

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«Моделювання систем»
для студентів спеціальності 123 Комп'ютерна інженерія
усіх форм навчання.
Моделювання систем (мовою Python)

Методичні вказівки до виконання лабораторних робіт з дисципліни «Моделювання систем» для студентів спеціальності 123 Комп'ютерна інженерія усіх форм навчання. Моделювання систем (мовою Python) / Укл.: Р. К. Кудерметов, О. В. Польська — Запоріжжя: НУ «Запорізька політехніка», 2025 — 64 с.

Укладачі: Р. К. Кудерметов, доцент, к.т.н.
О. В. Польська, ст. викладач

Рецензент: М. Ю. Тягунова, доцент, к.т.н.

Відповідальний
за випуск: О. В. Польська, ст. викладач

Затверджено:
на засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 6 від 29 січня 2025 р.

Рекомендовано до видання:
НМК факультету КНТ
Протокол № 6 від 12 лютого 2025 р.

Зміст

1	Моделювання динамічних систем	5
1.1	Теоретичні відомості	5
1.2	Дослідження екологічної системи	6
1.3	Практичне завдання (перша частина)	7
1.3.1	Моделювання екологічної системи	7
1.3.2	Стаціонарний стан екологічної системи	10
1.3.3	Екологічна система, в якій відсутні зустрічі між хижаками та жертвами	10
1.4	Дослідження процесу розповсюдження епідемії	10
1.5	Практичне завдання (друга частина)	11
1.5.1	Моделювання процесу розповсюдження епідемії	11
1.5.2	Моделювання процесу розповсюдження епідемії при своєчасній ізоляції хворих	13
1.6	Контрольні питання	14
1.7	Зміст звіту з лабораторної роботи	14
2	Моделювання лінійних безперервних систем	15
2.1	Теоретичні відомості	15
2.2	Практичне завдання	17
2.2.1	Моделювання коливальної системи	17
2.2.2	Моделювання системи, що описується диференціальним рівнянням четвертого порядку	20
2.2.3	Перевірка властивості адитивності системи	23
2.2.4	Перевірка властивості однорідності системи	25
2.3	Контрольні питання	28
2.4	Зміст звіту з лабораторної роботи	29
3	Моделювання просторово-розподілених процесів	30
3.1	Теоретичні відомості	30
3.2	Практичне завдання	33
3.2.1	Розв'язання контрольної задачі	33
3.2.2	Індивідуальне завдання	37

3.3	Контрольні питання	37
3.4	Зміст звіту з лабораторної роботи	38
4	Моделювання систем, що описуються рівняннями теплопровідності	39
4.1	Теоретичні відомості	39
4.2	Дослідження процесів зміни температури в стіні . . .	42
4.2.1	Розв'язання контрольної задачі (стіна із заліза)	42
4.2.2	Індивідуальне завдання	44
4.3	Контрольні питання	47
4.4	Зміст звіту з лабораторної роботи	48
5	Моделювання систем масового обслуговування	49
5.1	Теоретичні відомості	49
5.1.1	Поняття потоків у СМО	49
5.1.2	Короткі відомості про основні види СМО . . .	50
5.1.3	Алгоритм роботи одноканальної СМО з обмеженим часом очікування	54
5.2	Практичні завдання	57
5.2.1	Дослідження одноканальної СМО	57
5.2.2	Дослідження багатоканальної СМО	60
5.3	Контрольні питання	63
5.4	Зміст звіту з лабораторної роботи	63
	Рекомендована література	64

ЛАБОРАТОРНА РОБОТА 1

Моделювання динамічних систем

Мета роботи — опанувати методи комп'ютерного моделювання динамічних систем, що описуються системами звичайних диференціальних рівнянь.

1.1 Теоретичні відомості

За допомогою моделювання можна відтворити поведінку будь-якої фізичної або технічної системи. Це дозволяє отримати знання про характеристики об'єкта-оригіналу, який моделюється. Одним з різновидів моделей є математична модель – математичне представлення реальності для вилучення необхідної інформації щодо системи [1–3]. При цьому дослідження базується на використанні диференціальних рівнянь, які допомагають описати стан системи на кожному етапі вивчення її поведінки в залежності від часу.

Звичайне диференціальне рівняння (ЗДР) першого порядку має вигляд (1.1):

$$\dot{y}(x) = f(x, y), \quad (1.1)$$

де $\dot{y}(x)$ – перша похідна змінної $y(x)$, x – незалежна змінна.

Розв'язати диференціальне рівняння – означає знайти множину функцій, які задовольняють даному рівнянню.

Найбільш поширеним підходом до розв'язання ЗДР є їх чисельне інтегрування. Одним із найпростіших методів чисельного інтегрування є метод прямокутників (Ейлера), що реалізується за допомогою рекурентного співвідношення:

$$y_{n+1} = y_n + hf(x_n, y_n), \quad (1.2)$$

де y_n – значення змінної y на n -му кроці, x_n – значення змінної x на n -му кроці, h – крок інтегрування, отриманий шляхом поділу часової осі на рівні інтервали.

Для розв'язання ЗДР на практиці часто використовують метод чисельного інтегрування Рунге-Кутта четвертого порядку $O(h^4)$.

Порядок вказує на точність апроксимації розв'язку. Оскільки розв'язок рівняння – сукупність точок, що графічно формують криву, яка відповідає поведінці системи, то для кожного значення x необхідно обчислити y .

Для обчислення y_{n+1} методом Рунге-Кутта використовується формула:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (1.3)$$

де значення k_1, k_2, k_3, k_4 обчислюються за допомогою формул:

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\ k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\ k_4 &= f(x_n + h, y_n + hk_3). \end{aligned}$$

1.2 Дослідження екологічної системи

Для деяких екологічних систем характерним є наявність хижаків та жертв. Наприклад, досліджується компонент біосфери – ліс. Він є системою, оскільки містить багато складових: рослини різних ярусів, тварини. Розглядається закрите середовище, де існують тільки два види тварин: вовки та зайці.

Тварини і рослини взаємодіють між собою. Рослини (редуценти) – їжа для рослиноїдних тварин (консументів першого порядку, наприклад, зайців). Рослиноїдні тварини – їжа для хижаків (консументів другого порядку, наприклад, вовків).

За умови відсутності вовків приріст чисельності зайців описується рівнянням (1.4), де a_{11} – коефіцієнт приросту зайців:

$$\dot{x} = a_{11}x. \quad (1.4)$$

У випадку, якщо зайців немає, вовки вимирають і процес зменшення чисельності вовків відповідає рівнянню (1.5), де a_{22} – коефіцієнт зменшення вовків за відсутності зустрічей із зайцями:

$$\dot{y} = -a_{22}y. \quad (1.5)$$

Якщо припустити, що в початковий момент спостереження за системою $t = 0$ число зайців дорівнює $x(0) = x_0$, а число вовків дорівнює $y(0) = y_0$, то аналітичні розв'язки рівнянь (1.4) і (1.5) матимуть вигляд:

$$x = x_0 e^{a_{11}t}, \quad y = y_0 e^{-a_{22}t}.$$

Це означає, що число зайців зростає за експоненціальним законом, а число вовків збуває, також по експоненті.

Змоделюємо систему, в якій територіальні ресурси розподілені між зайцями та вовками. При зустрічі з хижаками кількість зайців буде зменшуватися, а кількість вовків – навпаки, збільшуватися. Цей процес описує система рівнянь Лотки-Вольтерри:

$$\begin{cases} \dot{x} = a_{11}x - a_{12}xy, \\ \dot{y} = -a_{22}y + a_{21}xy. \end{cases} \quad (1.6)$$

Перший член кожного рівняння системи (1.6) відповідає стану моделі за рівняннями (1.4) та (1.5). Елемент $a_{12}xy$ системи (1.6) описує процес зменшення чисельності зайців при зустрічі з вовками, а елемент $a_{21}xy$ – збільшення чисельності вовків, якщо зайців для цього достатньо. Таким чином, можна сказати, що a_{12} і a_{21} – коефіцієнти, що характеризують частоту зустрічей жертв з хижаками і, очевидно, що вони рівні.

1.3 Практичне завдання (перша частина)

1.3.1 Моделювання екологічної системи

Виконайте моделювання екологічної системи. Приклад програми чисельного моделювання системи, що описується рівняннями Лотки-Вольтерри, наведено у лістингу 1.1. В результаті виконання програми буде отримано графіки (див. рис. 1.1).

На правому графіку (залежності хижаків від жертв), зображеному на рис. 1.1, вісь y – популяція хижаків, вісь x – популяція жертв. Спочатку кількість зайців та вовків задана співвідношенням 500:300 ($x : y$). Вовки та зайці починають розмножуватися,

змінюючи це співвідношення. Але, коли вовків стає дуже багато, чисельність зайців не встигає відновлюватися. Як наслідок, у вовків не вистачає їжі, і вони гинуть. Тривалість спостереження (t_1) 150 днів.

Лістинг 1.1 Моделювання екологічної системи (файл **lab1-1.py**)

```

1  import numpy as np
2  from scipy.integrate import odeint
3  import matplotlib.pyplot as plt
4
5  def dvdt(v, t, c):
6      x1, x2 = v
7      alpha, beta, gamma, delta = c
8      dx1dt = alpha * x1 - beta * x1 * x2
9      dx2dt = delta * x1 * x2 - gamma * x2
10     return [dx1dt, dx2dt]
11
12 a11, a12, a22, a21 = 0.06, 0.0006, 0.24, 0.0006
13 coeff = (a11, a12, a22, a21)
14 t0, t1, h = 0, 150, 0.1
15 n = int((t1-t0) / h)
16 x0, y0 = 500, 300
17 v0 = [x0, y0]
18 t = np.linspace(t0, t1, n)
19 solution = odeint(dvdt, v0, t, args=(coeff, ))
20 x, y = solution.T
21
22 fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
23 ax1.plot(t, x, label='Жертви')
24 ax1.plot(t, y, label='Хижак')
25 ax1.set_xlabel('Час')
26 ax1.set_ylabel('Кількість')
27 ax1.legend(loc='upper right')
28 ax1.grid(True)
29
30 ax2.plot(x, y, color='black')
31 ax2.set_xlabel('Жертви')
32 ax2.set_ylabel('Хижак')
33 ax2.grid(True)
34 plt.show()
```

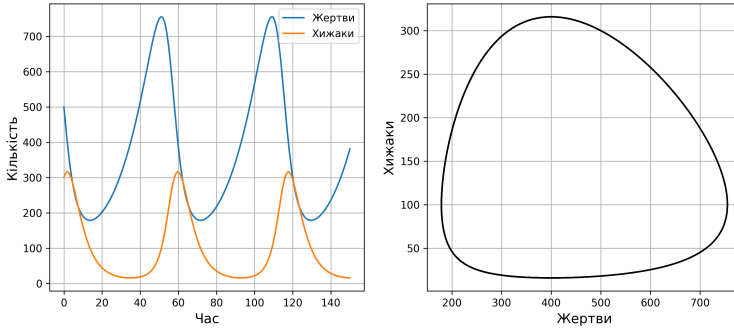


Рисунок 1.1 – Результат виконання програми з ліст. 1.1

Пояснення до реалізації програми (лістинг 1.1)

У реалізації програми мовою Python [4–6] для інтегрування ЗДР використовується метод `odeint(func, y0, t, args=(), ...)` (рядок 19), який реалізований у бібліотеці `scipy.integrate` [7].

Найважливішою особливістю цього метода є те, що він «адаптивний»: він може автоматично визначити яку саме схему інтегрування використовувати (вибір між методом Адамса-Мултона високого порядку, або іншим неявним методом, відомим як «формула зворотного диференціювання»).

Першим аргументом функції `odeint()` є функція, яка обчислює вектор похідних змінних від часу (праві частини системи (1.6)); другим аргументом має бути вектор початкових умов; третім аргументом має бути послідовність точок, у яких необхідно обчислити вектор невідомих змінних.

Зверніть увагу, що аргумент `args=()` – це додаткові параметри, необхідні для обчислення функції похідних. Цей параметр є типом даних мови Python, який називається «кортеж». Кортеж – це структура даних, яка не змінюється у процесі роботи програми. Запис `(a, )` означає, що кортеж складається з одного елемента `a`.

Метод `linspace(start, stop, num=, ...)` (рядок 18) з бібліотеки `numpy` повертає послідовність `num` рівномірно розподілених

значень (точок) в інтервалі `[start, stop]`.

В рядку 20 літера `T` у виразі `solution.T` означає транспонування вектора `solution`.

Починаючи з рядка 22 наведено код для побудови графіків результатів інтегрування.

1.3.2 Стаціонарний стан екологічної системи

Знайдіть стаціонарний стан системи, тобто таке співвідношення зайців та вовків, за яким чисельність зайців встигає відновлюватися, не дозволяючи вовкам гинути від нестачі їжі.

Для цього необхідно знайти розв'язок системи рівнянь (1.6) за умови $\dot{x} = 0$, $\dot{y} = 0$. Виконаємо заміну цих значень в (1.6) та отримаємо систему рівнянь у вигляді:

$$\begin{cases} a_{11}x - a_{12}xy = 0, \\ -a_{22}y + a_{21}xy = 0. \end{cases} \quad (1.7)$$

Виконайте необхідні перетворення системи рівнянь (1.7) та обчисліть координати стаціонарної точки (x, y) .

На основі програми з лістингу 1.1 створіть у новому файлі програму (файл **lab1-2.py**), яка моделює стаціонарний стан екологічної системи. Побудуйте графіки залежності $x(t)$, $y(t)$ та $y(x)$.

1.3.3 Екологічна система, в якій відсутні зустрічі між хижаками та жертвами

Виконайте моделювання системи, в якій відсутні зустрічі між хижаками та жертвами (файл **lab1-3.py**). Для цього складіть систему рівнянь з (1.4) та (1.5). Задайте тривалість спостереження ($t1$) 20 днів. Побудуйте графіки залежностей $x(t)$ і $y(t)$.

1.4 Дослідження процесу розповсюдження епідемії

Нехай є віддалений населений пункт, у якому знаходиться H людей, та почалася епідемія. Позначимо через x – кількість здоро-

вих людей, через y – кількість хворих людей, а через z – кількість людей, які вже переохворіли і в них виробився імунітет.

У момент виявлення епідемії $y(0)$ людей були хворими; $x(0)$ людей були здоровими, але потенційно схильними до захворювання. При цьому кількість переохворілих людей: $z(0) = H - x(0) - y(0)$.

