

Федеральное государственное автономное образовательное учреждение  
высшего образования «Национальный исследовательский университет  
ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа  
по ТФКП

**Построение и визуализация  
фрактальных множеств**

Выполнили:  
студенты потока 22.4

*Шумахер Демид Сергеевич  
Михайлов Петр Сергеевич  
Соболев Егор Викторович*

Санкт-Петербург - 2025

# Оглавление

1	Задание	3
2	Доказательство первого свойства множества Мандельброта	4
3	Доказательство второго свойства множества Мандельброта	5
4	Код функции для построения множеств Мандельброта	6
5	Изображения множества Мандельброта с разными параметрами	7
6	Код функции для построения множеств Жюлиа	9
7	Изображения множества Жюлиа с разными параметрами	10
8	Фрактал «Горящий корабль»	12

# 1 Задание

1. Докажите свойства 1 и 2 для множества Мандельброта.
2. Напишите программу, которая будет строить визуализацию множества Мандельброта. Выберите разумные ограничения, поварьируйте максимальное количество итераций. Попробуйте приблизить отдельные части множества, чтобы увидеть фрактальную структуру.
3. Напишите программу, которая по заданному  $c$  строит заполненное множество Жюлиа. Поварьируйте максимальное количество итераций, попробуйте проанализировать фрактальную структуру, рассмотрите множество при разных  $c$ . (Например, красиво получается при  $c = 0.5251993 + i \cdot 0.5251993$ ).
4. Найдите какой-нибудь неразобранный фрактал (например, бассейны Ньютона). Опишите его структуру, построение. Нарисуйте визуализации. Будьте готовы выступить с докладом перед своими одногруппниками.

## 2 Доказательство первого свойства множества Мандельброта

1.  $M = \{c \in \mathbb{C} \mid \exists M \in \mathbb{R} : \forall n \in \mathbb{N} |z_{n+1} - z_n|^2 + c| \leq M\}$  множество Мандельброта,  $z_n$  — последовательность комплексных чисел,  $z_0 = 0$ .  $M$  — конечное вещественное число.
2.  $\bar{c}$  — сопряжение  $c$ ,  $z'_n$  — последовательность с нашим  $\bar{c}$ .
3. Мы хотим доказать, что для любого  $c \in Q$  комплексное сопряжение этого  $c$  тоже принадлежит  $Q$ .
4. По определению множества,  $\bar{c} \in Q \Leftrightarrow |z'_{n+1} - z'_n|^2 + \bar{c}| < M$ .
5. Докажем, что  $z'_n = \overline{z_n}$  (все элементы последовательности  $z'_n$  это сопряжённые элементы  $z_n$ ), по индукции:

**База:**  $z_0 = z'_0 = 0$ .

**Переход:** пусть  $z_n = x + iy$ ,  $c = a + ib$ . Тогда

$$\begin{aligned} z'_{n+1} &= z'^2_n + \bar{c} = (x - iy)^2 + (a - ib) \\ &= x^2 - y^2 - i2xy + a - ib \\ &= (x^2 - y^2 + a) - i(2xy + b) \\ &= \overline{(x^2 - y^2 + a) + i(2xy + b)} \\ &= \overline{(z_n^2 + c)} = \overline{z_{n+1}} \end{aligned}$$

**Вывод:**  $z'_n = \overline{z_n} \Rightarrow |z'_n| = |\overline{z_n}| = |z_n| \Rightarrow$  если  $c \in Q$  то  $|z_n| < M$  и тогда для  $\bar{c}$   $|\overline{z_n}| < M$  значит  $\bar{c} \in Q$ .

### 3 Доказательство второго свойства множества Мандельброта

**Определение:** Точка  $c$  принадлежит множеству Мандельброта тогда и только тогда, когда рекуррентная последовательность  $z_{n+1} = z_n^2 + c$ , начинающаяся с  $z_0 = 0$ , остается ограниченной (то есть, существует такое  $M$ , что для всех  $n$  выполняется  $|z_n| < M$ ).

**Цель:** Доказать, что если  $|c| > 2$ , то последовательность  $z_n$  не ограничена, а значит,  $c$  не принадлежит множеству.

**Шаг 1:** Докажем, что последовательность  $|z_n|$  монотонно возрастает, начиная с  $n = 1$ , и все ее члены больше 2. Будем доказывать по индукции.

