

Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
ИТМО»**

Поток ТФКП 22.4

Дисциплина: Теория функций комплексного переменного (ТФКП)

**ЛАБОРАТОРНАЯ РАБОТА**  
по теме: «Построение конформных отображений»

Вариант 20

Выполнили:  
Михайлов Петр Сергеевич  
Соболев Егор Викторович

Преподаватель:  
Поздняков Семен Сергеевич

Санкт-Петербург, 2025  
15.12.2025

# Содержание

<b>1 Аналитическое описание множеств</b>	<b>3</b>
1.1 Исходное множество $D$ (Рисунок 5)	3
1.2 Целевое множество $G$ (Рисунок 8)	3
<b>2 Построение конформного отображения <math>D \rightarrow G</math></b>	<b>3</b>
2.1 Шаг 1. $D \rightarrow H$ : Отображение сектора на Верхнюю полуплоскость	3
2.2 Шаг 2. $H \rightarrow K$ : Отображение Верхней полуплоскости на Единичный круг	6
2.3 Шаг 3. $K \rightarrow G$ : Отображение Единичного круга на круг радиуса $\pi$	8
<b>3 Итоговое конформное отображение</b>	<b>10</b>
3.1 Прямое отображение $w = F(z)$	10
3.2 Обратное отображение $z = F^{-1}(w)$	10
<b>4 Комплексная визуализация процесса</b>	<b>12</b>
4.1 Результаты работы программы	12
<b>5 Исходный код полной визуализации</b>	<b>13</b>

# 1 Аналитическое описание множеств

## 1.1 Исходное множество $D$ (Рисунок 5)

Исходное множество представляет собой угловой сектор на комплексной плоскости  $z$ , ограниченный лучами, выходящими из начала координат. Углы наклона граничных лучей к положительному направлению вещественной оси составляют  $\frac{\pi}{4}$  и  $\pi - \frac{\pi}{4} = \frac{3\pi}{4}$ .

Аналитически область  $D$  описывается неравенством для аргумента комплексного числа:

$$D = \left\{ z \in \mathbb{C} \mid \frac{\pi}{4} < \arg z < \frac{3\pi}{4} \right\}$$

Угловая мера сектора составляет  $\alpha = \frac{3\pi}{4} - \frac{\pi}{4} = \frac{\pi}{2}$ .

## 1.2 Целевое множество $G$ (Рисунок 8)

Целевое множество представляет собой открытый круг на комплексной плоскости  $w$  с центром в начале координат. Судя по рисунку, граница проходит через точку  $\pi$  на вещественной оси, то есть радиус круга равен  $\pi$ .

Аналитически область  $G$  описывается неравенством для модуля комплексного числа:

$$G = \{w \in \mathbb{C} \mid |w| < \pi\}$$

# 2 Построение конформного отображения $D \rightarrow G$

Для перевода исходной области  $D$  (сектор) в целевую область  $G$  (круг радиуса  $\pi$ ) воспользуемся композицией трех элементарных конформных преобразований:

$$z \xrightarrow{f_1} z_1 \xrightarrow{f_2} z_2 \xrightarrow{f_3} w$$

где:

- $z \in D$  (Сектор);
- $z_1 \in H$  (Верхняя полуплоскость);
- $z_2 \in K$  (Единичный круг);
- $w \in G$  (Круг радиуса  $\pi$ ).

## 2.1 Шаг 1. $D \rightarrow H$ : Отображение сектора на Верхнюю полуплоскость

Для перевода области  $D$  в полуплоскость (угол  $\pi$ ) требуется «развернуть» сектор угловой мерой  $\frac{\pi}{2}$  (согласно справочному рисунку №4). Используем преобразование  $f_1(z) = z_1$ .

**1.1. Поворот:** Сначала повернем исходный сектор по часовой стрелке на угол  $\frac{\pi}{4}$ , чтобы одна из его сторон легла на положительную вещественную полуось. Преобразование поворота:

$$z' = z \cdot e^{-i\frac{\pi}{4}}$$

Проверим изменение аргументов:

$$\frac{\pi}{4} - \frac{\pi}{4} < \arg z' < \frac{3\pi}{4} - \frac{\pi}{4} \implies 0 < \arg z' < \frac{\pi}{2}$$

Область  $D'$  теперь представляет собой первую координатную четверть.

**1.2. Степенная функция:** Чтобы отобразить первую четверть (угол  $\frac{\pi}{2}$ ) на верхнюю полуплоскость (угол  $\pi$ ), нужно удвоить аргумент. Используем функцию  $z_1 = (z')^2$ .

$$z_1 = (z \cdot e^{-i\frac{\pi}{4}})^2 = z^2 \cdot e^{-i\frac{\pi}{2}}$$

Так как  $e^{-i\frac{\pi}{2}} = -i$ , получаем окончательную формулу отображения:

$$z_1 = -iz^2$$

**1.3. Визуализация:** Ниже представлены графики исходного множества и результата его первого отображения. Цветовая раскраска точек сохраняется при отображении, что позволяет отследить соответствие границ.

