МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВОДНОГО ГОСПОДАРСТВА ТА ПРИРОДОКОРИСТУВАННЯ

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії

„До захисту допущена”
Зав. кафедри комп’ютерних наук та прикладної математики
д.т.н., професор Турбал Ю.В.

___ _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ЧИСЛОВОГО РОЗВ’ЯЗАННЯ ЗАДАЧІ МАСОПЕРЕНЕСЕННЯ ПРИ НЕЛІНІЙНІЙ НЕІЗОТЕРМІЧНІЙ ФІЛЬТРАЦІЇ

Виконав: Басюк Максим Васильович _____
(прізвище, ім’я, по батькові) (підпис)

студент групи ІПЗ-41 інт

Керівник: к.т.н, доцент кафедри Мічута О.Р. _____
(науковий ступінь, вчене звання, посада, прізвище та ініціали) (підпис)

Рівне – 2025

Зміст

РЕФЕРАТ

Кваліфікаційна робота: стор., ілюстрацій, використаних джерел, 1 додаток.

Мета дослідження — розробити алгоритмічні підходи до чисельного моделювання масоперенесення в пористому середовищі з урахуванням температурної залежності та нелінійності закону фільтрації.

Об'єктом дослідження виступає фізичний процес нелінійної фільтрації рідини в неоднорідному пористому середовищі, що включає тонкі включення зі змінними характеристиками.

Предметом дослідження є математичні та чисельні методи моделювання масоперенесення в задачах фільтраційного типу, де враховано залежність проникності від температури та інші нелінійні ефекти.

Методи дослідження: чисельне інтегрування крайових задач на основі методу скінченних елементів (МСЕ), реалізація розв'язку за допомогою алгоритмів на мові програмування Python із використанням спеціалізованих бібліотек для обробки даних.

У результаті виконаної роботи було побудовано математичну модель, яка описує процес масоперенесення під впливом температурних коливань і нелінійних змін фізичних параметрів середовища. Розроблено програмний інструмент, що дозволяє досліджувати поведінку масивів ґрунту з тонкими включеннями при змінних умовах. Проведено серію чисельних експериментів, що ілюструють значний вплив температури на характер розподілу концентрації речовини.

Показано, що нехтування термічними ефектами у фільтраційних задачах призводить до відчутних похибок моделювання.

Ключові слова: МАСОПЕРЕНЕСЕННЯ, НЕІЗОТЕРМІЧНИЙ РЕЖИМ, НЕЛІНІЙНА ФІЛЬТРАЦІЯ, МЕТОД СКІНЧЕННИХ ЕЛЕМЕНТІВ, НЕІЗОТЕРМІЧНА ПРОНИКНІСТЬ, АЛГОРИТМІЧНА РЕАЛІЗАЦІЯ, ЧИСЕЛЬНЕ МОДЕЛЮВАННЯ, ТОНКЕ ВКЛЮЧЕННЯ, PYTHON.

ВСТУП

У сучасних умовах стрімкого розвитку промисловості та зростаючого антропогенного навантаження на навколишнє середовище виникає нагальна потреба у дослідженні процесів, що відбуваються у підземних та ґрунтових системах. Зокрема, важливим є моделювання масоперенесення у пористих середовищах, які слугують природними фільтрами, що затримують забруднювачі, важкі метали або нафтопродукти. Для підвищення надійності захисних конструкцій, таких як техногенні екрани або геобар'єри, необхідно глибоко розуміти механізми переносу маси з урахуванням впливу температурних змін і неоднорідної структури ґрунту.

У численних практичних ситуаціях, зокрема при захороненні побутових або промислових відходів, у підземному середовищі виникають значні температурні градієнти. Ці температурні збурення істотно впливають на процеси фільтрації та масоперенесення, змінюючи властивості середовища та флюїду. У зв'язку з цим виникає необхідність створення математичних моделей, які враховують як нелінійність законів переносу, так і термічні впливи.

