МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ

Факультет прикладной
математики и информатики

Кафедра теоретической и прикладной информатики

Практическое задание № 1

по дисциплине «Компьютерные технологии моделирования и анализа данных»

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ ДЛЯ РАБОТЫ С КОНЕЧНОЭЛЕМЕНТНЫМИ СЕТКАМИ

Вариант 1 ПММ-52 КУСАКИН АЛЕКСАНДР
ПММ-52 ЦИРКОВА АЛИНА
ПММ-52 БОРИСОВ ДМИТРИЙ

Преподаватели КОШКИНА ЮЛИЯ ИГОРЕВНА

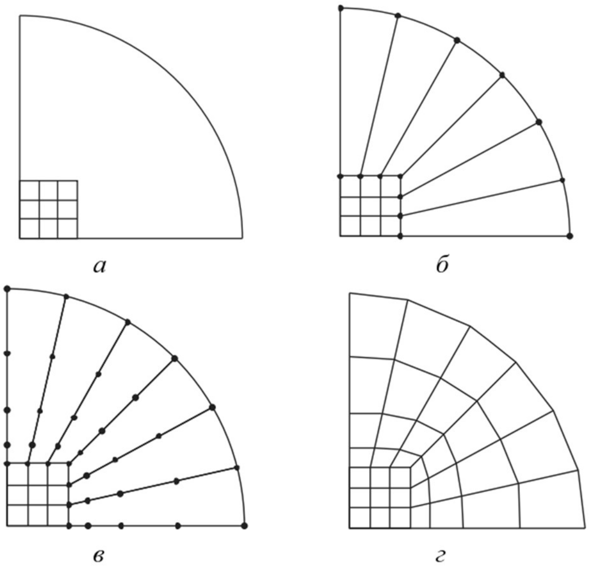
Новосибирск, 2025

Цель работы

Изучить структуры данных метода конечных элементов, способы хранения информации о сетке. Научиться разрабатывать алгоритмы построения конечноэлементных сеток.

Задание

Построить сетку из четырехугольников в расчетной области, изображенной на рисунке.



Алгоритм

1. Задание квадрата с обычной прямоугольной сеткой, лежащего в квадранте
2. На дуге окружности расставляется столько же узлов, сколько принадлежит двум сторонам квадрата, лежащим внутри квадранта
3. Соответствующие узлы соединяются отрезками
4. На отрезках, соединяющих узлы квадрата с узлами на дуге окружности, с некоторым коэффициентом разрядки расставляется заданное число узлов
5. Последовательно соединить соответствующие узлы отрезков

Формат входных данных

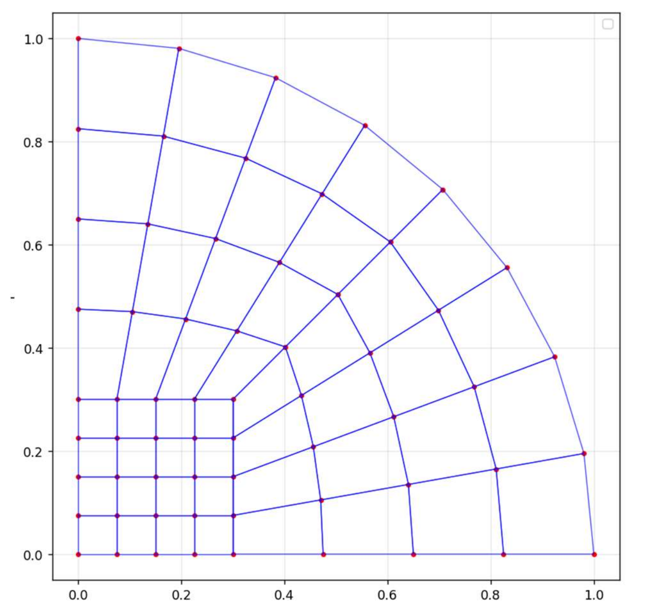
```
radius <значение> square_size <значение>
Kx Ky
<координаты узлов 1-й линии>
<координаты узлов 2-й линии>
...
<координаты узлов Ky-й линии>
<num_subregions>
<описание каждой подобласти>
```

