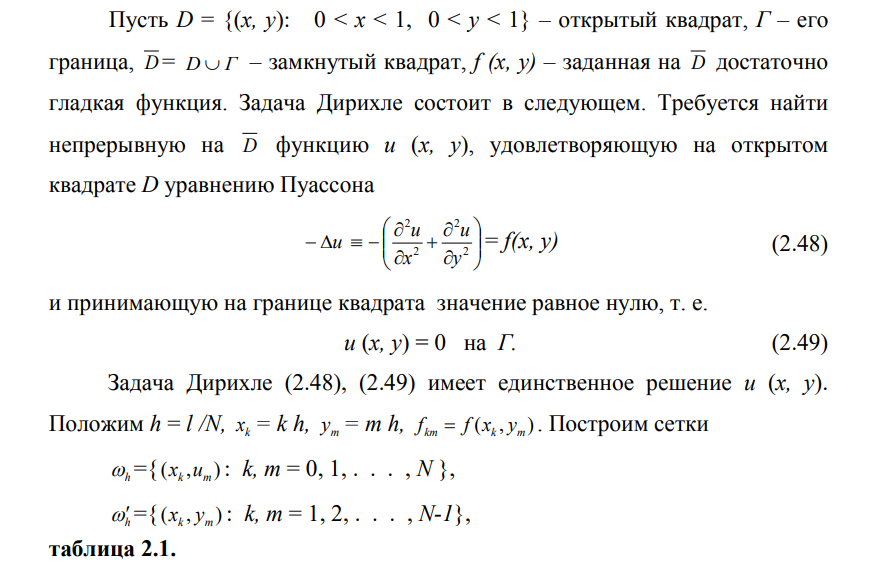
Проверил: Анисимов Владимир Яковлевич

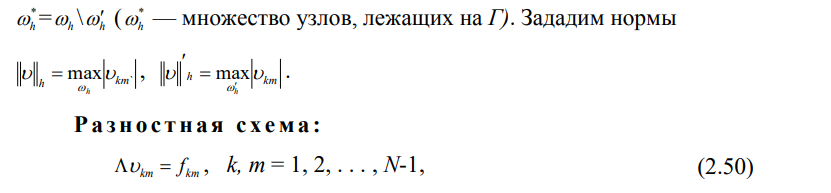
Минск 2023

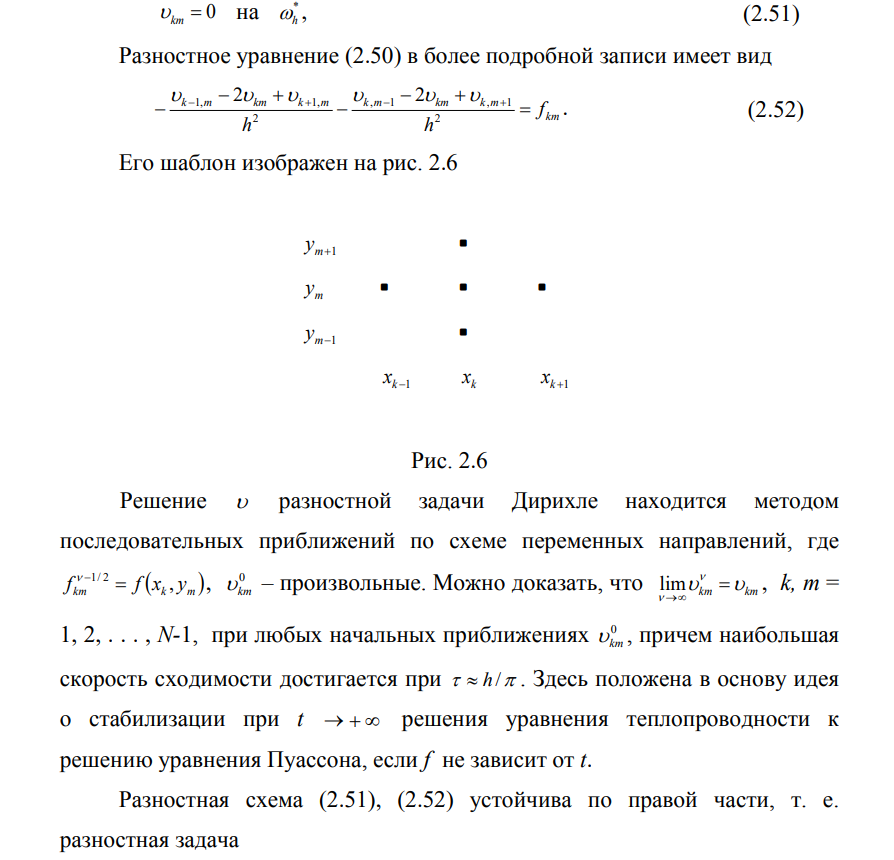
**Цели выполнения задания**

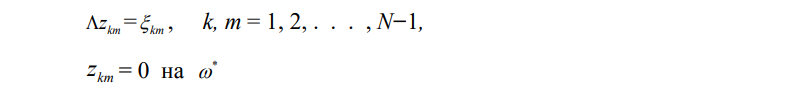


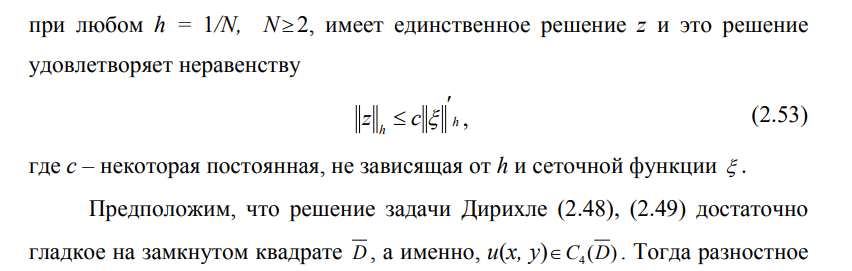
**Краткие теоретические сведения**

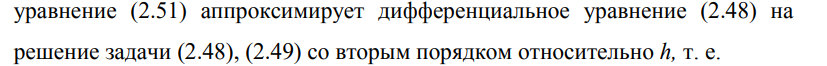


