НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

На правах рукопису
УДК 004.942

До захисту допущено
Завідувач кафедри ММСА

Оксана ТИМОЩУК

«___» _____ 2022 р.

Магістерська дисертація

на здобуття ступеня магістра за спеціальністю 124 «Системний аналіз»
на тему: «Математичні моделі в епідеміології»

Виконала:
студентка 2 курсу, групи КА-13мп
Клименко Анастасія Іллівна _____

Керівник: доцент кафедри ММСА,
к.ф.-м.н. Подколзін Гліб Борисович _____

Рецензент: Доцент кафедри МА та ТЙ,
КПІ ім. Ігоря Сікорського,
к.ф.-м.н. Орловський Ігор Володимирович _____

Засвідчую, що у цій магістерській дисертації
немає запозичень з праць інших авторів
без відповідних посилань
Студент (*підпис*): _____

Київ
2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Рівень вищої освіти — другий (магістерський)
Спеціальність (ОПП) — 124 «Системний аналіз» («Системний аналіз фінансового ринку)

ЗАТВЕРДЖУЮ
Завідувач кафедри ММСА
Оксана ТИМОЩУК
«___» _____ 2022 р.

ЗАВДАННЯ
на магістерську дисертацію студентці Клименко Анастасії Іллівні

1. Тема дисертації:« Математичні моделі в епідеміології», науковий керівник дисертації Подколзін Гліб Борисович, к.ф.-м.н., доцент, затверджені наказом по університету від «03» листопада 2022 р., № 4046-с.

2. Термін подання студентом дисертації: 12.12.2022 р.

3. Об'єкт дослідження: Процеси, пов'язані з динамікою поширення інфекційних хвороб.

4. Предмет дослідження: Засоби для дослідження еволюційної динаміки поширення інфекційних хвороб.

5. Перелік завдань, які потрібно розробити:
1) Дослідити актуальність обраної теми.
2) Здійснити огляд технічної літератури за темою дослідження.
3) Зробити дослідження та аналіз існуючих методів та моделей.
4) Здійснити огляд технічної літератури за темою математичні моделі в епідеміології.
5) Модифікувати вже існуючу модель, щоб вона могла бути використаною для більшої кількості випадків.
6) Провести аналіз результатів.

7) Підготувати ілюстративний матеріал.
8) Оформити пояснювальну записку.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:

1) Презентація.
2) Практичний приклад програми, розробленої в ході дослідження.
3) Графіки з результатами.
4) Таблиці у розділі стартап-проекту.

7. Орієнтовний перелік публікацій:

1. Клименко А.І., Подколзін Г.Б. Аналіз розповсюдження COVID-19 на території України з використанням модифікованої SEIRD-моделі. Університетський науковий збірник «Системні науки та інформатика», 22-29 листопада, Київ. — К.: НН ІПСА КПІ ім. Ігоря Сікорського, 2022. С.140-146.

8. Дата видачі завдання: 01.09.2022

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Отримання завдання на магістерську дисертацію.	01.09.2022 — 05.09.2022	Виконано
2.	Огляд технічної літератури за темою.	06.09.2022 — 12.09.2022	Виконано
3.	Концептуальний вступ дисертації. Формулювання об’єкту, предмета, цілі, завдань, новизни, практичної значущості результатів. Дослідження актуальності обраної теми.	13.09.2022 — 19.09.2022	Виконано
4.	Перший розділ. Дослідження предметної області.	20.09.2022 — 26.09.2022	Виконано
5.	Другий розділ. Математичні моделі в епідеміології.	27.09.2022 — 03.10.2022	Виконано
6.	Третій розділ. Розробка модифікованої моделі та програмного продукту.	04.10.2022 — 17.10.2022	Виконано
7.	Аналіз результатів.	18.10.2022 — 24.10.2022	Виконано
8.	Четвертий розділ. Проведення аналізу ринкових можливостей стартап-проекту.	25.10.2022 — 31.10.2022	Виконано
9.	Підготовка ілюстративного матеріалу.	01.11.2022 — 14.11.2022	Виконано
10.	Оформлення пояснювальної записки.	15.11.2022 — 18.11.2022	Виконано

Студентка Анастасія КЛИМЕНКО

Науковий керівник дисертації Гліб ПОДКОЛЗІН

РЕФЕРАТ

Магістерська дисертація: 174 с., 22 рис., 27 табл., 1 дод. та 67 джерел.

МАТЕМАТИЧНІ МОДЕЛІ, ЕПІДЕМІОЛОГІЯ, МОДЕЛЮВАННЯ COVID-19.

Тема: Математичні моделі в епідеміології.

У роботі розглянуто моделювання та побудову математичних моделей в епідеміології.

Об’єкт дослідження: Процеси, пов’язані з динамікою поширення інфекційних хвороб.

Предмет дослідження: Засоби для дослідження еволюційної динаміки поширення інфекційних хвороб.

Мета роботи: моделювання еволюційної динаміки поширення інфекційних хвороб.

Методи дослідження: аналіз, порівняння, статистики, дослідження, експерименту, математичного моделювання.

Було показано актуальність вивчення та моделювання інфекційних хвороб. Було розглянуто математичні моделі в епідеміології, еволюції та екології. Було створено програмний продукт на мові програмування Python та виконано моделювання.

В роботі використовувалися статистичні дані України з COVID-19.

ABSTRACT

Master’s thesis: 174 p., 22 fig., 27 tables, 1 appendice, 67 sources.

MATHEMATICAL MODELS, EPIDEMIOLOGY, SIMULATION OF COVID-19.

Theme: Mathematical models in epidemiology.

The work considers modeling and construction of mathematical models in epidemiology.

Object of research: Processes related to the dynamics of the spread of infectious diseases.

Subject of research: Means for studying the evolutionary dynamics of the spread of infectious diseases.

The purpose of the work: modeling of the evolutionary dynamics of the spread of infectious diseases.

Methods of research: analysis, comparison, statistics, research, experiment, mathematical modeling.

The relevance of studying and modeling infectious diseases was shown. Mathematical models in epidemiology, evolution and ecology were considered. A software product on Python was created and simulations were performed.

Statistical data of Ukraine on COVID-19 were used in the work.

ЗМІСТ

ВСТУП 9
1.1 Дискретні методи моделювання в екології 13
1.1.1 Моделі динаміки популяції 14
1.1.2 Урбаністичні моделі 21
1.2 Дискретні методи моделювання в еволюції 29
1.2.1 Популяційна генетика 30
1.2.2 Природний відбір 33
1.2.3 Баланс відбору та мутації 34
1.2.4 Випадковий дрейф 35
1.2.4 Міграція 37
1.2.5 Невипадкове спаровування 39
1.3 Висновки до розділу 1 40
РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ ПОШИРЕННЯ ІНФЕКЦІЙНИХ ХВОРОБ 42
2.1 Модель SIR 45
2.2 Модель SIS 51
2.3 Модель SRID 52
2.4 Модель SEIR 54
2.5 Модель SEIRD 57
2.6 Висновки до розділу 2 59
РОЗДІЛ 3 АНАЛІЗ ДАНИХ, ПОБУДОВА МОДИФІКОВАНИХ МОДЕЛЕЙ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 61
3.1 Запропоновані модифіковані моделі 62
3.2 Програмна реалізація 69
3.3 Модуляція ситуації для України 77
3.4 Висновки до розділу 3 84

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ 85
4.1 Вступ та опис стартап-проекту 85
4.2 Технологічний аудит проекту 88
4.4 Розроблення ринкової стратегії проекту 99
4.5 Розроблення маркетингової програми стартап-проекту 104
4.6 Висновки до розділу 4 108
ВИСНОВКИ 110
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 112
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ 119

ВСТУП

Актуальність теми. Протягом усієї історії людське суспільство жило з періодичними епідеміями та пандеміями. Численні спалахи захворювань призвели до смерті, суспільних потрясінь і економічних розладів. Таким чином, прогнозування того, як спалах може прогресувати, є важливим для пом'якшення його наслідків, і область епідеміологічного моделювання є центральною для цього.

Епідеміологічні моделі відіграли центральну роль у пандемії COVID-19, особливо коли потрібні були термінові рішення, а доступних доказів було мало. Вони використовувалися для прогнозування еволюції захворювання та для інформування при виробленні політики. Складність епідеміологічних моделей різна. Вони можуть бути простими детермінованими моделями або складними просторово-явними стохастичними моделюваннями. Підхід, обраний епідеміологами, залежить від кількох змінних, включаючи те, скільки відомо про епідеміологію захворювання, мету дослідження, кількість доступних даних і їх якість.

Епідеміологічні моделі зазвичай мають вхідні дані, що стосуються:

— характеристики вірусу, такі як інкубаційний період, заразність і віковий ризик симптомів, госпіталізації або смерті;

— характеристики населення, такі як вікова структура, етнічна приналежність та рівень імунітету від вакцинації або від попередньої інфекції;

— деякі оцінки частоти контактів між людьми, включаючи моделі взаємодії та соціальну структуру, а також те, як на них впливають різні заходи охорони здоров'я.

Отже, епідеміологічне моделювання є важливою частиною боротьби зі спалахами. Дані все більш складних моделей можуть допомогти визначити найефективніші втручання у сфері охорони здоров’я у 21 столітті.

У першому розділі роботи проведена постановка задачі, дослідження галузі та різних підходів до розв’язку задачі. Були розглянуті математичні моделі, що використовуються в екології та в еволюції Також були розглянуті урбаністичні моделі та дискретні методи моделювання в еволюції.

У другому розділі були описані загальні поняття, пов’язані із епідеміологічним моделюванням. Динаміка епідеміологічного процесу описана з допомогою системи диференційних рівнянь, розв’язки яких характеризують зміну чисельності у різних підгрупах популяції.

Третій розділ сформований з урахуванням програмної реалізації моделей. Були запрограмовані на Python шість моделей: стандартні SIR, SEIR, SEIRD моделі, дві модифіковані SEIR-моделі та запропонована у цій роботі модифікована SEIRD-модель.

У четвертому розділі йде мова про побудову власного стартап-проєкту. Він передбачає собою опис усіх особливостей, що потрібні для його застосування.

РОЗДІЛ 1 МЕТОДИ МОДЕЛЮВАННЯ В ЕКОЛОГІЇ ТА В ЕВОЛЮЦІЇ

Моделі — це зручні інструменти для узагальнення, організації та синтезу знань або даних у формах, що дозволяють формулювати кількісні, імовірнісні чи Байєсовські твердження про можливі або майбутні стани змодельованої сутності [1, с.15]. Моделювання має давню традицію в науках про Землю, де здатність передбачати екологічно значущі явища є давньою (наприклад, рух планет і зірок) [2, с.5]. Відтоді були розроблені моделі для вивчення явищ на багатьох рівнях складності, від фізіологічних систем і окремих організмів до цілих екосистем і земної кулі.

Попит на надійні прогнози, а отже, і моделі стрімко зростає, оскільки екологічні проблеми стають головною проблемою суспільства [2, с.4]. Крім того, величезні технологічні можливості для створення та обміну даними створюють значний тиск для асиміляції цих даних у послідовний синтез, який зазвичай забезпечується моделями.

Значна взаємодія між математикою, екологією та біологією почалася щонайменше століття тому в моделях динаміки населення [3, с.45]. Сьогодні математична екологія та популяційна біологія представляють собою злиття двох великих традицій [4, с.67], одна з яких походить від роботи Лотки та Вольтерри у їх вивченні динаміки популяції, а друга, розроблена Холдейном, Фішером і Райтом, — для вивчення механізмів успадкування [4, с.56].

Моделювання біологічних систем — це важлива частина математичної біології [4. с.34], що уявляє собою один із найактуальніших напрямків у дослідженнях. Зазвичай математичні моделі використовуються не тільки для перевірки гіпотез, розроблених на основі експериментальних даних, але й для

передбачення нових явищ та закономірностей, які здійснюються при використанні отриманих результатів тестування цих моделей [5, с.44].

Окремо можна виділити екологічне моделювання [4, с.5]. Екологічне моделювання може допомогти у впровадженні динамічного розвитку, математичних моделей і системного аналізу, які описують, як екологічні процеси можуть підтримувати стале управління ресурсами.

Рівень досягнення конкретної цілі з використанням математичної моделі залежить як від рівня знань про систему [6, с.66], так і від того, наскільки якісно здійснюється моделювання системи [5, с.76].

Окремо можна виділити моделювання еволюційної динаміки поширення інфекційних хвороб [7, с.24], що є актуальною темою у сучасному світі. Інфекційні захворювання — це захворювання, спричинені організмами, такими як бактерії, віруси, грибки або паразити, які передаються від заражених осіб здоровим і здатні до масового поширення [8, с.34-36].

Інфекційні захворювання накладають величезний тягар захворюваності та смертності на населення, особливо в країнах з низьким рівнем доходу [9, с.45]. Розуміння поширення інфекційних захворювань і розробка ефективних стратегій контролю все більше покладаються на математичні моделі [9, с.78], які описують процес передачі збудників через популяції. Наприклад, моделі використовуються для розуміння соціальних і біологічних факторів [4, 6], що спричиняють епідемії, для розробки стратегій втручання та прогнозування спалахів у майбутньому.

Для того, щоб успішно протистояти поширенню інфекції, необхідно аналізувати динаміку зростання захворюваності, розраховувати навантаження на охорону здоров'я та робити реалістичні прогнози [10, с.56]. Все це потребує розробки адекватної математичної моделі поширення епідемії [10, с.67], параметри якої легко підбираються зі статистичних даних та легко коригуються при зміні умов поширення. Математичне моделювання епідемій, що поширюються серед людей, має довгу історію як у вивченні математичних

властивостей та можливостей моделей, так і у моделюванні конкретних епідемій та контролі їх еволюції [11, с.67].

1.1 Дискретні методи моделювання в екології

Екологія — це наука про взаємозв'язки між живими організмами, включаючи людину, та їх фізичним середовищем [12, с.45].

Математична екологія — це область прикладної математики, яка займається застосуванням математичних концепцій, інструментів і методів, зазвичай у формі математичних моделей, до проблем, що виникають у динаміці популяції, екології та еволюції [12, с.12]. Математичне моделювання в екології відіграє особливу роль. Через дуже високу складність екологічних систем і непостійний характер умов навколишнього середовища регулярні повторювані експериментальні дослідження не завжди можливі [13, с.55]. Математичне моделювання створює віртуальну лабораторію, де можна перевіряти різні гіпотези та розглядати та детально вивчати різні сценарії екологічної динаміки, як доповнення (і іноді навіть як заміну) до емпіричних досліджень.

Одним із відповідних способів покращити наше розуміння функціонування екології та еволюції є надання більш формальних рамок. Вони виявилися цінними для прискорення та кращого контролю рішення процедури. Подібно до того, що відбувається в біології, звичайні диференціальні рівняння є домінуючою методологією моделювання екосистем. Недоліками таких моделей є те, що:

— вони зазвичай потребують кількісної оцінки різних параметрів (здебільшого невідомих) [14, с.34];

— вони не в змозі точно відобразити часовий масштаб, який зазвичай великий, необхідний для спостереження за екологічними процесами;

— крім того, аналітичні рішення зазвичай не існують [13, с.14], і моделі часто представляють усереднену та інколи нереалістичну поведінку екосистем.

З іншого боку, так як дискретні якісні моделі мають високий рівень спостережуваних процесів, вони дозволяють розгадувати заплутані причинно-наслідкові зв'язки між системою сутності (тобто матеріальні складові компоненти).

1.1.1 Моделі динаміки популяції

У деяких біологічних системах буде доречніше апелювати до термінів дискретного часу. Наприклад, якщо це моделювання розмноження тварин чи рослин, що трапляється протягом невеликого періоду раз на рік. У такому випадку часовий крок дискретного часу буде дорівнювати року.

Розглядається система дискретної часової динаміки в $\mathbb{R}^n$ (1.1):

$$x(t+1) = f\big(t, x(t)\big), t = \overline{t_0, T-1} \qquad (1.1)$$

Починаючи від початкового стану $x_0 \in \mathbb{R}^n$ у часі t_0.

Мальтус стверджував, що людська популяція p зростає швидше, ніж кількість запасів їжі [15, с.11]. І тоді бідність буде наслідком перенаселення. Він не передбачив існування межі в геометричному зростанні населення, заданому лінійним диференціальним рівнянням (1.2):

$$\frac{dp}{dt} = kp, \qquad (1.2)$$

де k — швидкість зростання;

p — популяція.

Модель (1.2) ще має назву експоненційної моделі [15, с.23], бо розв'язком має функцію (1.3):

$$P(t) = P(0)e^{kt}, \tag{1.3}$$

де *P(0)* — чисельність популяції у початковий момент часу (t=0);

При швидкості зростання більше нуля $k > 0$ буде нескінченний ріст популяції; при швидкості зростання менше нуля $k < 0$ — чисельність прямує до нуля, тобто популяція помре.

Вергюльст стверджував, що модель Мальтуса була надто спрощена, оскільки включала лише лінійні терміни, а обмеження все-таки існує, і воно є наслідком наявного матеріалу: землі та їжі [16. с.45]. Він встановив найпростішу гіпотезу, а саме те, що коефіцієнт зростання пропорційний відстані чисельності популяції від точки її насичення. Результатом є гальмівний член np, пропорційний квадрату розміру популяції, яка асимптотично прагне до постійної чисельності населення k/n (1.4):

$$\frac{dp}{dt} = kp - np^2, \tag{1.4}$$

де k — швидкість зростання;

$p(t)$ — популяція, що залежить від часу t;

n — коефіцієнт внутрішньовидової конкуренції (за їжу/ресурси тощо).

Рівняння (1.4) описує швидкість зростання чисельності популяції окремого виду. При малих p чисельність p росте експоненційно, проте при великих прямує до своєї границі.

Проте розмір популяції одного виду може мати фіксований інтервал між поколіннями або, можливо, фіксований інтервал між вимірюваннями. Наприклад, багато видів комах не мають збігів між послідовними поколіннями, і, отже, їхня популяція еволюціонує дискретних часових кроках. Така динаміка популяції описується послідовністю $\{p_n\}$, яка може бути змодельована логістичним різницевим рівнянням (1.5):

$$p_{n+1} = p_n + r_0 p_n(1 - \frac{p_n}{k}) \quad (1.5)$$

де r_0 — внутрішня швидкість росту;

$p(t)$ — популяція, що залежить від часу t;

n — кількість поколінь.

k — ємність популяції.

Вергюльст назвав це рівняння (1.4) логістичною функцією.

Наступного століття біолог Роберт Мей заявив, що розуміння логістичної моделі слід вважати важливою віхою в галузі нелінійних явищ [17, с.33].

Сформулюємо дискретне логістичне рівняння, тобто дискретну версію моделі Вергюльста, для еволюції популяції виду (в екології).

Таким чином, якщо x_n представляє популяцію ізольованого виду після n поколінь, припустимо, що ця змінна обмежена в діапазоні $0 < x_n < 1$. Дискретна логістична еволюція у такому випадку буде задана рівнянням (1.6):

$$x_{n+1} = \mu x_n(1 - x_n), \quad (1.6)$$

де μ — константа швидкості зростання, що знаходиться в діапазоні (0;4).

— $0 < \mu < 1$. Темп зростання недостатньо великий, щоб стабілізувати популяцію. Темп впаде, і вид вимре.

— $1 < \mu < 3$. Різка зміна відбувається, коли μ більше 1. Тепер можлива рівновага між двома конкуруючими силами, відтворенням з одного боку, та обмеженням ресурсів — з іншого. Популяція досягає, незалежно від початкових умов, фіксованого значення, яке підтримується в часі.

— $3 < \mu < 3.57$. Каскад раптових змін провокує коливання популяції в циклах періоду 2^n , де n зростає від 1, коли μ близьке до 3, до нескінченності, коли μ наближається до критичного значення 3.57. Це називається каскадом подвоєння періоду.

— $3.57 < \mu < 3.82$. Коли параметр змінюється, система чергує періодичну поведінку з великими періодами у вікнах інтервалів параметрів і хаотичні режими для значення параметрів не розташовані в інтервалах. Популяція може бути непередбачуваною, хоча система детермінована. Хаотичні режими спостерігаються при заданому значенні μ на підінтервалах [0,1].

— $3.82 < \mu < 3.85$. Орбіта періоду 3 з'являється на $\mu = 3.82$ після режиму, коли непередбачувані спалахи, які називаються перервами, стають рідшими до їх зникнення в триперіодичному часовому сигналі.

— $3.85 < \mu < 4$. У цьому інтервалі спостерігається хаотична поведінка з періодичними вікнами.

— $\mu = 4$. Хаотичний режим виходить на всьому інтервалі [0,1]. Цей специфічний режим створює динаміку, яка виглядає як випадкова. Динаміка втратила майже весь свій детермінізм, і популяція розвивається як генератор випадкових чисел.

Фаза активації або розширення контролюється μx_n, що пропорційний поточній популяції x_n і константі швидкості зростання μ. Обмеження ресурсів призводять систему до фази гальмування або скорочення, безпосередньо пов'язаної з перенаселенням. Термін (1 - x_n) може позначати, наскільки система переповнена.

Припустімо тепер, що два види (x_n, y_n) зараз живуть разом. Кожен з них розвивається відповідно до динаміки логістичного типу (1.7):

$$\begin{cases} x_{n+1} = \mu_x(y_n)x_n(1-x_n) \\ y_{n+1} = \mu_y(x_n)y_n(1-y_n) \end{cases} \quad (1.7)$$

де $\mu(z)$ — коефіцієнт зростання кількості популяції.

В залежності від значень μ_1 та μ_2 маємо три різні моделі:

— міжвидовий симбіоз, що може бути змодельовано симетричним зчепленням, що означає взаємну взаємодіючу вигоду, тоді $\mu_x = \mu_y = \mu_1$;

— взаємодія хижак-жертва, що базується на співвідношенні користь/шкода, встановлене між хижаком і жертвою, відповідно, тоді $\mu_x = \mu_1$ та $\mu_y = \mu_2$;

— конкуренція між видами, що викликає протилежне симетричне зчеплення $\mu_x = \mu_y = \mu_2$.

Усі три моделі за основу беруть модель Лотки-Вольтерра [13, с.44].

Мутуалізм — це, по суті, рівняння логістичного зростання та мутуалістична взаємодія [17, с.56]. Термін мутуалістичної взаємодії представляє збільшення зростання популяції першого виду в результаті присутності більшої кількості другого виду, і навпаки (1.8). Оскільки мутуалістичний термін завжди

позитивний, він може призвести до нереалістичного необмеженого зростання. Отже, важливо включити механізм насичення, щоб уникнути проблеми.

$$\begin{cases} \frac{dN_1}{dt} = r_1 N_1 (1 - \alpha_{11} N_1 + \beta_{12} N_2) \\ \frac{dN_2}{dt} = r_2 N_2 (1 + \alpha_{21} N_1 - \beta_{22} N_2) \end{cases}, \quad (1.8)$$

де α_{ij} — негативний вплив внутрішньовидової скупченості;

N_i — щільність популяції;

r_i — власний темп зростання популяції;

β_{ij} — сприятливий вплив щільності мутуалістичного партнера.

Модель Лотки-Вольтерра часто використовується для опису динаміки екологічних систем [18, с.14], у яких взаємодіють два види, один з яких є хижаком, а інший — його жертвою. Модель спрощена з такими припущеннями:

— Існують лише два види: хижак та жертва;

— Жертви народжуються, а потім гинуть через хижацтво або природну смерть;

— Жертви мають безмежну кількість їжі;

— Хижаки народжуються, і на їхню народжуваність позитивно впливає рівень хижацтва, і вони вмирають природним шляхом;

— На забезпечення популяції хижаків впливає тільки чисельність популяції здобичі;

— Хижаки мають безмежний апетит;

— Під час процесу їхньої взаємодії навколишнє середовище не змінюється.

Рівняння Лотки-Вольтерра має вигляд (1.9):

$$\begin{cases} \frac{dx}{dt} = \alpha x - \beta xy \\ \frac{dy}{dt} = \delta xy - \gamma y \end{cases}, \quad (1.9)$$

де β — швидкість, з якою хижаки знищують здобич;

α — швидкість зростання кількості здобичі;

δ — швидкість, з якою хижаки збільшуються, поглинаючи здобич;

γ — смертність хижаків.

У конкурентній моделі буде припущено, що особини різних видів також негативно впливають одна на одну. Класичною моделлю конкурентної взаємодії є безперервна модель міжвидової конкуренції Лотки-Вольтерра [19, с.5-50]. Буде побудовано безпосередньо логістичну модель зростання, додаючи додатковий термін для негативної залежності щільності, цього разу, з конкуруючими видами (1.10):

$$\begin{cases} \frac{dN_1}{dt} = r_1 N_1 (1 - \alpha_{11} N_1 - \alpha_{12} N_2) \\ \frac{dN_2}{dt} = r_2 N_2 (1 - \alpha_{21} N_1 - \alpha_{22} N_2) \end{cases}, \quad (1.10)$$

де r_i — миттєва швидкість збільшення; власна швидкість росту особин, вироблених на особину в одиницю часу;

α_{ii} — коефіцієнт внутрішньовидової конкуренції; негативний вплив особини виду і на власну швидкість росту;

N_i — щільність (кількість) популяції;

α_{ij} — ефект міжвидової конкуренції; негативний вплив особини виду j на швидкість росту виду і.

1.1.2 Урбаністичні моделі

Екологічний урбанізм виник наприкінці XX століття як стратегія зміни парадигми дизайну міст. При цьому міські проекти повинні розроблятися з урахуванням потенціалу та обмежень існуючих природних ресурсів. На відміну від інших попередніх рухів, в екологічному урбанізмі архітектура не є структурним елементом міста — ним є сам ландшафт [20, с.34]. Іншими словами, зелені зони мають існувати не лише для того, щоб прикрашати простір, але й бути справжніми інженерними артефактами, здатними, наприклад, зволожувати, утримувати та очищувати дощову воду. З екологічним урбанізмом міський дизайн стає визначеним природними елементами, властивими його структурі.

У 2022 році більше половини населення світу проживає в містах [21, с.129-152]. Ось чому моделювання зростання міст стало важливою проблемою. Саме тому планувальникам важливо розуміти розподіл міського населення. Аналіз щільності міст корисний для розуміння поточного розподілу населення та потреб у послугах. Громадський транспорт та інші служби (наприклад, школи та лікарні) добре підходять для цього типу аналізу [21, с.33]. Планувальники можуть тестувати різні сценарії, коли вони прагнуть контролювати та проектувати майбутню структуру щільності житлових районів.

Моноцентричному аналізу щільності міст приділено значну увагу з боку двох дисциплін: урбаністичної географії та регіональної науки. До цього підходили як теоретично, так і емпірично. Класичне дослідження Коліна Кларка призвело до великої кількості літератури, що стосується емпіричних реалізацій для широкого кола столичних районів і міст, у різних країнах і в різний час [22, с.44].

Щоб спростити дослідження щільності міського населення та зробити порівняння його результатів легшим для розуміння, Колін Кларк запропонував дві загальні гіпотези, достовірність яких зараз загальновизнана:

1. У кожному великому місті, за винятком центральної ділової зони, де проживає невелика кількість мешканців, у внутрішніх районах є густонаселені райони, щільність яких поступово падає, коли просуваємося до зовнішніх передмість.

2. У більшості (але не в усіх) містах з часом щільність має тенденцію до зменшення у найбільш густонаселених внутрішніх передмістях і зростання у зовнішніх передмістях, і все місто має тенденцію «розкидатися».

Кларк стверджує, що щільність міського населення можна правильно описати за допомогою негативної експоненціальної функції (1.11):

$$D(x) = D_0 e^{bx} \qquad (1.11)$$

де x — відстань до центру, виміряну в одиницях довжини;

$D(x)$ — означає густоту постійного населення на одиницю поверхні;

$D_0 > 0$ та $b < 0$.

Стюарт, географ, вже запропонував лінійну залежність між щільністю та відстанню (1.12):

$$D(x) = D_0 + bx \qquad (1.12)$$

Таннер і Смід запропонували дві нові функціональні форми у своїх дослідженнях міського руху. Їхній внесок ґрунтується на двох особливих випадках квадратичної гамма-функції. Модель Таннера передбачає (1.13):

$$D(x) = D_0 e^{cx^2} \qquad (1.13)$$

де $D_0 > 0$ та $c < 0$.

У випадку Сміда відношення виглядає наступним чином (1.14):

$$D(x) = D_0 x^e \tag{1.14}$$

де $D_0 > 0$ та $e < 0$.

Екологічні моделі міської форми описують і пояснюють просторові закономірності, що виникли в результаті розподілу людей, будівель і видів діяльності на території міста [23, с.55-67]. Цей упорядкований набір просторових домовленостей відомий як модель землекористування міста або просторова форма. Протягом багатьох років дослідники-екологи визначили три основні моделі геометрії форми міста: концентричну зону, сектор і кілька ядер.

Незважаючи на те, що три моделі концептуально відрізняються, у фактичному розвитку більшості міст різні елементи з трьох моделей стають унікальним чином об'єднані в просторову структуру, яка надає кожному місту власну індивідуальну просторову геометрію. Кожна з трьох моделей була розроблена для пояснення міської морфології в індустріальних містах двадцятого століття. Модель концентричної зони була представлена Ернестом Берджессом у 1925 році. Секторна і багатоядерна моделі були представлені пізніше як альтернатива моделі концентричної зони.

Модель концентричної зони передбачає, що місто зазнає швидкого зростання населення та економічного зростання [24, с.65-68]. Оскільки різні групи населення, промислові підприємства та організації приходять до міста, на ринку землі розвивається величезна конкуренція за високоцінні місця. Групи з найбільшою кількістю доступних ресурсів (наприклад, бізнес і промисловість, вищий клас) можуть отримати бажане місце розташування [22, 24], тоді як ті з

меншими ресурсами (наприклад, збіднілі групи іммігрантів) повинні задовольнятися в іншому місці [21, 24].

Переміщення людей і діяльності з однієї зони в іншу відповідно до цієї моделі нагадує схему, яка спостерігається, коли камінчик кидають в озеро (Рис. 1.1):

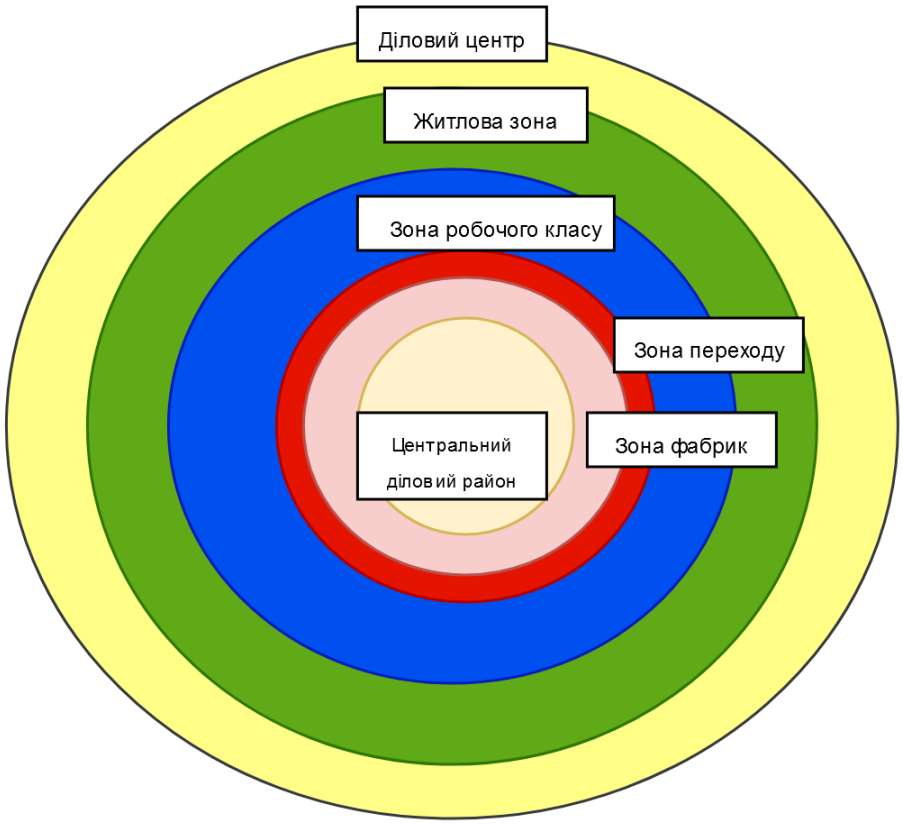

Рис. 1.1. — Модель концентричних кіл міста

Швидкість переміщення міських районів з одного типу населення або виду діяльності на інший регулюється кількома факторами. По-перше, це швидкість зростання людей і видів діяльності, які потребують житла або будівель. По-друге, темпи будівництва житла, промислових будівель і комерційних приміщень. По-третє, це інвестиційні рішення, прийняті забудовниками, фінансовими установами та політичними режимами. Якщо будівництво відстає від зростання населення та економіки, відбувається стагнація та накопичення людей і

діяльності. Попит на забудовані землі зростає, а ціни на забудовані ділянки зростають. Якщо будівництво перевищує чисельність населення та економічне зростання, рівень вакантності зростає, а ціни на землю знижуються, але створюються нові можливості, які можуть сприяти майбутньому зростанню. Або, в крайньому випадку, з високим рівнем вакансій і низьким зростанням може відбутися крах місцевої економіки розвитку, що посилає місто в економічну депресію.

На основі вивчення 142 американських міст Гомер Хойт стверджував, що, на відміну від моделі концентричних зон, міська геометрія міста краще описується секторною схемою розвитку території (Рис. 1.2):

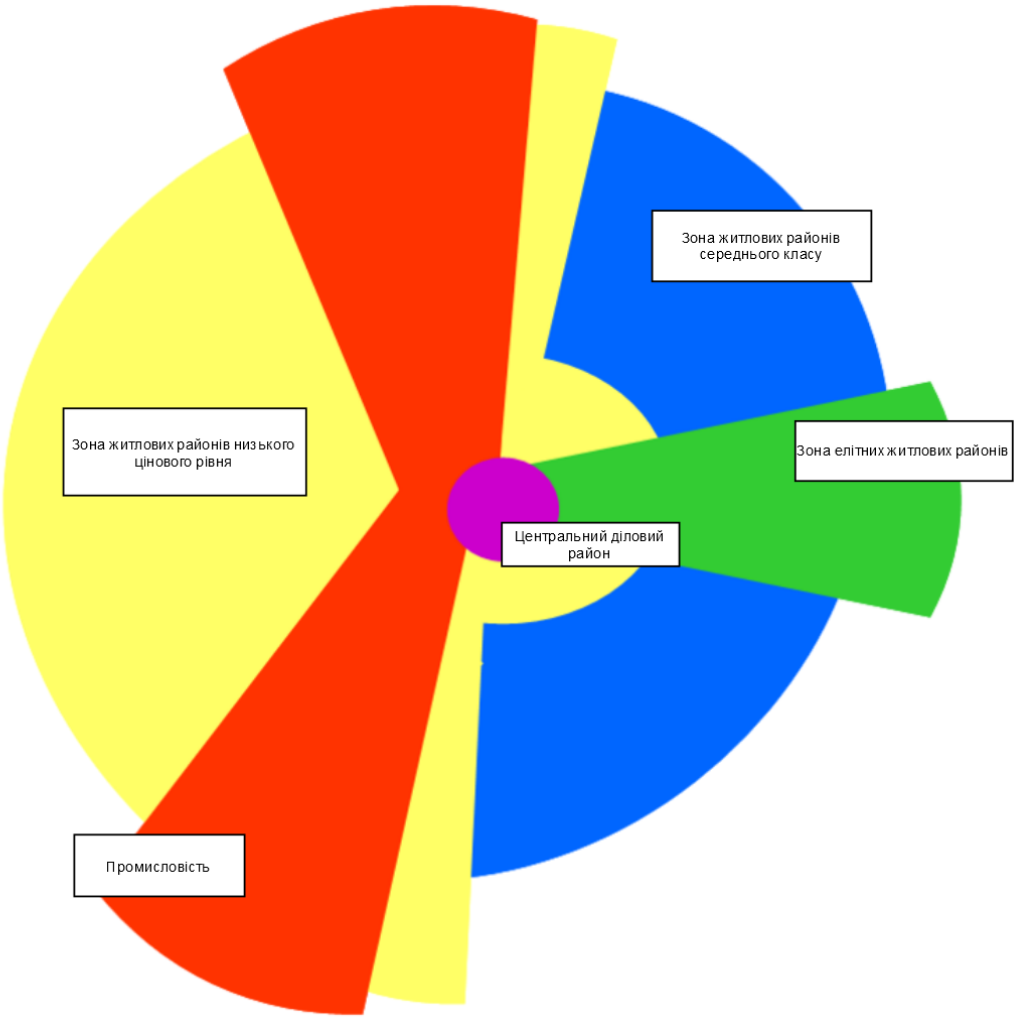

Рис. 1.2. — Секторна модель міста

Розподіл орендної плати та групи соціально-економічного статусу міста організовані в однорідні клини або сектори у формі пирога, які простягаються від центрального ділового району міста до периферії. Характерне землекористування, склад видів діяльності та склад населення для будь-якого сектора відрізняються від секторів, які до нього прилягають. Мається на увазі, що якщо їхати з центрального ділового району на периферію, залишаючись у тому самому секторі, він залишиться загалом із тим самим типом землекористування, складом постійного населення та структурою діяльності. Модель концентричної зони забезпечує водієві набагато інший погляд на місто. Подорожуючи від центрального ділового району до периферії, незалежно від напрямку, проходиться ту саму градацію постійно зростаючого статусного складу районів.

Ця модель включає ідею Річарда Херда про те, що зростання та розвиток спочатку відбуваються вздовж основних транспортних шляхів від центру міста до глибинки; до них відносяться залізничні лінії, шосейні дороги та судноплавні водойми. У якийсь момент освоєння відкритої землі між осями стає дешевшим у часі на дорогу та грошах, ніж продовжувати просування вздовж осей. Коли територія між осями заповнюється, починається інший цикл, коли розвиток знову зміщується до осей і виштовхується вздовж них у незабудовану глибинку.

