Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Магістерська робота

освітній ступінь – магістр

на тему: «**ВІЗУАЛІЗАЦІЯ ОРБІТ НА ДИВНИХ АТРАКТОРАХ ТА КОМП'ЮТЕРНИЙ ЕКСПЕРИМЕНТ У СЕРЕДОВИЩІ TWINE**»

Виконав: студент 2-го року навчання,
освітньо-наукової програми
«Інженерія програмного
забезпечення», 121

Войцеховський Євгеній
Олександрович

Керівник Авраменко О.В.,_____
доктор фіз.-мат. наук, професор

Рецензент Макарчук О.П._____
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою _____

Секретар ЕК _____
«____» _____ 20____ р.

Київ – 2024

Тема: **Візуалізація орбіт на дивних атракторах та комп’ютерний експеримент у середовищі Twine**

Графік узгоджено «___» _____ 20___ р.

№ з/п	Назва етапу дипломної роботи	Термін виконання етапу	Примітка
1.	Вибір теми та наукового керівника, отримання завдання на дипломну роботу.	16.10.2023	
2.	Огляд літератури за темою роботи.	листопад – грудень	
3.	Проведення досліджень	січень – лютий	
4.	Опис результатів дослідження, написання кваліфікаційної роботи	березень – квітень	
5.	Аналіз отриманих результатів з керівником	квітень	
6.	Корегування роботи	травень	
7.	Попередній захист кваліфікаційної роботи	17.05.2024	
8.	Створення презентації та написання доповіді.	до 01.06.2024	
9.	Подання роботи для перевірки на плагіат	до 03.06.2024	
10.	Подання на зовнішню рецензію	до 08.06.2024	
11.	Захист кваліфікаційної роботи	10.06.2024 – 11.06.2024	

Науковий керівник _____ (ПІБ)
Виконавець кваліфікаційної роботи _____ (ПІБ)

ЗМІСТ

АНОТАЦІЯ

У магістерській роботі було досліджено динамічні системи та дивні атрактори (Лоренца, Ресслера, Чен-Лі та Дадрас). Робота включає теоретичний аналіз, огляд засобів моделювання та розробку програмного продукту для візуалізації цих систем за допомогою середовища розробки Twine. Проведений ряд експериментів, порівняні властивості та знайдені закономірності у цих нелінійних динамічних системах.

Ключові слова: динамічні системи, хаос, комп’ютерне моделювання, візуалізація, дивні атрактори, середовище Twine.

ABSTRACT

The master's thesis explored dynamic systems and strange attractors (Lorenz, Rössler, Chen-Lee and Dadras). The work includes theoretical analysis, an overview of modelling tools, and the development of a software product for visualizing these systems using the Twine development environment. Experiments were conducted, properties were compared, and patterns were identified in these nonlinear dynamic systems.

Keywords: dynamic systems, chaos, computer modelling, visualization, strange attractors, Twine.

ВСТУП

Першим прообразом системи, що володіє хаотичними властивостями, стала «задача трьох тіл», що в деяких випадках проявляє неймовірну чутливість до початкових умов. Початок дослідження саме хаосу та дивних атракторів поклав Едвард Лоренц у 1963 році [18]. У 1970-х роках ці дослідження продовжили математики Флоріс Такенс та Девід Руелл, що першими ввели поняття «дивний атрактор» та використали його для побудови моделі турбулентності рідини [30].

Подальший розвиток обчислювальної техніки значно прискорив вивчення хаотичних систем, і у 1988 році з'явилася програма Fractint (початкова назва FRACT386), яка могла досить швидко будувати фрактали та подібні їм структури. У наступні роки з'явилися більш потужні програмні пакети, як Mathematica, MATLAB і Python з бібліотеками SciPy та NumPy.

Сучасне програмне забезпечення, зокрема середовище розробки Twine, дозволяє створювати інтерактивні програми візуалізації таких складних систем, як дивні атрактори, та спростити їхнє вивчення більш широким колом науковців.

Мета: розробити програмний продукт у середовищі Twine для візуалізації дивних атракторів та провести у ньому серію експериментів для підтвердження властивостей орбіт у динамічних системах з хаосом.

Для досягнення цієї мети були визначені наступні **завдання:**

- Провести огляд теоретичних основ динамічних систем, дивних атракторів та програмних продуктів, що моделюють динаміку їхньої поведінки.
- Обрати та обґрунтувати спосіб побудови проекцій орбіт та функції, що необхідні для просторового зображення декількох динамічних систем.
- Розробити програмний продукт для візуалізації дивних атракторів у середовищі Twine [37].
- Використати розроблений програмний продукт для експериментального аналізу орбіт у динамічних системах з хаосом.

Об’єкт дослідження: комп’ютерне моделювання хаотичного руху у динамічній системі за наявності дивних атракторів.

Наукова новизна: підтверджено поведінку та властивості дивних атракторів, побудовано проекції їхніх орбіт та розроблено програмний продукт для візуалізації з використанням сучасного середовища розробки Twine.

Практичне значення: результати роботи та створене програмне забезпечення можуть бути використані для моделювання хаотичних систем, подальшого вивчення дивних атракторів, їхньої візуалізації в навчальних цілях, розробки нових алгоритмів та програмних продуктів.

Апробація результатів магістерської: результати роботи були представлені на XII Всеукраїнській Науковій Конференції Молодих Математиків [41].

Текстова частина цієї роботи складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі представлено теоретичні відомості про динамічні системи. Описані різні види дивних атракторів, такі як атрактор Лоренца, атрактор Ресслера, атрактор Чен-Лі та атрактор Дадрас. Розглянуті засоби моделювання, що використовуються для візуалізації таких систем.

Другий розділ присвячений розробці програмного забезпечення. Тут проведений огляд інструментів розробки, зокрема середовища розробки Twine, а також представлена програмна реалізація проєкту. Докладно описані елементи програмної реалізації та можливості, що доступні користувачам.

Третій розділ містить результати експериментів для візуалізації та аналізу динаміки атракторів. Тут протестовані можливості розробленого програмного забезпечення, побудовані дивні атрактори та порівняні їхні характеристики.

РОЗДІЛ 1. ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ З МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ

1.1 Елементи теорії динамічних систем

Для початку варто зазначити основні визначення й терміни, які будуть використані у цій роботі та подальшому дослідженні.

Динамічна система – це математична абстракція, що описує та вивчає складні системи, які еволюціонують з часом [39]. Динамічна система складається з *правил*, що описують її поведінку, та *змінних*, які визначають поточний стан системи. Така система має деякий початковий стан, що під дією правил набуває нового значення. Зазвичай, правила задаються рівнянням або системою рівнянь, а сукупність всіх допустимих значень системи називаються *фазовим простором* системи.

Фазова траєкторія – це траєкторія переміщення точки у фазовому просторі таким чином, що кожна наступна точка цієї траєкторії відповідає значенню, яке система набуває на наступному кроці своєї еволюції. Також цю послідовність точок можуть називати *орбітою* динамічної системи.

У цій роботі розглядаються *нелінійні* динамічні системи, поведінка яких не є прямо пропорційною зміні їхніх параметрів. Вони так називаються тому, що в ролі правил у них виступає деяка система нелінійних рівнянь. Також ці системи мають бути *детермінованими* або тими, в яких майбутній стан системи повністю визначається її поточним станом та правилами переходу в новий стан.

Проте існують детерміновані динамічні системи, які через деяку кількість ітерації починають поводити себе хаотично та, здавалось, непередбачувано. Таке стається тому, що існують неймовірно малі неточності у заданні або вимірюванні деякої змінної, яку використовує система. Це явище називається *детермінованим* або *динамічним хаосом*.

Детермінований хаос – це поведінка динамічної системи, яка всупереч своїй детермінованій природі може продукувати складні та непередбачувані результати. Це стається тоді, коли динамічна система є нестійкою та дуже чутливою до значень

своїх параметрів. Оскільки такий хаос є детермінованим, поведінку системи можна з'ясувати шляхом послідовних обчислень, але ніколи не передбачити з самого початку. У таких системах не буває жодної невизначеності або випадковості, та їхня нерегулярна поведінка виникає виключно з нелінійності самої системи, а не в наслідок якихось шумів чи впливу зовнішніх факторів. Якщо траєкторії якоїсь системи прямують до нескінченності, вона не буде хаотичною, оскільки хаотична поведінка має бути аперіодичною та не мати фіксованих точок.

Робота [13] дає наступне визначення. Динамічна система F є *хаотичною,* якщо:

1. Періодичні точки для F є щільними.
2. F є транзитивною.
3. F дуже чутлива до початкових умов.

У детермінованих динамічних системах можуть виникати такі поняття, як притягуючі та відштовхуючі фіксовані точки, що називаються атракторами та репелерами відповідно. Робота [13] також розглядає нейтральні та періодичні точки, але це дослідження зосередиться на точках першого виду (атракторах).

Атрактор – це підмножина станів системи, що складається з наборів станів, до яких система має тенденцію розвиватися з часом, що прямує до нескінченності. Робота [33] вказує, що атрактор A має володіти такими властивостями:

1. A – це інваріантна множина. Будь-яка траєкторія x(t), що починається в A, постійно залишається в A.
2. A притягує до себе всі траєкторії, які починаються достатньо близько від неї.
3. A є мінімальною. Не існує іншої підмножини в A, що задовольняє перші дві умови.

Якщо об'єднати поняття атрактор та складну поведінку, буде утворена математична модель, що називається *дивний атрактор*. Це поняття було вперше вжито у роботі [30] для опису атрактора, що виникав у рідині під час процесу біфуркації (роздвоєння, після якого система стає непередбачуваною).

Дивний атрактор – це атрактор, що притягує до себе всі траєкторії динамічної системи та робить їхні рухи складними й неперіодичними [42]. Фазові траєкторії такого руху ніколи не перетнуться, не покинуть деякої обмеженої зони та можуть повернутися до околу початкового стану, характер чого не можна передбачити. Більша частина дивних атракторів є хаотичними (на них діє детермінований хаос), але можуть існувати дивні нехаотичні атрактори [1, 40]. Внаслідок цього динамічні системи з дивними атракторами (надалі вони вважатимуться хаотичними) є чутливими до початкових умов, але вони стійкі у тому сенсі, що захоплена атрактором траєкторія не покине простір його дії.

1.2 Відомості про деякі дивні атрактори

1.2.1 Атрактор Лоренца

Першим дивним атрактором (Lorenz attractor), що був відкритий та досліджений, стала система з трьох диференціальних рівнянь, які Едвард Лоренц запропонував у 1963 році [18], коли працював над спрощеною моделлю атмосферної конвенції:

$$\begin{cases} \dot{x} = -\sigma x + \sigma y \\ \dot{y} = -xz + rx - y \\ \dot{z} = \quad xy - bz \end{cases}$$

Ці рівняння описують динаміку атмосферної конвенції за допомогою трьох змінних: *x*, *y* та *z*, що представляють швидкість конвенції, різницю температур між висхідними та низхідними потоками повітря та відхилення розподілу температур відповідно. Параметри σ, r, та b керують поведінкою системи та в оригінальній статті Лоренца мали значення σ = 10, r = 28 та b = 8/3. На Рисунку 1.1. показані проекції даного атрактора на площини xy, xz та yz відповідно, де початкова точка має значення: x = 3, y = 1, z = 15.

Система Лоренца стала всесвітньо відомою зокрема через те, що навіть найменше зміщення початкової точки впливає на поведінку всього атрактора через деяку кількість повторних обчислень (це твердження буде підтверджене у Розділі 3). З моменту відкриття було проведено багато досліджень атрактора Лоренца, але

навіть зараз він залишається не до кінця вивченим. Дослідники переважно зосереджуються на зміні параметрів цього атрактора (зокрема значення r), пошуку стабільних точок та дослідженні інших властивостей.

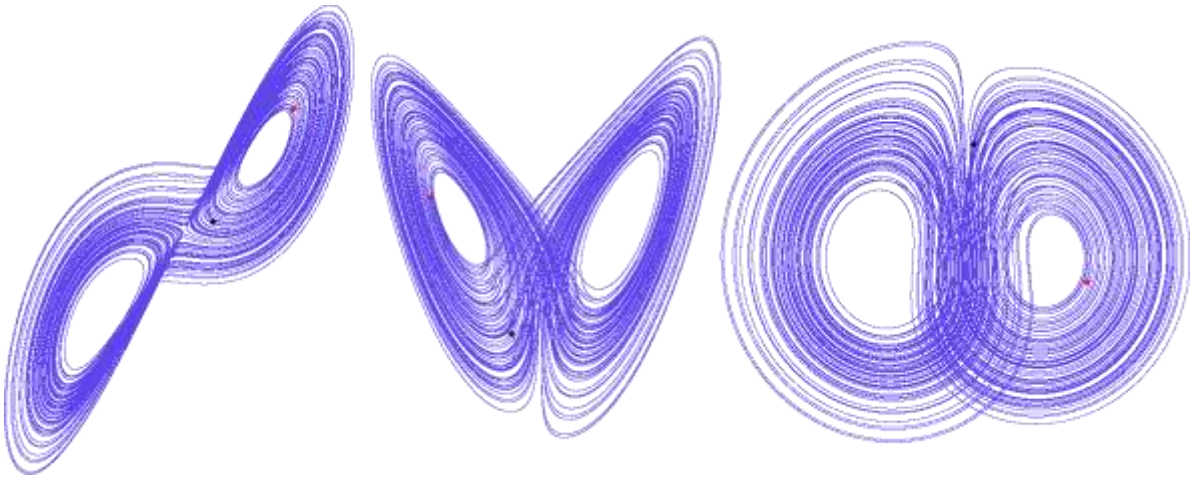


Рисунок 1.1. – Атрактор Лоренца

Наведемо приклади його використання. Деякий програмний код у відкритому доступі використовує набір σ = 5, r = 15 та b = 1, щоб позбутися нецілих значень класичного набору параметрів, дослідження [23] використовує систему Лоренца, щоб побудувати замкнуті цикли та періодичні структури, а в роботі [46] були розроблені синергічні закони для керування хаотичними моделями, що побудовані на атракторах Лоренца та Росслера.

1.2.2 Атрактор Росслера

Одним з наступних достатньо відомих атракторів, які почали досліджувати, став дивний атрактор, який запропонував Отто Росслер у 1976 році [29] для моделювання хімічних процесів (Rössler attractor). Він задається наступною системою рівнянь:

$$\begin{cases} \dot{x} = -(y + z) \\ \dot{y} = x + ay \\ \dot{z} = b + z(x - c) \end{cases}$$

При зміні значень параметрів цей атрактор показує як періодичну поведінку, так і хаотичну, але в порівнянні з атрактором Лоренца йому потрібно більше ітерацій, щоб орбіта атрактора змінилася достатньо помітно (підрозділ 3.4). Сам

атрактор має вигляд переплетених петель та спіралей, як показано на Рисунку 1.2 (проекції на площини xy, xz та yz).

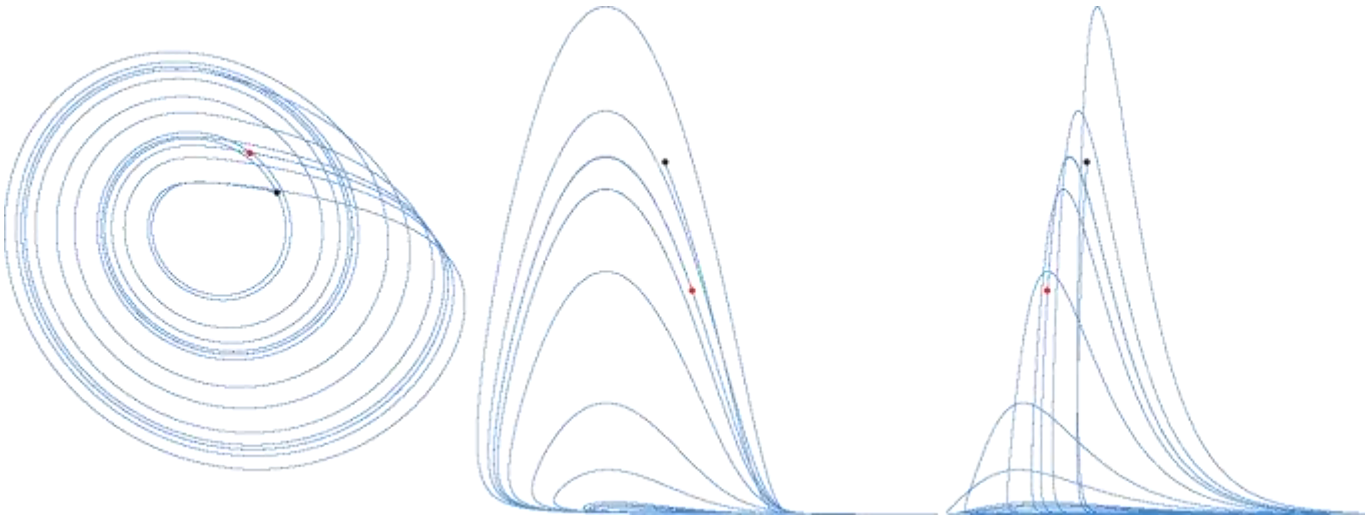


Рисунок 1.2. – Атрактор Ресслера

Стандартні значення його параметрів: a = 0.2, b = 0.2, c = 5.7. Цей атрактор використовувався для моделювання коливань хімічних реакцій з хаотичною поведінкою [19], шифрування та синхронізації хаотичних осциляторів [12, 43] та в інших галузях.

