УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ «ДНІПРОВСЬКИЙ ІНСТИТУТ ІНФРАСТРУКТУРИ І ТРАНСПОРТУ»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ НАУКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ «ДНІПРОВСЬКИЙ ІНСТИТУТ ІНФРАСТРУКТУРИ І ТРАНСПОРТУ»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

ЧИГІР РОБЕРТ РОМАНОВИЧ

УДК 004.42:[004.94:514.74+519.76]+004.92

ДИСЕРТАЦІЯ

КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ ФРАКТАЛІВ

122 – Комп’ютерні науки
12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Р.Р.Чигір

Науковий керівник: Шинкаренко Віктор Іванович, д.т.н., професор

Дніпро – 2025

АНОТАЦІЯ

Чигір Р.Р. Конструктивно-продукційне моделювання фракталів. — Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 – Комп’ютерні науки (галузь знань 12 – Інформаційні технології). – Український держаний університет науки і технологій, Дніпро, 2025.

Дисертація присвячена розробці та дослідженню методів і засобів моделювання фракталів на основі конструктивно-продукційного моделювання.

У дисертаційній роботі отримані нові науково обґрунтовані теоретичні та експериментальні результати, що у сукупності дозволять реалізувати принципи моделювання на основі конструкторів та проводити багаторазові дослідження зі фракталами та бієктивними відображеннями їх з різною елементною базою носія.

Фрактали знаходять практичне застосування в багатьох сферах діяльності, що підтверджує важливість розробки нових підходів до їх конструювання. У сфері комп'ютерної графіки фрактали є інструментом для створення складних візуальних ефектів і моделювання природних явищ, таких як річкові мережі та гірські ландшафти. Фрактальна графіка базується на фрактальній геометрії і дозволяє описати складні об'єкти за допомогою всього лише декількох математичних виразів. Для прогнозу фізико-механічних характеристик різних матеріалів на підставі аналізу їх структури застосовуються методи математичного моделювання, включаючи фрактальну геометрію.

Однак, фрактальні моделі, що базуються на класичних системах геометричних перетворень, мають обмеження: неможливо повторно використовувати прості моделі у складних та мультиструктурних композиціях. Це створює потребу в нових підходах. У дослідженні пропонується новий підхід до формування фракталів: використання проміжної форми у вигляді мультисимвольної послідовності, що отримується методами L-систем. Важливість L-систем у фрактальному моделюванні не можна недооцінювати, оскільки їх рекурсивна природа легко призводить до самоподібності.

Актуальність конструктивно-продукційного підходу до моделювання фракталів підтверджується сучасними дослідженнями. Цей підхід, заснований на формальних граматиках, забезпечує гнучкі можливості з налаштування генерації фракталів, дозволяє формувати їх з неоднорідних елементів і комбінувати різні підходи, що значно розширює можливості практичного застосування. Конструктивно-продукційне моделювання, що поєднує апарат формальних граматик із принципами об’єктно-орієнтованого та компонентного проєктування, надає можливості для породження та трансформації складних об’єктів на базі конструктору. Це дозволяє формувати ієрархії різнорідних фрактальних фігур з різною елементною базою носія, підтримувати адаптивну зміну параметрів і багаторазово використовувати вже розроблені фрагменти моделей. Така гнучкість є надзвичайно важливою для сучасних задач комп’ютерної графіки та симуляцій природних явищ.

Предметом дослідження є методи, моделі та програмні засоби для формування фракталів з проміжним представленням у вигляді мультисимвольних послідовностей на основі конструктивно-продукційного моделювання.

Наукова новизна отриманих результатів полягає в наступному:

- розроблено конструктивно-продукційний метод формування фракталів з проміжним конструюванням у вигляді мультисимвольних послідовностей. Як і найближчий аналог – L-системи – він забезпечує фрактальність (самоподібність) формуємих конструкцій різної елементної бази носія з неочевидною самоподібністю, та на відміну від L-систем вирізняється гнучкістю при формуванні модифікацій та споріднених фракталів. Встановлено, що більшість класичних підходів не забезпечують необхідного рівня керованості та універсальності для створення складних морфологічних структур, особливо в умовах зміни параметрів у режимі реального часу.;
- виконано класифікацію конструкторів. На відміну від інших, сформована таксономія поділяє конструктори за функціональним призначенням та можливостями їх комбінування;

- розроблено конструктивно-продукційні моделі фракталів з різною елементною базою з бієктивним відображенням у декількох формах. На відміну від класичних L-систем, формування моделей виконується з оглядом на предметну область конструктору та використовується більш складні форми представлення об’єктів;

- для формалізації зв’язків з керування та даних між сукупністю конструкторів, що вирішують єдину задачу конструювання, розроблено модель мультиконструктора з її візуалізацією;

- встановлено для класичних геометричних фракталів зі стохастичними відхиленнями при їх формуванні властивості самоподібності та фрактальної розмірності не взаємопов’язані. У результаті експериментальних досліджень встановлено, що при зміні довжини лінії на кожному кроці формування класичних геометричних фракталів з математичним очікуванням та дисперсією, що дорівнюють одиниці, фрактальна розмірність може варіюватися в межах 5…15%.;

- на основі розроблених моделей запропоновано спосіб встановлення бієктивного відображення конструкцій з різною елементною базою носія. Встановлені залежності фрактальної розмірності класичних геометричних фракталів від стохастичної варіативності довжини лінії та кута нахилу, та на їх основі зроблені висновки щодо того, що при їх формуванні властивості самоподібністі та фрактальної розмірності не мають однозначного зв’язку;

- розроблено кросплатформене програмне забезпечення «Конструктор 1.1» та браузерне «Конструктор 2.0», яке втілює запропоновані підходи та забезпечує повний цикл формування конструкторів. Розроблене програмне забезпечення є у деякій мірі уніфіковане, що надає можливість його використання у майбутніх дослідженнях з застосуванням конструктивно-продукційного моделювання.

- удосконалено методи формування та розробки конструкторів при дослідженні з використанням конструктивно-продукційного моделювання, зокрема, сформульовано принципи конструктивної парадигми сприйняття світу та відповідних моделей. В результаті узагальнення методів і засобів конструктивно-продукційного моделювання представлених у багатьох попередніх роботах різ-

них дослідників, сформульовані основні положення конструктивної парадигми та розроблена таксономія конструктивно-продукційного моделювання, що сприятиме подальшому використанню цієї парадигми

- отримала подальший розвиток теорія фракталів у частині взаємозв’язку властивостей фракталів: самоподібності та фрактальної розмірності.

У першому розділі проведено комплексний аналіз сучасного стану проблеми моделювання фракталів та конструктивно-продукційних систем. Розглянуто теоретичні засади фрактальної геометрії, методи на основі ітеративних функціональних систем, L-систем та теорії хаосу. Проаналізовано апарат формальних граматик та його еволюцію. Систематизовано теоретико-методологічні засади конструктивізму. За результатами аналізу виявлено ключові обмеження існуючих підходів, зокрема відсутність уніфікованого інструментарію для композиційного моделювання, та обґрунтовано доцільність розробки нової, універсальної парадигми на засадах конструктивізму.

У другому розділі викладено теоретичні засади запропонованого конструктивно-продукційного підходу. Сформульовано основоположні принципи конструктивної парадигми. Введено формальне визначення узагальненого конструктора та представлена послідовність уточнюючих перетворень (спеціалізація, інтерпретація, конкретизація, реалізація) як механізм переходу від абстрактної моделі до конкретної. Розроблено оригінальну таксономію конструкторів за функціональним призначенням та способом взаємодії. Запропоновано модель мультиконструктора як засобу композиції та оркестрації взаємопов'язаних конструкторів.

У третьому розділі описано розробку інструментального програмного забезпечення «Конструктор 1.1» та «Конструктор 2.0», що реалізує запропоновані теоретичні положення. Визначено функціональні вимоги та задачі програмного забезпечення. Продемонстровано функціональність середовища на прикладах створення та уточнення конструкторів для моделювання геометричних фракталів.

У четвертому розділі представлено результати моделювання фрактальних структур з використанням розробленого інструментарію. Розроблено конструктивно-продукційні моделі для класичних геометричних фракталів, фрактальних ча-

сових рядів (на прикладі моделювання грозового фронту), мультифракталів та тривимірних фрактальних поверхонь (на прикладі китайської пагоди). Продемонстровано можливості бієктивного відображення між фракталами різної природи. Проведено експериментальне дослідження залежності фрактальної розмірності від стохастичних відхилень, що підтвердило відсутність прямого зв'язку між самоподібністю та фрактальною розмірністю для стохастичних фракталів.

Таким чином, у дисертаційній роботі вирішено актуальну науково-прикладну задачу формалізованого представлення, генерації та візуалізації фрактальних структур на основі конструктивно-продукційного підходу. Розроблено відповідну теоретичну модель, визначено алгоритми побудови та трансформації конструкцій, а також створено програмне забезпечення, що забезпечує їхню реалізацію з урахуванням параметричної варіативності, контекстних залежностей та можливостей інтерактивного керування.

Ключові слова: фрактал, фрактальні часові ряди, фрактальна розмірність, конструктивно-продукційне моделювання, L-система, конструктор, формальні граматики, бієктивне відображення, програмне забезпечення, інформаційні технології.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Праці у фахових виданнях МОН України

1. Шинкаренко, В. І., ***Чигір, Р. Р.*** (2024). Інструментальні засоби конструктивно-продукційного моделювання. Проблеми програмування, (2–3), 107–115. https://doi.org/10.15407/pp2024.02-03.107

2. Шинкаренко, В. І., ***Чигір, Р. Р.*** (2025). Конструктивно-продукційне моделювання тривимірних фрактальних поверхонь. Системні технології, 1(156), 78–88. https://doi.org/10.34185/1562-9945-1-156-2025-09

3. Шинкаренко, В. І., ***Чигір, Р. Р.*** (2025). Конструктивно-продукційне моделювання грозового фронту з використанням генетичного алгоритму. Системні технології, 4(159), 189–199. https://doi.org/10.34185/1562-9945-4-159-2025-19

Праці міжнародних конференцій, включені до міжнародної наукометричної бази Scopus

4. Shynkarenko, V., Lytvynenko, K., ***Chyhir, R.***, Nikitina, I. (2020). Modeling of lightning flashes in thunderstorm front by constructive production of fractal time series. In Advances in Intelligent Systems and Computing IV: CSIT 2019 (AISC, Vol. 1080, pp. 173–185). Springer, Cham. https://doi.org/10.1007/978-3-030-33695-0_13

5. Shynkarenko, V., Lytvynenko, K., ***Chyhir, R.***, Sansiieva, I. (2019). Constructive modeling of lightning activity in thunderstorm front. In 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 92–95). IEEE. https://doi.org/10.1109/STC-CSIT.2019.8929754

6. Shynkarenko, V., Letuchyi, O., ***Chyhir, R.*** (2023). Constructive-synthesizing modeling of fractal crystal lattices. In 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) (pp. 1–4). IEEE. https://doi.org/10.1109/CSIT61576.2023.10324251

7. Shynkarenko, V., ***Chyhir, R.*** (2024). Constructive-synthesising modelling of multifractals based on multiconstructors. In Proceedings of the 14th International Scientific and Practical Programming Conference (UkrPROG 2024) (CEUR Workshop Proceedings, Vol. 3806, pp. 75–88).

Праця у міжнародному виданні

8. Шинкаренко, В. И., Литвиненко, К. В., ***Чигирь, Р. Р.*** (2019). Конструктивное соответствие мультисимвольных и линейных геометрических фракталов. Information Technologies & Knowledge, 13(1), 76–99.

Матеріали міжнародних наукових конференцій

9. Шинкаренко, В. И., Литвиненко, К. В., ***Чигирь, Р. Р.***, Жадан, А. А. (2018). Констуктивно-продукционное моделирование фракталов. В «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту». Матеріали міжнародної наукової конференції (с. 289–291). Херсон: ПП Вишемирський В.С.

10. Шинкаренко, В. И., Литвиненко, К. В., ***Чигирь, Р. Р.*** (2019). Восстановление фрактальных временных рядов. В «Матеріали міжнародної науково-технічної конференції «Інформаційні технології в металургії та машинобудуванні»» (с. 83). ІВК «Системні технології».

11. Shynkarenko, V., Nikitina, I., ***Chyhir, R.*** (2020). Constructive-synthesizing modeling of lightning flashes in the dynamic thunderstorm front. In «Advances in Intelligent Systems and Computing V»: CSIT 2020 (AISC, Vol. 1293, pp. 1128–1145). Springer, Cham. https://doi.org/10.1007/978-3-030-63270-0_76

12. Шинкаренко, В. І., Литвиненко, К. В., ***Чигір, Р. Р.***, Жадан, А. А. (2017). Вариативность уточняющих преобразований конструктивно-продукционного моделирования. В «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD'2017) (с. 225–230). Київ.

13. Шинкаренко, В. І., ***Чигір, Р. Р.*** (2023). Використання спрощеної форми проєкту предметно-орієнтованого програмного забезпечення. В Тези доповідей XVII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (с. 99). Дніпро, УДУНТ.

14. Шинкаренко, В. І., Литвиненко, К. В., ***Чигір, Р. Р.*** (2018). Констуктивно – продукційне моделювання стохастичних процесів. В Тези доповідей XII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (с. 118). Дніпропетровськ, ДНУЗТ.

Матеріали наукових конференцій

15. Шинкаренко, В. І., ***Чигір, Р. Р.*** (2020). Описання продукційних конструкторів, що породжують фрактальні зображення. В Всеукраїнська конференція студентів та молодих вчених «Інформаційно-управляючі технології і системи на залізничному транспорті» (с. 33). Дніпро.

16. Шинкаренко, В. И., Литвиненко, К. В., ***Чигирь, Р. Р.***, Жадан, А. А. (2018). Констуктивно-продукционное моделирование фрактальных временных рядов на

основе L-систем. В Проблеми математичного моделювання. Матеріали Всеукраїнської науково-методичної конференції (с. 161–163). Кам'янське: ДДТУ.

Свідоцтва про реєстрацію авторського права

17. Шинкаренко, В. І., ***Чигір, Р. Р.***, Жадан, А. А. (2019). Комп'ютерна програма «Рекурентний аналіз часових рядів породжуваних в L-системах». Рішення про реєстрацію договору, який стосується права автора на твір № 4367 від 29.05.2019.

18. Шинкаренко, В. І., ***Чигір, Р. Р.,*** Жадан, А. А. (2019). Комп'ютерна програма «Моделювання взаємопов'язаних часових рядів й геометричних фракталів породжуваних в L-системах». Рішення про реєстрацію договору, який стосується права автора на твір № 4369 від 31.05.2019.

19. Шинкаренко, В. І., ***Чигір, Р. Р.,*** Жадан, А. А. (2017). Комп'ютерна програма «Моделювання часових рядів із заданими фрактальними властивостями». Свідоцтво про реєстрацію авторського права на твір № 72579 від 27.06.2017.

Додаткова доповідь на міжнародній конференції, що включена до міжнародної наукометричної бази Scopus

20. Shynkarenko, V., Raznosilin, V., Snihur, Y., ***Chyhir, R.*** (2022). Experimental research of educational content tracking by students group for distance learning. In V. Ermolayev et al. (Eds.), Information and Communication Technologies in Education, Research, and Industrial Applications: ICTERI 2021, Revised Selected Papers (CCIS, Vol. 1698, pp. 229–251). Springer, Cham. https://doi.org/10.1007/978-3-031-20834-8_11

ABSTRACT

Chyhir R.R. Construction-synthesizing modeling of fractals. — Qualifying scientific work on manuscript rights.

Dissertation for the degree of PhD in specialty 122 "Computer Science" - Ukrainian State University of Science and Technology, Dnipro, 2025.

The dissertation is devoted to the research and development of various methods and tools for modeling fractals based on constructive-synthesizing modeling.

The thesis provides new scientifically grounded theoretical and experimental results that will allow implementing the principles of constructor-based modeling and conducting repeated studies with fractals and bijection reflections with different elemental carrier bases.

Fractals find practical application in many areas of activity, which confirms the importance of developing new approaches to their construction. In the field of computer graphics, fractals are a tool for creating complex visual effects and modeling natural phenomena, such as river networks and mountain landscapes. Fractal graphics is based on fractal geometry and allows you to describe complex objects using just a few mathematical expressions. Mathematical modeling methods, including fractal geometry, are used to predict the physical and mechanical characteristics of various materials based on the analysis of their structure.

However, fractal models based on classical systems of geometric transformations have limitations: it is impossible to reuse simple models in complex and multistructural compositions. This creates a need for new approaches. The study proposes a new approach to the formation of fractals: the use of an intermediate form in the form of a multi-character sequence obtained by L-systems. The importance of L-systems in fractal modeling cannot be underestimated, since their recursive nature easily leads to self-similarity.

The relevance of the constructive-synthesizing approach to fractal modeling is confirmed by modern research. This approach, based on formal grammars, provides flexible possibilities for setting up the generation of fractals, allows them to be formed from heterogeneous elements and combine different approaches, which significantly

expands the possibilities of practical application. Constructive-synthesizing modelling, which combines the apparatus of formal grammars with the principles of object-oriented and component design, provides opportunities for generating and transforming complex objects based on a constructor. This allows you to form hierarchies of heterogeneous fractal shapes with different elemental carrier bases, support adaptive parameter changes, and reuse already developed model fragments. Such flexibility is extremely important for modern computer graphics and natural phenomena simulation tasks.

The subject of the study is methods, models and software tools for the formation of fractals with intermediate representation in the form of multi-character sequences based on constructive-synthesizing modeling.

The scientific novelty of the results is as follows:

- a constructive-synthesizing method of forming fractals with intermediate construction in the form of multisymbol sequences has been developed. Like its closest analogue – L-systems – it provides fractality (self-similarity) of the formed structures of different element base of the carrier with non-obvious self-similarity, and unlike L-systems, it is flexible in the formation of modifications and related fractals. It has been established that most classical approaches do not provide the necessary level of controllability and versatility to create complex morphological structures, especially in the conditions of changing parameters in real time;
- classification of constructors. Unlike others, the formed taxonomy divides constructors by their functional purpose and possibilities of their combination;
- constructive-synthesizing models of fractals with different element base with bielectric display in several forms are developed. In contrast to classical L-systems, the formation of models is performed with regard to the subject area of the constructor and more complex forms of object representation are used;
- to formalise the control and data links between a set of constructors that solve a single constructive problem, a multiconstructor model with its visualisation has been developed;
- it is established that for classical geometric fractals with stochastic deviations in their formation, the properties of self-similarity and fractal dimensionality are not in-

terrelated. As a result of experimental studies, it was found that when the length of the line is changed at each step of the formation of classical geometric fractals with mathematical expectation and variance equal to one, the fractal dimension can vary within 5...15%;

- on the basis of the developed models, a method for establishing a bijection display of structures with different elemental base of the carrier is proposed. The dependences of the fractal dimension of classical geometric fractals on the hundred-hysteretic variability of the line length and the angle of inclination are established, and on their basis it is concluded that the properties of self-similarity and fractal dimension are not unambiguously related in their formation;
- cross-platform software ‘Constructor 1.1’ and desktop software ‘Constructor 2.0’ were developed, which embodies the proposed approaches and provides a full cycle of constructors' formation. The developed software is to some extent unified, which makes it possible to use it in future research with the use of constructive-synthesizing modeling.
- the methods of formation and development of constructors in research using constructive-synthesizing modeling have been improved, in particular, the principles of the constructive paradigm of world perception and corresponding models have been formulated. As a result of generalising the methods and tools of constructive-synthesizing modelling presented in many previous works by different researchers, the main provisions of the constructive paradigm were formulated and a taxonomy of constructive-synthesizing modeling was developed, which will facilitate the further use of this paradigm
- the theory of fractals was further developed in terms of the relationship between the properties of fractals: self-similarity and fractal dimension.

In the first section, a comprehensive analysis of the current state of the problem of modeling fractals and constructive-synthesizing systems is carried out. The theoretical foundations of fractal geometry, methods based on iterative functional systems, L-systems and chaos theory are considered. The apparatus of formal grammars and its evolution are analysed. The theoretical and methodological foundations of constructiv-

ism are systematised. The analysis reveals the key limitations of existing approaches, in particular the lack of unified tools for compositional modeling, and substantiates the expediency of developing a new, universal paradigm based on constructivism.

The second section outlines the theoretical foundations of the proposed constructive-synthesizing approach. The basic principles of the constructive paradigm are formulated. A formal definition of a generalised constructor is introduced and a sequence of refinement transformations (specialisation, interpretation, specification, realisation) is described as a mechanism for the transition from an abstract model to a concrete one. An original taxonomy of constructors by functional purpose and method of interaction is developed. The model of a multi-constructor as a means of composing and orchestrating interconnected constructors is proposed.

The third section describes the development of the instrumental software «Constructor 1.1» and «Constructor 2.0», which implements the proposed theoretical provisions. The functional requirements and tasks of the software are defined. The functionality of the environment is demonstrated on the examples of creating and refining constructors for modeling geometric fractals.

The chapter four presents the results of experimental modeling of fractal structures using the developed tools. Constructive-synthesizing models for classical geometric fractals, fractal time series (for example, modeling a thunderstorm front), multifractals and three-dimensional fractal surfaces (for example, a Chinese pagoda) are developed. The possibilities of bijection between fractals of different nature are demonstrated. An experimental study of the dependence of the fractal dimension on stochastic deviations is carried out, which confirms the absence of a direct relationship between self-similarity and fractal dimension for stochastic fractals.

Therefore, the thesis solves the actual scientific and applied problem of formalised representation, generation and visualisation of fractal structures based on the constructive-synthesizing approach. The corresponding theoretical model was developed, algorithms for constructing and transforming structures were determined, and software was created to ensure their implementation, taking into account parametric variability, contextual dependencies and interactive control capabilities.

Keywords: fractal, fractal time series, fractal dimension, constructive-synthesizing modeling, L-system, constructor, formal grammars, bijection, software, information technology.

ПЕРЕЛІК АБРЕВІАТУР ТА УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface (Інтерфейс прикладного програмування)

CAD/CAM – Computer-Aided Design / Computer-Aided Manufacturing (Системи автоматизованого проєктування / автоматизованого виробництва)

DAO – Data Access Object (Об'єкт доступу до даних)

DDD – Domain-Driven Design (Предметно-орієнтоване проєктування)

HTML – HyperText Markup Language (Мова гіпертекстової розмітки)

HTTP – Hypertext Transfer Protocol (Протокол передачі гіпертексту)

IFS – Iterated Function Systems (Системи ітераційних функцій)

JSON – JavaScript Object Notation (Об'єктна нотація JavaScript)

L-система – Система Ліндenмаєра

MDE/MDA – Model-Driven Engineering / Model-Driven Architecture (Модельно-орієнтована інженерія / Модельно-орієнтована архітектура)

OWL – Web Ontology Language (Мова веб-онтологій)

PDF – Portable Document Format (Портативний формат документів)

PIM – Platform-Independent Model (Платформо-незалежна модель)

PSM – Platform-Specific Model (Платформо-специфічна модель)

RDF – Resource Description Framework (Інфраструктура опису ресурсів)

SQL – Structured Query Language (Мова структурованих запитів)

UI – User Interface (Інтерфейс користувача)

UML – Unified Modeling Language (Уніфікована мова моделювання)

UUID – Universally Unique Identifier (Універсально унікальний ідентифікатор)

ІЗ – Інформаційне Забезпечення

КПМ – Конструктивно-продукційне моделювання

УЗ – Узагальнений Конструктор

$A\,|_X^Y$ – Алгоритм над даними з вхідної множини X зі значеннями з множини Y

C – Конструктор

C_A – Алгоритмічний конструктор

D – Фрактальна розмірність

M – Неоднорідний розширюваний носій

Ψ – Множина правил підстановки

Σ – Сигнатура відношень та пов'язаних з ними операцій

Λ – Інформаційне забезпечення конструювання

Ω – Множина конструкцій

ЗМІСТ

ВСТУП

Актуальність теми. Фрактали знаходять практичне застосування в багатьох сферах діяльності, що підтверджує важливість розробки нових підходів до їх конструювання. У сфері комп'ютерної графіки фрактали є інструментом для створення складних візуальних ефектів і моделювання природних явищ, таких як річкові мережі та гірські ландшафти. Фрактальна графіка базується на фрактальній геометрії і дозволяє описати складні об'єкти за допомогою всього лише декількох математичних виразів. Для прогнозу фізико-механічних характеристик різних матеріалів на підставі аналізу їх структури застосовуються методи математичного моделювання, включаючи фрактальну геометрію.

Однак, фрактальні моделі, що базуються на класичних системах геометричних перетворень, мають обмеження: неможливо повторно використовувати прості моделі у складних та мультиструктурних композиціях. Це створює потребу в нових підходах. У дослідженні пропонується новий підхід до формування фракталів: використання проміжної форми у вигляді мультисимвольної послідовності, що отримується методами L-систем. Важливість L-систем у фрактальному моделюванні не можна недооцінювати, оскільки їх рекурсивна природа легко призводить до самоподібності.

Фрактали та фрактальні часові ряди досліджуються у роботах A. Lindenmayer, P. Prusinkiewicz, B. Mandelbrot, H.-O. Peitgen, H. Jurgens, D. Saupe, В. Скалозуба, Я. Соколовського, О. Михальова, В. Белозерова, Л. Кіріченко, А. Гуди та інші.

Актуальність конструктивно-продукційного підходу до моделювання фракталів підтверджується сучасними дослідженнями. Цей підхід, заснований на формальних граматиках, забезпечує гнучкі можливості з налаштування генерації фракталів, дозволяє формувати їх з неоднорідних елементів і комбінувати різні підходи, що значно розширює можливості практичного застосування. Конструктивно-продукційне моделювання, що поєднує апарат формальних граматик із принципами об’єктно-орієнтованого та компонентного проєктування, надає можливості для породження та трансформації складних об’єктів на базі конструк-

тору. Це дозволяє формувати ієрархії різнорідних фрактальних фігур з різною елементною базою носія, підтримувати адаптивну зміну параметрів і багаторазово використовувати вже розроблені фрагменти моделей. Така гнучкість є надзвичайно важливою для сучасних задач комп'ютерної графіки та симуляцій природних явищ.

За останні роки конструктивно-продукційне моделювання знаходиться на розвитку, що забезпечується роботами В. Шинкаренко, В. Ільмана, В. Скалозуба, К. Литвиненко, Е. Куропятник, та інші.

Існує нагальна необхідність у розробці інструментарію, який би дозволив легко створювати та модифікувати фрактальні моделі для різних застосувань. Розробка моделі мультиконструктора, запропонована в роботі, дозволяє групувати конструктори в єдину систему, що поетапно, переходячи від одного до іншого, формує моделі фракталів у різних представленнях. Це забезпечує високий рівень варіативності та розширюваності за рахунок параметризації, дозволяючи породжувати цілі групи споріднених конструкцій.

Конструктивно-продукційне моделювання, що поєднує апарат формальних граматик із принципами об'єктно-орієнтованого та компонентного проєктування, надає можливості для породження та трансформації складних об'єктів на базі конструктору. Це дозволяє формувати ієрархії різнорідних фрактальних фігур з різною елементною базою носія, підтримувати адаптивну зміну параметрів і багаторазово використовувати вже розроблені фрагменти моделей. Така гнучкість підходить для сучасних задач комп'ютерної графіки та симуляцій природних явищ на прикладі росту рослин чи погодних явищ.

Даний підхід базується на засадах формальних граматик, що були сформовані A. Lindenmayer, K. Ruohonen, A. Isah, G. Rozenberg, C. Manning, H. Schutze, Н. Хомського, та інші.

Науково-прикладна задача формалізованого представлення, генерації та візуалізації фрактальних структур на основі конструктивно-продукційного підходу залишається не вирішеною.

Зв'язок роботи з науковими програмами, темами, планами. Результати роботи застосовані при розробці програмних засобів та дослідженнях у науково-дослідних роботах кафедри «Комп'ютерні інформаційні технології» Українського державного університету науки і технологій, а саме є частиною науково- дослідних робіт «Конструктивно-продукційне моделювання фракталів» (2018 р. № держреєстрації 0118U004215) та «Підвищення конкурентоспроможності залізничного транспорту на основі уніфікованих інтелектуальних технологій процесів перевезень та експлуатації парків технічних систем» (2018 р. № держреєстрації 0117U004392).

Мета та задачі дослідження. Метою роботи було розробити методи та засоби для моделювання фракталів на основі конструктивно-продукційного моделювання. Для досягнення мети були поставлені та вирішені задачі дисертаційного дослідження:

- виконати аналіз існуючих підходів та методів формування фракталів; конструктивно породжуючи систем;
- розробити модель мультиконструктору для поєднання конструкторів у єдину систему з узгодженням зв'язків між ними, з відповідною візуалізацією у вигляді логічної схеми;
- виконати таксономію конструктивно-продукційного моделювання;
- розробити метод формування фракталів з різною елементною базою носія на основі фрактальних мультисимвольних послідовностей;
- розробити інструментарій формування конструктивно-продукційних моделей фракталів;
- на основі розробленого інструментарію виконати моделювання фракталів з різною елементною базою носія: класичних геометричних фракталів, часових рядів, мультифракталів, об'ємних фракталів;
- визначити можливості бієктивного співставлення фракталів з різною елементною базою носія;
- провести дослідження залежності фрактальної розмірності класичних геометричних фракталів при наявності стохастичних відхилень при їх формуванні.

Об’єктом дослідження є процеси формування фракталів різного походження.

Предметом дослідження є методи, моделі та програмні засоби для формування фракталів з проміжним представленням у вигляді мультисимвольних послідовностей на основі конструктивно-продукційного моделювання.

Методи дослідження. Для вирішення задач, поставлених у дослідженні, були використані**:**

- конструктивно-продукційне моделювання при розробці конструкторів;
- функціональне моделювання IDEF0 для аналізу та розробки інструментарію;
- адаптоване предметно-орієнтоване проектування при визначені доменів предметної області дослідження;
- об’єктно-орієнтований аналіз визначених об’єктів предметної області.

Наукова новизна отриманих результатів.

Вперше:

- розроблено конструктивно-продукційний метод формування фракталів з проміжним конструюванням у вигляді мультисимвольних послідовностей. Як і найближчий аналог – L-системи – він забезпечує фрактальність (самоподібність) формуємих конструкцій різної елементної бази носія з неочевидною самоподібністю, та на відміну від L-систем вирізняється гнучкістю при формуванні модифікацій та споріднених фракталів;
- виконано класифікацію конструкторів. На відміну від інших, сформована таксономія поділяє конструктори за функціональним призначенням та можливостями їх комбінування;
- розроблено конструктивно-продукційні моделі фракталів з різною елементною базою з бієктивним відображенням у декількох формах. На відміну від класичних L-систем, формування моделей виконується з оглядом на предметну область конструктору та використовується більш складні форми представлення об’єктів;

- розроблено модель мультиконструктору, що на відміну від інших, поєднує конструктори та представляє схему обробки та передачі даних між ними;
- встановлено, що для класичних геометричних фракталів зі стохастичними відхиленнями при їх формуванні властивості самоподібності та фрактальної розмірності не взаємопов'язані;
- на основі розроблених моделей запропоновано спосіб встановлення бієктивного відображення конструкцій з різною елементною базою носія;
- розроблено інструментальний додаток для формування конструкторів. На відміну від інших, він представляє уніфікований інтерфейс до розробки конструкторів без прив'язки до конкретної предметної області.

Удосконалено методи формування та розробки конструкторів при дослідженні з використанням конструктивно-продукційного моделювання, зокрема, сформульовано принципи конструктивної парадигми сприйняття світу та відповідних моделей.

Отримала подальший розвиток теорія фракталів у частині взаємозв'язку властивостей фракталів: самоподібності та фрактальної розмірності.

Практичне значення отриманих результатів. Розроблений інструментарій для формування конструкторів дозволяє моделювати конструктивно-продукційні моделі конструкцій та конструктивних процесів різної природи та застосування.

Результати роботи застосовані при розробці програмних засобів та дослідженнях у науково-дослідних роботах кафедри «Комп'ютерні інформаційні технології» Українського державного університету науки і технологій, а саме є частиною науково- дослідних робіт № держреєстрації 0118U004215 та 0117U004392.

Впроваджено у навчальний процес та використовувалося для формування конструкторів моделей при викладені дисципліни «Ефективність інформаційних систем та комп'ютерних технологій» для аспірантів спеціальності 122 «Комп'ютерні науки».

Особистий внесок здобувача. Результати дисертаційної роботи опубліковано в статтях у співавторстві, де здобувачем виконано:

[69, 114, 115, 123, 126] – розробка програмного забезпечення та конструктивно-продукційних моделей фрактальних структур, проведення досліджень;

[121] – розробка програмного забезпечення, проведення досліджень з лінійних геометричних фракталів

[10, 70, 72, 127] – розробку програмного забезпечення, методів та моделей представлення блискавок та грозового фронту;

[116] – розробка програмного забезпечення для моделювання часових рядів;

[125] – розробка моделей, методів та програмного додатку моделювання конструкторів, проведення досліджень;

[124] – розробку методів організації програмного коду;

[68, 120] – формування моделей класичних лінійних фракталів з бієктивним відображенням.

А також, [128, 129, 130] – були отримані свідоцтва авторського права на комп'ютерні програми.

Апробація результатів дисертації. Основні положення та результати дисертаційної роботи доповідались та обговорювались на міжнародних та всеукраїнських наукових конференціях:

- «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD) (м. Київ, 2017);
- XII Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (м. Дніпропетровськ, 2018);
- Міжнародна наукова конференція «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту» (м. Херсон, 2018);
- Всеукраїнська науково-методична конференція «Проблеми математичного моделювання» (м. Кам'янське, 2018);
- IEEE 14th, 15th, 18th International Conference on Computer Sciences and Information Technologies (CSIT) (2019, 2020, 2023);
- Міжнародна науково-технічна конференція «Інформаційні технології в металургії та машинобудуванні» (м. Дніпро, 2019);

- "Information Theories and Applications" ITA 2019 (Varna, Bolgaria, 2019).
- Всеукраїнська конференція студентів та молодих вчених «Інформаційно-управляючі технології і системи на залізничному транспорті» (м. Дніпро, 2020);
- International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications (ICTERI) (2021);
- XVII Міжнародна науково-практична конференція «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (м. Дніпро, 2023);
- 14th International Scientific and Practical Programming Conference (UkrPROG) (2024).

Публікації. Результати дисертаційної роботи опубліковано в 19 наукових працях.

У фахових та рекомендованих Міністерством освіти і науки (МОН) України для публікацій результатів дисертації – 3.

Матеріали міжнародних конференцій, що індексуються МНМБ Scopus – 5.

У тезах доповідей міжнародних та всеукраїнських конференцій – 7.

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, 4 розділів, висновків, списку використаних джерел і додатків. Загальний обсяг дисертації становить 156 сторінок, в тому числі 113 сторінки основної частини, 72 рисунків, 10 таблиць, 3 додатки на 10 сторінках та список використаних джерел із 135 найменувань на 14 сторінках.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО МОДЕЛЮВАННЯ ФРАКТАЛІВ ТА КОНСТРУКТИВНО-ПРОДУКЦІЙНИХ СИСТЕМ

1.1 Історико-теоретичні засади фрактальної геометрії

Основні поняття фрактальної геометрії сформульовано в роботі Б. Мандельброта [38] як узагальнення і розвиток ідей А. Пуанкаре, П. Фату, Г. Кантора, Ф. Гаусдорфа. Зазначена робота призвела до появи багатьох праць прикладного характеру, в яких фрактальний підхід почали застосовувати для розв'язання практичних задач із теорії хаосу і динамічних систем [48, 87, 104], моделювання дендритів [86, 94, 96], фрактальних властивостей антен [100], трафіку відеосигналів, зв'язку та інтернету [106, 107], еластичності матеріалів [61], часових рядів [91], теплой вологоперенесенні [77, 101, 102, 103] та ін.

Відповідно до визначення Б. Мандельброта: фрактал – це структура, що складається з частин, які в якомусь сенсі подібні до цілого. Тому фракталом є такий об'єкт, який має властивість самоподібності, він більш-менш одноманітно влаштований на широкому діапазоні масштабів. У геометричному випадку самоподібність гарантує інваріантність за будь-якої зміни масштабу, однак це характерно тільки для регулярних, детермінованих фракталів. Якщо детермінований процес побудови фрактальної структури зашумлений випадковими впливами, тоді формуються стохастичні фрактали. Властивість самоподібності таких фракталів проявляється тільки при усередненні по всіх статистично незалежних реалізаціях об'єкта. Тому, частина фрактала при зміні масштабу не повністю інваріантна до початкового фрагмента, проте їхні статистичні характеристики збігаються.

Розроблення ефективних алгоритмів і програмних засобів для моделювання двовимірних фракталів і тривимірних фрактальних поверхонь продовжує залишатися в центрі уваги спеціалістів, які займаються розв'язанням інженерних, медичних, управлінських та інших задач [7, 9, 27, 84, 85, 92].

Розвиток фрактальної геометрії у напрямі формалізації методів побудови фрактальних структур привів до появи декількох принципово різних підходів, се-

ред яких найбільш поширеними є ітеративні функціональні системи, L-системи (системи Ліденмаєра) та методи, засновані на теорії хаосу. Кожен з цих методів має власну математичну природу, особливості застосування та область ефективного використання.

Ітеративний підхід у фрактальній геометрії є одним із фундаментальних методів побудови фракталів, який базується на повторному застосуванні певного трансформаційного правила або набору правил до початкової геометричної фігури. Цей підхід, відомий також як метод ітеративних функціональних систем (Iterated Function Systems, IFS), забезпечує ефективне моделювання самоподібних структур, що мають фрактальну природу.

Суть ітеративного підходу полягає в послідовному застосуванні до деякого початкового геометричного об'єкта (наприклад, точки або відрізка) набору функцій, які найчастіше є афінними стискаючими перетвореннями. На кожному ітераційному кроці результатом стає нова множина фігур, яка формується як об’єднання образів попередньої множини під дією кожної з функцій. При великій кількості ітерацій система сходиться до стійкої структури – атрактору, який і є фракталом.

Найбільш відомими прикладами, які ілюструють ітеративний підхід, є трикутник Серпінського, килими Серпінського, множина Барнслі чи фрактальне дерево. Усі ці структури утворюються шляхом багаторазового застосування простих правил, що дає змогу досягти високого рівня складності на глобальному рівні при збереженні локальної простоти.

Особливістю ітеративного підходу є можливість використання гри хаоса (chaos game) – стохастичного методу генерації фракталів, при якому на кожному кроці випадково вибирається одна з функцій з фіксованою ймовірністю і застосовується до поточної точки. Такий підхід забезпечує просту реалізацію побудови фракталів навіть без повного знання аналітичної структури.

У сучасних дослідженнях ітеративний підхід отримав розвиток у поєднанні з методами комп’ютерної графіки, штучного інтелекту та обробки зображень. Наприклад, представлено реалізацію IFS у реальному часі для синтезу зображень із

фрактальними характеристиками, що має застосування в генеративному мистецтві та ігрових рушіях [44] та запропонували використання фрактальних операторів як елементів еволюційних алгоритмів у задачах оптимізації [56].

Іншою значущою перевагою ітеративного підходу є його фрактальна компресія, яка дозволяє стискати графічну інформацію за рахунок опису зображення невеликим набором перетворень, здатних відтворити складну структуру з високою точністю. Це застосовується, зокрема, у задачах стиснення цифрових зображень без втрат або з контролем похибки [13].

L-системи, запропоновані А. Ліденмаєром для моделювання росту рослин, ґрунтуються на формальній граматиці, де символи перетворюються згідно з набором продукційних правил. Відмінною особливістю L-систем є можливість керування просторовими трансформаціями через інтерпретацію символів як інструкцій для «черепашачої» графіки. Завдяки цій властивості L-системи стали фундаментом для моделювання біологічних структур і синтезу природоподібних форм [8]. У сучасних дослідженнях розроблено ряд модифікацій класичних L-систем, зокрема стохастичні та контекстно-залежні, які дозволяють імітувати більш складні об’єкти [25]. Зокрема, продемонстровано формування фрактальних патернів шляхом чергування кількох L-систем, що дозволяє досягати багатшої морфології [21].

Ще одним шляхом генерації фракталів є використання ідей теорії хаосу, зокрема хаотичних ітерацій та атракторів.

Формування фракталів на основі теорії хаосу ґрунтується на властивостях динамічних систем, які, хоча й детерміновані, демонструють непередбачувану, чутливу до початкових умов поведінку. Фрактальні структури виникають як геометричне відображення динаміки таких систем і найчастіше пов’язані з хаотичними атракторами – стійкими множинами в фазовому просторі, до яких притягуються траєкторії системи при довготривалій еволюції. Ці атрактори мають складну структуру, яка є ніде не гладкою, самоподібною та фрактальною за своєю природою.

Одним із класичних прикладів фракталів, що виникають на основі теорії хаосу, є атрактор Лоренца – розв’язок тривимірної системи нелінійних диференціальних рівнянь, яка моделює конвекційні процеси в атмосфері. Цей атрактор утворює характерну "метеликову" форму в фазовому просторі і демонструє фрактальну розмірність приблизно 2.06 [35]. Інші приклади включають атрактори Рєсслера, Генона, а також множину Жюліа і множину Мандельброта, які виникають у результаті ітерації комплексних функцій.

Особливо ефективним з точки зору чисельної реалізації є метод хаотичної гри (chaos game), запропонований Майклом Барнслі. У цьому підході фрактал формується як множина точок, які генеруються шляхом повторного випадкового вибору і застосування одного з афінних перетворень. При цьому ймовірності вибору перетворення та його параметри можуть бути налаштовані таким чином, щоб моделювати динаміку хаотичної системи. Таким чином, метод хаотичної гри – це стохастична симуляція динаміки хаотичної системи, яка в результаті збігається до фрактальної структури [1].

У рамках дискретної динаміки особливої уваги заслуговують логістичне рівняння та інші ітеративні схеми, які демонструють перехід до хаосу через періодичне подвоєння.

У візуалізації фракталів із хаотичних систем використовують відображення Пуанкаре, які дозволяють проєктувати складну динаміку на нижчий вимір для зручнішого аналізу. У такій проєкції можна спостерігати фрактальні кластерні структури, які відповідають хаотичним режимам системи.

У новітніх дослідженнях хаос і фрактальність все частіше поєднуються в алгоритмах обчислювального моделювання. Наприклад, було запропоновано використання хаотичного фрактального пошуку (Chaotic Fractal Search) у задачах класифікації акустичних сигналів [26]. Такий підхід дозволяє ефективно досліджувати простори рішень високої розмірності за рахунок хаотичної чутливості до змін параметрів.