У місті з секторальною просторовою геометрією сектори промисловості, складського господарства та землі низької якості, як правило, оточені секторами жителів із низьким рівнем доходу та робітничого класу. Сектори житла середнього класу, як правило, відокремлюють тих, хто має вищий статус, від секторів з низьким доходом, промисловості та шкідливих видів діяльності. Населення з високим статусом керує найбажанішими місцями в місті. Сектори з високою орендною платою, як правило, займають височини, вільні від ризику повеней, а квартири класу люкс зазвичай розташовуються поблизу бізнес-центрів у старих житлових районах. Райони з низькою орендною платою та райони, зайняті бідними та маргіналізованими расовими та етнічними групами, як

правило, розташовані на протилежному боці міста від сектора з високим доходом.

Розташування та переміщення секторів, зайнятих заможними та вищими соціально-економічними групами, мають великий вплив на розташування інших секторів. Модель Хойта стверджує, що зростання житла з високою орендною платою, як правило, йде від даної точки походження вздовж встановлених маршрутів або до іншого існуючого ядра чи торгової зони. Сектори з високою орендною платою, як правило, поширюються вздовж озер, заток, річок і океанських портів, де набережні є непромисловими. Житлові райони з високою орендною платою, як правило, розвиваються у напрямку відкритої місцевості, подалі від «тупикових» секторів, які перешкоджають розширенню через природні чи штучні бар'єри, і в бік будинків лідерів громад. Зростання високоорендних мікрорайонів триває в тому ж напрямку протягом тривалого часу. Розробники нерухомості можуть змінювати напрямок високоякісного житлового будівництва, але вони не можуть заперечувати або скасовувати вплив загальних принципів, втілених у моделі.

На відміну від інших моделей, багатоядерна модель Чонсі Гарріса та Едварда Уллмана не розглядає місто як організоване навколо Центрального Ділового Району (ЦДР). Швидше, він постулює, що існує низка різних ядер зростання, кожне з яких впливає на розподіл людей, види діяльності та використання землі. Кожне ядро спеціалізується на помітно різних видах діяльності, починаючи від роздрібної торгівлі та закінчуючи виробництвом, освітою та охороною здоров'я, а також житловим. Ядра бувають різними за розміром. Деякі з них великі, наприклад промислові об'єкти; інші невеликі, наприклад торгові центри. Таким чином, просторова геометрія міста багато в чому нагадує клаптикову ковдру з різних ядер, які не організовані навколо єдиного центру. ЦДР є лише одним із кількох функціонально важливих ядер.

Кількість і склад ядер у місті дуже різняться. У великих містах більше ядер, ніж у менших, і вони, як правило, більш спеціалізуються на більшій громаді. Наприклад, невелике місто може мати ядро роздрібної торгівлі, але у великому місті окремі види роздрібної торгівлі можуть виникнути у власне ядро, як це видно в «діамантовому» районі Нью-Йорка. Деякі ядра існували з самого початку міста; інші розвивалися разом із зростанням міста, наприклад, етнічні анклави, створені прибулими групами іммігрантів, і через реконструкцію міст, коли один вид землекористування витісняє інший, як у випадку з проектом арени, що будується на місці колишньої в'язниці. На рис. 1.3 можна побачити модель кількох ядер міста:

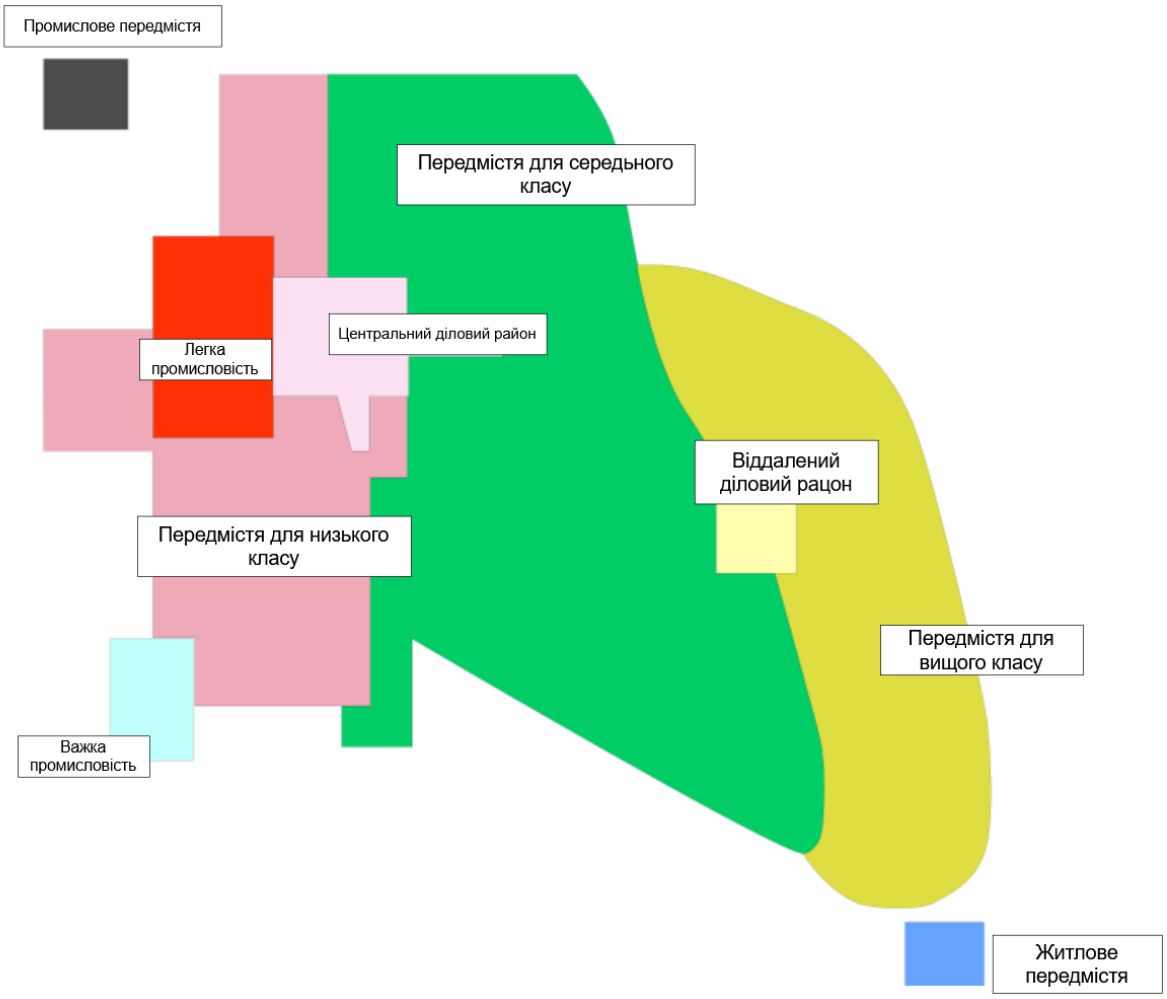

Рис. 1.3. — Модель кількох ядер міста

Модель кількох ядер використовує чотири основні принципи, щоб пояснити як появу окремих ядер, так і їх зміну в часі.

1. Певні види діяльності вимагають спеціалізованих об’єктів, розташованих лише в одній або кількох частинах мегаполісу, як це видно у випадку виробничих підприємств, які потребують великих масивів незабудованої землі, розташованих поблизу залізничних ліній.

2. Певні подібні види діяльності отримують прибуток від сусідніх конгрегацій, як видно з об’єднання роздрібних закладів у торговельні та торгові центри.

3. Деякі різні види діяльності є антагоністичними або завдають шкоди одна одній, як це видно у випадку виробничих підприємств і житлових забудов вищого класу.

4. Певні види діяльності не можуть дозволити собі витрати на найбажаніші місця, як це видно у випадку житлових районів із низьким рівнем доходу та високих ділянок із дуже популярними краєвидами.

З плином часу ці три інтелектуально пов’язані між собою моделі широко розглядаються як «класичні моделі міського землекористування». Вони є «класичними» в тому сенсі, що три моделі витримали випробування часом і виявилися каталізаторами. дослідження міст як у розвинених, так і в країнах, що розвиваються.

1.2 Дискретні методи моделювання в еволюції

Починаючи з сучасного еволюційного синтезу, математика відігравала незамінну роль у теорії еволюції [25, с.149-156]. Як правило, внесок математики полягає в розробці та аналізі математичних моделей. Представляючи еволюційні

сценарії в точний спосіб, математичне моделювання може прояснити концептуальні питання, з’ясувати механізми, що лежать в основі, і генерувати нові гіпотези [26, с.76].

Проте висновки з математичних моделей завжди повинні перевірятися на предмет їх стійкості. Альтернативний підхід полягає у використанні загальності, що стала можливою завдяки математичній абстракції. Якщо можна визначити мінімальний набір припущень, які застосовуються до широкого класу моделей, будь-яка теорема, доведена на основі цих припущень, застосовуватиметься до всього класу. Такі теореми усувають повторну роботу з виведення особливих випадків однієї моделі за раз. Що ще важливіше, чим більшу загальність доведена теорема, тим більша ймовірність, що вона представлятиме надійний науковий принцип [23, 24].

Теорія еволюції є об’єднуючою концепцією біології [26, с.55]. В основі сучасної теорії еволюції лежить ідея природного відбору, диференційованого відтворення між генотипами. Багато факторів сприяють залишенню різної кількості потомства; виживання, довголіття та плідність є одними з найважливіших. На біологічну теорію еволюції — «сучасний синтез» — глибоко вплинули математичні формулювання популяційної генетики Фішера, Райта та Голдена. Для перевірки різних аспектів теорії було розроблено багато лабораторних експериментів. Демонстрація того, що деякі генетичні зміни в природі можуть бути відтворені в експериментальних популяціях, відкрила шлях для багатогранної атаки на генетичні процеси в еволюції.

1.2.1 Популяційна генетика

Популяційна генетика — галузь біології, яка вивчає генетичний склад біологічних популяцій, а також зміни в генетичному складі, що є результатом дії різних факторів, у тому числі природного відбору [17, с.12-14]. Популяційна генетика є однією з найбільш динамічних областей досліджень у біологічних

науках. Крім того, ця дисципліна пропонує виклик, який не зустрічається в більшості інших біологічних наук, тому що його основний виклик є теоретичним, а не експериментальним. Проте його теоретичний розвиток випливає з аналізу емпіричних даних, хоча в деяких випадках емпірична підтримка виникає після теоретичного розвитку.

Отже, це можна розглядати як зворотний зв'язок між емпіричними даними та теоретичними розробками, щоб зрозуміти та якнайкраще пояснити, як еволюціонують природні популяції. Математичні моделі широко використовуються в популяційній генетиці; ці моделі являють собою спрощення складної ситуації і неминуче не в змозі показати всі зв'язки реальної ситуації. Отже, вибір кількох ідентифікованих факторів для опису реальної та складної ситуації є проблемою.

Популяційний генетик зацікавлений у складних ситуаціях, які включають кілька факторів, таких як народження, смерть, чисельність популяції, моделі спаровування, географічний розподіл організмів тощо. Таким чином, популяційний генетик намагається описати ефект великої кількості окремих подій шляхом повної роботи фізика чи хіміка із статистичним середнім значенням молекулярної поведінки кожної окремої молекули.

Популяційна генетика тісно пов'язана з вивченням еволюції та природного відбору [27, с.23] і часто вважається теоретичним наріжним каменем сучасного дарвінізму. Це пояснюється тим, що природний відбір є одним із найважливіших факторів, які можуть впливати на генетичний склад популяції [27, с.24]. Природний відбір відбувається, коли деякі варіанти в популяції перевершують відтворення інших варіантів через те, що вони краще пристосовані до навколишнього середовища або «більш пристосовані».

Популяційні генетики зазвичай визначають «еволюцію» як будь-яку зміну в генетичному складі популяції з часом. Чотири фактори, які можуть спричинити таку зміну: природний відбір (natural selection), мутація (mutation), випадковий

дрейф генів (random genetic drift) і міграція в популяцію або з неї (migration into or out of the population). (П’ятий фактор — зміни моделі спаровування (changes to the mating pattern) — може змінити генотип, але не частоти алелів; багато теоретиків не вважають це еволюційною зміною.)

Початок цієї дисципліни можна датувати 1908 роком, коли Годфрі Х. Харді та Вільгельм Вайнберг незалежно один від одного сформулювали свій принцип.

Розробивши математично наслідки відбору, що діє на популяцію, яка підкоряється менделівським правилам успадкування, Фішер, Холдейн і Райт показали, що дарвінізм і менделізм не просто сумісні, а й чудові товариші по ліжку; це зіграло ключову роль у формуванні «неодарвінівського синтезу» і пояснює, чому популяційна генетика стала займати таку ключову роль в еволюційній теорії.

При випадковому спаровуванні закон Харді-Вайнберга стверджує, що частоти алелів від батьків визначають частоти генотипу в нащадках [27, с.55].

Закон Харді-Вайнберга полягає у наступному: у популяції нескінченно великого розміру, в якій не діє природний відбір, не йде мутаційний процес, відсутній обмін особинами з іншими популяціями, не відбувається дрейф генів, всі схрещування випадкові — частоти генотипів за якимось геном (якщо в популяції є два алелі цього гена) будуть підтримуватись постійними з покоління в покоління та відповідати рівнянню (1.15):

$$p^2 + 2pq + q^2 = 1 \qquad (1.15)$$

де p^2 — доля гомозигот по одному із алелей;

p — частота цього алелю;

q^2 — доля гомозигот по альтернативному алелю;

q — частота відповідного алеля;

$2pq$ — доля гетерозигот.

Таким чином, цей принцип підкреслює генетичну безперервність у часі між поколіннями, а також розширює менделівське знайоме співвідношення до менделівського співвідношення рівня популяції. Таким чином, математична модель, що лежить в основі принципу Харді-Вайнберга, встановлює математичний зв'язок між частотами алелів і частотами генотипів: $AA{:}p^2$; $Aa{:}2pq$ і $aa{:}q^2$, у яких p^2; $2pq$ і q^2 — частоти генотипів АА, Аа і аа в зиготах будь-якого покоління, а p і q — частоти алелів А і а в гаметах попереднього покоління.

1.2.2 Природний відбір

Природний відбір відбувається, коли деякі генотипові варіанти в популяції користуються перевагою щодо виживання або розмноження над іншими. Найпростіша популяційно-генетична модель природного відбору передбачає наявність одного аутосомного локусу з двома алелями A_1 і A_2, як зазначено вище. Три диплоїдні генотипи A_1A_1, A_1A_2 і A_2A_2 мають різну придатність, позначену w_{11}, w_{12} і w_{22} відповідно. Передбачається, що ці придатності є постійними в поколіннях. У цьому контексті придатність генотипу можна визначити як середню кількість успішних гамет, яку організм цього генотипу дає наступному поколінню, що залежить від того, наскільки добре організм виживає, скільки спаровувань він досягає та наскільки він плідний [28, с.22]. Якщо w_{11}, w_{12} і w_{22} не рівні, тоді відбуватиметься природний відбір, що, можливо, призведе до зміни генетичного складу популяції.

Припустимо, що спочатку, тобто до початку селекції, генотипи зиготи відповідають пропорціям Харді-Вайнберга, а частоти алелів A_1 і A_2 дорівнюють p і q відповідно, де $p + q = 1$. Потім зиготи ростуть до дорослого стану і розмножуються, дає початок новому поколінню нащадків зигот. Наше завдання полягає в тому, щоб обчислити частоти A_1 і A_2 у другому поколінні; позначимо їх через p' і q' відповідно, де $p' + q' = 1$. (Зверніть увагу, що в обох поколіннях

розглянуто частоти генів на зиготичній стадії; вони можуть відрізнятися від частот генів дорослих, якщо є диференціальне виживання .)

У першому поколінні генотипові частоти на зиготичній стадії становлять p^2, $2pq$ і q^2 для A_1A_1, A_1A_2, A_2A_2 відповідно за законом Харді-Вайнберга. Три генотипи продукують успішні гамети пропорційно їх придатності, тобто у співвідношенні $w_{11}:w_{12}:w_{22}$. Середня пристосованість популяції виглядає як (1.16):

$$w = p^2w_{11} + 2pqw_{12} + q^2w_{22} \qquad (1.16)$$

Тому загальна кількість утворених успішних гамет дорівнює Nw, де N — розмір популяції. Якщо припустити, що немає мутації та виконується закон сегрегації Менделя, то організм A_1A_1 вироблятиме лише гамети A_1, організм A_2A_2 — лише гамети A_2, а організм A_1A_2 — гамети A_1 та A_2 в рівних пропорціях.

1.2.3 Баланс відбору та мутації

Мутація є основним джерелом генетичної варіації [29, с.33], яка перешкоджає популяціям стати генетично однорідними в ситуаціях, де вони б інакше. Після врахування мутації необхідно змінити висновки, зроблені в попередньому розділі. Навіть якщо один алель вибірково перевершує всі інші в даному локусі, він не закріпиться в популяції; рекурентна мутація забезпечить присутність інших алелів з низькою частотою, таким чином зберігаючи ступінь поліморфізму. Популяційні генетики вже давно зацікавлені в дослідженні того, що відбувається, коли відбір і мутація діють одночасно.

Продовжуючи нашу модель з одним локусом і двома алелями, припустімо, що алель A_1 вибірково перевершує A_2, але повторювана мутація з A_1 на A_2 запобігає поширенню A_1 до фіксації. Швидкість мутації від A_1 до A_2 на покоління, тобто частка алелів A_1, які мутують у кожному поколінні, позначається u.

Зворотну мутацію від A_2 до A_1 можна ігнорувати [29, с.55], оскільки припускається, що алель A_2 має дуже низьку частоту в популяції завдяки природному відбору. Оскільки певна частка (u) алелей A_1 мутує до A_2, це рівняння повторення має бути змінено таким чином (1.17), щоб врахувати мутацію:

$$p' = \frac{(p^2 w_{11} + pq w_{12})(1-u)}{\bar{w}} \qquad (1.17)$$

1.2.4 Випадковий дрейф

Випадковий генетичний дрейф стосується випадкових коливань частоти генів, які виникають у обмежених популяціях; це можна розглядати як тип «помилки вибірки». У багатьох еволюційних моделях популяція вважається нескінченною [30, с.12] або дуже великою саме для того, щоб абстрагуватися від випадкових коливань. Але хоча це припущення математично зручне, часто нереалістичне [25, с.67]. У реальному житті фактори випадковості незмінно відіграватимуть роль, особливо в невеликих популяціях. Термін «випадковий дрейф» іноді використовується в широкому значенні [30, с.56] для позначення будь-яких стохастичних факторів, які впливають на частоти генів у популяції, включаючи, наприклад, випадкові коливання у виживанні та успіху спаровування; а інколи у вужчому значенні — для випадкового відбору гамет для формування потомства (що виникає через те, що організми виробляють набагато більше гамет, ніж коли-небудь утворять запліднену зиготу). Тут використовується ширший сенс [31].

Щоб зрозуміти природу випадкового дрейфу, розглянемо простий приклад. Популяція містить лише десять організмів, п’ять типу A та п’ять типу B; організми розмножуються безстатевим шляхом і дають однотипне потомство. Припустимо, що жоден тип вибірково не перевершує інший — обидва однаково

добре пристосовані до середовища. Однак це не означає, що два типи дадуть однакову кількість нащадків, оскільки випадкові фактори можуть відігравати певну роль. Наприклад, цілком можливо, що всі типи B можуть випадково загинути до того, як розмножаться; в цьому випадку частота B у другому поколінні впаде до нуля. Якщо так, то занепад типу B (і, отже, поширення типу A) є результатом випадкового дрейфу. Еволюціоністи часто зацікавлені в тому, щоб дізнатися, чи є дана зміна частоти гена результатом дрейфу, відбору чи деякої комбінації двох.

Мітка «випадковий дрейф» дещо вводить в оману. Говорячи, що поширення типу A відбувається через випадковий дрейф або випадковість, не мається на увазі, що неможливо знайти причину його поширення [31]. Теоретично ми, ймовірно, могли б відкрити повну причинно-наслідкову історію про те, чому кожен організм у популяції залишив рівно стільки нащадків, скільки він залишив. Приписуючи еволюційні зміни випадковому дрейфу, нема заперечень, що існує така причинно-наслідкова історія, яку слід розповісти. Швидше, мається на увазі, що поширення типу A не відбулося через його адаптивну перевагу над типом B. Іншими словами, типи A і B мали однакову очікувану кількість нащадків, тому були однаково придатними; але типи A мали більшу фактичну кількість нащадків [32, с.473-475]. У обмеженій популяції фактичний репродуктивний результат майже завжди відхилятиметься від очікуваного, що призведе до еволюційних змін.

Аналогія з підкиданням монети може прояснити випадковий дрейф. Припустимо, чесну монету підкинули десять разів. Імовірність того, що під час будь-якого підкидання випаде решка, дорівнює ½, отже, очікувана частота решки в послідовності з десяти становить 50%. Але ймовірність фактично отримати половину орлів і половину решок становить лише 242/1024, або приблизно 23,6%. Таким чином, навіть якщо монета справедлива, навряд чи отримаємо однакові пропорції орлів і решок у послідовності з десяти кидків; деяке

відхилення від очікувань більш ймовірне, ніж ні. Таким же чином, навіть якщо типи А і В однаково підходять у наведеному вище прикладі, цілком ймовірно, що відбудуться певні еволюційні зміни. Ця аналогія також ілюструє роль чисельності населення. Якби кинули монету сто разів, а не десять, частка голів, ймовірно, була б дуже близькою до ½. Так само, чим більша популяція, тим менш важливий вплив випадкового дрейфу на частоти генів; у нескінченній межі дрейф не впливає.

1.2.4 Міграція

Міграція в популяцію або з неї є четвертим і останнім фактором, який може вплинути на її генетичний склад. Очевидно, якщо іммігранти генетично відрізняються від популяції, до якої вони мігрують, це призведе до зміни генетичного складу популяції. Еволюційне значення міграції пояснюється тим фактом, що багато видів складаються з ряду окремих субпопуляцій, значною мірою ізольованих одна від одної, але пов'язаних випадковою міграцією. (Для екстремального прикладу поділу популяції подумайте про колонії мурах.) Міграція між субпопуляціями породжує потік генів, який діє як свого роду «клей», обмежуючи ступінь, до якої субпопуляції можуть генетично відрізнятися одна від одної.

Найпростіша модель для аналізу міграції припускає, що дана популяція приймає певну кількість мігрантів кожне покоління, але не посилає емігрантів. Припустимо, що частота алеля A_1 у постійній популяції дорівнює p, а частота алеля A_1 серед мігрантів, які прибувають у популяцію, дорівнює p_m. Частка мігрантів, які приходять у популяцію кожного покоління, дорівнює m (тобто як частка постійного населення). Отже, після міграції частота алеля A_1 у популяції становить (1.18):

$$p' = (1 - m)p + mp_m \qquad (1.18)$$

Отже, зміна частоти генів у поколіннях виглядає як (1.19):

$$\Delta \mathrm{p} = p' - p = -m(p - p_m) \qquad (1.19)$$

Отже, міграція збільшить частоту алеля A_1, якщо $p_m > p$, зменшить його частоту, якщо $p > p_m$, і залишить його частоту незмінною, якщо $p = p_m$. Тоді легко вивести рівняння, яке дає частоту гена в поколінні t як функцію його початкової частоти та швидкості міграції. Рівняння має вигляд (1.20):

$$p_t = p_m + (p_0 - p_m)(1 - m)^t \qquad (1.20)$$

де p_0 — початкова частота алеля A_1 у популяції, тобто до того, як відбулася будь-яка міграція.

Оскільки вираз $(1 - m)^t$ прямує до нуля зі збільшенням t, легко побачити, що рівновага досягається, коли $p_t = p_m$, тобто коли частота генів мігрантів дорівнює частоті генів резидентної популяції.

Ця проста модель припускає, що міграція є єдиним фактором, що впливає на частоту генів у локусі, але це навряд чи так. Отже, необхідно розглянути, як міграція буде взаємодіяти з відбором, дрейфом і мутацією. Баланс між міграцією та відбором може призвести до збереження шкідливого алеля в популяції, у спосіб, дуже аналогічний балансу мутації та відбору, який обговорювався вище. Взаємодія між міграцією та дрейфом особливо цікава. Можна побачити, що дрейф може призвести до генетичного розходження окремих субпопуляцій виду. Міграція протистоїть цій тенденції — це гомогенізуюча сила, яка прагне зробити субпопуляції більш схожими [33, с.163-190] Математичні моделі припускають, що навіть досить невелика швидкість міграції буде достатньою, щоб запобігти

генетичному розходженню субпопуляцій виду. Деякі теоретики використовували це, щоб аргументувати еволюційну важливість групового відбору на тій підставі, що генетичні відмінності між групами, які є важливими для функціонування групового відбору, навряд чи збережуться в умовах міграції.

1.2.5 Невипадкове спаровування

Закон Харді-Вайнберга — відправна точка для більшості популяційно-генетичного аналізу — був отриманий на основі припущення про випадкове спаровування [35, с.252-256]. Але відхилення від випадкового спаровування насправді досить поширені. Організми можуть мати тенденцію вибирати партнерів, які схожі на них фенотипово або генотипово — система спарювання, відома як «позитивний асортимент». Крім того, організми можуть вибирати партнерів, не схожих на них, — «негативний асортимент». Іншим типом відхилення від випадкового спаровування є інбридинг, або переважно спаровування з родичами.

Аналізувати наслідки невипадкового спаровування досить складно, але деякі висновки можна побачити досить легко. По-перше, і найважливіше, невипадкове спарювання саме по собі не впливає на частоти генів (тому це не еволюційна «сила» на рівні з відбором, мутаціями, міграцією та дрейфом) [35, с.234-237]; швидше це впливає на частоти генотипів. Щоб зрозуміти цей момент, зауважте, що частота генів популяції на зиготичній стадії дорівнює частоті генів у пулі успішних гамет, з яких утворюються зиготи. Схема спарювання просто визначає спосіб, у який гаплоїдні гамети «упаковуються» в диплоїдні зиготи. Таким чином, якщо популяція випадкового спарювання раптом почне спарюватися невипадково, це не вплине на частоти генів [35, с.56-57].

По-друге, позитивне ассортативне спарювання буде мати тенденцію до зменшення частки гетерозигот у популяції, таким чином збільшуючи генотипову дисперсію. Щоб побачити це, знову розглянемо один локус із двома алелями, A_1

і A_2, із частотами p і q у даній популяції. Спочатку популяція знаходиться в рівновазі Харді-Вайнберга, тому частка гетерозигот A_1A_2 становить $2pq$. Якщо тоді популяція починає спаровуватися повністю асортативно, тобто спаровування відбувається лише між організмами однакового генотипу, очевидно, що частка гетерозигот повинна зменшитися. Для $A_1A_1 \times A_1A_1$ і $A_2A_2 \times A_2A_2$ спарювання не призведе до гетерозигот; і лише половина нащадків від спаровування $A_1A_2 \times A_1A_2$ буде гетерозиготною. Таким чином, частка гетерозигот у другому поколінні повинна бути менше $2pq$. І навпаки, негативний асортимент буде мати тенденцію до збільшення частки гетерозигот порівняно з рівновагою Харді-Вайнберга.

Стосовно інбридингу можна сказати наступне. Загалом, інбридинг має тенденцію до збільшення гомозиготності популяції, як позитивний асортимент. Причина цього очевидна: родичі, як правило, більш схожі за генотипом, ніж випадково обрані члени популяції. У більшості видів, включно з людиною, інбридинг негативно впливає на організм — явище, відоме як «інбридингова депресія». Пояснення цього полягає в тому, що шкідливі алелі часто мають тенденцію бути рецесивними, тому не мають фенотипічного ефекту, якщо їх виявляють у гетерозигот. Інбридинг зменшує частку гетерозигот, роблячи рецесивні алелі більш імовірними для гомозигот, де їхні негативні фенотипові ефекти стають очевидними. Зворотне явище — «гібридна сила», що виникає в результаті аутбридингу — широко використовується селекціонерами тварин і рослин.

1.3 Висновки до розділу 1

У розділі 1 були розглянуті поняття, що пов’язані із математичним моделюванням у екології та в еволюції. Були розглянуті поняття математичної

екології та основні математичні моделі: експоненційна модель (модель Мальтуса), модель зростання чисельності популяції окремого виду (модель Вергюльста з обмеженнями, логістичне рівняння), модель дискретної логістичної еволюції (модель Роберта Мея).

Були розглянуті також урбаністичні моделі: Модель кількох ядер, Секторна модель, Модель концентричних кіл. До кожної моделі було наведено малюнок та опис.

Були розглянуті також дискретні методи моделювання в еволюції: природний відбір, мутація, випадковий дрейф, мутація та невипадкове спаровування. До кожного методу були наведені математичні формули.

РОЗДІЛ 2 МАТЕМАТИЧНІ МОДЕЛІ ДЛЯ ПРОГНОЗУВАННЯ ПОШИРЕННЯ ІНФЕКЦІЙНИХ ХВОРОБ

Поширення епідемії COVID-19 у різних країнах та регіонах світу почалося з моменту першого спалаху хвороби в Китаї та інтенсивно проводиться нині. Для цієї мети використовуються різні моделі: моделі логістичного зростання Вергюльста та його модифікації [36-37], компланарні моделі типу SIR [10,16,17,18], SIRD, SIS, SEIR, SIRS [36, с.45] та ін. [37, с.24-76], які враховують різні уточнюючі фактори [38, 39, 40]. Всі ці моделі застосовні в однорідних ізольованих системах, коли має місце гарне перемішування в популяції та виконується закон діючих мас. Більш-менш це виконується у великих містах, де не вводяться обмежувальні заходи. Для поширення епідемії COVID-19 компланарні моделі з постійними параметрами не застосовні [41, с.54]. Вони описують поширення інфекції як однієї епідеміологічної хвилі з одним піком.

Реальні системи є відкритими та неоднорідними та демонструють дуже складну динаміку поширення інфекції.

Якщо згладити графіки щоденного приросту хворих, то у багатьох випадках виходять криві, що мають кілька локальних максимумів (піків), або містять «плечі», або «плато», що не вкладається в класичну модель з одним піком. Таку поведінку можна пояснити виникненням кількох локальних хвиль поширення інфекції, зрушених у часі. Локальні хвилі можуть виникати у різний час у різних регіонах країни [43, с.56-59], або в одному регіоні, але у різний час, коли інфекція приноситься ззовні, викликаючи новий спалах, або, коли в цьому.

У регіоні деякі жителі починають ігнорувати карантинні заходи. У загальному випадку поширення інфекції є суперпозицією багатьох локальних хвиль епідемії.

Пандемія небезпечна тим, що одночасне захворювання на інфекцію безлічі людей може призвести до перевантаженості системи охорони здоров'я [40, с.56] з підвищеною кількістю госпіталізацій та летальних наслідків. Системи охорони здоров'я можуть виявитися не готові до надзвичайно великої кількості тяжкохворих пацієнтів.

Найбільш важливим заходом у відповідь до інфекції є не лікувальні заходи, а зниження швидкості її поширення, щоб розтягнути її в часі і знизити, таким чином, навантаження на системи охорони здоров'я. Для розрахунків варіантів можливих заходів для зниження навантаження на медичні установи, а також термінів зняття карантинних заходів використовується математичне моделювання.

На даний момент існують різні класи математичних моделей, які застосовуються для прогнозування перебігу епідемій [42]. Клас моделей SRID (Susceptible — сприйнятливий, Recovered — одужалий, Infectious — інфікований, Deceased — померлий) передбачає формування стійкого імунітету до інфекції (повторне зараження неможливе). Клас моделей SIR (Susceptible – сприйнятливий, Infectious — інфікований, Recovered — одужав) — модель з формуванням стійкого імунітету є базовою моделлю для опису поширення епідемій Клас моделей SIS (Susceptible — сприйнятливий, Infected — одужалий, Susceptible — сприйнятливий) — модель без стійкого імунітету з можливим хронічним перебігом хвороби. SIS-моделі добре зарекомендували себе при описі небезпечних вірусних захворювань із хронічним перебігом, таких як вірус імунодефіциту людини (HIV), хронічні гепатити B (HBV) та C (HCV). Моделі SIR застосовуються для опису епідемічних процесів, що викликаються вірусами з відносно легким перебігом: групи вірусів, що викликають респіраторні інфекції (ГРВІ) та деякі штами вірусу грипу (influenza virus) [40, с.45-48].

Моделювання показує, з якою швидкістю поширюватиметься епідемія, скільки заразиться, жертв, а також хворих у критичному стані. Останній

показник порівнюється з потужностями медичної системи для визначення - чи здатна вона впоратися з напливом пацієнтів, які потребують спеціалізованої допомоги: у випадку з коронавірусною інфекцією це реанімація, штучна вентиляція легень тощо.

У наведених тут епідеміологічних моделях зроблені такі припущення [44, с.465]:

1 . Розглянута популяція має постійний розмір N, який є достатньо великим, щоб розміри кожного класу можна було вважати безперервними змінними. Якщо модель повинна включати життєву динаміку, то передбачається, що народжуваність і природна смертність відбуваються з однаковою частотою і що всі новонароджені сприйнятливі до вірусу. Індивідууми вилучаються через смерть з кожного класу зі швидкістю, пропорційною розміру класу з константою пропорційності μ, яка називається щоденною швидкістю видалення смерті. Це відповідає негативній експоненціальній віковій структурі із середньою тривалістю життя $1/\mu$.

2. Популяція однорідно перемішується. Добовий рівень контактів λ — це середня кількість адекватних контактів на одного інфікованого за день. Адекватний контакт інфікованого — це взаємодія, яка призводить до інфікування іншої особини, якщо вона сприйнятлива. Таким чином, середня кількість сприйнятливих осіб, інфікованих інфекційним за день, дорівнює λS, а середня кількість сприйнятливих осіб, заражених інфекційним класом із розміром NI на день, — λSNI. Щоденна норма контакту λ є фіксованою та не змінюється в залежності від сезону. Тип прямого чи непрямого контакту, адекватного для передачі, залежить від конкретного захворювання. Кількість випадків на день λSNI, що називається захворюваністю, є законом масової дії, оскільки включає добуток S та I.

3. Особи одужують і вилучаються з класу інфікованих відповідно до пропорційної кількості інфікованих з константою пропорційності γ, що

називається добовою нормою відновлення. Латентний період дорівнює нулю (його визначають як період між моментом зараження та часом, коли починається заразність). Таким чином пропорція осіб, які зазнали впливу (і негайно заразилися) у момент часу t_0, які все ще є інфекційними в момент часу $t_0 + t$ це $exp(-\gamma t)$, а середній період інфекційності становить l/γ.

Коефіцієнт виведення з класу інфекційних як за одужанням, так і за смертю, становить $\gamma + \mu$, так що скоригований на смерть середній період інфекційності становить $l/(\gamma+\mu)$. Таким чином, середня число адекватних контактів (як із сприйнятливими, так і з іншими) інфікованого протягом періоду зараження дорівнює $\sigma = \lambda/(\gamma+\mu)$, яке називається номером контакту. Цю величину також називають основним коефіцієнтом відтворення, хоча це число, а не курс. Оскільки середня кількість сприйнятливих осіб, заражених інфекцією протягом інфекційного періоду, дорівнює σS, кількість σS називається числом заміни.

2.1 Модель SIR

SIR модель включає поділ населення на три групи:

S(t) — кількість осіб, сприйнятливих до захворювання;

I(t) — кількість інфікованих осіб;

R(t) — кількість осіб, які одужали й мають імунітет або загинули [4].

Модель SIR була запропонована ще в 1920-х роках Керманом і Маккендріком, забезпечує основне якісне розуміння динаміки поширення інфекційних захворювань, але для кількісного моделювання цієї динаміки необхідні різні уточнення, з урахуванням особливостей перебігу конкретних захворювань. Отже, важлива особливість багато захворювань — це наявність інкубаційного періоду, протягом якого людина вже є носієм хвороби, але не

проявляє симптомів і не є заразною для оточуючих. Цю особливість можна врахувати, поділивши групу інфікованих (Infected) на дві підгрупи - експоновані (Exposed), тобто інфіковані в стадії інкубаційного періоду, і контагіозні (Infections). Отже, модель SIR трансформується в модель SEIR [40,41]

Ця модель не є складною при розв'язуванні й водночас дає змогу моделювати поширення багатьох інфекційних захворювань, в тому числі кору, ендемічного паротиту та краснухи, а також оцінити ефективність карантинних заходів різної тривалості. Крім того, SIR модель може використовуватися для моделювання динаміки антагоністичних протистоянь [5], а також як основа методології вирощування даних (Data Farming) для тренування і тестування нейронних мереж [40, с.45].

Для того щоб, показати, що значення S, I, R змінюються з часом (навіть якщо загальна чисельність населення залишається незмінною), їх слід позначити як залежні від часу функції S(t), I(t) та R(t). Ці функції будуть змінювати в залежності від захворювання та популяції, щоб мати змогу спрогнозувати можливі спалахи й взяти їх під контроль.

Класична модель епідемії виглядає наступним чином (2.1):

$$\begin{cases} \frac{dS}{dt} = -\beta \frac{SI}{N} \\ \frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases} \qquad (2.1)$$

де S(t) — кількість людей, які можуть бути заражені;

I(t) — кількість заражених людей;

R(t) — кількість людей, що були виокремлені із передачі (померли або вилікувалися);

N = S(t) + I(t)+R(t) — загальна кількість населення, константа.

На Рис. 2.1 зображено структуру SIR-моделі:

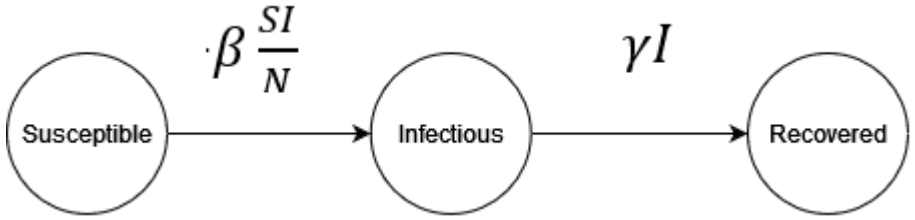

Рис. 2.1. — Схема можливих переходів між групами в SIR-моделі

Тобто, *S(t)/N* — частка населення, яка може бути інфікована. Припустімо, що кожна інфікована особа має в середньому μ контактів, тоді $\mu S(t)I(t)/N$ щоденних зустрічей можуть призвести до зараження. Однак розумно припустити, що лише частина цих зустрічей $\tau < 1$ ефективно призводить до зараження. Отже, кількість нових інфікованих за наступну добу становитиме *S(t)/N* – частка населення, яка може бути інфікована. Отже, кількість нових інфікованих за наступну добу становитиме (2.2):

$$\frac{\tau\mu S(t)}{N} = \lambda S(t), \qquad (2.2)$$

де $\lambda=\tau\mu N$ — рівень зараження.