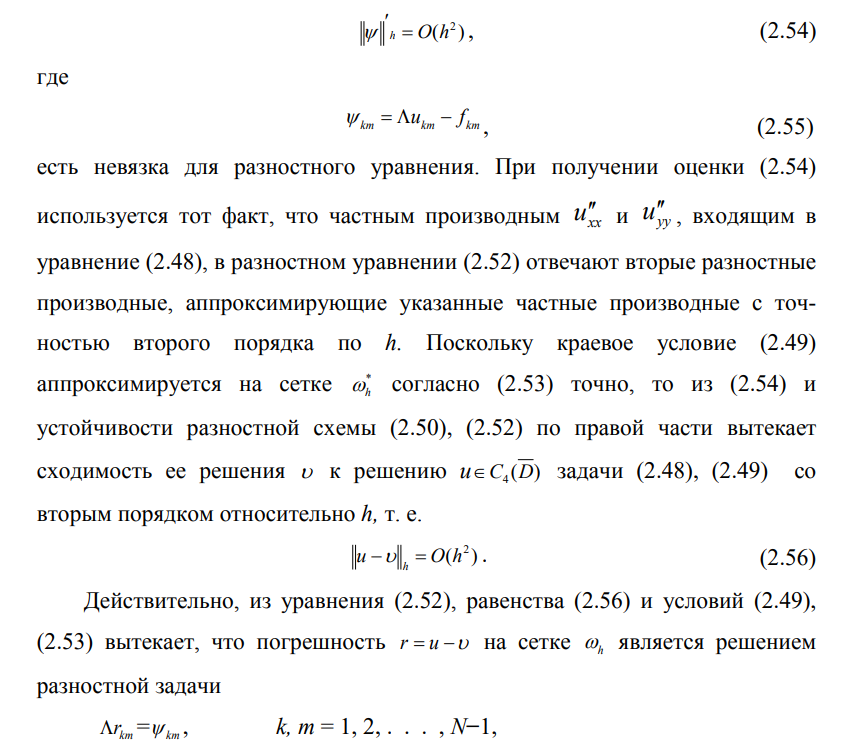




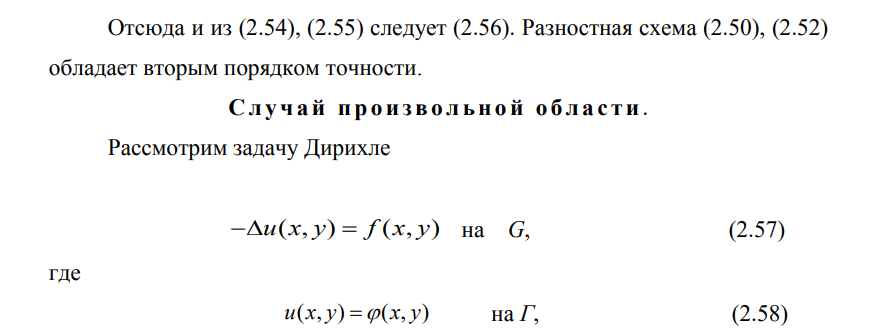


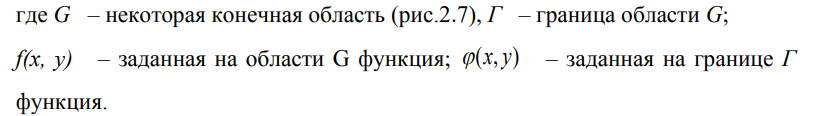


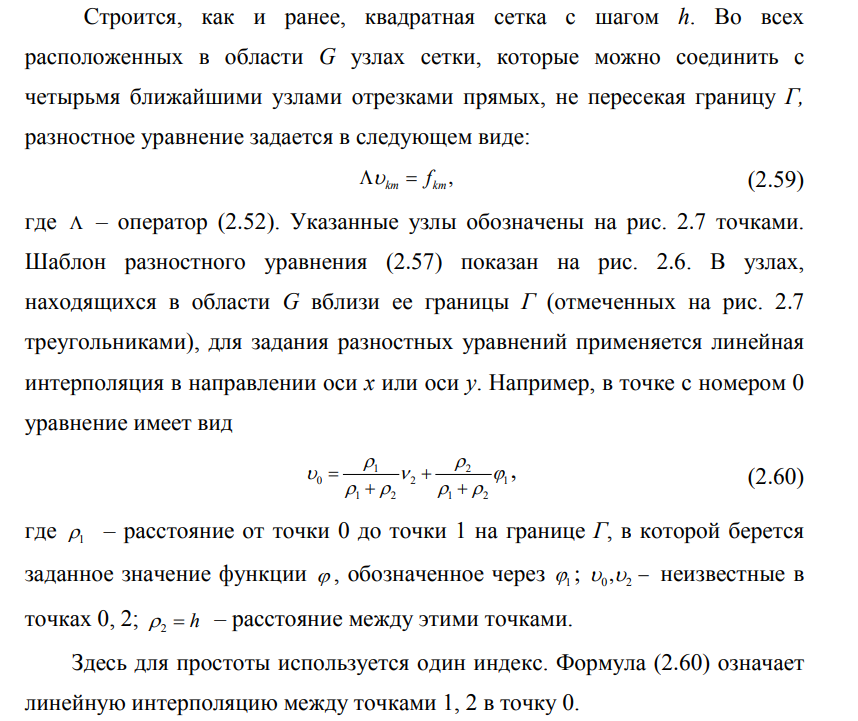


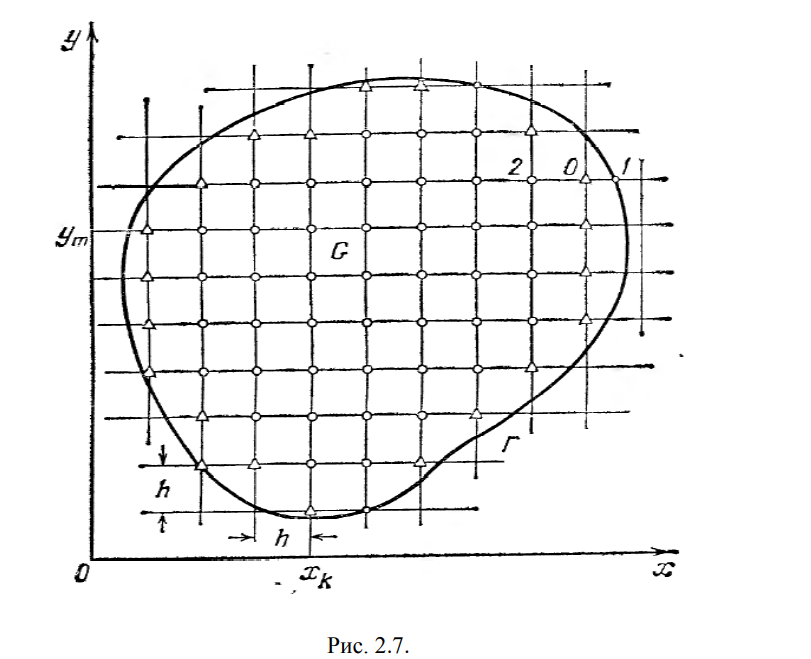


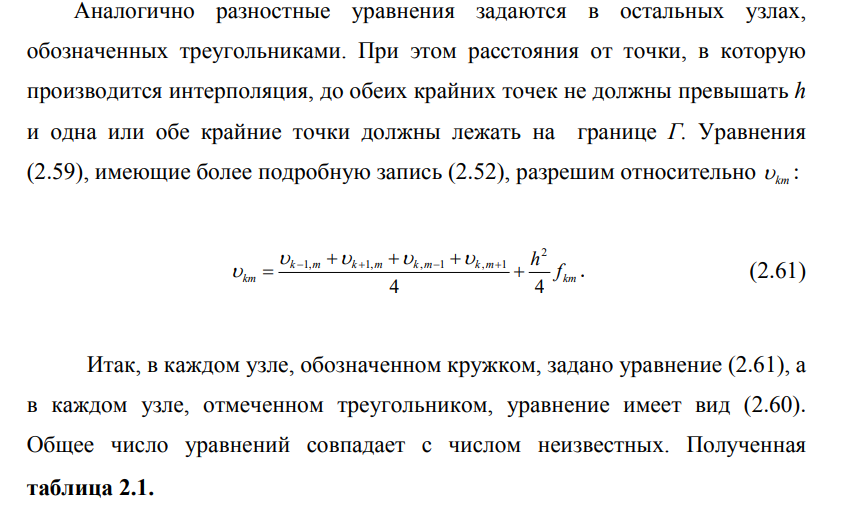


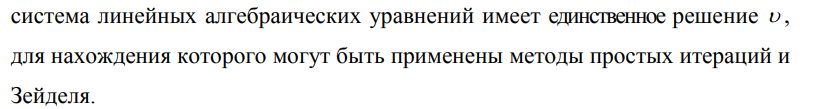


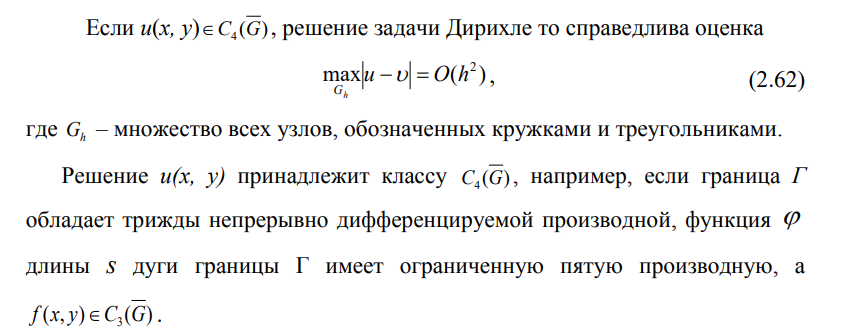


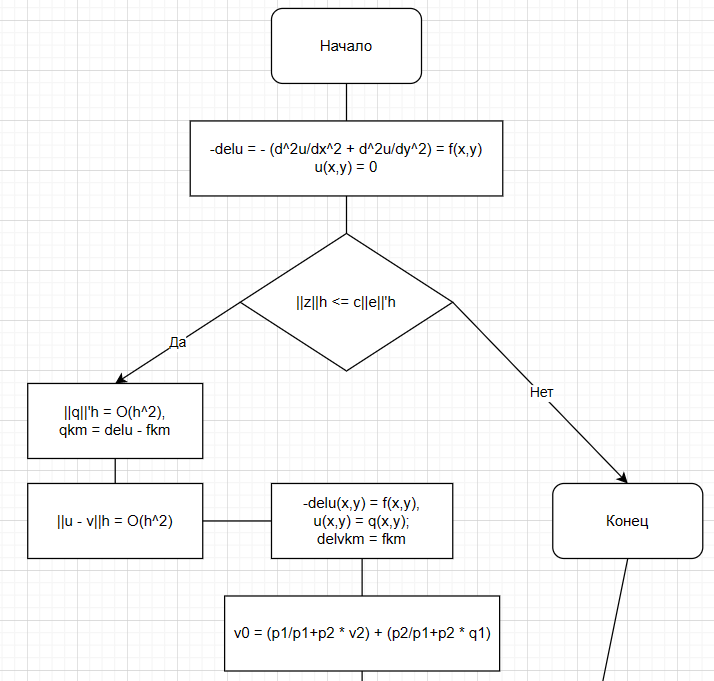


