**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра физики**

отчет

**по лабораторной работе №2**

**по дисциплине «Физические основы информационных технологий»**

Тема: Численное решение уравнения Лапласа

|  |  |  |
| --- | --- | --- |
| Студент гр. 1303 |  | Коренев Д. А. |
| Преподаватель |  | Альтмарк А.М. |

Санкт-Петербург

2023

**Условие задания.**

Дана электростатическая система, состоящая из трех электродов. Внешний электрод (на рисунке 1 отмечен синим цветом) обладает потенциалом 0 В. Внутренние электроды (на рисунке отмечены красным цветом и пронумерованы как 1 и 2) обладают потенциалами, отличными от 0. Исходные данные нужно взять в файле FOIT\_IDZ2.xlsx. Для одной из указанных в таблице эквипотенциальных линий необходимо найти длину и записать её в файл IDZ2.txt. Контуры электродов можно построить по формулам, указанным в таблице и сравнить с соответствующим изображением в jpeg – файле. Координаты в данном задании можно считать безразмерными.

Помимо текстового файла IDZ2.txt в папке IDZ2 должен находиться Word-файл с отчетом, а также файл с кодом (Python, Mathcad, Mathematica). Для лучшего понимания отчетности смотрите папку “Пример организации яндекс-папки студентов”.

Пример содержания файла IDZ2.txt:

4.53258

1

2

Рисунок 1. Пример электростатической системы

**Исходные данные**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Вар.** | **Уравнение внешнего электрода** | **Уравнения электрода 1** | **Уравнения электрода 2** | **Потенциал искомой эквипотенциали, В** | **Потенциал на электроде 1, В** | **Потенциал на электроде 2,В** | **Файл с картинкой** |
| 19 | x^2 + y^2 = 25 | Abs[-1.8 + x]^2 + 0.3\*Abs[1.8 + y]^2 = 0.6 | Abs[1.8 + x]^1.5 + 0.5\*Abs[-1.8 + y]^1.5 = 0.8 | 4 | 6 | 5 | 19.jpeg |

**Выполнение работы**

Для выполнения поставленной задачи лабораторная работа была разбита не несколько этапов.

1. Изображение выглядит как круг, диаграмма

   Автоматически созданное описаниеЗадать и отрисовать границы электродов. Границы представлены на рисунке 1.

Рисунок 1 Границы электродов

Полученные границы соответствуют изображению, выданному преподавателем, рисунок 2.

Изображение выглядит как круг, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 2 Границы электродов представленные преподавателем

1. Создать сетку, где на пересечении линий стоят рандомные значения в диапазоне от 0 до 6. Сетка с случайными значениями представлен на рисунке 3. Красные значения соответствуют более высокому значению потенциала (6), синие – более низкому (0).

**Изображение выглядит как снимок экрана, круг, диаграмма, Красочность

Автоматически созданное описание**

Рисунок 3 Сетка случайных значений.

1. Заданное количество раз происходит итерация: значение потенциала в точке равно среднему значению его соседних точек. При этом точки внутри 2 и 3 и на 1 электродах не меняются. Вид сетки после 1000 итераций представлен на рисунке 4.

**Изображение выглядит как снимок экрана, Красочность, круг, диаграмма

Автоматически созданное описание**

Рисунок 4 сетка после итерирования

1. Для нахождения длины эквипотенциали с заданным потенциалом был использован следующий метод:
   1. Сетка разбивается на треугольники в пространстве, таким образом можно реализовать поверхность (три точки в пространстве задают поверхность)
   2. Для каждой из сторон треугольника находится координата, в которой значение Z (в нашем случае это потенциал) равен 4. Очевидно, таких точек не больше двух. Для наглядности приведен рисунок 5 (координаты точек треугольника: (0.7, 0.9, 4.01), (0.8, 0.8, 4.06), (0.8, 0.9, 3.98), координаты точек (0.7475, 0.9, 4), (0.8, 0.88, 4)).

