КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра Обчислювальної Математики

Курсова робота

За спеціальністю 113 "Прикладна математика"

на тему:

Моделювання розвитку епідемії з використанням чисельних методів

Виконав студент 3-го курсу

Коломієць Микола

Науковий керівник:

Тимошенко Андрій

Київ, 2023

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта дослідження

У рамках даної курсової роботи, нашим об'єктом дослідження є розвиток епідемії. Для отримання глибокого розуміння даного об'єкта, ми здійснюємо оцінку його сучасного стану. Ця оцінка базується на дослідженні актуальних даних та використанні чисельних методів.

Ми вивчаємо важливі фактори, які впливають на поширення епідемії, такі як швидкість передачі, рівень вакцинації, ефективність заходів контролю та інші. Це дозволяє нам отримати більш точну картину та зробити адекватні прогнози.

Оцінка сучасного стану об'єкта дослідження також включає використання чисельних методів. Ми використовуємо математичні моделі для моделювання поширення епідемії на основі наявних даних. Ці моделі дозволяють нам розрахувати рівень інфікування, динаміку поширення та імітацію різних сценаріїв.

Отримані результати оцінки сучасного стану об'єкта дослідження стануть основою для подальшого моделювання розвитку епідемії та визначення ефективних стратегій контролю та мінімізації поширення захворювання.

Актуальність роботи та підстави для її виконання.

Наша курсова робота, що присвячена моделюванню розвитку епідемії з використанням чисельних методів, є дуже актуальною в контексті сучасних глобальних викликів у сфері громадського здоров'я. Ось деякі підстави, які обгрунтовують актуальність нашої роботи:

Пандемія COVID-19: Світ стикається з найпоширенішою епідемією за останнє десятиліття, яка має глибокий вплив на життя людей та економіку. Розуміння та моделювання розвитку епідемій, таких як COVID-19, є надзвичайно важливими для розробки ефективних стратегій контролю та прогнозування їхнього впливу.

Потреба у прогнозуванні: Розуміння майбутнього розвитку епідемій

дозволяє приймати обґрунтовані рішення щодо адаптації систем охорони здоров'я, вакцинації, медичного обладнання та інших аспектів. Моделювання розвитку епідемії з використанням чисельних методів надає можливість прогнозувати варіанти розвитку подій та оцінювати ефективність різних стратегій.

Розвиток наукових методів: Чисельні методи використовуються в багатьох наукових дисциплінах для моделювання складних процесів. Застосування цих методів до дослідження епідеміологічних моделей дозволяє отримати детальніші та точніші результати, що сприяє покращенню нашого розуміння поширення епідемій.

Перспектива майбутніх епідемій: Історія свідчить про те, що епідемії та пандемії є неминучою частиною нашого світу. Моделювання розвитку епідемій з використанням чисельних методів допоможе нам готуватися до майбутніх викликів і впроваджувати належні заходи передбачення, мінімізації поширення та зменшення наслідків епідемій.

Отже, наша робота є актуальною та важливою, оскільки вона спрямована на дослідження моделей розвиту епідемій. Вона відповідає сучасним викликам у галузі громадського здоров'я та може стати цінним інструментом для прогнозування та управління епідеміологічними заходами.

Мета й завдання роботи. Метою є

Метою роботи є дослідження різних моделей розвитку епідемі за допомогою чисельних методів. Завданнями роботи є: -огляд основних моделей розвитку епідемій -підбір параметрів моделей під реальні дані -отримання та оцінка прогнозів моделей

Об'єкт, методи та засоби дослідження.

Об'єктом нашого дослідження є розвиток епідемії. Ми зосереджуємося на моделюванні та прогнозуванні поширення захворювання з використанням чисельних методів.

Для досягнення наших цілей ми використовуємо наступні методи та засоби:

1. Математичне моделювання: Ми використовуємо математичні моделі для опису динаміки поширення епідемії. Ці моделі базуються на епідеміологічних принципах та інформації про характеристики захворювання. Вони дозволяють нам розрахувати рівень інфікування, швидкість

поширення та імітувати різні сценарії розвитку епідемії.

2. Чисельні методи: Для розв'язання математичних моделей ми використовуємо чисельні методи. Ці методи дозволяють нам апроксимувати розв'язок диференціальних рівнянь та виконувати чисельні симуляції розвитку епідемії. Вони забезпечують точність та ефективність обчислень.

3. Програмування на Python та C: Ми використовуємо мови програмування Python та C для реалізації наших чисельних методів та моделей. Python є широко використовуваною мовою програмування у сфері наукових обчислень, зокрема для моделювання епідемій. C є мовою програмування, що надає високу швидкодію, і може бути використана для оптимізації обчислювальних процесів.

4. Аналіз даних: Для оцінки сучасного стану об'єкта дослідження ми здійснюємо аналіз актуальних даних щодо поширення епідемії. Ми використовуємо статистичні методи та інструменти для обробки та визначення закономірностей у даних.

5. Комп'ютерне моделювання: Ми використовуємо комп'ютерні моделі та інструменти для виконання чисельних симуляцій та візуалізації результатів. Це дозволяє нам вивчати розвиток епідемії на основі введених параметрів та проводити аналіз різних сценаріїв.

Загалом, наш дослідницький підхід поєднує математичне моделювання, чисельні методи та програмування, зокрема використання мов Python та C, для досягнення наших цілей у моделюванні розвитку епідемії.

Можливі сфери застосування.

За допомогою подібних дій описаних у даній роботі можна оцінювати моделі розвитку епідемій і таким чином вибирати найкращі ідеї. Завдякт цьому

РОЗДІЛ 1 SIR МОДЕЛЬ

В першому розгляді розглянемо SIR-подібні моделі. Почнемо з базової SIR моделі і поступово будемо ускладнювати модель шляхом додавання нових груп, що дозволить поступово наблизити модель до реалій протікання хвороби.