**База индукции ( $n = 1$ ):**  $z_0 = 0$   $z_1 = z_0^2 + c = c$ . Следовательно,  $|z_1| = |c| > 2$  (по условию).  $z_2 = z_1^2 + c = c^2 + c$ . Проанализируем  $|z_2|$ . По неравенству треугольника для модуля ( $|A + B| \geq |A| - |B|$ ), получаем:  $|z_2| = |c^2 + c| \geq |c^2| - |c| = |c|^2 - |c|$ . Выносим  $|c|$  за скобку:  $|z_2| \geq |c|(|c| - 1)$ . Так как  $|c| > 2$ , то  $(|c| - 1) > 1$ . Это означает, что  $|z_2| > |c| \cdot 1 = |c| = |z_1|$ . Более того, так как  $|z_1| > 2$ , то и  $|z_2| > 2$ .

**База индукции доказана:**  $|z_2| > |z_1| > 2$ .

**Индукционное предположение:** Пусть для некоторого  $n \geq 2$  верно, что  $|z_n| > |z_{n-1}| > 2$ .

**Индукционный переход:** ( $n \rightarrow n + 1$ ) Рассмотрим  $|z_{n+1}| = |z_n^2 + c|$ . Снова применим неравенство треугольника:  $|z_{n+1}| = |z_n^2 + c| \geq |z_n^2| - |c| = |z_n|^2 - |c|$ . По индукционному предположению,  $|z_n| > 2$  и  $|z_n| > |c|$  (так как  $|z_n| > \dots > |z_1| = |c|$ ).

Вынесем  $|z_n|$  за скобку:  $|z_{n+1}| \geq |z_n|^2 - |c| = |z_n|(|z_n| - \frac{|c|}{|z_n|})$ . Мы знаем, что  $|z_n| > |c| \Rightarrow \frac{|c|}{|z_n|} < 1$ . Беря во внимание, что  $|z_n| > 2$ , следует, что  $(|z_n| - \frac{|c|}{|z_n|}) > (2 - 1) = 1$ .

Подставляем обратно:  $|z_{n+1}| \geq |z_n| \cdot (\text{число, большее } 1) > |z_n|$ . Кроме того, так как  $|z_n| > 2$ , то  $|z_{n+1}| > 2$ .

**Индукционный переход доказан:**  $|z_{n+1}| > |z_n| > 2$ .

**Вывод Шага 1:** Последовательность  $|z_n|$ , начиная с  $n = 1$ , является строго монотонно возрастающей, и все ее члены, начиная с  $z_1$ , строго больше 2.

**Шаг 2:** Покажем, что последовательность  $|z_n|$  не может быть ограниченной. Предположим, что  $|z_n|$  ограничена сверху, тогда по теореме Вейерштрасса у неё должен быть конечный предел  $M$ , то есть, начиная с какого-то  $n_0$ , для любого  $\epsilon$  верно  $M - \epsilon < |z_n| < M + \epsilon$ . Но как видно из доказательства монотонности, каждый следующий член больше предыдущего ... так что при  $0 < \epsilon < \frac{(|c|^2 - 2|c|)}{2}$  требование не будет выполняться, ведь если какой-то  $|z_{n_0}|$  отличается от  $M$  менее, чем на  $\epsilon$ , то  $|z_{n_0+1}|$  будет гарантированно больше  $M + \epsilon$ .

**Вывод Шага 2:** Получено противоречие при предположении об ограниченности последовательности.

**Заключение:** Так как предположение об ограниченности последовательности  $z_n$  при  $|c| > 2$  привело нас к противоречию, это предположение ложно. Следовательно, последовательность является неограниченной, и точка  $c$  не принадлежит множеству Мандельброта. Ч. т. д.