Крім того, фрактали на основі теорії хаосу знайшли широке застосування в криптографії, генеруванні псевдовипадкових послідовностей, аналізі фінансових

ринків та біоінформатиці. Наприклад, було продемонстровано побудову великої кількості маловідомих, але фрактально організованих атракторів, використовуючи прості нелінійні системи рівнянь [78].

1.2 Формальні граматики як апарат синтаксичного аналізу та специфікації мов

Формальні граматики становлять фундаментальну частину теорії формальних мов, яка широко застосовується у комп'ютерних науках, лінгвістиці, біоінформатиці та математичній логіці. Основна ідея формальної граматики полягає у визначенні набору правил для побудови правильних (допустимих) виразів або речень мови. Відповідно до класичної класифікації Ноама Хомського, всі формальні граматики поділяються на чотири основні типи: тип 0 (без обмежень), тип 1 (контекстно-залежні), тип 2 (контекстно-вільні) та тип 3 (регулярні) [55]. Ця ієрархія, відома як ієрархія Хомського, надає систематичний спосіб опису складності мов, які можуть бути згенеровані відповідними граматиками.

Тип 3, регулярні граматики, є найпростішими та використовуються у побудові лексичних аналізаторів. Контекстно-вільні граматики (тип 2), завдяки своїй здатності описувати вкладені структури, широко застосовуються у синтаксичному аналізі мов програмування. Типи 1 і 0 мають більшу обчислювальну потужність, але через свою складність рідко використовуються у практичних додатках [23].

Ієрархія Хомського, яка систематизує формальні граматики за зростаючим рівнем обчислювальної потужності, має широке застосування у різноманітних наукових дослідженнях. Її застосування не обмежується лише теоретичними дисциплінами, як-от математична логіка або інформатика, а простягається і до прикладних галузей, включаючи біоінформатику, нейронауку, лінгвістику та навіть еволюційну біологію.

Одним із головних напрямів є використання контекстно-вільних граматик (тип 2) у комп'ютерній лінгвістиці та мовознавстві. Наприклад, у роботах Huybregts та Bolhuis граматики Хомського застосовуються для аналізу синтаксичних структур у природній мові та порівняння мовної здатності людини з вокалізаціями птахів. Вони вказують на те, що ієрархічна організація, характерна для

людської мови, не має еквівалентів у вокалізаціях інших видів, підкреслюючи унікальність граматичного компонента мови [22].

У комп'ютерних науках ієрархія Хомського активно використовується для класифікації мов програмування та побудови компіляторів. Регулярні граматики (тип 3) застосовуються в лексичних аналізаторах, а контекстно-вільні – в синтаксичних парсерах. Це дозволяє ефективно описувати синтаксис мов, як-от Java або Python, з використанням BNF-подібних нотацій [54].

Іншою сферою застосування є біоінформатика, зокрема в моделюванні послідовностей ДНК і білків. Деякі дослідження пропонують використовувати контекстно-залежні граматики (тип 1) для опису структурних залежностей у вторинній структурі РНК, які не можна описати простими регулярними засобами [58].

Дослідницький інтерес також привертає поєднання граматик Хомського з теорією автоматів для побудови систем розпізнавання шаблонів, зокрема в штучному інтелекті та обробці зображень. Наприклад, стохастичні контекстно-вільні граматики використовуються для побудови моделей генерації природної мови, які можуть навчатися з текстових корпусів [39, 53].

Особливу увагу привертає клас L-систем, або систем Ліденмаєра, що виник у контексті моделювання біологічного росту, але згодом став окремим напрямом в теорії граматикоподібних систем.

L-системи (Lindenmayer systems) були запропоновані Аристидом Ліденмаєром у 1968 році для моделювання морфогенезу рослин [33, 49].

Основна відмінна риса L-систем щодо класичних граматик полягає у відсутності нетерміналів, атрибутивності терміналів, виконанні «паралельної» підстановки, порядку формування множини виведених конструкцій і аксіоми у вигляді початкової конструкції. Це дозволяє ефективно моделювати феномени, де важливими є одночасні процеси, як, наприклад, клітинний поділ у рослинах [12].

Класифікація L-систем відрізняється від класичної ієрархії Хомського. Наприклад, розрізняють детерміновані L-системи (D0L), стохастичні L-системи (S0L), а також контекстно-залежні та табличні варіанти. Ці типи не вписуються безпосередньо в стандартну ієрархію, оскільки використовують інші принципи

генерації мов, що, за словами деяких дослідників, робить їх ортогональними до Хомського поділу [29].

Однією з найважливіших сфер застосування L-систем є ботаніка та біомоделювання. L-системи використовуються для моделювання морфогенезу рослин, зокрема розгалуження, листкових структур, утворення квітів тощо. У роботах Prusinkiewicz та Lindenmayer детально описано, як детерміновані L-системи (D0L) і стохастичні L-системи (S0L) дозволяють відтворювати складні структури реальних рослин з високим ступенем візуальної точності [49]. Ці моделі знайшли застосування не тільки у біології, а й у комп'ютерній графіці, де використовуються для створення реалістичних візуалізацій флори.

Крім біології, L-системи застосовуються також у моделюванні розвитку тканин і органів, що важливо для біоінженерії та біомедицини. Наприклад, модифіковані контекстно-залежні L-системи дозволяють моделювати не тільки форму, а й функціональні характеристики органів, як-от судинні структури або нейронні мережі [19].

У сфері обчислювальної біології L-системи інтегруються з агентно-орієнтованими моделями для симуляції динаміки клітин і молекул у багатокомпонентних середовищах. Наприклад, L-системи використовуються для моделювання міжклітинної комунікації та процесів апоптозу – програмованої клітинної смерті [46].

У галузі архітектури та урбаністики дослідники також використовують розширені L-системи для автоматизованої генерації геометрій будівель і вуличних структур, які відповідають певним симетричним і рекурсивним законам розвитку [47]. Це дозволяє створювати складні та органічні структури, що імітують природні патерни росту.

У галузі штучного інтелекту та машинного навчання L-системи досліджуються як альтернатива до нейромереж для створення генеративних моделей. Наприклад, дослідження Krivochen показало, що L-системи можуть реалізовувати обчислювальні процеси, які складно моделювати класичними послідовними граматиками, включаючи породження вкладених структур і моделей пам'яті [30].

Також цікаво, що деякі модифікації L-систем використовуються в мистецтві та цифровому дизайні. Параметричні L-системи дають змогу художникам та дизайнерам створювати складні візуальні композиції з точним контролем над кожним етапом росту форми [76].

Таким чином, L-системи становлять унікальний підхід до формального моделювання, що об’єднує теоретичну строгість із прикладною універсальністю. Їх використання охоплює широкий спектр галузей – від біології до архітектури та інформатики.

Подальші дослідження зосереджуються на поєднанні формальних граматик із динамічними системами, клітинними автоматами та нейронними мережами. Наприклад, D. Krivochen зі співавторами описує шляхи від граматичних структур до хаосу через формалізм автоматів і паралельного обчислення [31]. З іншого боку, спостерігається тенденція до уніфікації формальних граматик із засобами символічного моделювання біологічних процесів, що відкриває нові перспективи для моделювання росту, розвитку та морфогенезу [30].

1.3 Теоретико-методологічні засади, формальний апарат та онтологічна підтримка конструктивізму

Конструктивно-продукційне моделювання є формальною парадигмою для систематичної побудови структур та процесів, що базується на концепції сприйняття світу як сукупності різноманітних структур [32]. Основна мета конструктивно-продукційного моделювання полягає в автоматизації формування таких структур шляхом формування строгої системи формалізації процесів і результатів побудови конструкцій із елементів, наділених певними властивостями [32]. Цей підхід глибоко укорінений у принципах математично-алгоритмічного конструктивізму, що акцентує увагу на структурованому та процедурному плані розробки моделей і систем [3].

На базі фундаментального принципу загальнонаукових досліджень – від часткового до загального, а потім від загального до часткового, виконано узагаль-

нення можливостей та особливостей різних модифікацій граматик і граматико-подібних систем із застосуванням конструктивного підходу [66].

Парадигма конструкційно-продукційного моделювання з’явилася у результаті спроби узагальнення відомих засобів формальних граматик. Як з’ясувалось мається велика кількість модифікацій граматик [42, 66], таких як мультисимвольні [24], програмні, стохастичні [50, 83], зважені, індексні [11], матричні [49], різні графічні [50] та граматико подібні L-системи [34], R-системи [62] тощо.

Основні переваги запропонованого підходу:

- засобами однієї моделі можна пов’язати конструкції з конструктивними процесами;
- оптимізувати або адаптувати структуру конструкції і процесів (складові, їх розташування, зв’язки);
- моделювати конструкції, які складно, або навіть неможливо отримати іншими засобами;
- виконувати співставлення конструкції;
- моделювати, аналізувати і прогнозувати конструктивні процеси.

Парадигма конструктивно-продукційного моделювання відрізняється від інших підходів своєю унікальною епістемологічною основою. Це не просто набір технік для формування систем, а прикладна епістемологія, що базується на філософії конструктивізму. Якщо багато інженерних парадигм, таких як модельно-орієнтована інженерія, зосереджені на абстрагуванні та трансформації артефактів, то конструктивно-продукційне моделювання фокусується на самому процесі творення. Ключові положення, що визначають теоретичне ядро конструктивно-продукційного моделювання, включають явне призначення властивостей елементам і операціям, концепцію розширюваного носія, а також інтеграцію моделі виконавця, що визначає алгоритмічне виконання операцій [74].

На відміну від багатьох інших формалізмів, де властивості (атрибути) асоціюються переважно з об'єктами, конструктивно-продукційне моделювання поширює цей принцип і на самі операції, що дозволяє створювати значно багатші та більш керовані моделі [63]. Одним із фундаментальних принципів є концепція ро-

зширюваного носія – динамічного середовища, яке розширюється в процесі побудови, що дозволяє природно моделювати відкриті та еволюціонуючі системи [74]. Крім того, конструктивно-продукційне моделювання явно включає в себе модель виконавця, так званий алгоритмічний конструктор, який визначає, як саме виконуються абстрактні операції, усуваючи розрив між декларативним описом системи («що будувати») та її процедурною реалізацією («як будувати») [64, 125].

Формальний апарат конструктивно-продукційного моделювання базується на двох ключових поняттях: «конструктори», що є абстрактними моделями, які втілюють елементи та їхні можливі операції, та «уточнюючі перетворення», що забезпечують перехід від абстрактної ідеї до конкретної реалізації [66, 125]. Конструктори систематично класифікуються за функціональним призначенням (генерувальні, перетворювальні, аналізуючі, оптимізуючі/адаптаційні, алгоритмічні) та за характером взаємодії із зовнішніми сутностями (автономні, параметричні, інтерактивні, багатокористувацькі) [125].

Процес розробки та застосування цих конструкторів підпорядковується строгій послідовності уточнюючих перетворень: специфікація (визначає точну предметну область побудови), інтерпретація (встановлює відповідність між абстрактними операціями та конкретними алгоритмами виконання) та конкретизація (охоплює визначення точних правил підстановки та початкових умов) [5, 125]. Ця тріада перетворень є нормативною основою, що забезпечує строгість та дисципліну в моделюванні складних систем [90].

З розвитком семантичних технологій конструктивно-продукційне моделювання отримало потужний імпульс завдяки інтеграції онтологічного представлення знань [32, 45]. На відміну від традиційних мов, таких як XML або діаграми сутність–зв’язок, онтології (що використовують мови OWL, RDF і SHACL) побудовані на логіці описів [16, 32]. Це дозволяє безпосередньо відображати більшу кількість галузевих правил і обмежень у самій моделі, а не вбудовувати їх неявно в програмний код [18, 32, 40]. У контексті конструктивно-продукційного моделювання онтології найчастіше використовуються для формалізації та представлення розширюваного носія конструктора, дозволяючи детально й однозначно описати

всі елементи предметної області, їхні класи, властивості та взаємозв'язки [32, 36]. Цей перехід до онтологічної підтримки перетворює конструктивно-продукційне моделювання на підхід, керований знаннями, де процес побудови може керуватися результатами логічного виведення [20]. Важливим свідченням потужності парадигми є її здатність до самомоделювання: сама онтологія конструктивно-продукційного моделювання була побудована за допомогою цієї ж методології, що демонструє рекурсивне застосування парадигми та її здатність до самомоделювання [98, 99].

1.4 Сфери застосування, практична реалізація та інструментальні засоби конструктивно-продукційного моделювання

Універсальність формального апарату конструктивно-продукційного моделювання дозволяє застосовувати його для вирішення широкого кола наукових та інженерних задач у різноманітних предметних областях [57]. Будь-яке явище, яке можна описати як процес побудови з елементарних частин за певними правилами, може бути формалізоване засобами конструктивно-продукційного моделювання [63].

Конструктивно-продукційне моделювання можна застосовувати для розв'язання задач формування, перетворення й аналізу конструкцій з різною елементною базою носія із застосуванням операцій зв'язування, підстановки, виведення та ін. операцій, а також правил підстановки. На основі такого підходу уявляється можливим моделювання та формалізація будь-яких конструктивних процесів у галузі інженерії, біології, інформаційних технологій, а також розширюється можливість урахування властивостей елементів, їхньої форми та зв'язку.

Відштовхуючись від загального підходу в зазначених роботах, вдалося розв'язати низку приватних завдань:

- адаптації алгоритмів стиснення до архівованих даних [67];
- удосконалення процесу ранжування альтернатив методом аналізу ієрархій [108];
- адаптації структур даних в оперативній пам'яті [109];

- вдосконалення структур зберігання даних у задачах виявлення плагіату [110].

Однією з найбільш розвинених сфер застосування є генерація та аналіз складних структур. Наприклад, у моделюванні геометричних фракталів конструктивно-продукційний підхід забезпечує значно більшу варіативність атрибутів та гнучкість порівняно з класичними методами [117]. Нещодавні дослідження демонструють моделювання складних тривимірних фрактальних поверхонь, таких як китайські пагоди [126], та кристалічних ґраток [65]. В обробці природної мови парадигма знаходить застосування в задачах лінгвістичного моделювання, зокрема для встановлення авторства та виявлення плагіату, де текст моделюється за допомогою конструкторів, що описують його як стохастичну граматику [131]. Для аналізу авторства розроблено дуальний підхід, що включає метод формування текстових моделей у вигляді правил підстановки з імовірнісними вагами та класичний метод прямого порівняння текстів [112].

В інженерії програмного забезпечення конструктивно-продукційне моделювання застосовується для побудови повної історії змін у вихідному коді [74], генерації коду для алгоритмів сортування [71] та встановлення відповідності між текстом програми та її алгоритмічним представленням [111]. Ідеї, близькі до конструктивно-продукційного моделювання, знаходять своє відображення і в інших інженерних дисциплінах, таких як конструктивний синтез логічних схем [5, 41] та конструктивна блокова геометрія (Constructive Solid Geometry) у системах CAD/CAM [51, 93].

Окрім моделювання статичних структур, конструктивно-продукційне моделювання ефективно застосовується для опису систем, що змінюються в часі, та для вирішення задач оптимізації. Яскравим прикладом є розробка моделі ділянки тягового електропостачання на залізничному транспорті для зниження споживання електроенергії шляхом оптимізації використання енергії рекуперації за допомогою методів нечіткої логіки [88, 89, 118, 122]. Конструктивні моделі також використовуються для аналізу, реконструкції та прогнозування стохастичних часових рядів, наприклад, для моделювання блискавичної активності в грозовому фронті, де для оптимізації параметрів моделі застосовуються генетичні алгоритми

[127, 133, 132]. Такий підхід знаходить застосування і в біомедичних дослідженнях для синтетичного моделювання властивостей тканин [2, 3, 81].

У роботах за конструктивно-продукційним моделюванням увага приділяється переважно формальним аспектам граматик або окремим алгоритмам генерації, тоді як саме підхід до створення, управління й повторного використання високорівневих конструкторів практично відсутній. Запропонована в дисертації концепція мультиконструктора та стандартизація до розробки на основі конструкторів заповнює цю прогалину, забезпечуючи інструментарій для опису зв’язаних конструкторів необхідною функціональною наповненістю.

Практична значущість роботи підтверджується розробкою програмної системи, що реалізує запропоновані принципи та надає зручний засіб швидкого прототипування та аналізу фрактальних структур. Система дозволяє варіювати правила продукцій, параметри генерації та перевиконувати уточнюючі перетворення, що скорочує трудомісткість моделювання.

1.5 Контекст сучасних наукових парадигм, невирішені проблеми та перспективи конструктивно-продукційного моделювання

Для повного розуміння сутності конструктивно-продукційного моделювання необхідно провести його порівняльний аналіз з іншими провідними парадигмами, зокрема з модельно-орієнтованою інженерією, а також дослідити його зв'язки з методами штучного інтелекту [3].

Модельно-орієнтована інженерія (Model-Driven Engineering) є методологією, що фокусується на створенні та використанні доменних моделей як первинних артефактів розробки, з яких автоматично генеруються інші артефакти, такі як вихідний код [4, 6, 57]. На перший погляд, обидві парадигми мають спільну мету, проте ключова відмінність полягає в тому, що модельно-орієнтована інженерія за своєю суттю є трансформаційною парадигмою, основний механізм якої – це перетворення однієї моделі в іншу [4]. Натомість конструктивно-продукційне моделювання є генеративною (продукційною) парадигмою, основний механізм якої – це побудова (конструювання) кінцевої структури з елементів за правилами [66]. В

модельно-орієнтованій інженерії центральною абстракцією є модель (наприклад, платформо-незалежна), а в конструктивно-продукційному моделюванні – конструктор, що функціонує в межах розширюваного носія [4, 90]. Детальний порівняльний аналіз представлено у табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика парадигм конструктивно-продукційного та модельно-орієнтованого моделювання

Критерій порівняння	Конструктивно-продукційне моделювання	Модельно-орієнтована інженерія (MDE/MDA)
Філософська основа	Математичний конструктивізм; світ як сукупність конструкцій, що будуються.	Абстрагування та розділення аспектів (separation of concerns).
Ключова абстракція	Конструктор (модель елементів та операцій) та розширюваний носій.	Платформо-незалежна модель (PIM) та платформо-специфічна модель (PSM).
Основний механізм	Генерація/побудова конструкцій за допомогою продукційних правил та уточнюючих перетворень.	Трансформація моделей з вищого рівня абстракції на нижчий (model-to-model, model-to-text).
Базовий формалізм	Формальні граматики, теорія множин, онтології (OWL, RDF).	UML, OCL, MOF; предметно-орієнтовані мови (DSL).
Роль виконавця	Явно моделюється через алгоритмічний конструктор.	Неявна; реалізована у трансформаційних рушіях та згенерованому коді.
Основний фокус	Процес побудови (конструювання) та його формалізація.	Артефакти розробки (моделі) та їх автоматизоване перетворення.

Інтеграція з методами штучного інтелекту є природним напрямом еволюції конструктивно-продукційного моделювання, оскільки продукційні правила є основою класичних експертних систем [95, 97, 105]. Еволюційні обчислення та ге-

нетичні алгоритми застосовуються для автоматизації вибору та налаштування конструкторів, наприклад, для створення адаптивних алгоритмів сортування [71] або для підбору параметрів фрактальних моделей блискавок [89, 135]. Нечітка логіка дозволяє моделям оперувати з нечіткими атрибутами та правилами, що продемонстровано в системі оптимізації розподілу енергії рекуперації, де на основі експертних даних та нечітких правил генеруються керуючі рішення [89].

Аналітичний огляд дозволяє виявити низку невирішених проблем: складність ручної побудови конструкторів для складних предметних областей [75], верифікація та валідація властивостей великомасштабних конструкцій, та недостатність високорівневого, предметно-орієнтованого інструментарію [59]. Це визначає перспективні напрями досліджень: автоматизований синтез конструкторів на основі методів машинного навчання [52], розвиток масштабованої онтологічної підтримки для верифікації [82], інтеграція з цифровими двійниками [28] та розробка уніфікованих середовищ моделювання [4, 64].

Таким чином, існує наукова лакуна у розробці моделей та методів, що дозволяють генерувати або адаптувати конструктивно-продукційні моделі на основі високорівневих специфікацій або даних моніторингу функціонування системи. Заповнення цієї прогалини дозволить суттєво знизити складність застосування конструктивно-продукційного моделювання та підвищити рівень автоматизації при розробці надійних та ефективних комп'ютерних систем, що і визначає актуальність та наукову новизну даного дисертаційного дослідження.

Висновки до першого розділу

Проведений аналітичний огляд сучасних підходів до генерації фракталів показав, що ітеративні функціональні системи, L-системи й методи хаотичного відображення формують репрезентативний набір інструментів для побудови самоподібних структур. Однак жоден з них не забезпечує повної підтримки повторного використання моделей та їх гнучкої інтеграції в багаторівневі композиції складних об’єктів.

Це зумовлює необхідність пошуку універсальнішої парадигми, яка поєднає переваги наведених методів і мінімізує їхні обмеження.

Еволюція формальних граматик, починаючи з класичної ієрархії Хомського та завершуючи спеціалізованими L-системами, засвідчила напрями розширення виражальних можливостей синтаксичних формальних апаратів. Зокрема, паралельна природа L-систем є ефективною для опису самоподібності, але потребує доповнення механізмами параметричного та контекстного керування, коли йдеться про реалістичні, неоднорідні структури.

Концепцію конструктивно-продукційного моделювання розглянуто як універсальний підхід, що поєднує граматичний апарат із принципами об’єктно-орієнтованого та компонентного проєктування.

У ході аналізу сформульовано базові вимоги до інструментарію моделювання фракталів у рамках конструктивно-продукційної парадигми. До таких вимог належать:

- підтримка уточнюючих перетворень, що охоплюють конкретизацію, спеціалізацію, інтерпретацію та реалізацію;
- можливість опису як автономних, так і взаємопов’язаних конструкторів;
- параметризація процесів генерації з подальшим адаптивним налаштуванням під різні класи фракталів.

На підставі узагальнених результатів окреслено наукову проблему, що полягає у відсутності єдиного формального та програмного середовища, здатного інтегрувати граматики, алгоритмічні конструктори й процедурні моделі для побудови плоских, тривимірних та стохастичних фракталів із гарантованою відтворюваністю результатів.

Подальші дослідження доцільно спрямувати на розроблення формальних моделей мультиконструкторів, методів породження фракталів на основі мультисимвольних послідовностей та створення програмної системи, що реалізує повний цикл «конструктор → модель → фрактальна конструкція». Запропоновані напрями становлять підґрунтя для формування нових алгоритмів і програмних засобів, розробка яких буде предметом наступних розділів дисертаційного дослідження.

Матеріали розділу опубліковані у роботах: [10, 69, 70, 125, 126, 127].

РОЗДІЛ 2
ТЕОРЕТИЧНІ ЗАСАДИ КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ ФРАКТАЛІВ

Маються різні інтерпретації поняття «Фрактал». Само поняття було введено Мандельбротом [37] та позначало структуру, що складається з частин, які в подібні до цілої структури.

Також, фракталом називають об’єкт, що має у собі прояв характеристики самоподібності у більший (повній) чи меншій самого себе (тобто повторення форми чи будови при виділені підоб’єкту з об’єкту).

Іншою можливою умовою класифікації об’єкту як фракталу є наявність дробової фрактальної розмірності в математичній чи графічній моделі об’єкту [17].

За Фальконом [15] визначають наступний перелік структурових, геометричних чи множинних властивостей об’єкту для означення його терміном «фрактал»:

- дробова фрактальна розмірність;
- ніде не диференційовані функції;
- **самоподібність – геометрична множина Q містить копії самої себе в багатьох різних масштабах;**
- множина Q має «тонку структуру», яка містить деталі в довільних малих масштабах;
- хоча Q має складну структуру, фактичне визначення Q дуже просте;
- для отримання Q застосовується рекурсивна процедура;
- геометрію Q нелегко описати в класичних термінах, це не геометричне місце точок, які задовольняють деяку просту умову і не множина розв’язків будь-якого простого рівняння;
- проблематично описати локальну геометрію Q ¬ поблизу кожної з її точок є велика кількість інших точок, розділених проміжками різної довжини;
- хоча Q в певному роді достатньо велика множина (незліченна), її розмір не визначається кількісно звичними мірами, такими як довжина, площа, об’єм.

При розгляді графічних фракталів (фракталів, що графічно представлені на площині), здебільшого як головну властивість ставлять розраховану дробову фрактальну розмірність отриманої фігури у межах мінімальної виділеної області, але важливо враховувати і елементну самоподібність в об’єкті.

У цій роботі основним критерієм фрактальності вважається наявність властивості само подібності.

В рамках ітеративних процедур генерації фрактальних структур виникає необхідність розрізнення між ідеальним математичним об'єктом та його скінченними наближеннями. У цьому контексті вводиться поняття «передфрактал». Передфрактал визначається як самоподібна фігура, в якій генеративний процес повторюється скінченну кількість разів. Кожен крок такого ітеративного процесу створює нову, більш детальну структуру, яка є передфракталом певного порядку. Теоретичний перехід від передфракталу до фракталу відбувається в границі, коли кількість ітерацій або порядок конструкції прямує до нескінченності.

У цьому дослідженні, як і в багатьох інших, будемо говорити про фрактал підрозуміваючи при цьому передфрактал.

У дослідженнях, що використовують конструктивно-продукційний підхід, властивість самоподібності посідає центральне місце. Самоподібність, яка визначається як властивість об'єкта, частина якого після масштабування виглядає як ціле, є не просто пасивною характеристикою кінцевого результату, а активним механізмом, закладеним в основу самого генеративного процесу.

Ітеративний підхід гарантує, що «кожен рівень структури є результатом модифікації попереднього згідно з чітко визначеними правилами», що дозволяє автоматично створювати нові рівні, «зберігаючи при цьому їхню самоподібність».

2.1 Основоположні принципи конструктивної парадигми

В основі методів та засобів, що пропонується в даній роботі, лежить конструктивно-продукційна парадигма, епістемологічне коріння якої сягає принципів математично-алгоритмічного конструктивізму [43, 125]. Центральною ідеєю цього підходу є твердження, що існування математичного або обчислювального об'є-

кта доводиться не через абстрактні міркування, а шляхом надання явного методу (алгоритму) для його побудови [5, 41]. Таким чином, доказ і конструкція стають синонімічними поняттями. Ця філософія знаходить пряме втілення в комп'ютерних науках, де програма є, по суті, конструктивним доказом існування розв'язку задачі.

В багатьох дослідженнях з використанням КПМ визначені деякі особливості цього підходу, які потребують узагальнення.

Сформулюємо основні положення конструктивної парадигми:

- світ сприймається як сукупність конструкцій і конструктивних процесів;
- конструкції складаються з деяких елементів і інших конструкцій;
- конструктивний процес складається з елементарних дій (дискретних або неперервних, детермінованих або стохастичних і т.д.) та інших процесів;
- елементи, конструкції, проміжні форми конструювання пов’язані між собою деякими відношеннями зв’язування;
- елементи конструкцій, проміжні форми конструювання, конструкції, відношення та операції – кожен має свій набір деякий атрибутів;
- при формуванні конструкцій використовується тріада: відношення → операція → відношення.

Останнє положення потребує деякого пояснення. У алгебрі, наприклад цілих чисел з заданою операцією додавання, запис $5-6$ сприймається як відношення: треба відняти 6 від 5, в результаті операції складання отримаємо результат -1. Тобто маємо відношення → операція: треба відняти і віднімання.

У конструктивній парадигмі: маємо відношення цвях можна забити у стіну. Це відношення є основою операції забиття цвяху у стіну. В результаті операції встановлюється нове відношення цвях забитий у стіну. Відношення між цвяхом і стіною змінилось за тріадою пов’язаних між собою відношень і операції: можна забити → операція забиття → забитий. У результаті операції залишаються всі аргумента (елементи) з новим відношенням.

2.2 Узагальнений конструктор та його уточнюючі перетворення

Основною конструювання є узагальнений конструктор [66]:

$$C = \langle M, \Sigma, \Lambda \rangle, \tag{2.1}$$

де M – неоднорідний розширюваний носій, Σ – сигнатура відношень та пов'язаних з ними операцій, Λ – інформаційне забезпечення конструювання: онтологія. призначення, умови початку та завершення конструювання, правила підстановки та обмеження.

Основні положення щодо узагальненого конструктору та його перетворень викладені у [64, 63].

Перехід від абстрактної ідеї, інкапсульованої в узагальненому конструкторі, до конкретної моделі, здатної генерувати фрактальні структури, здійснюється через послідовність уточнюючих перетворень [66, 121]: спеціалізації, інтерпретації, конкретизації та реалізації.

Ключовою особливістю цього процесу є чітке розмежування ролей між зовнішнім виконавцем, який вносить інтелектуальні рішення, та внутрішнім виконавцем – автоматизованою системою, яка виконує фінальну генерацію [125].

Першим кроком, який виконує зовнішній виконавець, є спеціалізація. Це перетворення обмежує універсальний потенціал узагальненого конструктора до конкретної предметної області. Спеціалізація визначає онтологію моделі, її словник та базові операції.

Спеціалізація конструктору визначається як:

$$C = \langle M, \Sigma, \Lambda \rangle \ {}_S\mapsto \ {}_S C = \langle M_S, \Sigma_S, \Lambda_S \rangle, \tag{2.2}$$

де M_S – включає у собі множину терміналів, Σ_S – набір операцій над потенційними елементами, та інформаційне забезпечення Λ_S – додатково до Λ містить множину елементів й операцій та їх опис.

Результатом спеціалізації є шаблон конструктора, адаптований до певного класу задач.

Наступний крок, інтерпретація, надає обчислювальний сенс абстрактним операціям. Зовнішній виконавець пов'язує кожну операцію з конкретним, виконуваним алгоритмом.

Інтерпретація конструктору ставиться як

$$\left\langle C_S = \left\langle M_S, \Sigma_S, \Lambda_S \right\rangle, C_A = \left\langle M_A, \Sigma_A, \Lambda_A \right\rangle \right\rangle_I \mapsto {}_I C_S = \left\langle M_{SI}, \Sigma_{SI}, \Lambda_{SI}, Z \right\rangle, \quad (2.3)$$

де C_A – алгоритмічний конструктор, Σ_{SI} – включає у собі сигнатуру послідовного та умовного виводу, та Λ_{SI} – інформаційне забезпечення, що включає множину операцій та алгоритмів, а також положення L-систем [49], Z – модель внутрішнього виконавця.

Інтерпретація полягає в збагаченні операцій з Σ_{SI} атрибутом: алгоритмом з алгоритмічного конструктору, здатним виконувати відповідну операцію.

Алгоритми над даними визначаються як $A\Big|_X^Y$, де X – вхідні дані, Y – вихідні дані.

До обов’язкових операцій конструктору належать операції повного та часткового виводу, а також операція підстановки.

Операція повного виводу, що позначається як $\|\Rightarrow$, визначається як послідовне виконання операцій часткового виводу. Процес починається з початкової конструкції (аксіоми) і триває доти, доки не буде виконано умову завершення, визначену в Λ. Результатом повного виводу є кінцева множина конструкцій Ω.

Операція часткового виводу, що позначається як $|\Rightarrow$, визначається як складена дія, що складається з трьох тісно пов'язаних кроків:

1) вибір відповідного правила підстановки з множини продукцій Ψ;

2) виконання цієї підстановки над конструкцією в носії M;

3) виконання операцій над атрибутами, що відповідають обраному правилу.

У межах ієрархії Хомського, центральною операцією є прямий вивід (однокроковий вивід), що полягає в локальній, контекстно-вільній заміні. У рядку, що складається із термінальних та нетермінальних символів, один нетермінал замінюється правою частиною відповідного продукційного правила. Цей процес є суто синтаксичним і оперує з лінійною послідовністю символів. Повний вивід ви-

значається як скінченна послідовність таких однокрокових виводів, що починається з початкового символу граматики (аксіоми) і завершується рядком, який містить виключно термінальні символи. Важливо підкреслити, що на кожному етапі об'єктом маніпуляції є символьний рядок, позбавлений внутрішніх семантичних атрибутів. Граматика визначає, що є синтаксично можливим, але не те, що згенерований рядок означає або якими властивостями він володіє.

Прикладом є початкова стрічка з символом F та правилами $F \to +fF - fF, F \to \varepsilon$

$$\begin{aligned} &n = 0: \ F \\ &n = 1: \ \overbrace{+fF - fF}^{F} \\ &n = 2: \ +fF - f\overbrace{+fF - fF}^{F} \\ &n = 3: +fF - f + f\overset{F}{\varepsilon} - fF \end{aligned} \tag{2.4}$$

У L-систем, замість послідовної заміни одного символу, усі застосовні правила продукції застосовуються одночасно до всіх символів у рядку протягом однієї ітерації. Цей паралельний перезапис є фундаментальною властивістю, що дозволяє моделювати процеси, де всі частини структури розвиваються одночасно.

Наприклад, при початковій стрічці F та правилах $F \to +F - G, G \to -G + F$ отримуємо:

$$\begin{aligned} &n = 0: F \\ &n = 1: \overbrace{+F - G}^{F} \\ &n = 2: +\overbrace{+F - G}^{F} - \overbrace{-G + F}^{G} \\ &n = 3: ++\overbrace{+F - G}^{F} - \overbrace{-G + F}^{G} - -\overbrace{-G + F}^{G} + \overbrace{+F - G}^{F} \end{aligned} \tag{2.5}$$

В більшості конструктивно-продукційних моделей операція часткового виводу аналогічна до класичних граматик. При конструктивно-продукційному конструюванні фракталів операція часткового виводу аналогічна до L-систем.

Третій крок – конкретизація. Зовнішній виконавець надає конструктивній системі всі необхідні вхідні та початкові дані для генерації однієї конкретної конструкції.

Конкретизація конструктору визначається як

$$C_{SI} = \langle M_{SI}, \Sigma_{SI}, \Lambda_{SI} \rangle \ {}_{K}\mapsto \ {}_{K}C_{SI} = \langle M_{SIK}, \Sigma_{SIK}, \Lambda_{SIK} \rangle, \tag{2.6}$$

де M_{SIK} – розширяємий носій що містить визначений набір термінальних та нетермінальних символів, Σ_{SIK} – множина операцій, Λ_{SIK} – інформаційне забезпечення процесу конструювання.

При конкретизації визначаються початкові значення даних, як початкова символьна послідовність та правила підстановки.

Носій M_{SIK} включає не термінальні та термінальні символи. Сигнатура операцій $\Sigma_{SIK} = \Sigma_{SI}$.

Множина правил підстановки визначається як

$$\psi = \langle s, g \rangle \in \Psi, \tag{2.7}$$

де s – відношення підстановки, g – набір операцій над атрибутами.

Визначається при конкретизації Λ_K наступне: мета конструювання, обмеження, початкові умови, правила підстановки, умова завершення.

Після визначення спеціалізації, інтерпретації та конкретизації, конструктивна система C_{SIK} готова для реалізації.

Останній крок, реалізація, виконується внутрішнім виконавцем (частіше за все апаратно-програмним середовищем). Система бере конкретизований конструктор і запускає процес генерації.

Реалізація конструктору визначається як

$$C_{SIK} = \langle M_{SIK}, \Sigma_{SIK}, \Lambda_{SIK} \rangle \ {}_{R}\mapsto \Omega(C_{SIK}), \tag{2.8}$$

де $\Omega(C_{SIK})$ – множина конструкцій породжувана конструктором C_{SIK}, яка в окремих випадках може складатися з однієї або жодної конструкції.

Наведена послідовність перетворень забезпечує керований підхід до моделювання, де кожен етап має чітке призначення та результат. Це дозволяє розділити складну задачу моделювання на низку простіших, логічно пов'язаних кроків.

2.3 Таксономія та композиція конструкторів

Конструктивно-продукційна парадигма полягає не лише в можливості створювати окремі моделі, а й у її модульній природі. На відміну від монолітних підходів, конструктивно-продукційна парадигма дозволяє розробляти спеціалізовані, багаторазово використовувані конструктори, які можна комбінувати для моделювання складних, багатоаспектних систем [125]. Ця модульність формалізується через складання таксономії конструкторів, яка класифікує їх за роллю та способом взаємодії, та через концепцію мультиконструктора, що забезпечує механізм їх композиції.

У роботі постає задача заповнити вказану прогалину, пропонуючи нову таксономію конструктивно-продукційних систем. За аналогією до успішних таксономічних підходів [73], запропонована система класифікує конструктори за шістьма ортогональними, але взаємозалежними вимірами: їхня функціональна мета («Що?»), метод їхнього уточнення та реалізації («Як?»), спосіб взаємодії та композиції («Чим?»), операційний домен («Де?»), керуючий агент («Хто?») та фаза життєвого циклу («Коли?»). Така багатовимірна структура дозволяє не лише каталогізувати існуючі типи конструкторів, але й забезпечує глибоке розуміння їхньої ролі та взаємозв'язків у складних моделюючих системах.

Кожен вимір, або «фасет», представляє окремий, ортогональний аспект конструктора, що дозволяє аналізувати його з різних точок зору. Фасети є ортогональними в тому сенсі, що будь-який конструктор можна класифікувати незалежно за кожним із шести вимірів. Водночас вони є взаємозалежними, оскільки вибір категорії в одному вимірі часто обмежує або передбачає можливі варіанти в інших, що відображає цілісність та системність проєктних рішень при створенні моделі.

Вимір «Що?» відповідає на питання: «Яка основна функція або мета цього конструктора в рамках загального процесу моделювання?» (рис. 2.1):

- **породжуючі**: основна функція – створення нових конструкцій "з нуля" на основі наявної елементної бази аксіоми та набору продукційних правил. Це основний тип конструкторів для генерації фрактальних структур. Приклад: конс-

труктор, що породжує мультисимвольний рядок для кривої Коха, ітеративно застосовуючи правило F→F+F− −F+F [117];

- **трансформуючі** : призначені для перетворення конструкції з одного представлення в інше, зберігаючи при цьому її семантичну суть. Вони виконують роль "перекладачів" між різними рівнями абстракції або форматами даних. Приклад: конструктор, що приймає на вхід мультисимвольний рядок і трансформує його в набір 2D-координат для візуалізації за допомогою "черепашачої" графіки [123], конструктор трансформації схеми структури даних у вихідний код на мові C#, чи чи конструктор що трансформує набір тексту у текст з тегами [131];

- **аналізуючі**: Використовуються для обчислення певних властивостей або метрик існуючої конструкції. Вони не змінюють саму конструкцію, а лише видобувають з неї інформацію. Приклад: конструктор, що аналізує згенерований набір точок фракталу і обчислює його фрактальну розмірність за методом підрахунку квадратів, транслятори, які аналізують вихідний код програми на наявність синтаксичних і семантичних помилок, чи конструктор, що аналізує подібність текстів [131];

- **оптимізуючі/адаптуючі**: цей тип конструкторів працює на мета-рівні, змінюючи параметри або структуру інших конструкцій та/або конструктивних процесів з метою досягнення певної цілі. Часто для цього використовуються евристичні алгоритми. Приклад: конструктор на основі генетичного алгоритму, який підбирає параметри (кут, правила) породжуючого конструктора для того, щоб згенерований фрактал максимально відповідав заданому природному зразку, наприклад, формі блискавки [10, 127, 132];

- **алгоритмічні**: слугують як бібліотеки виконуваних процедур. Вони не створюють конструкції самостійно, а надають реалізації для операцій, визначених у сигнатурах інших конструкторів під час етапу інтерпретації. Алгоритмічний конструктор зазвичай використовується для інтерпретації операцій інших конструкторів, але коли його алгоритми конструюються – то він є породжуючим. Прикладом є конструктор що реалізує операції генетичного алгоритму чи конс-

труктор, що містить операції формування зображення на основі «черепашок графіки» [123].

Використовуючи різні види конструкторів можливо отримувати групи споріднених конструкцій. Можна зафіксувати один трансформуючий конструктор (наприклад, стандартний інтерпретатор 2D-графіки) і варіювати параметри породжуючого конструктора. Змінюючи такі параметри, як кути повороту, коефіцієнти масштабування, кількість ітерацій або ймовірності застосування правил у стохастичних L-системах, можна легко та швидко генерувати цілі родини споріднених фракталів. Це дозволяє, наприклад, плавно переходити від детермінованої сніжинки Коха до її стохастичних варіацій, не змінюючи при цьому логіку візуалізації.

Рисунок 2.1 – Типи конструкторів за функціональністю

Вимір «Як?» є нововведенням запропонованої таксономії та базується на аналізі чотириетапного циклу уточнення. Він відповідає на питання: «Як саме абстрактний потенціал конструктора перетворюється на конкретну, виконувану мо-

дель, і на якому етапі цього процесу зосереджена основна складність або визначальні характеристики моделі?» (рис. 2.2):

- **керований спеціалізацією**: у таких конструкторах основна інтелектуальна складність та визначальні характеристики моделі зосереджені на етапі Спеціалізації. Головним завданням є формалізація предметної області: створення її онтології, визначення типів елементів, їхніх атрибутів та допустимих відношень. Подальші етапи (інтерпретація, конкретизація) можуть бути відносно тривіальними. Приклад: розробка конструктора для моделювання складної біологічної системи, наприклад, росту тканин. Основна складність полягатиме у визначенні типів клітин, їхніх станів, правил взаємодії та сигнальних шляхів. Самі алгоритми поділу чи переміщення можуть бути простими, але коректне визначення доменної моделі є критичним [32];
- **керований інтерпретацією**: для цього типу конструкторів ядро складності лежить на етапі Інтерпретації. Предметна область може бути простою, але її реалізація вимагає складних, унікальних або високопродуктивних алгоритмів. Приклад: трансформуючий конструктор для візуалізації тривимірної фрактальної пагоди [126]. Символьна L-система, що описує структуру, може бути простою (a→ab, b→c, c→a), але основна складність полягає в алгоритмах інтерпретації цих символів: розрахунок та рендеринг складних поверхонь Безьє, керування камерою, освітленням та деталізацією;
- **керований конкретизацією**: такі конструктори розробляються як гнучкі, універсальні шаблони, поведінка яких майже повністю визначається на етапі Конкретизації через надання початкових даних та параметрів. Вони є потужним інструментом для генерації цілих родин споріднених конструкцій. Приклад: параметричний конструктор для генерації класичних L-системних фракталів. Сам конструктор є універсальним інтерпретатором L-систем. Проте, надавши йому на етапі конкретизації аксіому f та правило f→f-f++f-f, ми отримуємо криву Коха, а надавши аксіому xf та правила для x та y, ми отримуємо криву Госпера. Вся унікальність кінцевого продукту визначається параметрами конкретизації [117];

- **керований реалізацією**: у цьому випадку визначальні властивості конструкції виникають як емерджентний результат самого процесу Реалізації. Поведінку системи важко або неможливо повністю передбачити, аналізуючи лише початкові правила. Це характерно для моделей хаотичних, стохастичних та складних адаптивних систем. Приклад: модель фрактального часового ряду, що імітує грозову активність. Хоча правила L-системи та параметри стохастичності задані, кінцева форма часового ряду (його піки, спади, квазіперіодичність) є емерджентною властивістю, що виникає в процесі симуляції. Неможливо точно передбачити значення в точці t, не провівши повну реалізацію до цього моменту [127].

