Санкт-Петербургский Государственный Электротехнический Университет "ЛЭТИ"

кафедра физики

Задание №2 по дисциплине

"Физические основы информационных технологий"

Название: Численное решение уравнения Лапласа

Фамилия И.О.:	Чернякова В.А.
Группа:	1304
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.

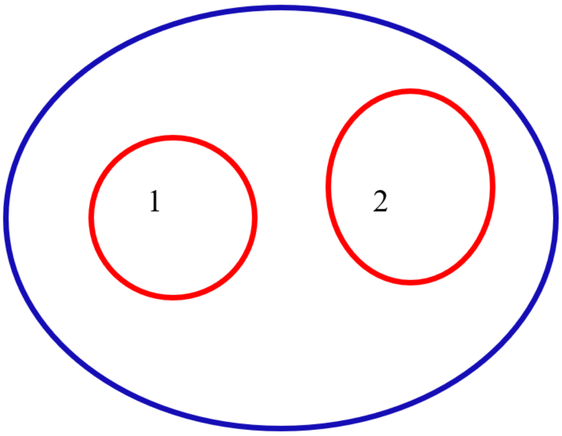

Рисунок 1 – пример электростатической системы.

Вариант 18.

Данные.

Уравнение внешнего электрода: $x^2 + y^2 = 25$

Уравнение электрода 1: $Abs[-1.8 + x]^2 + 0.8*Abs[y]^2 = 0.6$

Уравнение электрода 2: $Abs[1.8 + x]^4 + 0.5*Abs[y]^4 = 0.8$

Потенциал искомой эквипотенциали, В: 1

Потенциал на электроде 1, В: 6

Потенциал на электроде 2, В: -5

Файл с картинкой: 18.jpeg

Основные теоретические положения.

В данной задаче электроды – источники электростатического поля. Потенциал – исследуемая физическая величина.

Чтобы найти значения потенциалов в зависимости от удаленности от электродов можно использовать численный метод решения уравнения Лапласа: $\Delta\varphi = 0$.

Можно рассмотреть два метода сеток решения данного уравнения:

- Разбиение на прямоугольники. Является простым способом.
- Триангуляция. Узлы сетки, произвольно выбранные в области и на границе, соединяются не пересекающимися отрезками так, чтобы каждый внутренний узел был вершиной 6 треугольников (элементов). Данный способ более правильный и оптимальный.

Эквипотенциальная поверхность — это поверхность, на которой скалярный потенциал данного потенциального поля принимает постоянное значение.

Выполнение работы.

Были объявлены переменные, которые задают уравнения областей электродов в двумерном пространстве.

```
conditionExternalElectrode = x^2 + y^2 <= 25;
conditionElectode1 = Abs[-1.8 + x]^2 + 0.8*Abs[y]^2 <= 0.6;
conditionElectode2 = Abs[1.8 + x]^4 + 0.5*Abs[y]^4 <= 0.8;
```

Объявлены переменные, в которых создаются неявно заданные области, которые представляют собой области в двумерном пространстве, определенные на основе уравнений, заданных в переменных, описанных выше.

```
areaExternalElectrode=ImplicitRegion[conditionExternalElectrode,
{x, y}];
areaElectode1 = ImplicitRegion[conditionElectode1 , {x, y}];
areaElectode2  = ImplicitRegion[conditionElectode2 , {x, y}];
```

В итоге, после выполнения этих строк кода, создаются три неявно заданные области.

Для создания графика с контурами электродов были написаны следующие строки кода.

```
diffAreaExternalFirst=RegionDifference[areaExternalElectrode,
areaElectode1 ];
```

Создается новая область с именем *diffAreaExternalFirst* путем выполнения операции разности между областями *areaExternalElectrode* и *areaElectode1*. Фактически, это означает вычитание геометрической области, представленной *areaElectode1*, из геометрической области, представленной *areaExternalElectrode.* В результате, получается область, которая представляет собой оставшуюся часть *areaExternalElectrode* после удаления области *areaElectode1.*

```
fullArea=RegionDifference[diffAreaExternalFirst, areaElectode2 ];
```

В этой строке создается еще одна область с именем *fullArea.* Операция разности *(RegionDifference)* выполняется между областями *diffAreaExternalFirst* и *areaElectode2.* Таким образом, удаляется область areaElectode2 из области *diffAreaExternalFirst.* В результате получается область *fullArea*, которая представляет всю геометрическую область без области *areaElectode1* и *areaElectode2*.

