

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

Факультет прикладної математики

Кафедра прикладної математики

Звіт
із лабораторної роботи №2
з дисципліни «Теорія керування»
на тему
«Математичне моделювання епідемій»

Виконав:

студент групи КМ-03
Шаповалов Г. Г.

Перевірили:

Професор ПМА ФПМ
Норкін В.І.
Асистент ПМА ФПМ
Жук І.С.

Київ — 2024

Зміст

Постановка задачі.....	3
Теоретичні відомості.....	4
Порядок виконання роботи.....	5
Основна частина.....	6
Дослідження параметру N - загальне число індивідів	8
Дослідження параметру r - інтенсивність контактів індивіда	11
Дослідження параметру c - ймовірність зараження при контакті з хворим	14
Дослідження параметру u - відсоток вакцинованих за час	17
Висновок.....	20
Використана література	21
Додаток А – Код програми	22

Постановка задачі

Проста неперервна динамічна модель епідемії в популяції з N індивідів SIR-модель (Susceptible, Infected, Recovered)

Введемо наступні величини:

N - загальне число індивідів;

$S(t)$ - число здорових індивідів (без імунітету, susceptibles);

$I(t)$ - число хворих і заразних (Infected);

$R(t)$ - число індивідів з імунітетом (здорових с імунітетом та тих, що видужали з придбаним імунітетом);

$D(t)$ - число померлих;

$u(t)$ - доля вакцинованих (ізолюваних) в одиницю часу;

$p(t)dt = p(I(t)) \cdot dt$ - ймовірність заразитися здоровому за час dt ;

$S(t) \cdot p(I(t))dt$ - середнє число заражених за час dt ;

$\alpha \cdot dt$ - ймовірність одужання інфікованого за час dt ;

$\beta \cdot dt$ - ймовірність смерті інфікованого за час dt ;

$u(t) \cdot dt$ - (керований) відсоток вакцинованих за час dt ;

$S(t) \cdot u(t)dt$ - число вакцинованих за час dt .

Модель зараження (ймовірності зараження $p(I)$)

Нехай

r - інтенсивність контактів індивіда, $r \cdot dt$ - число контактів за час dt ;

c - ймовірність зараження при контакті з хворим.

Вочевидь, I/N - ймовірність зустріти хворого при будь-якому контакті.

Яка ймовірність заразитися за час dt , тобто за $r \cdot dt$ контактів?

Відмітимо, що

$rdt \frac{I}{N}$ - число контактів з хворими за час dt ,

$(1-c)$ - ймовірність не заразитися при контакті з хворим,

$(1-c)^{rdt \frac{I}{N}}$ - ймовірність не заразитися при $rdt \frac{I}{N}$ контактах,

$1 - (1-c)^{rdt \frac{I}{N}}$ - ймовірність заразитися при $rdt \frac{I}{N}$ контактах,

$$1 - (1-c)^{rdt \frac{I}{N}} = 1 - \exp\left(-\frac{r \log(1-c)I}{N} dt\right) \approx 1 - \left(1 - \frac{r \log(1-c)I}{N} dt\right) = \frac{r \log(1-c)I}{N} dt.$$

Таким чином інтенсивність зараження (ймовірність зараження в одиницю часу)

$$\text{дорівнює } p(I) = -\frac{rI \log(1-c)}{N}.$$

Теоретичні відомості

Модель 3. Модель зі зворотним зв'язком

В моделі зі зворотним зв'язком керовані параметри (цими параметрами можуть бути u, r, c) залежать від поточного стану системи (S, I, R, D) . Наприклад, інтенсивність контактів r може бути залежною від кількості інфікованих (або померлих), $r = r(I) = r_0 / (I/N)^\gamma$ або $r = r(I) = r_0 e^{-\gamma I/N}$, $\gamma \geq 0$; інтенсивність вакцинації (у випадку наявності вакцини) також може залежати від кількості інфікованих (або померлих), $u = u(I) = \min\{u_{\max}, u_0 (I/N)^\delta\}$, $\delta \geq 0$; ймовірність підхопити інфекцію c також може бути керованою величиною (залежить від правил соціального дистанціювання та строгості індивідуальних захисних заходів), $c = c(I) = c_0 / (I/N)^\lambda$, $\lambda \geq 0$. Пошук цих та інших функціональних форм $r(I)$, $u(I)$, $c(I)$ є важливою проблемою керування епідемією.

Порядок виконання роботи

Завдання.

Промодельовати розвиток епідемії, розв'язав задачу Коши для системи диференціальних рівнянь для різних наборів параметрів моделі

$(N, r, c, q, u(t) = \text{const})$ та керувань $u(t) \in U = \{u : 0 \leq u \leq u_{\max} < 1\}$

В тому числі зі зворотним зв'язком $(u(I/N), r(I/N), c(I/N))$.

Порядок виконання роботи.

- 1) Обрати модель M1 – M5.
- 2) Обрати значення параметрів моделі $(N, r, c, q, q_0, \alpha, \beta, u_{\max})$ відповідно їх змісту (орієнтуватися на пандемію COVID-19).
- 3) Обрати початкові значення, наприклад, $I(0) > 0$, $S(0) = N - I(0)$, $J(0) = 0$, $R(0) = 0$, $D(0) = 0$.
- 4) Обрати проміжок часу $[0, T]$.
- 5) Обрати програму для розв'язання нелінійної системи диференціальних рівнянь (solver, наприклад, ode23 в системі Matlab, або відповідну програму в інших системах програмування).
- 6) Розв'язати задачу Коши на відрізок часу $[0, T]$.
- 7) Візуалізувати розв'язок, тобто побудувати графіки функцій $S(t), I(t), R(t), D(t)$. На графіках вказати назву графіку, назву осей координат, легенду, значення основних параметрів моделі.
- 8) Обчислити основні характеристики епідемічного процесу для обраного набору параметрів моделі (пік процесу - $I_{\max} = \max_{t \in [0, T]} I(t)$, $t_{\max} = \arg \max_{t \in [0, T]} I(t)$; тривалість епідемії; кінцевий результат епідемії, $S(\infty), I(\infty), J(\infty), R(\infty), D(\infty)$).
- 9) Дослідити зміни розвитку епідемії (картину) для різних значень параметрів моделі $(N, r, c, q, q_0, u(t) = \text{const})$.
- 10) Дослідити зміни розвитку епідемії (картину) для різних керувань $u(t) \in U = \{u : 0 \leq u \leq u_{\max} < 1\}$, в тому числі зі зворотним зв'язком $u(I/N), r(I/N), c(I/N)$.
- 11) Підготувати звіт про роботу в електронному вигляді (з графіками, висновками і лістингом програми).
- 12) Надіслати звіт викладачеві на електронну адресу.

Основна частина

У цій дослідженні було використано Модель №3 зі зворотнім зв'язком для симуляції епідемії. Керовані параметри цієї моделі змінюються відповідно до поточного стану системи. Для моделювання процесу епідемії було розроблено систему диференціальних рівнянь, які вирішуються за допомогою методу Рунге-Кутта 4-го порядку.

Давайте спробуємо провести моделювання епідемії, використовуючи наступні дані:

$N = 1000$	$R = 0$	$C0 = 0.5$	$\alpha = 0.1$
$S = 900$	$D = 0$	$R0 = 5$	$\beta = 0.05$
$I = 100$		$U0 = 0.1$	$T = [0, 50]$

Де $\alpha \cdot dt$ - ймовірність одужання інфікованого за час dt ;
 $\beta \cdot dt$ - ймовірність смерті інфікованого за час dt ;

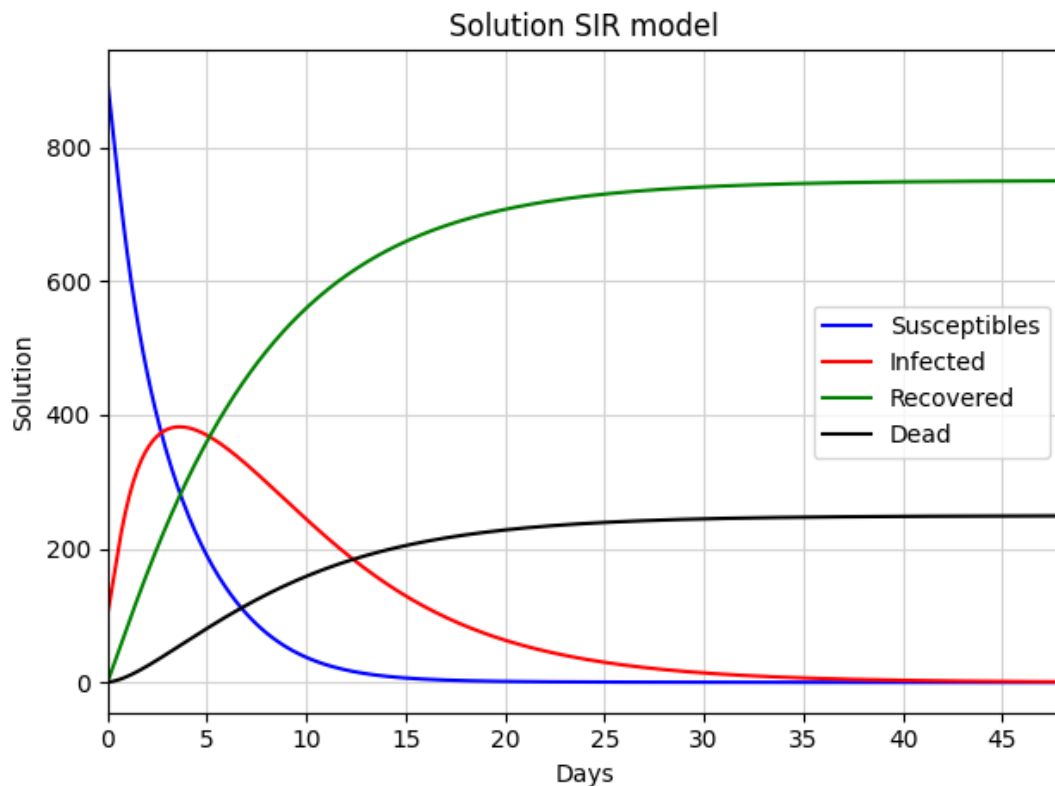


Рис. 1 – Змодельована епідемія

Дана епідемія мала такі результати:

Пік інфікованих = 382 у 4 день

Кількість suspected індивідів: 0

Кількість infected індивідів: 0

Кількість recovered індивідів: 751

Кількість dead індивідів: 249

Кількість днів, які тривала епідемія: 47

У результаті епідемії, за 47 днів імунітет набули 751 індивідів та 249 індивідів померли. Це $\frac{3}{4}$ та $\frac{1}{4}$ відповідно від загальної кількості населення.

Дослідження параметру N - загальне число індивідів

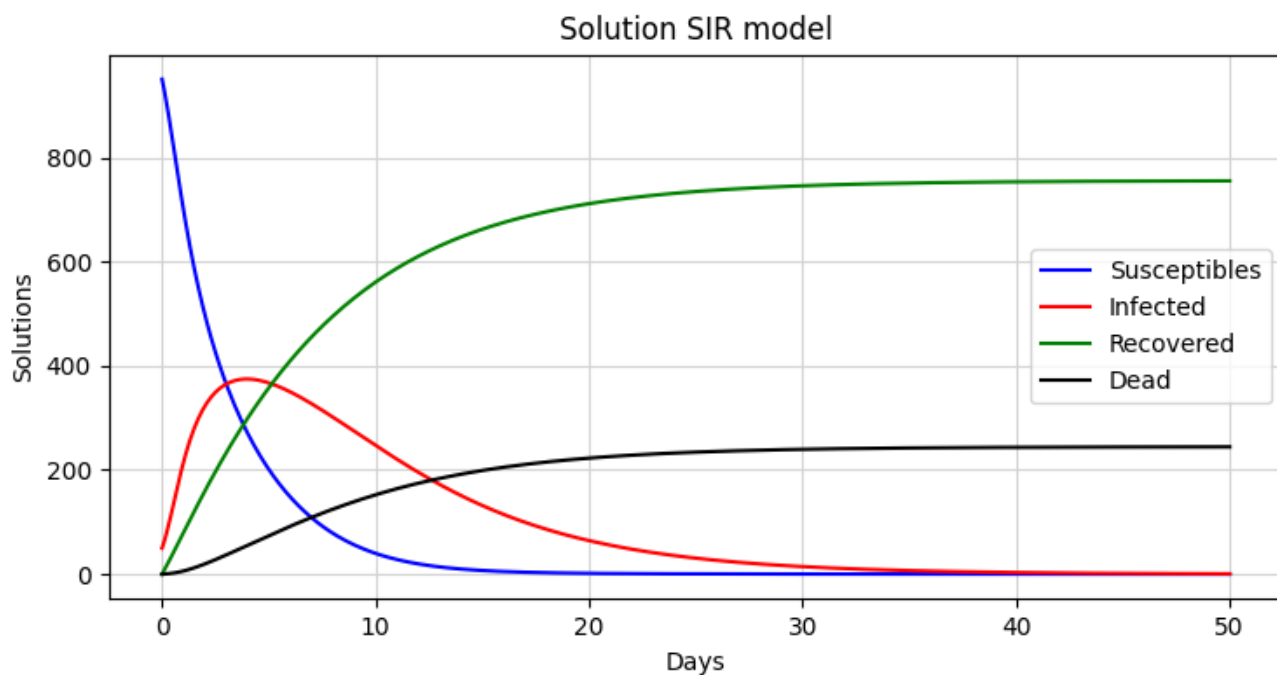


Рис. 2 – Змодельована епідемія при $N = 1000$

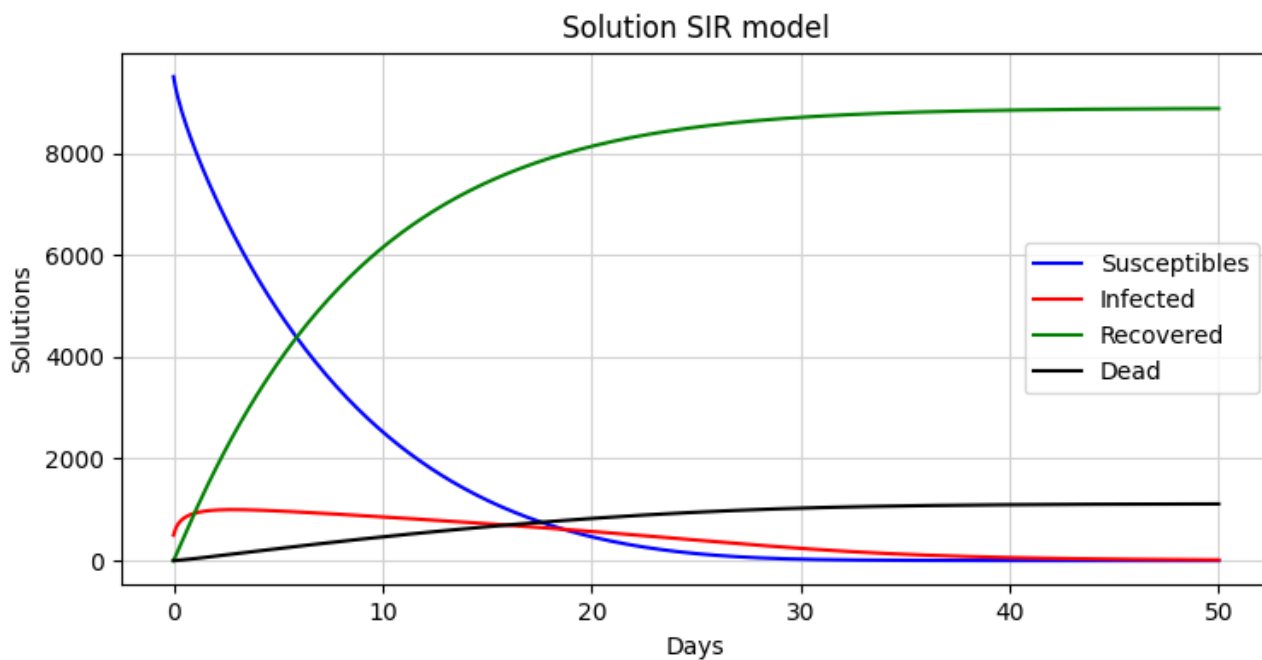


Рис. 3 – Змодельована епідемія при $N = 10000$

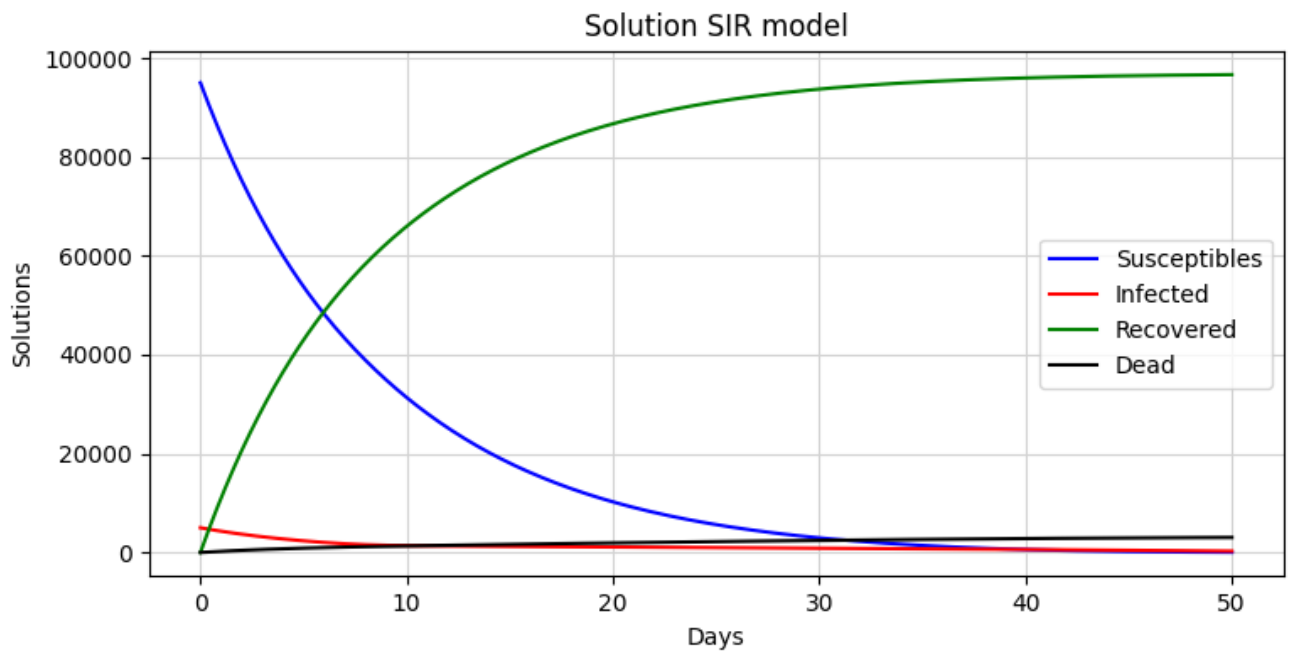


Рис. 4 – Змодельована епідемія при $N = 100000$

