ФГАОУ ВО «НИУ ИТМО»
Факультет программной инженерии
и компьютерной техники

Лабораторная работа

Построение и визуализация фрактальных множеств

(по дисциплине «Теория функции комплексного переменного»)

Выполнили студенты:

Горин Семён, 465592
Лабин Макар, 466449
Пивоваров Роман, 467082
Поток: 22.4

Проверил:

Поздняков Семён Сергеевич

г. Санкт-Петербург, Россия
2025

1 Постановка задач

В ходе лабораторной работы будут выполнены следующие задачи:

1. Доказать свойства множества Мандельброта.

2. Написать программу, которая строит визуализацию множества Мандельброта.

3. Написать программу, которая по заданному c строит заполненное множество Жюлиа.

4. Описать структуру и построение кривой Гильберта. Нарисовать ее визуализации.

2 Множество Мандельброта

2.1 Доказательство свойств

Для начала приведем определение множества Мандельброта:

Определение 1. *Множество всех $c \in \mathbb{C}$ при которых последовательность z_n, заданная как $z_{n+1} = z_n^2 + c, z_0 = 0$, остается ограниченной, называется* ***множеством Мандельброта****.*

Докажем первое свойство:

Свойство 1. *Множество Мандельброта переходит само в себя при сопряжении. Иными словами, оно симметрично относительно вещественной оси.*

Доказательство. Обозначим за M множество Мандельброта. Пусть $c \in M$. Рассмотрим последовательность z_n для этого c. Возьмем комплексно-сопряженное каждого члена последовательности: $w_n = \overline{z_n}$. Тогда:

$$w_0 = \overline{z_0} = \overline{0} = 0,$$

и

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

Также $\forall n : |w_n| = |\overline{z_n}| = |z_n|$, а значит, так как по определению последовательность z_n ограничена, то и последовательность w_n ограничена. Таким образом параметр $\overline{c} \in M$, а так как $\overline{\overline{z}} = z$, то утверждение верно и в обратную сторону. То есть:

$$c \in M \Longleftrightarrow \overline{c} \in M,$$

что и означает симметричность относительно вещественной оси. □

Докажем второе свойство:

Свойство 2. *Если $|c| > 2$, то c не принадлежит множеству Мандельброта.*

Доказательство. Обозначим за M множество Мандельброта. Докажем от обратного. Пусть $\exists c : |c| > 2$ и $c \in M$. Рассмотрим такое c. Тогда так как $z_0 = 0, z_1 = z_0^2 + c = c$, откуда $|z_1| > 2$.
Рассмотрим теперь следствие из неравенства треугольника для z_n^2 и c:

$$|z_n^2 + c| \geq ||z_n^2| - |c||$$

Используем далее в доказательстве математическую индукцию. Для $n = 1$:

$$||z_1^2| - |c|| = ||c^2| - |c|| = ||c|(|c| - 1)| = |c|(|c| - 1)$$

Так как $|c| > 2$ из предположения, то:

$$|z_1^2 + c| \geq |c|(|c| - 1) > |c| \Longleftrightarrow |z_2| > |c| \Longleftrightarrow |z_2| > |z_1|$$

Это база индукции. Теперь положим, что для некоторого $n \geq 1$ выполняется:

$$|z_n| \geq |c|$$

Тогда:

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

$$|z_{n+1}| \geq |z_n|(|z_n| - 1)$$

Так как $|z_n| \geq |c| > 2$ имеем:

$$|z_{n+1}| \geq k|z_n|, \text{ где } k > 1$$

Выполним оценку снизу. Для $n > 1$ имеем: $|z_{n+1}| = k|z_n|$. Выразим через общий член последовательности: $|z_n| = k^n \cdot |c|$. Такая последовательность неограничена сверху, поэтому исходная последовательность неограничена, а значит не выполняется определение понятия множества Мандельброта. Значит наше предположение неверно и $\forall c : |c| > 2, c \notin M$. $\square$