Результат работы программы

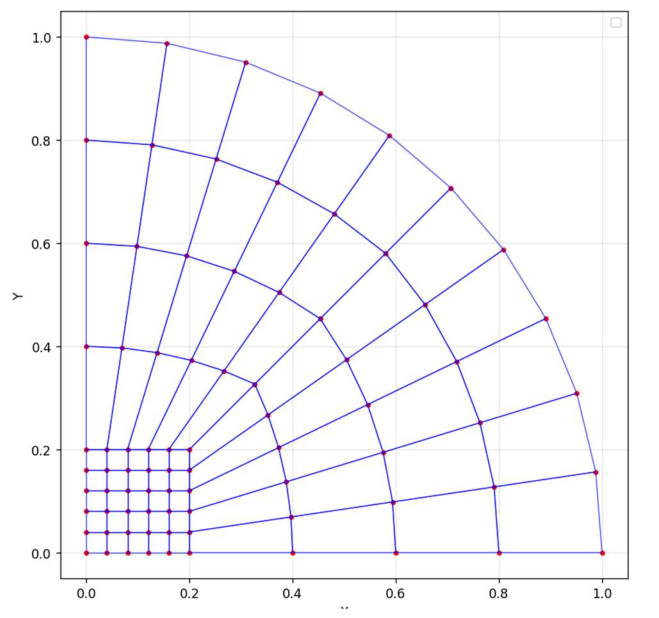
Коэффициент разрядки = 1.0

Количество узлов = 5

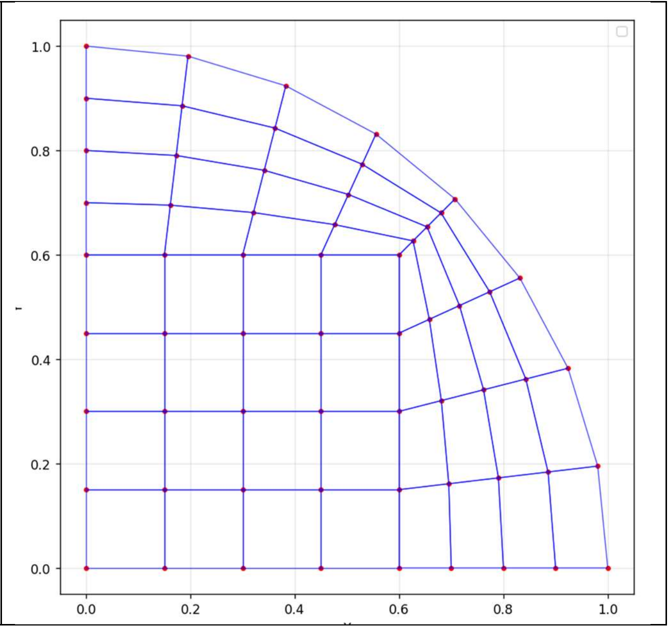
```
radius 1.0 square_size 0.3
5 5
0.0 0.0    0.075 0.0    0.15 0.0    0.225 0.0    0.3 0.0
0.0 0.075  0.075 0.075  0.15 0.075  0.225 0.075  0.3 0.075
0.0 0.15    0.075 0.15    0.15 0.15    0.225 0.15    0.3 0.15
0.0 0.225  0.075 0.225  0.15 0.225  0.225 0.225  0.3 0.225
0.0 0.3    0.075 0.3    0.15 0.3    0.225 0.3    0.3 0.3
1
1 1 4 1 4
```



```
radius 1.0 square_size 0.2
6 6
0.0 0.0   0.04 0.0   0.08 0.0   0.12 0.0   0.16 0.0   0.2 0.0
0.0 0.04  0.04 0.04  0.08 0.04  0.12 0.04  0.16 0.04  0.2 0.04
0.0 0.08  0.04 0.08  0.08 0.08  0.12 0.08  0.16 0.08  0.2 0.08
0.0 0.12  0.04 0.12  0.08 0.12  0.12 0.12  0.16 0.12  0.2 0.12
0.0 0.16  0.04 0.16  0.08 0.16  0.12 0.16  0.16 0.16  0.2 0.16
0.0 0.2   0.04 0.2   0.08 0.2   0.12 0.2   0.16 0.2   0.2 0.2
1
1 1 5 1 5
```