Відповідно до гіпотези захворювання передається «контактно», тобто при зустрічі. Тоді швидкість інфікування здорових людей x пропорційна частоті їх зустрічей із хворими y . Характер захворювання й умови життя в населеному пункті такі, що кожний хворий щодня в середньому передає інфекцію β здоровим людям із кількості H . Захворіла людина через γ днів одужує, набуваючи імунітет.

Тоді можна записати диференціальні рівняння розглянутої системи у вигляді:

$$\begin{cases} \dot{x} = -\beta/H \cdot xy, \\ \dot{y} = \beta/H \cdot xy - 1/\gamma \cdot y, \\ \dot{z} = 1/\gamma \cdot y. \end{cases} \quad (1.8)$$

1.5 Практичне завдання (друга частина)

1.5.1 Моделювання процесу розповсюдження епідемії

Для моделювання розглянутої системи (1.8) зробіть наступні позначення: $x_1 = x$, $x_2 = y$, $x_3 = z$. В результаті отримаєте систему рівнянь:

$$\begin{cases} \dot{x}_1 = -\beta/H \cdot x_1x_2, \\ \dot{x}_2 = \beta/H \cdot x_1x_2 - 1/\gamma \cdot x_2, \\ \dot{x}_3 = 1/\gamma \cdot x_2. \end{cases} \quad (1.9)$$

Створіть файл **lab1-4.py** з кодом лістингу 1.2 з наступними даними:

- кількість людей в населеному пункті **H=1000**;
- інтенсивність розповсюдження епідемії така, що за один день одна захворіла людина інфікує одну здорову людину **beta=1**;
- кількість днів, необхідних на одужання **gamma=10**;

– вектор початкових умов $v_0=[x_0,y_0,z_0]$, елементи якого:
 $x_0=900$ – кількість здорових людей, $y_0=90$ – кількість хворих людей,
 $z_0=H-x_0-y_0$ – кількість людей, які одужали;
 – тривалість спостереження $t_1=40$ днів.

Результат роботи програми наведено на рис. 1.2.

Лістинг 1.2 Моделювання розповсюдження епідемії (**lab1-4.py**)

```

1  import numpy as np
2  from scipy.integrate import odeint
3  import matplotlib.pyplot as plt
4
5  def dvdt(v, t, p, b, g):
6      x1, x2, x3 = v
7      dx1dt = -b/p * x1 * x2
8      dx2dt = b/p * x1 * x2 - 1/g * x2
9      dx3dt = 1/g * x2
10     return [dx1dt, dx2dt, dx3dt]
11
12  beta, gamma = 1, 10
13  H, x0, y0 = 1000, 900, 90
14  z0 = H - x0 - y0
15  v0 = [x0, y0, z0]
16  t0, t1 = 0, 40
17  t = np.linspace(t0, t1, 100)
18
19  solution = odeint(dvdt, v0, t, args=(H, beta, gamma))
20  x, y, z = solution.T
21
22  plt.figure(figsize=(8, 5))
23  plt.plot(t, x, label='Сприйнятливі')
24  plt.plot(t, y, label='Інфіковані')
25  plt.plot(t, z, label='Одужали')
26  plt.xlabel('Час (дні)')
27  plt.ylabel('Населення')
28  plt.title('Процес розповсюдження епідемії')
29  plt.xlim(t0, t1)
30  plt.ylim(0, H)
31  plt.legend()
32  plt.grid(True)
33  plt.show()

```

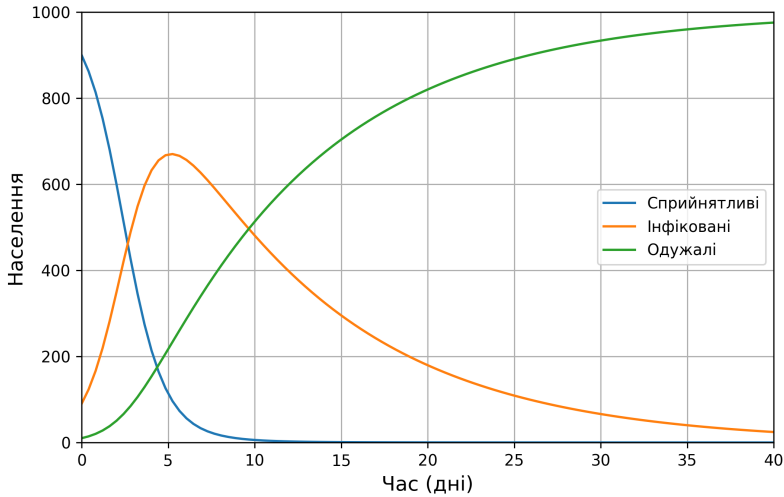


Рисунок 1.2 – Графік залежності кількості людей від часу

1.5.2 Моделювання процесу розповсюдження епідемії при своєчасній ізоляції хворих

Модифікуємо систему поширення епідемії таким чином, щоб у певний момент часу τ хворих людей ізолювали від здорової частини населення. В результаті буде зупинено процес поширення епідемії. Таку поведінку системи можна змоделювати шляхом обнулення у певний момент часу коефіцієнта інтенсивності поширення епідемії β . Один із способів змоделювати таку поведінку – ввести в систему рівнянь функцію Хевісайду $\Phi(\theta)$ (1.10). Це функція дійсної змінної, значення якої дорівнює нулю (0) для від’ємних і одиниці (1) для додатних значень аргументу.

$$\begin{cases} \dot{x} = -\beta/H \cdot xy\Phi(\tau - t), \\ \dot{y} = \beta/H \cdot xy\Phi(\tau - t) - 1/\gamma \cdot y, \\ \dot{z} = 1/\gamma \cdot y. \end{cases} \quad (1.10)$$

Для моделювання цієї системи в новому файлі (**lab1-5.py**) додайте визначення функції Хевісайду `heaviside(t, tau)` та реалізуйте розв’язання системи рівнянь (1.10). Це можна зробити наступним чином:

```
def heaviside(t, tau):
    if(t < tau):
        return 1
    else:
        return 0
```

Змодельуйте поведінку системи за кількома значеннями часу ізоляції, наприклад, три, п’ять і вісім днів.

1.6 Контрольні питання

1. Що таке моделювання? Його призначення?
2. Що таке модель? Наведіть деякі типи моделей.
3. Чи є модель системою?
4. Дайте визначення поняттю система та її чотирьом ознакам: ціле і частина, наявність зв’язків, організація та інтегральність.
5. Поясніть, у чому полягає стратифікований підхід до вивчення (дослідження, пізнання) систем.
6. Які завдання є предметом розв’язання загальної теорії систем і дисципліни «Системотехніка»?
7. Які припущення в моделях екологічної системи і системи розповсюдження епідемії дозволили представити їх як замкнуті (закриті, автономні) системи?
8. Поясніть поняття гомоморфізму та ізоморфізму моделей.

1.7 Зміст звіту з лабораторної роботи

1. Титульний лист, оформлений відповідно до вимог.
2. Мета роботи.
3. Лістинги програм до п. 1.3 і п. 1.5.
4. Результати виконання програм у вигляді графіків.
5. Відповіді на контрольні питання.
6. Висновки за результатами досліджень.

ЛАБОРАТОРНА РОБОТА 2

Моделювання лінійних безперервних систем

Мета роботи — опанувати методи моделювання лінійних систем, що описуються звичайними диференціальними рівняннями n -го порядку, дослідити основні властивості лінійних систем.

2.1 Теоретичні відомості

Для розв'язання звичайного диференціального рівняння (ЗДР) n -го порядку чисельними методами, його необхідно звести до системи, що складається з n диференціальних рівнянь першого порядку. Методику такого зведення розглянемо на прикладі ЗДР другого порядку:

$$\ddot{y}(t) + a_1\dot{y}(t) + a_2y(t) = 0 \quad (2.1)$$

з початковими умовами $\dot{y}(0) = \dot{y}_0$, $y(0) = y_0$.

Розв'яжемо рівняння (2.1) щодо старшої похідної:

$$\ddot{y}(t) = -a_1\dot{y}(t) - a_2y(t). \quad (2.2)$$

Для цього рівняння введемо дві нові змінні x_1 і x_2 таким чином:

$$x_1 = y(t), \quad x_2 = \dot{y}(t), \quad (2.3)$$

тоді очевидно, що $\dot{x}_1 = x_2$ і $\dot{x}_2 = \ddot{y}(t)$. Зробивши заміну нових змінних (2.3) у рівнянні (2.2), одержимо систему з двох ЗДР першого порядку:

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = -a_1x_2 - a_2x_1 \end{cases} \quad (2.4)$$

з початковими умовами $\dot{x}_1(0) = y(0)$, $\dot{x}_2(0) = \dot{y}(0)$.

Зверніть увагу, що у правих частинах рівнянь (2.4) відсутні похідні змінних x_1 і x_2 .

Для дослідження багатьох технічних і природних систем їх можна представити шляхом *лінеаризації* і допущень як лінійні системи. При цьому рівняння мають лінійний вигляд, тобто в них

відсутні нелінійні функції від змінних, а методи дослідження простіші [3].

Будь-яку лінійну динамічну систему можна представити у вигляді системи лінійних рівнянь:

$$\begin{cases} \dot{z}(t) = a(t)x(t) + b(t)z(t), \\ y(t) = c(t)x(t) + d(t)z(t), \end{cases} \quad (2.5)$$

де $z(t)$ – вектор стану системи; x – вектор зовнішніх сил, що діють на систему (входів); y – вектор виходів системи; $a(t)$, $b(t)$, $c(t)$, $d(t)$ – матриці коефіцієнтів.

Перше рівняння в (2.5) називається рівнянням стану системи та описує зміни стану системи у процесі її функціонування (або “часу життя”). Друге рівняння в системі (2.5) описує процес вимірювання, так як поведінку системи ми зазвичай оцінюємо за допомогою приладів і одиниць вимірювання (наприклад, електричний струм, який створює система). Якщо змінні $z(t)$, $y(t)$ і $x(t)$ є скалярами, то $a(t)$, $b(t)$, $c(t)$ і $d(t)$ також скаляри.

Якщо елементи a , b , c і d не залежать від часу, то система (2.5) описує *лінійну стаціонарну систему*. У даній роботі ми моделюватимемо саме такі системи. Також у цій роботі ми не будемо використовувати рівняння вимірювань, тобто ми будемо оцінювати поведінку системи безпосередньо за вектором стану системи.

Лінійні системи мають властивості *однорідності* й *адитивності*. Опишемо коротко ці властивості.

Для простоти наступних формул розглянемо лише перше рівняння системи (2.5) та запишемо його у вигляді $\dot{z}(t) = f(t, z, x)$. Нехай на інтервалі часу функціонування системи задані два фрагменти вхідних впливів x'_t і x''_t , а в момент часу початку інтервалу t_0 – два різні стани z'_0 і z''_0 . Нехай для фрагментів вхідних впливів та станів системи визначено операції складання та множення на постійне число k такі, що $x_t = x'_t + x''_t$ і kx_t , а також $z_0 = z'_0 + z''_0$ і kz_0 . Тоді однорідність означає справедливність наступного співвідношення для функції $f(t, z, x)$:

$$f(t, kz_0, kx_t) = kf(t, z_0, x_t), \quad (2.6)$$

а адитивність – співвідношення

$$f(t, z_0, x_t) = f(t, z'_0, x'_t) + f(t, z''_0, x''_t). \quad (2.7)$$

Якщо друге рівняння системи (2.5) представити як $y(t) = g(t, z, x)$, то аналогічні співвідношення однорідності та адитивності можна скласти для функції $g(t, z, x)$.

Властивості однорідності та адитивності лінійної системи суттєво спрощують її дослідження. Наприклад, якщо відома поведінка (стан) системи з вхідним впливом $x(t)$, то згідно з (2.6) можна визначити поведінку системи з вхідним впливом $kx(t)$. Або, якщо відома поведінка системи при початкових станах z'_0 і z''_0 , то можна визначити поведінку системи при сумі (або різниці) цих початкових станів $z'_0 + z''_0$ відповідно до (2.7). У цьому ви переконаєтесь, виконавши практичне завдання.

2.2 Практичне завдання

2.2.1 Моделювання коливальної системи

Нагадаємо, що найпростіша коливальна система (наприклад маятник або пружина) описується ЗДР

$$\ddot{y} + 2\xi\dot{y} + \omega_0^2 y = 0, \quad (2.8)$$

де ω_0 – власна частота системи (або незгасна кутова частота); ξ – коефіцієнт згасання. Аргумент t не вказано для спрощення зображення формули.

Як можна помітити, рівняння (2.8) є лінійним і аналогічним рівнянню (2.1) при $a_1 = 2\xi$ і $a_2 = \omega_0^2$. Тому будемо використовувати (2.1) для програмної реалізації моделі.

Виберемо значення коефіцієнтів $a_1 = 0.5$ і $a_2 = 39.4784176$. Такий вибір означає, що лінійна частота коливальності системи приблизно дорівнюватиме 1 Гц (одне коливання за секунду), так як лінійна частота пов'язана з круговою частотою формулою $\omega = 2\pi f$, тоді

$$a_2 = \omega_0^2 = 1 \cdot 4\pi^2 = 39.4784176,$$

а коефіцієнт згасання $\xi = a_1/2 = 0.25$.

Насправді частота коливань системи буде трохи меншою, оскільки вона визначається формулою $\omega = \sqrt{\omega_0^2 - \xi^2}$.

Для перевірки правильності програмної реалізації моделі коливальної системи можна використовувати такі формули:

$$y(t) = A_0 e^{-\xi t} \cos \omega t, \quad (2.9)$$

$$A(t) = A_0 e^{-\xi t}, \quad (2.10)$$

де $A_0 = y(0)$ – початкова амплітуда коливань.

Формула (2.9) є аналітичним розв’язком рівняння (2.8) при $\xi < \omega_0$. Формула (2.10) описує зміну в часі амплітуди згасаючих коливань.

Виконайте моделювання системи, яка описується рівнянням (2.8) з вихідними даними згідно з варіантом з табл. 2.1 (для всіх варіантів значення власної частоти, коефіцієнта згасання та кроку інтегрування ідентичні, тобто $a_1 = 0.5$, $a_2 = 39.4784176$, $h = 0.02$). Для цього:

- представте рівняння (2.8) у вигляді (2.1);
- перетворіть це рівняння на систему рівнянь згідно з методикою, викладеною вище (див. формули (2.2)–(2.4));
- розробіть програму для моделювання (файл **lab2-1.py**).

Приклад реалізації програми наведено у лістингу 2.1. Програмна реалізація у даному прикладі використовує ті самі методи, що й у лабораторній роботі № 1, які описані у поясненнях до лістингу 1.1. Графічний результат роботи програми наведено на рис. 2.1.