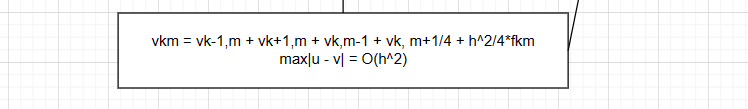




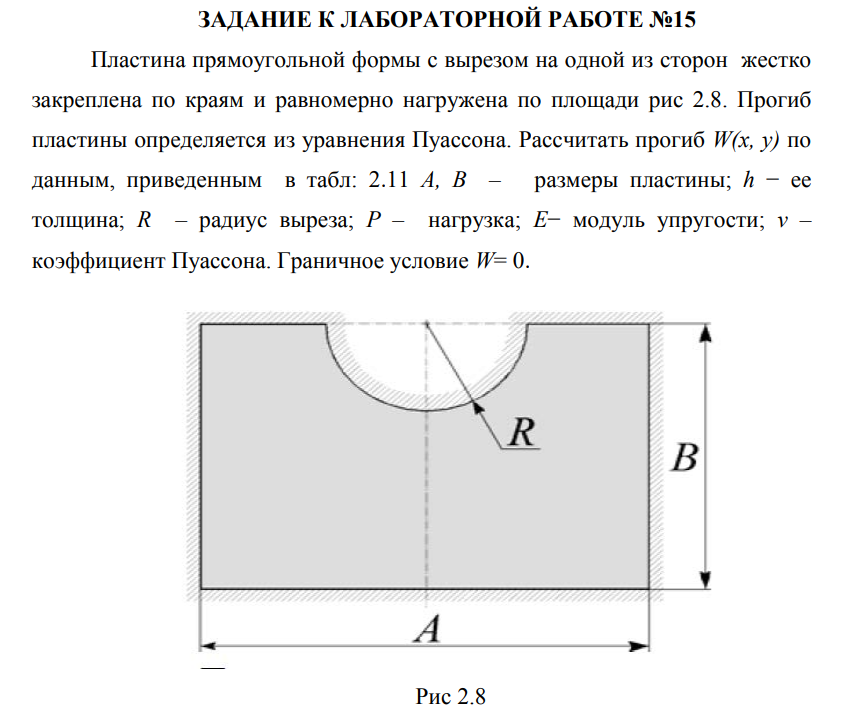


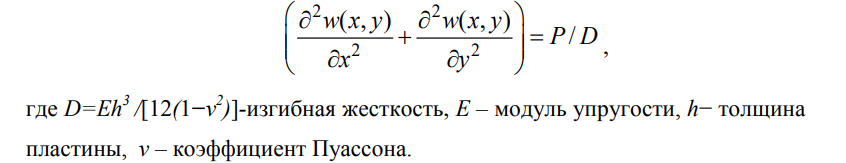


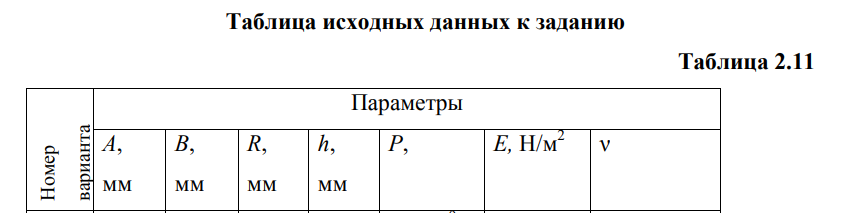




**Задание**





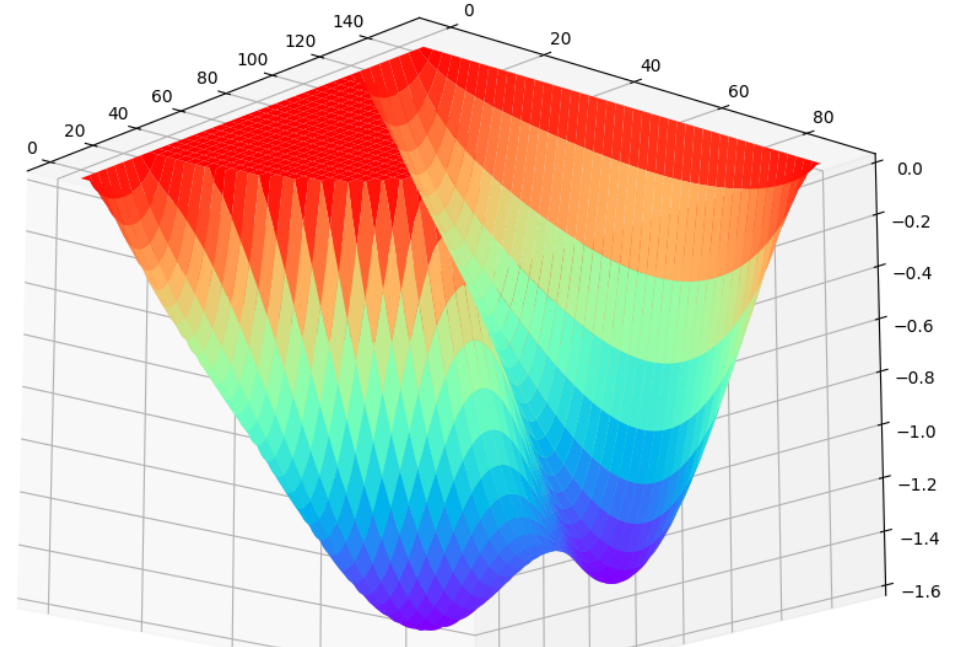


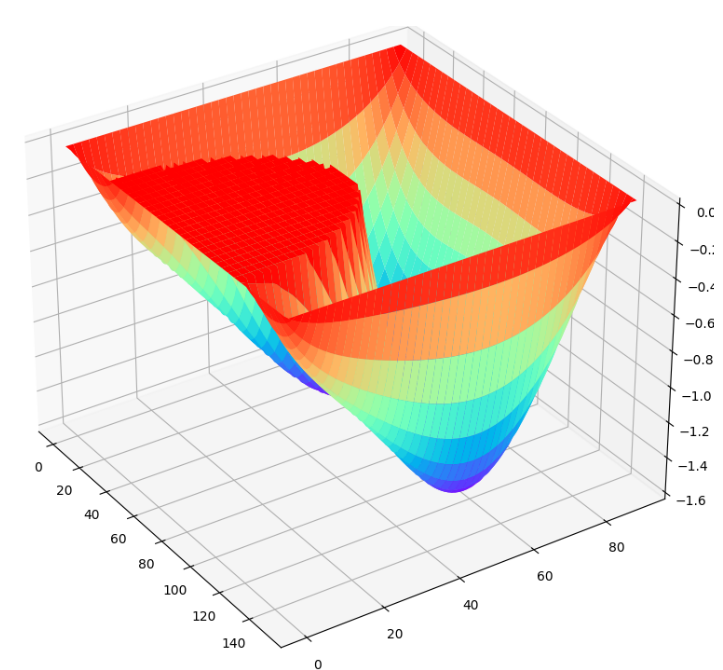


**Тестовое задание**

A = 150; B = 90; R = 45; h = 6; P = 65\*10^9; E = 140; v = 0.28

Шаг равен единице.





