

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

_____ А.І.Петренко
(підпис) (ініціали, прізвище)

“ _____ ” _____ 20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напрямку підготовки

6.050101 Комп'ютерні науки
(код і назва)

на тему: Моделювання еволюції екосистеми на базі рівнянь Вольтерри-Лотки

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-31
(шифр групи)

Петенок Олена Володимирівна

(прізвище, ім'я, по батькові)

_____ (підпис)

Керівник доцент, к.т.н., Булах Б.В.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант Економічний доцент, к.е.н., Рощина Н.В.

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали)

_____ (підпис)

Рецензент доц. каф. ММСА, к.т.н., доцент, Тимошук О.Л.

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

_____ (підпис)

Нормоконтроль старший викладач, Бритов О.А.

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2017 року

**Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»**

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрямок підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«__» _____ 20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Петенок Олені Володимирівні

(прізвище, ім'я, по батькові)

1. Тема роботи Моделювання еволюції екосистеми на базі рівнянь
Вольтерри-Лотки

керівник роботи Булах Богдан Вікторович, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» травня 2017 р. № 1477-с

2. Термін подання студентом роботи 09.06.2017.

3. Вихідні дані до роботи Моделювання екосистем з урахуванням довільної
кількості видів тварин та рослин та їх взаємозв'язків, застосування рівнянь
Вольтерра-Лотки.

4. Зміст роботи _____

1. Огляд існуючих рішень та предметної області.

2. Вибір засобів реалізації.

3. Програмна реалізація та аналіз результатів.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) _____

1. UMLдіаграма класів – плакат.

2. Модель Вольтерри-Лотки – плакат.

3. Результати роботи програмного продукту – плакат.

3. Презентація.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічно-організаційна частина	Рощина Н.В., к.е.н.		

7. Дата видачі завдання 01.02.2017.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Планування дипломного проекту	15.02.2017	
3	Дослідження існуючих рішень	28.02.2017	
4	Вибір математичної моделі	15.03.2017	
5	Реалізація програмного продукту	15.04.2017	
6	Тестування програмного продукту	30.04.2017	
7	Аналіз результатів роботи	15.05.2017	
8	Оформлення роботи	30.05.2017	
9	Проходження нормо-контролю	10.06.2017	

Студент

(підпис)

О.В. Петенок

(ініціали, прізвище)

Керівник роботи

(підпис)

Б.В. Булах

(ініціали, прізвище)

*Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Петенок Олени Володимирівни на тему
«Моделювання еволюції екосистеми на базі рівнянь Вольтерри-Лотки»

Метою дипломної роботи є створення програмного продукту, що забезпечував би імітаційне моделювання екосистем. Об'єктом дослідження є моделювання еволюції екосистеми на базі рівнянь Вольтерри-Лотки. Було виконано огляд існуючих аналогів готового продукту дипломної роботи. Створено програмний продукт для моделювання еволюції екосистеми за модифікованою моделлю Вольтерри-Лотки. Програмний продукт може бути використаний екологами для аналізу проблем екосистем, наприклад, небезпеки промислового вилову тварин, і прийняття рішень щодо можливих рішень таких проблем.

Загальний обсяг роботи: 87 сторінок, 58 ілюстрацій, 6 таблиць та 8 посилань.

Ключові слова: математична модель, імітаційна модель, популяційна динаміка, трофічна піраміда, екосистема, модифікована модель Вольтерри-Лотки.

АННОТАЦИЯ

бакалаврской дипломной работы Петенок Елены Владимировны на тему «Моделирование эволюции экосистемы на базе уравнений Вольтерры-Лотки»

Целью дипломной работы является создание программного продукта, который обеспечивал бы имитационное моделирование экосистем. Объектом исследования является моделирование эволюции экосистемы на базе уравнений Вольтера-Лотки. Выполнен обзор существующих аналогов готового продукта дипломной работы. Создан программный продукт для моделирования эволюции экосистемы на основе модифицированной модели Вольтерры-Лотки. Программный продукт может быть использован экологами для анализа проблем экосистем, например, опасности промышленного отлова животных, и принятие решений о возможных решениях таких проблем.

Общий объем работы: 87 страниц, 58 иллюстраций, 6 таблиц и 8 ссылок.

Ключевые слова: математическая модель, имитационная модель, популяционная динамика, трофическая пирамида, экосистема, модифицированная модель Вольтерры-Лотки.

ABSTRACT

to the bachelor thesis by Petenok Olena Volodymyrivna on “Modeling of the evolution of the ecosystem based on the Volterra-Lotka equations”

The aim of the thesis is to create ecosystem simulation software. The object of the study is modeling of the evolution of ecosystems based on Volterra-Lotka equations. There have been written review of existing software analogues. Was created the software product for simulating the evolution of ecosystems based on modified Volterra-Lotka equations. The software can be used by ecologists to analyze ecosystem's problems, such as the danger of industrial catch animals, and decide on possible solutions of these problems.

Includes: 87 pages, 58 illustrations, 6 tables and 8 references.

Keywords: mathematical model, simulation model, population dynamics, trophic pyramid, ecosystem, modified Volterra-Lotka model.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ.....	10
ВСТУП	11
1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ.....	13
1.1 Передмова.....	13
1.2 Моделювання в екології.....	13
1.2.1 Моделі популяційної динаміки	13
1.2.2 Імітаційне моделювання.....	14
1.2.3 Моделі міжвидової конкуренції	15
1.3 Модель Вольтерри-Лотки	15
1.3.1 Припущення про компоненти системи.....	15
1.3.2 Трофічні піраміди	16
1.3.3 Рівняння двовимірної моделі (хижак-жертва).....	17
1.3.4 Стаціонарна позиція двовимірної моделі.....	18
1.3.5 Матриця взаємозв'язків.....	18
1.3.6 Узагальнене рівняння моделі.....	19
1.4 Огляд існуючих програмних рішень	19
1.4.1 The Ecorpath with Ecosim (EWE) approach.....	19
1.4.2 Ecolego	21
1.5 Висновки.....	23
2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ.....	24
2.1 Модифікації моделі Вольтерри-Лотки	24
2.1.1 Матриця коефіцієнтів	24

	8
2.1.2	Внесення модифікацій в рівняння моделі 24
2.1.3	Явний метод Ейлера для розв'язку системи рівнянь 26
2.2	Вибір програмних засобів..... 28
2.2.1	Java..... 28
2.2.2	JDK 8 30
2.2.3	IntelliJ IDEA (CommunityEdition) 2017.1.3 30
2.2.4	Swing 30
2.2.5	JFreeChart 1.0.19 31
2.3	Висновки..... 32
3	РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ 33
3.1	Структура програмного продукту 33
3.1.1	UML-діаграма класів 33
3.1.2	Опис класівSwing, що були використані..... 33
3.1.3	Опис класів JFreeChart, що були використані 36
3.1.4	Опис класів логіки та обробки даних 37
3.1.5	Опис класів GUI..... 39
3.2	Результати роботи програмного продукту для моделювання еволюції екосистеми лісу 41
3.2.1	Обмеження можливих результатів..... 41
3.2.2	Опис моделюваної екосистеми..... 42
3.2.3	Результати моделювання екосистеми за умови нестачі трави.. 42
3.2.4	Результати моделювання екосистеми за нормальних умов 49
3.2.5	Результати моделювання екосистеми за умови істотного промислового видобутку оленів та відстрілу вовків..... 55
3.3	Висновки..... 63

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	64
4.1 Вступ	64
4.2 Постановка задачі техніко-економічного аналізу	65
4.2.1 Обґрунтування функцій програмного продукту	66
4.2.2 Варіанти реалізації основних функцій	67
4.3 Обґрунтування системи параметрів ПП.....	69
4.3.1 Опис параметрів.....	69
4.3.2 Кількісна оцінка параметрів	70
4.3.3 Аналіз експертного оцінювання параметрів	73
4.4 Аналіз рівня якості варіантів реалізації функцій	77
4.5 Економічний аналіз варіантів розробки ПП	78
4.6 Вибір кращого варіанта ПП техніко-економічного рівня	83
4.7 Висновки.....	83
ВИСНОВКИ.....	85
ПЕРЕЛІК ПОСИЛАНЬ	87

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

GUI	Graphic User Interface (графічний інтерфейс користувача).
Екосистема	Сукупність живих організмів, які пристосувалися до спільного проживання в певному середовищі існування, утворюючи з ним єдине ціле.
Імітаційна модель	Логіко-математичний опис об'єкта, який може бути використаний для експериментування на комп'ютері.
Модель Вольтерри-Лотки	Система диференціальних рівнянь першого порядку, яка описує кінетику чисельності популяцій.
Матриця взаємозв'язків	Повну структуру парних взаємодій між n видами можна зобразити за допомогою матриці з $n \times n$ елементів. Елемент (i, j) являє собою 1, -1 або 0 і показує вплив i -го виду на j -й.
Матриця коефіцієнтів	Повну структуру парних взаємодій між n видами можна зобразити за допомогою матриці з $n \times n$ елементів. Елемент (i, j) являє собою коефіцієнт виїдання j -го виду i -м.
IDE	Integrated Development Environment — Інтегроване середовище розробки.

ВСТУП

Бажання передбачати речі та впливати на майбутнє – невід’ємна частина людини з давніх давен. З розвитком комп’ютерних технологій передбачення стало певною мірою можливим, за рахунок імітаційного моделювання. Ця тема є актуальною і в наші дні.

Імітаційне моделювання — це метод, що дозволяє будувати моделі процесів, що описують, як ці процеси проходили б насправді. Імітаційне моделювання засноване на тому, що система, яка вивчається, замінюється імітатором і з ним проводяться експерименти з метою отримання інформації про цю систему.

При імітаційному моделюванні спочатку будується математична модель системи, а потім в ній запускаються різноманітні, можливі для цієї системи, процеси. Таким чином можна спостерігати реакцію системи на різні вхідні дані та робити висновки про поведінку реальної системи за модельованим прототипом, з певною вірогідністю.

Метою дипломної роботи є створення програмного забезпечення, що забезпечувало б імітаційне моделювання екосистем. Математичною моделлю виступає модифікована система рівнянь Вольтерри-Лотки для довільної кількості видів. Поставлена мета передбачає вирішення наступних задач:

- Аналіз предметної області;
- Аналіз існуючого програмного забезпечення для імітаційного моделювання;
- Модифікація моделі Вольтерри-Лотки за рахунок додавання до неї додаткових параметрів;
- Написання програмного забезпечення для імітаційного моделювання за модифікованою моделлю.

Об'єктом дослідження є моделювання еволюції екосистеми на базі рівнянь Вольтерри-Лотки. Предметом дослідження є аналіз і модифікація рівнянь Вольтерри-Лотки для імітаційного моделювання, еволюційне моделювання екосистем.

Для досягнення поставленої мети було використано середовище розробки IntelliJIdea для написання програмного забезпечення на мові Java.

Наукова новизна полягає в тому, що створений програмний продукт за модифікованою моделлю Вольтерри-Лотки розглядає багатовимірний випадок екосистеми з можливістю впливу людини на систему та не вимагає від користувача графічного створення системи (вводяться текстові дані).

Потенційні застосування та практична цінність результатів дипломної роботи:

Програмний продукт може бути використаний екологами для моделювання екосистем та аналізу проблем екосистем і пропонування рішень цих проблем на базі результатів роботи програми за різних вхідних даних;

Програмний продукт може бути використаний для аналізу небезпеки промислового видобутку тварин і прийняття рішень щодо можливої кількості такого видобутку, за якого екосистемі не буде нанесено шкоди.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Передмова

Екосистема — це сукупність живих організмів, які пристосувалися до спільного проживання в певному середовищі існування, утворюючи з ним єдине ціле.

Математичне моделювання широко застосовується для вирішення багатьох актуальних задач екології та біології. Одним з важливих напрямків в цих дослідженнях є математичне моделювання біологічних популяцій. Воно застосовується для вирішення таких завдань, як збереження зникаючих і рідкісних видів, прогнозування чисельності промислових популяцій і розробка оптимальних стратегій промислу, вивчення впливу антропогенних факторів на чисельність біологічних видів, і інших.

Перші дослідження в області популяційного моделювання з'явилися в 20-і роки ХХ століття. Ключовими роботами, які дали потужний поштовх подальшим дослідженням, були дослідження А. Лотки і В. Вольтерри (створені незалежно один від одного), в яких розглядалася модель взаємодії двох популяцій «хижак-жертва».

1.2 Моделювання в екології

1.2.1 Моделі популяційної динаміки

Сучасні математичні моделі в екології можна розбити на три класи.

Перший – описові моделі: регресивні і інші емпірично встановлені кількісні залежності, що не претендують на розкриття механізму описуваного процесу. Вони застосовуються зазвичай для опису окремих процесів і залежностей і включаються як фрагменти в імітаційні моделі.

Другий – моделі якісні, які будуються з метою з'ясування динамічного механізму досліджуваного процесу, здатні відтворити спостерігаються динамічні ефекти в поведінці систем, такі, наприклад, як коливальний характер зміни біомаси. Зазвичай ці моделі не дуже громіздкі, піддаються якісному дослідженню з застосуванням аналітичних і комп'ютерних методів.

Третій клас – імітаційні моделі конкретних екологічних систем, що враховують всю наявну інформацію про об'єкт. Мета побудови таких моделей – детальне прогнозування поведінки складних систем або рішення оптимізаційної задачі їх експлуатації.

1.2.2 Імітаційне моделювання

Імітаційне моделювання — це метод, що дозволяє будувати моделі процесів, що описують, як ці процеси проходили б насправді. Імітаційне моделювання засноване на тому, що система, яка вивчається, замінюється імітатором і з ним проводяться експерименти з метою отримання інформації про цю систему. Імітаційне моделювання — це окремий випадок математичного моделювання.

Імітаційна модель — (у вузькому значенні) логіко-математичний опис об'єкта, який може бути використаний для експериментування на комп'ютері в цілях проектування, аналізу і оцінки функціонування об'єкта.

Імітаційні моделі мають ряд можливостей, яких немає у аналітичних моделей:

- дозволяють використовувати в моделі залежності, які не виражаються в аналітичній формі;
- дозволяють програвати різні сценарії;
- дозволяють враховувати часові та просторові неоднорідності.

1.2.3 Моделі міжвидової конкуренції

Перші моделі динаміки популяцій – це:

- ряд Фібоначчі (1202);
- модель експоненціального зростання Мальтуса (1798);
- модель обмеженого зростання Ферхюльста (1838).

Напочатку 20 століття з'явилися перші моделі взаємодії видів (моделі хижак-жертва). Першою була модель Вольтерри-Лотки (1926, 1925). Наступні моделі були її модифікаціями, варіаціями, узагальненням або створювались для вирішення конкретної задачі:

- Колмагорова (1972);
- Розенцвейга-Макартура (1965);
- Базикіна (1969).

1.3 Модель Вольтерри-Лотки

Модель Вольтерри-Лотки — система диференціальних рівнянь першого порядку, яка описує кінетику чисельності популяцій. Рівняння запропонували незалежно Альфред Джеймс Лотка та Віто Вольтерра, в 1925 та 1926 роках, відповідно.

Система складається з декількох біологічних видів і запасу їжі, який використовують деякі з розглянутих видів.

1.3.1 Припущення про компоненти системи

Їжа або ϵ в необмеженій кількості, або її надходження з плином часу жорстко регламентовано.

Особини кожного виду вмирають так, що в одиницю часу гине постійна частка існуючих особин.

Хижак поїдають жертв, причому, за одиницю часу, кількість з'їдених жертв завжди пропорційна ймовірності зустрічі особин цих двох видів, тобто добутку кількості хижаків на кількість жертв.

Якщо є їжа в необмеженій кількості і кілька видів, які здатні її споживати, то частка їжі, споживана кожним видом за одиницю часу, пропорційна кількості особин цього виду, взятого з деяким коефіцієнтом, що залежить від виду.

Якщо вид харчується їжею, наявної в необмеженій кількості, приріст чисельності виду за одиницю часу пропорційний чисельності виду.

Якщо вид харчується їжею, наявною в обмеженій кількості, то його розмноження регулюється швидкістю споживання їжі, тобто за одиницю часу приріст пропорційний кількості з'їденої їжі.

1.3.2 Трофічні піраміди

Екосистеми мають складну будову: кілька рівнів, між якими існують різноманітні трофічні (харчові) і топічні (не пов'язані з ланцюгом харчування) зв'язки. Структура трофічної піраміди може бути різною, в залежності від клімату, ґрунту, ландшафту, тривалості існування біогеоценозу і інших чинників.

При аналізі екосистем прийнято будувати харчові або трофічні піраміди, тобто графи, вершини яких відповідають видам, що входять до системи, а ребра вказують трофічні зв'язки між видами. Зазвичай такі графи – орієнтовані: напрямок дуги між двома вершинами вказує на той з видів, який є споживачем іншого, тобто напрямок дуги збігається з напрямком потоку речовини або біомаси в системі.

На рисунку 1.1 наведено приклад трофічного графу (дворівневої трофічної піраміди). З 15 видів комах (верхній рівень) 5 видів харчуються тільки на одному з двох видів рослин, 2 види - тільки на другому, в раціон інших 8 видів

комахи входять обидва види рослин. На рисунку 1.2 наведено приклад чотирирівневої трофічної піраміди з описом рівнів.

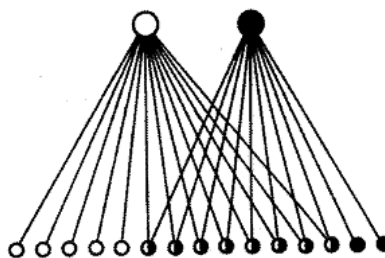


Рисунок 1.1 – Приклад дворівневої трофічної піраміди.



Рисунок 1.2 – Приклад реальної трофічної піраміди.

1.3.3 Рівняння двовимірної моделі (хижак-жертва)

$$\begin{cases} \frac{dx}{dt} = (\alpha - \beta y)x \\ \frac{dy}{dt} = (-\gamma + \delta x)y \end{cases}, \text{ де}$$

x – кількість жертв,

y – кількість хижаків,

α – коефіцієнт народжуваності жертв,

β – коефіцієнт вбивства жертв хижаками,

γ – коефіцієнт смертності хижаків,

δ – коефіцієнт народжуваності хижаків.