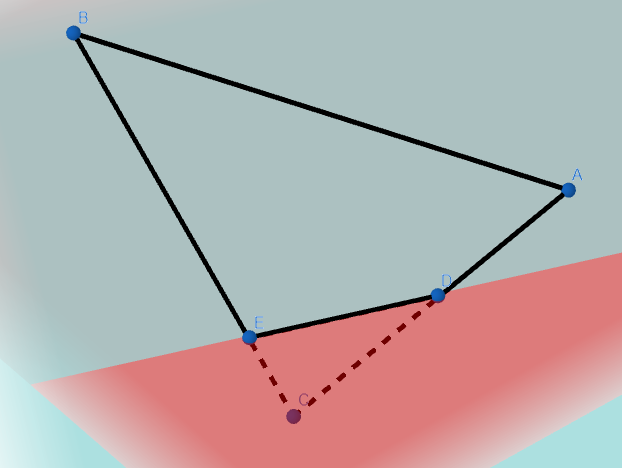
****

Рисунок 5 треугольник пересекает плоскость потенциала 4

* 1. Находится длина линии между двумя точками
  2. Суммируются длины всех линий, результат выводится в консоль. Итоговый результат представлен на рисунке 7, длина эквипотенциали с зарядом 4 равна 19.48 у.е.

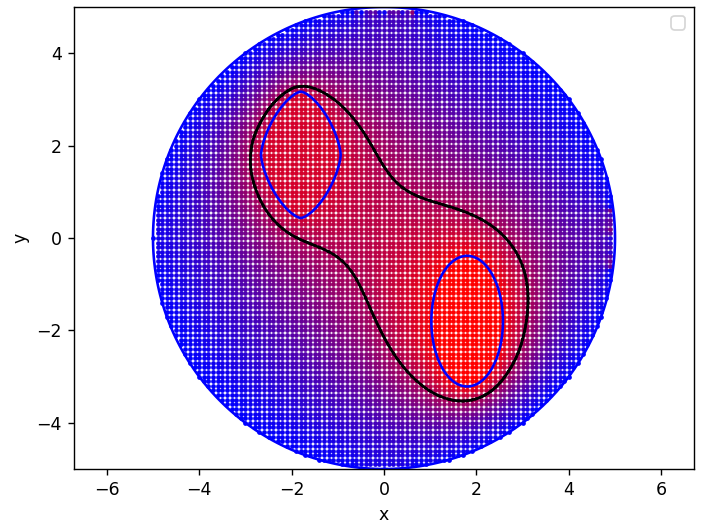
****

Рисунок 6 Итоговый результат сетки

Приложение A

Программа main.py

import random

import numpy as np

import matplotlib.pyplot as plt

p0 = 0

p1 = 6

p2 = 5

pf = 4

hop = 0.1

size = 10

lineNodes = int(size // hop) + 1

mesh = [[] for \_ in range(lineNodes)]

traceX = []

traceY = []

def electrode0(x, y):

return x \*\* 2 + y \*\* 2 - 25

def electrode1(x, y):

return np.abs(-1.8 + x) \*\* 2 + 0.3 \* np.abs(1.8 + y) \*\* 2 - 0.6

def electrode2(x, y):

return np.abs(1.8 + x) \*\* 1.5 + 0.5 \* np.abs(-1.8 + y) \*\* 1.5 - 0.8

def initPlot():

plt.xlabel('x')

plt.ylabel('y')

plt.title('Эквипотенциали')

plt.legend()

plt.grid(False)

plt.axis('equal')

def getColor(value):

if (value == 10): return "#000000"

red = int(value \* 255 / 6)

blue = 255 - int(value \* 255 / 6)

color\_hex = "#{:02x}{:02x}{:02x}".format(red, 0, blue)

return color\_hex

def drawElectrode():

x = np.linspace(-5, 5, 400)

y = np.linspace(-5, 5, 400)

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

Z1 = electrode1(X, Y)

Z2 = electrode2(X, Y)