Але кількість інфікованих також зменшується, якщо вони одужують або помирають. Якщо середній час одужання становить D днів, частка β=1/D інфікованих буде одужувати щодня. Нарешті, загальну кількість інфікованих за наступний день можна визначити за формулою (2.3):

$$I(t + \Delta t) = I(t) + [\lambda S(t)I(t) - \beta I(t)]\Delta t, \qquad (2.3)$$

де Δt — одиниця часу, що відповідає певному інтервалу часу, який може означати, наприклад, одну добу.

Якщо N має велике значення, то можна вважати змінні неперервними, коли зменшується інтервал часу, тобто (2.4):

$$\frac{dI(t)}{dt} = \lim_{\Delta t \to 0} \frac{I(t+\Delta t) - I(t)}{\Delta t} = \lambda S(t) I(t) - \beta I(t), \qquad (2.4)$$

На початку нової епідемії, що відповідає $t = 0$ у нашому математичному підході, можемо припустити загальну гіпотезу про те, що практично всі люди сприйнятливі, за винятком нульової особи — це означає $S(t=0) \approx N$. Це значення залишається досить постійним на перших етапах зараження.

Початкові умови (2.5):

$$\begin{cases} \mathrm{S}(0) = \mathrm{N} - 1 \\ \mathrm{I}(0) = 1 \\ \mathrm{R}(0) = 0 \end{cases} \qquad (2.5)$$

Маємо рисунок 2.2:

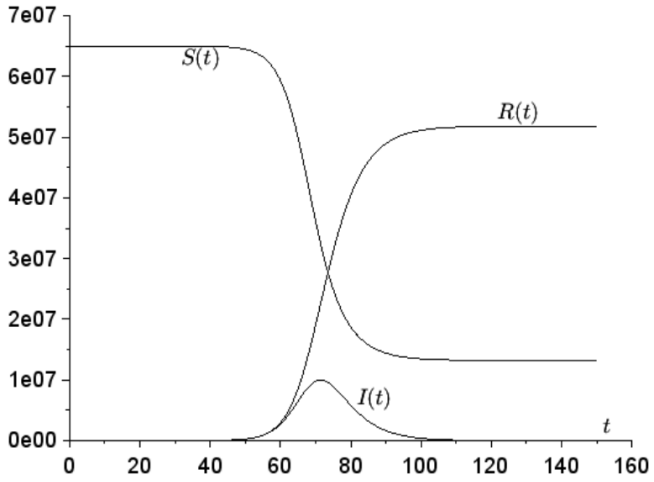

Рис. 2.2. — Імітація моделі SIR

При об’єднанні (2.1) та (2.5) маємо (2.6):

$$\frac{1}{S}\frac{dS}{dt} = -\beta\frac{1}{N} = -\frac{\beta}{\alpha N}\frac{dR}{dt} \qquad (2.6)$$

Звідси можна зробити висновок, що $\log\left(\frac{S(t)}{S(0)}\right) = -\frac{\beta}{\gamma N}R(t)$. З Першого рівняння знаходимо (2.7):

$$\frac{dS}{dt} = -\beta\frac{S}{N}(N - S - R) = -\beta\frac{S}{N}(N - S + \frac{\beta N}{\gamma}\log\left(\frac{S(t)}{S(0)}\right)) \qquad (2.7)$$

Пік епідемії на рис. 2.1 відповідає умові *dI/dt=0*, що відбувається із рівняння (2.5), коли *S(t)=Nb/a.* S(t) зменшується монотонно. Можемо зробити висновок, що якщо час *T* — час піка епідемії, то (2.8):

$$I(T) = \max_{t > 0} I(t) \qquad (2.8)$$

$$T = \int_0^T dt = \int_{S(0)}^{bN/a} \frac{dS}{-\beta\frac{S}{N}(N-S+\frac{\gamma N}{\beta}\log\left(\frac{S(t)}{S(0)}\right)} \quad (2.9)$$

Для прикладу, можемо навести кількість людей, інфікованих новим коронавірусом (SARS-CoV-2). Перший зареєстрований випадок стався в Ухані, Китай, 31 грудня 2019 року. Через місяць, 31 січня 2020 року, було 9826 інфікованих осіб, згідно з Ситуаційним звітом 11 ВООЗ [40, с.186]. Це відповідає мізерній частині світового населення — понад сім мільярдів людей, за даними Організації Об'єднаних Націй. Враховуючи це наближення, маємо (2.10):

$$\frac{dI(t)}{dt} = (\lambda S(0) - \beta)I(t), \quad (2.10)$$

Що дає (2.11) як рішення еволюції кількості інфікованих людей на початку епідемії.:

$$I(t) = I(0)\mathrm{e}^{(\lambda S(0)-\beta)t}, \quad (2.11)$$

За допомогою цього виразу можна отримати цінну інформацію. Якщо $\lambda S(0)-\beta > 0$, кількість інфікованих зростає експоненційно.

Але якщо $\lambda S(0)-\beta < 0$, кількість інфікованих зменшується до повного зникнення епідемії. Значення $\lambda S(0)/\beta = 1$ є епідемічним порогом, який розділяє дві різні фази епідемії. Коли початковий стан відповідає всім сприйнятливим людям, як це сталося, наприклад, під час поширення COVID-19, маємо окремий випадок, коли значення $\lambda S(0)/\beta = \lambda N/\beta$ відоме як базове репродуктивне число, і воно вимірює «інтенсивність зараження» — кількість зараження, яке може спричинити кожна інфікована особа. Це означає, що кількість інфікованих збільшиться, оскільки $\lambda S(t)/\beta > 1$. З іншого боку, коли дивимося, як кількість

сприйнятливих людей змінюється з часом, то можна дійти до висновку, що ця кількість завжди зменшуватиметься з часом, оскільки (2.12):

$$\frac{dS(t)}{dt} = -\lambda S(t)I(t), \qquad (2.12)$$

Тому буде час, коли величина $\lambda S(t)/\beta$ стане меншою за одиницю, а отже, кількість заражених почне зменшуватися. Тобто, кількість інфікованих осіб експоненційно швидко зростає на початку епідемії, досягає піку і починає знижуватися. Це природна поведінка епідемії. Проте чекати, поки значна частина населення заразиться, щоб пом'якшити епідемію, безумовно, не є найкращою стратегією, особливо коли хвороба має високі показники смертності та летальності.

Можна згадати, що $\lambda=\mu\tau/N$, $\beta=1/D$ і на початку епідемії $S(t) \approx N$, то якщо $\lambda S(t)/\beta=\mu\tau D<1$, то епідемія починає спадати.

2.2 Модель SIS

SIS модель включає поділ населення на три групи:

S(t) — кількість осіб, сприйнятливих до захворювання;

I(t) — кількість інфікованих осіб;

S(t) — кількість осіб, сприйнятливих до захворювання.

Модель SIS (Susceptible-Infected-Susceptible) можна отримати з моделі SIR у припущенні, що індивіди, що вилікувалися, не набувають імунітету до захворювання, тобто. знову можуть перейти до групи сприйнятливих. Це особливо актуально для ГРВІ, що характеризуються великою різноманітністю типів вірусу.

Схема SIS-моделі виглядає наступним чином (Рис. 2.3):

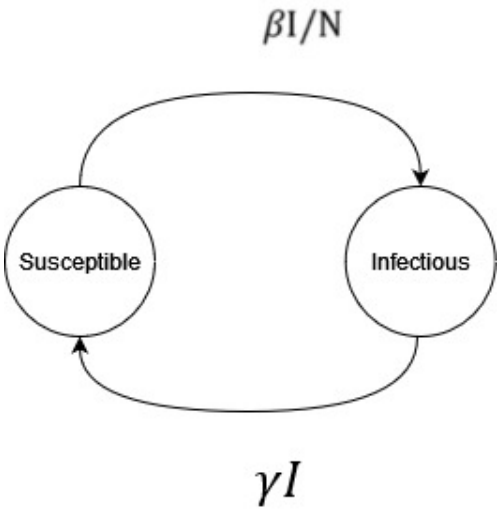

Рис. 2.3. — Схема можливих переходів між групами в SIS-моделі

Прибравши рівняння, що описує приріст одужуючої частини населення, отримаємо систему (2.13):

$$\begin{cases} \frac{dS}{dt} = -\beta \frac{SI}{N} \\ \frac{dI}{dt} = \beta \frac{SI}{N} - \gamma I \end{cases} \quad (2.13)$$

де β — швидкість одужання в одиницю часу;

γ — швидкість смерті за одиницю часу.

2.3 Модель SRID

Single radial immunodiffusion (SRID). Передбачається, що процес поширення інфекції відбувається у популяції людей з загальною чисельністю N осіб. Позначимо:

S(t) — кількість людей із популяції, які ще не проконтактували з патогеном;

R(t) — кількість одужалих людей, які проконтактували з патогеном та отримали стійкі імунітет;

I(t) — кількість хворіють чи вірусоносіїв;

D(t) — кількість померлих;

K(t) — загальна кількість інфікованих та одужалих (K=I+R+D,K=N-S);

Z(t) — кількість нових заражень протягом дня. Змінні у моделі повинні підходити під умови N=S+I+R+D, N=S+L.

Припустимо, що у моделі шаг зміни часу t дорівнює одному дню. Момент часу t=1 — це час появи у популяції першого захворівшего. Тоді система рівнянь буде виглядати наступним чином (2.14):

$$\begin{cases} S(t) = S(t-1) - \frac{\alpha}{N} S(t-1) I(t-1) \\ I(t) = I(t-1) + \frac{\alpha}{N} S(t-1) I(t-1) - \beta I(t-1) - \gamma I(t-1) \\ R(t) = R(t-1) + \beta I(t-1) \\ D(t) = D(t-1) + \gamma I(t-1) \\ Z(t) = \frac{\alpha}{N} S(t-1) I(t-1) \\ K(t) = N - S(t) \end{cases} \quad (2.14)$$

де α — швидкість зараження в одиницю часу (кількість заражених одним хворим інших людей за добу);

β — швидкість одужання в одиницю часу;

γ — швидкість смерті за одиницю часу.

Значення параметра α визначається заразністю вірусу (біологічні властивості патогену, що визначають ймовірність зараження при безпосередньому контакті хворого та здорового, стійкістю патогенна в навколишнє стері тощо) та кількістю контактів інфікованого з іншими здоровими

людьми (швидкістю перемішування людей у популяції). Значення параметрів β та γ визначаються біологічними властивостями патогену та ефективністю терапії, при виникненні захворювання.

Алгоритм розрахунку параметрів з урахуванням реальних даних. Якщо спостерігається початковий етап розвитку епідемії деякого інфекційного захворювання в популяції з N людина, то маємо ряд даних, що описують значення змінних S, R, K, D та I протягом ряду днів розвитку процесу. Нехай *td* — момент, до якого нам відомі реальні дані про розвиток епідемічного процесу. Тоді час *t* буде маняться від 1 до td (t , 2, 3, ... td). Позначимо реальні дані про епідемічний процес за час *td* як S', R', I' і D'. Виходячи з даних щодо зміни реальних значень змінних на кожному тимчасовому кроці та рівняння (2.7), можна розрахувати вибірки значень параметрів αt, βt та γt за формулами (2.15), (2.16), (2.17):

$$\alpha_t = -\frac{N\left(S'(t)-S'(t-1)\right)}{S'(t-1)I'(t-1)} \quad \overline{\alpha} = \frac{1}{td-1}\sum_{t=2}^{td}\alpha_t \qquad (2.15)$$

$$\beta_t = -\frac{\left(R'(t)-R'(t-1)\right)}{I'(t-1)} \quad \overline{\beta} = \frac{1}{td-1}\sum_{t=2}^{td}\beta_t \qquad (2.16)$$

$$\gamma_t = -\frac{\left(D'(t)-RD'(t-1)\right)}{I'(t-1)} \quad \overline{\gamma} = \frac{1}{td-1}\sum_{t=2}^{td}\gamma_t \qquad (2.17)$$

де $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ – це середнє значення параметрів, які можна використовувати у рівнянні (2.5).

2.4 Модель SEIR

SEIR-модель, в якій передбачається, що захворювання має латентний період, тобто перші ознаки захворювання з'являються у зараженого через певний проміжок часу. З урахуванням цього фактору населення ділиться на 4 групи:

$S(t)$ — кількість сприйнятливих до вірусу;

$E(t)$ — кількість людей з вірусом у інкубаційному періоді;

I(t) — кількість людей, що хворіють;

R(t) — кількість одужалих людей, які проконтактували з патогеном та отримали стійкі імунітет.

Ця модель є розвитком класичної SIR-моделі (2.1), запропонованої Кермаком і Маккендріком у 1927 р. [15].

Модель SEIR — найпоширеніший інструмент для прогнозування епідемій та дієвості заходів щодо їх придушення. Вона є математичною компартментальною моделлю, заснованою на середній поведінці досліджуваної популяції. Мета полягає в тому, щоб надати дослідникам краще розуміння значення математичного моделювання для епідемічних захворювань. За допомогою моделювання показано, як соціальні заходи, такі як дистанціювання, регіональні блокування та пильність громадської охорони здоров'я можуть впливати на параметри моделі, що, у свою чергу, змінює рівень смертності та активні випадки зараження з часом.

У 2020 році модель була доопрацьована Річардом Нейєром та його співробітниками у Базельському університеті з урахуванням особливостей епідемії нового коронавірусу. Модель реалізована як комп'ютерна програма для моделювання епідемії COVID-19, доступної інтернеті [42]. Розрахунки з цього варіант SEIR, зокрема, використовувалися при прийнятті рішення про введення обмежувальних заходів у штаті Іллінойс і в його найбільшому місті Чикаго. Основна властивість моделі SEIR — наявність так званого епідемічного переходу: модель поводиться радикально по-різному залежно від показника R_0 (базове репродуктивне число) — середньої кількості людей, яких один заражений встигає заразити за час, поки сам не одужає.

При $R_0 < 1$ епідемія загасає, за показника $R_0 > 1$ заражається значна частина населення. Значення R_0 залежить від особливостей вірусу, частки населення, яка

отримала імунітет внаслідок вакцинації або пережитого захворювання), а також заходів щодо придушення епідемії (різні форми карантину) [43].

У SEIR-моделі враховується можливість, що вірус може мати якийсь «латентний період», під час якого хвороба не задає шкоди інфікованій особині. Зазвичай вірус заражає вразливу особину (S) до входу до своєї латентної стадії. Протягом латентного періоду (E, Exposed) людина вважається зараженою, але не поширює вірус. Через деякий час хвороба переходить із латентної стадії до відкритої, і особа набуває можливості заражати інших (I, Infectious) і далі перехворіє хворобою та стає (R, Recovered).

Схема SEIR-моделі виглядає наступним чином (Рис. 2.4):

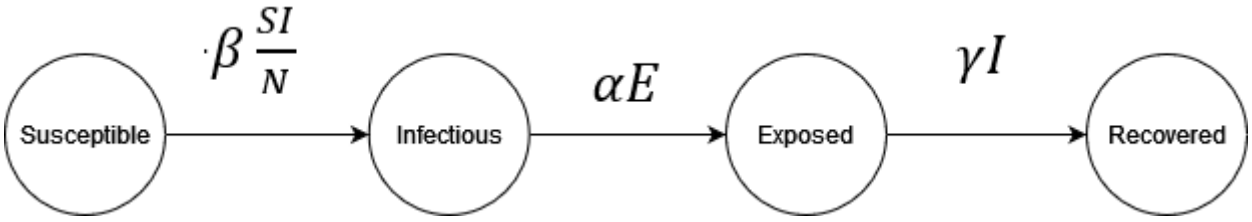

Рис. 2.4. — Схема можливих переходів між групами в SEIR-моделі

SEIR-модель описується наступною системою рівнянь (2.18):

$$\begin{cases} \frac{dS}{dt} = -\beta \frac{SI}{N} \\ \frac{dE}{dt} = \beta \frac{SI}{N} - \alpha E \\ \frac{dI}{dt} = \alpha E - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases} \quad (2.18)$$

де α — швидкість переходу хвороби з латентного періоду до відкритої стадії;

β — швидкість передачі інфекції;

γ — швидкість одужання.

2.5 Модель SEIRD

Для опису епідемії COVID-19 з вищезгаданих моделей найбільш підходящою є SEIRD-модель, в якій передбачається, що захворювання має латентний період, тобто перші ознаки захворювання з'являються у зараженого через певний проміжок часу. З урахуванням цього фактора населення ділиться на 5 груп:

- сприйнятливі ($S(t)$),
- латентні ($E(t)$),
– інфіковані ($I(t)$),
- несприйнятливі ($R(t)$),
– померлі ($D(t)$)

На рис. 2.5 можна побачити схему можливих переходів між даними групами:

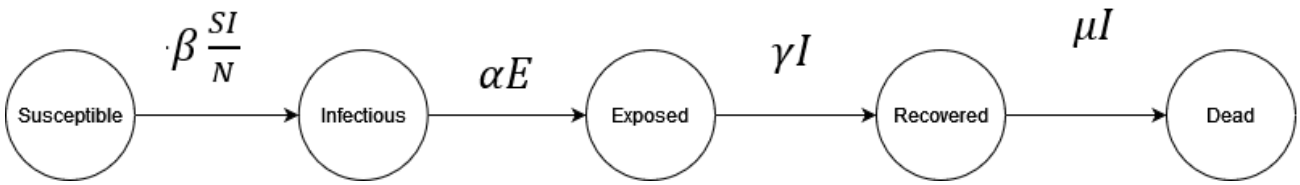

Рис. 2.5. — Схема можливих переходів між групами в SEIRD-моделі

Розглянемо фактори, що впливають на динаміку якихось груп. Очевидно, що інфіковані та перехід до латентної фази відбувається у результаті контакту уразливих та інфікованих людей. При чому, відносна частота такого контакту

дорівнює: $\frac{S(t)}{N}\frac{I(t)}{N}$. Тобто, швидкість зміни долі латентних людей $\frac{d}{dt}\left(\frac{E(t)}{N}\right)$ має складову, що пропорційна $-\frac{E(t)}{N}$.

Тобто, маємо (2.19):

$$\frac{d}{dt}\left(\frac{E(t)}{N}\right) = \beta\frac{SI}{N} - \alpha E \qquad (2.19)$$

Розмірковуючи аналогічним чином інших груп популяції, отримаємо систему диференціальних рівнянь, що описують динаміку епідеміологічного процесу (SEIRD-модель) (2.20):

$$\begin{cases} \frac{dS(t)}{dt} = -\beta\frac{SI}{N} \\ \frac{dE(t)}{dt} = \beta\frac{SI}{N} - \alpha E \\ \frac{dI(t)}{dt} = \alpha E - (\gamma + \mu)I \\ \frac{dR(t)}{dt} = \gamma I \\ \frac{dD(t)}{dt} = \mu I \end{cases} \qquad (2.20)$$

де β − коефіцієнт, який можна інтерпретувати як ймовірність отримання хвороби у разі контакту сприйнятливого індивіда з інфікованим;

μ — коефіцієнт смертності;

γ — швидкість одужання;

α — швидкість переходу хвороби з латентного періоду до відкритої стадії.

2.6 Висновки до розділу 2

У розділі 2 були описані загальні поняття, пов’язані із епідеміологічним моделюванням. Найбільше уваги було приділено математичним моделям SIS, SEIR, SEIRD, SRID, які застосовуються для прогнозування перебігу епідемій. Динаміка епідеміологічного процесу описана з допомогою системи диференційних рівнянь, розв’язки яких характеризують зміну чисельності у різних підгрупах популяції.

При цьому важливо відзначити, що збудник грипу постійно змінюється, що спричиняє необхідність частої зміни відповідної вакцини. Генетична структура вірусу модифікується з такою швидкістю, що на час закінчення розробки нової вакцини вона може виявитися не актуальною. У традиційній моделі йдеться лише про один тип вірусу, але нерідко відбуваються спалахи епідемій, коли відбувається поширення двох вірусних типів із різною силою.

Поведінкові зміни є важливим аспектом перебігу епідемії. Зміни в поведінці інфекційних членів популяції мають інші причини, ніж зміни в поведінці неінфікованих членів, і модель, що включає поведінкові зміни, повинна відображати це. Одним із наслідків є те, що модель, яка включає зміни поведінки, повинна включати гетерогенне змішування. Одним із наслідків цього є те, що кінцевий розмір епідемії не можна точно визначити на основі остаточного співвідношення розмірів, а лише приблизно. У багатьох випадках існує ефективне число відтворення, яке є меншим за базове число відтворення, але не обов’язково завжди.

Моделі епідемії з віковою структурою або іншими неоднорідностями в змішуванні також можуть бути розширені для включення поведінкових змін. Це призведе до моделей зі складною поведінкою змішування, які буде важко проаналізувати. Існувала б система рівнянь остаточного розміру, яку неможливо

було б точно розв’язати, але вона все одно давала б остаточні оцінки розміру. Важливим питанням, яке ще не розглядалося, є формулювання моделей, які включають поведінкові реакції, особливо неінфікованих членів населення, які залежать від стану та історії епідемії.

Усі модифіковані моделі, що були розроблені в рамках цієї магістерської дисертації будуть розглянуті у третьому розділі.

РОЗДІЛ 3 АНАЛІЗ ДАНИХ, ПОБУДОВА МОДИФІКОВАНИХ МОДЕЛЕЙ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Основним завданням цієї роботи та проекту була розробка моделі, яка зможе відобразити ситуацію із вірусними захворюваннями, коли є подільність на вакцинованих та невакцинованих людей. Для цього було розроблено та реалізовано програмне забезпечення.

Для моделювання було вирішено використати мову програмування Python та пакет PyQt5 для візуального відображення даних. Python має наступні переваги, через що було обрано саме його:

— Python має велику кількість вбудованих бібліотек для аналізу даних [45, с.56], маніпулювання даними та машинного навчання, зокрема бібліотека Pandas. Pandas забезпечує спрощені форми представлення даних та їхній аналіз [46, с.89-98]. Спрощене представлення даних сприяє кращим результатам проектів з обробки даних.

Для розробки інтерфейсу було використано бібліотеку PyQt5. Її було обрано, тому що:

— Qt пропонує декілька віджетів, таких як кнопки чи меню, усі вони розроблені з базовим виглядом на всіх підтримуваних платформах [46 с.78];

— Програмування на PyQt є гнучким, бо Qt розроблено навколо концепції сигналів і слотів для встановлення зв’язку між об’єктами, що забезпечує більш плавну кодову базу.

Для контролю версій було вирішено використовувати Git, зокрема — GitHub. По-перше, GitHub містить потужне програмне забезпечення для керування версіями, яке може легко виконувати розгалуження та злиття [47, с.88-

99]. По-друге, GitHub чудово підходить для віддаленої співпраці. Для тих, хто не перебуває в тому самому фізичному місці, онлайновий Git є чудовим рішенням. Використання онлайн-репозиторію надає гарний і простий спосіб мати код і історію версій доступними в Інтернеті, незалежно від того, що відбувається з локальною машиною.

3.1 Запропоновані модифіковані моделі

Використання епідеміологічних моделей досить різноманітне. Деякі моделі були орієнтовані на вирішення конкретних проблем і тому можуть бути більш корисними для одних цілей, ніж для інших. Моделі великої складності мають обмежене використання в практиці охорони здоров'я, оскільки вони зазвичай вимагають надання обширної інформації про фактичну епідеміологічну ситуацію, а така інформація може бути недоступною, зокрема в країнах, що розвиваються.

Більш практичними є відносно прості моделі, які вимагають лише необхідної інформації, наприклад, про чисельність населення, захворюваність і, якщо необхідно, обсяг і вартість запланованих медичних заходів. У цій роботі буде обговорення буде обмежено використанням моделей, що стосуються гострих бактеріальних інфекцій.

В рамках цієї магістерської дисертації було розглянуто та змодельовано п’ять моделей: SIR (Susceptible, Infectious, Recovered) (2.1), SEIR (Susceptible, Exposed, Infectious, Recovered) (2.18), SEIR з двома модифікаціями та модифікована SEIRD-модель.

Розглянемо модифіковані SEIR-моделі, які були змодельовані у рамках цієї роботи. Перша модифікована SEIR-модель має наступний вигляд (3.1):

$$\begin{cases} \frac{dS}{dt} = -(1-u)\beta\frac{SI}{N} \\ \frac{dE}{dt} = (1-u)\beta\frac{SI}{N} - \alpha E \\ \frac{dI}{dt} = \alpha E - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases} \quad (3.1)$$

де S — кількість ще не перехворівших людей;

E — кількість людей із захворюванням у латентному режимі;

I — кількість хворих людей;

R — кількість перехворівших та отримавших імунітет людей;

u —успіх дистанційних заходів для цілей моделювання;

β — швидкість одужання в одиницю часу;

α — швидкість переходу захворювання із латентної фази у відкриту;

γ — швидкість смертності за одиницю часу.

На відміну від звичайної SEIR-моделі, ця (3.1) має параметр, що відповідає за норму дистанціювання соціуму. Тобто, $(1-u)\beta\frac{SI}{N}$ — це швидкість, з якою сприйнятливе населення зустрічається з інфікованим населенням, що призводить до передати захворювання.

При $u = 0$ маємо звичайну SEIR-модель (2.18), тобто відсутність ефективності будь-якого втручання громадської охорони здоров'я для контролю передачі хвороби.

При $u = 1$ маємо повне усунення передачі хвороби.

Друга модифікована SEIR-модель має таку характеристику як ступінь заразності латентних носіїв інфекції у порівнянні із захворілими (3.2):

$$\begin{cases} \frac{dS}{dt} = -\beta \frac{S(I+\theta E)}{N} \\ \frac{dE}{dt} = \beta \frac{S(I+\theta E)}{N} - \alpha E \\ \frac{dI}{dt} = \alpha E - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases} \qquad (3.2)$$

де S — кількість ще не перехворівших людей;

E — кількість людей із захворюванням у латентному режимі;

I — кількість хворих людей;

R — кількість перехворівших та отримавших імунітет людей;

Θ — ступінь заразності латентних носіїв інфекції у порівнянні із захворілими;

β — швидкість одужання в одиницю часу;

α — швидкість переходу захворювання із латентної фази у відкриту;

γ — швидкість смертності за одиницю часу.

Пандемія COVID-19 вплинула на життя людей у всьому світі. Минулого року багато дослідників запропонували різні моделі та підходи для вивчення того, як можна пом’якшити поширення хвороби. Однією з моделей, яка широко використовується, є модель чутливих до інфекційних захворювань (SEIRD). Однак, ця модель не розглядає наявність вакцинованих особин у популяції. Саме тому у цій роботі була запропонована модифікована SEIRD-модель, що враховує цей фактор.

Недоліками моделей (3.1) (3.2) для модуляції є, по-перше, відсутність смертності, що не дає можливість відобразити реальну картину COVID-пандемії.

По-друге, у моделі (3.1) параметр u (успіх дистанційних заходів для цілей моделювання) надто загальний, він не має змоги відобразити повну картину, пов’язану з вакцинаціями населення. Саме тому акцент робиться на модифікованій SEIRD-моделі.

Переваги запропонованої моделі пов’язані із урахуванням того, що популяція може бути поділена на вакциновану та невакциновану. Крім того, модифікована SEIRD-модель враховує народжуваність та природню смертність, що не пов’язана з смертністю від хвороби, що дозволить якнайточніше відтворити ситуацію, максимально наблизивши її до реальних умов.

Базова SEIRD-модель має вигляд (2.20). У модифікованій моделі кожна складова ділиться на дві частини: вакциновані та невакциновані представники популяції (Susceptible, Exposed, Infectious, Recovered). Параметр народжуваності стосується лише невакцинованих сприятливих, тому що за замовчуванням люди народжуються невакцинованими.

Маємо модель (3.3):

$$\begin{cases} \dfrac{dS_{unvac}}{dt} = l - \mu S_{unvac} - \dfrac{S_{unvac}(\beta_{uu}I_u + \beta_{uv}I_v)}{N} \\ \dfrac{dS_{vac}}{dt} = -\mu S_{vac} - \dfrac{S_{vac}(\beta_{vu}I_u + \beta_{vv}I_v)}{N} \\ \dfrac{dE_{unvac}}{dt} = \dfrac{S_{unvac}(\beta_{uu}I_u + \beta_{uv}I_v)}{N} - (\mu + \alpha_{unvac})E_{unvac} \\ \dfrac{dE_{vac}}{dt} = \dfrac{S_{vac}(\beta_{vu}I_u + \beta_{vv}I_v)}{N} - (\mu + \alpha_{vac})E_{vac} \\ \dfrac{dI_{unvac}}{dt} = \alpha_{unvac}E_{unvac} - (\gamma_{unvac} + \mu + \theta_{unvac})I_{unvac} \\ \dfrac{dI_{vac}}{dt} = \alpha_{vac}E_{vac} - (\gamma_{vac} + \mu + \theta_{vac})I_{vac} \\ \dfrac{dR_{unvac}}{dt} = \gamma_{unvac}I_{unvac} - \mu R_{unvac} \\ \dfrac{dR_{vac}}{dt} = \gamma_{vac}I_{vac} - \mu R_{vac} \\ \dfrac{dD}{dt} = \theta_{vac}I_{vac} + \theta_{unvac}I_{unvac} + \mu(S_{vac} + S_{unvac} + E_{unvac} + E_{vac} + I_{unvac} \\ \qquad + I_{vac} + R_{unvac} + R_{vac}) \end{cases} \tag{3.3}$$

де S_{unvac} — сприятливі для вірусу невакциновані особи населення, які не інфіковані, але можуть заразитися при контакті з інфікованою особою (невакцинованою або вакцинованою);

S_{vac} — сприятливі для вірусу вакциновані особи населення, які не інфіковані, але можуть заразитися при контакті з інфікованою особою (невакцинованою або вакцинованою);

E_{unvac} — кількість невакцинованих людей із захворюванням у латентному режимі (контактували з інфікованою особою);

E_{vac} — кількість вакцинованих людей із захворюванням у латентному режимі (контактували з інфікованою особою);

I_{unvac} — кількість невакцинованих хворих людей, що передають вірус невакцинованим і вакцинованим сприятливим особам;

I_{vac} — кількість вакцинованих хворих людей що передають вірус невакцинованим і вакцинованим сприятливим особам;

R_{unvac} — кількість невакцинованих перехворівших, що сприйнятливі до повторного зараження, хоча ймовірність менша;

R_{vac} — кількість вакцинованих перехворівших, що сприйнятливі до повторного зараження, хоча ймовірність менша;

D — померлі і від вірусу, і з інших причин;

Θ_{unvac} — смертність від вірусу інфікованих невакцинованих;

Θ_{vac} — смертність від вірусу інфікованих вакцинованих;

β_{uu} — ймовірність передачі вірусу від інфікованих невакцинованих до невакцинованих;

β_{uv} — ймовірність передачі вірусу від інфікованих невакцинованих до вакцинованих;

β_{vu} — ймовірність передачі вірусу від інфікованих вакцинованих до невакцинованих;

β_{vv} — ймовірність передачі вірусу від інфікованих вакцинованих до вакцинованих;

α_{unvac} — ймовірність переходу захворювання із латентної фази у відкриту у невакцинованих;

α_{vac} — ймовірність переходу захворювання із латентної фази у відкриту у вакцинованих;

γ_{unvac} — одужання інфікованих невакцинованих від вірусу;

γ_{vac} — одужання інфікованих вакцинованих від вірусу;

μ — смертність не від інфекції;

l — народжуваність.

Як можна побачити, основна відмінність цієї моделі від класичної це поділення популяції на вакцинованих та невакцинованих. Ймовірність зараження вакцинованих осіб (S_{vac}) набагато нижча, ніж у невакцинованих осіб (S_{unvac}). Вакциновані особи, які зазнали контакту (E_{vac}), менш схильні до розвитку

симптомів і менш заразні, ніж невакциновані особи, які зазнали впливу (E_{unvac}). Хворі вакциновані особи (I_{vac}) менш заразні, менше може розвинути серйозні симптоми або померти, ніж у невакцинованих інфекційних осіб (I_{unvac}).

У випадку, якщо $l = \mu = \alpha_{vac} = \alpha_{unvac} = 0$, маємо класичну SIR-модель (2.1).

Концепція модифікованої SEIRD-моделі проілюстрована на (Рис. 3.1). Так само припускається, що часовий масштаб моделі досить короткий, щоб народжуваність і природна смертність (смерті, не спричинені COVID-19) були незначними, а кількість смертей від COVID-19 невелика порівняно з живим населенням.

Параметри переходу позначені стрілками між відсіками. Хоча знову існує технічний зв'язок між усіма живими відсіками та D (через природні смерті), додавання всіх із них ускладнило б діаграму, тому їх було залишено поза увагою.

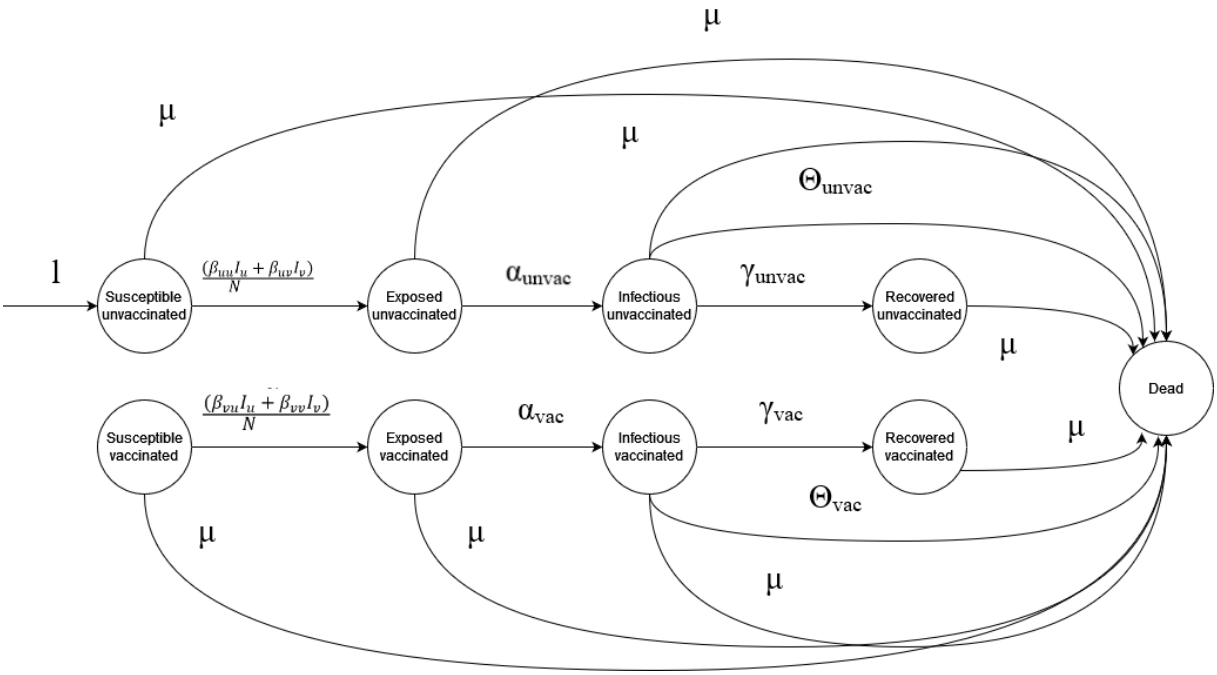

Рис. 3.1. — Ілюстрація модифікованої SEIRD-моделі

Всі моделі є спрощенням реальності. Вони не можуть вловити кожну деталь про те, хто з ким взаємодіє, як це змінюється з часом у зв'язку з урядовою політикою та змінами в поведінці, а також безліч відмінностей між людьми.

Натомість вони намагаються зосередитися на найважливіших факторах, які впливають на траєкторію епідемії та її вплив на рівні населення.

Рішення про те, які спрощення та припущення доцільно використовувати з конкретною моделлю, залежить від деталей питань, на які ви хочете отримати відповідь від моделі. Різні моделі включатимуть різні рівні деталізації епідемії. Часто рішення про те, які деталі включити, залежить від того, чи доступні достатньо якісні дані про цю змінну і чи ймовірно, що вона буде важливою для результатів, які цікавлять.

Моделі використовуються різними способами залежно від того, що намагаються зрозуміти вчені, працівники охорони здоров'я чи політики. Вони можуть використовувати останні дані про випадки захворювання, щоб створити короткостроковий прогноз (зазвичай на кілька тижнів), який може бути корисним для планування можливостей охорони здоров'я.

Моделі можна використовувати для створення середньострокових прогнозів (зазвичай на 2-3 місяці) траєкторії епідемії, якщо умови залишаться такими, як є, або зміняться певним чином, наприклад, пом'якшення заходів охорони здоров'я. Моделі також можна використовувати для дослідження довгострокових сценаріїв. Це може бути корисно, щоб подумати про запитання типу «що, якщо», наприклад, що, якщо з'явиться новий варіант із певними характеристиками, але за необхідності вони включають більше невизначеності.

3.2 Програмна реалізація

Програмно було змодельовано п'ять вищезазначених моделей. Для кожної передбачено введення необхідних коефіцієнтів та можливість побудувати графіки на довгостроковий термін.

