**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по лабораторной работе №5**

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

**Тема: Триангуляция пространства**

**Вариант 2**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Альтмарк А.М. |

Санкт-Петербург

2018

**Цель работы**

Изучить триангуляцию двухмерного пространства и ее программную реализацию, и визуализацию.

**Основные теоретические положения**

**Триангуляция Делоне** — триангуляция для заданного множества точек *S* на плоскости, при которой для любого треугольника все точки из *S* за исключением точек, являющихся его вершинами, лежат вне окружности, описанной вокруг треугольника.

Данный алгоритм основан на стандартной для многих алгоритмов методике сведения сложной задачи к более простым, в которых решение очевидно. Сам алгоритм для *N* > 1 состоит из 2 шагов:

1. Разбиение исходного множества на более мелкие множества. Для этого мы проводим вертикальные или горизонтальные прямые в середине множества и уже относительно этих прямых разделяем точки на две части примерно по *N*/2. После для каждой группы точек рекурсивно запускаем процесс деления.
2. Объединение оптимальных триангуляций. Сначала находятся две пары точек, отрезки которых образуют в совокупности с построенными триангуляциями выпуклую фигуру. Они соединяются отрезками, и один из полученных отрезков выбирается как начало для последующего обхода. Обход заключается в следующем: на этом отрезке мы как будто «надуваем пузырь» внутрь до первой точки, которую достигнет раздувающаяся окружность «пузыря». С найденной точкой соединяется та точка отрезка, которая не была с ней соединена. Полученный отрезок проверяется на пересечение с уже существующими отрезками триангуляции, и в случае пересечения они удаляются из триангуляции. После этого новый отрезок принимается за начало для нового «пузыря». Цикл повторяется до тех пор, пока начало не совпадёт со вторым отрезком выпуклой оболочки.

**Задание**

Провести триангуляцию пространства, ограниченного двумя электродами (синий и красный).

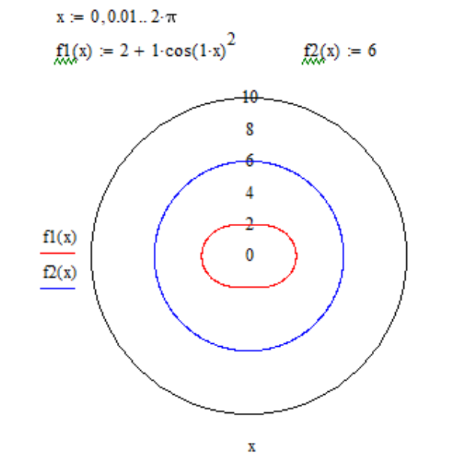


Рисунок 1 – Вариант задания

**Ход работы**

1. Определим функции, которые возвращают для отрисовки фигур
2. Зададим точки фигур в декартовой системе координат отдельно для триангуляции и отдельно для визуализации фигур
3. Проведём триангуляцию
4. Зададим маску для скрытия ненужных треугольников в красной фигуре

Исходный код программы находится в приложении А, а результаты триангуляции в приложении Б.

**Выводы**

В ходе выполнения данной лабораторной работы была произведена триангуляция двумерного пространства, ограниченного двумя кривыми.

**ПРИЛОЖЕНИЕ А**

**Код программы**

# coding: utf-8  
  
import numpy as np  
from matplotlib import pyplot as plt  
from matplotlib import tri  
  
  
def f1(x):  
 return 2 + np.cos(x) \*\* 2  
  
  
# точки для просчета углов  
fig\_angles = np.linspace(0, 2 \* np.pi, 50)  
  
r1 = 1.5  
r2 = 2.25  
r3 = 3  
  
# считаем точки вершин треугольников фигур  
inner\_fig = np.array([(r \* np.cos(alpha), r \* np.sin(alpha)) for alpha, r in zip(fig\_angles,  
 f1(fig\_angles))])  
outer\_fig = np.array([(6 \* np.cos(alpha), 6 \* np.sin(alpha)) for alpha in fig\_angles])  
  
inner\_ring\_1 = np.array([(r \* np.cos(alpha), r \* np.sin(alpha)) for alpha, r in zip(fig\_angles,  
 f1(fig\_angles) + r1)])  
inner\_ring\_2 = np.array([(r \* np.cos(alpha), r \* np.sin(alpha)) for alpha, r in zip(fig\_angles,  
 f1(fig\_angles) + r2)])  
inner\_ring\_3 = np.array([(r \* np.cos(alpha), r \* np.sin(alpha)) for alpha, r in zip(fig\_angles,  
 f1(fig\_angles) + r3)])  
  
  
# точки для построения высокоточных фигур  
fig\_draw\_angles = np.linspace(0, 2 \* np.pi, 2000)  
  
# рисуем фигуры  
inner\_fig\_draw = np.array([(r \* np.cos(alpha), r \* np.sin(alpha)) for alpha, r in zip(fig\_draw\_angles,  
 f1(fig\_draw\_angles))])  
outer\_fig\_draw = np.array([(6 \* np.cos(alpha), 6 \* np.sin(alpha)) for alpha in fig\_draw\_angles])  
  
# объединяем массивы фигур по x и y  
X = np.hstack((inner\_fig[:, 0], outer\_fig[:, 0], inner\_ring\_1[:, 0], inner\_ring\_2[:, 0],  
 inner\_ring\_3[:, 0]))  
Y = np.hstack((inner\_fig[:, 1], outer\_fig[:, 1], inner\_ring\_1[:, 1], inner\_ring\_2[:, 1],  
 inner\_ring\_3[:, 1]))  
  
triangulation = tri.Triangulation(X, Y)  
  
# объединяем координаты центров треугольников по x и y  
x\_mid = X[triangulation.triangles].mean(axis=1)  
y\_mid = Y[triangulation.triangles].mean(axis=1)  
  
# первод в полярные координаты  
R = (x\_mid \*\* 2 + y\_mid \*\* 2) \*\* 0.5  
alpha = np.arctan(y\_mid / x\_mid)  
  
# маска для скрытия треугольников внутри красной фигуры  
mask = np.where(2 + np.cos(2 \* alpha \* np.pi / 180) \*\* 2 >= R, True, False)  
triangulation.set\_mask(mask)  
  
plt.figure(figsize=(10, 10))  
plt.axes().set\_aspect('equal')  
  
plt.plot(inner\_fig\_draw[:, 0], inner\_fig\_draw[:, 1], c='red', lw=1.5)  
plt.plot(outer\_fig\_draw[:, 0], outer\_fig\_draw[:, 1], c='blue', lw=1.5)  
plt.title('Трианугляция пространства', fontsize=20)  
plt.triplot(triangulation, c='black', lw=0.5)  
plt.show()

**ПРИЛОЖЕНИЕ Б**

**Триангуляция**