Таблиця 2.1 – Варіанти індивідуальних завдань

№	1	2	3	4	5	6	7	8	9	10
$y(0)$	8	-4	10	3	6	-10	-5	6	-8	5
$\dot{y}(0)$	0.4	1.0	-12	0.2	0.7	1.5	0.8	-0.6	1.2	0.5
t	10	8	12	10	12	8	6	10	12	8

Лістинг 2.1 Програма моделювання коливальної системи

```

1  import numpy as np
2  from scipy.integrate import odeint
3  import matplotlib.pyplot as plt
4
5  def dvdt(v, t, c):
6      x1, x2 = v
7      c1, c2 = c
8      dx1dt = x2
9      dx2dt = -c1 * x2 - c2 * x1
10     return [dx1dt, dx2dt]
11
12  a1, a2 = 0.5, 39.4784176
13  coeff = (a1, a2)
14  x0, y0 = 4.0, 0.3
15  v0 = [x0, y0]
16  t0, t1, h = 0.0, 9.0, 0.02
17  n = int((t1-t0) / h)
18
19  t = np.linspace(t0, t1, n)
20  solution = odeint(dvdt, v0, t, args=(coeff, )) # args
      must be tuple!
21  x, y = solution.T # T is the transposition operator
22
23  fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
24  plt.grid(True)
25  ax1.plot(t, x, label=r'$y$')
26  ax1.plot(t, y, label=r'$\dot{y}$')
27  ax1.set_xlabel(r't')
28  ax1.set_ylabel(r'$y, \dot{y}$')
29  ax1.legend(loc='upper right')
30  ax1.grid(True)
31
32  ax2.plot(x, y, color='black')
33  ax2.set_xlabel(r'$y$')
34  ax2.set_ylabel(r'$\dot{y}$')
35  ax1.grid(True)
36  plt.show()

```

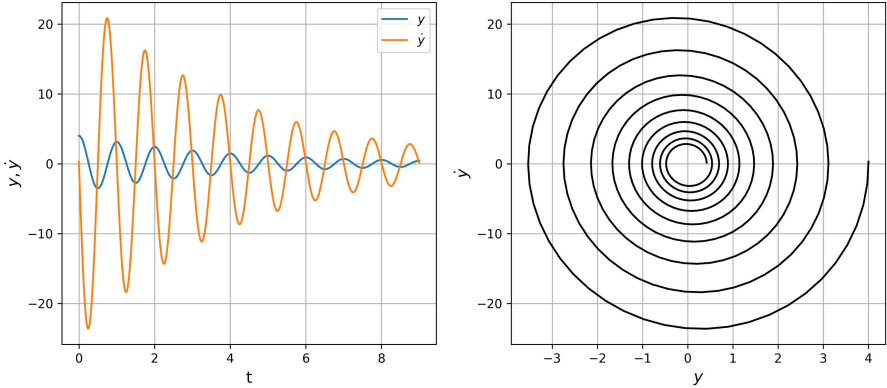


Рисунок 2.1 – Моделювання коливальної системи

2.2.2 Моделювання системи, що описується диференціальним рівнянням четвертого порядку

У цій частині роботи треба змоделювати систему, яка описується диференціальним рівнянням четвертого порядку¹:

$$y''''(t) + a_1 y'''(t) + a_2 y''(t) + a_3 y'(t) + a_4 y(t) = f(t), \quad (2.11)$$

де $f(t)$ – зовнішня сила, яка діє на систему та впливає на її поведінку і задана функцією

$$f(t) = A \sin(\omega t + \varphi).$$

Для моделювання цієї системи **виконайте наступне**:

- за допомогою методики, викладеної вище (див. рівняння (2.1)–(2.4)), перетворіть (2.11) в систему з чотирьох рівнянь першого порядку;
- на основі програми з лістингу 2.1 розробіть програму моделювання даної системи (файл **lab2-2.py**) з вихідними даними згідно з варіантом (див. табл. 2.2) і кроком інтегрування $h = 0.1$;
- у програмі забезпечте побудову графіків залежностей $y(t)$ і $y'(t)$.

¹У цьому розділі похідні позначаються за допомогою символу «апостроф»

Приклад реалізації програми (див. ліст. 2.2) з наступними значеннями вихідних даних:

- параметри системи: $a_1 = 0.8$, $a_2 = 2.0$, $a_3 = 0.6$, $a_4 = 0.4$;
- початкові умови (стан) системи: $y(0) = -5$, $y'(0) = 8$,
 $y''(0) = 1$, $y'''(0) = 2$;
- параметри інтегрування: час початку $t_0 = 0$, час закінчення $t = 80$, крок $h = 0.1$;
- параметри зовнішньої сили: амплітуда $A = 2$, кругова частота $\omega = \pi/4$, початкова фаза $\varphi = \pi/8$.

Графічний результат роботи програми наведено на рис. 2.2.

Лістинг 2.2 Моделювання системи четвертого порядку

```

1  import numpy as np
2  from scipy.integrate import odeint
3  import matplotlib.pyplot as plt
4
5  def f(t):
6      A, omega, fi = 2, np.pi / 4, np.pi / 8
7      return A*np.sin(omega*t + fi)
8
9  def dvdt(v, t, c, f):
10     x1, x2, x3, x4 = v
11     c1, c2, c3, c4 = c
12     dx1dt = x2
13     dx2dt = x3
14     dx3dt = x4
15     dx4dt = -c1*x4 - c2*x3 - c3*x2 - c4*x1 + f(t)
16     return [dx1dt, dx2dt, dx3dt, dx4dt]
17
18  a1, a2, a3, a4 = 0.8, 2.0, 0.6, 0.4
19  coeff = (a1, a2, a3, a4)
20  y1, y2, y3, y4 = -5.0, 8.0, 1.0, 2.0
21  v0 = [y1, y2, y3, y4]
22  t0, t1, h = 0.0, 80.0, 0.1
23  n = int((t1-t0) / h)
24  t = np.linspace(t0, t1, n)
25  solution = odeint(dvdt, v0, t, args=(coeff, f))
26  y = solution.T
27  lf_size = 13
28  label = [r"$y(t)$", r"$y'(t)$", r"$y''(t)$", r"$y'''(t)$"]
29  fig, ax = plt.subplots(figsize=(12, 5))

```

```

30 for i in range(0, len(y)):
31     ax.plot(t, y[i], label=label[i])
32 ax.legend(loc='upper right')
33 ax.set_xlabel(r'$t$', fontsize=lf_size)
34 ax.set_xlim(0, t1)
35 ax.grid(True)
36 plt.savefig('fig2-2.png', dpi=300)
37 plt.show()

```

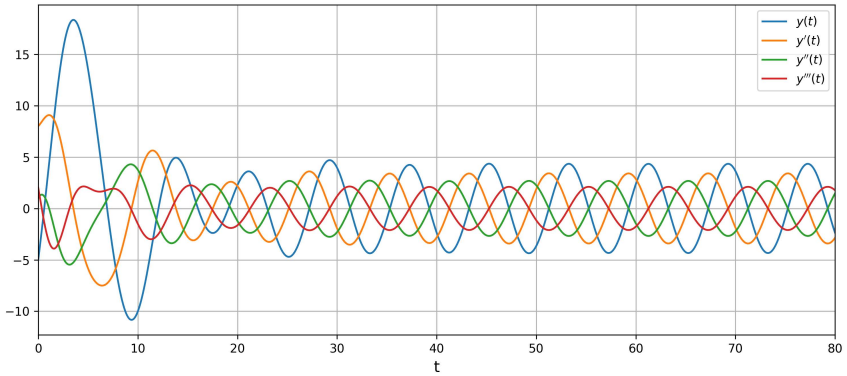


Рисунок 2.2 – Моделювання системи четвертого порядку

Таблиця 2.2 – Варіанти індивідуальних завдань

№	1	2	3	4	5	6	7	8	9	10
a_1	0.5	0.2	0.25	0.75	0.6	0.3	0.06	0.5	0.4	0.2
a_2	20	0.16	1	0.15	1.2	0.8	64	14	0.6	0.4
a_3	4	0.01	0.01	0.025	0.08	0.1	32	0.8	0.012	0.02
a_4	5	0.002	0.04	0.002	0.04	0.12	24	6	0.025	0.015
A	3	6	20	10	4	16	100	40	0.2	1.5
ω	$\pi/2$	$\pi/8$	$\pi/4$	$\pi/3$	$\pi/4$	$\pi/2$	$\pi/2$	π	$\pi/6$	$\pi/4$
φ	$\pi/4$	$\pi/6$	$\pi/8$	$\pi/12$	$\pi/8$	$\pi/4$	$-\pi/8$	$\pi/6$	$\pi/12$	$\pi/10$
$y(0)$	5	300	20	-40	-5	-20	-1	1.5	2.5	-4.5
$y'(0)$	0.5	-2	-5	10	8	40	4	-2.5	1.5	4
$y''(0)$	1	2	-8	-10	0	-5	-2	-1	-0.8	-0.2
$y'''(0)$	0	0	0	0	0	0	0	0	0	0
t	50	120	150	100	80	50	10	15	150	120

2.2.3 Перевірка властивості адитивності системи

Як було викладено в п. 2.1 лінійна система має властивість адитивності (2.7). Перевірку цієї властивості виконаємо за допомогою моделі та програми, розробленої згідно з варіантом попереднього розділу (п. 2.2.2), наступним чином:

- на основі програми, розробленої в п. 2.2.2, створіть новий файл програми (**lab2-3.py**);
- переробить програму так, щоб на екран виводилось одночасно чотири графічних вікна і в кожному з них будувались тільки графіки $y(t)$ і $y'(t)$ (приклад реалізації див. ліст. 2.3);
- код програми повинен створювати розв’язки, результати яких помістити послідовно у чотирьох графічних вікнах відповідно до нумерації розв’язків.

Розв’язок 1 – при заданих початкових умовах та наявності зовнішньої сили (це вихідні дані – розв’язок, отриманий у попередньому завданні, див. рядок 28 ліст. 2.3). Формально запишемо цей розв’язок у вигляді

$$\mathfrak{R}_1 : y^{(n)}(0) = y_0^{(n)}; f(t) \neq 0.$$

Розв’язок 2 – за умови, що зовнішня сила дорівнює нулю $f(t) = 0$. Формально:

$$\mathfrak{R}_2 : y^{(n)}(0) = y_0^{(n)}; f(t) = 0.$$

Цей розв’язок (рядок 31) можна отримати, якщо додати у програму функцію обчислення зовнішньої сили (рядок 9), яка повертає нульове значення, та використовувати саме цю функцію у виразі для обчислення похідної (рядок 12).

Розв’язок 3 – при нульових початкових умовах та заданій зовнішній силі (рядок 35). Формально:

$$\mathfrak{R}_3 : y^{(n)}(0) = 0; f(t) \neq 0.$$

Цей розв’язок можна отримати, якщо встановити вектор початкових умов з нульовими значеннями (рядок 34).

Розв'язок 4 – цей розв'язок є простою сумою розв'язку 2 і розв'язку 3:

$$\mathfrak{R}_4 = \mathfrak{R}_2 + \mathfrak{R}_3.$$

Цей розв'язок легко отримати, застосовуючи операцію додавання (*Python*), яка виконає векторне складання розв'язків, одержаних у результаті інтегрування (рядок 38).

Просте порівняння графіків у першому та четвертому вікнах має показати збіг графіків, що свідчить про наявність властивості адитивності (можна також говорити, що виконується *принцип суперпозиції*).

Лістинг 2.3 Перевірка властивості адитивності системи

```

1  import numpy as np
2  from scipy.integrate import odeint
3  import matplotlib.pyplot as plt
4
5  def f(t):
6      A, omega, fi = 2, np.pi / 4, np.pi / 8
7      return A*np.sin(omega*t + fi)
8
9  def f0(t):
10     return 0
11
12  def dvdt(v, t, c, f):
13     x1, x2, x3, x4 = v
14     c1, c2, c3, c4 = c
15     dx1dt = x2
16     dx2dt = x3
17     dx3dt = x4
18     dx4dt = -c1*x4 - c2*x3 - c3*x2 - c4*x1 + f(t)
19     return [dx1dt, dx2dt, dx3dt, dx4dt]
20
21  a1, a2, a3, a4 = 0.8, 2.0, 0.6, 0.4
22  coeff = (a1, a2, a3, a4)
23  y1, y2, y3, y4 = -5.0, 8.0, 1.0, 2.0
24  v0 = [y1, y2, y3, y4]
25  t0, t1, h = 0.0, 80.0, 0.1
26  n = int((t1-t0) / h)
27  t = np.linspace(t0, t1, n)
28  solution = odeint(dvdt, v0, t, args=(coeff, f))

```

```

29 R1 = solution.T
30
31 solution = odeint(dvdt, v0, t, args=(coeff, f0))
32 R2 = solution.T
33
34 v0 = [0, 0, 0, 0]
35 solution = odeint(dvdt, v0, t, args=(coeff, f))
36 R3 = solution.T
37
38 R4 = R2 + R3
39
40 lf_size = 13
41 fig, ax = plt.subplots(nrows=2,ncols=2,figsize=(12, 6))
42 axi = (ax[0, 0], ax[0, 1], ax[1, 0], ax[1, 1])
43 label = [r"$y(t)$", r"$y'(t)$"]
44 tl1 = 'Вихідний варіант'
45 tl2 = 'Нульова зовнішня сила'
46 tl3 = 'Нульові початкові умови'
47 tl4 = 'Суперпозиція'
48 title = [tl1, tl2, tl3, tl4]
49 for i in range(0, 2):
50     ax[0, 0].plot(t, R1[i], label=label[i])
51     ax[0, 1].plot(t, R2[i], label=label[i])
52     ax[1, 0].plot(t, R3[i], label=label[i])
53     ax[1, 1].plot(t, R4[i], label=label[i])
54 for i, ax in enumerate(axi):
55     ax.set_title(title[i])
56     ax.legend(loc='upper right')
57     ax.grid(True)
58 fig.subplots_adjust(hspace=0.35)
59 plt.savefig('fig2-3-1.png', dpi=300)
60 plt.show()

```

2.2.4 Перевірка властивості однорідності системи

Для перевірки однорідності лінійної системи (2.6) виконайте наступне:

- на основі програми, розробленої в п. 2.2.2, створіть новий файл програми (**lab2-4.py**);
- переробить програму так, щоб на екран виводилося одночасно два вікна з графіками і в кожному вікні будувалися графіки

$y(t)$ і $y'(t)$ (приклад реалізації див. ліст. 2.4);

– код програми повинен створювати розв’язки, результати яких помістять послідовно у двох графічних вікнах відповідно до нумерації розв’язків.

