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

Дисциплина «Теория функций комплексного переменного»

# Отчёт

## по лабораторной работе №1

Построение и визуализация фрактальных множеств

Выполнили:
Гаврилин Олег Сергеевич
Зорин Георгий Юрьевич
Матевосян Артур Русланович

Группа по ТФКП: 21.5

Преподаватель:
Шаврин Андрей Андреевич

Санкт-Петербург 2024г.

# Доказательства свойств для множества Мандельброта.

1. **Симметрия относительно вещественной оси.**
   **Определение множества Мандельброта:**

   Множество Мандельброта $M$ состоит из всех комплексных чисел $c$, для которых последовательность, определённая как:

   $$z_{n+1} = z_n^2 + c, \quad z_0 = 0,$$

   остаётся ограниченной, то есть не уходит на бесконечность при $n \to \infty$.

   **Доказательство симметрии:**

   (a) **Начало последовательности:** Для любого $c$, начальное значение последовательности $z_0$ равно 0. Заметим, что сопряжённое число от 0 равно самому себе:

   $$\overline{z_0} = \overline{0} = 0.$$

   (b) **Сопряжение и рекуррентное соотношение:** Пусть последовательность $z_n$ удовлетворяет рекуррентному соотношению $z_{n+1} = z_n^2 + c$. Для сопряжённого числа $\overline{c}$ аналогичная последовательность будет:

   $$\overline{z_{n+1}} = (\overline{z_n})^2 + \overline{c}.$$

   То есть последовательность для $\overline{c}$ образуется из последовательности $z_n$ сопряжением каждого элемента.

   (c) **Ограниченность и симметрия:** Множество Мандельброта $M$ состоит из чисел $c$, для которых последовательность $z_n$ ограничена. Если последовательность $z_n$ ограничена для $c$, то последовательность $\overline{z_n}$ также ограничена для $\overline{c}$, так как операция сопряжения не влияет на величину комплексных чисел:

   $$|z_n| = |\overline{z_n}|.$$

   (d) **Вывод:** Если $c \in M$, то $\overline{c} \in M$. Это означает, что множество Мандельброта $M$ симметрично относительно вещественной оси.

2. **Если $|c| > 2$, то $c \notin$ множеству Мандельброта.**

   **Доказательство:**

   (a) Рассмотрим первое значение последовательности:

   $$z_1 = z_0^2 + c = c.$$

   Тогда $|z_1| = |c|$. Если $|c| > 2$, то уже на первом шаге $|z_1| > 2$.

   (b) На следующем шаге:

   $$z_2 = z_1^2 + c = c^2 + c.$$

   Тогда:

   $$|z_2| \geq |z_1|^2 - |c| = |c|^2 - |c| > 2^2 - 2 = 2,$$

   так как $|c| > 2$.

   (c) В общем случае можно показать по индукции, что если $|z_n| > 2$, то:

   $$|z_{n+1}| = |z_n^2 + c| \geq |z_n|^2 - |c|.$$

   Поскольку $|z_n| > 2$ и $|c| > 2$, значение $|z_{n+1}|$ будет ещё больше.

   Таким образом, если $|c| > 2$, то последовательность $z_n$ не остаётся ограниченной, и $c \notin M$. Это доказывает свойство.

# Визуализация множества Мандельброта

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

width, height = 800, 800
x_min, x_max = -2.5, 1.5
y_min, y_max = -2.0, 2.0
max_iter = 100

def mandelbrot(c, max_iter):

    z = 0
    for n in range(max_iter):
        z = z * z + c
        if abs(z) > 2:
            return n
    return max_iter

x, y = np.linspace(x_min, x_max, width), np.linspace(y_min, y_max, height)
X, Y = np.meshgrid(x, y)
C = X + 1j * Y