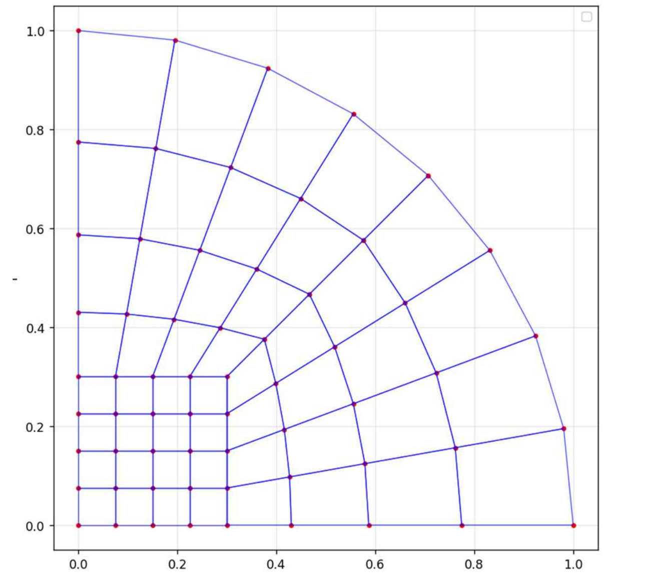


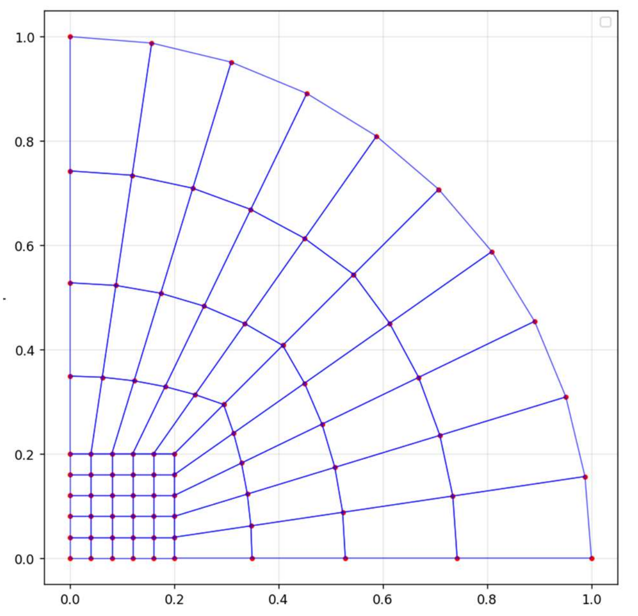
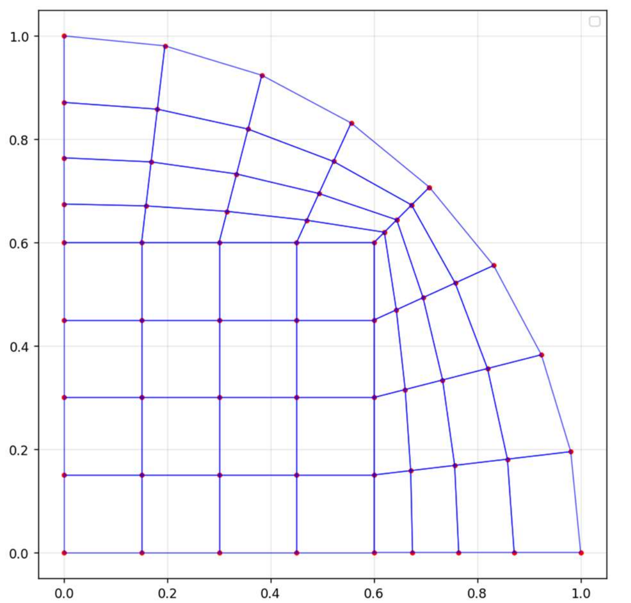
```
radius 1.0 square_size 0.6
5 5
0.0 0.0   0.15 0.0   0.3 0.0   0.45 0.0   0.6 0.0
0.0 0.15   0.15 0.15   0.3 0.15   0.45 0.15   0.6
0.15
0.0 0.3   0.15 0.3   0.3 0.3   0.45 0.3   0.6 0.3
0.0 0.45   0.15 0.45   0.3 0.45   0.45 0.45   0.6
0.45
0.0 0.6   0.15 0.6   0.3 0.6   0.45 0.6   0.6 0.6
1
1 1 4 1 4
```