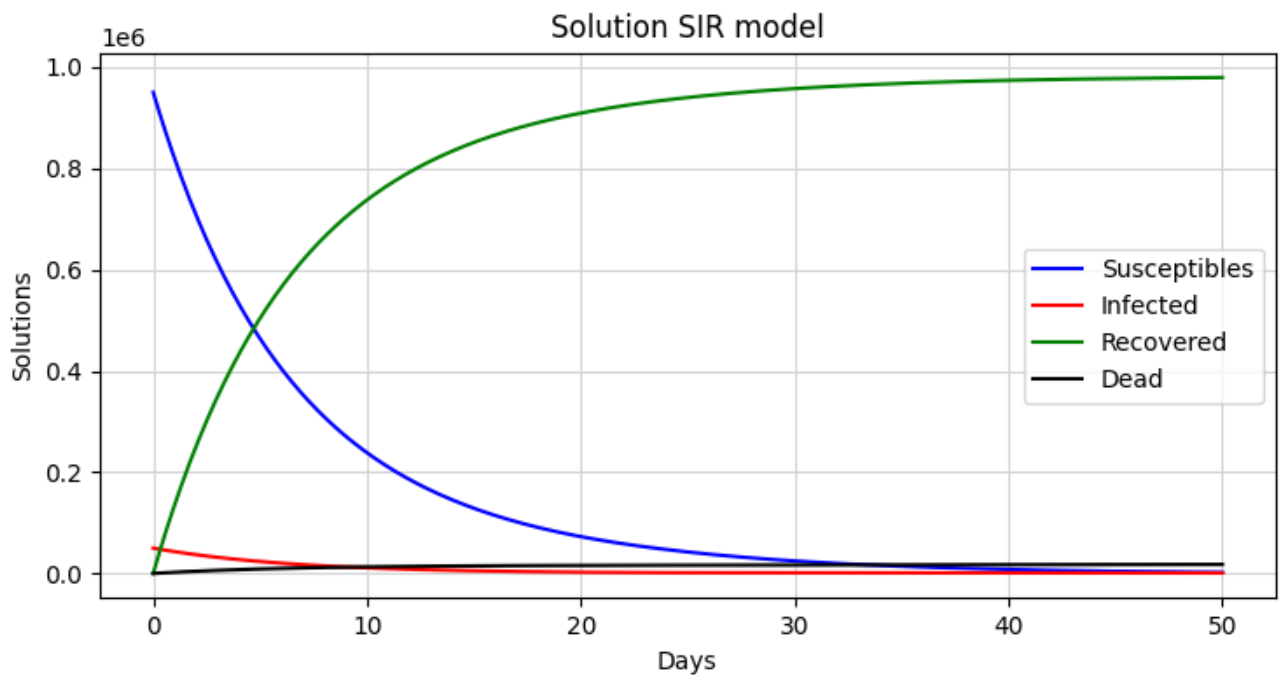


Рис. 5 – Змодельована епідемія при $N = 1000000$

Для цього дослідження було використано такі параметри:

$R = 0$	$C0 = 0.5$	$\alpha = 0.1$
$D = 0$	$R0 = 5$	$\beta = 0.05$
	$U0 = 0.1$	$T = [0, 50]$

Де $\alpha \cdot dt$ - ймовірність одужання інфікованого за час dt ;

$\beta \cdot dt$ - ймовірність смерті інфікованого за час dt ;

При $N = 1000$ та $N = 10000$ можемо побачити, що кількість інфікованих та померлих є суттєвою, але при $N = 100000$ та $N = 1000000$ ситуація зворотня. За великої кількості населення епідемія не поширюється так, як при малій кількості, це може бути пов'язано з тим, що коефіцієнт ймовірність зараження при контакті з хворим було обрано досить низьким, а саме 0.5