**Программная реализация**

import numpy as np

import matplotlib.pyplot as plt

import typing as tp

import scipy.sparse.linalg as spa\_linalg

from math import \*

VARIANT\_NUMBER = 5

width = 180

height = 90

radius = 35

thickness = 5

P = 110\*10^9

E = 120

v = 0.3

step = 1

function = lambda x, y: P / (E \* thickness\*\*3 / (12 \* (1 - v\*\*2)))

def solve(width: float, height: float, radius: float,

function: tp.Callable[[float, float], float], step: float) -> np.array:

cols\_amount = int(ceil(width / step)) + 1

rows\_amount = int(ceil(height / step)) + 1

total\_equations\_amount = cols\_amount \* rows\_amount

A = np.zeros((total\_equations\_amount, total\_equations\_amount))

b = np.zeros(total\_equations\_amount)

def get\_mapped\_index(i: int, j: int) -> int:

return cols\_amount \* i + j

def is\_bound\_point(i: int, j: int) -> bool:

x, y = i \* step, j \* step

if np.isclose(x, 0) or np.isclose(y, 0):

return True

elif x >= height or y >= width:

return True

elif x \*\* 2 + (y - width / 2) \*\* 2 <= radius \*\* 2:

return True

else:

return False

for i in range(rows\_amount):

for j in range(cols\_amount):

if not is\_bound\_point(i, j):

