Гафурова Фарангиз

Поток 21.3

Лабораторная работа № 2

«Построение конформных отображений»

В соответствии с вариантом возьмите два рисунка. На них изображены множества, между которыми следует построить некоторое конформное отображение. Множество 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 кон формное отображение, которое переводит первую область во вторую. Табличка с преобразованием может быть найдена в конце данного документа.
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 (учтите, что может понадобиться сделать набор «более плотным» в какой-то части множества).

Решения:

Задание № 1

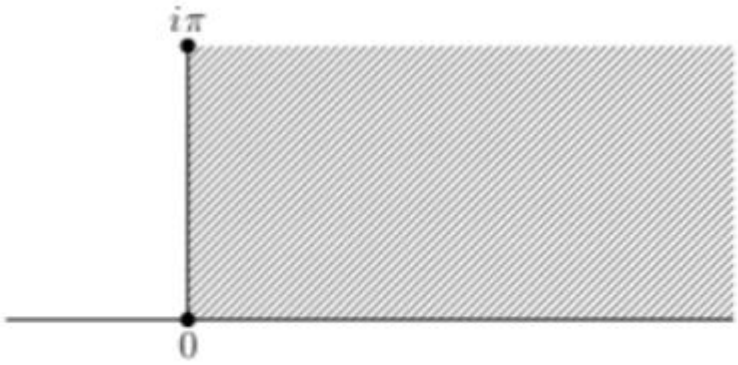


Рисунок 6

Рисунок 1

На рисунке изображена закрашенная область, которая представляет множество точек в комплексной плоскости. Это множество можно записать следующим образом:

$$z\epsilon\mathbb{C} \mid Re(z) \geq 0 \text{ и } 0 \leq Im(z) \leq \pi$$

Здесь:

- $Re(z)$ — действительная часть числа (z),
- $Im(z)$ — мнимая часть числа (z).

Иначе говоря: полуполоса, ограниченная π.

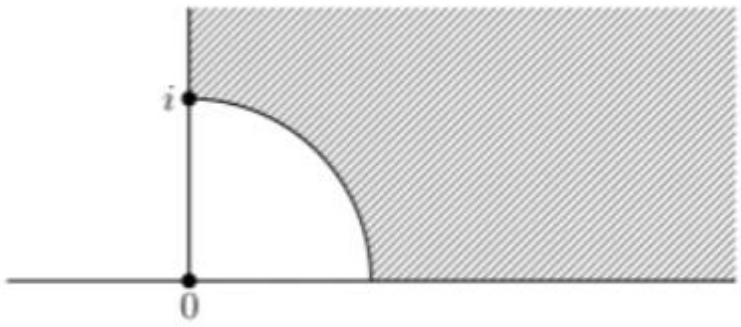


Рисунок 10

Рисунок 2

На рисунке 2 изображена закрашенная область, ограниченная вертикальной прямой, вещественной осью и дугой окружности радиуса 1. Это множество можно записать как:

$$z \in \mathbb{C} \mid Re(z) \geq 0, Im(z) \geq 0 \text{ и } |z| \leq 1$$

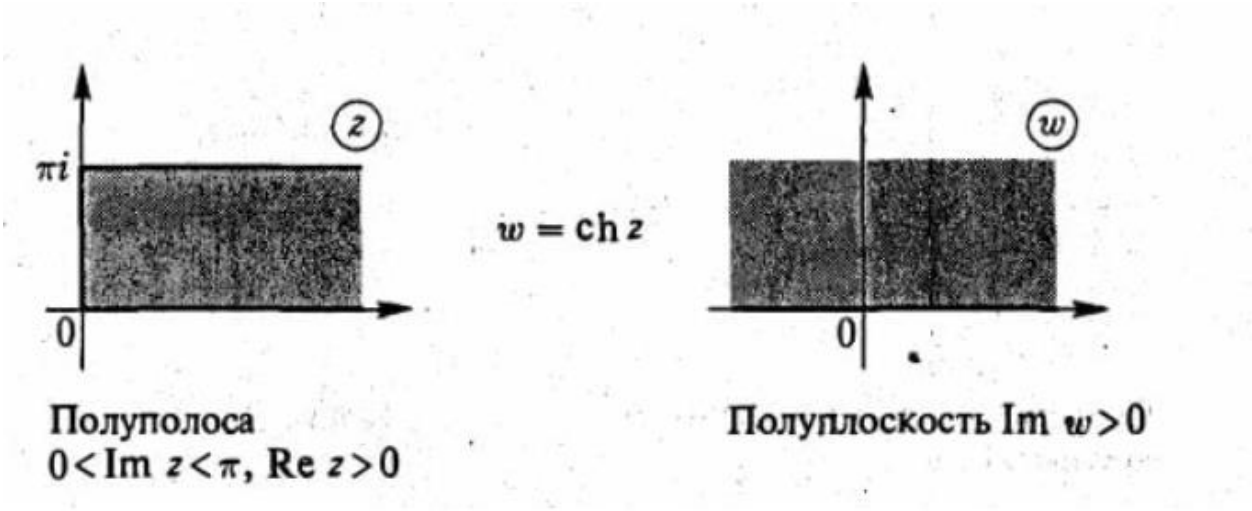
Здесь:

- $|z|$ — модуль комплексного числа (z), т. е. $|z| = \sqrt{Re(z)^2 + Im(z)^2}$

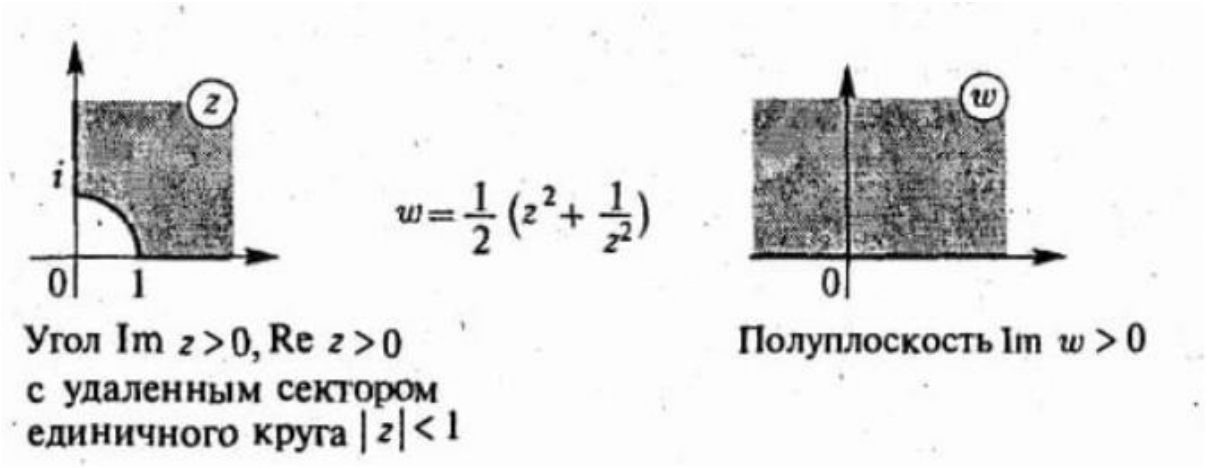
Иначе говоря, на этом рисунке область с выколотой частью единичного круга в первой четверти.

Задание № 2

Для того, чтобы получить из 6 рисунка 10 нужно сначала выполнить данное преобразование:



Потом нам нужно выполнить преобразование, обратное данному:



Для этого выведем нужную формулу

$$\omega = \frac{1}{2}\left(z^2 + \frac{1}{z^2}\right)$$

$$2\omega = z^2 + \frac{1}{z^2}$$

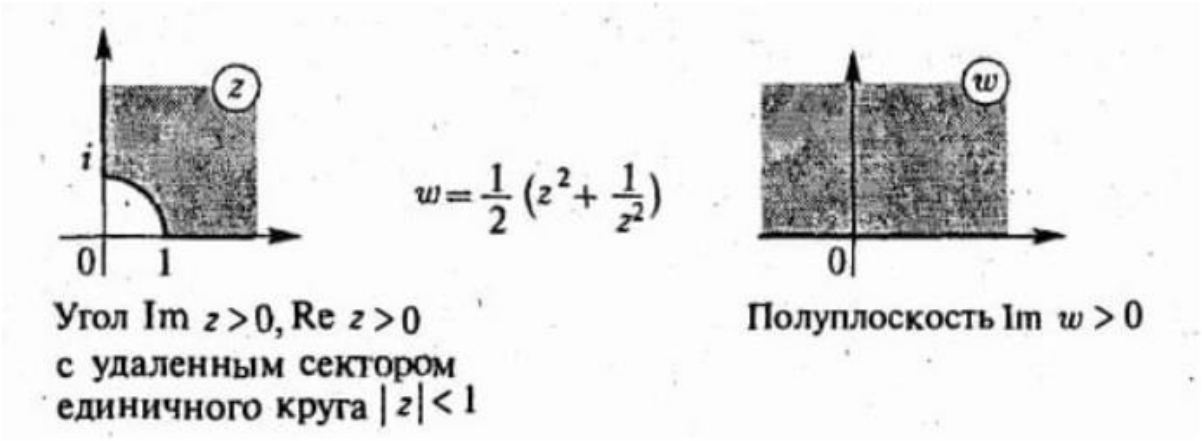
$$z^2 + \frac{1}{z^2} - 2\omega = 0$$

$$\frac{z^4 - 2\omega z^2 + 1}{z^2} = 0$$

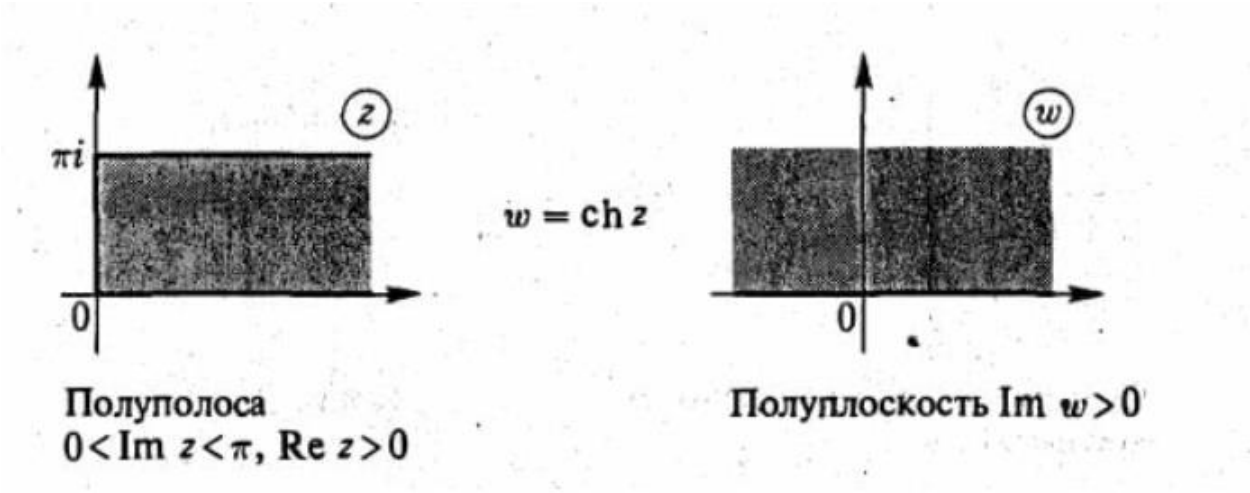
$$z^4 - 2\omega z^2 + 1 = 0$$

Задание № 3

Для обратного преобразования сначала преобразуем угол с удаленным сектором в полуплоскость



Потом нужно выполнить преобразование, обратное данному



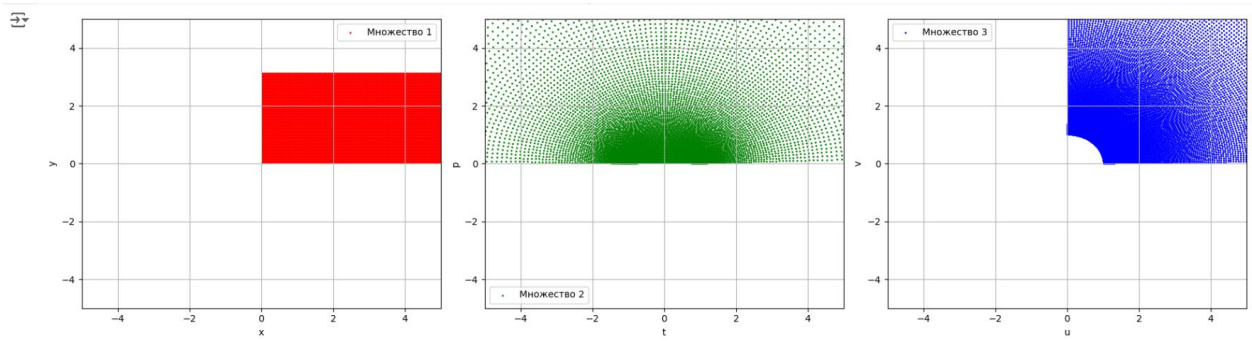
Это будет $arcch(\omega)$ Таким образом мы получаем исходную область.