Дослідження параметру r - інтенсивність контактів індивіда

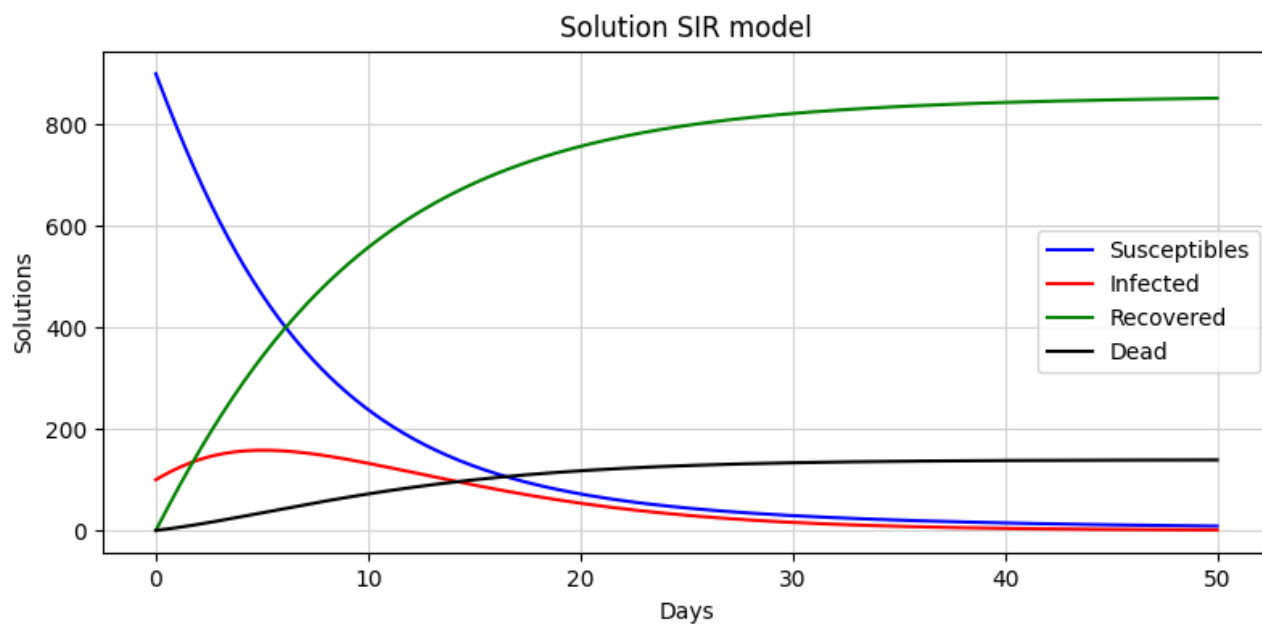


Рис. 6 – Змодельована епідемія при $r_0 = 1$

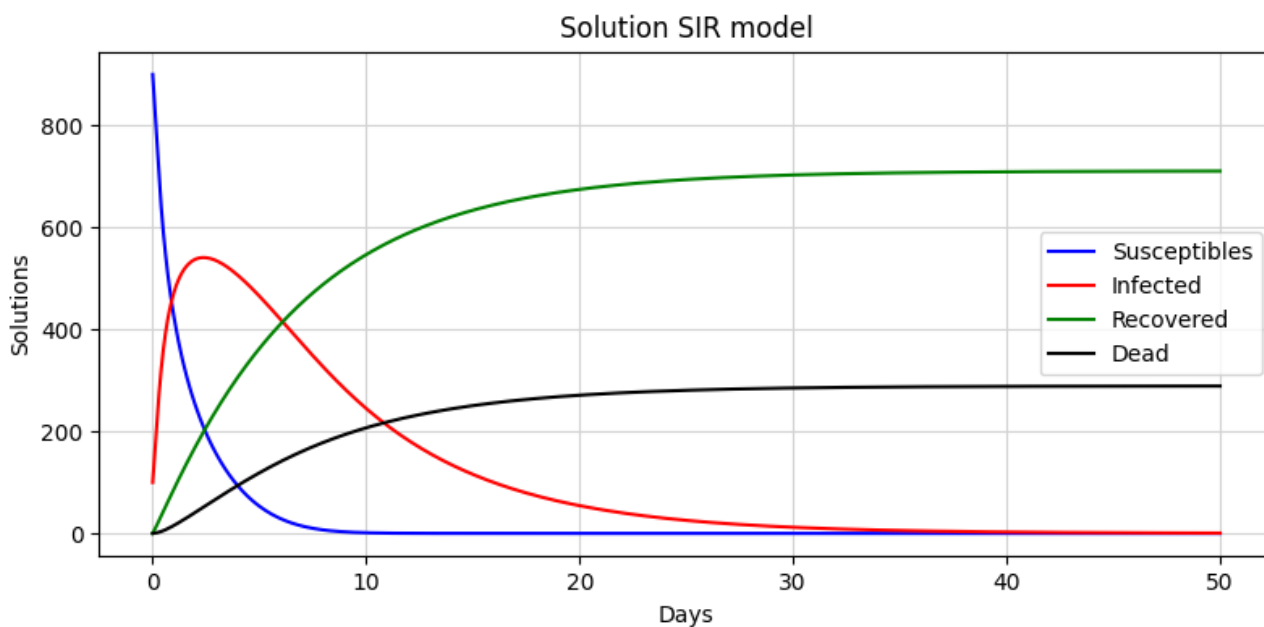


Рис. 7 – Змодельована епідемія при $r_0 = 15$

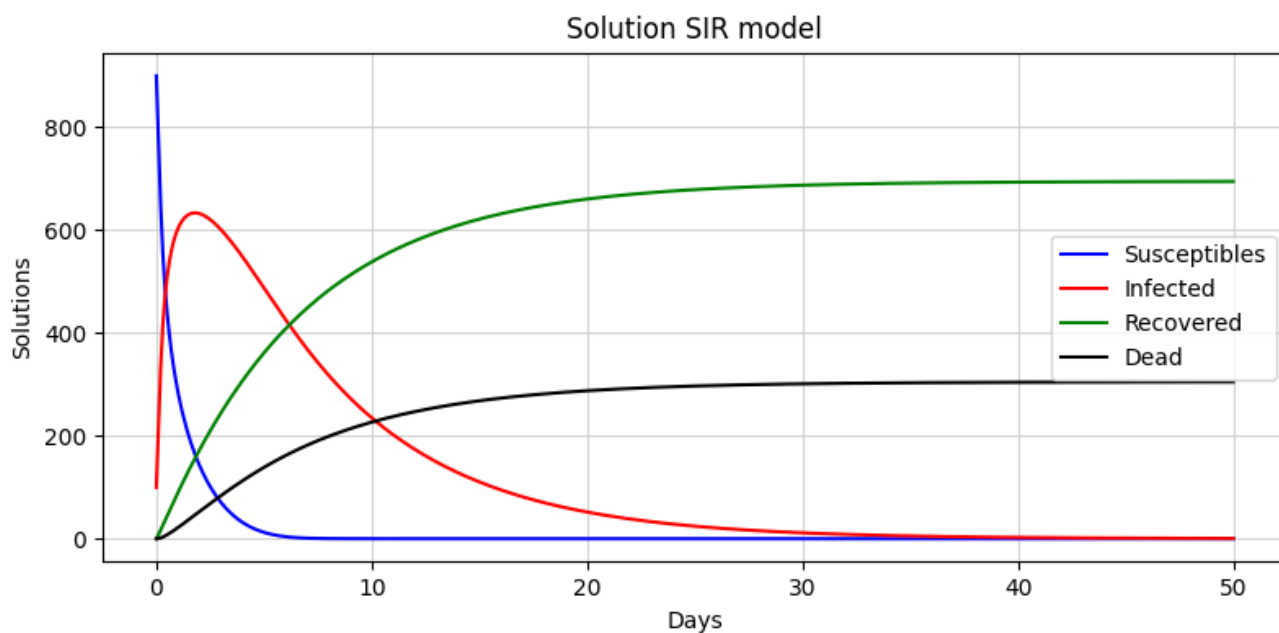


Рис. 8 – Змодельована епідемія при $r_0 = 30$

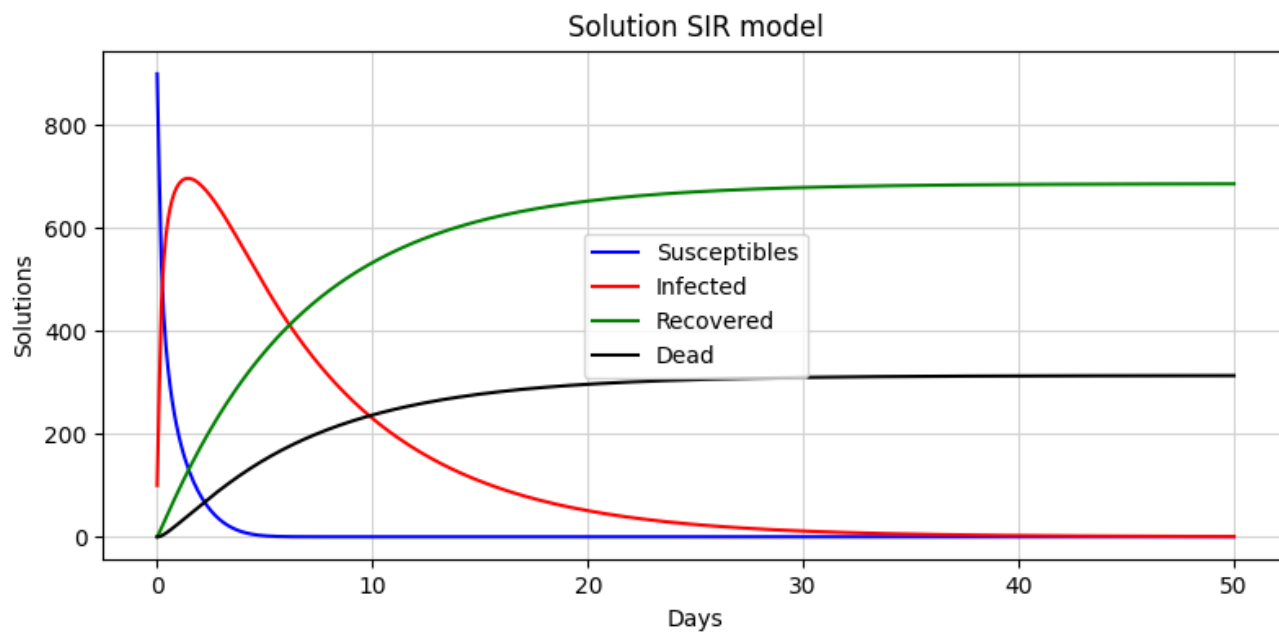


Рис. 9 – Змодельована епідемія при $r_0 = 50$

Для цього дослідження було використано такі параметри:

$$N = 1000$$

$$R = 0$$

$$C_0 = 0.5$$

$$\alpha = 0.1$$

$$S = 900$$

$$D = 0$$

$$U_0 = 0.1$$

$$\beta = 0.05$$

$$I = 100$$

$$T = [0, 50]$$

Де $\alpha \cdot dt$ - ймовірність одужання інфікованого за час dt ;
 $\beta \cdot dt$ - ймовірність смерті інфікованого за час dt ;

Можна зробити висновок, що кількість взаємодій, що відбуваються у людини, має великий вплив на темпи розповсюдження інфекції в середовищі та тривалість епідемії. Зі зростанням цієї кількості інфекція розповсюджується швидше, що може в результаті призвести до формування імунітету або навіть призвести до загибелі людей.

