МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему “Реализация алгоритмов построения фрактальных множеств”

Обучающийся А. А. Сопуева
(Инициалы Фамилия) (личная подпись)

Руководитель к.ф.м.н., доцент, Г. А. Тырыгина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Т. С. Якушева д. к. п. н.
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема бакалаврской работы является «Реализация алгоритмов построения фрактальных множеств».

Данная выпускная квалификационная работа рассматривает методы и инструменты, используемые для создания фрактальных множеств, которые являются примером самоподобия в геометрии. Также рассматриваются различные алгоритмы построения фрактальных множеств, такие как кривая Коха, множество Мандельброта и Жюлиа, L-системы и многие другие.

Работа состоит из трех глав. В первой главе описываются основные теоретические понятия фракталов. Рассматриваются различные виды фракталов, а также методы анализа фрактальных структур.

Во второй главе анализируются алгоритмы построения фрактальных множеств. Рассматриваются методы Линденмайера, алгоритмы Жюлиа и Мандельброта, используемые для создания фрактальных структур.

В третьей главе описываются программные реализации алгоритмов построения фрактальных множеств. Рассматриваются различные инструменты и технологии, используемые для создания программного обеспечения. Также описывается архитектура программного обеспечения и принципы его работы.

Итогом выпускной квалификационной работы является алгоритм для изображения множества фракталов. Алгоритм основан на математической модели множества, которая определяет, какие точки в комплексной плоскости принадлежат множеству фракталов.

Выпускная квалификационная работа состоит из пояснительной записки 59 страниц, введения, включая 37 рисунков, 2 таблиц, 27 источников и 6 формул.

Abstract

The title of the graduation work is "Implementation of algorithms for constructing fractal sets".

This graduation qualification work examines methods and tools used to create fractal sets, which are examples of self-similarity in geometry. Various algorithms for constructing fractal sets such as the Koch curve, the Mandelbrot and Julia sets.

The work consists of three chapters. The first chapter describes the basic theoretical concepts of fractals. The discussion covers the various types of fractals and the methods used for analyzing their structures.

The second chapter analyzes algorithms for constructing fractal sets. Methods such as Lindenmayer systems, algorithms for Julia and Mandelbrot sets used for creating fractal structures are considered.

The third chapter describes software implementations of algorithms for constructing fractal sets. The discussion revolves around various tools and technologies that utilizing for software creation. The architecture of the software and the principles of its operation are also detailed.

The outcome of the graduation qualification work is an algorithm for imaging fractal sets. The mathematical model of the set forms the basis of the algorithm, by which it is determined which points in the complex plane belong to the fractal set.

The graduation qualification work consists of an explanatory note of 59 pages, an introduction including 37 figures, 2 tables, 27 sources and 6 formulas.

Оглавление

Введение

Фрактальные множества давно привлекали внимание математиков. Эти математические объекты, обладающие свойством самоподобия, нашли применение в различных областях, таких как сжатие изображений, обработка сигналов, компьютерная графика, физика и биология, а также встречаются при изучении природных явлений. Этим и определяется актуальность.

В 1975 году математик 20-го века Бенуа Мандельброт предложил термин фрактал от латинского слова *fractus* – означающего неправильный или фрагментированный. Такие неправильные и фрагментированные формы окружают нас повсюду.

Фрактальный анализ позволяет построить математические модели природных объектов, описать их геометрические свойства с помощью фрактальной размерности. Однако применимость фрактального подхода к моделированию разнообразных объектов не исследована достаточно.

Объектом исследования выступают фрактальные множества. Предметом исследования являются алгоритмы построения фрактальных множеств и определение их фрактальной размерности.

Целью выпускной квалификационной работы является реализация алгоритмов построения фрактальных множеств.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать методы определения фрактальной размерности;
- проанализировать различные алгоритмы построения фрактальных множеств;
- реализация алгоритмов построения фрактальных множеств.

Данная бакалаврская работа включает введение, основную часть, заключение и библиографический список. Основная часть бакалаврской работы состоит из трех глав.

Глава 1 Теоретические основы фрактальных множеств

1.1 Основные понятия теории фракталов

Новые открытия в науке способны существенно изменить человеческую жизнь. К таким открытия можно отнести фракталы. «Фракталы вокруг нас повсюду, и в очертаниях гор, и в извилистой линии морского берега.» [15]

Мандельброт писал: «Фракталы — это математические, природные или созданные человеком объекты, которые мы называем неправильными, шероховатыми, пористыми или раздробленными, причём этими свойствами обладают фракталы в одинаковой степени в любом масштабе.» [10, 21]

Фракталы встречаются в природе, например, в форме растительных листьев, горных цепей и облачных формаций, а также используются в компьютерной графике, науке и других областях для моделирования и анализа сложных систем. Теория фракталов изучает свойства и структуру фракталов, их размерность и математические свойства, а также их применения в различных областях.

Хочется процитировать автора одной из статей этой книги Г. Айленбергера: «Наше ощущение прекрасного возникает под влиянием гармонии порядка и беспорядка в природных объектах — тучах, деревьях, горных грядах или кристалликах льда. Их очертания — это динамические процессы, застывшие в физических формах... наука и эстетика согласны в том, что именно теряется в технических объектах по сравнению с природными: роскошь некоторой нерегулярности, беспорядка и непредсказуемости. Понимание этого может здорово помочь нам в придании человеческого лица технологии, от которой все больше и больше зависит наше выживание.» [15]

Основные понятия теории фракталов включают:

– фрактал: математический объект, обладающий самоподобием на всех масштабах и имеющий фрактальную размерность;

- самоподобие: свойство фракталов, при котором они имеют похожую структуру на всех масштабах;
- фрактальная размерность: показатель, который отражает степень самоподобия фракталов и может быть нецелым числом;
- итерационная функция: функция, которая описывает преобразование фрактала при каждой итерации;
- мультифрактал: фрактал, который имеет различную фрактальную размерность на разных участках своей поверхности;
- фрактальный метод обеспечивает более адекватное описание сложной иерархической структуры природы. Природные системы, будь то рельеф, растительные сообщества, организмы или их морфологические структуры, обладают высоким уровнем вложенности и детализации на разных масштабах;
- метод Хаусдорфа-Безиковича: метод нахождения фрактальной размерности, основанный на понятии покрытия фрактала;
- детерминистический хаос: свойство системы, при котором малые изменения начальных условий приводят к значительным изменениям в ее поведении, что часто наблюдается в фрактальных системах;
- множество аттрактор: множество, к которому сходятся последовательности итераций итерационной функции;
- фрактальная геометрия: область математики, которая изучает фрактальные объекты и их свойства.

В.С. Секованов определяет: «Фрактальная геометрия – молодое быстроразвивающееся математическое направление, связанное не только с выдвижением новых математических идей, но и бурным развитием компьютерной графики, художественного компьютерного творчества.»

Для обозначения классификации фракталов, чаще всего используют способ их построения. Эта классификация включает в себя три основных типа фракталов: геометрические, алгебраические и стохастические.

Алгебраические фракталы - это объекты, которые можно описать с помощью алгебраических уравнений или систем уравнений. Они обычно имеют самоподобную структуру, то есть состоят из более мелких копий самих себя. «Получают их с помощью нелинейных процессов в n-мерных пространствах.» [3]

Примерами алгебраических фракталов являются фракталы Мандельброта и Жюлиа (рисунок 1), которые создаются путем повторного применения определенных алгебраических формул к комплексным числам.

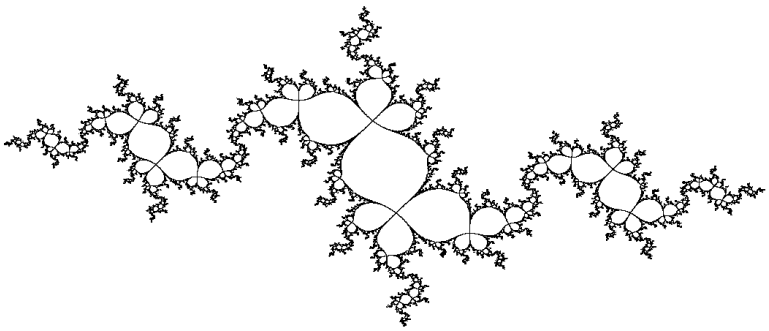


Рисунок 1 – Параболический фрактал Жюлиа

Геометрические фракталы - это объекты, которые можно описать с помощью геометрических преобразований или процедур. Они также обычно имеют самоподобную структуру, что означает, что они содержат более мелкие копии самих себя.

Примерами геометрических фракталов являются кривые Коха, кривые Хансена, снежинки Коха, фрактальные деревья и многие другие. Они создаются путем повторения определенных геометрических операций, таких как деление отрезка на несколько частей или замена части фигуры на более сложную.

Треугольник Серпинского получается путем разбиения равностороннего треугольника на четыре более мелких равносторонних треугольника, затем центральный треугольник удаляется, и этот процесс

повторяется на оставшихся треугольниках бесконечное число раз. Таким образом структура треугольника Серпинского (рисунок 2) повторяется на разных масштабах, и он является примером геометрического фрактала.

Рисунок 2 – Этапы построения треугольника Серпинского

Алгебраические и геометрические фракталы имеют широкий спектр приложений в разных науках, искусстве и дизайне.

Стохастические фракталы - это объекты, которые не могут быть описаны алгебраическими или геометрическими уравнениями, а создаются путем случайных процессов или стохастических алгоритмов. Пример стохастического фрактала изображен на рисунке 3:

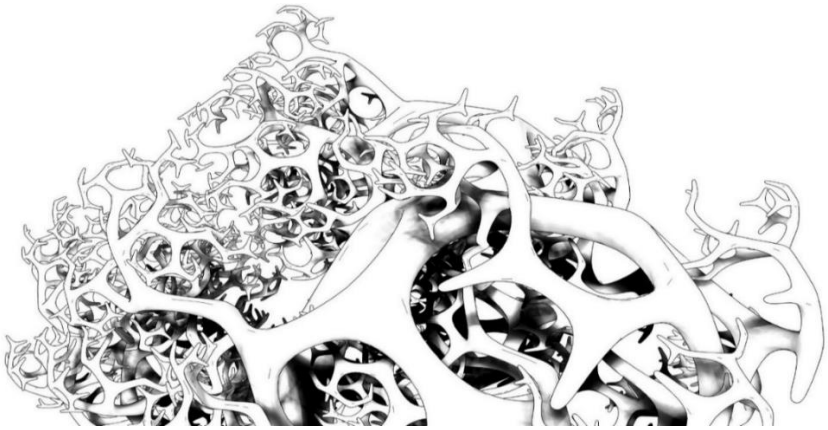


Рисунок 3 – Стохастический фрактал

Они также имеют самоподобную структуру, что означает, что они содержат более мелкие копии самих себя.

Примерами стохастических фракталов являются фракталы Брауна, которые создаются путем накопления случайных шумовых величин, и фракталы Перлина, которые создаются путем смешивания различных градиентов случайных шумовых функций.

1.2 Свойства фрактальных множеств

Фрактальные множества обладают рядом интересных свойств, которые делают их уникальными и полезными для изучения:

- самоподобие: фрактальные множества являются самоподобными, то есть они имеют похожую структуру на всех масштабах;
- фрактальная размерность: фрактальные множества имеют фрактальную размерность, которая может быть нецелой и отражает степень их самоподобия;
- бесконечность деталей: фрактальные множества содержат бесконечное число деталей на всех масштабах, что делает их сложными и красивыми;
- компактность: фрактальные множества могут быть компактными, то есть они полностью умещаются в некотором ограниченном пространстве;
- полезность в науке и технике: фрактальные множества используются в различных областях науки и техники, например, в физике, биологии, медицине, компьютерной графике, анализе данных и другие;
- фрактальный шум: фракталы могут использоваться для генерации фрактального шума, который имеет множество применений в компьютерной графике, анимации, синтезе звука и так далее;

– мультифрактальность: некоторые фракталы обладают свойством мультифрактальности, что означает, что они имеют различные фрактальные размерности для различных частей структуры;
– фрактальная геометрия: фрактальные геометрии могут использоваться для описания и моделирования различных явлений и процессов, таких как турбулентность, распределение галактик и так далее;
– фрактальная аналитическая геометрия: область математики, которая использует фракталы для решения различных задач и проблем в разных областях науки и техники;
– фрактальная компрессия: метод сжатия данных, который использует свойства самоподобия фракталов для уменьшения размеров изображений и других типов данных;
– фрактальные алгоритмы: фрактальные алгоритмы могут быть использованы для решения различных задач, таких как генерация случайных чисел, криптография, оптимизация и так далее.