Розв’язок 1 – його можна отримати, якщо вихідний розв’язок (рядок 26) помножити на коефіцієнт k (рядок 29). Формально запишемо цей розв’язок у вигляді

$$\mathfrak{R}_1 : k \cdot \mathfrak{R},$$

де $\mathfrak{R}$ – вихідний розв’язок.

Розв’язок 2 – цей розв’язок отримано шляхом попереднього множення початкових умов та зовнішньої сили на коефіцієнт k . Формально:

$$\mathfrak{R}_2 : y^{(n)}(0) = k \cdot y_0^{(n)}; \quad k \cdot f(t).$$

У прикладі реалізації (ліст. 2.4) до функції обчислення похідних додано параметр зі значенням за замовчуванням $k=1$ (див. рядки 9 і 15), який також необхідно передати у функцію інтегрування, якщо k відрізняється від одиниці (порівняйте рядки 26 і 31).

Порівняння графіків у першому та другому вікнах має показати їх збіг, що свідчить про наявність властивості однорідності системи. Такі системи ще називають *гомогенними*.

Лістинг 2.4 Перевірка властивості однорідності системи

```

1  import numpy as np
2  from scipy.integrate import odeint
3  import matplotlib.pyplot as plt
4
5  def f(t):
6      A, omega, fi = 2, np.pi / 4, np.pi / 8
7      return A*np.sin(omega*t + fi)
8
9  def dvdt(v, t, c, f, k=1):
10     x1, x2, x3, x4 = v
11     c1, c2, c3, c4 = c
12     dx1dt = x2
13     dx2dt = x3
14     dx3dt = x4
```

```

15     dx4dt = -c1*x4 - c2*x3 - c3* x2 - c4*x1 + k*f(t)
16     return [dx1dt, dx2dt, dx3dt, dx4dt]
17
18 a1, a2, a3, a4 = 0.8, 2.0, 0.6, 0.4
19 coeff = (a1, a2, a3, a4)
20 y1, y2, y3, y4 = -5.0, 8.0, 1.0, 2.0
21 v0 = np.array([y1, y2, y3, y4]) #!!!!
22 t0, t1, h = 0.0, 80.0, 0.1
23 n = int((t1-t0) / h)
24
25 t = np.linspace(t0, t1, n)
26 solution = odeint(dvdt, v0, t, args=(coeff, f, ))
27
28 k = 2
29 R1 = k*solution.T
30 v1 = k*v0
31 solution = odeint(dvdt, v1, t, args=(coeff, f, k))
32 R2 = solution.T
33
34 lf_size = 13
35 fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,
36     3))
37 axi = (ax[0], ax[1])
38 label = [r"$y(t)$", r"$y'(t)$"]
39 t11 = 'Вихідний варіант ' + r'$\times$ k$'
40 t12 = 'Початкові умови '+r'$\times$ k$'+ ' та зовнішня си
41     ла '+r'$\times$ k$'
42 title = [t11, t12]
43 for i in range(0, 2):
44     ax[0].plot(t, R1[i], label=label[i])
45     ax[1].plot(t, R2[i], label=label[i])
46 for i, ax in enumerate(axi):
47     ax.set_title(title[i])
48     ax.legend(loc='upper right')
49     ax.grid(True)
50     plt.savefig('fig2-3-2.png', dpi=300)
51 plt.show()

```

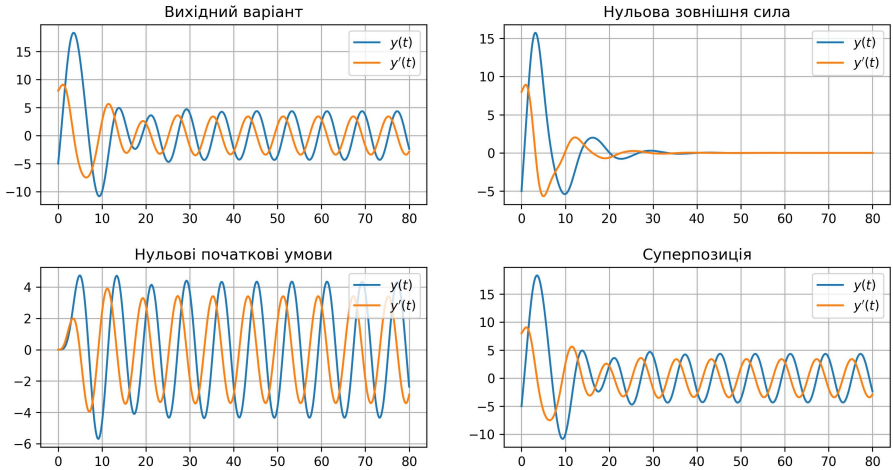


Рисунок 2.3 – Перевірка властивості адитивності системи

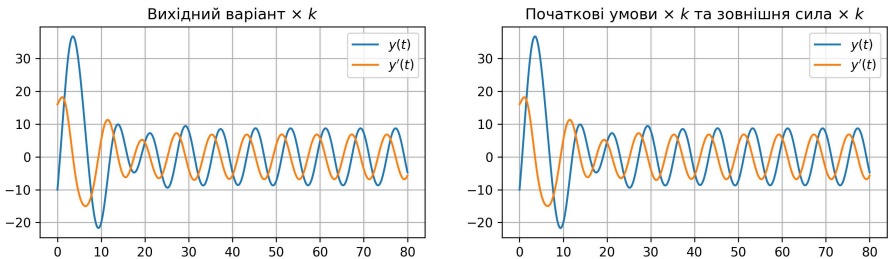


Рисунок 2.4 – Перевірка властивості однорідності системи

2.3 Контрольні питання

1. Які з розглянутих раніше моделей систем є лінійними, нелінійними? Відповідь обґрунтуйте.
2. Які властивості має лінійна система? Дайте визначення цих властивостей.
3. Які моделі систем є безперервними та дискретними, детермінованими та стохастичними, стаціонарними та нестаціонарними?
4. Чи дійсно існують лінійні системи у природі?

5. Чи може модель системи бути одночасно лінійною, стохастичною та нестационарною?
6. Наведіть приклад диференціального рівняння третього порядку для моделювання нестационарної системи.
7. Як описати функціонування системи у вигляді кінцевого автомата?
8. Що таке стан системи і чому його ще називають вектором станів?
9. З яких елементів складається вектор станів розглянутої у п. 2.2.1 коливальної системи?
10. Поясніть поняття «чорна скринька» та «біла скринька», які використовуються для опису моделей систем.
11. Як здійснюється перетворення диференціального рівняння n -го порядку до системи з n рівнянь першого порядку? Наведіть приклад перетворення диференціального рівняння четвертого порядку.
12. Напишіть фрагмент коду для ліст. 2.4, який перевіряє збіг графіків на рис. 2.4.

2.4 Зміст звіту з лабораторної роботи

1. Титульний лист, оформлений згідно з вимогами.
2. Мета роботи.
3. Лістинги програм, розроблених відповідно до завдань.
4. Результати виконання програм у вигляді графіків.
5. Відповіді на контрольні питання.
6. Висновки за результатами досліджень.

ЛАБОРАТОРНА РОБОТА 3

Моделювання просторово-розподілених процесів

Мета роботи — опанувати методи комп’ютерного моделювання просторово-розподілених процесів, дослідити моделі та їх властивості.

3.1 Теоретичні відомості

Більшість відомих у природі явищ і процесів поширюються в часі та просторі. Фізичні закони, що визначають їхню поведінку, мають безперервний характер та описуються диференціальними рівняннями в частинних похідних. Це електричні і магнітні поля, поширення тепла і дифузійні процеси, теорія пружності тощо.

Прикладами таких диференціальних рівнянь є:

– рівняння Пуассона

$$\nabla^2 V = \frac{\partial V^2}{\partial x^2} + \frac{\partial V^2}{\partial y^2} = f(x, y); \quad (3.1)$$

– рівняння теплового потоку і дифузії

$$\nabla^2 V = K \frac{\partial V}{\partial t}; \quad (3.2)$$

– хвильове рівняння

$$\nabla^2 V = \alpha^2 \frac{\partial^2 V}{\partial t^2}. \quad (3.3)$$

Роздивимось одну з методик розв’язання рівняння Лапласа, яке є частковим випадком рівняння Пуассона та має вигляд:

$$\nabla^2 V = \frac{\partial V^2}{\partial x^2} + \frac{\partial V^2}{\partial y^2} = 0. \quad (3.4)$$

Формулюємо задачу таким чином. Необхідно знайти в якійсь області (ділянці) Ω на площині xy безперервну функцію $V(x, y)$, яка

задовольняє (3.4) та приймає на межі області Ω значення згідно з граничними функціями: $\varphi_1(x)$ (на нижній межі), $\varphi_2(x)$ (на верхній межі), $\psi_1(y)$ (на лівій межі), $\psi_2(y)$ (на правій межі). Ці функції називають *граничними умовами*.

Така задача відома під назвою *задачі Діріхле* для рівняння Лапласа. Ця задача є класичною та часто використовується для демонстрації переваг її розв'язування паралельними методами в дисципліні «Паралельні та розподілені обчислення». В загальному випадку межа ділянки Ω може бути довільною, проте ми будемо розглядати задачу Діріхле на прямокутній ділянці, сторони якої дорівнюють a і b . Для розв'язування диференціальних рівнянь у частинних похідних частіше використовують кінцево-різницеві методи, в яких частинні похідні апроксимуються різницеви́ми операторами.

Отже, розглянемо прямокутну ділянку Ω на площині xy з рівномірною (для спрощення) сіткою з кроком h (рис. 3.1).

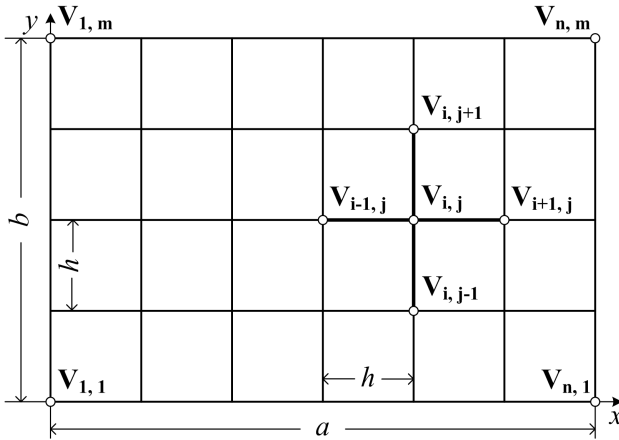


Рисунок 3.1 – Прямокутна ділянка Ω та позначення її вузлів

Позначимо значення функції в вузлах сітки через $V_{i,j}$. Тоді частинні похідні в рівнянні Лапласа можна апроксимувати централь-

ними кінцевими різницями другого порядку:

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{h^2} + O(h^2), \quad (3.5)$$

$$\frac{\partial^2 V}{\partial y^2} \approx \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{h^2} + O(h^2), \quad (3.6)$$

де $O(h^2)$ – члени другого порядку малості; $i = 1, 2, \dots, n$ – індекси точок уздовж осі x ; $j = 1, 2, \dots, m$ – індекси точок уздовж осі y .

Вузли сітки (i, j) внутрішньої області Ω будемо називати внутрішніми, вузли на межі – граничними.

Осі x і y в кутах області Ω перетинаються, тому очевидно, що повинні виконуватися такі граничні умови:

$$\begin{aligned} \varphi_1(0) &= \psi_1(0), \\ \varphi_1(a) &= \psi_2(0), \\ \varphi_2(0) &= \psi_1(b), \\ \varphi_2(a) &= \psi_2(b). \end{aligned} \quad (3.7)$$

Таким чином, оскільки граничні умови для задачі Діріхле є заданими, то нам відомі значення функції $V_{i,j}$ на граничних вузлах, тобто відомі значення $V_{i,1}$, $V_{i,m}$, $V_{1,j}$ та $V_{n,j}$ для усіх $i = 1, \dots, n$ та $j = 1, \dots, m$.

Очевидно, що індекси внутрішніх вузлів сітки мають значення $i = 2, \dots, n - 1$ та $j = 2, \dots, m - 1$. Якщо (3.5) та (3.6) підставити у (3.4), то, нехтуючи членами другого порядку малості, рівняння Лапласа для кожного внутрішнього вузла можна записати:

$$V_{i,j} = \frac{1}{4}(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}). \quad (3.8)$$

Наприклад, для внутрішнього вузла $(2, 2)$ рівняння (3.8) матиме вигляд

$$V_{2,2} = \frac{1}{4}(V_{3,2} + V_{1,2} + V_{2,3} + V_{2,1}).$$

Як можна помітити, значення функції $V_{i,j}$ в кожному внутрішньому вузлі є *середнім арифметичним* значень функції у чотирьох сусідніх вузлах. Такий вид рівнянь виник завдяки обраному способу апроксимації частинних похідних другого порядку центральними різницями другого порядку та зветься *чотирьох точковим* (або *хрестоподібним*) *шаблоном*. В залежності від способу апроксимації можна отримати інші шаблони, наприклад, 6-точковий, 8-точковий.

Одним з методів знаходження значень $V_{i,j}$ на внутрішніх вузлах сітки області Ω є *ітераційний метод Якобі*, який для кожного рівняння (3.8) може бути записаний у вигляді рекурентного виразу:

$$V_{i,j}^{k+1} = \frac{1}{4}(V_{i+1,j}^k + V_{i-1,j}^k + V_{i,j+1}^k + V_{i,j-1}^k), \quad (3.9)$$

де $k = 0, 1, 2, \dots$ – номер ітерації.

Щоб розпочати ітераційний процес (першу ітерацію) необхідно задати початкові значення функції $V_{i,j}^0$ на внутрішніх вузлах сітки. Якщо нам нічого невідомо про очікуваний розв'язок, то ці початкові значення можна обрати довільними.

Ітераційний процес закінчується, коли результати обчислення значень функції $V_{i,j}$ у *всіх внутрішніх* вузлах сітки на даній ітерації відрізняються від значень функції попередньої ітерації не більш ніж припустима похибка ε – деяке мале попередньо задане додатне число. Таким чином, умову закінчення ітераційного процесу можна записати у вигляді:

$$\max_{\forall i,j} |V_{i,j}^k - V_{i,j}^{k-1}| < \varepsilon. \quad (3.10)$$

3.2 Практичне завдання

3.2.1 Розв'язання контрольної задачі

Змодельуйте рівняння Лапласа, що описується (3.4), за допомогою ітераційного методу Якобі на області Ω з граничними функціями:

$$\varphi_1(x) = \sin(x) - x^3 \sin(1), \quad \varphi_2(x) = x^2, \quad \psi_1(y) = 0, \quad \psi_2(y) = y^3. \quad (3.11)$$

Для отримання навичок розв’язання рівняння Лапласа ітеративним методом Якобі реалізуйте програму моделювання контрольної задачі (див. ліст. 3.1) з наступними значеннями параметрів, які ви можете використати при виконанні індивідуальних завдань:

- прямокутна область Ω зі сторонами $a = 1$, $b = 1$ є областю визначення функції $V_{i,j}$;
- крок дискретизації по осях x і y $h = 0.1$;
- кількість точок по осі x визначається як $n = a/h + 1$, а по осі y – як $m = b/h + 1$;
- максимальна припустима похибка $\varepsilon = 10^{-4}$.