1.3.4 Стаціонарна позиція двовимірної моделі

Стаціонарна позиція системи: $x > 0, y > 0$, зміна чисельності популяцій рівна 0:

$$\begin{cases} (\alpha - \beta y)x = 0 \\ (-\gamma + \delta x)y = 0 \end{cases}$$

Тому, точка, навколо якої відбуваються коливання:

$$\begin{cases} x = \frac{\gamma}{\delta} \\ y = \frac{\alpha}{\beta} \end{cases}$$

1.3.5 Матриця взаємозв'язків

Повну структуру парних взаємодій між n видами можна зобразити за допомогою матриці з $n \times n$ елементів. Елемент (i, j) являє собою 1, -1 або 0 і показує вплив i -го виду на j -й. Симетричні пари елементів матриці S вказують на тип парної взаємодії між видами:

- +1 +1 симбіоз або мутуалізм;
- +1 -1 хижак - жертва (паразит-господар);
- +1 0 коменсалізм;
- -1 -1 конкуренція;
- -1 0 аменсалізм;
- 0 0 нейтралізм.

1.3.6 Узагальнене рівняння моделі

Модель складається з рівнянь виду:

$$\frac{dN_i}{dt} = N_i \left(\varepsilon_i - \sum_{j=0}^n \gamma_{ij} N_j \right), \text{ де}$$

n – кількість видів,

$i \in (0, n)$,

t – одиниця часу,

N_i – кількість особин i -го виду,

ε_i – коефіцієнт природного приросту або смертності,

γ_{ij} – коефіцієнт впливу j -го виду на i -й з матриці взаємозв'язків,

N_j – кількість особин j -го виду.

1.4 Огляд існуючих програмних рішень

1.4.1 The Ecosim with Ecosim (EWE) approach

Ecosim з Ecosim (EWE) - це програмний пакет для екологічного моделювання, який створювався і розширювався майже двадцять років. Розвиток зосереджено в Центрі рибальства Університету Британської Колумбії, а додатки широко поширені по всьому світу. EWE - перша імітаційна модель рівня екосистеми, яка широко і вільно доступна. Програмне забезпечення Ecosim було нещодавно визнано одним з десяти найбільших наукових досягнень NOAA за останні 200 років.

EWE має три основних компоненти:

- Ecosim – статичний, збалансований по масі знімок системи;

- Ecosim – модуль динамічного моделювання в часі для дослідження екосистем;
- Ecospace – просторовий і часовий динамічний модуль, призначений в першу чергу для вивчення впливу і розміщення охоронюваних територій.

Програмний пакет Ecorpath можна використовувати для:

- вирішення екологічних питань;
- оцінки екосистемних наслідків промислового вилову риби/відстрілу тварин;
- оцінки варіантів політики управління;
- оцінки впливу і розміщення морських охоронюваних районів;
- оцінки впливу змін у навколишньому середовищі.

Ecosim забезпечує динамічну симуляцію на рівні екосистеми з ключовими початковими параметрами, успадкованими від базової моделі Ecorpath. Основні обчислювальні аспекти представлені у вигляді:

- використання результатів масового балансу (від Ecorpath) для оцінки параметрів;
- поділ на групи зі змінною швидкістю розвитку дозволяє ефективно моделювати динаміку як «швидких» (фітопланктону), так і «повільних» груп (китів);
- вплив поведінки мікромасштабних організмів на макромасштабні: пряме управління знизу вгору або згори донизу;
- включення динаміки біомаси і структури розмірів для ключових груп екосистем, використовуючи поєднання диференціальних і різницевих рівнянь.

Ecosim використовує систему диференціальних рівнянь, які виражають швидкості потоку біомаси, в залежності від мінливих у часі біомаси та

швидкості збору. Виконуючи повторні симуляції, Ecosim дозволяє підганяти передбачені біомаси до даних часових рядів.

1.4.2 Ecolego

Ecolego – це потужний і гнучкий програмний інструмент для створення динамічних моделей і виконання детермінованих або імовірнісних симуляцій. Ecolego можна використовувати для проведення оцінок ризику складних динамічних систем, що розвиваються з часом з будь-якою кількістю видів. Ecolego має спеціалізовані бази даних та інші надбудови, розроблені для області оцінки радіологічних ризиків.

Графічний користувальницький інтерфейс допомагає користувачеві визначати будівельні блоки, параметрами, керувати видами і параметрами моделювання. Ecolego також допомагає створювати звіти, відображати результати моделювання, виконувати імовірнісні симуляції і аналіз чутливості.

Особливості:

- Просте керування блоками, параметрами, матеріалами та іншими налаштуваннями.
- Ієрархічні контейнери полегшують складання і документацію великих і складних моделей.
- Проектами легко керувати за допомогою браузера проекту, який відображає ієрархічне представлення блоків, параметрів, забруднень та інших об'єктів. Можна відкривати кілька проектів одночасно.
- Модна копіювати-вставляти параметри, контейнери, блоки та матеріали всередині і між проектами.
- Інструментальні підказки містять інформацію про обрані об'єкти: рівняння, одиниці виміру, тощо.
- Гіперпосилання дозволяють користувачу переходити до різних частин моделі.

- Створення моделей з використанням матриць взаємодії.
- Модулі підсистем можуть бути повторно використаними компонентами.
- Документація моделей з використанням зображень, гіперпосилань, категорій, одиниць і коментарів.
- Створює повні звіти, які можуть бути збережені в різних форматах, включаючи формат Adobe PDF.
- Можливість виклику призначених для користувача функцій в Java або Matlab під час моделювання.
- Потужні чисельні вирішувачі для складних і динамічних систем. Фіксовані і змінні статичні вирішувачі звичайних диференціальних рівнянь для жорстких задач.
- Підтримка імовірнісних симуляцій з використанням Монте-Карло або вибірки лабіринтів Hypercube.
- Розширений редактор функцій щільності ймовірності.
- Для результатів симуляції є багато різних типів діаграм результатів і таблиць.
- Підключення бази даних зовнішніх параметрів для спільного використання і управління параметрами проекту з гарантованою якістю. Інструментарій бази даних параметрів
- Експорт та імпорт значень параметрів і результатів в / з Microsoft Excel.

1.5 Висновки

У даному розділі досліджено предметну область та існуючі рішення (Ecosim та Ecolego).

Описані загальні засоби моделювання екосистем та поняття трофічних пірамід. Визначено модель Вольтерри-Лотки (в двовимірному та багатовимірному випадках).

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Модифікації моделі Вольтерри-Лотки

2.1.1 Матриця коефіцієнтів

Повну структуру парних взаємодій між n видами можна зобразити за допомогою матриці з $n \times n$ елементів. Елемент (i, j) являє собою коефіцієнт виїдання j -го виду i -м.

2.1.2 Внесення модифікацій в рівняння моделі

Узагальнене рівняння моделі з пункту 1.3.6:

$$\frac{dN_i}{dt} = N_i \left(\varepsilon_i - \sum_{j=0}^n \gamma_{ij} N_j \right), \text{ де}$$

n – кількість видів,

$i \in (0, n)$,

t – одиниця часу,

N_i – кількість особин i -го виду,

ε_i – коефіцієнт природного приросту або смертності,

γ_{ij} – коефіцієнт впливу j -го виду на i -й з матриці взаємозв'язків,

N_j – кількість особин j -го виду.

Внесемо модифікації в рівняння, що розширять варіанти впливу користувача на систему, спростять розуміння користувачем коефіцієнтів, зроблять систему логічнішою для написання програмного забезпечення, але не змінять принцип її роботи. Додамо коефіцієнти:

- g – коефіцієнт природного приросту;
- d – коефіцієнт природної смертності;
- h – вплив людини.

При програмній реалізації моделювання не доцільно розв'язувати систему диференціальних рівнянь, якщо відомі всі коефіцієнти (як в нашому випадку), в зв'язку з тим, що моделювання просто відбувається на певному Δt .

Також при програмній реалізації доцільно внести невелику частину випадковості в народжуваність і смертність, оскільки ці коефіцієнти досить важко балансувати, а подібна випадковість може видати трохи різні результати при однакових вхідних даних, що може бути корисно для аналізу екосистеми користувачем. Тобто, значення народжуваності/смертності, введене користувачем матиме вигляд $(d - g)_i$, а також матиме випадковий зсув на $(-0.25 ; 0.25) * (d - g)_i$.

Таким чином, узагальнене рівняння моделі (враховуючи внесені модифікації) приймає вигляд:

$$\frac{\Delta N_i}{\Delta t} = N_i(g - d)_i - \sum_{j=0}^n \gamma_{ij} N_j + h_i, \text{ де}$$

n – кількість видів,

$i \in (0, n)$,

Δt – одиниця часу (рік),

N_i – кількість особин i -го виду,

$(g - d)_i$ – коефіцієнт природного приросту (при $g > d$) або смертності (при $g < d$) i – го виду,

γ_{ij} – коефіцієнт виїдання j – го виду i – м з матриці коефіцієнтів,

N_j – кількість особин j – го виду,

h_i – кількість особин i – го виду, випущена на волю або вбита людиною щорічно (вплив людини на систему).

Така модифікована модель, однак, не враховує смертність хижаків при недоїданні, що у вихідній моделі регулювалась простим добутком кількості j – го виду на i – й та на коефіцієнт з матриці взаємозв'язків, але не враховувало різну поживність різних видів жертв для хижаків, тому додамо наступні коефіцієнти:

- m – поживність організму;
- f – необхідна кількість їжі.

В рамках коефіцієнтів m та f можна додати частину рівняння, що характеризувала б смертність хижаків при недостатній кількості їжі. Тобто, кожний вид має певну поживність і необхідну кількість їжі для кожної з особин виду.

В результаті, необхідна кількість їжі для хижаків вираховується, як кількість особин виду N_i , помножена на f_i , а реальна можлива кількість їжі – як сума всіх кількостей видів можливих жертв, помножених на їх поживність m_i .

У випадку недостатньої кількості їжі для видів, вони відмирають, в залежності від коефіцієнту, який вираховується як відношення реальної кількості їжі до необхідної. Якщо цей коефіцієнт більше одиниці, відмирання хижаків не відбувається, у інших випадках, особини видів вимирають пропорційно до коефіцієнту нестачі їжі.

2.1.3 Явний метод Ейлера для розв'язку системи рівнянь

Нехай дана задача Коші для рівняння першого порядку:

$$\frac{dy}{dx} = f(x, y)$$

Рішення шукається на інтервалі $(x_0, b]$. На цьому інтервалі введемо вузли $x_0 < x_1 < \dots < x_n \leq b$. Наближене рішення в вузлах x_i , яке позначимо через y_i , визначається за формулою:

$$y_i = y_{i-1} + (x_i - x_{i-1})f(x_{i-1}, y_{i-1}), i \in (1, n)$$

Ці формули безпосередньо узагальнюються на випадок систем звичайних диференціальних рівнянь.

Похибка на кроці або локальна похибка—це різниця між чисельним рішенням після одного кроку обчислення y_i і точним рішенням в точці $x_i = x_{i-1} + h$. Чисельне рішення задається формулою:

$$y_i = y_{i-1} + hf(x_{i-1}, y_{i-1}), h = (x_i - x_{i-1})$$

Точне рішення можна розкласти в ряд Тейлора:

$$y(x_{i-1} + h) = y(x_{i-1}) + hy'(x_{i-1}) + O(h^2)$$

Локальну помилку L отримуємо, віднімаючи від другого рівняння перше:

$$L = O(h^2)$$

Це справедливо, якщо функція $f(x, y)$ має безперервну другу похідну. Іншою достатньою умовою справедливості цієї оцінки є безперервна диференційованість за обома аргументами.

Похибка в цілому, глобальна або накопичена похибка — це похибка в останній точці довільного кінцевого відрізка інтегрованого рівняння. Для обчислення рішення в цій точці потрібно $\frac{S}{h}$ кроків, де S — довжина відрізка. Тому глобальна похибка методу:

$$G = O\left(\frac{h^2 S}{h}\right) = O(h)$$

Таким чином, метод Ейлера є методом першого порядку – має похибку кроку $O(h^2)$ та загальну похибку $O(h)$.

2.2 Вибір програмних засобів

2.2.1 Java

Java — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

«Oracle» надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції. Цього можна досягти, компілюючи початковий Java код у байт-код, який є спрощеними машинними командами. Потім програму можна

виконати на будь-якій платформі, що має встановлену віртуальну машину Java, яка інтерпретує байткод у код, пристосований до специфіки конкретної операційної системи і процесора. Зараз віртуальні машини Java існують для більшості процесорів і операційних систем.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної» мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова.

Java використовує автоматичний збирач сміття для керування пам'яттю під час життєвого циклу об'єкта. Програміст вирішує, коли створювати об'єкти, а віртуальна машина відповідальна за звільнення пам'яті після того, як об'єкт стає непотрібним. Коли до певного об'єкта вже не залишається посилань, збирач сміття може автоматично прибирати його із пам'яті. Проте, витік пам'яті все ж може статися, якщо код, написаний програмістом, має посилання на вже непотрібні об'єкти, наприклад на об'єкти, що зберігаються у діючих контейнерах.

Збирання сміття дозволене у будь-який час. В ідеалі воно відбувається під час бездіяльності програми. Збірка сміття автоматично форсується при нестачі вільної пам'яті в купі для розміщення нового об'єкта, що може призводити до кількасекундного зависання. Тому існують реалізації віртуальної машини Java з прибиральником сміття, спеціально створеним для програмування систем реального часу.

2.2.2JDK 8

Java Development Kit, скорочено JDK — безкоштовний розповсюджуваний Oracle комплект розробника застосунків на мові Java, який включає до себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різноманітні утиліти і виконавчу систему Java (JRE). В склад JDK не входить інтегроване середовище розробки на Java (IDE), тому розробник, що використовує тільки JDK, повинен використовувати текстовий редактор і компілювати та виконувати свої програми через утиліти командного рядка.

Програмний продукт написаний з використанням JDK 8.

2.2.3IntelliJ IDEA (CommunityEdition) 2017.1.3

IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Двійкові збірки підготовлені для Linux, Mac OS X і Windows.

Програмний продукт написаний у IntelliJ IDEA(Community Edition), версії 2017.1.3.

2.2.4Swing

Swing — інструментарій для створення графічного інтерфейсу користувача (GUI) мовою програмування Java. Це частина бібліотеки базових класів Java (JFC, Java Foundation Classes).

Swing розробляли для забезпечення функціональнішого набору програмних компонентів для створення графічного інтерфейсу користувача, ніж у ранішого інструментарію AWT. Компоненти Swing підтримують специфічні look-and-feel модулі, що динамічно підключаються. Завдяки ним

можлива емуляція графічного інтерфейсу платформи (тобто до компоненту можна динамічно підключити інші, специфічні для даної операційної системи вигляд і поведінку). Основним недоліком таких компонентів є відносно повільна робота, хоча останнім часом це не вдалося підтвердити через зростання потужності персональних комп'ютерів. Позитивна сторона — універсальність інтерфейсу створених програм на всіх платформах.

В програмному продукті Swing використовується для створення GUI.

2.2.5JFreeChart 1.0.19

JFreeChart — відкрита бібліотека для мови програмування Java, що спрощує створення різноманітних складних діаграм. Через різноманітні методи набору класів надає майже повний контроль над областю діаграми. Реалізовані механізми збільшення/зменшення, обробки подій, можливості створення кількох діаграм на одній області, текстові підказки, задання вигляду кривих, точок та фону і т.п.

Підтримуються такі типи діаграм:

- Лінійні графіки
- Діаграми часових рядів
- Кругові діаграми
- Графіки Гранта
- Гістограми (стовбцеві діаграми)
- Комбінована гістограма
- Графіки різниці
- Покрокові діаграми
- Комбіновані діаграми

У програмному продукті для побудови графіків використовується дана бібліотека, версії 1.0.19.

2.3 Висновки

У даному розділі модифіковано модель Вольтерри-Лотки так, щоб вона забезпечувала можливість створення конкретизованої багатовимірної моделі з великою кількістю видів та аналіз їх поведінки з можливістю втручання людини до системи.

Також обрано середовище розробки програмного продукту (IntelliJ IDEA, Community Edition, v. 2017.1.3) та засоби Java для реалізації програмного продукту: JDK 8, Swing (для створення GUI), JFreeChart 1.0.19 (для побудови графіків).

3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Структура програмного продукту

3.1.1 UML-діаграма класів

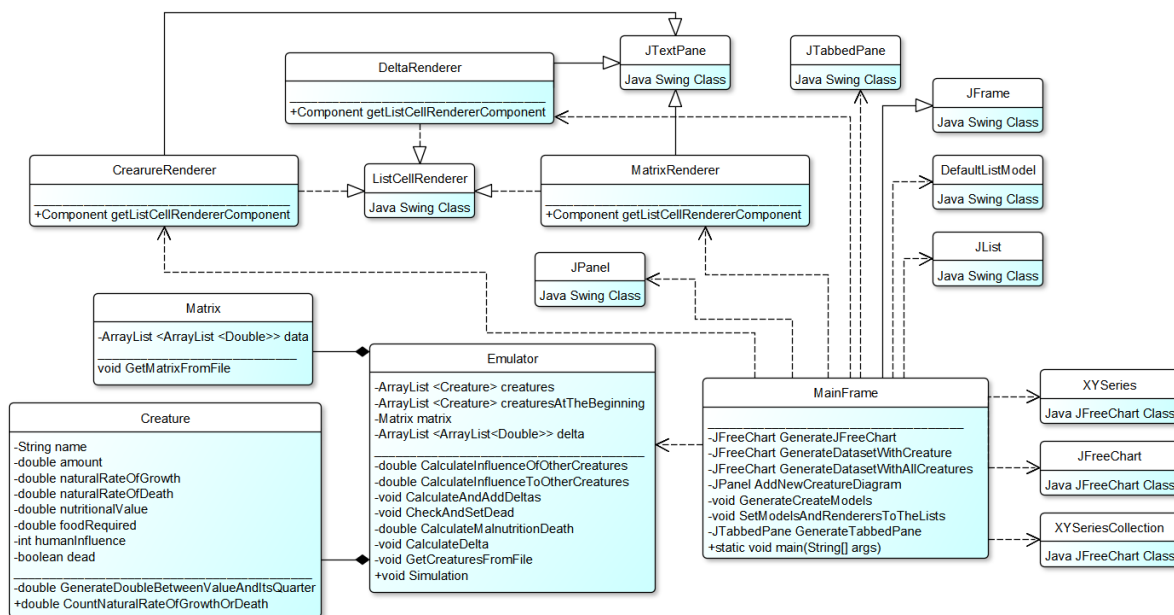


Рисунок 3.1 – UML діаграма класів.

3.1.2 Опис класів Swing, що були використані

3.1.2.1 JFrame

```
public class JFrame
```

```
extends Frame
```

```
implements WindowConstants, Accessible, RootPaneContainer
```

Розширена версія `java.awt.Frame`, яка додає підтримку архітектури компонентів JFC / Swing.

Клас `JFrame` трохи несумісний з `Frame`. Як і всі інші контейнери верхнього рівня JFC / Swing, `JFrame` містить `JRootPane` як єдиний його дочірній елемент.

Панель вмісту, що надається кореневої панеллю, повинна, як правило, містити всі компоненти не-меню, які відображаються JFrame.