Для SIR-моделі графік програмний продукт виглядає наступним чином (Рис. 3.2):

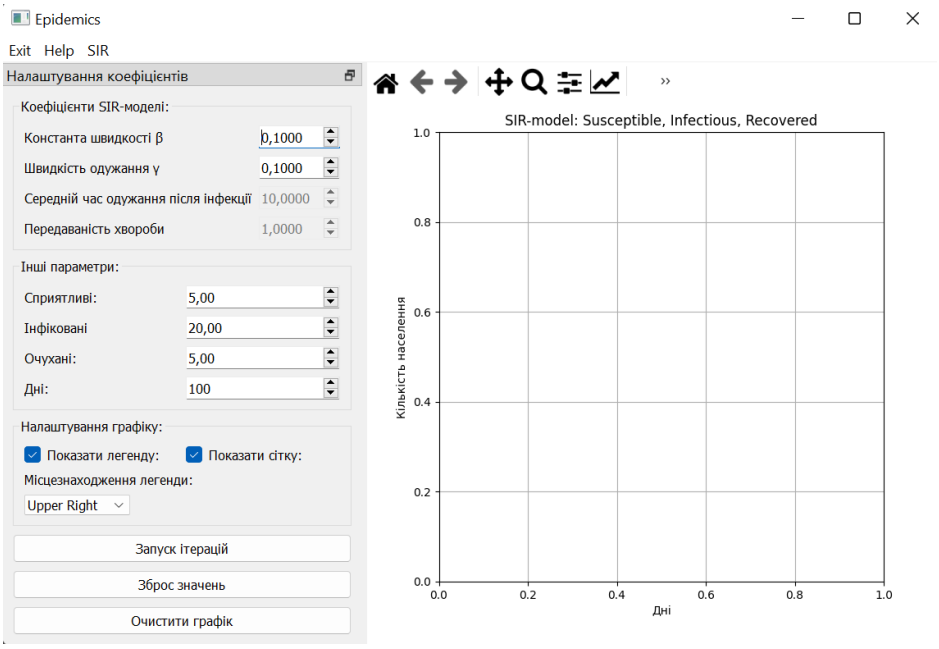

Рис. 3.2. — Інтерфейс SIR-моделі

У кожної моделі є інформація про неї з описом (Рис. 3.3):

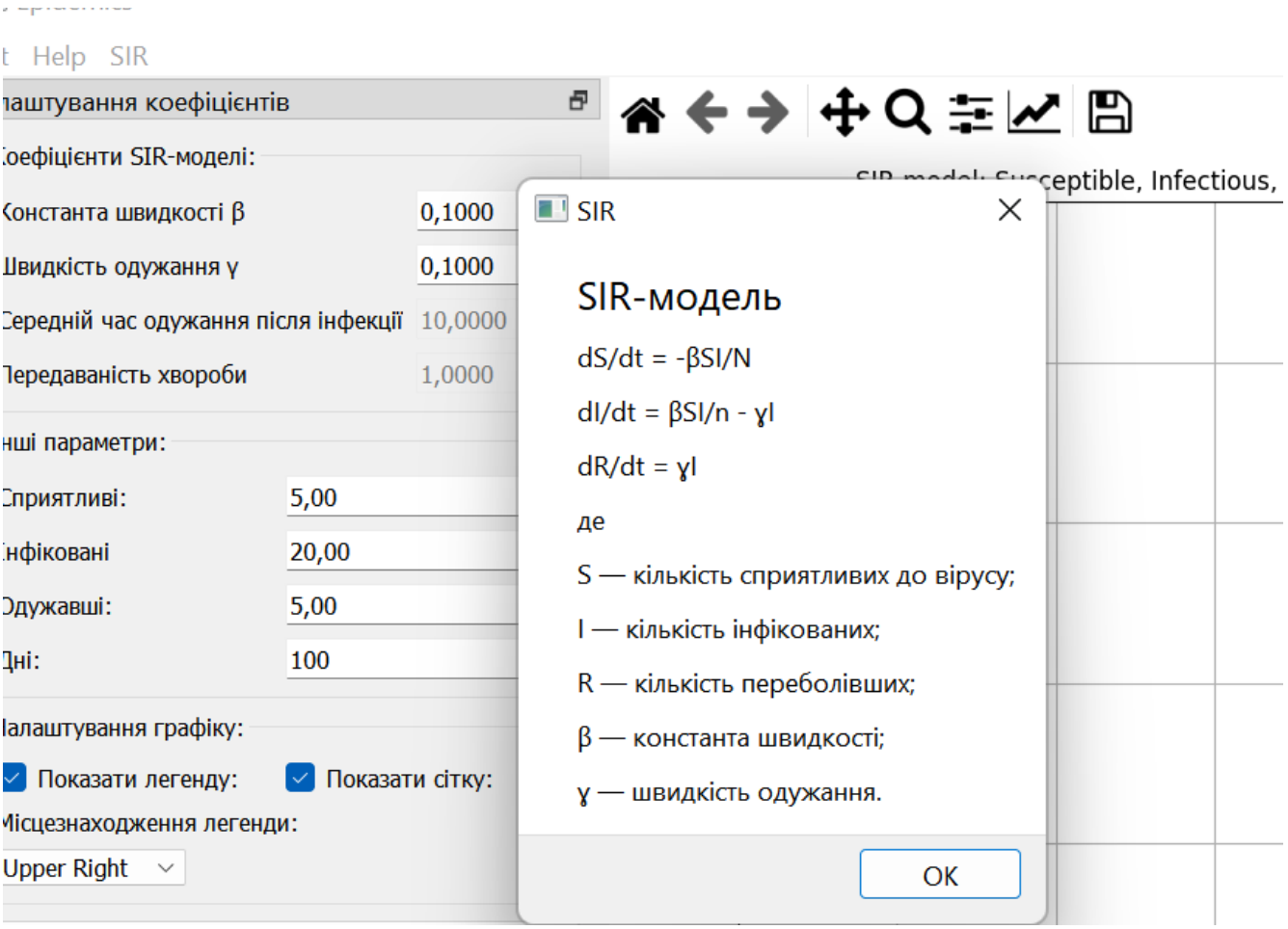

Рис. 3.3. — Опис SIR-моделі

Зараз буде змодельовано ситуацію з наступними параметрами: S = 20, I = 1, R = 0, оберемо швидкість одужання 0.1 та швидкість передавання хвороби 0.5. Хочемо побачити динаміку розвитку хвороби протягом ста днів (рис. 3.4):

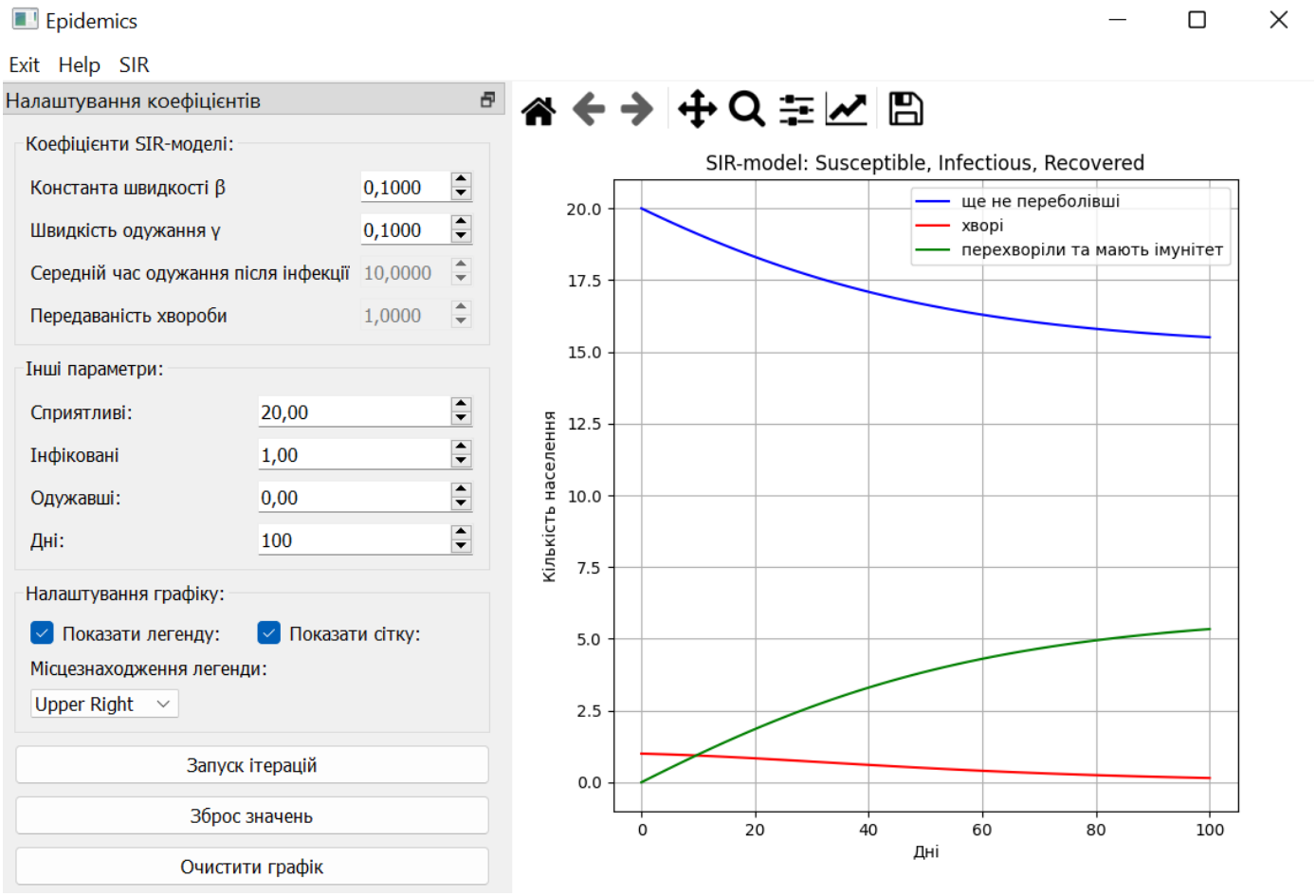

Рис. 3.4. — SIR-модель з початковими коефіцієнтами

Як можна побачити з графіку, йде динаміка спаду ще не хворівших людей та зростання — перехворівших. Давайте змоделюємо не для ста днів, а для 1250, щоб побачити повну картину (Рис. 3.5):

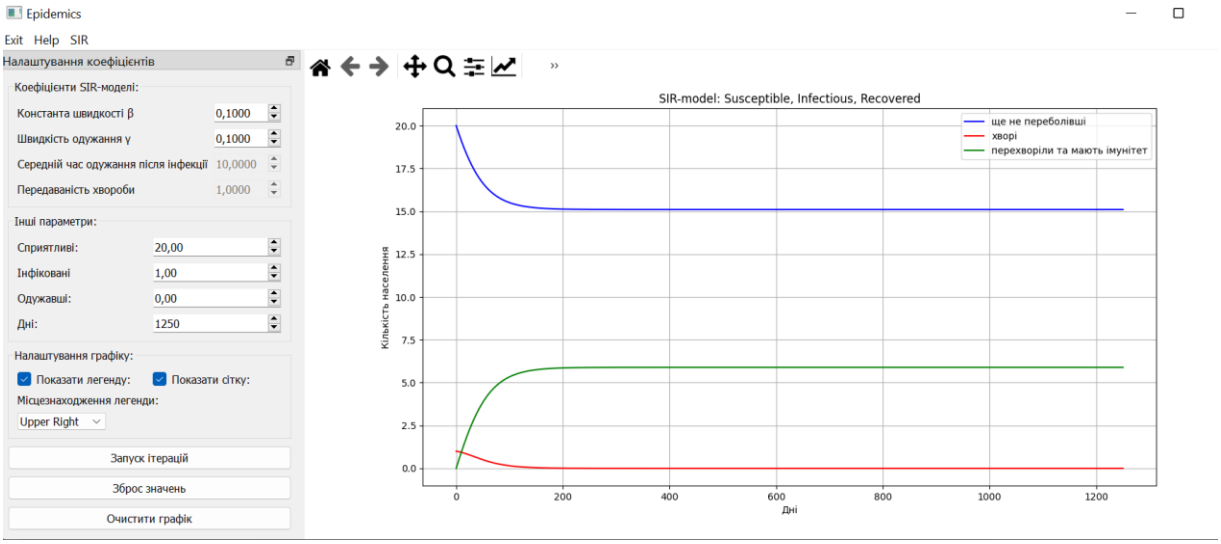

Рис. 3.5. — SIR-модель з коефіцієнтами по 0.1 за 1250 днів

Тобто, як можна побачити, при наступних коефіцієнтах у нас хвороба зупиняється на 6 перехворівших та 15 не перехворівших. Давайте змоделюємо ситуацію, при яких всі перехворіють. Для цього треба підвищити передаваємість хвороби, наприклад, візьмемо $\beta = 0.6$. Як можна побачити з Рис. 3.6, навіть 100 днів вистачить, щоб усі 21 людина перехворіли та вилікувалися при вищезазначеній швидкості передачі хвороби:

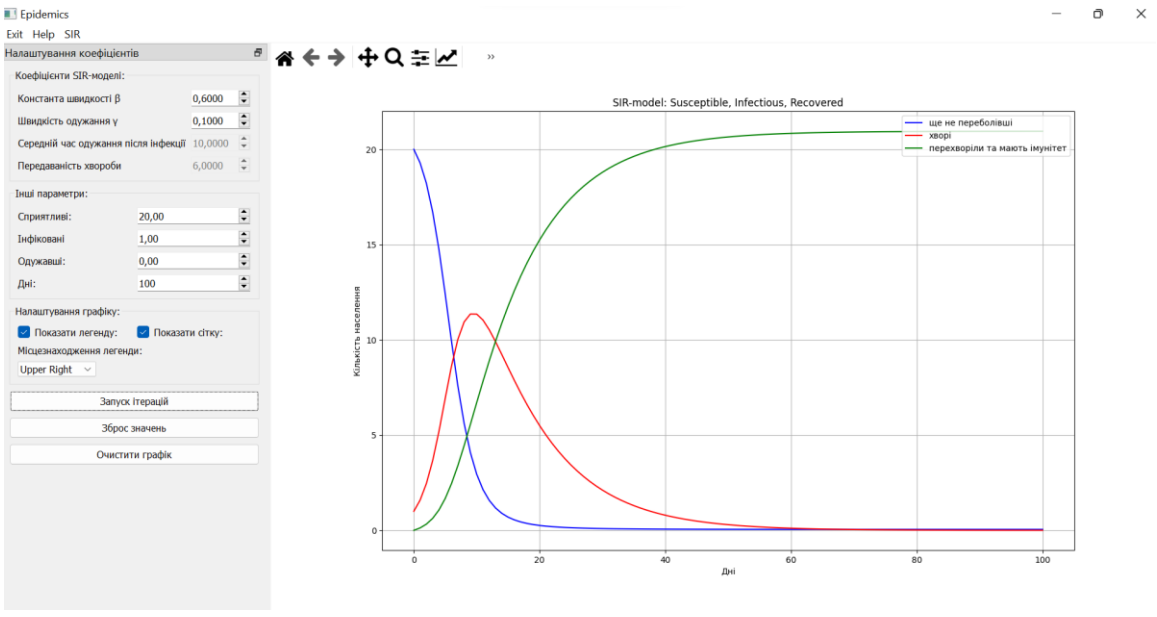

Рис. 3.6. — SIR-модель при β = 0.6

Для стандартної SEIR-моделі графік з параметрами S = 25, I = 20, E = 10, $\beta = 0.1$, $\gamma = 0.1$, $\alpha = 0.1$ виглядає наступним чином (Рис. 3.7):

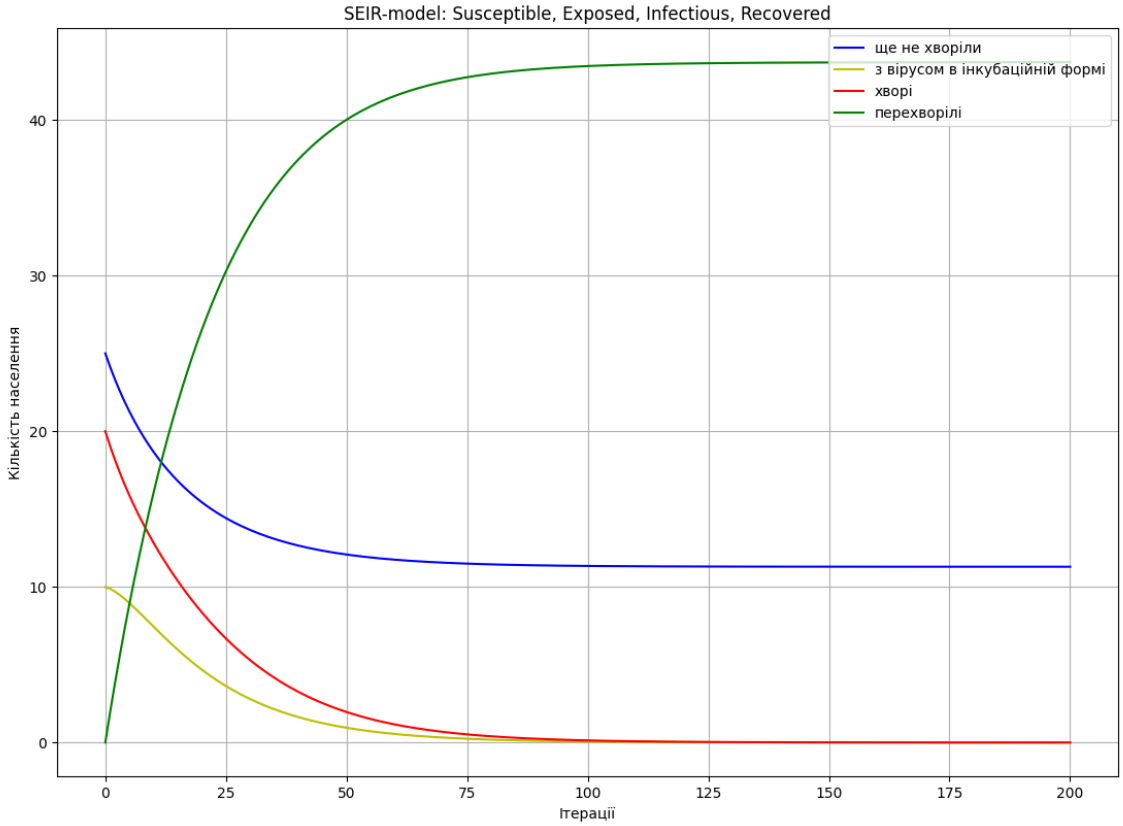

Рис. 3.7. — SEIR-модель

Тобто, приблизно через 100 днів 43 особини з популяції перехворіють та отримають імунітет, а 12 залишаться не хворівшими. Якщо треба, щоб перехворіла уся популяція, треба знову-таки підвищити швидкість передачі хвороби (Рис. 3.8):

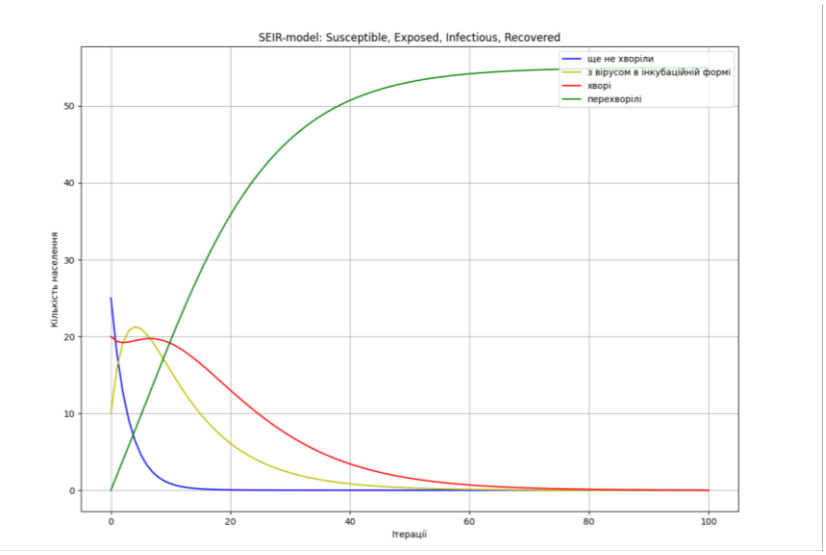

Рис. 3.8. — SEIR-модель з іншими параметрами

Тепер змоделюємо модифіковану SEIR-модель (3.2), у якої латентні захворівші теж можуть розповсюджувати вірус. Якщо зробимо такі самі коефіцієнти, як для стандартної SEIR-моделі ($S = 25, I = 20, E = 10, \beta = 0.1, \gamma = 0.1, \alpha = 0.1$), але зробимо ступінь заразності $\Theta = 0.3$, то побачимо, що, на відміну від Рис. 3.5, кількість інфікованих зменшується швидше (тобто люди хворіють і одужують швидше, бо вони можуть захворіти від латентних носіїв теж), Рис. 3.8:

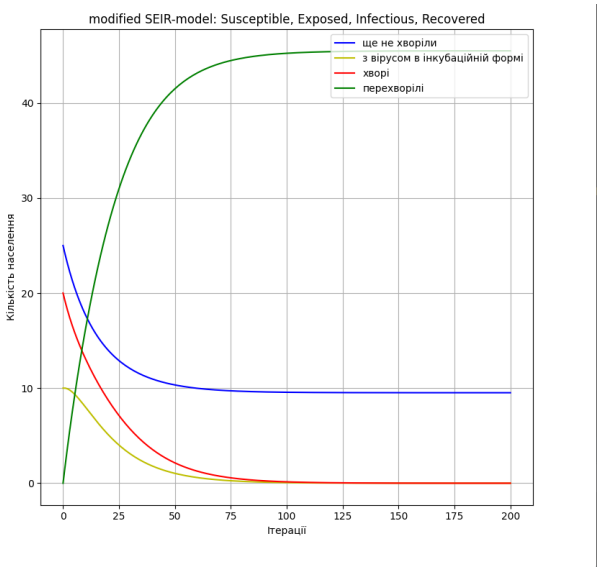

Рис. 3.9. — Модифікована SEIR-модель

Тепер розглянемо другу модифікацію SEIR-моделі, у якій є запобіжні заходи для передачі захворювання через взаємодію населення (u). При $u = 0$ (тобто коли немає запобіжних заходів від охорони здоров'я таких як вакцинація чи не допущення великих скупчень людей) і таких самих коефіцієнтах будемо мати SEIR-модель (як на рис. 3.4) (Рис. 3.10):

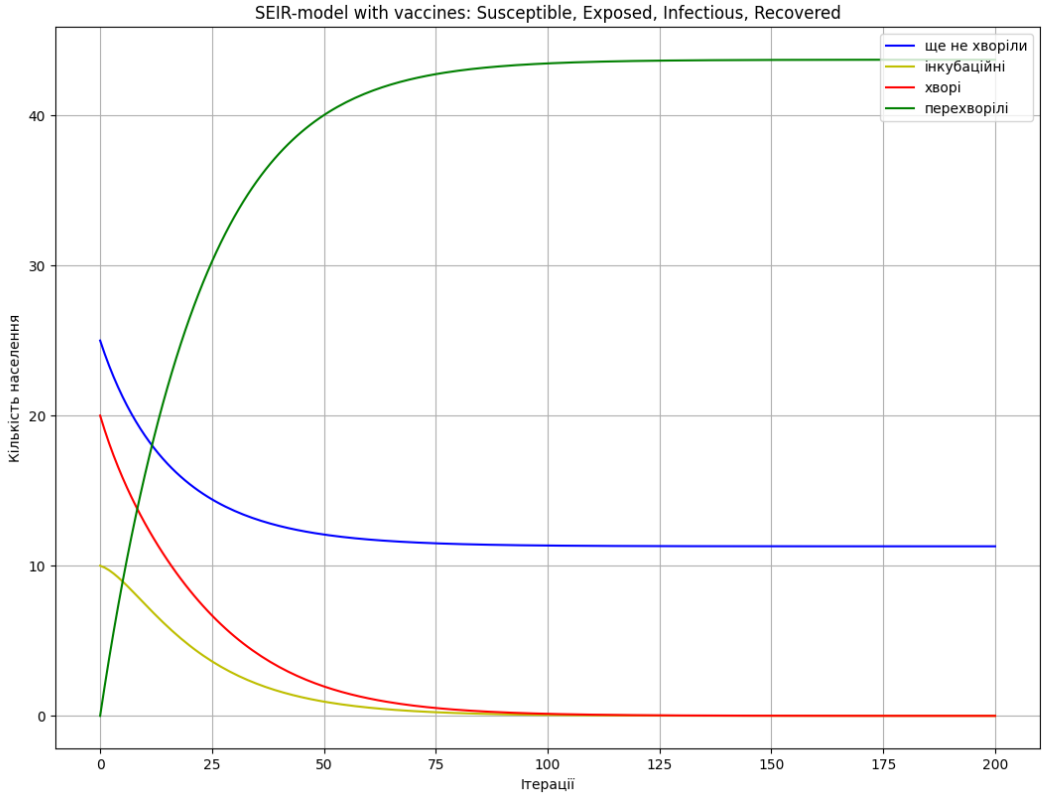

Рис. 3.10. — Модифікована SEIR-модель з урахуванням запобіжних заходів з u =0

Якщо ж поставимо $u = 1$, то перехворівших буде менше, буде більш різкий спад кількості хворих та людей з хворобою у інкубаційному періоді (Рис. 3.11):

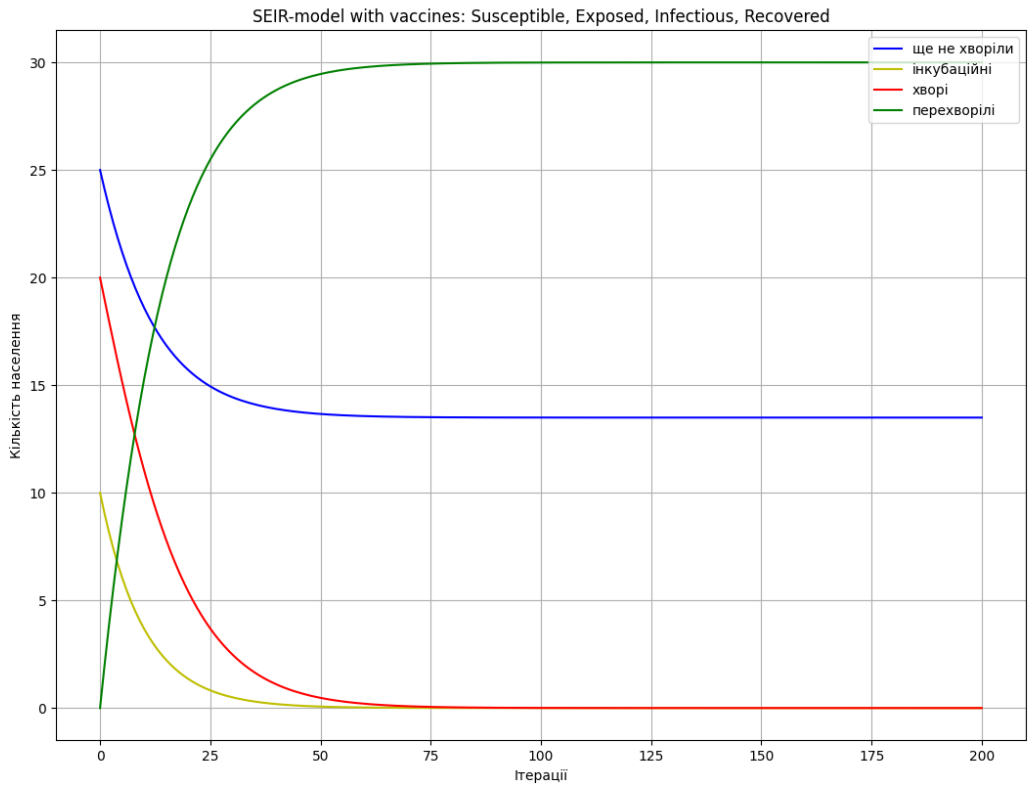

Рис. 3.11. — Модифікована SEIR-модель з урахуванням запобіжних заходів з u =1

3.3 Модуляція ситуації для України

Змоделюємо ситуацію з COVID-19 в Україні. Для цього треба розрахувати відповідні коефіцієнти, щоб підставити їх. Для цього скористуємося інформацією за 2021 рік.

Кількість українців за 2021 (без урахування окупованих територій) становить 41 млн 167 тисяч людей [48]. Смертність за 2021 рік становить 714 263 людей, з яких COVID-19 становить 86 015 випадків [49]. Народжуваність за 2021 рік становить 271 984 дитини [50].

Кількість усіх смертей за день у середньому становить 714 263/365 = 1 956.88 людей. Тобто, за день від загальної кількості населення помирають 1 956.88/41 167 300 = 0.0000475349. У цю цифру також входить смертність від COVID-19. Смертність від COVID-19 становить (86 015/365)/41 167 300 = 0.0000057244.

На жаль, статистики смертності вакцинованих та невакцинованих саме по Україні не було знайдено. Проте можна розрахувати цей коефіцієнт самостійно, якщо врахувати, що дані, що були зазначені в інтерв'ю з професоркою Школи громадської охорони здоров'я Леанни Вен [58], правдиві: що вакциновані люди мають у шість разів менше шансу заразитися, ніж у невакцинованих, і мають ймовірність померти від коронавірусу у 11 разів менше. У цьому випадку смертність вакцинованих за день становить 0.0000004770, а невакцинованих 0.0000052474. Відповідно, маємо Θ_{unvac} = 0.0000052474, Θ_{vac} = 0.0000004770.

Відповідно, смертність за день не від COVID-19 становить 0.0000475349 – 0.0000057244 = 0.0000418105, тобто μ = 0.0000418105.

Народжуваність за день становить (271 984 /365)/41 167 300 = 0.0000181008, тобто, l = 0.0000181008.

Так як для моделювання треба і вакцинована, і невакцинована популяції, треба навести дані стосовно цього. Станом на 2021 рік в Україні вакциновано двома вакцинами 15 201 112 людей [51]. Тобто, 15 201 112/41 167 300 = 0.3692521006. Відповідно, якщо треба хочеться змоделювати популяцію, де всього 100 людей, а по одному з вакцинованих та невакцинованих хворіють, 36.1867058563 будуть ще не хворівшими вакцинованими, а 61.8132941437 — ще не хворівшими невакцинованими.

Інкубаційний період — це кількість днів між моментом зараження чимось і моментом появи симптомів. Для розрахунку коефіцієнту, що відповідає за швидкість переходу вірусу із латентного періоду у повноцінний інфікований, скористаємося формулою (3.4):

$$\alpha = \frac{1}{T_{inc}}, \qquad (3.4)$$

де α — швидкість переходу захворювання із латентної фази у відкриту;

T_{inc} — середній час інкубаційного періоду вірусу.

Віруси постійно змінюються, що іноді призводить до появи нових штамів. Різні штами COVID-19 можуть мати різні інкубаційні періоди. У середньому симптоми з'явилися у новоінфікованої людини приблизно через 5.6 дня після контакту [52]. Тобто, T_{inc} = 5.6, відповідно, α = 1/5.6 = 0.17858. Відповідно до досліджень, інкубаційний період у вакцинованих та невакцинованих складає однакову кількість днів [53], тобто $\alpha_{vac} = \alpha_{unvac}$ = 0.17858.

Одні дослідження показали, що організму може знадобитися 2 тижні, щоб подолати легку хворобу, або до 6 тижнів у важких або критичних випадках [54]. Інші джерела кажуть [55], що одужання зазвичай займає один-два тижні. Отже, візьмемо час одужання за два тижні у середньому.

Розрахуємо γ відповідно до формули розрахунку коефіцієнту швидкості одужання (3.5):

$$\gamma = \frac{1}{T_{rec}} \qquad (3.5)$$

де γ — коефіцієнт одужання інфікованих людей від вірусу;

T_{rec} — середній час одужання.

Тобто, T_{rec} = 14 днів. Відповідно, $\gamma = \gamma_{unvac}$ = 1/14 = 0.0714.

У випадку, коли населення поділяється на вакцинованих та невакцинованих, час одужання буде різний. Різні джерела кажуть різне стосовно

цих періодів: одні опити показують, загальна тривалість хвороби була на шість-сім днів коротшою, ніж у невакцинованих людей [56]. Інше дослідження 2021 року, що було проведено Центрами з контролю та профілактики захворювань, зазначено, що вакциновані учасники проводили в ліжку в середньому від двох до шести днів хвороби менше, ніж невакциновані [57]. Візьмемо в середньому, що вакциновані хворіють на шість днів менше, тобто $\gamma_{vac} = 1/8 = 0.125$.

Розрахуємо тепер ймовірності передачі захворювання. Для розрахування даних для вакцинованих людей, знадобиться статистика по ефективності вакцини (візьмемо вакцину Pfizer). Відповідно до даних [60], Pfizer має захист від легкої форми COVID-19 95%. Тобто, ймовірність захворіти, якщо ти вакцинований, $p_{\text{заразитися, якщо ти вакцинований}} = 0.05$.

Також треба зазначити, що вакциновані люди менш схильні до передачі хвороби, навіть якщо вони інфікуються. На листопадовій прес-конференції Тедрос Гебреєсус, генеральний директор ВООЗ, заявив, що вакцини на 60 відсотків захищають від поширення вірусу до появи дельта-варіанту [61][64]. Тобто $p_{\text{передати хворобу, якщо ти вакцинований}} = 0.4$

Відповідно до того ж дослідження [61][62], у вакцинованих людей у десять разів менше шансів заразитися [63], а ймовірність заразити мене вдвічі менша, судячи з наведених вище цифр.

Тобто, $p_{\text{заразитися, якщо ти невакцинований}} = p_{\text{захворіти, якщо ти вакцинований}} \times 10 = 0.5$.

А $p_{\text{передати хворобу, якщо ти невакцинований}} = 0.4 \times 2 = 0.8$.

Тепер можемо розрахувати коефіцієнти передачі захворювання.

$\beta_{uu} = p_{\text{передати хворобу, якщо ти невакцинований}} \times p_{\text{заразитися, якщо ти невакцинований}} = 0.8 \times 0.5 = 0.4$

$\beta_{uv} = p_{\text{передати хворобу, якщо ти невакцинований}} \times p_{\text{заразитися, якщо ти вакцинований}} = 0.8 \times 0.05 = 0.04$

$\beta_{vu} = p_{\text{передати хворобу, якщо ти вакцинований}} \times p_{\text{заразитися, якщо ти невакцинований}} = 0.4 \times 0.5 = 0.2$

$\beta_{vv} = p_{\text{передати хворобу, якщо ти вакцинований}} \times p_{\text{заразитися, якщо ти вакцинований}} = 0.4 \times 0.05 = 0.02$

Змоделюємо з цими параметрами модель. На рис. 3.12 можна побачити результати для ста днів:

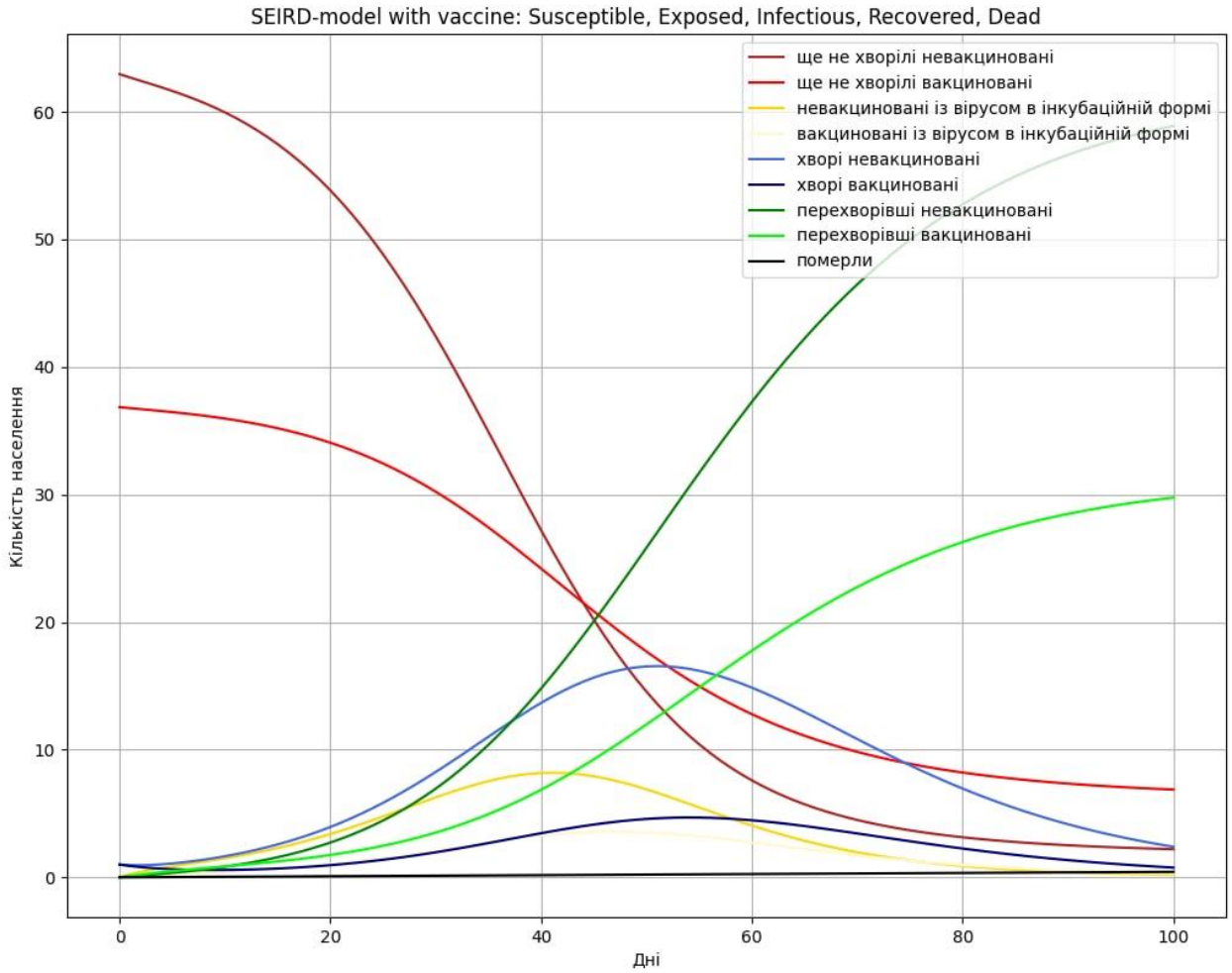

Рис. 3.12. — Модифікована SEIRD-модель для 100 днів

З Рис. 3.12 можна побачити, як кількість ще не хворівших людей зменшується і одночасно кількість перехворівших зростає. Кількість перехворівших невакцинованих зростає швидше, і з урахуванням вакцинаці [57] це виглядає логічно. Кількість померших відносно повільно зростає. Кількість не хворівших невакцинованих спадає швидше, бо у них ймовірність підхватити хворобу вище, у той час як вакциновані хворіють повільніше.

