Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники



Вариант **№6**

Расчетно-графическая работа **№6**

По дисциплине

**Теория функций комплексной переменной**

*Выполнили студенты*:

Хромов Даниил Тимофеевич 373336

Кузьмина Ольга Игоревна 412986

Садовой ГРИГОРИЙ Владимирович 368748

*Преподаватель*:

Ким Эрик Евгеньевич

Санкт-Петербург 2023 г.

1. **Текст задания**

В соответствии с вариантом возьмите два рисунка. На них изображены множества, между которыми следует построить некоторое конформное отображение. Множе ство на рисунке закрашено штриховкой, а все граничные точки подразумеваются не принадлежащими ему.

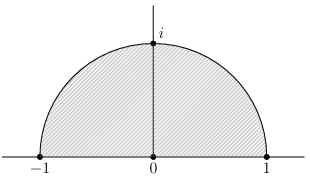
**Этапы работы**

1. Аналитически опишите заданные множества.
2. Воспользовавшись композицией классических преобразований, составьте конформное отображение, которое переводит первую область во вторую. Табличка с преобразовании может быть найдена в конце данного докумнета.
3. Составьте обратное отображение, переводящее второе множество в первое.
4. На любом удобном вам языке программирования напишите программу, которая нарисует первого множества и все этапы его преобразования во второе. Достаточно наглядным будет взять набор точек множества, передающий его форму (учитите, что может понадобится сделать набор «более плотным» в какой-то части множества)

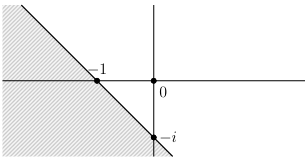
**2. Выполнение пунктов**

* **Аналитическое описания заданных множеств.**

На рисунках ниже представлены множесва (рис.2 и рис.7 соответственно):



(Рисунок 2)



(Рисунок 7)

Давайте аналитически опишем множество №1 (Рисунок 2):

Полюса в комплексной плоскости находятся следующим образом:

,

где граничные линии: и (по условию мы их исключаем); а в верхней полуплоскости должна быть ограничена по действительной части .

Теперь также опишем множество №2 (Рисунок 7):

Верхняя полуплоскость задается следующим множеством:

,

* **Построим конформное отображение**

Для этого давайте подберем последовательности преобразований:

Поскольку наша задача состоит трансформировании множества из Рисунка 2 в множество из Рисунка 7, а также имеются в приложении последовательности элементарных конформных преобразований, можем сделать следующее:

1. **Линейное преобразование сдвига:**

Приведем полосы в единичный отрезок.

1. **Нормализация ширины полосы:**

Давайте для удобства растянем ее по горизонтальной используя масштабирование:

Теперь будет в полосе

1. **Преобразование к полуплоскости:**

Используем экспоненциальное преобразование, чтобы преобразовать ограниченную область в верхнюю полуплоскость:

После этого преобразования мы получаем верхнюю полуплоскость (а именно ).

Таким образом итоговое отображение будет выглядеть так (после

объединения всех этапов):

* **Построим обратное отображение**

Дабы вернуть нашу вторую область (Рисунок 7) в первую (Рисунок 2), давайте воспользуемся последовательными обратными преобразованиями:

1. **Обратное экспоненциальное преобразование:**
2. **Масштабирование:**
3. **Сдвиг назад:**

Таким образом обратное отображение будет иметь вид:

* **Программная реализация**

Вот код для реализации наших конформных преобазований:

function generatePoints() {

const points = [];

const realParts = d3.range(-1, 1, 0.02);

const imagParts = d3.range(0.1, 2, 0.02);

realParts.forEach((re) => {

imagParts.forEach((im) => {

points.push([re, im]);

});

});

return points;

}

function transformZtoW(points) {

return points.map(([re, im]) => {

const z1 = { re: re + 1, im: im };

const z2 = { re: z1.re / 2, im: z1.im / 2 };

const magnitude = Math.exp(-Math.PI \* z2.im);

const phase = Math.PI \* z2.re;

const w = {

re: magnitude \* Math.cos(phase),

im: magnitude \* Math.sin(phase),

};

return [w.re, w.im];

});

}

function plotTransformations() {

const svgWidth = 800;

const svgHeight = 400;

const margin = { top: 20, right: 20, bottom: 20, left: 20 };

const points = generatePoints();

const transformedPoints = transformZtoW(points);

const xScale = d3

.scaleLinear()

.domain([-2, 2])

.range([margin.left, svgWidth - margin.right]);

const yScale = d3

.scaleLinear()

.domain([-2, 2])

.range([svgHeight - margin.bottom, margin.top]);

const svg = d3

.select("body")

.append("svg")

.attr("width", svgWidth)

.attr("height", svgHeight);

svg

.append("g")

.selectAll("circle")

.data(points)

.enter()

.append("circle")

.attr("cx", (d) => xScale(d[0]))

.attr("cy", (d) => yScale(d[1]))

.attr("r", 1)

.attr("fill", "blue")

.attr("opacity", 0.6);

svg

.append("g")

.selectAll("circle")

.data(transformedPoints)

.enter()

.append("circle")

.attr("cx", (d) => xScale(d[0]))

.attr("cy", (d) => yScale(d[1]))

.attr("r", 1)

.attr("fill", "red")

.attr("opacity", 0.6);

const axisBottom = d3.axisBottom(xScale).ticks(10);

const axisLeft = d3.axisLeft(yScale).ticks(10);

svg

.append("g")

.attr("transform", `translate(0, ${yScale(0)})`)

.call(axisBottom);

svg

.append("g")

.attr("transform", `translate(${xScale(0)}, 0)`)

.call(axisLeft);

svg

.append("text")

.attr("x", svgWidth / 4)

.attr("y", margin.top)

.attr("fill", "blue")

.text("Исходное множество (Рис. 2)");

svg

.append("text")

.attr("x", (3 \* svgWidth) / 4)

.attr("y", margin.top)

.attr("fill", "red")

.text("Преобразованное множество (Рис. 7)");

}

plotTransformations();

**3. Заключение**

Мы познакомились на практике с конформными преобразованиями нескольких множеств и их отображений из одного в другое. Также смогли реализовать программно алгоритм действий описанных математически для трансформации одного множества в другое, что дает наглядное и более точное понимание происходящего.