Рисунок 2.2 – Типи конструкторів за уточненням

Вимір «Чим?» класифікує конструктори за тим, як вони взаємодіють із зовнішнім світом (виконавцями, даними) та як вони поєднуються один з одним для створення складніших систем [125] (рис. 2.3):

- **автономні**: повністю самодостатні моделі, що містять всю необхідну інформацію для реалізації самостійно. Вони не потребують зовнішніх даних для запуску. Приклад: конструктор, що генерує трикутник Серпінського з фіксованою глибиною рекурсії та заздалегідь визначеними правилами [69], чи конструктор відображення розкладу [60];

- **параметричні**: найбільш поширений тип для забезпечення гнучкості та багаторазового використання. Такі конструктори приймають зовнішні дані у вигляді параметрів перед початком реалізації. Ці параметри задаються на етапі конкретизації і можуть визначати глибину ітерацій, кути, початкові умови тощо. Це основний механізм для передачі даних між конструкторами. Такими конструкторами є конструктор, що формує на основі набору правил підстановки, аксіоми, кількості ітерацій та іншого мультисимвольну стрічку [126], чи конструктор, що на основі вхідного файлу дисциплін, навантаження вчителів, аудиторій та обмежень формує розклад занять [60];
- **інтерактивні**: мають здатність взаємодіяти із зовнішнім виконавцем (наприклад, користувачем) під час процесу реалізації для отримання даних. Їх виконання може бути призупинене до отримання відповіді. Приклад: конструктор фрактального дерева, який на кожному кроці рекурсії запитує у користувача кут розгалуження, чи конструктор на основі методу аналізу ієрархій, який у процесі роботи запитує у експерта попарні порівняння критеріїв для визначення їхньої вагомості [108];
- **мультиконструктори**: це конструктори, що містять інші конструктори та здатні керувати й забезпечувати зв’язки по керуванню та по даним між конструкторами. Прикладом є конструктор складання розкладу, що складається з конструктору створення популяції, конструктору оцінки та конструктору оптимізації [60]. Чи мультиконструктор породження графічних фракталів [127].

Зв'язок між конструкторами у складі мультиконструктора може бути забезпечений трьома механізмами:

- передача даних через параметри від зовнішнього виконавця або засобами мультиконструктора;
- наявність в одному конструкторі серед початкових умов конструювання результату реалізації іншого конструктора;
- наявність у відповідному алгоритмічному конструкторі алгоритму, який виконує реалізацію іншого конструктору (не того з яким він поєднаний в конструктивну систему).

Рисунок 2.3 – Типи конструкторів за взаємодією

Вимір «Де відповідає на питання: «Де, в якому концептуальному просторі, оперує конструктор?». Класифікація за операційним доменом дозволяє зрозуміти природу артефактів, з якими працює конструктор (рис. 2.4).

- **символьний домен**: носій та операції конструктора базуються на формальних мовах, алфавітах, рядках та правилах граматик. Його основний продукт — це структуровані послідовності символів. Приклад: будь-який конструктор на основі L-систем, основною метою якого є генерація мультисимвольного рядка, наприклад, f-f++f-f. Він оперує виключно в символьному просторі, не маючи уявлення про геометрію чи числові дані [126];
- **геометричний домен**: конструктор оперує з геометричними примітивами (точками, лініями, поверхнями, об'ємами) у дво- або тривимірному координатному просторі. Приклад: трансформуючий конструктор, що візуалізує 2D-фрактал, оперує з координатами на площині. Конструктор, що генерує 3D-пагоду, працює з контрольними точками та полігонами поверхонь Безьє у тривимірному просторі [126];
- **дата-центричний домен**: Основна мета конструктора — генерація, перетворення або аналіз числових наборів даних, таких як часові ряди, матриці або статистичні розподіли. Приклад: конструктор для моделювання фрактальних ча-

сових рядів. Його кінцевим продуктом є не символьний рядок і не геометрична фігура, а послідовність числових значень, що представляють, наприклад, інтенсивність блискавок у часі. Чи конструктор що аналізує популяцію на основі фітнес-функції [127];

- **мета-рівневий домен**: конструктор оперує не з даними, символами чи геометрією, а з іншими конструкторами. Його носій складається з конструкторів або їхніх параметрів, а операції — це дії з їх модифікації чи налаштування. Приклад: оптимізуючий конструктор «Генетичний пошук». Його «популяція» складається з хромосом, кожна з яких є набором параметрів для іншого, породжуючого конструктора. Операції «схрещування» та «мутації» відбуваються над цими наборами параметрів. Він працює на один рівень абстракції вище, ніж конструктор, який він оптимізує [127]. Чи конструктор формування коду, який керує його формування на основі інших [109].

Рисунок 2.4 – Типи конструкторів за природою домену

Вимір «Хто?» фокусується на природі виконавця, що керує процесом конструювання, відповідаючи на питання: «Хто або що приймає рішення та керує ви-

конанням конструктивного процесу?». Категорії виводяться з розмежування між «зовнішнім» та «внутрішнім» виконавцями, згаданого в дисертації (рис. 2.5):

- **зовнішній виконавець:** забезпечує уточнення (спеціалізація, інтерпретація, конкретизація) повністю керується людиною-проєктувальником. Людина вручну визначає правила, цілі, мету, онтологію, правила підстановки, обмеження, початковий стан та умови завершення, обирає алгоритми та задає параметри через інтерфейс програмного середовища;
- **внутрішній виконавець:** забезпечує процес виконання є повністю автоматизованим і детермінованим на основі заздалегідь наданої конфігурації. Це стосується етапу Реалізації, коли програмне середовище бере на себе керування і виконує повністю конкретизований конструктор без втручання ззовні.

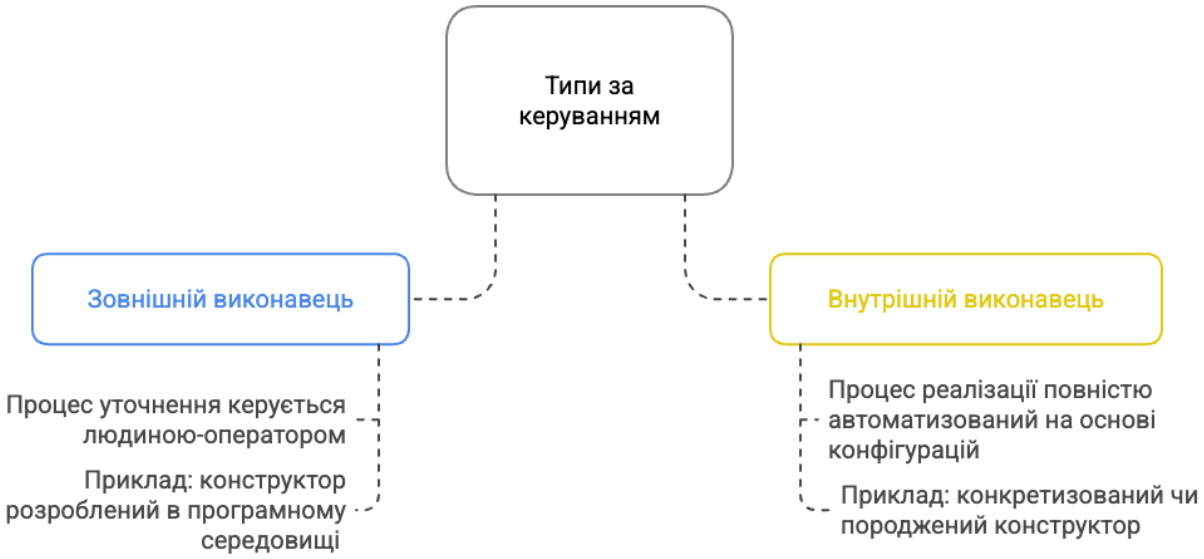


Рисунок 2.5 – Типи конструкторів за керуванням

Вимір «Коли?» надає часовий контекст для функції конструктора, відповідаючи на питання: «Коли в рамках загального моделюючого конвеєра застосовується цей конструктор?». Означений підходами до класифікації процесів за життєвим циклом, цей вимір дозволяє зрозуміти роль конструктора у схемі використання (рис. 2.6):

- **початкова генерація**: конструктор знаходиться на самому початку конвеєра. Його завдання — створити первинний артефакт з нуля (з аксіоми), який слу-

гуватиме вхідними даними для наступних етапів. Приклад: породжуючий конструктор, що створює мультисимвольний рядок для фракталу. Він є першим кроком у процесі візуалізації. Чи конструктор що створює початкову популяцію [127];

- **проміжна трансформація**: конструктор знаходиться в середині конвеєра. Він отримує на вхід артефакт, створений на попередньому етапі, перетворює його в нове представлення і передає далі. Приклад: трансформуючий конструктор, що перетворює мультисимвольний рядок на набір координат. Він не є ані початком, ані кінцем процесу, а слугує проміжною ланкою [127]. Чи конструктор що перетворює схему структури даних на програмний код. Він отримує схему та повертає представлення у вигляді коду [109];
- **фінальний аналіз/візуалізація**: конструктор є термінальним етапом конвеєра. Він створює кінцевий продукт, призначений для споживання зовнішнім агентом (наприклад, візуальне зображення для людини або числовий результат для звіту). Приклад: конструктор «Графічне відображення» в моделі грозового фронту. Він отримує готовий часовий ряд і створює фінальну візуалізацію, яка є кінцевим результатом усього моделювання [127]. Чи конструктор, що візуалізує оптимальний розклад у вигляді документу PDF чи HTML [60];
- **безперервна адаптація**: конструктор працює не в лінійному конвеєрі, а в ітеративному циклі. Він багаторазово виконується для поступового уточнення моделі, пошуку рішення або симуляції процесу в часі. Приклад: оптимізуючий конструктор «Генетичний пошук». Він не виконується один раз, а працює в циклі протягом багатьох поколінь, на кожній ітерації наближаючись до оптимального рішення. Його життєвий цикл є циклічним, а не лінійним [127]. Чи інтерактивний конструктор на основі методу аналізу ієрархій, який циклічно взаємодіє з експертом до досягнення узгодження оцінок [108].

Представлені шість вимірів утворюють комплексну фасетну таксономію, яка дозволяє всебічно описати будь-який конструктор у рамках конструктивно-продукційної парадигми. Сгрупована таксономія у табличному виді представлено на табл. 2.1.

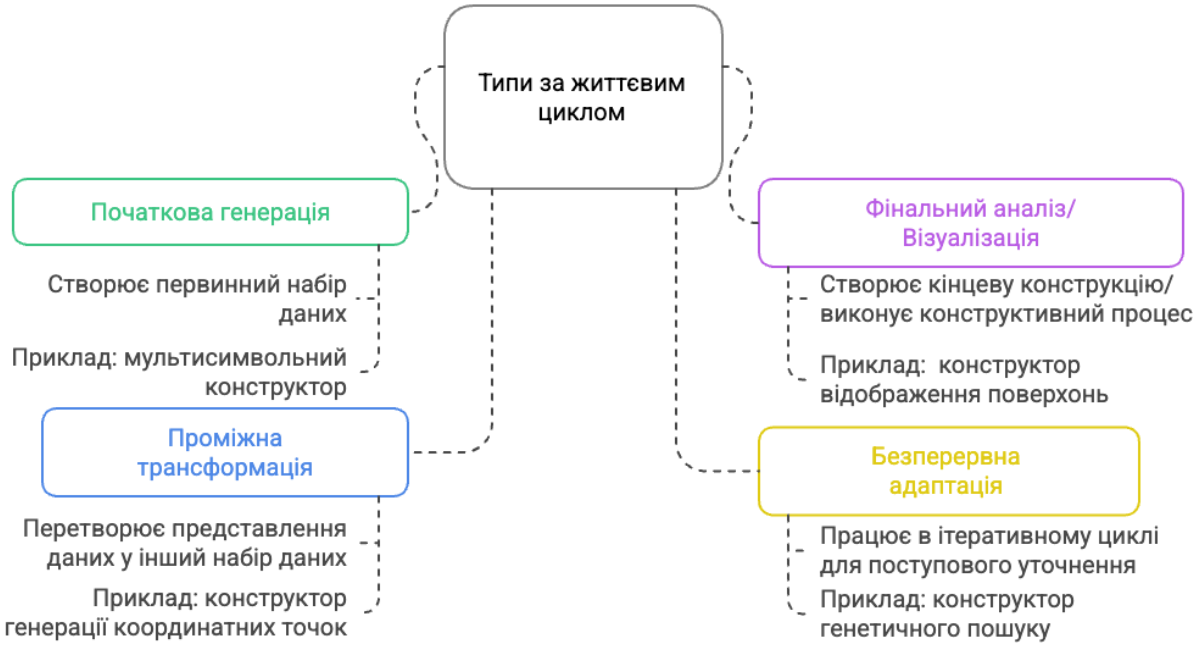


Рисунок 2.6 – Типи конструкторів за життєвим циклом

Таблиця 2.1 Таксономія конструкторів

Вимір	Категорія	Визначення	Приклад конструктору
Що? (Функціональне призначення)	Породжуючий	Створює нові конструкції на основі аксіоми та продукційних правил.	Конструктор, що генерує мультисимвольний рядок.
	Трансформуючий	Перетворює конструкцію з одного представлення в інше.	Конструктор, що перетворює мультисимвольний рядок на 2D-координати для візуалізації.
	Аналізуючий	Обчислює властивість або метрику заданої конструкції.	Конструктор, що обчислює фрактальну розмірність згенерованого набору точок.
	Оптимізуючий/Адаптуючий	Модифікує параметри іншого конструктора для досягнення цільової функції.	Конструктор «Генетичний пошук», що підбирає параметри для моделі грозового фронту.
	Алгоритмічний	Зберігає процедури, що зв'язуються з операціями під час інтерпретації.	Конструктор «Генетичні алгоритми», що містить реалізації мутації та схрещування.

Продовження табл.2.1.

Вимір	Категорія	Визначення	Приклад конструктору
Як? (Метод уточнення)	Керований спеціалізацією	Складність зосереджена у визначенні предметної області та її онтології.	Модель складної біологічної системи, де головне — правильно описати типи клітин.
	Керований інтерпретацією	Складність полягає в реалізації складних алгоритмів для операцій.	Конструктор для візуалізації 3D-пагоди з використанням поверхонь Безьє.
	Керований конкретизацією	Поведінка визначається переважно параметрами, наданими на останньому етапі.	Універсальний конструктор L-систем, що генерує різні фрактали залежно від аксіоми/правил.
	Керований реалізацією	Визначальні властивості є емерджентним результатом процесу виконання.	Модель стохастичного фрактального часового ряду.
Чим? (Спосіб взаємодії)	Автономний	Самодостатній, вся інформація для виконання є внутрішньою.	Конструктор, що генерує базову сітку трикутника Серпінського.
	Параметричний	Приймає зовнішні дані у вигляді параметрів перед виконанням.	Конструктор, що вбудовує фрактал у трикутник, отримуючи його координати як параметр.
	Інтерактивний	Може запитувати дані у зовнішнього виконавця під час виконання.	Конструктор, що дозволяє користувачу інтерактивно керувати розгалуженням дерева.
	Мультиконструктор	Містить та оркеструє виконання інших конструкторів.	Мультиконструктор «Моделювання грозового фронту», що поєднує три інші конструктори.
Де? (Операційний домен)	Символьний	Оперує з рядками, алфавітами та правилами формальних граматик.	Будь-який конструктор, що генерує мультисимвольний рядок L-системи.

Продовження табл.2.1.

Вимір	Категорія	Визначення	Приклад конструктору
	Геометричний	Оперує з геометричними примітивами у координатному просторі.	Конструктор, що візуалізує фрактал на площині або в 3D.
	Дата-центричний	Оперує з числовими наборами даних (напр., часовими рядами).	Конструктор, що генерує модельний часовий ряд грозової активності.
	Мета-рівневий	Оперує з іншими конструкторами або їхніми параметрами.	Конструктор «Генетичний пошук», що оптимізує параметри іншого конструктора.
Хто? (Керуючий агент)	Зовнішній виконавець	Процес уточнення керується людиною-проєктувальником.	Стандартний режим роботи в середовищі «Конструктор», де зовнішній виконавець задає всі параметри.
	Внутрішній виконавець	Процес реалізації повністю автоматизований на основі конфігурації.	Запуск будь-якого повністю конкретизованого конструктора.
Коли? (Фаза життєвого циклу)	Початкова генерація	Є стартовим етапом конвеєра, створює первинний артефакт.	Конструктор, що генерує початковий мультисимвольний рядок.
	Проміжна трансформація	Є середньою ланкою конвеєра, перетворює представлення.	Конструктор, що перетворює рядок у набір координат.
	Фінальний аналіз/Візуалізація	Є кінцевим етапом конвеєра, створює фінальний продукт.	Конструктор, що рендерить фінальне зображення фракталу.
	Безперервна адаптація	Працює в ітеративному циклі для поступового уточнення або пошуку.	Конструктор «Генетичний пошук», що працює протягом багатьох поколінь.

Висновки до другого розділу

У даному розділі представлено узагальнений конструктор як базову формальну схему моделювання фракталів. Показано, що реальні конструктори утворю-

ються крізь послідовність уточнюючих перетворень – спеціалізації, інтерпретації, конкретизації та реалізації.

Така чотирикрокова процедура забезпечує перехід від абстрактної граматичної форми до прикладного інструмента, здатного породжувати або трансформувати цільові конструкції.

Представлено нову фасетну таксономію для систематизації конструкторів конструктивно-продукційного моделювання. Базуючись на фундаментальних концепціях, викладених у першоджерелах, було розроблено шестивимірну класифікаційну систему. Ця система аналізує конструктори з точки зору їхнього функціонального призначення («Що?»), методу уточнення («Як?»), способу взаємодії («Чим?»), операційного домену («Де?»), керуючого агента («Хто?») та фази життєвого циклу («Коли?»).

На основі узагальненого підходу сформовано класи функціональних ролей конструкторів: породжуючі, трансформуючі, аналізуючі, оптимізуючі / адаптуючі та алгоритмічні.

Завдяки цьому стало можливим чітко розмежувати логіку обробки даних на різних етапах формування фракталів.

Здійснено класифікацію конструкторів за характером взаємодії із середовищем: автономні (працюють у замкненому вигляді), параметричні (отримують стартові дані через параметри), інтерактивні (запитують інформацію під час виконання) та мультиконструктори (поєднують кілька конструкторів у єдину систему).

Такий поділ окреслює механізми обміну даними і визначає, яким чином конструктори інтегруються у більші програмні комплекси.

Окрему увагу приділено механізмам передавання інформації в мультиконструкторах. Залежно від конкретного сценарію це може бути: передавання даних через параметри, використання результату одного конструктора як початкової умови іншого, або спільна робота над єдиною конструкцією, що поступово змінюється кількома виконавцями.

Отже, розділ сформував теоретичний каркас, що:

- визначає формальні етапи перетворення узагальненого конструктора у прикладні моделі;
- описує типологію конструкторів;
- регламентує способи інтеграції конструкторів у складні мультиагреговані системи.

Сукупність цих результатів забезпечує методологічну основу для наступних досліджень, де розроблені класи конструкторів використовуються для генерації та аналізу плоских, просторових і стохастичних фрактальних структур.

Матеріали розділу опубліковані у роботах: [69, 125, 126, 127].

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ КОНСТРУКТОРІВ

3.1 Визначення задач програмного забезпечення

Для реалізації методів, описаних у другому розділі, були розроблені програмні середовища «Конструктор 2.0» та «Конструктор 1.1».

Практична реалізація та поширення методології неможливі без відповідних інструментальних засобів [57]. Центральним програмним продуктом, що реалізує принципи конструктивно-продукційного моделювання, є середовище «Конструктор» [125]. Його перша версія, що є повністю десктопним кросплатформеним застосунком, була розроблена для систематичного формування конструкторів на прикладі генерації геометричних фракталів [119], а удосконалена версія «Конструктор 2.0», що представляє браузерним мікросервісним програмним забезпеченням, використовується в останніх дослідженнях для моделювання грозового фронту [92]. Технологічний стек, на якому базуються ці інструменти, включає мову програмування Python з використанням технології Qt [119]. Використання поширених, сучасних технологій запобігає ізоляції методології та робить її доступною для широкого кола розробників, підтверджуючи, що конструктивно-продукційне моделювання є зрілою, практично реалізованою парадигмою [113].

В основі інструментарію – реалізація конструкторів та проходження їх повного життєвого циклу від створення до формування конструкцій після виконання уточнюючих перетворень над ними.

Важливою функцією є опрацювання алгоритмів, описаних зовнішнім виконавцем на основі мови програмування Python, що вводяться в форми «Конструкторів».

Архітектура програмного забезпечення базується на основі domain-drive design [14, 79, 80], що реалізують основні рівні розмежування коду на основі предметної області. Структура розмежовує зони відповідальності програмних об’єктів, й дозволяє організувати тестування частин коду незалежно одна від одної і проходити окремі модульні та інтеграційні тести.

3.2 Архітектурне багаторівневе представлення інструментарію «Конструктор 2.0»

Враховуючи функціональні вимоги, а також необхідність забезпечити розширюваність програмного забезпечення, архітектуру інструментарію було поділено на три логічні рівні.

Перший рівень – Data Access Layer – поєднує у собі Domain Layer, де описані інформаційні об’єкти предметної області, та Data Access Object Layer, де організовується доступ до джерел даних, таких як бази даних, й представлені операції на отримання даних з підтримкою транзакційності операцій й логування процесів обробки.

Другий рівень – Business Service Layer – сформований з сервісних класів, які організовують виконання бізнес-завдань з використанням DAO, виконанням валідації вхідних та вихідних даних.

Третій рівень – Presentation Layer – поєднує FastAPI-маршрутизатори та Pydantic-валидацію для побудови REST API. Використовується Jinja2-шаблони для серверного рендерингу інтерфейсу програмного забезпечення та інтегровано адміністративні панелі через starlette-admin і sqladmin.

Рівні оформлені відповідно до правил domain-drive design з реалізацією механізму впровадження залежностей Dependency Injection, що забезпечує інжекцію DAO і сервісів у контролери без порушення принципу єдиної відповідальності (рис. 3.1). Налаштування рівнів виконується у Application Layer, де конфігуруються підключення до бази даних, реєструються всі роутери та підключаються middleware.

Data Access Layer складається з двох взаємопов’язаних компонентів: Domain Layer, у якому декларативно описані всі моделі предметної області, та Data Access Object Layer, що надає об’єктні інтерфейси для взаємодії з джерелами інформації.

У Domain Layer кожна сутність предметної області реалізована як клас, з певним набором атрибутів, та програмно реалізований на основі базового класу

SQLModel, що спрощує інтеграцію з локальною базою даних. Виділяються основні моделі предметної області.

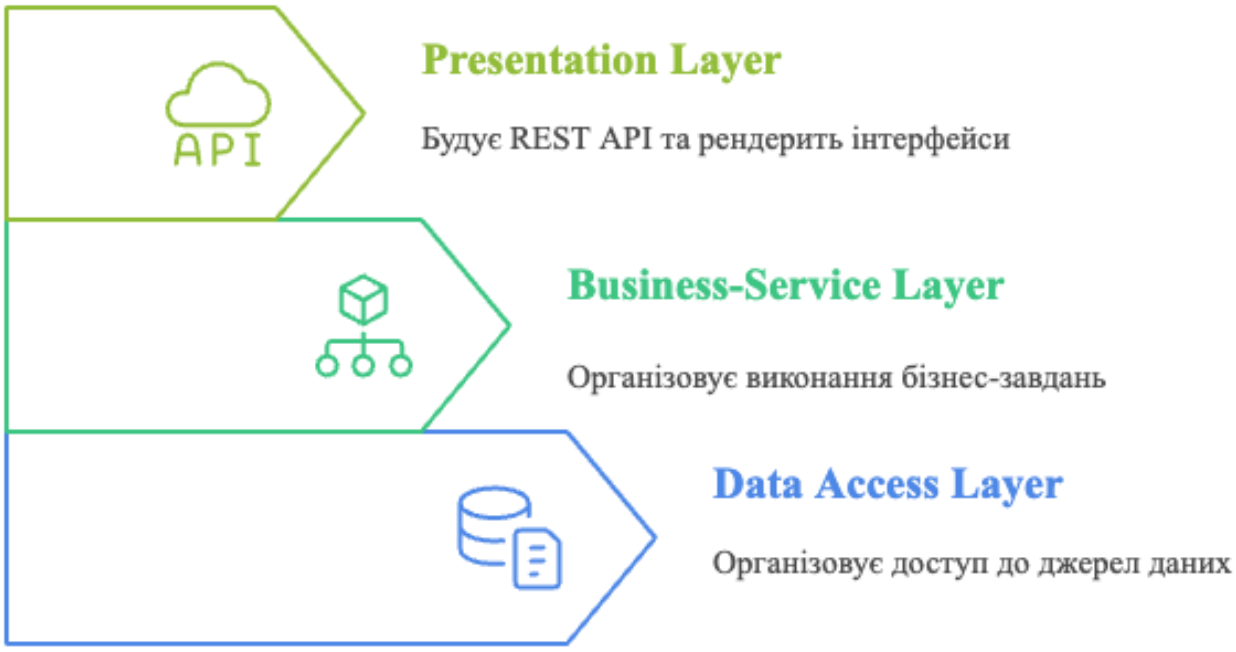


Рисунок 3.1 – Представлення архітектури верхнього рівня програмного забезпечення

Для кожної моделі вказані назва таблиці даних для збереження інформації, атрибути у вигляді полів з відповідними типами даних та обмеження у вигляді зовнішніх ключів, що налаштовані із каскадним видаленням. Зв’язки “один-до-багатьох” і “багато-до-багатьох” реалізовані через взаємозв’язки основних предметних моделей зі зв’язуючими, щоб забезпечити цілісність реляційної схеми без необхідності ручного написання SQL запитів. У моделях присутні службові поля унікальних ідентифікаторів на основі UUID та дати створення об’єктів та їх зміни з автоматичним заповненням.

Data Access Object Layer формалізує доступ до інформаційних джерел, оформлюючи дані відповідно до предметних моделей Domain Layer. DAO-класи, як ConstructorDAO, OperationDAO, організовано від єдиного базового інтерфейсу RelatedDAO, який реалізує CRUD-методи get_all(), get(id), create(obj), update(id, data), delete(id). У їхній реалізації для забезпечення транзакційності використовується організація сесій доступу на основі Session від SQLModel, що гарантує автоматичне збереження дій над даними та роботу з транзакціями в разі винятків.

Для обробки помилок передбачено перехоплення специфічних виключень із трансляцією програмних повідомлень в читабельні повідомлення єдиного виду логування.

Business Service Layer відповідає за реалізацію бізнес-логіки, що лежить над рівнем доступу до даних та безпосередньо готує результати для Presentation Layer. Сервісний рівень складається з класів, кожен з яких інжектує відповідний DAO.

Усі сервіси мають уніфікований інтерфейс помилок на основі HTTP-кодів із певним форматом повідомлень та логуються з рівнем ERROR.

Presentation Layer реалізує взаємодію зовнішнього виконавця з сервісом через три компоненти – API, UI та Admin Panel.

У API зосереджені маршрутизатори, які оголошують REST-ендпоінти для всіх доменних сутностей. Для кожного ресурсу існує окремий APIRouter. Вхідні та вихідні дані валідуються Pydantic-схемами, що гарантують перевірку наявності обов'язкових полів та правильність типів. Документація по користуванню та інформація по контрактах доступна у Swagger UI, що формується на основі описаних docsting у коді програмного забезпечення.

UI формується у вигляді серверного рендерінгу веб-сторінок. Використовується Jinja2Templates, що спрощує формалізацію інтерфейсу. Кожен контролер приймає HTTP-запити, звертається до відповідного сервісу, а потім повертає TemplateResponse із заповненим контекстом – списком записів, формою для введення даних або повідомленням про помилку.

Реалізацію адміністративної панелі забезпечує Admin Panel, що інтегрується у вигляді starlette-admin, так і sqladmin. Кожен доменний клас доповнюється адміністративною конфігурацією – стовпцями, фільтрами та груповими діями. Панель доступна у веб-інтерфейсі, і в ній реалізовано можливості пошуку, сортування, масових оновлень і перегляд журналу аудиту.

3.3 Використання програмного забезпечення «Конструктор 2.0» при розробці конструкторів

У межах побудови універсальної системи конструктивно-продукційного моделювання, виконується послідовність дій з формування та уточнення конструктору. При роботі з програмним забезпеченням, зовнішній виконавець проходить типову послідовність дій, починаючи від аутентифікації виконавця в застосунку до отримання готової конструкції, із акцентом на те, які компоненти системи та рівні даних задіюються на кожному етапі.

Оскільки програмне забезпечення передбачено використовувати на локальних технічних комп'ютерах користувачем, після запуску веб-сервера, передбачається безумовний доступ до інтерфейсу програмного забезпечення. Після підтвердження прав користувача відбувається ініціалізація головного інтерфейсу, і на екран виводиться перелік уже наявних конструкторів в головному меню (рис. 3.2). Отримання списку виконується через виклик API-методу «отримати всі конструктори», який звертається до рівню доступу до даних: репозиторій видає SQL-запит на читання таблиці «Constructor», повертає набір записів, що далі перетворюється в модель представлення (ViewModel), і відображається у вигляді табличного списку. У серверній частині ці дані потрапляють у сесію читання ORM, транслюються в об'єкти доменної моделі й передаються вище для форматування JSON-відповіді.

Надалі надається можливість вибору сценарія створення нового конструктора або модифікації існуючого. Для створення нового конструктору, дія на створення «Створити новий конструктор» представлена у піктографічному меню застосунку. Інтерфейс відкриває пусту форму введення базових параметрів: назва конструктора, опис, початкові атрибути (рис. 3.3). Після заповнення обов'язкових полів та вибору «Зберегти» відбувається POST-запит до API-методу «створити конструктор». Цей метод приймає дані форми у вигляді структурованого тіла запиту, викликає сервісний рівень, де створюється новий екземпляр об'єкта Constructor, ініціалізується його інформаційний модуль, присвоюється унікальний ідентифікатор та записується в базу даних.

Рисунок 3.2 – Список конструкторів

Після успішного завершення клієнт отримує оновлений список конструкторів із включенням щойно створеного порожнього конструктору, який можна далі редагувати.

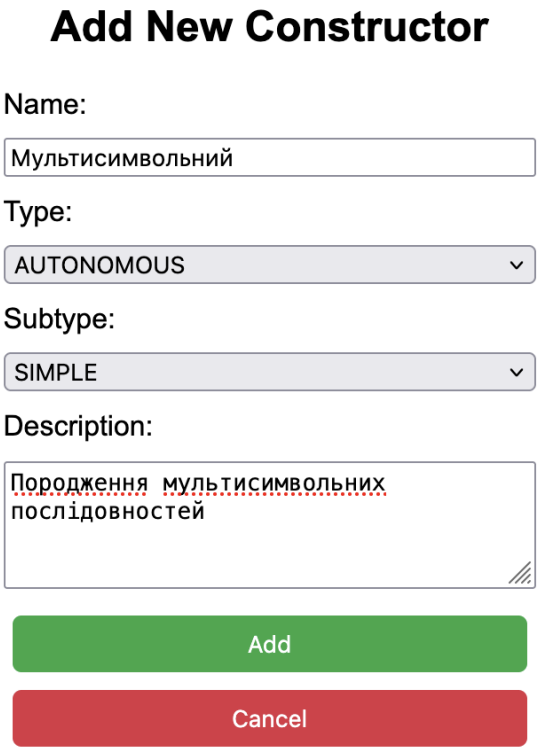


Рисунок 3.3 – Форма опису нового конструктора

Коли порожній конструктор створено, зовнішній виконавець має перейти до тонкого налаштування уточнюючих перетворень. З головного списку конструкторів обирається потрібний – і відразу відкривається панель редагування, поділена на вкладки: «Спеціалізація», «Інтерпретація» та «Конкретизація». Кожна вкладка вирішує окрему частину формалізації (рис. 3.4).

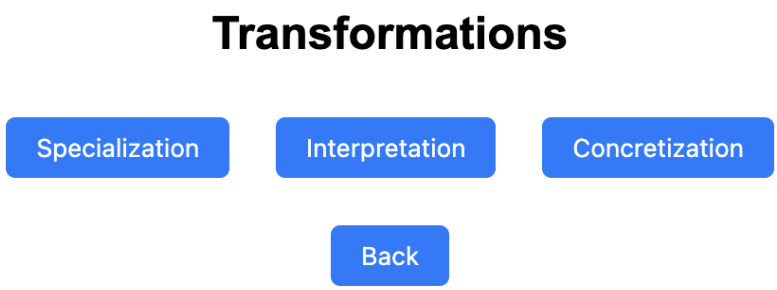


Рисунок 3.4 – Уточнюючі перетворення

У розділі «Спеціалізація» інтерфейс відображає множину доступних елементів конструктор у вигляді змінних та операцій, які були ініціалізовані за замовчуванням (рис. 3.5, 3.6).

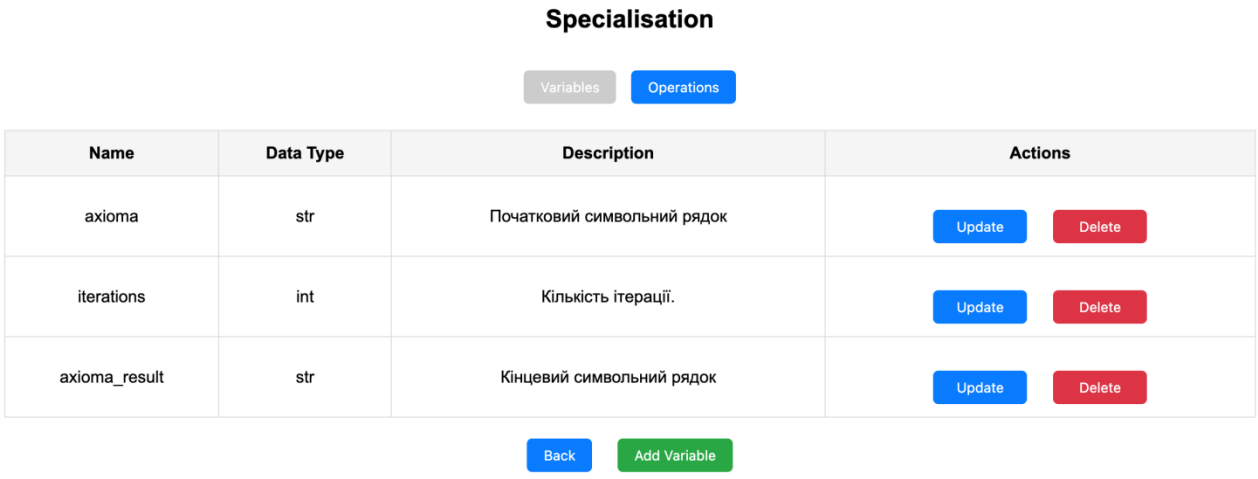


Рисунок 3.5 – Форма керування змінними конструктору при спеціалізації

Для кожноної змінної передбачено редагування та дії над нею. У змінні можливо додати лише ті термінали й нетермінали, що зовнішній виконавець вважає релевантними для конкретної предметної області. Після вибору й натискання «Застосувати» формується PUT-запит до API-методу «оновити спеціалізацію», у

якому передається масив відфільтрованих символів і відповідних правил. На сервері ці дані порівнюються з поточним станом у репозиторії, відбувається видалення зайвих зв'язків у таблицях «ConstructorVariable» чи «ConstructorOperation», а нові записи про спеціалізовані елементи заносяться в базу даних. Після цього рівень бізнес-логіки оновлює в пам'яті об'єкт конструктора та повертає результат користувацькому інтерфейсу для підтвердження успіху операції.

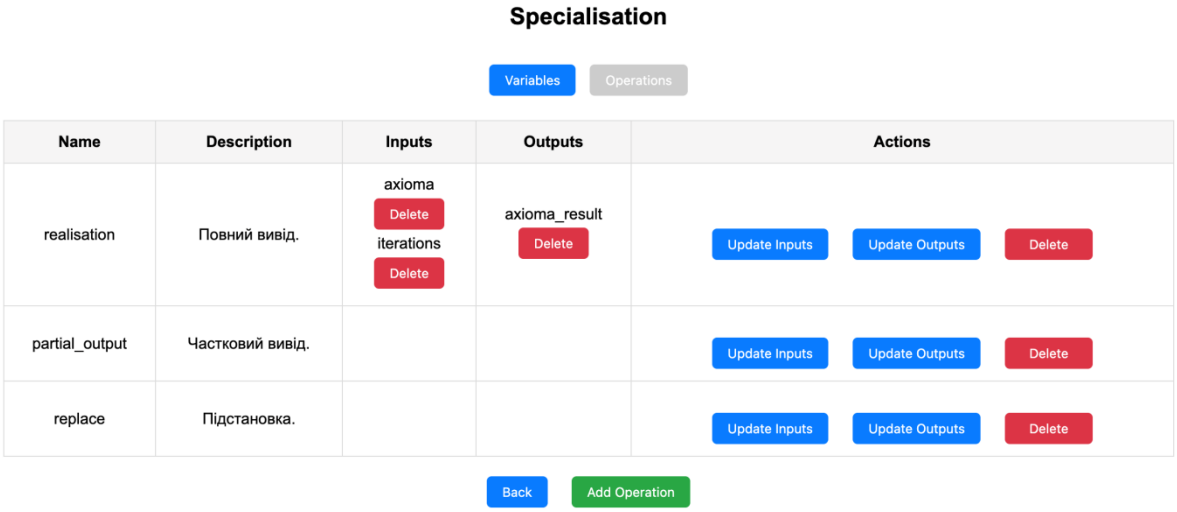


Рисунок 3.6 – Опис операцій при спеціалізації конструктору

Далі зовнішній виконавець переходить у вкладку «Інтерпретація» (рис. 3.7). Тут кожному операції описаній при «Спеціалізації» пропонується призначити конкретний алгоритм інтерпретування – опис роботи у вигляді текстового коду з вибору з переліку доступних функцій. Інтерфейс може надавати текстове поле, куди вставляються фрагменти формальних описів або ключові назви процедур. Після натискання «Зберегти інтерпретацію» кожне зіставлення записується у таблицю «OperationAlgorithmBind», яка зв'язує оператор і назву/ідентифікатор обчислювальної функції. Сервер повинен перевірити валідність цих призначень (наприклад, чи існує згаданий алгоритм у реєстрі), оновити зв'язки й повернути новий стан конструктора для відображення виконавцю.

У вкладці «Конкретизація» зовнішній виконавець формулює правила формування кінцевої конструкції із символічних представлень та встановлює початкові значення змінних, описаних при «Спеціалізації» (рис. 3.8, 3.9). Після вибору та налаштування параметрів і збереження виконується PUT-запит, що оновлює

інформаційний модуль у базі даних: у таблиці «Rule» зберігаються атрибутивні параметри конкретизації. Таким чином, для кожного нетермінала миттєво фіксується спосіб його перетворення у кінцевій об’єктній моделі.

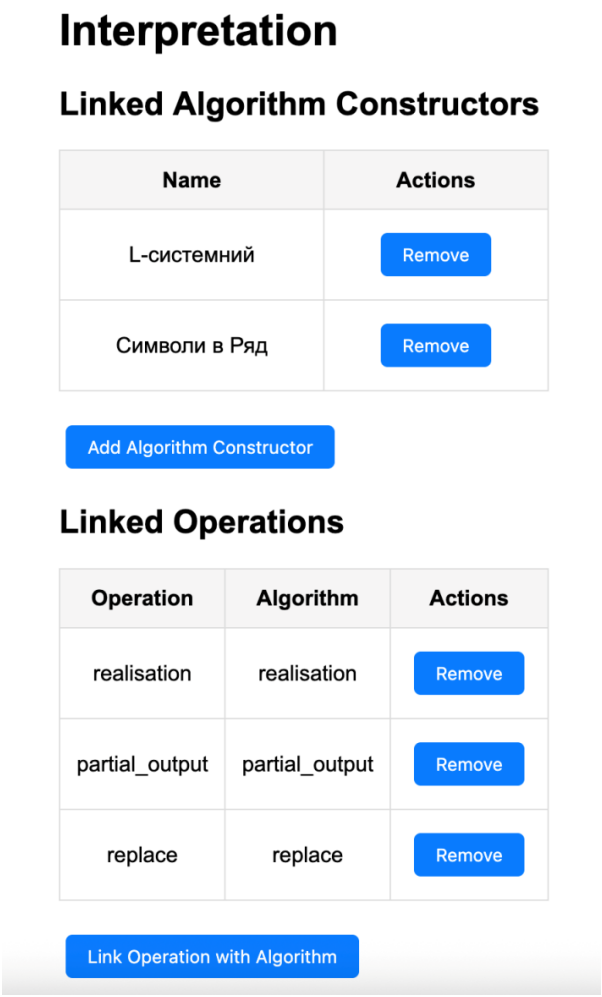


Рисунок 3.7 – Уточнення інтерпретації конструктору

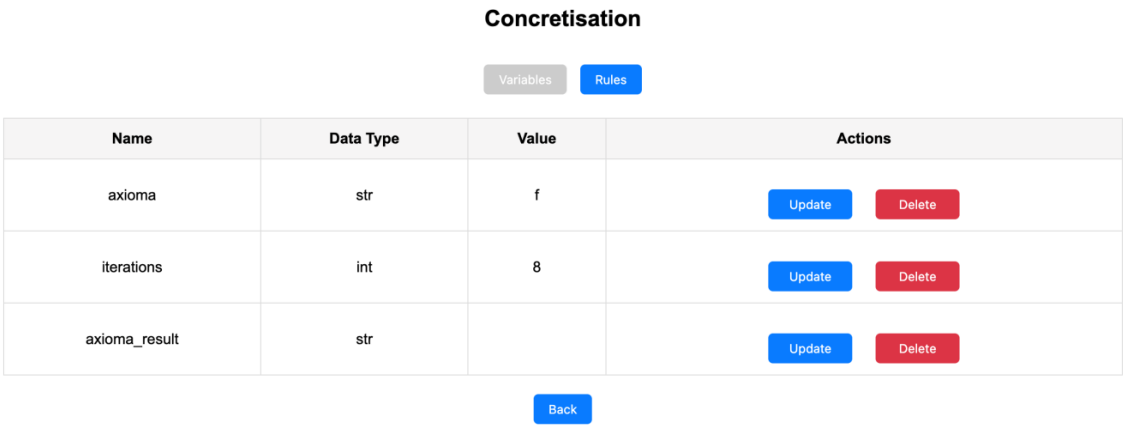


Рисунок 3.8 – Конкретизація змінних конструктора

Після проходження всіх трьох форм уточнення зовнішній виконавець готовий сформувати фактичну конструкцію чи конструктивний процес. Він звертається до функції «Реалізація» – обирає «Сформувати реалізацію» у піктографічному меню конструкторів.