Область вмісту завжди буде не нульова. Спроба встановити її в значення null призведе до того, що JFrame видасть виняток. На панелі вмісту за замовчуванням буде встановлено менеджер BorderLayout.

На відміну від Frame, JFrame має деяке уявлення про те, як реагувати, коли користувач намагається закрити вікно. Поведінка за замовчуванням - приховати JFrame, коли користувач закриває вікно. Щоб змінити поведінку за замовчуванням, викликається метод `setDefaultCloseOperation (int)`.

3.1.2.2 JTextPane

```
public class JTextPane
```

```
extends JEditorPane
```

Текстовий компонент, який може бути позначений атрибутами, які представлені графічно. Цей компонент моделює параграфи, які складаються з прогонів атрибутів рівня символів. Кожен абзац може мати прикріплений до нього логічний стиль, який містить атрибути за замовчуванням, які слід використовувати, якщо вони не перевизначені атрибутами, встановленими в абзаці або символічному прогоні. Компоненти і зображення можуть бути вбудовані в потік тексту.

3.1.2.3 JList

```
public class JList<E>
```

```
extends JComponent
```

```
implements Scrollable, Accessible
```

Компонент, який відображає список об'єктів і дозволяє користувачеві вибирати один або кілька елементів. Окрема модель `ListModel` підтримує вміст списку.

3.1.2.4 `DefaultListModel`

```
public class DefaultListModel<E>
    extends AbstractListModel<E>
```

Цей клас вільно реалізує API `java.util.Vector`, оскільки він реалізує версію `java.util.Vector` версії 1.1.x, не має підтримки класу колекції і повідомляє `ListDataListeners` про те, коли відбуваються зміни.

3.1.2.5 `ListCellRenderer`

```
public interface ListCellRenderer<E>
```

Визначає компоненти, які можуть використовуватися як «гумові штампи» ("rubber stamps") для малювання «комірок» ("cells") в `JList`.

3.1.2.6 `JTabbedPane`

```
public class JTabbedPane
    extends JComponent
    implements Serializable, Accessible, SwingConstants
```

Компонент, який дозволяє користувачеві перемикатися між групою компонентів, натиснувши на вкладку з цим заголовком і / або значком.

Вкладки / компоненти додаються в об'єкт `TabbedPane` за допомогою методів `addTab` і `insertTab`. Вкладка представлена індексом, відповідним позиції, починаючи з 0.

3.1.2.7 JPanel

```
public class JPanel
```

```
extends JComponent
```

```
implements Accessible
```

JPanel – загальний легкий (lightweight) контейнер.

3.1.3 Опис класів JFreeChart, що були використані

3.1.3.1 JFreeChart

```
public class JFreeChart
```

```
extends java.lang.Object
```

```
implements org.jfree.ui.Drawable, TitleChangeListener, PlotChangeListener,
java.io.Serializable, java.lang.Cloneable
```

Клас діаграми, реалізований з використанням Java 2D API. Поточна версія підтримує гістограми, лінійні діаграми, кругові діаграми і графіки ху (включаючи дані часових рядів).

JFreeChart координує кілька об'єктів, щоб намалювати діаграму на 2D-графічному пристрої Java: список об'єктів (Title), графік (Plot) і точки на осях OX та OY (Dataset).

3.1.3.2 XYSeriesCollection

```
public class XYSeriesCollection
```

```
extends AbstractIntervalXYDataset
```

```
implements IntervalXYDataset, DomainInfo, RangeInfo,
java.beans.VetoableChangeListener, org.jfree.util.PublicCloneable,
java.io.Serializable
```

Представляє колекцію об'єктів XYSeries, які можуть використовуватися в якості набору даних.

3.1.3.3 XYSeries

```
public class XYSeries
```

```
extends Series
```

```
implements Cloneable, Serializable
```

Являє собою послідовність з нуля або більше елементів даних в формі (x, y). За замовчуванням елементи в серії будуть відсортовані в порядку зростання за значенням x, а допустимі повторювані значення x дозволяються. Як сортування, так і повторювані значення, що встановлені за замовчуванням, можуть бути замінені в конструкторі. Значення Y можуть бути порожніми, щоб представити відсутні значення.

3.1.4 Опис класів логіки та обробки даних

3.1.4.1 Creature

```
public class Creature
```

Поля та методи класу зображені на елементі Рисунок UML діаграми з пункту 3.1.1, а також – на Рисунок 3.2.

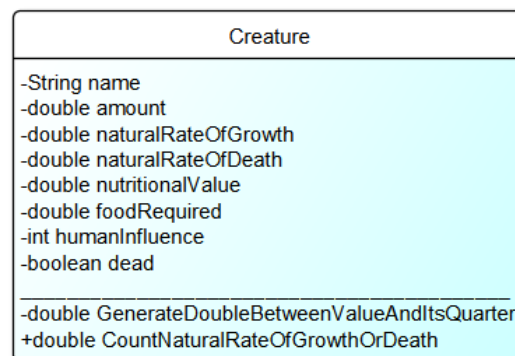


Рисунок 3.2 – Поля та методи класу Creature.

Клас являє собою опис кожного виду:

- назва виду,
- кількість особин,
- коефіцієнт природного приросту,
- коефіцієнт природної смертності,
- коефіцієнт поживності організму,
- коефіцієнт необхідної кількості їжі для організму,
- вплив людини,
- коефіцієнт, що вказує, чи живий вид на даний ітерації.

3.1.4.2 Matrix

```
public class Matrix
```

Поля та методи класу зображені на елементі Рисунок UML діаграми з пункту 3.1.1, а також – на Рисунок 3.3.

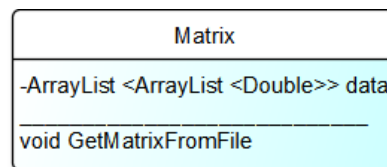


Рисунок 3.3 – Поля та методи класу Matrix.

Клас являє собою матрицю коефіцієнтів, призначення якої описано в пункті 2.1.1.

3.1.4.3 Emulator

```
public class Emulator
```

Поля та методи класу зображені на елементі Рисунок UML діаграми з пункту 3.1.1, а також – на Рисунок 3.4.

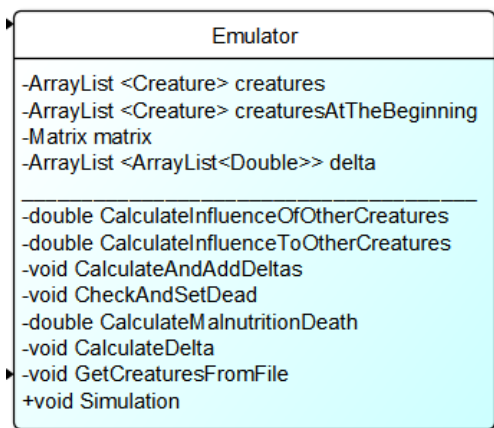


Рисунок 3.4 – Поля та методи класу Emulator.

Клас являє собою емулятор, що проводить симуляцію математичної моделі, описаної в пункті 2.1.2.

3.1.5 Опис класів GUI

3.1.5.1 MainFrame

```
public class MainFrame
```

```
extends JFrame
```

Поля та методи класу зображені на елементі Рисунку UML діаграми з пункту 3.1.1, а також – на Рисунку 3.5.

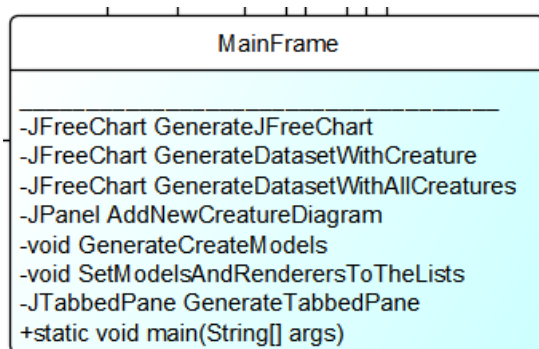


Рисунок 3.5 – Методи класу MainFrame.

Клас являє собою GUI, який представлений у вигляді вікна з вкладеннями (“Существа вначале”, “Существа вконец”, “Матрица”, “Дельта”, “Справка”, “Графики”, а також – вкладення з назвами усіх істот поточної симуляції).

CrearureRenderer, DeltaRenderer, MatrixRenderer

```
public class CrearureRenderer
```

```
extends JTextPane
```

```
implements ListCellRenderer<Creature>
```

Поля та методи класу зображені на елементі Рисунку UML діаграми з пункту 3.1.1, а також – на Рисунку 3.6.

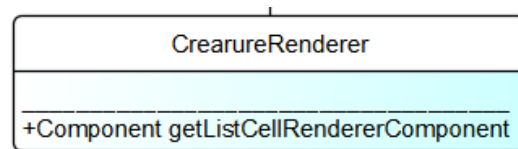


Рисунок 3.6– Метод класу CrearureRenderer.

```
public class DeltaRenderer
```

```
extends JTextPane
```

```
implements ListCellRenderer <ArrayList<Double>>
```

Поля та методи класу зображені на елементі Рисунку UML діаграми з пункту 3.1.1, а також – на Рисунку 3.7.

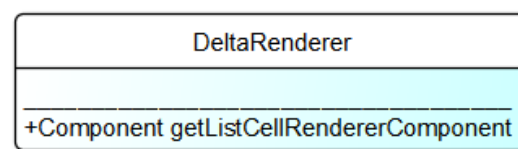


Рисунок 3.7 – Метод класу DeltaRenderer.


```
public class MatrixRenderer
extends JPanel
implements ListCellRenderer<ArrayList<Double>>
```

Поля та методи класу зображені на елементі Рисунку UML діаграми з пункту 3.1.1, а також – на Рисунку 3.8.

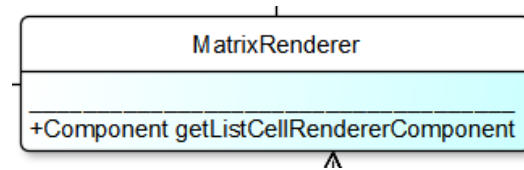


Рисунок 3.8 – Метод класу DeltaRenderer.

Класи являють собою Renderer-и для різних типів JList, що використані в класі MainFrame (Renderer-и описують форматування елементів JList).

3.2 Результати роботи програмного продукту для моделювання еволюції екосистеми лісу

3.2.1 Обмеження можливих результатів

Оскільки у відкритому доступі немає необхідних даних для введення у програму, то неможливо:

- задати коректні у всіх відношеннях початкові дані;
- перевірити результати роботи програмного продукту, з точки зору передбачення еволюції екосистеми та аналізу коливань популяцій.

Однак, у можливого користувача програмного продукту, з великою вірогідністю, буде можливість отримати коректні початкові дані, тому ця проблема не розглядається.

3.2.2 Опис моделюваної екосистеми

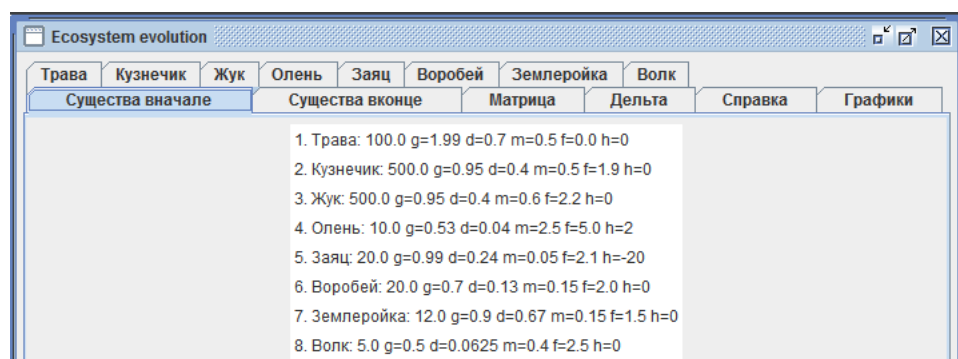
В розділах 3.2.3-3.2.5 розглянуто приклад роботи програмного продукту для симуляції еволюції екосистеми лісу.

Зробимо припущення про модельовану екосистему:

- нехай екосистему населяють такі істоти: “Трава”, “Кузнечик”, “Жук”, “Олень”, “Заяц”, “Воробей”, “Землеройка”, “Волк”;
- нехай, моделюється еволюція екосистеми на наступні 5 років.

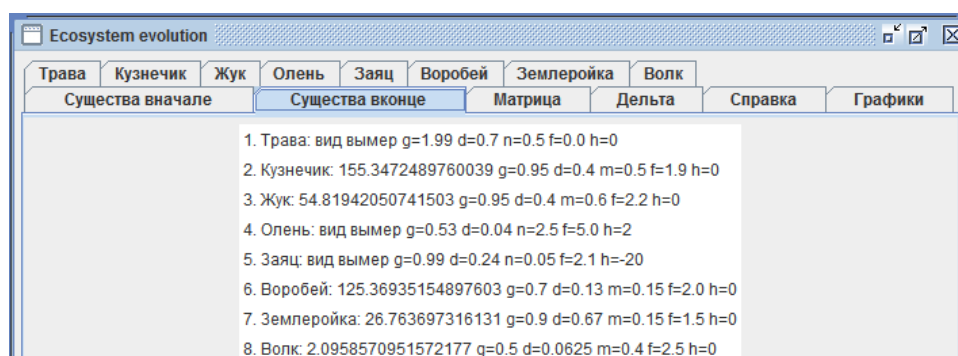
3.2.3 Результати моделювання екосистеми за умови нестачі трави

На Рисунках 3.9 – 3.22 показано результати роботи програмного продукту, за умови вказання недостатньої кількості трави користувачем.



Трава	Кузнечик	Жук	Олень	Заяц	Воробей	Землеройка	Волк
Существа вначале	Существа вконец	Матрица	Дельта	Справка	Графики		
1. Трава: 100.0 g=1.99 d=0.7 m=0.5 f=0.0 h=0							
2. Кузнечик: 500.0 g=0.95 d=0.4 m=0.5 f=1.9 h=0							
3. Жук: 500.0 g=0.95 d=0.4 m=0.6 f=2.2 h=0							
4. Олень: 10.0 g=0.53 d=0.04 m=2.5 f=5.0 h=2							
5. Заяц: 20.0 g=0.99 d=0.24 m=0.05 f=2.1 h=-20							
6. Воробей: 20.0 g=0.7 d=0.13 m=0.15 f=2.0 h=0							
7. Землеройка: 12.0 g=0.9 d=0.67 m=0.15 f=1.5 h=0							
8. Волк: 5.0 g=0.5 d=0.0625 m=0.4 f=2.5 h=0							

Рисунок 3.9 – Кількість видів з їх коефіцієнтами до симуляції.



1. Трава: вид вымер g=1.99 d=0.7 n=0.5 f=0.0 h=0							
2. Кузнечик: 155.3472489760039 g=0.95 d=0.4 m=0.5 f=1.9 h=0							
3. Жук: 54.81942050741503 g=0.95 d=0.4 m=0.6 f=2.2 h=0							
4. Олень: вид вымер g=0.53 d=0.04 n=2.5 f=5.0 h=2							
5. Заяц: вид вымер g=0.99 d=0.24 n=0.05 f=2.1 h=-20							
6. Воробей: 125.36935154897603 g=0.7 d=0.13 m=0.15 f=2.0 h=0							
7. Землеройка: 26.763697316131 g=0.9 d=0.67 m=0.15 f=1.5 h=0							
8. Волк: 2.0958570951572177 g=0.5 d=0.0625 m=0.4 f=2.5 h=0							

Рисунок 3.10 – Кількість видів з їх коефіцієнтами після симуляції.

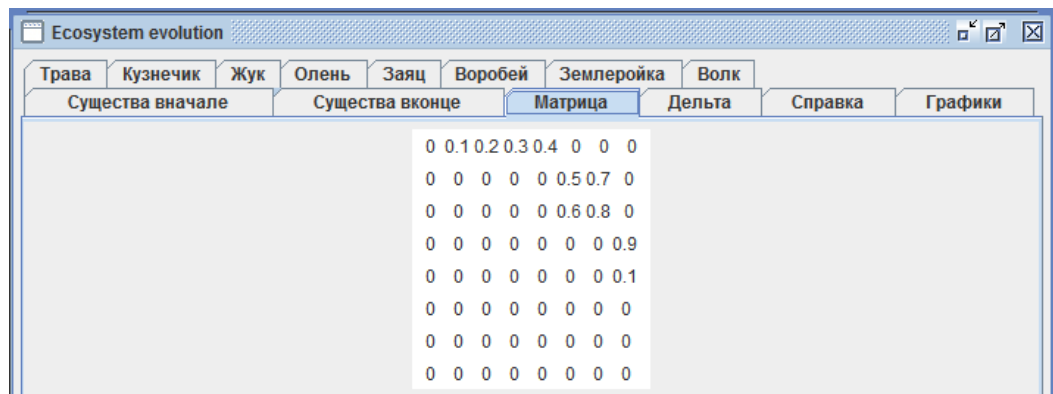


Рисунок 3.11 – Матриця коефіцієнтів виідання.