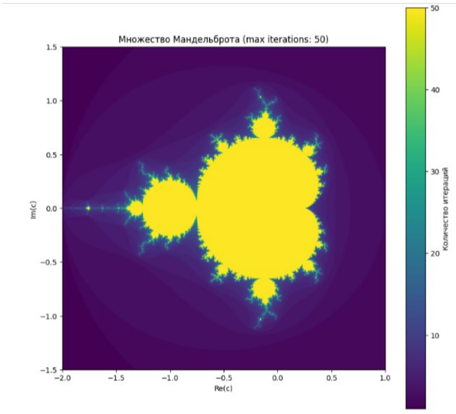
2.2 Реализация

Листинг 1: Построение множества Мандельброта

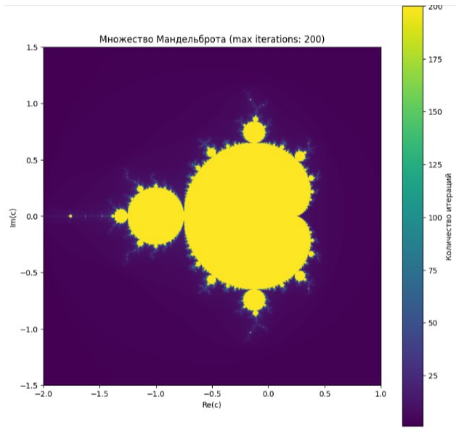
```
import numpy as np
import matplotlib.pyplot as plt

class Mandelbrot:
    def __init__(self, width=800, height=800, max_iterations=100,
                 xmin=-2.0, xmax=1.0, ymin=-1.5, ymax=1.5):
        self.width = width
        self.height = height
        self.max_iterations = max_iterations
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax

    def compute(self):
        # take a bounded part of the complex plane and construct a grid on it
        x = np.linspace(self.xmin, self.xmax, self.width)
        y = np.linspace(self.ymin, self.ymax, self.height)
        X, Y = np.meshgrid(x, y)
        C = X + 1j * Y

        # init arrays
        z = np.zeros_like(C)
        iterations = np.zeros(C.shape, dtype=int)

        # construct a sequence iteratively
        for i in range(self.max_iterations + 1):
            mask = np.abs(z) <= 2.0
            z[mask] = z[mask]**2 + C[mask]
            iterations[mask] = i

```

```python
        return iterations

    def plot(self, cmap='viridis'):
        iterations = self.compute()

        plt.figure(figsize=(10, 10))

        # create cmap
        cmap_obj = plt.cm.get_cmap(cmap)
        cmap_obj.set_under('black')

        #extent - borders, origin - position (0,0), vmin - scope of visibility for cmap
        plt.imshow(iterations,
                  extent=[self.xmin, self.xmax, self.ymin, self.ymax],
                  cmap=cmap_obj,
                  origin='lower',
                  vmin=1,
                  vmax=self.max_iterations)

        plt.colorbar(label='Количество')
        plt.xlabel('Re(c)')
        plt.ylabel('Im(c)')
        plt.title(f'Множество Мандельброта (max iterations: {self.max_iterations})')
        plt.show()

# example of a function call
mandel3 = Mandelbrot(
  xmin=-0.75, xmax=-0.65,
  ymin=0.1, ymax=0.2,
  max_iterations=300
)
mandel3.plot()
```