A[get\_mapped\_index(i, j), get\_mapped\_index(i, j - 1)] = 1

A[get\_mapped\_index(i, j), get\_mapped\_index(i, j + 1)] = 1

A[get\_mapped\_index(i, j), get\_mapped\_index(i - 1, j)] = 1

A[get\_mapped\_index(i, j), get\_mapped\_index(i + 1, j)] = 1

A[get\_mapped\_index(i, j), get\_mapped\_index(i, j)] = -4

b[get\_mapped\_index(i, j)] = function(i \* step, j \* step) \* step \*\* 2

else:

A[get\_mapped\_index(i, j), get\_mapped\_index(i, j)] = 0

solution = spa\_linalg.cg(A, b)[0]

return solution.reshape((rows\_amount, cols\_amount))

def plot\_solution(solution: np.array, width: float, height: float, step: float) -> None:

y = np.arange(0, height + step, step)

x = np.arange(0, width + step, step)

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

# print('X:',X,'Y:',Y)

fig = plt.figure(figsize=(10, 10))

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

ax.plot\_surface(X, Y, solution, cmap='rainbow')

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

# solution = solve(width, height, radius, function, thickness)

# plot\_solution(solution, width, height, thickness)

solution = solve(width, height, radius, function, step)

print(solution)

plot\_solution(solution, width, height, step)