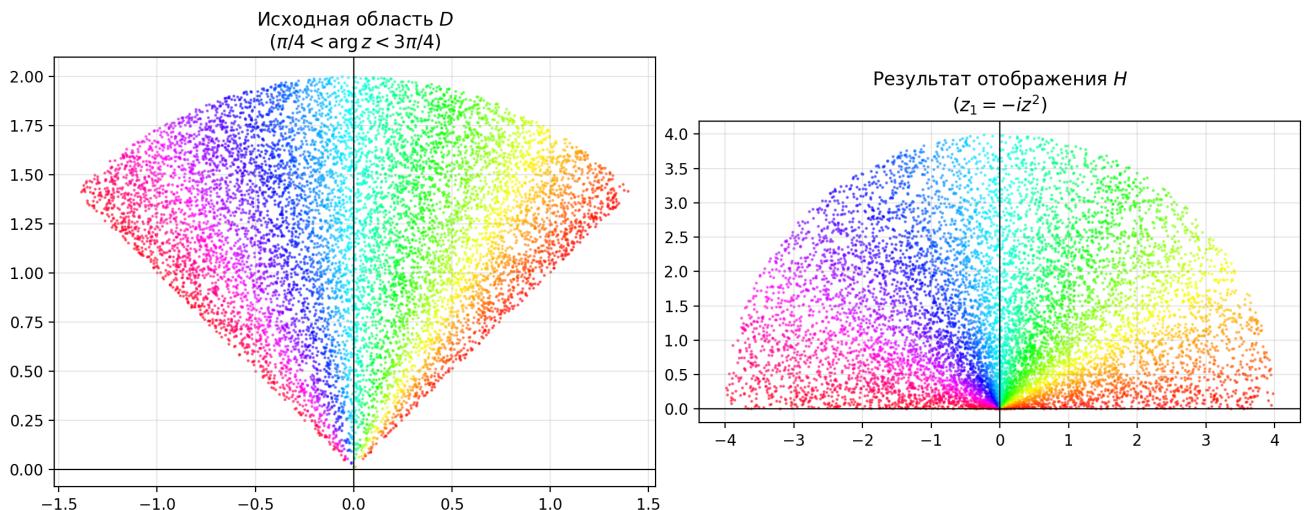


Рис. 1: Слева: Исходный сектор  $D$  ( $\pi/4 < \arg z < 3\pi/4$ ). Справа: Промежуточная область  $H$  ( $\operatorname{Im} z_1 > 0$ ).

## Исходный код программы (Шаг 1)

```
1 def get_grid_points():
2     """
3         Создает сетку точек в секторе  $\pi/4 < \arg(z) < 3\pi/4$ .
4         Используем полярную сетку для красивых линий сетки.
5     """
6     rs = np.linspace(0.1, 2.0, 15) # Радиусы от 0.1 до 2.0
7     thetas = np.linspace(np.pi / 4, 3 * np.pi / 4, 30) # Углы от  $\pi/4$  до  $3\pi/4$ 
8
9     # 1. Радиальные линии (лучи)
10    lines = []
11    for t in thetas:
12        r_line = np.linspace(0, 2.0, 100)
13        lines.append(r_line * np.exp(1j * t))
14
15    # 2. Дуговые линии (окружности)
16    for r in rs:
17        t_arc = np.linspace(np.pi / 4, 3 * np.pi / 4, 100)
18        lines.append(r * np.exp(1j * t_arc))
19
20    # Преобразуем список массивов в один длинный массив для анимации
21    # Добавляем NaN разделятели, чтобы линии не соединялись
22    Z_lines = []
23    for line in lines:
24        Z_lines.extend(line)
25        Z_lines.append(np.nan + 1j * np.nan)
26    return np.array(Z_lines)
27
28 def mapping(z):
29     return -1j * (z ** 2) #  $z_1 = -i z^2$ 
30
31 Z = get_grid_points()
32 W = mapping(Z)
33
34 def save_static_report_image():
35     # Генерируем плотное облако точек для красивой картинки (в отличие от линий сетки выше)
36     num_pts = 10000
37     r_rand = np.sqrt(np.random.uniform(0, 4, num_pts)) # sqrt для равномерности круга
38     t_rand = np.random.uniform(np.pi / 4, 3 * np.pi / 4, num_pts)
39     Z_cloud = r_rand * np.exp(1j * t_rand)
40     W_cloud = mapping(Z_cloud)
41
42     fig, ax = plt.subplots(1, 2, figsize=(12, 5))
43
44     colors = np.angle(Z_cloud) # Раскраска по углу (чтобы видеть, куда переходят границы)
45
46     # Левая часть: исходный рисунок (gifted power)
47     ax[0].scatter(Z_cloud.real, Z_cloud.imag, c=colors, cmap='hsv', s=1, alpha=0.5)
48     ax[0].set_title("Исходная область $D$\n $\pi/4 < \arg z < 3\pi/4$ ")
49     ax[0].axhline(0, color='k', lw=0.8)
50     ax[0].axvline(0, color='k', lw=0.8)
51     ax[0].grid(True, alpha=0.3)
52     ax[0].set_aspect('equal')
53
54     # Правая часть: образ конформного отображения (pure effort)
55     ax[1].scatter(W_cloud.real, W_cloud.imag, c=colors, cmap='hsv', s=1, alpha=0.5)
56     ax[1].set_title("Результат отображения $H$\n $z_1 = -i z^2$ ")
57     ax[1].axhline(0, color='k', lw=0.8)
58     ax[1].axvline(0, color='k', lw=0.8)
59     ax[1].grid(True, alpha=0.3)
60     ax[1].set_aspect('equal')
61
62     plt.tight_layout()
63     plt.savefig("output/img/static_mapping1.png", dpi=200)
64     print("Картина 'static_mapping1.png' сохранена.")
65     plt.close()
66
67 save_static_report_image()
```