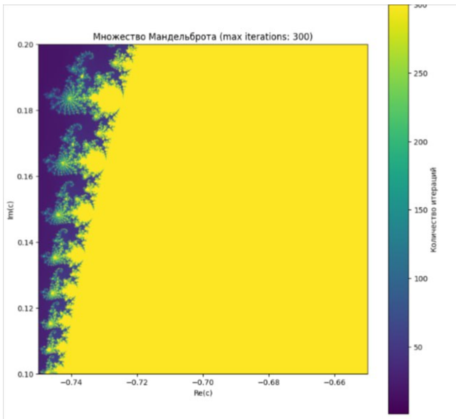
2.3 Примеры визуализации



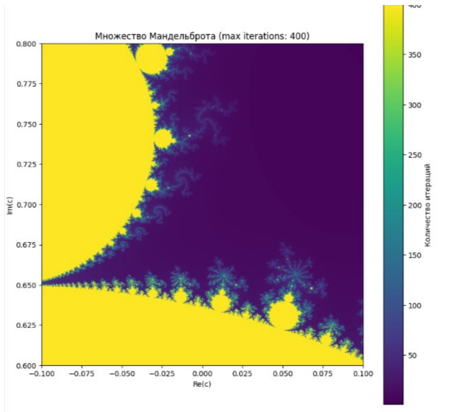
Множество Мандельброта, 50 итераций



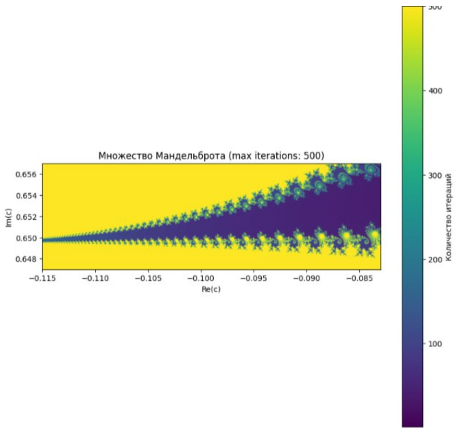
Множество Мандельброта, 200 итераций



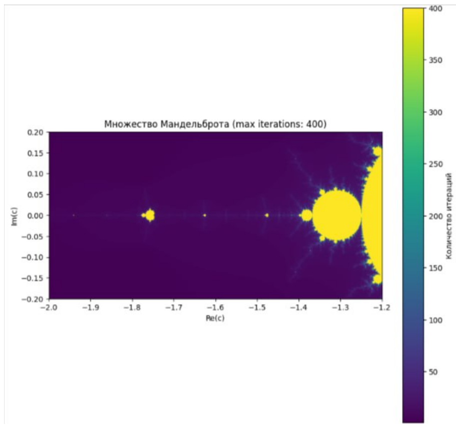
Приближение множества Мандельброта, 300 итераций



Приближение множества Мандельброта на стыке, 400 итераций



Приближение множества Мандельброта по центру слева, 400 итераций



Максимальное приближение множества Мандельброта, 500 итераций

3 Множество Жюлиа

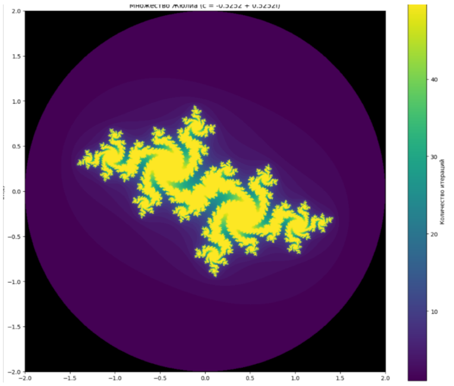
3.1 Реализация

Листинг 2: Построение множества Жюлиа

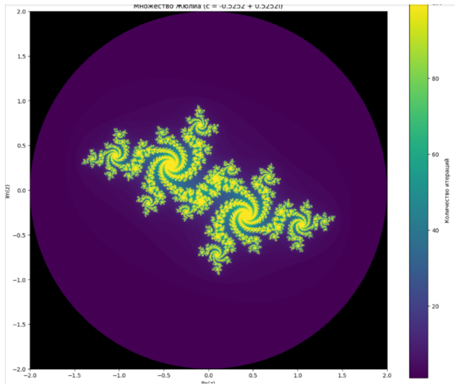
```
import numpy as np
import matplotlib.pyplot as plt

class JuliaSet:
    def __init__(self, c, width=800, height=800, max_iterations
        =100,
                 xmin=-2.0, xmax=2.0, ymin=-2.0, ymax=2.0):
        self.c = complex(c)
        self.width = width
        self.height = height
        self.max_iterations = max_iterations
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax

    def compute_julia(self):
        x = np.linspace(self.xmin, self.xmax, self.width)
        y = np.linspace(self.ymin, self.ymax, self.height)
        X, Y = np.meshgrid(x, y)
        Z = X + 1j * Y

        iterations = np.zeros(Z.shape, dtype=int)
        for i in range(1, self.max_iterations + 1):
            mask = np.abs(Z) <= 2.0
            Z[mask] = Z[mask]**2 + self.c
            iterations[mask] = i

        return iterations

    def plot(self, cmap='hot', show_info=True):
        iterations = self.compute_julia()

        plt.figure(figsize=(12, 10))

        cmap_obj = plt.cm.get_cmap(cmap)
        cmap_obj.set_under('black')

        im = plt.imshow(iterations,
                       extent=[self.xmin, self.xmax, self.ymin,
                           self.ymax],
                       cmap=cmap_obj,
                       origin='lower',
                       vmin=1,
                       vmax=self.max_iterations)