Коэффициент разрядки = 1.2
Количество узлов = 5

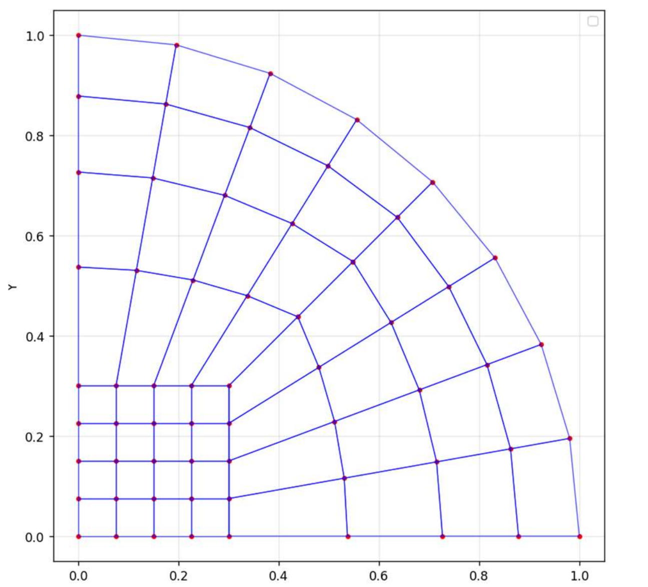
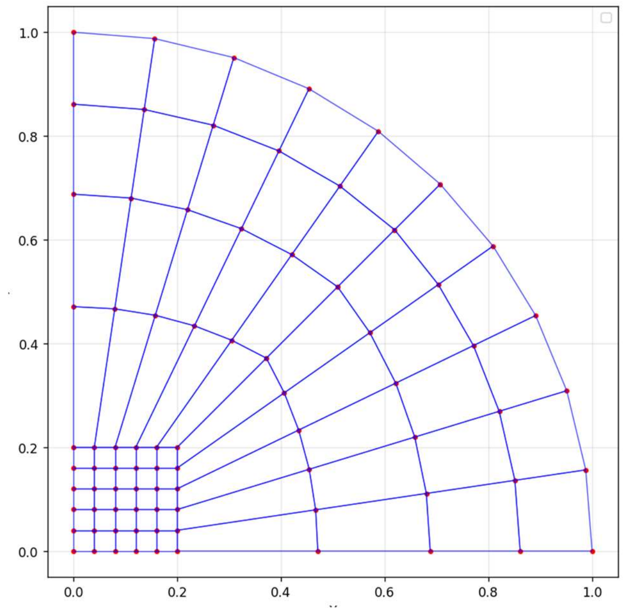
```
radius 1.0 square_size 0.3
5 5
0.0 0.0   0.075 0.0   0.15 0.0   0.225 0.0   0.3 0.0
0.0 0.075  0.075 0.075  0.15 0.075  0.225 0.075
0.3 0.075
0.0 0.15    0.075 0.15    0.15 0.15    0.225 0.15
0.3 0.15
0.0 0.225  0.075 0.225  0.15 0.225  0.225 0.225
0.3 0.225
0.0 0.3    0.075 0.3    0.15 0.3    0.225 0.3    0.3
0.3
1
1 1 4 1 4
```



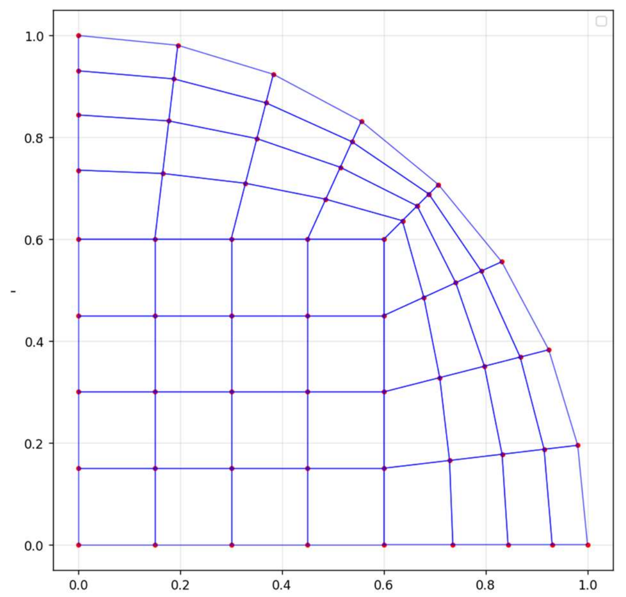
radius 1.0 square_size 0.2 6 6 0.0 0.0 0.04 0.0 0.08 0.0 0.12 0.0 0.16 0.0 0.2 0.0 0.0 0.04 0.04 0.04 0.08 0.04 0.12 0.04 0.16 0.04 0.2 0.04 0.0 0.08 0.04 0.08 0.08 0.08 0.12 0.08 0.16 0.08 0.2 0.08 0.0 0.12 0.04 0.12 0.08 0.12 0.12 0.12 0.16 0.12 0.2 0.12 0.0 0.16 0.04 0.16 0.08 0.16 0.12 0.16 0.16 0.16 0.2 0.16 0.0 0.2 0.04 0.2 0.08 0.2 0.12 0.2 0.16 0.2 0.2 0.2 1 1 1 5 1 5	 1.0 0.8 0.6 0.4 0.2 0.0 0.0 0.2 0.4 0.6 0.8 1.0
radius 1.0 square_size 0.6 5 5 0.0 0.0 0.15 0.0 0.3 0.0 0.45 0.0 0.6 0.0 0.0 0.15 0.15 0.15 0.3 0.15 0.45 0.15 0.6 0.15 0.0 0.3 0.15 0.3 0.3 0.3 0.45 0.3 0.6 0.3 0.0 0.45 0.15 0.45 0.3 0.45 0.45 0.45 0.6 0.45 0.0 0.6 0.15 0.6 0.3 0.6 0.45 0.6 0.6 0.6 1 1 1 4 1 4	 1.0 0.8 0.6 0.4 0.2 0.0 0.0 0.2 0.4 0.6 0.8 1.0