1.2.3 Атрактор Чен-Лі

У попередніх підрозділах були розглянуті більш давні та краще вивчені атрактори, а зараз ми зосередимося на новітніх відкриттях. Першим з таких буде розглянутий атрактор Чен-Лі (Chen-Lee attractor), що був запропонований дослідниками Hsien-Keng Chen та Ching-I Lee у 2004 році [7]:

$$\begin{cases} \dot{x} = ax - yz \\ \dot{y} = by + xz \\ \dot{z} = cz + \dfrac{xy}{3} \end{cases}$$

Стандартні значення його параметрів: a = 5, b = -10, c = -0.38. Рисунок 1.3 демонструє зовнішній вигляд цього атрактора. Дослідження [17] вказує, що для появи хаотичної поведінки ці параметри мають відповідати наступним умовам: $a > 0$, $b < 0$, $c < 0$ та $0 < a < -(b + c)$, хоч це лише один з трьох випадків. Робота [2]

з’ясувала, що наведена вище система рівнянь має п’ять точок рівноваги та обчислила їх для значень a = 5, b = -10, c = -3.8.

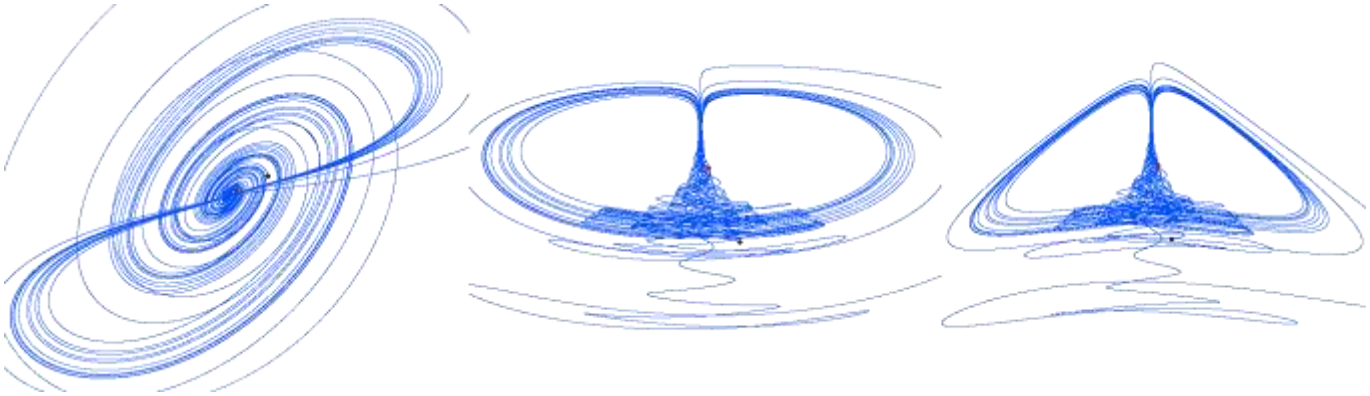


Рисунок 1.3. – Атрактор Чен-Лі

Завдяки неймовірній чутливості атрактора Чен-Лі до змін в початкових умовах, він знайшов застосування у різних галуззях. Наприклад, робота [15] використала його систему рівнянь для аналізу якості підшипників та знаходження дефектів різного типу (зовнішній або внутрішній).

Подібну роботу проведено у роботі [22], автори якої використовують систему Чен-Лі для знаходження критичних проблем під час обробки металів, шляхом розгляду різних сигналів вібрацій. Також були запропоновані методи для синхронізації двох хаотичних систем Чен-Лі [9], що може бути корисно для початкової ініціалізації кількох атракторів, які потім будуть використані для деякої спільної роботи.

1.2.4 Атрактор Дадрас

Останній дивний атрактор, що розглянутий у цій роботі, запропонували у 2009 році Сара Дадрас та Гамід Реза Момені [11], через що у літературі можна зустріти як «атрактор Дадрас» (Dadras attractor), так і «атрактор Дадрас-Момені» (Dadras-Momeni attractor). Цей атрактор задається формулою:

$$\begin{cases} \dot{x} = y - ax + byz \\ \dot{y} = cy - xz + z \\ \dot{z} = dxy - ez \end{cases}$$

Стандартні значення його параметрів: a = 3, b = 2.7, c = 1.7, d = 2, e = 9 (в оригінальній статті параметр c = 4.7). На ці значення накладаються обмеження, щоб

всі вони були більше нуля. Ця система має п’ять точок рівноваги та може генерувати дивні атрактори з двома, трьома та чотирма завитками (scrolls). Рисунок 1.4 демонструє зовнішній вигляд цього атрактора.

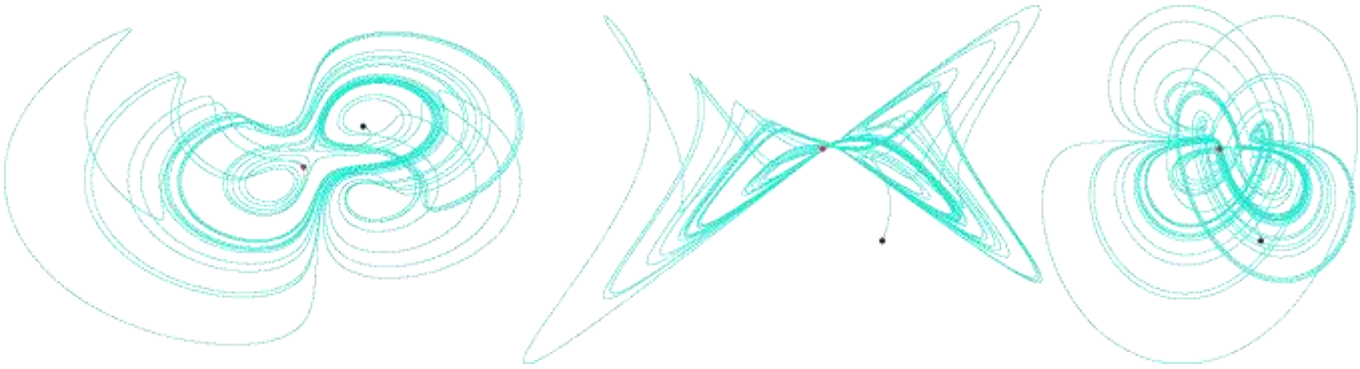


Рисунок 1.4. – Атрактор Дадрас

Оскільки цей атрактор найновіший з розглянутих, було проведено меншу кількість досліджень, які вивчали його поведінку та застосування. З наявних можна відзначити хіба що роботу 2023 року [3], що аналізує даний атрактор у контексті оператора Капуто-Фабриціо.

1.3 Огляд програмного забезпечення

У цьому підрозділі розглянемо наявні засоби для візуалізації орбіт дивних атракторів, порівняємо їх між собою та з’ясуємо, якими характеристиками має володіти кінцевий програмний продукт. Для такої цілі було обрано 17 різних програм, досліджено їхні особливості та результати порівняння внесені у Таблицю 1.1. Розглянемо по черзі стовпці цієї таблиці.

№ (номер). У цьому стовпці пронумеровані усі програми, щоб було зручніше говорити про окремі властивості деяких з них. Зіставлення номерів з програмами: №1 [4], №2 [48], №3 [25], №4 [21], №5 [10], №6 [28], №7 [20], №8 [31], №9 [24], №10 [5], №11 [38], №12 [8], №13 [27], №14 [16], №15 [44, 45], №16 [36] та №17 [26].

Мова програмування. Цей стовпчик був використаний для сортування таблиці, та він показує мову, якою була написана та чи інша програма. Як бачимо, більшість програм використовують мову JavaScript, що можна пояснити тим, що

такі програми не створюють проблем під час запуску користувачем. На другому місці стоїть популярна мова Python, та є декілька інших мов.

Кількість атракторів. Можна помітити, що більша частина програм здатна побудувати більш ніж десять атракторів, а максимальна кількість становить 62. Велика кількість атракторів збільшує час на розробку програми, який можна було б використати на створення інших функцій, тому це потрібно враховувати.

Проекція. Спосіб візуалізації дивних атракторів. Майже всі програми надають 3D проекцію, а деякі окрім того дозволять спроектувати атрактор на площину.

Взаємодія. Дії, які користувач може робити з візуалізацією атрактора, поділяються на обертання, масштабування та зсув. Можна помітити, що якщо якась взаємодія є, то це точно обертання. Тобто, немає таких програм, які можуть масштабувати атрактор, але не можуть повернути його.

№	Мова програ-мування	Кількість атракто-рів	Проекція	Взаємодія		Вид	Побудова	Початкова точка	Зміна параметрів
				Обертання	Масштаб				
1	C, інші	18	3D	Ні	Ні	Фото, код	Лінії, точки	Одна	Ні
2	Delphi	2	3D	Невідомо	Невідомо	Фото програми, код	Лінії	Одна	Так
3	GLSL	10	3D	Ні	Ні	Відео, код	Точки	Багато	Ні
4	JavaScript	6	3D	Так	Ні	Онлайн, код	Точки	Багато	Ні
5	JavaScript	1	3D	Так	Так	Онлайн	Лінії, точки	Багато	Випадкові
6	JavaScript	31	3D	Так	Так	Онлайн	Точки	Багато	Деякі
7	JavaScript	62	3D	Авто	Так	Онлайн	Лінії	Одна	Ні
8	JavaScript	10	3D	Так	Так	Онлайн	Лінії, точки	Багато, одна	Ні
9	JavaScript	4	1 площина	Ні	Ні	Онлайн, програма	Точки	Багато	Так
10	JavaScript	16	3D	Так	Так	Онлайн, програма	Точки	Багато	Так
11	JavaScript	31	3D, 6 площин	Так (3D)	Так (3D)	Програма	Лінії, точки	Одна	Так
12	JavaScript	12	3D	Так	Так	Онлайн, програма	Лінії	Одна	Так
13	JavaScript	43	3D	Так	Так	Онлайн, код	Точки	Багато	Так
14	Python	8	3D, 3 площини	Так	Так	Програма	Лінії	Одна	Так
15	Python	2	3D	Так	Так	Фото програми	Лінії	Одна	Так
16	Python	26	3D	Невідомо	Невідомо	Програма	Лінії	Одна	Невідомо
17	Невідомо	10	3D	Так	Ні	Відео, фото	Точки	Багато	Ні

Таблиця 1.1. – Порівняння програм

Зсув – це досить специфічна операція, яка часто конфліктує з обертанням, бо для цього використовується одна і та ж операція (перетягування мишкою). Зсув доступний лише у програмах №7 та №14, тому він не винесений в окремий стовпчик таблиці. Програма №7 використовує для цього кнопки A та D, а програма №14 дозволяє обрати, яку дію користувач може зараз робити: обертати чи переміщувати атрактор.

Вид. Цей стовпчик має наступні значення: онлайн (програма працює як інтернет-сайт), код (наявні лише частини коду, а не вся програма), відео або фото (немає програми, але є результат її візуалізації), програма (можна скачати й запустити) та фото програми (наявний лише інтерфейс програми, а не вона сама).

Побудова. Основні способи побудувати атрактор – це показати рух або траєкторію точки/точок. Як правило, побудова траєкторії використовується лише для одної початкової точки, тоді як для багатьох показується лише їхній рух.

Початкова точка. Таблиця показує, що приблизно половина програм будує атрактор з одної точки, та половина – з багатьох. Деякі програми дозволяють налаштовувати кількість «багатьох» точок, тоді як інші завжди генерують деяку сталу кількість точок.

Зміна параметрів. Цей критерій розділяє програми на дві категорії: програми візуалізації (немає зміни параметрів) та програми аналізу (є зміна параметрів). Програми другого типу є більш широким класом, оскільки вони можуть не тільки візуалізувати деякий заданий атрактор, а й змінювати параметри цього атрактора та його зовнішній вигляд.

Далі будуть розглянуті інші можливості та властивості цих програм, що не описані в загальній таблиці, але важливі для програм аналізу.

Збереження. Цю властивість можна поділити на такі види: збереження візуалізації атрактора та збереження параметрів атрактора. У першому випадку може зберегти лише фото або відео атрактора (програма №6, №9, №12, №13, №14), а в другому зберігаються налаштування атрактора, які можна використати при наступному запуску (програма №10).

Вибір початкової точки. Вісім програм можуть змінювати параметри атрактора, але лише деякі з них (програма №2, №10, №11, №14) дають користувачу можливість однозначно встановити координати початкової точки, що неймовірно важливо для настільки чутливих до змін об’єктів, як дивні атрактори. Без такої можливості користувач не отримає повну інформацію про атрактор, оскільки початкова точка залишиться схованою в коді програми.

Інші функції:

- Можливість задати рівняння для атрактора (програма №10).
- Побудова двовимірних атракторів (програма №11).
- Графіки для опису атракторів (програма №14, №15).
- Позначення початкової та кінцевої точки (програма №16).

Окремим пунктом варто зазначити вимогливість програм та складнощі в їхньому використанні. Програми можуть займати багато місця на жорсткому диску (програма №11), потрібно багато часу для створення фото (програма №9) чи відео (програма №13), складний запуск або встановлення (програма №11, №14, №16) та велике навантаження на оперативну пам’ять комп’ютера, через що програма довго вантажиться та відповідає на запити користувача (програма №9, №12, №17).

Причина більшості з цих проблем полягає в тому, що дивні атрактори можуть складатися з мільйонів точок, які програмі потрібно обробити та візуалізувати. Не всі програми використовують достатньо ефективні алгоритми, які дозволяють швидко та якісно будувати атрактори. Найбільше проблем викликала програма №12, що має широкий ряд можливостей для налаштування зовнішнього вигляду атрактора, але це потребує неймовірно великих ресурсів для роботи.

1.4 Висновки до Розділу 1

Розглянуті основні теоретичні поняття, що фігурують в роботі. Описані характеристики чотирьох дивних атракторів (система рівнянь, особливості, проекції, використання). Проведений огляд програмних засобів для візуалізації хаотичних систем. Зібрана інформація для створення нового програмного продукту.

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ДЛЯ ВІЗУАЛІЗАЦІЇ ДИНАМІКИ

2.1 Постановка задачі

У підрозділі 1.3 були проаналізовані наявні інструменти для візуалізації орбіт дивних атракторів та з'ясовані їхні особливості, що дозволяють іншим користувачам будувати дивні атрактори. На основі такого аналізу спробуємо виділити ті аспекти та функції, що повинен мати програмний продукт, аби задовольнити ці потреби.

Мова програмування. 10 з 17 програм використовують для роботи мову JavaScript через простоту цієї мови та можливість проектувати вебсайти, що будуть візуалізувати дивні атрактори.

Програми на мові Python мають більш широкий спектр можливостей, але разом з цим їх складніше запустити. Користувач повинен мати на своєму пристрої встановлене середовище мови Python та запускати ці програми через «Командний рядок». Також не кожну програму вдається запустити, як це було з програмою №16.

Окрім цих двох мов наявні деякі інші мови програмування, але вони більш специфічні та складніші у використанні. Таким чином дослідження постало перед питанням, яку з двох мов (JavaScript або Python) обрати, і в результаті було прийнято рішення обрати головною мовою JavaScript.

Кількість атракторів. Обрахуємо деякі значення на основі даних Таблиці 1.1. Середня кількість атракторів, яку можуть побудувати всі програми (17 програм), становить 17.17 атрактори. Середня кількість атракторів, яку можуть побудувати програми, де користувач може налаштовувати параметри (8 програм), становить 14.75 атрактори. Тобто, ускладнення програми спричиняє зменшення кількості атракторів, які вона може побудувати (причина цього описана у підрозділі 1.3).

Тоді яка кількість атракторів має бути доступна у нашому програмному застосунку? Це не може бути один атрактор, та написання коду для проектування багатьох атракторів займе надто багато часу. В ході обговорень цього питання з науковим керівником було вирішено зупинитися на чотирьох атракторах, що будуть доступні у першій версії програми (на запити користувачів програма може бути

розширена на більшу кількість атракторів). Така кількість дозволить продемонструвати усі можливості програми й буде простір до майбутніх покращень та доповнень.

Розглянемо решту пунктів, щоб визначитися з необхідним функціоналом програмного продукту:

- Побудова 3D-проекції атрактора, як і більшість програм.
- Побудова проекції атрактора на координатні площини (корисний спосіб візуалізації).
- Взаємодія з атрактором: масштабування, зсув та обертання (доступне лише для 3D-проекції).
- Побудова атрактора окремими точками. Якщо робити достатньо малі кроки, ці точки зіллються в лінію, що буде орбітою дивного атрактора.
- Лише одна початкова точка. Багато точок допомагають у візуалізації дивних атракторів, але для аналізу найкраще підходить спостереження за життєвим циклом одної конкретної точки.
- Можливість змінювати параметри атрактора.
- Два режими: побудова та порівняння орбіт дивного атрактора.
- Інструменти для аналізу атракторів.

2.2 Інструменти розробки

2.2.1 Загальний огляд

Отже, JavaScript був обраний в якості мови програмування, але залишилося визначитися з середовищем, в якому буде розроблятися програмний продукт, та засобами розробки, що будуть застосовуватися. Звісно, можна створити вебсайт (програму) на чистій комбінації мов вебпрограмування (HTML, JavaScript та CSS), але деякі функції давно запрограмовані іншими людьми, тому потрібно використовувати вже готові інструменти, а не «вигадувати велосипед з нуля».

Середовище розробки. На ринку існує багато програм, які полегшують написання коду та взаємодію з програмним проєктом. Деякі найбільш відомі

представники: JetBrains WebStorm, IntelliJ IDEA, VS Code, Notepad++ та інші. Такі середовища самі по собі не визначають кінцевий вигляд програми, а лише використовуються для її створення. Тобто, кожен програміст обирає те середовище, що задовольняє його потреби та володіє потрібним йому функціоналом.

Засоби розробки. Цей аспект має великий вплив на архітектуру, зовнішній вигляд, можливості та функції програми. Наприклад, програма №11 використовувала фреймворк Node.js, щоб створювати локальний сервер та працювати на ньому. Ця платформа корисна для розробки й роботи сайтів, але запуск сервера – це робота програмістів, а не звичайних користувачів, які хочуть одразу використовувати деяку програму. Деякі інші популярні фреймворки: React, Vue, Angular.

Разом з Node.js часто використовується менеджер пакетів npm, що дозволяє скачувати та підключати різні бібліотеки, які використовує програмний продукт. Це також корисний інструмент, але не для користувача, бо розмір цих скачаних бібліотек зазвичай займає значно більше місця, ніж сама програма чи сайт.