Z = np.zeros(C.shape, dtype=int)
for i in range(height):
    for j in range(width):
        Z[i, j] = mandelbrot(C[i, j], max_iter)


plt.figure(figsize=(10, 10))
plt.imshow(Z, extent=[x_min, x_max, y_min, y_max], cmap='hot', origin='lower')
plt.colorbar(label='Количество итераций до расходимости')
plt.title("Множество Мандельброта")
plt.xlabel("Re(c)")
plt.ylabel("Im(c)")
plt.show()
```

**Набор изображений при разном числе итераций и приближений.**

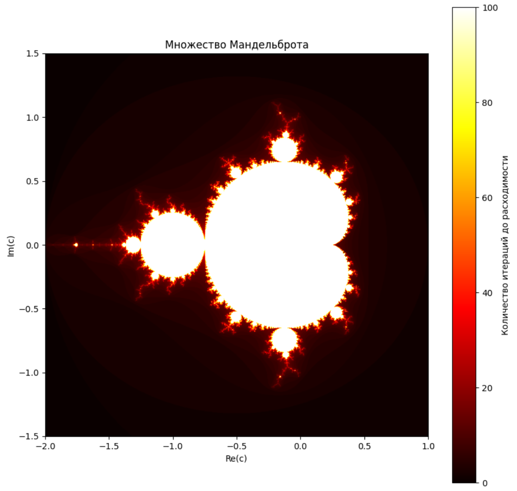


Рис. 1: Mandelbrot: 100 iterations, Zoom 1

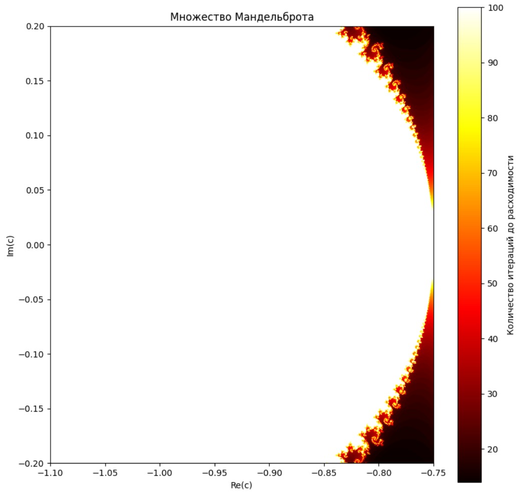


Рис. 2: Mandelbrot: 100 iterations, Zoom 2

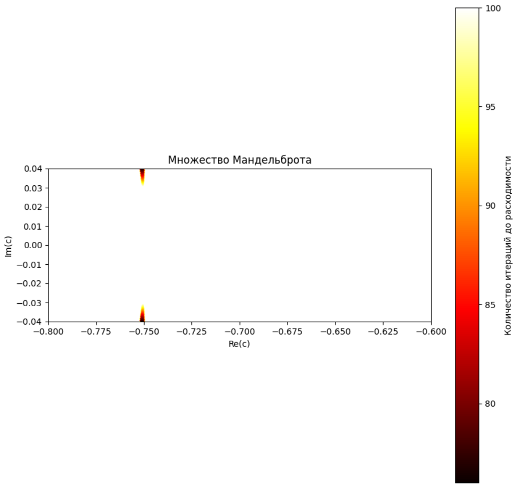


Рис. 3: Mandelbrot: 100 iterations, Zoom 3

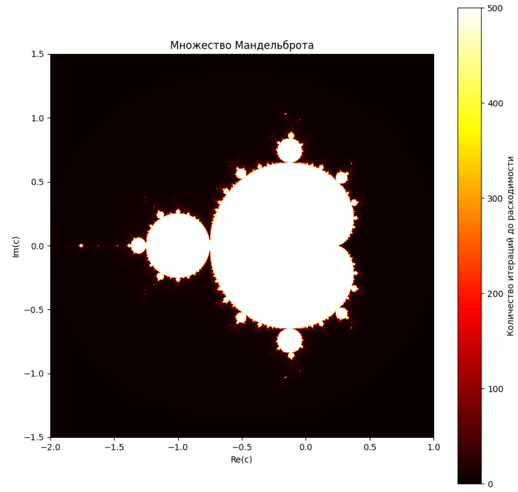


Рис. 4: Mandelbrot: 500 iterations, Zoom 1

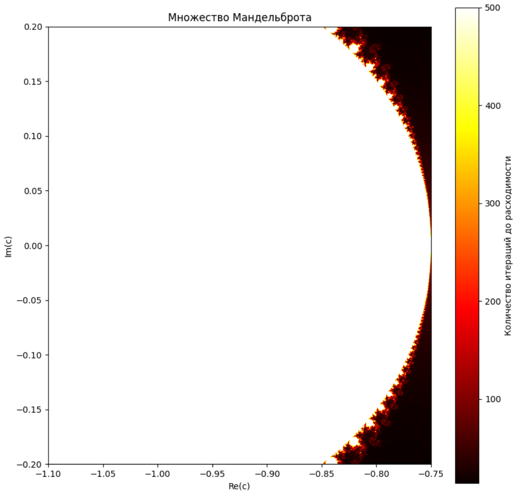