Коэффициент разрядки = 0.8

Количество узлов = 5

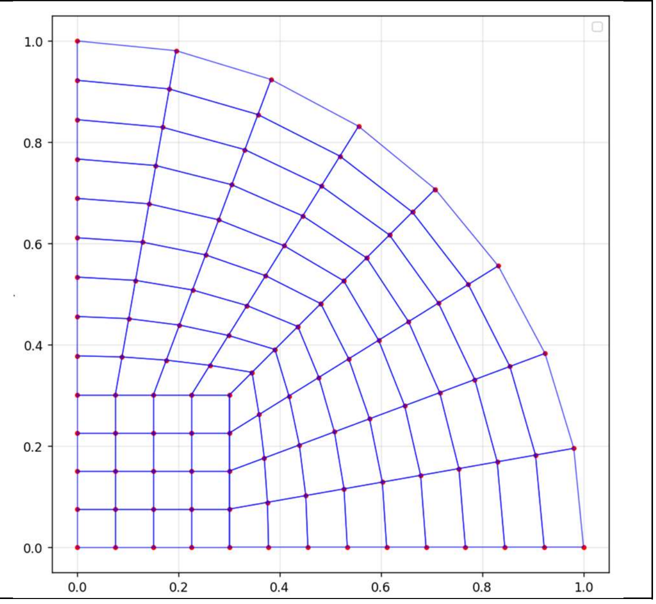
radius 1.0 square_size 0.3 5 5 0.0 0.0 0.075 0.0 0.15 0.0 0.225 0.0 0.3 0.0 0.0 0.075 0.075 0.075 0.15 0.075 0.225 0.075 0.3 0.075 0.0 0.15 0.075 0.15 0.15 0.15 0.225 0.15 0.3 0.15 0.0 0.225 0.075 0.225 0.15 0.225 0.225 0.225 0.3 0.225 0.0 0.3 0.075 0.3 0.15 0.3 0.225 0.3 0.3 0.3 1 1 1 4 1 4	 1.0 0.8 0.6 0.4 0.2 0.0 Y 0.0 0.2 0.4 0.6 0.8 1.0
radius 1.0 square_size 0.2 6 6 0.0 0.0 0.04 0.0 0.08 0.0 0.12 0.0 0.16 0.0 0.2 0.0 0.0 0.04 0.04 0.04 0.08 0.04 0.12 0.04 0.16 0.04 0.2 0.04 0.0 0.08 0.04 0.08 0.08 0.08 0.12 0.08 0.16 0.08 0.2 0.08 0.0 0.12 0.04 0.12 0.08 0.12 0.12 0.12 0.16 0.12 0.2 0.12 0.0 0.16 0.04 0.16 0.08 0.16 0.12 0.16 0.16 0.16 0.2 0.16 0.0 0.2 0.04 0.2 0.08 0.2 0.12 0.2 0.16 0.2 0.2 0.2 1 1 1 5 1 5	 1.0 0.8 0.6 0.4 0.2 0.0 0.0 0.2 0.4 0.6 0.8 1.0