Бібліотека jQuery – це ще один інструмент, який значно спрощує написання інструкцій мовою JavsScript, є простим у використанні, сумісний з багатьма інтернет браузерами, дозволяє обробляти різні події, створює анімації тощо.

2.2.2 Середовище розробки Twine

Twine – це інструмент, що використовується для створення інтерактивних історій, текстових ігор, візуальних новел та інших комп’ютерних програм. В основі цього інструмента лежать базові мови вебпрограмування (HTML, JavaScript та CSS), тому зовнішній вигляд програми залежить лише від того, який браузер та пристрій використовується для її запуску.

Twine дозволяє швидко та просто створювати програмні продукти, в основі яких лежить гіпертекст. Тобто, в таких програмах користувач буде натискати на посилання чи кнопки та переходити на деяку нову сторінку з іншим функціоналом та можливостями. Twine називає такі сторінки *«уривками»* (passage) та об’єднує їх в

одну робочу програму. Саме уривки лежать в основі будь-якої програми (назва в термінах Twine – «історія»), що написана на Twine.

Такі уривки виконують функцію «модулів» програми. Тобто, можна винести деякий функціонал (програмний код, елементи зовнішнього вигляду, текстові фрагменти) в окремі уривки, а потім використовувати їх у потрібному місці та порядку. Програма на Twine починається з того, що користувач бачить вміст деякого початкового уривка, може взаємодіяти з ним та переходити на інші уривки. У такій програмі завжди є лише один поточний уривок, що є головним на момент часу, і до нього підключаються інші уривки. Натискання на інтерактивний елемент (посилання або кнопка) може зробити інший уривок поточним, оновити поточний, довантажити новий контент або активувати якусь функцію.

Розглянемо структуру уривка, що зображений на Рисунку 2.1. Уривок має: унікальну назву (№1 на рисунку), текстовий вміст (№2) та теги (№3, за потреби). Назва уривка використовується для його ідентифікації, вміст зберігає деякий програмний код, а теги використовуються для групування уривків у деякі категорії (така функція може бути як для косметичних цілей, так і для активації деякого коду лише для уривків з однаковим тегом).

Рисунок 2.1. – Уривок у Twine (формат SugarCube)

Наступний елемент, що використовується у Twine, це *змінні*, що є надбудовою над змінними JavaScript. Вони поділяються на постійні (глобальні) та тимчасові (доступні, поки відображається деякий уривок). Набір змінних визначає поточний стан програми та використовується для збереження будь-яких результатів її роботи.

Формат історії – це набір стилів, функцій та можливостей, що визначає зовнішній вигляд програми на Twine. Кожна програма може використовувати лише один формат історії. Основні формати Twine: Harlowe [14], SugarCube [34], Snowman [32] та Chapbook [6]. Кожен формат пропонує свій набір css-стилів та *макро* (надбудова над деяким програмним кодом, щоб спростити його використання).

Harlowe надає гарний набір початкових макро, але виступає проти використання JavaScript. *SugarCube* має як великий набір макро, так і тут широко використовується JavaScript. *Snowman* не має макро, але сильно зосереджений на JavaScript та деяких його бібліотеках (Underscore.js, Marked та jQuery). *Chapbook* розділяє свої функції на модифікатори та вставки (modifiers and inserts), що показують деякі елементи та змінюють їх.

Таким чином, кожен формат має свою сферу застосувань та вимоги до знання мов програмування. Синтаксис та можливості форматів сильно відрізняються одне від одного, тому якщо програма написана з використанням деякого формату, її буде дуже складно переробити на інший формат. Надалі буде розглядатися формат SugarCube та його особливості.

Розглянемо зовнішній вид програми, що використовує формат SugarCube (Рисунок 2.2). Програма складається з основного вікна (№1 на рисунку) та бокової панелі (№2), яку можна згорнути кнопкою №4. В основному вікні розташовується поточний уривок (№3), дві кнопки стрілок (№5) використовуються для переходу між уривками та станами програми, а група кнопок №6 дозволяє зберегти стан програми в об’єкті вебсховища localStorage та перезапустити програму.

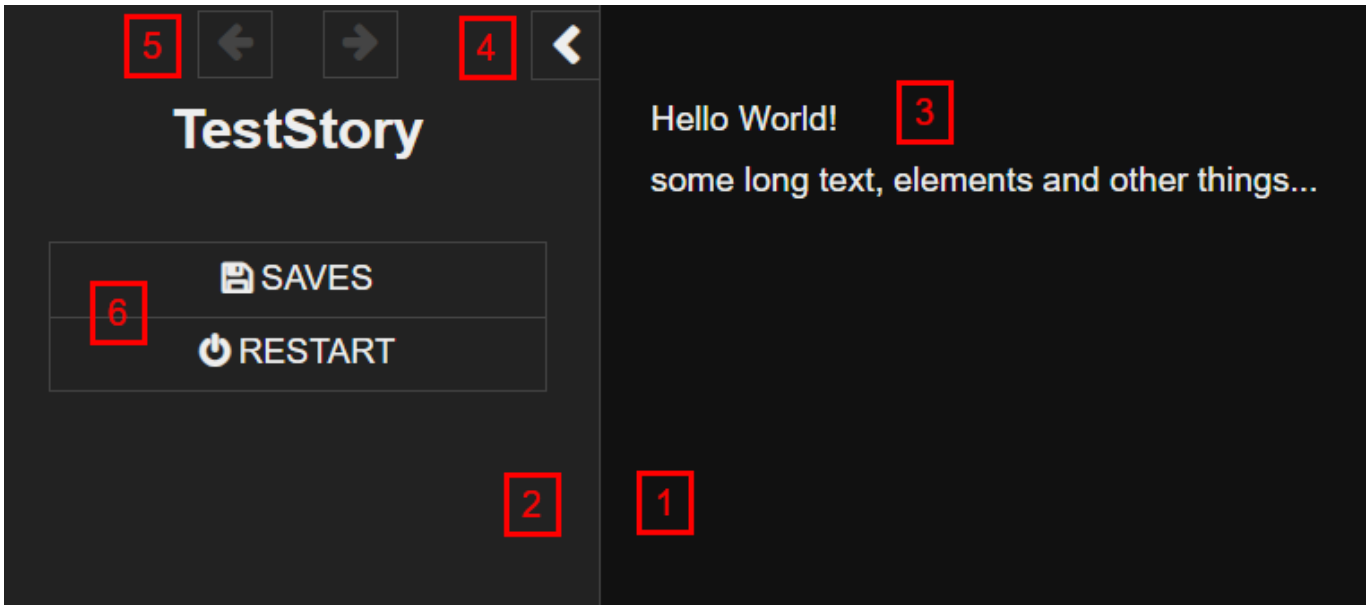


Рисунок 2.2. – Базова будова програми формату SugarCube

Наявність бокової панелі та можливість збереження стану стали одними з причин, чому був обраний саме формат SugarCube. Як було описано у підрозділі 1.3, функція «збереження» дуже важлива для програми візуалізації дивних атракторів, і лише програма №10 володіє подібним функціоналом збереження, як Twine та SugarCube. Звісно, можна запрограмувати з нуля таку функцію, але ми вирішили скористатися вже наявними інструментами, що є досить простими у використанні.

Це далеко не всі можливості, які доступні у Twine, а лише деякі з них, щоб сформувати базове розуміння речей, про які мова піде у наступних підрозділах. Більш детально про Twine можна дізнатися на його головному вебсайті [35] або на сайтах, що присвячені опису можливостей конкретного формату історій [6, 14, 32, 34].

2.3 Програмна реалізація

2.3.1 Системні елементи

Після огляду інших програм та можливостей середовища розробки можна перейти до самого створення потрібної програми, яка буде візуалізувати дивні атрактори та їхні орбіти. Пройдемося по тим уривкам, сукупність яких утворює в результаті потрібний програмний продукт (Додаток А), що отримав назву Attralyzer.

Start. Початковий уривок, з якого запускається програма. У версії Twine 1 мав обов’язково бути уривок з назвою «Start», що був початком програми, але тут використовується Twine 2, де уривок з будь-якою назвою може стати початковим. На Рисунку 2.1. можна побачити пункт «Start Story Here», що використовується для задання початкового уривка. У випадку цієї програми на уривку Start буде доступний перелік доступних атракторів та посилання для переходу на уривки відповідних атракторів.

StoryInit. Уривок з системною назвою, що використовується для ініціалізації змінних одразу після першого запуску програми. Twine успадковує поведінку змінних JavaScript, тож змінні в ньому можуть набувати таких значень: рядки, числа, логічні значення (true та false), об’єкти, масиви (Array) та мапи (Map). В ході роботи з’ясувалося, що програма має мати дуже велику кількість змінних, тому їх потрібно групувати для більш простого доступу та використання.

Розглянемо деякі змінні, що використовує програма Attralyzer (Рисунок 2.3). Всі значення зберігаються у змінних типу «об’єкт», які містять в собі примітивні змінні або інші об’єкти. Об’єкт sysVars зберігає змінні для відображення атрактора, compareMode відповідає за режим порівняння атракторів, об’єкти LorenzAttr, LorenzCompare (та ще 6 об’єктів такого типу для трьох інших атракторів) задають значення атракторів. Також є масив SavedAttrs для збереження атракторів.

```
<<set $sysVars to {
   wasDrawn: false,
   scale: 20,
   shift: {x: 0, y: 0},
   view: {x: 3, y: 2, z: -2},
   angle: {theta: 0, phi: 0},
   constColor: false,
   hasGrid: true,
   mode: "3d"
}>>
<<set $compareMode to {
   epsilon: 0.001,
   wasMoreThanEps: false,
   showAttr1: true,
   showAttr2: true
}>>
<<set $SavedAttrs to []>>
<<set $LorenzAttr to {
   color: [],
   params: {a: 10, b: 28, c: "8/3"},
   initPoint: {x: 3.051522, y: 1.582542, z: 15.62388},
   dt: 0.0001,
   steps: 1000000
}>>
<<set $LorenzCompare to {
   color: [],
   params: {a: 10, b: 28, c: "8/3"},
   initPoint: {x: 3.051522, y: 1.582542, z: 15.62388},
   dt: 0.0001,
   steps: 1000000
}>>
```

Рисунок 2.3. – Структура змінних програми

Розглянемо структуру об’єкта LorenzAttr. За допомогою цієї змінної програма зберігає колір атрактора, його параметри, початкову точку, значення dt та кількість кроків, скільки пройде атрактор. Кількість параметрів (об’єкт params) тут становить три, але один за атракторів (атрактор Дадрас) зберігає в цьому об’єкті цілих п’ять своїх параметрів.

Всі ці змінні важливо ініціалізувати перед початком роботи, тому задаємо їх в уривку StoryInit, який гарантує, що ці змінні точно матимуть значення. Можна задавати змінні й на початковому уривку (уривок Start у нашому випадку), але деколи для тестування обирають інший початковий уривок, і тоді змінні першого початкового уривка не будуть ініціалізовані. Уривок StoryInit гарантує, що такої проблеми не буде, оскільки він завжди запускається на початку програми незалежно від того, який уривок є початковим.

StoryCaption. Це важливий уривок, що використовується для задання бічної панелі навігації (Рисунок 2.2, елемент №2). Програма Attralyzer використовує бічну панель, щоб розміщати на ній елементи для керування об’єктом sysVars (масштаб атрактора, його зсув, кут та положення спостерігача).

Також у програму вбудовані приховані уривки **Story JavaScript** та **Story Stylesheet**, які не відображені серед уривків Додатка А, але наявні в кожній програмі. Ці уривки використовуються для задання JavaScript коду та CSS стилів елементів відповідно. Будь-який уривок програми за замовчуванням має доступ до цієї пари уривків, оскільки вони завжди є підключеними (не потрібно додатково підключати їх кудись).

Інші системні уривки, які використовуються в нашій програмі; *StoryDisplayTitle* (відображає назву поточного атрактора), *StoryAuthor* (відображає поточний режим програми), *PassageReady* (ініціалізує віджети на кожному уривку) та *StoryMenu* (створює кнопку «Допомога» на бічній панелі навігації, натиснення на яку покаже звичайний уривок *Help*). Від програми до програми ці уривки можуть використовуватися для різних цілей, або їх взагалі може не бути.

2.3.2 Головні елементи

Перейдемо до уривків, що безпосередньо відповідають за візуалізацію та побудову атракторів. Як було зазначено у підрозділі 2.1, цей програмний продукт розглядатиме чотири атрактори, а саме атрактор Лоренца, Ресслера, Чен-Лі та Дадрас (властивості яких розглядалися у підрозділі 1.2). Для кожного атрактора створено три уривки: два режими побудови та уривок з програмним кодом. Ці уривки мають подібний код між різними атракторами, тому буде достатньо розглянути уривки деякого одного атрактора. Наприклад, розглянемо уривки атрактора Ресслера.

Rossler Build. Цей уривок задає зовнішній вигляд для режиму «Побудова». Спочатку до цього уривку підключається програмний код з уривків Attr JS та Rossler JS, а потім задаються значення для об’єкта sysVars, що є найкращими для атрактора Ресслера. Якщо підключати Attr JS та Rossler JS не на поточному уривку, а в іншому місці програми, то перезавантаження сторінки програми зітре підвантажені раніше уривки, і програма не зможе отримати до них доступ, видаючи повідомлення з помилкою. Далі розташовані html-елементи зовнішнього вигляду сторінки: об’єкт canvas, на якому візуалізується атрактор, таблиця для зміни параметрів атрактора (об’єкт RosslerAttr) та кнопки взаємодії.

Rossler Compare. Задає зовнішній вигляд режиму «Порівняння». Майже ідентичний уривок до попереднього, але лише з тим винятком, що у таблиці потрібно задавати значення не одного атрактора, а двох атракторів. Якщо говорити у контексті змінних, тут задаються параметри об’єктів RosslerAttr та RosslerCompare. Також тут наявна текстова область, куди виводяться результати порівняння цих двох атракторів.

Rossler JS. Уривок з кодом JavaScript для побудови атракторів. Підключається допоміжний програмний код уривка Helper JS та задаються функції візуалізації атракторів у двох режимах: setup.buildRossler та setup.compareRossler. Приставка setup – це спеціальний об’єкт Twine, через який задаються функції, які потрібно використовувати для різних уривків. Тут потрібно використовувати setup, оскільки існують інші уривки (уривок Story JavaScript), які посилаються на ці функції.

Алгоритм, що використовує функція buildRossler:

- Отримання копії об’єкта RosslerAttr та створення локальних версій його змінних.
- Підбір випадкового кольору атрактора.
- Цикл за кількістю кроків атрактора для його побудови.
- Малювання сітки або осей.
- Малювання початкової та кінцевої точки атрактора.
- Візуалізація отриманих даних в об’єкті canvas.

Алгоритм для функції compareRossler аналогічний та відрізняється тим, що одночасно обчислюються орбіти двох атракторів, та цей результат зберігається у тому ж самому об’єкті, який буде передано у canvas. Завдяки цьому орбіта другого атрактора будо накладатися на орбіту першого, поки вони не розійдуться (якщо значення обох атракторів не є однаковими).

Attr JS. У цьому уривку знаходяться функції, які оновлюють значення повзунків бічної панелі (уривок StoryCaption), та функції, що дозволяють користувачу використовувати мишку для взаємодії з відображенням атрактора (масштабування та зсув).

Helper JS. Деякі головні функції, що відповідають за візуалізацію атрактора.

setup.drawAttractor знаходить координати, де знаходиться конкретна точка атрактора, в залежності від того, який вид проекції був обраний (3D чи проекція на якусь площину).

setup.draw3D обраховує перспективну проекцію атрактора з одної точки за наступними формулами, що описані в роботі [47]:

$$\overrightarrow{r_E} = V\overrightarrow{r_v}$$

$$V = \begin{bmatrix} -sin\theta & cos\theta & 0 \\ -cos\varphi cos\theta & -cos\varphi sin\theta & sin\varphi \\ -sin\varphi cos\theta & -sin\varphi sin\theta & -cos\varphi \end{bmatrix}$$

$$x' = d\frac{x - x_c}{z - z_c}, y' = d\frac{y - y_c}{z - z_c}$$

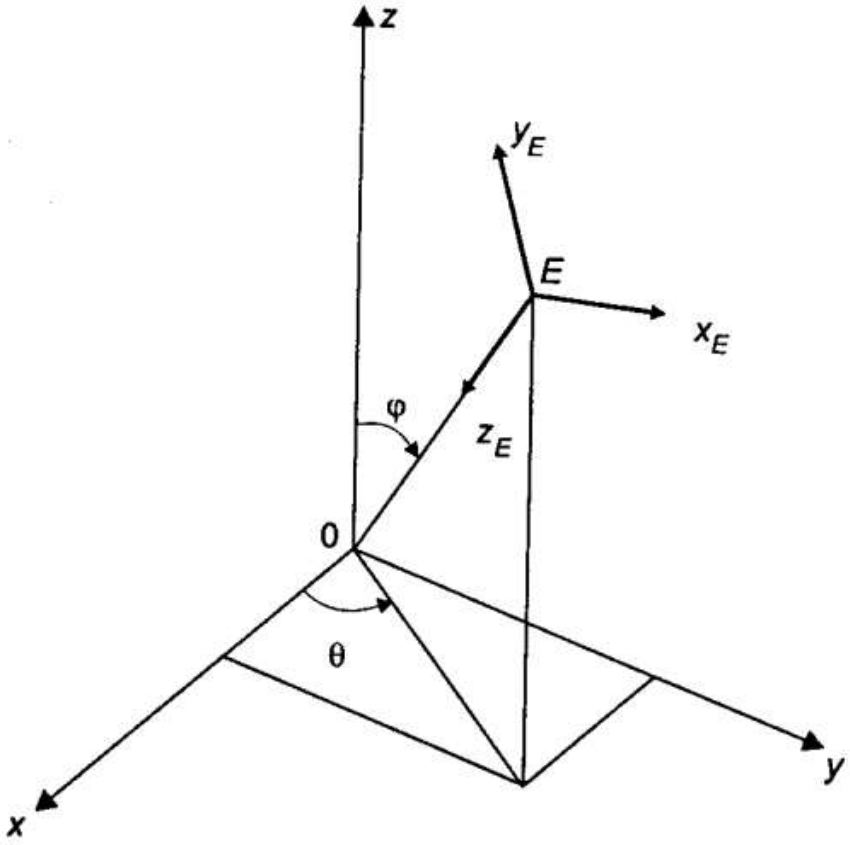


Рисунок 2.4. – Система видових координат [47]

де $\overrightarrow{r_v}$ – початкові координати атрактора у просторі, $\overrightarrow{r_E}$ – координати атрактора у видовій системі координат, V – матриця перетворення, θ та φ – кути сферичної системи координат (зображені на Рисунку 2.4), x, y, z – координати вектора $\overrightarrow{r_E}$ у розгорнутому вигляді, x_c, y_c, z_c – координати точки спостерігача, з якої відбувається спостереження за атрактором, d – відстань від екрана до точки атрактора, x' та y'– координати точки атрактора на екрані програми.

setup.drawBoldPoint та *setup.drawRedPoint* малюють початкову та кінцеву точки атрактора по заданим координатам відповідно. Також перша з них використовується для точки з координатами (0, 0, 0) у 3D-проекції.

setup.drawGrid малює сітку для проекцій на площини xy, xz та yz та ставить масштабовані засічки на осях.

setup.draw3DAxis малює координатні осі для 3D-проекції. Вісь Ox зображена синім кольором, Oy – червоним, а Oz – зеленим.