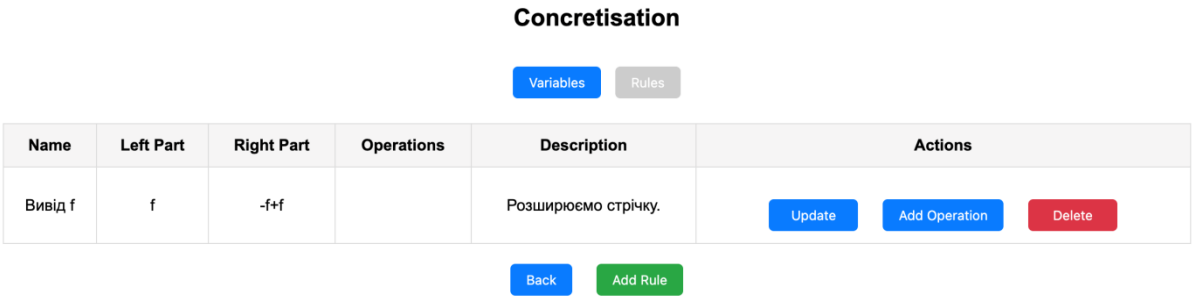


Рисунок 3.9 – Додавання правил підстановки при конкретизації конструктора

Це формує POST-запит до API-методу «запустити конструктор», у якому передаються ідентифікатор конструктора та актуальні параметри середовища. На сервері внутрішній виконавець оркеструє процес формування конструкції: модуль (у термінах доменного моделювання) послідовно завантажує всі спеціалізовані операції, виконує інтерпретовані функції над носіями та матеріалізує результат згідно з конкретизацією. При цьому кожен крок логічно фіксується в таблиці «Log», що дає змогу ретельно відстежити історію виконання. Після завершення оркестрації система повертає результат у вигляді структурованих даних (JSON-об’єкт із представленням виконання внутрішнім користувачем реалізації конструктору), які клієнтському інтерфейсу належить відобразити у відповідному візуальному компоненті.

Ключовим при такому підході є те, що кожна дія виконавця одразу ж транслювана в операцію над базою даних і доменною моделлю.

Ініціалізація форм інтерфейсу виконується GET-запитами до серверу, для отримання відображення, наповнення інтерфейсу з десеріалізованих JSON-відповідей, збереження даних введених через них – PUT/POST-запити, що викликають методи серверу, які в кінцевому підсумку трансформуються в SQL-запити INSERT/UPDATE/DELETE.

Таким чином, послідовність дій зовнішнього виконавця: від відкриття списку конструкторів до отримання готової конструкції, відображає багаторівневу взаємодію інтерфейсу, сервісного рівню, доменної логіки й рівня збереження даних (рис. 3.10).

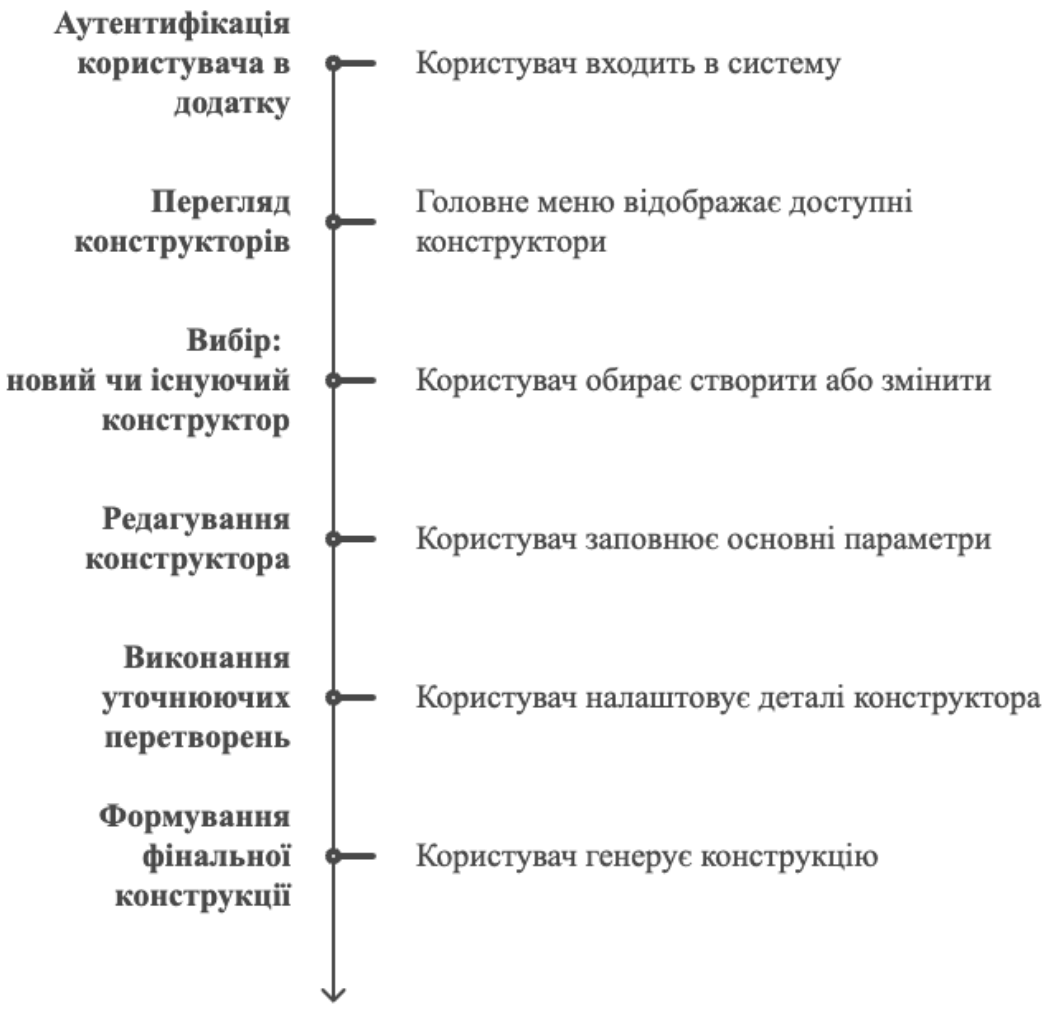


Рисунок 3.10 – Базовий процесс дій зовнішнього виконавця

3.4 Використання настільного застосунку «Конструктор 1.1»

Програмне середовище «Конструктор 1.1» розроблено засобами мови Python з використанням технології Qt для забезпечення кросплатформеності.

Продемонструємо його функціональність на прикладі моделювання геометричного фракталу «Сніжинка Коха».

На головному екрані додаємо до списку конструкторів почергово конструктори (рис. 3.11). Кожен з них наслідує всі властивості узагальненого конструктора.

Рисунок 3.11 – Перелік конструкторів

Спочатку додамо мультисимвольний автономний конструктор. Перейдемо його уточнення на наступному екрані (рис. 3.12).

Рисунок 3.12 – Уточнюючі перетворення

При спеціалізації задається предметна область, опис потрібних даних, елементів, відношень і операцій. У цьому конструкторі додаткових операцій не передбачено, дані задаються як на рис. 3.13. Без спеціалізації, інтерпретації та конкретизації виконання реалізація внутрішнім користувачем не доступна (не можлива).

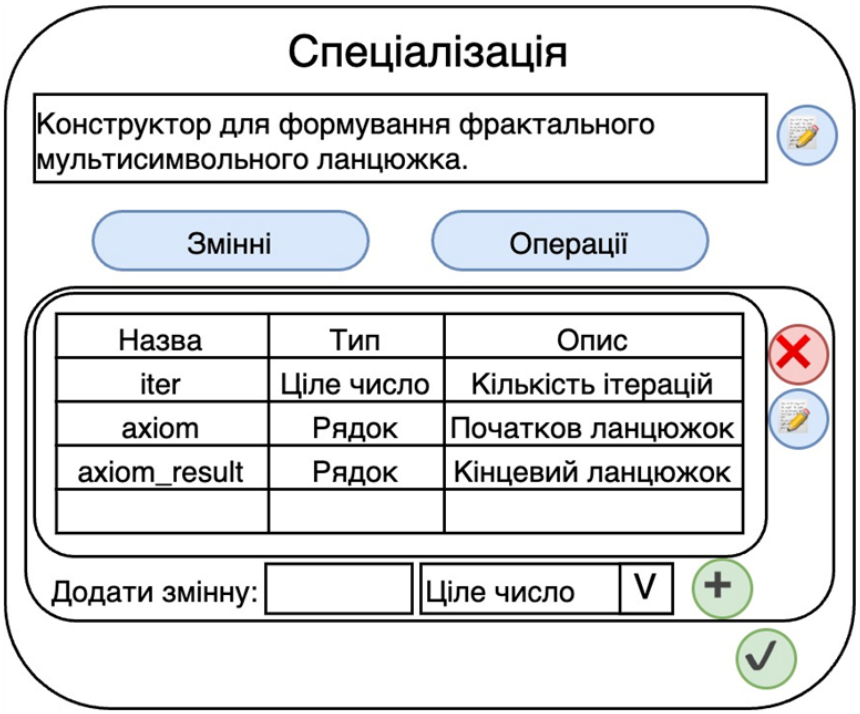


Рисунок 3.13 – Спеціалізація мультисимвольного конструктора

Для інтерпретації мультисимвольного конструктора потрібен відповідний алгоритмічний конструктор. Задаємо його у списку конструкторів. На рис. 3.11 він названий як L-системний. Його спеціалізація (рис. 3.14) полягає у наданні налагоджених процедур, які спроможні виконувати всі операції мультисимвольного конструювання.

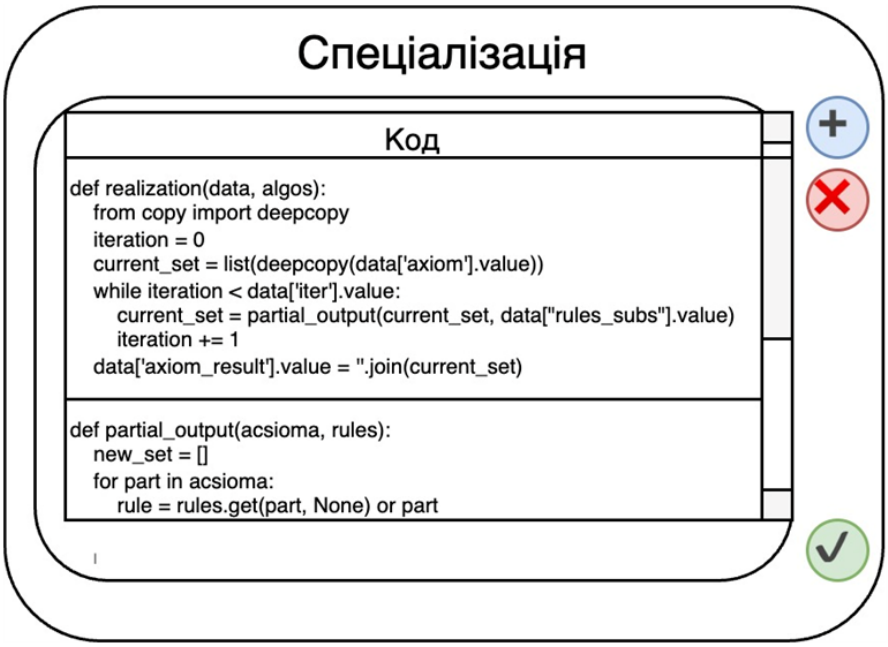


Рисунок 3.14 – Спеціалізація алгоритмічного конструктора

Тепер можна повернутись до інтерпретації мультисимвольного конструктора (рис. 3.15).

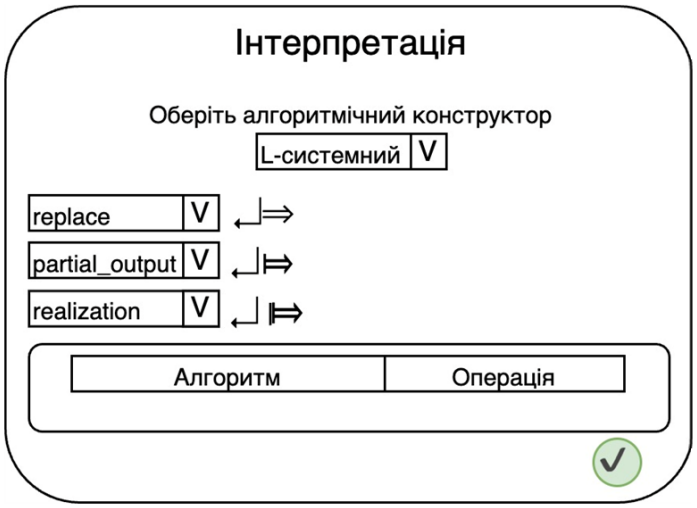


Рисунок 3.15 – Інтерпретація мультисимвольного конструктора

При інтерпретації кожній допустимій операції мультисимвольного конструктора (вказаній при його спеціалізації) ставиться у відповідність алгоритм алгоритмічного конструктора. Обов’язково повинні інтерпретуватись операції підстановки, часткового і повного виводу, які успадковуються від узагальненого конструктора. У результаті інтерпретації формується конструктивна система яка поєднує конструктор з елементами і можливими операціями і модель внутрішнього виконавця, здатного ці операції виконувати.

Конкретизація (рис. 3.16, 3.17) задає правила підстановки (конструювання), які складаються з відношень підстановки і операцій над атрибутами та початкові умови формування.

Так як мультисимвольний конструктор автономний то після всіх уточнюючих перетворень стає можливою його реалізація внутрішнім виконавцем, тобто формування фрактальної мультисимвольної послідовності за правилами формування сніжинки Коха. Мультисимвольна послідовність буде збережена у заданий файл.

Рисунок 3.16 – Встановлення значень змінних при конкретизації мультисимовольного конструктора

Рисунок 3.17 – Додавання правил підстановки при конкретизації мультисимвольного конструктору

Для відображення за цією послідовністю геометричного фракталу розробляється параметричний конструктор, параметром якого буде конструкція сформована мультисимвольним конструктором.

Як на рис. 3.3 заданий параметричний конструктор. Він спеціалізується як на рис. 3.18 і рис. 3.19. В цьому конструкторі вже задані деякі операції.

Аналогічно з мультисивольним конструктором для параметричного задається відповідний алгоритмічний з відображення фракталу і виконується інтепретація як на рис. 3.20 (на відміну від першого конструктора є додаткові інтерпретовані операції, пов’язані з відображенням ліній та зміною поточного кута нахилу).

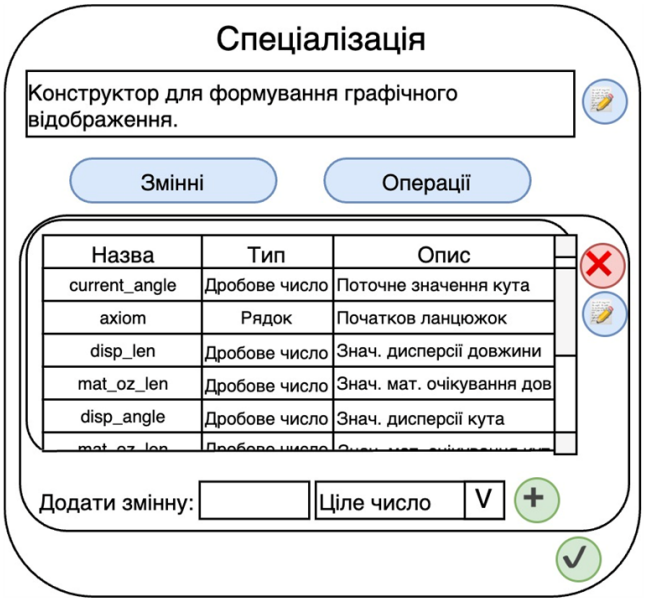


Рисунок 3.18 – Встановлення значень змінних при конкретизації графічного конструктора

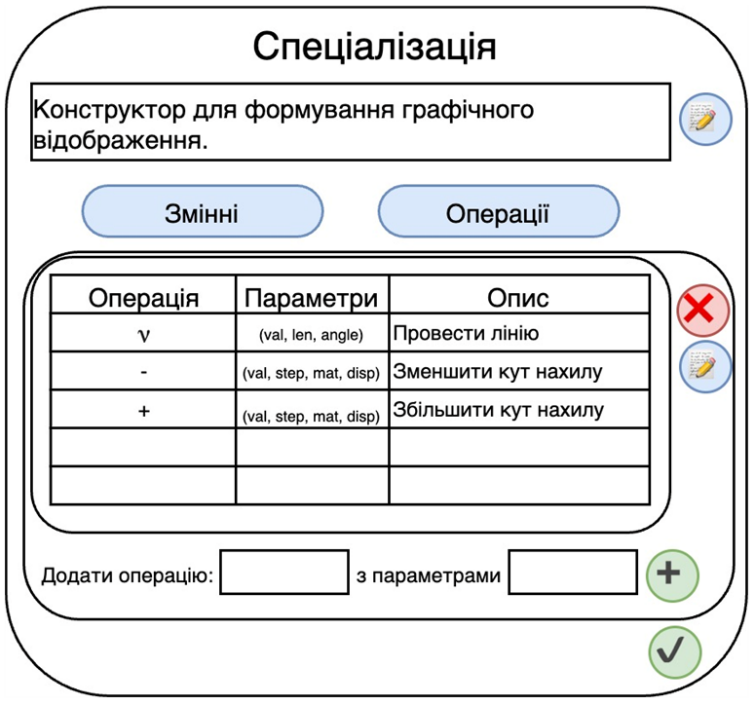


Рисунок 3.19 – Завдання операцій при спеціалізації графічного конструктору

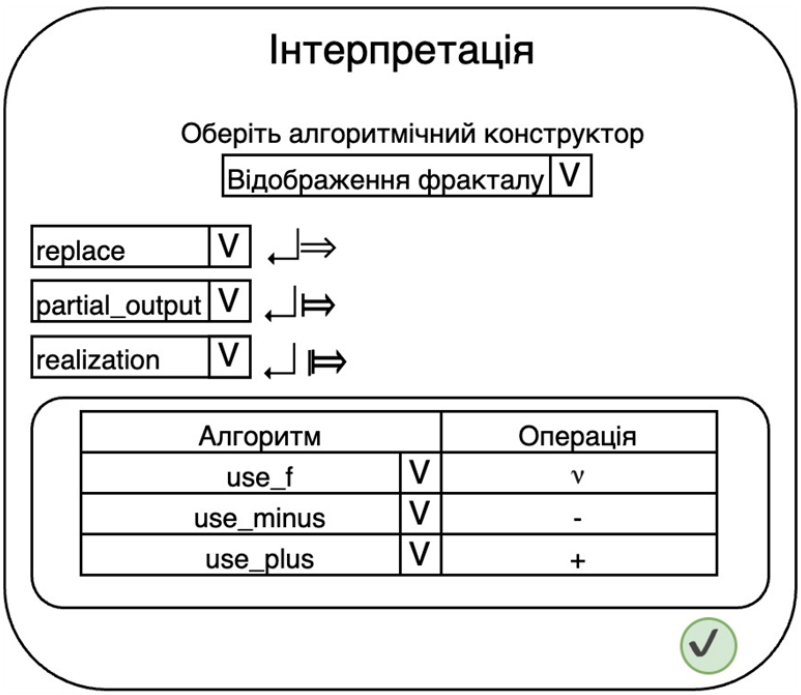


Рисунок 3.20 – Інтерпретація операцій графічного конструктора

Конкретизація графічного конструктора наведена на рис. 3.21 і рис. 3.22.

Рисунок 3.21 – Ініціалізація змінних графічного конструктора

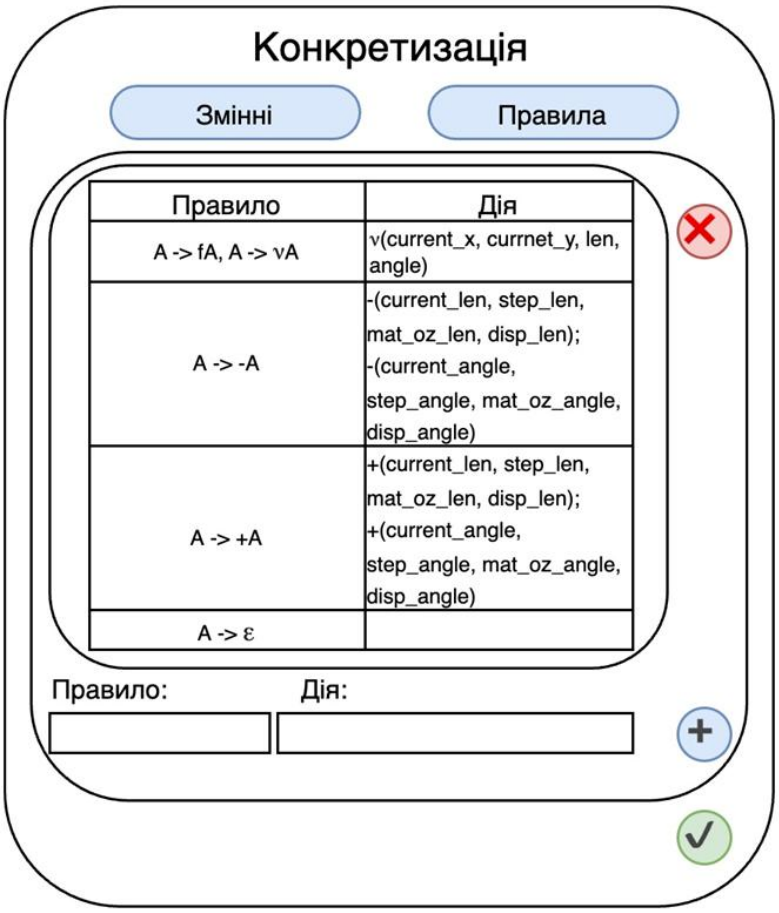


Рисунок 3.22 – Завдання правил підстановки графічного конструктора

Виконати графічний (етап реалізації) неможливо автономно. Він може працювати лише у зв’язці з мультисимвольним. Тому задається мультиконструктор «Сніжинка Коха». Він спеціалізується як на рис. 3.23 і рис. 3.24. Спочатку вказується як мультисимвольний і графічний конструктор поєднуються (рис. 3.23), а потім які дії слід виконати для повного формування зображення сніжинки Коха.

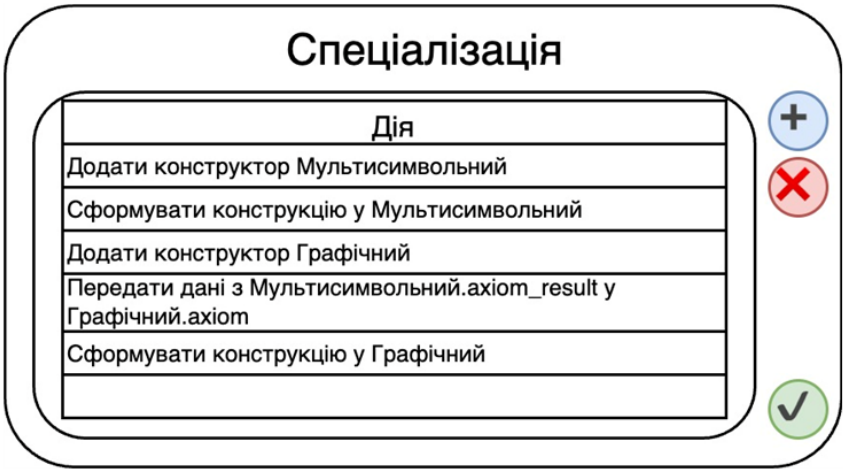


Рисунок 3.23 – Порядок виконання дій мультисимвольного конструктора

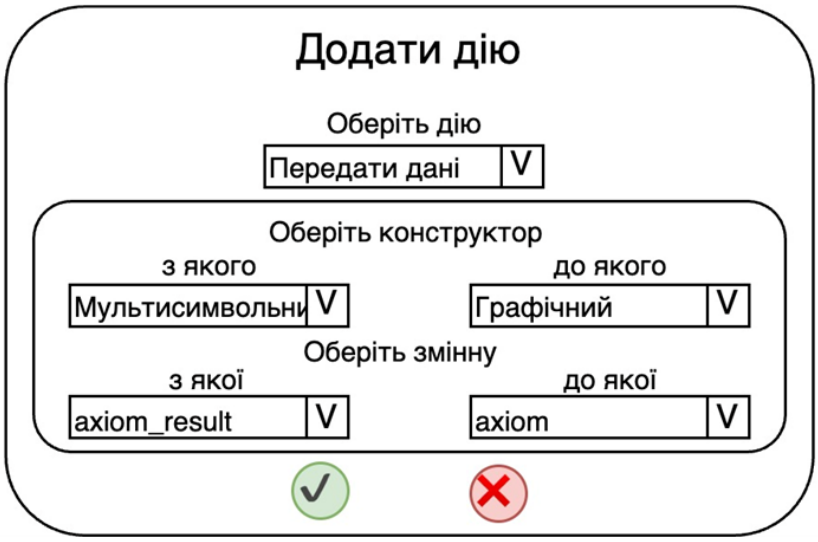


Рисунок 3.24 – Встановлення зв'язку між конструкторами

Так як графічний конструктором є трансформуючим, задається два відношення підстановки: перше використовується для розбору вхідної конструкції (мультисимвольного ланцюжка), друге – для формування вихідної конструкції (зображення сніжинки Коха).

Задані операції над атрибутами змінюють поточний кут нахилу лінії з можливими стохастичними збуреннями. При реалізації мультиконструктора отримаємо файл з зображенням сніжинки Коха (рис. 3.25).

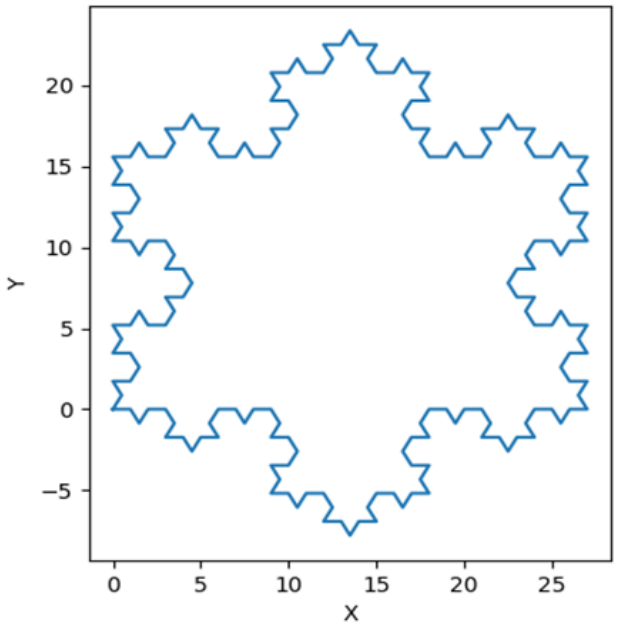


Рисунок 3.25 – "Сніжинка Коха" отримана при реалізації мультиконструктора

3.5 Організаційна структура коду

Програмне забезпечення реалізовано на базі архітектури, що поєднує принципи Domain-Driven Design (DDD) із підходом Porto.

Використання DDD (Domain-Driven Design) обумовлено тим, що система працює з чітко вираженою предметною областю, пов’язаною з описом, трансформацією та комбінацією конструкторів, моделей фракталів, правил продукцій і параметрів. DDD дозволяє побудувати програмне ядро навколо предметних об’єктів, що мають стабільну термінологію, логіку та поведінку:

- конструктори – як агрегати предметної області;
- правила – як поведінкові модулі;
- параметри – як об’єкти значень.

Таким чином, модель побудована не довкола баз даних чи UI, а навколо семантичних понять, що є ключовими у конструктивно-продукційному підході. Завдяки цьому ядро системи залишається незалежним від інтерфейсів і технічної реалізації, що забезпечує стабільність при зміні інфраструктурних компонентів

Архітектура Porto (Port + Container) використовується для модульної організації логіки за допомогою контейнерів, які відповідають окремим підсистемам, і портів, через які вони взаємодіють зі світом. Порти (інтерфейси до контейнерів) використовуються у вигляді сервісних шарів, що дозволяє зберігати слабке зчеплення між компонентами та легко масштабувати систему.

Поєднання DDD та Porto забезпечує:

- декларативність предметної логіки в ядрах контейнерів;
- ізоляцію змін у разі оновлення окремих модулів;
- підтримку інверсії залежностей;
- уніфіковану структуру репозиторіїв та менеджерів фабрик.

Завдяки цьому архітектурному рішенню програмна система є розширюваною, тестованою і стабільною при впровадженні нових видів конструкторів, правил чи графічних репрезентацій.

Відповідно до DDD кожний клас у системі належить до одного з трьох основних шарів:

- доменний шар (Domains) – містить сутності, які формально описують бізнес-об’єкти й їхні атрибути.
- шар доступу до даних (DAO/Repositories) – класи, що інкапсулюють взаємодію з джерелами даних.
- шар бізнес-логіки (Business-Model/Services) – класи, які реалізують бізнес-правила, координують роботу декількох репозиторіїв та формують відповіді на запити.

Нижче наведено групування основних сутностей відповідно до цих шарів і показано, як вони укладені в директорії згідно з принципами Porto.

Доменний шар (Domains) розміщений у папці domains знаходяться SQLModel-класи, що задають внутрішню структуру бізнес-об’єктів без жодної залежності від логіки збереження:

- Constructor – об’єкт конструктору, що поєднує в собі базову інформацію про нього;
- Operation – об’єкт операції.
- Variable – носій даних у вигляді змінних у конструктору.
- Algorithm – код алгоритму обробки інформації.
- Rule – умовне правило граматики обробки текстового рядку.
- Action – запис про побічні ефекти виконання (створення об’єктів, зміну змінних).
- Log – аудит-сутність для збереження інформації про послідовність кроків.
- Для рівня доступу до даних (DAO) у папці dao розміщені класи, що реалізують патерн Repository для кожної доменної сутності та працюють із сесіями SQLModel: ConstructorDAO, OperationDAO, VariableDAO, AlgorithmDAO, RuleDAO, ActionDAO, LogDAO – базових репозиторії з методами get, list, create, update, delete до відповідних доменних сутностей. Ці DAO-класи реалізують порт

для доступу до сховища даних, а будь-який інший шар взаємодіє з ними через чітко визначені інтерфейси.

Шар бізнес-логіки (Business-Model**)** розміщений у папці models представлений класами моделей, які інкапсулюють доменну логіку і координують роботу DAO: ConstructorModel, OperationModel, VariableModel, AlgorithmModel, RuleModel, ActionModel, LogModel, що відповідають логіці обробки над доменними об’єктами. Кожен з цих класів використовує один або кілька DAO для читання та запису даних, та описані функції обробки даних.

Папка protocols містить абстрактні базові класи (RelatedDAO, RelatedModel, ApiModel), що визначають контракт для DAO, моделей і API-рівня. Реалізації цих протоколів знаходяться в dao та models.

Такий поділ гарантує, що зміну сховища даних або способу надання API можна виконати, просто замінивши відповідні адаптери, не змінюючи протоколи.

Організація API , UI та Admin, відповідно до архітектурного дизайну Porto, оформлено у вигляді двох наборів адаптерів:

UI-адаптери, що розміщуються у ui, надає Jinja2-шаблонізований веб-інтерфейс для інтерактивного управління конструкторами. Формується уніфікований HTML-інтерфейс, що представляється зовнішньому виконавцю на кожному кроці роботи з конструкторами, як форми для заповнення та сторінки з відображенням існуючої інформації о конструкторі. Ці адаптери викликають відповідні методи API-рівню та отримують дані у форматі моделей.

Admin-адаптери у admin інтегрується з панелями управління SQLAdmin та Starlette-Admin для швидкого перегляду й редагування інформації у джерелах даних, як бази даних. У routers визначені маршрути FastAPI, які зв’язують HTTP-шляхи (/constructor, /operation, /variable тощо) із відповідними API-класами (адаптерами портів ApiModel). Кожний маршрут використовує методи бізнес-моделі для виконання операцій і повертає JSON-відповіді або редиректи на UI.

Таким чином, у відповідності з DDD сутності чітко розділені на доменні класи, репозиторії та сервіси, а архітектура Porto гарантує, що кожен із цих шарів взаємодіє через визначені порти – абстрактні інтерфейси в protocols/ – і може бути

змінений незалежно від інших, завдяки адаптерам у відповідних папках. API організовано у вигляді двох адаптерів – UI та Admin – що надають як веб-інтерфейс, так і стандартні CRUD-панелі, без порушення бізнес-логіки чи доменної моделі.

Висновки до третього розділу

Згідно теоретичних положень другого розділу, тут сформовано та реалізовано програмні середовища «Конструктор 1.1» та «Конструктор 2.0», які забезпечують повний життєвий цикл роботи з фрактальними конструкторами – від створення первинних правил до отримання готових конструкцій після уточнюючих перетворень.

Представлено розроблене програмне забезпечення для формування фракталів з різною елементною базою носія. Програмне забезпечення апробовано на задачах моделювання грозового фронту [10, 70, 72, 127], різних плоских графічних фракталів [68, 120, 121], та тривимірних фракталів [126].

Поставлено й вирішено такі прикладні завдання, як інтерактивне налаштування параметрів конструкторів через узагальнені перетворення, виконання користувацьких алгоритмів, описаних на мові програмування Python і збереження результатів моделювання у базі даних .

Архітектуру інструментарію поділено на три логічні рівні – Data Access Layer, Business Service Layer та Presentation Layer – що відповідають підходу Domain-Driven Design і гарантують ізоляцію доменних об’єктів від транспортних та інфраструктурних деталей.

Композиційний характер середовища посилено використанням архітектурного стилю Porto: зовнішня взаємодія забезпечується двома наборами адаптерів. UI-адаптери формують уніфікований HTML-інтерфейс для покрокового керування конструкторами.

Трирівнева архітектура представляє: по-перше, декларативну побудову базових DAO-класів; по-друге, консолідацію бізнес-правил у сервісах із чітко визначеними інтерфейсами; по-третє, наявність кількох веб-компонентів, що забезпечують однотипний доступ до внутрішніх процесів системи. Така структура сут-

тєво спрощує модульне й інтеграційне тестування та дає змогу змінювати будь-який шар незалежно від інших.

Матеріали розділу опубліковані у роботах: [10, 68, 70, 72, 120, 121, 126, 127].

РОЗДІЛ 4
КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ ФРАКТАЛІВ

4.1 Метод моделювання фракталів на основі мультисимвольних послідовностей

Спеціалізацію узагальненого конструктора для формування мультисимвольних фрактальних послідовностей можна розглядати як

$$C = \langle M, \Sigma, \Lambda \rangle \; {}_S\mapsto \; {}_S C = \langle M_L, \Sigma_L, \Lambda_L \rangle, \tag{4.1}$$

де M_L включає символьні термінали, а також проміжні форми та мультисимвольні конструкції, Σ_L складається з єдиної операції – конкатенації символів і символьних ланцюжків (знак операції між операндами зазвичай опускається); Λ_L – інформаційне забезпечення включає основи конструктивно-продукційного моделювання та особливості L-систем $\Lambda_L = \Lambda \bigcup \Lambda_1$.

Онтологія Λ_1 включає наведені вище позначення та їхню семантику, поняття «символ», «конкатенація», «ланцюжок символів» та інші відомі поняття мультисимвольного опрацювання, а також наведеними нижче положеннями.

Уточнюється операція часткового виведення $|\Rightarrow(\Psi, {}_{W_l} l)$: виконуються всі допустимі операції підстановки з Ψ, застосовні до терміналів із форми ${}_{w_l} l$, переглядаючи її зліва направо за винятком рекурсії.

Початкові умови задаються у вигляді ланцюжка символів (аксіоми).

Множина нетеміналів порожня.

Конструктивна система дає змогу конструювати деяку множину конструкцій (можливо, й одну) або виконувати перевірку належності заданої конструкції цій множині.

У низці випадків виникає необхідність схожим чином формувати дві або більше відмінних одна від одної множини конструкцій. Інакше кажучи, множини формованих конструкцій різні, а процеси їх формування мають незначну варіативність.

Аналогічно, виникає необхідність одну множину конструкцій відобразити різними варіантами, провівши ставлення одних чи інших атрибутів до певного параметру відображення. Тобто, виконати бієктивну трансформацію даних.

Бієкція – це відображення, яке ставить кожному елементу однієї множини рівно один елемент іншої множини.

Бієктивні відображення є прикладом повного перенесення даних та їх кількості однієї множини на іншу. Кінцева множина зберігає кількість атрибутів початкової множини. Тобто, кожному елементу деякої множини B ставиться лише один і тільки один елемент множини A.

Також, окрім бієкції, вирізняють відображення ін'єктивні та сюр'єктивні.

Ін'єктивні відображення характеризується тим, що кожному елементу деякої множини B ставиться не більше одного елементу множини A.

Сюр'єктивне відображення характеризується тим, що кожному елементу деякої множини A ставиться щонайменше один елемент множини A.

4.2 Конструктивно-продукційні моделі класичних геометричних фракталів

У таких випадках доцільно застосовувати параметричні конструктори. Назвемо сімейством конструкторів множину конструкторів, що відрізняються обмеженою кількістю положень інформаційного забезпечення. Під час визначення сімейства в круглих дужках задають параметри конструкторів (перелічують варіативні в межах сімейства елементи ІЗ). Визначимо сімейство параметричних конструкторів таким чином:

$$C(a_1, a_2 \ldots a_n) = \langle M, \Sigma, \Lambda \rangle \tag{4.2}$$

де $a_i \in \Lambda$ – ідентифікатори положення інформаційного забезпечення. Їх значення встановлюються зовнішнім виконавцем доповненням конкретизації.

Конкретизуємо C_L до рівня сімейства параметричних мультисимвольних конструкторів

$$C_L = \left\langle M_L, \Sigma_L, \Lambda_L \right\rangle_K \mapsto C_{MS}(B, P, n) = \left\langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \right\rangle \tag{4.3}$$

де B – початковий символьний рядок (аксіома), P – множина правил підстановки, n – кількість операцій часткового виводу, $M_{MS} = M_L \bigcup \{f, z, \text{y}, +, -\} \bigcup M_1$, M_1 включає ланцюжки з символів $\{f, z, \text{y}, +, -\}$, $\Sigma_{MS} = \Xi_{MS} = \{\circ\}$, $\circ$ – операція конкатенації, $\Lambda_{MS} = \Lambda_L \bigcup \Lambda_2$, Онтологічна складова Λ_2 включає зазначені вище позначення й їх семантику, а також наступні положення:

- ціль конструювання – формування мультисивольних ланцюжків фрактальної структури;
- правила підстановки задаються параметром P;
- обмеження – операції над атрибутами відсутні, правила підстановки містять єдине відношення підстановки;
- початкові умови – аксіома задається параметром B;
- умова завершення – виконання n операцій часткового виводу.

У результаті інтерпретації формуємо конструктивну систему як сукупність двох моделей: конструктора і внутрішнього виконавця (остання у вигляді конструктора алгоритмів, які здатний виконати виконавець)

$$\begin{aligned} &\left\langle C_{MS}(\text{B},\text{P},\text{n}) = \left\langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \right\rangle, C_A = \left\langle M_A, \Sigma_A, \Lambda_A \right\rangle \right\rangle_I \mapsto \\ &C_{A,MS}(\text{B},\text{P},\text{n}) = \left\langle M_{A,MS}, \Sigma_{A,MS}, \Lambda_{A,MS} \right\rangle, \end{aligned} \tag{4.4}$$

де C_A – модель виконавця у виглядів конструктору, що здатен виконувати базові та сконструйовані алгоритми; M_A – множина базових $\{A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j}, A_2^0 \mid_{Z_1, Z_2, A_i}^{:A_i}, A_3^0 \mid_{l_i, l_j}^{l_i \circ l_j}\} \subset \text{M}_A$ й сконструйованих $C = \langle M\{A_4 \mid_{l_h, l_q, l_i}^{l_j}$, $A_5 \mid_{l_i, \Psi}^{l_j}$, $A_6 \mid_{l_i, \Psi}^{l_j}\} \subset \Omega(C_A)$ алгоритмів; $\Sigma_A = \{\cdot, :\}$ включає сигнатуру операцій послідовного й умовного виконання алгоритмів; інформаційне забезпечення Λ_A, $M_{A,MS} = \left\langle M_{MS}, M_A \right\rangle$, $\Sigma_{A,MS} = \left\langle \Sigma_{MS}, \Sigma_A \right\rangle$, $\Lambda_{A,MS} = \Lambda_{MS} \bigcup \Lambda_A \bigcup \Lambda_3$.

Алгоритми M_A:

- виконання операції композиції алгоритмів $A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j}$ ($A \mid_X^Y$ – алгоритм над даними з вхідної множини X зі значеннями з множини Y, A^0 – алгоритм утворен-

ня), $A_i, A_j \in \Omega(C_{A,MS})$, $A_i \cdot A_j$ – послідовне виконання алгоритма A_j після алгоритма A_i;

- умовне виконання $A_2^0 \mid_{Z_1,Z_2,A_i}^{A_i}$, яке полягає у виконання алгоритма A_i при умові $Z_1 \supseteq Z_2$;

- конкатенація ланцюжків символів $A_3^0 \mid_{l_i,l_j}^{l_i \circ l_j}$, $l_i, l_j \in M_{MS}$;

- виконання операції підстановки $A_4 \mid_{l_h,l_q,l_i}^{l_j}$, , $l_i, l_j, l_h, l_q \in M_{MS}$, l_i, l_j – поточна форма, в якій виконується операція підстановки до і після її виконання, l_h, l_q – ланцюжки в лівій і правій частині відношення підстановки, згідно з яким виконується;

- виконання операції часткового та повного виводу $A_5 \mid_{l_i,\Psi}^{l_j}$, $A_6 \mid_{l_i,\Psi}^{l_j}$, $\Psi \subset \Lambda_{MS}$ – множина правил підстановки.