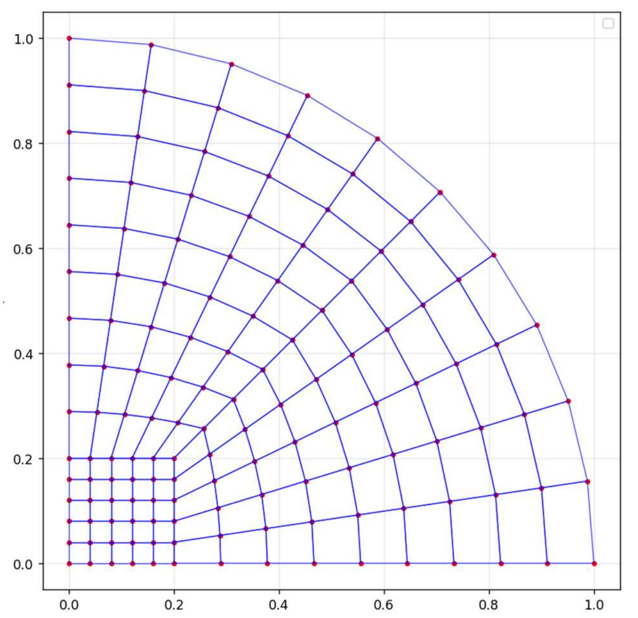
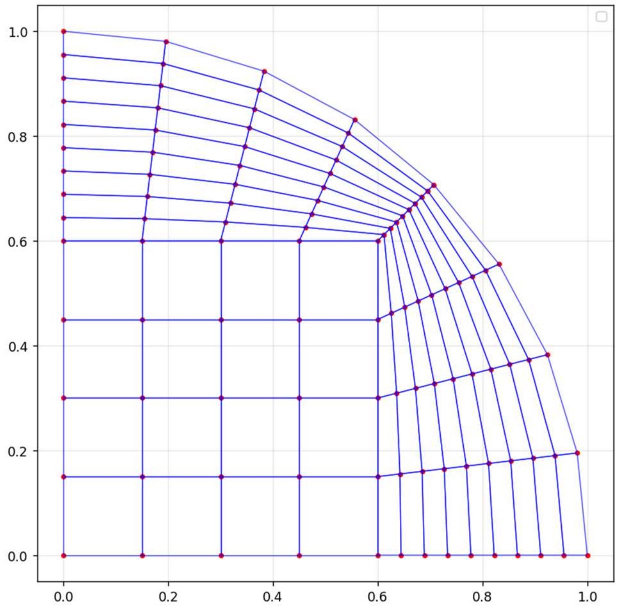
```
radius 1.0 square_size 0.6
5 5
0.0 0.0   0.15 0.0   0.3 0.0   0.45 0.0   0.6 0.0
0.0 0.15  0.15 0.15  0.3 0.15  0.45 0.15  0.6 0.15
0.0 0.3   0.15 0.3   0.3 0.3   0.45 0.3   0.6 0.3
0.0 0.45  0.15 0.45  0.3 0.45  0.45 0.45  0.6 0.45
0.0 0.6   0.15 0.6   0.3 0.6   0.45 0.6   0.6 0.6
1
1 1 4 1 4
```



Коэффициент разрядки = 1.0
Количество узлов = 10

```
radius 1.0 square_size 0.3
5 5
0.0 0.0   0.075 0.0   0.15 0.0   0.225 0.0   0.3 0.0
0.0 0.075  0.075 0.075  0.15 0.075  0.225 0.075
0.3 0.075
0.0 0.15    0.075 0.15    0.15 0.15    0.225 0.15
0.3 0.15
0.0 0.225  0.075 0.225  0.15 0.225  0.225 0.225
0.3 0.225
0.0 0.3    0.075 0.3    0.15 0.3    0.225 0.3    0.3
0.3
1
1 1 4 1 4
```



radius 1.0 square_size 0.2 6 6 0.0 0.0 0.04 0.0 0.08 0.0 0.12 0.0 0.16 0.0 0.2 0.0 0.0 0.04 0.04 0.04 0.08 0.04 0.12 0.04 0.16 0.04 0.2 0.04 0.0 0.08 0.04 0.08 0.08 0.08 0.12 0.08 0.16 0.08 0.2 0.08 0.0 0.12 0.04 0.12 0.08 0.12 0.12 0.12 0.16 0.12 0.2 0.12 0.0 0.16 0.04 0.16 0.08 0.16 0.12 0.16 0.16 0.16 0.2 0.16 0.0 0.2 0.04 0.2 0.08 0.2 0.12 0.2 0.16 0.2 0.2 0.2 1 1 1 5 1 5	
radius 1.0 square_size 0.6 5 5 0.0 0.0 0.15 0.0 0.3 0.0 0.45 0.0 0.6 0.0 0.0 0.15 0.15 0.15 0.3 0.15 0.45 0.15 0.6 0.15 0.0 0.3 0.15 0.3 0.3 0.3 0.45 0.3 0.6 0.3 0.0 0.45 0.15 0.45 0.3 0.45 0.45 0.45 0.6 0.45 0.0 0.6 0.15 0.6 0.3 0.6 0.45 0.6 0.6 0.6 1 1 1 4 1 4	