Примеры фрактальных множеств: примерами фрактальных множеств являются множество Кантора, кривая Коха, фрактал Мандельброта, множество Жюлиа и другие.

Фрактальные множества также могут быть использованы для создания новых математических моделей и решения различных научных и технических задач. Использование фракталов позволяет создавать более точные и детальные модели, которые могут описывать сложные структуры и процессы.

В целом, фракталы обладают уникальными свойствами, которые делают их полезными для описания и моделирования сложных систем и явлений в разных областях.

Для количественного описания фракталов необходимо и достаточно всего лишь одной величины — размерности Хаусдорфа (показатель скейлинга). Данный показатель описывает сохраняемость геометрии или статистических характеристик при изменении масштаба. Однако в биологии,

химии, физики и даже в телекоммуникационных отраслях нередко встречаются явления, требующие распространения понятия фрактала на более сложные структуры, которые имеют более одного показателя скейлинга. Данные структуры, как правило, характеризуются сразу спектром показателей, где размерность Хаусдорфа лишь одна из многих величин. Для таких фракталов был придуман термин «мультифракталы» [1, 27].

1.3 Самоподобие

Фракталы - это фигуры, которые состоят из нескольких подобных частей, каждая из которых похожа на всю фигуру. В математике фракталы определяются как множества точек в метрическом пространстве, у которых размерность не является целым числом по метрике Минковского или Хаусдорфа. Таким образом, метрическая размерность фрактала отличается от его топологической размерности.

Эти самоподобные паттерны являются результатом простого уравнения или математического утверждения. Фракталы создаются путем повторения этого уравнения через цикл обратной связи в процессе, называемом итерацией, где результаты одной итерации формируют входное значение для следующей. Например, если посмотреть на внутреннюю часть раковины наутилуса, можно увидеть, что каждая камера раковины в основном является точной копией предыдущей камеры, только меньшего размера, если проследить их от внешней части к внутренней.

Во-первых, фракталы могут быть самоподобными в геометрическом смысле. Это означает, что они имеют одинаковую геометрическую форму на разных масштабах. Например, фрактальное множество Мандельброта имеет одинаковую форму на всех уровнях детализации.

Самоподобие фрактальных множеств может быть описано с помощью математических функций.

Во-вторых, фракталы могут быть самоподобными в статистическом смысле. Это означает, что их свойства, такие как структура, размеры и распределение, могут повторяться на разных масштабах. Например, ковер Серпинского имеет одинаковую структуру на всех масштабах, и его площадь уменьшается в два раза с каждой итерацией.

Самоподобие фрактальных множеств проявляется на различных масштабах, что делает их полезными для описания и моделирования явлений в природе и технике. Например, самоподобие может быть использовано для описания формы облаков, поверхности листьев или кривых дорог. Также самоподобие может быть использовано для создания искусственных текстур и ландшафтов, которые имеют естественный и нерегулярный вид.

Свойство самоподобия может быть использовано для решения различных задач. Например, оно может быть использовано для описания и моделирования сложных систем, таких как финансовые рынки, экономические процессы и так далее. Также может быть использовано для создания более точных и детальных моделей, которые могут описывать сложные структуры и процессы.

Для фрактальных множеств, обладающих свойством самоподобия, можно рассчитать различные характеристики, которые связаны с этим свойством.

Одной из таких характеристик является фрактальная размерность. Фрактальная размерность описывает, как изменяется размер фрактального множества при изменении масштаба. Для фрактальных множеств, обладающих свойством самоподобия, фрактальная размерность может быть рассчитана с помощью различных методов, таких как метод боксов, метод Ренье или метод Хаусдорфа-Безиковича.

Также для фрактальных множеств, обладающих свойством самоподобия, можно рассчитать коэффициент подобия, который определяет, как изменяется размер фрактального множества при изменении масштаба.

Коэффициент подобия может быть рассчитан с помощью отношения размеров двух копий фрактального множества на разных масштабах.

Также можно рассчитать другие характеристики, такие как плотность фрактального множества, геометрический центр фрактального множества и так далее. Эти характеристики могут быть использованы для описания и анализа свойств фрактальных множеств и их применения в различных областях науки, техники и искусства.

1.4 Фрактальная размерность

Большинство из нас выросли, когда нас учили, что длина, ширина и высота — это три измерения, вот и все. Фрактальная геометрия превращает эту концепцию в кривую, создавая неправильные формы во фрактальном измерении; фрактальная размерность формы — это способ измерения сложности этой формы.

Фрактальная размерность (также известная как размерность Хаусдорфа) - это концепция, используемая для описания сложности геометрических объектов, которые не могут быть описаны классической евклидовой геометрией.

Фракталы - это геометрические объекты, которые обладают свойством самоподобия, то есть их структура повторяется на различных масштабах. Фрактальная размерность позволяет оценить, насколько быстро меняется сложность объекта при изменении масштаба.

Для определения фрактальной размерности используются различные методы, например, метод Хаусдорфа-Безиковича, метод счета ящиков и другие. В результате применения этих методов получается число, которое может быть не целым и отражает степень самоподобия объекта.

«Размерность Хаусдорфа–Безиковича определяется по формуле (1):

$$D_{il} = \lim_{r \to 0} \frac{\ln N}{\ln 1/r} \quad (1)$$

где N – число уменьшенных копий k-го этапа построения фрактала, помещающихся в одной копии $k + l$–го этапа построения,

$1/r$ – коэффициент подобия этих копий. При этом k может быть любым целым числом.» [20]

В начале каждого

Формально математический фрактал определяется как любой ряд, для которого размерность Хаусдорфа (непрерывная функция) превышает дискретную топологическую размерность. Топологически линия одномерна. Однако размерность D фрактального «следа» на плоскости представляет собой непрерывную функцию с интервалом $1 \leq D \leq 2$ [13, 17].

Полностью дифференцируемый ряд имеет фрактальную размерность D = 1 (такую же, как и топологическая размерность), а броуновский след полностью занимает двумерное топологическое пространство и, следовательно, имеет фрактальную размерность D = 2. Фрактальные размерности $1 \leq D \leq 2$ количественно определяют степень, в которой след «заполняет» плоскость. Точно так же плоская криволинейная поверхность топологически двумерна, а фрактальная поверхность имеет размерность $2 \leq D \leq 3$ [25].

Например, фрактальная размерность прямой линии равна 1, фрактальная размерность кривой Коха равна примерно 1.2619, а фрактальная размерность фрактала Мандельброта равна примерно 1.58.

Существует множество методов, которые можно использовать для расчета так называемой «размерности» набора данных, которая в контексте динамических систем называется фрактальной размерностью.

Существует несколько методов расчета размерности фракталов, включая:

- метод рассеяния: в этом методе фрактальный объект рассматривается как совокупность малых частей, и размерность вычисляется как отношение между числом малых частей и их размером. Этот метод может быть применен к фрактальным объектам, которые можно разбить на множество малых частей, таких как кривые Коха или фрактальные линии;
- метод покрытия: в этом методе фрактальный объект покрывается множеством непересекающихся шаров или кубиков, и размерность вычисляется как отношение между объемом покрытия и его линейными размерами. Этот метод применим к фрактальным объектам, которые имеют сложные формы, такие как множество Мандельброта;
- метод корреляционной функции: в этом методе измеряется, как сильно коррелированы точки на фрактальном объекте, и размерность вычисляется как степень корреляции. Этот метод может быть применен к фрактальным объектам, которые имеют структуру с самоподобием;
- метод Хаусдорфа: в этом методе размерность определяется с помощью понятия меры Хаусдорфа, которая позволяет измерить размерность непрерывных множеств. Этот метод является наиболее общим и применим к любым фрактальным объектам.

Одним из ключевых принципов при расчете размерности фракталов является понимание самоподобия объекта на разных масштабах. Фрактальные объекты имеют одинаковую структуру на всех масштабах, что позволяет использовать один и тот же метод для измерения их размерности. Также важно иметь точную модель генерации фрактального объекта, чтобы можно было применить соответствующий метод расчета размерности.

В настоящее время фракталы являются объектом интенсивных научных исследований, имеющих важные и разнообразные приложения в таких областях, как медицина, биология, физиология, физика полимеров,

геоморфология, теория броуновского движения, астрофизика и теория катастроф.

Физик Жан Перрен попытался измерить скорость, являющуюся производной от положения частицы по времени, но обнаружил, что скорость частицы «самым диким образом меняется по величине и направлению и не стремится к пределу, как время, необходимое для наблюдения, уменьшается», а также сказал, что «природа содержит предположения как о недифференцируемых, так и о дифференцируемых процессах».

Ботаник Роберт Браун заметил почти случайное движение маленькой частицы, когда она погружена в жидкость или газ. Жидкость или газ состоят из постоянно движущихся молекул, которые ударяются о маленькую частицу с разных направлений. Это движение теперь носит его имя и называется броуновским движением (рисунок 4). «Броуновское движение – один из самых известных примеров стохастического процесса.» [4]

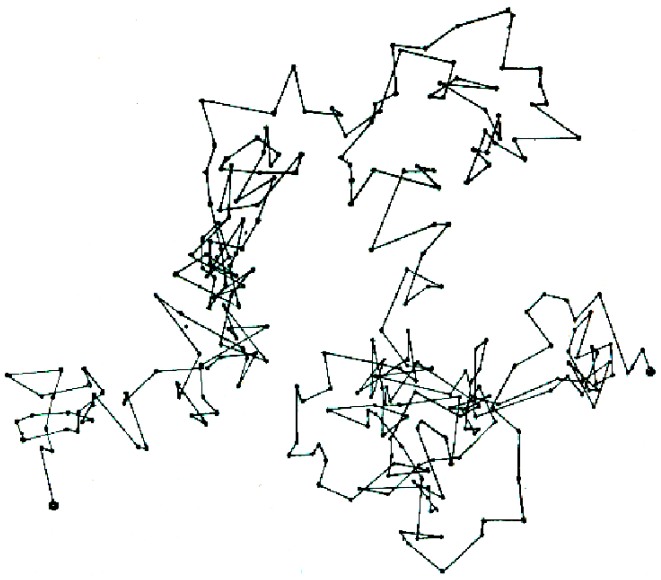


Рисунок 4 – Броуновское движение

На рисунке 4 показано, как сгенерированная компьютером частица наблюдается в разные промежутки времени. При уменьшении временных интервалов расчетная длина пути фактически увеличивается.

«Обобщенное броуновское движение является моделью, описывающей движение гипотетических частиц, которые случайным образом перемещаются в пространстве.» [14]

Это исследование броуновского движения показывает нам, что некоторые процессы в природе лучше всего моделируются недифференцируемыми функциями. Эти функции будут иметь бесконечную длину.

Мандельброт интерпретировал береговую линию как фрактальный объект - объект, размерность которого совпадает с показателем степени в. Такая смелая интерпретация нуждается в разъяснении. Прежде всего, поясним, при чём тут размерность, и, во-вторых, разберёмся с дробной размерностью. Мы помним, что топологическую размерность мыслители от Евклида до Пуанкаре определяли, мысленно разрезая его поверхностями, сечениями, кривыми и точками. [6]

На основе фрактального анализа речной системы было проведено моделирования наводнений. Наводнения являются следствием недостаточной развитости речной сети, значительное затопление определённой территории земли в результате подъёма уровня воды в реке, озере и так далее.

Расчет затоплений строится по оценке уровня подъема воды Δh от ординара, который равен средней скорости подъема X, умноженное на среднее время подъема T [13]. После преобразований получили формулу (2) расчета максимально возможного подъема уровня воды:

$$\Delta h = k(D_0 - D) \tag{2}$$

При выборе $D_0 = 1$, уровень воды будет обусловлен фрактальной размерностью речной системы. Далее ввели относительную характеристику θ, на территориях с максимальным и минимальным уровнями подъема воды:

$$\theta_{1,2} = \frac{\Delta h_1}{\Delta h_1} = \frac{D_1 - 1}{D_2 - 1}$$

Выявили, что на величину θ не влияют величины осадков, температурный режим, снежных покровов. Это означает, что главным параметром регулированием водоотвода является фрактальная размерность речной системы – D.