Листинг 1: Python код для визуализации отображения области D на область H

## 2.2 Шаг 2. $H \rightarrow K$ : Отображение Верхней полуплоскости на Единичный круг

Требуется отобразить Верхнюю полуплоскость  $H = \{z_1 \mid \operatorname{Im} z_1 > 0\}$  на Единичный круг  $K = \{z_2 \mid |z_2| < 1\}$ . Для этого используем Дробно-линейное преобразование (Преобразование Мёбиуса), соответствующее справочному рисунку №11.

**2.1. Выбор преобразования:** Общий вид преобразования, отображающего верхнюю полуплоскость на единичный круг:

$$z_2 = e^{i\alpha} \frac{z_1 - z_0}{z_1 - \bar{z}_0}$$

где  $z_0$  — любая точка в  $H$  ( $\operatorname{Im} z_0 > 0$ ), которая отображается в центр круга ( $z_2 = 0$ ), а  $\alpha \in \mathbb{R}$  — поворот.

Для максимальной простоты выберем:

- $z_0 = i$  (центр, лежащий в  $H$ );
- $\alpha = 0$  (отсутствие дополнительного поворота).

Подставляя эти значения, получаем:

$$\begin{aligned} z_2 &= \frac{z_1 - i}{z_1 - \bar{i}} = \frac{z_1 - i}{z_1 - (-i)} \\ z_2 &= \frac{z_1 - i}{z_1 + i} \end{aligned}$$

**2.2. Проверка граничных точек (для наглядности):** Проверим, как преобразуется вещественная ось  $\operatorname{Im} z_1 = 0$  (граница  $H$ ) в границу круга  $|z_2| = 1$ .

- Точка  $z_1 = 0 \in \mathbb{R}$  переходит в  $z_2 = \frac{0-i}{0+i} = -1$ .
- Точка  $z_1 = 1 \in \mathbb{R}$  переходит в  $z_2 = \frac{1-i}{1+i} = \frac{(1-i)^2}{(1+i)(1-i)} = \frac{1-2i-1}{1+1} = \frac{-2i}{2} = -i$ .

Таким образом, вещественная ось отображается на окружность  $|z_2| = 1$ .

**2.3. Визуализация:** Ниже представлен результат второго этапа: отображение верхней полуплоскости  $H$  на единичный круг  $K$ .