Особливу складність становлять ситуації, де у ґрунтовій товщі присутні тонкі включення — шари з іншими фізичними характеристиками, які можуть мати низьку проникність або бути напівпроникними. На межах таких включень виникають стрибки параметрів, що потребують спеціального математичного підходу до формулювання умов спряження.

Метою цієї роботи є розробка математичної моделі та алгоритмічної реалізації чисельного розв'язання задачі масоперенесення з урахуванням термічної неоднорідності, нелінійності фільтрації та присутності тонких включень. Основний акцент зроблено на побудову чисельного алгоритму з використанням методу скінченних елементів, адаптованого до задач із складною геометрією та нелінійною фізикою процесу.

Обчислювальна частина реалізована з використанням мови Python, яка має зручний інструментарій для наукового програмування завдяки бібліотекам NumPy, SciPy і Matplotlib. Вибір Python обумовлений також його доступністю, простотою синтаксису та широким застосуванням у сфері технічного моделювання.

Структура роботи включає аналіз теоретичних аспектів задач фільтрації та масоперенесення, формулювання моделі з умовами спряження, чисельну реалізацію задачі та аналіз результатів моделювання. Дослідження має міждисциплінарний характер і може бути застосоване у галузях екологічної інженерії, гідрогеології, будівництва захисних споруд, а також у наукових проєктах, пов’язаних із моніторингом підземного середовища.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ НЕЛІНІЙНОГО МАСОПЕРЕНЕСЕННЯ ПРИ НЕІЗОТЕРМІЧНІЙ ФІЛЬТРАЦІЇ ТА ПІДХОДИ ДО ЧИСЕЛЬНОГО МОДЕЛЮВАННЯ

У цьому розділі розглядаються фундаментальні теоретичні аспекти, що лежать в основі процесів масоперенесення в пористих середовищах за умов нелінійної та неізотермічної фільтрації. Проаналізовано ключові фізичні закони, що описують рух флюїдів, вплив температурних ефектів на параметри системи, а також математичні підходи до моделювання неоднорідних структур, зокрема з тонкими включеннями. Окрему увагу приділено обґрунтуванню вибору методу скінченних елементів (МСЕ) як основного інструменту для чисельного розв’язання поставленої задачі. Ці теоретичні положення є основою для розробки математичної моделі та її подальшої алгоритмічної реалізації

1.1. Фізичні процеси масоперенесення у пористих середовищах

Процес масоперенесення у пористих середовищах пов'язаний з рухом рідин або газів через структуру, що містить порожнини та капіляри. Такий рух виникає внаслідок різниці тиску, концентрації чи температури.

Закон Дарсі та його значення

Основою для опису фільтраційних процесів є **закон Дарсі**, сформульований французьким інженером Анрі Дарсі в 1856 році для опису руху води через пісок. Цей закон встановлює лінійну пропорційну залежність між швидкістю фільтрації та градієнтом гідравлічного напору. У класичному вигляді він записується так

$$\vartheta = -\text{К} * \nabla H, \tag{1.1}$$

де:

ϑ – швидкість фільтрації (м/с);

К – коефіцієнт фільтрації, що характеризує проникність пористого середовища та залежить від властивостей рідини (наприклад, в'язкості);

∇H – градієнт напору, що є рушійною силою процесу.

Закон Дарсі (1.1) є фундаментальним у гідрогеології, нафтовидобуванні, ґрунтознавстві та інших галузях, де моделюється рух рідин.

Рівняння конвекції-дифузії

Процес масоперенесення не обмежується лише конвективним рухом, зумовленим фільтрацією. Важливу роль відіграють також дифузійні процеси, пов'язані з хаотичним рухом молекул речовини. Для комплексного опису цих явищ використовується конвективно-дифузійне рівняння:

$$\frac{\partial(\varphi C)}{\partial t} + \nabla * (\vartheta C) = \nabla * (D \nabla C),$$

де:

φ – пористість середовища (безрозмірна величина);

C – концентрація розчиненої речовина (кг/м^3);

ϑ – швидкість фільтрації, яка визначається законом Дарсі;

D – коефіцієнт дифузії речовини в пористому середовищі (м²/с);

t – час (с).