Якщо ж розглянути набагато більший часовий проміжок (наприклад, 8 000 днів), то з рис. 3.13 можна побачити, як зростає кількість померших, і зменшується — усієї іншої популяції. Кількість перехворівших людей також зменшується, бо модель (3.3) враховує також природну смертність, у яку у даному випадку не входить смертність від COVID-19.

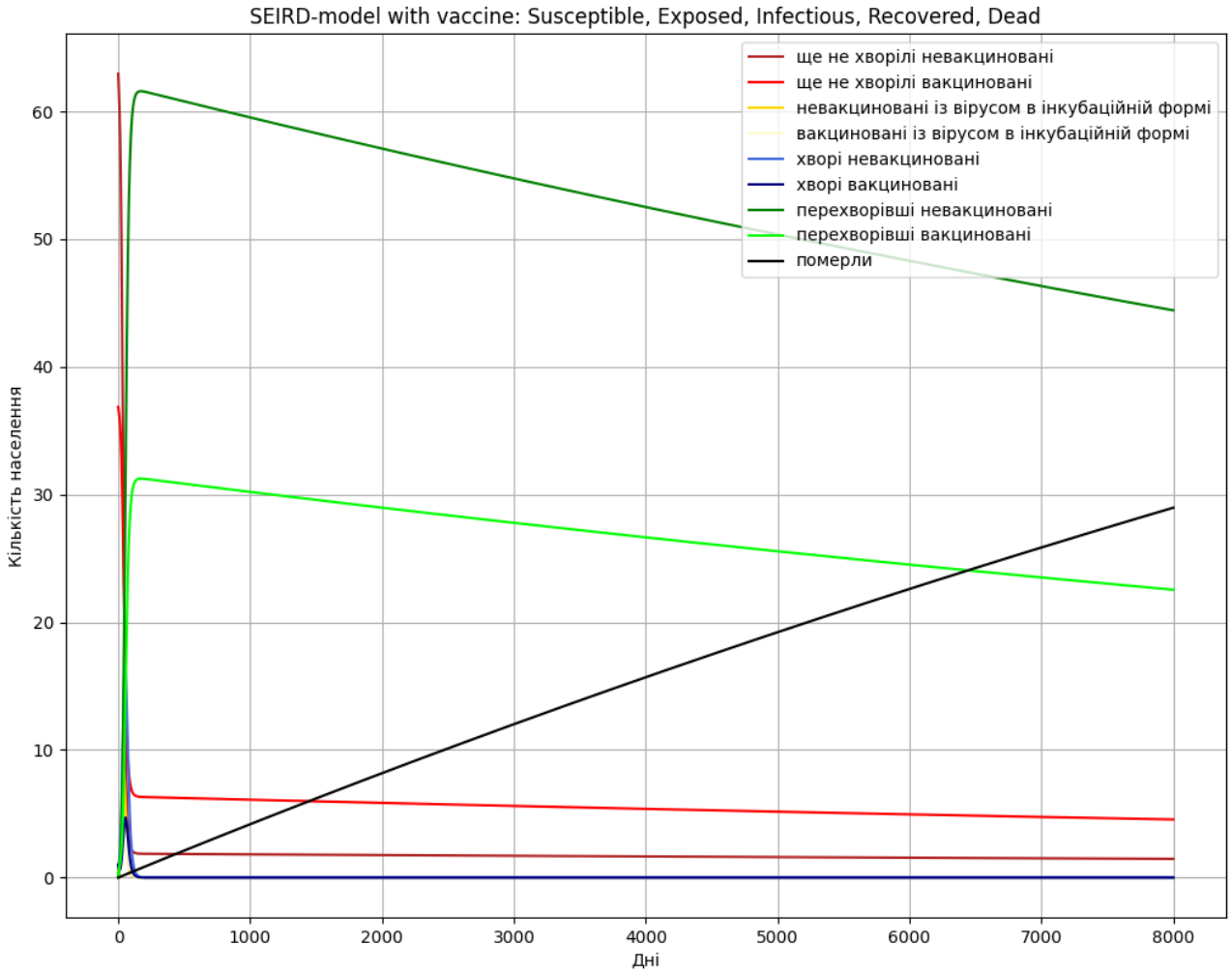

Рис. 3.13. — Модифікована SEIRD-модель для 8000 днів

Така ситуація із народжуваністю та смертністю виходить тому, що в Україні народжуваність менше, ніж смертність [65]. Якщо ж збільшити

коефіцієнт народжуваності, щоб він перевищив смертність, то ситуація буде виглядати інакше (Народжуваність l = 0,0500181008) (Рис. 3.14):

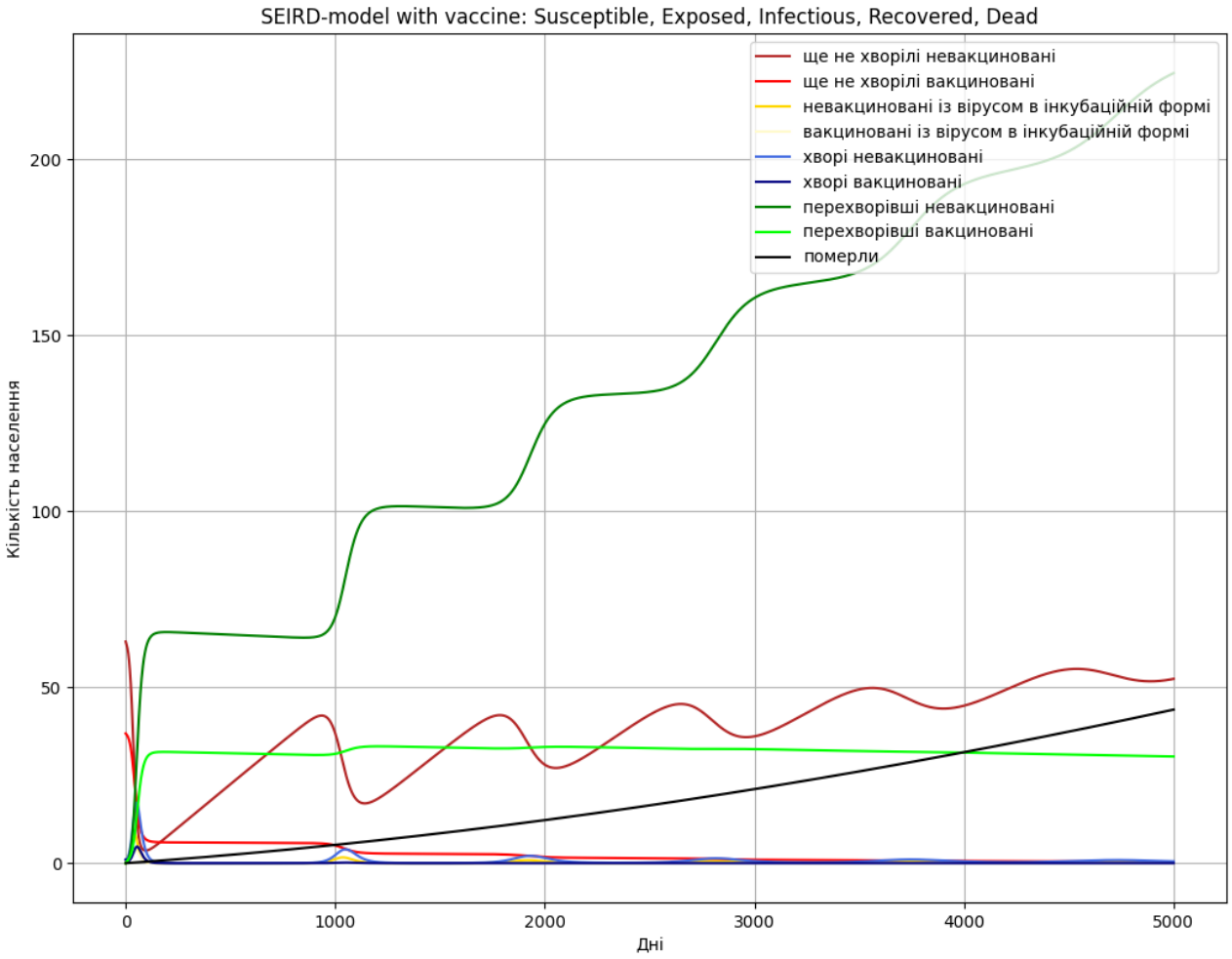

Рис. 3.14. — Модифікована SEIRD-модель для 5000 днів з народжуваністю l = 0,0500181008

Зростає лише кількість ще не перехворівших невакцинованих, бо дана модель не передбачає вакцинацію і відповідно перехід від невакцинованих до вакцинованих. Уся популяція, що народжується, за замовчуванням є невакцинованою. З Рис. 3.14 ще можна побачити, як з періодами йде процес захворювання: на тисячному і двохтисячному дні можна побачити хвилі з

хворих невакцинованих, і відповідно періодами зменшуються ще не перехворівші невакциновані.

3.4 Висновки до розділу 3

У розділі 3 були наведені модифіковані SEIR та SEIRD-модель, були зазначена їхня важливість. Було обґрунтовано вибір мови програмування та засобів для розробки програмного забезпечення, був наведений інтерфейс програми, що була написана в рамках цієї роботи, та описано її функціонал.

Були змодельовані та наведені графіки з різними параметрами для SIR, SEIR, двох модифікованих SEIR-моделі та модифікованої SEIRD-моделі. Були описані рівняння модифікованих моделей. Зазначено важливість змодельованої у рамках цієї роботи SEIRD-моделі: вона дозволяє моделювати ситуацію, коли уся популяція ділиться на вакцинованих та невакцинованих. Модифікована SEIRD-модель також має коефіцієнти, що відповідають за народжуваність та смертність, саме тому вона може відображати реалістичну картину.

Для модифікованої SEIRD-моделі були прораховані усі параметри з реальних статистичних даних України, щоб мати можливість змоделювати ситуацію, максимально можливо близьку до реальності. Усе було проілюстровано графіками.

РОЗДІЛ 4 РОЗРОБКА СТАРТАП-ПРОЕКТУ

4.1 Вступ та опис стартап-проекту

Стартапи — це зерна уяви та пристрасті, які проростають у вигляді побічної метушні або значних витрат часу та грошей.

Крім того, як саджанцю потрібне сонячне світло, вода та поживні речовини з ґрунту, стартапу також потрібен певний набір ресурсів для росту. Ключ полягає в забезпеченні оптимальних ресурсів, а потім у їх блискучому використанні. Важливим способом досягнення успіху є бездоганне управління стартап-проектом.

Менеджмент — чим млявіше звучить це слово, тим більше він може викликати занепокоєння у будь-якого керівника чи генерального директора. Особливо коли справа доходить до стартапів, квінтесенція процесу управління стає життєво важливою.

Сучасні системи прогнозування не враховують комплексно кількісні та якісні фактори, що впливають на зміну курсів акцій або враховують у векторному вигляді, який обмежує можливості представлення вхідних даних до моделей машинного навчання. Такі системи прогнозування зазвичай мають точність 53-58% вірного прогнозу. Цієї точності замало для того, щоб використовувати такі системи як повноцінний інструмент для економічного аналізу та прогнозування. Отже, ідеєю стартап проекту є створення і розповсюдження системи прогнозування прогнозування кредитоспроможності юридичних осіб із більшою ніж у конкурентів точністю прогнозування тренду (табл. 4.1).

Таблиця 4.1. — Опис ідеї стартап-проекту

Зміст ідеї	**Вигоди для користувача**	**Напрямки застосування**
Розроблену систему для моделювання можна застосовувати для аналізу та прогнозування поведінки епідемій типу COVID-19 та їй аналогічних.	Аналоги (на українському ринку) відсутні. Застосування математичного апарату з метою покращення якості вибору клієнта та збільшення прибутків. Можливість прогнозування поведінки вірусів та відповідно впровадження запобіжних заходів, що дозволить зберегти людські життя та обмежити поширення.	Можна застосовувати для перевірки успішності різних вакцин (використовуючи знання про успішність вакцини). Можна застосовувати для моделювання епідемій та відповідно розуміння, як буде відбуватися подальше поширення. Урахування багатьох факторів дозволить прорахувати декілька ситуацій, щоб отримати повну картину.

Тепер можна провести аналіз потенційних техніко-економічних переваг ідеї у порівнянні з конкурентами.

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

а) визначення переліку техніко-економічних властивостей та характеристик ідеї;

б) визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться

збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до зазначеного вище переліку;

в) проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі);

г) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (табл. 4.2)

Таблиця 4.2. — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№п/п	Технічноекономічні характеристики ідеї	(потенційні) товари/концепції конкурентів		Weak (слабка сторона)	Neutral (нейтральна сторона)	Strength (сильна сторона)
		Мій проєкт	Matlab			
1.	Вартість програмного забезпечення	Низька	Середня			+
2.	Доступність	Середня	Середня		+	
3.	Кроссплатформність	Так	Так		+	

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит проекту

Метою технічного аудиту є визначення переліку технологій, за допомогою який реалізована система і їх аналіз. Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 4.3):

1. за допомогою якої технологією розроблена система згідно ідеї проекту;
2. чи існують у відкритому доступі ці технології, чи їх потрібно додатково розробляти або купувати;
3. чи має доступ розробник до описаних технологій.

Таблиця 4.3. — Технологічна здійсненність ідеї проекту

№п/п	**Ідея проекту**	**Технології реалізації**	**Наявність технологій**	**Доступність технологій**
1.	Створення програмного продукту для моделювання епідемій.	Використання мови програмування C#	Необхідність доопрацювання.	Не доступні
2.		Використання мови програмування Python	Наявна	Доступні
3.		Використання мови програмування Java	Необхідність доопрацювання	Не доступні

Продовження таблиці 4.3.

Для реалізації програмного продукту було обрано мову програмування Python.

За результатами аналізу можна зробити висновок щодо можливості технологічної реалізації проекту. Технологічним шляхом реалізації проекту було обрано мову програмування Python через її доступність та безкоштовність.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Метою аналізу ринкових можливостей є виявлення та дослідження обмежень, можливостей та розрахунок конкретних показників реалізації розробленої системи прогнозування фінансових ринків як стартап-проекту.

Спочатку було проведено аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4):

Таблиця 4.4. — Попередня характеристика потенційного ринку стартап проекту

№ п/п	**Показники стану ринку(найменування)**	**Характеристика**
1.	Кількість конкурентів на ринку, од	10
2.	Загальний обсяг продаж, грн/ум.од	100
3.	Динаміка ринку	Повільний приріст

Продовження таблиці 4.4.

4.	Наявність обмежень для входу (вказати характер обмежень)	Нема
5.	Специфічні вимоги до стандартизації та сертифікації	Нема
6.	Середнє значення рентабельності в галузі(або по ринку), %	50%

Проаналізувавши результати, можна зробити висновок, що проект є придатним до інвестицій, оскільки вкладені кошти будуть повернені за 2 роки.

За результатами порівняння, що було наведене у таблиці 4.4, було зроблено висновок, що ринок є придатним для розповсюдження системи як стартап-проекту. Після цього були досліджені потенційні категорії клієнтів, їх особливості та затверджено перелік вимог до системи для кожної категорії клієнтів (табл. 4.5):

Таблиця 4.5. — Характеристика потенційних клієнтів стартап-проект

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія та потенційні клієнти**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги до споживачів товару**

Продовження таблиці 4.5.

	COVID-19	Медичні компанії, компанії, що займаються вакцинацією	Цільові групи клієнтів мають різні доходи, і потенційний прибуток від цього ПЗ буде різний	Оптимізовані та автоматизовані алгоритми, кроссплатформенність

Після дослідження потенційних категорій клієнтів було проведено дослідження ринкового середовища: складено таблиці факторів, що сприяють реалізації розробленої системи як стартап-проекту, та факторів, що йому заважають (табл. 4.6, 4.7).

Таблиця 4.6. — Фактори загроз

№п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1.	Конкуренція	Вихід на ринок систем з кращими Характеристиками та моделлю надання послуг	Вийти на ринок, зосередивши увагу на переваги власної системи. Покращення якісних характеристик системи прогнозування. Обрати цільову аудиторію
2.	Зміна потреб користувачів	Клієнту потрібна буде система з більшою точністю прогнозування і новими функціями	Передбачити можливість розширення системи та підвищення точності прогнозування

Таблиця 4.7. — Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Конкуренція	Відсутність аналогів на українському ринку для вітчизняного користувача	Адаптація системи до особливостей потреб на українському ринку
2	Поява альтернативних методів моделювання	Нові методи моделювання, більш легкі в освоєні праці	Розширення можливостей, максимальне спрощення

Аналіз пропозиції та загальні риси конкуренції на ринку можна побачити у таблиці 4.8:

Таблиця 4.8. — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому виражена дана характеристика**	**Вплив на діяльність компанії (можливі дії для підвищення конкурентоспроможності)**
1.Вказати тип конкуренції - олігополія	На ринку присутні декілька постачальників-конкурентів, але їх товар дещо відрізняється від нашої системи.	Підтримка якості системи, безперервний розвиток, покращення, вдосконалення, оновлення та підтримка.
2. За рівнем конкурентної боротьби - міжнародний	Компанії-конкуренти з інших країн	Створити основу системи прогнозування таким чином, щоб можна було легко локалізувати її для використання у інших країнах.
3.Загалузевою ознакою - внутрішньогалузева	Система може застосовуватися в одній галузі, але її різних сферах.	Постійне вдосконалення системи прогнозування, що не має прив’язки до сфери, але має до галузі
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між видами систем прогнозування, їх особливостями.	Створити систему прогнозування, враховуючи недоліки конкурентів

Продовження таблиці 4.8.

5. За характером конкурентних переваг - цінова	Покращення процесу створення програмного продукту, мінімізація витрат на оновлення, застосування безперервної інтеграції	Використання відкритих та дешевших технологій для побудови системи в порівнянні з системами-конкурентами, але тільки якщо ці технології відповідають необхідним критеріям якості.
6. За інтенсивністю - не марочна	Бренд присутній, але його роль незначна	Реклама, участь у конференціях, семінарах, виставках.

Таблиця 4.9. — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
	Навести перелік безпосередніх конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постачальників	Визначити фактори сили клієнтів	Фактори загроз з боку товарів-замінників

Продовження таблиці 4.9.

	Інші системи для побудови хворих популяцій	Наявність вже існуючих рішень	-	Якість системи та її підтримка оновлення	Більш відомий розробник, що підтримує свою систему
Висновки:	На даний момент немає конкурентів на українському ринку	Вихід на український ринок буде легшим через відсутність конкуренції	Постачальники відсутні	Вимоги клієнтів такі, як зручний інтерфейс, якість Програмного продукту	Випустити систему, що буду не гірше, ніж у конкурента, але мати кращу точність розрахунків

За результатами порівняльного дослідження що наведене у табл. 4.9 було зроблено висновок про доцільність виходу на український та міжнародні ринки.

На основі аналізу ситуації на ринках, проведеного в табл. 4.9, а також із урахуванням характеристик розробленої системи, вимог потенційних клієнтів до товару (табл. 4.5) та особливостей ринкового середовища (таблиці 4.6, 4.7) досліджується та визначається перелік факторів конкурентоспроможності розробленої системи прогнозування фінансових ринків.

За визначеними факторами конкурентоспроможності проведено аналіз сильних та слабких сторін стартап-проекту (табл. 4.10).

Таблиця 4.10. — Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1.	Ціна	Безкоштовність Python.
2.	Кросплатформність	Загальнодоступність, зручність та точність
3.	Орієнтованість на кінцевого споживача	Періодичні оновлення програмного продукту

Таблиця 4.11. — Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг систем-конкурентів у порівнянні з розробленою системою прогнозування						
			-3	-2	-1	0	+1	+2	+3
1	Ціна	15					+		
2	Кросплатформність	20			+				
3	Орієнтованість на кінцевого споживача	7					+		

Кінцевим кроком маркетингового дослідження можливостей реалізації системи прогнозування фінансових ринків як стартап-проекту є побудова SWOT-аналізу (матриці сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (таблиця 4.12) на основі описаних конкурентних та маркетингових загроз та можливостей, та сильних і слабких сторін (таблиця 4.11). Список маркетингових загроз та можливостей було складено на основі дослідження факторів загроз та факторів можливостей ринкової ситуації. Маркетингові загрози та можливості є наслідками (прогнозованими результатами) впливу ринкових факторів.

Таблиця 4.12. — SWOT-аналіз стартап-проекту

Сильні сторони: Ціна, орієнтованість на кінцевого споживача, кросплатформеність	Слабкі сторони: Складність розповсюджувати продукцію за кордоном.
Можливості: Відсутність конкуренції на українському ринку	Загрози: Зміна основних потреб клієнтів, при відсутності конкуренції необхідно підтримувати інтерес аудиторії до продукту

За результатами SWOT-аналізу було сформовано альтернативи ринкової стратегії (перелік заходів) для виведення розробленої системи як стартап-проекту на український та міжнародні ринки та розрахований оптимальний час їх ринкової реалізації з урахуванням потенційних розробок конкурентів, що можуть бути виведені на ринок (див. таблицю 4.9 аналіз потенційних конкурентів). Запропоновані альтернативи були проаналізовані з точки зору часу реалізації та ймовірності отримання ресурсів (таблиця 4.13).

Таблиця 4.13. — Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний Комплекс заходів) ринкової Поведінки	Приблизна ймовірність отримання ресурсів	Приблизні строки реалізації
1	Безкоштовне розповсюдження Обмеженої версії створеного програмного продукту	85%	12 місяців
2	Створення Програмної системи з подальшим платним розповсюдженням (продаж платної ліцензії)	45%	12 місяців

Після аналізу було обрано альтернативу №2.

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим етапом передбачає визначення стратегії охоплення ринку: дослідження цільових груп потенційних клієнтів, які приведенні в таблиці 4.14.

Таблиця 4.14. — Вибір цільових груп потенційних споживачів

№п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота входу у сегмент**
1.	Компанії що займаються моделюванням епідемій	Готові	Необхідно	Висока	Середня

Визначена цільова група клієнтів: Компанії, що займаються моделюванням епідемій. Дослідивши потенційні групи клієнтів було визначено цільові категорію, для яких буде пропонуватися розроблена система, та визначено стратегію охоплення ринку — стратегію диференційованого маркетингу.

Для роботи в обраних сегментах ринку сформовано базову стратегію розвитку (таблиця 4.15).

Таблиця 4.15. — Визначення базової стратегії розвитку

№ п/п	**Обрана альтернатива розвитку стартап-проекту**	**Обрана альтернатива розвитку стартап-проекту**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку**
1		Визначити потреби сучасного ринку для кожної з груп.	Цінова політика, універсальність продукту.	Стратегія Диференціації

Наступним кроком обрано стратегію конкурентної поведінки (таблиця 4.16).

Таблиця 4.16. — Визначення базової стратегії конкурентної поведінки

№ п/п	**Чи є проект першою подібною системою на ринку?**	**Чи буде розробник системи шукати нових споживачів, або забирати існуючих у конкурентів?**	**Чи буде розробник копіювати властивості товару конкурента**	**Стратегія конкурентної поведінки**
1	Ні	Шукати Нових	Ні	Заняття конкурентної ніші

За результатами аналізу вимог клієнтів визначених категорій до розробника стартап-проекту та до продукту (див. таблицю 4.5), а також в залежності від обраної базової стратегії розвитку (таблиця 4.15) та стратегії конкурентної поведінки (таблиця 4.6) розроблено стратегію позиціонування (таблиця 4.17), що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17. — Визначення стратегії позиціонування

№ п/п	**Вимоги до системи з боку потенційних клієнтів**	**Основна стратегія розвитку**	**Ключові конкурентоспроможні позиції власного стартап-проекту**	**Вибір основних асоціацій**
1	Проста побудова моделі прогнозування, довгий час навчання моделі, проте вища точність прогнозу та інтуїтивно зрозумілий інтерфейс, докладне керівництво користувача	Стратегія диференціації	Позиція на основі порівняння фірми з товарами конкурентів; Відмінні особливості споживача	Економія часу; Зручність застосування; Практичність

За результатами дослідження була сформована система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії українському та міжнародному ринках.

4.5 Розроблення маркетингової програми стартап-проекту

Визначено маркетингову концепцію системи використання моделі машинного навчання, яку буде купувати споживач. Основна концепція продукту — письмовий опис фізичних та програмних характеристик системи, які сприймаються споживачем, і набору вигід, які він обіцяє певній групі споживачів.

За результатами дослідження було сформовано трирівневу маркетингову модель системи: ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання:

- 1-й рівень вирішує питання щодо того, засобом вирішення якої потреби і / або проблеми буде система, яка її основна вигода;
- 2-й рівень являє рішення того, як буде реалізована система в реальному ринковому середовищі. Рівень включає в себе якість, властивості, дизайн, упаковку, ціну;
- 3-й рівень визначає додаткові послуги та переваги для клієнта системи, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 4.18).

Таблиця 4.18. — Визначення ключових переваг концепції потенційного товару

№ п/п	**Потреба**	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)**
1.	Моделювання розповсюдження епідемій	Інструментарій для моделювання	Спеціалізація, модифіковані моделі
2.	Пошук долі технічного прогресу, що впливає на ВВП	Моделі для використання	Унікальність

Таблиця 4.19. — Опис трьох рівнів моделі товару

Рівні товару	**Сутність та складові**		
I. Товар за задумом	Дослідження та моделювання епідемій		
II. Товар у реальному виконанні	Властивості / характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Програмний продукт — складна, комплексна система		
	Якість: відповідно до стандарту ISO9000 тестування.		
	Пакування — інсталятор		
	Марка: назва організації-розробника + назва товару		

Продовження таблиці 4.19.

III. Товар із підкріпленням	До продажу: гнучка оплата, доставка
	Після продажу: гарантія якості та оновлення
За рахунок чого товар буде захищено від копіювання: захист інтелектуальної власності	

Після цього визначимо цінові межи, якими необхідно керуватись при встановленні ціни на систему, яке включає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової категорії клієнтів (таблиця 4.20). Аналіз проведено експертним методом.

Таблиця 4.20. — Визначення меж встановлення ціни

№ п/п	**Приблизна вилка вартості товарів-замінників**	**Приблизна вилка вартості товарів-аналогів**	**Приблизний рівень доходів цільової групи клієнтів**	**Верхня та нижня межі вартості системи**
1	850-2000$	850-2000$	2000$+	300-800$

Після цього визначимо оптимальну систему збуту, за допомогою якої буде розповсюджуватися система як статрап-проект.(таблиця 4.21)

Таблиця 4.21. — Формування системи збуту

№ п/п	Особливості ринкової поведінки потенційних клієнтів	Функції збуту, які має виконувати постачальник системи	Глибина каналу збуту	Оптимальний канал збуту
1	Цільові клієнти – компанії, які бажають впровадити у своїй роботі сучасні засоби допоможуть прогнозувати фінансові ринкі із кращою точністю	Побудова прямих контактів із потенційними клієнтами і їх підтримка. формування попиту і стимулювання продажів.	Один (від виробника одразу споживачу)	Прямий канал збуту до клієнта, мінімізувати збутові витрати. Розвиток нових маркетингових концепцій.

Фінальною складовою маркетингової програми є визначення концепції маркетингових комунікацій.

Результатом дослідження стала робоча ринкова програма, що включає в себе концепції системи, збуту, просування та попереднє дослідження ціноутворення на продукт, базується на цінностях та потребах категорій покупців, конкурентні переваги системи, стан та динаміку маркетингового середовища, в межах якого впроваджено системи прогнозування фінансових ринків як стартап-проект, та відповідну обрану альтернативу ринкової поведінки.

Концепція маркетингових комунікацій відображена у таблиці 4.22:

Таблиця 4.22. — Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
Купівля ПЗ напряму, а не через інтернет	Фізичні носії ПЗ	Прогнозування, аналіз, точність, простота	Показати переваги ПЗ	Презентація з результатами роботи

4.6 Висновки до розділу 4

В цьому розділі було проведено аналіз розробленої прогнозування кредитоспроможності юридичних осіб методами машинного навчання у якості стартап-проекту. Варто зауважити, що проект має можливість ринкової комерціалізації, через те, що ринок потребує якісного та інноваційного продукту для прогнозування поведінки клієнта.

Результатом роботи є розроблений стартап-проект, план виходу на ринки програмного забезпечення та маркетингова стратегія. Розроблений стартап-проект доцільно застосувати при комерціалізації розробки.

Висока конкуренція зумовлює необхідність виваженого підходу до просування продукту. Для впровадження ринкової реалізації проекту була обрана стратегія, яка включає розробку програмної системи із класичною моделлю ліцензування за певну плату.

Можна сказати, що подальший розвиток проекту є доцільним, оскільки кількість потенційних користувачів системи зростає кожного року.

ВИСНОВКИ

В результаті виконання даної магістерської роботи в першому розділі були були розглянуті поняття, що пов’язані із математичним моделюванням у екології та в еволюції. Були розглянуті поняття математичної екології та основні математичні моделі: експоненційна модель (модель Мальтуса), модель зростання чисельності популяції окремого виду (модель Вергюльста з обмеженнями, логістичне рівняння), модель дискретної логістичної еволюції (модель Роберта Мея). Були розглянуті також урбаністичні моделі: Модель кількох ядер, Секторна модель, Модель концентричних кіл. До кожної моделі було наведено малюнок та опис.

У другому розділі були описані загальні поняття, пов’язані із епідеміологічним моделюванням. Найбільше уваги було приділено математичним моделям SIS, SEIR, SEIRD, SRID, які застосовуються для прогнозування перебігу епідемій. Динаміка епідеміологічного процесу описана з допомогою системи диференційних рівнянь, розв’язки яких характеризують зміну чисельності у різних підгрупах популяції.

У третьому розділі були наведені модифіковані SEIR та SEIRD-моделі, що були змодельовані в рамках цієї магістерської дисертації. Була зазначена їхня важливість. Було обґрунтовано вибір мови програмування та засобів для розробки програмного забезпечення, був наведений інтерфейс програми та описано її функціонал. Були змодельовані та наведені графіки з різними параметрами для SIR, SEIR, двох модифікованих SEIR-моделі та модифікованої SEIRD-моделі. Були описані рівняння модифікованих моделей. Зазначено важливість змодельованої у рамках цієї роботи SEIRD-моделі: вона дозволяє моделювати ситуацію, коли уся популяція ділиться на вакцинованих та невакцинованих. Модифікована SEIRD-модель також має коефіцієнти, що

відповідають за народжуваність та смертність, саме тому вона може відображати реалістичну картину. Для модифікованої SEIRD-моделі були прораховані усі параметри з реальних статистичних даних України, щоб мати можливість змоделювати ситуацію, максимально можливо близьку до реальності. Усе було проілюстровано графіками.

На базі модифікованої SEIRD-моделі було зроблено наступні висновки, що корелюються з реальними статистичним даними України, а саме:

- Кількість захворілих невакцинованих в період піку епідемії приблизно в 3.5 рази більше, ніж захворілих вакцинованих;
- Смертність в Україні перевищує народжуваність;
- Розвиток епідемії має періодичний характер з тенденцією до затухання з часом.

Отримані результати підтверджують коректність роботи розробленої моделі, яка дозволяє моделювати ситуацію розповсюдження захворювання COVID-19 з урахуванням таких додаткових факторів, як розподілення населення на вакцинованих та невакцинованих, та використання додаткових коефіцієнтів, які дозволяють також врахувати вплив народжуваності та смертності населення. Використання перелічених додаткових факторів при моделюванні дозволяє прогнозувати більш реалістичну картину як розповсюдження захворювання, так і прорахування піків епідемії та її спадів. Результати даного дослідження можуть використовуватися в якості прогнозу розвитку епідемії для організації відповідних запобіжних заходів в медичних закладах, громадських місцях та ін.

Представлені розрахунки на базі модифікованої SEIRD-моделі використовували статистичні дані з українських джерел.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Hartl, D. A Primer of Population Genetics (3rd Ed), —Sunderland, MA: Sinauer Associates Inc., 2000. — 454 p.
2. Otto, S. & Day, T. A Biologist's Guide to Mathematical Modeling in Ecology and Evolution. — Princeton, NJ: Princeton University Press, 2007. — 450 p.
3. Glasser, J., Meltzer, M. & Levin, B. Mathematical modeling and public policy: responding to health crises. — London: Emerg. Infect. Dis. 2004. — 2050 p.
4. May, R. M. Uses and abuses of mathematics in biology. — New Jersy: Science, 2004. — 837 p.
5. Стеценко І. В. Моделювання систем — Черкаси : ЧДТУ, 2009. — 399 с.
6. А.Є.Кононюк. Узагальнена теорія моделювання. — Київ: освіта України, 2012. — 302 с.
7. Томашевський В. М. Моделювання систем — К. : Видавнича група BHV, 2005. — 352 с.
8. Franzese, P.P, Liu, G., & Arico, S. Environmental accounting models and nature conservation strategies. Ecological Modelling, 2019, — 397p.
9. Anderson RM, May RM. Infectious disease of humans: dynamics and control. — Oxford: Oxford University Press; 1991. — 2036 p.
10. Becker NG. Analysis of infectious disease data. — Chapman and Hall: London; 1989. — 457 p.
11. Diekmann O, Heesterbeek JAP. Mathematical epidemiology of infectious diseases. — Wiley: Chichester; 2000. — 475 p.
12. S.A. Levin, T.G. Hallam and L.J. Gross, Eds. Applied Mathematical Ecology. — Springer-Verlag: New York, 1989. — 738 p.

13.J.D. Murray, Mathematical Biology. — Springer-Verlag: New York, 2nd Edition, 1993. — 454 p.

14.D.J. Daley and J. Gani. Epidemic Modelling: An Introduction. — Cambridge: University Press Cambridge, UK, 1999. — 898 p.

15.Cappuccino, N., and Price, P. W. Population Dynamics: New Approaches and Synthesis. — San Diego, CA: Academic Press. 1995. — 485 p.

16. Ніколя Бакаер : Коротка історія математичної динаміки населення. — Aubervilliers, France: Institut national d’ ́etudes d ́emographiques, 2021. — 423 p.

17.Vandermeer, J. H.; Goldberg, D. E. Population ecology: First principles. — Woodstock, Oxfordshire: Princeton University Press, 2003. — 383 p.

18.A. J. Lotka. Elements of physical biology. — Williams and Wilkins, Baltimore, 1925. — 747 p.

19. Volterra, V. Variations and Fluctuations of the Number of Individuals in Animal Species living together, 1928. — 154 p.

20. Graves, C.P. The Genealogy of Cities, Har/Cdr edition. ed. — Kent, Ohio: The Kent State University Press, 2009. — 837 p.

21.Hillier, B.. The Genetic Code for Cities: Is it Simpler than We Think? In Complexity Theories of Cities Have Come of Age, — Berlin, Heidelberg: Springer. 2012. — 206 p.

22. Marshall, S. Cities Design & Evolution. — Abingdon, Oxon; New York: Routledge, 2008. — 123 p.

23.Rubinow, S.I. Introduction to Mathematical Biology. — New York: J. Wiley & Sons, 1975. — 245 p.

24. L. Benguiguia and E. Blumenfeld-Lieberthal, A dynamic model for city size distribution beyond Zipf’s law, — Physica A: Statistical Mechanics and its Applications, 2007, — 384 p.

25.Kendall, D. J. Deterministic and stochastic epidemics in closed populations. In: Proceedings of the Third Berkeley Symposium on Mathematical Statistics and

Probability, — Berkeley & Los Angeles: University of Califonmia Press, 1956, — 324 p.

26. Воронцов Н. Н. Адаптивність та нейтралізм в еволюції // Екологічна генетика та еволюція. — Кишинів: Штіінца, 1987. — 265 с.

27. Чураєв Р. Н. Гіпотеза про епіген // Дослідження з математичної генетики. — Новосибірськ: Ін-т цитології та генетики ЗІ АН СРСР, 1975. — 121 с.

28. Букатова І. Л., Макрусєв В. В. Теорія цілісно-еволюційної інтелектуалізації соціальних систем. — М.: МИГКУ, 2004. — 125 с.

29. Голубовський М. Д. Деякі аспекти взаємодії генетики та теорії еволюції // Методологічні та філософські проблеми біології. — Новосибірськ: Наука, 1981. — 192 с.

30. Любищев А. А. Про постулати сучасного селектогенезу // Проблеми еволюції. — Новосибірськ: Наука, 1973. — 90 с.

31. Дрейф генів [Електронний ресурс]: Режим досутпу https://en.wikipedia.org/wiki/Genetic_drift

32. Purves, W. K., Sadava, D., Orians, G. H., and Heller, H. C.. Genetic drift may cause large changes in small populations. In Life: — The science of biology, 2003. — 498 p.

33. Williams, C. B. Insect Migration. Annual Review of Entomology. — 1957. — 859 p.

34. А. В. Сиволоб, С.Р. Рушковський, С.С. Кир'яченко, Генетична структура популяції. Закон Харді–Вайнберга. Генетика. — К: Видавничо-поліграфічний центр "Київський університет", 2000. — 345 с.

35. Стрельчук С. І., Демидов С. В., Бердишев Г. Д., Голда Д. М. Генетика з основами селекції. — К., 2000. — 279 с.