1.1 SIR модель

Одною з найпростіших моделей є SIR модель. В ній всі люди розглядаються у вигляді трьох груп: здорові (susceptible), хворі (infectious) та ті, що одужали (removed). Форсування людей між цими групами відбувається зі здорових у хворі і з хворих до одужаних.[9](див рис. 1.1)

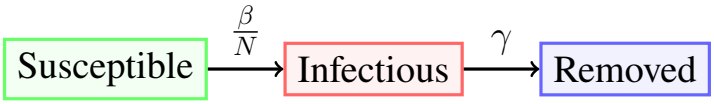

Рис. 1.1: SIR model

Зв'язок між цими групами записують за допомогою системи диференціальних рівняннь

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

Перше рівняння описує, що кількість здорових зменшується на величину пропорційну добутку хворих на здорових - кількість можливих контактів. Друге рівняння показує, що кількість хворих зростає на ту саму величну і зменшується на величину пропорційну до кількості хворих (кількість одужаних). Третє рівняння зібльшується на ту саму кількість одужаних.

Додавши всі рівняння системи отримаємо - $\frac{dS}{dt} + \frac{dI}{dt} + \frac{dR}{dt} = 0$. Тож населення у нашій моделі не змінюється з часом: $N = S + I + R = const$.[8]

1.2 SEIR модель

Часто хвороба характеризується не лише станами - хворий, здоровий, перехворівший. Для подібних випадків додають додаткові стани. Наприклад, якщо додати у SIR модель ще одну групу «заражені, але ще не взмозі інфікувати інших» то вийде SEIR модель. Тоді у порівнянні з SIR моделлю у схему додасться проміжний етап. (див рис. 1.2)

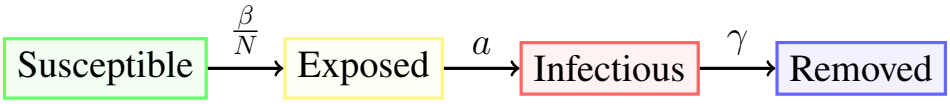

Рис. 1.2: Схема роботи SIER моделі

У вихідну систему додається додаткове рівняння

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta}{N} SI \\ \frac{dE}{dt} = \frac{\beta}{N} SI - aE \\ \frac{dI}{dt} = aE - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases}$$

E(t) функція кількості заражених, коефіцієнт a відповідає оберненому середньому періоду переходу людини від зараженої, до повноцінного інфікованого. Так само населення залишається незмінним: $N = S + E + I + R$.

Подібні прийоми застосовують для більш точного моделювання відповідно до протікання хвороби - часто є певний період під час якого інфікований, ще не заражає оточуючих або заражає з меншим шансом, більш тісним контаком, або період до проявів перших симптомів і початком домашньої ізоляції, носіння маски. Це впливає на параметри та рівняння моделі і на точність резульата. Для даної моделі буде складніше підібрати вірні параметри, але це може позетивно вплинути на точність. [7]

1.3 SEI модель

Іноді моделі не ускладнюють, а спрощують. Наприклад, якщо для хвороби виробити імунітет навічно не можливо, як от нежить, одужаних розглядати немає сенсу.[3] Таким чином схема спрощується(див рис. 1.3)

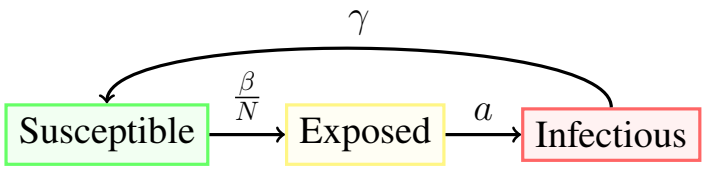

Рис. 1.3: Схема роботи SEI моделі

Отже і система буде складатись з меншої кількості рівнянь.

$$\begin{cases} \frac{dS}{dt} = -\frac{\beta}{N}SI + \gamma I \\ \frac{dE}{dt} = \frac{\beta}{N}SI - aE \\ \frac{dI}{dt} = aE - \gamma I \end{cases}$$

1.4 SEIPR

Додамо нову групу інфікованих, яка заражає здорових людей з більшою чи меншою ймовірністю ніж звичайні інфіковані.

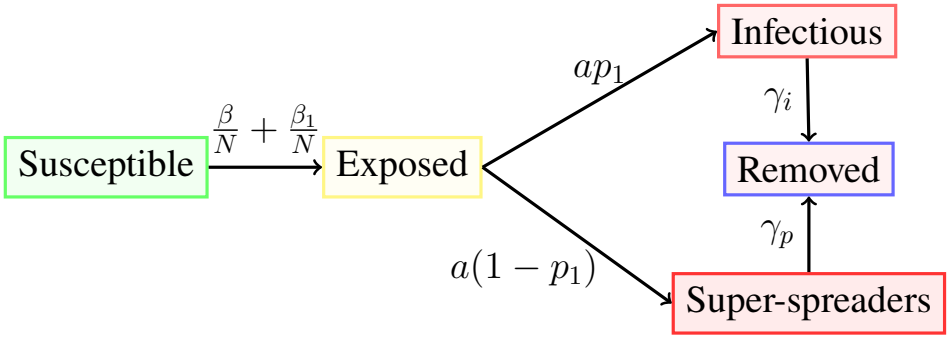

Рис. 1.4: Схема роботи SEIPR моделі

Тоді кожен заражений може стати або звичайним інфікованим з ймовірністю p_1 або іншим інфікованим з ймовірністю $1-p_1$. Параметр, що відповідає за інфікування інших інфікованих β_1. Для спрощення моделі часто припускають, що ці дві групи інфікованих мають однакову ймовірність одужати і беруть однакові коефіцієнти $\gamma_i = \gamma_p = \gamma$, але загальна система виглядає так:

$$\begin{cases} \frac{dS}{dt} = -\beta\frac{I}{N}S - \beta'\frac{P}{N}S, \\ \frac{dE}{dt} = \beta\frac{I}{N}S + \beta'\frac{P}{N}S - \kappa E, \\ \frac{dI}{dt} = \kappa\rho_1 E - \gamma_i I, \\ \frac{dP}{dt} = \kappa(1-\rho_1)E - \gamma_p P, \\ \frac{dR}{dt} = \gamma_i I + \gamma_p P \end{cases}$$

1.5 SEIAPR

Додамо групу людей, які заражені проте не мають ніяких симптомів і не можуть заражати інших.