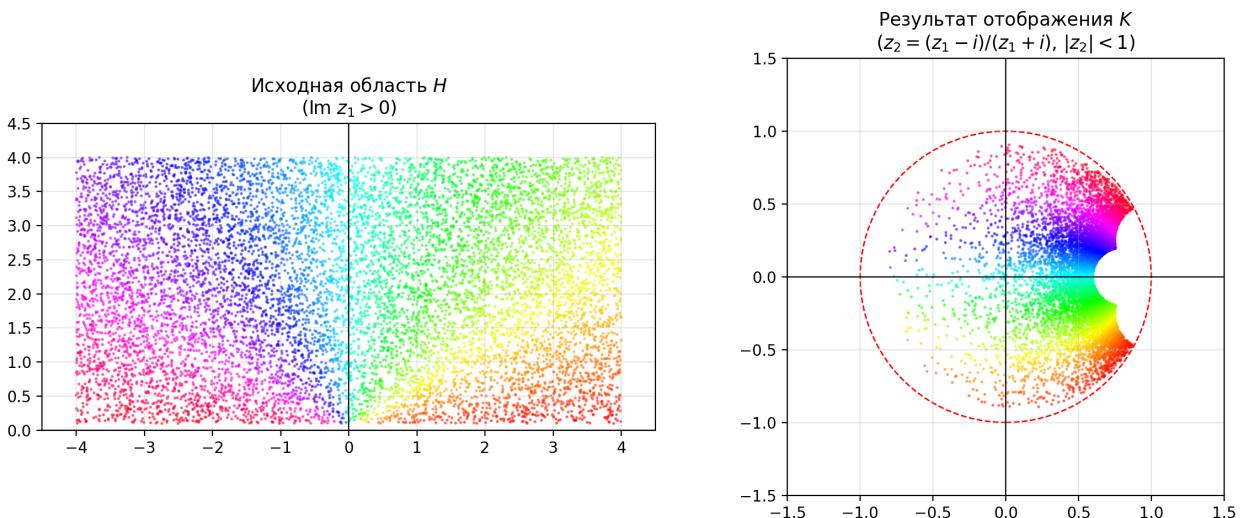


Рис. 2: Слева: Промежуточная область  $H$  ( $\operatorname{Im} z_1 > 0$ ). Справа: Результат отображения во второй промежуточный круг  $K$  ( $|z_2| < 1$ ).

## Исходный код программы (Шаг 2)

```
1 def get_grid_points():
2     """
3         Создает сетку точек в Верхней полуплоскости H.
4         Используем прямоугольную сетку.
5     """
6     reals = np.linspace(-4, 4, 15) # вещественные линии (вертикальные)
7     imgs = np.linspace(0.1, 4, 10) # мнимые линии (горизонтальные)
8
9     lines = []
10
11    # 1. Вертикальные линии (Re = const)
12    for r in reals:
13        im_line = np.linspace(0, 4, 100)
14        lines.append(r + 1j * im_line)
15
16    # 2. Горизонтальные линии (Im = const)
17    for i in imgs:
18        re_line = np.linspace(-4, 4, 100)
19        lines.append(re_line + 1j * i)
20
21    # Преобразуем список массивов в один длинный массив для анимации
22    # Добавляем NaN разделятели, чтобы линии не соединялись
23    Z_lines = []
24    for line in lines:
25        Z_lines.extend(line)
26        Z_lines.append(np.nan + 1j * np.nan)
27
28    return np.array(Z_lines)
29
30 def mapping(z1):
31     return (z1 - 1j) / (z1 + 1j) # z2 = (z1 - i) / (z1 + i)
32
33 Z1 = get_grid_points()
34 Z2 = mapping(Z1)
35
36 def save_static_report_image():
37     num_pts = 10000 # Генерируем плотное облако точек
38     Z1_real = np.random.uniform(-4, 4, num_pts)
39     Z1_imag = np.random.uniform(0.1, 4, num_pts)
40     Z1_cloud = Z1_real + 1j * Z1_imag
41     Z2_cloud = mapping(Z1_cloud)
42     fig, ax = plt.subplots(1, 2, figsize=(12, 5))
43     colors = np.angle(Z1_cloud) # Раскраска по углу (чтобы отследить конформность)
44     # Левая часть: Исходная область H
45     ax[0].scatter(Z1_cloud.real, Z1_cloud.imag, c=colors, cmap='hsv', s=1, alpha=0.5)
46     ax[0].set_title("Исходная область $H$\n\$text{Im } z_1 > 0$")
47     ax[0].axhline(0, color='k', lw=0.8)
48     ax[0].axvline(0, color='k', lw=0.8)
49     ax[0].set_xlim(-4.5, 4.5)
50     ax[0].set_ylim(0, 4.5)
51     ax[0].grid(True, alpha=0.3)
52     ax[0].set_aspect('equal')
53     # Правая часть: Образ конформного отображения K
54     ax[1].scatter(Z2_cloud.real, Z2_cloud.imag, c=colors, cmap='hsv', s=1, alpha=0.5)
55     ax[1].set_title("Результат отображения $K$\n\$z_2 = (z_1 - i)/(z_1 + i)$, $|z_2| < 1$")
56     ax[1].axhline(0, color='k', lw=0.8)
57     ax[1].axvline(0, color='k', lw=0.8)
58     # Добавляем границу единичного круга
59     ax[1].add_patch(plt.Circle((0, 0), 1.0, color='red', fill=False, linestyle='--'))
60     ax[1].set_xlim(-1.5, 1.5)
61     ax[1].set_ylim(-1.5, 1.5)
62     ax[1].grid(True, alpha=0.3)
63     ax[1].set_aspect('equal')
64     plt.tight_layout()
65     plt.savefig("output/img/static_mapping2.png", dpi=200)
66     print("Картина 'output/img/static_mapping2.png' сохранена.")
67     plt.close()
68
69 save_static_report_image()
```

Листинг 2: Python код для визуализации отображения области H на область K

## 2.3 Шаг 3. $K \rightarrow G$ : Отображение Единичного круга на круг радиуса $\pi$

Требуется отобразить Единичный круг  $K = \{z_2 \mid |z_2| < 1\}$  на целевой круг  $G = \{w \mid |w| < \pi\}$ . Это преобразование является Гомотетией (растяжением), соответствующим справочному рисунку №3.