Рис. 5: Mandelbrot: 500 iterations, Zoom 2

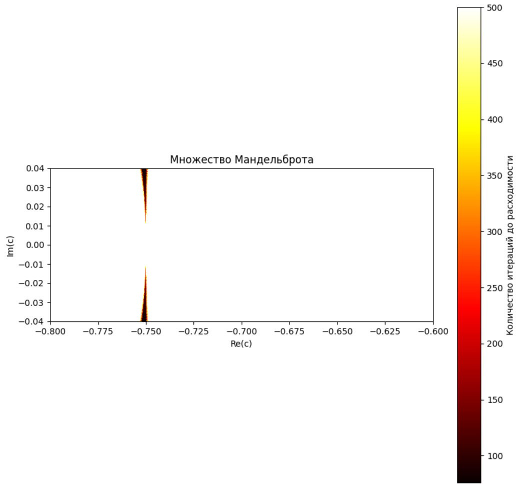


Рис. 6: Mandelbrot: 500 iterations, Zoom 3

# Визуализация множества Жюлиа

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

width, height = 800, 800
x_min, x_max = -2.0, 2.0
y_min, y_max = -2.0, 2.0
max_iter = 500
c = complex(-0.5251993, 0.5251993)

x = np.linspace(x_min, x_max, width)
y = np.linspace(y_min, y_max, height)
X, Y = np.meshgrid(x, y)
Z = X + 1j * Y

iterations = np.zeros(Z.shape, dtype=int)

mask = np.full(Z.shape, True, dtype=bool)
for i in range(max_iter):
    Z[mask] = Z[mask] ** 2 + c
    mask[np.abs(Z) > 2] = False
    iterations[mask] = i

plt.figure(figsize=(8, 8))
plt.imshow(iterations, extent=(x_min, x_max, y_min, y_max), cmap="magma")
plt.colorbar(label="Iterations to escape")
plt.title("Julia Set")
plt.xlabel("Re(z)")
plt.ylabel("Im(z)")
plt.show()
```