Но были выявлены и другие способы влияния на изменение фрактальной размерности речной системы. Больше всего влияет противопаводковых емкостей вокруг искусственных водоемов. Также и мелиоративные мероприятия: спрямление русла и устройства дренажным систем. [19]

1.5 Нахождение фрактальной размерности

Нахождение фрактальной размерности может быть достаточно сложной задачей, так как это требует использования специальных методов и алгоритмов. Однако, в общих чертах процесс нахождения фрактальной размерности выглядит следующим образом:

- выбрать фрактал, для которого нужно найти фрактальную размерность;
- разбить фрактал на множество более мелких фрагментов;
- для каждого фрагмента вычислить его линейный размер, например, длину или площадь;
- построить график зависимости логарифма количества фрагментов от логарифма их размера;
- приближенно определить фрактальную размерность как наклон этого графика.

«Первоначально различные определения фрактальной размерности были детально исследованы для самоподобных фракталов, и оказалось, что величина размерности не зависит от выбора определения.» [9]

Существует множество алгоритмов для нахождения фрактальной размерности, таких как метод Хаусдорфа-Безиковича, метод счета ящиков и другие.

Один из самых распространенных методов - это метод счета ящиков (Box-Counting method), который можно использовать для измерения фрактальной размерности двумерных фракталов. Этот метод один из самых простых, чаще всего применяется для оценки размерности множества на плоскости [5], состоящий из 5 шагов.

Шаг 1. Разбить изображение фрактала на квадраты одинакового размера (ящики).

Шаг 2. Подсчитать количество ящиков, которые пересекаются с фракталом (то есть содержат часть фрактала).

Шаг 3. Повторить шаги 1 и 2 для ящиков меньшего размера.

Шаг 4. Построить график логарифма количества ящиков в зависимости от логарифма размера ящиков.

Шаг 5. Фрактальная размерность определяется как угол наклона полученной прямой на графике.

«Иногда ящики не могут равномерно распределять данные, тогда некоторые ячейки будут пропущены, если размеры ящиков будут увеличиваться геометрически. Из-за этой потенциальной проблемы некоторые авторы предлагают сначала сопоставить необработанные данные с квадратной единицей. Однако, если исходные данные образуют неквадратную область, эта опция будет недоступна.» [11, 16]

Для фракталов трех и более измерений можно использовать другие методы, например, метод генерации случайных блужданий или метод Монте-Карло.

В целом, процесс нахождения фрактальной размерности требует тщательной обработки данных и может быть сложен, особенно для более сложных фракталов.

Размерность Минковского определяется как подсчет ящиков, точнее оценка «подсчета кубов» для фрактальной размерности. «Данная размерность тесно связана с размерностью Хаусдорфа, хоть и существуют множества, для которых они различаются, во многих случаях эти размерности совпадают, поэтому говоря о размерностях, чаще всего не имеет значение, какая размерность имеется в виду.» [7]

Проведем расчет для треугольника Серпинского методом подсчета ящиков. Начальная точка, этап 0, представляет собой треугольник S_0 с единичными сторонами. На более поздних этапах S_k получается путем удаления равносторонних треугольников из оставшихся треугольников в S_{k-1}, как показано ниже. Фигура S_k состоит из 3^k треугольников со стороной 2^{-k}. Треугольник Серпинского определяется как $S = \bigcap_{i=0}^{\infty} S_i$.

«При построении салфетки Серпинского каждый треугольник фиксированного размера заменяется тремя треугольниками вдвое меньшего размера.» [2]

Мы определяем $dim_B(S)$ путем оценки $\overline{dim}_B(S)$ и $\underline{dim}_B(S)$. Пусть $\delta>0$ и выбрать $k \in N$ такое, что $2^{-k} < \delta \leq 2^{-k+1}$. Тогда треугольники S_k являются δ-покрытием S, поэтому $N_\delta(S) \leq 3^k$. Из этого следует:

$$\overline{dim}(S) = \overline{\lim}_{\delta \to 0} \frac{\log N_\delta(S)}{-\log \delta} \leq \overline{\lim}_{k \to \infty} \frac{\log 3^k}{-\log 2^{-k+1}} = \frac{\log 3}{\log 2}$$

В качестве альтернативы можно выбрать $l \in N$ так, что $2^{-l-1} \leq \delta < 2^{-l}$. Тогда множество диаметра δ может пересекать только треугольники в S_l, расстояние между которыми меньше 2^l. Следовательно, любое такое множество пересекает не более трех треугольников в S_l. Это означает, что δ-покрытие S содержит не менее $3^l/3$ множеств, откуда следует:

$$\underline{dim}_B F = \lim_{\delta \to 0} \frac{\log \mathrm{N}_\delta(\mathrm{F})}{-\log \delta} \geq \lim_{l \to 0} \frac{\log 3^{l-1}}{-\log 2^{-l-1}} = \frac{\log 3}{\log 2}$$

Следовательно, $\frac{\log 3}{\log 2} \leq \underline{dim}_B(F) \leq \overline{dim}_B(F) \leq \frac{\log 3}{\log 2}$, что показывает, что размерность треугольника Серпинского с учетом квадратов равна $\frac{\log 3}{\log 2}$.

Теперь вычислим размерность Хаусдорфа для множества Кантора F. Разделим F на две части, левую часть $F_L = F \cap \left[0, \frac{1}{3}\right]$ и правую часть $F_R = F \cap \left[\frac{2}{3}, 1\right]$. Эти части полностью аналогичны самой F, но масштабируется с коэффициентом в $\frac{1}{3}$ раз. $F = F_L \cup F_R$, где объединение не пересекаются. Теперь для любого s,

$$\mathcal{H}^s(F) = \mathcal{H}^s(F_L) + \mathcal{H}^s(F_R) = \frac{1}{3}^S \mathcal{H}^s(F) + \frac{1}{3}^S \mathcal{H}^s(F),$$

по свойству масштабирования будем считать, что для критического значения $s = dim_H(F),\ 0 < \mathcal{H}^s(F) < \infty$. Теперь можем разделить на $\mathcal{H}^s(F)$, чтобы получить $1 = 2\left(\frac{1}{3}\right)^s$. Это естественным образом приводит к размерности Хаусдорфа множества Кантора, которая равна $s = \frac{\log 2}{\log 3}$ [25].

Аналогичным образом можно вычислить размерность Хаусдорфа для кривой Коха. Кривая Коха состоит из четырех копий самой себя, назовем их F_c, масштабированных в $\frac{1}{3}$ раз. Теперь для любого s, $\mathcal{H}^s(F) = 4\mathcal{H}^s(F_\mathrm{c}) = 4\left(\frac{1}{3}\right)^s \mathcal{H}^s(F)$.

Предположим, что для критического значения $s = dim_H(F),\ 0 < \mathcal{H}^s(F) < \infty$. После деления на $\mathcal{H}^s(F)$, получим $1 = 4\left(\frac{1}{3}\right)^s$. В итоге получим размерность Хаусдорфа для кривой Коха, равная $dim_H(F) = s = \frac{\log 4}{\log 3}$.

Кривая Коха, треугольник Серпинского и множество Кантора являются классическими примерами фрактальных объектов с известными фрактальными свойствами. Сравним их основные характеристики: фрактальная размерность, количество самоподобных элементов, пространственное заполнение.

По расчетам фрактальной размерности фрактальных объектов, получилось:

- кривая Коха: $dim_H(F) = s = \frac{\log 4}{\log 3}$;
- треугольник Серпинского: $dim(F) = s = \frac{\log 3}{\log 2}$;
- множество Кантора: $dim(F) = s = \frac{\log 2}{\log 3}$.

Таким образом, треугольник Серпинского имеет наибольшую размерность и самую сложную фрактальную структуру. Высокая размерность означает, что фрактал заполняет пространство сложным, иерархическим образом на многих масштабных уровнях.

Количество самоподобных элементов:

- кривая Кохи: на каждой итерации число элементов увеличивается в 4 раза;
- треугольник Серпинского: на каждой итерации число треугольников увеличивается в 3 раза;
- множество Кантора: на каждой итерации промежуточный интервал разбивается пополам, так что число интервалов удваивается.

Быстрый рост числа деталей при итерации. При каждой итерации число треугольников увеличивается в 3 раза, число элементов кривой Коха увеличивается в 4 раза, а число интервалов множества Кантора увеличивается в 2 раза. Это приводит к тому, что даже на первых итерациях фракталы содержат большое число деталей, придающих ему сложную структуру.

Пространственное заполнение: Треугольник Серпинского равномерно заполняет пространство на всех масштабах, что создает плотную иерархическую ткань. Это придает фракталу высокую визуальную сложность.

Пространственное заполнение множества Кантора имеет наименьшее пространственное заполнение, поскольку на каждой итерации удаляется центральная часть.

Пространственное заполнение кривой Коха является ограниченным и иерархическим, но не достигает полного заполнения всего пространства, как у треугольника Серпинского. Это придает Кривой Кохи своеобразную «лоскутную» структуру заполнения.

Таким образом, можно сделать вывод, что треугольник Серпинского обладает наибольшей фрактальной сложностью, в то время как множество Кантора имеет наименьшую размерность и наименьшее пространственное заполнение среди рассмотренных фракталов.

Для описания сложных объектов необходимо уметь определять фрактальную размерность. Нахождение фрактальных размерностей – сложная теоретическая задача, поэтому представляет интерес более удобные простые методы для определения фрактальных размерностей.

Выводы

В первой главе рассмотрены теоретические основы фрактальных множеств. В частности, определены основные понятия теории фракталов, рассмотрены свойства фрактальных множеств, описано понятие самоподобия и фрактальной размерности.

Также рассмотрены методы нахождения фрактальной размерности Минковского и Хаусдорфа-Безиковича для треугольника Серпинского, множества Кантора и кривой Коха. Проведен сравнительный анализ этих фракталов.

Глава 2 Анализ алгоритмов построения фрактальных множеств

2.1 Множество Жюлиа

Множество Жюлиа назван в честь французского математика Гастона Жюлиа, который исследовал их свойства примерно в 1915 году и завершился своей знаменитой работой в 1918 году.

Множество Жюлиа теперь связано с теми точками $z = x + iy$ на комплексной плоскости, для которых ряд $z_{n+1} = z_n^2 + c$ не стремится к бесконечности. Где c — комплексная константа, для каждого с получается свой набор Жюлиа. Начальным значением z_0 для серии является каждая точка плоскости изображения.

Итерации множества Жюлиа имеют вид $z_{n+1} = f(z_n), n = 0,1,2 \ldots$ с простой функцией на комплексной плоскости. Функции f – это граница множества тех точек z, которые при итерировании $f(z)$ стремятся к бесконечности [8].

Далее следует разделить вещественную и мнимую часть отображения, тогда отображение будет иметь вид:

$$x_{n+1} = x_n^2 - y_n^2 + a, \qquad y_{n+1} = 2x_n y_n + b$$

Это отображение дает множество фракталов, соответствующих множеству Жюлиа $J(c) = J(a, b)$.

Пусть значение комплексной константы $c = 0$. В этом случае отображение $f(z) = z^2$ имеет две неподвижные точки $z_1 = 0$ и $z_2 = 1$, которые получаются их уравнения $z^2 = z$. Так как $f'(z_1) = 0$, $f'(z_2) = 2$, то первая точка является притягивающей, а вторая – отталкивающей по теореме о характере неподвижной точки [8].

В более широком смысле точная форма повторяемой функции может быть любой, общая форма $z_{n+1} = f(z_n)$, интересные множества возникают с нелинейными функциями $f(z)$. К часто используемым функциям относятся следующие:

$$z_{n+1} = c \cdot sin(z_n) \qquad z_{n+1} = c \cdot exp(z_n)$$
$$z_{n+1} = c \cdot i \cdot cos(z_n) \qquad z_{n+1} = c \cdot z_n(1 - z_n)$$

Хорошо известное множество Мандельброта образует своего рода индекс к множеству Жюлиа. Набор Жюлиа либо связан, либо разъединен, значения с, выбранные из множества Мандельброта, связаны, а значения вне набора Мандельброта - разъединены. Несвязные множества часто называют «пылью», они состоят из отдельных точек независимо от того, в каком разрешении они просматриваются. На рисунке 5 изображено «пыль» множества Мандельброта:

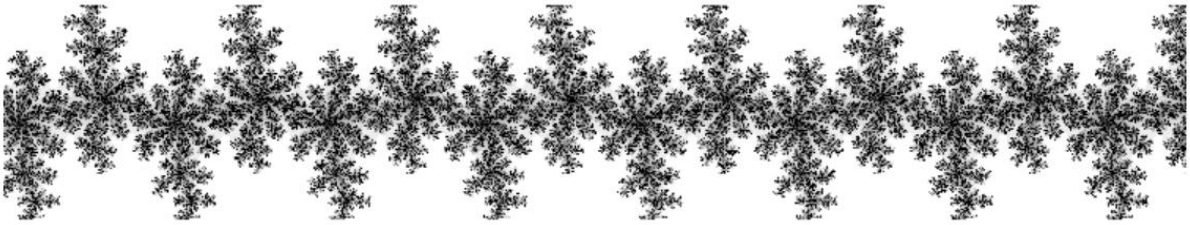


Рисунок 5 – Ненулевой мнимый компонент

Другое свойство множеств Жюлиа относится к различным областям c. Если c действительно, то множество Жюлиа зеркально отражается относительно действительной оси. Другие значения c с ненулевым мнимым компонентом имеют 180-градусную вращательную симметрию.

