

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»

Виконала:

студентка групи КМ-01
Резниченко Є. С.

Перевірили:

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

Зміст

Постановка задачі.....	3
Теоретичні відомості.....	4
Порядок виконання роботи.....	5
Основна частина.....	6
Дослідження параметру N	8
Дослідження параметру r	9
Дослідження параметру c	10
Дослідження параметру u	11
Висновок.....	12
Використана література	13
Додаток А – Код програми	14

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;

$p(I)dt = p(I(t)) \cdot dt$ - 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.

Промодельювати 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 відрізку 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 подальшому дослідженні в цій лабораторній роботі буде застосовуватись третя модель для імітації епідемії, яка ґрунтується на зворотному зв'язку. Параметри моделі, які піддаються регулюванню, змінюються відповідно до поточного стану системи. Для симуляції процесу епідемії використано систему диференціальних рівнянь, обчислених за допомогою методу Рунге-Кутта четвертого порядку точності.

Змодельюємо епідемію на наступних даних:

$N = 100000$	$R = 0$	$C0 = 0.6$	$\alpha = 0.1$
$S = 90000$	$D = 0$	$R0 = 16$	$\beta = 0.05$
$I = 10000$		$U0 = 0.1$	$T = [0, 80]$

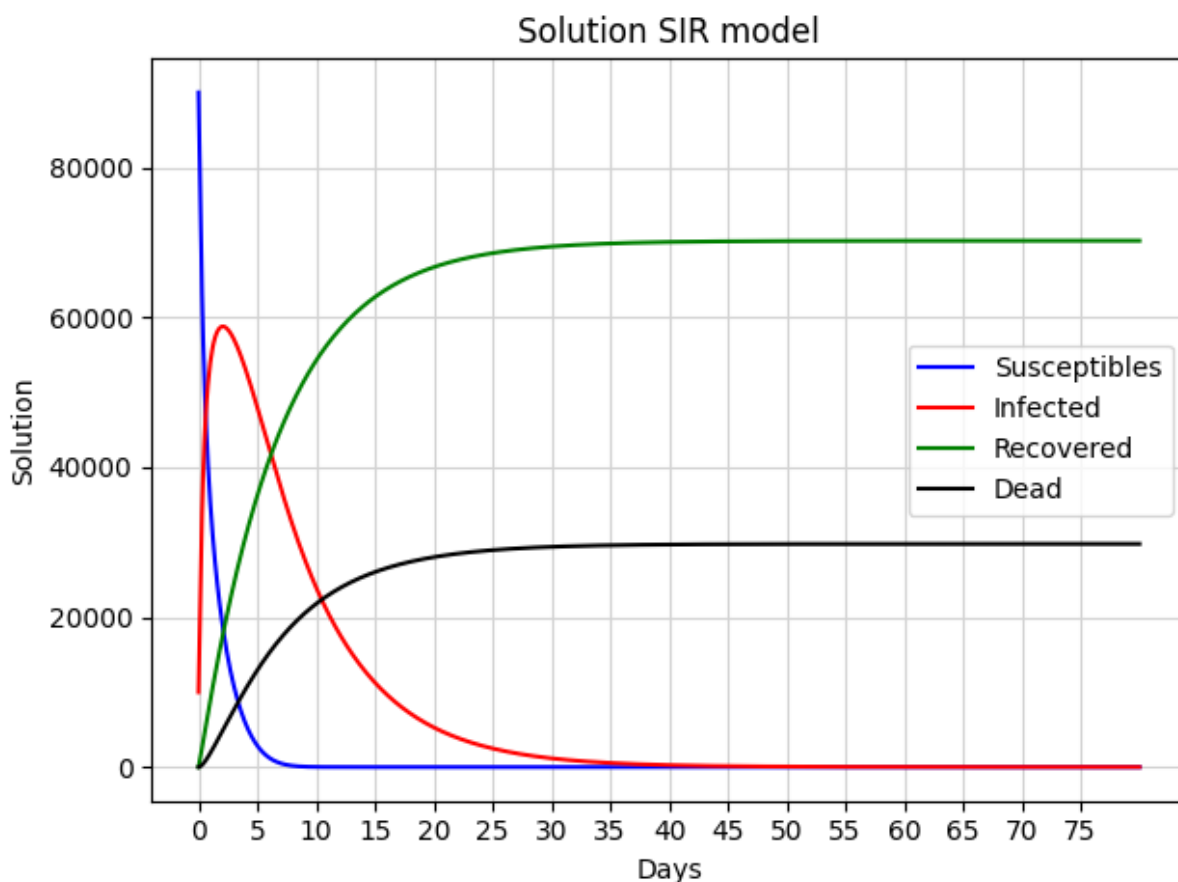


Рис. 1 – Протікання епідемії

Основні характеристики даної епідемії:

Peak of infected people: 58783 on 2 day

Suspected people: 0

Infected people: 0

Recovered people: 70223

Dead people: 29775

Days: 76

Епідемія тривала 76 дні, а пік інфікованих настав на 2 день.

Приблизно 30% населення померло та 70% отримало імунітет від вірусу.

Дослідження параметру N

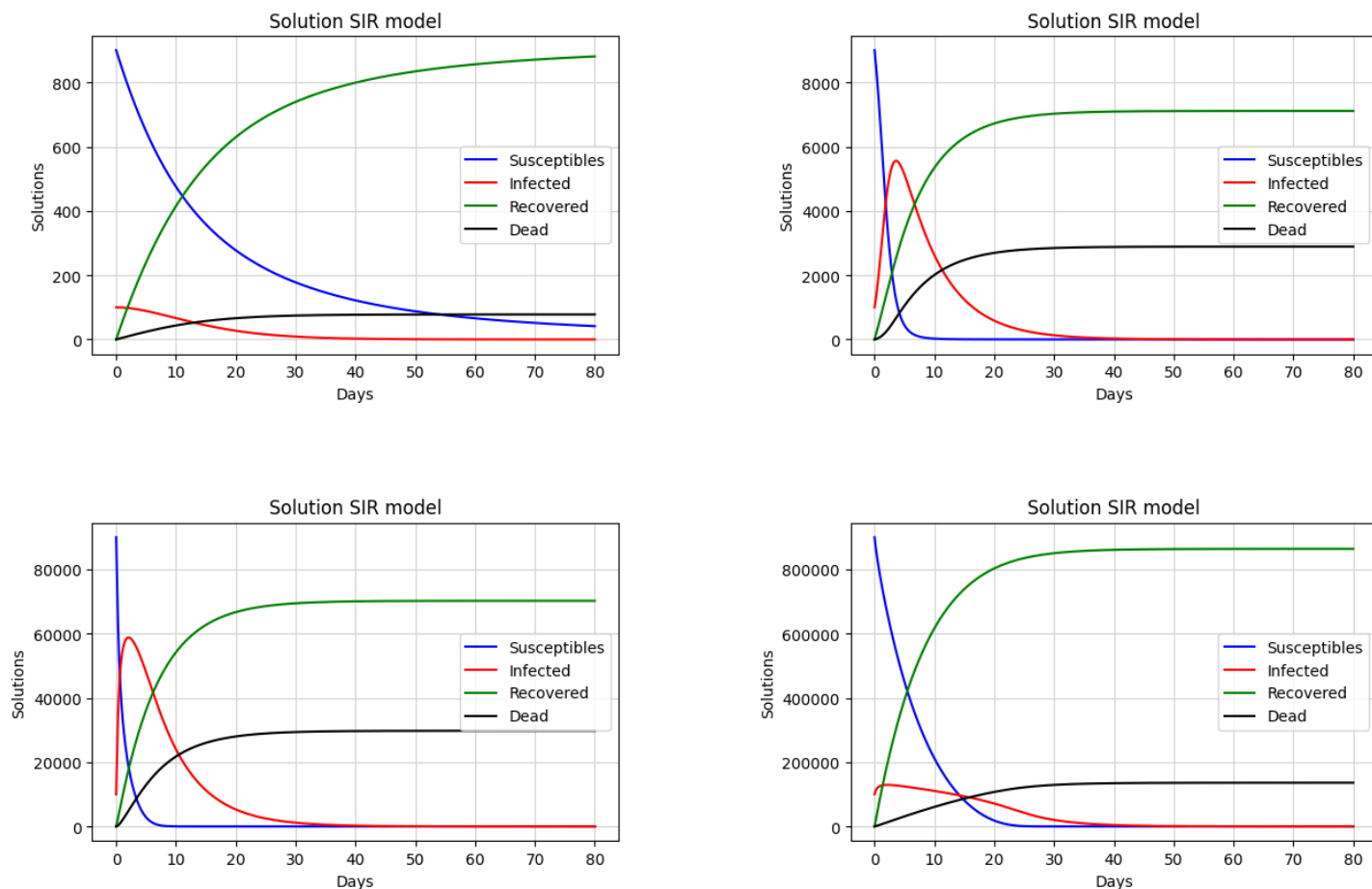


Рис. 2 – Протікання епідемії при $N = [1000, 10000, 100000, 1000000]$

Параметр N відповідає за загальне число індивідів.

Для цього дослідження було використано такі параметри:

$R = 0$	$C0 = 0.6$	$\alpha = 0.1$
$D = 0$	$R0 = 16$	$\beta = 0.05$
	$U0 = 0.1$	$T = [0, 80]$

При значеннях $N = 10000$ та $N = 100000$ ми спостерігаємо значну кількість інфікованих та померлих. Однак при $N = 1000$ та $N = 1000000$ ситуація інша. Це може бути пов'язано з коефіцієнтами інтенсивності контакту індивідів та ймовірністю зараження при контакті.