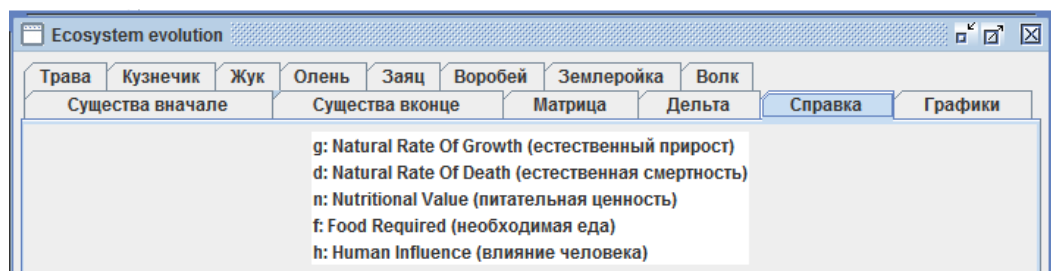
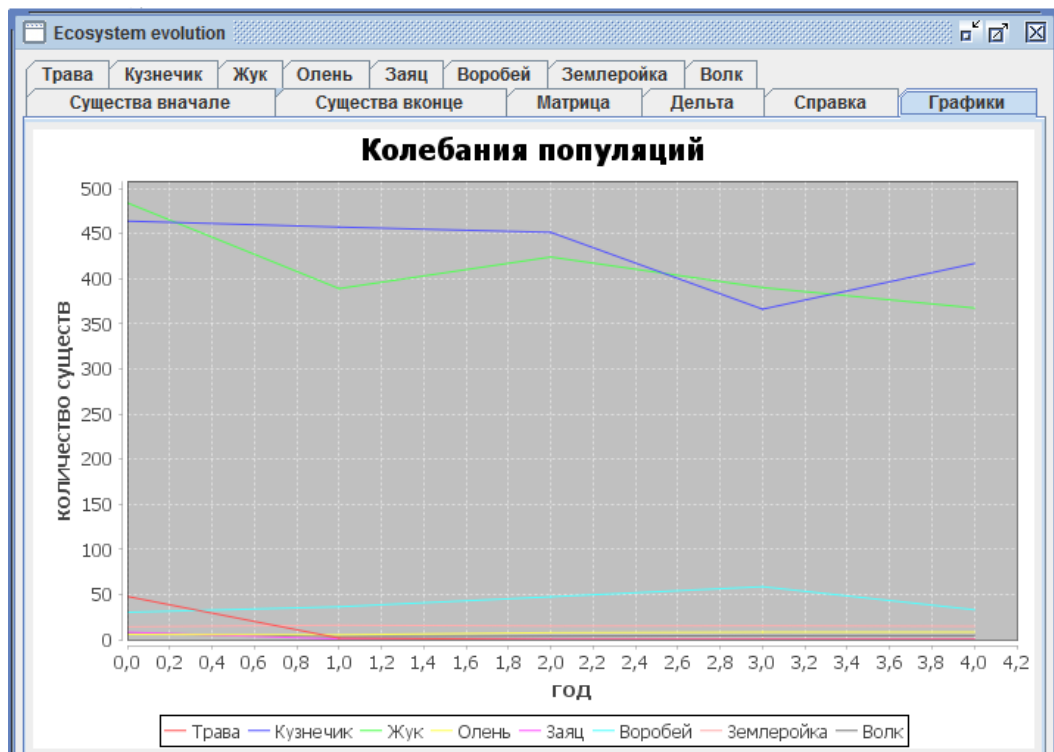


Рисунок 3.12 – Опис призначення коефіцієнтів.

Рисунок 3.13 – Загальний графік видозміни популяцій за Δt років.

Ecosystem evolution							
Трава	Кузнечик	Жук	Олень	Заяц	Воробей	Землеройка	Волк
Существа в начале		Существа в конце		Матрица	Дельта	Справка	Графики
год: 1							
1. -52.24190367030462							
2. -36.19024223621875							
3. -15.898182604709802							
4. -4.3860624580963625							
5. -11.93473821261843							
6. 10.172224973573769							
7. 2.1081682337105687							
8. 1.619543145293159							
год: 2							
1. -98.42534260813974							
2. -42.80768272069841							
3. -110.97117391780819							
4. -4.811429295380434							
5. -19.425424254891016							
6. 16.298108176240703							
7. 3.9032293277117636							
8. -1.8510095968065916							
год: 3							
1. -169.60371486050778							
2. -48.62360605186268							
3. -75.96097510386096							
4. -2.3963450278555136							
5. -27.537643841628775							
6. 27.35914716396728							
7. 2.92396063705702							
8. -1.2346159369501823							
год: 4							
1. -296.80370275713744							
2. -133.81687022878285							
3. -109.85400164231478							
4. -1.8548468911836993							
5. -49.393262853938005							
6. 38.51880927549709							
7. 3.167694425196219							
8. -0.7500635542570697							
год: 5							
1. -721.174643623879							
2. -83.21434978643344							
3. -132.49624622389123							
4. -1.827156547667328							
5. -73.93597765344285							
6. 13.021061959697192							
7. 2.660644692455427							
8. -0.6879969621220985							

Рисунок 3.14 – Список ΔN_i всіх видів за всі роки симуляції.



Рисунок 3.15 – Графік, що відображає коливання популяції трави.

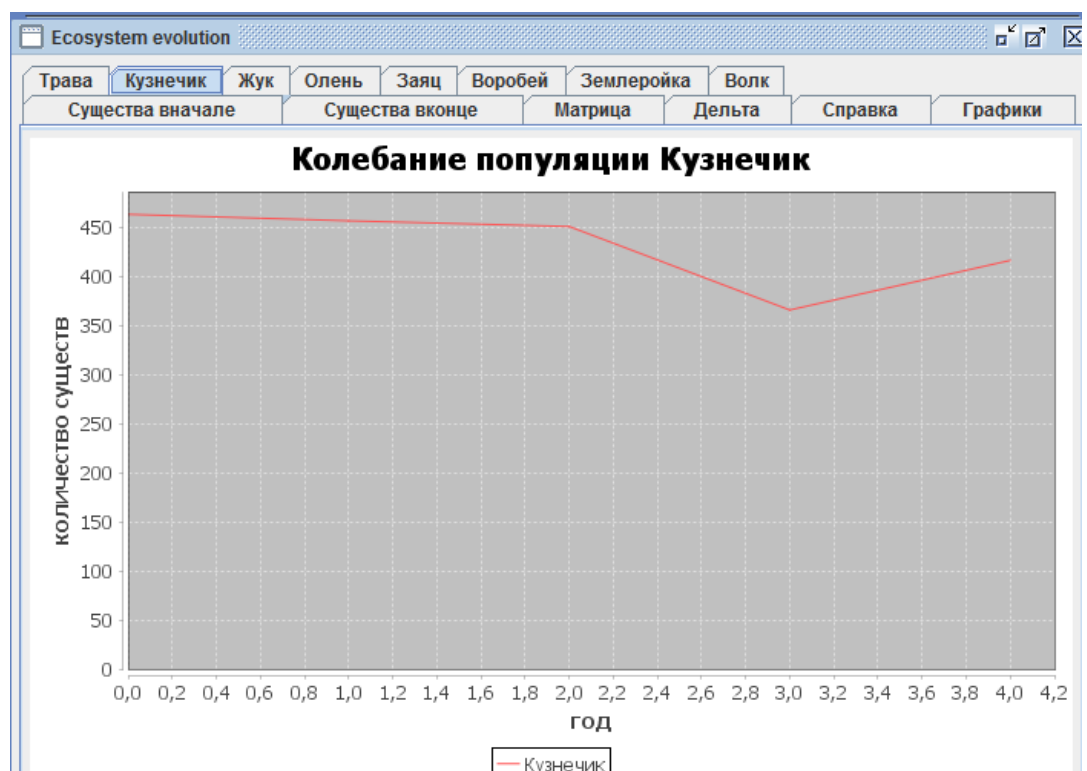


Рисунок 3.16 – Графік, що відображає коливання популяції коників.

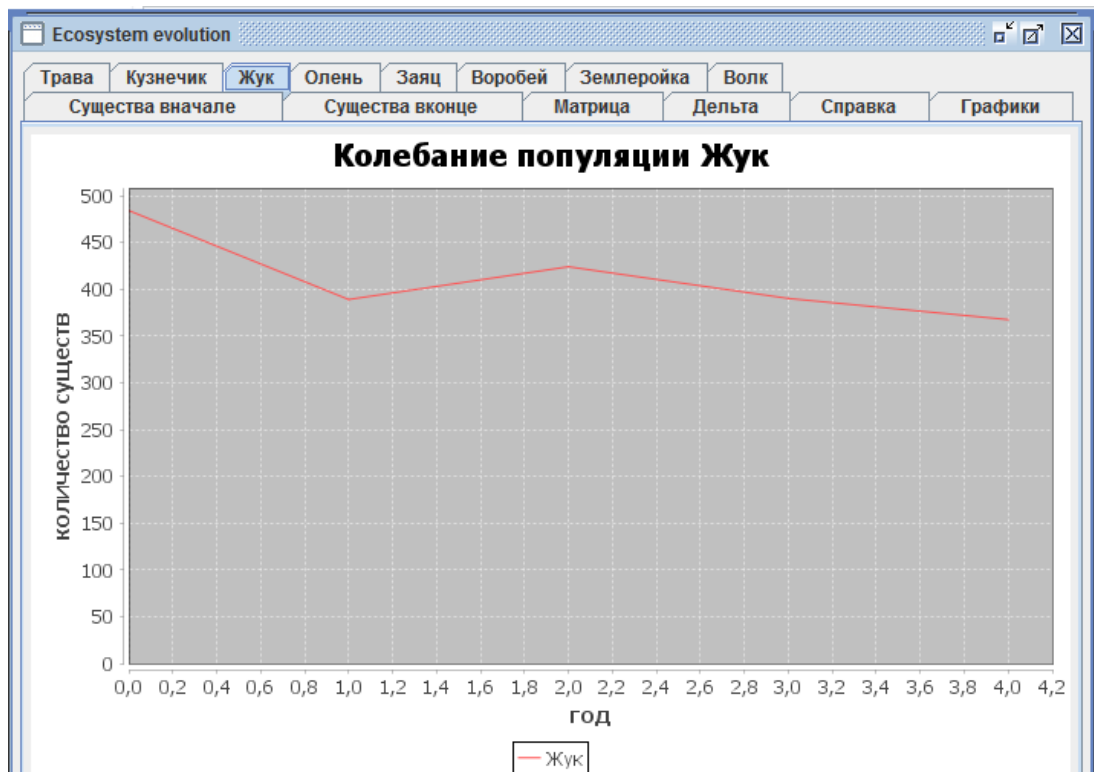


Рисунок 3.17 – Графік, що відображає коливання популяції жуків.



Рисунок 3.18 – Графік, що відображає коливання популяції оленів.

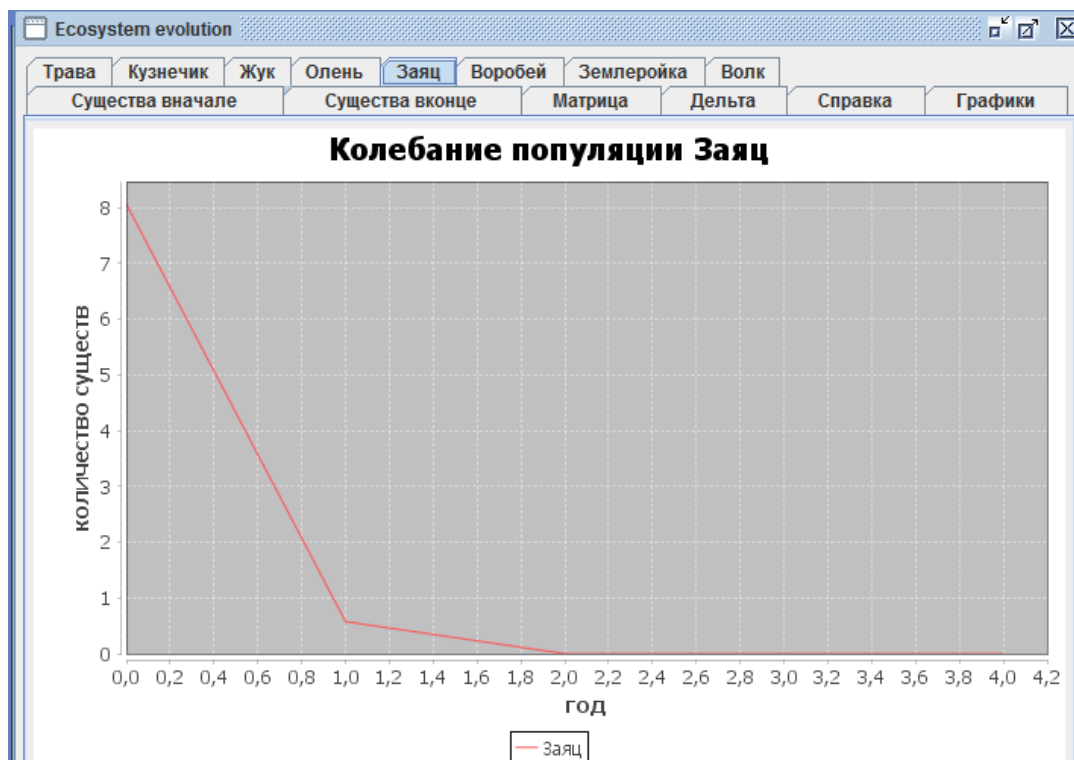


Рисунок 3.19 – Графік, що відображає коливання популяції зайців.

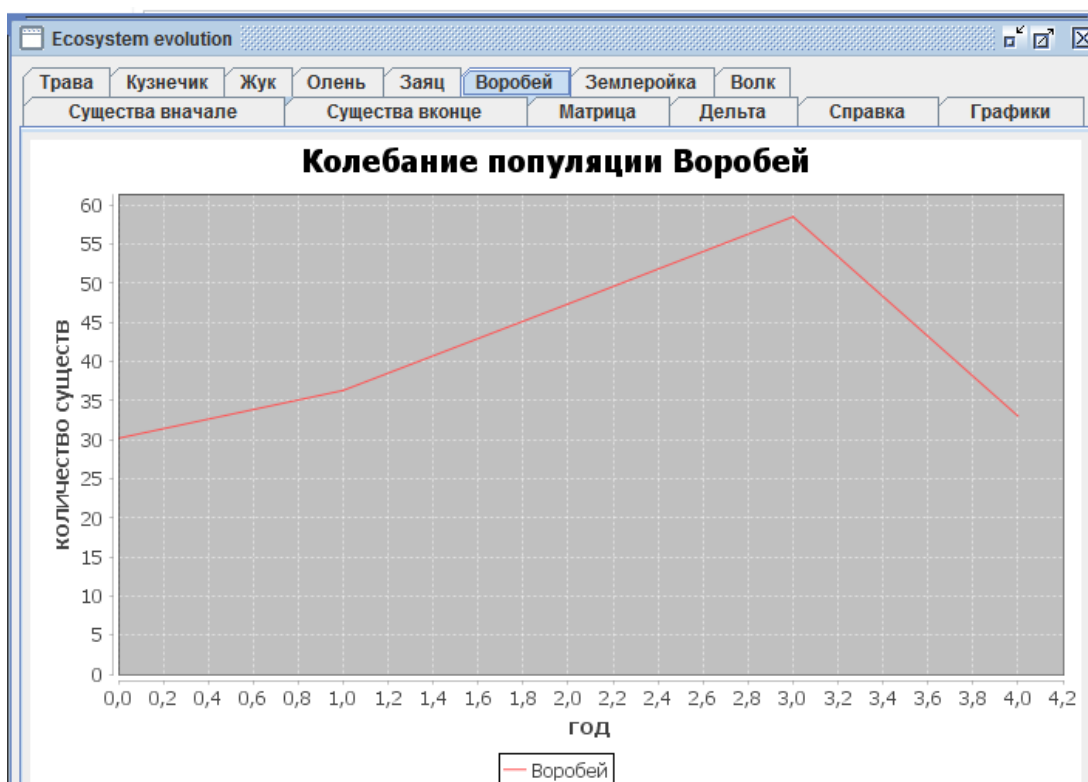


Рисунок 3.20 – Графік, що відображає коливання популяції горобців.

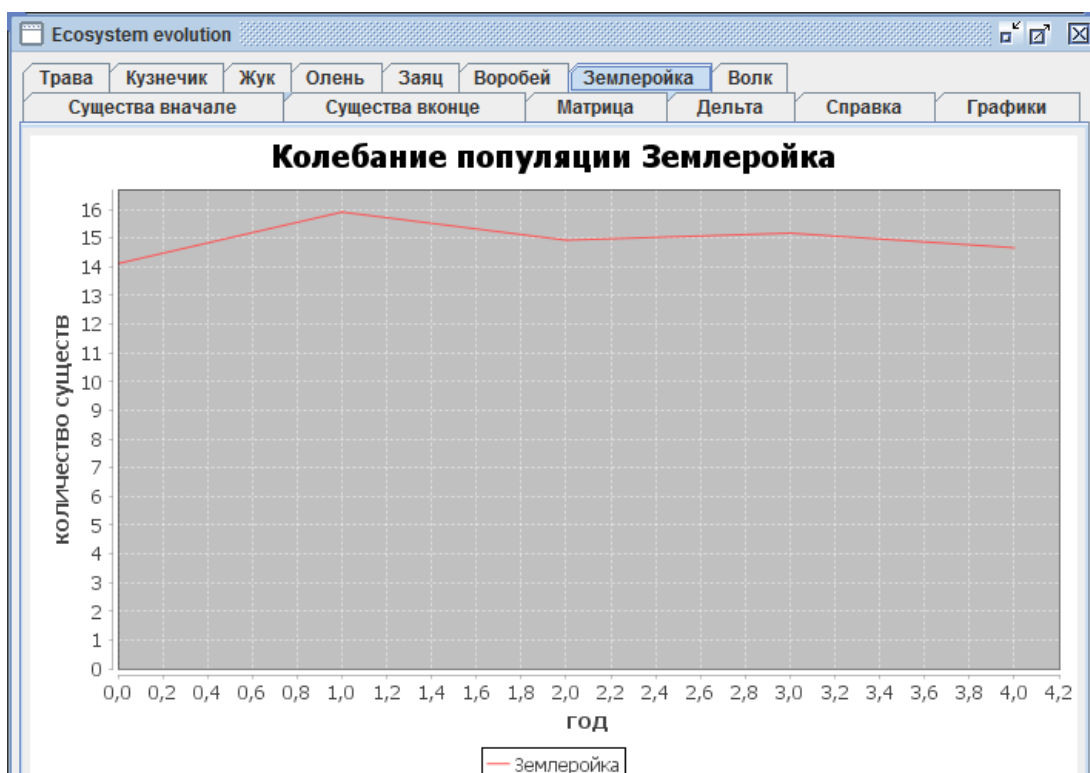


Рисунок 3.21 – Графік, що відображає коливання популяції землерийок.