36. Ross, R. Report on the prevention of malaria in Mauritius. London, UK: Waterlow and Sons Limited, 1908. — 369 p.

37. Macdonald, G. Theory of the eradication of malaria. Bulletin of the World Health Organization, LA, 1956. — 387 p.
38. N.T.J. Bailey. The Mathematical Theory of Infectious Diseases. — Hafner, New York, 2nd Edition, 1975. — 475 p.
39. Bailey, N. T. J. The mathematical theory of infectious diseases and its applications, — London, Grifn, 1975. — 738 p.
40. Elvback, L., Varma, A. Simulation of mathematical models for public health problems. — Public Health Rep., 1965. — 837 p.
41. Waale, H. T., Piar, M. A. The use of an epidemiological model for estimating the effectiveness of tuberculosis control measures. Sensitivity of the effectiveness of tuberculosis control measures to the coverage of the population. — Bull. World Health Organ., 1969. — 354 p.
42. Kermack, w., McKendrick, A. G. A contribution to the mathematical theory of epidemics // The problem of endemicity. — Proc. r. Soc. A, 1932. — 475 p.
43. Nunes B, Viboud C, Machado A, Ringholz C, Rebelo-de-Andrade H, Nogueira P, et al. Excess mortality associated with influenza epidemics in Portugal, — 1980. — 465 p.
44. Stroup DF, Thacker SB, Herndon JL. Application of multiple time series analysis to the estimation of pneumonia and influenza mortality by age — LA, 1962. — 869 p.
45. Рудченко В.І. Інформатика. Основи алгоритмізації та програмування мовою Python. —Київ: Ранок, 2001. — 170 с.
46. Сьєрра К., Бейтс Б., Беррі П. Python Head First. — LA: Fabula, 2017. — 374 с.
47. С. Чакона, Страуба Б. Git для професійного програміста. — Бостон: MIT Press, 2016. — 800 с.
48. Населення України. URL: https://index.minfin.com.ua/ua/reference/people/ (дата звернення: 18.11.2022)

49. Смертність України за 2021. URL: https://index.minfin.com.ua/ua/reference/people/deaths/2021/ (дата звернення: 18.11.2022)
50. Народжуваність в Україні. URL: https://uk.wikipedia.org/wiki/Народжуваність_в_Україні (дата звернення: 15.11.2022)
51. Вакцинація в Україні. URL: https://index.minfin.com.ua/ua/reference/coronavirus/vaccination/ukraine/ (дата звернення: 13.11.2022)
52. Coronavirus Incubation Period. URL: https://www.webmd.com/lung/coronavirus-incubation-period#1 (дата звернення: 18.11.2022)
53. SARS-Cov-2 incubation period according to vaccination status during the fifth COVID-19 wave in a tertiary-care center in Spain: a cohort study. URL: https://bmcinfectdis.biomedcentral.com/articles/10.1186/s12879-022-07822-4 https://www.webmd.com/lung/coronavirus-incubation-period#1 (дата звернення: 18.11.2022)
54. Coronavirus Recovery. URL: https://www.webmd.com/lung/covid-recovery-overview#2 (дата звернення: 15.11.2022)
55. Diagnosed with covid-19 what to expect. URL: https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/diagnosed-with-covid-19-what-to-expect (дата звернення: 14.11.2022)
56. COVID-19 Vaccine Reduces Severity, Length, Viral Load for Those Who Still Get Infected. URL: https://news.arizona.edu/story/covid-19-vaccine-reduces-severity-length-viral-load-those-who-still-get-infected (дата звернення: 14.11.2022)
57. It’s not just severity—the types of Covid symptoms you get depend on the vaccines you’ve received, new data says. URL: https://www.cnbc.com/2022/10/26/covid-

symptoms-differ-based-on-vaccination-status-zoe-health-study.html (дата звернення: 14.11.2022)

58. Why vaccinated people dying from Covid-19 doesn't mean the vaccines are ineffective. URL: https://edition.cnn.com/2021/10/18/health/covid-19-vaccination-colin-powell-death-wen-wellness/index.html (дата звернення: 12.11.2022)
59. the first stage of the COVID-19 pandemic in Ukraine using epidemiological and genomic data. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8474758/ (дата звернення: 14.11.2022)
60. COVID-19 vaccine efficacy explained. URL: https://www.nebraskamed.com/COVID/covid-19-vaccine-efficacy-explained (дата звернення: 12.11.2022)
61. Modelling of COVID-19. URL: https://www.doherty.edu.au/our-work/institute-themes/viral-infectious-diseases/covid-19/covid-19-modelling/modelling (дата звернення: 10.11.2022)
62. Your unvaccinated friend is roughly 20 times more likely to give you COVID. https://theconversation.com/your-unvaccinated-friend-is-roughly-20-times-more-likely-to-give-you-covid-170448 (дата звернення: 12.11.2022)
63. Vaccinated NSW residents to be allowed into Victoria as the state records 2,179 COVID-19 cases and six deaths. URL: https://www.skynews.com.au/australia-news/coronavirus/watch-live-vic-health-officials-to-provide-covid19-update/news-story/863cdc24d57dd787251a8ccff26b5ec5 (дата звернення: 10.11.2022)
64. The Risk of Vaccinated COVID Transmission Is Not Low. URL: https://www.scientificamerican.com/article/the-risk-of-vaccinated-covid-transmission-is-not-low/ (дата звернення: 10.11.2022)
65. Народжуваність в Україні продовжує падати і готується поставити новий антирекорд останніх 30-ти років. URL: https://opendatabot.ua/analytics/depopulation-2021 (дата звернення: 10.11.2022)

66. Клименко А.І., Подколзін Г.Б. Аналіз розповсюдження COVID-19 на території України з використанням модифікованої SEIRD-моделі. Університетський науковий збірник «Системні науки та інформатика», 22-29 листопада, Київ. — К.: НН ІПСА КПІ ім. Ігоря Сікорського, 2022. С.140-146.
67. Клименко А.І. Математичні моделі взаємодії популяцій типу хижак-жертва: дипломна робота бакалавра: 124 Системний аналіз / Клименко Анастасія Іллівна. — Київ, 2021. — 123 с. https://ela.kpi.ua/handle/123456789/45272