**Набор изображений при разном числе итераций и приближений.**

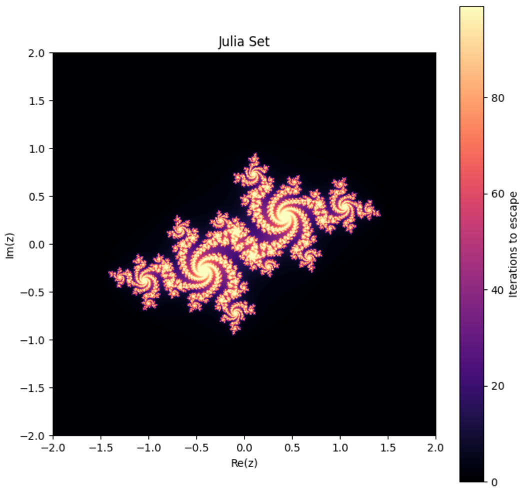


Рис. 7: Julia: 100 iterations, Zoom 1

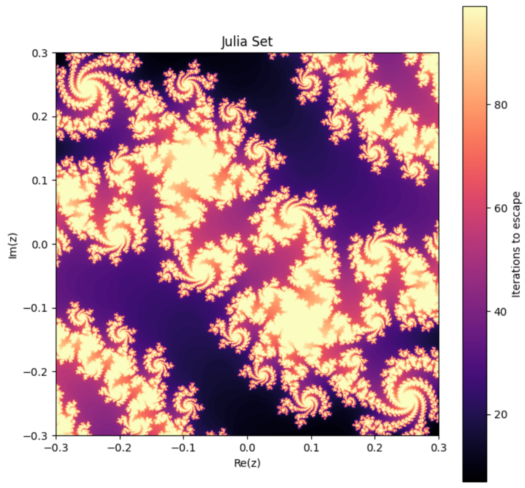


Рис. 8: Julia: 100 iterations, Zoom 2

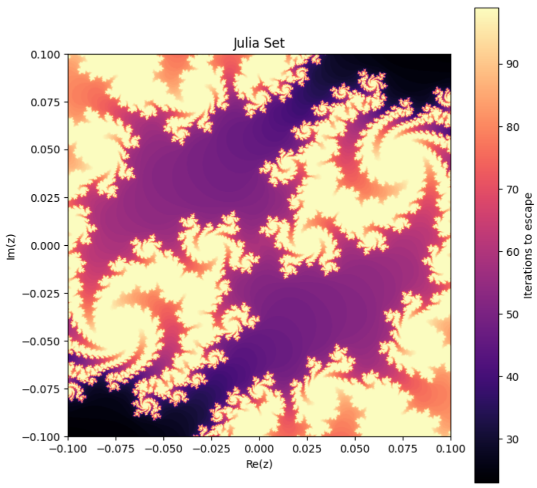


Рис. 9: Julia: 100 iterations, Zoom 3

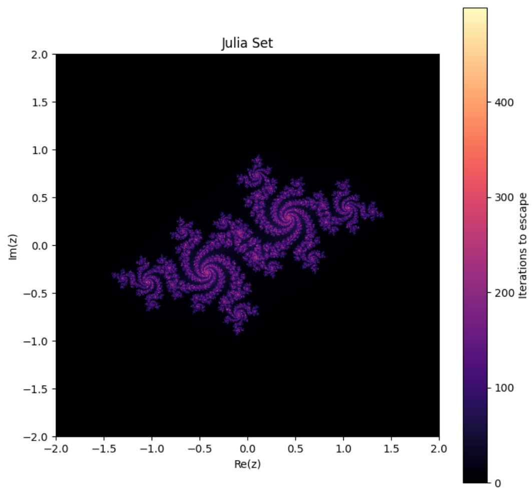


Рис. 10: Julia: 500 iterations, Zoom 1

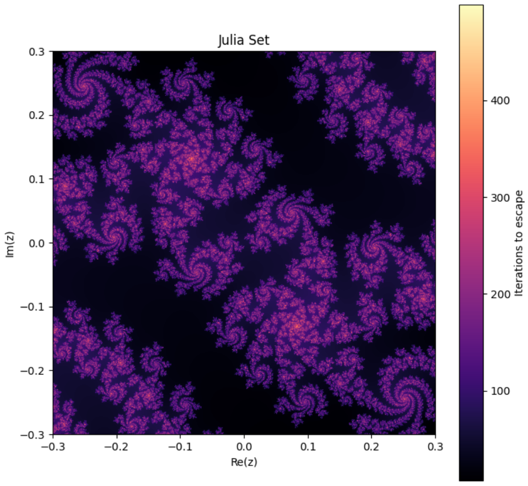


Рис. 11: Julia: 500 iterations, Zoom 2

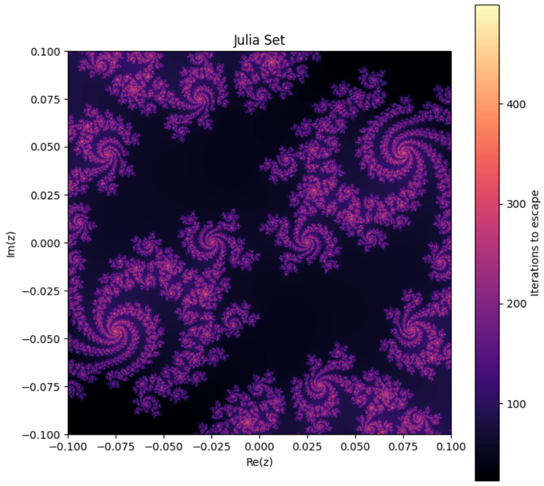


Рис. 12: Julia: 500 iterations, Zoom 3

# Неразобранный фрактал - дракон Хартера-Хейтуэя

## Описание

Дракон Хартера-Хейтуэя (Harter–Heighway Dragon) — это фрактал, который создаётся с помощью рекурсивных изгибов линии. Он получил известность благодаря тому, что стал основой для визуальных иллюстраций в математике и информатике.