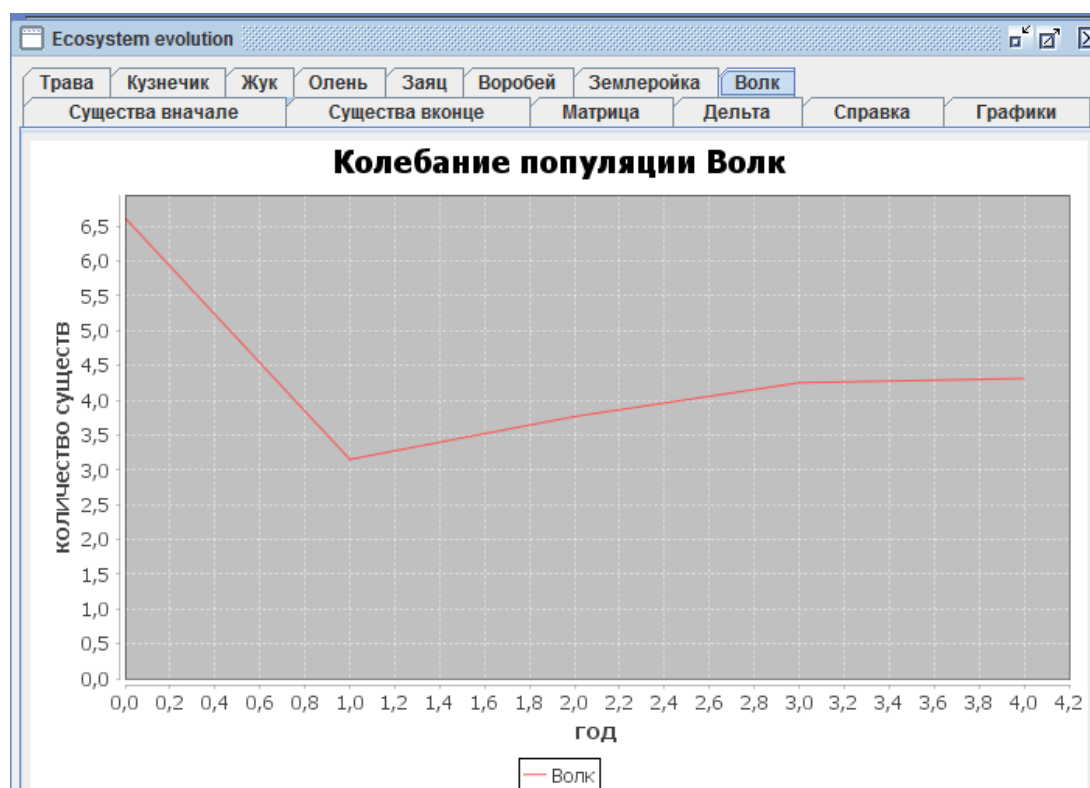


Рисунок 3.22 – Графік, що відображає коливання популяції вовків.

3.2.4 Результати моделювання екосистеми за нормальних умов

На Рисунках 3.23 –3.36 показано результати роботи програмного продукту, за умови вказання відносно нормальних умов існування для видів користувачем.

№	Вид	g	d	m	f	h
1.	Трава	10000.0	1.99	0.7	0.0	0
2.	Кузнечик	500.0	0.95	0.4	0.5	1.9
3.	Жук	500.0	0.95	0.4	0.6	2.2
4.	Олень	20.0	0.53	0.04	2.5	5.0
5.	Заяц	50.0	0.99	0.01	0.15	0.3
6.	Воробей	20.0	0.7	0.13	0.15	2.0
7.	Землеройка	12.0	0.9	0.67	0.15	1.5
8.	Волк	10.0	0.5	0.0625	0.4	2.5

Рисунок 3.23 – Кількість видів з їх коефіцієнтами до симуляції.

№	Вид	g	d	m	f	h
1.	Трава	478331.63655373023	1.99	0.7	0.0	0
2.	Кузнечик	3099.272138540467	0.95	0.4	0.5	1.9
3.	Жук	3590.4477539844547	0.95	0.4	0.6	2.2
4.	Олень	123.22609180307839	0.53	0.04	2.5	5.0
5.	Заяц	577.9322819492576	0.99	0.01	0.15	0.3
6.	Воробей	155.02461879856685	0.7	0.13	0.15	2.0
7.	Землеройка	28.60825248223039	0.9	0.67	0.15	1.5
8.	Волк	27.699585715202062	0.5	0.0625	0.4	2.5

Рисунок 3.24 – Кількість видів з їх коефіцієнтами після симуляції.

0	0.1	0.2	0.3	0.4	0	0	0
0	0	0	0	0	0.5	0.7	0
0	0	0	0	0	0.6	0.8	0
0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0.1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Рисунок 3.25 – Матриця коефіцієнтів видання.

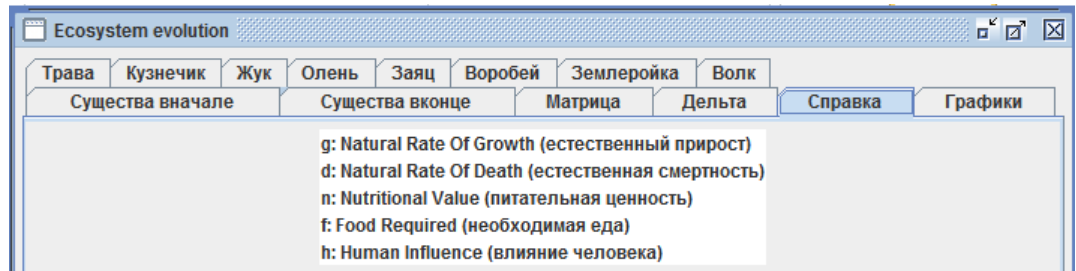


Рисунок 3.26 – Опис призначення коефіцієнтів.

Рисунок 3.27 – Список ΔN_i всіх видів за всі роки симуляції.

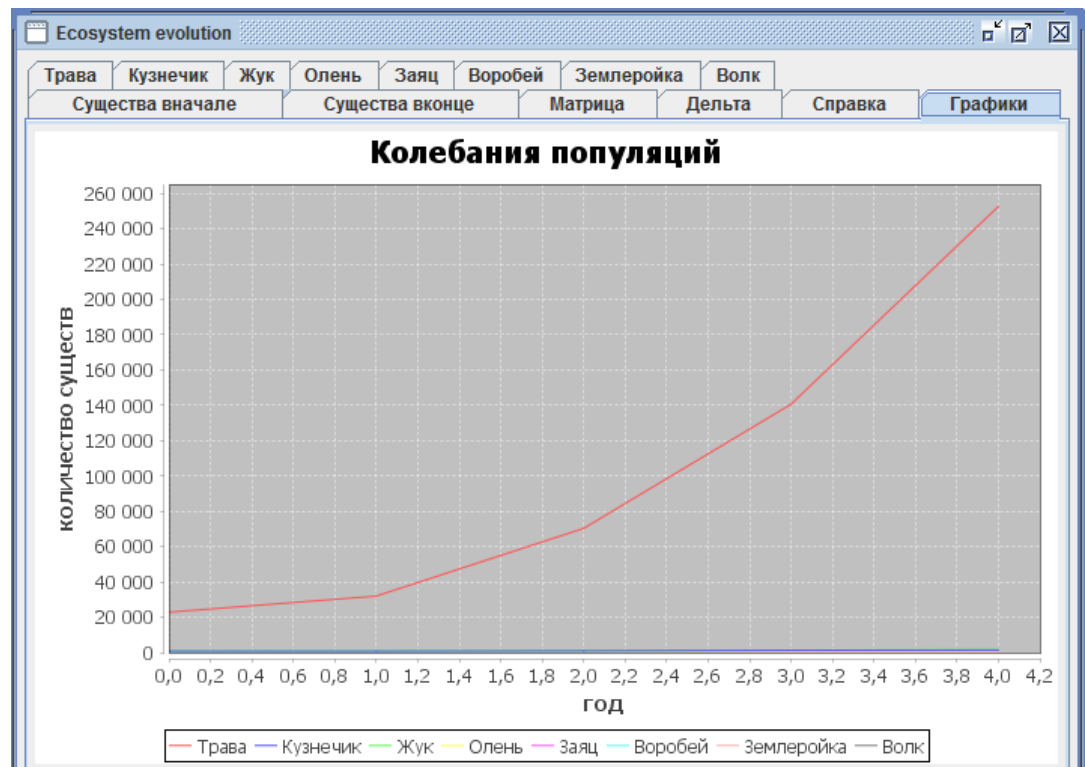


Рисунок 3.28 – Загальний графік видозміни популяцій за Δt років.



Рисунок 3.29 – Графік, що відображає коливання популяції трави.



Рисунок 3.30 – Графік, що відображає коливання популяції коників.

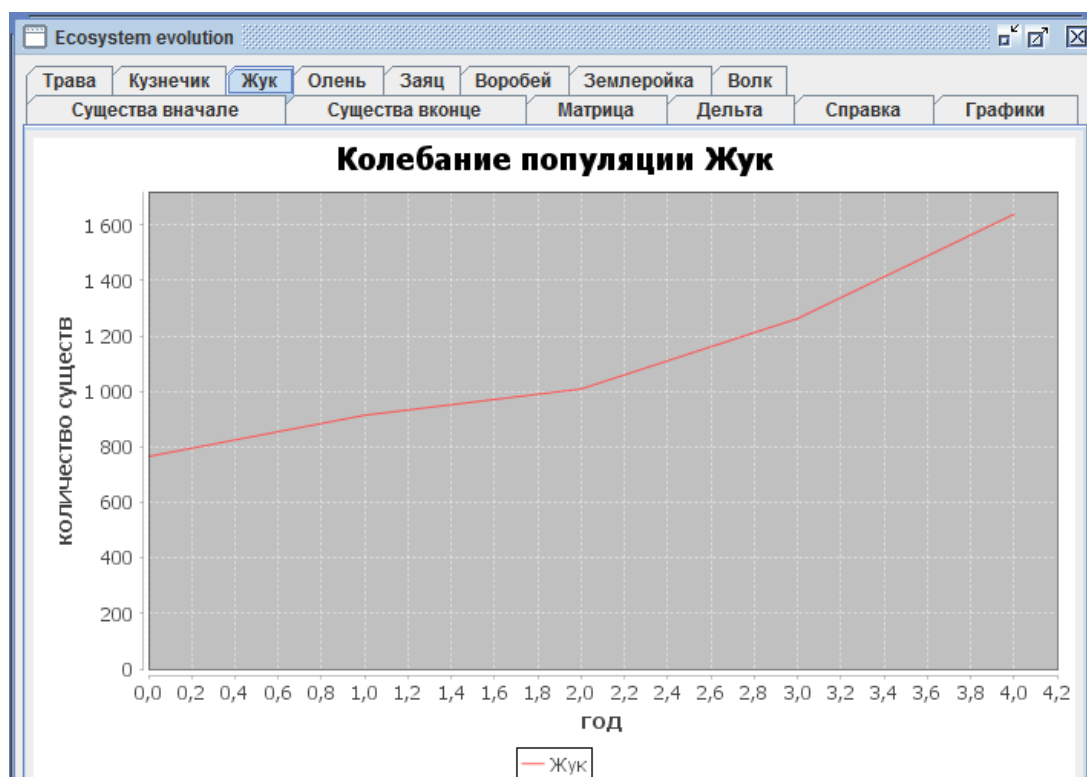


Рисунок 3.31 – Графік, що відображає коливання популяції жуків.

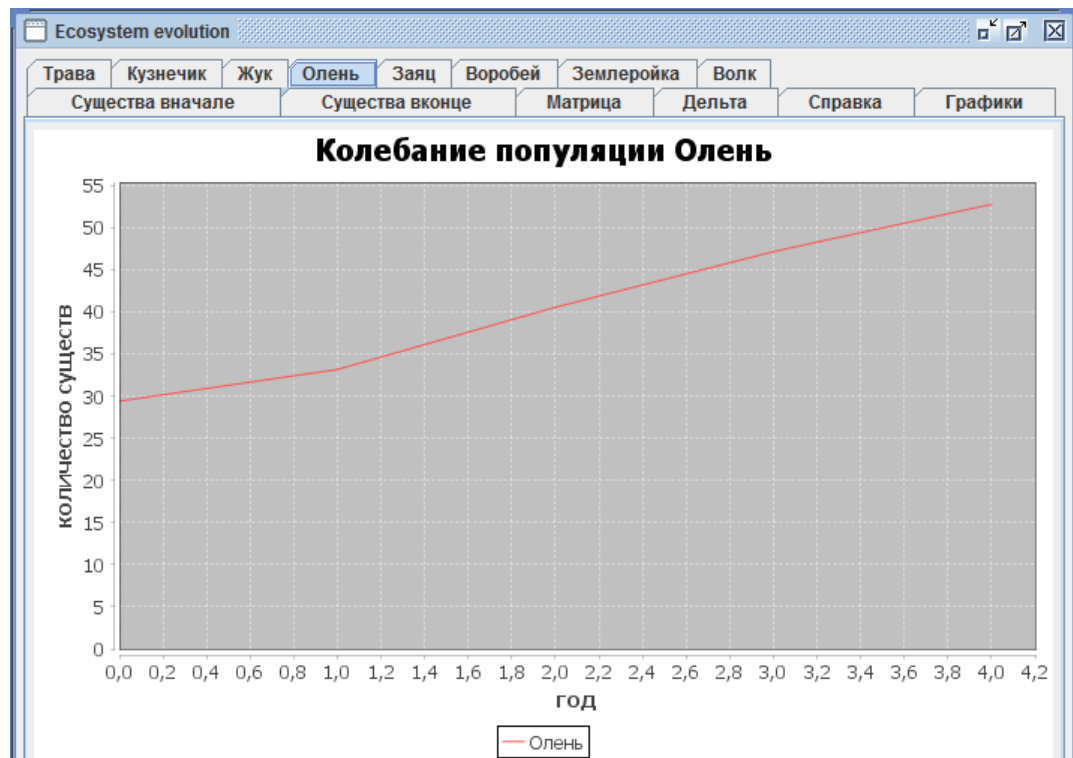


Рисунок 3.32 – Графік, що відображає коливання популяції оленів.

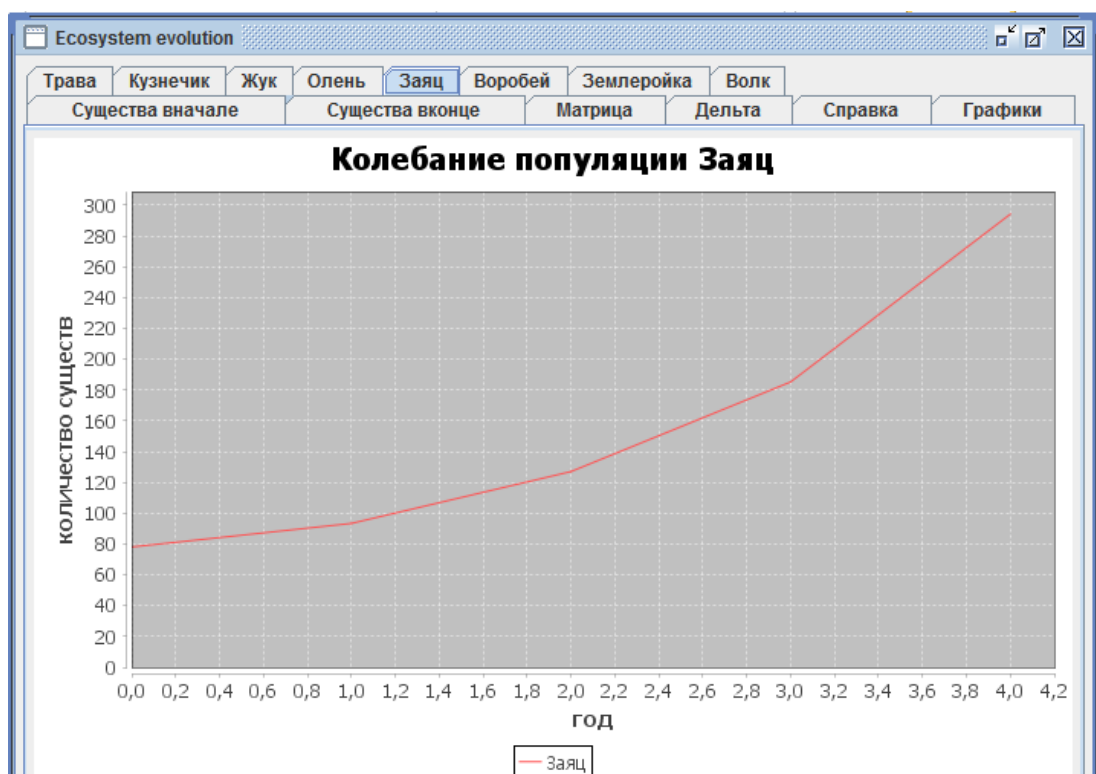


Рисунок 3.33 – Графік, що відображає коливання популяції зайців.

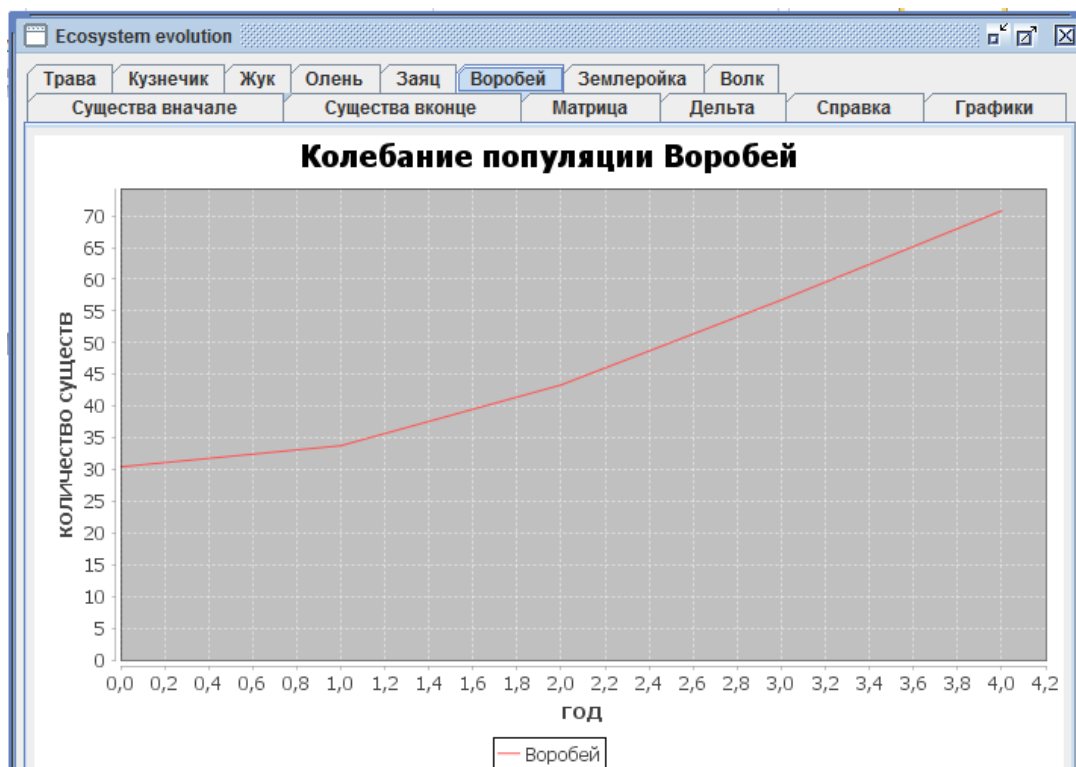


Рисунок 3.34 – Графік, що відображає коливання популяції горобців.