Далее были объявлены переменные, которые определяют уравнения для электродов и значения потенциалов на них.

```
equationExternalElectrode = x^2 + y^2 == 25;
equationElectode1 = Abs[-1.8 + x]^2 + 0.8*Abs[y]^2 == 0.6;
equationElectode2 = Abs[1.8 + x]^4 + 0.5*Abs[y]^4 == 0.8;
\[Phi]External  = 0;
\[Phi]1  = 6;
\[Phi]2 = -5;
```

Для достижения основной цели – нахождение длины эквипотеницали с заданным потенциалом, необходимо определить все эквипотенциали. Для их нахождения необходимо решить дифференциальное уравнение с наложенными условиями Дирихле. При решении уравнения использовался метод сеток, а именно триангуляция.

```
laplaceEquation = Laplacian[u[x, y], {x, y}] == 0;
```

Это уравнение Лапласа, которое говорит о том, что лапласиан (вторая производная) функции *u(x, y)* по *x* и *y* должен быть равен нулю. Это уравнение описывает распределение потенциалов внутри области.

```
conditions = {
   DirichletCondition[u[x, y] == \[Phi]External,
```

```
 equationExternalElectrode ],
DirichletCondition[u[x, y] == \[Phi]1, equationElectode1],
DirichletCondition[u[x, y] == \[Phi]2 , equationElectode2 ]
};
```

В этой строке создается список conditions, который содержит граничные условия для решения уравнения Лапласа. В данном случае, используются условия Дирихле (*DirichletCondition*), где задается значение функции *u(x, y)* на границе каждого из электродов в соответствии с определенными значениями потенциала.

```
numericalSolution = NDSolve[{laplaceEquation, conditions}, u, {x, y} \[Element] fullArea];
```

В этой строке кода решается система уравнений, которая включает в себя уравнение Лапласа (*laplaceEquation*) и граничные условия *(conditions).* Решение численно вычисляется с помощью функции *NDSolve*.

```
plotWithEquipotentials = ContourPlot[
  u[x, y] /. First[numericalSolution ],
  {x, y} \[Element] fullArea,
  Contours -> 30,
  ColorFunction -> "TemperatureMap",
  PlotLegends -> Automatic
  ]
```

Переменная *plotWithEquipotentials* содержит графическое представление потенциального поля с равными потенциалами (контурами). *u[x, y] /. First[numericalSolution]* извлечение решение потенциального поля *u[x, y]* из переменной *numericalSolution. First[numericalSolution]* используется для извлечения первого решения из возможных решений, если их несколько. *{x, y} \[Element] fullArea* указывает, что график будет построен для переменных *x* и *y*, принадлежащих области *fullArea*, которая была определена ранее.

```
plotWithFindEquipotential = ContourPlot[
   Evaluate[u[x, y] /. numericalSolution ] == \[Phi]Equipotential,
   {x, y} \[Element] fullArea,
   Contours -> {\[Phi]Equipotential},
   PlotLegends -> Automatic,
   ContourStyle -> Green
   ];
```

Этот код создает графическое представление линий равного потенциала внутри области *fullArea*, где *\[Phi]Equipotential* — это заданное значение потенциала, для которого находятся соответствующие линии равного потенциала. *Evaluate[u[x, y] /. numericalSolution] == \[Phi]Equipotential* уравнение сравнивает потенциал, вычисленный с помощью решения *numericalSolution (выражение u[x, y] /. numericalSolution),* с заданным значением *\(\[Phi]Equipotential\).*

Чтобы найти значение длины эквипотенциали, описанной в графике выше, необходимо разбить линию эквипотенциали на бесконечное число точек, и просуммировать эвклидово расстояние между ними.

```
points = Cases[Normal@plotWithFindEquipotential, Line[points_] :>
points, Infinity];
```

В этой строке кода переменной *points* присваивается список координат точек, которые представляют собой линии равного потенциала на графике *plotWithFindEquipotential*. Затем применяется функция *Cases*, которая ищет в графическом объекте *plotWithFindEquipotential* все элементы, которые соответствуют шаблону *Line[points_]* и извлекает их содержимое *points.*

```
pointsPairs = Flatten[points, 1];
```

Здесь создается переменная *pointsPairs,* которая содержит плоский список координат точек. *Flatten[points, 1]* преобразует двумерный список в одномерный список координат точек.

```
For[index = 1, index <= Length[pointsPairs] - 1, index++,
  equipotentialsLength +=
   EuclideanDistance[pointsPairs[[index]],pointsPairs[[index+1]]]
  ];
```

Внутри цикла вычисляется расстояние между текущей точкой *(pointsPairs[[index]])* и следующей точкой *(pointsPairs[[index + 1]])* с помощью функции *EuclideanDistance*. Затем это расстояние прибавляется к переменной *equipotentialsLength.*

Разработанный программный код смотри в приложении А.