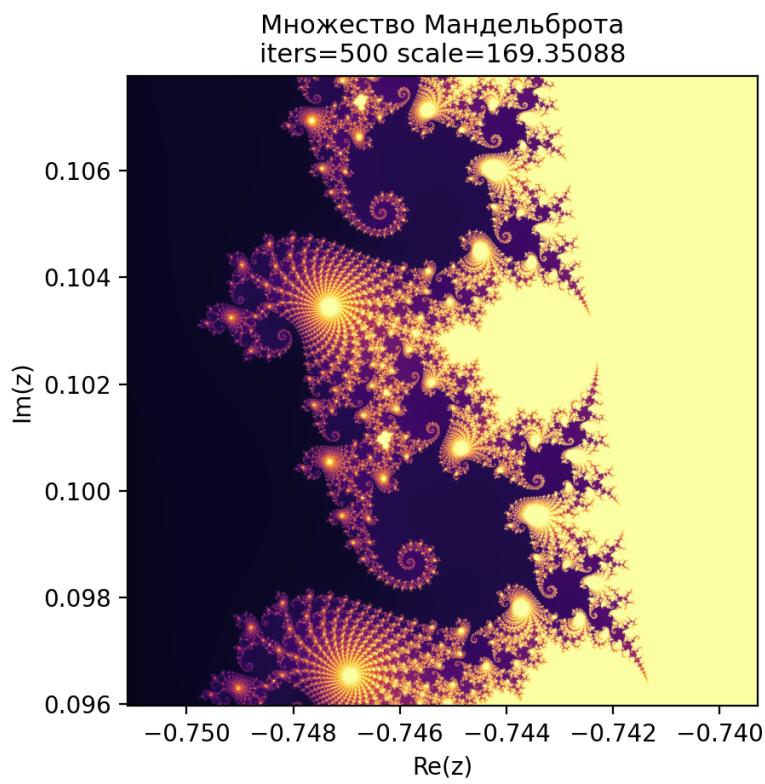
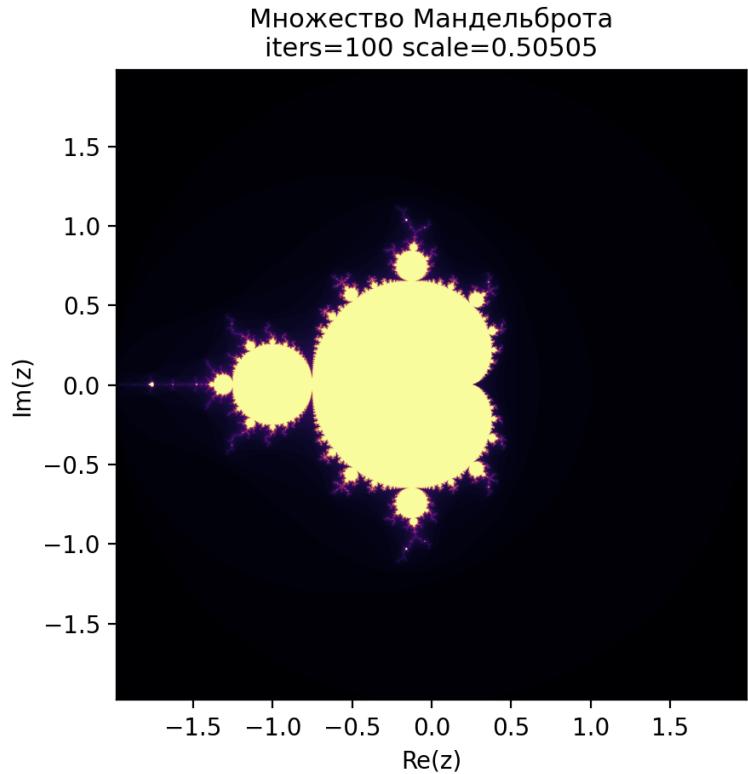
## 4 Код функции для построения множеств Мандельброта

