Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №2 по дисциплине

"Физические основы информационных технологий"

Название: Численное решение уравнения Лапласа.

|  |  |
| --- | --- |
| Фамилия И.О.: | Куклина Ю.Н. |
| группа: | 1303 |
| Преподаватель: | Альтмарк А.М. |
| Итоговый балл: |  |
|  |  |

Крайний срок сдачи: 05.11.23

.

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

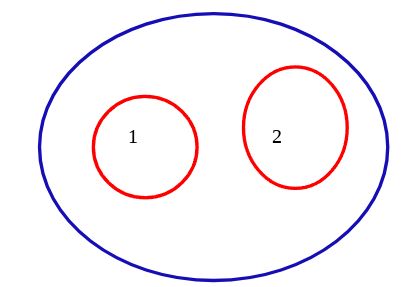


Рисунок 1. Пример электростатической системы

Таблица с исходными данными

Вариант 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Уравнение внешнего электрода | Уравнение первого электрода | Уравнение второго электрода | Потенциал искомой эквипотенциали | Потенциал на электроде 1  Ф1 | Потенциал на электроде 2  Ф2 |
| x^2 + y^2 = 25 | Abs[-1.5 + x]^2.5 + 0.8 Abs[y]^2.5 = 0.7 | Abs[1.5 + x]^1.5 + Abs[y]^1.5 = 0.8 | -4 | -5 | 5 |

Теоретические положения

Алгоритм решения:

1. Разбиваем область внутри внешнего электрода и вне маленьких на прямоугольники.
2. Каждому узлу сетки присваиваем значение потенциала. Если узлы попали на границы электродов, то им устанавливается потенциал этого электрона, иначе случайном образом в диапазоне от потенциала 1 и 2 электродов.
3. Затем все значения на узлах пересчитываются как среднее арифметическое соседних узлов. И повторяется пока разность потенциалов на s-1 и s итерации не станет меньше заданной точности.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

import matplotlib.pyplot as plt

import random

import numpy as np

class Point:

def \_\_init\_\_(self, x, y, f):

self.x = x

self.y = y

self.f = f

F1 = -5

F2 = 5

f = -4

step = 0.1

arrStart = np.arange(F1, F2 + 1, step)

arr\_x = []

arr\_y = []

arr\_point = []

def isIn(x, y):

res1 = x \*\* 2 + y \*\* 2 <= 25

res2 = 0.8 \* abs(-1.5 + x) \*\* 2.5 + 0.8 \* abs(y) \*\* 2.5 >= 0.7

res3 = abs(1.5 + x) \*\* 1.5 + abs(y) \*\* 1.5 >= 0.8

return res1 and res2 and res3

def init():

for x in arrStart:

arr\_point.append([])

for y in arrStart:

if isIn(x, y):

if (abs(x \*\* 2 + y \*\* 2 - 25) <= 1):

f = 0

elif (abs(0.8 \* abs(-1.5 + x) \*\* 2.5 + 0.8 \* abs(y) \*\* 2.5 - 0.7) <= 0.4):

f = F1

elif (abs(abs(1.5 + x) \*\* 1.5 + abs(y) \*\* 1.5 - 0.8) <= 0.4):

f = F2

else:

f = random.uniform(min(F1, F2), max(F1, F2))

else:

f = None

point = Point(x, y, f)

arr\_point[-1].append(point)

def iteration():

n = 0

while (n <= 500):

for y in range(1, len(arr\_point) - 1):

for x in range(1, len(arr\_point[y]) - 1):

neighboors = [arr\_point[y][x - 1].f,

arr\_point[y][x + 1].f,

arr\_point[y - 1][x].f,

arr\_point[y + 1][x].f]

if (any(potent is None for potent in neighboors)):

continue

arr\_point[y][x].f = sum(neighboors) / 4

n += 1

def findPoint(tmp, neighboor, flag):

diapason = abs(tmp.f - neighboor.f)

s = abs(f - tmp.f)

if (flag == 'horisontal'):

arr\_x.append(tmp.x + (s / diapason) \* step)

arr\_y.append(tmp.y)

else:

arr\_x.append(tmp.x)

arr\_y.append(tmp.y + (s / diapason) \* step)

def find(f):

for i in range(1, len(arr\_point) - 2):

for j in range(1, len(arr\_point[i]) - 2):

tmp = arr\_point[i][j]

right = arr\_point[i + 1][j]

down = arr\_point[i][j + 1]

if (tmp.f is None or right.f is None or down.f is None):

continue

if (tmp.f <= f <= right.f or tmp.f >= f >= right.f):

findPoint(tmp, right, 'horisontal')

if (tmp.f <= f <= down.f or tmp.f >= f >= down.f):

findPoint(tmp, down, 'vertical')

def sort():

points = list(zip(arr\_x, arr\_y))

leftmost\_point, \*\_, rightmost\_point = sorted(points, key=lambda lst: lst[0])

x1, y1 = leftmost\_point

x2, y2 = rightmost\_point

tan = (y2 - y1) / (x2 - x1)

points\_below = [(x1, y1)]

points\_above = [(x2, y2)]

for x, y in points:

border = tan \* (x - x1) + y1

if y < border:

points\_below.append((x, y))

elif y > border:

points\_above.append((x, y))

points\_below.sort()

points\_above.sort(reverse=True)

points\_above.append((x1, y1))

correct\_points = points\_below + points\_above

x\_lst = [x for x, \_ in correct\_points]

y\_lst = [y for \_, y in correct\_points]

plt.plot(x\_lst, y\_lst, c='black')

return correct\_points

def findLength(merged\_points):

x0 = merged\_points[-1][0]

y0 = merged\_points[-1][1]

l = 0

for x1, y1 in merged\_points:

l += np.sqrt((x1 - x0) \*\* 2 + (y1 - y0) \*\* 2)

x0, y0 = x1, y1

return l

init()

iteration()

arr = []

for sublist in arr\_point:

for elem in sublist:

if elem.f is not None:

arr.append(elem)

plt.rcParams.update({'figure.figsize': (6, 6)})

plt.scatter([elem.x for elem in arr], [elem.y for elem in arr])

find(f)

pointsArr = sort()

lenLine = findLength(pointsArr)

print("Длина = ", lenLine)

plt.show()