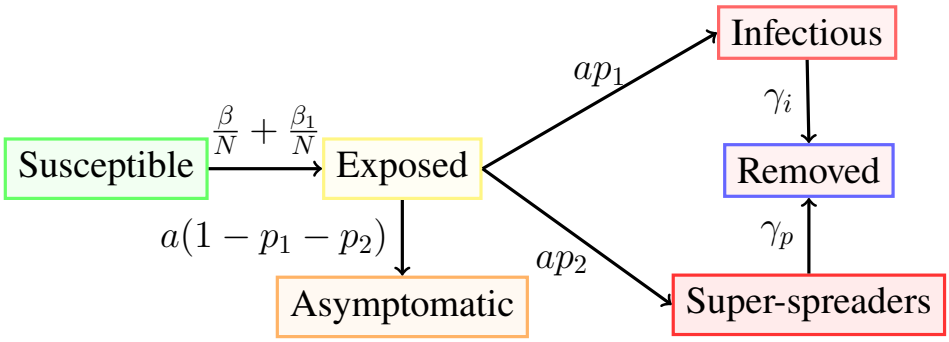

Рис. 1.5: Схема роботи SEIAPR моделі

В данному випадку нас не цікавить чи вилікуються вони чи ні адже вони впливають на систему так само як і виліковані. Тоді кожен заражений з категорії E з ймовірністю p_1 зможе стати зараженим I з ймовірністю p_2 може стати зараженим P і з ймовірністю $1-p_1-p_2$ може стати зараженим A.[6]

$$\begin{cases} \frac{dS}{dt} = -\beta\frac{I}{N}S - \beta'\frac{P}{N}S, \\ \frac{dE}{dt} = \beta\frac{I}{N}S + \beta'\frac{P}{N}S - aE, \\ \frac{dI}{dt} = a\rho_1 E - \gamma_i I, \\ \frac{dP}{dt} = a\rho_2 E - \gamma_p P, \\ \frac{dA}{dt} = a\left(1-\rho_1-\rho_2\right)E, \\ \frac{dR}{dt} = \gamma_i I + \gamma_p P \end{cases}$$

1.6 SEIPHRD

Остання модель була взята з статті прогнозування захворюваності у Вухані.[5] Тут додані дві нові групи H- госпіталізовані та F- летальні випад-

ки. Це допоможе нам конфігурувати моделі адже подібні дані дуже часто зберігаються і вони є більш точними ніж кількість інфікованих.

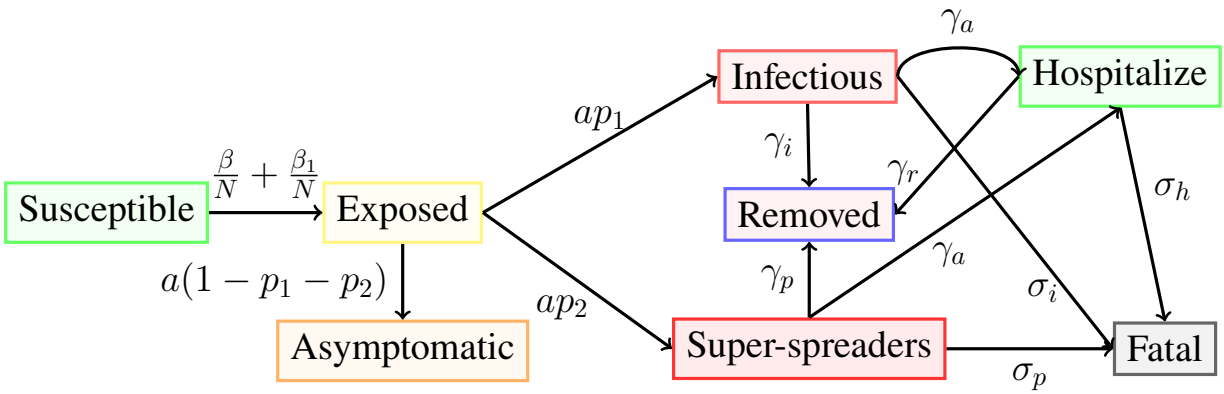

Рис. 1.6: Схема роботи SEIAPHRF моделі

За смерть різниї груп відповідають параметри $\sigma_i, \sigma_p, \sigma_h$, а за госпіталізацію параметр γ_a.

$$
\begin{cases}
\frac{dS}{dt} = -\beta\frac{I}{N}S - l\beta\frac{H}{N}S - \beta'\frac{P}{N}S, \\
\frac{dE}{dt} = \beta\frac{I}{N}S + l\beta\frac{H}{N}S + \beta'\frac{P}{N}S - aE, \\
\frac{dI}{dt} = a\rho_1 E - (\gamma_a + \gamma_i)\, I - \delta_i I, \\
\frac{dP}{dt} = a\rho_2 E - (\gamma_a + \gamma_i)\, P - \delta_p P, \\
\frac{dA}{dt} = a\,(1 - \rho_1 - \rho_2)\, E, \\
\frac{dH}{dt} = \gamma_a(I + P) - \gamma_r H - \delta_h H, \\
\frac{dR}{dt} = \gamma_i(I + P) + \gamma_r H, \\
\frac{dF}{dt} = \delta_i I + \delta_p P + \delta_h H,
\end{cases}
$$

РОЗДІЛ 2 ЧИСЕЛЬНІ МЕТОДИ

У другому розділі розглянемо чисельні методи для розв'язання данних сисетем диф рівняннь та для визначення оптимальних параметрів, що якнайкраще симулюють розвиток захворюванстi.