Код программы

```
import matplotlib.pyplot as plt
import math
import numpy as np

class MeshData:
    def __init__(self):
        self.radius = 1.0
        self.square_size = 0.7
        self.Kx = 0
        self.Ky = 0
        self.coord_lines = []
        self.subregions = []
        self.nodes = []
        self.elements = []
        self.n_radial_div = 5
        self.spacing = 1.0

class Subregion:
    def __init__(self):
        self.material = 0
        self.nxb = self.nxe = 0
        self.nyb = self.nye = 0

def read_area_file(filename):
    mesh = MeshData()

    with open(filename, 'r') as f:
        lines = [line.strip() for line in f if line.strip() and not line.strip().startswith('#')]

    current_line = 0

    if 'radius' in lines[current_line]:
        params = lines[current_line].split()
        for i in range(0, len(params), 2):
            if params[i] == 'radius':
                mesh.radius = float(params[i+1])
            elif params[i] == 'square_size':
                mesh.square_size = float(params[i+1])
        current_line += 1

    mesh.Kx, mesh.Ky = map(int, lines[current_line].split())
    current_line += 1

    mesh.coord_lines = []
    for i in range(mesh.Ky):
        coords = list(map(float, lines[current_line].split()))
        line_points = []
        for j in range(0, len(coords), 2):
            line_points.append([coords[j], coords[j+1]])
        mesh.coord_lines.append(line_points)
        current_line += 1

    num_subregions = int(lines[current_line])
    current_line += 1

    mesh.subregions = []
    for i in range(num_subregions):
        data = list(map(int, lines[current_line].split()))
        subregion = Subregion()
        subregion.material = data[0]
        subregion.nxb, subregion.nxe = data[1], data[2]
        subregion.nyb, subregion.nye = data[3], data[4]
        mesh.subregions.append(subregion)
        current_line += 1

    return mesh

def interpolate_points(p1, p2, n_div, q):

    x1, y1 = p1
    x2, y2 = p2
    pts = []
    if n_div < 2:
        return [p1, p2]

    if abs(q - 1.0) < 1e-6:
        for i in range(n_div):
            t = i / (n_div - 1)
            x = x1 + (x2 - x1) * t
            y = y1 + (y2 - y1) * t
            pts.append([x, y])
    else:
        total = (q ** (n_div - 1) - 1) / (q - 1)
        for i in range(n_div):
            l = (q ** i - 1) / (q - 1)
            t = l / total
            x = x1 + (x2 - x1) * t
            y = y1 + (y2 - y1) * t
            pts.append([x, y])
    return pts

def generate_radial_mesh(mesh):
    all_nodes = []
    elements = []
```

```
square_nodes = []
for i in range(mesh.Ky):
    for j in range(mesh.Kx):
        square_nodes.append(mesh.coord_lines[i][j])

all_nodes.extend(square_nodes)
square_start_idx = 0
square_count = len(square_nodes)

arc_nodes = []

right_square_nodes = [mesh.coord_lines[i][-1] for i in range(mesh.Ky)]
for i, node in enumerate(right_square_nodes):
    y = node[1]
    ratio = y / mesh.square_size
    angle = (math.pi / 4) * ratio  # от 0 до π/4
    x_arc = mesh.radius * math.cos(angle)
    y_arc = mesh.radius * math.sin(angle)
    arc_nodes.append([x_arc, y_arc])

top_square_nodes = mesh.coord_lines[-1]
for j, node in enumerate(top_square_nodes):
    x = node[0]
    ratio = 1 - x / mesh.square_size
    angle = (math.pi / 4) + (math.pi / 4) * ratio
    x_arc = mesh.radius * math.cos(angle)
    y_arc = mesh.radius * math.sin(angle)
    arc_nodes.append([x_arc, y_arc])

arc_start_index = len(all_nodes)
all_nodes.extend(arc_nodes)

intermediate_layers = []
total_layers = mesh.n_radial_div - 2

for layer_idx in range(1, mesh.n_radial_div - 1):
    current_layer = []
    t = layer_idx / (mesh.n_radial_div - 1)

    if mesh.spacing != 1.0:
        total = (mesh.spacing ** (mesh.n_radial_div - 1) - 1) / (mesh.spacing - 1)
        l = (mesh.spacing ** layer_idx - 1) / (mesh.spacing - 1)
        t = l / total

    for i, square_node in enumerate(right_square_nodes):
        arc_node = arc_nodes[i]
        x = square_node[0] + (arc_node[0] - square_node[0]) * t
        y = square_node[1] + (arc_node[1] - square_node[1]) * t
        current_layer.append([x, y])
    for j, square_node in enumerate(top_square_nodes):
        arc_node = arc_nodes[len(right_square_nodes) + j]
        x = square_node[0] + (arc_node[0] - square_node[0]) * t
        y = square_node[1] + (arc_node[1] - square_node[1]) * t