# solution = solve(width, height, radius, function, 8)

# plot\_solution(solution, width, height, 8)

#

# solution = solve(width, height, radius, function, 16)

# plot\_solution(solution, width, height, 16)

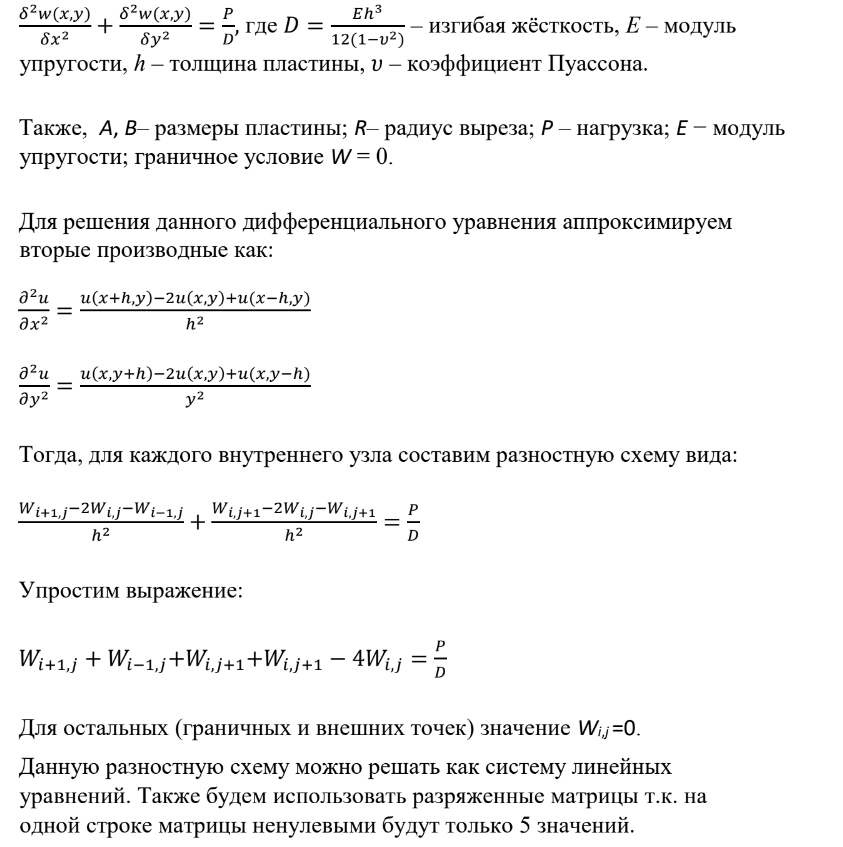
**Полученные результаты**

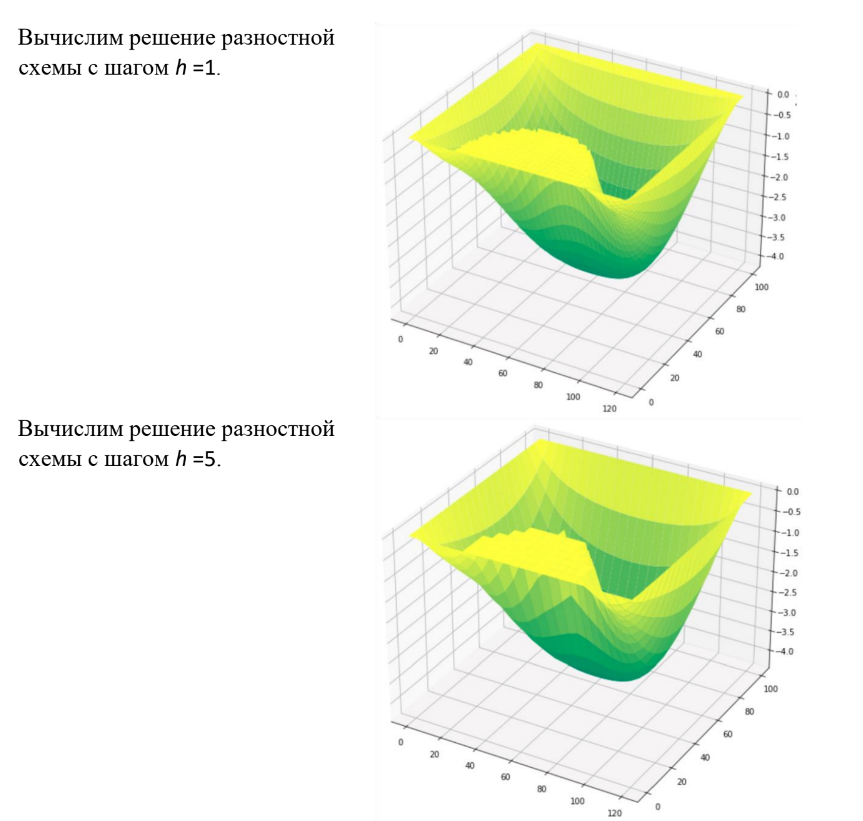
Требуется промоделировать следующий процесс: пластина

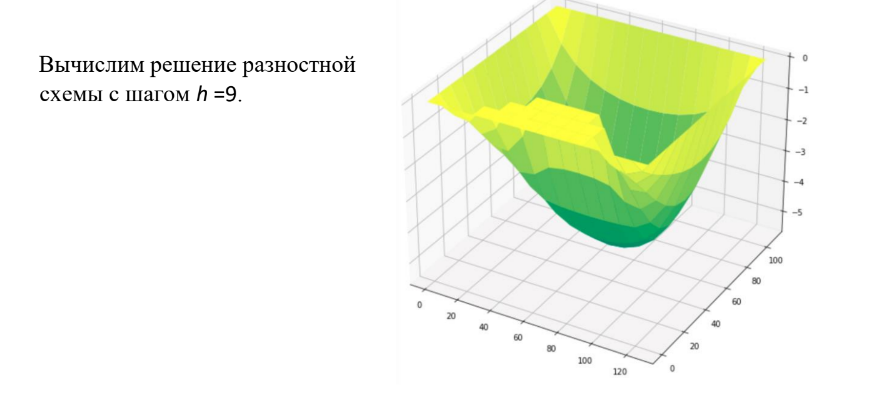
прямоугольной формы с вырезом на одной из сторон жестко закреплена по

краям и равномерно нагружена по площади рис 2.8. Прогиб пластины

определяется из уравнения Пуассона как функции W(x, y).







**Выводы**

В данной лабораторной работе был разработан программный продукт для решения задачи Дирихле для уравнения Пуассона и проверена корректность работы на примере реального физического процесса.

Также, в ходе работы я изучил зависимость качества аппроксимации функции от шага разностной сетки, а также получил визуальное подтверждение гипотезы в виде графика: при увеличении шага качество аппроксимации функции снижается.

Стоит отметить, что для улучшения скорости выполнения программы, использовались разряженные матрицы.