Z0 = electrode0(X, Y)

plt.contour(X, Y, Z1, levels=[0], colors='b')

plt.contour(X, Y, Z2, levels=[0], colors='b')

plt.contour(X, Y, Z0, levels=[0], colors='b')

def drawMesh():

global mesh

for height in range(0, lineNodes):

for width in range(0, lineNodes):

value = mesh[height][width]

if value is not None:

x = -5 + width \* hop

y = -5 + height \* hop

plt.scatter(x, y, color=getColor(value), s=3)

def isOn(x, y):

if -5 \* hop <= electrode0(x, y) <= 0:

return 0 # на 0 электроде

if electrode0(x, y) > 0:

return -2 # снаружи

if electrode1(x, y) <= 0:

return 1 # внутри 1 электрода

if electrode2(x, y) <= 0:

return 2 # внутри 2 электрода

else:

return -1 # внутри 0 электрода

def setMash(w, h, value):

global mesh

mesh[h][w] = value

def fillRandomNode():

for height in range(0, lineNodes):

for width in range(0, lineNodes):

mesh[height].append(None)

x = -5 + width \* hop

y = -5 + height \* hop

pos = isOn(x, y)

if pos != -1:

match pos:

case 0:

setMash(width, height, p0)

case 1:

setMash(width, height, p1)

case 2:

setMash(width, height, p2)

elif pos == -1:

mesh[height][width] = random.random() \* ((p0 + p1 + p2) / 3)

else:

mesh[height][width] = None

def iteration(count):

for i in range(count):

# print(i)

for height in range(1, lineNodes - 1):

for width in range(1, lineNodes - 1):

x = -5 + width \* hop

y = -5 + height \* hop

if isOn(x, y) == -1:

neighbours = [mesh[height - 1][width],

mesh[height + 1][width],

mesh[height][width - 1],

mesh[height][width + 1]]

neighbours = [value for value in neighbours if value is not None]

mesh[height][width] = sum(neighbours) / len(neighbours)

def toR(v):

return -(size / 2) + v \* hop

def getPointCoord(point1, point2):

x1, y1, z1 = point1

x2, y2, z2 = point2

if z1 == pf:

return point1

elif z2 == pf:

return point2

if z1 == z2:

return None

t = (pf - z1) / (z2 - z1)

if 0 <= t <= 1:

x = x1 + t \* (x2 - x1)

y = y1 + t \* (y2 - y1)

z = pf

return x, y, z

return None

def calcDistance(point1, point2):

return ((point2[0] - point1[0]) \*\* 2 + (point2[1] - point1[1]) \*\* 2 + (point2[2] - point1[2]) \*\* 2) \*\* 0.5

def calcLenLineInTriangle(triangle):

length = 0

coord = [getPointCoord(triangle[0], triangle[1]), getPointCoord(triangle[0], triangle[2]),

getPointCoord(triangle[1], triangle[2])]

coord = [point for point in coord if point is not None]

if len(coord) == 2:

length += calcDistance(coord[0], coord[1])

traceX.append(coord[0][0])

traceX.append(coord[1][0])

traceY.append(coord[0][1])

traceY.append(coord[1][1])

plt.plot([traceX[-2], traceX[-1]], [traceY[-2], traceY[-1]], color="black")

return length

def findLength():

length = 0

for height in range(0, lineNodes - 1):

for width in range(0, lineNodes - 1):

if mesh[height][width] is None: continue

if mesh[height + 1][width] is None: continue

if mesh[height][width + 1] is None: continue

if mesh[height + 1][width + 1] is None: continue

triangleLow = [[toR(width), toR(height), mesh[height][width]],

[toR(width + 1), toR(height), mesh[height][width + 1]],

[toR(width), toR(height + 1), mesh[height + 1][width]]]

triangleTop = [[toR(width), toR(height + 1), mesh[height + 1][width]],

[toR(width + 1), toR(height), mesh[height][width + 1]],

[toR(width + 1), toR(height + 1), mesh[height + 1][width + 1]]]

if max(triangleTop[0][2],

triangleTop[1][2],

triangleTop[2][2]) > pf > min(triangleTop[0][2],

triangleTop[1][2],

triangleTop[2][2]):

length += calcLenLineInTriangle(triangleTop)

if max(triangleLow[0][2],

triangleLow[1][2],

triangleLow[2][2]) > pf > min(triangleLow[0][2],

triangleLow[1][2],

triangleLow[2][2]):

length += calcLenLineInTriangle(triangleLow)

return length

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

initPlot()

drawElectrode()

fillRandomNode()

iteration(2000)

length = findLength()

print(length)

drawMesh()

plt.show()