2.1 Методи вирішення системи диф рівняннь

Розв'язати подібні системи в загальному вигляді від параметрів не можна через нелінійність та не стійкість розв'язків, тож у данному розділі ми розглянемо деякі чисельні методи для розв'язку подібних систем відносно наперед відомих параметрів.

2.1.1 Метод Ейлера

Метод Ейлера найбільш базовий чисельний метод для розв'язання нелінійних систем диференціальних рівняннь, що застосувується при відомих початкових значеннях. В данному випадку нам відома кількість хворих на початку епідемії. Це частковий випадок методів Рунге Кутті. Загалом метод можна описати формулою $dS \approx S_{n+1} - S_n$. Зазвичай береться певний крок dt і дифиренціальне рівняння апроксимується відносно нього:

$$\frac{dS}{dt} = f(t, S) \Rightarrow S_{n+1} = S_n + f(t_n, S_n)$$

Ітерації починаються при відомому S_0. Метод є не стабільним при великому кроці dt.[2, 1]

Як приклад візьмемо SIR модель при розв'язанні моделі отримаємо:

$$\begin{cases} S_{n+1} = S_n - \frac{\beta}{N} S_n I_n dt \\ I_{n+1} = I_n + \frac{\beta}{N} S_n I_n dt - \gamma I_n dt \\ R_{n+1} = R_n + \gamma I_n dt \\ S_0 = N - 1, I_0 = 1, R_0 = 0 \end{cases}$$

Функція написана на C для отримання данних з моделі відносно її параметрів кроку та початкових умов.

```
float** sir(int size, float beta, float gama, int N, float dt){
    float* S = malloc(size * sizeof(float));
    float* I = malloc(size * sizeof(float));
    float* R = malloc(size * sizeof(float));
    float** answer = malloc(3 * sizeof(float*));
    answer[0] = S; answer[1] = I; answer[2] = R;
    S[0] = N - 1;I[0] = 1;R[0] = 0;answer[0] = 1;
    int step = (int)(1 / dt);float sum = 1;
    for (int i = 0; i < size - 1; i++){
        S[i + 1] = S[i] - beta * S[i] * I[i] * dt / N;
        I[i + 1] = I[i] + (beta * S[i] * I[i] / N - gama * I[i]) * dt;
        R[i + 1] = R[i] + gama * I[i] * dt;}
    return answer;
}
```

Мова програмування C була обрана для написання цієї функції завдяки її швидкості виконання та відносної простоти написанння коду. Потреба у високій швидкості виконання обумовлена потребами задля швидкого знахордження оптимальних параметрів - через складну залежність від параметрів моделі використати градієнтні методи не вийде, а не градієнтні методи можуть багато разів викликати данну функцію на кожній ітерації.

Функції, що розв'язують інші моделі відрізняються лише кількістю масивів та основним циклом, загальна структура функцій однакова. Час виконання програми такий то.

2.1.2 Класичний метод Рунге Кутті

Ідеологічно методи Рунге Кутті це алгоритми влучного застосування методу Ейлера певну кількість разів. Від кількості разів, яку він застосовується залежить порядок методу. Наприклад класичним методом Рунге Кутті називають саме метод четвертого порядку. [2, 1]

Нехай у нас є система диференціальних рівняннь $\frac{dx}{dt} = f(x, t)$. Тоді,

якщо скористатися методом Ейлера отримаємо $x_{k+1} = x_k + f(x_k, t_k)$. Замість цього ми рахуємо лише доданок $k_1 = f(x_k, t_k)$. Далі рахуємо, ще декілька доданків

$$\begin{cases} k_1 = f(x_k, t_k) \\ k_2 = f(x_k + k_1\frac{dt}{2}, t_k + \frac{dt}{2}) \\ k_3 = f(x_k + k_2\frac{dt}{2}, t_k + \frac{dt}{2}) \\ k_4 = f(x_k + k_3 dt, t_k + dt) \end{cases}$$

І вже тоді додаємо, отримуючи наближене значення

$$x_{k+1} = x_k + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Розглянемо роботу методу на звичайній SIR моделі. Для простоти введемо функції:

$$\begin{cases} f_1(S_k, I_k, R_k) = -\beta\frac{S_k}{N}I_k \\ f_2(S_k, I_k, R_k) = \beta\frac{S_k}{N}I_k - \gamma I_k \\ f_3(S_k, I_k, R_k) = \gamma I_k \end{cases}$$

Розрахунок наближень 4-рьох порядків:

$$\begin{cases} s_1 = f_1(S_k, I_k, R_k) \\ i_1 = f_2(S_k, I_k, R_k) \\ r_1 = f_3(S_k, I_k, R_k) \end{cases}$$

$$\begin{cases} s_2 = f_1(S_k + \frac{dt}{2}s_1, I_k + \frac{dt}{2}i_1, R_k + \frac{dt}{2}r_1) \\ i_2 = f_2(S_k + \frac{dt}{2}s_1, I_k + \frac{dt}{2}i_1, R_k + \frac{dt}{2}r_1) \\ r_2 = f_3(S_k + \frac{dt}{2}s_1, I_k + \frac{dt}{2}i_1, R_k + \frac{dt}{2}r_1) \end{cases}$$

$$\begin{cases} s_3 = f_1(S_k + \frac{dt}{2}s_2, I_k + \frac{dt}{2}i_2, R_k + \frac{dt}{2}r_2) \\ i_3 = f_2(S_k + \frac{dt}{2}s_2, I_k + \frac{dt}{2}i_2, R_k + \frac{dt}{2}r_2) \\ r_3 = f_3(S_k + \frac{dt}{2}s_2, I_k + \frac{dt}{2}i_2, R_k + \frac{dt}{2}r_2) \end{cases}$$

$$\begin{cases} s_4 = f_1(S_k + dts_3, I_k + dti_3, R_k + r_3) \\ i_4 = f_2(S_k + dts_3, I_k + dti_3, R_k + r_3) \\ r_4 = f_3(S_k + dts_3, I_k + dti_3, R_k + r_3) \end{cases}$$

Отримане наближення:

$$\begin{cases} S_{k+1} = S_k + \frac{dt}{6}(s_1 + 2s_2 + 2s_3 + s_4) \\ I_{k+1} = I_k + \frac{dt}{6}(i_1 + 2i_2 + 2i_3 + i_4) \\ R_{k+1} = R_k + \frac{dt}{6}(r_1 + 2r_2 + 2r_3 + r_4) \end{cases}$$

Задля застосуваннння данного методу на інших моделях треба лише додати потрібні допоміжні функції та розрахунок додаткових змінних, принцип залишається незмінним.