Уривок **Memory Attr** використовується для показу атракторів, значення яких користувач зберіг у масив SavedAttrs, але показується лише той тип атракторів, який

зараз є поточним. Тобто, якщо поточний атрактор Чен-Лі, то цей уривок покаже лише значення інших збережених атракторів Чен-Лі. Якщо жодний атрактор поточного типу не збережено, то уривок буде порожнім. Уривок **Saved Attr. Template** слугує контейнером для значень якогось одного атрактора, що будуть показані у Memory Attr, та створює кнопки взаємодії (використання значень атрактора або їхнє видалення з масиву).

2.4 Користувацькі можливості

Запустимо програму та подивимося, як користувачі побачать описані вище елементи та можливості. Для запуску програми Attralyzer використовується файл index.html, який відкриється в інтернет-браузері, що обраний у користувача за замовчуванням. Наступні фото інтерфейсу зроблені у браузері Opera 110.0.5130.39.

Програма відкривається як нова вкладка браузера під назвою «Attralyzer» та має вигляд, як Додаток Б. Це показаний початковий уривок Start. В межах цього уривка бічна панель прихована програмними засобами та буде показана, коли користувач запустить режим якогось із доступних атракторів, яких є чотири. Програма показує назву атрактора, фото його візуалізації (зроблені самою програмою) та кнопки з посиланнями для переходу в режим Побудови або Порівняння відповідного атрактора. Також доступна англійська мова інтерфейсу.

Натиснемо на кнопку «Побудова» під атрактором Лоренца, щоб запустити відповідний режим програми (Додаток В). Поточний уривок був змінений на Lorenz Build, назва вкладки змінилася на «Лоренц» та з’явилася решта елементів (бічна панель та основний простір уривку). На бічній панелі показана назва поточного атрактора й режиму, повзунки та кнопки керування. В основному просторі праворуч розташоване полотно для візуалізації, текстові поля для введення даних про атрактор та кнопки керування. Розглянемо ці елементи.

Одразу після переходу у цей режим полотно є порожнім та не відповідає на дії користувача, поки він не натисне на кнопку «Порахувати». Після цього користувач може зсувати (затискання лівої кнопки миші та перетягування) та масштабувати

атрактор (прокручування колесика миші). На порожньому полотні ці дії неактивні, стан «порожній» визначається значенням змінної sysVars.wasDrawn (Рисунок 2.3), що приймає значення true або false. Також під час таких дій автоматично оновлюється значення повзунків масштабування та зсуву на бічній панелі ліворуч. Можна перетягувати ці повзунки вручну, але результат буде показаний лише після натискання кнопки «Порахувати» або якоїсь взаємодії з полотном. Якщо змінити значення деякого повзунка за допомогою миші, його значення надалі можна підкорегувати клавішами «←» та «→» на клавіатурі.

Для режиму 3D-проекції перетягування мишею на полотні дозволяє не зсувати, а обертати атрактор. Горизонтальне перетягування змінює кут θ, а вертикальне – кут φ. Можливість зсуву також залишилася, але для цього потрібно перетягувати відповідні повзунки на бічній панелі.

Одразу під полотном знаходиться табличка з інформацією та полями введення. *Перший* стовпець – система рівнянь, що визначають поточний атрактор. *Другий* стовпець – координати початкової точки атрактора (x, y та z) та параметр dt (щільність атрактора). *Третій* стовпець – параметри атрактора (a, b та c) та кількість кроків (ітерацій обрахунків) steps. У ці рядки можна вносити не тільки числові значення, а й математичні формули (наприклад, 8/3). Якщо ввести некоректне значення або якийсь текст, буде виведено повідомлення про помилку. *Четвертий* стовпець – довідкова інформація з напрямком осей координат для різних проекцій. Рядок активних зараз напрямків буде автоматично виділений жирним шрифтом.

Під таблицею знаходяться кнопки керування. Їхній функціонал:

- **Назад.** Повернення до початкового вікна вибору атракторів та режимів.
- **Пам’ять.** Показ спливаючого вікна зі збереженими атракторами (уривок Memory Attr).
- **Зберегти.** Збереження поточних параметрів, що показані в таблиці вище, до масиву збережених атракторів. Після натискання у правому верхньому куту екрана буде показане спливаюче віконечко з текстом «Атрактор збережено у Пам'ять!».

- **Порахувати.** Активує малювання або перемальовку атрактора на полотні.
- **Скачати.** Скачує вміст видимого користувачу полотна у вигляді фото формату .png, назва якого складається з назви поточного атрактора та часу збереження у форматі «години_хвилини_секунди».

Зазначимо деякі інші властивості основного простору. Розмір полотна становить 800 пікселів у ширину та 600 у висоту. Видимий розмір полотна може бути меншим, якщо користувач зменшить екран програми, але запрограмований розмір 800x600 залишиться незмінним. Також зміна розміру програми буде переносити деякі елементи таблиці на новий рядок, що дозволяє використовувати програму Attralyzer на екранах з різним розширенням (навіть мобільні пристрої).

Бічна панель складається з таких елементів: назва атрактора та поточний режим, повзунки, параметри та допоміжні кнопки. Опишемо призначення та властивості цих елементів.

Масштаб. Діапазон значень: від 10 до 50 одиниць, на які множаться координати атрактора, щоб зобразити його на полотні.

Зсув графіка. Два повзунки: зсув по горизонталі («h» – horizontal) та вертикалі («v» – vertical). Діапазон значень: -38 – +38 горизонталь та -28 – +28 вертикаль. 1 умовна одиниця зсуву дорівнює 10 пікселям полотна, на які зміщається центр полотна по відношенню до свого початкового положення. Наприклад, якщо центр полотна опиниться у лівому верхньому куту, то координати зсуву h та v дорівнюватимуть -38 та -28 відповідно, а якщо у лівому правому куту – +38 та -28 відповідно.

Точка зору. Тут визначаються координати, з яких спостерігач буде будувати 3D-проекцію орбіт атрактора. Діапазон значень для кожної координати (x, y та z) становить від -15 до +15.

Кут спостерігача. Тут визначаються кути повороту видової системи координат, що представлена на Рисунку 2.4. Діапазон значень: від 0° до 360°.

Сталий колір. За замовчуванням для кожної візуалізації атрактора обирається випадковий колір. Якщо цей перемикач знаходиться в активному стані, алгоритм програми буде використовувати попередній колір, а не генерувати випадковий.

Сітка. Відповідає за те, чи показувати під час візуалізації сітку (проекція на площини) або координатні осі (3D-проекція). Окрім візуалізації на полотні, кольори осей показані як колір пояснювального тексту біля повзунків у пункті «Точка зору».

Вид проекції. Вибір режиму проекції має наступний набір значень: 3d, xy, xz та yz, де 3d відповідає режиму 3D-проекції, а решта варіантів – проекція на відповідні координатні площини, де показуються дві ординати точки та ігнорується третя.

Допомога. Відкриває спливаюче вікно з коротким поясненням можливостей програми.

Saves. Використовується для збереження поточного стану програми у localStorage браузера або у виді файлу з розширенням .save. За допомогою такого файлу можна відновити роботу з цією програмою після її закриття. Ця функція корисна для збереження значень масиву SavedAttrs, та у назві збереження відображається кількість об’єктів у цьому масиві. Є десять слотів для збережень, які можна видаляти, робити нові та завантажувати деякий стан програми.

Restart. Запускає програму з початку, обнуляючи всі данні до варіантів за замовчуванням.

Описану вище структуру мають всі інші атрактори у режимі «Побудова», за винятком лише атрактора Дадрас, що має не три параметри (a, b та c), а п’ять, тому компонування його таблиці має дещо іншу структуру.

Розглянемо режим «Порівняння» (Додаток Г) та відзначимо його відмінності від режиму «Побудова».

Збільшення кількості полів введення. У цьому режимі порівнюються між собою два дивні атрактори деякого однакового типу (вони використовують однакову систему рівнянь, але можуть мати різні параметри цієї системи), тому потрібні поля для введення значень обох атракторів.

Видимість атрактора. Параметри першого атрактора згруповані в другий стовпець, а другого – в третій. Кожен стовпчик має кнопку «Показати/Сховати», що

дозволяє приховати відповідний атрактор з полотна. Коли атрактор прихований, його заголовок у першому рядку таблиці стане перекресленим.

Обмін значень. Дозволяє обміняти параметри атракторів. Дуже потрібна функція, щоб поєднувати її з «Пам’ять». Коли програма застосовує значення деякого збереженого атрактора, вона ставить їх лише у поля першого атрактора. Немає можливості вибрати, у які поля будуть вставлені ці значення (можливе місце для покращення програми). Тобто, якщо користувач зберіг деякі два атрактора, він повинен застосувати значення першого атрактора, обміняти значення та застосувати значення другого атрактора.

Розходження атракторів. У четвертий стовпець таблиці додана текстова область, куди виводяться дані порівняння двох атракторів у виді відстані між їхніми відповідними точками. Формат виведення: «Порядковий номер точок – відстань між ними». Нижче цієї області знаходиться текстове поле для введення похибки ε, та програма виведе номер точки, на якому відстань між точками атракторів буде більше величини ε. Якщо параметри обох атракторів повністю ідентичні, такої точки не буде знайдено, але якщо вони хоч трохи відрізняються, момент розходження їхніх орбіт неодмінно настане та буде показаний користувачу.

2.5 Висновки до Розділу 2

Розглянуті інструменти програмування та обрано Twine як середовище розробки. Виділені особливості Twine та обґрунтовано доцільність використання формату SugarCube. Описаний функціонал та призначення уривків програмного продукту. Пояснено інтерфейс користувача та його можливості.

РОЗДІЛ 3. ВІЗУАЛІЗАЦІЯ АТРАКТОРІВ ТА ЕКСПЕРИМЕНТ У ATTRALYZER

3.1 Базові можливості

Перед більш глибоким дослідженням потрібно перевірити правильність роботи створеного програмного продукту та інструментів, що він надає для візуалізації та аналізу дивних атракторів. Наступні перевірки будуть проведені у режимі «Побудова». Для цього запустимо програму та оберемо відповідний режим під зображенням атрактора Лоренца.

Можна побачити, що при наведенні курсора мишки на інтерактивні елементи програми (кнопки, текстові поля, повзунки) програма показує деяку взаємодію (зміна кольору рамок, зміна кольору заднього фону, зміна вигляду курсора). З усіма елементами, які подібним чином реагують, користувач може взаємодіяти. Також можна помітити, що користувач не може виділити пояснювальний текст всередині таблиць та назви кнопок, але без проблем може виділяти вміст текстових полів. Це зроблене для того, щоб зменшити кількість випадкових виділень тексту, які можуть заважати нормальній роботі користувача.

Коли екран програми має вигляд, як у Додатку В, користувач може взаємодіяти з різними елементами, але це ніяк не вплине на полотно. Наприклад, можна міняти значення повзунків на панелі навігації, але це ні на що не впливає, поки користувач не натиснув кнопку «Порахувати». Якщо користувач перезавантажить вкладку браузера з програмою, вміст текстових полів та значення повзунків буде скинуто на значення за замовчуванням. Такий же ефект спостерігається, якщо натиснути кнопку «Назад» (повернення до початкового екрана), а потім знову обрати режим «Побудова».

Побудуємо атрактор. За замовчуванням обраний режим 3D-проекції, але для цих перевірок оберемо режим проекції на площину xy. Спробуємо знову порухати повзунки чи змінити значення. Знову полотно не змінюється, поки не натиснути на кнопку «Порахувати». Така поведінка запрограмована за такою ціллю, щоб не було

непотрібних ітерацій обчислень, поки користувач не скаже програмі, що потрібно побудувати атрактор.

Якщо навести курсор на полотно, можна маніпулювати з ним та атрактором. Колесико мишки масштабує полотно, а затиснення лівої кнопки миші та перетягування у деякому напрямі зсуває полотно. Одночасно з цим змінюються значення повзунків на панелі навігації. Якщо не стоїть галочка на пункті «Сталий колір», то кожна така взаємодія буде змінювати колір атрактора.

Також можна помітити те, що програма будує атрактор досить швидко для проекцій xy, xz та yz, але 3D-проекція реагує трохи повільніше. Причина такого в тому, що для кожної такої взаємодії програма повинна заново будувати атрактор та візуалізувати його на полотні. Візуалізація у тривимірному просторі займає більше обчислювальних операцій, ніж проекція на площини. Якщо зменшувати або збільшувати кількість точок (параметр steps), то програма буде реагувати швидше та повільніше відповідно.

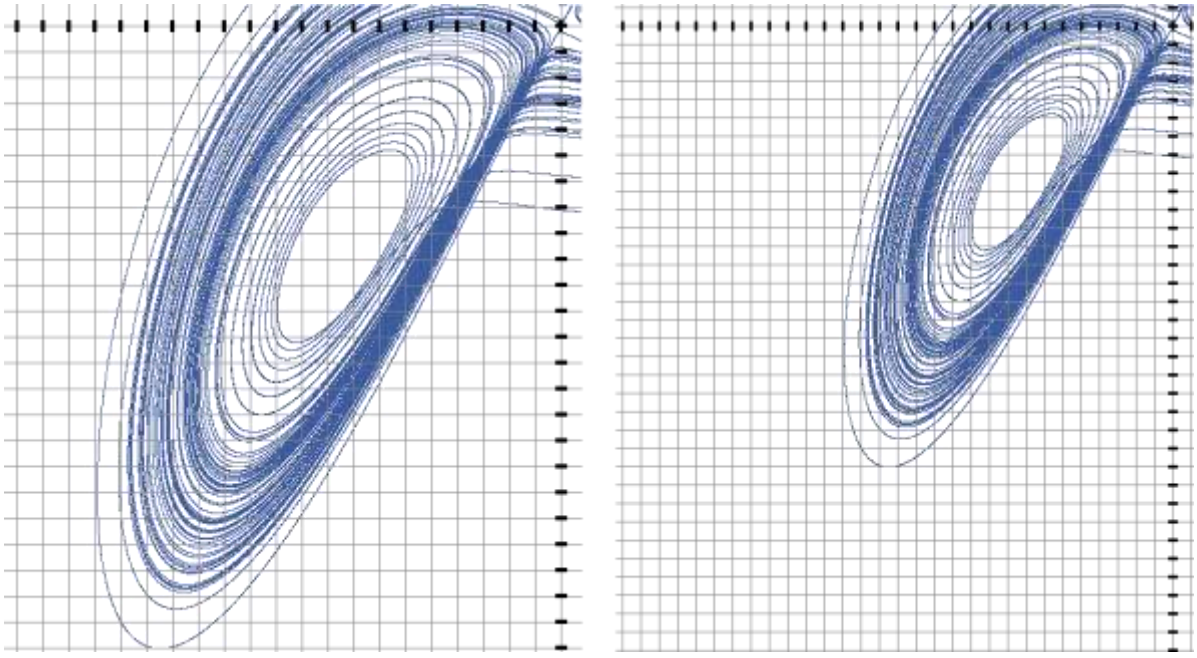


Рисунок 3.1. – Граничний зсув та масштабування

Спробуємо зсунути полотно до максимуму в якусь зі сторін та побачимо, що координатні осі та точка їх перетину (початок координат) завжди знаходяться в межах видимої частини полотна. Якщо змінювати масштаб, ця властивість зберігається (Рисунок 3.1). Таке обмеження програми введено для того, щоб

спростити взаємодію користувача з полотном та завжди тримати точку (0, 0) на полотні.

При масштабуванні змінюється не лише розмір атрактора та кількість показаних точок, а й розмір засічок на координатних осях та розмір початкової та кінцевої точки атрактора. Координатна сітка завжди має ширину в один піксель, і дія масштабування впливає лише на кількість ліній та клітинок сітки, які будуть показані.

3.2 Проекції

Наступний елемент, що потрібно протестувати, це режими проекцій. Почнемо з проекцій на площини xy, xz та yz (Рисунок 3.2). Параметри атрактора, що зображений на рисунку: a = 10, b = 28, c = 8/3, x = 3.051, y = 1.582, z = 5.623. Масштаб програми становить 12 одиниць, а зі створених фото були вирізані частини шириною в 500 пікселів та скомпоновані в один рисунок.