2.2 Множество Мандельброта

Множество Мандельброта - это одно из наиболее известных и изученных фрактальных множеств. Оно было открыто и исследовано американским математиком Бенуа Мандельбротом в 1970-х годах.

Вот изображение (рисунок 6), представляющее фрактал множества Мандельброта, один из самых знаковых математических фракталов:

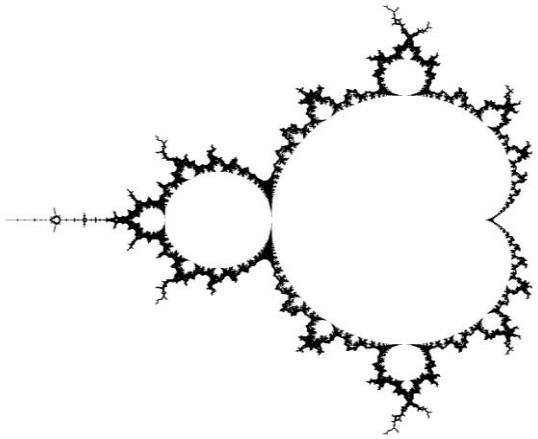


Рисунок 6 – Множество Мандельброта

«Классы одномерных комплексных отображений, сопоставляющих одному комплексному числу $z_n = x_n + iy_n$ другое комплексное число $z_{n+1} = x_{n+1} + iy_{n+1}$ по итерационному правилу $z_{n+1} = f(z_n)$, где $f(z)$ – некоторая нелинейная функция переменной z, n – номер итерации» [8, 12].

«Неподвижной точкой $\tilde{z}$ отображения $z_{n+1} = f(z_n)$ или неподвижной точкой функции $f(z)$ называется корень уравнения $f(z) = z$» [18].

Примером квадратного комплексного отображения $f(z) = z^2 + c$, где $c = a + ib$ – ненулевая комплексная константа, представлена в формуле (3):

$$z_{n+1} = z_n^2 + c \tag{3}$$

где z - комплексное число,

c - постоянное комплексное число,

n - количество итераций.

Каждая точка в множестве Мандельброта представляет собой значение c, для которого $J(c)$ связно. Множество Мандельброта M для полинома $f(z) = z^2 + c$ есть множество таких чисел c, для которых последовательность итераций $z = 0$ ограничена, то есть:

$$M = \left\{c \in \mathbb{C}: \left\{f^{\{n\}}(0)\right\}_{n=0}^{\infty} \text{ограничена}\right\}.$$

Точка $z = 0$ в качестве начальной связан с тем, что это единственная точка, в которой производная функции $f(z) = z^2 + c$ обращается в нуль.

Это уравнение определяет структуру и форму фрактала Мандельброта, которые повторяются на всех масштабах.

Множество Мандельброта строится на основе итеративной функции комплексной переменной z, где $z(0) = 0$. Для каждой точки плоскости комплексных чисел, процесс итерации повторяется до тех пор, пока значение z не станет бесконечно большим, либо не достигнет определенного ограничения. Если значение z стремится к бесконечности, то точка считается не принадлежащей множеству Мандельброта, в противном случае - принадлежащей.

Изображение множества Мандельброта (рисунок 6) можно получить, применив эту итеративную функцию для каждой точки в ограниченной области плоскости комплексных чисел, например, в квадрате от -2 до 2 на вещественной оси и от -2i до 2i на мнимой оси. Каждый пиксель на изображении представляет собой точку в этой области, и его цвет определяется количеством итераций, необходимых для того, чтобы значение z превысило определенный порог.

Множество Мандельброта обладает многими интересными свойствами, в том числе бесконечной детализацией на всех масштабах и самоподобием. Его красивые и сложные формы содержат множество деталей и структур, которые повторяются на разных масштабах. Множество Мандельброта также имеет множество приложений в математике, физике, биологии, компьютерной графике и других областях.

2.3 L-системы

L-система или система Линденмайера — это тип формальной грамматики. Систематическое изучение и формализация были предприняты П. Прусинкевичем.

L-системы открывают путь к бесконечному разнообразию новых фракталов, что и послужило причиной их широкого применения в компьютерной графике для построения фрактальных деревьев и растений. Рассмотренные в данной курсовой работе L-системы ограничиваются случаем детерминированных L-систем и графикой на плоскости.

L-система состоит из алфавита символов, набора продукционных правил, которые расширяют каждый символ в некоторую большую строку символов, исходной строки, с которой начинается построение, и алгоритма для преобразования сгенерированных строк в геометрические структуры. Центральным понятием L-систем является понятие перезаписи. Переписывание — это метод создания сложных объектов путем замены частей простого исходного объекта с использованием локальных правил перезаписи.

L-системы могут использоваться для создания различных фрактальных структур, таких как деревья (рисунок 7), кустарники, кристаллы и многие другие. Они также могут быть использованы для генерации фрактальных изображений, которые могут быть использованы в различных приложениях, таких как компьютерные игры, архитектура и дизайн.

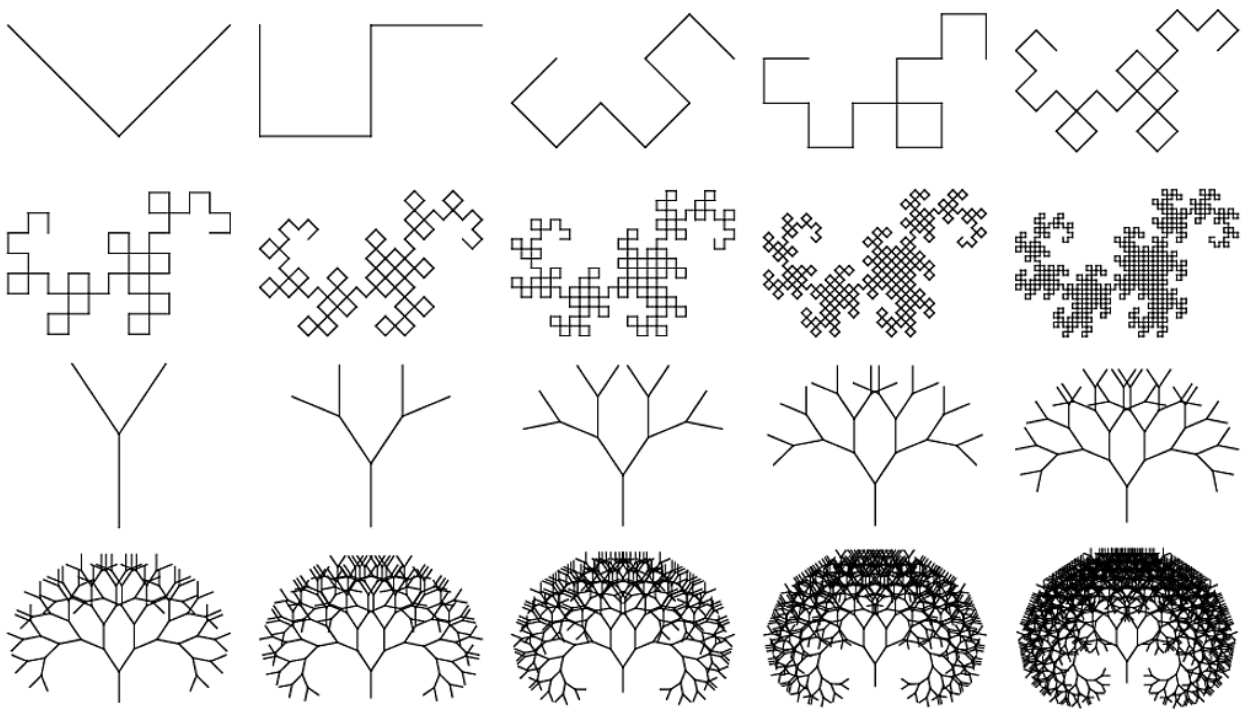


Рисунок 7 – Фрактальная структура дерева

«Система Линденмайера, стала параллельной перезаписывающей системой и типом формальной грамматики. Ее можно определить по формуле (4), как кортеж из трех элементов:

$$G = (V, w, P) \tag{4}$$

где V – алфавит,

w – аксиома,

P – порождающее правило.» [8]

Форма и структура фрактальной структуры, созданной с помощью L-системы, зависят от правил замены символов. Каждое правило состоит из символов, которые заменяются на другие символы, и угла поворота, который определяет направление рисования структуры.

Например, рассмотрим следующую L-систему:

– axiom: F;

– rules: F → F+F-F-F+F.

Начальная строка (axiom) состоит из одного символа F. Правило (rule) гласит, что каждый символ F в строке должен быть заменен на строку F+F-F-F+F. Здесь символы плюс и минус указывают на поворот направо и налево соответственно.

«Обновление букв в данном слове предполагается одновременным, то есть буквы слова одного уровня обновляются раньше любой буквы следующего уровня» [8].

При выполнении правил замены символов для этой L-системы, получим следующую последовательность:

1. F;
2. F+F-F-F+F;
3. F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F+F-F-F+F.

Эта последовательность символов определяет фрактальную кривую, которая называется кривой Коха. Эта кривая является примером фрактала, поскольку она имеет самоподобную структуру и может быть бесконечно увеличена и детализирована.

L-системы могут быть классифицированы по нескольким характеристикам:

Контекстно-независимые L-системы (или КНФ-L-системы) - это системы, в которых правила замены символов не зависят от контекста, в котором эти символы находятся. В контекстно-зависимых L-системах (или КЗФ-L-системах) правила замены символов зависят от контекста, т.е. от символов, которые находятся рядом с заменяемым символом.

Детерминированные L-системы - это системы, в которых каждое правило замены применяется строго определенное количество раз. Стохастические L-системы - это L-системы, в которых правила замены применяются случайным образом с определенной вероятностью.

Параметрические L-системы - это системы, в которых символы не только заменяются на другие символы, но и влияют на положение и ориентацию элементов структуры. Например, символ "F" может означать передвижение вперед на некоторое расстояние и поворот на некоторый угол. Непараметрические L-системы - это L-системы, в которых символы заменяются на другие символы, но не влияют на положение и ориентацию элементов структуры.

Каждый тип L-системы имеет свои особенности и может быть использован для создания разных видов структур и форм. Например, контекстно-зависимые L-системы могут быть использованы для создания более сложных и реалистичных моделей, в то время как стохастические L-системы могут быть использованы для создания более естественных и случайных форм.

2.4 Кривая Коха

Кривая Коха, также известная как снежинка Коха, является одним из простейших и наиболее известных фрактальных множеств. Она была впервые предложена шведским математиком Хельге фон Кохом в 1904 году.

«Снежинка Коха» – фрактал включает в себя взятие треугольника и превращение центральной трети каждого сегмента в треугольную выпуклость таким образом, чтобы сделать фрактал симметричным. Каждая выпуклость, конечно, длиннее исходного сегмента, но все же содержит внутри конечное пространство.

Кохова кривая может быть построена путем последовательного замещения каждого отрезка на треугольник с центральным отрезком такой же длины. Такой процесс называется итерацией. На каждой итерации длина кривой увеличивается в 4 раза.

Первая итерация Коховой кривой представляет собой обычный треугольник. Выразим размерность D по формуле (5):

$$L(\delta) = \left(\frac{4}{3}\right)^n \quad (5)$$

где $L(\delta)$ – длина ,

n – номер шага,

δ – звено, равная 3^{-n}.