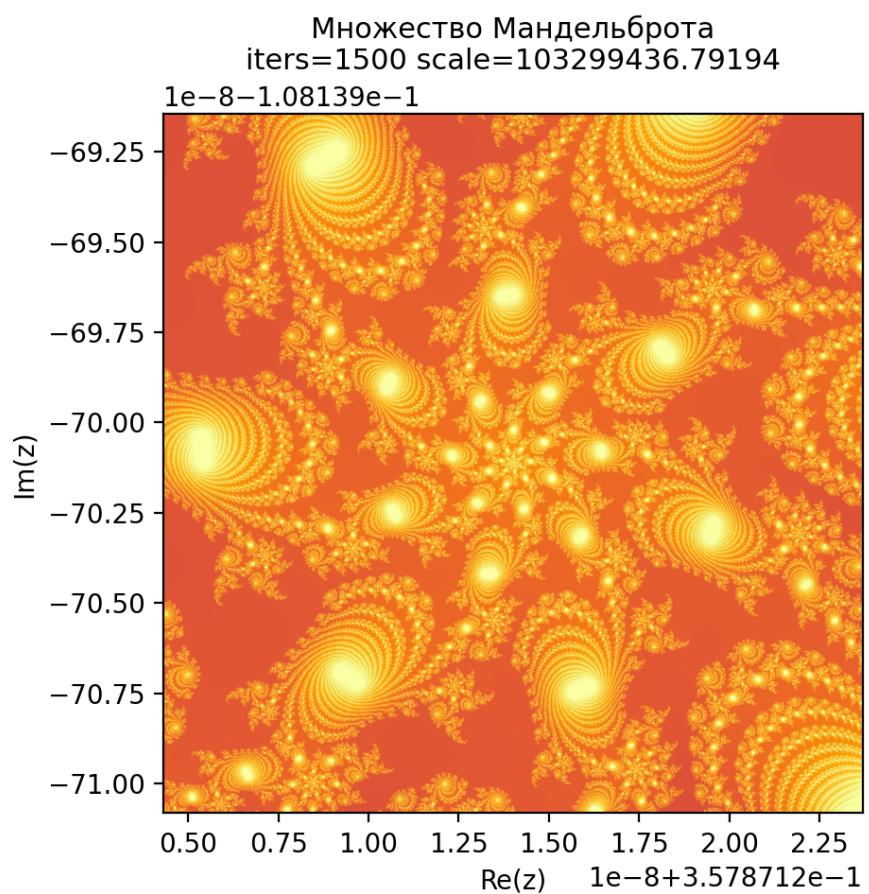
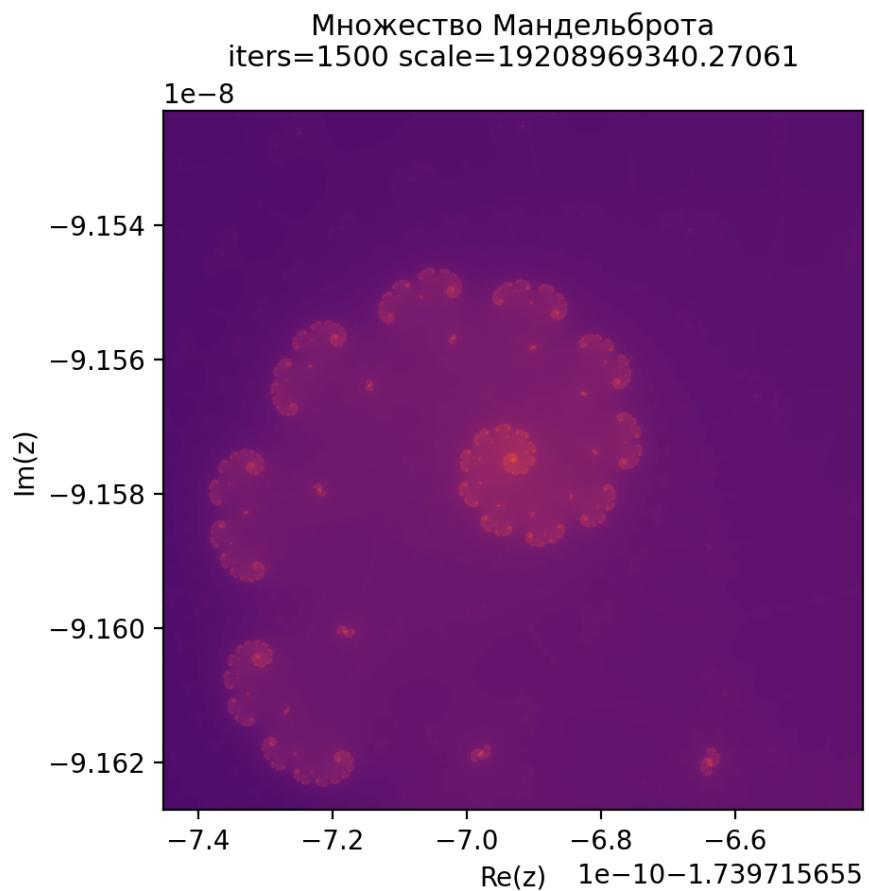
```
1 def mandelbrot_matrix(fractal_params, iterations, borders, density):
2     z0 = fractal_params.get("z0")
3     r = fractal_params.get("r")
4
5     plane_start, plane_end = borders
6
7     # Вектор действительных частей комплексной плоскости
8     re_vec = np.linspace(plane_start.real, plane_end.real, density,
9                          dtype=np.float32)
10
11    # Вектор мнимых частей комплексной плоскости
12    im_vec = np.linspace(plane_start.imag, plane_end.imag, density,
13                          dtype=np.float32) * 1j
14
15    # Формируем дискретную комплексную плоскость точек C
16    # с помощью декартового сложения двух векторов
17    # В итоговой плоскости получаем density^2 точек
18    plane = np.add.outer(im_vec, re_vec)
19
20    # Результирующая матрица для вывода и окраски
21    output = np.zeros(plane.shape, dtype='uint16')
22
23    # Промежуточная матрица из комплексных чисел для формирования
24    # последовательности, заполненная z0
25    # В ней будем хранить п-ые члены последовательности для всех
26    # подходящий точек с (plane)
27    z = np.full(plane.shape, z0, np.complex64)
28
29    for i in range(iterations):
30        # Матрица из точек, в которых текущее z еще не "убежало"
31        mask = np.less(np.abs(z), r)
32        # Чем больше чисел попало в точку, тем большее число для
33        # данной точке будет в результирующей матрице
34        output[mask] = i
35
36        # Формируем следующий член последовательности для чисел,
37        # которые отвечают радиусу
38        # z_{n+1} = z_n^2 + c
39        z[mask] = z[mask]**2 + plane[mask]
40
41    return output
```

Листинг 1: Функция для построения множества Мандельброта

Полный код вы можете увидеть на [Github](#).