Інформаційне забезпечення конструктору Λ_A включає в собі приведені вище визначення, позначення та їхню семантику

$$\Lambda_3 = \{(A_1^0 \mid_{A_i,A_j}^{A_i \cdot A_j} \hookleftarrow \cdot), (A_2^0 \mid_{Z_1,Z_2,A_i}^{A_i} \hookleftarrow :), (A_3^0 \mid_{l_i,l_j}^{l_i \circ l_j} \hookleftarrow \circ); (A_4 \mid_{l_h,l_q,l_i}^{l_j} \hookleftarrow \Rightarrow);$$
$$(A_5 \mid_{l_i,\Psi}^{l_j} \hookleftarrow \mid\Rightarrow); (A_6 \mid_{l_i,\Psi}^{l_j} \hookleftarrow \mid\mid\Rightarrow)\}. \quad (4.5)$$

Виконаємо завершаючу конкретизацію та реалізацію двопороджуючих сімей параметричних конструкторів: мультисимвольного дракона Гартера–Гайвея та сніжинки Коха.

Реалізація мультисимвольного дракона Гартера–Гайвея

$$\left\langle C_{MS}(fz, \{y \to -fz - y;\ z \to z + y\ f +\}, 7), C_A \right\rangle_R \mapsto \Omega_1(C_{MS}) \quad (4.6)$$

Позначається в паралельному виконані підстановок:

Початковий стан $(n = 0)$ поточної форми

$$fz\ ;$$

у результаті першої операції часткового виводу $(n = 1)$ отримуємо

$$fz + yf +;$$

При $n = 2$ маємо

$$fz+yf++-fz-yf+;$$

при $n=3$ маємо

$$fz+yf++-fz-yf++-fz+yf+--fz-yf+$$

й так далі.

У результаті реалізації отримуємо мультисимвольну конструкцію $\Omega_1(C_{MS})$, яка має властивість самоподібності, що наочно представлено процедурою її формування.

Реалізацію мультисимвольної сніжинки Коха виконаємо в такий самий спосіб

$$\left\langle C_{MS}(f++f++f,\{f\to f-f++f-f\},4),C_A\right\rangle_R\mapsto\Omega_2(C_{MS}) \tag{4.7}$$

Сімейство параметричних конструкторів-перетворювачів із конструкції у вигляді ланцюжка символів у конструкцію у вигляді зображення геометричного фрактала

$$C_L=\left\langle M_L,\Sigma_L,\Lambda_L\right\rangle_K\mapsto C_{GF}(\Omega_i(\mathrm{C}_{MS}),M_x,\mathrm{d}M_x,\mathrm{D}_x,\alpha,\mathrm{m})=\left\langle M_{GF},\Sigma_{GF},\Lambda_{GF}\right\rangle \tag{4.8}$$

де $\Omega_i(\mathrm{C}_{MS})$ – ланцюжки символів, отримані у результаті реалізації конструктора C_{MS}; M_x – початкова довжина відрізку, $\mathrm{d}M_x$ – прирощування довжини відрізку (%), D_x – дисперсія довжини відрізку, α – кут, m – кількість формуємих геометричних фігур, M_{GF} – включає множину терміналів T (всіх можливих ломаних на площині й символів $\{f,z,\mathrm{y},+,-\}$), нетерминалов $N=\{A\}$, правила підстановки, $\Sigma_{GF}=\Xi_{GF}\bigcup\Phi_{GF}$, $\Xi_{GF}=\{\circ,f\}$, $\Phi_{GF}=\{*,\wedge,+,-,\times,:\}$, $\Lambda_{GF}=\Lambda_L\bigcup\Lambda_4$.

Позначемо відрізок ломаної ν з атрибутами $i\hookleftarrow\nu$ – порядковий номер при формувані ломаної, $X_i\hookleftarrow\nu, Y_i\hookleftarrow\nu$ – координати початку, $l\hookleftarrow\nu$ – довжина, $\beta\hookleftarrow\nu$ кут нахилу.

Введемо операції над атрибутами:

- додавання, віднімання, множення і ділення відповідно $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ й $:(c,a,b)$ з операндами a, b й результатом c;

- обчислення кінця поточного відрізку (й початку наступного) $*(\mathrm{t}\,M_x, \beta, X_i, Y_i, X_{i+1}, Y_{i+1})$ з початковими координатами X_i, Y_i, довжиною $\mathrm{t}\,M_x$ й кутом нахилу β;

- генерація випадкового, нормально розподіленого числа $\wedge(c,a,b)$ з математичним очікуванням a і дисперсією b.

Інформаційне забезпечення конструктора Λ_4 включає наведені вище визначення, позначення та їхню семантику, а також такі положення:

- онтологія доповнюється відомими поняттями «площина», «відрізок», «дійсне число», “координати”, «кут», тощо, що дають змогу оперувати як з лінійними геометричними фігурами, так і з дійсними числами;

- мета конструювання – формування лінійного геометричного фрактала на площині;

- правила підстановки:

$\{\langle\langle A \to fA,\ A \to \nu A\rangle, \langle *(\mathrm{t}\,M_x, \beta, X_i, Y_i, X_{i+1}, Y_{i+1}), = (i,i,1)\rangle\rangle,$
$\langle\langle A \to \mathrm{z}A\rangle, \langle\varepsilon\rangle\rangle,$
$\langle\langle A \to yA\rangle, \langle\varepsilon\rangle\rangle,$
$\langle\langle A \to +A\rangle, \langle \times(\mathrm{qM}, M_x, dM_x), : (\mathrm{qM}, \mathrm{qM}, 100), +(\mathrm{t}\,M_x, \mathrm{t}\,M_x, qM), \wedge(l, \mathrm{t}\,M_x, D_x), +(\beta, \beta, \alpha)\rangle\rangle,$
$\langle\langle A \to -A\rangle, \langle \times(\mathrm{qM}, M_x, dM_x), : (\mathrm{qM}, \mathrm{qM}, 100), -(\mathrm{t}\,M_x, \mathrm{t}\,M_x, qM), \wedge(l, \mathrm{t}\,M_x, D_x), -(\beta, \beta, \alpha)\rangle\rangle,$
$\langle\langle A \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\};$

- обмеження – правило $\langle A \to \varepsilon, \langle\varepsilon\rangle\rangle$ виконується, якщо не застосовуються всі інші;

- початкові умови – ланцюжок $\Omega_i(\mathrm{C}_{MS})$; початкова точка $X_0 = 0, Y_0 = 0$; початковий кут $\beta = 0$, початковий номер точки $i = 0$, початкова довжина відрізку $\mathrm{t}\,M_x = M_x$;

- умова завершення – виконання правила $\langle A \to \varepsilon, \langle\varepsilon\rangle\rangle$.

Визначимо конструктивну систему, інтерпретувавши C_{GF}:

$$\left\langle C_{GF}(\Omega_i(\mathrm{C}_{MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \alpha, \mathrm{m}) = \left\langle M_{GF}, \Sigma_{GF}, \Lambda_{GF} \right\rangle, C_B = \left\langle M_B, \Sigma_B, \Lambda_B \right\rangle \right\rangle_I \mapsto$$
$$C_{B,GF}(\Omega_i(\mathrm{C}_{MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \alpha, \mathrm{m}) = \left\langle M_{B,GF}, \Sigma_{B,GF}, \Lambda_{B,GF} \right\rangle, \quad (4.9)$$

де C_B – конструктор, розширюючий можливості C_A наявністю сформованих алгоритмів $\{A_7 \mid^{c}_{a,b}, A_8 \mid^{c}_{a,b}, A_9 \mid^{X_{i+1},Y_{i+1}}_{\mathrm{t}M_x,\beta,X_i,Y_i}, A_{10} \mid^{\mathrm{qM}}_{dM_x,D_x}\} \subset \Omega(C_B)$, $M_B \supset M_A$, $M_{B,GF} = \left\langle M_{GF}, M_B \right\rangle$, $\Sigma_B = \Sigma_A$, $\Sigma_{B,GF} = \left\langle \Sigma_{GF}, \Sigma_B \right\rangle$, $\Lambda_B = \Lambda_A$, $\Lambda_{B,GF} = \Lambda_{GF} \bigcup \Lambda_B \bigcup \Lambda_5$

Інформаційне забезпечення конструктора Λ_5 включає наведені вище позначення та їхню семантику, а також визначення атрибутики операцій (алгоритмів, що їх реалізують):

$$\Lambda_5 = \{(A_7 \mid^{c}_{a,b} \hookleftarrow +), (A_8 \mid^{c}_{a,b} \hookleftarrow -), (A_9 \mid^{c}_{a,b} \hookleftarrow \times), (A_{10} \mid^{c}_{a,b} \hookleftarrow :),$$
$$(A_{11} \mid^{X_{i+1},Y_{i+1}}_{\mathrm{l},\beta,X_i,Y_i} \hookleftarrow *), (A_{12} \mid^{\mathrm{qM}}_{dM_x,D_x} \hookleftarrow \wedge), (A_{13} \mid^{\nu}_{\mathrm{l},\beta,X_i,Y_i} \hookleftarrow f)\}. \quad (4.10)$$

Надалі виконується конкретизація шляхом завдання значень параметрів у конструктивній системі та відповідна реалізація.

Розглянемо їх на прикладах.

Детермінований фрактал «сніжинка Коха», представлений на рис. 4.1, є реалізацією одного х сімї параметричних конструктора з параметрами $M_x = 1,\ dM_x = 0,\ D_x = 0,\ \alpha = 60^o,\ m = 1\ \left\langle C_{GF}(\Omega_2, 1, 0, 0, 60, 1), C_B \right\rangle_R \mapsto \Omega_3$.

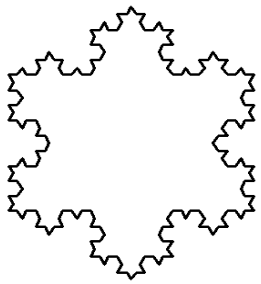


Рисунок 4.1 – Реализація детермінованого фрактала "Снежинка Коха"

Один з 20 стохастичних варіантів цього фракталу реалізованого конструктором $\left\langle C_{GF}(\Omega_2, 1, 50, 50, 60, 20), C_B \right\rangle_R \mapsto \Omega_4$, представлено на рис. 4.2.

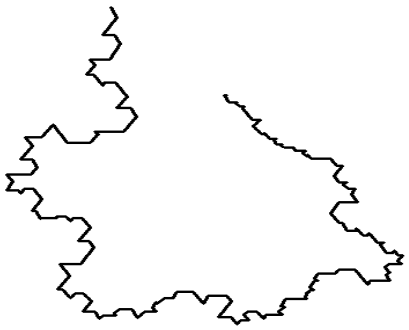


Рисунок 4.2 – Одна з реалізацій стохастичного фрактала "Снежинка Коха"

Детермінований фрактал «Дракон Гартера-Гайвея», представлений на рис. 4.3, є реалізацією конструктора $\left\langle C_{GF}(\Omega_1,1,0,0,90,1), C_B \right\rangle_R \mapsto \Omega_5$.

Один з 20 стохастичних варіантів цього фракталу реалізованого конструктором $\left\langle C_{GF}(\Omega_1,1,50,50,90,20), C_B \right\rangle_R \mapsto \Omega_6$, представлено на рис. 4.4.

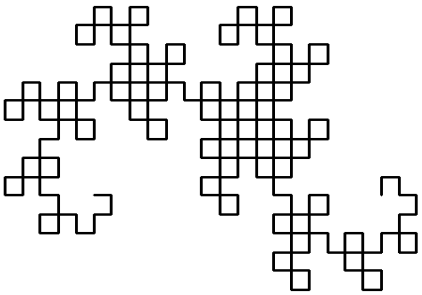


Рисунок 4.3 – Реалізація детермінованого
фрактала «Дракон Гартера – Гайвея»

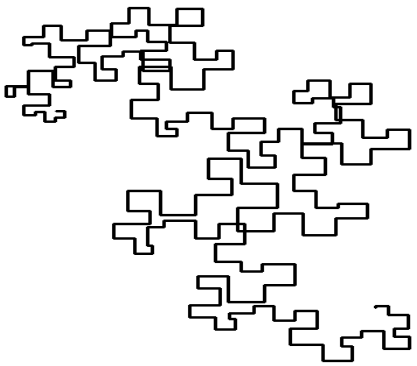


Рисунок 4.4 – Одна з реалізацій стохастичного
фрактала "Дракон Гартера – Гайвея"

Таким чином, за конструктивно-продукційним підходом виконання моделювання лінійних геометричних фракталів виконується шляхом перетворень двух параметричних сімей конструкторів:

$$C_{S} \mapsto C_{L_K} \mapsto \begin{cases} C_{MS}(\mathrm{B},\mathrm{P},\mathrm{n}),\ C_{A_I} \mapsto C_{A,MS_R} \mapsto \Omega_i(C_{MS}) \\ C_{GF}(\Omega_i(\mathrm{C}_{MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \alpha, \mathrm{m})_I \mapsto C_{B,GF_R} \mapsto \Omega_i(C_{GF}). \end{cases} \quad (4.11)$$

Наведемо інші класичні фрактали, реалізовані даним методом.

Формування фракталу «Крива Коха» наступне:

- аксіома: f;
- правила підстановки: f⟶f-f++f-f;
- кінечний мультисимвольний ланцюг: f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f++ …;
- детермінований фрактал «Крива Коха» (рис. 4.5) є реалізацією конструктора з параметрами $M_x = 1, dM_x = 0, D_x = 0, \alpha = 60^\circ$;
- стохастичний фрактал «Крива Коха» (рис. 4.6) є реалізацією конструктора з параметрами $M_x = 1, dM_x = 0{,}5, D_x = 0{,}5, dDx = 0{,}25, \alpha = 60^\circ, d\alpha = 30^\circ$.

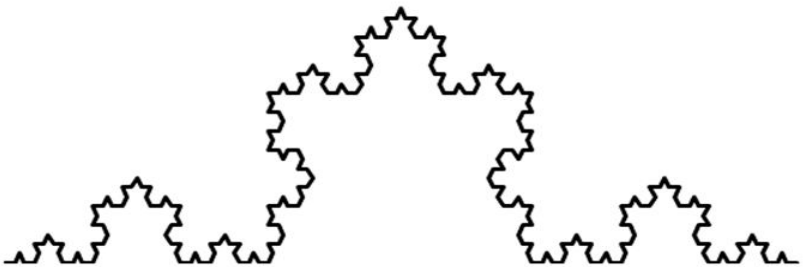


Рисунок 4.5 – Реалізація детермінованого фрактала «Крива Коха»

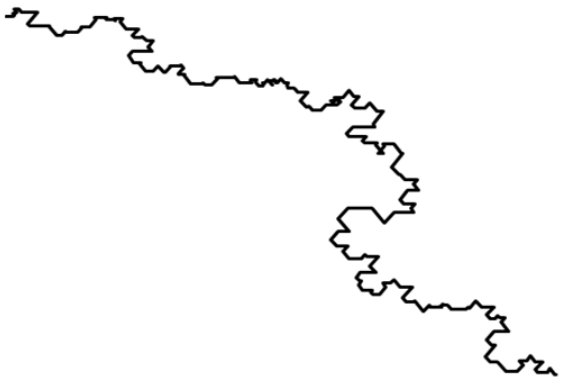


Рисунок 4.6 – Одна з реалізацій стохастичного фрактала «Крива Коха»

Формування фракталу «Крива Госпера» наступне:

- аксіома: xf;
- правила підстановки: y->-fx+yfyf++yf+fx--fx-y, x->x+yf++yf-fx--fxfx-yf+ ;
- кінечний мультисимвольний ланцюг: x+yf++yf-fx--fxfx-yf++-fx+yfyf++yf+fx--fx-yf++-fx+yfyf++yf+fx--fx-yf-fx+yf++yf-fx--fxfx-yf+--fx+yf++yf-fx--fxfx-yf+fx+yf++yf-fx--fxfx-y…;
- детермінований фрактал «Крива Госпера» (рис. 4.7) є реалізацією конструктора с параметрами $M_x = 1, dM_x = 0, D_x = 0, \alpha = 60^\circ$;
- стохастичний фрактал «Крива Госпера» (рис. 4.8) є реалізацією конструктора с параметрами $M_x = 1, dM_x = 0{,}5, D_x = 0{,}5, dDx = 0{,}25, \alpha = 60^\circ, d\alpha = 30^\circ$.

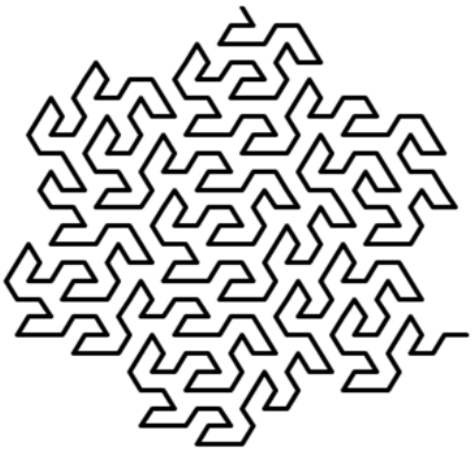


Рисунок 4.7 – Реалізація детермінованого фрактала "Крива Госпера"

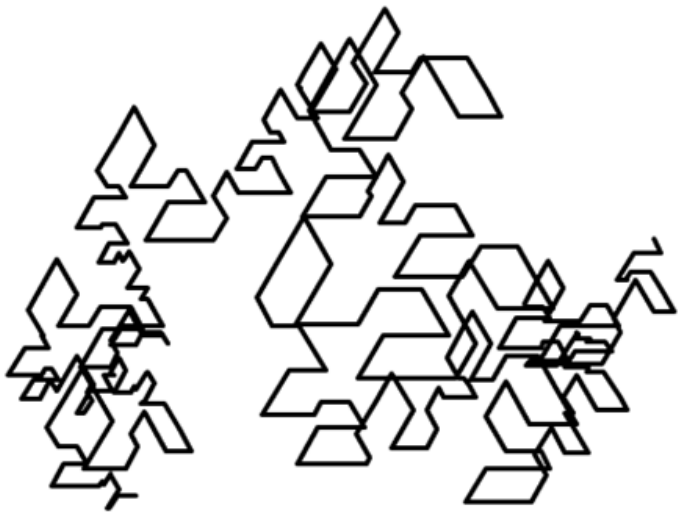


Рисунок 4.8 – Одна з реалізацій стохастичного фрактала "Крива Госпера"

Формування фракталу «Льодовий фрактал» наступне.

- аксіома: f+f+f+f;
- правила підстановки: f→ ff+f++f+f;
- кінечний мультисимвольний ланцюг: ff+f++f+fff+f++f+f+ff+f++f+f++ff+f++f+f+ff+f++f+fff+f++f+fff+f++f+f+ff+f++f+f++ff+f++f+f+ff+f++f+f++ff+f …;
- детермінований фрактал «Льодовий фрактал» (рис. 4.9) є реалізацією конструктора с параметрами $M_x = 1, dM_x = 0, D_x = 0, \alpha = 90^\circ$;
- стохастичний фрактал «Льодовий фрактал» (рис. 4.10) є реалізацією конструктора с параметрами $M_x = 1, dM_x = 0{,}5, D_x = 0{,}5, dDx = 0{,}25, \alpha = 90^\circ, d\alpha = 45^\circ$.

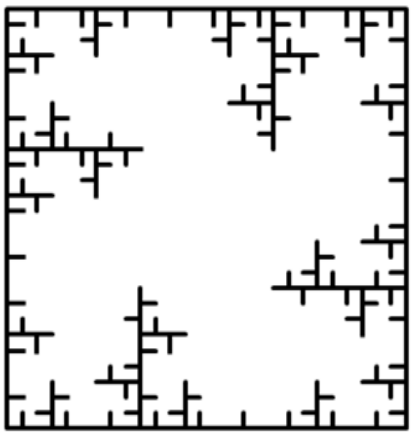


Рисунок 4.9 – Реалізація детермінованого фрактала "Льодовий фрактал"

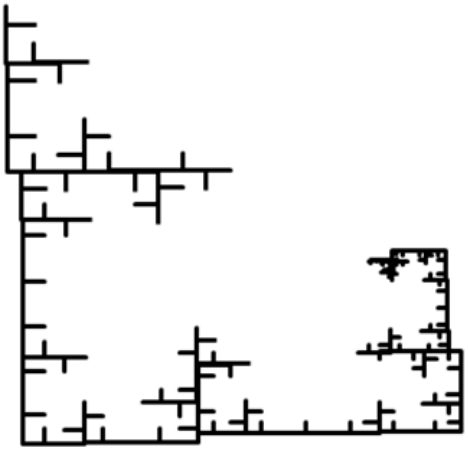


Рисунок 4.10 – Одна з реалізацій стохастичного фрактала "Льодовий фрактал"

Формування фракталу «Крива Серпінського» наступне:

- аксіома: f+xf+f+xf;
- правила підстановки: x→ xf-f+f-xf+f+xf-f+f-x;
- кінечний мультисимвольний ланцюг: f+xf-f+f-xf+f+xf-f+f-xf-f+f-xf-f+f-xf+f+xf-f+f-xf+f+xf-f+f-xf+f+xf-f+f-xf-f+f-xf+f+xf-f+f-xf+f+xf-f+f-xf-f+f-xf-f+f-xf+f+xf-f+f-xf-f+f-xf+f+xf…;
- детермінований фрактал «Крива Серпінського» (рис. 4.11) є реалізацією конструктора з параметрами $M_x = 1, dM_x = 0, D_x = 0, \alpha = 90^\circ$;
- стохастичний фрактал «Крива Серпінського» (рис. 4.12) є реалізацією конструктора з параметрами $M_x = 1, dM_x = 0{,}5, D_x = 0{,}5, dDx = 0{,}25, \alpha = 90^\circ, \ d\alpha = 45^\circ$

.

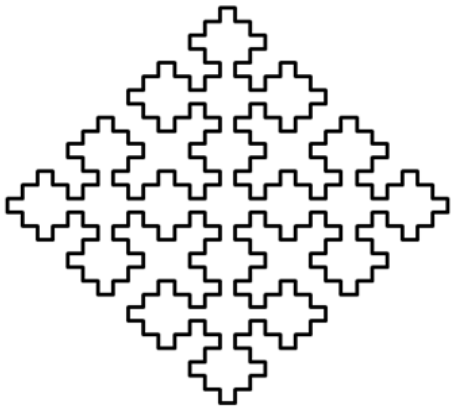


Рисунок 4.11 – Реалізація детермінованого фрактала "Крива Серпінського"

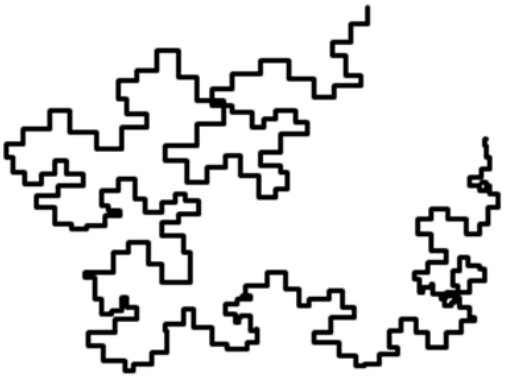


Рисунок 4.12 – Одна з реалізацій стохастичного фрактала "Крива Серпінського"

Формування фракталу «Трикутники» наступне:

- аксіома: f+f+f;
- правила підстановки: f→ f-f+f;
- кінечний мультисимвольний ланцюг: f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f+f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f-f-f+f-f-f+f+f-f+f+f-f+f-f-f+f+f-f+f+f-f+f-f-f+f+f …;
- детермінований фрактал «Трикутники» (рис. 4.13) є реалізацією конструктора з параметрами $M_x = 1, dM_x = 0, D_x = 0, \alpha = 120^{\circ}$;
- стохастичний фрактал «Трикутники» (рис. 4.14) , є реалізацією конструктора з параметрами $M_x = 1, dM_x = 0,5, D_x = 0,5, dDx = 0,25, \alpha = 120^{\circ}, d\alpha = 60^{\circ}$.

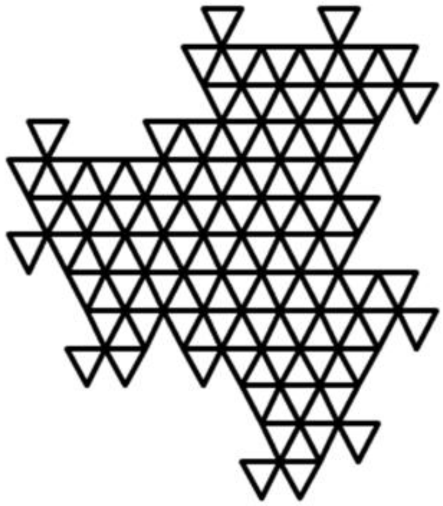


Рисунок 4.13 – Реалізація детермінованого фрактала «Трикутники»

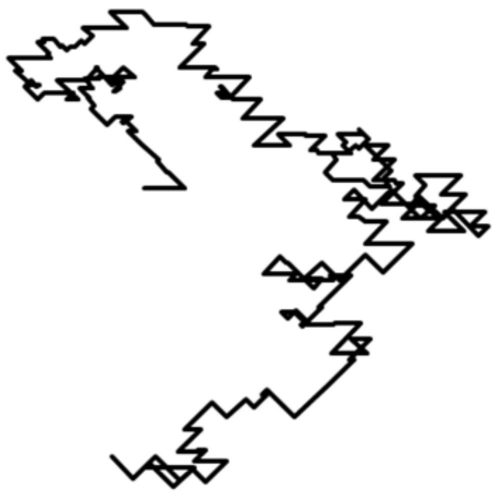


Рисунок 4.14 – Одна з реалізацій стохастичного фрактала «Трикутники»

4.3 Конструктивно-продукційні моделі фрактальних часових рядів

Назвемо L-структурою наступну спеціалізацію узагальненого конструктору:

$$C = \langle M, \Sigma, \Lambda \rangle \ {}_S\mapsto \ C_{LS} = \left\langle M_{LS}, \Sigma_{LS}, \Lambda_{LS} \right\rangle,$$

де носій M_{LS} структури C_{LS} складається з конструктивних елементів із набором атрибутів, які пов'язані зі статичними, динамічними або складовими властивостями елементів. Сигнатура Σ_{LS} складається з імен операцій, що володіють набором атрибутів, і складається з безлічі операцій: зв'язування, підстановки і виведення, операцій над атрибутами.

Розглянемо можливості застосування УК для моделювання фрактальних часових рядів на основі L-систем.

Нехай носій M_K узагальненого конструктору ${}_KC$ містить термінали $\{f\} \in M_K$ й атрибути конструювання $M_x, dM_x, D_x, \alpha, \alpha t$; сигнатура операцій $\{+,-\} \in \Sigma_K$; інформаційне забезпечення Λ_K містить: онтологію з Λ_G, доповнену онтологією L-систем, правила конструювання: аксіому и правила підстановки, обмеження: $n = 8, M_x = 1, dM_x = 0, D_x = 0, \alpha = 90, \alpha t = 90$.

На першому кроці узагальнений конструктор формує мультисимвольні послідовності, що містять властивість самоподібності. Розглянемо уточнюючі перетворення узагальненого конструктору:

$$_{KS}C = \left\langle M_{S_1}, \Sigma_{S_1}, \Lambda_{S_1} \right\rangle \mapsto \left\langle {}_{KS_1}C, C_A \right\rangle \ {}_I\mapsto {}_{KS_1I_1}\ C = \left\langle M_{I_1}, \Sigma_{I_1}, \Lambda_{I_1} \right\rangle \ {}_R\mapsto {}_{KS_1I_1R_1}\ C \qquad (4.12)$$

Специалізація узагальненого конструктору: онтологію в Λ_{S_1} доповнимо семантикою елементів носія й операцій. Їх значеннями є зображення відповідних символів $\{f, +, -\}$. f – покажчик необхідності обчислення значення часового ряду, операції $+, -$ змінюють значення M_x на величину dM_x.

Інтерпретація узагальненого конструктору:

$$\left(A_k \Big|_{M_x, dM_x, i}^{x(t_i), i} \hookleftarrow f \right); \left(A_{k+1} \Big|_{M_x, dM_x}^{M_x} \hookleftarrow + \right); \left(A_{k+2} \Big|_{M_x, dM_x}^{M_x} \hookleftarrow - \right), \qquad (4.13)$$

де A_k – видає нормально розподілене випадкове значення часового ряду, що є нормально розподіленим $x(t_i)$ з математичним очікуванням M_x й дисперсією dM_x, а

також збільшую лічільник i на одиницю, A_{k+1}, A_{k+2} – збільшують та зменшують (відповідно) M_x на величину dM_x.

Розглянемо конкретний приклад узагальненого конструктору, описаного для формування часових рядів у вигляді автономного породжуючого . Якщо початкова аксіома має вигляд f++f++f , правила підстановки f→f-f++f--f, тоді узагальнений конструктор формує послідовність f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f-f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-f++f-f++f-f-f-f++f-.

Для значень атрибутів конструювання: початкове математичне очікування величини та зміна математичного очікування відповідно $M_x = 1, dM_x = 0.5$, початкова дисперсія величини та зміна величини дисперсії відповідно $D_x = 0, dD_x = 0$, початкове математичне очікування періоду та зміна періоду $\alpha = 60, \alpha t = 30$. Реалізацією є часовий ряд, представлений на рис. 4.15. На рис. 4.16 представлено фрактальний часовий ряд зі стохастичними шумами $D_x = 0.5, dD_x = 0.25$.

Рисунок 4.15 – Реалізація детермінованого часового ряду

Рисунок 4.16 – Одна з реалізацій стохастичного часового ряду

Для реалізації задачі моделювання грозового фронту була розроблена низка конструкторів.

У результаті спеціалізації узагальненого конструктора спочатку були розроблені наступні конструктори:

- параметричний породжуючий конструктор «Генетичний пошук» призначений для формування чергового покоління хромосом та визначення їх якості. Па-

раметрами цього конструктору є часовий ряд реального грозового фронту, який моделюється, та параметри генетичного пошуку – гранична кількість поколінь, коефіцієнт мутацій та схрещувань та допустима межа функції допасованості (фітнес функції);

- алгоритмічний конструктор «Генетичні алгоритми», призначений для інтерпретації операцій з конструктору «Генетичний пошук».

Спеціалізація конструктору (у даному випадку конструктору «Генетичний пошук») визначається як:

$$C = \langle M, \Sigma, \Lambda \rangle_S \mapsto {}_S C(rts, lts) = \langle M_S, \Sigma, \Lambda_S \rangle, \quad (4.14)$$

де M_S – включає у собі множину терміналів, нетерміналів, проміжні форми, та інформаційне забезпечення Λ_S – додатково до Λ містить множину змінних й операцій та їх опис. Параметри конструктору: rts – реальний часовий ряд, для якого шукається модель, максимально наближена до нього, lts – кількість елементів ряду rts.

Визначимо предметну область конструктору як формування хромосом, послідовність їх поколінь з застосування схрещувань й мутацій, та визначенням якості хромосом (за функцією допасованості).

Множина змінних згідно Λ_S:

- $acsioma$ – початковий символьний рядок;
- $chromosome\_list$ – список хромосом поточного покоління;
- $iter$ – кількість поколінь;
- $chromosome\_count$ – кількість хромосом в поколінні;
- $efficiency\_coef$ – допустима межа функції допасованості;
- $mutation\_prob$ – кількість операцій мутації;
- $crossover\_prob$ – кількість операцій схрещування;
- $best\_chromosome$ – найкраща хромосома;
- $compare\_data$ – дані реального часового ряду для визначення порівняльної якості хромосом.

Множина операцій згідно Λ_S, де:

- $\lhd(list, count, signature, restrictions)$ – формування покоління хромосом, де $list$ – список хромосом, $count$ – кількість хромосом в поколінні, $signature$ – позначення шуканих терміналів, $restrictions$ – обмеження, які складаються з мінімальних та максимальних значень математичного очікування та дисперсії зміни значення наступної точки ряду до попередньої, кількості символів та множини можливих символів формування початкового символьного рядка та правила підстановки, мінімального та максимального значення кількості ітерації;
- $\prec(list, list, count)$ – мутації хромосом, де $count$ – кількість операцій;
- $\succ(list, list, count)$ – схрещування хромосом;
- $\triangleright(list, list, data, coef)$ – операція визначення допасованості (фітнес функції), де $data$ – реальній часовий ряд для порівняння, $coef$ – допустима межа функції допасованості;
- $*(chromosome, list)$ – визначення найякіснішої хромосоми, де $chromosome$ – визначена хромосома.

Для інтерпретації, розроблено алгоритмічний конструктор «Генетичні алгоритми» з визначеними алгоритмами для формування хромосом – мутація, схрещування, фітнес-функція.

Алгоритми над даними визначаються як $A\big|_X^Y$, де X – вхідні дані, Y – вихідні дані.

Визначили алгоритми виконання операцій:

- $A_1\big|_{list,count,signature,restrictions}^{list}$ – формування хромосом в покоління;
- $A_2\big|_{list,count}^{list}$ – виконання операції схрещування;
- $A_3\big|_{list,count}^{list}$ – виконання операції мутації;
- $A_4\big|_{list,data,coef}^{list}$ – визначення допасованості хромосом;
- $A_5\big|_{list}^{chromosome}$ – визначення найякіснішої хромосоми;
- $A_6^0\big|_{l_j,l_h,l_q}^{l_i}$ – підстановка;
- $A_7^0\big|_{l_j,\psi}^{l_i}$ – частковий вивід;

- $A_8^0\,|_{l_j,\psi}^{l_i}$ – повний вивід.

На мові програмування Python реалізовані ці алгоритми.

Інтерпретація конструктору «Генетичний пошук» за алгоритмічним конструктором «Генетичні алгоритми» визначається як

$$\left\langle C_S(rts,\, lts) = \left\langle M_S, \Sigma, \Lambda_S \right\rangle, C_A = \left\langle M_A, \Sigma_A, \Lambda_A \right\rangle \right\rangle_I \mapsto {}_I C_S(rts,\, lts) = \left\langle M_{SI}, \Sigma_{SI}, \Lambda_{SI}, Z \right\rangle, \quad (4.15)$$

де C_A – алгоритмічний конструктор, M_A – неоднорідний розширюваний носій алгоритмічного конструктору, Σ_A – набір операцій над символами, та інформаційне забезпечення алгоритмічного конструктору, Λ_A – інформаційне забезпечення конструювання алгоритмічного конструктору, Σ_{SI} що включає у собі сигнатуру операцій підстановки, часткового та повного виводу, та інформаційне забезпечення Λ_{SI}, що включає множину операцій та алгоритмів.

Поєднали алгоритми за операціями (кожен алгоритми є атрибутом деякої операції): $\{A_1\,|_{list,count,signature,restrictions}^{list} \hookleftarrow \lhd\}$, $\{A_2\,|_{list,count}^{list} \hookleftarrow \prec\}$, $\{A_3\,|_{list,count}^{list} \hookleftarrow \succ\}$, $\{A_4\,|_{list,data,coef}^{list,chromosome} \hookleftarrow \rhd\}$, $\{A_5\,|_{list}^{chromosome} \hookleftarrow *\}$, $\{A_6^0\,|_{l_j,l_h,l_q}^{l_i} \hookleftarrow \Rightarrow\}$, $\{A_7^0\,|_{l_j,\psi}^{l_i} \hookleftarrow |\Rightarrow\}$, $\{A_8^0\,|_{l_j,\psi}^{l_i} \hookleftarrow ||\Rightarrow\}$.

При інтерпретації конструктори «Генетичний пошук» та «Генетичні алгоритми» поєднуються, та після визначення перетворень формується конструктивна система, що здатна виконувати пошук значень на основі реальних даних. Наявність параметрів конструктору «Генетичний пошук» дало можливість застосувати цей конструктор для відновлення шести часових рядів не змінюючи всі складові конструктору.

Конкретизація конструктору визначається як

$$C_{SI}(rts,\, lts) = \left\langle M_{SI}, \Sigma_{SI}, \Lambda_{SI} \right\rangle_K \mapsto {}_K C_{SI}(rts,\, lts) = \left\langle M_{SIK}, \Sigma_{SIK}, \Lambda_{SIK} \right\rangle, \quad (4.16)$$

де M_{SIK} – розширяємий носій що містить визначений набір термінальних та нетермінальних символів, Σ_{SIK} – множина операції формування рядка, Λ_{SIK} – інформаційне забезпечення процесу конструювання.

При конкретизації визначимо початкові значення даних, як початкова символьна послідовність, та задамо правила підстановки.

Носій M_{SIK} включає нетермінальний символ B та термінальний символ $list$.

Сигнатура операцій $\Sigma_{SIK} = \Sigma_{SI}$.

Визначається при конкретизації Λ_K наступне:

- мета конструювання – знаходження хромосоми, у якій закодовано найкращий модельний ряд;
- обмеження – максимальна кількість ітерацій (поколінь) $iter = 1000$;
- початкові умови – початкова аксіома $acsioma = "B"$, набір хромосом $chromosome\_list$ не містить хромосом;
- правила підстановки:

$$\psi_1 = \left\langle s_1 = \left\langle B \to list\ B \right\rangle, g_1 = \left\langle \begin{matrix} \lhd(list, count, signature, restrictins), \\ \prec(list, count), \succ(list, count), \\ \rhd(list, data, coef) \end{matrix} \right\rangle \right\rangle,$$

$$\psi_2 = \left\langle s_2 = \left\langle B \to \varepsilon \right\rangle, g_2 = \left\langle *(chromosome, list) \right\rangle \right\rangle$$

- умова завершення – виконання кількості ітерацій $iter$.

Після визначення спеціалізації, інтерпретації та конкретизації, конструктивна система $C_{SIK}(rts, lts)$ готова для реалізації.

Реалізація конструктору визначається як

$$C_{SIK}(rts, lts) = \left\langle M_{SIK}, \Sigma_{SIK}, \Lambda_{SIK} \right\rangle_R \mapsto \Omega(C(rts, lts)), \qquad (4.17)$$

де $\Omega(C_{SIK}(rts, lts))$ – конструкція породжувана конструктором $C_{SIK}(rts, lts)$ у даному випадку результуючою конструкцією є найліпша знайдена хромосома, у якій закодовано інформація для відтворення часового ряду, наближеного до реального заданого параметрами.

Для інтеграції результатів конструктивної системи $C_{SIK}(rts, lts)$ й формування та відображення шести модельних часових рядів, розроблені наступні автономні конструктори:

- розрахунку точок на основі хромосом «Грозовий фронт»;
- відображення блискавок у вигляді грозового фронту «Графічне відображення».

Ці конструктори інтерпретуються відповідними алгоритмічними конструкторами:

- розрахунку точок «Часовий ряд»;
- формування графічного відображення «Графічні алгоритми».

Всі розроблені конструктори об’єднані в мультиконструкторі «Моделювання грозового фронту».

У даному випадку, зв'язок між конструкторами «Генетичний пошук» та «Грозовий фронт» й «Грозовий фронт» та «Графічне відображення» виконаний через передачу конструкцій в якості початкових умов. Передача забезпечується засобами мультиконструктора.

Послідовність виконання конструкторів та передача даних конструювання виконується у мультиконструкторі «Моделювання грозового фронту» представлено схемою конструкторів на рис. 4.17.

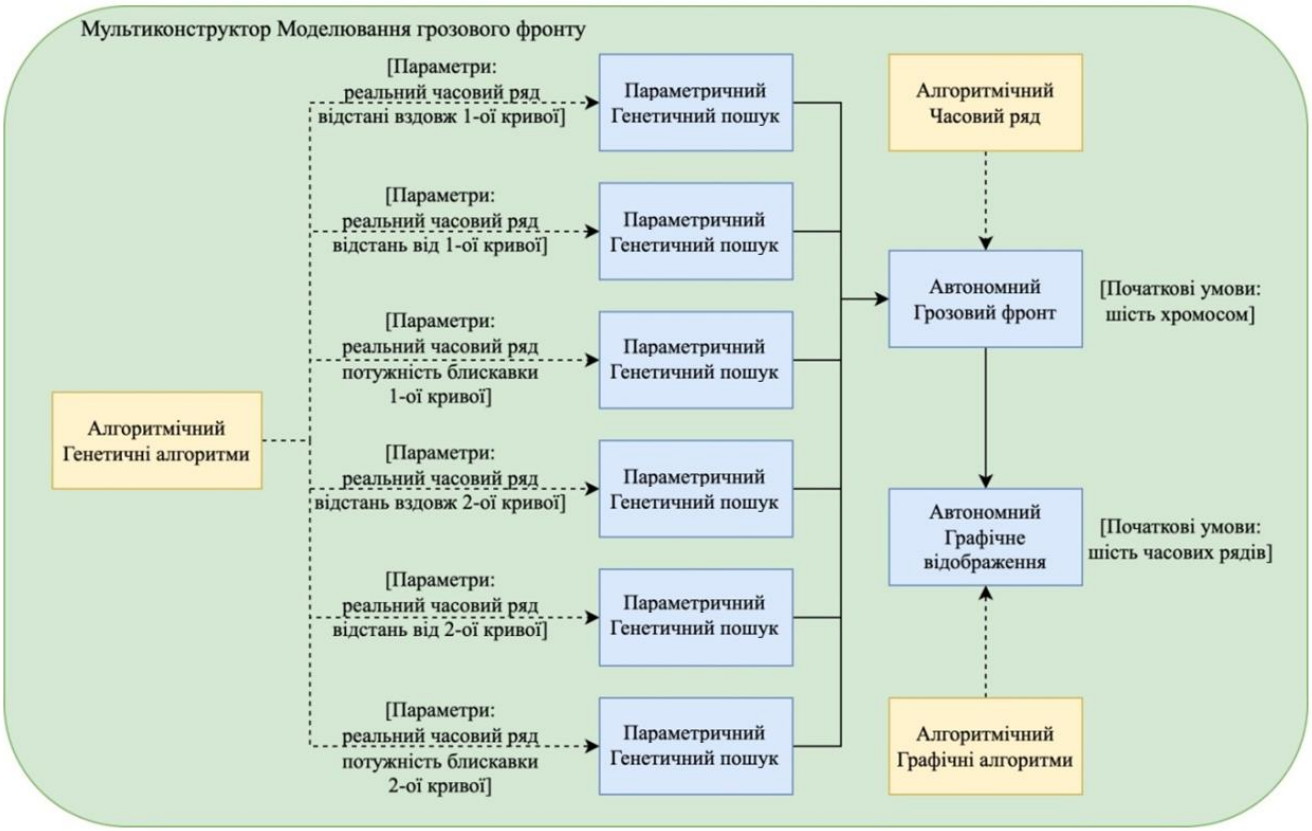


Рисунок 4.17 – Схема конструкторів моделювання грозового фронту

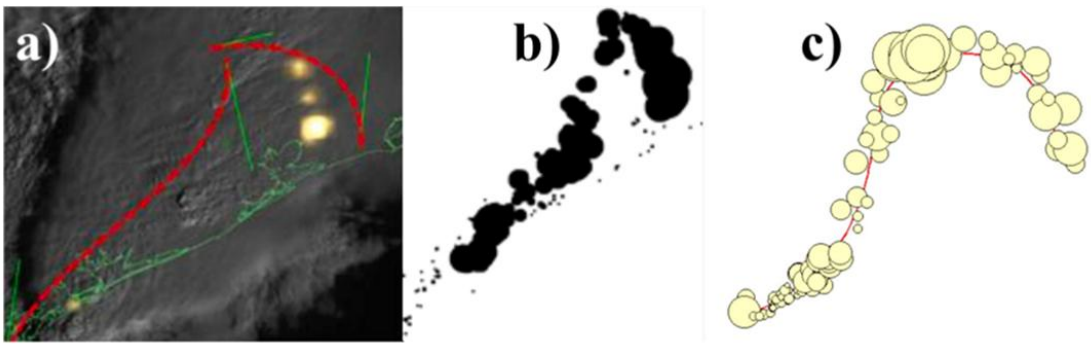


Рисунок 4.18 – Модель грозового фронту

Одна з реалізацій мультиконструктору «Моделювання грозового фронту» моделі грозового фронту приведена на (без анімації, як зведене зображення з усіх кадрів).