Дослідження параметру c - ймовірність зараження при контакті з хворим

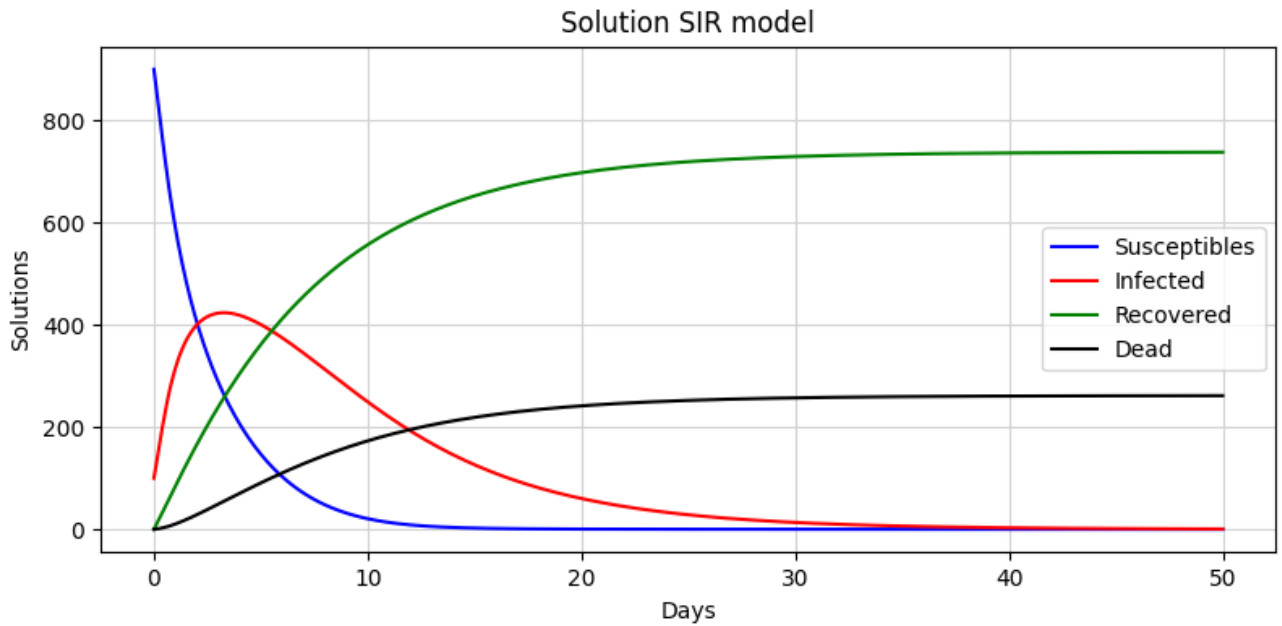


Рис. 10 – Змодельована епідемія при $c_0 = 0.6$

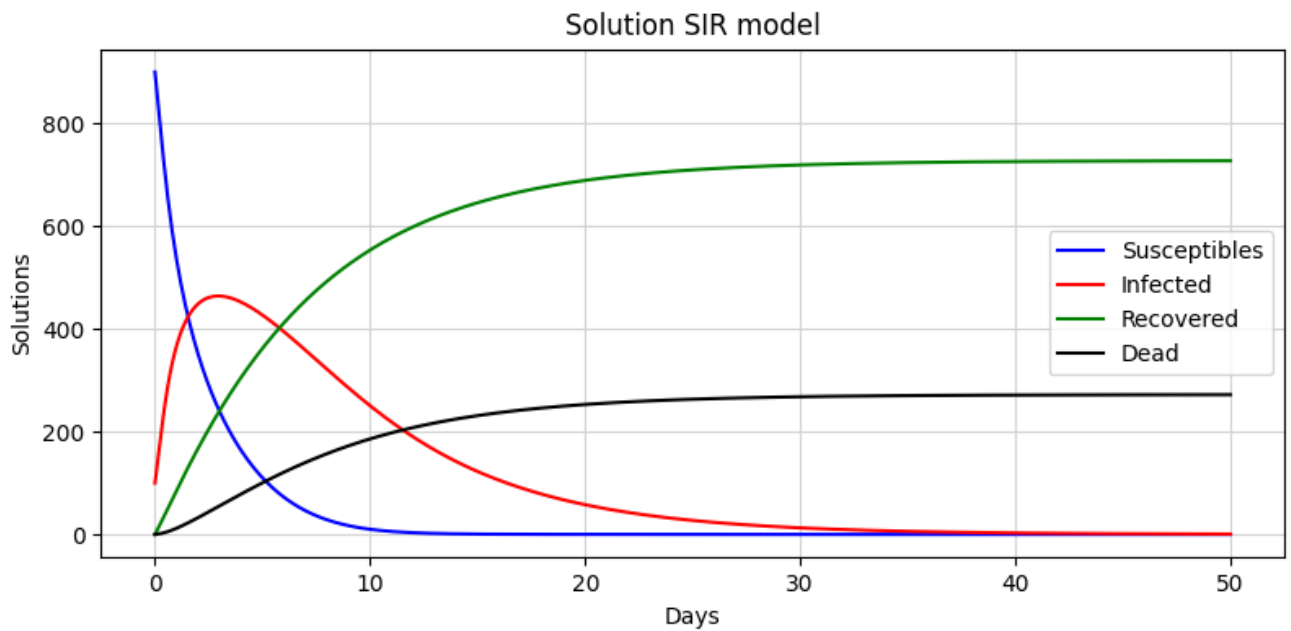


Рис. 11 – Змодельована епідемія при $c_0 = 0.7$

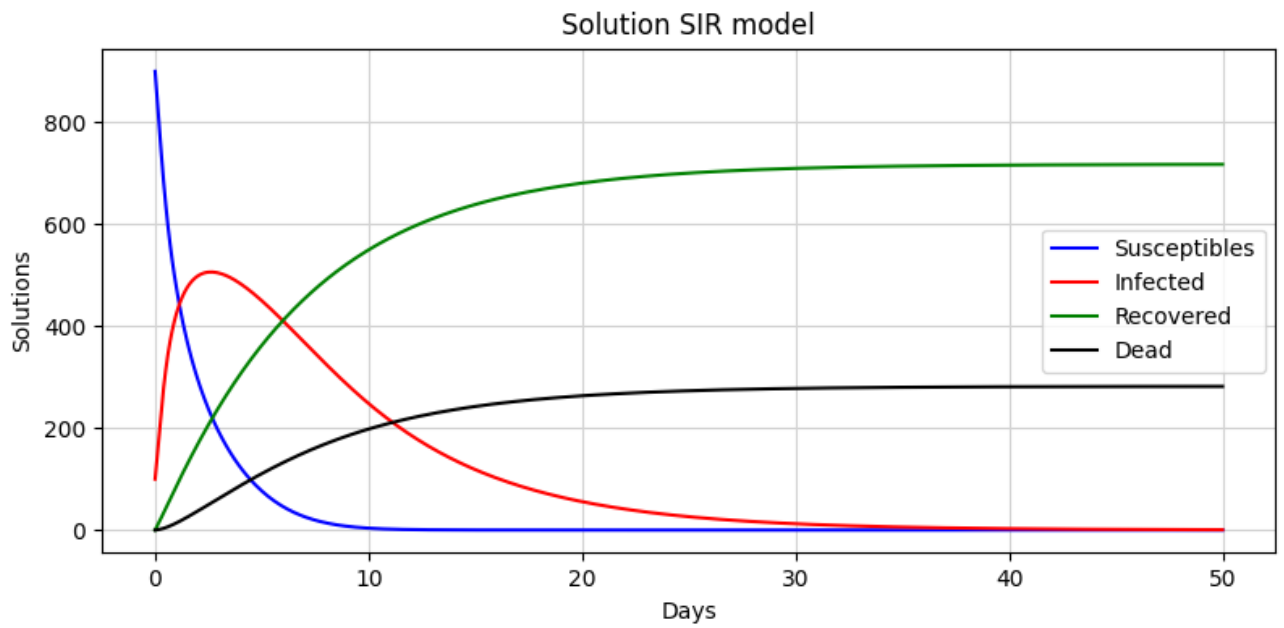


Рис. 12 – Змодельована епідемія при $c_0 = 0.8$

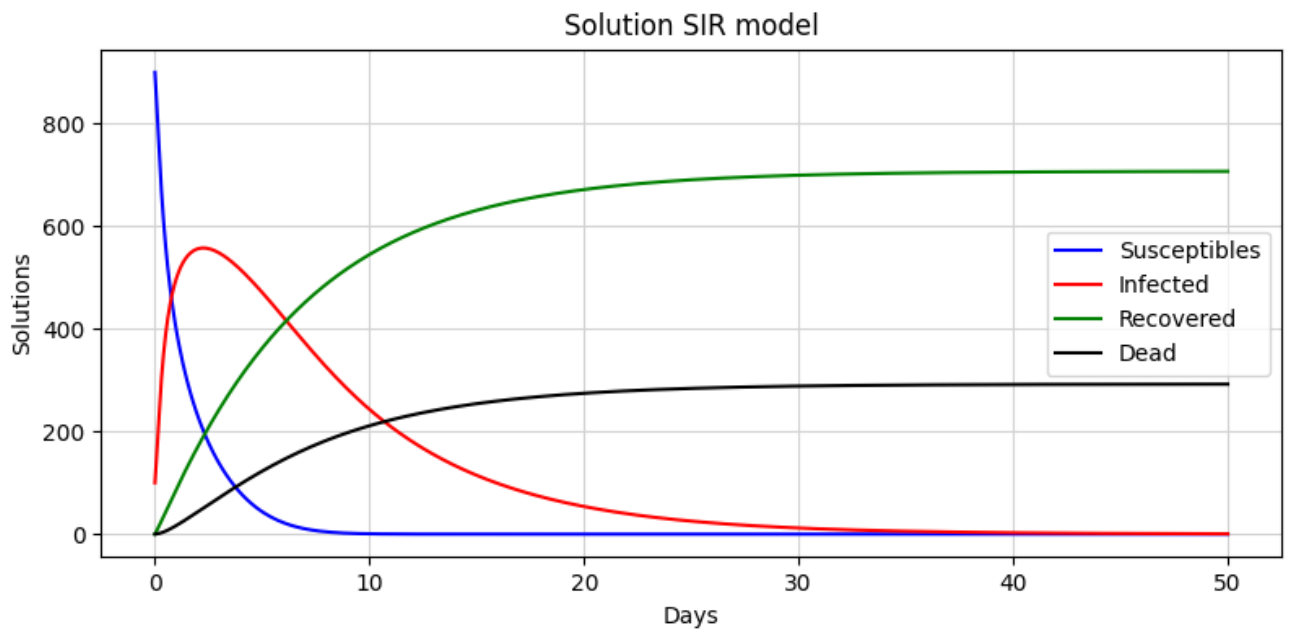


Рис. 13 – Змодельована епідемія при $c_0 = 0.9$

Для цього дослідження було використано такі параметри:

$N = 1000$	$R = 0$	$R_0 = 5$	$\alpha = 0.1$
$S = 900$	$D = 0$	$U_0 = 0.1$	$\beta = 0.05$
$I = 100$			$T = [0, 50]$

Де $\alpha \cdot dt$ - ймовірність одужання інфікованого за час dt ;

$\beta \cdot dt$ - ймовірність смерті інфікованого за час dt ;

Зміна цього параметра також впливає на швидкість поширення інфекції серед населення, час досягнення піку інфікованості людей та тривалість епідемії. Проте, для конкретних умов моделі, зміна цього параметра має незначний вплив на загальну епідеміологічну ситуацію.