Графічний результат роботи програми наведено на рис. 3.2.

Лістинг 3.1 Контрольна задача

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  h = 0.1
5  n = int(1 / h) + 1
6  x = np.linspace(0, 1, n)
7  y = np.linspace(0, 1, n)
8
9  v = np.random.random_sample(size=(n, n))
10 v[:, 0] = np.sin(x) - x**3 * np.sin(1.0)
11 v[:, n-1] = x**2
12 v[0, :] = 0.0
13 v[n-1, :] = y**3
14 v = v.T
15 v_initial = np.copy(v)
16 u = np.copy(v_initial)
17
18 x, y = np.meshgrid(x, y)
19
20 eps = 0.0001
21 err = 1.0
22 k = 0
23 while err > eps:
24     err = 0.0
25     for i in range(1, n-1):
26         for j in range(1, n-1):
27             v[i, j] = 0.25*(u[i+1, j] + u[i-1, j] +
                u[i, j+1] + u[i, j-1])

```

```

28         dif = np.abs(v[i, j] - u[i, j])
29         if dif > eps:
30             err = dif
31     u = np.copy(v)
32     k += 1
33
34     lf_size = 13
35     fig = plt.figure(figsize=plt.figaspect(0.45))
36     ax = fig.add_subplot(1, 2, 1, projection='3d')
37     ax.set_xlabel('x', fontsize=lf_size)
38     ax.set_ylabel('y', fontsize=lf_size)
39     ax.set_zlabel('v', fontsize=lf_size)
40     ax.view_init(azim=-118, elev=43)
41     ax.plot_wireframe(x, y, v_initial, color='black')
42     ax = fig.add_subplot(1, 2, 2, projection='3d')
43     ax.set_xlabel('x', fontsize=lf_size)
44     ax.set_ylabel('y', fontsize=lf_size)
45     ax.set_zlabel('v', fontsize=lf_size)
46     ax.view_init(azim=-118, elev=43)
47     ax.plot_wireframe(x, y, v, color='black')
48     plt.savefig('fig3-1.png', dpi=300)
49     plt.show()
50
51     print('azim=', ax.azim)
52     print('elev=', ax.elev)

```

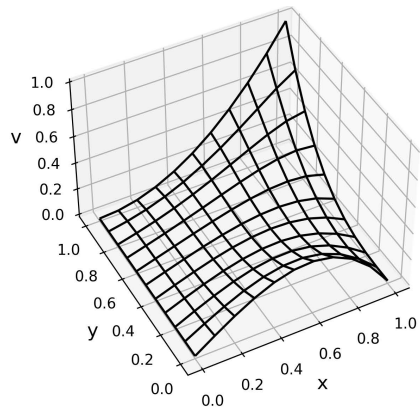
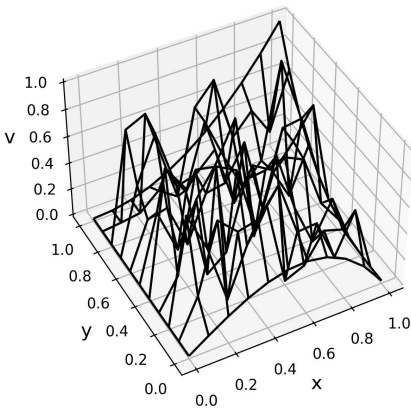


Рисунок 3.2 – Графіки розв’язку контрольної задачі

Пояснення до реалізації програми (лістинг 3.1)

Нагадаємо, що у мовах програмування номери індексів масивів починаються з нуля. Отже, точки сітки у програмі, наприклад по осі x , мають номери від 0 до $n - 1$, а граничні значення задачі на цій осі будуть, відповідно, у точках $x[0]=0$ і $x[n-1]=a$. Таким чином, внутрішні точки, в яких ітераційно обчислюються значення функції $V_{i,j}$, матимуть індекси від 1 до $n - 2$.

У рядках 6 і 7 створюються точки, що рівномірно розподілені по осях x і y (від 0 до $n - 1$) та є координатами області визначення.

У рядку 9 створюється двовимірний масив розміром $n \times n$ та заповнюється випадковими дійсними числами, що знаходяться в діапазоні $0 \leq v < 1.0$, які у нашому прикладі є початковими значеннями функцій у точках області, тобто значеннями $V_{i,j}^0$ з яких почнеться ітераційний процес.

У рядках з 10 по 13 задаються граничні умови $\varphi_1(x)$, $\varphi_2(x)$, $\psi_1(y)$, $\psi_2(y)$.

Початковий стан, тобто область із визначеними (випадковими) значеннями функцій $V_{i,j}^0$ у внутрішніх вузлах і заданими на межі області, зберігається в додатковому масиві для наступного виведення на графік (рядок 15). Зверніть увагу, що для збереження даних використовується функція копіювання (при використанні простої операції присвоєння буде створено тільки нове посилання на наявний масив). Такий саме метод використовується для створення робочого масиву для ітерацій (рядок 16).

Функція `meshgrid()` з бібліотеки `numpy` (рядок 18) у програмі призначена для створення прямокутної сітки з двох заданих одно-вимірних масивів, що представляють декартову або матричну індексацію, та полегшує побудову тривимірних графіків.

Після виведення на екран тривимірної фігури можна повернути її мишкою та вибрати зручну точку спостереження. Висота та азимут у градусах виведуться на друк (рядки 51, 52) після закриття графічного вікна. Ці значення можна підставити у функцію `view_init()` (рядок 40) для подальшого використання.

3.2.2 Індивідуальне завдання

Для виконання індивідуального завдання за варіантами (див. табл. 3.1, a і b по кожному) виконайте наступне:

- перевірте виконання граничних умов, використовуючи рівняння (3.7);
- створіть власну програму з вихідними даними згідно з варіантом та отримайте результат її роботи;
- зробіть оцінку середнього значення кількості необхідних ітерацій, виконавши моделювання задачі декілька разів.

Таблиця 3.1 – Варіанти індивідуальних завдань

№	$\varphi_1(x)$	$\varphi_2(x)$	$\psi_1(y)$	$\psi_2(y)$
1a	x^3	$x + 1$	y^2	$\cos y + (2 - \cos y)y$
1b	0	$30(1 - x^2)$	$30y$	0
2a	$1 - x^3$	x^2	$e^y - ey^2$	y
2b	$50 - \sin \pi x$	$50x$	$50(1 - y^2)$	50
3a	$\sin x - (1 + \sin 1)x^3$	$x - 1$	$-y^2$	$y - 1$
3b	$50x(1 - x)$	20	$20y$	$20y^2$
4a	1	$x + 1$	$e^y + (1 - e)y^2$	$y + 1$
4b	$50x(1 - x)$	$50x(1 - x)$	0	$50y(1 - y^2)$
5a	$\sin x - x^3 \sin 1$	x^2	0	y
5b	$30(1 - x)$	$20\sqrt{x}$	$30(1 - y)$	$20y$
6a	$2 - x^3$	$x - 1$	$2e^y - (1 + 2e)y^2$	$1 - y$
6b	$50 \sin \pi x$	$30\sqrt{x}$	$50 \sin \pi y$	$30y^2$
7a	$9x^2 + 7x + 6$	$9x^2 - 15x - 12$	$-10y^2 - 8y + 6$	$-10y^2 - 30y + 22$
7b	0	$40(1 - x)$	$40\sqrt{y}$	$20y(1 - y)$
8a	$6x^2 + 4x + 3$	$6x^2 - 12x - 9$	$-7y^2 - 5y + 3$	$-7y^2 - 21y + 13$
8b	$e^x - (1 + e)x^2$	0	$1 - y^2$	$y - 1$
9a	$e^{x \cdot x}$	$e^{x \cdot x - 1}$	$e^{-y \cdot y}$	$e^{1 - y \cdot y}$
9b	$-x^3$	$1 - x^3$	y^2	$y^2 - 1$
10a	x^2	$x^2 - 1$	$-y^3$	$1 - y^3$
10b	0	x	$\sin y - y^3 \sin 1$	y^2

3.3 Контрольні питання

1. Наведіть приклади диференціальних рівнянь, які описуються частинними похідними.
2. Наведіть формули апроксимації частинних похідних різницеви операторами.

3. Що таке шаблон апроксимації? Які шаблони бувають? Який шаблон застосовується в даній задачі?

4. Чим визначається кількість ітерацій при пошуку розв'язків рівнянь у частинних похідних методом Якобі?

5. Що є джерелом похибки розв'язку рівнянь у частинних похідних кінцево-різницевиими методами?

6. Що таке граничні умови? Які особливості граничних умов даної задачі?

3.4 Зміст звіту з лабораторної роботи

1. Титульний лист, оформлений відповідно до вимог.
2. Мета роботи.
3. Розрахунок перевірки граничних умов за варіантом.
4. Лістинги програм, розроблених відповідно до завдань.
5. Результати виконання програм у вигляді графіків.
6. Відповіді на контрольні питання.
7. Висновки за результатами досліджень.

ЛАБОРАТОРНА РОБОТА 4

Моделювання систем, що описуються рівняннями теплопровідності

Мета роботи — опанувати методи моделювання систем, що описуються диференціальними рівняннями в частинних похідних.

4.1 Теоретичні відомості

При розробці технічних систем роль відіграє моделювання розповсюдження тепла в деякому середовищі у часі. Швидкість зміни температури в точках середовища описується за допомогою рівняння теплопровідності, яке для одновимірного випадку має вигляд:

$$\frac{\partial T}{\partial t} = a \frac{\partial^2 T}{\partial y^2}, \quad (4.1)$$

де $a = \lambda / c\rho$ – коефіцієнт температуропровідності речовини, з якої складається середовище та який характеризує швидкість зміни температури в середовищі, $\text{м}^2/\text{с}$;

λ – коефіцієнт теплопровідності речовини, $\text{Вт}/(\text{м}\cdot\text{град})$;

c – питома теплоємність, $\text{Дж}/(\text{кг}\cdot\text{град})$ або $\text{Вт}\cdot\text{с}/(\text{кг}\cdot\text{град})$;

ρ – щільність речовини, $\text{кг}/\text{м}^3$.

Розроблено багато методів розв’язування рівняння теплопровідності, таких як метод Фур’є, метод відокремлення змінних, метод кінцевих різниць, що аналогічний викладеному в лабораторній роботі №3. В останньому випадку, одновимірна задача розв’язується на двовимірній сітці, один вимір якої являє товщину простору поширення тепла, а інший – час. У даній лабораторній роботі використовується метод, в якому зміна температури у часі обчислюється за допомогою чисельного інтегрування.

Розглянемо стіну, що складається з однорідної речовини, яка розділяє два середовища з різними температурами (рис. 4.1). Стіна має товщину L . Припустимо, що ширина і висота стіни нескінченні.

Температура середовища зліва від стіни може змінюватися в часі за деяким законом $\varphi_1(t)$, а справа – за законом $\varphi_2(t)$. Функції $\varphi_1(t)$ та $\varphi_2(t)$ назовемо *граничними умовами*. Початкове розподілення температури в точках стіни задається функцією $\varphi(y)$, яку будемо називати *початковими умовами*.

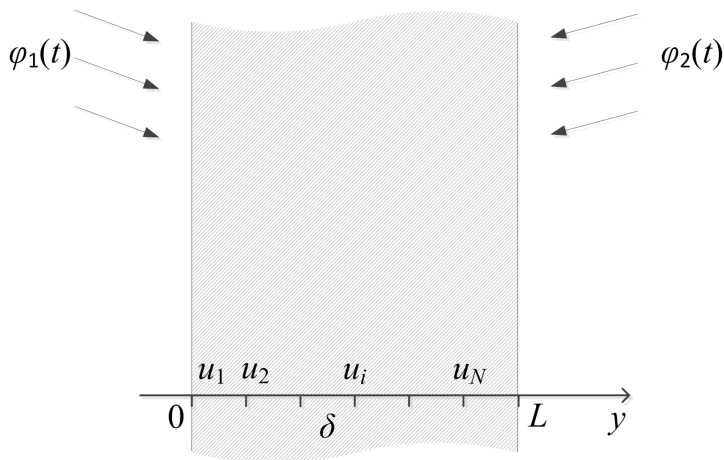


Рисунок 4.1 – Схема розбиття товщини стіни на шари

Будемо позначати температуру в точках стіни через $u(t, y)$. Припускаємо, що площа стіни нескінченна, а речовина – однорідна. Отже точки стіни, що знаходяться на одній вертикалі, мають однакову температуру. З урахуванням введених позначень, процес поширення температури в стіні буде описуватися рівнянням теплопровідності у вигляді

$$\frac{\partial u}{\partial t} = a \frac{\partial^2 u}{\partial y^2} \quad (4.2)$$

та умовами

$$u(0, y) = \varphi(y), \quad u(t, 0) = \varphi_1(t), \quad u(t, L) = \varphi_2(t). \quad (4.3)$$

Для отримання системи звичайних диференціальних рівнянь, що апроксимують рівняння теплопровідності, розділимо стіну вертикальними лініями на N шарів однакової товщини δ . На перетині

цих прямих з віссю y утвориться ряд точок, які пронумеруємо від $i = 1$ до $N + 1$. Тепер для кожної з позначених точок можна записати рівняння теплопровідності:

$$\frac{\partial u(t, y_i)}{\partial t} = a \frac{\partial^2 u(t, y_i)}{\partial y_i^2}. \quad (4.4)$$

Позначимо $u(t, y_i)$ через $u_i(t)$. Частинну похідну $\frac{\partial^2 u(t, y_i)}{\partial y_i^2}$ в кожній i -й точці можна апроксимувати за допомогою центральної різниці другого порядку:

$$\frac{\partial^2 u(t, y_i)}{\partial y_i^2} \approx \frac{u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)}{\delta^2} + O(\delta^2). \quad (4.5)$$

Введемо позначення $\mu = a/\delta^2$. Нехтуючи членами другого порядку малості $O(\delta^2)$ та враховуючи введені вище позначення, отримаємо ЗДР для кожної внутрішньої точки y_i :

$$\frac{du_i(t)}{dt} = \mu(u_{i+1}(t) - 2u_i(t) + u_{i-1}(t)). \quad (4.6)$$

Таких рівнянь буде стільки, скільки точок було визначено на осі y . Якщо кількість точок N , то всі рівняння разом складуть систему з N диференціальних рівнянь. Наприклад, для чотирьох точок система рівнянь буде виглядати наступним чином:

$$\begin{aligned} \dot{u}_1 &= \mu(u_2(t) - 2u_1(t) + \varphi_1(t)), \\ \dot{u}_2 &= \mu(u_3(t) - 2u_2(t) + u_1(t)), \\ \dot{u}_3 &= \mu(u_4(t) - 2u_3(t) + u_2(t)), \\ \dot{u}_4 &= \mu(\varphi_2(t) - 2u_4(t) + u_3(t)) \end{aligned} \quad (4.7)$$

при початкових умовах $u_i(0) = \varphi(y_i)$ та граничних умовах $u(t, 0) = \varphi_1(t)$, $u(t, L) = \varphi_2(t)$.

Таку систему рівнянь можна розв'язати за допомогою будь-яких методів чисельного інтегрування (наприклад, Рунге-Кутта).

4.2 Дослідження процесів зміни температури в стіні

4.2.1 Розв’язання контрольної задачі (стіна із заліза)

Змодельуйте процес зміни температури у залізній стіні методом зведення до системи звичайних диференціальних рівнянь.

Нехай в початковий момент температура у внутрішніх точках стіни розподілена за законом

$$\varphi(y) = 5 \sin(2\pi y),$$

тобто на лівій та правій границях стіни температура дорівнює нулю, а всередині – максимальна і дорівнює 5°C .

Від стіни ліворуч температура навколишнього середовища змінюється згідно з умовою

$$\varphi_1(t) = 2 \cos\left(\pi \frac{t}{T} - \frac{\pi}{2}\right),$$

де $T=60$ хв. – час спостереження за процесом (час моделювання).

Температура середовища від стіни праворуч дорівнює нулю:

$$\varphi_2(t) = 0^{\circ}\text{C}.$$

Розробіть для цієї задачі програму моделювання, за основу якої можна узяти приклад реалізації з ліст. 4.1. Графічний результат моделювання наведено на рис. 4.2. Змодельуйте за допомогою розробленої програми процеси, що описані у табл. 4.1.

Лістинг 4.1 Програма моделювання зміни температури у стіні