Отриманий результат (рис. 4.18, c) був порівняний зі згрупованими блискавками (рис. 4.18, b), отриманими після покадрової обробки оригінального відеофайлу (рис. 4.18, a).

Порівняння реального та модельного рядів наведено на рис. 4.19. Позитивні та негативні значення потужності слугують для відображення блискавок з однієї чи іншої сторони кривої Безьє.

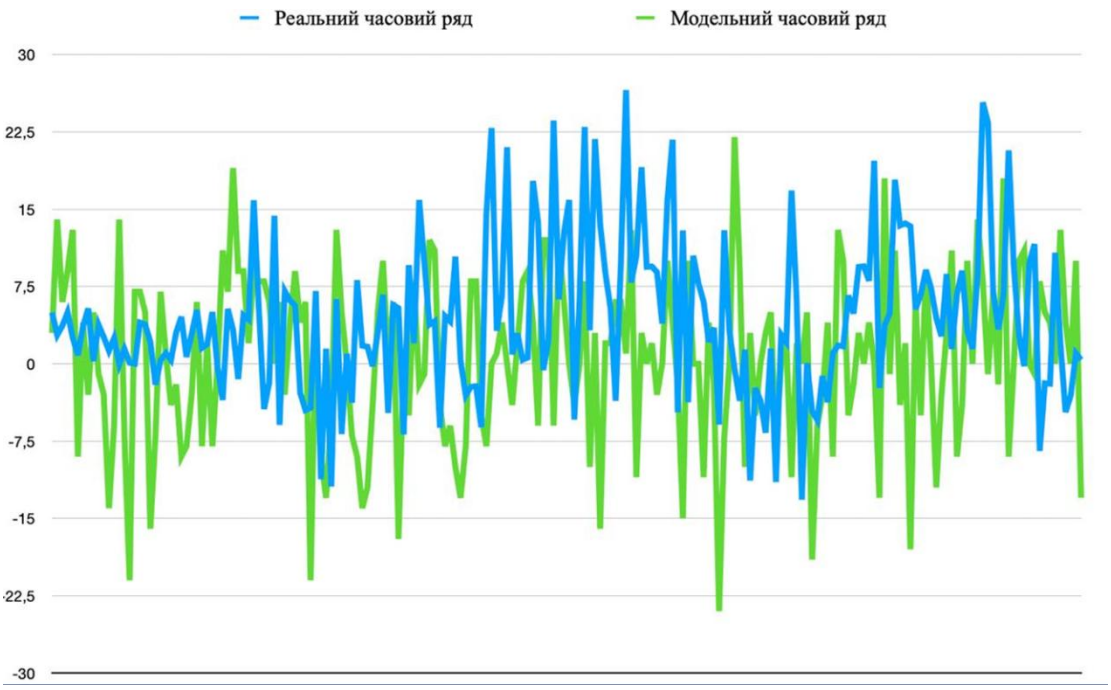


Рисунок 4.19 – Порівняння значень еталоного та модельного рядів блискавок

4.4 Бієктивні відображення фракталів

Були проведені формування різних відображень класичних фракталів. Сформовані форми у вигляді мультисимвольної послідовності, отримані з правил підстановок на основі аксіоми, графічного фракталу та часового ряду. Приклад формування детермінованих відображень на основі фракталу «Сніжинка Коха» наведено на рис. 4.20 .

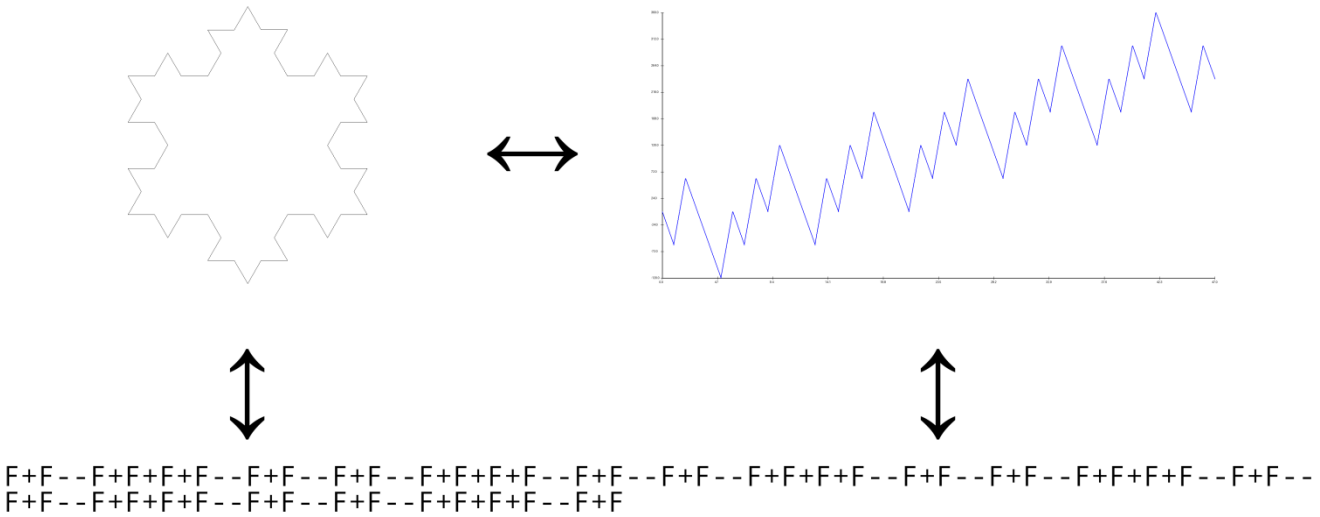
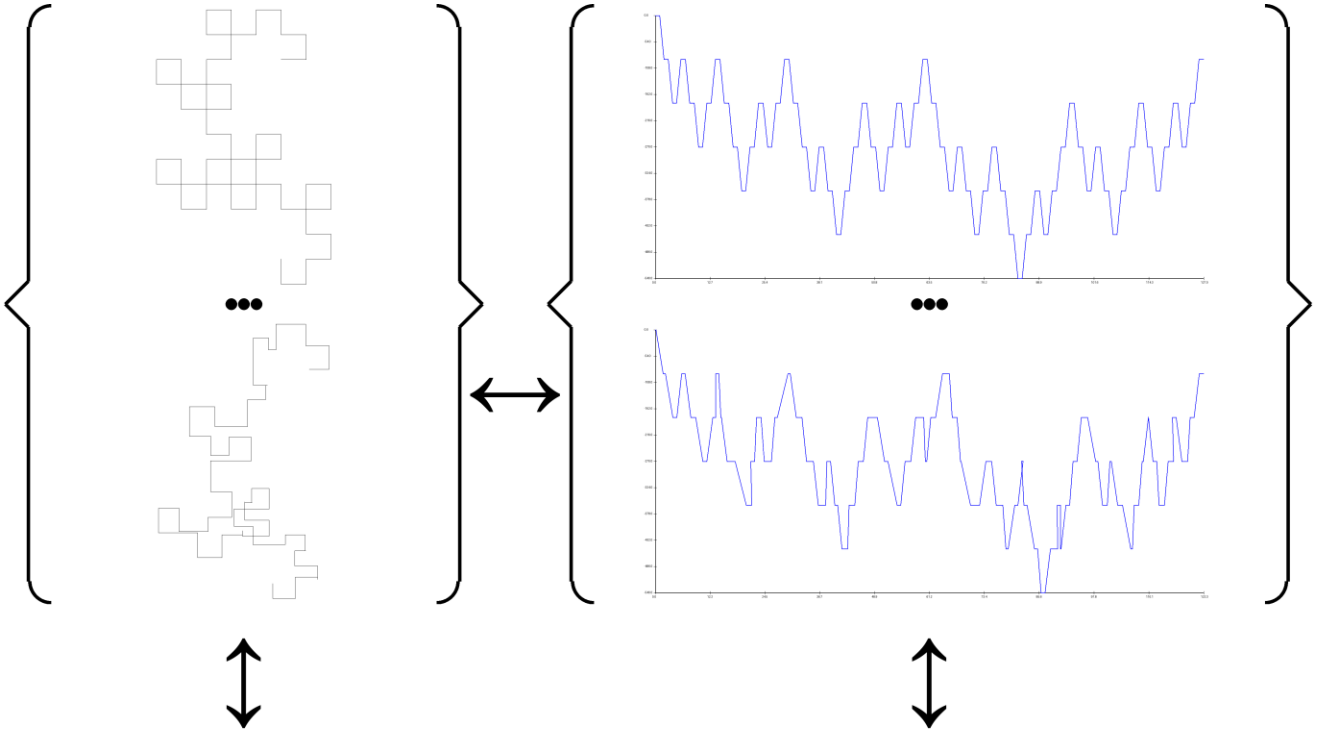


Рисунок 4.20 – Бієктивні відображення детермінованого фракталу "Сніжинка Коха"

Кожному детермінованому мультисимвольному фракталу ставиться у відповідність графічне відображення «Сніжинки Коха» та деякого часового ряду. Наявність мультисимвольного фракталу забезпечує бієкцію між графічним зображенням та часовим рядом сформованих одним конструктором при різних: елементарній базі носія і інтерпретації операції відображення елементів мультисимвольної послідовності.

Також, проведено формування для часового ряду та двовимірної фігури формувалися зображення при зміні дисперсії довжини лінії та кута повороту, та їх бієктивне відображення в шаг часу та значення рівня ряду.

Наведено бієкції для фракталів «Крива Гартера-Гейвея» (рис. 4.21), «Крива Госпера» (рис. Б.1), «Льодовий фрактал» (рис. Б.2), «Крива Коха» (рис. Б.3), «Сніжинка Коха» (рис. Б.4), «Крива Серпінського» (рис. Б.5), «Трикутники Серпінського» (рис. Б.6), «Трикутники» (рис. Б.7).



```
FX+YF++-FX-YF++-FX+YF+--FX-YF++-FX+YF++-FX-YF+--FX+YF+--FX-YF++-FX+YF++-FX-Y
F++-FX+YF+--FX-YF+--FX+YF++-FX-YF+--FX+YF+--FX-YF++-FX+YF++-FX-YF++-FX+YF+--
FX-YF++-FX+YF++-FX-YF+--FX+YF+--FX-YF+--FX+YF++-FX-YF++-FX+YF+--FX-YF+--FX+Y
F++-FX-YF+--FX+YF+--FX-YF+
```

Рисунок 4.21 – Бієктивні відображення фракталу "Крива Гартера-Гайвея"

Кожному детермінованому мультисимвольному фракталу ставиться у відповідність деяка множина стохастичних графічних відображень «Крива Гартера-Гайвея» та множини стохастичних часових рядів. Наявність мультисимвольного фракталу забезпечує бієкцію між множинами графічних зображень та часових рядів сформованих одним конструктором при різних: елементарній базі носія і інтерпретації операції відображення елементів мультисимвольної послідовності.

4.5 Застосування мультиконструкторів для конструювання мультифракталів

Спочатку додаємо автономний породжуючий конструктор для породження плоского фракталу «Трикутник Серпінського».

Перейдемо до уточнюючих перетворень конструктора.

При спеціалізації визначаємо основні елементи, що відповідають за збереження трикутників та встановлюємо призначення кожного з них. Спеціалізація

означує атрибутивність конструктора та його використання у подальшому.Без спеціалізації, інтерпретації та конкретизації реалізація не доступна (не можлива), але вони можуть бути задані у будь-якій послідовності та зміна одного може привести до зміни іншого.

При інтерпритації додаємо відповідний алгоритмічний конструктор для творення нових трикутників.

Визначемо основні алгоритми у алгоритмічному конструкторі. Його спеціалізація полягає у наданні налагоджених процедур, які спроможні виконувати всі операції конструювання даного плоского фракталу. Даний конструктор поєднується з автономним конструктором, та може бути використаний не лише у одному конструкторі, оскільки він виконує функцію групування алгоритмічних процесів за означеною предметною областю. Зміна алгоритмічного конструктору приведе до зміни у всіх процесах на його основі в інших конструкторах.

При творені трикутників будемо виймати трикутники з основного контейнера, для кожного з них створювати нові чорні та білі трикутники та додавати до проміжкових.

Після створення алгоритмічного конструктору повернемося до інтерпретації автономного конструктора де додамо створений алгоритмічний конструктор та встановимо відповідність алгоритмів до операцій. У результаті інтерпретації формується конструктивна система яка поєднує конструктор з елементами і можливими операціями і модель внутрішнього виконавця, здатного ці операції виконувати.

Конкретизація задає правила підстановки (конструювання), які складаються з відношень підстановки і операцій над атрибутами та початкові умови формування. Також, задаються початкові значення параметрів-атрибутів, що були означені у спеціалізації. Значення установлюються відповідно до типів атрибутів. Встановлюємо початкову кількість трикутників та їх розташування на координатній площині у відповідних атрибутах.

Так як конструктор формування плоского фракталу автономний то після всіх уточнюючих перетворень стає можливою його реалізація, тобто формування

множини трикутників, що відображають плоский фрактал «Трикутник Серпінського». Значення координат фігур на координатній площині та їх атрибутика буде збережена у заданий файл для подальшого використання. Формування набору даних виконується на основі встановлених правил та означених алгоритмів у конструкторах.

Далі, кожному з трикутників з отриманого набору ставиться у відповідність певна фрактальна фігура як «Сніжинка Коха» чи «Крива Серпінського». Дані фрактали формуються на основі відповідних еталонних значень L-систем.

Створимо параметричні конструктори формування мультисимвольних стрічок для таких фракталів.

Спочатку, створимо параметричний мультисимвольний конструктор для «Сніжинки Коха».

При спеціалізації визначемо предметну область конструкторів як формування фракталів такого виду та встановимо основні атрибути, що відповідні для формування стрічок на основі вхідного набору даних параметричного конструктору.

Для інтерпретації, додамо алгоритмічний конструктор з визначеними алгоритмами для формування мультисимвольних стрічок на основі правил L-систем.

Додамо новостворений алгоритмічний конструктор до даного параметричного конструктору.

При конкретизації визначемо правила підстановки та початкову символьну стрічку для конкретного фракталу «Сніжинка Коха».

При реалізації, даний конструктор створить мультисимвольні конструкції у кожному з трикутників, які будуть передані у нього як параметри.

Аналогічно створено конструктор формування «Кривої Серпінського».

Наступною дією створюємо параметричні конструктори розрахунку координат для фракталів на основі мультисимвольних стрічок.

Додамо параметричний конструктор для розрахунку координат для «Сніжинки Коха».

Визначемо спеціалізацію при якій встановлюється основний елементом набір трикутників з їх розташуванням на координатній площині з мультисимвольною стрічкою фракталу.

Створюємо для даного конструктору алгоритмічний конструктор з описаними алгоритмами розрахунку точок координат при трикутниках.

Приєднаємо конструктор до розглядаємого параметричного.

Реалізацією даного конструктору є конструкція з набору точок для «Сніжинки Коха».

Відповідно створюємо параметричний та автономний конструктор для формування точок для фракталу «Крива Серпінського».

Для графічного відображення фігур, утворених у наслідок реалізацій конструкторів, створено параметричний графічний конструктор для відображення мультифракталу «Трикутник Серпінського», що містить у собі чорні трикутники з фракталом «Сніжинка Коха» та білі трикутники з фракталом «Крива Серпінського».

При спеціалізації встановимо отримання параметрами набори чорних та білих трикутників.

Створимо алгоритмічний конструктор, що містить алгоритми відображення даних на координатній площині.

Додамо зв’язки між алгоритмами та операціями у інтерпретації параметричного конструктора.

Після визначення усіх автономних та параметричних конструкторів з відповідними алгоритмічними конструкторами, поєднаємо їх в один послідовний процес формування конструкції у мультиконструкторі. Додамо мультиконструктор до списку конструкторів.

На етапі спеціалізації визначемо послідовність дій зі створення конструкторів, формування конструкцій та передачі даних між конструкторами.

При реалізації мультиконструктора отримаємо файл з зображенням мультфракталу «Трикутник Серпінського» з фракталами «Сніжинка Коха» та «Крива

Серпінського» у трикутниках (рис. 4.22). Зображення відповідне до вигляду еталоних відповідних фракталів. [69]

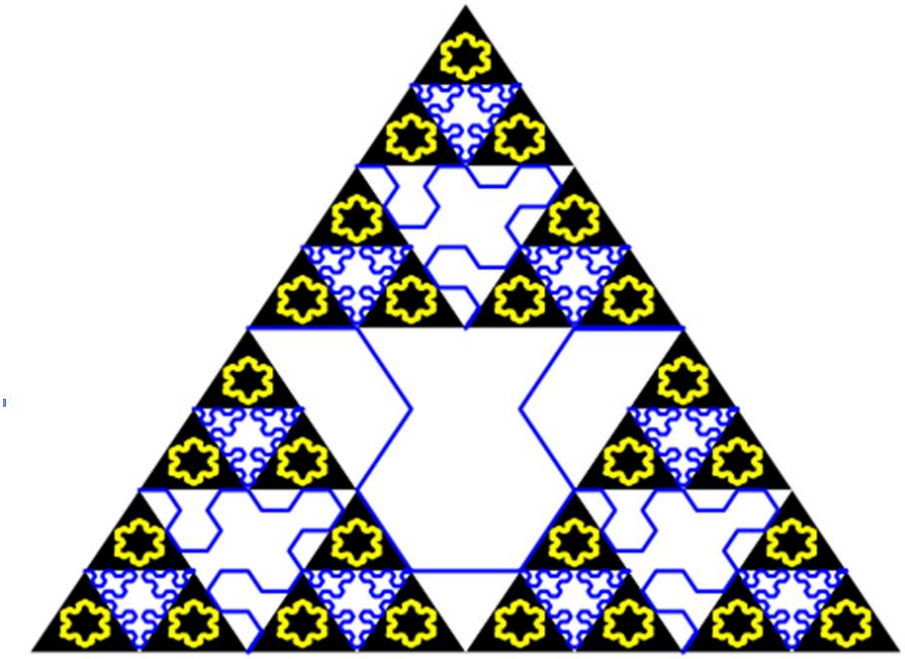


Рисунок 4.22 – Мультифрактал з трикутників Серпінського
та вкладених сніжинок Коха

4.6 Модель ітеративної генерації тривимірних фрактальних поверхонь

Для представлення тривимірних об’єктів розглянули різні представлення у 3D-графіці. Критеріями до вибору становили можливість налаштування деталізації згладжування форми об’єкту та простота моделювання складної фігури з декількох части.

Для формування тривимірних об’єктів, які будуть використовуватися у конструкторах, було розроблено програмний застосунок «Поверхні Безьє». Через інтерфейс налаштовуються опорні точки та виконується попередній перегляд поверхні (рис. 4.23).

Розглянемо модель пагоди для подальшого перетворення у вигляд конструктивної моделі. За одиницю відтворення виділимо один рівень споруди, що буде зменшуватися у розмірі до попереднього поверху. Представимо рівень у вигляди множини поверхонь, поєднаних між собою певними сторонами одна до одної фо-

рмуючи тривимірне зображення. Чотири поверхні поєднуються між собою у представлення рівня, як представлено на рис. 4.24.

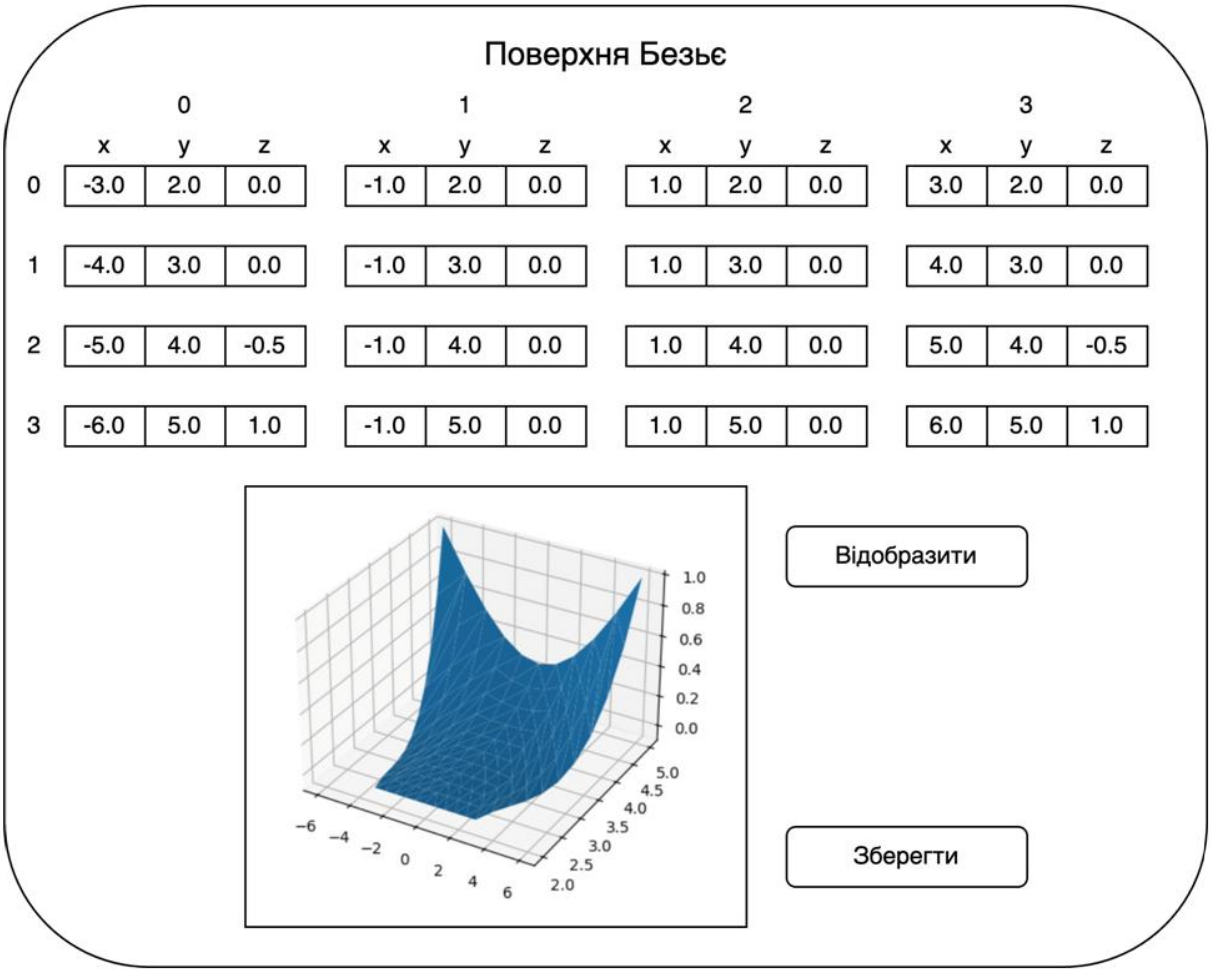


Рисунок 4.23 – Підготовка поверхні

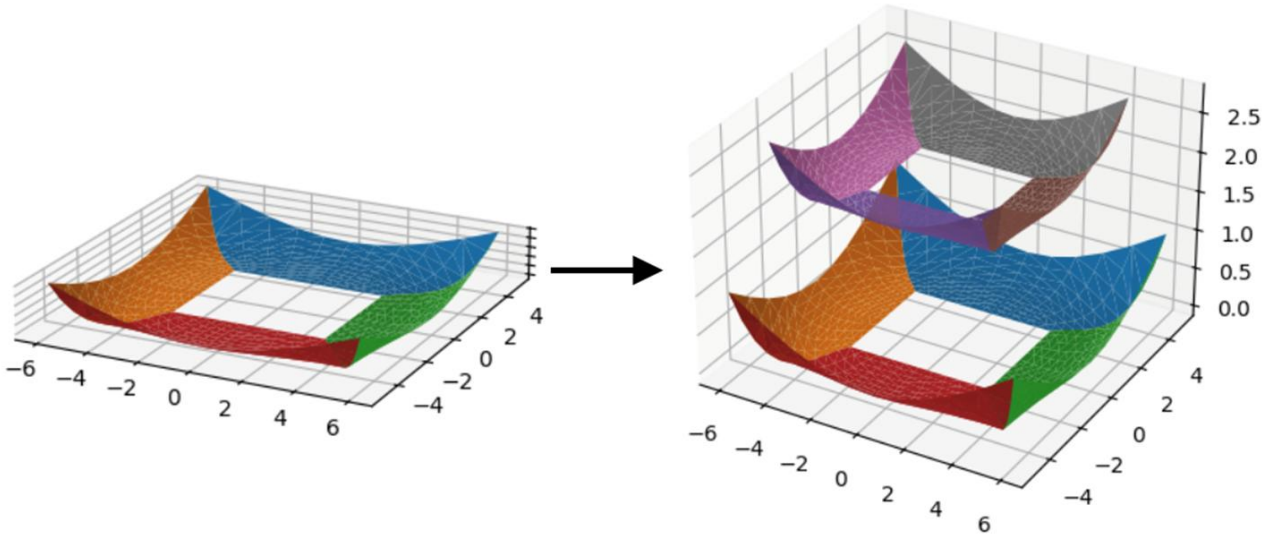


Рисунок 4.24 – Перетворення рівня поверхонь

Для відтворення даної структури за принципами конструктивно-продукційного моделювання розроблено низку конструкторів та їх взаємозв'язки. Схема конструкторів представлена на рис. 4.25.

Розробили автономний породжуючий конструктор «Мультисимвольний». Даний автономний конструктор породжує мультисимвольний рядок на базі аксіоми.

Для отримання даних саме для пагоди, встановлюємо $axioma = "aaaa"$, кількість ітерацій $iter = 8$ та відношення підстановки – $\{a \to bc\}$, $\{b \to a\}$.

Кінцевий рядок використаємо при породжені поверхонь об'ємної фігури. Для породження фігури у вигляді пагоди передаватимемо отриманий рядок $axioma\_result$ до конструктору породжуючого поверхні.

Розробили автономний конструктор «Фігура з поверхонь», що породжує поверхні на основі мультисимвольного рядку.

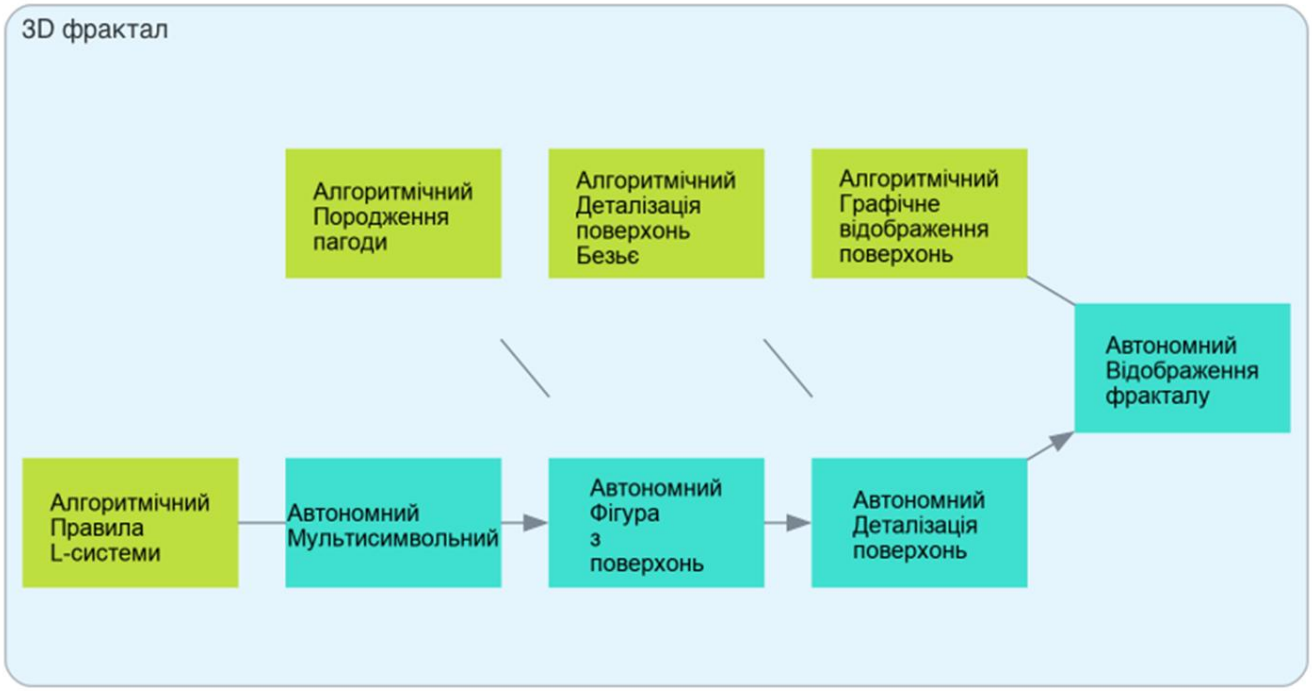


Рисунок 4.25 – Схема конструкторів формування тривимірного фракталу

Визначимо предметну область конструктору як формування структури фігури з набору поверхонь та встановимо основні атрибути, що відповідні для формування послідовностей на основі вхідного набору даних конструктору.

Визначимо дані конструювання: кількість ітерацій *iter* – граничне число кількості ітерацій, *axioma* та *axioma_result* – початковий символьний рядок та результуючий рядок символів після виконання ітераційного процесу замін згідно правил заміни, та набори поверхонь *list_begin*, *list_temp* та *list_res* – відповідно початковий, проміжків та результуючий набір. Значення змінних будуть надані при уточнюючому перетворені конкретизації.

Для інтерпретації, додамо алгоритмічний конструктор «Породження поверхонь» з визначеними алгоритмами для формування набору поверхонь з мультисимвольних послідовностей на основі правил L-систем. Алгоритми формування структури зазначеної форми пагоди також описуються при наповнені.

На мові програмування Python опишемо алгоритми наповнення поверхонь: use_a – алгоритм формування нової поверхні при якому береться одна поверхня з набору початкових поверхонь та розраховується переміщена та зменшена копія даної з додаванням до набору проміжкових поверхонь, use_b – переміщення поверхонь з проміжкового набору до кінцевого набору поверхонь з додатковим копіюванням до набору початкових поверхонь, та алгоритми підстановки, часткового та повного виводу.

Пов’яжемо алгоритмічний конструктор C_A з автономним конструктором «Фігура з поверхонь».

Визначимо алгоритми виконання операцій:

- $A_1^0 \mid_a^{b,c}$ – формування нової частини фігури;
- $A_2^0 \mid_b^a$ – наповнення початкового набору поверхонь;
- $A_3^0 \mid_c^{a,d}$ – наповнення результуючого набору поверхонь;
- $A_4^0 \mid_{l_j,l_h,l_q}^{l_i}$ – підстановка;
- $A_5^0 \mid_{l_j,\psi}^{l_i}$ – частковий вивід;

- $A_6^0 \mid_{l_j,\psi}^{l_i}$ – повний вивід.

Визначається при конкретизації Λ_K наступне:

- мета конструювання – наповнення набору поверхонь для формування багаторівневої тривимірної фігури на основі мультисимвольного рядка;
- обмеження – операції над атрибутами відсутні, кожне правило підстановки містить лише одне відношення підстановки;
- початкові умови – початкова аксіома $axioma = ""$, набір поверхонь $list\_begin$ та $list\_res$ містить по чотири моделі поверхні розміщені у вигляді рівня пагоди, кількість ітерацій $iter = 1$;
- умова завершення – виконання кількості ітерацій $iter$.

Пов'яжемо основні операції необхідні для формування пагоди з символами. При кожному символі «a» розраховується для однієї з поточних поверхонь рівня зменшена копія з перенесенням у здовж осі, після чого зберігається поверхня у двох списках – у списку нових поверхонь $list\_begin$ та у списку усіх поверхонь споруди $list\_res$. При кожному символі «b» переноситься одна з поверхонь списку нових поверхонь $list\_temp$ до списку поточних поверхонь $list\_begin$. При кожному символі «c» переноситься одна усі поверхні зі списку $list\_res$ до списку поточних поверхонь $list\_begin$.

При формуванні фігури за описаними правилами, для нового рівня потрібно взяти кожну з поверхонь поточного рівня та зменшити на відповідний коефіцієнт зменшення з перенесенням усіх точок поверхні у здовж координатної осі паралельно до попереднього стану. Попередній стан заздалегідь зберегти для відтворення структури поверхонь.

При реалізації, даний конструктор сформує набір поверхонь, Безьє для кількості рівнів-ітерацій, описаної при конкретизації.

Аналогічно до розробки конструктору «Фігура з поверхонь» розроблюємо конструктор формування набору точок тривимірного зображення поверхонь «Деталізація поверхонь» та конструктор для формування тривимірного зображення «Відображення фракталу».

Таким же чином, визначаємо параметричні конструктори та алгоритмічні конструктори. Після визначення усіх автономних та параметричних конструкторів з відповідними алгоритмічними конструкторами, поєднаємо їх в один послідовний процес формування конструкції у мультиконструкторі. Додамо мультиконструктор «3D фрактал» до списку конструкторів.

На етапі спеціалізації визначимо послідовність дій з додавання конструкторів, формування конструкцій та передачі даних між конструкторами.

Після ітеративного процесу утворюється зображення багатоповерхневої пагоди у відповідності до кількості ітерацій. Кількість ітерацій відповідає кількості рівнів що доповнюють пагоду від початкового рівня.

Отримана конструкція порівнюється з зображення реальної пагоди та представлена на рис. 4.26 [126].

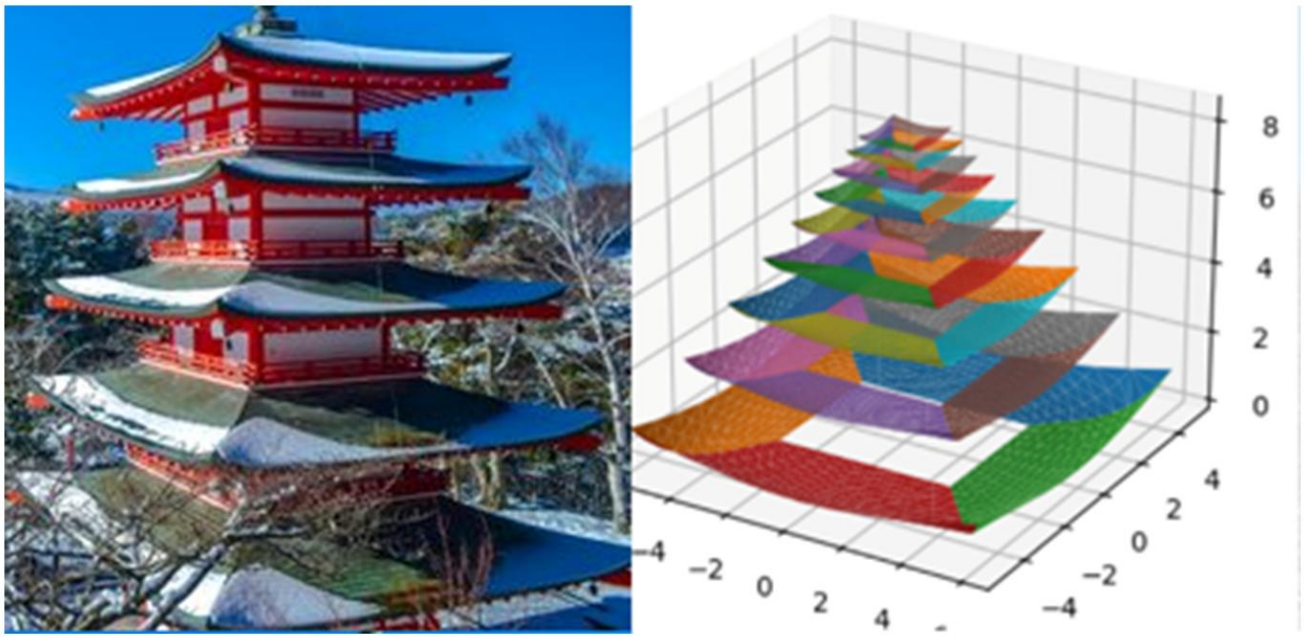


Рисунок 4.26 – Порівняння реального та модельно зображення пагоди

4.7 Експериментальні дослідження фрактальної розмірності стохастичних класичних геометричних фракталів

Підготовка експерименту:

- розроблено програмне забезпечення підрахунку фрактальної розмірності Міньковського. Вхідними даними для розрахунку обрано графічні зображення фракталів, отриманих при реалізації мультиконструкторів формування фракталів;

- виконано тестування на зображеннях детермінованих класичних фракталів з наперед відомими теоретичними значеннями фрактальної розмірності, таких як «Сніжинка Коха», «Крива Коха», «Дракона Гартера-Гайвея», «Трикутник Серпінського», «Крива Серпінського», «Крива Госпера», «Льодовий фрактал» та «Трикутники».

Виконання експерименту: для кожного з вказаних фракталів варіювалася дисперсія довжини лінії з кроком 0.02 від 0.0 до 1.0 з зафіксованим кутом повороту. Виконувалось 100 паралельних замірів для кожного фракталу при кожному значенні дисперсії довжини лінії.

Результати експериментів наведені у пп. 4.7.1 – 4.7.8.

Для аналізу результатів розраховано відхилення інтервалу фрактальної розмірності:

$$\Delta D = \frac{D_{\max} - D_{\min}}{D_{mean}} * 100\%, \tag{4.18}$$

де D_{mean}, $D_{\min}$, $D_{\max}$ – відповідно середнє, мінімальне та максимальне значення фрактальної розмірності.

Результати наведені у графічному та табличному вигляді для кожного з обраних фракталів.

4.7.1 Варіативність фрактальної розмірності стохастичного фракталу «Крива Коха»

Для фракталу «Крива Коха» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 1.26. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.1. Графічне представлення експериментальних розрахунків представлено на рис. 4.27.

Таблиця 4.1 Фрактальна розмірність Кривої Коха з варіативністю дисперсії довжини лінії

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD

0.00	1.29	1.29	1.29	0.00
0.10	1.29	1.25	1.33	6.20
0.20	1.29	1.24	1.32	6.20
0.30	1.28	1.25	1.32	5.47
0.40	1.29	1.23	1.34	8.53
0.50	1.28	1.21	1.33	9.37
0.60	1.28	1.22	1.34	9.37
0.70	1.28	1.24	1.35	8.60
0.80	1.28	1.23	1.33	7.81
0.90	1.28	1.23	1.35	9.37
1.00	1.28	1.23	1.33	7.81

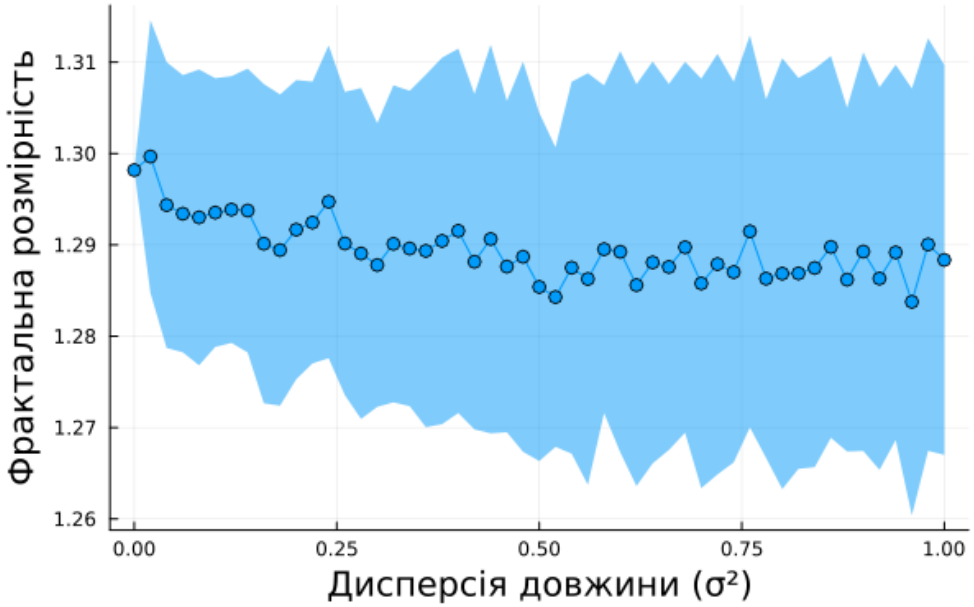


Рисунок 4.27 – Фрактальна розмірность Кривої Коха з варіативністю дисперсії довжини лінії

4.7.2 Варіативність фрактальної розмірності стохастичного фракталу «Сніжинка Коха»

Для фракталу «Крива Коха» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 1.26. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.2. Графічне представлення експериментальних розрахунків представлено на рис. 4.28.