**3.1. Выбор преобразования:** Преобразование имеет вид  $w = kz_2$ , где  $k$  — коэффициент растяжения. Поскольку радиус  $R_K = 1$  должен перейти в радиус  $R_G = \pi$ , коэффициент  $k$  должен быть равен  $\pi$ .

$$w = \pi z_2$$

**3.2. Проверка граничных точек:** Проверим модуль:

$$|w| = |\pi z_2| = \pi |z_2|$$

Если  $|z_2| < 1$ , то  $|w| < \pi \cdot 1 = \pi$ . Целевая область достигнута.

**3.3. Визуализация:** Ниже представлен результат третьего и последнего этапа: растяжение единичного круга  $K$  в круг радиуса  $\pi$  (область  $G$ ).

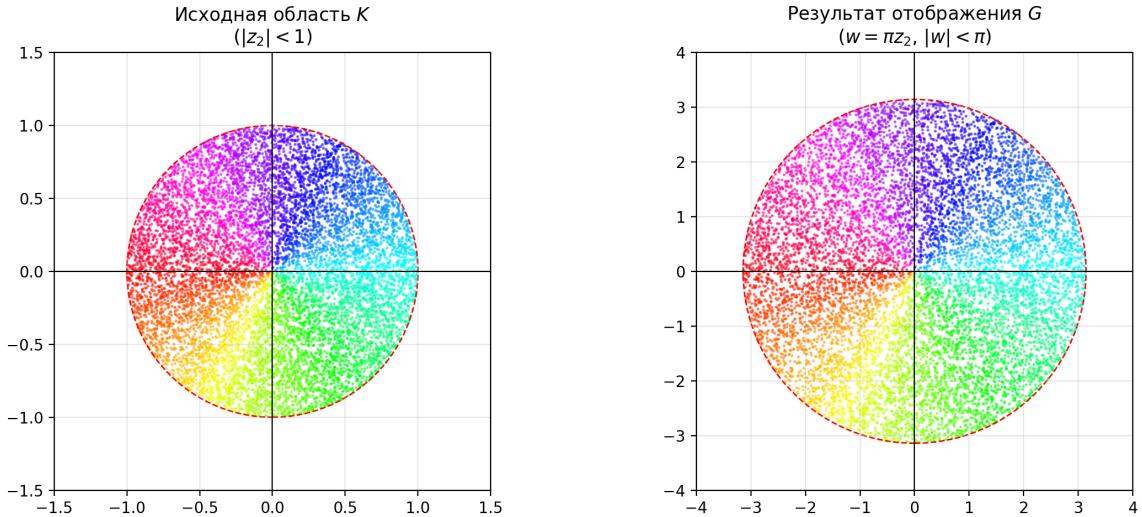


Рис. 3: Слева: Промежуточная область  $K$  ( $|z_2| < 1$ ). Справа: Результат отображения в целевую область  $G$  ( $|w| < \pi$ ).