```
1 import numpy as np
2 import math
3 import matplotlib.pyplot as plt
4 from scipy.integrate import odeint
5
6 PI = math.pi
7 l = 0.5      # товщина стіни [м]
8 n = 101     # число точок
9 delta = 1/n  # товщина шару
```

```

10 a = 23.0E-6 # температуропровідність заліза [м*м/с]
11 s = 60      # кількість хвилин у годині
12 a = a*s     # перерахунок на хвилини
13 mu = a / delta**2 # параметр
14 t0, t1, h = 0.0, 60.0, 1.0
15 u0 = np.array([(5*math.sin(2*PI*i*delta)) for i in
    range(0, n)])
16
17 def fi1(t, t1):
18     return 2*math.cos(-PI/2 + PI*t/t1)
19
20 def fi2():
21     return 0
22
23 def dudt(u, t, n, mu, t1):
24     du = np.zeros(n)
25     du[0] = mu*(u[1] - 2*u[0] + fi1(t, t1))
26     for i in range(1, n-1):
27         du[i] = mu * (u[i + 1] - 2 * u[i] + u[i - 1])
28     du[n - 1] = mu * (fi2() - 2 * u[n - 1] + u[n - 2])
29     return du
30
31 t = np.arange(t0, t1, h)
32 y = np.arange(0, 1, 1 / n)
33 solution = odeint(dudt, u0, t, args=(n, mu, t1))
34 t, y = np.meshgrid(t, y)
35 u = solution.T
36
37 lf_size = 13
38 fig = plt.figure()
39 ax = plt.axes(projection='3d')
40 ax.plot_wireframe(t, y, u, color='black')
41 ax.set_xlabel(r'$t$', fontsize=lf_size)
42 ax.set_ylabel(r'$y$', fontsize=lf_size)
43 ax.set_zlabel(r'$u$', fontsize=lf_size)
44 ax.view_init(azim=-35, elev=23)
45 ax.set_xlim(0, t1)
46 ax.set_ylim(0, 1)
47 ax.set_zlim(0, )
48 plt.savefig('fig4-2.png', dpi=300)
49 plt.show()
50 print('azim=', ax.azim)
51 print('elev=', ax.elev)

```

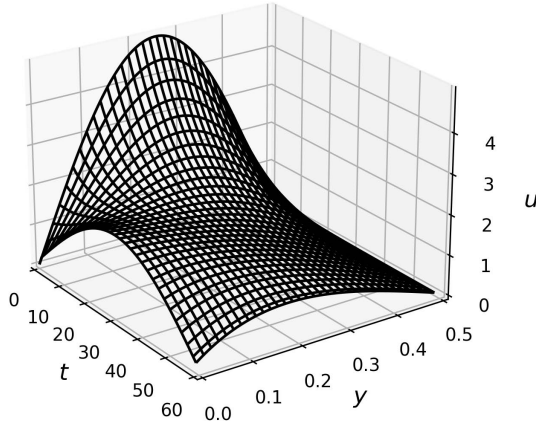


Рисунок 4.2 – Процес зміни температури у стіні із заліза

Таблиця 4.1 – Дослідження процесів зміни температури у стіні

Варіанти процесів	Початкова температура стіни	Температура від стіни ліворуч	Температура від стіни праворуч
Двостороннє охолодження	$\varphi(y) = 5$	$\varphi_1(t) = 0$	$\varphi_2(y) = 0$
Одностороннє примусове охолодження	$\varphi(y) = 5$	$\varphi_1(t) = 5$	$\varphi_2(y) = -3$
Одностороннє примусове нагрівання	$\varphi(y) = 0$	$\varphi_1(t) = 5$	$\varphi_2(y) = 0$
Двостороннє примусове нагрівання	$\varphi(y) = 0$	$\varphi_1(t) = 5$	$\varphi_2(y) = 5$
Нагрівання з охолодженням	$\varphi(y) = 2$	$\varphi_1(t) = 5$	$\varphi_2(y) = -1$

4.2.2 Індивідуальне завдання

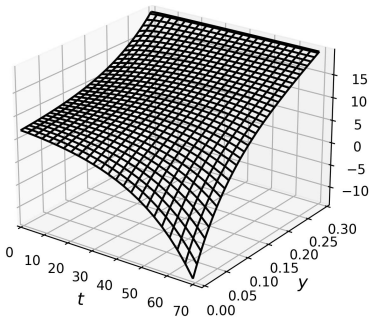
Розробіть програму моделювання зміни температури у стіні згідно з призначеним варіантом, для якого умови моделювання задані у табл. 4.2, а фізичні параметри – у табл. 4.3. Отримані результати можна порівняти з графіками на рис. 4.3 і рис. 4.4.

Таблиця 4.2 – Початкові та граничні умови

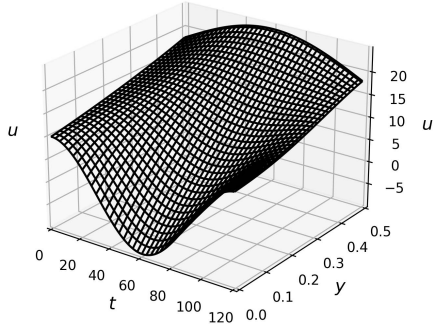
№	$\varphi(y)$	$\varphi_1(t)$	$\varphi_2(t)$
1	$6 + e^{8.8y}$	$8 - e^{0.045t}$	20
2	$10e^{1.1756y}$	$10 \cos(\pi t/60)$	$18 + 7 \cos(3\pi/2 + \pi t/120)$
3	$10e^{194.6y}$	$10(1 + \sin(\pi^2 t/62))$	$35(1 + \cos(\pi^2 t/62))$
4	$50(1 - \cos(7.1437\pi y))$	$10 \sin(\pi^2 t/50)$	$10(4 + \cos(\pi^3 t/120))$
5	$100y$	$1 - \cos(\pi t)$	$2e^{-1.5t}$
6	$20y^2$	$0.2(e^{0.03t} - 1)$	$20(1 - 0.1\sqrt{t})$
7	$100(0.1 + y^3)$	$10 + \sqrt{t}$	$52.19e^{-0.01t}$
8	$12\sqrt[3]{y}$	$e^{-0.25t} \sin(\pi t)$	$3.5e^{-2t}$
9	$-100y$	$15 \sin(\pi t/25)$	$-60(1 + \sin(\pi t/30))$
10	$36.6(e^{-693y})$	$36.6e^{-0.08t}$	$18.3e^{-2t}$

Таблиця 4.3 – Вихідні дані для моделювання

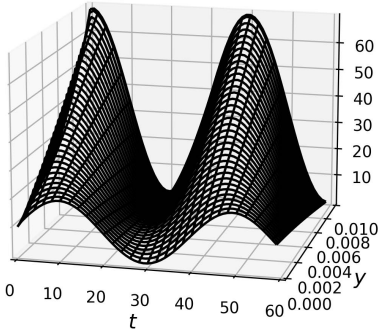
№	Матеріал	$a \cdot 10^{-6}, \text{ м}^2/\text{с}$	$L, \text{ м}$	T	h
1	Деревина	0.082	0.3	3 доби	3 год.
2	Цегла	0.52	0.5	5 діб	1 год.
3	Скло	0.34	0.01	1 год.	1 хв.
4	Мідь	111.0	0.07	1 год.	1 хв.
5	Алюміній	84.18	0.02	5 хв.	0.1 хв.
6	Повітря	19.0	1.0	3 год.	5 хв.
7	Вода	0.143	0.75	10 год.	20 хв.
8	Гума	0.13	0.025	5 хв.	0.1 хв.
9	Свинець	23.6	0.6	1 год.	1 хв.
10	Нейлон	0.09	0.001	5 хв.	0.1 хв.



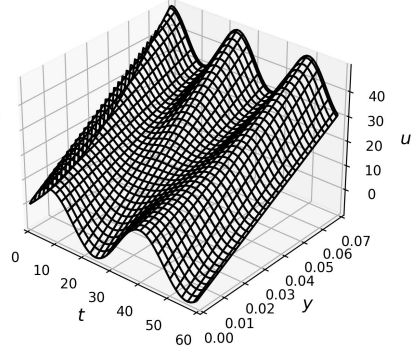
(а) варіант 1 (деревина)



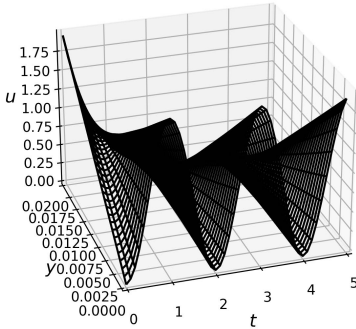
(b) варіант 2 (цегла)



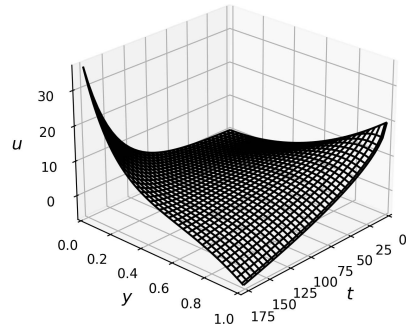
(с) варіант 3 (скло)



(d) варіант 4 (мідь)

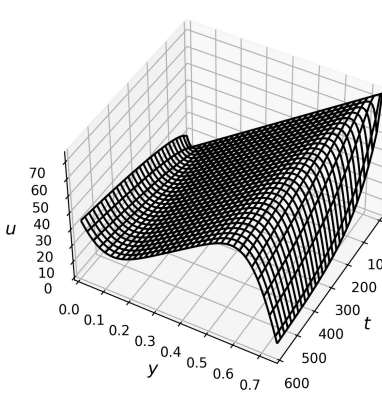


(е) варіант 5 (алюміній)

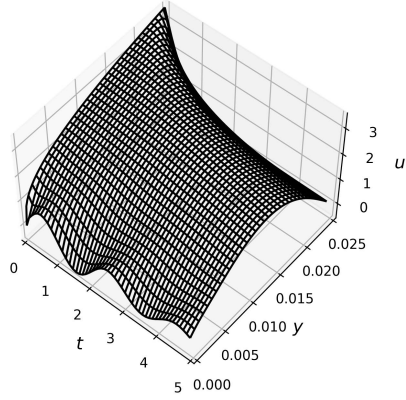


(f) варіант 6 (повітря)

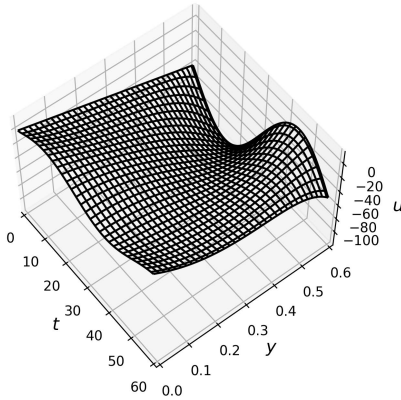
Рисунок 4.3 – Результати моделювання (варіанти 1–6)



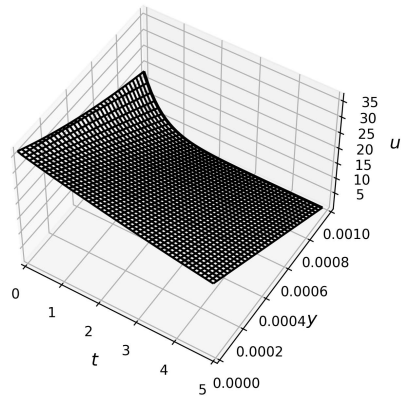
(a) варіант 7 (вода)



(b) варіант 8 (гума)



(c) варіант 9 (свинець)



(d) варіант 10 (нейлон)

Рисунок 4.4 – Результати моделювання (варіанти 7–10)

4.3 Контрольні питання

1. Наведіть формули апроксимації рівняння теплопровідності.
2. Чим визначається порядок системи звичайних диференціальних рівнянь для моделювання задачі теплопровідності за наведеною методикою?
3. Яку кількість шарів розбиття було використано для розв'я-

зання контрольної задачі з п. 4.2.1? Запишіть диференціальні рівняння для першої, другої та останньої точок системи звичайних диференціальних рівнянь для цієї задачі.

4. Назвіть основні джерела похибок, які впливають на результат моделювання.

5. Назвіть переваги й недоліки даної методики моделювання.

4.4 Зміст звіту з лабораторної роботи

1. Титульний лист, оформлений відповідно до вимог.
2. Мета роботи.
3. Результати моделювання процесів зміни температури в стіні із заліза з параметрами із табл. 4.1, у вигляді графіків.
4. Лістинг до п. 4.2.2 (індивідуальне завдання) та результати моделювання у вигляді графіку.
5. Відповіді на контрольні питання.
6. Висновки за результатами досліджень.

ЛАБОРАТОРНА РОБОТА 5

Моделювання систем масового обслуговування

Мета роботи — опанувати методи моделювання систем масового обслуговування, дослідити їх математичні моделі на прикладі найпростіших систем.

5.1 Теоретичні відомості

5.1.1 Поняття потоків у СМО

Системи масового обслуговування (СМО) – це системи, функціонування яких можна представити як процес обслуговування деякого потоку вимог, подій, заявок одного типу [8–10].

Потік вимог – послідовність вимог (подій, заявок) на обслуговування. *Вхідний потік* – потік вимог, які надходять до СМО. *Вихідний потік* – потік вимог, які виходять із СМО. *Джерело вимог* – першопричина виникнення вимог (подій, заявок) незалежно від їх фізичної природи.

Інтенсивність потоку – це середнє число вимог (подій, заявок), що виникають в одиницю часу.

При масовому надходженні вимог у СМО можуть виникнути черги.

Стаціонарний потік вимог – це потік, інтенсивність якого не залежить від часу.

Ординарний потік вимог – це потік, у якому вимоги виникають по одній, без одночасної появи.

Потік вимог без післядії – це потік, кількість вимог якого, за умови їх існування в інтервалах часу, що не перетинаються, не залежить одна від одної. Відсутність післядії означає, що вимоги надходять до системи незалежно одна від одної.

Найпростіший потік вимог – це потік, що одночасно стаціонарний, ординарний та без післядії.

Рекурентний потік вимог – це потік вимог, що є стаціонарним та ординарним, а інтервали часу між вимогами у такому потоці

є випадковими величинами з однаковим (довільним) розподілом.

Окремі випадки рекурентного потоку: *найпростіший потік*, коли інтервали часу між подіями розподілені за експоненціальним законом, та *нормальний потік*, коли інтервали часу між подіями розподілені за нормальним (Гаусовим) розподілом.

5.1.2 Короткі відомості про основні види СМО

СМО класифікують:

- за числом обслуговуваних підсистем: *одноканальні* та *багатоканальні*;
- за характером джерела вимог: *з кінцевою* та *нескінченною кількістю вимог*;
- залежно від відсутності або наявності можливості очікування для вимог: *з відмовами* та *з очікуванням (необмеженим, обмеженим)*.

У системі з відмовами вимога, що надходить у той час, коли всі канали СМО зайняті, отримує відмову, залишає систему та не приймає участі у подальшому процесі обслуговування. У системі з очікуванням така вимога стає в чергу та очікує поки не звільниться будь-яка підсистема.

У системах з обмеженим очікуванням можуть накладатися обмеження (на довжину черги, на час перебування в черзі, на загальний час перебування вимоги у системі тощо).

n-канальна СМО з відмовами

Ймовірність, що всі канали будуть вільні:

$$p_0 = \left(1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \dots + \frac{\rho^n}{n!} \right)^{-1}. \quad (5.1)$$

де $\rho = \lambda/\mu$ – зведена інтенсивність; n – кількість каналів у системі.

Після обчислення за формулою (5.1) ймовірності p_0 можна знайти ймовірності для всіх інших станів системи:

$$p_1 = \rho p_0, \quad p_2 = \frac{\rho^2}{2!} p_0, \quad \dots, \quad p_n = \frac{\rho^n}{n!} p_0. \quad (5.2)$$

Ці формули для ймовірностей станів n -канальної СМО із відмовами називаються *формулами Ерланга*.

Ймовірність того, що чергова заявка, що надійшла, отримає відмову в обслуговуванні дорівнює ймовірності зайнятості всіх каналів СМО, тобто p_n :

$$P_L = p_n = \frac{\rho^n}{n!} p_0. \quad (5.3)$$

Відносною пропускною здатністю СМО називається середня частка заявок, що будуть обслуговані:

$$Q = 1 - P_L = 1 - \frac{\rho^n}{n!} p_0. \quad (5.4)$$

Абсолютною пропускною здатністю СМО називається середня кількість заявок, що обслуговуються в одиницю часу:

$$A = \lambda Q. \quad (5.5)$$

Середнє число зайнятих каналів обчислюється за формулою:

$$k = \frac{A}{\mu}. \quad (5.6)$$

На рис. 5.1 наведена граф-схема станів для n -канальної СМО з відмовами.

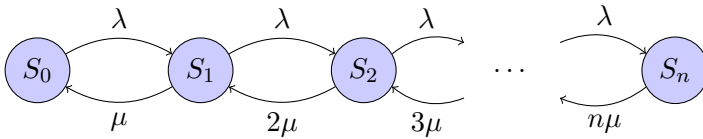


Рисунок 5.1 – Граф-схема станів n -канальної СМО з відмовами

Одноканальна СМО з необмеженою чергою

Ймовірність, що канал буде вільний:

$$p_0 = 1 - \rho. \quad (5.7)$$

Формули ймовірностей всіх інших станів системи:

$$p_1 = \rho(1 - \rho), \quad p_2 = \rho^2(1 - \rho), \quad \dots, \quad p_k = \rho^k(1 - \rho). \quad (5.8)$$

Середня кількість заявок та середній час перебування заявки у системі:

$$L_s = \frac{\rho}{1 - \rho}, \quad W_s = \frac{1}{\lambda} \frac{\rho}{1 - \rho}. \quad (5.9)$$

Довжина черги та середній час перебування заявки в черзі:

$$L_q = \frac{\rho^2}{1 - \rho}, \quad W_q = \frac{1}{\lambda} \frac{\rho^2}{1 - \rho}. \quad (5.10)$$

На рис. 5.2 наведена граф-схема станів для одноканальної СМО з необмеженою чергою.

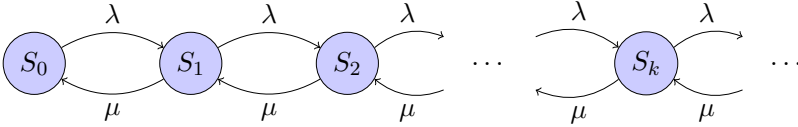


Рисунок 5.2 – Граф-схема одноканальної СМО з необмеженою чергою

n -канальна СМО з необмеженою чергою

Ймовірність, що всі канали будуть вільні:

$$p_0 = \left(1 + \frac{\rho}{1!} + \frac{\rho^2}{2!} + \dots + \frac{\rho^n}{n!} + \frac{\rho^{n+1}}{n!(n - \rho)} \right)^{-1}. \quad (5.11)$$

Ймовірності станів системи, коли немає черги:

$$p_1 = \rho p_0, \quad p_2 = \frac{\rho^2}{2!} p_0, \quad \dots, \quad p_n = \frac{\rho^n}{n!} p_0, \quad (5.12)$$

де p_1 – ймовірність того, що один канал зайнятий; p_2 – ймовірність того, що два канали зайняті; $\dots$, p_n – ймовірність того, що всі n каналів зайняті.

Ймовірність станів системи, коли всі канали зайняті і є черга:

$$p_{n+1} = \frac{\rho^{n+1}}{nn!} p_0, \quad p_{n+2} = \frac{\rho^{n+2}}{n^2 n!} p_0, \quad \dots \quad (5.13)$$

p_{n+1} – ймовірність того, що всі канали зайняті і одна заявка в черзі;
 p_{n+2} – ймовірність того, що всі канали зайняті і дві заявки в черзі,
і так далі.

Число заявок та час перебування заявки в черзі:

$$L_q = \frac{\rho^{n+1}}{nn!(1 - \rho/n)^2} p_0, \quad W_q = \frac{1}{\lambda} L_q; \quad (5.14)$$

Число заявок та час перебування заявки у системі:

$$L_s = L_q + \rho, \quad W_s = \frac{1}{\lambda} L_s. \quad (5.15)$$

На рис. 5.3 наведена граф-схема станів n -канальної СМО з необмеженою чергою.

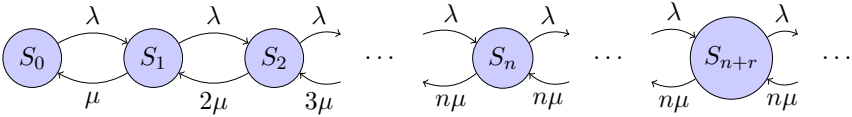


Рисунок 5.3 – Граф-схема станів n -канальної СМО з необмеженою чергою

Одноканальна СМО з обмеженим часом очікування

Ймовірність, що канал вільний:

$$p_0 = \left(1 + \rho + \rho \sum_{i=2}^{\infty} \frac{\lambda^{i-1}}{\prod_{j=1}^{i-1} (\mu + j\xi)} \right)^{-1}, \quad (5.16)$$

де λ – інтенсивність потоку заявок, μ – інтенсивність обслуговування каналу, ξ – інтенсивність очікування заявок.

Примітка: при використанні (5.16) достатньо вибрати максимальне значення індексу $i = 4$, оскільки члени ряду швидко зменшуються зі збільшенням i .

Ймовірність одночасного перебування r вимог в черзі, якщо $r \geq 2$:

$$p_r = \rho \frac{\lambda^{r-1}}{\prod_{j=1}^{r-1} (\mu + j\xi)} p_0. \quad (5.17)$$

5.1.3 Алгоритм роботи одноканальної СМО з обмеженим часом очікування

Розглянемо одноканальну систему, в яку на обслуговування надходять заявки (вимоги). Інтервал часу між двома послідовними заявками τ^E є випадковою величиною з заданим законом розподілу $f(\tau^E)$. Час зайнятості каналу (час обслуговування) τ^F також є випадковою величиною із законом розподілу $f(\tau^F)$.

Якщо наступна заявка, що надійшла, застає канал зайнятим, то вона очікує звільнення каналу протягом часового інтервалу, не більш ніж τ^W , який теж є випадковою величиною розподіленою за законом $f(\tau^W)$. Якщо за цей час канал звільняється, то він одразу починає обслуговувати заявку, інакше – заявка залишає систему (отримує відмову). Така система зветься *СМО з обмеженим часом очікування*.

В результаті моделювання такої системи необхідно отримати характеристики якості обслуговування:

- середню кількість обслугованих заявок;
- середню кількість заявок, які отримали відмову;
- середній час очікування;
- середній час обслуговування.

Алгоритм моделювання СМО та організацію роботи системи надано на рис. 5.4. Процес функціонування системи розглядається в інтервалі часу $[0, T]$. Зробимо деякі позначення моделі системи:

- t_i^E – момент часу надходження i -ї заявки. Заявки, що надійшли у момент часу $t_i^E \geq T$, у моделі не розглядаються;

– t^B – момент часу початку обслуговування. Якщо канал вільний, то обслуговування чергової заявки починається у момент її появи, тобто $t^B = t_i^E$, і канал вважається зайнятим;

– t^F – момент часу звільнення каналу (закінчення обслуговування). Якщо момент закінчення обслуговування $t^F > T$, вважаємо, що заявка отримала відмову;

– t^W – момент часу закінчення очікування поточної заявки. Якщо $t^W > t^F$, то заявка починає обслуговуватись, причому $t^B = t_i^E$ і система знову вважається зайнятою;

– заявка вважається обслугованою, якщо час закінчення обслуговування $t^F < T$;

– заявка отримала відмову, якщо $t^W < t^F$.

Нехай λ – інтенсивність потоку заявок, μ – інтенсивність обслуговування каналу, ξ – інтенсивність очікування заявок. Будемо вважати, що час між появою заявки, час обслуговування та час очікування для кожної заявки розподілені за експоненціальним законом:

$$f(\tau) = \lambda e^{-\lambda\tau}, \quad f(\tau^F) = \mu e^{-\mu\tau}, \quad f(\tau^W) = \xi e^{-\xi\tau}. \quad (5.18)$$

Обчислити випадкові значення, розподілені за експоненціальним законом, можна за формулою:

$$z = -\frac{1}{\theta} \ln(1 - x), \quad (5.19)$$

де x – випадкові *рівномірно* розподілені числа; θ – інтенсивність потоку.

Якщо діапазон розподілення $x \in (0, 1)$, то таке співвідношення еквівалентне виразу

$$z = -\frac{1}{\theta} \ln x. \quad (5.20)$$

Таким чином, випадкові моменти появи заявок, закінчення обслуговування та закінчення очікування можна обчислити за формулами:

$$t_{i+1}^E = t_i^E - \frac{1}{\lambda} \ln x, \quad t^F = t^B - \frac{1}{\mu} \ln x, \quad t^W = t_i^E - \frac{1}{\xi} \ln x. \quad (5.21)$$

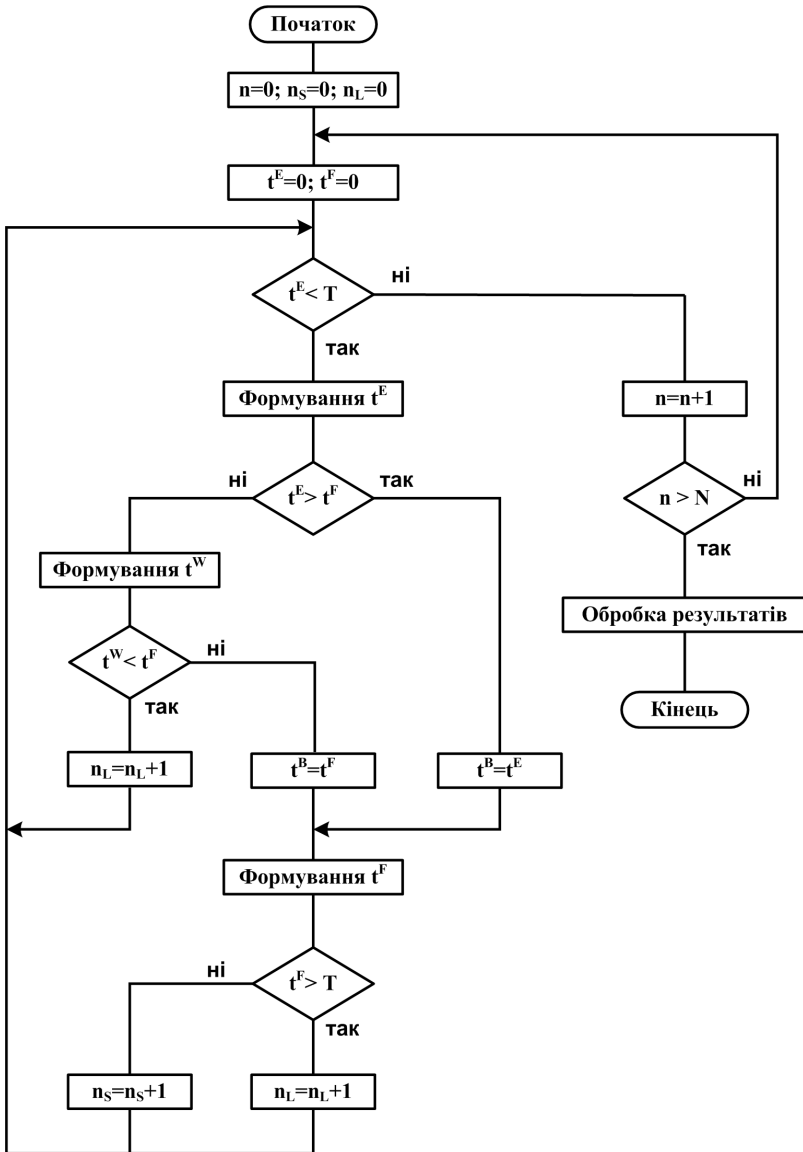


Рисунок 5.4 – Блок-схема алгоритму моделювання СМО

Рекомендації щодо використання алгоритму

За допомогою наведеного алгоритму можна приблизно моделювати поведінку СМО з відмовами та систему з необмеженим очікуванням.

При необмеженому збільшенні інтенсивності очікувань ξ , що означає те, що заявки інтенсивніше залишають систему, не дочекавшись обслуговування, система з обмеженим часом очікування за своїми характеристиками наближається до системи з відмовами.

При необмеженому зменшенні інтенсивності очікувань ця модель системи наближається до СМО з необмеженим очікуванням (з необмеженою чергою).

5.2 Практичні завдання

5.2.1 Дослідження одноканальної СМО

Розробіть програму моделювання одноканальної СМО відповідно до алгоритму представленому на рис. 5.4.

Для розробки власної програми моделювання можна використовувати елементи коду з ліст. 5.1.

Лістинг 5.1 Приклад реалізації одноканальної СМО