Ці величини, як правило, змінні і залежать від фізичних характеристик середовища, його структури, температури, а також властивостей речовини, що переноситься.

1.2. Нелінійність та температурні ефекти в задачах фільтрації

У реальних умовах класичний закон Дарсі часто потребує модифікацій, оскільки фільтраційні процеси можуть демонструвати **нелінійну** поведінку. **Нелінійність** виникає, коли фізичні параметри середовища або рідини змінюються під впливом зовнішніх факторів.

<u>Причини нелінійності:</u>

Основними причинами нелінійної залежності між швидкістю фільтрації та градієнтом напору є:

1. Зміна властивостей середовища**:** проникність пор може змінюватися внаслідок стиснення або розширення під дією тиску.
2. Температурні ефекти**:** в умовах неізотермічної фільтрації температура суттєво впливає на фізичні властивості рідини, зокрема на її в'язкість і густину, що безпосередньо змінює швидкість руху. Наприклад, підвищення температури зазвичай зменшує в'язкість рідини, що збільшує її рухливість.
3. Хімічна взаємодія**:** розчинені речовини можуть осідати в порах, змінюючи їхню структуру та ефективну проникність.
4. Пороговий градієнт**:** у деяких середовищах, наприклад, у глинистих ґрунтах, фільтрація починається лише після того, як градієнт напору перевищить певне критичне (порогове) значення.

<u>Вплив температури на параметри фільтрації</u>

У задач неізотермічної фільтрації врахування температурних градієнтів є критично важливим. Температура впливає не лише на в'язкість рідини, але й на гідравлічну провідність та пороговий градієнт середовища. Ці залежності часто описуються експоненційними функціями.

1. Температурозалежний коефіцієнт фільтрації (K_T): зі зростанням температури гідравлічна провідність збільшується, що полегшує рух рідини. Цю залежність можна описати формулою:

$$K(T) = K_0 e^{\alpha T}$$

де:

K_0 — початковий коефіцієнт фільтрації при базовій температурі;

α — емпіричний коефіцієнт температурної залежності;

T — температура (°C).

2. Температурозалежний пороговий градієнт ($\nabla H_{\text{кр}}(T)$): зі зростанням температури пороговий градієнт, навпаки, знижується, що полегшує початок фільтраційного потоку. Ця залежність може бути виражена як:

$$\nabla H_{\text{кр}}(T) = \nabla H_o e^{-\beta T} \quad (1.4)$$

де:

∇H_o — пороговий градієнт при базовій температурі;

β — коефіцієнт, що характеризує чутливість порогового градієнта до зміни температури.

Таким чином, для точного опису реальних фізичних явищ необхідно використовувати нелінійні моделі, які враховують взаємозв'язок між рухом рідини та теплопередачею.

1.3. Умови спряження на тонких включеннях

У багатьох геологічних та інженерних системах наявні неоднорідності, такі як багатошарові структури або тонкі прошарки з відмінними фізичними властивостями (наприклад, геотехнічні бар'єри, шари ущільненої глини). Моделювання таких систем вимагає спеціального підходу до опису процесів на межах контакту між різними матеріалами.

Умови спряження – це математичні співвідношення, що забезпечують узгодженість фізичних параметрів (напору, температури, концентрації) на межах контакту, враховуючи можливі стрибки цих величин.

Для тонкого включення, що має значний опір потоку, стрибки напору $[H]$, концентраціх порової рідини $[C]$ та температури $[T]$ на його межах можна описати наступним чином:

$$[H] = R_H * \nabla H * n$$

$$[T] = R_T * \nabla T * n$$

$$[C] = R_C * \nabla C * n$$

де:

$[f]$ — позначає стрибок величини f на межі (тобто різницю значень по обидва боки від включення);

R_H, R_T, R_C — коефіцієнти спряження (опору), що характеризують гідравлічні та теплові властивості включення;

n — нормальний вектор до межі включення.

Ці умови є ключовими для коректного моделювання систем із геобар'єрами або іншими тонкими прошарками, оскільки дозволяють врахувати їх фільтраційні та ізоляційні властивості.