        current_layer.append([x, y])

    intermediate_layers.append(current_layer)

intermediate_start_indices = []
for layer in intermediate_layers:
    intermediate_start_indices.append(len(all_nodes))
    all_nodes.extend(layer)

for i in range(mesh.Ky - 1):
    for j in range(mesh.Kx - 1):
        n1 = i * mesh.Kx + j
        n2 = i * mesh.Kx + j + 1
        n3 = (i + 1) * mesh.Kx + j + 1
        n4 = (i + 1) * mesh.Kx + j
        elements.append([n1, n2, n3, n4, 1])

square_right_indices = [i * mesh.Kx + (mesh.Kx - 1) for i in range(mesh.Ky)]

arc_right_indices = [arc_start_index + i for i in range(len(right_square_nodes))]

for i in range(len(right_square_nodes) - 1):
    for layer in range(mesh.n_radial_div - 1):
        if layer == 0:
            n1 = square_right_indices[i]
            n2 = square_right_indices[i + 1]
            if total_layers > 0:
                n3 = intermediate_start_indices[0] + i + 1
                n4 = intermediate_start_indices[0] + i
            else:
                n3 = arc_right_indices[i + 1]
                n4 = arc_right_indices[i]
            elements.append([n1, n2, n3, n4, 2])

        elif layer == mesh.n_radial_div - 2:
            n1 = intermediate_start_indices[layer - 1] + i
            n2 = intermediate_start_indices[layer - 1] + i + 1
            n3 = arc_right_indices[i + 1]
            n4 = arc_right_indices[i]
            elements.append([n1, n2, n3, n4, 2])

        else:
            n1 = intermediate_start_indices[layer - 1] + i
            n2 = intermediate_start_indices[layer - 1] + i + 1
            n3 = intermediate_start_indices[layer] + i + 1
            n4 = intermediate_start_indices[layer] + i
            elements.append([n1, n2, n3, n4, 2])

square_top_indices = [(mesh.Ky - 1) * mesh.Kx + j for j in range(mesh.Kx)]

arc_top_start = arc_start_index + len(right_square_nodes)
arc_top_indices = [arc_top_start + j for j in range(len(top_square_nodes))]

top_offset = len(right_square_nodes)

for j in range(len(top_square_nodes) - 1):
    for layer in range(mesh.n_radial_div - 1):
        if layer == 0:
            n1 = square_top_indices[j]
            n2 = square_top_indices[j + 1]
            if total_layers > 0:
                n3 = intermediate_start_indices[0] + top_offset + j + 1
                n4 = intermediate_start_indices[0] + top_offset + j
            else:
                n3 = arc_top_indices[j + 1]
                n4 = arc_top_indices[j]
            elements.append([n1, n2, n3, n4, 2])

        elif layer == mesh.n_radial_div - 2:
            n1 = intermediate_start_indices[layer - 1] + top_offset + j
            n2 = intermediate_start_indices[layer - 1] + top_offset + j + 1
            n3 = arc_top_indices[j + 1]
            n4 = arc_top_indices[j]
            elements.append([n1, n2, n3, n4, 2])

        else:
            n1 = intermediate_start_indices[layer - 1] + top_offset + j
            n2 = intermediate_start_indices[layer - 1] + top_offset + j + 1
            n3 = intermediate_start_indices[layer] + top_offset + j + 1
            n4 = intermediate_start_indices[layer] + top_offset + j
            elements.append([n1, n2, n3, n4, 2])

mesh.nodes = all_nodes
mesh.elements = elements
return mesh
def visualize_radial_mesh(mesh):
plt.figure(figsize=(8, 10))

x_nodes = [node[0] for node in mesh.nodes]
y_nodes = [node[1] for node in mesh.nodes]
plt.plot(x_nodes, y_nodes, 'ro', markersize=3)

for element in mesh.elements:
    polygon = element[:4] + [element[0]]
    x = [mesh.nodes[i][0] for i in polygon]
    y = [mesh.nodes[i][1] for i in polygon]
    plt.plot(x, y, 'b-', linewidth=0.8, alpha=0.7)

plt.axis('equal')
plt.grid(True, alpha=0.3)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.show()

def save_mesh_to_file(mesh, filename):
with open(filename, 'w') as f:
    f.write("NODES\n")
    f.write(f"{len(mesh.nodes)}\n")
    for i, node in enumerate(mesh.nodes):
        f.write(f"{i} {node[0]:.6f} {node[1]:.6f}\n")

    f.write("\nELEMENTS\n")
    f.write(f"{len(mesh.elements)}\n")
    for i, element in enumerate(mesh.elements):
        nodes_str = " ".join(str(n) for n in element[:4])
        f.write(f"{i} {nodes_str} {element[4]}\n")

if __name__ == "__main__":
mesh = read_area_file("computational_domain.txt")
mesh = generate_radial_mesh(mesh)
save_mesh_to_file(mesh, "radial_mesh.txt")
visualize_radial_mesh(mesh)
```