```

1  import random
2
3  def f(intensity, event):
4      return event + random.expovariate(intensity)
5
6  t = 1000 # тривалість однієї реалізації
7  la, mu, xi = 0.4, 0.6, 1.2 # лямбда, мю, кси
8  n = 100 # загальна кількість реалізацій
9  i = 0 # номер поточної реалізації
10 nE = 0 # кількість заявок, що надійшли
11 nS = 0 # кількість обслугованих заявок
12 nL = 0 # кількість заявок, що отримали відмову
13 while i < n:
14     te = 0.0 # час надходження заявки
15     tf = 0.0 # час звільнення каналу
16     while te < t:
17         te = f(la, te)
```

```

18         nE = nE + 1
19         if te > tf:
20             tb = te # час початку обслуговування
21             tf = f(mu, tb) # час кінця обслуговування
22             if tf > t:
23                 nL = nL + 1
24             else:
25                 nS = nS + 1
26         else:
27             tw = f(xi, te) # час очікування
28             if tw < tf:
29                 nL = nL + 1
30             else:
31                 tb = tf
32                 tf = f(mu, tb)
33                 if tf > t:
34                     nL = nL + 1
35                 else:
36                     nS = nS + 1
37         i = i + 1
38 # середня кількість заявок, що надійшли, обслуговані
39 # або отримали відмову
40 print('nE =', nE/n, 'nS =', nS/n, 'nL =', nL/n)

```

У програмі з ліст. 5.1, на відміну від формули (5.20), для генерації інтервалів часу, розподілених за експоненціальним законом, використовується бібліотечна функція `random.expovariate()` з аргументом, рівним відповідній інтенсивності λ , μ або ξ .

Для перевірки правильності реалізації можна скористатися результатами роботи цієї програми, згідно з якими:

- середнє значення заявок, що надійшли, приблизно дорівнює 400 зі стандартним відхиленням 18;
- середнє значення обслугованих заявок приблизно дорівнює 270 (68%) зі стандартним відхиленням 13;
- середня кількість заявок, що отримали відмову, приблизно дорівнює 130 (32%) зі стандартним відхиленням 16.

Параметри СМО для моделювання за індивідуальними варіантами наведено у табл. 5.1, де λ , μ і ξ – інтенсивності надходження заявки у систему, обслуговування заявки та очікування заявкою обслуговування $[1/c]$ відповідно.

Таблиця 5.1 – Інтенсивності потоків для одноканальної СМО

№	λ	μ	ξ	№	λ	μ	ξ
1	0.2	0.25	1.3	6	0.3	0.5	1.0
2	0.5	0.7	1.4	7	0.25	0.4	0.8
3	0.6	0.8	1.6	8	0.75	1.5	3.0
4	0.7	1.0	2.0	9	0.1	0.15	0.3
5	0.8	1.1	1.6	10	0.16	0.2	0.5

Загальне завдання для моделювання різних одноканальних СМО складається з трьох задач (*задача 1 – задача 3*).

Задача 1. Виконайте моделювання одноканальної СМО з відмовами:

- складіть граф-схему станів системи;
- обчисліть за допомогою формул, наведених у теоретичній частині: середню кількість заявок, що надійшли у систему; середню кількість обслугованих заявок; середню кількість заявок, що отримали відмову в обслуговуванні;
- виконайте імітаційне моделювання цієї системи за допомогою розробленої програми та рекомендацій щодо використання алгоритму. Для оцінки середніх статистичних характеристик слід виконувати не менше 100 реалізацій моделювання при тривалості однієї реалізації не менше 1000 с. (див. коментарі ліст. 5.1);
- порівняйте результати моделювання з результатами виконаних аналітичних розрахунків та зробіть висновки. Отримані результати по кількості заявок також надайте у вигляді відсотків (які були обслуговані; які отримали відмову).

Задача 2. Для одноканальної СМО з необмеженою чергою:

- складіть граф-схему станів системи;
- обчисліть: середню кількість заявок, що надійшли в систему; середню кількість заявок, які знаходяться у системі; середній час перебування заявки у системі; середню довжину черги заявок в системі; середній час перебування заявки в черзі;
- виконайте імітаційне моделювання цієї системи за допомо-

гою розробленої програми та рекомендацій щодо використання алгоритму, як ви це робили у *задачі 1*, не менше 100 реалізацій моделювання при тривалості однієї реалізації не менше 1000 с (див. ліст. 5.1);

– порівняйте результати моделювання з результатами виконаних аналітичних розрахунків та зробіть висновки. Отримані результати по кількості заявок також надайте у вигляді відсотків.

Задача 3. Для одноканальної СМО з обмеженим часом очікування за допомогою (5.16) та (5.17) обчисліть ймовірність того, що канал буде вільний, ймовірність зайнятості каналу та ймовірність одночасного перебування r заявок в черзі (для $r = 2$ і $r = 3$).

5.2.2 Дослідження багатоканальної СМО

За допомогою формул, наведених у теоретичній частині, виконайте необхідні розрахунки та розв'яжіть задачі для багатоканальних СМО за індивідуальними варіантами, кожен з яких складається з двох задач: *задача 4* – n -канальна СМО з відмовами, *задача 5* – n -канальна СМО з необмеженою чергою.

Для кожної задачі складіть граф-схему станів.

Варіант 1.

Задача 4. Для 3-х канальної СМО з відмовами визначте ймовірність обслуговування, якщо у систему надходять у середньому 18 заявок за хвилину та кожна заявка обслуговується 8 с.

Задача 5. Для 2-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи заявки в черзі буде знаходитись 2 заявки, якщо у систему надходять у середньому 12 заявок за хвилину і кожна заявка обслуговується 6 с.

Варіант 2.

Задача 4. Для 2-х канальної СМО з відмовами визначте ймовірність відмови в обслуговуванні, якщо у систему надходять у середньому 8 заявок за хвилину та кожна заявка обслуговується 10 с.

Задача 5. Для 3-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи заявки система буде зайнята, якщо у систему надходять у середньому 15 заявок за хвилину і кожна заявка обслуговується 6 с.

Варіант 3.

Задача 4. Для 4-х канальної СМО з відмовами визначте ймовірність того, що буде зайнято два канали, якщо у систему надходять у середньому 20 заявок за хвилину та кожна заявка обслуговується 10 с.

Задача 5. Для 2-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи заявки в черзі буде знаходитись 1 заявка, якщо у систему надходять у середньому 8 заявок за хвилину, а кожна заявка обслуговується 10 с.

Варіант 4.

Задача 4. Для 2-х канальної СМО з відмовами визначте ймовірність обслуговування, якщо у систему надходять у середньому 3 заявки за хвилину, а кожна заявка обслуговується 30 с.

Задача 5. Для 4-х канальної СМО з необмеженою чергою визначте середню кількість заявок та середній час перебування заявки у системі, якщо у систему надходять у середньому 4 заявки за хвилину, а кожна заявка обслуговується 40 с.

Варіант 5.

Задача 4. Для 3-х канальної СМО з відмовами визначте ймовірність того, що будуть зайняті 2 канали, якщо інтенсивність вхідного потоку – 1 заявка за 1 с, а кожна заявка обслуговується в середньому 2,5 с.

Задача 5. Для 3-х канальної СМО з необмеженою чергою визначте середню кількість заявок та середній час перебування заявки у системі, якщо у систему надходять у середньому 10 заявок за хвилину, а кожна заявка обслуговується 15 с.

Варіант 6.

Задача 4. Для 4-х канальної СМО з відмовами визначте ймовірність того, що система буде вільною, якщо інтенсивність вхідного потоку – 3 заявки за 2 с, а кожна заявка обслуговується в середньому 2 с.

Задача 5. Для 2-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи заявки у системі в черзі буде знаходитись 3 заявки, якщо у систему надходять у середньому 6 заявок за хвилину, а кожна заявка обслуговується 10 с.

Варіант 7.

Задача 4. Для 2-х канальної СМО з відмовами визначте ймовірність відмови в обслуговуванні, якщо інтенсивність вхідного потоку – 1 заявка за 12 с, а кожна заявка обслуговується в середньому 8 с.

Задача 5. Для 4-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи заявки в системі буде зайнято три канали, якщо у систему надходять у середньому 18 заявок за хвилину, а кожна заявка обслуговується 10 с.

Варіант 8.

Задача 4. Для 2-х канальної СМО з відмовами визначте ймовірність відмови в обслуговуванні, якщо у систему надходять у середньому 10 заявок за хвилину, а кожна заявка обслуговується 8 с.

Задача 5. Для 3-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи поточної заявки система буде зайнята, якщо у систему надходять у середньому 10 заявок за хвилину, а кожна заявка обслуговується 12 с.

Варіант 9.

Задача 4. Для 4-х канальної СМО з відмовами визначте ймовірність того, що система буде вільною, якщо у систему надходять у середньому 7 заявок за хвилину, а кожна заявка обслуговується 30 с.

Задача 5. Для 2-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи поточної заявки в черзі буде знаходитися 2 заявки, якщо у систему надходять у середньому 10 заявок за хвилину, а кожна заявка обслуговується 10 с.

Варіант 10.

Задача 4. Для 3-х канальної СМО з відмовами визначте ймовірність того, що в системі буде зайнятий один канал, якщо у систему надходять у середньому 8 заявок за хвилину, а кожна заявка обслуговується 20 с.

Задача 5. Для 4-х канальної СМО з необмеженою чергою визначте ймовірність того, що під час появи поточної заявки в системі буде зайнято 2 канали, якщо у систему надходять у середньому 16 заявок за хвилину, а кожна заявка обслуговується 12 с.

5.3 Контрольні питання

1. Що називають потоком вимог у теорії СМО?
2. Як практично оцінити значення інтенсивності потоку вимог?
3. Що таке найпростіший потік вимог? Як на практиці оцінити, чи є потік найпростішим? Як розподілені інтервали часу між вимогами у найпростішому потоці?
4. Складіть приклад розміченого графа та системи рівнянь Колмогорова для системи з трьома можливими станами, причому, система може переходити з кожного стану в будь-який з двох інших станів.
5. Поясніть поняття фінальної ймовірності стану СМО. У якому випадку вони існують? Як це визначити для конкретної СМО?
6. Чи є ймовірності у рівняннях Колмогорова фінальними?
7. Чи існують фінальні ймовірності станів одноканальної СМО з необмеженим часом очікування? Якщо так, то за якої умови?
8. Чи існують фінальні ймовірності станів n -канальної СМО з необмеженим часом очікування? Якщо так, то за якої умови?
9. Що означає твердження, що у черзі перебуває не більше трьох вимог?
10. Які саме канали триканальної СМО зайняті обслуговуванням, якщо на даний момент зайняті лише два канали?

5.4 Зміст звіту з лабораторної роботи

1. Титульний лист.
2. Мета роботи.
3. Лістинг програми, результати аналітичних розрахунків та висновки, які виконані за п. 5.2.1.
4. Результати розв'язання задач із п. 5.2.2.
5. Аргументовані відповіді на контрольні питання.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

- [1] Томашевський В. М. Моделювання систем / В. М. Томашевський. – Київ : Видавнича група BHV, 2005. – 352 с.
- [2] Хусаїнов Д. Я. Введення в моделювання динамічних систем : навч. посіб. / Д. Я. Хусаїнов, І. І. Харченко, А. В. Шатирко – Київ : КНУ ім. Т. Шевченка, 2010. – 130 с.
- [3] De Silva C. W. Modeling of Dynamic Systems with Engineering Applications. 2nd ed. / C. W. De Silva – Boca Raton : CRC Press, 2022. – 420 p.
- [4] Васильєв О. М. Програмування в Python. Теорія і практика / О. М. Васильєв – Київ : Ліра-К, 2023. – 462 с.
- [5] Зеленьак О. Програмування мовою Python. Алгоритмічні структури і стратегії. Теорія та практика / О. Зеленьак – Київ : Ліра-К, 2025. – 338 с.
- [6] Python Software Foundation. URL: <https://www.python.org/>.
- [7] SciPy API: scipy.integrate. URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html>.
- [8] Литвинов А. Л. Теорія систем масового обслуговування : навч. посіб. / А. Л. Литвинов – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 141 с.
- [9] Математичне моделювання систем і процесів : навч. посіб. / [П. М. Павленко, С. Ф. Філоненко, О. М. Чередніков та ін.] – Київ : НАУ, 2017. – 392 с.
- [10] Chan W. C. An Elementary Introduction to Queueing Systems / W. C. Chan. – Singapore : World Scientific, 2014. – 116 p.