Дослідження параметру r

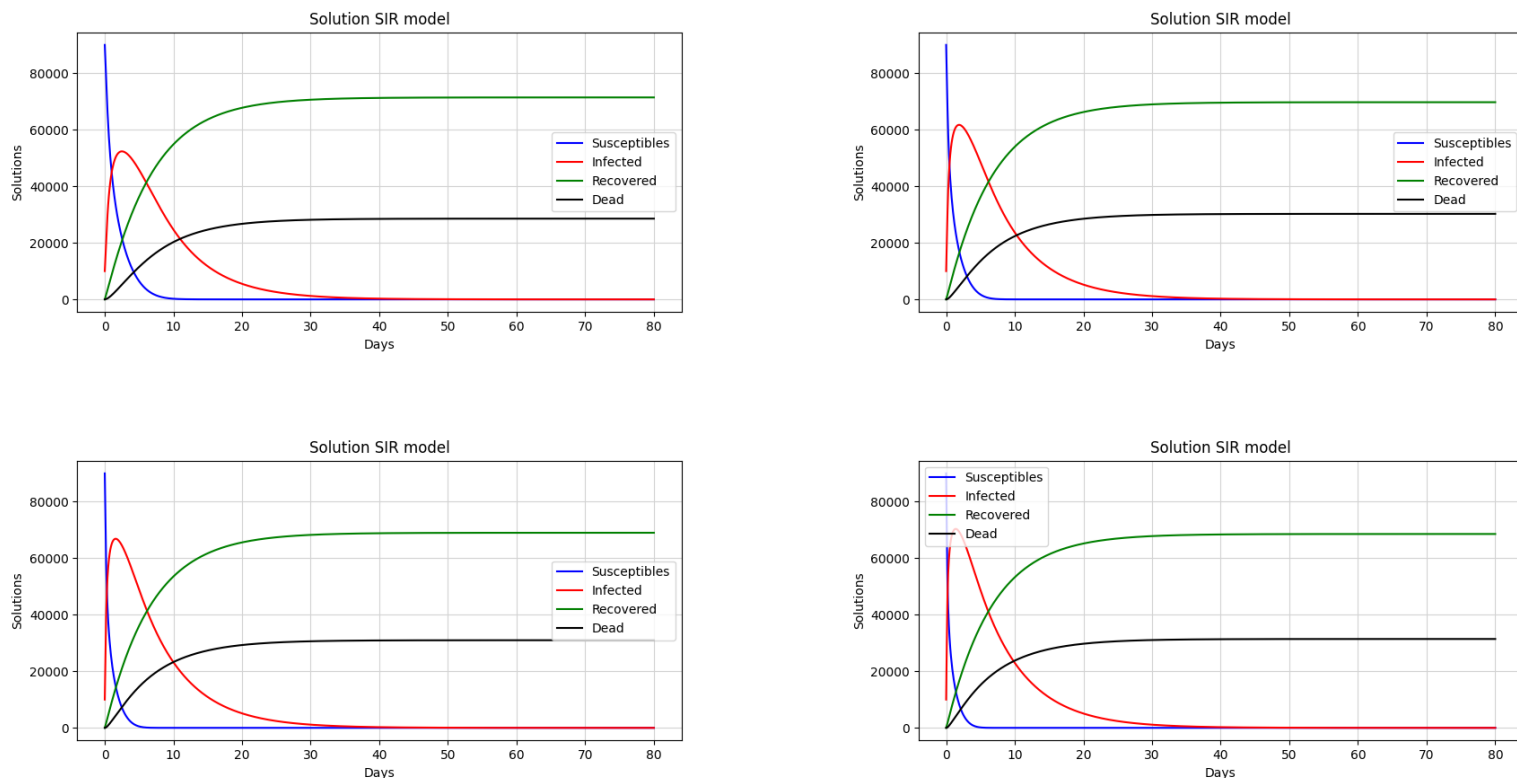


Рис. 3 – Протікання епідемії при $r = [10, 20, 30, 40]$

Параметр r відповідає за інтенсивність контактів індивіда

Для цього дослідження було використано такі параметри:

$N = 100000$	$R = 0$	$C0 = 0.6$	$\alpha = 0.1$
$S = 90000$	$D = 0$	$U0 = 0.1$	$\beta = 0.05$
$I = 10000$			$T = [0, 80]$

Можна припустити, що кількість контактів людини має вплив на швидкість поширення інфекції серед населення та тривалість епідемії: чим більше контактів, тим швидше розповсюджується інфекція.