```

```
        plt.colorbar(im, label='Количество итераций')
        plt.xlabel('Re(z)')
        plt.ylabel('Im(z)')

        if show_info:
            title = f'Множество Жюлиадля  c = {self.c.real:.7f} + {self.c.imag:.7f}i\n'
            title += f'Макс. итераций: {self.max_iterations}, Разрешение: {self.width}×{self.height}'
            plt.title(title)
        else:
            plt.title(f'Множество Жюлиа (c = {self.c.real:.4f} + {self.c.imag:.4f}i)')

        plt.tight_layout()
        plt.show()
# example of a function call
julia1 = JuliaSet(c=-0.5251993 + 0.5251993j, max_iterations=200)
julia1.plot(cmap='plasma')
```

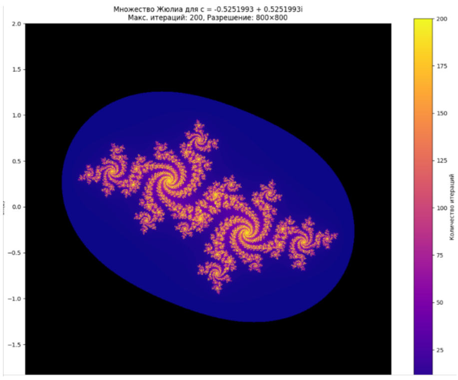
3.2 Примеры визуализации



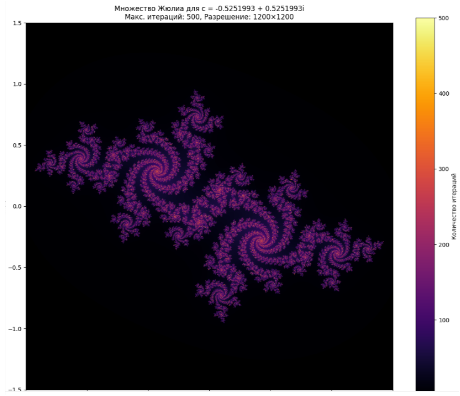
Пример из т/з, 50 итераций



Пример из т/з, 100 итераций

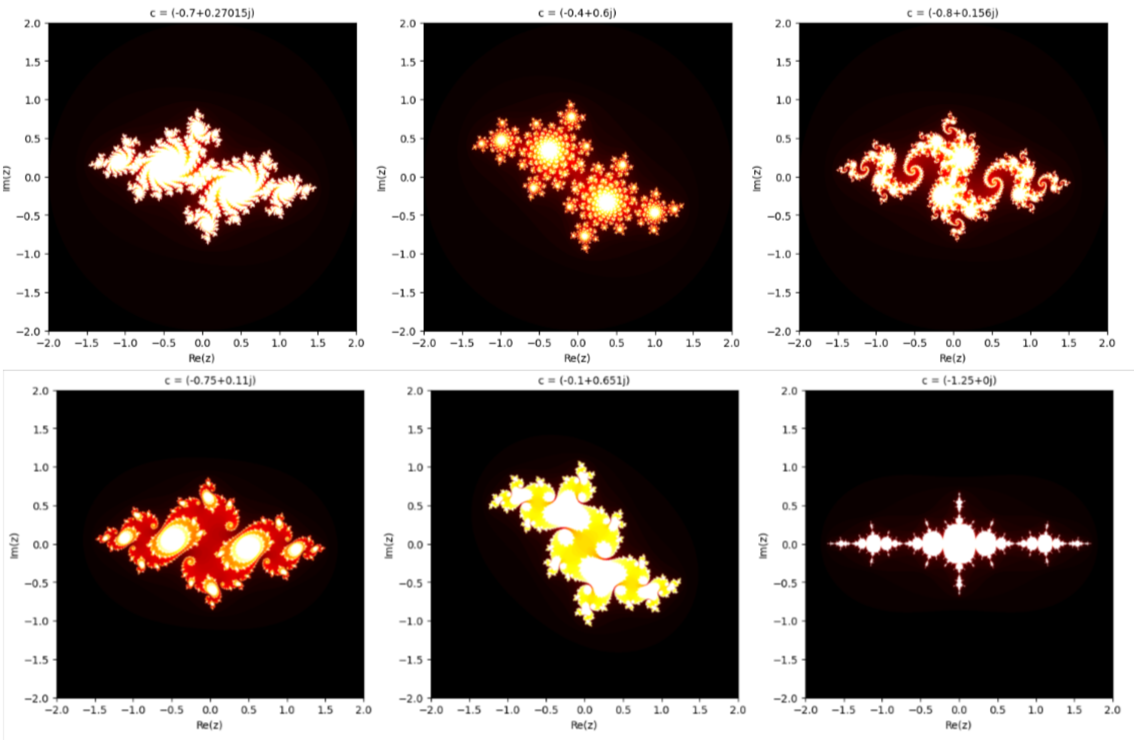


Пример из т/з, 200 итераций



Пример из т/з, 500 итераций

Рис. 2: примеры при различных C, 100 итераций



4 Кривая Гильберта

4.1 Описание структуры и построения

Кривая Гильберта была описана немецким математиком Давидом Гильбертом в 1891 году. Она тесно связана с понятием *всюду плотных кривых.*

Определение 2. *Кривая на плоскости называется* ***всюду плотной*** *в некоторой области, если она проходит через любую сколь угодно малую окрестность каждой точки этой области.*