Ці проекції проектують лише дві координати на полотно програми (x та y для першої проекції на рисунку, x та z – другої, y та z – третьої). Координатна сітка відповідає масштабу атрактора, тож її можна використовувати для знаходження координат деякої точки на проекції. Якщо важливий лише зовнішній вигляд атрактора, користувач має можливість приховати сітку, поставивши на панелі навігації галочку у пункті «Сітка».

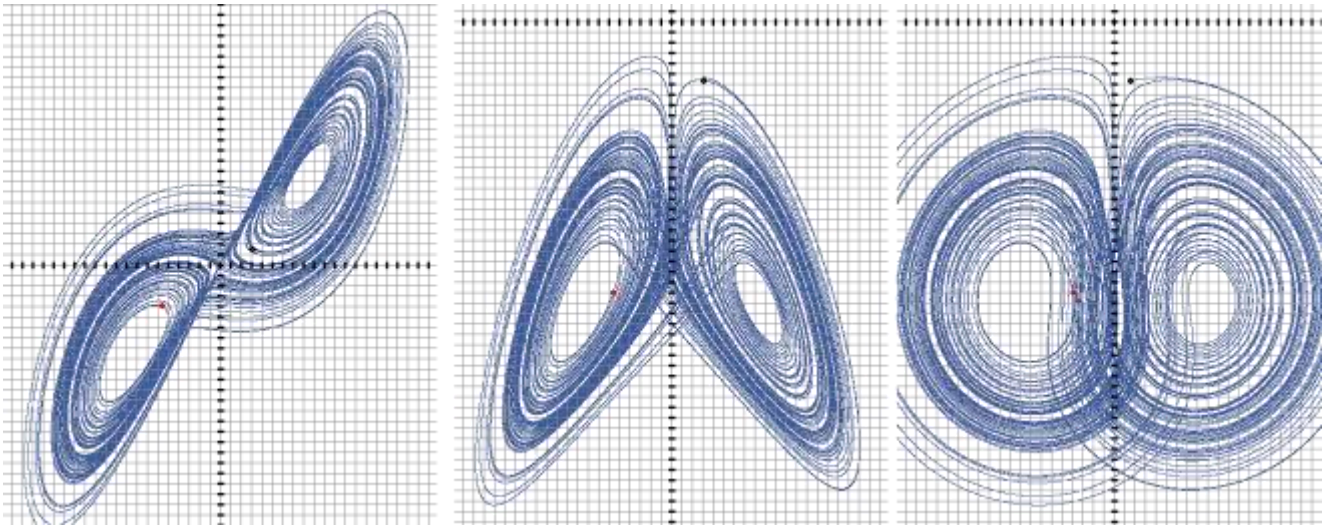


Рисунок 3.2. – Проекції атрактора Лоренца на площини

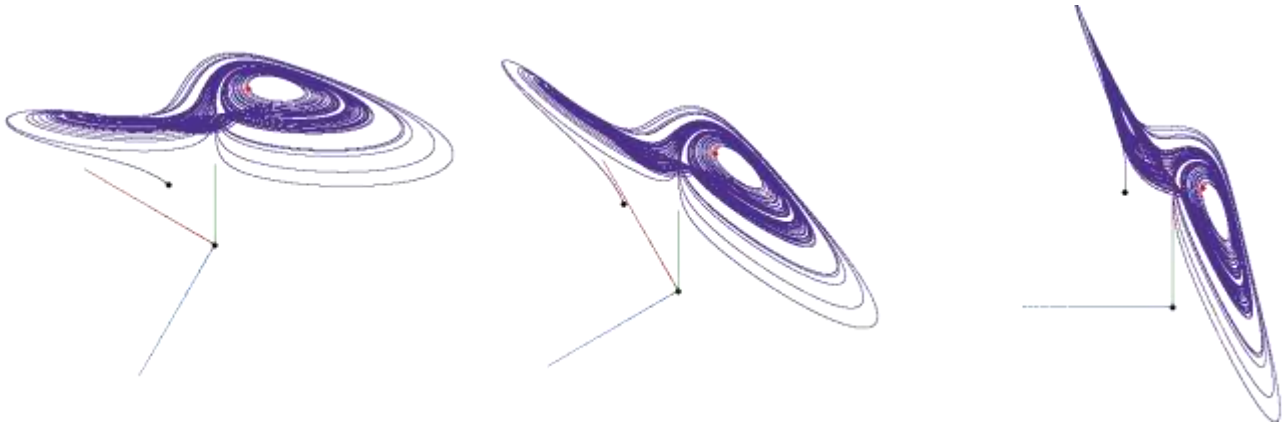


Рисунок 3.3. – Зміна кута θ для атрактора Лоренца

Такі проекції показують правильний вигляд атрактора Лоренца, який можна побачити у роботах інших дослідників [16, 38], тому не будемо зупинятися на цьому, й одразу перейдемо до тестування 3D-проекції.

Програми, розглянуті у підрозділі 1.3, мали різні способи візуалізувати орбіти дивних атракторів, зокрема в тривимірному просторі, тому розглянемо основні властивості пропонованого нами способу (описаний у підрозділі 2.3.2). Головні елементи, що його визначають, – це точка розташування спостерігача та кути, на які він повернутий. З точки спостерігача направлений конус у напрямку прямої, що визначається двома кутами повороту, де кут θ відповідає за поворот навколо осі Oz, а кут φ – це кут між Oz та прямою спостерігача.

Рисунок 3.3 показує 3D-проекцію атрактора Лоренца, параметри якого задані вище. Відзначимо решту параметрів, що використовуються в цій візуалізації: точка зору спостерігача (0, 5, 5), кут φ = 0, а кут θ є змінним та дорівнює 330°, 300° та 270° відповідно на рисунках. Окрім атрактора тут показані координатні осі, де зеленим показана нерухома вісь Oz, навколо якої обертається вся проекція.

Важливо зазначити факт, що осі у 3D-проекції мають однакову довжину та складаються з 300 точок. Під час повороту ці осі можуть по-різному проектуватися на екран програми, а деколи взагалі зникати з полотна, якщо спостерігач повернутий в іншому напрямку. Таке викривлення можна помітити на Рисунку 3.4, де змінюється значення кута φ, що відповідно дорівнює 20°, 50° та 80°. Зміна кута φ є дуже незвичним способом візуалізації тривимірного простору, тому потрібна практика, щоб отримувати потрібні результати. Деякі результати зміни цього кута

можуть здаватися неправильними, з чого слідує те, що пряма спостерігача направлена не на атрактор, а в інший бік.

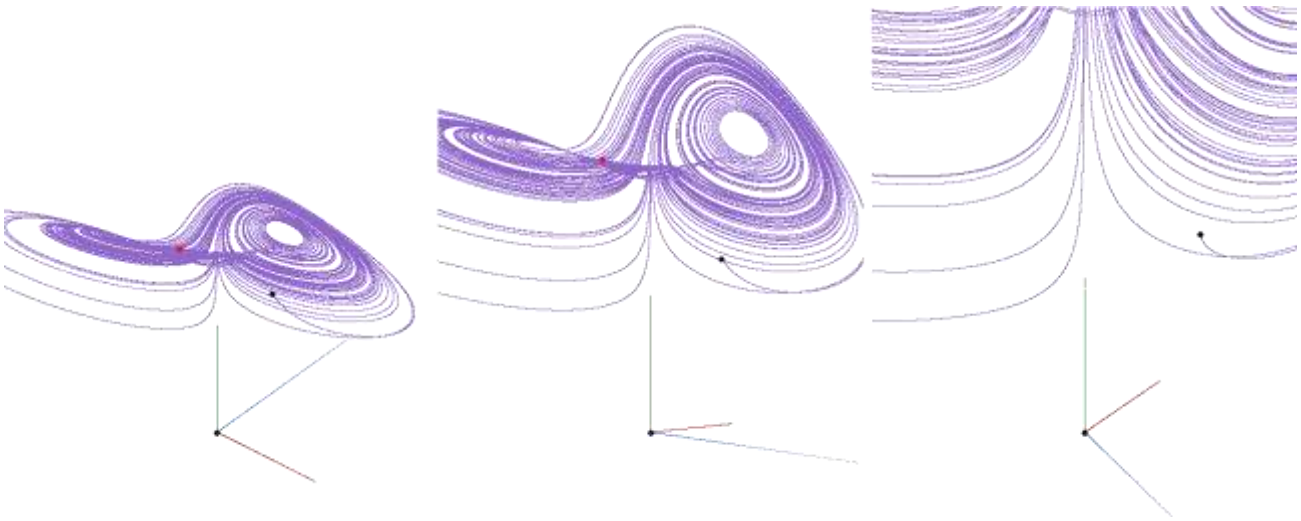


Рисунок 3.4. – Зміна кута φ для атрактора Лоренца

Й останній параметр такої проекції – це точка розташування спостерігача. Пошук оптимальних координат цієї точки – це проблемна задача, оскільки для кожного атрактора вона може набувати різних найкращих значень. Потрібно підбирати як цю точку, так і кути θ та φ, що разом дає цілих п’ять змінних. Деяка порада, що може спростити завдання підбору: визначте за допомогою проекцій на площини розташування атрактора та оберіть координати точки, що знаходиться в стороні від атрактора. Якщо спостерігач буде знаходитися всередині атрактора, побудована проекція буде незрозумілою та складною. Найкращі координати спостерігача знаходяться за межами атрактора та на деякій відстані від нього.

Атрактор	Точка зору			Кут спостерігача	
	x	y	z	θ	ϕ
Лоренц	0	5	5	140°	50°
Ресслер	-1	6	11	0°	70°
Чен-Лі	0	7	11	130°	90°
Дадрас	-2	-2	13	110°	110°

Таблиця 3.1. – Оптимальні значення 3D-проекції

У цьому підрозділі була зроблена спроба знайти комбінації кутів та точки спостерігача, коли атрактор буде виглядати достатньо зрозуміло та наочно, і

результат таких спроб викладено у Таблиці 3.1, а зовнішній вигляд спроектованих атракторів показаний у Додатку Д.

3.3 Побудова орбіт атракторів

Перевіримо вплив деяких параметрів на орбіти атракторів у режимі «Побудова». Візьмемо атрактор Лоренца зі стандартними значеннями a = 10, b = 28, c = 8/3 та початковою точкою (3.051, 1.582, 5.622). Спробуємо визначити допустимі границі параметра dt, steps та залежності між ними, використавши проекцію атрактора на площину xy.

Як показують формули, що використовуються для обрахунку орбіти атрактора (Додаток В, перший стовпчик таблиці під полотном), змінна dt визначає наскільки великі кроки робитиме атрактор після кожної ітерації. Знайдемо межі цієї змінної.

Була проведена спроба підібрати різні значення, щоб знайти верхню межу dt, отримані вписані значення в Таблицю 3.2, а виділені червоним варіанти зображені на Рисунку 3.5 (параметри: steps = 100,000, масштаб = 11). З таблиці можна побачити, що коли dt прямує до одиниці, то кількість точок, які можна побачити на полотні програми, прямує до нуля. Тобто, ітерації атрактора знаходяться занадто далеко одна від одної, щоб вони потрапляли на полотно та будували атрактор.

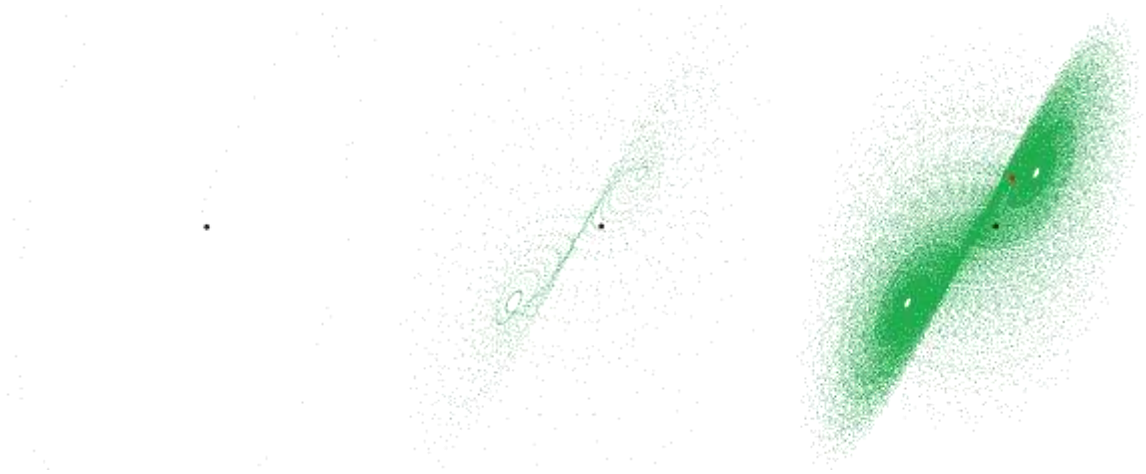


Рисунок 3.5. – Проекції верхніх меж dt

dt	Кількість точок	dt	Кількість точок
0.04	9	0.028143	3120
0.035	12	0.028142	1045
0.03	17	0.028141	833
0.029	22	0.02813	1986
0.0285	23	0.02812	1955
0.0283	27	0.027	3471
0.0282	31	0.026	25509
0.02815	37	0.025	46595
0.028145	46	0.02	44059
0.028144	49	0.01	41552

Таблиця 3.2. – Верхня межа dt

При значенні dt > 0.028143 не помічено, аби ці точки складалися в атрактор Лоренца. Проте коли dt = 0.028143 (Рисунок 3.5, проекція по центру), то уже помітна структура необхідного атрактора. Що більше будемо зменшувати dt, то більше точок опиниться на проекції атрактора. Кількість точок тут також є хаотичною величиною, що можна помітити на дуже близьких значеннях 0.028143, 0.028142 та 0.028141.

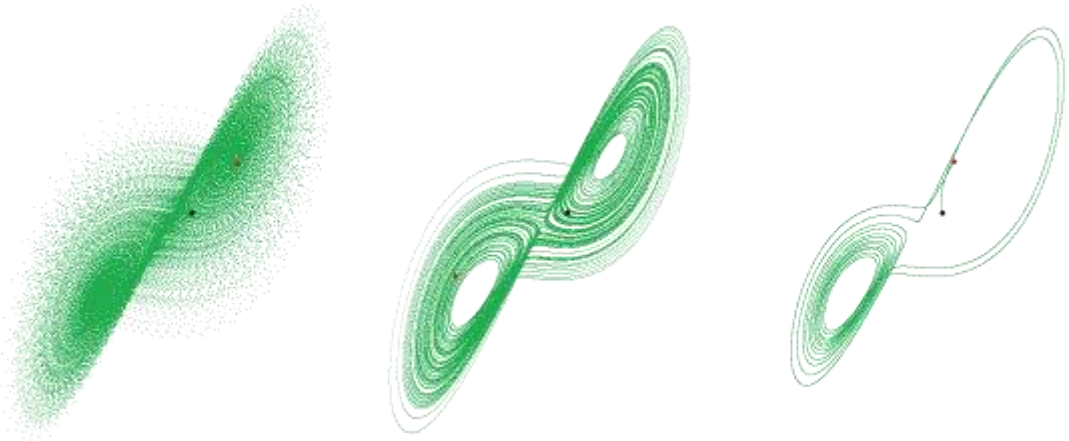


Рисунок 3.6. – Значення dt 0.01, 0.001 та 0.0001

При значенні dt <= 0.026 кількість точок стрімко виростає, і структура атрактора Лоренца стає ще більш помітною. Спробуємо змінити цей параметр на порядки, протестувавши значення 0.01, 0.001 та 0.0001 (Рисунок 3.6). Якщо на першій проекції можна побачити окремі точки, то друга та третя не має окремих точок у тому сенсі, що ці точки знаходяться достатньо близько одна до одної, щоб

злитися у суцільну лінію. На третьому малюнку показаний дуже повільний атрактор, якому не вистачило кількості кроків steps, щоб утворити структуру другого рисунка. Яка ж відповідність між цими змінними?

У Додатку Е наведені деякі приклади атрактора Лоренца з різними значеннями параметрів dt та steps. Розглядаються ті варіанти, коли проекція атрактора складається з суцільних або близьких до цього ліній (не точок) та ці лінії не повинні зливатися в єдину область деякого кольору. Під час експериментів було висунуте припущення, що залежність між цими двома параметрами є такою величиною, що значення dt та steps мають бути у деякому сенсі пропорційними кількості своїх розрядів. Тобто, якщо dt = 0.001 і steps = 100,000, то атрактор з dt = 0.0001 і steps = 1,000,000 також має мати подібний вигляд. Як ми бачимо на цьому рисунку, це припущення підтвердилося, але чи справді це завжди працює?

Ряд експериментів підтвердив, що це працює в багатьох випадках, але цю залежність потрібно розглянути більш глибоко. Наступне припущення було знайти добуток змінних dt та steps перших двох варіантів, і в результаті було отримане число 100.