Дослідження параметру u - відсоток вакцинованих за час

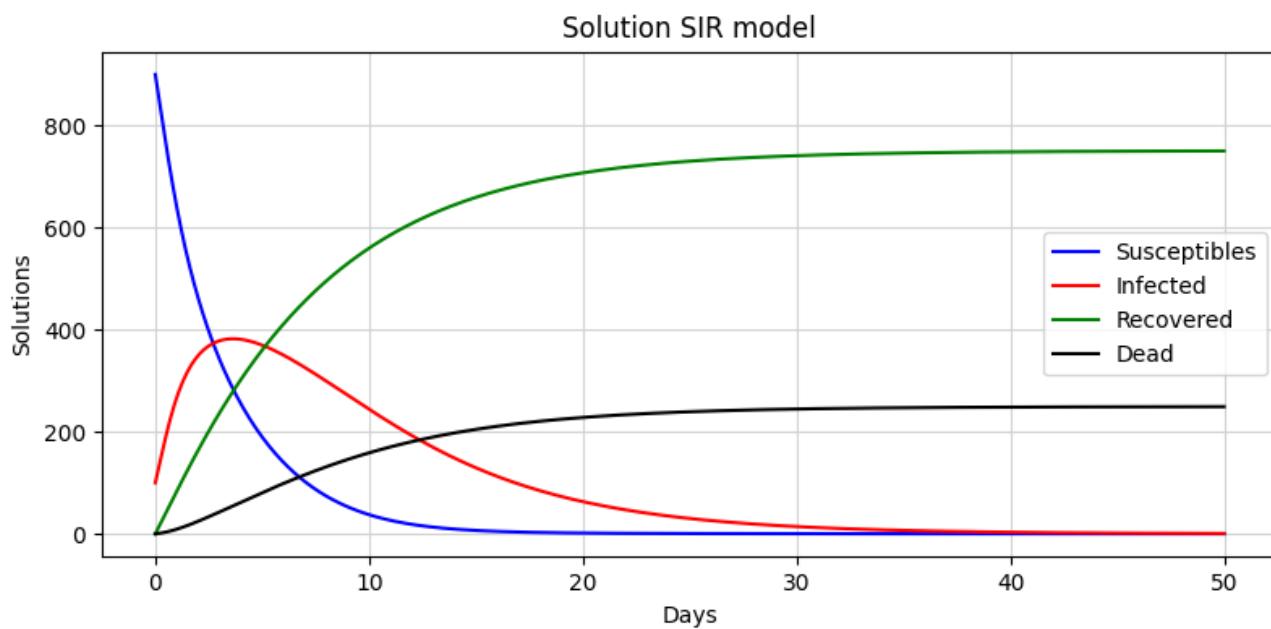


Рис. 14 – Змодельована епідемія при $u_0 = 0.1$

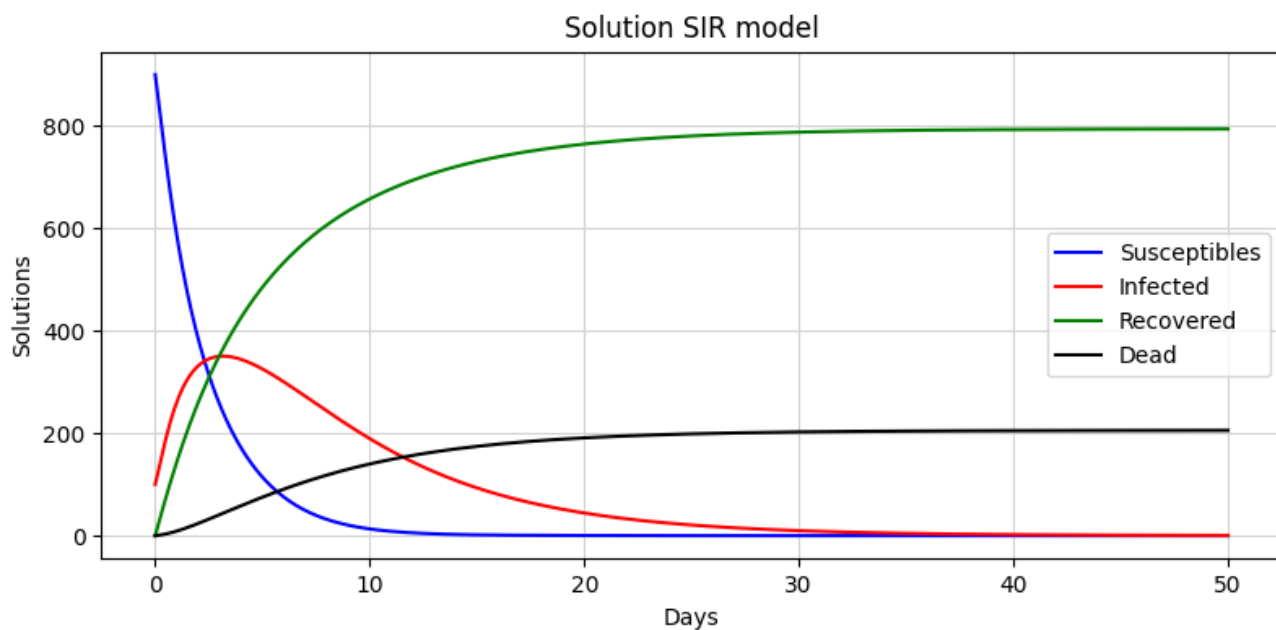


Рис. 15 – Змодельована епідемія при $u_0 = 0.2$

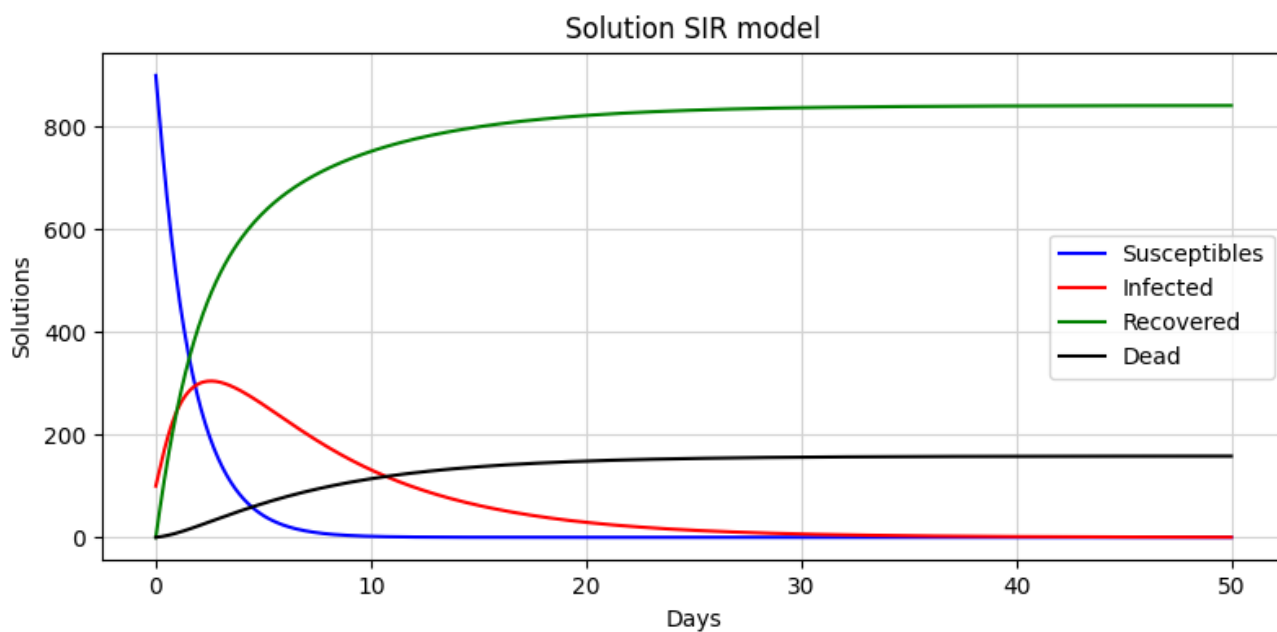


Рис. 16 – Змодельована епідемія при $u_0 = 0.4$

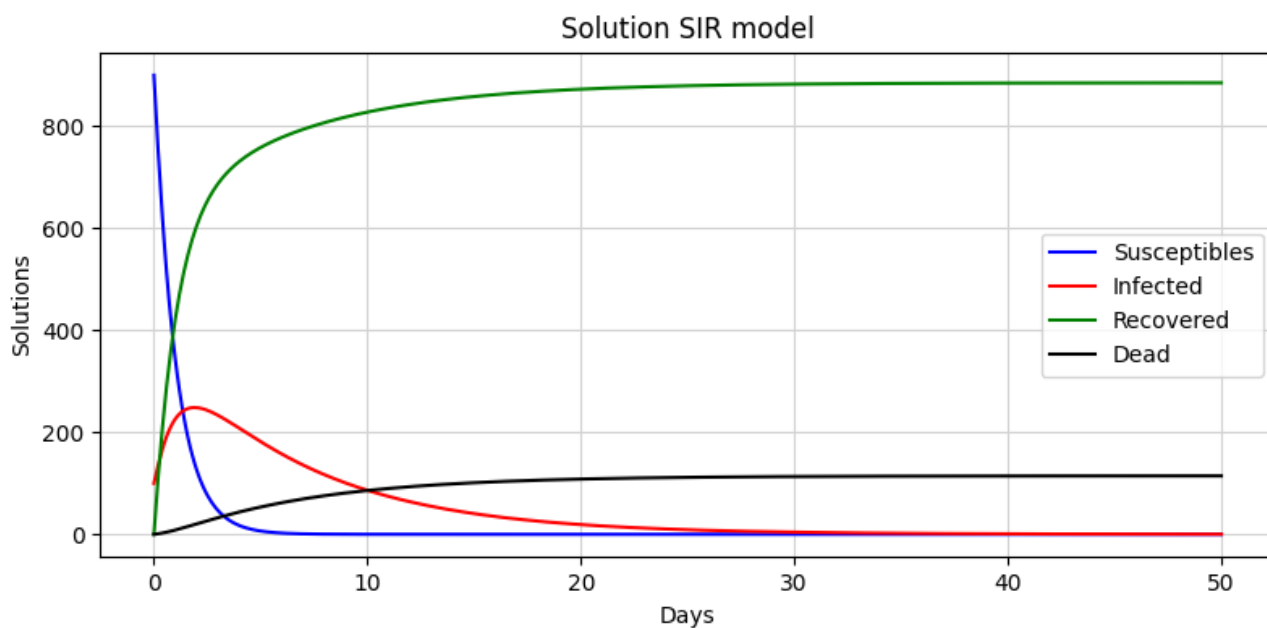


Рис. 17 – Змодельована епідемія при $u_0 = 0.8$

Для цього дослідження було використано такі параметри:

$$N = 1000$$

$$R = 0$$

$$C_0 = 0.5$$

$$\alpha = 0.1$$

$$S = 900$$

$$D = 0$$

$$R_0 = 5$$

$$\beta = 0.05$$

$$I = 100$$

$$T = [0, 50]$$

Де $\alpha \cdot dt$ - ймовірність одужання інфікованого за час dt ;
 $\beta \cdot dt$ - ймовірність смерті інфікованого за час dt ;

Це дослідження показує вплив даного регулюючого параметра на епідеміологічну обстановку, зі збільшенням значення параметра зростає кількість осіб, що набувають імунітет, та спостерігається прискорене закінчення епідемії.

Це передбачувано, оскільки зі зростанням цього коефіцієнта більше людей отримують вакцинацію від хвороби.

Висновок

Під час виконання даної лабораторної роботи було проаналізовано основні концепції формування математичних моделей поширення захворювань, вивчено вплив різноманітних факторів на тривалість та інтенсивність захворювання. Розроблено програму для математичного моделювання епідемій.

Основні висновки щодо досліджених параметрів:

1. При низькому значення коефіцієнта ймовірності зараження при контакті з хворим епідемія поширюється швидше в малочисельних групах. Коли кількість людей велика люди встигають отримати імунітет швидше.
2. Кількість контактів, що відбуваються у людини, має великий вплив на темпи розповсюдження інфекції в середовищі та тривалість епідемії. Зі зростанням цієї кількості інфекція розповсюджується швидше.
3. Ймовірність зараження при контакті з хворим також впливає на швидкість поширення інфекції серед населення, час досягнення піку інфікованості людей та тривалість епідемії.
4. Зі збільшенням відсотку вакцинованих зростає кількість осіб, що набувають імунітет, та спостерігається прискорене закінчення епідемії.

Використана література

1. Методичні вказівки до лабораторної роботи.
2. A. Huppert, G. Katriel, Mathematical modelling and prediction in infectious disease epidemiology, Clin. Microbiol. Infection 19 (2013), 999-1005.
3. Яушева О.А. 2016 - Математическая модель эпидемии лихорадки Эбола (SIR модель). Санкт-Петербургский государственный университет, Кафедра диагностики функциональных систем. Санкт-Петербург, 2016.
4. И.Д.Колесин, Е.М.Житкова. Математические модели эпидемий: Учебное пособие. - СПб.: НИИФ СПбГУ, 2004. - с.92.

Додаток А – Код програми

```
import numpy as np
import matplotlib.pyplot as plt

def u(_i, _u0):
    return min(umax, _u0*((_i/N)**delta))

def p(_i, _r0, _c0):
    _r = _r0*np.exp(-gamma*_i/N)
    _c = _c0/((_i/N)**lamda)
    _p = -_r*_i*np.log(1-_c)/N
    return _p

def st(_s, _i, _r0, _c0, _u0):
    return -_s*p(_i, _r0, _c0) - _s*u(_i, _u0)