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

```
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets


class OptionsMenu_SEIR(QtWidgets.QWidget):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)

        # Create the "SIR Coefficients" options
        self.beta_sb = QtWidgets.QDoubleSpinBox()
        self.gamma_sb = QtWidgets.QDoubleSpinBox()
        self.alpha_sb = QtWidgets.QDoubleSpinBox()

        self.t_incubation = QtWidgets.QDoubleSpinBox()
        self.t_recovery = QtWidgets.QDoubleSpinBox()
        self.R0 = QtWidgets.QDoubleSpinBox()

        self.t_incubation.valueChanged.connect(self.onTIncChanged)
        self.alpha_sb.valueChanged.connect(self.onAlphaChanged)
        self.gamma_sb.valueChanged.connect(self.onGammaChanged)
        self.t_recovery.valueChanged.connect(self.onTrecChanged)

        self.beta_sb = QtWidgets.QDoubleSpinBox()
        self.beta_sb.valueChanged.connect(self.onBetaChanged)

        for widget in (self.beta_sb, self.gamma_sb, self.alpha_sb):
            widget.setRange(0, 10)
            widget.setDecimals(4)
            widget.setSingleStep(0.001)

        for widget in (self.t_recovery, self.R0, self.t_incubation):
            widget.setRange(0, 100)
            widget.setDecimals(4)
            widget.setSingleStep(0.001)

        coeff_grid = QtWidgets.QGridLayout()
        coeff_grid.addWidget(QtWidgets.QLabel('Константа швидкості β'), 0, 0)
        coeff_grid.addWidget(self.beta_sb, 0, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість одужання γ'), 1, 0)
        coeff_grid.addWidget(self.gamma_sb, 1, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Середній час одужання після інфекції'), 2, 0)
        coeff_grid.addWidget(self.t_recovery, 2, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Передаваність хвороби'), 3, 0)
        coeff_grid.addWidget(self.R0, 3, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Інкубаційний період'), 4, 0)
        coeff_grid.addWidget(self.t_incubation, 4, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість переходу від інкубаційної стадії до відкритої
α'), 5, 0)
        coeff_grid.addWidget(self.alpha_sb, 5, 1)
```

```
coeff_gb = QtWidgets.QGroupBox('Коефіцієнти SEIR-моделі:')
coeff_gb.setLayout(coeff_grid)

# Create the "Other Parameters" options
self.sus_sb = QtWidgets.QDoubleSpinBox()
self.sus_sb.setRange(0, 100000)
self.sus_sb.setSingleStep(1)

self.exp_sb = QtWidgets.QDoubleSpinBox()
self.exp_sb.setRange(0, 100000)
self.exp_sb.setSingleStep(1)

self.inf_sb = QtWidgets.QDoubleSpinBox()
self.inf_sb.setRange(0, 100000)
self.inf_sb.setSingleStep(1)

self.rec_sb = QtWidgets.QDoubleSpinBox()
self.rec_sb.setRange(0, 100000)
self.rec_sb.setSingleStep(1)

self.days_sb = QtWidgets.QSpinBox()
self.days_sb.setRange(0, 100000)
self.days_sb.setSingleStep(100)

other_grid = QtWidgets.QGridLayout()
other_grid.addWidget(QtWidgets.QLabel('Сприятливі:'), 0, 0)
other_grid.addWidget(self.sus_sb, 0, 1)
other_grid.addWidget(QtWidgets.QLabel('Із хворобою у інкубаційному періоді:'), 1, 0)
other_grid.addWidget(self.exp_sb, 1, 1)
other_grid.addWidget(QtWidgets.QLabel('Інфіковані'), 2, 0)
other_grid.addWidget(self.inf_sb, 2, 1)
other_grid.addWidget(QtWidgets.QLabel('Перехворівші:'), 3, 0)
other_grid.addWidget(self.rec_sb, 3, 1)

other_grid.addWidget(QtWidgets.QLabel('Дні:'), 4, 0)
other_grid.addWidget(self.days_sb, 4, 1)
#other_grid.addWidget(QtWidgets.QLabel('Час дельта:'), 5, 0)
#other_grid.addWidget(self.timedelta_sb, 5, 1)

other_gb = QtWidgets.QGroupBox('Інші параметри:')
other_gb.setLayout(other_grid)

# Create the "Graph Options" options
self.legend_cb = QtWidgets.QCheckBox('Показати легенду:')
self.legend_cb.setChecked(True)
self.legend_cb.stateChanged.connect(self.legend_change)

self.grid_cb = QtWidgets.QCheckBox('Показати сітку:')
self.grid_cb.setChecked(True)
self.legend_loc_lbl = QtWidgets.QLabel('Місцезнаходження легенди:')
self.legend_loc_cb = QtWidgets.QComboBox()
self.legend_loc_cb.addItems([x.title() for x in [
    'right',
```

```
            'center',
            'lower left',
            'center right',
            'upper left',
            'center left',
            'upper right',
            'lower right',
            'upper center',
            'lower center',
            'best',
        ]])
        self.legend_loc_cb.setCurrentIndex(6)

        cb_box = QtWidgets.QHBoxLayout()
        cb_box.addWidget(self.legend_cb)
        cb_box.addWidget(self.grid_cb)

        legend_box = QtWidgets.QHBoxLayout()
        legend_box.addWidget(self.legend_loc_cb)
        legend_box.addStretch()

        graph_box = QtWidgets.QVBoxLayout()
        graph_box.addLayout(cb_box)
        graph_box.addWidget(self.legend_loc_lbl)
        graph_box.addLayout(legend_box)

        graph_gb = QtWidgets.QGroupBox('Налаштування графіку:')
        graph_gb.setLayout(graph_box)

        # Create the update/reset buttons
        self.update_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/calculator.png'),
            'Запуск ітерацій')

       # self.inf_btn = QtWidgets.QPushButton('Інфіковані')
       # self.preyPredator_btn = QtWidgets.QPushButton('Хищники-жертвы')

        self.reset_values_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/arrow_undo.png'),
            'Зброс значень')
        self.clear_graph_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/chart_line_delete.png'),
            'Очистити графік')

        self.reset_values_btn.clicked.connect(self.reset_values)

#       self.preySuperpredator_btn.clicked.connect(self.preySuperpredator)
#       self.preyPredator_btn.clicked.connect(self.preyPredator)

        # Create the main layout
        container = QtWidgets.QVBoxLayout()
        container.addWidget(coeff_gb)
        container.addWidget(other_gb)
```

```
    container.addWidget(graph_gb)
    container.addWidget(self.update_btn)
  #  container.addStretch()
 #   container.addWidget(self.preySuperpredator_btn)
 #   container.addWidget(self.preyPredator_btn)
 #
    container.addWidget(self.reset_values_btn)
    container.addWidget(self.clear_graph_btn)
    container.addStretch()
    self.setLayout(container)

    # Populate the widgets with values
    self.reset_values()

 def onAlphaChanged(self):
    if self.alpha_sb.value() == 0:
        return
    newVal = 1/self.alpha_sb.value()
    if newVal != self.t_incubation.value():
        self.t_incubation.setValue(newVal)

 def onTIncChanged(self):
    if self.t_incubation.value() == 0:
        return

    newVal = 1/self.t_incubation.value()
    if newVal != self.alpha_sb.value():
        self.alpha_sb.setValue(newVal)

 def onGammaChanged(self):
    if self.gamma_sb.value() == 0:
        return
    newVal = 1/self.gamma_sb.value()
    new = self.beta_sb.value() / self.gamma_sb.value()
    if newVal != self.t_recovery.value():
        self.t_recovery.setValue(newVal)
        self.R0.setValue(new)

 def onBetaChanged(self):
    if self.gamma_sb.value() == 0:
        return
    newVal = self.beta_sb.value() / self.gamma_sb.value()
    if newVal != self.gamma_sb.value():
        self.R0.setValue(newVal)

# def onR0Changed(self):
#     if self.gamma_sb.value() == 0:
#         return
#     newVal = self.beta_sb.value()/self.gamma_sb.value()
#     self.R0.setValue(newVal)

 def onTrecChanged(self):
    if self.t_recovery.value() == 0:
```

```
            return
        newVal = 1 / self.t_recovery.value()
        if newVal != self.gamma_sb.value():
            self.gamma_sb.setValue(newVal)

    def reset_values(self):
        self.beta_sb.setValue(0.1)
        self.gamma_sb.setValue(0.1)
        self.sus_sb.setValue(5)
        self.inf_sb.setValue(20)
        self.rec_sb.setValue(5)
        self.alpha_sb.setValue(0.1)
        self.exp_sb.setValue(10)
        self.days_sb.setValue(100)
        #self.timedelta_sb.setValue(0.02)

    def legend_change(self):
        self.legend_loc_cb.setEnabled(self.legend_cb.isChecked())
        self.legend_loc_lbl.setEnabled(self.legend_cb.isChecked())
# 3rd party modules
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets

class OptionsMenu_SEIR_mod(QtWidgets.QWidget):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)

        # Create the "SIR Coefficients" options
        self.beta_sb = QtWidgets.QDoubleSpinBox()
        self.gamma_sb = QtWidgets.QDoubleSpinBox()
        self.alpha_sb = QtWidgets.QDoubleSpinBox()
        self.theta_sb = QtWidgets.QDoubleSpinBox()

        self.t_incubation = QtWidgets.QDoubleSpinBox()
        self.t_recovery = QtWidgets.QDoubleSpinBox()
        self.R0 = QtWidgets.QDoubleSpinBox()

        self.t_incubation.valueChanged.connect(self.onTIncChanged)
        self.alpha_sb.valueChanged.connect(self.onAlphaChanged)
        self.gamma_sb.valueChanged.connect(self.onGammaChanged)
        self.t_recovery.valueChanged.connect(self.onTrecChanged)

        self.beta_sb = QtWidgets.QDoubleSpinBox()
        self.beta_sb.valueChanged.connect(self.onBetaChanged)

        for widget in (self.beta_sb, self.gamma_sb, self.alpha_sb, self.theta_sb):
            widget.setRange(0, 10)
            widget.setDecimals(4)
            widget.setSingleStep(0.001)

        for widget in (self.t_recovery, self.R0, self.t_incubation):
            widget.setRange(0, 100)
```

```
            widget.setDecimals(4)
            widget.setSingleStep(0.001)

        coeff_grid = QtWidgets.QGridLayout()
        coeff_grid.addWidget(QtWidgets.QLabel('Константа швидкості β'), 0, 0)
        coeff_grid.addWidget(self.beta_sb, 0, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість одужання γ'), 1, 0)
        coeff_grid.addWidget(self.gamma_sb, 1, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Середній час одужання після інфекції'), 2, 0)
        coeff_grid.addWidget(self.t_recovery, 2, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Передаваність хвороби'), 3, 0)
        coeff_grid.addWidget(self.R0, 3, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Інкубаційний період'), 4, 0)
        coeff_grid.addWidget(self.t_incubation, 4, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість переходу від інкубаційної стадії до відкритої
α'), 5, 0)
        coeff_grid.addWidget(self.alpha_sb, 5, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Ступінь заразності латентних носіїв у порівнянні з
захворілими Θ'), 6, 0)
        coeff_grid.addWidget(self.theta_sb, 6, 1)

        coeff_gb = QtWidgets.QGroupBox('Коефіцієнти модифікованої SEIR-моделі:')
        coeff_gb.setLayout(coeff_grid)

        # Create the "Other Parameters" options
        self.sus_sb = QtWidgets.QDoubleSpinBox()
        self.sus_sb.setRange(0, 100000)
        self.sus_sb.setSingleStep(1)

        self.exp_sb = QtWidgets.QDoubleSpinBox()
        self.exp_sb.setRange(0, 100000)
        self.exp_sb.setSingleStep(1)

        self.inf_sb = QtWidgets.QDoubleSpinBox()
        self.inf_sb.setRange(0, 100000)
        self.inf_sb.setSingleStep(1)

        self.rec_sb = QtWidgets.QDoubleSpinBox()
        self.rec_sb.setRange(0, 100000)
        self.rec_sb.setSingleStep(1)

        self.days_sb = QtWidgets.QSpinBox()
        self.days_sb.setRange(0, 100000)
        self.days_sb.setSingleStep(100)

        #self.timedelta_sb = QtWidgets.QDoubleSpinBox()
        #self.timedelta_sb.setRange(0, 100)
        #self.timedelta_sb.setSingleStep(0.05)

        other_grid = QtWidgets.QGridLayout()
        other_grid.addWidget(QtWidgets.QLabel('Сприятливі:'), 0, 0)
        other_grid.addWidget(self.sus_sb, 0, 1)
        other_grid.addWidget(QtWidgets.QLabel('Із хворобою у інкубаційному періоді:'), 1, 0)
```

```
other_grid.addWidget(self.exp_sb, 1, 1)
other_grid.addWidget(QtWidgets.QLabel('Інфіковані'), 2, 0)
other_grid.addWidget(self.inf_sb, 2, 1)
other_grid.addWidget(QtWidgets.QLabel('Очухані:'), 3, 0)
other_grid.addWidget(self.rec_sb, 3, 1)

other_grid.addWidget(QtWidgets.QLabel('Дні:'), 4, 0)
other_grid.addWidget(self.days_sb, 4, 1)
#other_grid.addWidget(QtWidgets.QLabel('Час дельта:'), 5, 0)
#other_grid.addWidget(self.timedelta_sb, 5, 1)

other_gb = QtWidgets.QGroupBox('Інші параметри:')
other_gb.setLayout(other_grid)

# Create the "Graph Options" options
self.legend_cb = QtWidgets.QCheckBox('Показати легенду:')
self.legend_cb.setChecked(True)
self.legend_cb.stateChanged.connect(self.legend_change)

self.grid_cb = QtWidgets.QCheckBox('Показати сітку:')
self.grid_cb.setChecked(True)
self.legend_loc_lbl = QtWidgets.QLabel('Місцезнаходження легенди:')
self.legend_loc_cb = QtWidgets.QComboBox()
self.legend_loc_cb.addItems([x.title() for x in [
    'right',
    'center',
    'lower left',
    'center right',
    'upper left',
    'center left',
    'upper right',
    'lower right',
    'upper center',
    'lower center',
    'best',
]])
self.legend_loc_cb.setCurrentIndex(6)

cb_box = QtWidgets.QHBoxLayout()
cb_box.addWidget(self.legend_cb)
cb_box.addWidget(self.grid_cb)

legend_box = QtWidgets.QHBoxLayout()
legend_box.addWidget(self.legend_loc_cb)
legend_box.addStretch()

graph_box = QtWidgets.QVBoxLayout()
graph_box.addLayout(cb_box)
graph_box.addWidget(self.legend_loc_lbl)
graph_box.addLayout(legend_box)

graph_gb = QtWidgets.QGroupBox('Налаштування графіку:')
graph_gb.setLayout(graph_box)
```

```
    # Create the update/reset buttons
    self.update_btn = QtWidgets.QPushButton(
        QtGui.QIcon(':/resources/calculator.png'),
        'Запуск ітерацій')

   # self.inf_btn = QtWidgets.QPushButton('Інфіковані')
   # self.preyPredator_btn = QtWidgets.QPushButton('Хищники-жертвы')

    self.reset_values_btn = QtWidgets.QPushButton(
        QtGui.QIcon(':/resources/arrow_undo.png'),
        'Зброс значень')
    self.clear_graph_btn = QtWidgets.QPushButton(
        QtGui.QIcon(':/resources/chart_line_delete.png'),
        'Очистити графік')

    self.reset_values_btn.clicked.connect(self.reset_values)

 #   self.preySuperpredator_btn.clicked.connect(self.preySuperpredator)
  #  self.preyPredator_btn.clicked.connect(self.preyPredator)

    # Create the main layout
    container = QtWidgets.QVBoxLayout()
    container.addWidget(coeff_gb)
    container.addWidget(other_gb)
    container.addWidget(graph_gb)
    container.addWidget(self.update_btn)
  #  container.addStretch()
 #   container.addWidget(self.preySuperpredator_btn)
 #   container.addWidget(self.preyPredator_btn)
 #
    container.addWidget(self.reset_values_btn)
    container.addWidget(self.clear_graph_btn)
    container.addStretch()
    self.setLayout(container)

    # Populate the widgets with values
    self.reset_values()

def onAlphaChanged(self):
    if self.alpha_sb.value() == 0:
        return
    newVal = 1/self.alpha_sb.value()
    if newVal != self.t_incubation.value():
        self.t_incubation.setValue(newVal)

def onTIncChanged(self):
    if self.t_incubation.value() == 0:
        return

    newVal = 1/self.t_incubation.value()
    if newVal != self.alpha_sb.value():
        self.alpha_sb.setValue(newVal)
```

```
    def onGammaChanged(self):
        if self.gamma_sb.value() == 0:
            return
        newVal = 1/self.gamma_sb.value()
        new = self.beta_sb.value() / self.gamma_sb.value()
        if newVal != self.t_recovery.value():
            self.t_recovery.setValue(newVal)
            self.R0.setValue(new)

    def onBetaChanged(self):
        if self.gamma_sb.value() == 0:
            return
        newVal = self.beta_sb.value() / self.gamma_sb.value()
        if newVal != self.gamma_sb.value():
            self.R0.setValue(newVal)

    # def onR0Changed(self):
    #     if self.gamma_sb.value() == 0:
    #         return
    #     newVal = self.beta_sb.value()/self.gamma_sb.value()
    #     self.R0.setValue(newVal)

    def onTrecChanged(self):
        if self.t_recovery.value() == 0:
            return
        newVal = 1 / self.t_recovery.value()
        if newVal != self.gamma_sb.value():
            self.gamma_sb.setValue(newVal)

    def reset_values(self):
        self.beta_sb.setValue(0.1)
        self.gamma_sb.setValue(0.1)
        self.sus_sb.setValue(5)
        self.inf_sb.setValue(20)
        self.rec_sb.setValue(5)
        self.alpha_sb.setValue(0.1)
        self.exp_sb.setValue(10)
        self.days_sb.setValue(100)
        #self.timedelta_sb.setValue(0.02)

    def legend_change(self):
        self.legend_loc_cb.setEnabled(self.legend_cb.isChecked())
        self.legend_loc_lbl.setEnabled(self.legend_cb.isChecked())
# 3rd party modules
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets


class OptionsMenu_SEIR_vacc(QtWidgets.QWidget):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
```

```
        # Create the "SIR Coefficients" options
        self.beta_sb = QtWidgets.QDoubleSpinBox()
        self.gamma_sb = QtWidgets.QDoubleSpinBox()
        self.alpha_sb = QtWidgets.QDoubleSpinBox()
        self.u_sb = QtWidgets.QDoubleSpinBox()
        self.t_incubation = QtWidgets.QDoubleSpinBox()
        self.t_recovery = QtWidgets.QDoubleSpinBox()
        self.R0 = QtWidgets.QDoubleSpinBox()

        self.t_incubation.valueChanged.connect(self.onTIncChanged)
        self.alpha_sb.valueChanged.connect(self.onAlphaChanged)
        self.gamma_sb.valueChanged.connect(self.onGammaChanged)
        self.t_recovery.valueChanged.connect(self.onTrecChanged)

        self.beta_sb = QtWidgets.QDoubleSpinBox()
        self.beta_sb.valueChanged.connect(self.onBetaChanged)

        for widget in (self.beta_sb, self.gamma_sb, self.alpha_sb):
            widget.setRange(0, 10)
            widget.setDecimals(4)
            widget.setSingleStep(0.001)

        self.u_sb.setRange(0, 1)
        self.u_sb.setSingleStep(0.001)

        for widget in (self.t_recovery, self.R0, self.t_incubation):
            widget.setRange(0, 100)
            widget.setDecimals(4)
            widget.setSingleStep(0.01)

        coeff_grid = QtWidgets.QGridLayout()
        coeff_grid.addWidget(QtWidgets.QLabel('Константа швидкості β'), 0, 0)
        coeff_grid.addWidget(self.beta_sb, 0, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість одужання γ'), 1, 0)
        coeff_grid.addWidget(self.gamma_sb, 1, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Середній час одужання після інфекції'), 2, 0)
        coeff_grid.addWidget(self.t_recovery, 2, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Передаваність хвороби'), 3, 0)
        coeff_grid.addWidget(self.R0, 3, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Інкубаційний період'), 4, 0)
        coeff_grid.addWidget(self.t_incubation, 4, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість передачі захворювання через взаємодію
населення u'), 5, 0)
        coeff_grid.addWidget(self.u_sb, 5, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість переходу від інкубаційної стадії до відкритої
α'), 6, 0)
        coeff_grid.addWidget(self.alpha_sb, 6, 1)

        coeff_gb = QtWidgets.QGroupBox('Коефіцієнти SEIR-моделі з вакцинацією:')
        coeff_gb.setLayout(coeff_grid)

        # Create the "Other Parameters" options
```

```
 self.sus_sb = QtWidgets.QDoubleSpinBox()
 self.sus_sb.setRange(0, 100000)
 self.sus_sb.setSingleStep(1)

 self.exp_sb = QtWidgets.QDoubleSpinBox()
 self.exp_sb.setRange(0, 100000)
 self.exp_sb.setSingleStep(1)

 self.inf_sb = QtWidgets.QDoubleSpinBox()
 self.inf_sb.setRange(0, 100000)
 self.inf_sb.setSingleStep(1)

 self.rec_sb = QtWidgets.QDoubleSpinBox()
 self.rec_sb.setRange(0, 100000)
 self.rec_sb.setSingleStep(1)

 self.days_sb = QtWidgets.QSpinBox()
 self.days_sb.setRange(0, 100000)
 self.days_sb.setSingleStep(100)

# self.timedelta_sb = QtWidgets.QDoubleSpinBox()
# self.timedelta_sb.setRange(0, 100)
# self.timedelta_sb.setSingleStep(0.05)

 other_grid = QtWidgets.QGridLayout()
 other_grid.addWidget(QtWidgets.QLabel('Сприятливі:'), 0, 0)
 other_grid.addWidget(self.sus_sb, 0, 1)
 other_grid.addWidget(QtWidgets.QLabel('Із хворобою у інкубаційному періоді:'), 1, 0)
 other_grid.addWidget(self.exp_sb, 1, 1)
 other_grid.addWidget(QtWidgets.QLabel('Інфіковані'), 2, 0)
 other_grid.addWidget(self.inf_sb, 2, 1)
 other_grid.addWidget(QtWidgets.QLabel('Очухані:'), 3, 0)
 other_grid.addWidget(self.rec_sb, 3, 1)

 other_grid.addWidget(QtWidgets.QLabel('Дні:'), 4, 0)
 other_grid.addWidget(self.days_sb, 4, 1)
 #other_grid.addWidget(QtWidgets.QLabel('Час дельта:'), 5, 0)
 #other_grid.addWidget(self.timedelta_sb, 5, 1)

 other_gb = QtWidgets.QGroupBox('Інші параметри:')
 other_gb.setLayout(other_grid)

 # Create the "Graph Options" options
 self.legend_cb = QtWidgets.QCheckBox('Показати легенду:')
 self.legend_cb.setChecked(True)
 self.legend_cb.stateChanged.connect(self.legend_change)

 self.grid_cb = QtWidgets.QCheckBox('Показати сітку:')
 self.grid_cb.setChecked(True)
 self.legend_loc_lbl = QtWidgets.QLabel('Місцезнаходження легенди:')
 self.legend_loc_cb = QtWidgets.QComboBox()
 self.legend_loc_cb.addItems([x.title() for x in [
    'right',
```

```
            'center',
            'lower left',
            'center right',
            'upper left',
            'center left',
            'upper right',
            'lower right',
            'upper center',
            'lower center',
            'best',
        ]])
        self.legend_loc_cb.setCurrentIndex(6)

        cb_box = QtWidgets.QHBoxLayout()
        cb_box.addWidget(self.legend_cb)
        cb_box.addWidget(self.grid_cb)

        legend_box = QtWidgets.QHBoxLayout()
        legend_box.addWidget(self.legend_loc_cb)
        legend_box.addStretch()

        graph_box = QtWidgets.QVBoxLayout()
        graph_box.addLayout(cb_box)
        graph_box.addWidget(self.legend_loc_lbl)
        graph_box.addLayout(legend_box)

        graph_gb = QtWidgets.QGroupBox('Налаштування графіку:')
        graph_gb.setLayout(graph_box)

        # Create the update/reset buttons
        self.update_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/calculator.png'),
            'Запуск ітерацій')

       # self.inf_btn = QtWidgets.QPushButton('Інфіковані')
       # self.preyPredator_btn = QtWidgets.QPushButton('Хищники-жертвы')

        self.reset_values_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/arrow_undo.png'),
            'Зброс значень')
        self.clear_graph_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/chart_line_delete.png'),
            'Очистити графік')

        self.reset_values_btn.clicked.connect(self.reset_values)

#       self.preySuperpredator_btn.clicked.connect(self.preySuperpredator)
#       self.preyPredator_btn.clicked.connect(self.preyPredator)

        # Create the main layout
        container = QtWidgets.QVBoxLayout()
        container.addWidget(coeff_gb)
        container.addWidget(other_gb)
```

```
        container.addWidget(graph_gb)
        container.addWidget(self.update_btn)
     #  container.addStretch()
    #   container.addWidget(self.preySuperpredator_btn)
    #   container.addWidget(self.preyPredator_btn)
    #
        container.addWidget(self.reset_values_btn)
        container.addWidget(self.clear_graph_btn)
        container.addStretch()
        self.setLayout(container)

        # Populate the widgets with values
        self.reset_values()

    def onAlphaChanged(self):
        if self.alpha_sb.value() == 0:
            return
        newVal = 1/self.alpha_sb.value()
        if newVal != self.t_incubation.value():
            self.t_incubation.setValue(newVal)

    def onTIncChanged(self):
        if self.t_incubation.value() == 0:
            return

        newVal = 1/self.t_incubation.value()
        if newVal != self.alpha_sb.value():
            self.alpha_sb.setValue(newVal)

    def onGammaChanged(self):
        if self.gamma_sb.value() == 0:
            return
        newVal = 1/self.gamma_sb.value()
        new = self.beta_sb.value() / self.gamma_sb.value()
        if newVal != self.t_recovery.value():
            self.t_recovery.setValue(newVal)
            self.R0.setValue(new)

    def onBetaChanged(self):
        if self.gamma_sb.value() == 0:
            return
        newVal = self.beta_sb.value() / self.gamma_sb.value()
        if newVal != self.gamma_sb.value():
            self.R0.setValue(newVal)

    # def onR0Changed(self):
    #     if self.gamma_sb.value() == 0:
    #         return
    #     newVal = self.beta_sb.value()/self.gamma_sb.value()
    #     self.R0.setValue(newVal)

    def onTrecChanged(self):
        if self.t_recovery.value() == 0:
```

```
            return
        newVal = 1 / self.t_recovery.value()
        if newVal != self.gamma_sb.value():
            self.gamma_sb.setValue(newVal)

    def reset_values(self):
        self.beta_sb.setValue(0.1)
        self.u_sb.setValue(1)
        self.gamma_sb.setValue(0.1)
        self.sus_sb.setValue(5)
        self.inf_sb.setValue(20)
        self.rec_sb.setValue(5)
        self.alpha_sb.setValue(0.1)
        self.exp_sb.setValue(10)
        self.days_sb.setValue(100)
        # self.timedelta_sb.setValue(0.02)

    def legend_change(self):
        self.legend_loc_cb.setEnabled(self.legend_cb.isChecked())
        self.legend_loc_lbl.setEnabled(self.legend_cb.isChecked())
# 3rd party modules
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets


class OptionsMenu_SEIRD_full_with_vacc(QtWidgets.QWidget):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)

        # Create the "SIR Coefficients" options
        self.beta_unvac_unvac_sb = QtWidgets.QDoubleSpinBox()
        self.beta_unvac_vac_sb = QtWidgets.QDoubleSpinBox()
        self.beta_vac_unvac_sb = QtWidgets.QDoubleSpinBox()
        self.beta_vac_vac_sb = QtWidgets.QDoubleSpinBox()

        self.gamma_unvac_sb = QtWidgets.QDoubleSpinBox()
        self.gamma_vac_sb = QtWidgets.QDoubleSpinBox()

        self.alpha_unvac_sb = QtWidgets.QDoubleSpinBox()
        self.alpha_vac_sb = QtWidgets.QDoubleSpinBox()

        self.mu_sb = QtWidgets.QDoubleSpinBox()
        self.l_sb = QtWidgets.QDoubleSpinBox()

        self.theta_unvac_sb = QtWidgets.QDoubleSpinBox()
        self.theta_vac_sb = QtWidgets.QDoubleSpinBox()

        self.t_incubation = QtWidgets.QDoubleSpinBox()
        self.t_recovery = QtWidgets.QDoubleSpinBox()
        self.R0 = QtWidgets.QDoubleSpinBox()

       #  self.t_incubation.valueChanged.connect(self.onTIncChanged)
```

```
        # self.alpha_sb.valueChanged.connect(self.onAlphaChanged)
        # self.gamma_sb.valueChanged.connect(self.onGammaChanged)
        # self.t_recovery.valueChanged.connect(self.onTrecChanged)

       #  self.beta_sb = QtWidgets.QDoubleSpinBox()
      #   self.beta_sb.valueChanged.connect(self.onBetaChanged)

        for widget in (self.beta_unvac_unvac_sb, self.beta_unvac_vac_sb, self.beta_vac_unvac_sb,
self.beta_vac_vac_sb,
                       self.gamma_unvac_sb, self.gamma_vac_sb, self.alpha_unvac_sb, self.alpha_vac_sb,
self.mu_sb, self.l_sb,
                       self.theta_unvac_sb, self.theta_vac_sb):
            widget.setRange(0, 2)
            widget.setDecimals(10)
            widget.setSingleStep(0.001)

        for widget in (self.t_recovery, self.R0, self.t_incubation):
            widget.setRange(0, 100)
            widget.setDecimals(10)
            widget.setSingleStep(0.01)



        coeff_grid = QtWidgets.QGridLayout()
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість передачі захворювання від невакцинованих до
невакцинованих'), 0, 0)
        coeff_grid.addWidget(self.beta_unvac_unvac_sb, 0, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість передачі захворювання від невакцинованих до
вак'), 1, 0)
        coeff_grid.addWidget(self.beta_unvac_vac_sb, 1, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість передачі захворювання від вакцинованих до
невакцинованих'), 2, 0)
        coeff_grid.addWidget(self.beta_vac_unvac_sb, 2, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість передачі захворювання від вакцинованих до
невакцинованих'), 3, 0)
        coeff_grid.addWidget(self.beta_vac_vac_sb, 3, 1)

        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість одужання невакцинованих'), 4, 0)
        coeff_grid.addWidget(self.gamma_unvac_sb, 4, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість одужання вакцинованих'), 5, 0)
        coeff_grid.addWidget(self.gamma_vac_sb, 5, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість переходу від інкубаційної стадії до відкритої
у невакцинованих'), 6, 0)
        coeff_grid.addWidget(self.alpha_unvac_sb, 6, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Швидкість переходу від інкубаційної стадії до відкритої
у вакцинованих'), 7, 0)
        coeff_grid.addWidget(self.alpha_vac_sb, 7, 1)

        coeff_grid.addWidget(QtWidgets.QLabel('Природня смертність'), 8, 0)
        coeff_grid.addWidget(self.mu_sb, 8, 1)
        coeff_grid.addWidget(QtWidgets.QLabel('Народжуваність'), 9, 0)
        coeff_grid.addWidget(self.l_sb, 9, 1)
```

```
coeff_grid.addWidget(QtWidgets.QLabel('Смертність від вірусу невакцинованих'), 10, 0)
coeff_grid.addWidget(self.theta_unvac_sb, 10, 1)
coeff_grid.addWidget(QtWidgets.QLabel('Смертність від вірусу вакцинованих'), 11, 0)
coeff_grid.addWidget(self.theta_vac_sb, 11, 1)

coeff_gb = QtWidgets.QGroupBox('Коефіцієнти SEIRD-моделі з вакцинацією:')
coeff_gb.setLayout(coeff_grid)


# Create the "Other Parameters" options
self.sus_unvac_sb = QtWidgets.QDoubleSpinBox()
self.sus_unvac_sb.setRange(0, 10000000)
self.sus_unvac_sb.setSingleStep(1)

self.sus_vac_sb = QtWidgets.QDoubleSpinBox()
self.sus_vac_sb.setRange(0, 10000000)
self.sus_vac_sb.setSingleStep(1)

self.exp_unvac_sb = QtWidgets.QDoubleSpinBox()
self.exp_unvac_sb.setRange(0, 10000000)
self.exp_unvac_sb.setSingleStep(1)

self.exp_vac_sb = QtWidgets.QDoubleSpinBox()
self.exp_vac_sb.setRange(0, 10000000)
self.exp_vac_sb.setSingleStep(1)

self.inf_unvac_sb = QtWidgets.QDoubleSpinBox()
self.inf_unvac_sb.setRange(0, 10000000)
self.inf_unvac_sb.setSingleStep(1)

self.inf_vac_sb = QtWidgets.QDoubleSpinBox()
self.inf_vac_sb.setRange(0, 10000000)
self.inf_vac_sb.setSingleStep(1)

self.rec_unvac_sb = QtWidgets.QDoubleSpinBox()
self.rec_unvac_sb.setRange(0, 10000000)
self.rec_unvac_sb.setSingleStep(1)

self.rec_vac_sb = QtWidgets.QDoubleSpinBox()
self.rec_vac_sb.setRange(0, 10000000)
self.rec_vac_sb.setSingleStep(1)

self.dead_sb = QtWidgets.QDoubleSpinBox()
self.dead_sb.setRange(0, 100000)
self.dead_sb.setSingleStep(1)

self.days_sb = QtWidgets.QSpinBox()
self.days_sb.setRange(0, 100000)
self.days_sb.setSingleStep(100)

# self.timedelta_sb = QtWidgets.QDoubleSpinBox()
# self.timedelta_sb.setRange(0, 100)
# self.timedelta_sb.setSingleStep(0.05)
```

```
        other_grid = QtWidgets.QGridLayout()
        other_grid.addWidget(QtWidgets.QLabel('Сприятливі невакциновані:'), 0, 0)
        other_grid.addWidget(self.sus_unvac_sb, 0, 1)
        other_grid.addWidget(QtWidgets.QLabel('Сприятливі вакциновані:'), 1, 0)
        other_grid.addWidget(self.sus_vac_sb, 1, 1)

        other_grid.addWidget(QtWidgets.QLabel('Із хворобою у інкубаційному періоді невакциновані:'), 2,
0)
        other_grid.addWidget(self.exp_unvac_sb, 2, 1)
        other_grid.addWidget(QtWidgets.QLabel('Із хворобою у інкубаційному періоді вакциновані:'), 3, 0)
        other_grid.addWidget(self.exp_vac_sb, 3, 1)

        other_grid.addWidget(QtWidgets.QLabel('Інфіковані невакциновані:'), 4, 0)
        other_grid.addWidget(self.inf_unvac_sb, 4, 1)

        other_grid.addWidget(QtWidgets.QLabel('Інфіковані вакциновані:'), 5, 0)
        other_grid.addWidget(self.inf_vac_sb, 5, 1)

        other_grid.addWidget(QtWidgets.QLabel('Одужавші невакциновані:'), 6, 0)
        other_grid.addWidget(self.rec_unvac_sb, 6, 1)

        other_grid.addWidget(QtWidgets.QLabel('Одужавші вакциновані:'), 7, 0)
        other_grid.addWidget(self.rec_vac_sb, 7, 1)

        other_grid.addWidget(QtWidgets.QLabel('Померші:'), 8, 0)
        other_grid.addWidget(self.dead_sb, 8, 1)

        other_grid.addWidget(QtWidgets.QLabel('Дні:'), 9, 0)
        other_grid.addWidget(self.days_sb, 9, 1)
        #other_grid.addWidget(QtWidgets.QLabel('Час дельта:'), 5, 0)
        #other_grid.addWidget(self.timedelta_sb, 5, 1)

        other_gb = QtWidgets.QGroupBox('Інші параметри:')
        other_gb.setLayout(other_grid)

        # Create the "Graph Options" options
        self.legend_cb = QtWidgets.QCheckBox('Показати легенду:')
        self.legend_cb.setChecked(True)
        self.legend_cb.stateChanged.connect(self.legend_change)

        self.grid_cb = QtWidgets.QCheckBox('Показати сітку:')
        self.grid_cb.setChecked(True)
        self.legend_loc_lbl = QtWidgets.QLabel('Місцезнаходження легенди:')
        self.legend_loc_cb = QtWidgets.QComboBox()
        self.legend_loc_cb.addItems([x.title() for x in [
            'right',
            'center',
            'lower left',
            'center right',
            'upper left',
            'center left',
            'upper right',
```

```
            'lower right',
            'upper center',
            'lower center',
            'best',
        ]])
        self.legend_loc_cb.setCurrentIndex(6)

        cb_box = QtWidgets.QHBoxLayout()
        cb_box.addWidget(self.legend_cb)
        cb_box.addWidget(self.grid_cb)

        legend_box = QtWidgets.QHBoxLayout()
        legend_box.addWidget(self.legend_loc_cb)
        legend_box.addStretch()

        graph_box = QtWidgets.QVBoxLayout()
        graph_box.addLayout(cb_box)
        graph_box.addWidget(self.legend_loc_lbl)
        graph_box.addLayout(legend_box)

        graph_gb = QtWidgets.QGroupBox('Налаштування графіку:')
        graph_gb.setLayout(graph_box)

        # Create the update/reset buttons
        self.update_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/calculator.png'),
            'Запуск ітерацій')

       # self.inf_btn = QtWidgets.QPushButton('Інфіковані')
       # self.preyPredator_btn = QtWidgets.QPushButton('Хищники-жертвы')

        self.reset_values_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/arrow_undo.png'),
            'Зброс значень')
        self.clear_graph_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/chart_line_delete.png'),
            'Очистити графік')
    def ukrainian_values(self):
        self.beta_unvac_unvac_sb.setValue(0.4)
        self.beta_unvac_vac_sb.setValue(0.04)
        self.beta_vac_unvac_sb.setValue(0.2)
        self.beta_vac_vac_sb.setValue(0.02)

        self.mu_sb.setValue(0.0000418105)
        self.l_sb.setValue(0.0000181008)

        self.gamma_unvac_sb.setValue(0.0714)
        self.gamma_vac_sb.setValue(0.125)

        self.sus_unvac_sb.setValue(62.949)
        self.sus_vac_sb.setValue(36.851)

        self.inf_unvac_sb.setValue(1)
```

```
        self.inf_vac_sb.setValue(1)

        self.rec_unvac_sb.setValue(0)
        self.rec_vac_sb.setValue(0)

        self.alpha_unvac_sb.setValue(0.17858)
        self.alpha_vac_sb.setValue(0.17858)

        self.theta_vac_sb.setValue(0.0000052474)
        self.theta_unvac_sb.setValue(0.0000052474)
        self.days_sb.setValue(100)

    def reset_values(self):
        self.beta_unvac_unvac_sb.setValue(0.1)
        self.beta_unvac_vac_sb.setValue(0.1)
        self.beta_vac_unvac_sb.setValue(0.1)
        self.beta_vac_vac_sb.setValue(0.1)

        self.mu_sb.setValue(1)
        self.l_sb.setValue(1)

        self.gamma_unvac_sb.setValue(0.1)
        self.gamma_vac_sb.setValue(0.1)

        self.sus_unvac_sb.setValue(120)
        self.sus_vac_sb.setValue(100)

        self.inf_unvac_sb.setValue(20)
        self.inf_vac_sb.setValue(20)

        self.rec_unvac_sb.setValue(0)
        self.rec_vac_sb.setValue(0)

        self.alpha_unvac_sb.setValue(0.1)
        self.alpha_vac_sb.setValue(0.1)

        self.theta_vac_sb.setValue(1)
        self.theta_unvac_sb.setValue(1)
        self.days_sb.setValue(100)
       # self.timedelta_sb.setValue(0.02)

    def legend_change(self):
        self.legend_loc_cb.setEnabled(self.legend_cb.isChecked())
        self.legend_loc_lbl.setEnabled(self.legend_cb.isChecked())
# 3rd party modules
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets


class OptionsMenu_SIR(QtWidgets.QWidget):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
```

```
# Create the "SIR Coefficients" options
self.beta_sb = QtWidgets.QDoubleSpinBox()
self.gamma_sb = QtWidgets.QDoubleSpinBox()
self.t_recovery = QtWidgets.QDoubleSpinBox()
self.R0 = QtWidgets.QDoubleSpinBox()

self.gamma_sb.valueChanged.connect(self.onGammaChanged)
self.t_recovery.valueChanged.connect(self.onTrecChanged)
self.t_recovery.valueChanged.connect(self.onR0Changed)
self.R0.valueChanged.connect(self.onR0Changed)

self.beta_sb.valueChanged.connect(self.onBetaChanged)

self.R0.setEnabled(False)
self.t_recovery.setEnabled(False)

for widget in (self.beta_sb, self.gamma_sb):
    widget.setRange(0, 10)
    widget.setDecimals(4)
    widget.setSingleStep(0.001)

for widget in (self.t_recovery, self.R0):
    widget.setRange(0, 100)
    widget.setDecimals(4)
    widget.setSingleStep(0.001)

coeff_grid = QtWidgets.QGridLayout()
coeff_grid.addWidget(QtWidgets.QLabel('Константа швидкості β'), 0, 0)
coeff_grid.addWidget(self.beta_sb, 0, 1)

coeff_grid.addWidget(QtWidgets.QLabel('Швидкість одужання γ'), 1, 0)
coeff_grid.addWidget(self.gamma_sb, 1, 1)

coeff_grid.addWidget(QtWidgets.QLabel('Середній час одужання після інфекції'), 2, 0)
coeff_grid.addWidget(self.t_recovery, 2, 1)

coeff_grid.addWidget(QtWidgets.QLabel('Передаваність хвороби'), 3, 0)
coeff_grid.addWidget(self.R0, 3, 1)

coeff_gb = QtWidgets.QGroupBox('Коефіцієнти SIR-моделі:')
coeff_gb.setLayout(coeff_grid)

# Create the "Other Parameters" options
self.sus_sb = QtWidgets.QDoubleSpinBox()
self.sus_sb.setRange(0, 100000)
self.sus_sb.setSingleStep(1)

self.inf_sb = QtWidgets.QDoubleSpinBox()
self.inf_sb.setRange(0, 100000)
self.inf_sb.setSingleStep(1)

self.rec_sb = QtWidgets.QDoubleSpinBox()
```

```
        self.rec_sb.setRange(0, 100000)
        self.rec_sb.setSingleStep(1)

        self.days_sb = QtWidgets.QSpinBox()
        self.days_sb.setRange(0, 100000)
other_grid = QtWidgets.QGridLayout()
        other_grid.addWidget(QtWidgets.QLabel('Сприятливі:'), 0, 0)
        other_grid.addWidget(self.sus_sb, 0, 1)
        other_grid.addWidget(QtWidgets.QLabel('Інфіковані'), 1, 0)
        other_grid.addWidget(self.inf_sb, 1, 1)
        other_grid.addWidget(QtWidgets.QLabel('Одужавші:'), 2, 0)
        other_grid.addWidget(self.rec_sb, 2, 1)

        other_grid.addWidget(QtWidgets.QLabel('Дні:'), 3, 0)
        other_grid.addWidget(self.days_sb, 3, 1)
        #other_grid.addWidget(QtWidgets.QLabel('Час дельта:'), 4, 0)
        #other_grid.addWidget(self.timedelta_sb, 4, 1)

        other_gb = QtWidgets.QGroupBox('Інші параметри:')
        other_gb.setLayout(other_grid)

        # Create the "Graph Options" options
        self.legend_cb = QtWidgets.QCheckBox('Показати легенду:')
        self.legend_cb.setChecked(True)
        self.legend_cb.stateChanged.connect(self.legend_change)

        self.grid_cb = QtWidgets.QCheckBox('Показати сітку:')
        self.grid_cb.setChecked(True)
        self.legend_loc_lbl = QtWidgets.QLabel('Місцезнаходження легенди:')
        self.legend_loc_cb = QtWidgets.QComboBox()
        self.legend_loc_cb.addItems([x.title() for x in [
            'right',
            'center',
            'lower left',
            'center right',
            'upper left',
            'center left',
            'upper right',
            'lower right',
            'upper center',
            'lower center',
            'best',
        ]])
        self.legend_loc_cb.setCurrentIndex(6)

        cb_box = QtWidgets.QHBoxLayout()
        cb_box.addWidget(self.legend_cb)
        cb_box.addWidget(self.grid_cb)

        legend_box = QtWidgets.QHBoxLayout()
        legend_box.addWidget(self.legend_loc_cb)
        legend_box.addStretch()
```

```
        graph_box = QtWidgets.QVBoxLayout()
        graph_box.addLayout(cb_box)
        graph_box.addWidget(self.legend_loc_lbl)
        graph_box.addLayout(legend_box)

        graph_gb = QtWidgets.QGroupBox('Налаштування графіку:')
        graph_gb.setLayout(graph_box)

        # Create the update/reset buttons
        self.update_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/calculator.png'),
            'Запуск ітерацій')

       # self.totalCases_btn = QtWidgets.QPushButton('Статистика хвороб')
        #self.onlyInfectious_btn = QtWidgets.QPushButton('onlyInfectious')

       # self.inf_btn = QtWidgets.QPushButton('Інфіковані')
       # self.preyPredator_btn = QtWidgets.QPushButton('Хищники-жертвы')

        self.reset_values_btn = QtWidgets.QPushButton(
            QtGui.QIcon(':/resources/arrow_undo.png'),
            'Зброс значень')
        self.clear_graph_btn = QtWidgets.QPushButton(
        # Create the main layout
        container = QtWidgets.QVBoxLayout()
        container.addWidget(coeff_gb)
        container.addWidget(other_gb)
        container.addWidget(graph_gb)
        container.addWidget(self.update_btn)
#         container.addWidget(self.totalCases_btn)
 #        container.addWidget(self.onlyInfectious_btn)
     #  container.addStretch()
    #   container.addWidget(self.preySuperpredator_btn)
    #   container.addWidget(self.preyPredator_btn)
    #
        container.addWidget(self.reset_values_btn)
        container.addWidget(self.clear_graph_btn)
        container.addStretch()
        self.setLayout(container)

        # Populate the widgets with values
        self.reset_values()

    def onGammaChanged(self):
        if self.gamma_sb.value() == 0:
            return

        newVal = 1/self.gamma_sb.value()
        new = self.beta_sb.value() / self.gamma_sb.value()
        if newVal != self.t_recovery.value():
            self.t_recovery.setValue(newVal)
            self.R0.setValue(new)
```

```
    def onBetaChanged(self):
        if self.gamma_sb.value() == 0:
            return
        newVal = self.beta_sb.value() / self.gamma_sb.value()
        if newVal != self.gamma_sb.value():
            self.R0.setValue(newVal)

    def onR0Changed(self):
        if self.t_recovery.value() == 0:
            return
        newVal = self.R0.value() / self.t_recovery.value()
        self.beta_sb.setValue(newVal)

    def onTrecChanged(self):
        if self.t_recovery.value() == 0:
            return
        newVal = 1 / self.t_recovery.value()
        if newVal != self.gamma_sb.value():
            self.gamma_sb.setValue(newVal)

    def reset_values(self):
        self.beta_sb.setValue(0.1)
        self.gamma_sb.setValue(0.1)
        self.sus_sb.setValue(5)
        self.inf_sb.setValue(20)
        self.rec_sb.setValue(5)
        self.days_sb.setValue(100)
        #self.timedelta_sb.setValue(0.02)

    def legend_change(self):
        self.legend_loc_cb.setEnabled(self.legend_cb.isChecked())
        self.legend_loc_lbl.setEnabled(self.legend_cb.isChecked())
import numpy as np
np.seterr(all='raise')

class SEIRCalculator(object):
    def __init__(self):
        # SIR equation coefficients
      #  self.Susceptible = 20.0 #сприятливі до інфекції
      #  self.Recovered = 10.0 #очухавшиєся
        self.gamma = 1.0 #швидкість одужання
        self.beta = 1.0 #константа швидкості
        self.alpha = 1.0
      #  self.Infectious = 10.0
        # Other parameters
        #self.dt = 0.02
        self.days = 100
        self.exp = 5
        self.sus = 5
        self.inf = 4
        self.rec = 10

    def dS(self, Susceptible, Exposed, Infectious, Recovered):
```

```
        dS_dt = -(self.beta*Susceptible*Infectious)/(Susceptible+Infectious+Recovered+Exposed)
        return dS_dt

    def dE(self, Susceptible, Exposed, Infectious, Recovered):
        dE_dt = self.beta*Susceptible*Infectious/(Susceptible+Infectious+Recovered+Exposed) -
self.alpha*Exposed
        return dE_dt

    def dI(self, Susceptible, Exposed, Infectious, Recovered):
        dI_dt = self.alpha*Exposed - self.gamma*Infectious
        return dI_dt

    def dR(self, Susceptible, Exposed, Infectious, Recovered):
        dR_dt = self.gamma*Infectious
        # Calculate the predator population change
        return dR_dt

    def calculate(self):
         # import json

        susceptible_history = []
        exposed_history = []
        infectious_history = []
        recovered_history = []

        y0 = np.array([self.sus, self.exp, self.inf, self.rec], dtype='double')
        tspan = np.array([0.0, self.days], dtype='double')

        try:
            t, y = self.rk4(self.derivatives, tspan, y0, self.days)
            susceptible_history = y[:, 0]
            exposed_history = y[:, 1]
            infectious_history = y[:, 2]
            recovered_history = y[:, 3]

            print('t = ', t)
            #print(predator_history)
            #res = [dict([(ti, yi)]) for ti, yi in zip(t, predator_history)]
            #print(json.dumps(res, indent=2))
        except (RuntimeError, OverflowError, FloatingPointError):
            print("")

        return {
            'Susceptible': susceptible_history,
            'Exposed': exposed_history,
            'Infectious': infectious_history,
            'Recovered': recovered_history
        }


    def derivatives(self, t, rf):

        Susceptible = rf[0]
```

```
        Exposed = rf[1]
        Infectious = rf[2]
        Recovered = rf[3]

        dSdt = self.dS(Susceptible, Exposed, Infectious, Recovered)
        dEdt = self.dE(Susceptible, Exposed, Infectious, Recovered)
        dIdt = self.dI(Susceptible, Exposed, Infectious, Recovered)
        dRdt = self.dR(Susceptible, Exposed, Infectious, Recovered)

        drfdt = np.array([dSdt, dEdt, dIdt, dRdt], dtype='double')
        return drfdt


    def rk4(self, dydt, tspan, y0, n):
        # RK4 approximates the solution to an ODE using the RK4 method.
        #  Input:
        #    function dydt: points to a function that evaluates the right
        #                   hand side of the ODE.
        #    real tspan[2]: contains the initial and final times.
        #    real y0[m]: an array containing the initial condition.
        #    integer n: the number of steps to take.

        #  Output:
        #    real t[n+1], y[n+1,m]: the times and solution values.
        #

        if np.ndim(y0) == 0:
            m = 1
        else:
            m = len(y0)

        tfirst = tspan[0]
        tlast = tspan[1]
        dt = (tlast - tfirst) / n
        t = np.zeros(n + 1)
        y = np.zeros([n + 1, m])
        y[0, :] = y0

        for i in range(0, n):
            f1 = dydt(t[i], y[i, :])
            f2 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f1 / 2.0)
            f3 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f2 / 2.0)
            f4 = dydt(t[i] + dt, y[i, :] + dt * f3)

            t[i + 1] = t[i] + dt
            y[i + 1, :] = y[i, :] + dt * (f1 + 2.0 * f2 + 2.0 * f3 + f4) / 6.0

        return t, y
import numpy as np
np.seterr(all='raise')

class SEIRModCalculator(object):
    def __init__(self):
```

```
        # SIR equation coefficients
       #  self.Susceptible = 20.0 #сприятливі до інфекції
       #  self.Recovered = 10.0 #очухавшиєся
        self.gamma = 1.0 #швидкість одужання
        self.beta = 1.0 #константа швидкості
        self.alpha = 1.0
        self.theta = 1.0
       #  self.Infectious = 10.0
        # Other parameters
        #self.dt = 0.02
        self.days = 100
        self.exp = 5
        self.sus = 5
        self.inf = 4
        self.rec = 10

    def dS(self, Susceptible, Exposed, Infectious, Recovered):
        dS_dt = -
(self.beta*Susceptible*(Infectious+self.theta*Exposed))/(Susceptible+Infectious+Recovered+Exposed)
        return dS_dt

    def dE(self, Susceptible, Exposed, Infectious, Recovered):
        dE_dt =
self.beta*Susceptible*(Infectious+self.theta*Exposed)/(Susceptible+Infectious+Recovered+Exposed) -
self.alpha*Exposed
        return dE_dt

    def dI(self, Susceptible, Exposed, Infectious, Recovered):
        dI_dt = self.alpha*Exposed - self.gamma*Infectious
        return dI_dt

    def dR(self, Susceptible, Exposed, Infectious, Recovered):
        dR_dt = self.gamma*Infectious
        # Calculate the predator population change
        return dR_dt

    def calculate(self):
         # import json

        susceptible_history = []
        exposed_history = []
        infectious_history = []
        recovered_history = []

        y0 = np.array([self.sus, self.exp, self.inf, self.rec], dtype='double')
        tspan = np.array([0.0, self.days], dtype='double')

        try:
            t, y = self.rk4(self.derivatives, tspan, y0, self.days)
            susceptible_history = y[:, 0]
            exposed_history = y[:, 1]
            infectious_history = y[:, 2]
            recovered_history = y[:, 3]
```

```
            print('t = ', t)
            #print(predator_history)
            #res = [dict([(ti, yi)]) for ti, yi in zip(t, predator_history)]
            #print(json.dumps(res, indent=2))
        except (RuntimeError, OverflowError, FloatingPointError):
            print("")

        return {
            'Susceptible': susceptible_history,
            'Exposed': exposed_history,
            'Infectious': infectious_history,
            'Recovered': recovered_history
        }


    def derivatives(self, t, rf):

        Susceptible = rf[0]
        Exposed = rf[1]
        Infectious = rf[2]
        Recovered = rf[3]

        dSdt = self.dS(Susceptible, Exposed, Infectious, Recovered)
        dEdt = self.dE(Susceptible, Exposed, Infectious, Recovered)
        dIdt = self.dI(Susceptible, Exposed, Infectious, Recovered)
        dRdt = self.dR(Susceptible, Exposed, Infectious, Recovered)

        drfdt = np.array([dSdt, dEdt, dIdt, dRdt], dtype='double')
        return drfdt


if np.ndim(y0) == 0:
            m = 1
        else:
            m = len(y0)

        tfirst = tspan[0]
        tlast = tspan[1]
        dt = (tlast - tfirst) / n
        t = np.zeros(n + 1)
        y = np.zeros([n + 1, m])
        y[0, :] = y0

        for i in range(0, n):
            f1 = dydt(t[i], y[i, :])
            f2 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f1 / 2.0)
            f3 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f2 / 2.0)
            f4 = dydt(t[i] + dt, y[i, :] + dt * f3)

            t[i + 1] = t[i] + dt
            y[i + 1, :] = y[i, :] + dt * (f1 + 2.0 * f2 + 2.0 * f3 + f4) / 6.0
```

```
        return t, y
#!/usr/bin/env python

# Python standard library modules
import sys

# 3rd party modules
import matplotlib
import matplotlib.backends.backend_qt5agg as backend_qt5agg
from matplotlib.figure import Figure
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets

#
class AppForm_SEIR_mod(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)

        # название окна
        self.setWindowTitle(APP_NAME)

        # Создание в меню параметров в виджете док-станции
        self.options_menu = OptionsMenu_SEIR_mod()
        dock = QtWidgets.QDockWidget('Налаштування коефіцієнтів', self)
        dock.setFeatures(
            QtWidgets.QDockWidget.NoDockWidgetFeatures |
            QtWidgets.QDockWidget.DockWidgetMovable |
            QtWidgets.QDockWidget.DockWidgetFloatable
        )
        dock.setAllowedAreas(
            QtCore.Qt.LeftDockWidgetArea | QtCore.Qt.RightDockWidgetArea,
        )
        dock.setWidget(self.options_menu)
        self.addDockWidget(QtCore.Qt.LeftDockWidgetArea, dock)

        # Подключение сигналов из меню параметров
        self.options_menu.update_btn.clicked.connect(self.clear_graph)
        self.options_menu.clear_graph_btn.clicked.connect(self.clear_graph)
        self.options_menu.legend_cb.stateChanged.connect(self.redraw_graph)
        self.options_menu.grid_cb.stateChanged.connect(self.redraw_graph)
        self.options_menu.legend_loc_cb.currentIndexChanged.connect(self.redraw_graph)

        # создание графика
        fig = Figure((7.0, 3.0), dpi=100)
        self.canvas = backend_qt5agg.FigureCanvasQTAgg(fig)
        self.canvas.setParent(self)
        self.axes = fig.add_subplot(111)