Дослідження параметру c

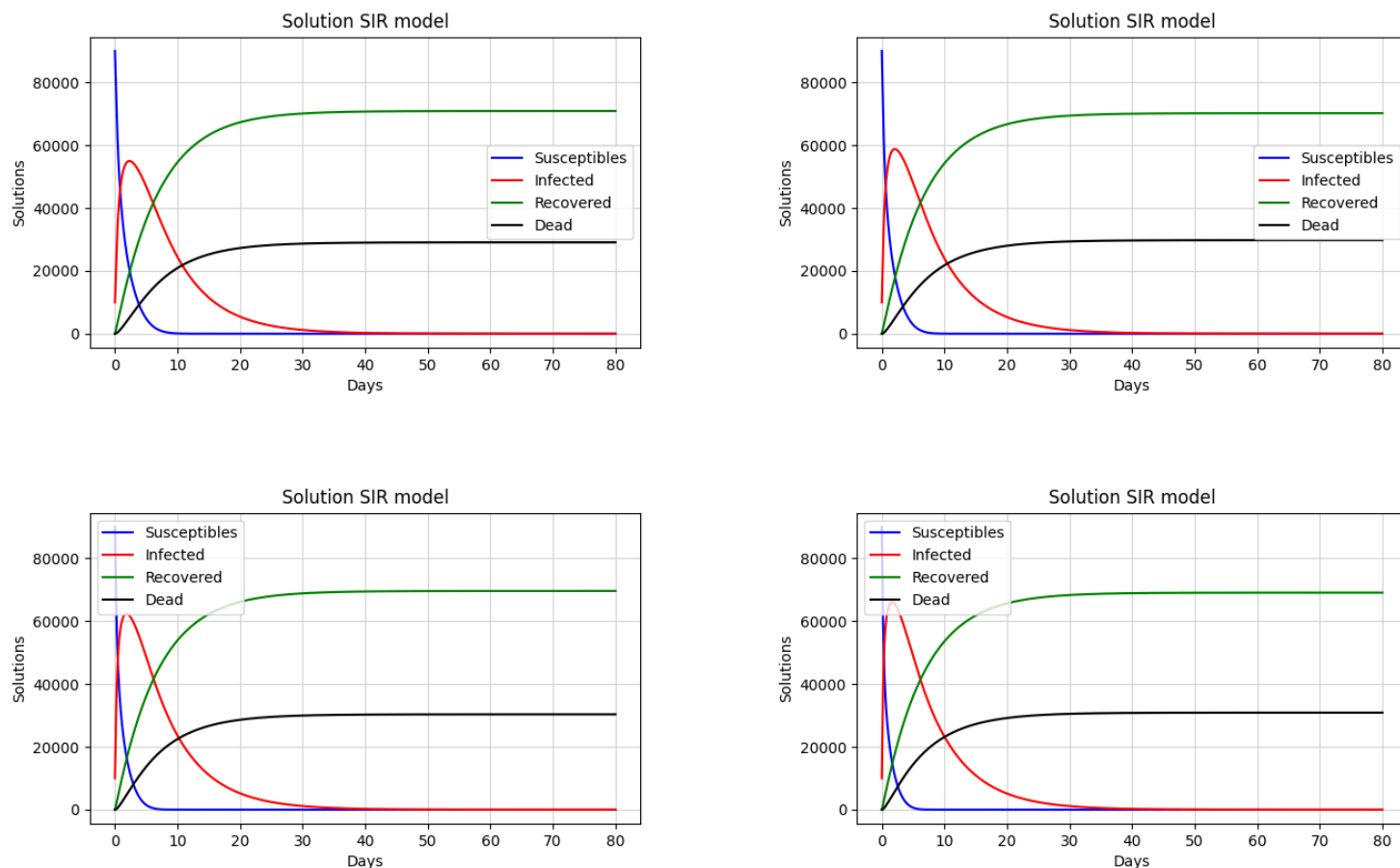


Рис. 4 – Протікання епідемії при $c = [0.5, 0.6, 0.7, 0.8]$

Параметр c відповідає за ймовірність зараження при контакті з хворим

Для цього дослідження було використано такі параметри:

$N = 100000$	$R = 0$	$R_0 = 16$	$\alpha = 0.1$
$S = 90000$	$D = 0$	$U_0 = 0.1$	$\beta = 0.05$
$I = 10000$			$T = [0, 80]$

Варто вважати, що ймовірність передачі захворювання під час контакту з інфікованим індивідом впливає на темп розповсюдження інфекції серед населення та тривалість епідемії. Зі зростанням ймовірності зараження під час контакту з інфікованою особою зростає швидкість поширення інфекції.