Кривая Гильберта — пример непрерывной сюръекции вида $p\colon [0,1] \twoheadrightarrow [0,1]^2$, причём она является фракталом из-за $D_T < D_F$:

- топологическая размерность D_T равна 1, поскольку прообраз — отрезок;
- метрическая размерность D_F равна 2, поскольку образ — квадрат.

Вышеуказанные утверждения требуют более формального обоснования, которое выходит за рамки курса. Оставим их доказательство в качестве упражнения читателю.

4.1.1 Алгоритм построения

Итеративный процесс построения кривой Гильберта удобно описать при помощи *L-системы.*

Определение 3. *Детерминированной контекстнонезависимой* ***L-системой*** *называют набор, состоящий из алфавита, аксиомы, и множества правил.*

Исторически она впервые была предложена биологом Аристидом Линденмайером в качестве математической модели развития растений.

Определение 4. ***Алфавитом*** *называется конечное множество, а его элементы —* ***символами****.*

Природа символов не важна, их единственная функция — отличаться друг от друга.

Определение 5. ***Аксиомой*** *называется некоторая строка над алфавитом, определяющая начальное состояние системы.*

Определение 6. ***Правилом*** *называется пара, состоящая из предшественника (символа алфавита) и последователя (строки над алфавитом).*

Опираясь на определения, *L-система* для кривой Гильберта выглядит следующим образом:

Алфавит: $A, B, F, +, -$

Аксиома: A

Правила:
$$\begin{cases} A \to -BF + AFA + FB- \\ B \to +AF - BFB - FA+ \end{cases}$$