Можно выразить номер шага по формуле (6):

$$n = -\frac{\ln \delta}{\ln 3} \quad (6)$$

Тогда длину можно записать так:

$$L(\delta) = \left(\frac{4}{3}\right)^n = \exp\left(-\frac{\ln \delta\,(\ln 4 - \ln 3)}{\ln 3}\right) = \exp\left(\ln \delta\left(1 - \frac{\ln 4}{\ln 3}\right)\right)$$

$$L(\delta) = \delta^{1-D} = \exp\left(\ln \delta\left(1 - \frac{\ln 4}{\ln 3}\right)\right) \Rightarrow D = \frac{\ln 4}{\ln 3}$$

На второй итерации каждый отрезок заменяется на две части: линию и уголок, который представляет собой меньший треугольник.

$$D = \left(\frac{\ln 4}{\ln 3}\right)^2$$

На следующих итерациях процесс повторяется, и каждый уголок заменяется на еще более мелкий треугольник.

«Одно из важных свойств снежинки Коха – ее бесконечная длина. В связи с этим Мандельброт опубликовал несколько работ, в которых он изучил

вопрос об измерении британской береговой линии. В качестве модели он использовал фигуру, подобную снежинке Коха.» [22, 24].

Кохова кривая является самоподобной, что означает, что она имеет одинаковую форму на разных масштабах. При этом, ее длина бесконечна, хотя на каждой итерации длина увеличивается в 4 раза. Таким образом, Кохова кривая не является фракталом в строгом смысле этого слова, так как она не обладает свойством бесконечной детализации на всех масштабах. Однако, она является простым примером фрактальной формы и часто используется для иллюстрации фрактальных свойств. На рисунке 8 изображена снежинка Коха:

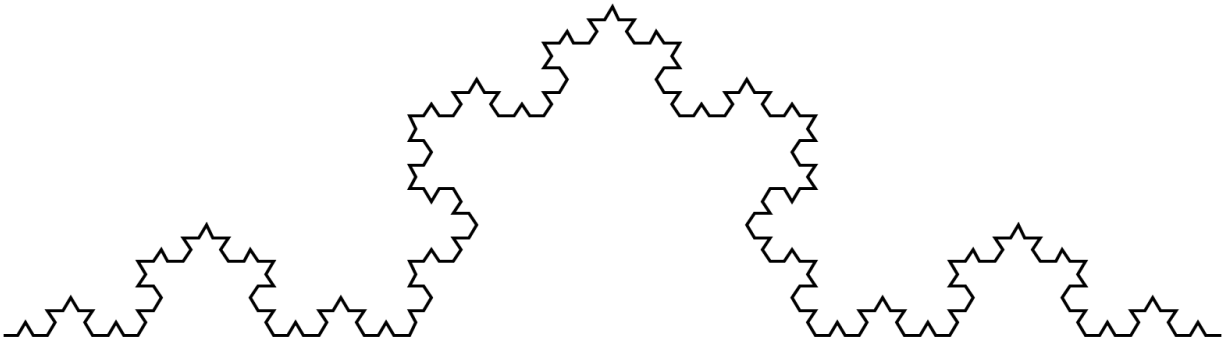


Рисунок 8 – Снежинка Коха

Кохова кривая также имеет множество интересных математических свойств и применений. Она может быть использована для описания многих естественных явлений, таких как фрактальная структура льда или облаков. Она также может быть использована в качестве генератора случайных чисел и в криптографии.

Выводы

Во второй главе представлены математические модели фрактальных множеств. Далее предложены алгоритмы построения множеств Мандельброта, Жюлиа и кривой Коха. Все эти алгоритмы открывают новые возможности для создания сложных фрактальных структур.

Глава 3 Реализация программного обеспечения для построения фрактальных множеств

3.1 Описание архитектуры программы

Для построения фрактальных множеств используются различные математические методы, такие как рекурсия, итерации, а также геометрические преобразования, такие как масштабирование, поворот и сдвиг. Реализация алгоритмов построения фрактальных множеств может быть выполнена на языках программирования, таких как Python, Java, C++ и других.

Для разработки алгоритмов выберем язык Python. Построение многих множеств на Python может быть реализовано с помощью графической библиотеки Matplotlib и черепашьей графики Turtle.

Множество Мандельброта представляет собой набор точек на комплексной плоскости, каждая из которых соответствует определенному значению функции, определенной для комплексных чисел. Архитектура кода, будет следующей:

- задание области на комплексной плоскости, для которой будет строиться множество Мандельброта;
- задание количества точек по осям действительных и мнимых чисел;
- создание двумерного массива значений, соответствующих комплексным числам на графике;
- определение множества на основе заданного количества итераций для каждой точки комплексной плоскости;
- отображение результатов на графике с помощью функции imshow библиотеки Matplotlib.

На рисунке 9 представлена блок-схема алгоритма Мандельброта.

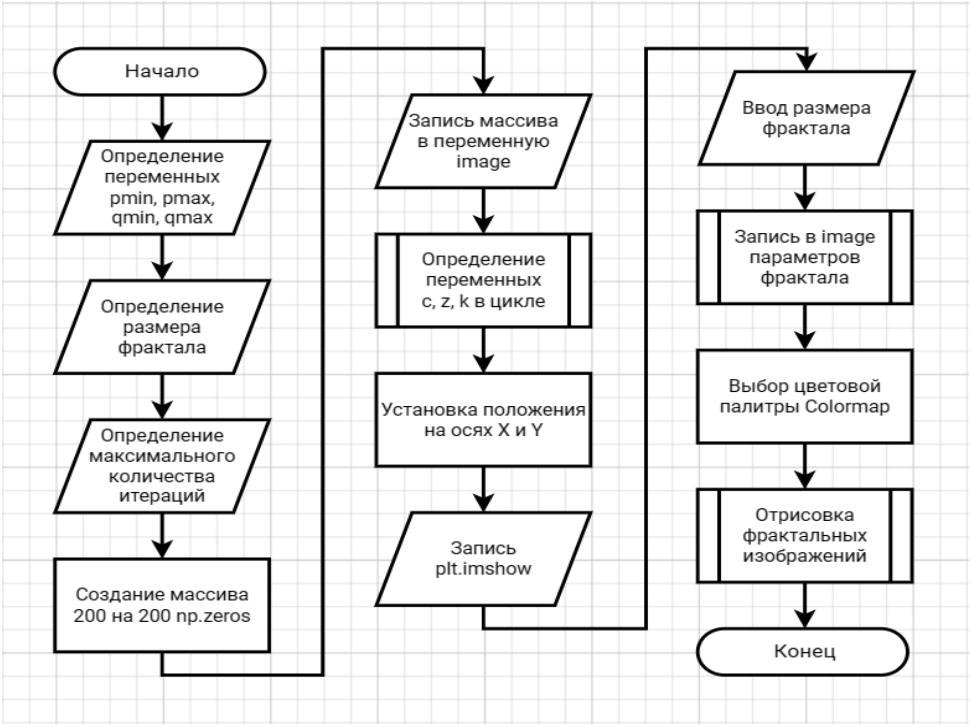


Рисунок 9 – Блок-схема алгоритма Мандельброта

L-система (или Линденмайеровская система) - это формальный язык, который используется для генерации изображений путем итеративного применения правил замены. Для построения L-системы на Python можно использовать следующий алгоритм:

- определить начальную строку, которая будет заменяться в процессе итераций;
- определить правила замены для каждого символа;
- определить количество итераций;
- написать функцию, которая будет применять правила замены на каждой итерации.

В результате получится изображение на основе L-системы с начальной строкой, правилом замены и итерациями. Можно будет изменять начальную строку, правила замены и количество итераций, чтобы получать различные изображения.

На рисунке 10 представлена блок-схема алгоритма L-систем.

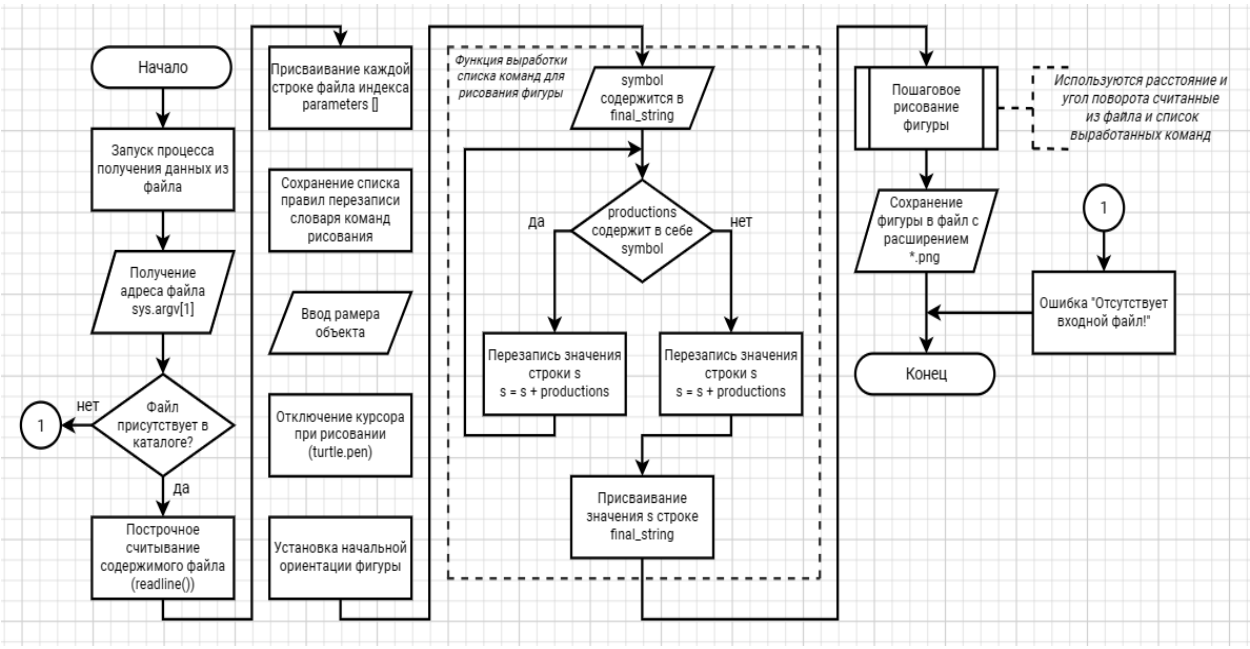


Рисунок 10 – Блок-схема алгоритма L-систем

Кривая Коха - это фрактальная кривая, которая получается из начального отрезка путем деления его на три равные части и замены средней части на равносторонний треугольник. Этот процесс повторяется для каждой из полученных частей в течение нескольких итераций. Для построения кривой Коха на Python можно использовать следующий алгоритм:

- импортируем модуль Turtle;
- написать функцию, которая будет рисовать один шаг кривой Коха;
- задать начальную длину отрезка;
- написать функцию, которая будет рисовать всю кривую Коха до заданной глубины рекурсии.

Множество Жюлиа - это фрактальное множество, которое возникает при итеративном применении к заданной точке комплексной плоскости функции $f(z) = z^2 + c$, где c - это заданная константа комплексной плоскости. Для построения множества Жюлиа на Python можно использовать следующий алгоритм:

– импортируем модуль Matplotlib;
– написать функцию, которая будет вычислять значение функции $f(z)$ для заданной точки z и константы c;
– написать функцию, которая будет проверять, принадлежит ли точка к множеству Жюлиа;
– написать функцию, которая будет рисовать множество Жюлиа для заданной константы c;
– вызвать функцию для рисования множества Жюлиа для заданной константы c.

На рисунке 11 представлена блок-схема алгоритма Жюлиа.

