Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Изображение выглядит как Шрифт, Графика, Цвет электрик, логотип

Автоматически созданное описание

Лабораторная работа № 2

По дисциплине “ТФКП”

Вариант №24 (7 => 8)

Выполнил студент группы P3208, поток 22.3:

Петров Вячеслав Маркович

Преподаватель:

Милюшин Александр Сергеевич

Санкт-Петербург 2025 г.

Этапы работы

[Задание 1. 3](#_Toc187168582)

[Задание 2 4](#_Toc187168583)

[Задание 3 5](#_Toc187168584)

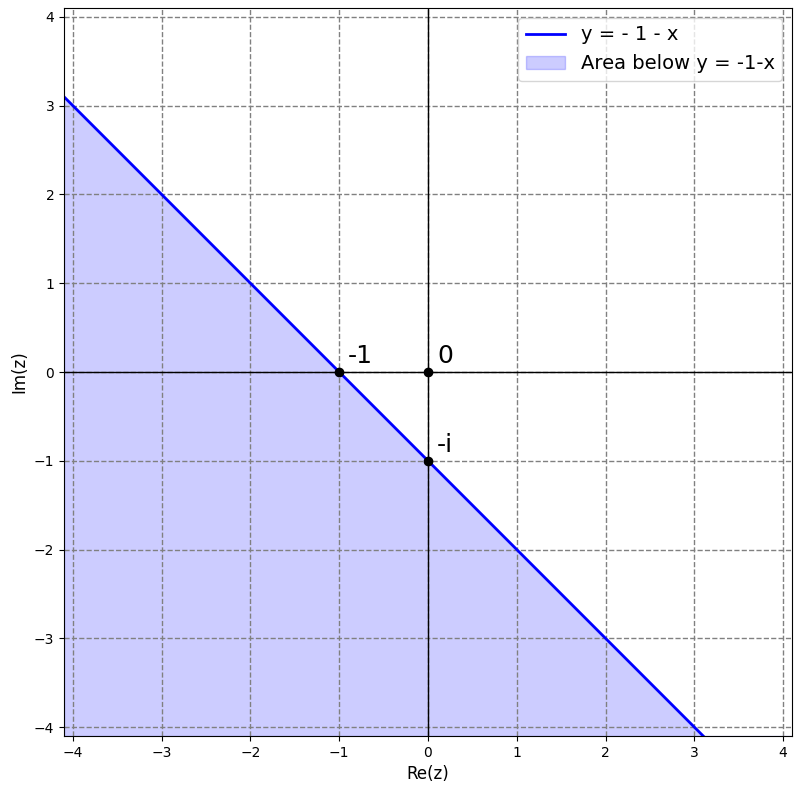
[Задание 4 6](#_Toc187168585)

[Программа на языке Python 6](#_Toc187168586)

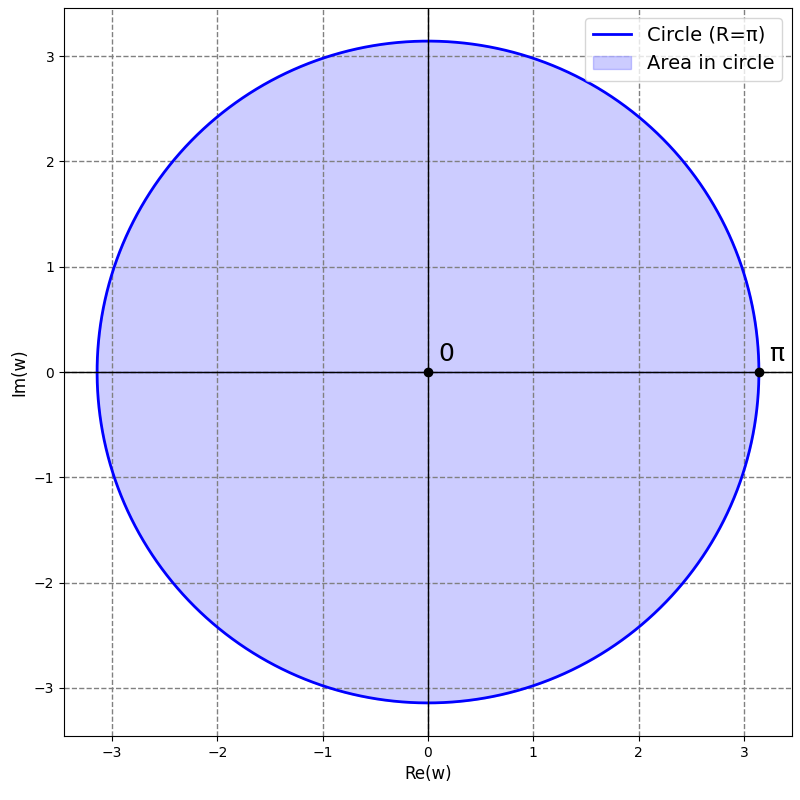
[Вывод 9](#_Toc187168587)