1.4. Метод скінченних елементів для розв’язання задач масоперенесення

Математичні моделі, що описують нелінійне масоперенесення при неізотермічній фільтрації, являють собою складні системи диференціальних рівнянь у частинних похідних. Отримати аналітичні розв'язки таких систем можливо лише для дуже спрощених випадків, які не відображають реальних умов. Тому для їх розв'язання застосовуються чисельні методи.

Метод скінченних елементів (МСЕ) є одним із найпотужніших та універсальних чисельних інструментів для розв'язання крайових задач. Його суть полягає в розбитті складної розрахункової області на скінченну кількість простих підобластей (елементів), у межах яких шукані величини апроксимуються за допомогою поліноміальних функцій.

Основні етапи застосування МСЕ включають:

1. **Дискретизація області.** Розрахункова область розбивається на скінченні елементи (наприклад, відрізки, трикутники, тетраедри).

2. **Вибір базисних функцій.** У межах кожного елемента шукана невідома функція (напір, температура, концентрація) наближається лінійною комбінацією простих функцій (базисних).
3. **Формулювання слабкої (варіаційної) постановки.** Диференціальне рівняння перетворюється на інтегральне, що дозволяє послабити вимоги до гладкості шуканого розв'язку та природно врахувати граничні умови.
4. **Побудова системи алгебраїчних рівнянь.** На основі варіаційної постановки для всієї області формується глобальна система лінійних або нелінійних алгебраїчних рівнянь.
5. **Розв'язання системи.** Отримана система розв'язується чисельними методами лінійної алгебри для знаходження значень шуканих величин у вузлах сітки.

Завдяки своїй універсальності та адаптивності, МСЕ ефективно застосовується для моделювання задач зі складною геометрією, неоднорідними матеріалами та нелінійними властивостями, що робить його ідеальним вибором для дослідження поставленої проблеми.

РОЗДІЛ 2
МАТЕМАТИЧНА МОДЕЛЬ ЗАДАЧІ МАСОПЕРЕНЕСЕННЯ ПРИ НЕЛІНІЙНІЙ НЕІЗОТЕРМІЧНІЙ ФІЛЬТРАЦІЇ З УМОВАМИ СПРЯЖЕННЯ

У цьому розділі на основі теоретичних положень, викладених у попередньому розділі, формулюється комплексна математична модель, що описує взаємопов'язані процеси переносу рідини, тепла та розчинених речовин у неоднорідному пористому середовищі. Модель враховує нелінійність закону фільтрації, температурну залежність фізичних параметрів та наявність тонкого напівпроникного включення, на якому виконуються спеціальні умови спряження. Сформульована модель є основою для подальшої розробки чисельного алгоритму.

2.1. Постановка фізичної задачі

Розглядається задача моделювання процесів переносу в неоднорідному, повністю насиченому пористому середовищі, що має практичне значення для багатьох інженерних та екологічних застосувань. Прикладом такої системи може слугувати ґрунтовий масив під полігоном для захоронення твердих побутових відходів. Внаслідок біохімічних процесів у тілі полігону відбувається підвищення температури , що впливає на фільтраційні властивості ґрунту та може створювати нерівномірний розподіл тиску.

Об'єктом моделювання є обмежена розрахункова область Ω, що складається з двох підобластей Ω_1 та Ω_2, які представляють шари ґрунту. Ці підобласті розділені тонким включенням Ω товщиною d, яке імітує геотехнічний бар'єр (геобар'єр). Геобар'єри використовуються для захисту підземних вод від проникнення забруднюючих речовин. Схематичне зображення розрахункової області наведено на рис. 2.1.