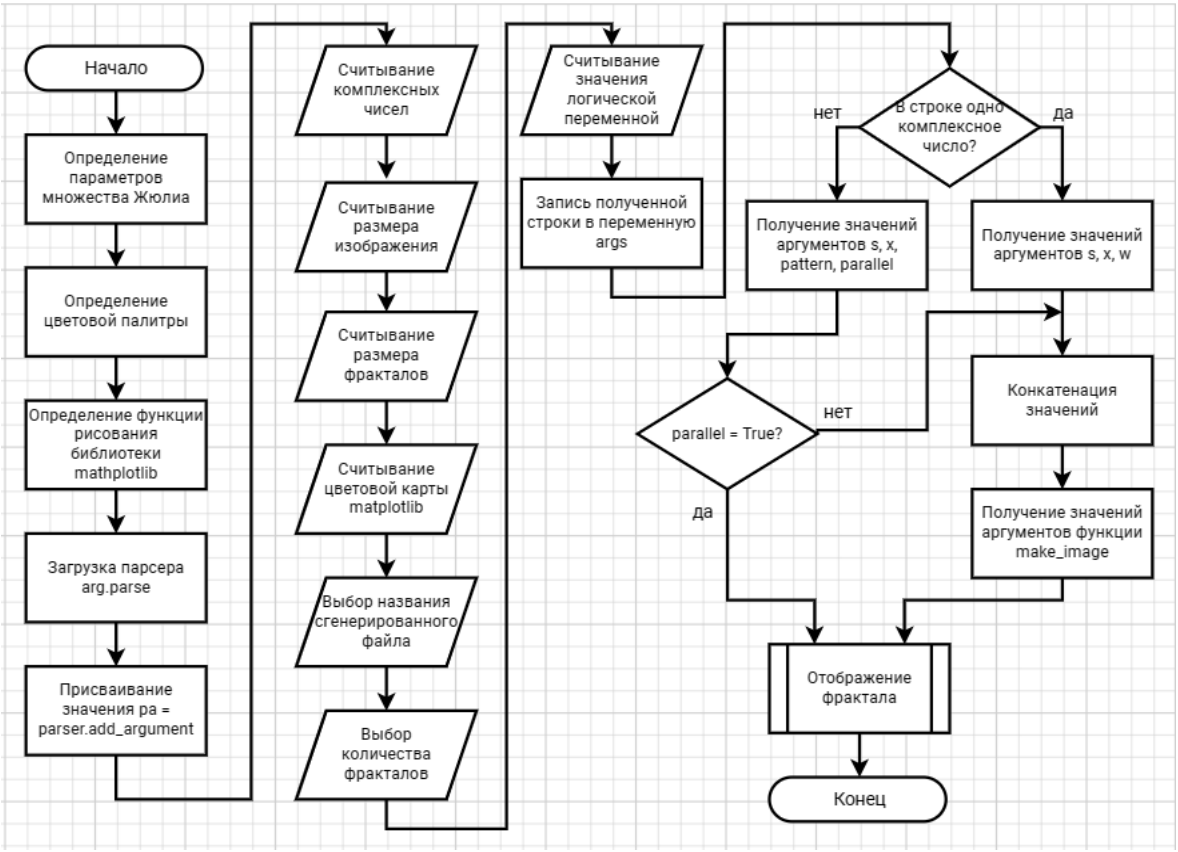


Рисунок 11 – Блок-схема алгоритма Жюлиа

В результате реализации будет рисунок множества Жюлиа для заданной константы с определенным разрешением и числом итераций для проверки.

Можно будет изменять константу, разрешение изображения и число итераций, чтобы получать различные варианты множества Жюлиа.

3.2 Реализация алгоритмов на выбранном языке программирования

Для реализации множества Мандельброта на python нужно подключить библиотеки numpy и Matplotlib, представленных на рисунке 12. Numpy – библиотека для работы с вычислениями в python, которая предоставляет массивы и матрицы для хранения и обработки данных, а также функции для работы с ними. Matplotlib – библиотека для визуализации данных, которая позволяет создавать различные типы графиков, диаграмм и другие.

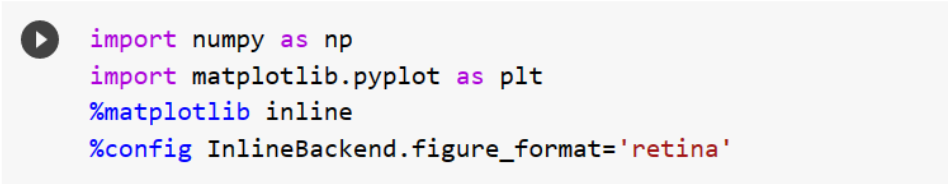


Рисунок 12 – Импорт библиотек

Для вычисления множества Мандельброта инициализируем параметры (рисунок 13). Переменные pmin, pmax, qmin и qmax задают диапазон значений для действительной и мнимой частей комплексного числа c. Ppoints и qpoints устанавливают количество точек на горизонтальной и вертикальной оси. Max_iterations устанавливает максимальное количество итераций, которые используются для определения того, принадлежит ли комплексное число с множеству Мандельброта. Infinity_border задает расстояние, после которого комплексное число с считается "бесконечным".

```
[ ]  # инициализиация параметров
     # пусть c = p + iq и p меняется в диапазоне от pmin до pmax, а q меняется в диапазоне от qmin до qmax
     pmin, pmax, qmin, qmax = -2.5, 1.5, -2, 2

     ppoints, qpoints = 200, 200 # число точек по горизонтали и вертикали

     max_iterations = 300 # максимальное количество итераций

     infinity_border = 10 # расстояние
```

Рисунок 13 – Инициализация параметров

При реализации множества Мандельброта сначала создается двумерный массив image размером ppoints на qpoints, заполненный нулями. Каждый элемент этого массива представляет собой точку на комплексной плоскости, изображенный на рисунке 14.

Затем вложенными циклами перебираются все точки на комплексной плоскости, используя np.linspace для генерации равномерно распределенных значений для p и q из диапазонов pmin, pmax, qmin и qmax.

```
image = np.zeros((ppoints, qpoints))
for ip, p in enumerate(np.linspace(pmin, pmax, ppoints)):
    for iq, q in enumerate(np.linspace(qmin, qmax, qpoints)):
        c = p + 1j * q
        z = 0
        for k in range(max_iterations):
            z = z**2 + c
            if abs(z) > infinity_border:
                image[ip,iq] = k
                break
plt.xticks([])
plt.yticks([])
plt.imshow(-image.T, cmap='flag')
```

Рисунок 14 – Реализация множества Мандельброта

Для каждой точки c на комплексной плоскости вычисляется z = 0 и затем применяется формула z = z**2 + c многократно (не более max_iterations раз) до тех пор, пока не произойдет "бег в бесконечность" (то есть abs(z) не станет больше infinity_border). Если "бег в бесконечность" не произошел, то значение

k записывается в соответствующий элемент массива image, где k - число итераций до достижения "бега в бесконечность".

Наконец, с помощью imshow из библиотеки matplotlib.pyplot создается изображение множества Мандельброта. Параметр cmap задает цветовую палитру, а plt.xticks([]) и plt.yticks([]) отключают метки на осях.

В результате будет рисунок 15 со стандартным размером 200×200 пикселей:

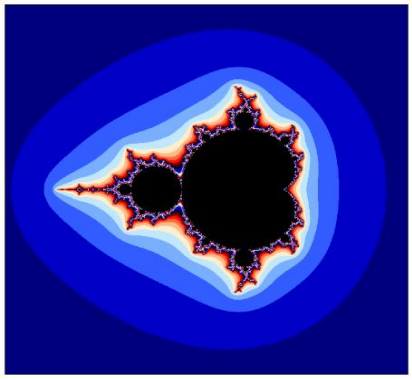


Рисунок 15 – Множество Мандельброта

Можно изменить размер рисунка, увеличить или уменьшить. Для этого используется функция plt.figure (рисунок 16) для создания нового изображения размером (10, 10). Функция mandelbrot вызывается с заданными параметрами, и результат сохраняется в переменной image.

```
plt.figure(figsize=(10, 10))
image = mandelbrot(-2.5, 1.5, 1000, -2, 2, 1000)
```

Рисунок 16 – Функция для создания изображения с размером

Также можно изменить и цветовую палитру изображения, созданный в предыдущем коде. Реализация цветовой палитры изображен на рисунке 17.

```
[16] from itertools import cycle
     import matplotlib.colors as clr
     colorpoints = [(1-(1-q)**4, c) for q, c in zip(np.linspace(0, 1, 20),
                                                    cycle(['green', 'orange',
                                                           'yellow',]))]
     cmap = clr.LinearSegmentedColormap.from_list('mycmap',
                                                  colorpoints, N=2048)
```

Рисунок 17 – Реализация цветовой палитры изображения

Сначала импортируется модуль itertools и matplotlib.colors с псевдонимом clr. Затем создается список colorpoints, который состоит из кортежей (q, c), где q - значение от 0 до 1, а c - соответствующий цвет. Значения q равномерно распределяются на интервале от 0 до 1 с помощью функции np.linspace, а цвета задаются в списке cycle([]), который бесконечно повторяется с помощью функции cycle. Это означает, что палитра будет состоять из трех цветов: зеленого, оранжевого и желтого.

Затем используется функция clr.LinearSegmentedColormap.from_list для создания новой линейной сегментированной цветовой палитры mycmap из списка colorpoints. Параметр N устанавливает количество цветов в палитре, в данном случае 2048.

В итоге изображение будет иметь вид (рисунок 18):

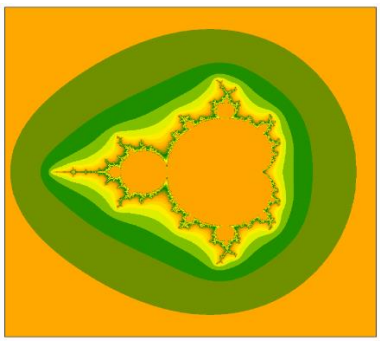


Рисунок 18 – Новая палитра множества Мандельброта

Для реализации множества Жюлиа на python нужно подключить библиотеки numpy и matplotlib.

Данный код реализует функцию "julia_set", которая генерирует изображение множества Жюлиа. Это множество точек на комплексной плоскости, которые не выходят за пределы заданного значения при повторном применении определенной формулы.

Функция принимает несколько параметров (рисунок 19), в том числе ширину и высоту изображения, диапазон значений для действительной и мнимой частей комплексной плоскости, параметр "уровень", который влияет на количество итераций расчета, и параметр "c", который задает константу формулы для построения множества Жюлиа.

```
def julia_set(**kwargs):
    w = kwargs.get('w', 401)
    h = kwargs.get('h', 401)
    c = kwargs.get('c', complex(0.0, 0.65))
    level = kwargs.get('level', 255)

    re_min = kwargs.get('re_min', -2.0)
    re_max = kwargs.get('re_max', +2.0)
    im_min = kwargs.get('im_min', -2.0)
    im_max = kwargs.get('im_max', +2.0)
```

Рисунок 19 – Параметры изображения множества Жюлиа

Функция генерирует массивы значений (рисунок 20) для действительной и мнимой частей комплексной плоскости, создает сетку из этих значений и выполняет расчеты для каждой точки на сетке, используя формулу множества Жюлиа. Результаты расчетов сохраняются в массивы "Z" и "Z_level", которые содержат информацию о точках на комплексной плоскости и их уровне яркости (цвете) на изображении.

Данный код реализует функцию "make_image", которая создает изображение на основе массива данных, полученного из функции "julia_set". Функция принимает несколько параметров, в том числе массив данных "data"

и имя выходного файла "outputname". Если задано имя файла, то функция создаст директорию для файла.

```
real_range = np.linspace(re_min, re_max, w)
imag_range = np.linspace(im_min, im_max, h)
Y, X = np.meshgrid(imag_range, real_range)
Z = X + 1j * Y
Z_level = level * np.ones(Z.shape, dtype=np.uint8)
max_abs = 6
for _ in range(level):
    norm = np.abs(Z)
    bb = norm < max_abs
    Z[bb] = Z[bb] * Z[bb] + c
    Z_level[bb] -= 5
    Z_level[Z_level < 0] = 0
return Z, Z_level
```

Рисунок 20 – Массивы значений

Функция также принимает параметры для задания границ изображения, разрешения dpi, цветовой карты и т.д. Если цветовая карта задана как "all", то функция создаст изображение для каждой доступной цветовой карты и сохранит их в отдельных файлах. В противном случае функция создаст изображение с заданными параметрами. Функция приведена на рисунке 21:

```
if colormap == 'all':
    for c in list_of_colormaps():
        fig = plt.figure()
        fig.set_size_inches(1 * (xmax - xmin), 1 * (ymax - ymin))
        ax = plt.Axes(fig, [0, 0, 1, 1])
        ax.set_axis_off()
        fig.add_axes(ax)
        plt.set_cmap(c)
        ax.contourf(data, aspect='normal')
```

Рисунок 21 – Параметры изображения множества Жюлиа

Функцию "create_one_julias_set" (рисунок 22) создает изображение фрактала Жюлиа на основе заданного комплексного числа "c". Функция

принимает несколько параметров, в том числе цветовую карту "colormap" и имя выходного файла "outputname". Если имя файла не задано, изображение не сохраняется.