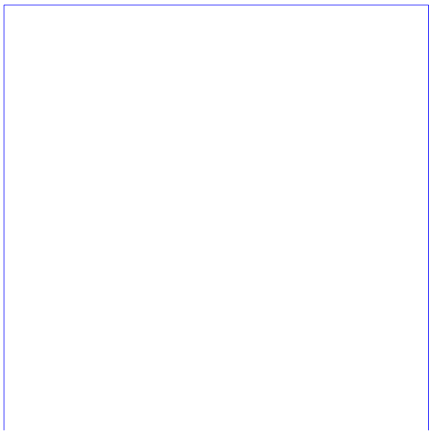
Здесь F означает «движение вперёд», «$-$» — поворот влево на $90°$, «$+$» — поворот вправо на $90°$, а A и B игнорируются при рисовании.

Листинг 3: Построение кривой Гильберта

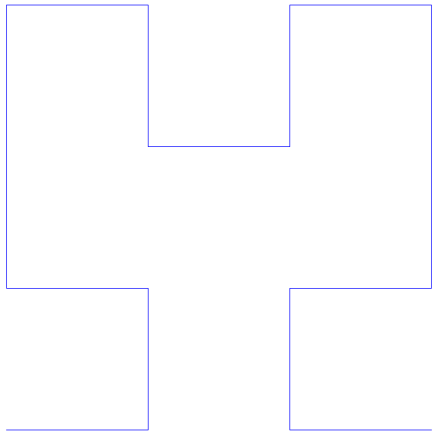
```
import matplotlib.pyplot as plt
import numpy as np

def generate_hilbert_string(level):
  """
  Generate Hilbert curve string using L-system rules
  """
  def apply_rules(char):
    if char == 'A':
      return '-BF+AFA+FB-'
    elif char == 'B':
      return '+AF-BFB-FA+'
    else:
      return char

  current_string = 'A' # axiom

  for _ in range(level):
    new_string = ''
    for char in current_string:
      new_string += apply_rules(char)
    current_string = new_string

  return current_string

def draw_hilbert_from_string(hilbert_string, level, step=10,
   angle=90, filename="hilbert_curve.png"):
  """
  Draw Hilbert curve by interpreting the generated string using
     matplotlib
  """
  x, y, direction = 0, 0, 0
  points = [(x, y)]

  for char in hilbert_string:
    if char == 'F':
      rad = np.radians(direction)
      x += step * np.cos(rad)
      y += step * np.sin(rad)
      points.append((x, y))
    elif char == '+':
      direction -= angle
    elif char == '-':
      direction += angle

  plt.figure(figsize=(10, 10))
  plt.plot(
      [p[0] for p in points],
      [p[1] for p in points],
      'b-',
```

```
        linewidth=1)
    plt.axis('off')

    total_segments = len([c for c in hilbert_string if c == 'F'])
    print(f'Hilbert Curve - Level {level}')
    print(f'Total segments: {total_segments:,}')
    print(f'Saving plot to: {filename}')
    plt.tight_layout()
    plt.savefig(filename, dpi=150, bbox_inches='tight', facecolor=
        'white')
    plt.close()

def main():
    try:
        level = int(input("Enter the level for Hilbert curve: "))
        if level < 1:
            print("Level must be at least 1. Using level 1.")
            level = 1
        elif level > 7:
            print("Warning: High levels may take long to compute and
                render.")

        output_file = f"hilbert_curve_level{level}.png"
        draw_hilbert_from_string(generate_hilbert_string(level),
            level=level, step=8, filename=output_file)

    except ValueError:
        print("Invalid input. Please enter a valid integer.")
        return

if __name__ == "__main__":
    main()
```