Задание № 4

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

# ---------------------------------------------------------------------------------------------------------------
POINT_SIZE = 1
# ---------------------------------------------------------------------------------------------------------------
EPS = 1e-3
# ---------------------------------------------------------------------------------------------------------------
POINTS_NUMBER = 1000
# ---------------------------------------------------------------------------------------------------------------
X_MIN = -10
X_MAX = 20
Y_MIN = -10
Y_MAX = 20
# ---------------------------------------------------------------------------------------------------------------
X_VIEW_MIN = -5
X_VIEW_MAX = 5
Y_VIEW_MIN = -5
Y_VIEW_MAX = 5
# ---------------------------------------------------------------------------------------------------------------
RED = 'red'
GREEN = 'green'
BLUE = 'blue'
# ---------------------------------------------------------------------------------------------------------------
x = np.linspace(X_MIN, X_MAX, POINTS_NUMBER)
y = np.linspace(Y_MIN, Y_MAX, POINTS_NUMBER)
X, Y = np.meshgrid(x, y)
area = X + 1j * Y
# ---------------------------------------------------------------------------------------------------------------
basic_area = area[(area.real >= EPS) & (area.imag >= EPS) & (area.imag - pi <= EPS)]
```

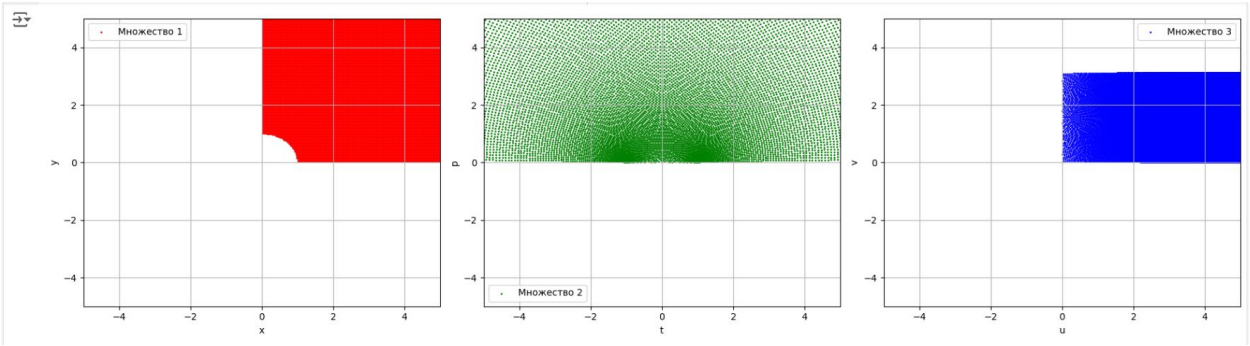
```
area_1 = np.cosh(basic_area)
# ---------------------------------------------------------------------------------------------------------------
t = np.sqrt(area_1 ** 2 - 1)
z1 = np.sqrt(area_1 + t)
z2 = np.sqrt(area_1 - t)
z3 = -np.sqrt(area_1 + t)
z4 = -np.sqrt(area_1 - t)
z = np.concatenate((z1, z2))
area_2 = z[(z.real >= EPS) & (z.imag >= EPS)]
# ---------------------------------------------------------------------------------------------------------------
fig, axs = plt.subplots(1, 3, figsize=(18, 5))
# ---------------------------------------------------------------------------------------------------------------
axs[0].scatter(basic_area.real, basic_area.imag, color=RED, s=POINT_SIZE, label='Множество 1')
axs[0].set_xlabel('x')
axs[0].set_ylabel('y')
# ---------------------------------------------------------------------------------------------------------------
axs[1].scatter(area_1.real, area_1.imag, color=GREEN, s=POINT_SIZE, label='Множество 2')
axs[1].set_xlabel('t')
axs[1].set_ylabel('p')
# ---------------------------------------------------------------------------------------------------------------
axs[2].scatter(area_2.real, area_2.imag, color=BLUE, s=POINT_SIZE, label='Множество 3')
axs[2].set_xlabel('u')
axs[2].set_ylabel('v')
# ---------------------------------------------------------------------------------------------------------------
for ax in axs:
    ax.legend()
    ax.grid(True)
    ax.set_xlim(X_VIEW_MIN, X_VIEW_MAX)
    ax.set_ylim(Y_VIEW_MIN, Y_VIEW_MAX)
# ---------------------------------------------------------------------------------------------------------------
plt.tight_layout()
plt.show()
```




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

# ---------------------------------------------------------------------------------------------------------------
POINT_SIZE = 1
# ---------------------------------------------------------------------------------------------------------------
EPS = 1e-3