2.2 Методи підбору параметрів

Тепер коли ми можемо порахувати різницю між прогнозованою величиною, яку видає наша модель і реальними данними, можемо використати алгоритми оптимізації для підбору оптимальних параметрів. За функцію ціни візьмемо відстань між графіками реальних данних та прогнозованих. Розглянемо лише неградієнтні методи оптимізації, адеж порахувати градієнт функції наближеної методами з першої частини глави не можна через складність обрахунків та кількість задіяної пам'яті.

2.2.1 Метод рою часток

Метод рою частинок не передбачає розрахунок градієнта функції, тому його швидкість роботи більш ніж задовільняє нашу задачу. Данний метод використовує "частинки кожна з яких символізує один розв'язок задачі. На кожній ітерації частинка намагається рухатись у оптимальному напрямі, тут присутній елемент стахастики адже випадковим є те чи буде вона рухатись до глобального оптимума (за весь час роботи алгоритма) чи до локального оптимума данної частинки. Функцію зміни параметрів частинки можна описати такою формулою:

$$\Delta x_{k+1}^{(i)} = \alpha \Delta x_k^{(i)} + \beta(x_{best}^{(i)} - x_k) + \gamma(x_{best} - x_k^{(i)})$$

У формулі $x_{best}^{(i)}$ найкраще значення цієї частинки, x_{best} найкраще значення всіх частинок. [4]

Моя реалізація на мові Python:

```
    def roy(f, n, N, iteration, w,
    a1, a2, A, B, eps, first):
        X = np.array([[random.uniform(A[i], B[i])
        for i in range(n)] for j in range(N)])
        V = np.array([[random.uniform(-eps, eps)
        for i in range(n)]
        for j in range(N)])
        for i in range(n):
            X[0][i] = first[i]
        P = np.copy(X)
        res = np.array([f(X[i]) for i in range(N)])
        b = P[np.argmin(res)]
        for i in tqdm(range(iteration)):
            V = w * V +
            a1 * random.random() * (P - X) +
            a2 * random.random() * (b - X)
            X = X + V
            for j in range(N):
                res1 = f(X[j])
                if res1 < res[j]:
                    P[j] = X[j]
                    res[j] = res1
            b = P[np.argmin(res)]
        return b
```

РОЗДІЛ 3 ОЦІНКА І ПОРІВНЯННЯ МОДЕЛЕЙ

У ході курсової роботи для оцінки та візуалізації моделей була створена програма, яка дозволяє користувачу регулювати параметри та одразу бачити, як вони впливають на прогнози. Також реалізована кнопка, яка починає оптимізацію початкових параметрів, налаштованих користувачем, методом рою часток (одна з часток ініціалізована початковими параметрами). Також є опції зберігання та загрузки параметрів у форматі json. Основна програма написана на мові програмування Python, функції поделювання написані на C задля збільшення швидкості оптимізації.

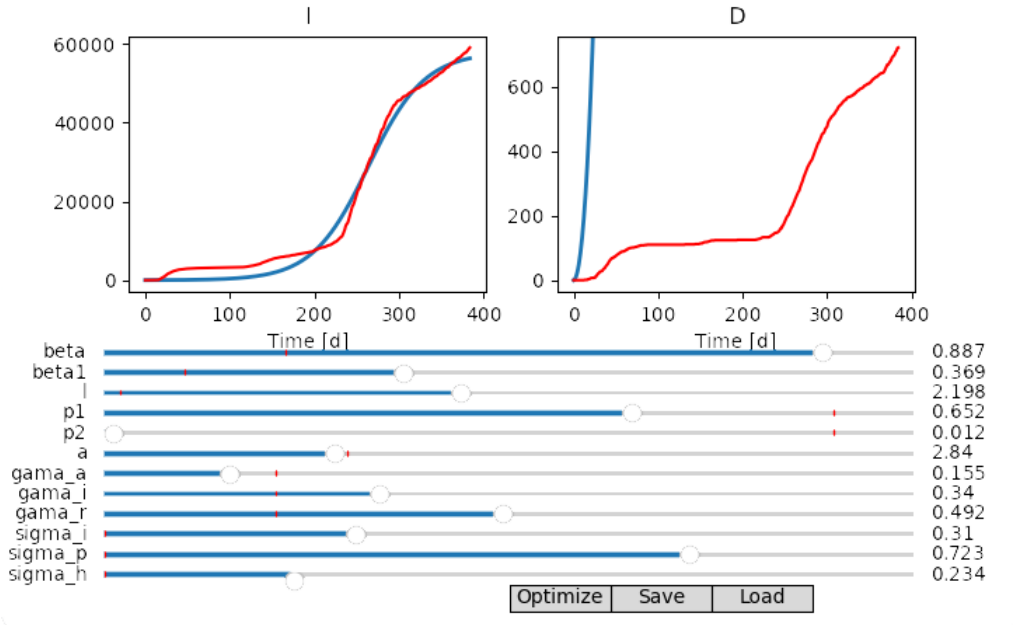

Рис. 3.1: Програма для підбору і оптимізації параметрів моделей

3.1 Справжні дані та оцінка ефективності

Прогнозувати будемо розвиток коронавірусу у Люксимбургу через те, що країна досить не велика та має середню густоту населення. Візуалізація данних за один рік.