Для обработки массива данных фрактала функция применяет различные операции, включая вычисление модуля, уровня и взвешивания данных. Операции выполняются в соответствии с выбранным шаблоном данных, заданным параметром "pattern".

```
def create_one_julias_set(c=complex(0.0, 0.65), colormap='magma', outputname=None, **kwargs):
    s = kwargs.get('s', 401)
    x = kwargs.get('x', 2.0)
    w = kwargs.get('w', 0.75)  # define ponderation between level and norm
    pattern = kwargs.get('pattern', 'abcde')  #
    Z, Z_level = julia_set(w=s, h=s, c=c, re_min=-x, re_max=+x, im_min=-x, im_max=+x, **kwargs)
    Z /= np.max(np.abs(Z))
    Z_abs = np.abs(Z)
    Z_level = Z_level.astype(float)
    Z_level = (Z_level - np.min(Z_level)) / (np.max(Z_level) - np.min(Z_level))
    Z_weighted = 0.5 / (w + 1.0) * (Z_level + w * Z_abs)
    Z_weighted = (Z_weighted - np.min(Z_weighted)) / (np.max(Z_weighted) - np.min(Z_weighted))
    dictPattern = {'a': np.real(Z), 'b': Z_abs, 'c': np.imag(Z), 'd': Z_level, 'e': Z_weighted}
    ZZZ = np.concatenate([dictPattern[p] for p in pattern], axis=1)
    make_image(ZZZ, outputname=outputname, xmax=len(pattern), colormap=colormap, dpi=s)
```

Рисунок 22 – Генерация одного изображения

Итоговый массив данных объединяется в единый массив и передается в функцию "make_image", которая создает изображение с заданными параметрами.

Функция "create_several_julias_set" (рисунок 23) создает несколько изображений фракталов Жюлиа для заданных комплексных чисел "cn".

Функция принимает несколько параметров, включая количество изображений "n", цветовую карту "colormap" и флаг "parallel", который указывает, следует ли создавать изображения параллельно. Также принимает параметры для задания размеров изображения, границ фракталов и т.д.

Параметр "cn" задает комплексные числа, для которых необходимо создать изображения фракталов. Функция генерирует имена выходных файлов для каждого изображения с помощью функции "get_filenames".

Если флаг "parallel" установлен в значение True, функция использует многопроцессорность для создания нескольких изображений параллельно. В противном случае функция создает изображения последовательно.

Если флаг "parallel" установлен в значение True, функция использует список входных параметров для создания нескольких процессов, каждый из которых вызывает функцию "create_one_julias_set" с соответствующими параметрами.

```
def create_several_julias_set(n, cn, **kwargs):
    parallel = kwargs.get('parallel', False)
    colormap = kwargs.get('colormap', 'magma')
    s = kwargs.get('s', 401)
    x = kwargs.get('x', 2.0)
    pattern = kwargs.get('pattern', 'abcde')
    cn = np.linspace(cn[0], cn[1], n)
    outputnames = get_filenames(colormap, cn)
    if parallel:
        from multiprocessing import cpu_count
        from multiprocessing import Pool
        ncores = max(1, cpu_count() - 1)
        listOfInputs = [{'c':complex(c), 's':s, 'x':x, 'pattern':pattern, 'colormap':colormap, 'outputname':outputname}
                        for c, outputname in zip(cn, outputnames)]
        p = Pool(ncores)
        p.map(create_one_julias_set_to_expand, listOfInputs)
    else:
        for c, outputname in zip(cn, outputnames):
            create_one_julias_set(c, outputname=outputname, **kwargs)
```

Рисунок 23 – Функция генерирования

Итоговые изображения сохраняются с помощью функции "make_image" в соответствующие выходные файлы.

Функция get_filename возвращает строку, содержащую имя файла для сохранения изображения фрактала, приведенный на рисунке 24.

Функция get_filenames возвращает список имен файлов для сохранения нескольких изображений фракталов, используя функцию get_filename.

```
def get_filename(colormap, c, suffix=''):
    return colormap + '/res_{0:06d}_{1:06d}{2}.png'.format(int(100000 * c.real),
                                                           int(100000 * c.imag), suffix)


def get_filenames(colormap, cn, suffix=''):
    return [get_filename(colormap, c, suffix) for c in cn]
```

Рисунок 24 – Файл для хранения изображений

В отличие от функции get_filename, функция get_filenames принимает список комплексных чисел cn вместо одного числа c, и для каждого числа в списке формирует соответствующее имя файла, используя переданные параметры colormap и suffix.

С помощью модуля argparse (рисунок 25) скрипт задает аргументы командной строки, такие как размер изображения, диапазон значений комплексных чисел, используемых для создания фрактала, название цветовой карты, используемой для отображения изображения, количество изображений, которые нужно сгенерировать, и т.д.

```
def main():
    class CustomFormatter(argparse.ArgumentDefaultsHelpFormatter, argparse.RawDescriptionHelpFormatter):
        pass
    parser = argparse.ArgumentParser(formatter_class=CustomFormatter)
    pa = parser.add_argument
    pa('-k', type=complex, default=[complex(0.285, 0.01)], nargs='*')
    pa('-s', '--size', type=int, default=401)
    pa('-x', type=float, default=2.0)
    pa('-c', '--colormap', type=str)
    pa('--pattern', type=str)
    pa('-o', '--output', default=None)
    pa('-n', '--number', type=int, default=2)
    pa('-p', '--parallel', action='store_true')
    args = parser.parse_args()
    output = args.output
    if len(args.k) == 1:
        create_one_julias_set(c=args.k[0], colormap=args.colormap, outputname=output, s=args.size, x=args.x, pattern=args.pattern)
    else:
        create_several_julias_set(n=args.number, cn=args.k, colormap=args.colormap,
              s=args.size, x=args.x, pattern=args.pattern, parallel=args.parallel)
```

Рисунок 25 – Аргументы командной строки

Затем скрипт использует заданные аргументы для вызова функций, которые генерируют изображения фракталов, и сохраняет их в файлы с заданными именами или выводит на экран, если имя файла не указано.

В зависимости от заданных аргументов скрипт может генерировать одно или несколько изображений фракталов и использовать все доступные ядра процессора для ускорения процесса. Результатом будет рисунок 26 - изображение множества Жюлиа из пяти генераций:

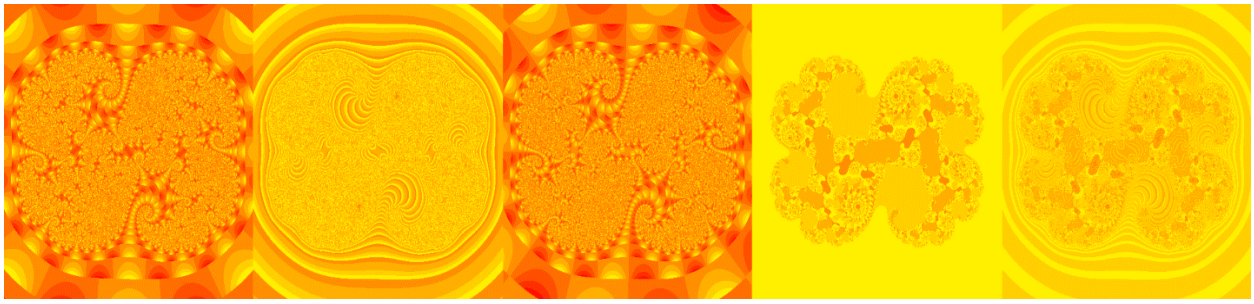


Рисунок 26 – Набор из пяти генераций

Также еще одним результатом будет генерация одного изображения множества Жюлиа, приведенный на рисунке 27:

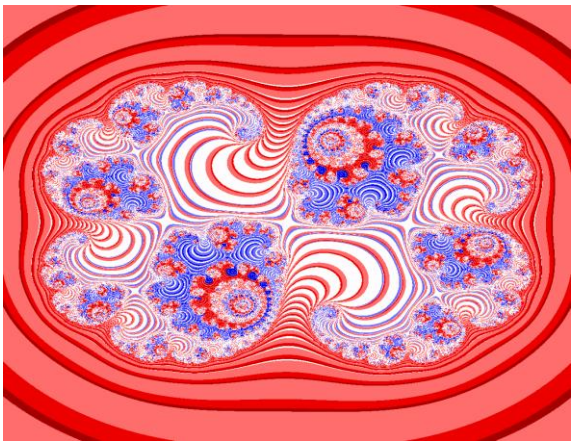


Рисунок 27 – Генерация одного изображения

Для реализации алгоритма построения изображения с использованием L-систем нужно подключить библиотеки sys. Sys - это стандартный модуль в Python, который предоставляет доступ к некоторым переменным и функциям, связанным с интерпретатором Python и системой. «Для графической

реализации L-систем в качестве подсистемы вывода используется так называемая тертил-графика (turtle — черепаха).» [9]

Функция rewrite (рисунок 28) принимает на вход три аргумента: n - количество итераций, axiom - начальная строка, и productions - словарь правил замены символов.

```
def rewrite(n, axiom, productions):
    final_string = axiom
    for _ in range(n):
        s = ""  # Строка для построения
        for symbol in final_string:
            if symbol in productions:  # Заменить
                s += productions[symbol]
            else:  # Константа
                s += symbol
        final_string = s
    return final_string
```

Рисунок 28 – Функция итераций символов

Функция создает конечную строку, начиная с заданной начальной строки axiom, и на каждой итерации заменяет символы в этой строке согласно правилам, заданным в словаре productions. Если символ не может быть заменен, то он остается в строке без изменений.

Процесс продолжается до тех пор, пока не будет достигнуто заданное количество итераций n. Функция возвращает конечную строку, полученную после всех итераций замены символов.

Данная функция draw (рисунок 29 и 30) использует модуль turtle для рисования графики. Модуль принимает на вход три аргумента: distance - длина шага, angle - угол поворота в градусах и commands_list - список команд для рисования.

```
def draw(distance, angle, commands_list):
    stack = []
    for command in commands_list:
        if command in ["F", "l", "r"]:  # l это F_l, r это F_r
            # Переместиться вперед на шаг длины 'distance'
            turtle.forward(distance)

        elif command == "f":
            # Переместиться вперед на шаг длины 'distance' без рисования линии
            turtle.penup()
            turtle.forward(distance)
            turtle.pendown()

        elif command == "+":
            turtle.left(angle)

        elif command == "-":
            turtle.right(angle)
```

Рисунок 29 – Функция рисования графики

В таблице 1 приведены команды, использующие для входного файла.

Таблица 1 – Список команд для входного файла

Символ	Команда
F	Переместиться вперед на шаг длины d
f	Продвинуться на шаг длины d, не оставляя следа
+	Повернуть влево на угол δ
-	Повернуть направо на угол δ
l	F_l, переместиться вперед
r	F_r, переместиться вперед
L	Игнорировать, ничего не делать
R	Игнорировать, ничего не делать
[	Поместите текущее состояние черепахи в стек
]	Извлечь состояние из стека и сделать его текущим состоянием черепахи

«Все остальные символы, которые могут содержаться в кодовом слове, игнорируются. Размер шага и величина θ заданы заранее и остаются неизменными для всех перемещений точки.» [23, 26]

Функция проходит по каждой команде в списке commands_list и выполняет соответствующие действия.

Если команда является одной из F, l, r, то черепашка перемещается вперед на заданное расстояние distance. Если команда f, то черепашка перемещается вперед на заданное расстояние без рисования линии. Если команда +, то черепашка поворачивает влево на заданный угол angle, а если команда -, то вправо на заданный угол angle.

```
elif command == "L" or command == "R":
    pass

elif command == "[":
    # Нажмите текущее состояние черепашки (положение и ориентация)
    # в стек
    stack.append((turtle.pos(), turtle.heading()))

elif command == "]":
    # Извлекаем последнее состояние из стека и делаем его текущим
    new_position, new_orientation = stack.pop()
    turtle.penup()
    turtle.goto(new_position)
    turtle.setheading(new_orientation)
    turtle.pendown()
```

Рисунок 30 – Продолжение функции рисования графики

Если команда L или R, то она игнорируется, т.к. эти команды не определены в данной функции. Когда встречается команда [, текущее положение и ориентация черепашки сохраняются в стеке. Когда встречается команда], последнее сохраненное состояние черепашки извлекается из стека и делается текущим. Это позволяет черепашке вернуться к предыдущему положению и продолжить рисование с этой точки.