## Исходный код программы (Шаг 3)

```
1 def get_grid_points():
2     """
3         Создает сетку точек в Единичном круге K.
4         Используем полярную сетку для красивых линий сетки.
5     """
6     rs = np.linspace(0.1, 1.0, 10) # Радиусы от 0.1 до 1.0
7     thetas = np.linspace(0, 2 * np.pi, 30, endpoint=False) # Углы от 0 до 2pi
8     lines = []
9     # 1. Радиальные линии (лучи)
10    for t in thetas:
11        r_line = np.linspace(0, 1.0, 100)
12        lines.append(r_line * np.exp(1j * t))
13    # 2. Дуговые линии (окружности)
14    for r in rs:
15        t_arc = np.linspace(0, 2 * np.pi, 100)
16        lines.append(r * np.exp(1j * t_arc))
17    # Преобразуем список массивов в один длинный массив для анимации
18    # Добавляем NaN разделители, чтобы линии не соединялись
19    Z_lines = []
20    for line in lines:
21        Z_lines.extend(line)
22        Z_lines.append(np.nan + 1j * np.nan)
23
24    return np.array(Z_lines)
25
26 def mapping(z2):
27     return np.pi * z2 # w = pi * z2 (Гомотетия)
28
29 Z2 = get_grid_points()
30 W = mapping(Z2)
31
32 def save_static_report_image():
33     num_pts = 10000 # Генерируем плотное облако точек
34     # Используем sqrt для равномерного распределения точек внутри круга
35     r_rand = np.sqrt(np.random.uniform(0, 1, num_pts))
36     t_rand = np.random.uniform(0, 2 * np.pi, num_pts)
37     Z2_cloud = r_rand * np.exp(1j * t_rand)
38     W_cloud = mapping(Z2_cloud)
39     fig, ax = plt.subplots(1, 2, figsize=(12, 5))
40     # Раскраска по углу
41     colors = np.angle(Z2_cloud)
42     # Левая часть: Исходная область K
43     ax[0].scatter(Z2_cloud.real, Z2_cloud.imag, c=colors, cmap='hsv', s=1, alpha=0.5)
44     ax[0].set_title("Исходная область $K$\n$(|z_2| < 1)$")
45     ax[0].axhline(0, color='k', lw=0.8)
46     ax[0].avxline(0, color='k', lw=0.8)
47     ax[0].add_patch(plt.Circle((0, 0), 1.0, color='red', fill=False, linestyle='--'))
48     ax[0].set_xlim(-1.5, 1.5)
49     ax[0].set_ylim(-1.5, 1.5)
50     ax[0].grid(True, alpha=0.3)
51     ax[0].set_aspect('equal')
52
53     # Правая часть: Образ конформного отображения G
54     ax[1].scatter(W_cloud.real, W_cloud.imag, c=colors, cmap='hsv', s=1, alpha=0.5)
55     ax[1].set_title("Результат отображения $G$\n$(w = \pi z_2, |w| < \pi)$")
56     ax[1].axhline(0, color='k', lw=0.8)
57     ax[1].avxline(0, color='k', lw=0.8)
58     # Добавляем границу целевого круга
59     ax[1].add_patch(plt.Circle((0, 0), np.pi, color='red', fill=False, linestyle='--'))
60     ax[1].set_xlim(-4, 4)
61     ax[1].set_ylim(-4, 4)
62     ax[1].grid(True, alpha=0.3)
63     ax[1].set_aspect('equal')
64     plt.tight_layout()
65     plt.savefig("output/img/static_mapping3.png", dpi=200)
66     print("Картина 'output/img/static_mapping3.png' сохранена.")
67     plt.close()
68
69 save_static_report_image()
```

Листинг 3: Python код для визуализации отображения области K на область G

### 3 Итоговое конформное отображение

#### 3.1 Прямое отображение $w = F(z)$

Составим композицию, последовательно подставляя преобразования:

$$w = f_3(f_2(f_1(z)))$$

1. Вспомним Шаг 1:  $z_1 = -iz^2$

2. Подставим  $z_1$  в Шаг 2 ( $z_2$ ):

$$z_2 = \frac{z_1 - i}{z_1 + i} = \frac{(-iz^2) - i}{(-iz^2) + i} = \frac{-i(z^2 + 1)}{-i(z^2 - 1)} = \frac{z^2 + 1}{z^2 - 1}$$

3. Подставим  $z_2$  в Шаг 3 ( $w$ ):

$$\mathbf{w} = \pi \mathbf{z}_2 = \pi \frac{\mathbf{z}^2 + 1}{\mathbf{z}^2 - 1}$$

Это и есть искомая формула конформного отображения области  $D$  на область  $G$ .

#### 3.2 Обратное отображение $z = F^{-1}(w)$

Требуется составить обратное отображение  $z = F^{-1}(w)$ , переводящее целевое множество  $G$  (круг радиуса  $\pi$ ) обратно в исходный сектор  $D$ .

Обратная композиция:

$$w \xrightarrow{f_3^{-1}} z_2 \xrightarrow{f_2^{-1}} z_1 \xrightarrow{f_1^{-1}} z$$

##### 1. Обратный Шаг 3: $G \rightarrow K$ (Обратная гомотетия)

$$w = \pi z_2 \implies z_2 = \frac{1}{\pi} w$$

$$f_3^{-1}(w) = \frac{w}{\pi}$$

2. Обратный Шаг 2:  $K \rightarrow H$  (Обратное преобразование Мёбиуса) Используем прямое преобразование  $z_2 = \frac{z_1 - i}{z_1 + i}$  и выразим  $z_1$  через  $z_2$ :

$$z_2(z_1 + i) = z_1 - iz_2 + iz_2 = z_1 - iz_2 z_1 - z_1 = -i - iz_2 z_1 (z_2 - 1) = -i(1 + z_2)z_1 = -i \frac{1 + z_2}{z_2 - 1} = i \frac{1 + z_2}{1 - z_2}$$

$$f_2^{-1}(z_2) = i \frac{1 + z_2}{1 - z_2}$$

3. Обратный Шаг 1:  $H \rightarrow D$  (Обратная степенная функция и поворот) Используем прямое преобразование  $z_1 = -iz^2$ . Выразим  $z$  через  $z_1$ :

$$z^2 = \frac{z_1}{-i} = iz_1$$

$$z = \sqrt{iz_1}$$

Для выбора ветви корня необходимо учесть, что обратное отображение должно вернуть нас в исходную область  $D$ :  $\frac{\pi}{4} < \arg z < \frac{3\pi}{4}$ . Поскольку  $\arg(\sqrt{u}) = \frac{1}{2}\arg u$ , и  $\arg(iz_1) = \frac{\pi}{2} + \arg z_1$ , то:

$$\arg z = \frac{1}{2} \left( \frac{\pi}{2} + \arg z_1 \right)$$

Так как  $z_1 \in H$ ,  $\arg z_1 \in (0, \pi)$ .