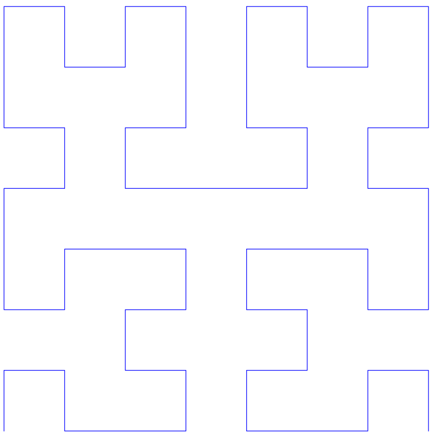
4.2 Визуализации



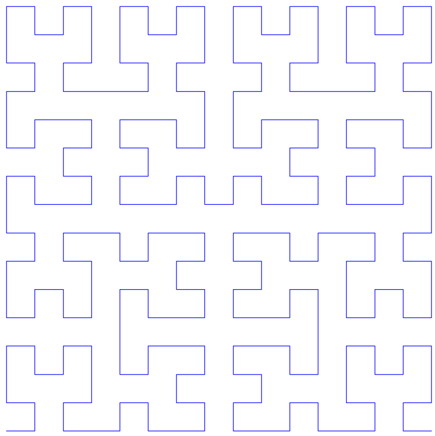
Кривая Гильберта 1-го порядка
Всего сегментов: 3



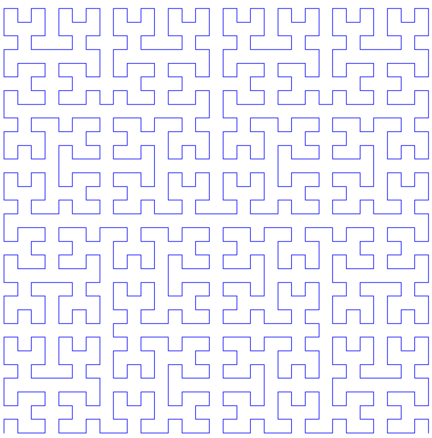
Кривая Гильберта 2-го порядка
Всего сегментов: 15



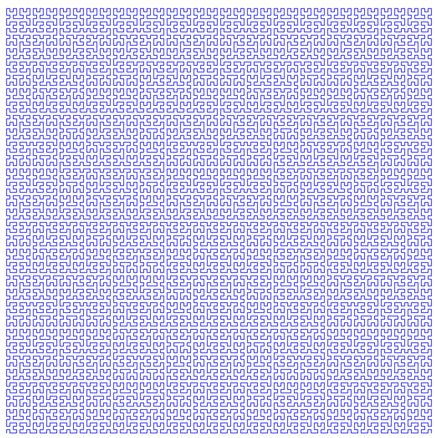
Кривая Гильберта 3-го порядка
Всего сегментов: 63



Кривая Гильберта 4-го порядка
Всего сегментов: 255



Кривая Гильберта 5-го порядка
Всего сегментов: 1,023



Кривая Гильберта 7-го порядка
Всего сегментов: 16,383

Рис. 4: Построения кривой Гильберта разных порядков

4.3 Анализ структуры

Кривая Гильберта обладает рядом интересных свойств:

- **Сюръективность.** Кривая Гильберта не позволяет биективно отобразить отрезок в квадрат, потому что $p_n\colon [0,1] \twoheadrightarrow [0,1]^2$ при $n \to \infty$ имеет бесконечное число самопересечений. В частности, центральной точке соответствуют $p(\frac{1}{6})$, $p(\frac{1}{2})$ и $p(\frac{5}{6})$.[1]

- **Недифференцируемость.** Несмотря на непрерывность отображения p, оно нигде не дифференцируемо в силу более сильного утверждения для всюду плотных кривых.[2]

- **Самоподобие.** Если увеличить любой подквадрат в $2n$ раз, мы получим кривую в точности похожую на всю кривую.

---

[1] That's Maths: Space-Filling Curves, Part II

[2] Sagan H. Space-Filling Curves. Springer-Verlag, 1994. Глава 3, стр. 34-36

5 Заключение

В ходе лабораторной работы мы познакомились с фракталами, написали программы для их визуализации и построения фрактальных множеств, смогли доказать некоторые из свойств множества Мандельброта.

Были исследованы методы построения кривой Гильберта и показано, как простое рекурсивное правило приводит к образованию сложной, самоподобной структуры.