Дослідження параметру u

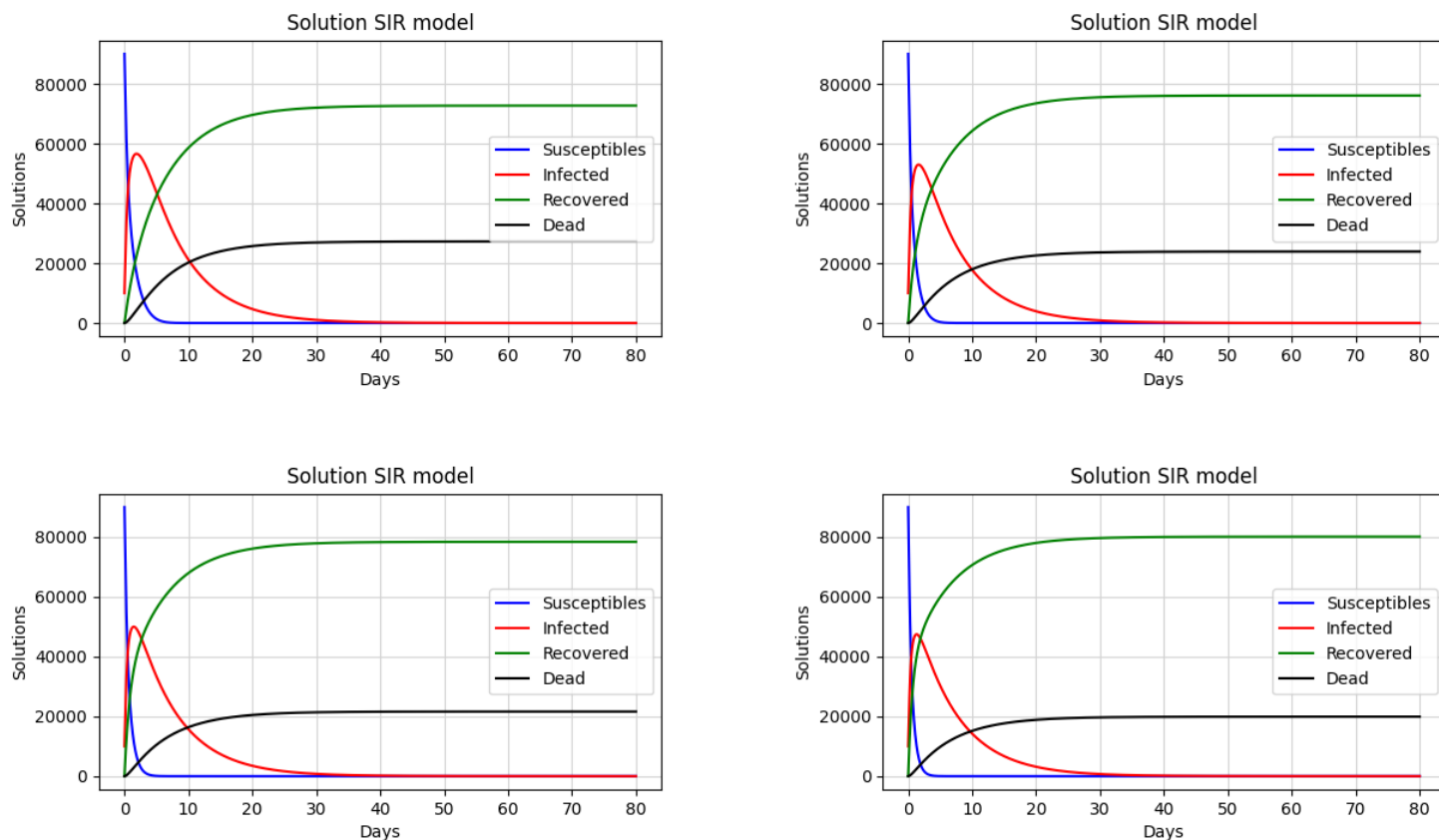


Рис. 5 – Протікання епідемії при $u = [0.2, 0.4, 0.6, 0.8]$

Параметр u відповідає за відсоток вакцинованих за час

Для цього дослідження було використано такі параметри:

$N = 100000$	$R = 0$	$C0 = 0.6$	$\alpha = 0.1$
$S = 90000$	$D = 0$	$R0 = 16$	$\beta = 0.05$
$I = 10000$			$T = [0, 80]$

Отже, можна зробити наступний висновок – збільшення цього параметра призводить до зміцнення імунітету у населення та прискорює завершення епідемії, оскільки більше людей отримує вакцинацію.

Висновок

Під час цієї лабораторної роботи було вивчено ключові принципи створення математичних моделей поширення епідемій, і вивчено вплив різних параметрів на тривалість та інтенсивність епідемії. Крім того, було розроблено програмне забезпечення для математичного моделювання епідемій.

Основні висновки стосовно досліджених параметрів такі:

1. Не можна впевнено сказати, що кількість населення грає ключову роль. Із дослідження слідує, що при малій та великій кількості населення смертність була низькою, а при середній кількості населення смертність значно виросла.
2. Чим більше контактів має людина, тим швидше поширюється інфекція і довше триває епідемія, оскільки інфекція поширюється активніше при збільшенні контактів.
3. Підвищення ймовірності зараження при контакті з хворим призводить до прискореного розповсюдження інфекції.