$$\frac{1}{2} \left( \frac{\pi}{2} + 0 \right) < \arg z < \frac{1}{2} \left( \frac{\pi}{2} + \pi \right)$$

$$\frac{\pi}{4} < \arg z < \frac{3\pi}{4}$$

Выбор главной ветви квадратного корня  $z = \sqrt{iz_1}$  является корректным, так как он возвращает нас в требуемый сектор  $D$ .

$$f_1^{-1}(z_1) = \sqrt{iz_1}$$

**4. Окончательная формула обратного отображения:** Собираем композицию  $z = f_1^{-1}(f_2^{-1}(f_3^{-1}(w)))$ :

$$z = \sqrt{i \cdot \left( i \frac{1 + \frac{w}{\pi}}{1 - \frac{w}{\pi}} \right)}$$

Упростим выражение под корнем:

$$i \cdot i \frac{1 + w/\pi}{1 - w/\pi} = -1 \cdot \frac{(\pi + w)/\pi}{(\pi - w)/\pi} = -\frac{\pi + w}{\pi - w} = \frac{\pi + w}{w - \pi}$$

Итоговая формула обратного отображения:

$$\mathbf{z} = \sqrt{\frac{\pi + \mathbf{w}}{\mathbf{w} - \pi}}$$

**Обоснование выбора ветви:** Функция комплексного квадратного корня является двузначной. Поскольку обратное отображение  $F^{-1} : G \rightarrow D$  должно быть однозначным и переводить точки из  $G$  строго в исходную область  $D$ , необходимо выбрать конкретную ветвь корня. Для обеспечения попадания в сектор  $D = \{z \mid \pi/4 < \arg z < 3\pi/4\}$ , выбирается та ветвь  $\sqrt{\cdot}$ , которая соответствует непрерывному изменению аргумента  $\arg z$  в требуемом диапазоне.

## 4 Комплексная визуализация процесса

### 4.1 Результаты работы программы

Ниже представлен итоговый график, который объединяет все три этапа в одну наглядную схему, демонстрируя композицию преобразований  $D \rightarrow H \rightarrow K \rightarrow G$ .

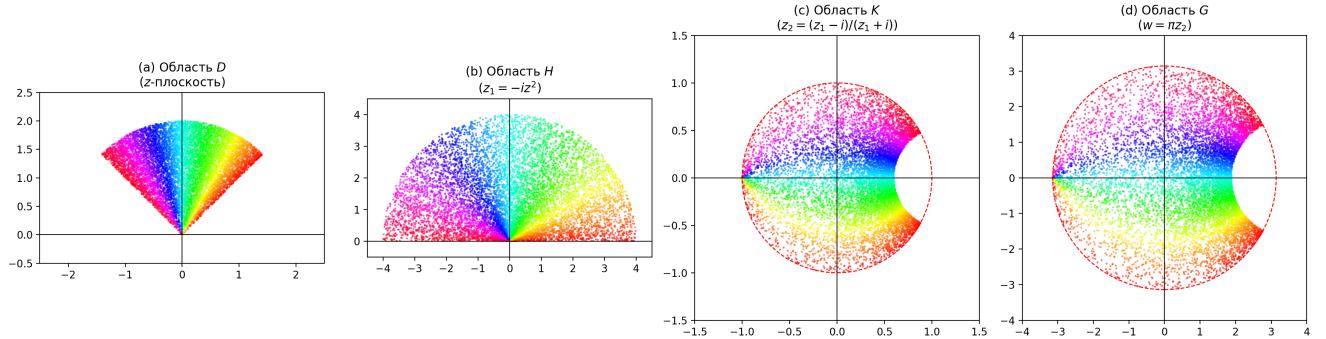


Рис. 4: Последовательная визуализация конформного отображения  $D \rightarrow G$ .

(a) Исходный сектор  $D$ . (b) Промежуточная область  $H$  ( $z_1 = -iz^2$ ). (c) Промежуточная область  $K$  ( $z_2 = (z_1 - i)/(z_1 + i)$ ). (d) Целевой круг  $G$  ( $w = \pi z_2$ ).