Данная функция main является точкой входа в программу и используется для чтения входных параметров из файла, вызова функций rewrite и draw и сохранения результата в файл.

Функция начинается с чтения имени входного файла из аргументов командной строки. Если аргументы не переданы, то программа выводит сообщение об ошибке и завершается.

Затем функция открывает входной файл, считывает параметры и сохраняет их в соответствующие переменные. Входной файл содержит следующие параметры (рисунок 30):

– n - количество итераций рекурсивной функции rewrite
– d - длина шага для черепашки
– delta - угол поворота для черепашки
– axiom - аксиома для начальной строки
– productions - список правил производства/перезаписи

Затем функция создает словарь productions (рисунок 31) из списка правил производства/перезаписи, где ключами являются символы, которые нужно заменить, а значениями - строки, на которые нужно заменить эти символы.

```
# Сохраняем список правил производства/перезаписи в словаре
productions = {}
for p in parameters[4:]:
    k, v = p.split("->")
    productions[k] = v
```

Рисунок 31 – Функция создания словаря

Далее функция настраивает черепашку (рисунок 32), устанавливая размер и начальную ориентацию, и вызывает функцию rewrite, которая генерирует новую строку на основе аксиомы и правил производства. Полученная строка передается в функцию draw, которая использует черепашку для рисования графики на экране.

Наконец, функция сохраняет вывод в файл в формате EPS и завершает работу черепашки.

```
# Настройки черепахи
turtle.Screen().screensize(4000, 3000)
turtle.Screen().setup(1200, 900)
turtle.pen(shown=False)
turtle.left(90)  # Установите начальную ориентацию на положительную ось Y

final_string = rewrite(n, axiom, productions)

draw(d, delta, final_string)

# Сохраняем вывод в файл
ts = turtle.getscreen()
ts.getcanvas().postscript(file=filename.replace(".txt", ".eps"))
turtle.done()
```

Рисунок 32 – Функция настройки черепахи

Следующая последовательность символов на рисунке 33 определяет фрактальную кривую Коха:

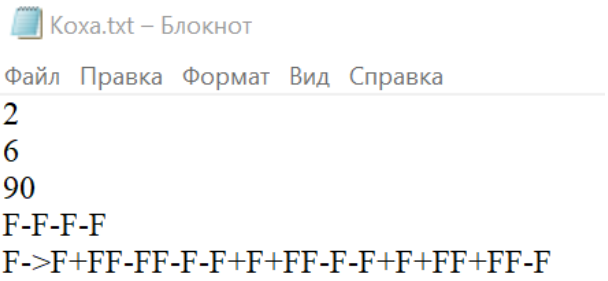


Рисунок 33 – Последовательность символов кривой Коха

Эта последовательность символов определяет фрактальную кривую, которая называется кривой Коха. Эта кривая является примером фрактала, поскольку она имеет самоподобную. Результат генерирования кривой Коха изображен на рисунке 34.

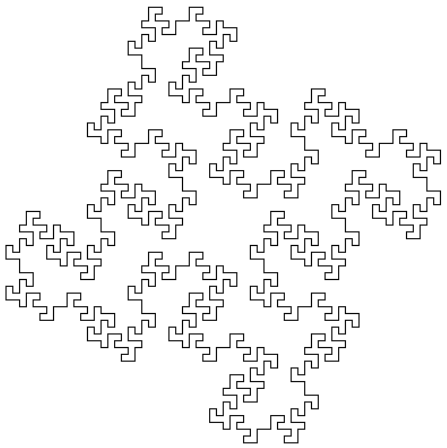
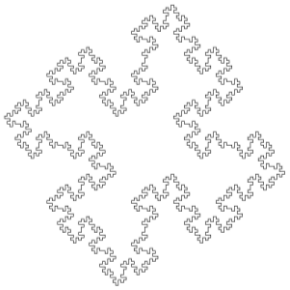
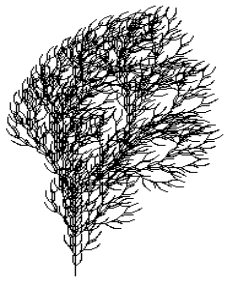


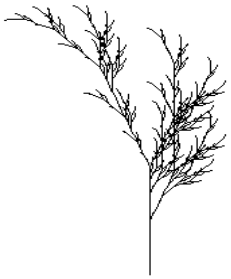
Рисунок 34 – Изображение кривой Коха №1

Изменяя количество итераций, длину шага, угол поворота и все остальное можно получить разные изображения кривой Коха. Также с помощью L-систем можно изобразить структуры растений. Далее в таблице 2 будут приведены примеры последовательностей и их результаты (рисунки 35-37).

Таблица 2 – Примеры с использованием L-систем

Параметры входного файла	Результат изображения
3 5 90 F-F-F-F F->F-F+F+FF-F-F+F	 Рисунок 35 – Изображение кривой Коха №2
4 5 22.5 F F->FF-[-F+F+F]+[+F-F-F]	 Рисунок 36 – Изображение структуры растения №1

Продолжение таблицы 2

Параметры входного файла	Результат изображения
5 3 22.5 X X->F-[[X]+X]+F[+FX]-X F->FF	 Рисунок 37 – Изображение структуры растения №2

Подводя итог данного этапа работы, мы рассмотрели реализацию программного кода, алгоритмов построения фрактальных множеств и его функций.

Выводы

В третьей главе описаны архитектуры программ и принципы их работ. Выбрана технология PyCharm Community Edition 2022.3.3 для реализации алгоритмов на языке программирования Python. Рассмотрены инструменты, библиотеки и фреймворки.

Протестированы программы реализующие алгоритмы построения множеств Жюлиа, Мандельброт, кривой Коха и растительных структур.

Заключение

В настоящей выпускной квалификационной работе реализованы алгоритмы построения фрактальных множеств. В ходе выполнения бакалаврской работы выполнены поставленные задачи и достигнута цель.

В первой главе рассмотрены теоретические основы фрактальных множеств. В частности, определены основные понятия теории фракталов, рассмотрены свойства фрактальных множеств, описано понятие самоподобия и фрактальной размерности. Также рассмотрены методы нахождения фрактальной размерности Минковского и Хаусдорфа-Безиковича для треугольника Серпинского, множества Кантора и кривой Коха. Проведен сравнительный анализ этих фракталов.

Во второй главе представлены математические модели фрактальных множеств. Далее предложены алгоритмы построения множеств Мандельброта, Жюлиа и кривой Коха. Таким образом, в результате анализа алгоритмов построения фрактальных множеств было установлено, что каждый из них может быть использован для создания различных типов фракталов в зависимости от поставленной задачи. Все эти алгоритмы открывают новые возможности для создания сложных фрактальных структур.

В третьей главе описаны архитектуры программ и принципы их работ. Выбрана технология PyCharm Community Edition 2022.3.3 для реализации алгоритмов на языке программирования Python. Рассмотрены инструменты, библиотеки и фреймворки. Протестированы программы реализующие алгоритмы построения множеств Жюлиа, Мандельброт, кривой Коха и растительных структур.

Результатом данной бакалаврской работы являются программы построения фрактальных множеств.

Список используемой литературы и используемых источников

1. Божокин С.В., Паршин Д.А. Фракталы и мультифракталы. - Ижевск: НИЦ «Регулярная и хаотическая динамика», - 2001. - 128 с.
2. Гелашвили Д.Б. Фракталы и мультифракталы в биоэкологии: Монография / Гелашвили Д.Б., Иудин Д.И., Розенберг Г.С., Якимов В.Н., Солнцев Л.А.— Нижний Новгород: Изд-во Нижегородского госуниверситета, 2013. — 370 с.: ил.
3. Графическое моделирование живой природы. Фракталы [Электронный ресурс]. Режим доступа: Министерство Образования РФ (unn.ru) (Дата обращения 11.02.2023)
4. Деменок С.Л. Просто фрактал. – СПб.: ООО «Страта», 2012. – 168 с.
5. Иудин Д.И. Фракталы: от простого к сложному / Д.И. Иудин, Е.В. Копосов; Нижегор. гос. архитектур. -строит. ун-т – Н. Новгород: ННГАСУ, 2012. –200 с.
6. Иудин Д.И., Гелашвили Д.Б., Розенберг Г.С. и др. Биологические и экологические аспекты теории перколяции // Успехи соврем. биол. – 2010. – Т. 130. – № 5. – С. 446–460.
7. Кроновер Ричард М. Фракталы и хаос в динамических системах: Пер. с англ. М.: Техносфера, 2006. 488 с.
8. Латыпова Н.В. Фрактальный анализ: учеб. Пособие / Латыпова Н.В. – Ижевск: Издательский центр «Удмуртский университет», 2020. – 120 с.
9. Мандельброт Б. Самоаффинные фрактальные множества, «Фракталы в физике». М.: Мир 1988 г.
10. Мандельброт Б. Фрактальная геометрия природы. – Москва: Институт компьютерных исследований, 2002, 656 стр.
11. Моисеев К.Г., Бойцова Л.В., Гончаров В.Д. Анализ динамики гумусного состояния почв фрактальными методами [Текст]// Агрофизика. - 2014. - №1(13).

12. Мульфрактальный математический анализ синергетических структур // Труды Международной научно-практической интернет-конференции «Перспектива и развитие». - М.: МФТИ, 2004. (соавторы Малинников В. А., Никольский А. Е., Учаев Д. В.).

13. Насонов А.Н., Цветков И.В., Жогин И.М., Кульнев В.В., Репина Е.М., Кирносов С.Л., Звягинцева А.В., Базарский О.В. Фракталы в науках о земле: учебное пособие. Воронеж: Изд-во «Ковчег». – 2018. – 82 с.

14. Остапчук А.К, Овсянников В.Е. Применение теории фракталов в математическом моделировании и технике. - Курган: Курганский гос. ун-т, 2009.

15. Пайтген Х. –О., Рихтер П. Х. Красота фракталов. Образы комплексных динамических систем: Пер. с англ. –М.: Мир, 1993. – 176 с., ил.

16. Пригожин И., Стенгерс И. Порядок из хаоса. Новый диалог человека с природой: Пер. с англ. М.: Эдиториал УРСС, 2001. 312 с.

17. Принцип разработки алгоритма построения изображений фракталов [Электронный ресурс] – Режим доступа: https://studopedia.ru/19_376587_printsip-razrabotki-algoritma-postroeniya-izobrazheniy-fraktalov.html (Дата обращения: 23.04.2023)

18. Тарасенко В.В. Фрактальная логика. – М.: «Прогресс-Традиция», 2002. – 160.

19. Фракталы как искусство: сборник статей / пер. Николаева Е.В. - Санкт-Петербург: Страта, 2015. - 244 с.

20. Фрактальный анализ в флювиальной геоморфологии: монография / Под ред. Иванова А.И., Позднякова А.В. – М.: Издательство «Университетская книга», 2013. 188 с.

21. Шайдук А.М. Влияние фрактальной размерности сигнала на распределение энергии в его спектре / А.М. Шайдук, С.А. Останин // Журнал радиоэлектроники. - 2016. - №2.

22. Alessandro Chiaudani, Diego Di Curzio, William Palmucci, Antonio Pasculli, Maurizio Polemio, Sergio Rusi Statistical and Fractal Approaches on Long

Time-Series to Surface-Water/Groundwater Relationship Assessment: A Central Italy Alluvial Plain Case Study // Water. - 2017. - №9 (11).

23. Deng Y., Liu X., Zhang Y. Fractal Dimension Analysis of the Julia Sets of Controlled Brusselator Model // Discrete Dynamics in Nature and Society. - 2016.

24. Devaney, Robert L. A first course in chaotic dynamical systems: theory and experiment / Robert L. Devaney, 1948.

25. Fractals in the Biological Sciences [Электронный ресурс] – Режим доступа: 1996.pdf - Yandex.Documents (Дата обращения: 21.04.2023)

26. Fractals, Lindenmayer-Systems and Dimensions [Электронный ресурс] – Режим доступа: MSc_EducationMath_2017_Lok_K.

27. Theiler J. Estimation fractal dimension [Электронный ресурс] / LiLincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts. – Vol. 7. - №6. – 1990, 19 с. Режим доступа: https://public.lanl.gov/jt/Papers/est-fractal-dim.pdf (Дата обращения: 02.05.2023)