# Задание 1.

На первом графике изображено множество точек



На втором графике изображено множество точек



# Задание 2

Изображение выглядит как линия, График, диаграмма, снимок экрана

Автоматически созданное описаниеКонформное отображение, переводящее первое множество во второе:

1. . Поднимаем график на i.

Изображение выглядит как текст, снимок экрана, линия, прямоугольный

Автоматически созданное описание

1. . Поворачиваем график на по часовой.

Изображение выглядит как диаграмма, линия, График, круг

Автоматически созданное описание

1. . Преобразование в окружность, где точка (0; 1) переходит в центр окружности.
2. . Увеличение радиуса окружности с 1 до

# Задание 3

Изображение выглядит как диаграмма, линия, График, круг

Автоматически созданное описаниеКонформное отображение, переводящее второе множество в первое:

1. . Уменьшение радиуса с до 1

Изображение выглядит как текст, снимок экрана, линия, прямоугольный

Автоматически созданное описание

1. . Переход от окружности к верхней полуплоскости, где центр окружности перешёл в точку (0; 1)

Изображение выглядит как линия, График, диаграмма, снимок экрана

Автоматически созданное описание

1. . Поворот полуплоскости против часовой на
2. . Опускаем на график на i.

# Задание 4

Преобразование первого множества во второе с дополнительными 11 точками, показывающими форму промежуточных и конечных множеств.

Изображение выглядит как текст, диаграмма, График, линия

Автоматически созданное описание

## Программа на языке Python

*import math*

*import numpy as np*

*import matplotlib.pyplot as plt*

*# Задаем значения x*

*x = np.linspace(-4.1, 4.1, 50) # Диапазон от -4 до 4, 50 точек*

*# Создаем график*

*plt.figure(figsize=(8, 8))*

*# фигуры*

*plt.subplot(2, 2, 1)*

*plt.plot(x, -1 - x, label='y = - 1 - x', color='magenta', linewidth=2)*

*plt.subplot(2, 2, 2)*

*plt.plot(x, -x, label='y = - x', color='blue', linewidth=2)*

*plt.subplot(2, 2, 3)*

*plt.plot(x, 0 \* x, label='y = 0', color='red', linewidth=2)*

*plt.subplot(2, 2, 4)*

*theta = np.linspace(0, 2 \* np.pi, 500) # Угол от 0 до 2π*

*radius = np.pi # 1*

*circle\_x = radius \* np.cos(theta) # x-координаты окружности*

*circle\_y = radius \* np.sin(theta) # y-координаты окружности*

*plt.plot(circle\_x, circle\_y, label='Circle (R=π)', color='green', linewidth=2)*

*# Закрашиваем*

*plt.subplot(2, 2, 1)*

*plt.fill\_between(x, -1 - x, -4.1, color='magenta', alpha=0.2, label='Area below y = - 1 - x')*

*plt.subplot(2, 2, 2)*

*plt.fill\_between(x, -x, -4.1, color='blue', alpha=0.2, label='Area below y = -x')*

*plt.subplot(2, 2, 3)*

*plt.fill\_between(x, 0, 6, color='red', alpha=0.2, label='Area above y = 0')*

*plt.subplot(2, 2, 4)*

*plt.fill\_between(circle\_x, circle\_y, color='green', alpha=0.2, label='Area in circle')*