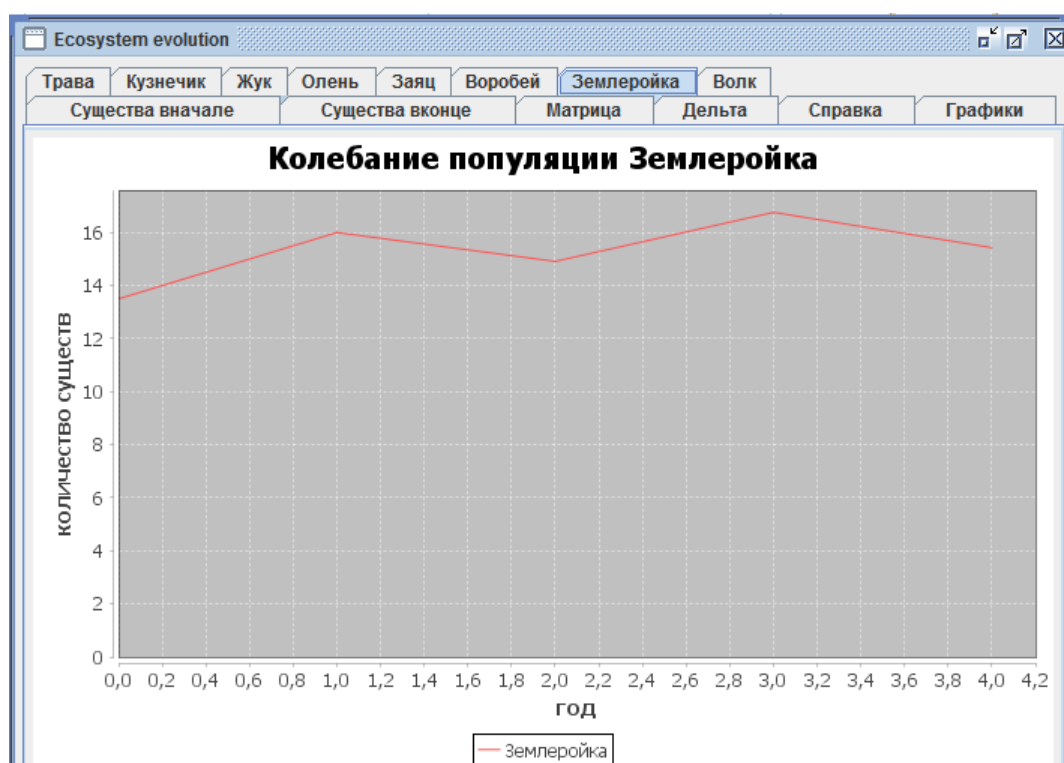


Рисунок 3.35 – Графік, що відображає коливання популяції землерийок.

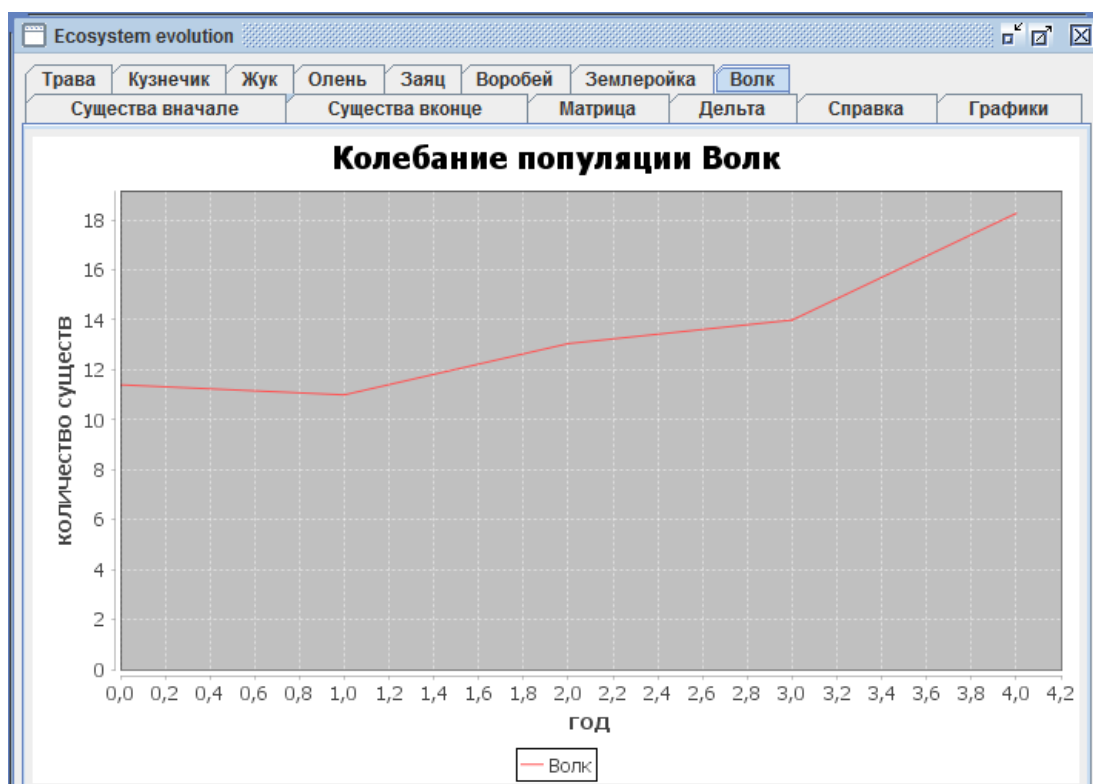


Рисунок 3.36 – Графік, що відображає коливання популяції вовків.

3.2.5 Результати моделювання екосистеми за умови істотного промислового видобутку оленів та відстрілу вовків

На Рисунках 3.37 – 3.50 показано результати роботи програмного продукту, за умови вказання користувачем істотного промислового видобутку оленів та відстрілу вовків.



Рисунок 3.37 – Кількість видів з їх коефіцієнтами до симуляції.

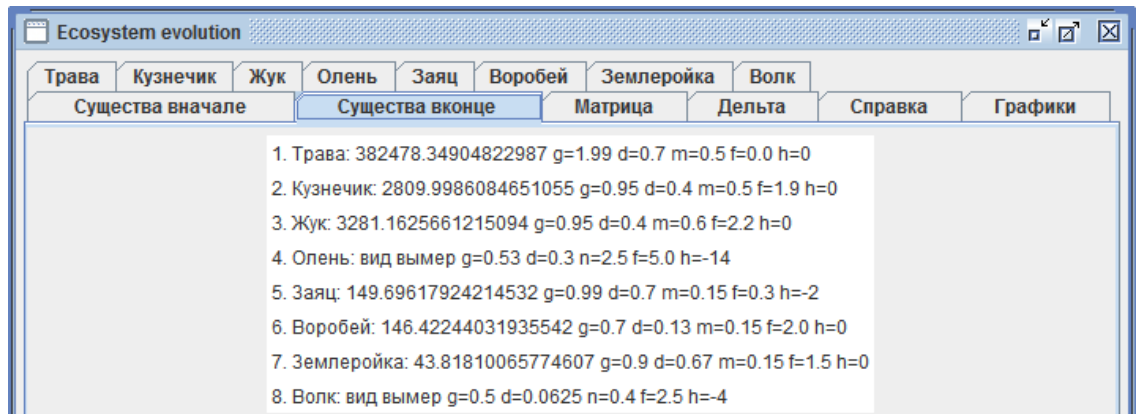


Рисунок 3.38 – Кількість видів з їх коефіцієнтами після симуляції.

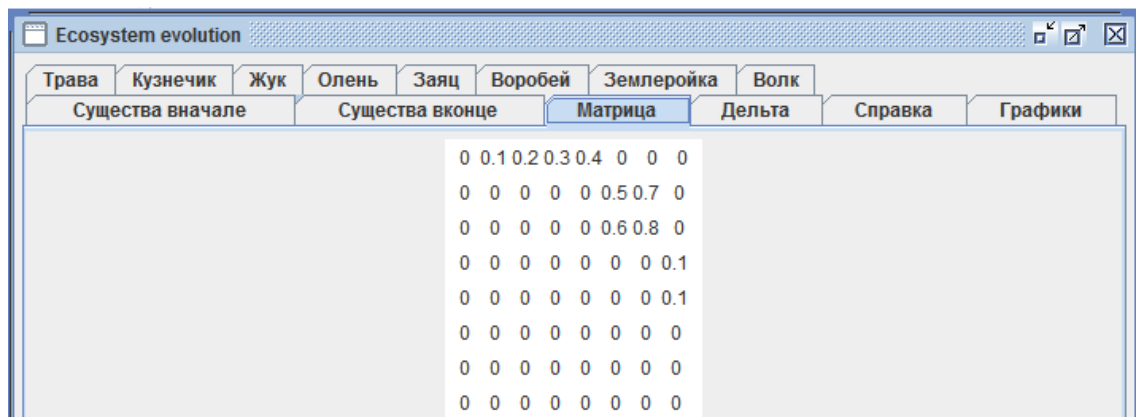


Рисунок 3.39 – Матриця коефіцієнтів виїдання.

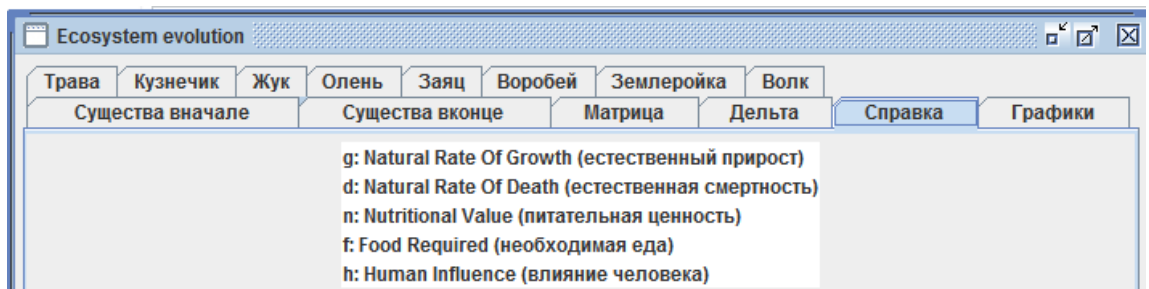


Рисунок 3.40 – Опис призначення коефіцієнтів.



Рисунок 3.41 – Загальний графік видозміни популяцій за Δt років.

Ecosystem evolution						
Трава	Кузнечик	Жук	Олень	Заяц	Воробей	Землеройка
Волк	Существа в начале		Существа в конце		Матрица	Дельта
Справка		Графики				
год: 1						
1. 13844.457642420282						
2. 168.55755479215435						
3. 234.98044735086384						
4. -11.128370157774263						
5. 7.152577361684001						
6. 10.126439724146845						
7. 1.7735341354111296						
8. -6.827348034611836						
год: 2						
1. 22477.22984524887						
2. 330.336714104388						
3. 326.1455159455915						
4. -12.82643335720427						
5. 12.387790484732209						
6. 13.712471859373803						
7. 5.8841983229442985						
8. -5.852221082427542						
год: 3						
1. 59395.811317669046						
2. 411.9865973780516						
3. 418.4830416693411						
4. -14.080948817035276						
5. 22.605706546309683						
6. 25.30432562046937						
7. 6.099757320282088						
8. -4.41166848549907						
год: 4						
1. 84452.13804140242						
2. 569.7061995578129						
3. 750.7449291814746						
4. -16.291592483585408						
5. 23.56576315951229						
6. 32.799902707812						
7. 6.174364388615227						
8. -4.796197327982194						
год: 5						
1. 192308.71220148925						
2. 829.4115426326989						
3. 1050.808631974238						
4. -20.538112853629592						
5. 33.984341689907154						
6. 44.47930040755339						
7. 5.886246490493337						
8. -6.7641141564443155						

Рисунок 3.42 – Список ΔN_i всіх видів за всі роки симуляції.

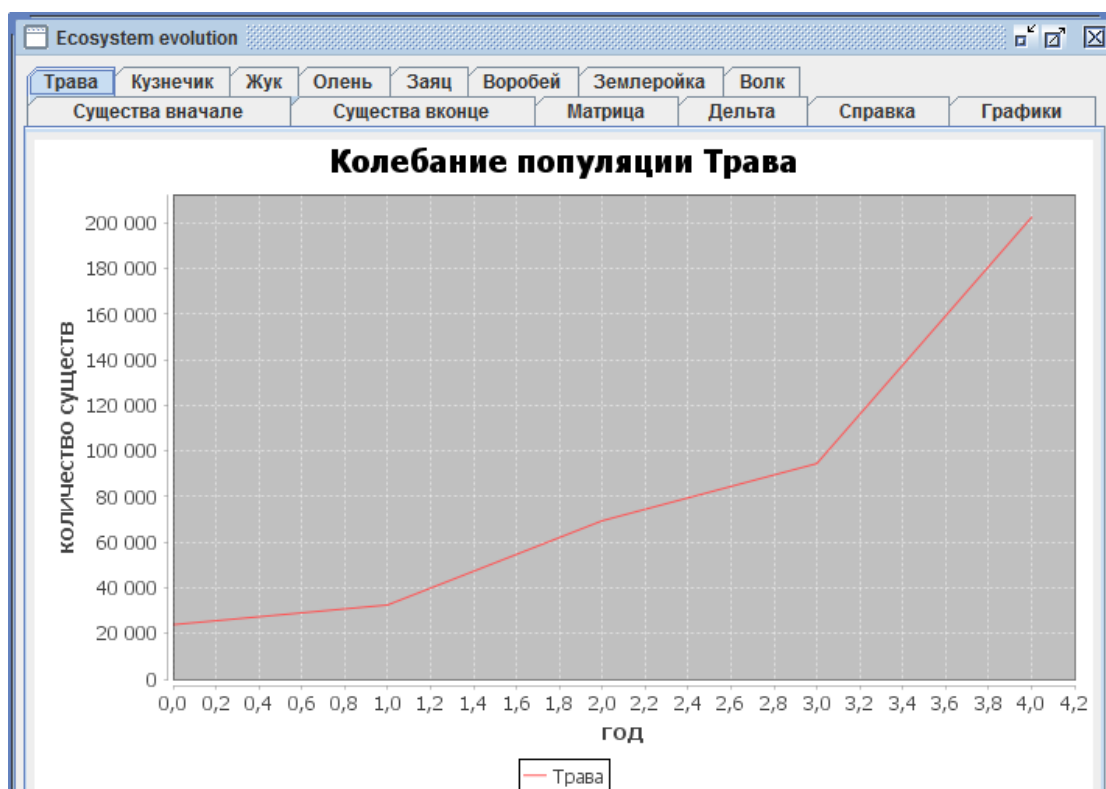


Рисунок 3.43 – Графік, що відображає коливання популяції трави.

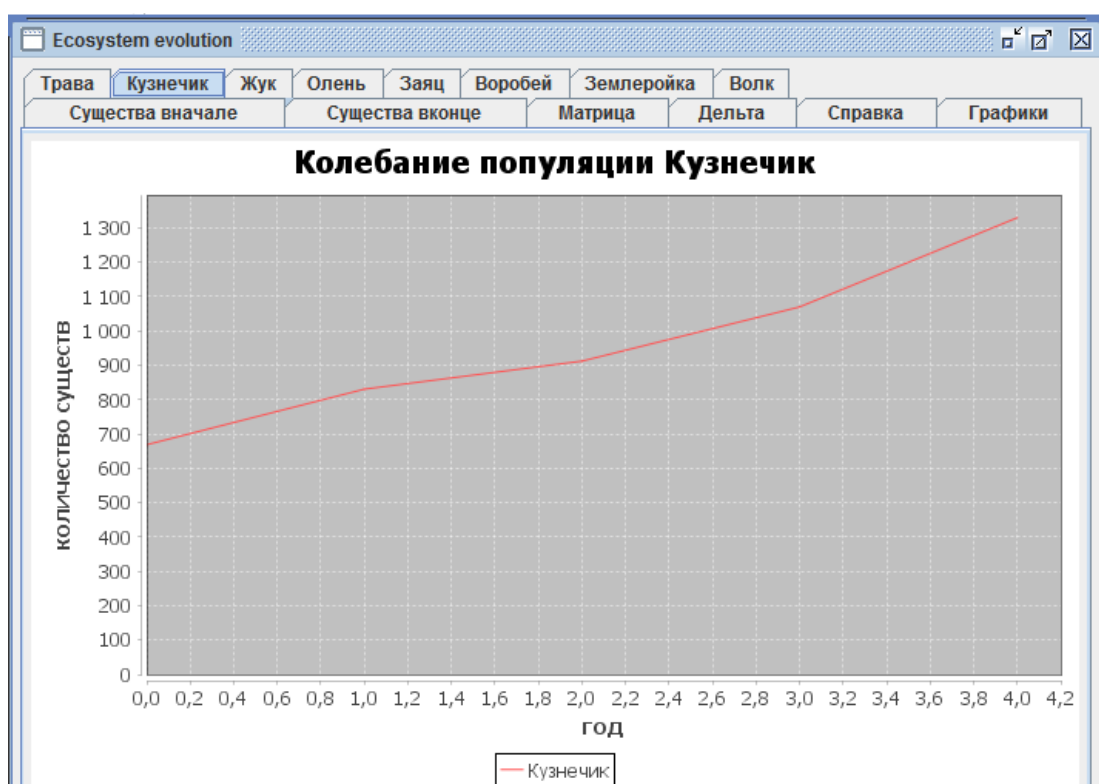


Рисунок 3.44 – Графік, що відображає коливання популяції коників.

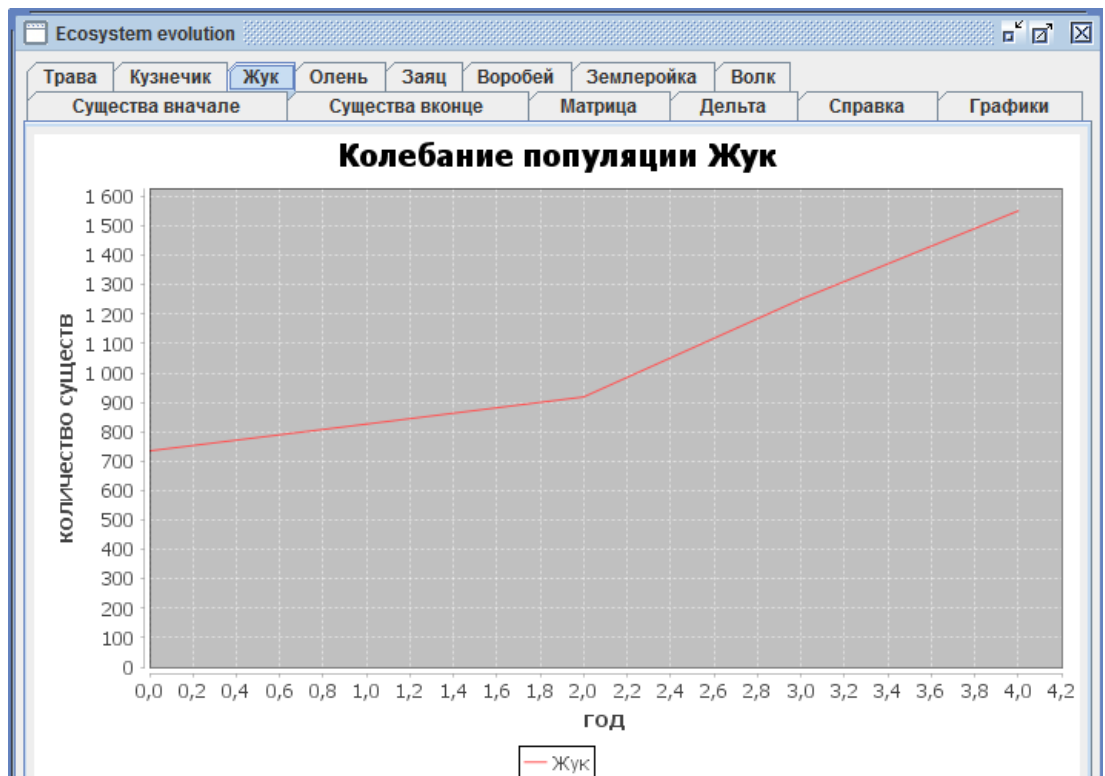


Рисунок 3.45 – Графік, що відображає коливання популяції жуків.