## 5 Исходный код полной визуализации

```
1 def get_grid_points():
2     """
3         Создает сетку точек в исходном секторе D: pi/4 < arg(z) < 3pi/4.
4     """
5     rs = np.linspace(0.1, 2.0, 15) # Радиусы от 0.1 до 2.0
6     thetas = np.linspace(np.pi / 4, 3 * np.pi / 4, 30) # Углы от pi/4 до 3pi/4
7
8     lines = []
9     # 1. Радиальные линии
10    for t in thetas:
11        r_line = np.linspace(0, 2.0, 100)
12        lines.append(r_line * np.exp(1j * t))
13    # 2. Дуговые линии
14    for r in rs:
15        t_arc = np.linspace(np.pi / 4, 3 * np.pi / 4, 100)
16        lines.append(r * np.exp(1j * t_arc))
17
18    Z_lines = []
19    for line in lines:
20        Z_lines.extend(line)
21        Z_lines.append(np.nan + 1j * np.nan)
22    return np.array(Z_lines)
23
24
25 # Функции отображения
26
27 def f1(z):
28     # D -> H: z1 = -i * z^2
29     return -1j * (z ** 2)
30
31
32 def f2(z1):
33     # H -> K: z2 = (z1 - i) / (z1 + i)
34     return (z1 - 1j) / (z1 + 1j)
35
36
37 def f3(z2):
38     # K -> G: w = pi * z2
39     return np.pi * z2
40
41
42 # Исходные точки и все промежуточные/конечные результаты
43 Z = get_grid_points()
44 Z1 = f1(Z) # H
45 Z2 = f2(Z1) # K
46 W = f3(Z2) # G
47
48 def get_cloud_points(num_pts=10000):
49     """Генерирует облако точек в исходном секторе D."""
50     r_rand = np.sqrt(np.random.uniform(0, 4, num_pts))
51     t_rand = np.random.uniform(np.pi / 4, 3 * np.pi / 4, num_pts)
52     return r_rand * np.exp(1j * t_rand)
53
54
55 def save_full_static_image():
56     Z_cloud = get_cloud_points()
57     Z1_cloud = f1(Z_cloud)
58     Z2_cloud = f2(Z1_cloud)
59     W_cloud = f3(Z2_cloud)
60
61     clouds = [Z_cloud, Z1_cloud, Z2_cloud, W_cloud]
62     titles = [
63         "(a) Область $D$\n($z$-плоскость"),
64         "(b) Область $H$\n($z_1 = -i z^2$"),
65         "(c) Область $K$\n($z_2 = (z_1-i)/(z_1+i)$"),
66         "(d) Область $G$\n($w = \pi z_2$")"
67     ]
68     x_limits = [(-2.5, 2.5), (-4.5, 4.5), (-1.5, 1.5), (-4, 4)]
69     y_limits = [(-0.5, 2.5), (-0.5, 4.5), (-1.5, 1.5), (-4, 4)]
70     colors = np.angle(Z_cloud)
71
72     fig, ax = plt.subplots(1, 4, figsize=(18, 5))
```

```

74     for i in range(4):
75         ax[i].scatter(clouds[i].real, clouds[i].imag, c=colors, cmap='hsv', s=1, alpha=0.5)
76         ax[i].set_title(titles[i])
77         ax[i].set_xlim(x_limits[i])
78         ax[i].set_ylim(y_limits[i])
79         ax[i].set_aspect('equal')
80         ax[i].axhline(0, color='k', lw=0.8)
81         ax[i].axvline(0, color='k', lw=0.8)
82
83     # Добавляем границы для кругов
84     if i == 2: # K
85         ax[i].add_patch(plt.Circle((0, 0), 1.0, color='red', fill=False, linestyle='--'))
86     if i == 3: # G
87         ax[i].add_patch(plt.Circle((0, 0), np.pi, color='red', fill=False, linestyle='--'))
88
89 plt.tight_layout()
90 plt.savefig(os.path.join("output/img", "full_mapping.png"), dpi=200)
91 print("Статическое изображение 'output/img/full_mapping.png' сохранено.")
92 plt.close()
93
94
95 save_full_static_image()

```

Листинг 4: Python код для визуализации всей композиции отображения  $D \rightarrow G$

## Доступность исходного кода

Полный исходный код, использованный для генерации статических изображений и GIF-анимаций (файлы `conformal_mapping1.py`, `conformal_mapping2.py`, `conformal_mapping3.py` и `full_mapping.py`), доступен в публичном репозитории на GitHub.

**Ссылка на репозиторий GitHub:**

<https://github.com/Axe-On-You/tfkp-lab2>

Структура проекта:

- `conformal_mapping[1, 2, 3].py`: Код для визуализации отдельных шагов  $D \rightarrow H$ ,  $H \rightarrow K$ ,  $K \rightarrow G$ .
- `full_mapping.py`: Код для генерации итоговой схемы и GIF-анимации  $D \rightarrow H \rightarrow K \rightarrow G$ .
- `output/img/`: Сохраненные статические изображения.
- `output/gif/`: Сохраненные GIF-анимации.