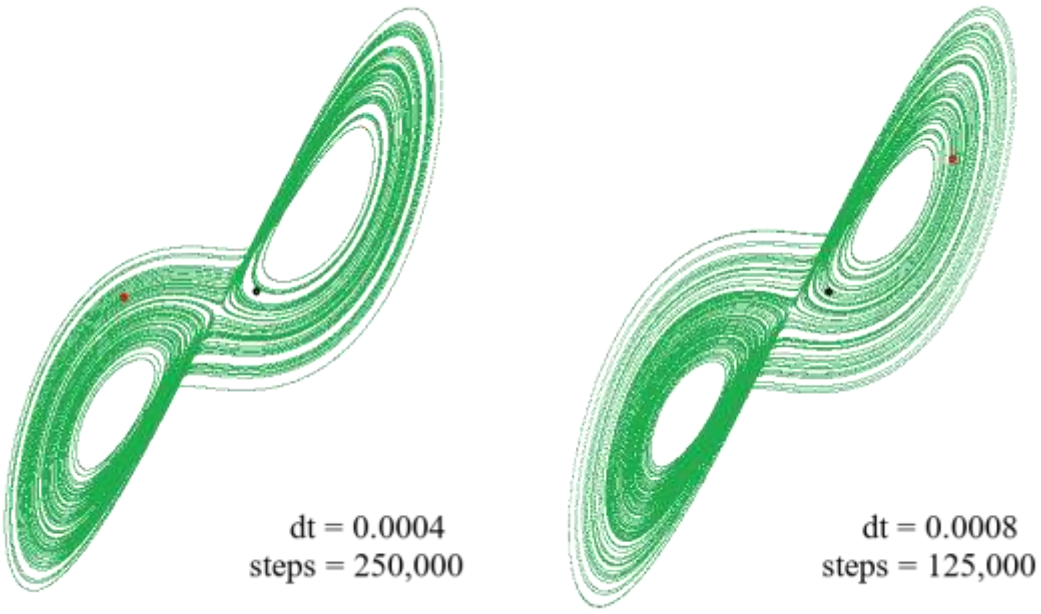


Рисунок 3.7. – Перевірка припущення

Тобто, якщо dt * steps = 100, можна очікувати, що дивний атрактор матиме подібний вигляд з суцільних ліній, що не зливаються. Але ж можна множити не

тільки числа, що починаються з одиниці, щоб отримати 100. Тому був запропонований третій (dt = 0.0005 і steps = 200,000) та четвертий варіант (dt = 0.0002 і steps = 500,000) цих параметрів, а також кілька варіантів Рисунка 3.7.

Таким чином було з’ясовано, що при значенні dt <= 0.001 та умові dt * steps = 100 проекція дивного атрактора матиме найкращий вигляд. Така поведінка буде спостерігатися навіть при кількості кроків у 10,000,000, але тоді надмірно збільшиться час та ресурси на обрахунок орбіт атрактора, тому рекомендовано не перевищувати мільйон кроків обчислень. Явних обмежень на це немає, оскільки користувач може мати більш потужний пристрій, що дозволить робити більшу кількість розрахунків, але потрібно мати на увазі це спостереження.

3.4 Порівняння атракторів

Перевіримо вплив початкової точки на орбіти атракторів у режимі «Порівняння». Візьмемо атрактор Лоренца зі стандартними значеннями a = 10, b = 28, c = 8/3 та початковою точкою (3.051522, 1.582542, 5.623881), та атрактор з такими ж параметрами, але початковою точкою (3.051523, 1.582542, 5.623881). Результат порівняння відображено на Рисунку 3.8.

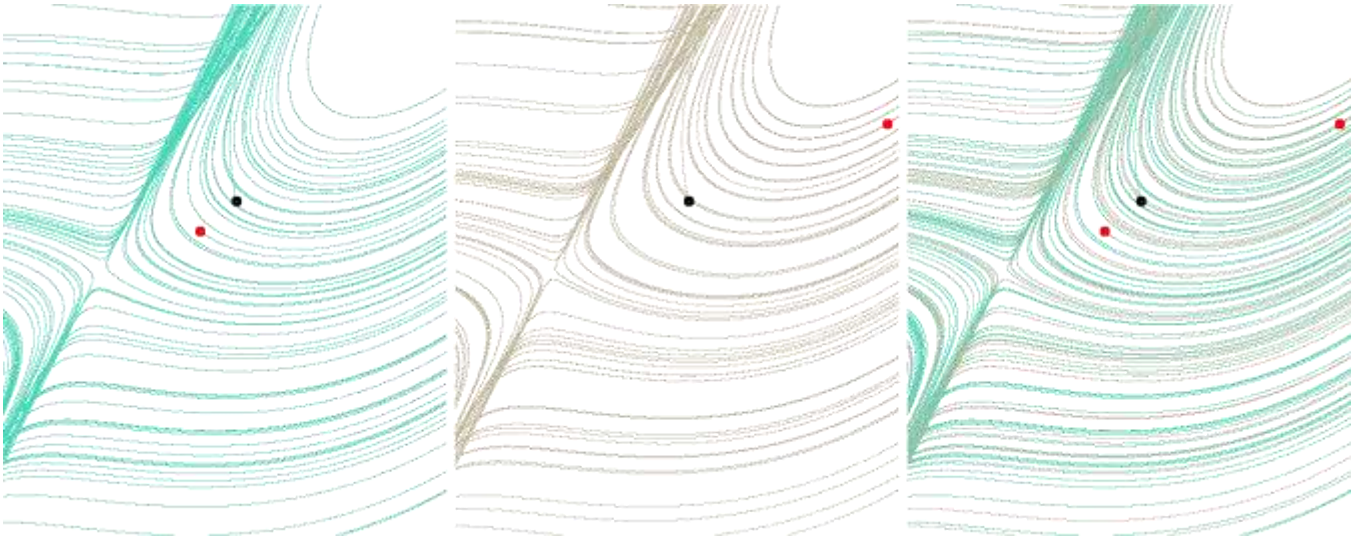


Рисунок 3.8. – Вплив зміни початкової точки

Рисунок 3.9. – Приклад роботи режиму Порівняння

Чорна точка на рисунках – це початкова точка атрактора, а червона – кінцева. Одне лише розташування кінцевої точки показує нам, що практично ідентичні атрактори розійшлися через кількість кроків, що дорівнює 1,000,000. Третя проекція на Рисунку 3.8 – це побудова одразу двох атракторів на одному полотні. З такого можна зробити спостереження, що лінії атракторів не накладаються одна на одну, а розвиваються по-різному, будуючи спочатку схожі, а потім все більш різні атрактори. Проте коли саме відбувається розходження?

На це питання дасть відповідь унікальна функція, що доступна у режимі «Порівняння», – визначення відстані між точками та знаходження моменту, коли останні точки двох атракторів розійшлися більше, ніж на відстань ε, яку задав користувач. На рисунку 3.9 показаний результат таких обчислень, де у четвертому стовпчику таблиці відбувається виведення обчисленої відстані. Програма знайшла точку під номером 107763, коли відстань між атракторами вперше перевищила наперед задане значення ε, що дорівнює 0.001.

ε	dt	Розход-ження	ε	dt	Розход-ження	ε	dt	Розход-ження
0.0001	0.001	7865	0.001	0.001	9466	0.01	0.001	12378
0.0001	0.002	3203	0.001	0.002	3318	0.01	0.002	6029
0.0001	0.005	1115	0.001	0.005	1944	0.01	0.005	2513
0.0001	0.01	576	0.001	0.01	845	0.01	0.01	1055
0.0001	0.02	102	0.001	0.02	187	0.01	0.02	303

Таблиця 3.3. – Розходження двох атракторів Лоренца

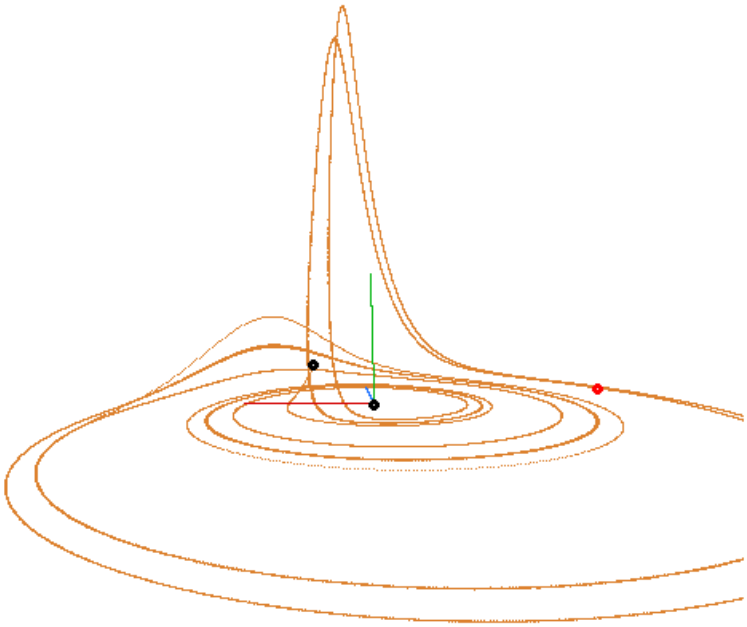
Знайдемо точки розходження для різних значень dt та ε (Таблиця 3.3), де стовпчик «Розходження» – це номер точки, яку знайшла програма. Такий результат дозволяє нам зробити наступні висновки:

- Відстань ε з меншим значенням завжди буде досягнута раніше при будь-яких значеннях dt. Винятком є лише ситуація, якщо вказати ε, яка більше або приблизно дорівнює границям фазового простору атрактора. У такому випадку атрактори ніколи не зможуть перевищити значення ε.
- Збільшення dt завжди призводить до того, що розходження відбудеться раніше. Що більше збільшення, то раніше вони розійдуться.
- Два атрактори Лоренца завжди розійдуться, якщо їхні початкові умови не є однаковими. Проте якщо обрати значення steps меншим, ніж точка розходження, то візуально буде здаватися, що атрактори є однаковими, оскільки деякий час від початку обчислень вони накладаються одне на одного.

В результаті такого аналізу було підтверджено, що атрактор Лоренца має хаотичну поведінку, але чи стосується це інших атракторів, які доступні у програмі Attralyzer? Щоб з’ясувати це, проведемо велике порівняння усіх чотирьох атракторів (Лоренц, Ресслер, Чен-Лі, Дадрас) та внесемо дані у таблицю. Для досягнення такої цілі зафіксуємо деякі значення атракторів та будемо змінювати кілька параметрів для різних атракторів. Нехай перший атрактор має такі параметри: x = 1.396311, y = 2.251014, z = 1.153022, ε = 0.001. Параметри другого атрактора будуть такими ж, але ми будемо змінювати значення x та dt. Результат порівняння винесений у Додаток Ж.

Атрактор Лоренца. Жодних неочікуваних результатів, все відбувається так, як і було прогнозовано та перевірено в експериментах вище. Що більш сильна зміна в початкових даних відбувається, то швидше відбувається розходження, і навпаки.

Атрактор Ресслера. Загалом, тут спостерігається така ж тенденція до змін, але є кілька особливостей. По-перше, на значенні dt = 0.01, коли x = 1.3963112 та x = 1.396312 розходження атракторів взагалі не відбулося (Рисунок 3.10). По логу програми видно, що значення міняються, але 10 мільйонів кроків не вистачило, щоб атрактори з такими параметрами розійшлися (схоже на те, що вони ніколи не розійдуться). Така поведінка спостерігається, коли dt ∈ [0.0091208, 0.0135506]. По-друге, цей атрактор на порядок менш чутливий до змін, ніж атрактор Лоренца та інші два атрактори.



...
8300000 - 0.0000094483;
8400000 - 0.0000102066;
8500000 - 0.0000023823;
8600000 - 0.0000028067;
8700000 - 0.0000031512;
8800000 - 0.0000038414;
8900000 - 0.0000044333;
9000000 - 0.0000053195;
9100000 - 0.0000062332;
9200000 - 0.0000067488;
9300000 - 0.0000082481;
9400000 - 0.0000093479;
9500000 - 0.0000553649;
9600000 - 0.0000029064;
9700000 - 0.0000032396;
9800000 - 0.0000040841;
9900000 - 0.0000044227;
10000000 - 0.0000057027;

Рисунок 3.10. – Атрактор Ресслера, відсутність розходження

Атрактор Чен-Лі. При невеликих змінах цей атрактор, коли dt = 0.001, поводить себе подібно до атрактора Лоренца, але більші зміни або збільшення dt різко прискорюють розходження атракторів. Проте збільшення dt > 0.005 привело до того, що атрактор не будувався правильним чином, а його орбіта прямувала в нескінченність (програма видавала дуже велике число, а потім значення NaN).

Детальніше дослідження виявило, що така поведінка стається, коли dt > 0.0037414. Цей атрактор показаний на Рисунку 3.11.а), де dt = 0.0037414, а x = 1.3963112. Радикальна зміна початкової точки повертає звичний вид атрактора.

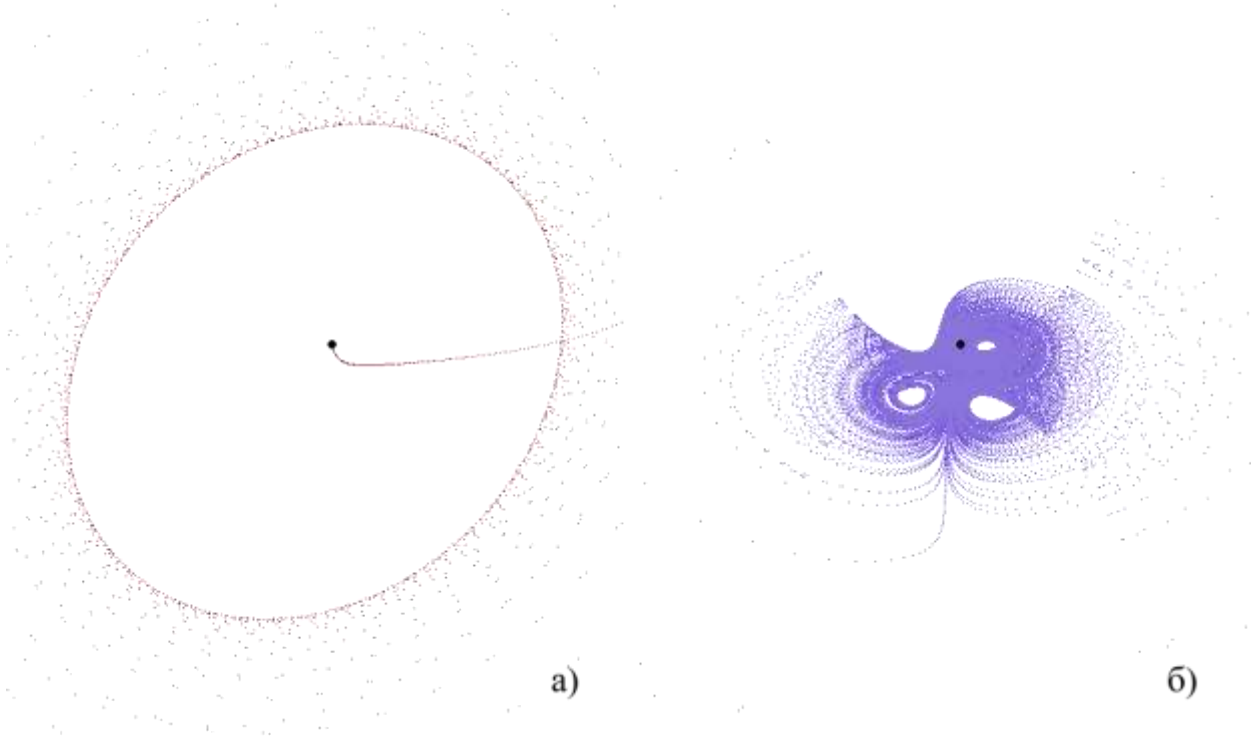


Рисунок 3.11. – Атрактор Чен-Лі (а) та Дадрас (б), проекції на xy

Атрактор Дадрас. Цей атрактор розбігається швидше Лоренца, але на деяких значеннях поступається Чен-Лі. Тут також наявна проблема Чен-Лі, що при деякому значенні dt у ході обчислень з'являється значення NaN. Таблиця каже, що для цього потрібно аби dt > 0.01, але додаткові обчислення показали, що вже при dt = 0.005 з'являється значення NaN приблизно на 5,700,000 кроці. Збільшення dt зменшує кількість кроків, які потрібні для досягнення такого значення. Приклад такого атрактора зі значенням dt = 0.01 та x = 1.3963112 показаний на Рисунку 3.11.б).

3.5 Висновки до Розділу 3

Перевірені основні функції візуалізації. Проілюстрована робота 3D-проекції. Знайдені закономірності між параметрами dt та steps. Доведена хаотична природа дивних атракторів. Проведене порівняння чотирьох атракторів дозволило відкрити деякі обмеження атрактора Ресслера, Чен-Лі та Дадрас.

ВИСНОВКИ

У магістерській роботі проведене комплексне дослідження теоретичних та практичних аспектів динамічних систем та візуалізації орбіт дивних атракторів. Робота охоплює наступні завдання:

1. Проаналізовані основні теоретичні понятя, що пов’язані з нелінійними динамічними системами. Розглянуті характеристики атракторів Лоренца, Ресслера, Чен-Лі та Дадрас. Розглянуті особливості наявних програмних засобів стали основою для подальшого створення нового програмного продукту.
2. Обрано середовище розробки Twine та обґрунтоване використання формату SugarCube. Розроблено програмний продукт Attralyzer, описано функціонал його програмного коду та інтерфейс користувача.
3. Проведена серія експериментів з розробленим програмним продуктом дозволила встановити наступні факти:
   3.1. Знайдені оптимальні значення для 3D-проекції орбіт атракторів.
   3.2. Визначені значення параметра dt, на яких характер візуалізації кардинально змінюється.
   3.3. Виявлена залежність dt * steps = 100, де dt <= 0.001 для побудови орбіт дивних атракторів суцільними лініями.
   3.4. Підтверджена висока чутливість дивних атракторів до початкових умов.
   3.5. Атрактор Ресслера містить ділянки (dt ∈ [0.0091208, 0.0135506]), на яких його поведінка стає періодичною.
   3.6. Деякі набори значень або дуже велика кількість кроків призводять до помилки обчислень у випадку атрактора Чен-Лі та Дадрас.

Перспективи: збільшення кількості дивних атракторів, що доступні для візуалізації, додавання нових можливостей та режимів, оптимізація алгоритмів та інтерфейсу, інтеграція з іншими засобами аналізу.