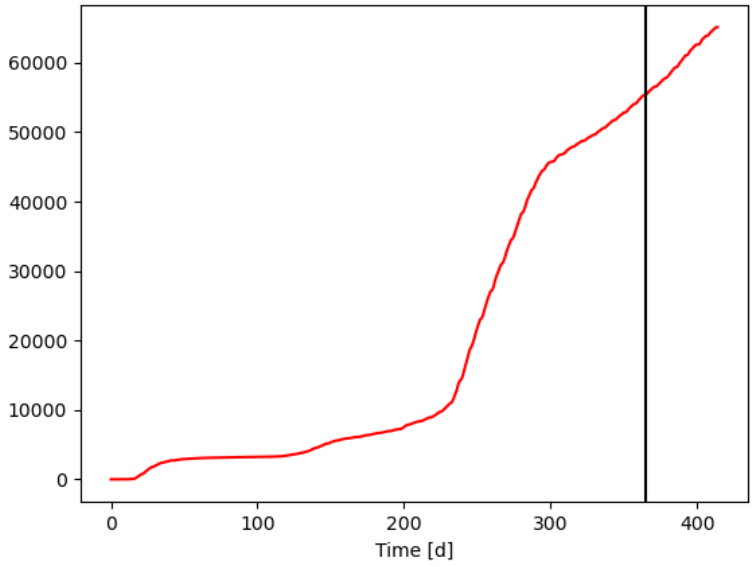

Рис. 3.2: Реальні дані захворюваності у Люксимбургу

Проміжкок після чорної прямої - період прогнозування. Підбирати параметри будемо за перший рік епідемії і відповідний прогноз будемо робити на наступні двадцять днів. Для оцінки моделей будемо використовувати дві похибки, похибка моделювання - сума квадратів відстаней між графіками кожного дня (при кожному цілому t) і похибку прогнозування - та сама величина але саме за давдцять днів прогнозування.

3.2 SIR

SIR модель погано моделює навіть ту частину графіка, яку ми використали для налаштування її параметрів. Середня різниця між графіками на відрізку прогнозування (20 днів) має порядк сотень людей. Похибка моделювання - 36695055916, похибка прогнозування - 142262882.

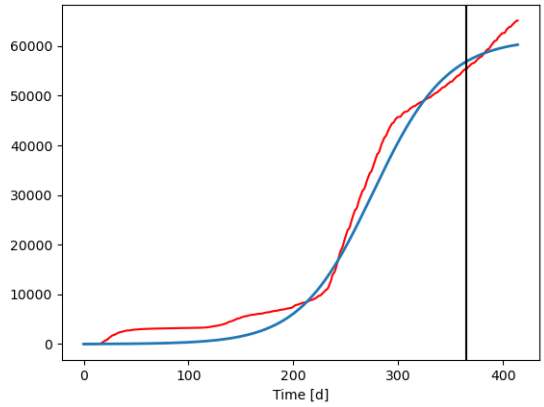

Рис. 3.3: Прогноз SIR моделі після 365 днів

3.3 SEIR

З SEIR моделлю вже можливо досить непогано апроксимувати графік, проте прогноз і реальні дані мають зовсім різну динаміку, порядок похибки - тисячі, отже дану модель не можна використовувати для пронозів. Похибка моделювання 36697160718, похибка прогнозування 131123602.

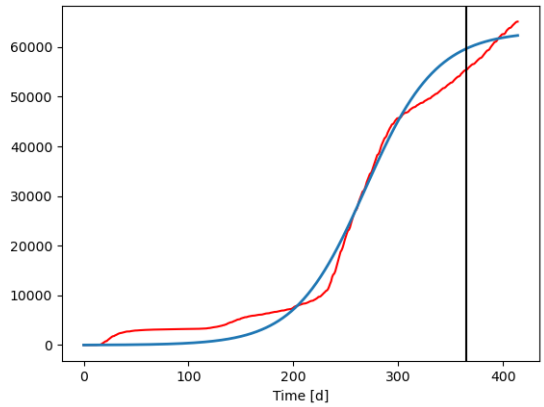

Рис. 3.4: Прогноз SEIR моделі після 365 днів

3.4 SEIPR

Модель має велику похибку на 365 день моделювання, через це прогноз на наступні дні має велику різницю з реальними данними. Похибка моделювання 36603451786, похибка прогнозування 241067088.

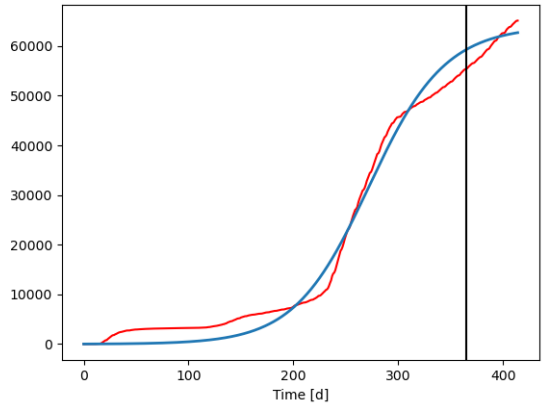

Рис. 3.5: Прогноз SIPR моделі після 365 днів

3.5 SEIAPR

Модель краще апроксимує графік, проте все ще погано моделює кількість хворих на короткі періоди часу. Загальна похибка 36663218005 Похибка моделювання 36772792167, похибка прогнозування 107349315.