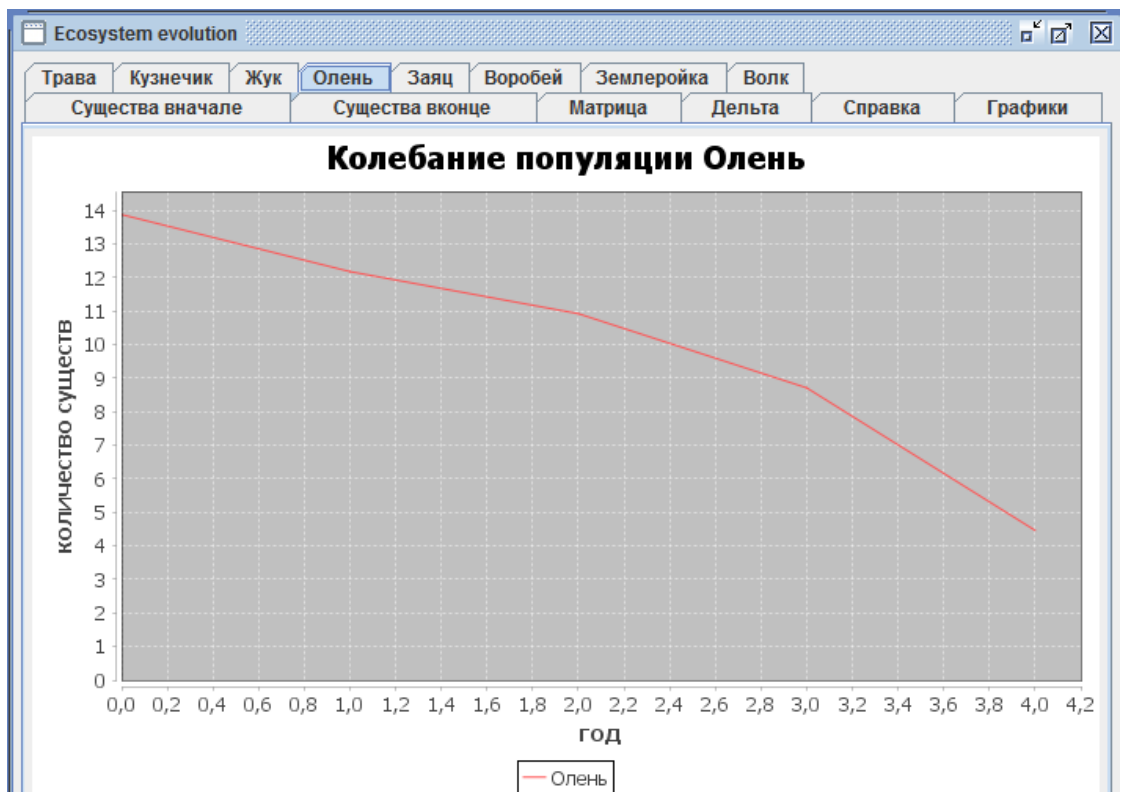


Рисунок 3.46 – Графік, що відображає коливання популяції оленів.

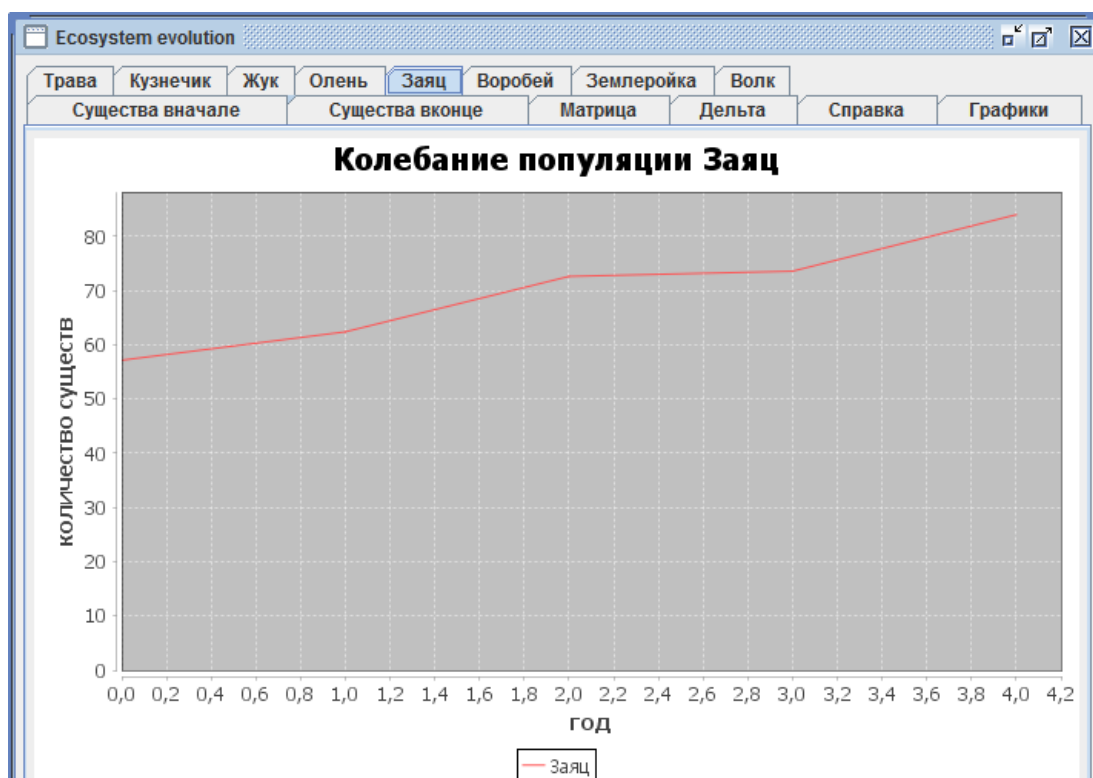


Рисунок 3.47 – Графік, що відображає коливання популяції зайців.

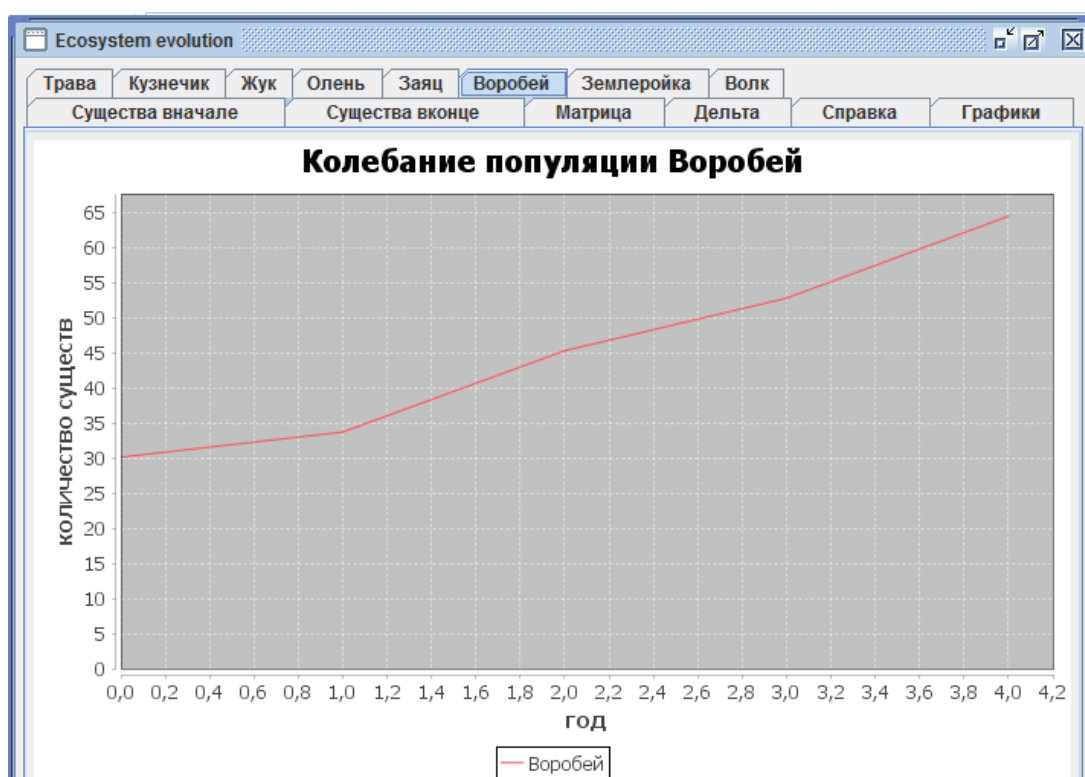


Рисунок 3.48 – Графік, що відображає коливання популяції горобців.

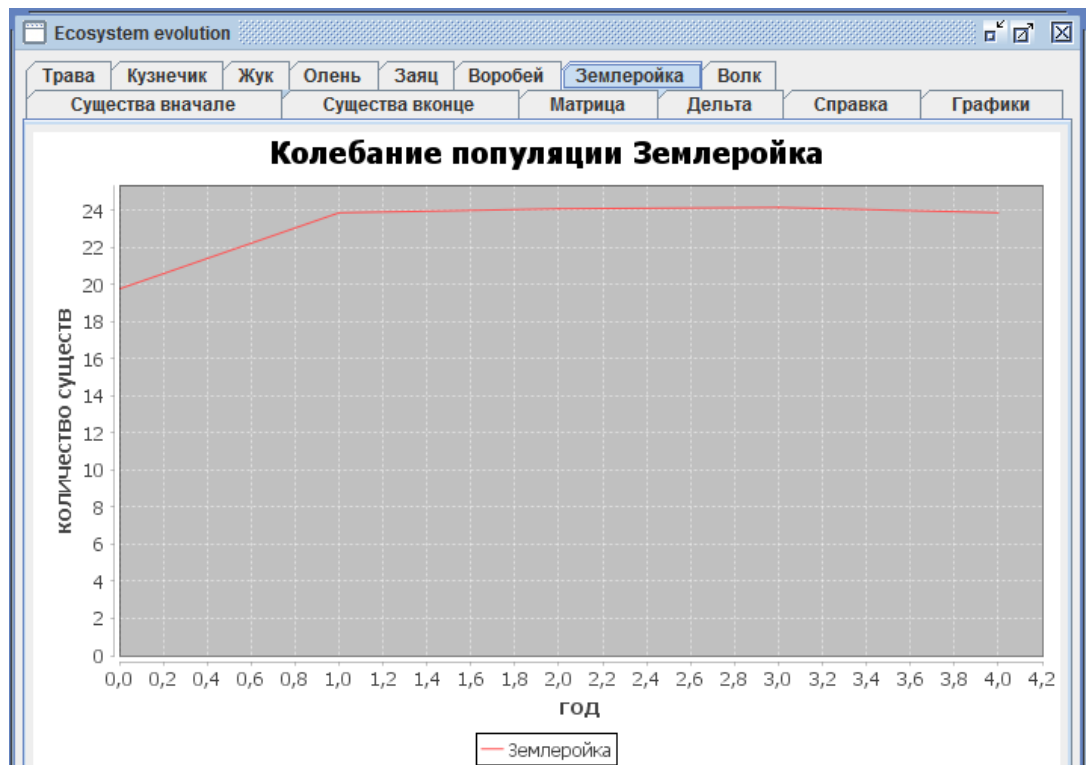


Рисунок 3.49 – Графік, що відображає коливання популяції землерийок.

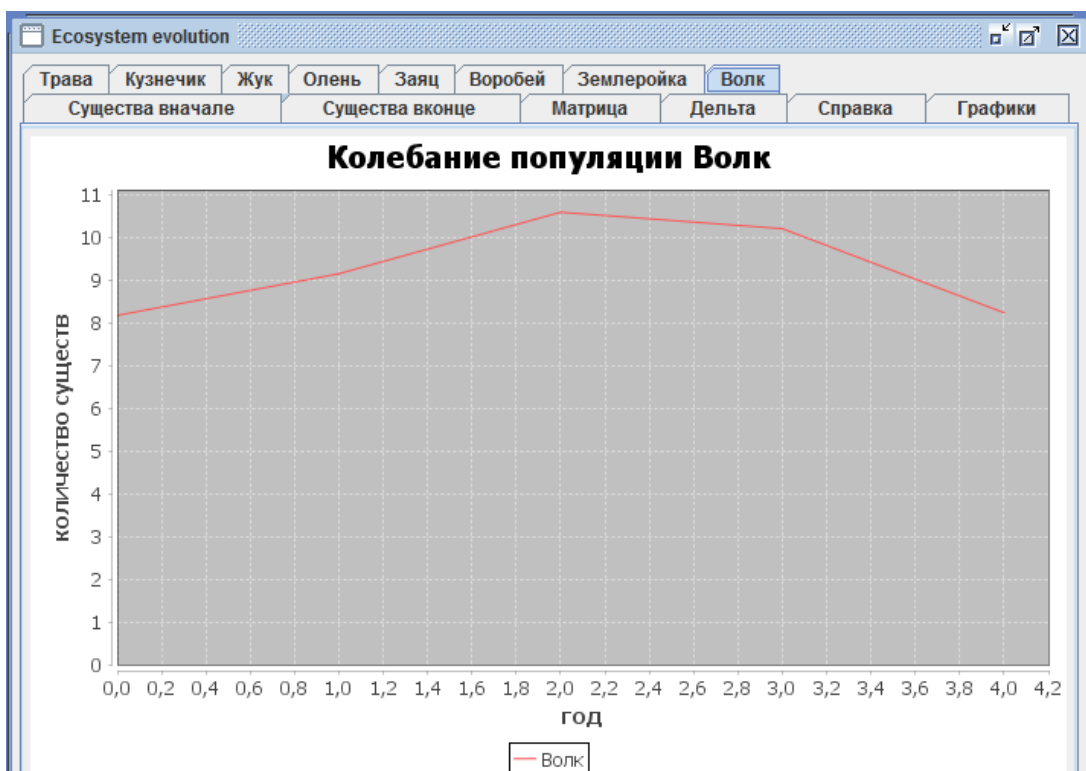


Рисунок 3.50 – Графік, що відображає коливання популяції вовків.

3.3 Висновки

У даному розділі визначено структуру програмного продукту:

- побудовано UML-діаграму класів;
- проведено опис класів бібліотек Swing та JFreeChart, що були використані при розробці програмного продукту;
- проведено опис створених класів логіки та обробки даних;
- проведено опис створених класів GUI.

Також, за допомогою розробленого програмного продукту, змодельовано еволюцію екосистеми лісу за різних вхідних параметрів:

- за умови нестачі трави;
- відносно нормальних умов;
- за умови істотного промислового вилову оленів та відстрілу вовків.

За результатами роботи можна зробити висновок, що майбутній користувач програмного продукту матиме можливість моделювати екосистему саме так, як передбачає модифікована модель Вольтерри-Лотки з пункту 2.1.2.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Вступ

У даному розділі проводиться оцінка основних характеристик програмного продукту. Інтерфейс користувача був розроблений за допомогою мови програмування Java у середовищі розробки IntelliJ IDEA (Community Edition) 2017.1.3.

Програмний продукт кросплатформенний.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.
- для кожної функції визначаються повні річні витрати й кількість робочих часів.
- для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.
- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.2 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на різних платформах;
- забезпечувати високу швидкість обробки великих об'ємів даних у реальному часі;

- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використання його як модуля;
- передбачати мінімальні витрати на впровадження програмного продукту.

4.2.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – вибір IDE;

F_3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1 :

- а) мова програмування Java;
- б) мова програмування JavaScript;

Функція F_2 :

- а) IntelliJ IDEA;
- б) Sublime Text 3.

Функція F_3 :

- а) інтерфейс користувача, створений за технологією Swing;
- б) інтерфейс користувача, створений за технологією Bootstrap.

4.2.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (Рисунок 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (Таблиця 4.1).

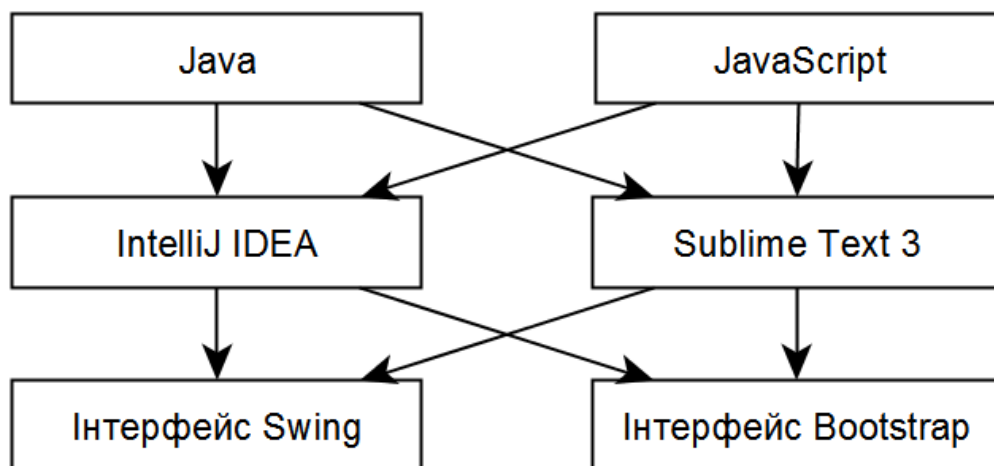


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	А	Явна типізація даних	Написання коду займає більше часу
	Б	Написання коду займає менше часу	Немає явної типізації даних
F2	А	Середовище розробки, проект збирається і компілюється автоматично, показуються помилки компіляції, в разі їх виявлення	Під час написання коду витрачаються додаткові ресурси системи на обробку інформації в середовищі розробки
	Б	Текстовий редактор (під час написання коду не витрачаються додаткові ресурси системи)	Збирати і компілювати проект потрібно сторонніми засобами
F3	А	Використання як самостійне вікно	Важчий у створенні
	Б	Легкий у створенні	Використання тільки для WEB-додатків

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1:

Оскільки розрахунки проводяться над системами диференціальних рівнянь, то точність і строга типізація є необхідними, тому варіант б) має бути відкинтий.