Тестирование.

На рисунках 1 – 4 представлены результат работы программы.

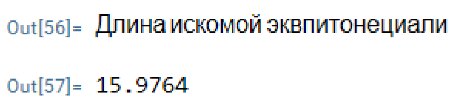

Рисунок 1 – значение длины эквипотенциали.

На рисунке 2 представлены контуры электродов, построенные по уравнениям, соответствующим варианту.

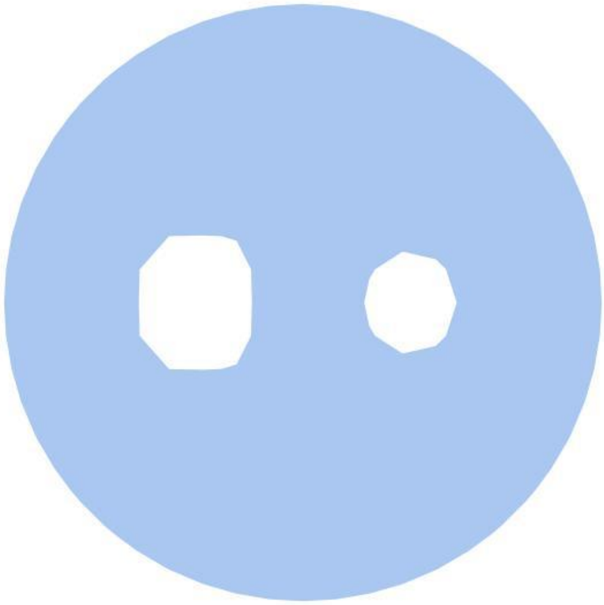

Рисунок 2 – контуры электродов.

На рисунке 3 представлены контуры электродов и соответствующих эквипотенциалей в области.

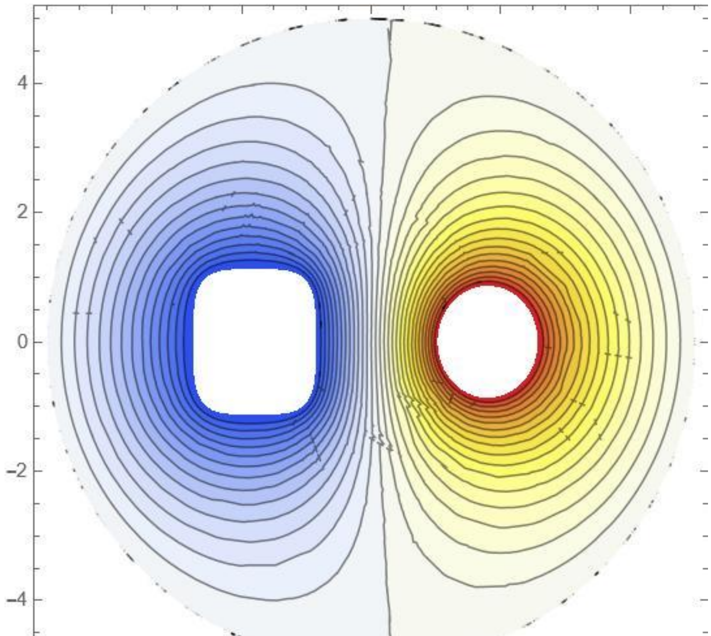

Рисунок 3 – контуры электродов с эквипотенциалями.

На рисунке 4 представлены контуры электродов и искомой эквипотенциали в области, которая обозначена зелёным цветом.