*# точки*

*x\_arr = [-0.7, -1, 1, -3, -0.5, 0.5, 0, 0, -4, -1, 0, 0.5]*

*y\_arr = [-1.7, 0, -2, -4, -0.5, -1.5, -1, -4, -1, -2, -2, -1.87]*

*color\_states = ["red", "green", "blue", "cyan", "magenta", "yellow", "black", "#FAAC07", "#AD9CF1", "#3A8C97", "#4A7AAA", "#6C1C99"]*

*for i in range(len(x\_arr)):*

*x1 = x\_arr[i]*

*y1 = y\_arr[i]*

*color\_state = color\_states[i]*

*plt.subplot(2, 2, 1)*

*plt.plot(x1, y1, color=color\_state, marker='o')*

*plt.annotate('p' + str(i), xy=(x1, y1), xytext=(x1 - 0.1, y1 + 0.2), fontsize=8, color='black')*

*plt.subplot(2, 2, 2)*

*plt.plot(x1, (y1 + 1), color=color\_state, marker='o')*

*plt.annotate('p' + str(i), xy=(x1, (y1 + 1)), xytext=(x1 - 0.2, (y1 + 1) - 0.3), fontsize=8, color='black')*

*plt.subplot(2, 2, 3)*

*plt.plot(-(x1 - y1 - 1) / math.sqrt(2), -(x1 + y1 + 1) / math.sqrt(2), color=color\_state, marker='o')*

*plt.annotate('p' + str(i),*

*xy=(-(x1 - y1 - 1) / math.sqrt(2), -(x1 + y1 + 1) / math.sqrt(2)),*

*xytext=(-(x1 - y1 - 1) / math.sqrt(2) - 0.2, -(x1 + y1 + 1) / math.sqrt(2) - 0.3),*

*fontsize=9, color='black')*

*plt.subplot(2, 2, 4)*

*x\_here = 0.5 \* (x1 - y1 - 1) \*\* 2 + (0.5 \* (x1 + y1 + 1) \*\* 2 - 1)*

*y\_here = math.sqrt(2) \* (x1 - y1 - 1)*

*znamenatel = (0.5 \* (x1 - y1 - 1) \*\* 2 + ((-1 / math.sqrt(2)) \* (x1 + y1 + 1) + 1) \*\* 2)*

*plt.plot(math.pi \* x\_here / znamenatel, math.pi \* y\_here / znamenatel, color=color\_state, marker='o')*

*plt.annotate('p' + str(i),*

*xy=(math.pi \* x\_here / znamenatel, math.pi \* y\_here / znamenatel),*

*xytext=(math.pi \* x\_here / znamenatel + 0.2, math.pi \* y\_here / znamenatel + 0.2),*

*fontsize=12, color='black')*

*# основное*

*for i in range(4):*

*plt.subplot(2, 2, i + 1)*

*plt.xlim(-4.1, 4.1)*

*if i == 2:*

*plt.ylim(-2.1, 6)*

*else:*

*plt.ylim(-4.1, 4.1)*

*plt.axhline(0, color='black', linewidth=1) # Горизонтальная ось*

*plt.axvline(0, color='black', linewidth=1) # Вертикальная ось*

*if i == 3:*

*plt.xlabel('Re(w)', fontsize=12)*

*plt.ylabel('Im(w)', fontsize=12)*

*else:*

*plt.xlabel('Re(z' + "'" \* i + ")", fontsize=12)*

*plt.ylabel('Im(z' + "'" \* i + ")", fontsize=12)*

*plt.xticks(np.arange(-4, 4, 1))*

*plt.grid(color='gray', linestyle='--', linewidth=1)*

*plt.legend(fontsize=8)*

*plt.subplots\_adjust(left=0.08, right=0.99, bottom=0.08, top=0.99)*

*plt.show()*

# Вывод

В ходе выполнения лабораторной работы я изучил построение конформных отображений и их построение при помощи языка программирования Python.