Використання: дослідження менш відомих дивних атракторів, детальний аналіз впливу параметрів на орбіти атракторів, побудова хаотичних моделей, інструмент для навчальних цілей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alsedà L., Costa S. On the definition of strange nonchaotic attractor. *Fundamenta mathematicae*. 2009. Vol. 206. P. 23–39. URL: https://doi.org/10.4064/fm206-0-2 (date of access: 02.06.2024).
2. Alternative implementation of the chaotic Chen–Lee system / L.-J. Sheu et al. *Chaos, solitons & fractals*. 2009. Vol. 41, no. 4. P. 1923–1929. URL: https://doi.org/10.1016/j.chaos.2008.07.053 (date of access: 02.06.2024).
3. Baishya C., Veeresha P. Dynamics of the Dadras-Momeni system in the frame of the Caputo-Fabrizio fractional derivative. *Special functions in fractional calculus and engineering*. Boca Raton, 2023. P. 241–269. URL: https://doi.org/10.1201/9781003368069-11 (date of access: 02.06.2024).
4. Bourke P. Fractals, chaos, self similarity. *paulbourke.net*. URL: https://paulbourke.net/fractals/ (date of access: 02.06.2024).
5. Breadloafsky. Phase space. *GitHub*. URL: https://github.com/breadloafsky/phase-space (date of access: 02.06.2024).
6. Chapbook introduction. *GitHub Pages*. URL: https://klembot.github.io/chapbook/guide/ (date of access: 02.06.2024).
7. Chen H.-K., Lee C.-I. Anti-control of chaos in rigid body motion. *Chaos, solitons & fractals*. 2004. Vol. 21, no. 4. P. 957–965. URL: https://doi.org/10.1016/j.chaos.2003.12.034 (date of access: 02.06.2024).
8. Christensen M. H. Strange attractors. *GitHub*. URL: https://syntopia.github.io/StrangeAttractors/ (date of access: 02.06.2024).
9. Complete synchronization of two Chen-Lee systems / L.-J. Sheu et al. *Journal of physics: conference series*. 2008. Vol. 96. P. 012138. URL: https://doi.org/10.1088/1742-6596/96/1/012138 (date of access: 02.06.2024).
10. Dadras attractor. *Dynamic Mathematics*. URL: https://www.dynamicmath.xyz/calculus/velfields/Dadras/ (date of access: 02.06.2024).

11. Dadras S., Momeni H. R. A novel three-dimensional autonomous chaotic system generating two, three and four-scroll attractors. *Physics letters A*. 2009. Vol. 373, no. 40. P. 3637–3642. URL: https://doi.org/10.1016/j.physleta.2009.07.088 (date of access: 02.06.2024).
12. Design and analysis bidirectional chaotic synchronization of Rossler circuit and its application for secure communication / A. Sambas et al. *Applied mathematical sciences*. 2013. Vol. 7. P. 11–21. URL: https://doi.org/10.12988/ams.2013.13002 (date of access: 02.06.2024).
13. Devaney R. L. First course in chaotic dynamical systems. Taylor & Francis Group, 2020. 318 p.
14. Harlowe 3.3.8 manual. *Harlowe 3.3.8 manual*. URL: https://twine2.neocities.org/ (date of access: 02.06.2024).
15. Inspection on ball bearing malfunction by Chen-Lee chaos system / C.-J. Lin et al. *IEEE access*. 2020. Vol. 8. P. 28267–28275. URL: https://doi.org/10.1109/access.2020.2971554 (date of access: 02.06.2024).
16. Kapitanov A. Генераторы хаоса на FPGA. *Хабр*. URL: https://habr.com/ru/articles/273915/ (дата звернення: 02.06.2024).
17. Liu W., Chen G. A new chaotic system and its generation. *International journal of bifurcation and chaos*. 2003. Vol. 13, no. 01. P. 261–267. URL: https://doi.org/10.1142/s0218127403006509 (date of access: 02.06.2024).
18. Lorenz E. N. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*. 1963. Vol. 20, no. 2. P. 130–141. URL: https://doi.org/10.1175/1520-0469(1963)020%3C0130:dnf%3E2.0.co;2 (date of access: 02.06.2024).
19. Malinetskii G. G., Klochkov A. K. Simulation of oscillatory chemical reactions with chaotic behavior. *Keldysh institute preprints*. 2020. No. 12. P. 1–18. URL: https://doi.org/10.20948/prepr-2020-12 (date of access: 02.06.2024).
20. Meier J. A gallery of strange attractors in webxr. *Analytic Physics*. URL: https://analyticphysics.com/Catastrophe%20Theory/A%20Gallery%20of%20Strange%20Attractors%20in%20WebXR.htm (date of access: 02.06.2024).

21. Nguyen B. G. Strange attractors. *brandon.nguyen.vc*. URL: https://brandon.nguyen.vc/2021/06/attractors/ (date of access: 02.06.2024).
22. Nonlinear Chen-Lee chaotic system based deep convolutional generative adversarial nets for chatter diagnosis / P.-H. Kuo et al. 2021. URL: https://doi.org/10.21203/rs.3.rs-1066378/v1 (date of access: 02.06.2024).
23. Pchelintsev A. N. A numerical-analytical method for constructing periodic solutions of the Lorenz system. *Electronic journal*. 2020. No. 4.
24. Piellardj. Performant 2D strange attractors plotter. *GitHub*. URL: https://github.com/piellardj/strange-attractors-webgl (date of access: 02.06.2024).
25. Pietri S. Strange attractors GPU. *OPENFUSE**. URL: https://fusefactory.github.io/openfuse/strange%20attractors/particle%20system/Strange-Attractors-GPU/ (date of access: 02.06.2024).
26. PurpleCat. Visualization of 3D strange attractors. *Reddit*. URL: https://www.reddit.com/r/math/comments/z0dmms/visualization_of_3d_strange_attractors/?rdt=60708 (date of access: 02.06.2024).
27. Reusser R. Strange attractors on the GPU, Part 2: Fun!. *Observable*. URL: https://observablehq.com/@rreusser/strange-attractors-on-the-gpu-part-2 (date of access: 02.06.2024).
28. Ricky Reusser. Strange attractors. *github.io*. URL: https://rreusser.github.io/strange-attractors (date of access: 02.06.2024).
29. Rössler O. E. An equation for continuous chaos. *Physics letters A*. 1976. Vol. 57, no. 5. P. 397–398. URL: https://doi.org/10.1016/0375-9601(76)90101-8 (date of access: 02.06.2024).
30. Ruelle D., Takens F. On the nature of turbulence. *Communications in mathematical physics*. 1971. Vol. 20, no. 3. P. 167–192. URL: https://doi.org/10.1007/bf01646553 (date of access: 02.06.2024).
31. Sketches by jcponcemath. *OpenProcessing*. URL: https://openprocessing.org/user/91533?view=sketches&o=48 (date of access: 02.06.2024).

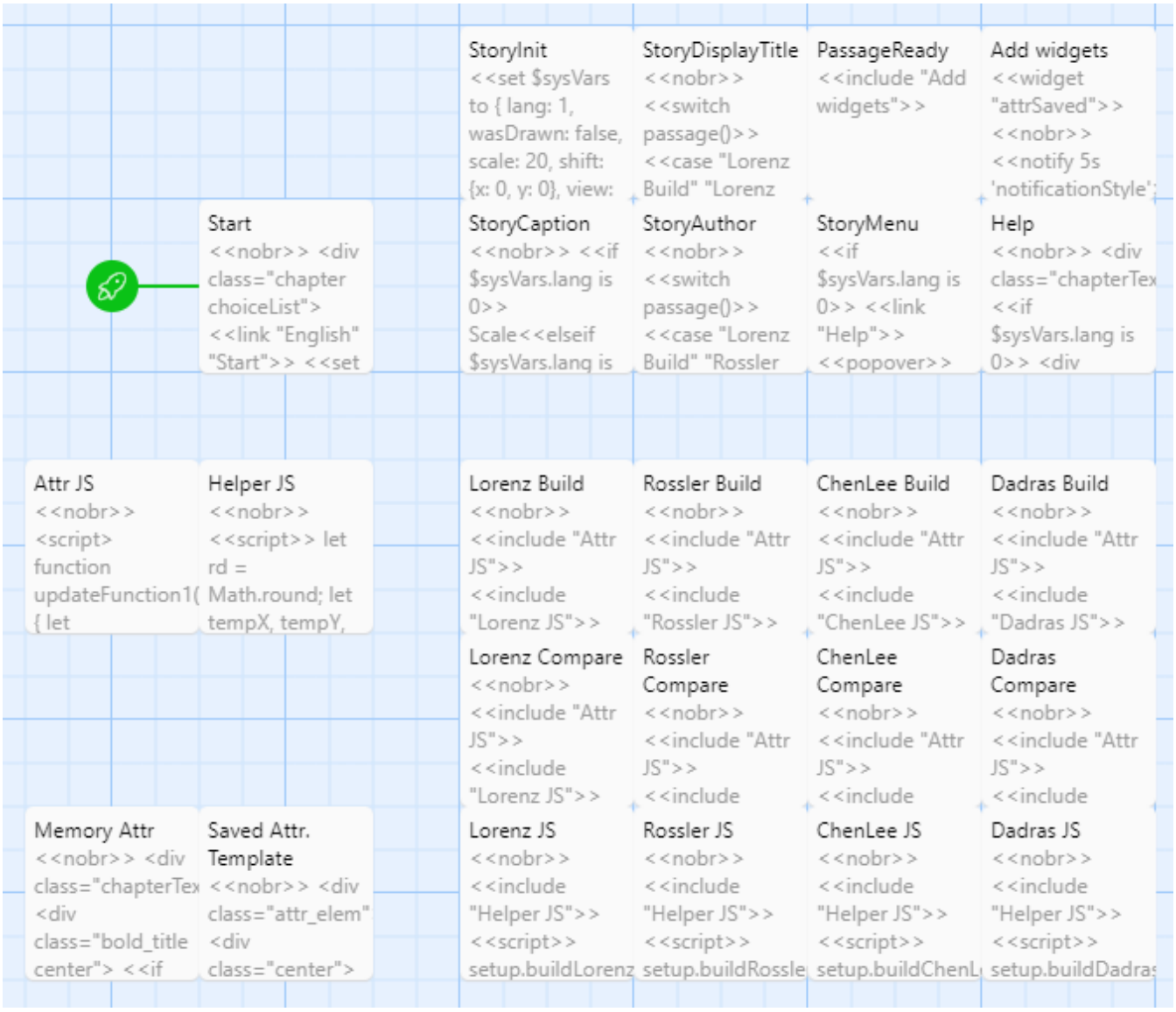
32. Snowman. *Dr. Daniel Cox*. URL: https://videlais.github.io/snowman/#/ (date of access: 02.06.2024).
33. Strogatz S. H. Nonlinear dynamics and chaos. 2nd ed. Boca Raton : CRC Press, 2018. 532 p. URL: https://doi.org/10.1201/9780429492563 (date of access: 02.06.2024).
34. SugarCube. *Motoslave.net*. URL: http://www.motoslave.net/sugarcube/2/ (date of access: 02.06.2024).
35. Twine Cookbook. *Twinery.org*. URL: https://twinery.org/cookbook/index.html (date of access: 02.06.2024).
36. Vdesmond. Attractors. *GitHub*. URL: https://github.com/vdesmond/attractors (date of access: 02.06.2024).
37. Voytsik. Attralyzer. *itch.io*. URL: https://voytsik.itch.io/attralyzer (date of access: 02.06.2024).
38. Wshahbaz. Strange attractor generator. *GitHub*. URL: https://github.com/wshahbaz/Strange_Attractors (date of access: 02.06.2024).
39. Бекман И. Н. Синергетика. Курс лекций. Москва : Междисциплинарный университет Бекм., 2010.
40. Бекман И. Н. ф. Нелинейная динамика сложных систем: теория и практика. Москва : Московс. Государственный Университет им. М.В. Ломонос., 2018. 613 с.
41. Войцеховський Є. О. Візуалізація дивних атракторів: реалізація засобами Javascript. Матеріали XII Всеукраїнської Наукової Конференції Молодих Математиків, 9–11 трав. 2024 р. С. 62.
42. Данилов В. Я., Зінченко А. Ю. Синергетичні методи аналізу практикум. Київ : КПІ ім. Ігоря Сікорського, 2023. 113 с.
43. Деменков Д. А. Синергетический метод обработки и защиты информации на основе реконструкции системы типа Рёсслера. *Известия ЮФУ. Технические науки*. 2008. Т. 88, № 11. URL: https://cyberleninka.ru/article/n/sinergeticheskiy-metod-obrabotki-i-

zaschity-informatsii-na-osnove-rekonstruktsii-sistemy-tipa-resslera (дата звернення: 02.06.2024).

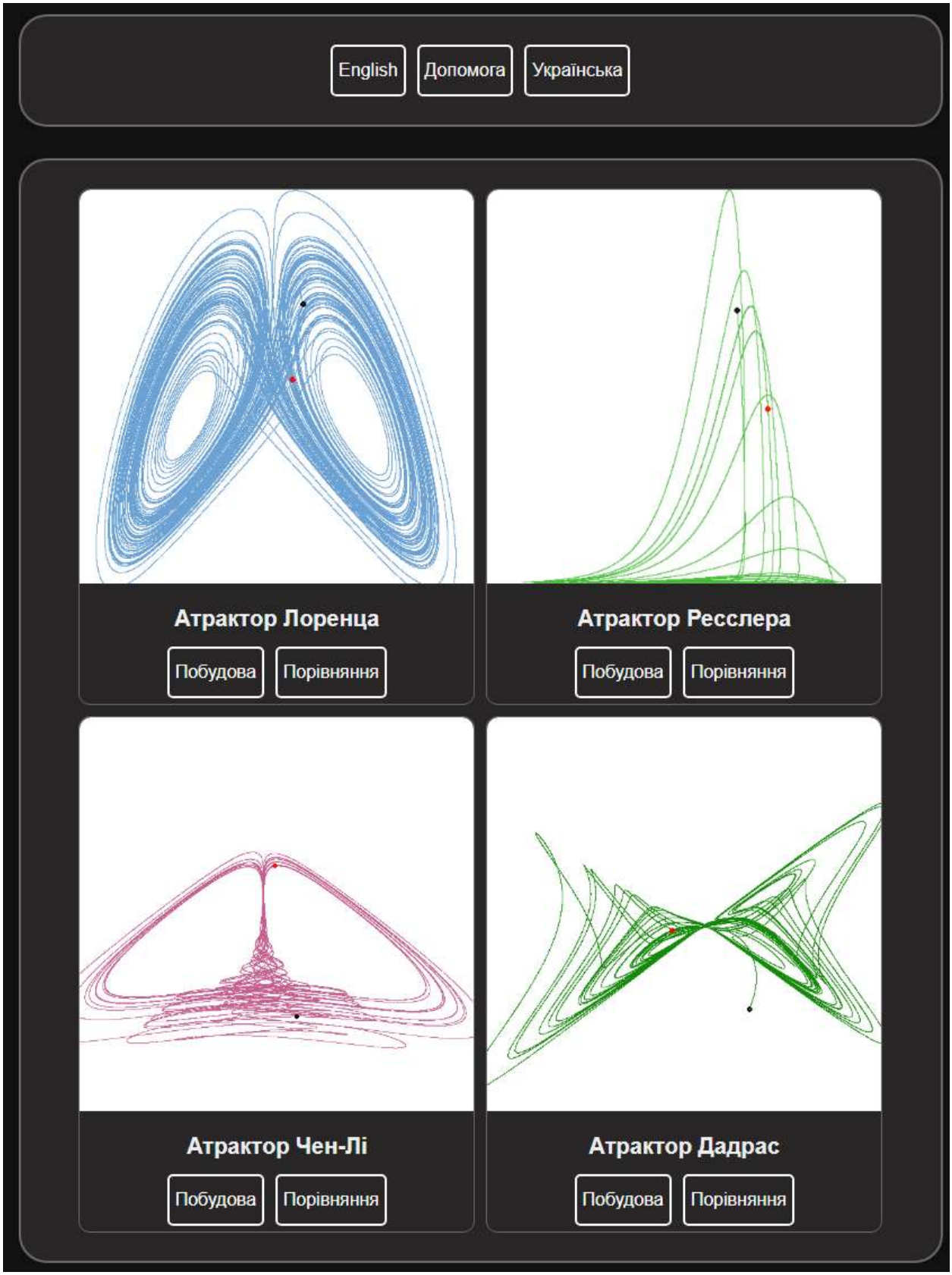
44. Ильичев В. Разработка программы для исследования аттрактора Лоренца и ее использование. *Сложные системы*. 2021. Т. 1, № 38. С. 56–63.
45. Ильичев И. Создание программных средств исследования аттрактора Рёсслера. *International journal of humanities and natural sciences*. 2021. Т. 5-1.
46. Кучерова В., Петьков В., Артамонов П. Применение метода АКАР для решения задачи стабилизации состояний равновесия типовых нелинейных систем. *Фундаментальные исследования*. 2016. № 5. С. 264–268.
47. Немнюгин С. А. Turbo pascal. Санкт-Петербург; Москва : Питер, 2000. 496 р.
48. Широкова О. А. Обучения учащихся объектно-ориентированному программированию при создании объектов фрактальной графики средствами delphi. *Математическое образование в школе и вузе: теория и практика (MATHEDU - 2016)* : материалы VI Международной научно-практической конференции, м. Казань, 25–26 листоп. 2016 р.