## Алгоритм построения

1. Начинаем с прямой линии.
2. На каждом шаге делим линию пополам.
3. Добавляем угол 90°, чтобы новая половина изгибалась по часовой или против часовой стрелки.
4. Повторяем процесс для всех новых отрезков.

Каждая итерация создаёт всё более сложную и самоподобную структуру.

## Свойства

- **Самоподобие**: Фрактал состоит из частей, каждая из которых подобна всей структуре.
- **Фрактальная размерность**: Размерность дракона Хартера-Хейтуэя составляет:

$$D = \frac{\log(2)}{\log(\sqrt{2})} = 2.$$

- **Геометрия**: Дракон обладает строгой симметрией и, несмотря на сложность формы, его легко описать итеративно.

## Применение

Дракон Хартера-Хейтуэя используется в компьютерной графике и для визуализации математических понятий, таких как самоподобие и рекурсия. Его можно найти в художественных работах, научных визуализациях и образовательных проектах.

**Визуализация фрактала**

```python
import matplotlib.pyplot as plt

def dragon_curve(iterations):
    points = [0 + 0j, 1 + 0j]
    for _ in range(iterations):
        new_points = []
        for i in range(len(points) - 1):
            start, end = points[i], points[i + 1]

            mid = (start + end) / 2

            diff = end - start
```

```
            rotated = mid + diff * 1j / 2

            new_points.extend([start, rotated])
        new_points.append(points[-1])
        points = new_points

    return points

iterations = 30
points = dragon_curve(iterations)

x = [p.real for p in points]
y = [p.imag for p in points]

plt.figure(figsize=(10, 10))
plt.plot(x, y, color='blue', linewidth=0.8)
plt.title(f'Дракон Хартера-Хейтуэя (итерации = {iterations})')
plt.axis('equal')
plt.grid(True)
plt.show()
```

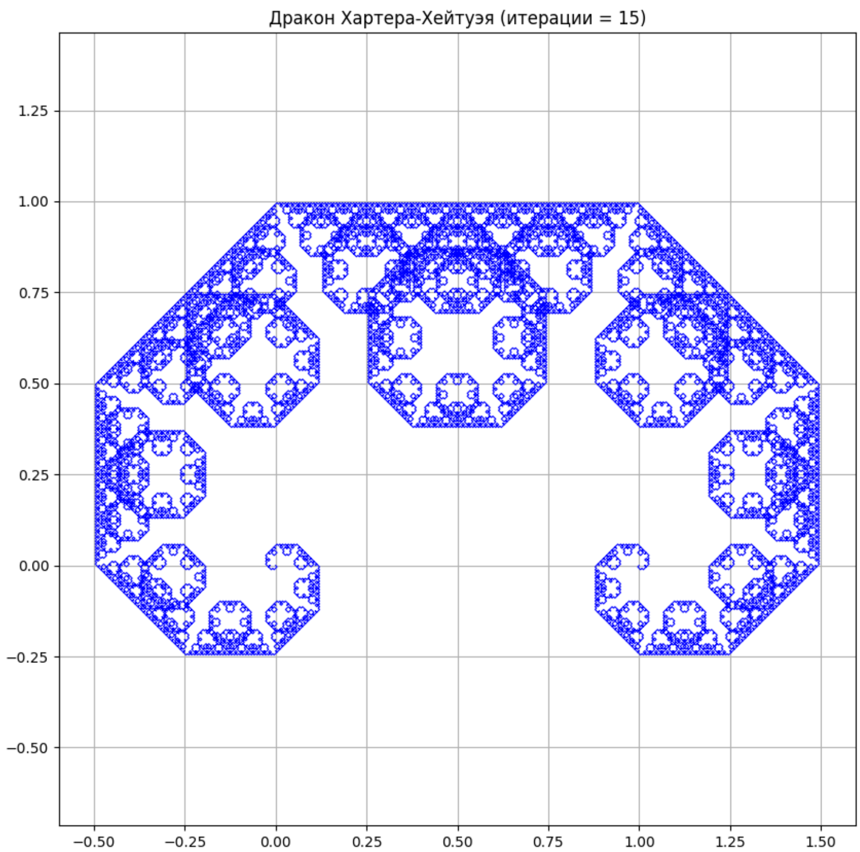


Рис. 13: Реализация фрактала Дракон Хартера-Хейтуэя.