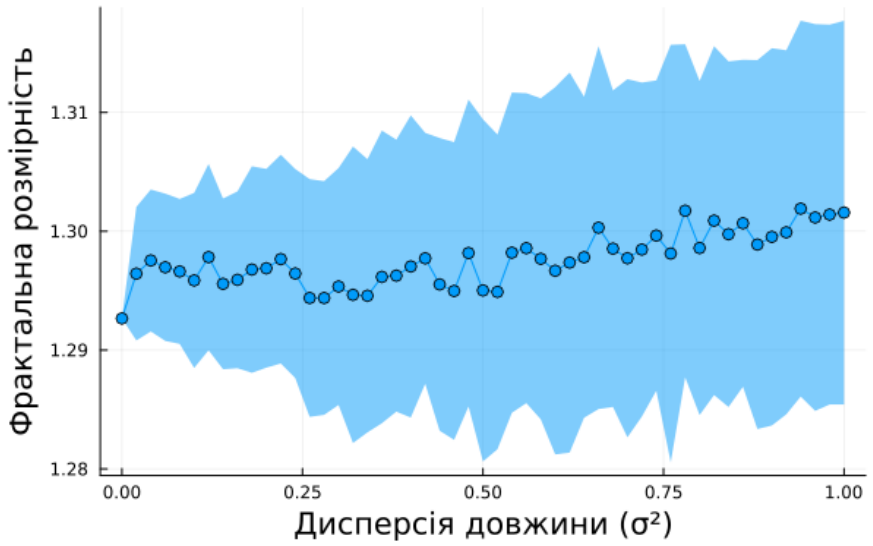


Рисунок 4.28 – Розподілення фрактальної розмірності Сніжинки Коха

Таблица 4.2 Розподілення фрактальної розмірності Сніжинки Коха

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.29	1.29	1.29	0.00
0.10	1.29	1.27	1.31	3.10
0.20	1.29	1.26	1.31	3.87
0.30	1.29	1.26	1.33	5.43
0.40	1.29	1.24	1.33	6.98
0.50	1.29	1.25	1.32	5.43
0.60	1.29	1.26	1.34	6.20
0.70	1.29	1.25	1.33	6.20
0.80	1.29	1.26	1.32	4.65
0.90	1.29	1.25	1.33	6.20
1.00	1.30	1.26	1.33	5.38

4.7.3 Варіативність фрактальної розмірності стохастичного фракталу «Дракон Гартера-Гайвея»

Для фракталу «Дракон Гартера-Гайвея» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 2. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.3. Графічне представлення експериментальних розрахунків представлено на рис. 4.29.

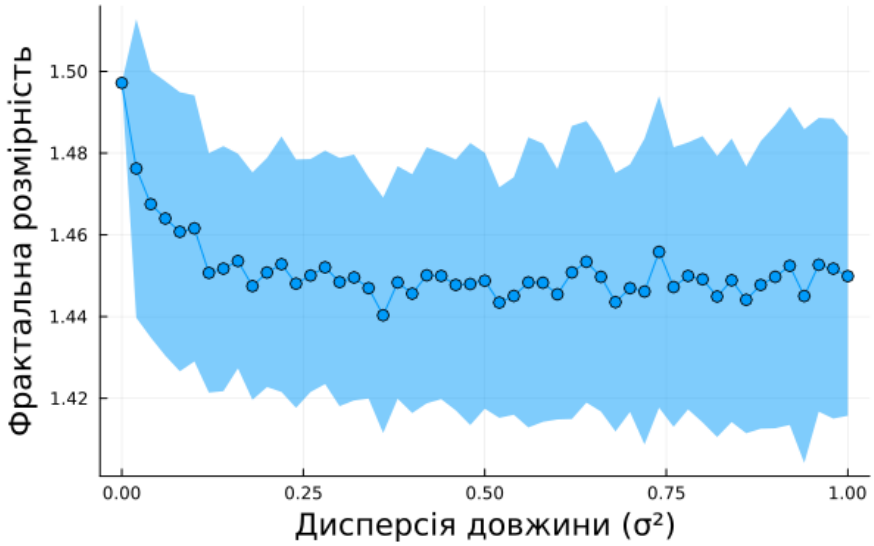


Рисунок 4.29 – Розподілення фрактальної розмірності Дракона Гартера-Гайвея

Таблица 4.3 Розподілення фрактальної розмірності Дракона Гартера-Гайвея

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.49	1.49	1.49	0.00
0.10	1.46	1.40	1.53	8.90
0.20	1.45	1.38	1.52	9.65
0.30	1.44	1.38	1.54	11.11
0.40	1.44	1.37	1.52	10.42
0.50	1.44	1.38	1.54	11.11
0.60	1.44	1.37	1.52	10.42
0.70	1.44	1.38	1.53	10.42
0.80	1.44	1.38	1.52	9.72
0.90	1.44	1.37	1.54	11.80
1.00	1.44	1.38	1.54	11.11

4.7.4 Варіативність фрактальної розмірності стохастичного фракталу «Крива Госпера»

Для фракталу «Крива Госпера» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 2. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.4. Графічне представлення експериментальних розрахунків представлено на рис. 4.30.

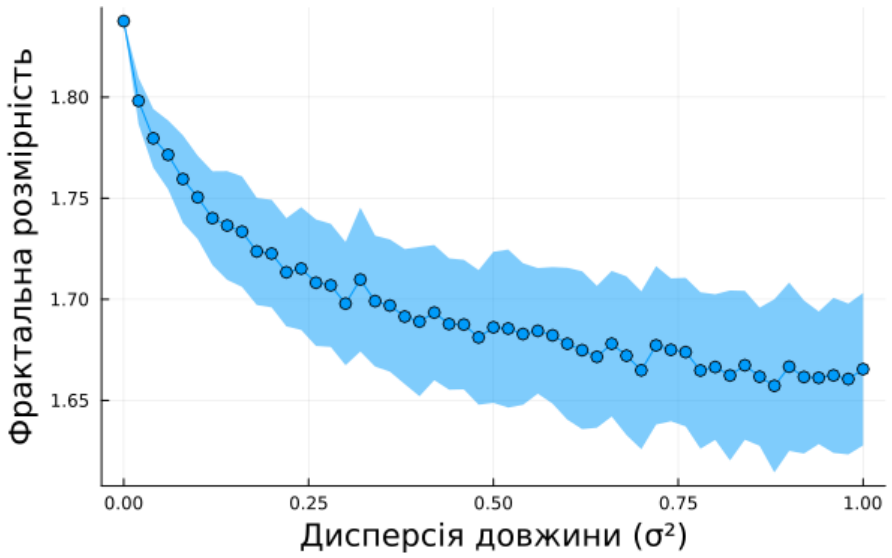


Рисунок 4.30 – Розподілення фрактальної розмірності Кривої Госпера

Таблица 4.4 Розподілення фрактальної розмірності Кривої Госпера

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.83	1.83	1.83	0.00
0.10	1.75	1.69	1.78	6.21
0.20	1.72	1.66	1.77	6.40
0.30	1.69	1.63	1.77	8.30
0.40	1.68	1.61	1.77	9.52
0.50	1.68	1.58	1.78	11.90
0.60	1.67	1.56	1.75	11.38
0.70	1.66	1.57	1.74	10.24
0.80	1.66	1.56	1.75	11.44
0.90	1.66	1.54	1.75	12.65
1.00	1.66	1.53	1.74	12.65

4.7.5 Варіативність фрактальної розмірності стохастичного фракталу «Трикутник Серпінського»

Для фракталу «Трикутник Серпінського» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 1.58. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.5. Графічне представлення експериментальних розрахунків представлено на рис. 4.31.

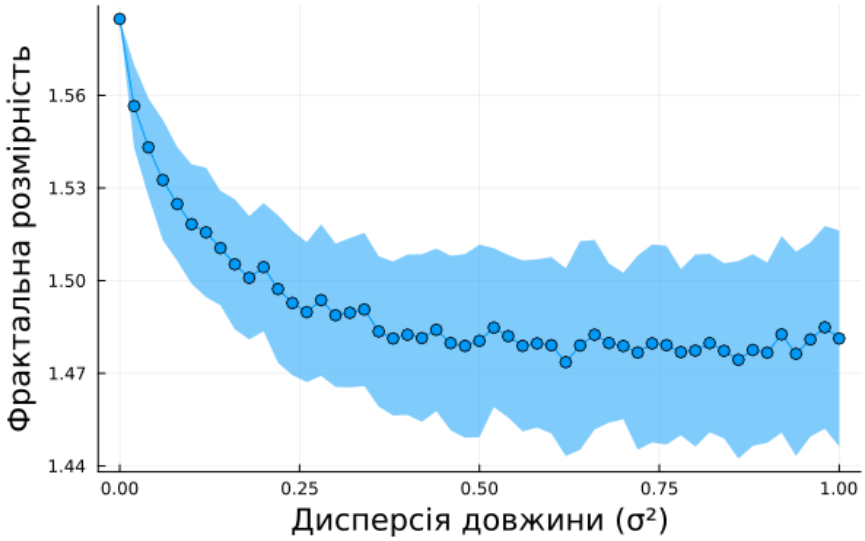


Рисунок 4.31 – Розподілення фрактальної розмірності Трикутника Серпінського

Таблиця 4.5 Розподілення фрактальної розмірності Трикутника Серпінського

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.58	1.58	1.58	0.00
0.10	1.51	1.47	1.56	5.96
0.20	1.50	1.45	1.56	7.33
0.30	1.48	1.43	1.54	7.43
0.40	1.48	1.40	1.54	9.46
0.50	1.48	1.39	1.55	10.81
0.60	1.47	1.40	1.55	10.20
0.70	1.47	1.42	1.53	7.48
0.80	1.47	1.38	1.57	12.92
0.90	1.47	1.39	1.55	10.88
1.00	1.48	1.38	1.55	11.49

4.7.6 Варіативність фрактальної розмірності стохастичного фракталу «Крива Серпінського»

Для фракталу «Крива Серпінського» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 1.58. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.6. Графічне представлення експериментальних розрахунків представлено на рис. 4.32.

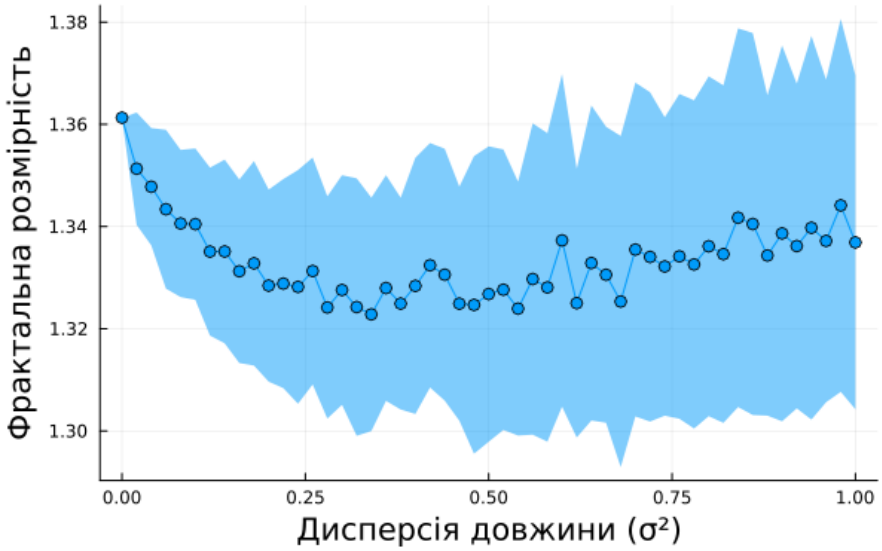


Рисунок 4.32 – Розподілення фрактальної розмірності Кривої Серпінського

Таблиця 4.6 Розподілення фрактальної розмірності Кривої Серпінського

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.36	1.36	1.36	0.00
0.10	1.34	1.29	1.37	5.97
0.20	1.32	1.28	1.36	6.06
0.30	1.32	1.28	1.38	7.57
0.40	1.32	1.28	1.40	9.09
0.50	1.32	1.26	1.40	10.60
0.60	1.33	1.28	1.46	13.53
0.70	1.33	1.25	1.42	12.78
0.80	1.33	1.27	1.44	12.78
0.90	1.33	1.27	1.47	15.04
1.00	1.33	1.27	1.41	10.53

4.7.7 Варіативність фрактальної розмірності стохастичного фракталу «Льодовий фрактал»

Для фракталу «Льодовий фрактал» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 1.72. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.7. Графічне представлення експериментальних розрахунків представлено на рис. 4.33.

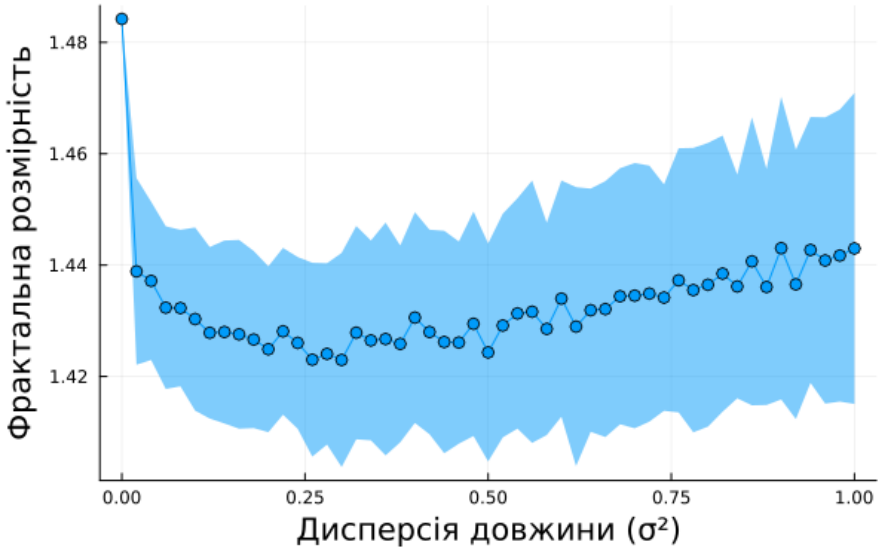


Рисунок 4.33 – Розподілення фрактальної розмірності Льодового фракталу

Таблица 4.7 Розподілення фрактальної розмірності Льодового фракталу

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.48	1.48	1.48	0.00
0.10	1.43	1.38	1.47	6.29
0.20	1.42	1.39	1.46	4.93
0.30	1.42	1.37	1.47	7.04
0.40	1.43	1.38	1.48	6.99
0.50	1.42	1.37	1.47	7.04
0.60	1.43	1.38	1.48	6.99
0.70	1.43	1.38	1.49	7.69
0.80	1.43	1.38	1.50	8.39
0.90	1.44	1.37	1.51	9.72
1.00	1.44	1.38	1.54	11.11

4.7.8 Варіативність фрактальної розмірності стохастичного фракталу «Трикутники»

Для фракталу «Трикутники» теоретично визначена з інших досліджень у детермінованому стані без стохастичних змін довжини лінії й кута повороту та складає 2. Зміна фрактальної розмірності при різних значеннях дисперсії довжини лінії наведено в табл. 4.8. Графічне представлення експериментальніх розрахунків представлено на рис. 4.34.

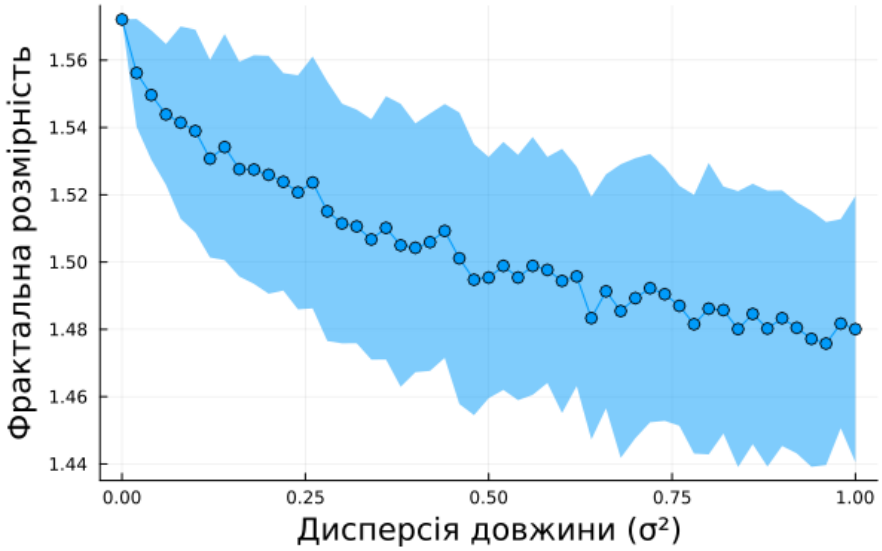


Рисунок 4.34 – Розподілення фрактальної розмірності Фракталу Трикутники

Таблиця 4.8 Розподілення фрактальної розмірності Фракталу Трикутники

Дисперсія	Середня розмірність D_{mean}	Мінімальна розмірність $D_{\min}$	Максимальна розмірність $D_{\max}$	Відхилення від середньої ΔD
0.00	1.57	1.57	1.57	0.00
0.10	1.53	1.46	1.62	10.46
0.20	1.52	1.44	1.63	12.50
0.30	1.51	1.43	1.59	10.60
0.40	1.50	1.43	1.60	11.33
0.50	1.49	1.41	1.58	11.41
0.60	1.49	1.41	1.59	12.08
0.70	1.48	1.39	1.57	12.16
0.80	1.48	1.39	1.60	14.19
0.90	1.48	1.40	1.59	12.84
1.00	1.48	1.38	1.60	14.86

З наведених результатів наочно виявлено, що зі збільшенням дисперсії довжини лінії та кута повороту (з одночасним зменшенням властивості самоподібності), фрактальна розмірність знаходиться в досить великих проміжках. Що в свою чергу свідчить про те, що властивості самоподібності і фрактальної розмірності не співпадають. Можливо, мається якийсь кореляційний зв'язок, але перевірка цієї гіпотези потребує окремого дослідження.

Висновки до четвертого розділу

У четвертому розділі підтверджено, що конструктивно-продукційна парадигма здатна охопити широкий спектр фрактальних структур – від класичних плоских до просторових та стохастичних, уніфікувавши їх через представлення у вигляді мультисимвольних послідовностей і спільний набір конструкторів.

Перший блок досліджень стосувався формування тривимірних фрактальних поверхонь. Створений застосунок «Поверхні Безьє» дає змогу інтерактивно задавати опорні точки, коригувати рівень згладжування й одразу переглядати результат, що спростило доведення конструкцій до потрібної деталізації.

На основі «Конструктор 2.0» розроблено низку конструкторів, яка генерує багаторівневу пагоду: кожна ітерація додає новий, пропорційно зменшений «поверх», а фінальне зображення корелює з реальною архітектурною формою.

Усі вони об’єднані мультиконструктором «Моделювання грозового фронту», що автоматизує передачу даних між стадіями та породжує шість синтетичних часових рядів, наближених до реального грозового сигналу .

Матеріали розділу опубліковані у роботах: [10, 68, 70, 72, 120, 121, 126, 127].

ЗАГАЛЬНІ ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-прикладну задачу формалізованого представлення, генерації та візуалізації фрактальних структур на основі конструктивно-продукційного підходу. Розроблено відповідну теоретичну модель, визначено алгоритми побудови та трансформації конструкцій, а також створено програмне забезпечення, що забезпечує їхню реалізацію з урахуванням параметричної варіативності, контекстних залежностей та можливостей інтерактивного керування.

За результатами дослідження одержано такі основні наукові та прикладні результати:

1. Проаналізовано сучасні підходи до моделювання фракталів, зокрема на основі L-систем, ітераційних функціональних систем, рекурсивних алгоритмів та стохастичних методів. Встановлено, що більшість класичних підходів не забезпечують необхідного рівня керованості та універсальності для створення складних морфологічних структур, особливо в умовах зміни параметрів у режимі реального часу.

2. У результаті експериментальних досліджень встановлено, що при зміні довжини лінії на кожному кроці формування класичних геометричних фракталів з математичним очікуванням та дисперсією, що дорівнюють одиниці, фрактальна розмірність може варіюватися в межах 5…15%.

3. Реалізовано кросплатформене програмне забезпечення «Конструктор 1.1» та браузерне «Конструктор 2.0», яке втілює запропоновані підходи та забезпечує повний цикл формування конструкторів. Розроблене програмне забезпечення є у деякій мірі уніфіковане, що надає можливість його використання у майбутніх дослідженнях з застосуванням конструктивно-продукційного моделювання.

4. Проведено конструктивно-продукційне моделювання фракталів з різною елементною базою носія, а саме класичних плоских та об’ємних геометричних фракталів, часових рядів та мультифракталів з проміжною мультисимвольною

послідовністю, що дозволило, зокрема, встановити бієктивні відображення між фракталами різної природи.

5. В результаті узагальнення методів і засобів конструктивно-продукційного моделювання представлених у багатьох попередніх роботах різних дослідників, сформульовані основні положення конструктивної парадигми та розроблена таксономія конструктивно-продукційного моделювання, що сприятиме подальшому використанню цієї парадигми.

6. Для формалізації зв’язків з керування та даних між сукупністю конструкторів, що вирішують єдину задачу конструювання , розроблено модель мультиконструктора з її візуалізацією.

7. Встановлені залежності фрактальної розмірності класичних геометричних фракталів від стохастичної варіативності довжини лінії та кута нахилу, та на їх основі зроблені висновки щодо того, що при їх формуванні властивості самоподібності та фрактальної розмірності не мають однозначного зв’язку.

8. Розроблені в роботі моделі, методи та програмне забезпечення отримали впровадження у навчальному процесі та науковій діяльності Українського державного університету науки та технологій, зокрема, у науково-дослідних роботах «Конструктивно-продукційне моделювання фракталів» (2018 р. № держреєстрації 0118U004215) та «Підвищення конкурентоспроможності залізничного транспорту на основі уніфікованих інтелектуальних технологій процесів перевезень та експлуатації парків технічних систем» (2018 р. № держреєстрації 0117U004392).

Таким чином, результати дисертаційної роботи є науково обґрунтованими, структурно завершеними та мають високий рівень практичної придатності для задач фрактального моделювання, програмної інженерії та освітньої діяльності. Розроблений підхід дозволяє не лише описувати вже відомі фрактальні структури, але й створювати нові, керовані, адаптивні моделі, що відкриває широкі перспективи подальших досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Barnsley, M. F. (1988). Fractals Everywhere. Academic Press.

2. Bassingthwaighte, J. B., Hunter, P. J., Noble, D. (2009). At the Biological Modeling and Simulation Frontier. PMC - PubMed Central, PMC2763179.

3. Bechtel, W., Abrahamsen, A. (2013). Synthetic Modeling and Mechanistic Account: Material Recombination and Beyond. Philosophy Compass, 8(7), 687-699.

4. Brambilla, M., Cabot, J., Wimmer, M. (2017). Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers. DOI: 10.2200/S00751ED2V01Y201701SWE004.

5. Brayton, R. K., Hachtel, G. D., McMullen, C., Sangiovanni-Vincentelli, A. (1984). M32: A Constructive Multilevel Logic Synthesis System. University of California, Berkeley.

6. Cabot, J. (2020). Clarifying concepts: MBE vs MDE vs MDD vs MDA. Modeling Languages.

7. Camps-Raga B., Islam N. E. (2010) Optimized simulation algorithms for fractal simulation and analysis. Progress In Electromagnetics Research M. Vol. 11, 2010. - P. 225 -240.

8. Chandra, M., Rani, M. (2009). Categorization of fractal plants. Chaos, Solitons & Fractals, 41(2), 906–914. DOI:10.1016/j.chaos.2008.03.004

9. Chiu W.K., Yeung, Y.C., Yu, K.M. (2006) Toolpath generation for layer manufacturing of fractal objects. Rapid Prototyping Journal. Vol. 12, № 4, 2006. P. 214 – 221.

10. Chyhir, R., Nikitina, I., Shynkarenko, V., Lytvynenko, K. (2019). Modeling of lightning flashes in thunderstorm front by constructive. In Advances in Intelligent Systems and Computing IV: Proceedings of the International Conference on Computer Science and Information Technologies (CSIT 2019), 17–20 September 2019, Lviv, Ukraine (AISC, Vol. 1080, pp. 173–185). Cham: Springer. https://doi.org/10.1007/978-3-030-33695-0_13.

11. D'Alessandro F., Ibarra O.H., McQuillan I. (2021) On finite-index indexed grammars and their restrictions, International Conference on Language and Automata Theory and Application, №279. С.287–298. doi: 10.1016/j.ic.2020.104613.
12. Dassow, J., Rozenberg, G. (2013). Grammar Systems. In: Handbook of Formal Languages. https://link.springer.com/chapter/10.1007/978-3-662-07675-0_4
13. Dube, S., Culik, K. (1990). Methods for generating deterministic fractals and image compression. Springer Lecture Notes in Computer Science, 464, 292–298. DOI:10.1007/3-540-53414-8_27
14. Evans, E. (2004). Domain-Driven Design: Tackling Complexity in the Heart of Software. Boston, MA: Addison-Wesley Professional.
15. Falconer, K. (1999). Fractal Geometry: Mathematical Foundations and Applications. John Wiley & Sons. P 288.
16. Fensel, D., Hendler, J. A., Lieberman, H., Wahlster, W. (Eds.) (2005). Spinning the Semantic Web: bringing the World Wide Web to its full potential. Mit Press.
17. Fraktal. Стаття з відкритого джерела Вікіпедія. Посилання дійсне на 05.07.2025 - https://de.wikipedia.org/wiki/Fraktal
18. Frye, L., Cheng, L., Heflin, J. (2014). TRIDSO: Traffic-based Reasoning Intrusion Detection System using Ontology. Journal of Research and Practice in Information Technology, 46(4), 215–233.
19. Godin, C., & Kurth, W. (1999). Modeling and simulation of plant architecture using L-systems. In Computational Modeling of Plants. Annals of Botany, 83(5), 231–250. https://doi.org/10.1006/anbo.1999.0989
20. Gruninger M., Fox M. (1995). Methodology for the Design and Evaluation of Ontologies. Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, 10–17.
21. Husain, D., Rani, M. (2023). Alternated L-system Fractals. In: Symposium on Mathematical Analysis of Fractals and Dynamical Systems. Springer. DOI:10.1007/978-3-031-58641-5_5

22. Huybregts, M.A.C., Beckers, G.J.L., Bolhuis, J.J. (2024). Response: Commentary: No evidence for language syntax in songbird vocalizations. Frontiers in Ecology and Evolution. https://doi.org/10.3389/fevo.2024.1474971

23. Isah, A.I., Singh, D. (2014). An Outline of the Development of the Theory of Formal Languages. IJLTC. PDF

24. Iwashokun O., Ade-Ibijola A. (2024) Parsing of Research Documents into XML Using Formal Grammars, Applied Computational Intelligence and Soft Computing. doi: 10.1155/2024/6671359.

25. Jacob, C. (1994). Genetic L-system programming. Parallel Problem Solving from Nature, Springer.

26. Khishe, M., Mosavi, M. R., Moridi, A. (2018). Chaotic Fractal Walk Trainer for Sonar Dataset Classification Using Multi-Layer Perceptron Neural Network and Its Hardware Implementation. Applied Acoustics, 138, 172–181. DOI:10.1016/j.apacoust.2018.04.018

27. Kravchenko G. (2017) Modeling the External Structure of a Fractals. IOP Conf. Series: Earth and Environmental Science, doi.10.1088/1755-1315/90/1/012100.

28. Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine, 51(11), 1016-1022. DOI: 10.1016/j.ifacol.2018.08.474.

29. Krivochen, D. (2018). Towards a classification of Lindenmayer systems. arXiv:1809.10542. PDF

30. Krivochen, D. (2021). From n-grams to trees in Lindenmayer systems. arXiv preprint arXiv:2104.01363. https://arxiv.org/abs/2104.01363

31. Krivochen, D.G. (2023). Routes to computation and chaos: Cellular automata, formal grammars, and symbolic encoding. ResearchGate PDF

32. Kuropiatnyk, O., Shynkarenko, V., Zhuchyi, L., Lyakhova, M. (2024). Geometric Fractals' Constructive-Synthesizing Models using Ontological Means. CEUR Workshop Proceedings, 3909, 419-431.

33. Lindenmayer A. (1968) Mathematical models for cellular interaction in development. Parts I and II. Journal of Theoretical Biology. V. 18, 1968. P. 280 - 315.

34. Lisovik L.P., Karnaukh T.A. (2009) A method of specification of fractal sets, Cybernetics and Systems Analysis, 2009. № 45(3). pp..365–372. doi: 10.1007/s10559-009-9117-1.

35. Lorenz, E. N. (1963). Deterministic Nonperiodic Flow. Journal of the Atmospheric Sciences, 20(2), 130–141. DOI:10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2

36. Luke, S., Spector, L., Rager, D. (1996). Ontology-Based Knowledge Discovery on the World-Wide Web. AAAI Technical Report WS-96-06, 96–102.

37. Mandelbrot, B. (1975). Objets fractals, p. 4

38. Mandelbrot, B. B. (1982). The Fractal Geometry of Nature. San Francisco, CA: W. H. Freeman.

39. Manning, C.D., Schütze, H. (1999). Foundations of Statistical Natural Language Processing. MIT Press. ISBN: 9780262133609

40. Manuja, M., Garg, D. (2015). Intelligent text classification system based on self-administered ontology. Turkish Journal of Electrical Engineering & Computer Sciences, 23, 1393–1404. DOI: 10.3906/elk-1305-112.

41. Miller, D. M., Thornton, M. A. (2006). A Constructive Algorithm for Reversible Logic Synthesis. Multiple-Valued Logic and Soft Computing, 12(3-4), 267-280.

42. Müller S. (2023) Grammatical theory: From transformational grammar to constraint-based approaches, Language Science Press. doi: 10.5281/zenodo.3992307.

43. Nersessian, N. J. (1999). The Role of Generic Models in Conceptual Change. Model-based reasoning in scientific discovery, 129-153.

44. Nikiel, S. (2007). Iterated Function Systems for Real-Time Image Synthesis. Springer. DOI:10.1007/1-84628-686-7_2

45. Palagin, A. V. (2016). Ontologicheskaya kontseptsiya informatizatsii nauchnyh issledovaniy. Kibernetika i sistemnyy analiz, 52(1), 3–9.

46. Palubicki, W., Horel, K., Prusinkiewicz, P. (2009). Modeling tissues and organs with L-systems. ACM Transactions on Graphics, 28(3), 1–10. DOI: 10.1145/1531326.1531364

47. Parish, Y.I.H., Müller, P. (2001). Procedural modeling of cities. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 301–308. DOI: 10.1145/383259.383292

48. Peitgen, H.-O., Jürgens, H., & Saupe, D. (2004). Chaos and Fractals: New Frontiers of Science (2nd ed.), 864. New York: Springer.

49. Prusinkiewicz P., Lindenmayer A. (2004) The algorithmic beauty of plants, Springer Science & Business Media. pp.228. doi: 10.1007/978-1-4613-8476-2.

50. Pu X., Kay M. (2020) A probabilistic grammar of graphics. CHI Conference on Human Factors in Computing Systems. pp.1–13. doi: 10.1109/TCBB.2023.3247144.

51. Qin, H., Terzopoulos, D. (1996). Constructive Sculpting of Heterogeneous Volumetric Objects Using Trivariate B-splines. Computer Graphics and Interactive Techniques, 25, 145-154. DOI: 10.1145/237170.237225.

52. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140), 1-67.

53. Rozenberg, G., Salomaa, A. (1980). The Mathematical Theory of L Systems. Springer.

54. Rozenberg, G., Salomaa, A. (1997). Handbook of Formal Languages. Springer. DOI: 10.1007/978-3-662-03339-5

55. Ruohonen, K. (2009). Formal Languages. CiteseerX. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=592e35edb677dbc00b169a78b0d5f77de1bf5c2d

56. Salcedo-Sanz, S., Aybar-Ruíz, A. (2018). Efficient fractal-based mutation in evolutionary algorithms from iterated function systems. Communications in Nonlinear Science and Numerical Simulation, 57, 147–158. DOI:10.1016/j.cnsns.2017.08.003

57. Schmidt, D. C. (2006). Guest Editor's Introduction: Model-Driven Engineering. IEEE Computer, 39(2), 25-31. DOI: 10.1109/MC.2006.58.

58. Searls, D. B. (2002). The language of genes. Nature, 420(6912), 211–217. DOI: 10.1038/nature01254

59. Selic, B. (2003). The pragmatics of model-driven development. IEEE Software, 20(5), 19-25.

60. Shinkarenko, V. I., Zhevago, O. O. (2019). Составление расписания занятий университета на основе конструктивного моделирования. Radio Electronics, Computer Science, Control, (3), pp. 152-162.

61. Shymanskyi, V., Sokolovskyy, Y. (2021). Finite element calculation of the linear elasticity problem for biomaterials with fractal structure. The Open Bioinformatics Journal, 14(1).

62. Shynkarenko V. I., (2019) Constructive-Synthesizing Representation of Geometric Fractals, Cybernetics and Systems Analysis, 2019. № 55. pp.186-199. doi: 10.1007/s10559-019-00123-w

63. Shynkarenko V. I., Ilman V. M. (2009). Structural Models of Algorithms in Problems of Applied Programming. I. Formal Algorithmic Structures. Cybernetics and Systems Analysis, 45(3), 329-339. DOI: 10.1007/s10559-009-9118-0.

64. Shynkarenko V. I., Ilman V. M., Skalozub V. V. (2009). Structural models of algorithms in problems of applied programming. II. Structural-algorithmic approach to software simulation. Cybernetics and Systems Analysis, 45(4), 544–550. DOI: 10.1007/s10559-009-9122-4.

65. Shynkarenko V. I., Letuchyi O. Y., Chyhir R.R. (2023). Constructive-synthesizing modeling of fractal crystal lattices. Proceedings of the 18th IEEE International Conference on Computer Science and Information Technologies (CSIT). DOI: 10.1109/CSIT61576.2023.10324251.

66. Shynkarenko V.I., Ilman V.M. (2014) Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure. Cybernetics and Systems Analysis, Vol. 50, No 5. Springer, 2014. P. 665 – 662. Part II. Refining Transformations. –Vol. 50, No 6, 2014. P. 829 – 841. ISSN: 1060-0396 (Print) 1573-8337 (Online), doi: 10.1007/s10559-014-9655-z, https://link.springer.com/article/10.1007/s10559-014-9655-z, doi: 10.1007/s10559-014-9674-9, https://link.springer.com/article/10.1007/s10559-014-9674-9

67. Shynkarenko V.I., VasetskaT.M. (2015) Modeling the Adaptation of Compression Algorithms by Means of Constructive-Synthesizing Structures. Cybernetics and Systems Analysis, Vol. 51, No 6. Springer, 2015. P. 849-861. doi: 10.1007/s10559-015-9778-x https://link.springer.com/article/10.1007/s10559-015-9778-x

68. Shynkarenko, V. I., Lytvynenko, K. V., Chyhir, R. R. (2019). Конструктивное соответствие мультисимвольных и линейных геометрических фракталов. Information Technologies & Knowledge, 13(1), 76–99.

69. Shynkarenko, V., Chyhir, R. (2024). Constructive-synthesising modelling of multifractals based on multiconstructors. In Proceedings of the 14th International Scientific and Practical Programming Conference (UkrPROG 2024) (CEUR-WS Proceedings, Vol. 3806, pp. 75–88). Aachen: CEUR-WS.org.

70. Shynkarenko, V., Lytvynenko, K., Chyhir, R., Sansiieva, I. (2019). Constructive modeling of lightning activity in thunderstorm front. In 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 92–95). IEEE. https://doi.org/10.1109/STC-CSIT.2019.8929754

71. Shynkarenko, V., Makarov, D. (2024). Structural Adaptation of Sorting Algorithms Based on Constructive Fragments. CEUR Workshop Proceedings, 3806, 16-29.

72. Shynkarenko, V., Nikitina, I., Chyhir, R. (2020). Constructive-synthesising modelling of lightning flashes in the dynamic thunderstorm front. In Advances in Intelligent Systems and Computing V: CSIT 2020 (AISC, Vol. 1293, pp. 1128–1145). Cham: Springer. https://doi.org/10.1007/978-3-030-63270-0_76.

73. Shynkarenko, V., Zhevago, O. (2025). A Taxonomy of Software Debugging Process. In Proceedings of the 15th International Scientific and Practical Programming Conference (UkrPROG 2025)

74. Shynkarenko, V., Zhevaho, O. (2021). Application of Constructive Modeling and Process Mining Approaches to the Study of Source Code Development in Software Engineering Courses. Journal of Communications Software and Systems, 17(4), 342-349. DOI: 10.24138/jcomss-2021-0046.

75. Skalozub, V. V., Ilman, V. M. (2020). Недостатність методологічної бази та інформаційно-технологічного забезпечення спеціалізованих процесів конструктивно-продукційного моделювання. Science and Transport Progress.

76. Smith, A.R., Bloomenthal, J. (1985). Visualization of fractals generated by L-systems. ACM SIGGRAPH Computer Graphics, 19(3), 163–170. DOI: 10.1145/325165.325214

77. Sokolowskyi, Ya., Shymanskyi, V. (2014). Mathematical modelling of non-isothermal moisture transfer and rheological behavior in cappilary-porous materials with fractal structure during drying. Computer and Information Science. Canadian Center of Science and Education, Vol. 7, No. 4 , pp. 111-122.

78. Sprott, J. C. (1993). Strange Attractors: Creating Patterns in Chaos. M&T Books.

79. Vernon, V. (2013). Implementing Domain-Driven Design. Boston, MA: Addison-Wesley Professional.

80. Vernon, V. (2016). Domain-Driven Design Distilled. Boston, MA: Addison-Wesley Professional.

81. Vogt, W. C., Jia, C., Wear, K. A., Garra, B. S., Pfefer, T. J. (2021). Modeling and synthesis of breast cancer optical property signatures with generative models. Scientific Reports, 11, 11376.

82. Vysotska, V., Burov, Y., Chyrun, L., Chyrun, S., Brodyak, O., Panasyuk, V., Karpyn, D., Pohreliuk, L., Shykh, N. (2024). Using Knowledge Presentation Models to Solve the Problems of Managing Complex Ontologies. CEUR Workshop Proceedings, 3722.

83. Wandr R., Ferreira A.L.S., Pessoa R.V.S., Galv A., Machado-Lima A. (2023) A stochastic grammar approach to mass classification in mammograms, IEEE/ACM Transactions on Computational Biology and Bioinformatics.

84. Zhou J., Vas A., Blackmore D. (2010) Fractal geometry surface modeling and measurement for musical cymbal surface texture design and rapid manufacturing. – Режим доступа: https://www.researchgate.net/publication/250330003.

85. Zhou J.; Leu M.; Blackmore D.: (1995) Fractal Geometry Modeling with Applications in Surface Characterization and Wear Prediction. / International Journal of Machine Tools & Manufacture. Vol. 35, No. 2, 1995. P. 203-209.

86. Безносюк С.А., Лерх Я.В., Жуковская Т.М. (2002) Компьютерное моделирование самоорганизации фрактальных кластерных нанодендритов. Ползуновский вестник. С. 160 - 166.

87. Божокин, С. В., Паршин, Д. А. (2001). Фракталы и мультифракталы, 128. М–Ижевск: НИЦ «Регулярная и хаотическая динамика».

88. Гуда, А. І., Іванов, О. П., Шинкаренко, В. І., Саблін, О. І. (2025). Атрибутивне насичення конструктивно-продукційної моделі ділянки системи електропостачання тяги постійного струму. Системні технології, 3(158), 96-107. DOI: 10.34185/1562-9945-3-158-2025-10.

89. Іванов, О. П., Гуда, А. І., Саблін, О. І., Шинкаренко, В. І. (2025). Конструктивно-продукційне моделювання розподілу енергії рекуперації на основі нечіткої логіки. Системні технології, 4(159), 200-211. DOI: 10.34185/1562-9945-4-159-2025-20.

90. Ільман В. М., Скалозуб, В. В., Шинкаренко, В. І. (2009). Формальні структури та їх застосування: монографія. Дніпро: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна.

91. Кириченко, Л. О., Чалая, Л. Э. (2014). Комплексный подход к исследованию фрактальных временных рядов. International Journal “INFORMATION TECHNOLOGIES & KNOWLEDGE, 8(1), 22-28.

92. Кравченко Г.М., Васильев С.Э., Пуданова Л.И. (2016) Моделирование фракталов. Инженерный вестник Дона. №4, 2016. - Режим доступа: https://www ivdon.ru/ru/magazine/archive/n4y2016/3930.

93. Кравченко, Г. М., Васильев, С. Э., Пуданова, Л. И. (2023). Графічна культура конструктивного моделювання фігур у метричній стереометрії. ResearchGate.

94. Кроновер Р. М. (2000) Фракталы и хаос в динамических системах. Основы теории, 352. М.: Постмаркет.

95. Лубко, Д. В., Шаров, С. В. (2019). Методи та системи штучного інтелекту. Навчальний посібник. Мелітополь: ФОП Однорог Т.В.

96. Помулев В.В., Михалев А. И., Бондаренко Я. С., Деревянко А. И. (2002) Моделирование и фрактальная параметризация дендритов нейронов. Адаптивные системы автоматического управления, №5(25), 2002. С. 160 - 166. ISSN 1560-8956

97. Рассел, С., Норвіг, П. (2021). Штучний інтелект: сучасний підхід (4-е вид.). Pearson.

98. Скалозуб, В., Ільман, В., Шинкаренко, В. (2017). Онтологічна підтримка формування для конструктивно-продукційного моделювання інформаційних систем. Eastern-European Journal of Enterprise Technologies, 6(4(90)), 58-69. DOI: 10.15587/1729-4061.2017.119497.

99. Скалозуб, В., Ільман, В., Шинкаренко, В. (2018). Формування онтологічної підтримки для конструктивно-продукційного моделювання процесів розробки інформаційних систем. Eastern-European Journal of Enterprise Technologies, 5(4(94)), 55-63. DOI: 10.15587/1729-4061.2018.143968.

100. Слюсар В. (2007) Фрактальные антены – принципиально новый тип ломаных антенн. Электроника: Наука, Технология, Бизнес. № 5, 2007. С.78 - 83. - Режим доступа: https://www.elektronics.ru/files/article_pdf/0/article_611_312.pdf.

101. Соколовський, Я. І., Шиманський, В. М. (2011). Фрактальна модель тепло- і масоперенесення у капілярно-пористих матеріалах. Вісник Національного університету «Львівська політехніка», Комп’ютерні науки та інформаційні технології. № 694. Львів : НУ «ЛП», с. 424-428.

102. Соколовський, Я. І., Шиманський, В. М. (2011). Чисельне моделювання неізотермічного вологоперенесення у середовищах з фрактальною структурою. Вісник Національного університету «Львівська політехніка» : Комп’ютерні науки та інформаційні технології., № 710., С. 159-164.

103. Соколовський, Я. І., Шиманський, В. М. (2012). Математична модель тепловологоперенесення та напружено-деформівного стану у капілярно-пористих матеріалах із фрактальною структурою. Фізико-математичне моделювання та інформаційні технології. Львів : Центр математичного моделювання Інституту прикла-

дних проблем механіки і математики ім. Я. С. Підстригача НАН України, Вип. 16., С. 133-142.