ДОДАТКИ

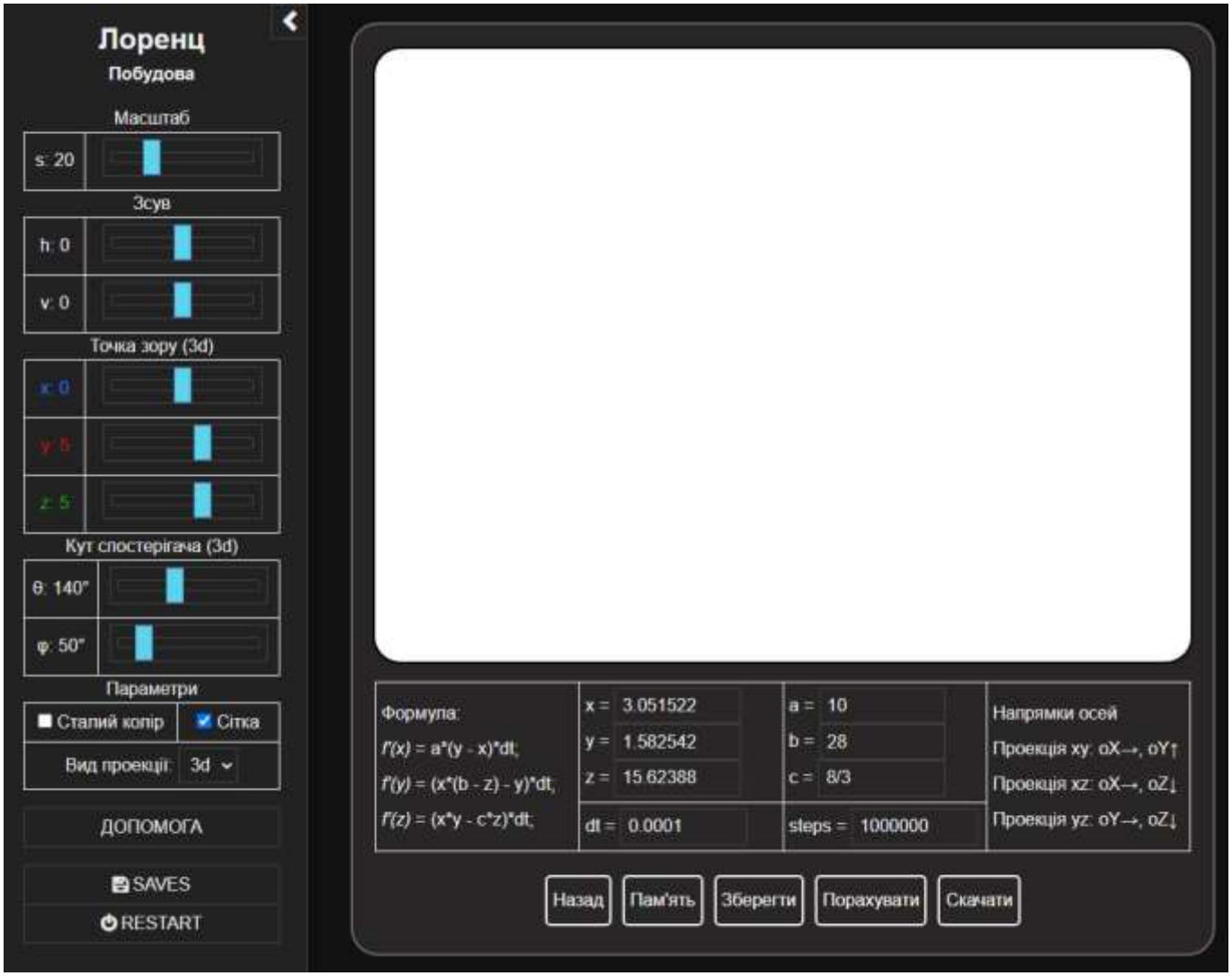
Додаток А. – Структура уривків програми



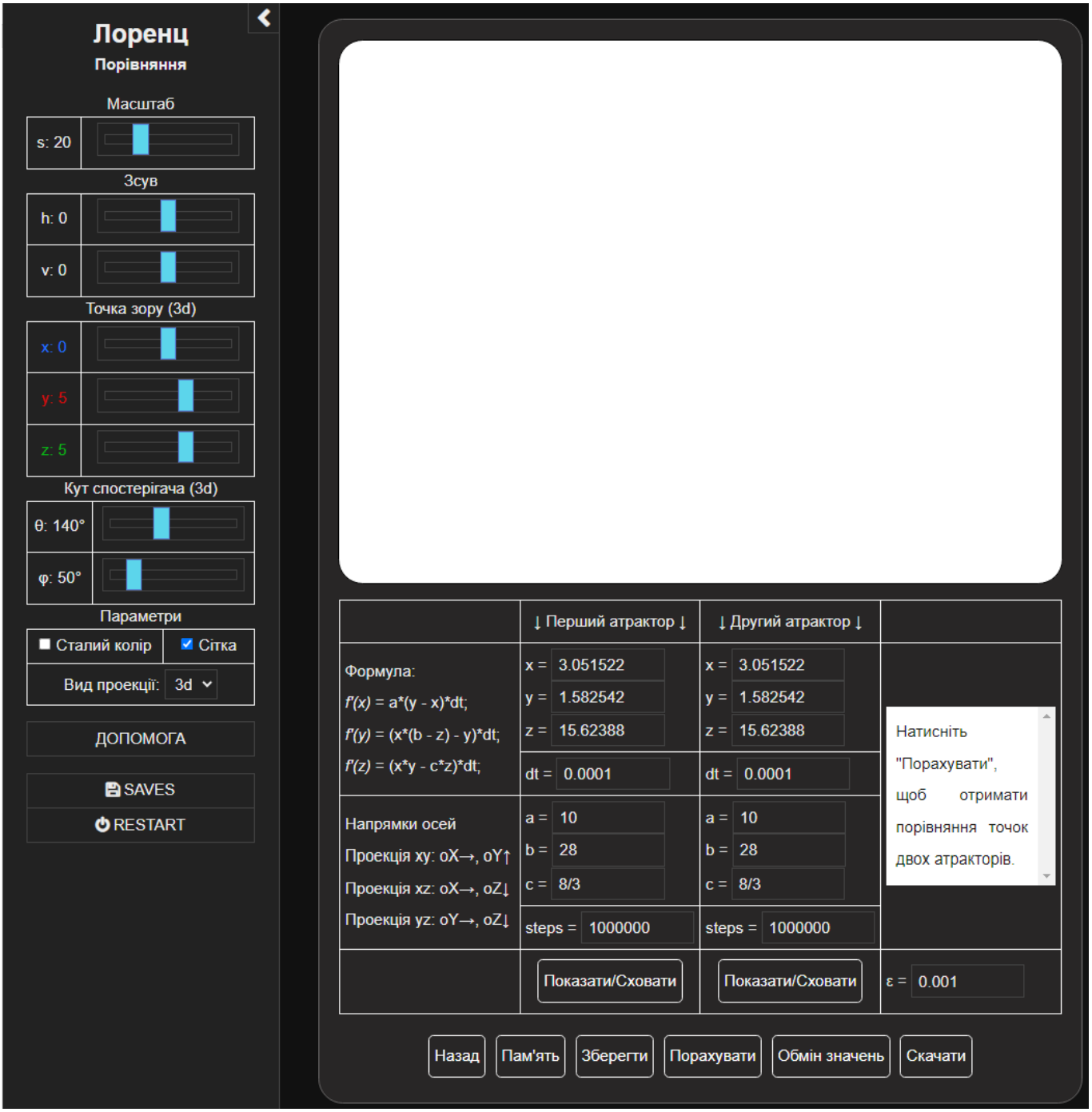
Додаток Б. – Початкова сторінка програми



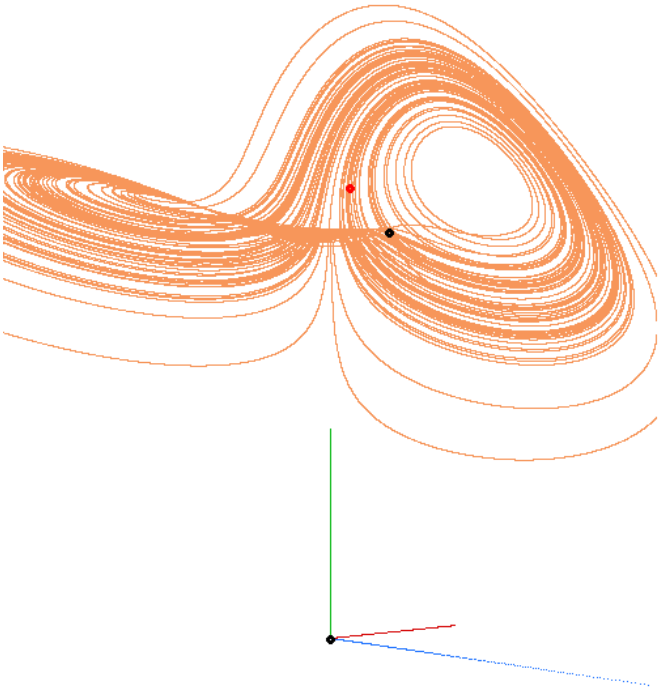
Додаток В. – Режим Побудова, атрактор Лоренца

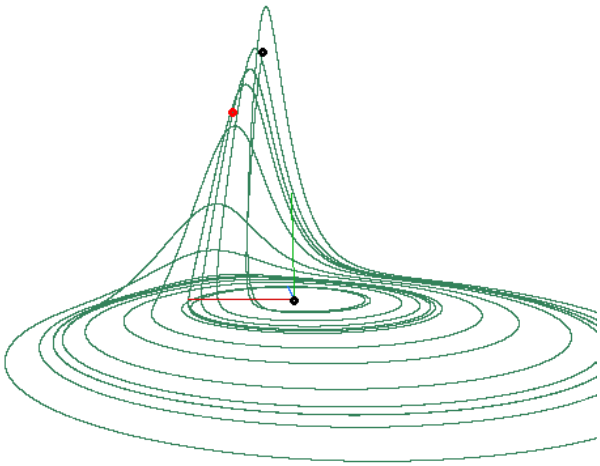


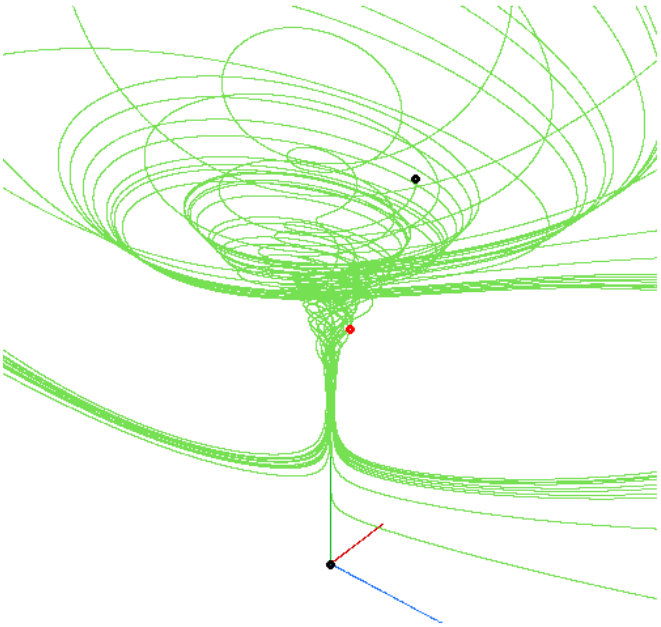
Додаток Г. – Режим Порівняння, атрактор Лоренца

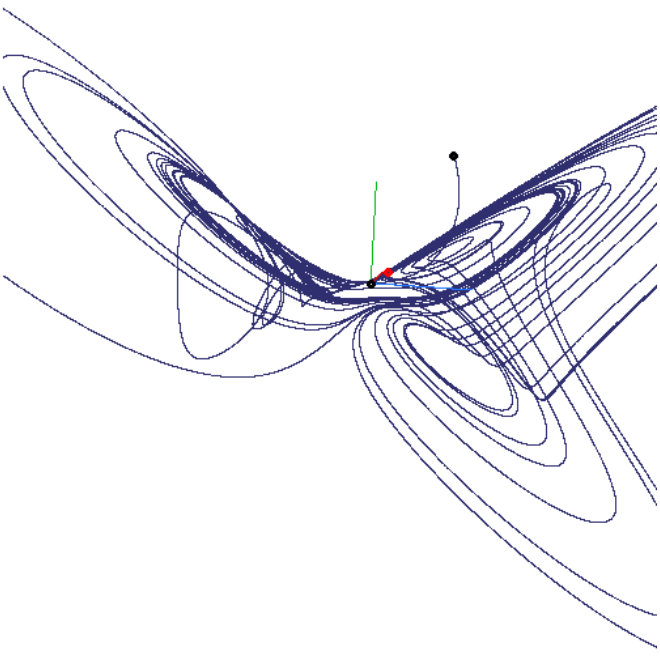


Додаток Д. – Найкращі 3D-проекції атракторів

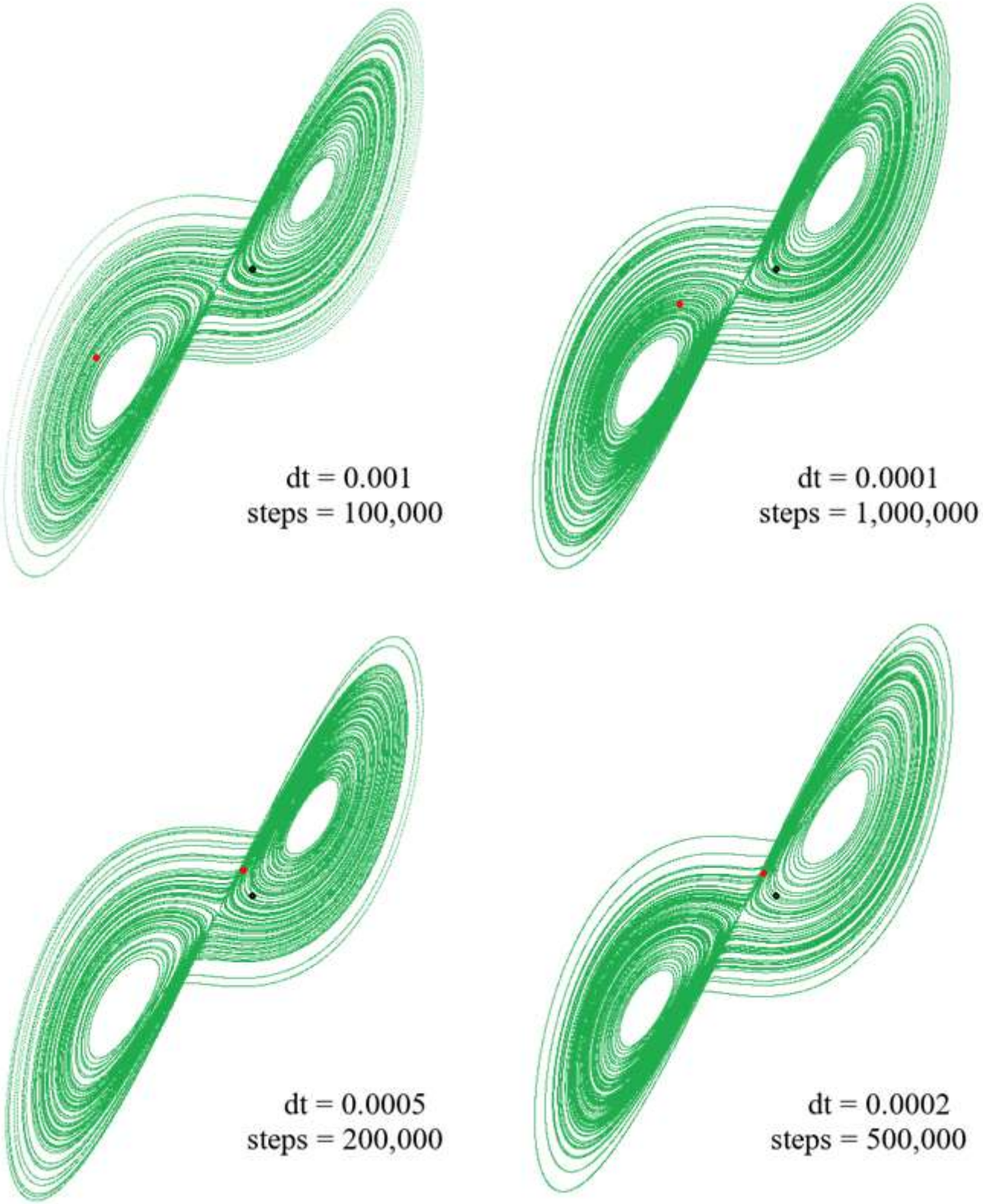








Додаток Е. – Залежність між dt та steps



Додаток Ж. – Перевірка атракторів на чутливість до змін

Лоренц		
x	**dt**	**Розход-ження**
1.3963112	0.001	18504
1.3963112	0.002	9506
1.3963112	0.005	2312
1.3963112	0.01	821
1.3963112	0.02	394
1.396312	0.001	14970
1.396312	0.002	8372
1.396312	0.005	2277
1.396312	0.01	804
1.396312	0.02	326
1.39632	0.001	14728
1.39632	0.002	6931
1.39632	0.005	1969
1.39632	0.01	537
1.39632	0.02	281

Ресслер		
x	**dt**	**Розход-ження**
1.3963112	0.001	140378
1.3963112	0.002	108192
1.3963112	0.005	31378
1.3963112	0.01	нема
1.3963112	0.02	3477
1.396312	0.001	105613
1.396312	0.002	58591
1.396312	0.005	24399
1.396312	0.01	нема
1.396312	0.02	2620
1.39632	0.001	70424
1.39632	0.002	43925
1.39632	0.005	14039
1.39632	0.01	5250
1.39632	0.02	1756

Чен-Лі		
x	**dt**	**Розход-ження**
1.3963112	0.001	18273
1.3963112	0.002	6928
1.3963112	0.005	342
1.3963112	0.01	116
1.3963112	0.02	53
1.396312	0.001	13358
1.396312	0.002	6758
1.396312	0.005	328
1.396312	0.01	111
1.396312	0.02	50
1.39632	0.001	776
1.39632	0.002	389
1.39632	0.005	157
1.39632	0.01	80
1.39632	0.02	42

Дадрас		
x	**dt**	**Розход-ження**
1.3963112	0.001	15139
1.3963112	0.002	6588
1.3963112	0.005	1518
1.3963112	0.01	1478
1.3963112	0.02	783
1.396312	0.001	5281
1.396312	0.002	2706
1.396312	0.005	1031
1.396312	0.01	1069
1.396312	0.02	520
1.39632	0.001	3345
1.39632	0.002	1731
1.39632	0.005	772
1.39632	0.01	521
1.39632	0.02	223