# ---------------------------------------------------------------------------------------------------------------
POINTS_NUMBER = 1000
# ---------------------------------------------------------------------------------------------------------------
X_MIN = -10
X_MAX = 20
Y_MIN = -10
Y_MAX = 20
# ---------------------------------------------------------------------------------------------------------------
X_VIEW_MIN = -5
X_VIEW_MAX = 5
Y_VIEW_MIN = -5
Y_VIEW_MAX = 5
# ---------------------------------------------------------------------------------------------------------------
RED = 'red'
GREEN = 'green'
BLUE = 'blue'
# ---------------------------------------------------------------------------------------------------------------
x = np.linspace(X_MIN, X_MAX, POINTS_NUMBER)
y = np.linspace(Y_MIN, Y_MAX, POINTS_NUMBER)
X, Y = np.meshgrid(x, y)
area = X + 1j * Y
# ---------------------------------------------------------------------------------------------------------------
basic_area = area[(area.real >= EPS) & (area.imag >= EPS) & (np.abs(area) - 1 >= EPS)]
# ---------------------------------------------------------------------------------------------------------------
area_1 = 0.5 * (np.square(basic_area) + 1 / np.square(basic_area))
```

```
# ------------------------------------------------------------------------------------------------------
area_2 = np.arccosh(area_1)
# ------------------------------------------------------------------------------------------------------
fig, axs = plt.subplots(1, 3, figsize=(18, 5))
# ------------------------------------------------------------------------------------------------------
axs[0].scatter(basic_area.real, basic_area.imag, color=RED, s=POINT_SIZE, label='Множество 1')
axs[0].set_xlabel('x')
axs[0].set_ylabel('y')
# ------------------------------------------------------------------------------------------------------
axs[1].scatter(area_1.real, area_1.imag, color=GREEN, s=POINT_SIZE, label='Множество 2')
axs[1].set_xlabel('t')
axs[1].set_ylabel('p')
# ------------------------------------------------------------------------------------------------------
axs[2].scatter(area_2.real, area_2.imag, color=BLUE, s=POINT_SIZE, label='Множество 3')
axs[2].set_xlabel('u')
axs[2].set_ylabel('v')
# ------------------------------------------------------------------------------------------------------
for ax in axs:
    ax.legend()
    ax.grid(True)
    ax.set_xlim(X_VIEW_MIN, X_VIEW_MAX)
    ax.set_ylim(Y_VIEW_MIN, Y_VIEW_MAX)
# ------------------------------------------------------------------------------------------------------
plt.tight_layout()
plt.show()
```



Выводы по проделнной работе:

В ходе выполнения работы были успешно достигнуты поставленные цели по построению конформных отображений между заданными множествами.

1. Анализ множеств: Подробно описаны геометрические и топологические свойства множеств.

2. Конформное отображение: Создано отображение, переводящее первое множество во второе с учетом сохранения углов и форм.

3. Обратное отображение: Разработано отображение для перехода от второго множества к первому, подтверждающее взаимосвязь.

4. Программная реализация: Создана программа, визуализирующая преобразование множества и демонстрирующая все его этапы.

Таким образом, работа продемонстрировала не только теоретические основы конформных отображений, но и практическое применение этих знаний в виде программного инструмента для визуализации и анализа.