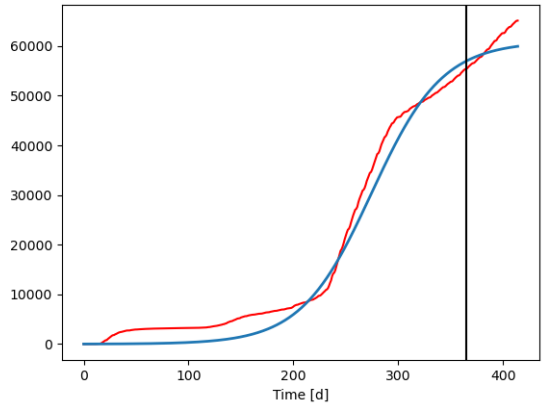

Рис. 3.6: Прогноз SEIAPR моделі після 365 днів

3.6 SEIAPHRD

Данна модель непогано апроксимує графік, проте початок у них досить сильно розбігається. Також сильно розбігаються графіки смертей та госпіталізованих. Модель спромоглася зробити досить точний прогноз на декілька днів. Похибка моделювання 2121319813, похибка прогнозування 40889031.

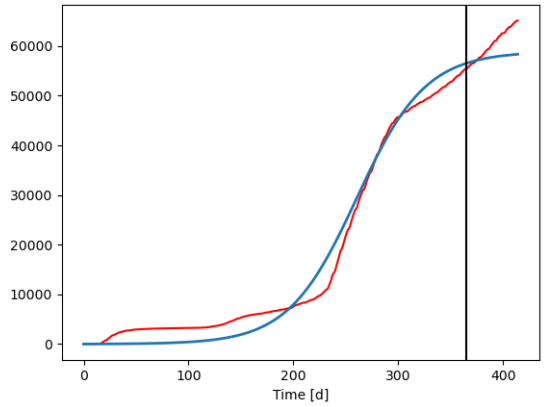

Рис. 3.7: Прогноз SEIAPHRD моделі після 365 днів

3.7 Порівняння моделей та результати

В усіх моделей є проблеми з моделюванням початку епідемії та в місцях різкої зміни динаміки розвитку захворюваності. Можливо варто використовувати змінні параметри моделі або застосувати більш складну механіку захворювання. Не зважаючи на це моделі спроможні дати досить точний короткостроковий прогноз. Даля наведена таблиця для порівняння результатів.

назва моделі	моделювання	прогнозування	прогнозування 3 дні
SIR	36695055916	142262882	787
SEIR	36697160718	131123602	5415
SEIPR	36603451786	241067088	7507
SEIAPR	36772792167	107349315	7560
SEIAPHRD	2121319813	40889031	6344

ВИСНОВКИ

Підсумовуючи нашу роботу, ми встановили, що всі розглянуті моделі моделювання розвитку епідемії, мають проблеми, особливо при моделюванні початку епідемії та в місцях різкої зміни динаміки розвитку захворюваності. Ці проблеми можуть виникати через неповну або неточну інформацію про початкові умови, особливості поширення та виявлення захворювання на початку епідемії та появи різних заходів захисту населення.

Незважаючи на проблеми, які виникають, ми також виявили, що деякі моделі спроможні дати досить точний короткостроковий прогноз. Вони можуть бути використані для оцінки поточного стану епідемії, планування необхідних заходів для контролю та протидії захворюванню та оцінки заходів захисту населення. Це може бути особливо корисним для стратегічного планування та прийняття рішень у галузі громадського здоров'я.

Чисельні методи виявляються корисними інструментами для дослідження розвитку епідемії та прогнозування короткострокових сценаріїв. Вони можуть допомогти нам краще розуміти та управляти епідеміологічними ситуаціями, сприяючи здоров'ю та безпеці громадськості.

БІБЛІОҐРАФІЯ

[1] Kaw Autar, Kalu Egwu, and Nguyen Duc. Textbook: Numerical Methods with Applications – Holistic Numerical Methods.

[2] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, January 1996. Google-Books-ID: 7Zofw3SFTWIC.

[3] Kwang Ik Kim and Zhigui Lin. Asymptotic behavior of an SEI epidemic model with diffusion. *Mathematical and Computer Modelling*, 47(11):1314–1322, 2008.

[4] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, 1 edition, November 2021.

[5] Faïçal Ndaïrou, Iván Area, Juan J. Nieto, and Delfim F. M. Torres. Mathematical modeling of COVID-19 transmission dynamics with a case study of Wuhan. *Chaos, Solitons & Fractals*, 135:109846, June 2020.

[6] Piotr Pa lka, Robert Olszewski, Ma lgorzata Kęsik-Brodacka, Agnieszka Wendland, Karolina Nowak, Ula Szczepankowska, and David Liebers. Using multiagent modeling to forecast the spatiotemporal development of the COVID-19 pandemic in Poland. *Scientific Reports*, 12:11314, July 2022.

[7] Iman Rahimi, Fang Chen, and Amir H. Gandomi. A review on COVID-19 forecasting models. *Neural Computing and Applications*, February 2021.

[8] Ayoob Salimipour, Toktam Mehraban, Hevi Seerwan Ghafour, Noreen Izza Arshad, and M. J. Ebadi. SIR model for the spread of COVID-19: A case study. *Operations Research Perspectives*, 10:100265, 2023. Publisher: Elsevier.

[9] Howard (Howie) Weiss. The SIR model and the Foundations of Public Health. *Materials Matemàtics*, 2006.

ДОДАТОК

```
float** sir(int size, float beta, float gama, int N, float dt){
    float* S = malloc(size * sizeof(float));
    float* I = malloc(size * sizeof(float));
    float* R = malloc(size * sizeof(float));
    float** answer = malloc(3 * sizeof(float*));
    answer[0] = S; answer[1] = I; answer[2] = R;
    S[0] = N - 1;I[0] = 1;R[0] = 0;answer[0] = 1;
    int step = (int)(1 / dt);float sum = 1;
    for (int i = 0; i < size - 1; i++){
        S[i + 1] = S[i] - beta * S[i] * I[i] * dt / N;
        I[i + 1] = I[i] + (beta * S[i] * I[i] / N - gama * I[i]) * dt;
        R[i + 1] = R[i] + gama * I[i] * dt;}
    return answer;
}
```