## 5 Изображения множества Мандельброта с разными параметрами





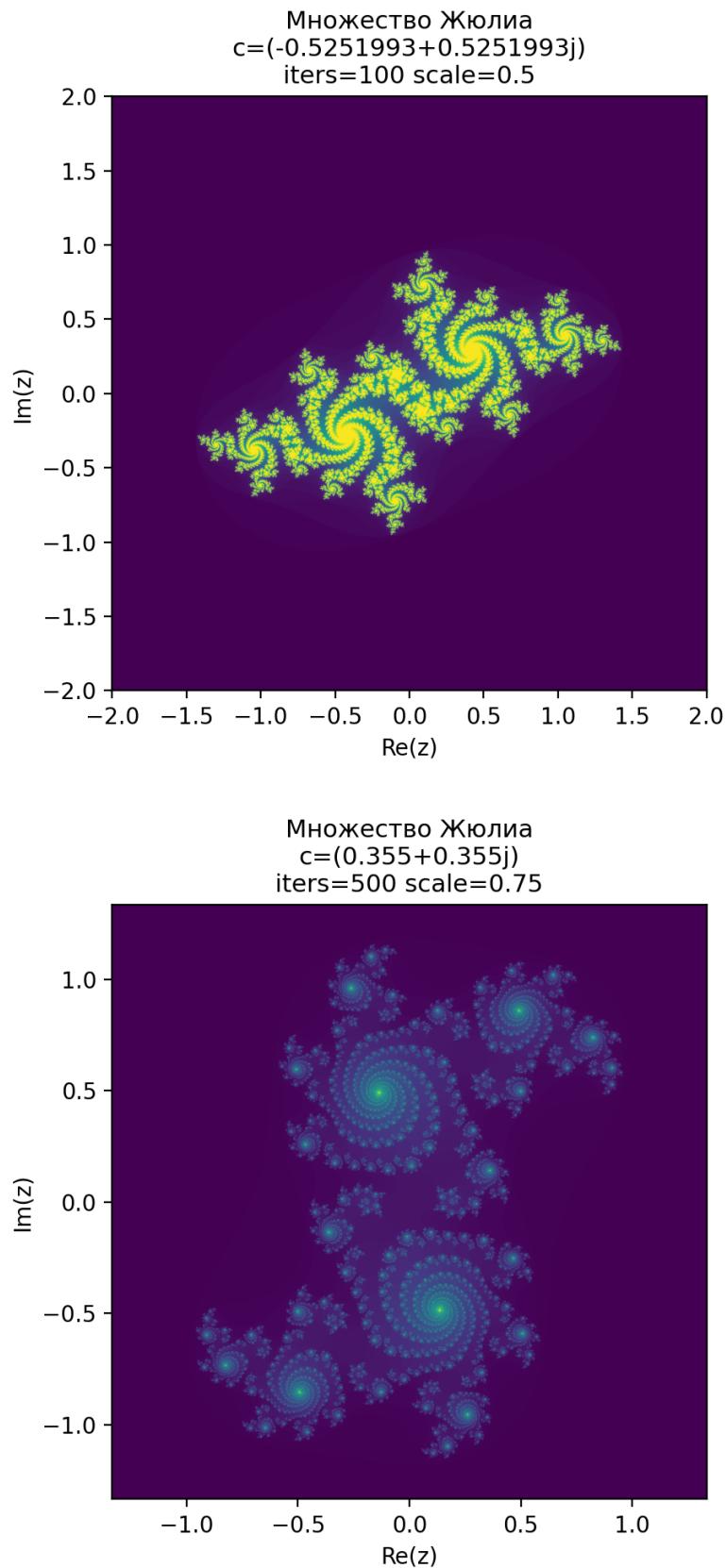
## 6 Код функции для построения множеств Жюлиа