Функція F2:

Оскільки немає апаратних обмежень, а збирати і компілювати проект сторонніми засоби досить трудомістко, то варіант б) має бути відкинтий.

Функція F3:

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

F1a – F2a – F3a

F1a – F2a – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів ПП

4.3.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X1 – швидкодія мови програмування;

X2 – об'єм пам'яті для збереження даних;

X3 – час обробки даних;

X4 – потенційний об'єм програмного коду;

X5 – складність створення графіків.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об'єм пам'яті в оперативній пам'яті персонального комп'ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

X5: Відображає складність написання коду графічної частини.

4.3.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 4.2 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія програми	X1	мс	6000	4200	800
Об'єм пам'яті для збереження даних	X2	Мб	32	16	8
Точність результату	X3	%	10	5	0
Потенційний об'єм програмного коду	X4	кількість строк коду	2000	1000	500
Складність створення графіків	X5	мкс	200	70	20

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.6.

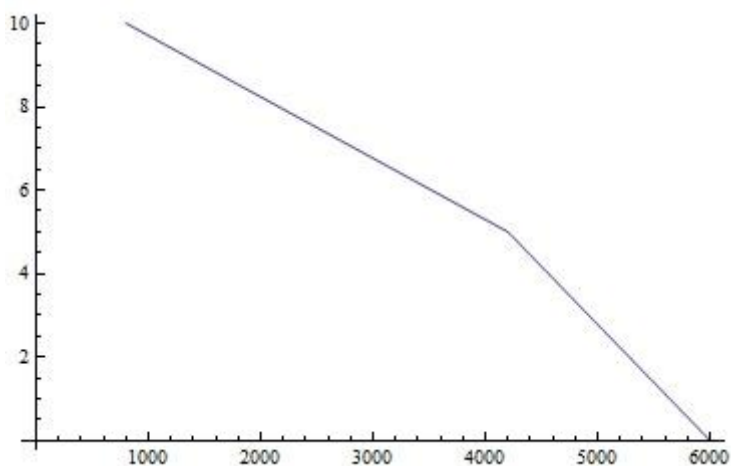


Рисунок 4.2 – X1, швидкодія мови програмування

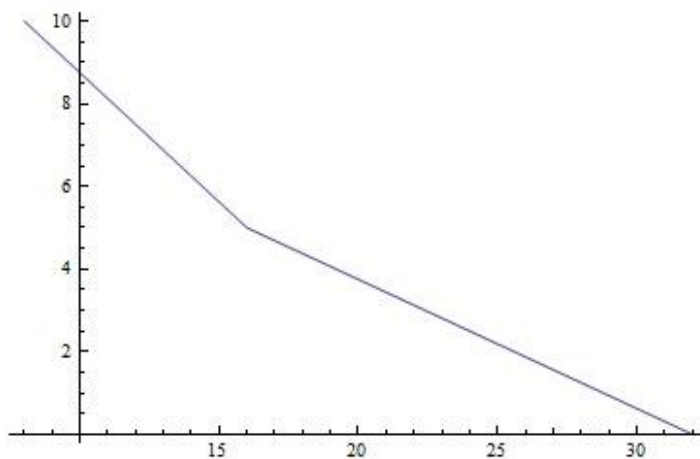


Рисунок 4.3– X2, об'єм пам'яті для збереження даних

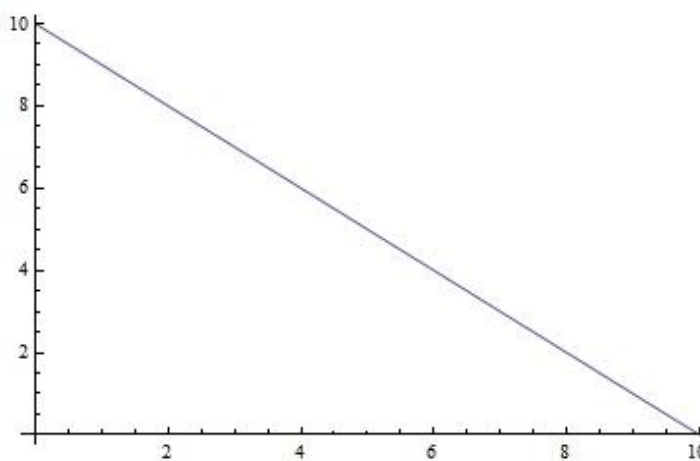


Рисунок 4.4 – X3, час обробки даних алгоритмом

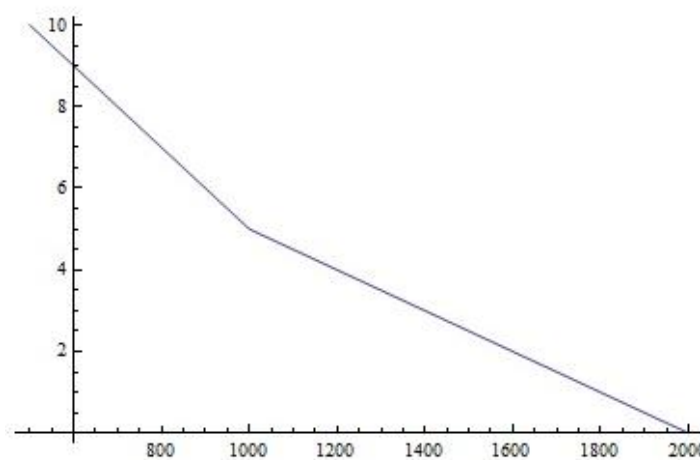


Рисунок 4.5 – X4, потенційний об'єм програмного коду

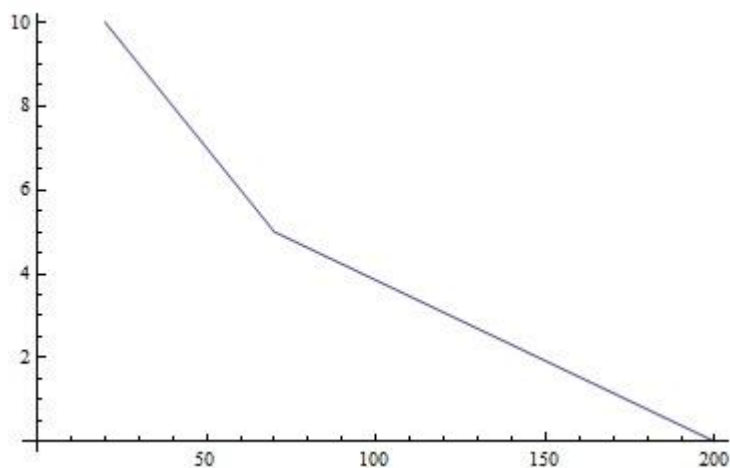


Рисунок 4.6 – X5, складність створення графіків

4.3.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія програми	мс	5	6	5	6	5	5	6	38	17	289
X2	Об'єм пам'яті для збереження даних	Мб	3	2	3	3	3	4	2	20	-1	1
X3	Точність результату	%	2	2	3	2	2	2	3	16	-5	25
X4	Потенційний об'єм програмного коду	кількість строк коду	3	2	2	2	3	2	1	15	-6	36
X5	Час розрахунку значення функції	мкс	2	3	2	2	2	2	3	16	-5	25
	Разом		5	5	5	5	5	5	5	105	0	376

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 21$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 376$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420,75}{7^2(5^3 - 5)} = 0,767 > W_k = 0,67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X1 і X5	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	=	=	>	>	>	<	>	1,5
X2 і X4	=	=	>	=	=	>	>	=	1
X2 і X5	>	<	>	>	>	>	<	>	1,5
X3 і X4	<	=	>	=	<	=	>	=	1
X3 і X5	=	<	>	=	=	=	=	=	1
X4 і X5	>	<	=	=	>	=	<	=	1

Числове значення, що визначає ступінь переваги і-го параметра над j-тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 & \text{при } x_i > x_j \\ 1.0 & \text{при } x_i = x_j \\ 0.5 & \text{при } x_i < x_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості K_{bi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i}, \text{ де } b_i = \sum_{i=1}^N a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{bi} = \frac{b'_i}{\sum_{i=1}^n b'_i}, \text{ де } b'_i = \sum_{i=1}^N a_{ij} b_j.$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j					Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	X5	b_i	K_{bi}	b_i^1	K_{bi}^1	b_i^2	K_{bi}^2
X1	1,0	1,5	1,5	0,5	1,5	7,0	0,28	34	0,287	160,75	0,287
X2	0,5	1,0	1,5	0,5	1,5	5,5	0,22	25,5	0,215	120,25	0,215
X3	0,5	0,5	1,0	0,5	1,0	4,0	0,16	18,75	0,158	88,75	0,158
X4	0,5	1,0	1,0	0,5	1,0	4,5	0,18	21,5	0,181	101,5	0,181
X5	0,5	0,5	1,0	0,5	1,0	4,0	0,16	18,75	0,158	88,75	0,158
Всього:						25	1	118,5	1	560	1

4.4 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X2(об'єм пам'яті для збереження даних) та X1 (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3 (час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так(таблиця 4.6):

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j},$$

де n – кількість параметрів; K_{ei} – коефіцієнт вагомості i-го параметра; B_i – оцінка i-го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X4)	А	500	6,8	0,181	1,231
F2(X3)	А	3	7,2	0,158	1,138
F3(X2,X1)	А	16	5	0,215	1,075
	Б	800	10	0,287	2,870

За даними з таблиці 4.6, за формулою:

$$K_K = K_{TY}[F_{1k}] + K_{TY}[F_{2k}] + \dots + K_{TY}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

$$KK1 = 1,231+1,138+1,075=3,444;$$

$$KK2 = 1,231+1,138+2,870=5,239.$$

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.5 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_0 = T_P \cdot K_{\Pi} \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М},$$

де T_P – трудомісткість розробки ПП; K_{Π} – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови

програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 90$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{П} = 1.7$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.8$. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_P = 27$ людино-днів, $K_{П} = 0.9$, $K_{СК} = 1$, $K_{СТ} = 0.8$:

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328.64 \text{ людино-годин;}$$

$$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин;}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 6000 грн., один фінансовий аналітик з окладом 9000грн. Визначимо зарплату за годину за формулою:

$$C_q = \frac{M}{T_m \cdot t} \text{ грн.},$$

де M – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$C_q = \frac{20000 + 20000 + 25000}{3 \cdot 21 \cdot 8} = 128,97 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$C_{зп} = C_q \cdot T_i \cdot K_d,$$

де C_q – величина погодинної оплати праці програміста; T_i – трудомісткість відповідного завдання; K_d – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I.} \quad C_{зп} = 128,97 \cdot 1328,64 \cdot 1,2 = 205622,86 \text{ грн.}$$

$$\text{II.} \quad C_{зп} = 128,97 \cdot 1345,52 \cdot 1,2 = 208238,06 \text{ грн.}$$

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 36,77%:

$$\text{I.} \quad C_{вд} = C_{зп} \cdot 0,22 = 205622,86 \cdot 0,22 = 45237,03 \text{ грн.}$$

$$\text{II.} \quad C_{вд} = C_{зп} \cdot 0,22 = 208238,06 \cdot 0,22 = 45812,37 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (C_M)

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 20000 \cdot 0,2 = 48000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{3П} = C_{\Gamma} \cdot (1 + K_3) = 48000 \cdot (1 + 0,2) = 57600 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$C_{ВІД} = C_{3П} \cdot 0,22 = 57600 \cdot 0,22 = 12672 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 30000 грн.

$$C_A = K_{TM} \cdot K_A \cdot C_{ПР} = 1,15 \cdot 0,25 \cdot 30000 = 8625 \text{ грн.,}$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; K_A – річна норма амортизації; $C_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot C_{ПР} \cdot K_P = 1,15 \cdot 30000 \cdot 0,05 = 1725 \text{ грн.,}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$T_{ЕФ} = (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 104 - 8 - 16) \cdot 8 \cdot 0,9 = 1706,4$ годин, де D_K – календарна кількість днів у році; D_B , D_C – відповідно кількість вихідних та святкових днів; D_P – кількість днів планових ремонтів устаткування; t_3 – кількість робочих годин в день; K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{ЕЛ} = T_{ЕФ} \cdot N_C \cdot K_3 \cdot C_{ЕН} = 1706,4 \cdot 1,94 = 3310,42 \text{ грн.,}$$

де N_C – середньо-споживча потужність приладу; K_3 – коефіцієнтом зайнятості приладу; C_{EH} – тариф за 1 кВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = C_{ПР} \cdot 0,67 = 30000 \cdot 0,67 = 20100 \text{ грн.}$$

$$C_{EKC} = C_{ЗП} + C_{ВІД} + C_A + C_P + C_{ЕЛ} + C_H$$

$$C_{EKC} = 57600 + 12672 + 8625 + 1725 + 3310,42 + 20100 = 104032,42 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{M-Г} = C_{EKC} / T_{ЕФ} = 104032,42 / 1706,4 = 60,97 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{M-Г} \cdot T$$

$$I. \quad C_M = 64,28 \cdot 1328,64 = 85404,98 \text{ грн.};$$

$$II. \quad C_M = 64,28 \cdot 1345,52 = 86490,03 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0,67$$

$$I. \quad C_H = 205622,86 \cdot 0,67 = 137767,32 \text{ грн.};$$

$$II. \quad C_H = 208238,06 \cdot 0,67 = 139519,50 \text{ грн.};$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_M + C_H$$

- I. $C_{\text{ПП}} = 205622,86 + 45237,03 + 85404,98 + 137767,32 = 474032,19$ грн.
- II. $C_{\text{ПП}} = 208238,06 + 45812,37 + 86490,03 + 139519,50 = 480059,96$ грн.

4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{\text{К}j} / C_{\Phi j},$$

$$K_{\text{ТЕР}1} = 5,214 / 474032,19 = 1,10 \cdot 10^{-5};$$

$$K_{\text{ТЕР}2} = 3,569 / 480059,96 = 0,74 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР}1} = 1,1 \cdot 10^{-5}$.

4.7 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишилися після першого відбору двох варіантів виконання програмного комплексу, оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{\text{TEP}} = 1,1 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Java;
- IDE – IntelliJ IDEA;
- інтерфейс користувача, створений за технологією Swing.

Даний варіант виконання програмного продукту дає хороші показники швидкодії, чудову візуалізацію та високу точність результатів.

ВИСНОВКИ

В ході дипломного проектування було досліджено моделювання еволюції екосистеми на базі рівнянь Вольтерри-Лотки.

Було досліджено предметну область, існуючі рішення (Ecosim, Ecolego) та описано загальні засоби моделювання екосистем, поняття трофічних пірамід. Визначено модель Вольтерри-Лотки (в двовимірному та багатовимірному випадках).

Після цього було модифіковано модель Вольтерри-Лотки так, щоб вона забезпечувала можливість створення конкретизованої багатовимірної моделі з великою кількістю видів та аналіз їх поведінки з можливістю втручання людини до системи.

Для програмної реалізації було обрано середовище розробки програмного продукту IntelliJ IDEA, Community Edition, v. 2017.1.3 та засоби Java для реалізації програмного продукту: JDK 8, Swing (для створення GUI), JFreeChart 1.0.19 (для побудови графіків).

Також було визначено і описано структуру програмного продукту:

- побудовано UML-діаграму класів;
- проведено опис класів бібліотек Swing та JFreeChart, що були використані при розробці програмного продукту;
- проведено опис створених класів логіки та обробки даних;
- проведено опис створених класів GUI.

За допомогою розробленого програмного продукту було змодельовано еволюцію екосистеми лісу за різних вхідних параметрів:

- за умови нестачі трави;
- за відносно нормальних умов;

- за умови істотного промислового вилову оленів та відстрілу вовків.

За результатами роботи програмного продукту можна зробити висновок, що майбутній користувач матиме можливість моделювати екосистему саме так, як передбачає модифікована модель Вольтерри-Лотки.

Було проведено повний функціонально-вартісний аналіз програмного продукту, після виконання якого можна зробити висновок, що альтернатив, що залишилися після першого відбору варіантів, оптимальним є перший варіант реалізації. У нього виявився найкращий показник техніко-економічного рівня якості $K_{\text{TEP}} = 1,1 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Java;
- IDE – IntelliJ IDEA;
- інтерфейс користувача, створений за технологією Swing.

Наукова новизна програмного продукту полягає в тому, що він створений за модифікованою моделлю Вольтерри-Лотки і розглядає багатовимірний випадок екосистеми з можливістю впливу людини на систему. А також – те, що програмний продукт не вимагає від користувача графічного створення системи (вводяться лише текстові дані).

Потенційні застосування та практична цінність результатів дипломної роботи:

Програмний продукт може бути використаний екологами для моделювання екосистем та аналізу проблем екосистем і пропонування рішень цих проблем на базі результатів роботи програми за різних вхідних даних;

Програмний продукт може бути використаний для аналізу небезпеки промислового видобутку тварин і прийняття рішень щодо можливої кількості такого видобутку, за якого екосистемі не буде нанесено шкоди.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вольтерра, В. Математическая теория борьбы за существование / В. Вольтерра. – пер. с итал. П.П. Лазарев – М.; Наука, 1976. – 288 с.
2. Горковенко, Н.Е. Математическое моделирование в экологии : курс лекций для аспирантов / Н.Е. Горковенко. – М.: КубГАУ, 2015. – 45 с.
3. Офіційний сайт компанії Ecolego. – Режим доступу:
<http://ecolego.facilia.se/ecolego/show/Ecolego>. – Дата доступу : 03.03.2017.
4. Офіційний сайт компаніїEcopath International(The Ecopath with Ecosim (EWE) approach). – Режим доступу: <http://ecopath.org/about/#toggle-id-3>. – Дата доступу : 05.03.2017.
5. Офіційний сайт компанії JFree (Javadoc). – Режим доступу:
<http://www.jfree.org/jfreechart/api/javadoc/>. – Дата доступу : 24.04.2017.
6. Офіційний сайт компанії Oracle(Java Documentation). – Режим доступу:
<https://docs.oracle.com/en/java/>. – Дата доступу : 16.04.2017.
7. Law, A.M. Simulation modeling and analysis / M.A. Law, D.W. Kelton. – Second edition. – McGraw-Hill, 1991. – 155 p.
8. Tian, S. Amathematicalmodelforpopulationdynamicsofantibiotictreatment/ S. Tian. – GeorgiaStateUniversity, 2014. – 37 p.