104. Федер, Е. (1991). Фракталы. М: Мир. 262 с.

105. Шаховська, Н. Б., Камінський, Р. М., Вовк, О. Б. (2018). Системи штучного інтелекту: навчальний посібник. Львів: Видавництво Львівської політехніки.

106. Шелухин О.И. (2011) Мультфракталы. Инфокоммуникационные приложения. М. Горячая линия – Телеком. 576 с.

107. Шелухин О.И., Осин А.В., Смольский С.М. (2008) Самоподобие и фракталы. Телекомуникационные приложения. М. Физматлит. 365 с.

108. Шинкаренко В. И., Васецкая Т. Н. (2016) Моделирование процесса ранжирования альтернатив методом анализа иерархий средствами конструктивно-продукционных структур. Математические машины и системы. № 1, 2016. С. 39-47. ISSN 1028-9763

109. Шинкаренко В. И., Забула Г. В. (2016) Конструктивная модель адаптации структур данных в оперативной памяти. ЧАСТЬ I. Конструирование текстов программ. Наука и прогресс транспорта. № 1 (61), 2016. С. 109-121.; ЧАСТЬ II. Конструкторы сценариев и процессов адаптации. № 2 (62), 2016. С. 88-97. ISSN 2307-6666

110. Шинкаренко В. И., Куропятник Е. С. (2016) Конструктивно-продукционная модель графового представления текста. Проблемы программирования. № 2-3, 2016. С. 63-72. ISSN 1727- 4907, http://dspace.nbuv.gov.ua/bitstream/handle/123456789/126391/07-Shinkarenko.pdf?sequence=1

111. Шинкаренко, В. (2021). Ідентифікація відповідності програмного тексту та алгоритму на основі конструктивно-продукційної моделі графа керування. Наука та прогрес транспорту.

112. Шинкаренко, В. (2024). Дуальний підхід до встановлення авторитетності технічних текстів природної мови. Український державний університет науки і технологій.

113. Шинкаренко, В. И. (2009). Экспериментальные исследования алгоритмов в программно-аппаратных средах: монография. Д.: Изд-во Днепропетр. нац. ун-та ж.-д. трансп. им. акад. В. Лазаряна.

114. Шинкаренко, В. И., Литвиненко, К. В., Чигирь, Р. Р. (2019). Восстановление фрактальных временных рядов. В «Матеріали міжнародної науково-технічної конференції «Інформаційні технології в металургії та машинобудуванні»» (с. 83). ІВК «Системні технології».

115. Шинкаренко, В. И., Литвиненко, К. В., Чигирь, Р. Р., Жадан, А. А. (2018). Констуктивно-продукционное моделирование фрактов. В «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту». Матеріали міжнародної наукової конференції (с. 289–291). Херсон: ПП Вишемирський В.С.

116. Шинкаренко, В. И., Литвиненко, К. В., Чигирь, Р. Р., Жадан, А. А. (2018). Констуктивно-продукционное моделирование фрактальных временных рядов на основе L-систем. В Проблеми математичного моделювання. Матеріали Всеукраїнської науково-методичної конференції (с. 161–163). Кам'янське: ДДТУ.

117. Шинкаренко, В. І. (2019). Конструктивно-продукційне представлення геометричних фракталів. Кібернетика та системний аналіз, 55(2), 186-199. DOI: 10.1007/s10559-019-00123-w.

118. Шинкаренко, В. І., Гуда, А. І., Саблін, О. І., Іванов, О. П. (2024). Конструктивно-продукційне моделювання системи електропостачання тяги постійного струму. Системні технології, 6(155), 145-158. DOI: 10.34185/1562-9945-6-155-2024-14.

119. Шинкаренко, В. І., Ільман, В. М. (2014). Програмний засіб конструктивно-продукційного моделювання. Проблеми програмування, (1), 3-17.

120. Шинкаренко, В. І., Литвиненко, К. В., Чигір, Р. Р. (2018). Констуктивно – продукційне моделювання стохастичних процесів. В Тези доповідей XII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (с. 118). Дніпропетровськ, ДНУЗТ.

121. Шинкаренко, В. І., Литвиненко, К. В., Чигір, Р. Р., Жадан, А. А. (2017). Вариативность уточняющих преобразований конструктивно-продукционного модели-

рования. В «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD’2017) (с. 225–230). Київ.

122. Шинкаренко, В. І., Саблін, О. І., Іванов, О. П. (2016). Конструктивна модель зони рекуперації в системі тягового електропостачання постійного струму. Наука та прогрес транспорту, (5(65)), 99-111. DOI: 10.15802/stp2016/84036.

123. Шинкаренко, В. І., Чигір, Р. Р. (2020). Описання продукційних конструкторів, що породжують фрактальні зображення. В Всеукраїнська конференція студентів та молодих вчених «Інформаційно-управляючі технології і системи на залізничному транспорті» (с. 33). Дніпро.

124. Шинкаренко, В. І., Чигір, Р. Р. (2023). Використання спрощеної форми проєкту предметно-орієнтованого програмного забезпечення. В Тези доповідей XVII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (с. 99). Дніпро, УДУНТ.

125. Шинкаренко, В. І., Чигір, Р. Р. (2024). Інструментальні засоби конструктивно-продукційного моделювання. Проблеми програмування, (2–3), 107–115. https://doi.org/10.15407/pp2024.02-03.107

126. Шинкаренко, В. І., Чигір, Р. Р. (2025). Конструктивно-продукційне моделювання тривимірних фрактальних поверхонь. Системні технології, 1(156), 78–88. https://doi.org/10.34185/1562-9945-1-156-2025-09

127. Шинкаренко, В. І., Чигір, Р. Р. (2025). Конструктивно-продукційне моделювання грозового фронту з використанням генетичного алгоритму. Системні технології, 4(159), 189–199. https://doi.org/10.34185/1562-9945-4-159-2025-19.

128. Шинкаренко, В. І., Чигір, Р. Р., Жадан, А. А. (2017). Комп’ютерна програма «Моделювання часових рядів із заданими фрактальними властивостями». Свідоцтво про реєстрацію авторського права на твір № 72579 від 27.06.2017.

129. Шинкаренко, В. І., Чигір, Р. Р., Жадан, А. А. (2019). Комп’ютерна програма «Моделювання взаємопов’язаних часових рядів й геометричних фракталів породжуваних в L-системах». Рішення про реєстрацію договору, який стосується права автора на твір № 4369 від 31.05.2019.

130. Шинкаренко, В. І., Чигір, Р. Р., Жадан, А. А. (2019). Комп’ютерна програма «Рекурентний аналіз часових рядів породжуваних в L-системах». Рішення про реєстрацію договору, який стосується права автора на твір № 4367 від 29.05.2019.

131. Шинкаренко, В., Демидович, І. (2023). Конструктивно-продукційне моделювання текстів природної мови. Комп'ютерні системи та інформаційні технології, 19(3), 81-89. DOI: 10.31891/csit-2023-3-10.

132. Шинкаренко, В., Жадан, А. (2024). Мультиагентна система для реконструкції конструктивних моделей стохастичних фрактальних часових рядів. CEUR Workshop Proceedings, 3955.

133. Шинкаренко, В., Литвиненко, К., Чигир, Р., Нікітіна, І. (2020). Конструктивно-продукційне моделювання блискавок у грозовому фронті за допомогою конструктивного продукування фрактальних часових рядів. Український державний університет науки і технологій.

134. Шинкаренко, В., Литвиненко, К., Чигирь, Р., Сансиева, І. (2019). Constructive modeling of lightning activity in thunderstorm front. In 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 92–95). IEEE. https://doi.org/10.1109/STC-CSIT.2019.8929754.

135. Шинкаренко, В.І, Демидович, І.М. (2020). Використання генетичного алгоритму для покращення визначення авторства природньомовних текстів. Збірка тез XIV Міжнародної науково-практичної конференції “Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті”, 127.

Додаток А Список публікацій здобувача та відомості про апробацію результатів дисертації

Праці у фахових виданнях МОН України

1. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2] (2024). Інструментальні засоби конструктивно-продукційного моделювання. Проблеми програмування, (2–3), 107–115. https://doi.org/10.15407/pp2024.02-03.107 ([1] – керування дослідженням, обговорення результатів , [2] – розробка моделей, методів та програмного додатку моделювання конструкторів, проведення досліджень)
2. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2] (2025). Конструктивно-продукційне моделювання тривимірних фрактальних поверхонь. Системні технології, 1(156), 78–88. https://doi.org/10.34185/1562-9945-1-156-2025-09 ([1] – керування дослідженням, обговорення результатів , [2] – розробка програмного забезпечення та конструктивно-продукційних моделей фрактальних структур, проведення досліджень)
3. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2] (2025). Конструктивно-продукційне моделювання грозового фронту з використанням генетичного алгоритму. Системні технології, 4(159), 189–199. https://doi.org/10.34185/1562-9945-4-159-2025-19 ([1] – керування дослідженням, обговорення результатів , [2] – проведення досліджень, розробку програмного забезпечення, методів та моделей представлення блискавок та грозового фронту)

Праці міжнародних конференцій, включені до міжнародної наукометричної бази Scopus

4. Shynkarenko, V.[1], Lytvynenko, K.[2], ***Chyhir, R.***[3], Nikitina, I.[4] (2019, 17-20 вересня). Modeling of lightning flashes in thunderstorm front by constructive production of fractal time series. In Advances in Intelligent Systems and Computing IV: CSIT 2019 (AISC, Vol. 1080, pp. 173–185). Springer, Cham. https://doi.org/10.1007/978-3-030-33695-0_13. Форма участі – очна, усна доповідь. ([1] – керування дослідженням, обговорення результатів, [2] – підготовка матеріалов до формату документа, [3] – проведення досліджень, розробка програмного забезпечення, методів та моделей пред-

ставлення блискавок та грозового фронту, [4] – розробка методів та обробка первинних даних відеофайлів)

5. Shynkarenko, V.[1], Lytvynenko, K.[2], ***Chyhir, R.***[3], Sansiieva, I.[4] (2019, 17-20 вересня). Constructive modeling of lightning activity in thunderstorm front. In 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 92–95). IEEE. https://doi.org/10.1109/STC-CSIT.2019.8929754. Форма участі – очна, усна доповідь. ([1] – керування дослідженням, обговорення результатів, [2] – підготовка матеріалов до формату документа, [3] – проведення досліджень, розробка програмного забезпечення, методів та моделей представлення блискавок та грозового фронту, [4] – розробка методів та обробка первинних даних відеофайлів)

6. Shynkarenko, V.[1], Letuchyi, O.[2], ***Chyhir, R.***[3] (2023, 22-24 листопада). Constructive-synthesizing modeling of fractal crystal lattices. In 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT) (pp. 1–4). IEEE. https://doi.org/10.1109/CSIT61576.2023.10324251. Форма участі – дистанційна, усна доповідь. ([1] – керування дослідженням, обговорення результатів, [2] – проведення досліджень, розробка програмного забезпечення, підготовка рукопису статті, [3] – розробка моделей кристалічних граток)

7. Shynkarenko, V.[1], ***Chyhir, R.***[2] (2024, 21-23 травня). Constructive-synthesising modelling of multifractals based on multiconstructors. In Proceedings of the 14th International Scientific and Practical Programming Conference (UkrPROG 2024) (CEUR Workshop Proceedings, Vol. 3806, pp. 75–88). Форма участі – дистанційна, усна доповідь. ([1] – керування дослідженням, обговорення результатів, [2] – розробка програмного забезпечення та конструктивно-продукційних моделей фрактальних структур, проведення досліджень)

Праця у міжнародному виданні

8. Шинкаренко, В. И.[1], Литвиненко, К. В.[2], ***Чигирь, Р. Р.***[3] (2019). Конструктивное соответствие мультисимвольных и линейных геометрических фракталов. Information Technologies & Knowledge, 13(1), 76–99. ([1] – теоретичні положення, [2] – підготовка матеріалов до друку, обговорення результатів, [3] – розробка програмно-

го забезпечення та моделей класичних лінійних фракталів з бієктивним відображенням, проведення досліджень)

Матеріали міжнародних наукових конференцій

9. Шинкаренко, В. И.[1], Литвиненко, К. В.[2], ***Чигирь, Р. Р.***[3], Жадан, А. А.[4] (2018, 21-27 травня). Констуктивно-продукционное моделирование фракталов. В «Інтелектуальні системи прийняття рішень і проблеми обчислювального інтелекту». Матеріали міжнародної наукової конференції (с. 289–291). Херсон: ПП Вишемирський В.С. Форма участі – очна, усна доповідь. ([1] – керування дослідженням, обговорення результатів, [2] – підготовка матеріалов до друку, обговорення результатів, [3] – розробка програмного забезпечення та конструктивно-продукційних моделей фрактальних структур, проведення досліджень, [4] – розробка частини програми, обговорення результатів)

10. Шинкаренко, В. И.[1], Литвиненко, К. В.[2], ***Чигирь, Р. Р.***[3] (2019, 28-30 травня). Восстановление фрактальных временных рядов. В «Матеріали міжнародної науково-технічної конференції «Інформаційні технології в металургії та машинобудуванні»» (с. 83). ІВК «Системні технології». Форма участі – очна, усна доповідь. ([1] – теоретичні положення, [2] – підготовка матеріалов до друку, обговорення результатів, [3] – розробка програмного забезпечення та конструктивно-продукційних моделей фрактальних структур, проведення досліджень)

11. Shynkarenko, V.[1], Nikitina, I.[2], ***Chyhir, R.***[3] (2020, 23-26 вересня). Constructive-synthesizing modeling of lightning flashes in the dynamic thunderstorm front. In «Advances in Intelligent Systems and Computing V»: CSIT 2020 (AISC, Vol. 1293, pp. 1128–1145). Springer, Cham. https://doi.org/10.1007/978-3-030-63270-0_76. Форма участі – очна, усна доповідь. ([1] – керування дослідженням, обговорення результатів, [2] – розробка моделей та обробка первинних даних відеофайлів, [3] – проведення досліджень, розробку програмного забезпечення, методів та моделей представлення блискавок та грозового фронту)

12. Шинкаренко, В. І.[1], Литвиненко, К. В.[2], ***Чигір, Р. Р.***[3], Жадан, А. А.[4] (2017, 04-08 грудня). Вариативность уточняющих преобразований конструктивно-продукционного моделирования. В «Теоретичні та прикладні аспекти побудови програмних систем» (TAAPSD’2017) (с. 225–230). Київ. Форма участі – очна, усна доповідь. ([1] – теоретичні положення, [2] – підготовка матеріалов до друку, обгово-

рення результатів, [3] – розробка програмного забезпечення, проведення досліджень з лінійних геометричних фракталів, [4] – проведення дослідження з фрактальних часових рядів)

13.Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2] (2023, 29 листопада – 1 грудня). Використання спрощеної форми проєкту предметно-орієнтованого програмного забезпечення. В Тези доповідей XVII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (с. 99). Дніпро, УДУНТ. Форма участі – дистанційна, усна доповідь. ([1] – оцінював та аналізував результати, [2] – розробка методів організації програмного коду)

14. Шинкаренко, В. І.[1], Литвиненко, К. В.[2], ***Чигір, Р. Р.***[3] (2018, 12-13 грудня). Констуктивно – продукційне моделювання стохастичних процесів. В Тези доповідей XII Міжнародної науково-практичної конференції «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті» (с. 118). Дніпропетровськ, ДНУЗТ. Форма участі – очна, усна доповідь. ([1] – теоретичні положення, [2] – підготовка матеріалов до друку, обговорення результатів, [3] – розробка програмного забезпечення та моделей класичних лінійних фракталів з бієктивним відображенням, проведення досліджень)

Матеріали наукових конференцій

15. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2] (2020, 23-27 березня). Описання продукційних конструкторів, що породжують фрактальні зображення. В Всеукраїнська конференція студентів та молодих вчених «Інформаційно-управляючі технології і системи на залізничному транспорті» (с. 33). Дніпро. ([1] – керування дослідженням, обговорення результатів, [2] – розробка програмного забезпечення та конструктивно-продукційних моделей фрактальних структур, проведення досліджень)

16. Шинкаренко, В. И.[1], Литвиненко, К. В.[2], ***Чигирь, Р. Р.***[3], Жадан, А. А.[4] (2018, 23-25 травня). Констуктивно-продукционное моделирование фрактальных временных рядов на основе L-систем. В Проблеми математичного моделювання. Матеріали Всеукраїнської науково-методичної конференції (с. 161–163). Кам'янське: ДДТУ. Форма доповіді – очна доповідь. ([1] – керівництво роботою, [2] – підготовка матеріалов до друку, обговорення результатів, [3] – розробка програмного забезпечення для моделювання часових рядів, [4] – планування та проведення дослідження)

Свідоцтва про реєстрацію авторського права

17. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2], Жадан, А. А.[3] (2019). Комп'ютерна програма «Рекурентний аналіз часових рядів породжуваних в L-системах». Рішення про реєстрацію договору, який стосується права автора на твір № 4367 від 29.05.2019. ([1] – керівництво розробкою, [2] – розробка частини програми, тестування, [3] – розробка частини програми)

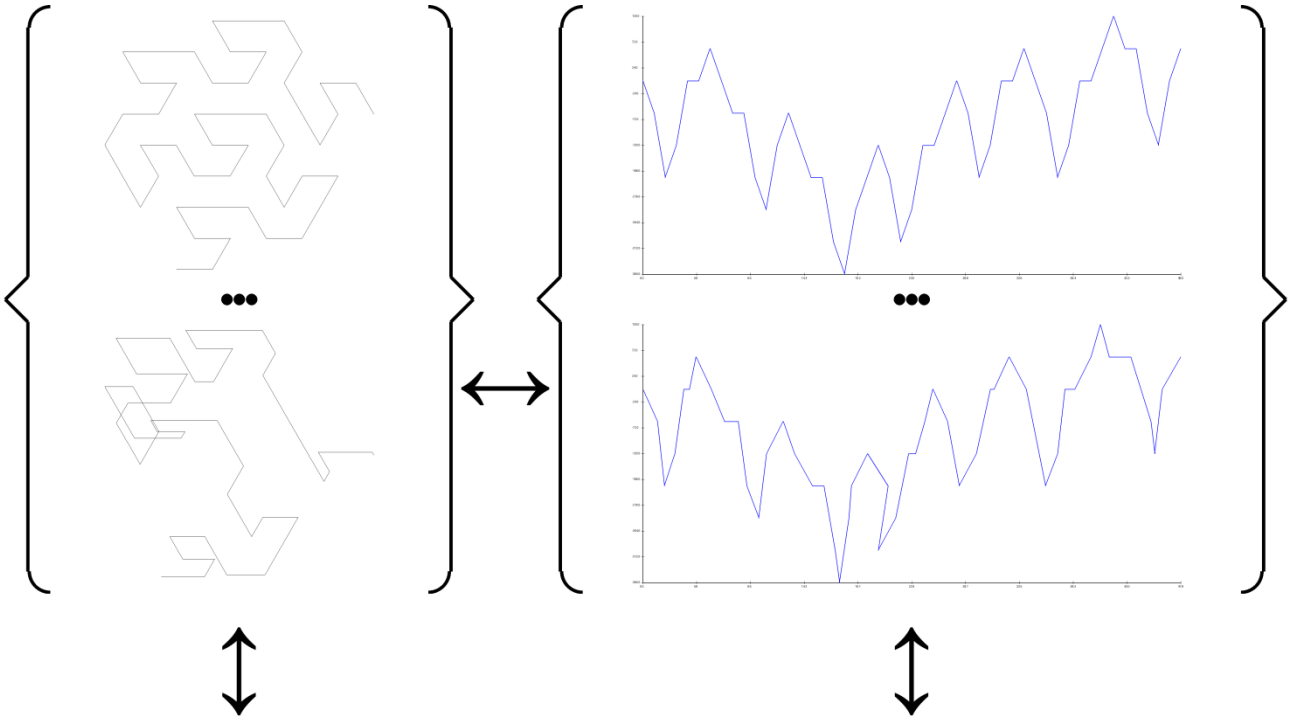
18. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2], Жадан, А. А.[3] (2019). Комп'ютерна програма «Моделювання взаємопов'язаних часових рядів й геометричних фракталів породжуваних в L-системах». Рішення про реєстрацію договору, який стосується права автора на твір № 4369 від 31.05.2019. ([1] – керівництво розробкою, [2] – розробка частини програми, [3] – розробка частини програми, тестування)

19. Шинкаренко, В. І.[1], ***Чигір, Р. Р.***[2], Жадан, А. А.[3] (2017). Комп'ютерна програма «Моделювання часових рядів із заданими фрактальними властивостями». Свідоцтво про реєстрацію авторського права на твір № 72579 від 27.06.2017. ([1] – керівництво розробкою, [2] – розробка частини програми, тестування, [3] – розробка частини програми)

Додатково праця у міжнародній конференції, що включена до міжнародної наукометричної бази Scopus

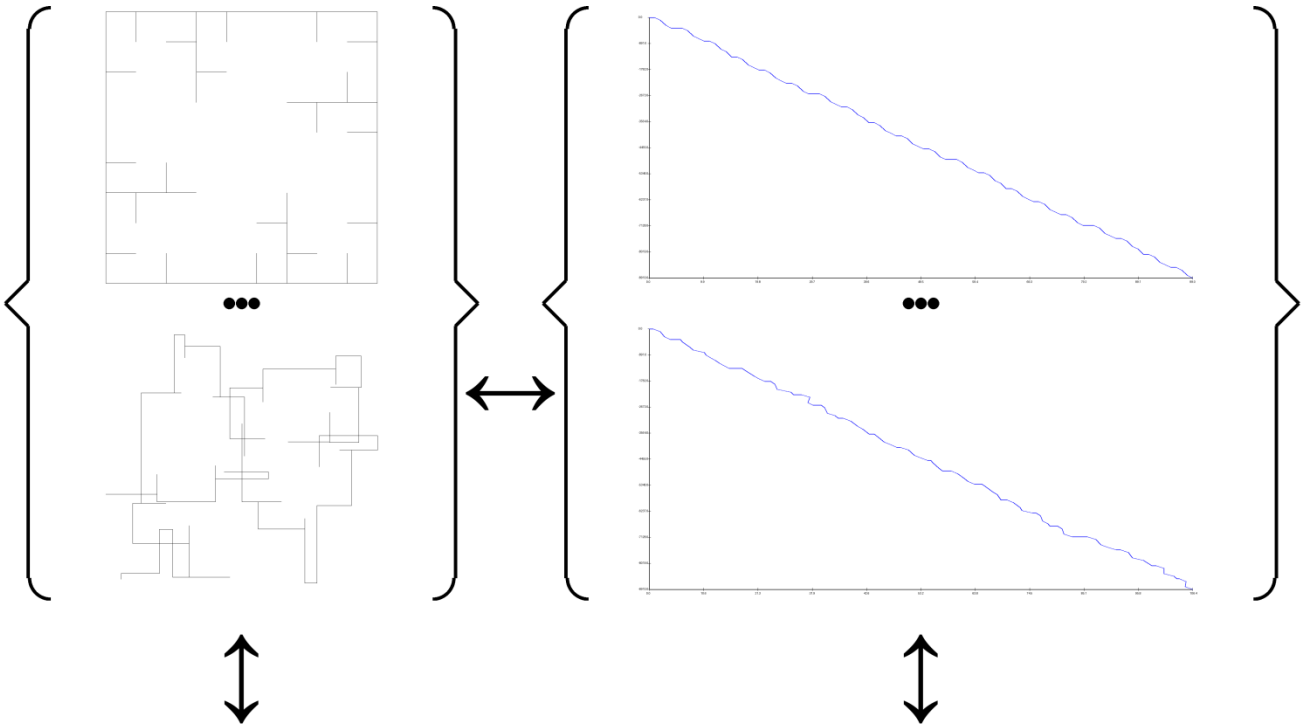
20. Shynkarenko, V.[1], Raznosilin, V.[2], Snihur, Y.[3], ***Chyhir, R.***[4] (2022). Experimental research of educational content tracking by students group for distance learning. In V. Ermolayev et al. (Eds.), Information and Communication Technologies in Education, Research, and Industrial Applications: ICTERI 2021, Revised Selected Papers (CCIS, Vol. 1698, pp. 229–251). Springer, Cham. https://doi.org/10.1007/978-3-031-20834-8_11 ([1] – теоретичні положення, [2] – підготовка матеріалов до друку, обговорення результатів, проведення досліджень, [3] – розробка частини програмного забезпечення, [4] – розробка частини програмного забезпечення, проведення досліджень)

Додаток Б Бієктивні відображення класичних фракталів



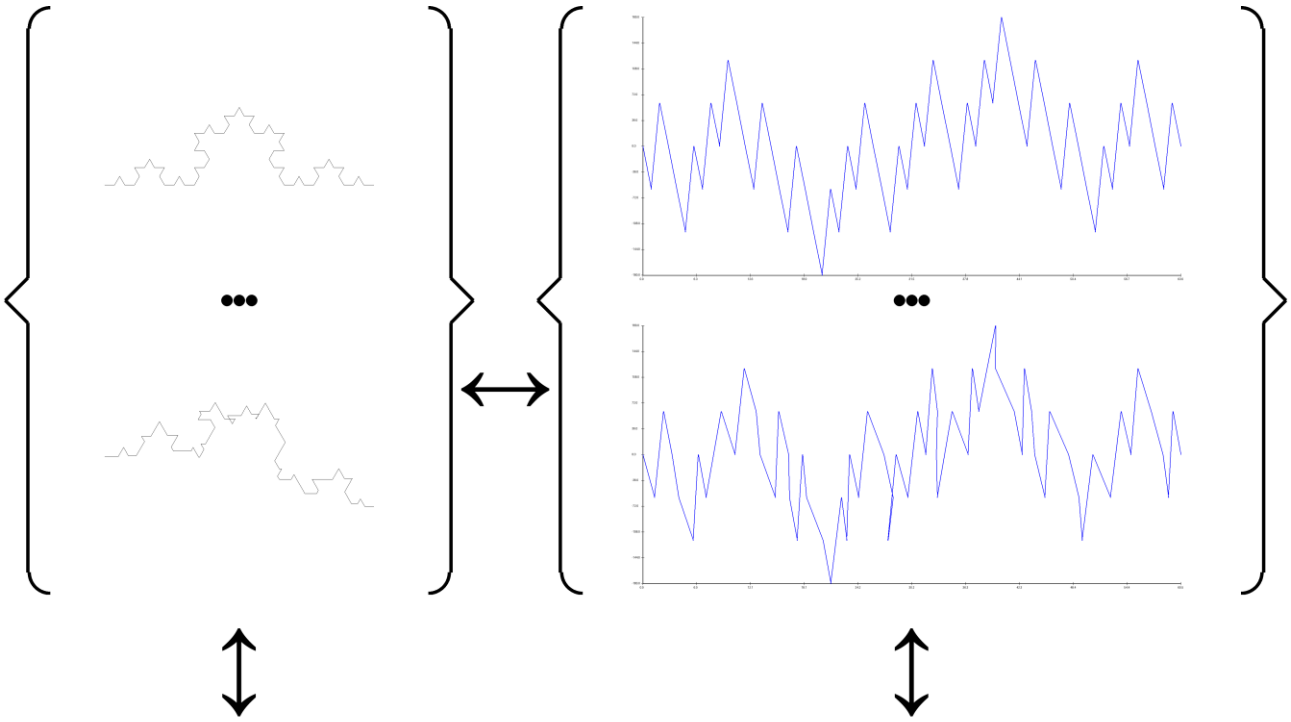
A+B++B-A--AA-B++-A+BB++B+A--A-B++-A+BB++B+A--A-B-A+B++B-A--AA-B+--A+B++B-A--
AA-B+A+B++B-A--AA-B+--A+BB++B+A--A-B+

Рисунок Б.1 – Бієктивні відображення фракталу "Крива Госпера"



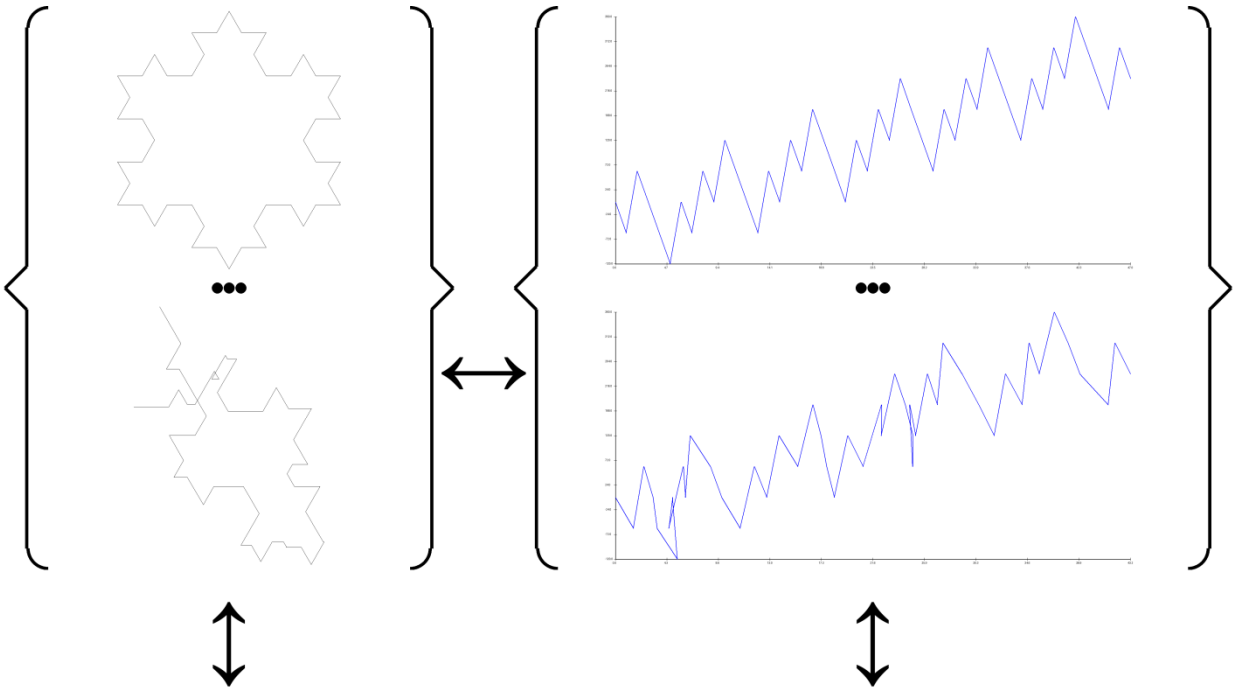
FF+F++F+FFF+F++F+F+FF+F++F+F++FF+F++F+F+FF+F++F+F+FF+F++F+FFF+F++F+F+FF+F++F
+F++FF+F++F+F+FF+F++F+F+FF+F++F+FFF+F++F+F+FF+F++F+F++FF+F++F+F+FF+F++F+F+FF
+F++F+FFF+F++F+F+FF+F++F+F++FF+F++F+F+FF+F++F

Рисунок Б.2 – Бієктивні відображення фракталу "Льодовий фрактал"



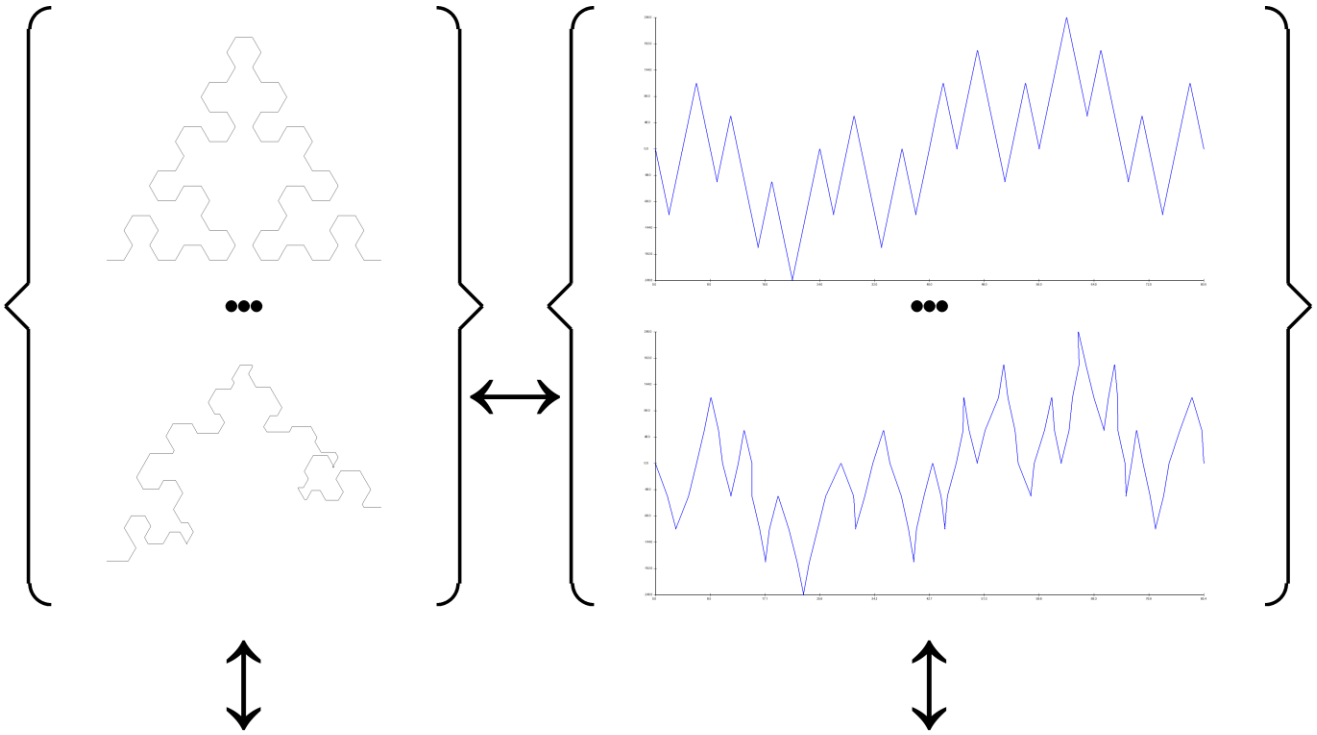
F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F+F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F--F
+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F+F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

Рисунок Б.3 – Бієктивні відображення фракталу "Крива Коха"



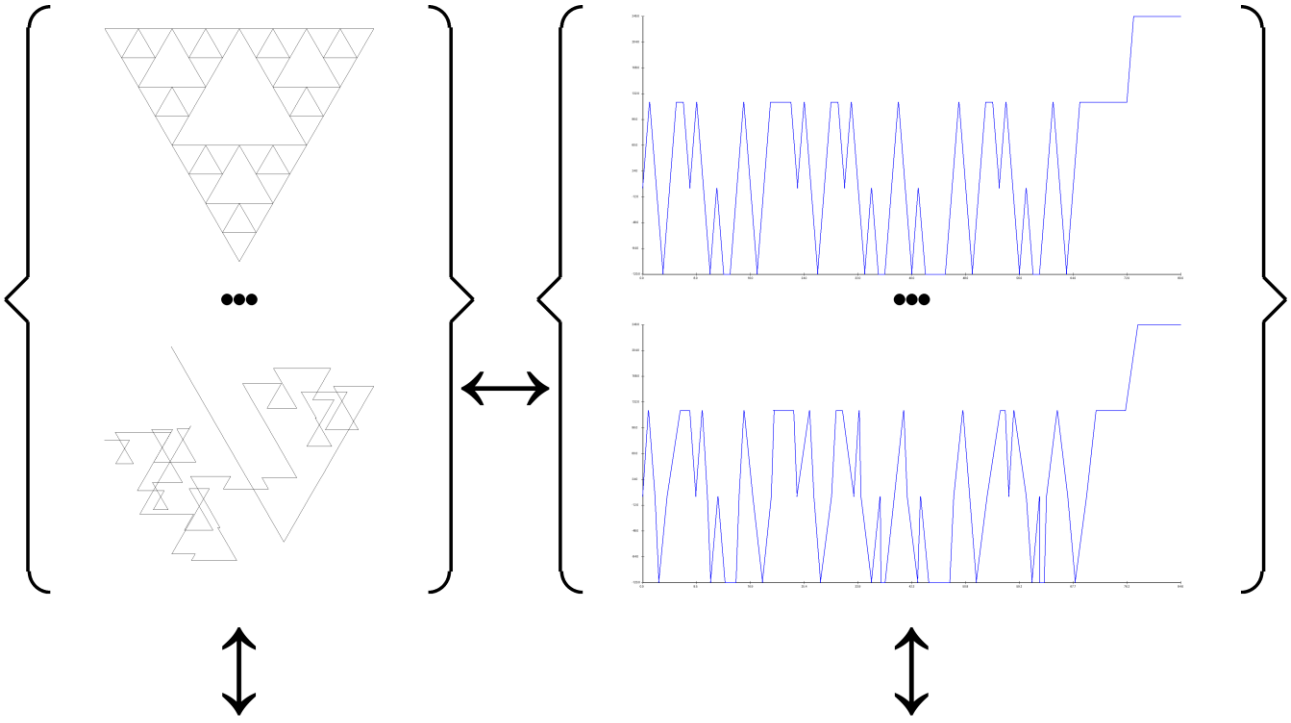
F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F--
F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F

Рисунок Б.4 – Бієктивні відображення фрактал "Сніжинка Коха"



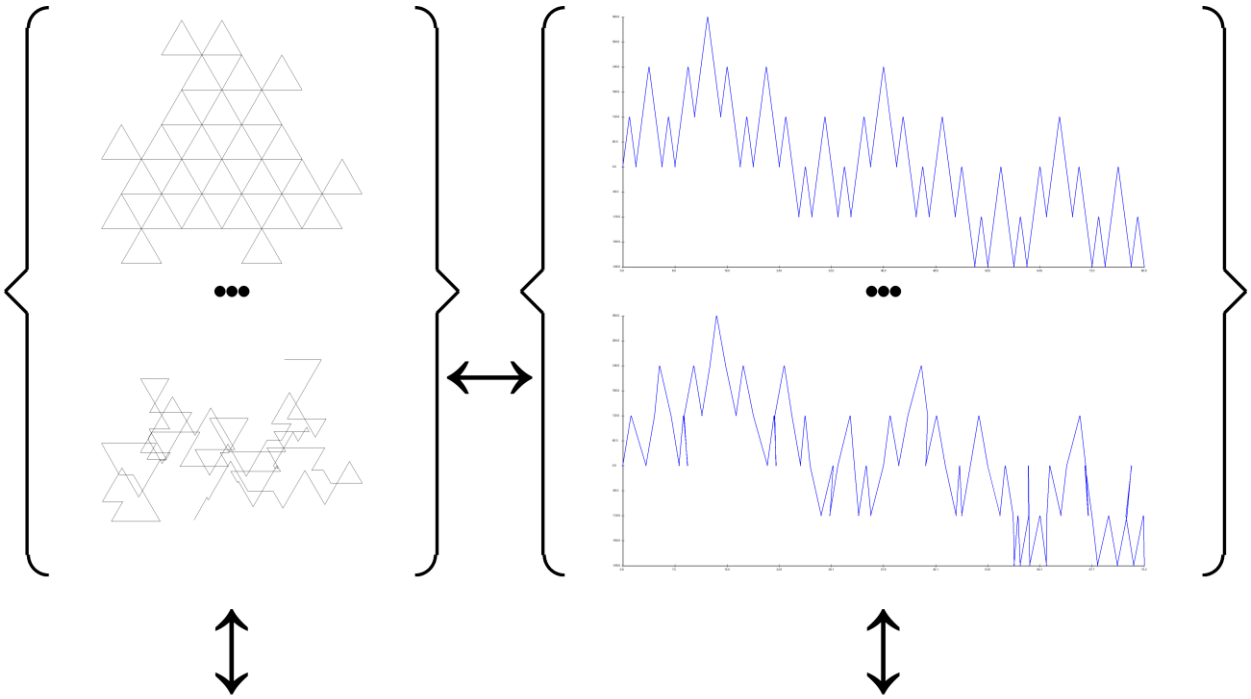
+-+-A+B+A--+B-A-B+--A+B+A-++-+B-A-B++-A+B+A-++B-A-B+-++-A+B+A--+B-A-B+--A+B+
A-+--+-+B-A-B++-A+B+A-++B-A-B+--+-A+B+A--+B-A-B+--A+B+A-+--+B-A-B++-A+B+A-++
B-A-B+-+--+-A+B+A--+B-A-B+--A+B+A-++-+B-A-B++-A+B+A-++B-A-B+-++-A+B+A--+B-A-
B+--A+B+A-+-+

Рисунок Б.5 – Бієктивні відображення фракталу "Крива Серпінського"



F-G+F+G-F-GG+F-G+F+G-F+GG-F-G+F+G-F-GGGG+F-G+F+G-F-GG+F-G+F+G-F+GG-F-G+F+G-F
+GGGG-F-G+F+G-F-GG+F-G+F+G-F+GG-F-G+F+G-F-GGGGGGGG-GGGGGGGG

Рисунок Б.6 – Бієктивні відображення фракталу "Трикутник Серпінського"



F-F+F-F-F+F+F-F+F-F-F+F-F-F+F+F-F+F+F-F+F-F-F+F+F-F+F+F-F+F-F-F+F+F-F+F-F-F+
F-F-F+F+F-F+F+F-F+F-F-F+F+F-F+F+F-F+F-F-F+F+F-F+F-F-F+F-F-F+F+F-F+F+F-F+F-F-
F+F+F-F+F

Рисунок Б.7 – Бієктивні відображення фракталу "Трикутники"

Додаток В Акт впровадження

«ЗАТВЕРДЖУЮ»
Перший проректор
Українського державного університету
науки і технологій
_____ д.т.н., проф. Анатолій РАДКЕВИЧ
_____ 03. 06. 2025 р.

АКТ

Про використання результатів дисертації
«КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ ФРАКТАЛІВ»
Чигіра Роберта Романовича,
представленої на здобуття наукового ступеня доктора філософії
за спеціальністю 122 «Комп'ютерні науки»

Цей акт складений про те, що у навчальному процесі Дніпровському інституту інфрастуктури і транспорту Українського державного університету науки і технологій при підготовці аспірантів за спеціальністю 122 «Комп'ютерні науки» використовуються наукові та практичні результати, що отримані в дисертації Чигіра Р.Р., при викладені дисципліни «Ефективність інформаційних систем та комп'ютерних технологій».

Зав. кафедри
Комп'ютерні інформаційні технології
к.т.н., доцент _____ Вадим ГОРЯЧКІН

Декан факультету
Комп'ютерні технології і системи
к.т.н., доцент _____ Володимир МАЛОВІЧКО