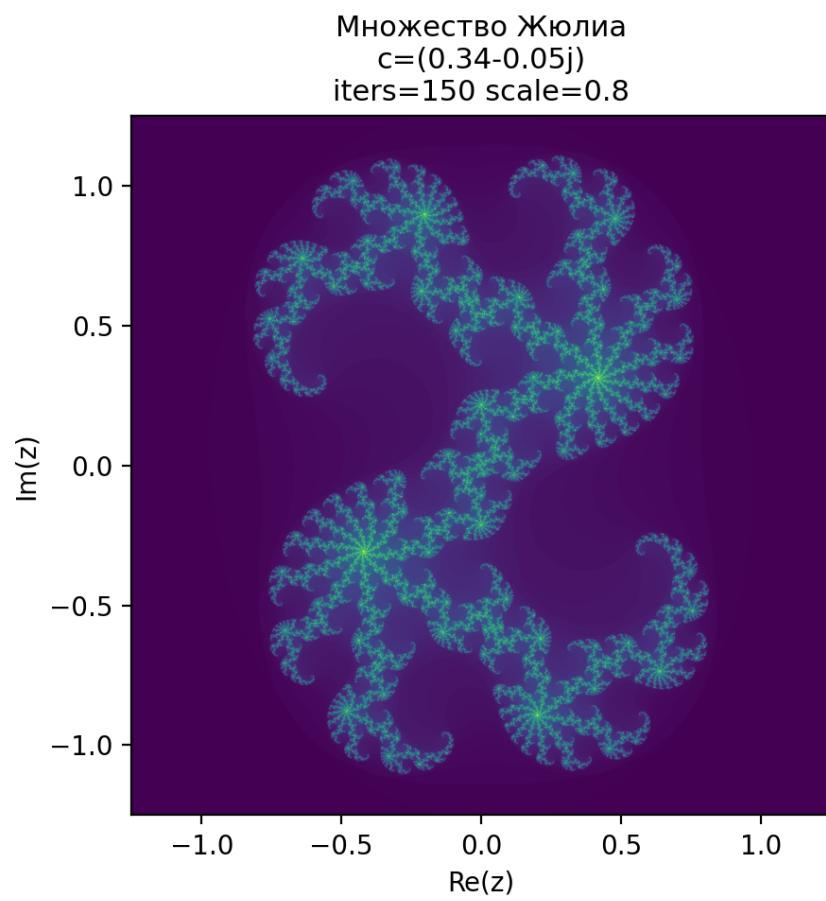
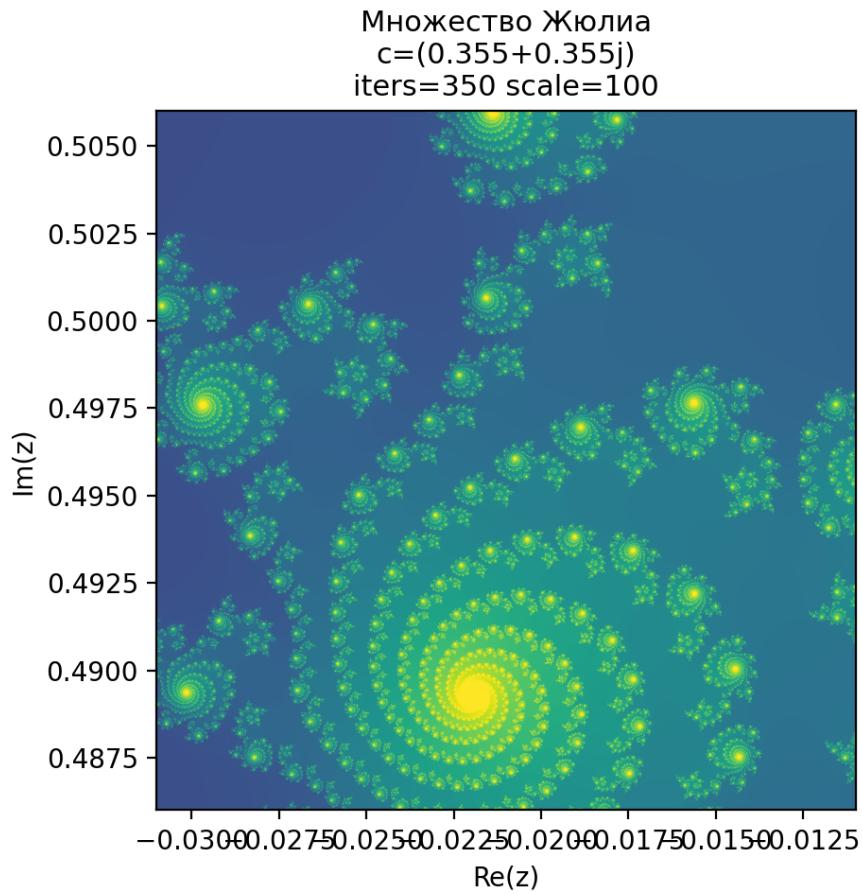
```
1 def julia_matrix(fractal_params, iterations, borders, density):
2     c = fractal_params.get("c")
3     r = fractal_params.get("r")
4     plane_start, plane_end = borders
5
6     # Выбор оптимального r
7     # Имеем:  $r^2 - r \geq |c|$ 
8     #  $r \geq (1 + \sqrt{1 + 4|c|})/2 + \text{eps}$ 
9     # Если передано значение r меньшее, чем оптимальное - оставляем как есть для иссле-
10    # дования области с меньшим r
11    # Иначе рассчитываем оптимальное значение, так как исследование области с большим
12    # r не имеет смысла
13    r = min(r, (1 + np.sqrt(1+4*np.abs(c)))/2 + 0.01)
14
15    # Вектор действительных частей комплексной плоскости
16    re_vec = np.linspace(plane_start.real, plane_end.real, density, dtype=np.float64)
17    # Вектор мнимых частей комплексной плоскости
18    im_vec = np.linspace(plane_start.imag, plane_end.imag, density, dtype=np.float64)
19    * 1j
20
21    # Формируем дискретную комплексную плоскость с помощью декартового сложения двух
22    # векторов
23    # В итоговой плоскости получаем density^2 точек
24    z = np.add.outer(im_vec, re_vec)
25
26    # Результирующая матрица для вывода и окраски
27    output = np.zeros(z.shape, dtype='uint16')
28
29    for i in range(iterations):
30        # Матрица из точек, в которых текущее z еще не "убежало"
31        mask = np.less(np.abs(z), r)
32        # Чем больше чисел попало в точку, тем большее число для данной точке будет в
33        # результирующей матрице
34        output[mask] = i
35        # Формируем следующий член последовательности для чисел, которые отвечают радиусу
36        #  $z_{n+1} = z_n^2 + c$  (const)
37        # Аналогично первому фракталу, но используем c как const и применяем функцию
38        # к каждой точке z
39        z[mask] = z[mask] ** 2 + c
40
41    return output
```