def it(_s, _i, _r0, _c0):
    return _s*p(_i, _r0, _c0) - alpha*_i - beta*_i

def rt(_s, _i, _u0):
    return alpha*_i + _s*u(_i, _u0)

def dt(_i):
    return beta*_i

def rk4(_s, _i, _r, _d, _h, _r0, _c0, _u0):
    k1 = _h * st(_s, _i, _r0, _c0, _u0)
    q1 = _h * it(_s, _i, _r0, _c0)
    l1 = _h * rt(_s, _i, _u0)
    m1 = _h * dt(_i)
    k2 = _h * st(_s, _i, _r0, _c0, _u0)
    q2 = _h * it(_s, _i, _r0, _c0)
    l2 = _h * rt(_s, _i, _u0)
    m2 = _h * dt(_i)
    k3 = _h * st(_s, _i, _r0, _c0, _u0)
    q3 = _h * it(_s, _i, _r0, _c0)
```

```

l3 = _h * rt(_s, _i, _u0)
m3 = _h * dt(_i)
k4 = _h * st(_s, _i, _r0, _c0, _u0)
q4 = _h * it(_s, _i, _r0, _c0)
l4 = _h * rt(_s, _i, _u0)
m4 = _h * dt(_i)
s_next = _s + (k1 + 2 * k2 + 2 * k3 + k4) / 6
i_next = _i + (q1 + 2 * q2 + 2 * q3 + q4) / 6
r_next = _r + (l1 + 2 * l2 + 2 * l3 + l4) / 6
d_next = _d + (m1 + 2 * m2 + 2 * m3 + m4) / 6
return s_next, i_next, r_next, d_next

def test_n():
    N = [1_000, 10_000, 100_000, 1_000_000]
    s = [950, 9_500, 95_000, 950_000]
    i = [ 50,   500,   5_000,  50_000]
    s_res = []
    i_res = []
    r_res = []
    d_res = []
    for k in range(len(N)):
        s_list = [s[k]]
        i_list = [i[k]]
        r_list = [r]
        d_list = [d]
        for j in range(step):
            res1, res2, res3, res4 = rk4(s_list[j], i_list[j], r_list[j],
d_list[j], h, r0, c0, u0)
            s_list.append(res1)
            i_list.append(res2)
            r_list.append(res3)
            d_list.append(res4)

        s_list = np.array(s_list)
        i_list = np.array(i_list)
        r_list = np.array(r_list)
        d_list = np.array(d_list)
        s_res.append(s_list)
        i_res.append(i_list)
        r_res.append(r_list)
        d_res.append(d_list)

    x = np.arange(t0, t + h, h)
    plt.subplot(2, 2, 1)
    plt.title('Solution SIR model')
    plt.plot(x, s_res[0], color='blue', label="Susceptibles")

```

```

plt.plot(x, i_res[0], color='red', label="Infected")
plt.plot(x, r_res[0], color='green', label="Recovered")
plt.plot(x, d_res[0], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')

plt.subplot(2, 2, 2)
plt.title('Solution SIR model')
plt.plot(x, s_res[1], color='blue', label="Susceptibles")
plt.plot(x, i_res[1], color='red', label="Infected")
plt.plot(x, r_res[1], color='green', label="Recovered")
plt.plot(x, d_res[1], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')

plt.subplot(2, 2, 3)
plt.title('Solution SIR model')
plt.plot(x, s_res[2], color='blue', label="Susceptibles")
plt.plot(x, i_res[2], color='red', label="Infected")
plt.plot(x, r_res[2], color='green', label="Recovered")
plt.plot(x, d_res[2], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')

plt.subplot(2, 2, 4)
plt.title('Solution SIR model')
plt.plot(x, s_res[3], color='blue', label="Susceptibles")
plt.plot(x, i_res[3], color='red', label="Infected")
plt.plot(x, r_res[3], color='green', label="Recovered")
plt.plot(x, d_res[3], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')