4. Чим більше людей вакцинуються, тим більше осіб отримує імунітет, що призводить до швидшого припинення епідемії.

Використана література

1. Методичні вказівки до лабораторної роботи.
2. Яушева О.А. 2016 - Математическая модель эпидемии лихорадки Эбола (SIR модель). Санкт-Петербургский государственный университет, Кафедра диагностики функциональных систем. Санкт-Петербург, 2016.
3. 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 = [900, 9_000, 90_000, 900_000]
    i = [100, 1_000, 10_000, 100_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 = [10, 20, 30, 40]
    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 = [0.5, 0.6, 0.7, 0.8]
    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 = [0.2, 0.4, 0.6, 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('\nPeak of infected people: ', int(max_infected), 'on',
int(round(max_time[0], 0)), 'day')
    end_time = t0 + h*np.where(i_list < 1)[0][0]
    print("Suspected people: ", int(s_list[-1]))
    print("Infected people: ", int(i_list[-1]))
    print("Recovered people: ", int(r_list[-1]))
    print("Dead people: ", int(d_list[-1]))
    print("Days: ", 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 = 100_000
    s = 90_000
    i = 10_000
    r = 0
    d = 0
    c0 = 0.6
    r0 = 16
    u0 = 0.1
    alpha = 0.1
    beta = 0.05
    t0, t = 0, 80

```

```
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()
```