        backend_qt5agg.NavigationToolbar2QT(self.canvas, self.canvas)
        about_action.setToolTip('About')
        about_action.setIcon(QtGui.QIcon(':/resources/icon_info.png'))
        about_action.triggered.connect(self.show_about)
```

```
    sir_action = QtWidgets.QAction('&SEIRmod', self)
    sir_action.setToolTip('SEIRmod')
    sir_action.triggered.connect(self.show_seir_mod)

  # создать менюбар
    file_exit_action = QtWidgets.QAction('&Exit', self)
    file_exit_action.setToolTip('Exit')
    file_exit_action.setIcon(QtGui.QIcon(':/resources/door_open.png'))
    file_exit_action.triggered.connect(self.close)

    file_menu = self.menuBar().addMenu('&Exit')
    file_menu.addAction(file_exit_action)

    help_menu = self.menuBar().addMenu('&Help')
    help_menu.addAction(about_action)

    sir_menu = self.menuBar().addMenu('&SEIRmod')
    sir_menu.addAction(sir_action)


def calculate_data(self):
    # объект GrowthCalculator
    growth = SEIRModCalculator()

    # Update the GrowthCalculator parameters from the GUI options
    growth.gamma = self.options_menu.gamma_sb.value()
    growth.beta = self.options_menu.beta_sb.value()
    growth.alpha = self.options_menu.alpha_sb.value()
    growth.theta = self.options_menu.theta_sb.value()
    growth.sus = self.options_menu.sus_sb.value()
    growth.exp = self.options_menu.exp_sb.value()
    growth.inf = self.options_menu.inf_sb.value()
    growth.rec = self.options_menu.rec_sb.value()

    growth.days = self.options_menu.days_sb.value()
    #growth.dt = self.options_menu.timedelta_sb.value()

    # Calculate the population growths
    results = growth.calculate()
    self.infectious_history.extend(results['Infectious'])
    self.exposed_history.extend(results['Exposed'])
    self.susceptible_history.extend(results['Susceptible'])
    self.recovered_history.extend(results['Recovered'])

    if (len(self.infectious_history) == 0 and
            len(self.susceptible_history) == 0 and
            len(self.recovered_history) == 0 and
            len(self.exposed_history) == 0):
        QtWidgets.QMessageBox.information(self, 'Error', 'Помилка')

def clear_graph(self):
    # очистить историю популяций
```

```
        self.infectious_history = []
        self.exposed_history = []
        self.susceptible_history = []
        self.recovered_history = []

        # перерисовать граф
        self.redraw_graph()

    def redraw_graph(self):
        # очистить граф
        self.axes.clear()

        # Create the graph labels
        self.axes.set_title('modified SEIR-model: Susceptible, Exposed, Infectious, Recovered')
        self.axes.set_xlabel('Ітерації')
        self.axes.set_ylabel('Кількість населення')

        # Plot the current population data

        if self.susceptible_history:
            self.axes.plot(self.susceptible_history, 'b-', label='ще не хворіли')
        if self.exposed_history:
            self.axes.plot(self.exposed_history, 'y-', label='з вірусом в інкубаційній формі')
        if self.infectious_history:
            self.axes.plot(self.infectious_history, 'r-', label='хворі')
        if self.recovered_history:
            self.axes.plot(self.recovered_history, 'g-', label='перехворілі')
        # если нужно, создаём легенду
        if self.options_menu.legend_cb.isChecked():
            if self.recovered_history or self.susceptible_history or self.infectious_history or self.exposed_history:
                legend_loc = str(
                    self.options_menu.legend_loc_cb.currentText()
                ).lower()
                legend = matplotlib.font_manager.FontProperties(size=10)
                self.axes.legend(loc=legend_loc, prop=legend)

        # если нужно, сетки на графике
        self.axes.grid(self.options_menu.grid_cb.isChecked())

        # рисуем график
        self.canvas.draw()

    def show_seir_mod(self):

        message = '''<font size="+2">SEIR-модель</font>
            <p>dS/dt = -βSI/N
            <p>dE/dt = βSI/N - αE
            <p>dI/dt = αE - ɣI
            <p>dR/dt = ɣI
            <p>де
            <p>S — кількість сприятливих до вірусу;
            <p>E — кількість населення із хворобою у інкубаційному періоді;
            <p>I — кількість інфікованих;
```

```
            <p>R — кількість переболівших;
            <p>Θ — характеризує ступінь заразності латентних носіїв у порівнянні із захворілими;
            <p>β — константа швидкості;
            <p>α — швидкість переходу хвороби від інкубаційної стадії до відкритої;
            <p>ɣ — швидкість одужання.
            '''
        QtWidgets.QMessageBox.about(self, 'SEIR modified' + '', message)

    def show_about(self):

        message = '''<font size="+2">%s</font>
            <p>Магістерська дисертація на тему "моделювання епідеміологічних моделей".
            <p>Написана by %s, група КА-13мп.
            ''' %(APP_NAME, AUTHOR)

        QtWidgets.QMessageBox.about(self, 'About' + APP_NAME, message)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(':/resources/icon.svg'))
    form = AppForm_SEIR_mod()
    form.show()
    app.exec_()
#!/usr/bin/env python

# Python standard library modules
import sys

# 3rd party modules
import matplotlib
import matplotlib.backends.backend_qt5agg as backend_qt5agg
from matplotlib.figure import Figure
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets

# Local application modules
APP_NAME = 'Epidemics'
AUTHOR = 'Клименко Анастасія'

class AppForm_SEIR(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)

        self.options_menu = OptionsMenu_SEIR()
        dock = QtWidgets.QDockWidget('Налаштування коефіцієнтів', self)
        dock.setFeatures(
            QtWidgets.QDockWidget.NoDockWidgetFeatures |
            QtWidgets.QDockWidget.DockWidgetMovable |
            QtWidgets.QDockWidget.DockWidgetFloatable
        )
        dock.setAllowedAreas(
            QtCore.Qt.LeftDockWidgetArea | QtCore.Qt.RightDockWidgetArea,
```

```
    )
    dock.setWidget(self.options_menu)
    self.addDockWidget(QtCore.Qt.LeftDockWidgetArea, dock)
    self.options_menu.update_btn.clicked.connect(self.clear_graph)
    self.options_menu.update_btn.clicked.connect(self.calculate_data)
    self.options_menu.legend_cb.stateChanged.connect(self.redraw_graph)
    self.options_menu.grid_cb.stateChanged.connect(self.redraw_graph)
    self.options_menu.legend_loc_cb.currentIndexChanged.connect(self.redraw_graph)
    self.canvas.setParent(self)
    self.axes = fig.add_subplot(111)
    backend_qt5agg.NavigationToolbar2QT(self.canvas, self.canvas)
    self.clear_graph()

    self.setCentralWidget(self.canvas)
    about_action = QtWidgets.QAction('&About', self)
    about_action.setToolTip('About')
    about_action.setIcon(QtGui.QIcon(':/resources/icon_info.png'))
    about_action.triggered.connect(self.show_about)

    sir_action = QtWidgets.QAction('&SEIR', self)
    sir_action.setToolTip('SEIR')
    sir_action.triggered.connect(self.show_seir)

 # создать менюбар
    file_exit_action = QtWidgets.QAction('&Exit', self)
    file_exit_action.setToolTip('Exit')
    file_exit_action.setIcon(QtGui.QIcon(':/resources/door_open.png'))
    file_exit_action.triggered.connect(self.close)

    file_menu = self.menuBar().addMenu('&Exit')
    file_menu.addAction(file_exit_action)

    help_menu = self.menuBar().addMenu('&Help')
    help_menu.addAction(about_action)

    sir_menu = self.menuBar().addMenu('&SEIR')
    sir_menu.addAction(sir_action)


def calculate_data(self):
    growth = SEIRCalculator()
    growth.gamma = self.options_menu.gamma_sb.value()
    growth.beta = self.options_menu.beta_sb.value()
    growth.alpha = self.options_menu.alpha_sb.value()
    growth.sus = self.options_menu.sus_sb.value()
    growth.exp = self.options_menu.exp_sb.value()
    growth.inf = self.options_menu.inf_sb.value()
    growth.rec = self.options_menu.rec_sb.value()

    growth.days = self.options_menu.days_sb.value()
    #growth.dt = self.options_menu.timedelta_sb.value()

    # Calculate the population growths
```

```
    results = growth.calculate()
    self.infectious_history.extend(results['Infectious'])
    self.exposed_history.extend(results['Exposed'])
    self.susceptible_history.extend(results['Susceptible'])
    self.recovered_history.extend(results['Recovered'])

    if (len(self.infectious_history) == 0 and
            len(self.susceptible_history) == 0 and
            len(self.recovered_history) == 0 and
            len(self.exposed_history)==0):
        QtWidgets.QMessageBox.information(self, 'Error', 'Помилка')
        return
def clear_graph(self):
    # очистить историю популяций
    self.infectious_history = []
    self.exposed_history = []
    self.susceptible_history = []
    self.recovered_history = []

    # перерисовать граф
    self.redraw_graph()

def redraw_graph(self):
    # очистить граф
    self.axes.clear()

    # Create the graph labels
    self.axes.set_title('SEIR-model: Susceptible, Exposed, Infectious, Recovered')
    self.axes.set_xlabel('Ітерації')
    self.axes.set_ylabel('Кількість населення')

    # Plot the current population data

    if self.susceptible_history:
        self.axes.plot(self.susceptible_history, 'b-', label='ще не хворіли')
    if self.exposed_history:
        self.axes.plot(self.exposed_history, 'y-', label='з вірусом в інкубаційній формі')
    if self.infectious_history:
        self.axes.plot(self.infectious_history, 'r-', label='хворі')
    if self.recovered_history:
        self.axes.plot(self.recovered_history, 'g-', label='перехворілі')
    # если нужно, создаём легенду
    if self.options_menu.legend_cb.isChecked():
        if self.recovered_history or self.susceptible_history or self.infectious_history or self.exposed_history:
            legend_loc = str(
                self.options_menu.legend_loc_cb.currentText()
            ).lower()
            legend = matplotlib.font_manager.FontProperties(size=10)
            self.axes.legend(loc=legend_loc, prop=legend)

    # если нужно, сетки на графике
    self.axes.grid(self.options_menu.grid_cb.isChecked())
```

```
        # рисуем график
        self.canvas.draw()

    def show_seir(self):

        message = '''<font size="+2">SEIR-модель</font>
            <p>dS/dt = -βSI/N
            <p>dE/dt = βSI/N - αE
            <p>dI/dt = αE - ɣI
            <p>dR/dt = ɣI
            <p>де
            <p>S — кількість сприятливих до вірусу;
            <p>E — кількість населення із хворобою у інкубаційному періоді;
            <p>I — кількість інфікованих;
            <p>R — кількість переболівших;
            <p>β — константа швидкості;
            <p>α — швидкість переходу хвороби від інкубаційної стадії до відкритої;
            <p>ɣ — швидкість одужання.
            '''
        QtWidgets.QMessageBox.about(self, 'SEIR' + '', message)

    def show_about(self):

        message = '''<font size="+2">%s</font>
            <p> Магістерська дисертація на тему "моделювання епідеміологічних моделей".
            <p>Написана by %s, група КА-13мп.
            ''' %(APP_NAME, AUTHOR)

        QtWidgets.QMessageBox.about(self, 'About' + APP_NAME, message)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(':/resources/icon.svg'))
    form = AppForm_SEIR()
    form.show()
    app.exec_()
import numpy as np
np.seterr(all='raise')



    def dS(self, Susceptible, Exposed, Infectious, Recovered):
        dS_dt = -(self.beta*Susceptible*Infectious)/(Susceptible+Infectious+Recovered+Exposed)
        return dS_dt

    def dE(self, Susceptible, Exposed, Infectious, Recovered):
        dE_dt = (1-self.u)*self.beta*Susceptible*Infectious/(Susceptible+Infectious+Recovered+Exposed) -
self.alpha*Exposed
        return dE_dt

    def dI(self, Susceptible, Exposed, Infectious, Recovered):
        dI_dt = self.alpha*Exposed - self.gamma*Infectious
        return dI_dt
```

```
def dR(self, Susceptible, Exposed, Infectious, Recovered):
    dR_dt = self.gamma*Infectious
    # Calculate the predator population change
    return dR_dt

def calculate(self):
     # import json

    susceptible_history = []
    exposed_history = []
    infectious_history = []
    recovered_history = []

    y0 = np.array([self.sus, self.exp, self.inf, self.rec], dtype='double')
    tspan = np.array([0.0, self.days], dtype='double')

    try:
        t, y = self.rk4(self.derivatives, tspan, y0, self.days)
        susceptible_history = y[:, 0]
        exposed_history = y[:, 1]
        infectious_history = y[:, 2]
        recovered_history = y[:, 3]

        print('t = ', t)
        #print(predator_history)
        #res = [dict([(ti, yi)]) for ti, yi in zip(t, predator_history)]
        #print(json.dumps(res, indent=2))
    except (RuntimeError, OverflowError, FloatingPointError):
        print("")

    return {
        'Susceptible': susceptible_history,
        'Exposed': exposed_history,
        'Infectious': infectious_history,
        'Recovered': recovered_history
    }


def derivatives(self, t, rf):

    Susceptible = rf[0]
    Exposed = rf[1]
    Infectious = rf[2]
    Recovered = rf[3]

    dSdt = self.dS(Susceptible, Exposed, Infectious, Recovered)
    dEdt = self.dE(Susceptible, Exposed, Infectious, Recovered)
    dIdt = self.dI(Susceptible, Exposed, Infectious, Recovered)
    dRdt = self.dR(Susceptible, Exposed, Infectious, Recovered)

    drfdt = np.array([dSdt, dEdt, dIdt, dRdt], dtype='double')
    return drfdt
```

```
    def rk4(self, dydt, tspan, y0, n):
            m = len(y0)

        tfirst = tspan[0]
        tlast = tspan[1]
        dt = (tlast - tfirst) / n

import sys

# 3rd party modules
import matplotlib
import matplotlib.backends.backend_qt5agg as backend_qt5agg
from matplotlib.figure import Figure
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets

# Local application modules
from SEIR_vacc_calculator import SEIRvaccCalculator
from OptionsMenu_SEIR_vacc import OptionsMenu_SEIR_vacc
#from options_menu_4_species import OptionsMenu_4_species
#from options_menu_2 import OptionsMenu_2_species

#import resources

APP_NAME = 'Epidemics'
AUTHOR = 'Клименко Анастасія'

class AppForm_SEIR_vacc(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)
        self.setWindowTitle(APP_NAME)
        self.options_menu = OptionsMenu_SEIR_vacc()
        dock = QtWidgets.QDockWidget('Налаштування коефіцієнтів', self)
        dock.setFeatures(
            QtWidgets.QDockWidget.NoDockWidgetFeatures |
            QtWidgets.QDockWidget.DockWidgetMovable |
            QtWidgets.QDockWidget.DockWidgetFloatable
        )
        dock.setAllowedAreas(
            QtCore.Qt.LeftDockWidgetArea | QtCore.Qt.RightDockWidgetArea,
        )
        dock.setWidget(self.options_menu)
        self.addDockWidget(QtCore.Qt.LeftDockWidgetArea, dock)
        self.options_menu.update_btn.clicked.connect(self.clear_graph)
        self.options_menu.update_btn.clicked.connect(self.calculate_data)
        self.options_menu.legend_cb.stateChanged.connect(self.redraw_graph)
        self.options_menu.grid_cb.stateChanged.connect(self.redraw_graph)
        self.options_menu.legend_loc_cb.currentIndexChanged.connect(self.redraw_graph)
        fig = Figure((7.0, 3.0), dpi=100)
        self.canvas = backend_qt5agg.FigureCanvasQTAgg(fig)
```

```
    self.canvas.setParent(self)
    self.axes = fig.add_subplot(111)
    backend_qt5agg.NavigationToolbar2QT(self.canvas, self.canvas)
    self.clear_graph()
    self.setCentralWidget(self.canvas)
    about_action = QtWidgets.QAction('&About', self)
    about_action.setToolTip('About')
    about_action.setIcon(QtGui.QIcon(':/resources/icon_info.png'))
    about_action.triggered.connect(self.show_about)

    sir_action = QtWidgets.QAction('&SEIR vaccined', self)
    sir_action.setToolTip('SEIR vaccined')
    sir_action.triggered.connect(self.show_seir)
    file_exit_action.setToolTip('Exit')
    file_exit_action.setIcon(QtGui.QIcon(':/resources/door_open.png'))
    file_exit_action.triggered.connect(self.close)

    file_menu = self.menuBar().addMenu('&Exit')
    file_menu.addAction(file_exit_action)

    help_menu = self.menuBar().addMenu('&Help')
    help_menu.addAction(about_action)

    sir_menu = self.menuBar().addMenu('&SEIR vaccined')
    sir_menu.addAction(sir_action)


def calculate_data(self):
    # объект GrowthCalculator
    growth = SEIRvaccCalculator()

    # Update the GrowthCalculator parameters from the GUI options
    growth.gamma = self.options_menu.gamma_sb.value()
    growth.beta = self.options_menu.beta_sb.value()
    growth.alpha = self.options_menu.alpha_sb.value()
    growth.u = self.options_menu.u_sb.value()
    growth.sus = self.options_menu.sus_sb.value()
    growth.exp = self.options_menu.exp_sb.value()
    growth.inf = self.options_menu.inf_sb.value()
    growth.rec = self.options_menu.rec_sb.value()

    growth.days = self.options_menu.days_sb.value()
    #growth.dt = self.options_menu.timedelta_sb.value()

    # Calculate the population growths
    results = growth.calculate()
    self.infectious_history.extend(results['Infectious'])
    self.exposed_history.extend(results['Exposed'])
    self.susceptible_history.extend(results['Susceptible'])
    self.recovered_history.extend(results['Recovered'])

    if (len(self.infectious_history) == 0 and
            len(self.susceptible_history) == 0 and
```

```
            len(self.recovered_history) == 0 and
            len(self.exposed_history)==0):
        QtWidgets.QMessageBox.information(self, 'Error', 'Помилка')
        return

    # последнее в векторе количество популяции на панель инструментов параметров
    #  print('self.predator_history[-1]', self.predator_history[-1])
    #  self.options_menu.predator_sb.setValue(self.predator_history[-1])
    #  self.options_menu.prey_sb.setValue(self.prey_history[-1])
    #  self.options_menu.superpredators_sb.setValue(self.superpredator_history[-1])
    if self.options_menu.legend_cb.isChecked():
        if self.recovered_history or self.susceptible_history or self.infectious_history or self.exposed_history:
            legend_loc = str(
                self.options_menu.legend_loc_cb.currentText()
            ).lower()
            legend = matplotlib.font_manager.FontProperties(size=10)
            self.axes.legend(loc=legend_loc, prop=legend)

    # если нужно, сетки на графике
    self.axes.grid(self.options_menu.grid_cb.isChecked())

    # рисуем график
    self.canvas.draw()

def show_seir(self):

    message = '''<font size="+2">SEIR-модель з вакцинацією</font>
        <p>dS/dt = -βSI/N
        <p>dE/dt = (1-u)βSI/N - αE
        <p>dI/dt = αE - ɣI
        <p>dR/dt = ɣI
        <p>де
        <p>S — кількість сприятливих до вірусу;
        <p>E — кількість населення із хворобою у інкубаційному періоді;
        <p>I — кількість інфікованих;
        <p>R — кількість переболівших;
        <p>β — константа швидкості;
        <p>u — ефективність втручань громадської охорони здоров'я;
        <p>α — швидкість переходу хвороби від інкубаційної стадії до відкритої;
        <p>ɣ — швидкість одужання.
        '''
    QtWidgets.QMessageBox.about(self, 'SEIR vaccined' + '', message)

def show_about(self):

    message = '''<font size="+2">%s</font>
        <p> Магістерська дисертація на тему "моделювання епідеміологічних моделей".
        <p>Написана by %s, група КА-13мп.
        ''' %(APP_NAME, AUTHOR)

    QtWidgets.QMessageBox.about(self, 'About' + APP_NAME, message)

if __name__ == "__main__":
```

```
    app = QtWidgets.QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(':/resources/icon.svg'))
import matplotlib
import matplotlib.backends.backend_qt5agg as backend_qt5agg
from matplotlib.figure import Figure
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets
from PyQt5.QtWidgets import QLabel
from PyQt5.QtGui import QPixmap
#from PyQt5.QtWidgets import QLabel
from PyQt5.QtWidgets import QHBoxLayout
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget
# Local application modules
from SEIRD_full_with_vacc_calculator import SEIRDvaccCalculator
from OptionsMenu_SEIRD_full_with_vacc import OptionsMenu_SEIRD_full_with_vacc
#from options_menu_4_species import OptionsMenu_4_species
#from options_menu_2 import OptionsMenu_2_species

#import resources

APP_NAME = 'Epidemics'
AUTHOR = 'Клименко Анастасія'

class AppForm_SEIRD_vacc(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)

        # название окна
        self.setWindowTitle(APP_NAME)

        # Создание в меню параметров в виджете док-станции
        self.options_menu = OptionsMenu_SEIRD_full_with_vacc()
        dock = QtWidgets.QDockWidget('Налаштування коефіцієнтів', self)
        dock.setFeatures(
            QtWidgets.QDockWidget.NoDockWidgetFeatures |
            QtWidgets.QDockWidget.DockWidgetMovable |
            QtWidgets.QDockWidget.DockWidgetFloatable
        )
        dock.setAllowedAreas(
            QtCore.Qt.LeftDockWidgetArea | QtCore.Qt.RightDockWidgetArea,
        )

        self.scrollArea = QtWidgets.QScrollArea()
        self.scrollArea.setWidget(self.options_menu)
        dock.setWidget(self.scrollArea)
        self.addDockWidget(QtCore.Qt.LeftDockWidgetArea, dock)

        # Подключение сигналов из меню параметров
        self.options_menu.update_btn.clicked.connect(self.clear_graph)
        self.options_menu.update_btn.clicked.connect(self.calculate_data)
        self.options_menu.legend_cb.stateChanged.connect(self.redraw_graph)
```

```
        self.options_menu.grid_cb.stateChanged.connect(self.redraw_graph)
        self.options_menu.legend_loc_cb.currentIndexChanged.connect(self.redraw_graph)

        # создание графика
        fig = Figure((7.0, 3.0), dpi=100)
        self.canvas = backend_qt5agg.FigureCanvasQTAgg(fig)
        self.canvas.setParent(self)
        self.axes = fig.add_subplot(111)
        backend_qt5agg.NavigationToolbar2QT(self.canvas, self.canvas)

        # инициализация графа
        self.clear_graph()

        self.setCentralWidget(self.canvas)

        # создать менюбар действий

        about_action = QtWidgets.QAction('&About', self)
        about_action.setToolTip('About')
        about_action.setIcon(QtGui.QIcon(':/resources/icon_info.png'))
        about_action.triggered.connect(self.show_about)

        sir_action = QtWidgets.QAction('&SEIRD vaccined', self)
        sir_action.setToolTip('SEIRD vaccined')
        sir_action.triggered.connect(self.show_seird)

    # создать менюбар
        file_exit_action = QtWidgets.QAction('&Exit', self)
        file_exit_action.setToolTip('Exit')
        file_exit_action.setIcon(QtGui.QIcon(':/resources/door_open.png'))
        file_exit_action.triggered.connect(self.close)

        file_menu = self.menuBar().addMenu('&Exit')
        file_menu.addAction(file_exit_action)

        help_menu = self.menuBar().addMenu('&Help')
        help_menu.addAction(about_action)

        sir_menu = self.menuBar().addMenu('&SEIRD vaccined')
        sir_menu.addAction(sir_action)



        growth.mu = self.options_menu.mu_sb.value()
        growth.l = self.options_menu.l_sb.value()

        growth.theta_vac = self.options_menu.theta_vac_sb.value()
        growth.theta_unvac = self.options_menu.theta_unvac_sb.value()

        growth.sus_unvac = self.options_menu.sus_unvac_sb.value()
        growth.sus_vac = self.options_menu.sus_vac_sb.value()

        growth.exp_unvac = self.options_menu.exp_unvac_sb.value()
```

```
        growth.exp_vac = self.options_menu.exp_vac_sb.value()

        growth.inf_unvac = self.options_menu.inf_unvac_sb.value()
        growth.inf_vac = self.options_menu.inf_vac_sb.value()

        growth.rec_unvac = self.options_menu.rec_unvac_sb.value()
        growth.rec_vac = self.options_menu.rec_vac_sb.value()

        growth.dead = self.options_menu.dead_sb.value()

        growth.days = self.options_menu.days_sb.value()
        #growth.dt = self.options_menu.timedelta_sb.value()

        # Calculate the population growths
        results = growth.calculate()
        self.infectious_unvac_history.extend(results['Infectious_unvac'])
        self.infectious_vac_history.extend(results['Infectious_vac'])
        self.exposed_unvac_history.extend(results['Exposed_unvac'])
        self.exposed_vac_history.extend(results['Exposed_vac'])

        self.susceptible_unvac_history.extend(results['Susceptible_unvac'])
        self.susceptible_vac_history.extend(results['Susceptible_vac'])

        self.recovered_unvac_history.extend(results['Recovered_unvac'])
        self.recovered_vac_history.extend(results['Recovered_vac'])

        self.dead_history.extend(results['Dead'])

        if (len(self.infectious_unvac_history) == 0 and
            len(self.infectious_vac_history) == 0 and
            len(self.exposed_unvac_history) == 0 and
            len(self.exposed_vac_history) == 0 and
            len(self.susceptible_unvac_history) == 0 and
            len(self.susceptible_vac_history) == 0 and
            len(self.recovered_unvac_history) == 0 and
            len(self.recovered_vac_history) == 0 and
            len(self.dead_history) == 0):
          QtWidgets.QMessageBox.information(self, 'Error', 'Помилка')
          return

    def clear_graph(self):
        # очистить историю популяций
        self.susceptible_unvac_history = []
        self.susceptible_vac_history = []
        self.exposed_unvac_history = []
        self.exposed_vac_history = []
        self.infectious_unvac_history = []
        self.infectious_vac_history = []
        self.recovered_unvac_history = []
        self.recovered_vac_history = []
        self.dead_history = []

        # перерисовать граф
```

```
        self.redraw_graph()

    def redraw_graph(self):
        # очистить граф
        self.axes.clear()

        # Create the graph labels
        self.axes.set_title('SEIRD-model with vaccine: Susceptible, Exposed, Infectious, Recovered, Dead')
        self.axes.set_xlabel('Дні')
        self.axes.set_ylabel('Кількість населення')

        # Plot the current population data

        if self.susceptible_unvac_history:
            self.axes.plot(self.susceptible_unvac_history, 'firebrick', label='ще не хворілі невакциновані')
        if self.susceptible_vac_history:
            self.axes.plot(self.susceptible_vac_history, 'red', label='ще не хворілі вакциновані')

        if self.exposed_unvac_history:
            self.axes.plot(self.exposed_unvac_history, 'gold', label='невакциновані із вірусом в інкубаційній
формі')
        if self.exposed_vac_history:
            self.axes.plot(self.exposed_vac_history, 'orange', label='вакциновані із вірусом в інкубаційній
формі')

        if self.infectious_unvac_history:
            self.axes.plot(self.infectious_unvac_history, 'royalblue', label='хворі невакциновані')
        if self.infectious_vac_history:
            self.axes.plot(self.infectious_vac_history, 'navy', label='хворі вакциновані')

        if self.recovered_unvac_history:
            self.axes.plot(self.recovered_unvac_history, 'green', label='перехворівші невакциновані')
        if self.recovered_vac_history:
            self.axes.plot(self.recovered_vac_history, 'lime', label='перехворівші вакциновані')
        if self.dead_history:
            self.axes.plot(self.dead_history, 'black', label='померли')

        # если нужно, создаём легенду
        if self.options_menu.legend_cb.isChecked():
            if self.susceptible_unvac_history or self.susceptible_vac_history or self.exposed_unvac_history or \
                    self.exposed_vac_history or self.infectious_unvac_history or self.infectious_vac_history or \
                    self.recovered_unvac_history or self.recovered_vac_history or self.dead_history:

                legend_loc = str(
                    self.options_menu.legend_loc_cb.currentText()
                ).lower()
                legend = matplotlib.font_manager.FontProperties(size=10)
                self.axes.legend(loc=legend_loc, prop=legend)

        # если нужно, сетки на графике
        self.axes.grid(self.options_menu.grid_cb.isChecked())

        # рисуем график
```

```
        self.canvas.draw()

    def show_seird(self):

        message = '''<font size="+2">SEIRD-модель з вакцинацією</font>
            <p>dS<font size="-5">unvac</font>/dt = l - μ*S<font size="-5">unvac</font> - S<font size="-5">unvac</font>(β<font size="-5">unvac->unvac</font>*I<font size="-5">unvac</font>
            + β<font size="-5">unvac->vac</font>*I<font size="-5">vac</font>)/N
            <p>dS<font size="-5">vac</font>/dt = - μ*S<font size="-5">vac</font> - S<font size="-5">vac</font>
            (β<font size="-5">vac->unvac</font>*I<font size="-5">vac</font>
            + β<font size="-5">vac->vac</font>*I<font size="-5">vac</font>)/N

            <p>dE<font size="-5">unvac</font>/dt = S<font size="-5">unvac</font>(β<font size="-5">unvac->unvac</font>*I<font size="-5">unvac</font>
            + β<font size="-5">unvac->vac</font>*I<font size="-5">vac</font>)/N - (μ+α<font size="-5">unvac</font>)E<font size="-5">unvac</font>
            <p>dE<font size="-5">vac</font>/dt = S<font size="-5">vac</font>(β<font size="-5">unvac->vac</font>*I<font size="-5">vac</font>
            + β<font size="-5">vac->vac</font>*I<font size="-5">vac</font>)/N - (μ+α<font size="-5">vac</font>)E<font size="-5">vac</font>

            <p>dI<font size="-5">unvac</font>/dt = α<font size="-5">unvac</font>E<font size="-5">unvac</font> - (ɣ<font size="-5">unvac</font> + μ + θ<font size="-5">unvac</font>)I<font size="-5">unvac</font>
            <p>dI<font size="-5">vac</font>/dt = α<font size="-5">vac</font>E<font size="-5">vac</font> - (ɣ<font size="-5">vac</font> + μ + θ<font size="-5">vac</font>)I<font size="-5">vac</font>

            <p>dR<font size="-5">unvac</font>/dt = ɣ<font size="-5">unvac</font>I<font size="-5">unvac</font> - μR<font size="-5">unvac</font>
            <p>dR<font size="-5">vac</font>/dt = ɣ<font size="-5">vac</font>I<font size="-5">vac</font> - μR<font size="-5">vac</font>

            <p>dD/dt = θ<font size="-5">unvac</font>I<font size="-5">unvac</font> + θ<font size="-5">vac</font>I<font size="-5">vac</font> +
            μ(S<font size="-5">unvac</font> + S<font size="-5">vac</font> + E<font size="-5">unvac</font> + E<font size="-5">vac</font> +
            I<font size="-5">unvac</font> + I<font size="-5">vac</font> + I<font size="-5">unvac</font> + I<font size="-5">vac</font> +
            R<font size="-5">unvac</font> + R<font size="-5">vac</font>)


            <p>де
            <p>S — кількість сприятливих до вірусу;
            <p>E — кількість населення із хворобою у інкубаційному періоді;
            <p>I — кількість інфікованих;
            <p>R — кількість переболівших;
            <p>β — константа швидкості;
            <p>u — ефективність втручань громадської охорони здоров'я;
            <p>α — швидкість переходу хвороби від інкубаційної стадії до відкритої;
            <p>ɣ — швидкість одужання.
            '''
        QtWidgets.QMessageBox.about(self, 'SEIRD vaccined', message)
```

```
    def show_about(self):

        message = '''<font size="+2">%s</font>
            <p> Магістерська дисертація на тему "моделювання епідеміологічних моделей".
            <p>Написана by %s, група КА-13мп.
            ''' %(APP_NAME, AUTHOR)

        QtWidgets.QMessageBox.about(self, 'About' + APP_NAME, message)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(':/resources/icon.svg'))
    form = AppForm_SEIRD_vacc()
    form.show()
    app.exec_()
import numpy as np
np.seterr(all='raise')

class SEIRDvaccCalculator(object):
    def __init__(self):
        # SIR equation coefficients
      #  self.Susceptible = 20.0 #сприятливі до інфекції
      #  self.Recovered = 10.0 #очухавшиєся
        self.gamma_unvac = 1.0 #швидкість одужання
        self.gamma_vacc = 1.0

        self.beta_unvac_unvac = 1.0 #константа швидкості
        self.beta_unvac_vac = 1.0
        self.beta_vac_unvac = 1.0
        self.beta_vac_vac = 1.0

        self.alpha_unvac = 1.0
        self.alpha_vac = 1.0

        self.theta_unvac = 1.0
        self.theta_vac = 1.0

        self.mu = 1.0
        self.l = 1.0

      #  self.Infectious = 10.0
        # Other parameters
        #self.dt = 0.02

        self.days = 100

        self.exp_vac = 5
        self.exp_unvac = 5
        self.sus_unvac = 5
        self.sus_vac = 5
        self.inf_unvac = 4
        self.inf_vac = 4
```

```
        self.rec_unvac = 0
        self.rec_vac = 0
        self.dead = 0

    def dS_unvac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dS_unvac_dt = self.l - self.mu*Susceptible_unvac \
                - (self.beta_unvac_unvac*Infectious_unvac +
self.beta_unvac_vac*Infectious_vac)*Susceptible_unvac\

/(Susceptible_unvac+Susceptible_vac+Exposed_unvac+Exposed_vac+Infectious_unvac+Infectious_vac+Rec
overed_unvac+Recovered_vac)
        return dS_unvac_dt

    def dS_vac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dS_vac_dt = - self.mu*Susceptible_vac \
                - (self.beta_vac_unvac*Infectious_unvac + self.beta_vac_vac*Infectious_vac)*Susceptible_vac\

/(Susceptible_unvac+Susceptible_vac+Exposed_unvac+Exposed_vac+Infectious_unvac+Infectious_vac+Rec
overed_unvac+Recovered_vac)
        return dS_vac_dt

    def dE_unvac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dE_unvac_dt = (self.beta_unvac_unvac*Infectious_unvac +
self.beta_unvac_vac*Infectious_vac)*Susceptible_unvac\

/(Susceptible_unvac+Susceptible_vac+Exposed_unvac+Exposed_vac+Infectious_unvac+Infectious_vac+Rec
overed_unvac+Recovered_vac) \
                - (self.mu + self.alpha_unvac)*Exposed_unvac
        return dE_unvac_dt

    def dE_vac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dE_vac_dt = (self.beta_vac_unvac*Infectious_unvac +
self.beta_vac_vac*Infectious_vac)*Susceptible_vac\

/(Susceptible_unvac+Susceptible_vac+Exposed_unvac+Exposed_vac+Infectious_unvac+Infectious_vac+Rec
overed_unvac+Recovered_vac) \
                - (self.mu + self.alpha_vac)*Exposed_vac
        return dE_vac_dt

    def dI_unvac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dI_unvac_dt = self.alpha_unvac*Exposed_unvac \
                - (self.gamma_unvac+self.mu+self.theta_unvac)*Infectious_unvac
        return dI_unvac_dt

    def dI_vac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dI_vac_dt = self.alpha_vac*Exposed_vac \
                - (self.gamma_vac+self.mu+self.theta_vac)*Infectious_vac
```

```
        return dI_vac_dt

    def dR_unvac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dR_unvac_dt = self.gamma_unvac*Infectious_unvac \
                - self.mu*Recovered_unvac
        # Calculate the predator population change
        return dR_unvac_dt

    def dR_vac(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dR_vac_dt = self.gamma_vac*Infectious_vac \
                - self.mu*Recovered_vac
        # Calculate the predator population change
        return dR_vac_dt

    def dD(self, Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
Infectious_vac, Recovered_unvac, Recovered_vac, Dead):
        dD_dt = self.theta_unvac*Infectious_unvac \
                + self.theta_vac*Infectious_vac \
                +
self.mu*(Susceptible_unvac+Susceptible_vac+Exposed_unvac+Exposed_vac+Infectious_unvac+Infectious_
vac+Recovered_unvac+Recovered_vac)
        # Calculate the predator population change
        return dD_dt


    def calculate(self):
         # import json

        susceptible_unvac_history = []
        susceptible_vac_history = []
        exposed_unvac_history = []
        exposed_vac_history = []
        infectious_unvac_history = []
        infectious_vac_history = []
        recovered_unvac_history = []
        recovered_vac_history = []
        dead_history = []

        y0 = np.array([self.sus_unvac, self.sus_vac, self.exp_unvac, self.exp_vac,
                       self.inf_unvac, self.inf_vac, self.rec_unvac, self.rec_vac,
                       self.dead], dtype='double')
        tspan = np.array([0.0, self.days], dtype='double')

        try:
            t, y = self.rk4(self.derivatives, tspan, y0, self.days)

            susceptible_unvac_history = y[:, 0]
            susceptible_vac_history = y[:, 1]
            exposed_unvac_history = y[:, 2]
            exposed_vac_history = y[:, 3]
            infectious_unvac_history = y[:, 4]
```

```
            infectious_vac_history = y[:, 5]
            recovered_unvac_history = y[:, 6]
            recovered_vac_history = y[:, 7]
            dead_history = y[:, 8]


            print('t = ', t)
            #print(predator_history)
            #res = [dict([(ti, yi)]) for ti, yi in zip(t, predator_history)]
            #print(json.dumps(res, indent=2))
        except (RuntimeError, OverflowError, FloatingPointError):
            print("")

        return {
            'Susceptible_unvac': susceptible_unvac_history,
            'Susceptible_vac': susceptible_vac_history,
            'Exposed_unvac': exposed_unvac_history,
            'Exposed_vac': exposed_vac_history,
            'Infectious_unvac': infectious_unvac_history,
            'Infectious_vac': infectious_vac_history,
            'Recovered_unvac': recovered_unvac_history,
            'Recovered_vac': recovered_vac_history,
            'Dead': dead_history
        }

    def derivatives(self, t, rf):
        Susceptible_unvac = rf[0]
        Susceptible_vac = rf[1]
        Exposed_unvac = rf[2]
        Exposed_vac = rf[3]
        Infectious_unvac = rf[4]
        Infectious_vac = rf[5]
        Recovered_unvac = rf[6]
        Recovered_vac = rf[7]
        Dead = rf[8]

        dS_unvac_dt = self.dS_unvac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                    Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dS_vac_dt = self.dS_vac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dE_unvac_dt = self.dE_unvac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                    Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dE_vac_dt = self.dE_vac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dI_unvac_dt = self.dI_unvac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                    Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dI_vac_dt = self.dI_vac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
```

```
                                Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dR_unvac_dt = self.dR_unvac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                    Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dR_vac_dt = self.dR_vac(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac,
Infectious_unvac,
                                Infectious_vac, Recovered_unvac, Recovered_vac, Dead)
        dD_dt = self.dD(Susceptible_unvac, Susceptible_vac, Exposed_unvac, Exposed_vac, Infectious_unvac,
                        Infectious_vac, Recovered_unvac, Recovered_vac, Dead)

        drfdt = np.array([dS_unvac_dt, dS_vac_dt, dE_unvac_dt, dE_vac_dt, dI_unvac_dt,
                          dI_vac_dt, dR_unvac_dt, dR_vac_dt, dD_dt], dtype='double')
        return drfdt


    def rk4(self, dydt, tspan, y0, n):

        if np.ndim(y0) == 0:
            m = 1
        else:
            m = len(y0)

        tfirst = tspan[0]
        tlast = tspan[1]
        dt = (tlast - tfirst) / n
        t = np.zeros(n + 1)
        y = np.zeros([n + 1, m])
        y[0, :] = y0

        for i in range(0, n):
            f1 = dydt(t[i], y[i, :])
            f2 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f1 / 2.0)
            f3 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f2 / 2.0)
            f4 = dydt(t[i] + dt, y[i, :] + dt * f3)

            t[i + 1] = t[i] + dt
            y[i + 1, :] = y[i, :] + dt * (f1 + 2.0 * f2 + 2.0 * f3 + f4) / 6.0

        return t, y
import numpy as np
np.seterr(all='raise')

class SIRCalculator(object):
    def __init__(self):
        # SIR equation coefficients
      #  self.Susceptible = 20.0 #сприятливі до інфекції
      #  self.Recovered = 10.0 #очухавшиєся
        self.gamma = 1.0 #швидкість одужання
        self.beta = 1.0 #константа швидкості
      #  self.Infectious = 10.0
        # Other parameters
        self.days = 160
       # self.dt = 0.02
```

```
  # self.iterations = 1000

   self.sus = 5
   self.inf = 4
   self.rec = 10

def dS(self, Susceptible, Infectious, Recovered):
   dS_dt = -(self.beta*Susceptible*Infectious)/(Susceptible+Infectious+Recovered)
   return dS_dt

def dI(self, Susceptible, Infectious, Recovered):
   dI_dt = (self.beta*Susceptible*Infectious)/(Susceptible+Infectious+Recovered) - self.gamma*Infectious
   return dI_dt

def dR(self, Susceptible, Infectious, Recovered):
   dR_dt = self.gamma*Infectious
   # Calculate the predator population change
   return dR_dt

def calculate(self):
    # import json

   susceptible_history = []
   infectious_history = []
   recovered_history = []

   y0 = np.array([self.sus, self.inf, self.rec], dtype='double')
   tspan = np.array([0.0, self.days], dtype='double')

   try:
      t, y = self.rk4(self.derivatives, tspan, y0, self.days)
      susceptible_history = y[:, 0]
      infectious_history = y[:, 1]
      recovered_history = y[:, 2]

      print('t = ', t)
      #print(predator_history)
      #res = [dict([(ti, yi)]) for ti, yi in zip(t, predator_history)]
      #print(json.dumps(res, indent=2))
   except (RuntimeError, OverflowError, FloatingPointError):
      print("")

   return {
      'Susceptible': susceptible_history,
      'Infectious': infectious_history,
      'Recovered': recovered_history
   }

def calculate_r0(self, beta, gamma):
   R0 = beta / gamma
   return R0

def calculate_trecovery(self, gamma):
```

```
    t_recovery = 1 / gamma
    return t_recovery

def derivatives(self, t, rf):

    Susceptible = rf[0]
    Infectious = rf[1]
    Recovered = rf[2]

    dSdt = self.dS(Susceptible, Infectious, Recovered)
    dIdt = self.dI(Susceptible, Infectious, Recovered)
    dRdt = self.dR(Susceptible, Infectious, Recovered)

    drfdt = np.array([dSdt, dIdt, dRdt], dtype='double')
    return drfdt


def rk4(self, dydt, tspan, y0, n):
    # RK4 approximates the solution to an ODE using the RK4 method.
    #  Input:
    #    function dydt: points to a function that evaluates the right
    #                   hand side of the ODE.
    #    real tspan[2]: contains the initial and final times.
    #    real y0[m]: an array containing the initial condition.
    #    integer n: the number of steps to take.

    #  Output:
    #    real t[n+1], y[n+1,m]: the times and solution values.
    #

    if np.ndim(y0) == 0:
        m = 1
    else:
        m = len(y0)

    tfirst = tspan[0]
    tlast = tspan[1]
    dt = (tlast - tfirst) / n
    t = np.zeros(n + 1)
    y = np.zeros([n + 1, m])
    y[0, :] = y0

    for i in range(0, n):
        f1 = dydt(t[i], y[i, :])
        f2 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f1 / 2.0)
        f3 = dydt(t[i] + dt / 2.0, y[i, :] + dt * f2 / 2.0)
        f4 = dydt(t[i] + dt, y[i, :] + dt * f3)

        t[i + 1] = t[i] + dt
        y[i + 1, :] = y[i, :] + dt * (f1 + 2.0 * f2 + 2.0 * f3 + f4) / 6.0

    return t, y
```

```
#!/usr/bin/env python

# Python standard library modules
import sys
import os
import pandas as pd

# 3rd party modules
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.backends.backend_qt5agg as backend_qt5agg
from matplotlib.figure import Figure
import PyQt5.QtCore as QtCore
import PyQt5.QtGui as QtGui
import PyQt5.QtWidgets as QtWidgets

# Local application modules
from SIR_calculator import SIRCalculator
from OptionsMenu_SIR import OptionsMenu_SIR
#from options_menu_4_species import OptionsMenu_4_species
#from options_menu_2 import OptionsMenu_2_species

#import resources

APP_NAME = 'Epidemics'
AUTHOR = 'Клименко Анастасія'

class AppForm_SIR(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QMainWindow.__init__(self, parent)

        # название окна
        self.setWindowTitle(APP_NAME)

        # Создание в меню параметров в виджете док-станции
        self.options_menu = OptionsMenu_SIR()
        dock = QtWidgets.QDockWidget('Налаштування коефіцієнтів', self)
        dock.setFeatures(
            QtWidgets.QDockWidget.NoDockWidgetFeatures |
            QtWidgets.QDockWidget.DockWidgetMovable |
            QtWidgets.QDockWidget.DockWidgetFloatable
        )
        dock.setAllowedAreas(
            QtCore.Qt.LeftDockWidgetArea | QtCore.Qt.RightDockWidgetArea,
        )
        dock.setWidget(self.options_menu)
        self.addDockWidget(QtCore.Qt.LeftDockWidgetArea, dock)

        self.options_menu.update_btn.clicked.connect(self.clear_graph)
        self.options_menu.update_btn.clicked.connect(self.calculate_data)
        self.options_menu.clear_graph_btn.clicked.connect(self.clear_graph)
        self.options_menu.legend_cb.stateChanged.connect(self.redraw_graph)
        self.options_menu.grid_cb.stateChanged.connect(self.redraw_graph)
```

```
    self.options_menu.legend_loc_cb.currentIndexChanged.connect(self.redraw_graph)

    # создание графика
    fig = Figure((7.0, 3.0), dpi=100)
    self.canvas = backend_qt5agg.FigureCanvasQTAgg(fig)
    self.canvas.setParent(self)
    self.axes = fig.add_subplot(111)
    backend_qt5agg.NavigationToolbar2QT(self.canvas, self.canvas)

    self.clear_graph()

    self.setCentralWidget(self.canvas)
    about_action = QtWidgets.QAction('&About', self)
    about_action.setToolTip('About')
    about_action.setIcon(QtGui.QIcon(':/resources/icon_info.png'))
    about_action.triggered.connect(self.show_about)

    sir_action = QtWidgets.QAction('&SIR', self)
    sir_action.setToolTip('SIR')
    sir_action.triggered.connect(self.show_sir)

    file_exit_action = QtWidgets.QAction('&Exit', self)
    file_exit_action.setToolTip('Exit')
    file_exit_action.setIcon(QtGui.QIcon(':/resources/door_open.png'))
    file_exit_action.triggered.connect(self.close)

    file_menu = self.menuBar().addMenu('&Exit')
    file_menu.addAction(file_exit_action)

    help_menu = self.menuBar().addMenu('&Help')
    help_menu.addAction(about_action)

    sir_menu = self.menuBar().addMenu('&SIR')
    sir_menu.addAction(sir_action)


def calculate_data(self):
    # объект GrowthCalculator
    growth = SIRCalculator()

    # Update the GrowthCalculator parameters from the GUI options
    growth.gamma = self.options_menu.gamma_sb.value()
    growth.beta = self.options_menu.beta_sb.value()

    growth.sus = self.options_menu.sus_sb.value()
    growth.inf = self.options_menu.inf_sb.value()
    growth.rec = self.options_menu.rec_sb.value()
    growth.days = self.options_menu.days_sb.value()
   # growth.iterations = self.options_menu.iterations_sb.value()
   # growth.dt = self.options_menu.timedelta_sb.value()

    # Calculate the population growths
    results = growth.calculate()
```

```
    self.infectious_history.extend(results['Infectious'])
    self.susceptible_history.extend(results['Susceptible'])
    self.recovered_history.extend(results['Recovered'])

    if (len(self.infectious_history) == 0 and
            len(self.susceptible_history) == 0 and
            len(self.recovered_history) == 0):
        QtWidgets.QMessageBox.information(self, 'Error', 'Помилка')
        return

    self.redraw_graph()

def statisticsExcel(self):
    self.axes.clear()
    os.chdir("C:/Users/asja6/Documents/Projects/Epidemic_dyploma/data/")
    file = "Sweden.xlsx"
    Ukraine = pd.read_excel(file)
    new_cases = Ukraine.values[:, 5]
    days = Ukraine.values[:, 3]
    days_filtered = []
    new_cases_filtered = []
    for i in range(1, len(days)):
        if "2021" not in days[i] and "2020" not in days[i]:
            if "-01-" in days[i]:
                days_filtered.append(days[i])
                new_cases_filtered.append(new_cases[i])
    plt.figure(figsize=(15, 15))
    plt.plot(days_filtered, new_cases_filtered)
    plt.suptitle('Стастика захворюваності', fontsize=20)
    plt.show()

def onlyInfectious(self):
    plt.close()
    plt.figure(figsize=(15, 15))
    plt.plot(self.infectious_history)
    plt.suptitle('Стастика захворюваності', fontsize=20)
    plt.show()

def clear_graph(self):
    # очистить историю популяций
    self.infectious_history = []
    self.susceptible_history = []
    self.recovered_history = []

    # перерисовать граф
    self.redraw_graph()

def redraw_graph(self):
    # очистить граф
    self.axes.clear()

    # Create the graph labels
    self.axes.set_title('SIR-model: Susceptible, Infectious, Recovered')
```

```
        self.axes.set_xlabel('Дні')
        self.axes.set_ylabel('Кількість населення')

        # Plot the current population data

        if self.susceptible_history:
            self.axes.plot(self.susceptible_history, 'b-', label='ще не переболівші')
        if self.infectious_history:
            self.axes.plot(self.infectious_history, 'r-', label='хворі')
        if self.recovered_history:
            self.axes.plot(self.recovered_history, 'g-', label='перехворіли та мають імунітет')
        # если нужно, создаём легенду
        if self.options_menu.legend_cb.isChecked():
            if self.recovered_history or self.susceptible_history or self.infectious_history:
                legend_loc = str(
                    self.options_menu.legend_loc_cb.currentText()
                ).lower()
                legend = matplotlib.font_manager.FontProperties(size=10)
                self.axes.legend(loc=legend_loc, prop=legend)

        # если нужно, сетки на графике
        self.axes.grid(self.options_menu.grid_cb.isChecked())

        # рисуем график
        self.canvas.draw()

    def show_sir(self):

        message = '''<font size="+2">SIR-модель</font>
            <p>dS/dt = -βSI/N
            <p>dI/dt = βSI/n - ɣI
            <p>dR/dt = ɣI
            <p>де
            <p>S — кількість сприятливих до вірусу;
            <p>I — кількість інфікованих;
            <p>R — кількість переболівших;
            <p>β — константа швидкості;
            <p>ɣ — швидкість одужання.
            '''
        QtWidgets.QMessageBox.about(self, 'SIR' + '', message)

    def show_about(self):

        message = '''<font size="+2">%s</font>
            <p> Магістерська дисертація на тему "моделювання епідеміологічних моделей".
            <p>Написана by %s, група КА-13мп.
            ''' %(APP_NAME, AUTHOR)

        QtWidgets.QMessageBox.about(self, 'About' + APP_NAME, message)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    app.setWindowIcon(QtGui.QIcon(':/resources/icon.svg'))
```

```
    form = AppForm_SIR()
    form.show()
    app.exec_()
import pandas as pd
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# The SIR model differential equations.
def deriv(state, t, N, beta, gamma):
    S, I, R = state
    # Change in S population over time
    dSdt = -beta * S * I / N
    # Change in I population over time
    dIdt = beta * S * I / N - gamma * I
    # Change in R population over time
    dRdt = gamma * I
    return dSdt, dIdt, dRdt
effective_contact_rate = 0.5
recovery_rate = 1/4

# We'll compute this for fun
print("R0 is", effective_contact_rate / recovery_rate)

# What's our start population look like?
# Everyone not infected or recovered is susceptible
total_pop = 1000
recovered = 0
infected = 1
susceptible = total_pop - infected - recovered

# A list of days, 0-160
days = range(0, 160)

# Use differential equations magic with our population
ret = odeint(deriv,
             [susceptible, infected, recovered],
             days,
             args=(total_pop, effective_contact_rate, recovery_rate))
S, I, R = ret.T

# Build a dataframe because why not
df = pd.DataFrame({
    'suseptible': S,
    'infected': I,
    'recovered': R,
    'day': days
})

plt.style.use('ggplot')
df.plot(x='day',
        y=['infected', 'suseptible', 'recovered'],
        color=['#bb6424', '#aac6ca', '#cc8ac0'],
```

```
        kind='area',
        stacked=True)
plt.show()
```