Листинг 2: Функция для построения множества Жюлиа

Полный код вы можете увидеть на [Github](#).

## 7 Изображения множества Жюлиа с разными параметрами





## 8 Фрактал «Горящий корабль»

В качестве неразобранного фрактала был выбран «Горящий корабль». Он был открыт Майклом Михеличем в 1992 году и является вариантом множества Мандельброта. Его структура отличается большей хаотичностью из-за небольшого изменения в итерационной формуле.

### Множество «Горящий корабль»

Рассмотрим последовательность комплексных чисел, заданную следующим образом:

$$z_{n+1} = (|\operatorname{Re}(z_n)| + i|\operatorname{Im}(z_n)|)^2 + c, \quad \text{при } z_0 = 0$$

Ключевое отличие от формулы для множества Мандельброта ( $z_{n+1} = z_n^2 + c$ ) состоит в том, что перед возведением в квадрат на каждой итерации берутся абсолютные значения действительной и мнимой частей комплексного числа  $z_n$ .

**Определение.** Множество всех  $c \in \mathbb{C}$ , при которых заданная выше последовательность  $z_n$  остается ограниченной, называется **множеством «Горящий корабль»**.

### Алгоритм построения

Алгоритм практически идентичен алгоритму построения множества Мандельброта:

1. Берется ограниченная часть комплексной плоскости (например, от  $-2 - 2i$  до  $2 + 2i$ ), которая разбивается равномерной сеткой. Каждый узел сетки соответствует пикслю на изображении и представляет собой комплексное число  $c$ .
2. Для каждой точки  $c$  запускается итеративный процесс, начиная с  $z_0 = 0$ . Вводится ограничение на максимальное число итераций  $M$  и условие «убегания»: если модуль  $|z_n|$  превосходит 2, итерации прекращаются.
3. Если после  $M$  итераций последовательность не «убежала», считается, что точка  $c$  принадлежит множеству, и соответствующий пиксель закрашивается в черный цвет.
4. Для получения цветного изображения пиксели, не принадлежащие множеству, окрашиваются в цвет, зависящий от номера итерации, на которой произошло «убегание».