```



```

plt.ylabel('Solutions')
plt.tight_layout()
plt.show()

def test_r():
    r_test = [1, 15, 30, 50]
    s_res = []
    i_res = []
    r_res = []
    d_res = []
    for k in range(len(r_test)):
        s_list = [s]
        i_list = [i]
        r_list = [r]
        d_list = [d]
        for j in range(step):
            res1, res2, res3, res4 = rk4(s_list[j], i_list[j], r_list[j],
d_list[j], h, r_test[k], c0, u0)
            s_list.append(res1)
            i_list.append(res2)
            r_list.append(res3)
            d_list.append(res4)

        s_list = np.array(s_list)
        i_list = np.array(i_list)
        r_list = np.array(r_list)
        d_list = np.array(d_list)
        s_res.append(s_list)
        i_res.append(i_list)
        r_res.append(r_list)
        d_res.append(d_list)

    x = np.arange(t0, t + h, h)
    plt.subplot(2, 2, 1)
    plt.title('Solution SIR model')
    plt.plot(x, s_res[0], color='blue', label="Susceptibles")
    plt.plot(x, i_res[0], color='red', label="Infected")
    plt.plot(x, r_res[0], color='green', label="Recovered")
    plt.plot(x, d_res[0], color='black', label="Dead")
    plt.legend()
    plt.grid(c='lightgrey')
    ax = plt.gca()
    ax.set_axisbelow(True)
    plt.xlabel('Days')
    plt.ylabel('Solutions')
    plt.tight_layout()

```

```

plt.subplot(2, 2, 2)
plt.title('Solution SIR model')
plt.plot(x, s_res[1], color='blue', label="Susceptibles")
plt.plot(x, i_res[1], color='red', label="Infected")
plt.plot(x, r_res[1], color='green', label="Recovered")
plt.plot(x, d_res[1], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 3)
plt.title('Solution SIR model')
plt.plot(x, s_res[2], color='blue', label="Susceptibles")
plt.plot(x, i_res[2], color='red', label="Infected")
plt.plot(x, r_res[2], color='green', label="Recovered")
plt.plot(x, d_res[2], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 4)
plt.title('Solution SIR model')
plt.plot(x, s_res[3], color='blue', label="Susceptibles")
plt.plot(x, i_res[3], color='red', label="Infected")
plt.plot(x, r_res[3], color='green', label="Recovered")
plt.plot(x, d_res[3], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()
plt.show()

def test_c():
    c_test = [0.6, 0.7, 0.8, 0.9]

```

```

s_res = []
i_res = []
r_res = []
d_res = []
for k in range(len(c_test)):
    s_list = [s]
    i_list = [i]
    r_list = [r]
    d_list = [d]
    for j in range(step):
        res1, res2, res3, res4 = rk4(s_list[j], i_list[j], r_list[j],
d_list[j], h, r0, c_test[k], u0)
        s_list.append(res1)
        i_list.append(res2)
        r_list.append(res3)
        d_list.append(res4)

    s_list = np.array(s_list)
    i_list = np.array(i_list)
    r_list = np.array(r_list)
    d_list = np.array(d_list)
    s_res.append(s_list)
    i_res.append(i_list)
    r_res.append(r_list)
    d_res.append(d_list)

x = np.arange(t0, t + h, h)
plt.subplot(2, 2, 1)
plt.title('Solution SIR model')
plt.plot(x, s_res[0], color='blue', label="Susceptibles")
plt.plot(x, i_res[0], color='red', label="Infected")
plt.plot(x, r_res[0], color='green', label="Recovered")
plt.plot(x, d_res[0], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 2)
plt.title('Solution SIR model')
plt.plot(x, s_res[1], color='blue', label="Susceptibles")
plt.plot(x, i_res[1], color='red', label="Infected")
plt.plot(x, r_res[1], color='green', label="Recovered")
plt.plot(x, d_res[1], color='black', label="Dead")

```

```

plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 3)
plt.title('Solution SIR model')
plt.plot(x, s_res[2], color='blue', label="Susceptibles")
plt.plot(x, i_res[2], color='red', label="Infected")
plt.plot(x, r_res[2], color='green', label="Recovered")
plt.plot(x, d_res[2], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 4)
plt.title('Solution SIR model')
plt.plot(x, s_res[3], color='blue', label="Susceptibles")
plt.plot(x, i_res[3], color='red', label="Infected")
plt.plot(x, r_res[3], color='green', label="Recovered")
plt.plot(x, d_res[3], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()
plt.show()

def test_u():
    u_test = [0.1, 0.2, 0.4, 0.8]
    s_res = []
    i_res = []
    r_res = []
    d_res = []
    for k in range(len(u_test)):
        s_list = [s]
        i_list = [i]

```

```

        r_list = [r]
        d_list = [d]
        for j in range(step):
            res1, res2, res3, res4 = rk4(s_list[j], i_list[j], r_list[j],
d_list[j], h, r0, c0, u_test[k])
            s_list.append(res1)
            i_list.append(res2)
            r_list.append(res3)
            d_list.append(res4)

        s_list = np.array(s_list)
        i_list = np.array(i_list)
        r_list = np.array(r_list)
        d_list = np.array(d_list)
        s_res.append(s_list)
        i_res.append(i_list)
        r_res.append(r_list)
        d_res.append(d_list)

x = np.arange(t0, t + h, h)
plt.subplot(2, 2, 1)
plt.title('Solution SIR model')
plt.plot(x, s_res[0], color='blue', label="Susceptibles")
plt.plot(x, i_res[0], color='red', label="Infected")
plt.plot(x, r_res[0], color='green', label="Recovered")
plt.plot(x, d_res[0], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 2)
plt.title('Solution SIR model')
plt.plot(x, s_res[1], color='blue', label="Susceptibles")
plt.plot(x, i_res[1], color='red', label="Infected")
plt.plot(x, r_res[1], color='green', label="Recovered")
plt.plot(x, d_res[1], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

```

```

plt.subplot(2, 2, 3)
plt.title('Solution SIR model')
plt.plot(x, s_res[2], color='blue', label="Susceptibles")
plt.plot(x, i_res[2], color='red', label="Infected")
plt.plot(x, r_res[2], color='green', label="Recovered")
plt.plot(x, d_res[2], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()

plt.subplot(2, 2, 4)
plt.title('Solution SIR model')
plt.plot(x, s_res[3], color='blue', label="Susceptibles")
plt.plot(x, i_res[3], color='red', label="Infected")
plt.plot(x, r_res[3], color='green', label="Recovered")
plt.plot(x, d_res[3], color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solutions')
plt.tight_layout()
plt.show()

def first():
    s_list = [s]
    i_list = [i]
    r_list = [r]
    d_list = [d]
    for j in range(step):
        res1, res2, res3, res4 = rk4(s_list[j], i_list[j], r_list[j], d_list[j], h,
r0, c0, u0)
        s_list.append(res1)
        i_list.append(res2)
        r_list.append(res3)
        d_list.append(res4)
    s_list = np.array(s_list)
    i_list = np.array(i_list)
    r_list = np.array(r_list)
    d_list = np.array(d_list)

```

```

max_infected = max(i_list)
max_time = t0 + h*(np.where(i_list == max_infected)[0])
print('\nПік інфікованих = ', int(max_infected), 'y', int(round(max_time[0],
0)), 'день')
end_time = t0 + h*np.where(i_list < 1)[0][0]
print("Кількість suspected індивідів: ", int(s_list[-1]))
print("Кількість infected індивідів: ", int(i_list[-1]))
print("Кількість recovered індивідів: ", int(r_list[-1]))
print("Кількість dead індивідів: ", int(d_list[-1]))
print("Кількість днів, які тривала епідемія: ", int(end_time))
x = np.arange(t0, t+h, h)
plt.title('Solution SIR model')
plt.plot(x, s_list, color='blue', label="Susceptibles")
plt.plot(x, i_list, color='red', label="Infected")
plt.plot(x, r_list, color='green', label="Recovered")
plt.plot(x, d_list, color='black', label="Dead")
plt.legend()
plt.grid(c='lightgrey')
ax = plt.gca()
ax.set_axisbelow(True)
plt.xlabel('Days')
plt.ylabel('Solution')
plt.xlim(0, int(end_time)+1)
plt.xticks(np.arange(0, t, 5))
plt.tight_layout()
plt.show()

if __name__ == '__main__':
    N = 1_000
    s = 900
    i = 100
    r = 0
    d = 0
    c0 = 0.5
    r0 = 5
    u0 = 0.1
    alpha = 0.1
    beta = 0.05
    t0, t = 0, 50

    h = 0.1
    step = int((t-t0)/h)

    gamma = 5
    delta = 0.1
    lamda = 0.01

```

```
umax = 0.9
```

```
# first()
```

```
# test_n()
```

```
# test_r()
```

```
# test_c()
```

```
test_u()
```