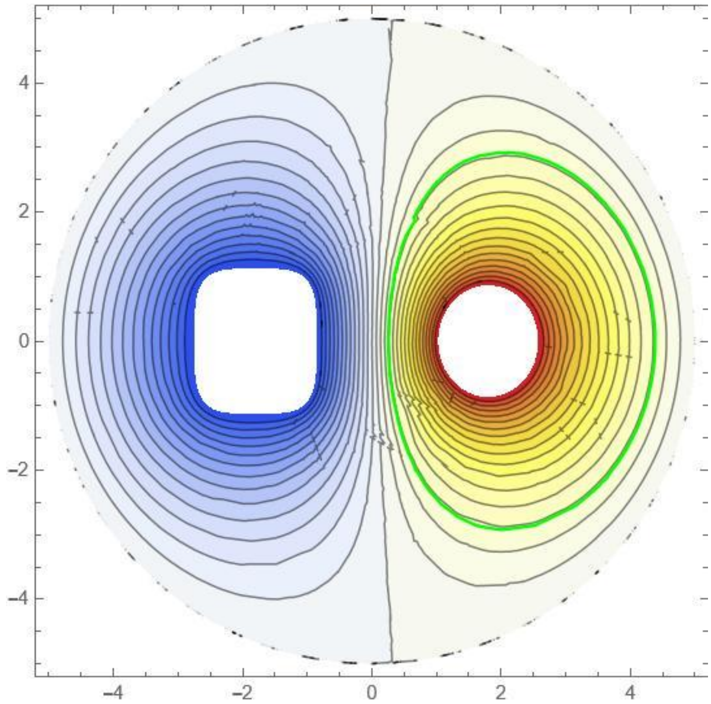

Рисунок 4 – контуры электродов с искомой эквипотенциалью.

Выводы.

В ходе лабораторной работы написана программа, которая вычисляет длину для указанной в таблице эквипотенциальной линии.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: IDZ2.nb

```
conditionExternalElectrode = x^2 + y^2 <= 25;
conditionElectode1 = Abs[-1.8 + x]^2 + 0.8*Abs[y]^2 <= 0.6;
conditionElectode2 = Abs[1.8 + x]^4 + 0.5*Abs[y]^4 <= 0.8;
areaExternalElectrode  =
  ImplicitRegion[conditionExternalElectrode, {x, y}];
areaElectode1 = ImplicitRegion[conditionElectode1 , {x, y}];
areaElectode2  = ImplicitRegion[conditionElectode2 , {x, y}];
diffAreaExternalFirst =
  RegionDifference[areaExternalElectrode, areaElectode1 ];
fullArea  = RegionDifference[diffAreaExternalFirst,  areaElectode2
];
Region[fullArea]
equationExternalElectrode = x^2 + y^2 == 25;
equationElectode1 = Abs[-1.8 + x]^2 + 0.8*Abs[y]^2 == 0.6;
equationElectode2 = Abs[1.8 + x]^4 + 0.5*Abs[y]^4 == 0.8;
\[Phi]External  = 0;
\[Phi]1  = 6;
\[Phi]2 = -5;
laplaceEquation = Laplacian[u[x, y], {x, y}] == 0;
conditions = {
   DirichletCondition[u[x, y] == \[Phi]External,
    equationExternalElectrode ],
   DirichletCondition[u[x, y] == \[Phi]1, equationElectode1],
   DirichletCondition[u[x, y] == \[Phi]2 , equationElectode2 ]
   };
numericalSolution =
  NDSolve[{laplaceEquation, conditions},
   u, {x, y} \[Element] fullArea];
plotWithEquipotentials = ContourPlot[
  u[x, y] /. First[numericalSolution ],
  {x, y} \[Element] fullArea,
  Contours -> 30,
  ColorFunction -> "TemperatureMap",
  PlotLegends -> Automatic
  ]
\[Phi]Equipotential = 1;
```

```
plotWithFindEquipotential = ContourPlot[
   Evaluate[u[x, y] /. numericalSolution ] == \[Phi]Equipotential,
   {x, y} \[Element] fullArea,
   Contours -> {\[Phi]Equipotential},
   PlotLegends -> Automatic,
   ContourStyle -> Green
   ];
Show[plotWithEquipotentials, plotWithFindEquipotential ]
points =
  Cases[Normal@plotWithFindEquipotential, Line[points_] :> points,
   Infinity];
pointsPairs = Flatten[points, 1];
equipotentialsLength = 0;
For[index = 1, index <= Length[pointsPairs] - 1, index++,
  equipotentialsLength +=
   EuclideanDistance[pointsPairs[[index]],  pointsPairs[[index  +
1]]]
  ];
Text[Длина искомой эквпитонециали]
equipotentialsLength
```