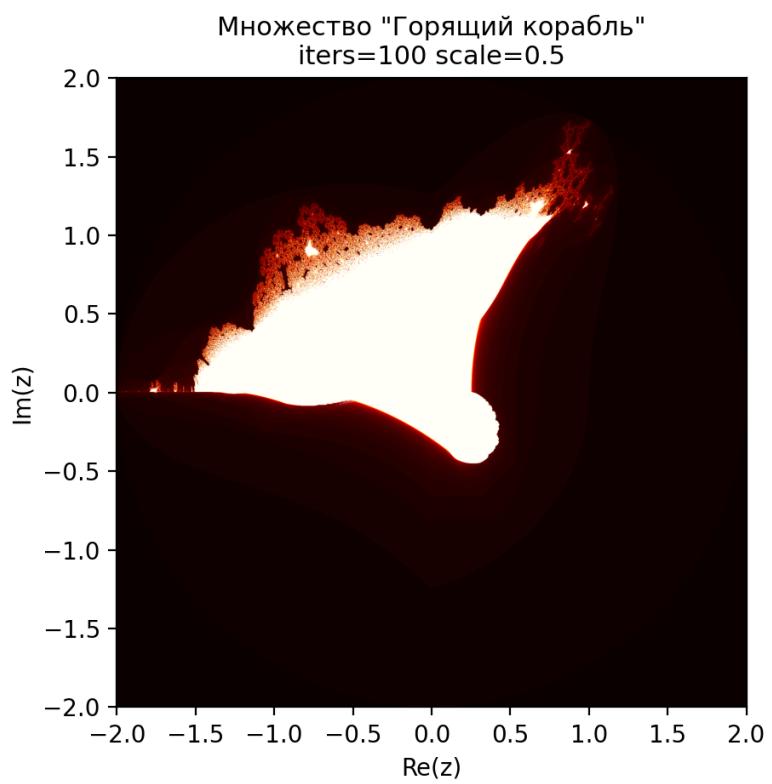
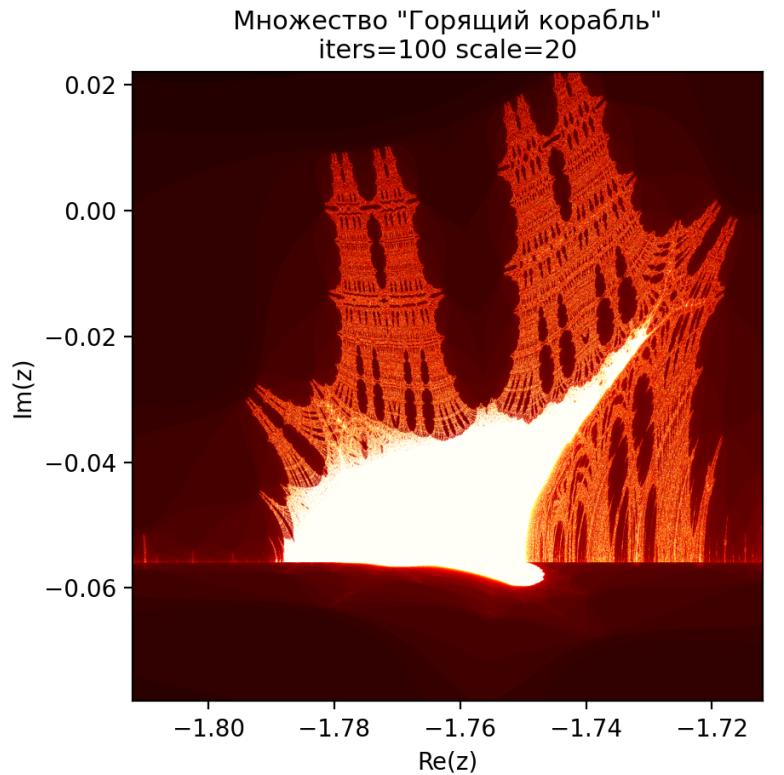
```

1 def burning_ship_matrix(fractal_params, iterations, borders, density):
2     z0 = fractal_params.get("z0")
3     r = fractal_params.get("r")
4
5     plane_start, plane_end = borders
6
7     # Вектор действительных частей
8     re_vec = np.linspace(plane_start.real, plane_end.real, density,
9                           dtype=np.float64)
10
11    # Вектор мнимых частей
12    im_vec = np.linspace(plane_start.imag, plane_end.imag, density,
13                          dtype=np.float64) * 1j
14
15    # Плоскость точек C
16    plane = np.add.outer(im_vec, re_vec)
17
18    # Результирующая матрица
19    output = np.zeros(plane.shape, dtype='uint16')
20
21    # Промежуточная матрица, заполненная z0
22    z = np.full(plane.shape, z0, np.complex128)
23
24    for i in range(iterations):
25        # Мaska точек, которые не "убежали"
26        mask = np.less(np.abs(z), r)
27
28        output[mask] = i
29
30        # z_{n+1} = (|Re(z)| + i|Im(z)|)^2 + c
31        z[mask] = (np.abs(z[mask].real) + 1j * np.abs(z[mask].imag))**2 + plane[mask]
32
33    return output

```

Листинг 3: Функция для построения множества "Горящий корабль"

Полный код вы можете увидеть на [Github](#).



Множество "Горящий корабль"  
iters=250 scale=250000