Лістинг 3.1: Метод Ейлера для SIR моделі

```
import numpy as np
import random
from tqdm import tqdm


def roy(f, n, N, iteration, w, a1, a2, A, B, eps, first):
    X = np.array([[random.uniform(A[i], B[i]) for i in range(n)] for j in
    range(N)])
    V = np.array([[random.uniform(-eps, eps) for i in range(n)] for j in range
    (N)])
    for i in range(n):
        X[0][i] = first[i]
    P = np.copy(X)
    res = np.array([f(X[i]) for i in range(N)])
    b = P[np.argmin(res)]
    for i in tqdm(range(iteration)):
        V = w * V + a1 * random.random() * (P - X) + a2 * random.random() * (b
    - X)
        X = X + V
        for j in range(N):
            res1 = f(X[j])
            if res1 < res[j]:
```

```
                P[j] = X[j]
                res[j] = res1
        b = P[np.argmin(res)]
    return b
```

Лістинг 3.2: Ройова оптимізація

```
import numpy as np
import json
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button

# my files
from optimizer_roy import *


class Param():

    def __init__(self, fig, label, init_val, min, max, n) -> None:
        self.ax = fig.add_axes([0.1, 0.4 - n * 0.03, 0.8, 0.02])
        self.slider = Slider(
                            ax=self.ax,
                            label=label,
                            valmin=min,
                            valmax=max,
                            valinit=init_val)
        self.name = label


class Graph():

    def __init__(self, name, f, T, T1, N) -> None:
        self.fig, self.ax = plt.subplots()
        self.n = 0
        self.params = []
        self.name = name
        self.function = f
        self.T = T
        self.N = N
        self.inits = []
        self.T1 = T1
        self.save = Button(self.fig.add_axes([0.6, 0.025, 0.1, 0.04]), 'Save', hovercolor='0.975')
        self.load = Button(self.fig.add_axes([0.7, 0.025, 0.1, 0.04]), 'Load', hovercolor='0.975')
        self.optimize = Button(self.fig.add_axes([0.5, 0.025, 0.1, 0.04]), 'Optimize', hovercolor='0.975')
        self.path = "C:\\Users\\nickk\\course_work\\code\\sliders\\"

```

```
    def add_parametr(self, label, init_val, min, max):
        self.params.append(Param(self.fig, label, init_val, min, max, self.n))
        self.inits.append(init_val)
        self.n += 1

    def set_params(self, params):
        for i in range(len(params)):
            self.params[i].slider.set_val(params[i])

    def save_params(self, event):
        data ={}
        for param in self.params:
            data[param.name] = param.slider.val
        with open(self.path + self.name +"_params.json", 'w') as file:
            json.dump(data, file, indent= 3)
        file.close()

    def load_params(self, event):
        with open(self.path + self.name +"_params.json") as file:
            file_content = file.read()
            file_content = json.loads(file_content)
            self.set_params(list(file_content.values()))
        file.close()

    def update(self, event):
        values = []
        for param in self.params:
            values.append(param.slider.val)
        self.line.set_ydata(self.function(*values))
        self.fig.canvas.draw_idle()

    def diff(self, params):
        values = (param for param in params )
        res = np.array(self.function(*values))[:self.T]
        return np.sum((res - self.data[:self.T])**2) + (res[-1] - self.data
[-1])**2 * 0.5

    def prognoz(self, params):
        values = (param for param in params )
        res = np.array(self.function(*values))[self.T:self.T1 + self.T]
        return np.sum((res - self.data[self.T:self.T + self.T1])**2)


    def optimizing(self, event):
        A = [param.slider.valmin for param in self.params]
        B = [param.slider.valmax for param in self.params]
        first = [param.slider.val for param in self.params]
        new_val = roy(self.diff, n = len(self.params), N = 100, iteration =
```

```
    100, w = 0.9, a1 = 1.5, a2 = 1.7, A = A, B = B, eps = 0.01, first = first)
87          print(self.diff(new_val))
88          print("prognoz =", self.prognoz(new_val))
89          self.set_params(new_val)
90
91      def load_file(self):
92          file = open("data.txt")
93          self.data = file.readline().split()
94          for i in range(len(self.data)):
95              self.data[i] = int(self.data[i])
96          file.close()
97          self.data = np.array(self.data)
98
99      def drew(self):
100         self.t = np.array([i for i in range(self.T + self.T1)])
101         self.ax.plot(self.t, self.data[:self.T + self.T1], color='r')
102         self.line, = self.ax.plot(self.t, self.function(*self.inits)[: self.T
    + self.T1], lw=2)
103         self.ax.set_xlabel('Time [d]')
104         plt.axvline(x = self.T, color = 'black')
105         plt.ylim(0, 2* self.N / 3)
106         self.fig.subplots_adjust(bottom=0.5)
107
108     def buttons(self):
109         self.save.on_clicked(self.save_params)
110         self.load.on_clicked(self.load_params)
111         self.optimize.on_clicked(self.optimizing)
112         for param in self.params:
113             param.slider.on_changed(self.update)
114         plt.show()
115
116     def preset(self):
117         self.load_file()
118         self.drew()
119         self.buttons()
```

Лістинг 3.3: Допоміжна візуалізація

```
1  from slider import *
2  import ctypes
3  from ctypes import c_float, c_int, POINTER
4
5  sir = ctypes.CDLL("C:\\Users\\nickk\\course_work\\code\\sliders\\sir\\sir.so")
6  sir.sir.argtypes = [c_int, c_float, c_float, c_int, c_float]
7  sir.sir.restype = POINTER(c_float)
8
9  T = 365
10 T1 = 20
11 dt = 0.1
```

```
N = 647601

def g(beta, gama):
    size = int((T + T1)/dt)
    result = sir.sir(size, beta, gama, N, dt)
    answer = result[:T + T1]
    sir.free_memory(result)
    return answer

init_beta = 0.12
init_gama = 0.11

graph = Graph("sir", g, T, T1, N)
graph.add_parametr("gama", init_gama, 0, 3)
graph.add_parametr("beta", init_beta, 0, 3)


graph.preset()
```

Лістинг 3.4: Приклад використання візуалізації