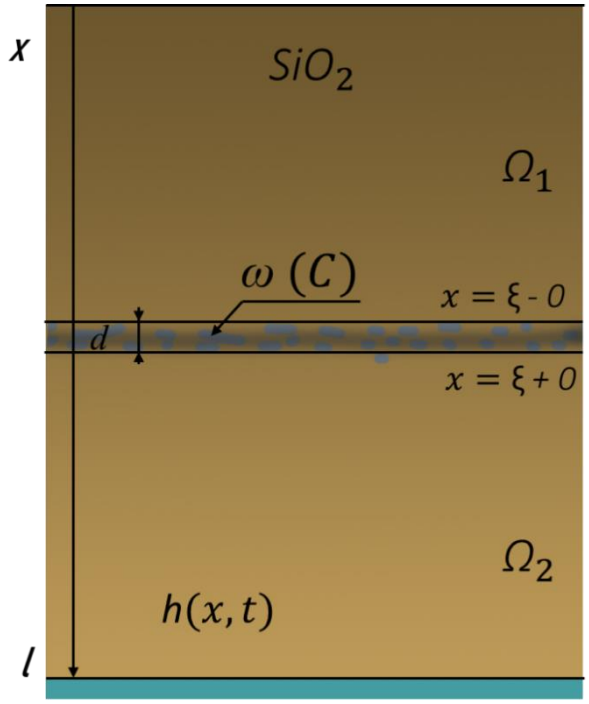

Рис. 2.1. Шар ґрунту товщиною l з тонким включенням ω товщиною d

Фізична складність системи полягає у значному контрасті властивостей між основним середовищем і включенням, що призводить до появи стрибків у функціях температури, напору та концентрації. Це вимагає введення відповідних умов спряження.

Основна мета — розробити математичну модель для прогнозування та аналізу розподілу напорів $h(x,t)$, температури $T(x,t)$ та концентрації розчиненої речовини $C(x,t)$ у ґрунтовій товщі з урахуванням нелінійних та неізотермічних ефектів.

2.2. Основні рівняння моделі

Математична модель, що описує поставлену задачу, є системою пов'язаних диференціальних рівнянь у частинних похідних, які виражають закони збереження маси, енергії та домішок.

1. Рівняння фільтраційної консолідації

Це рівняння описує зміну напору рідини з урахуванням стисливості ґрунту, а також впливу температури (термоосмос). Воно має вигляд:

$$\frac{\partial h}{\partial t} = \frac{1+e}{\gamma a}\frac{\partial}{\partial x}\left(k^*(\mathrm{C}, h, T)\frac{\partial h}{\partial x} - \nu\frac{\partial \mathrm{C}}{\partial x} - \mu(h)\frac{\partial T}{\partial x}\right), x \in \Omega_1 \cup \Omega_2, t \in (0, \underline{t}], t > 0 \quad (2.1)$$

де:

- $h(t, x)$ — напір рідини в поровій рідині;
- $T(x, t)$ — температура;
- $k^*(\mathrm{C}, h, T)$ — коефіцієнт фільтрації в нелінійному законі Дарсі, що залежить від напорів, температури та концентрації;
- $\mu(h)$ — термоосмотичний коефіцієнт;
- e — коефіцієнт пористості ґрунту;
- γ — питома вага порової рідини;
- α — коефіцієнт стискуваності ґрунту.

Початкова умова для напору (при $t = 0$):

$$h(x, 0) = h_0(x), \qquad x \in \underline{\Omega_1} \cup \underline{\Omega_2}$$

Граничні умови для напору (на межах $x = 0$ та $x = 1$):

$$h(x, t)|_{x=0} = \underline{h_0}(t),\ t \geq 0;$$

$$u(x, t)|_{x=l} = \left(-k^*(\mathrm{C}, h, T)\frac{\partial h}{\partial x} + \nu\frac{\partial \mathrm{C}}{\partial x} + \mu(h)\frac{\partial T}{\partial x}\right)|_{x=l} = 0,\ t \geq 0;$$

2. Рівняння теплоперенесення та його умови

Зміна температури внаслідок теплопровідності описується рівнянням:

$$c_s(h)\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\lambda(h)\frac{\partial T}{\partial t}\right) - \rho_\omega c_\omega u(h, T)\frac{\partial T}{\partial t},\ \ x \in \Omega_1 \cup \Omega_2, t > 0;$$

де:

- $C_s(h)$ — об'ємна теплоємність ґрунту;
- $\lambda(h)$ — коефіцієнт теплопровідності, що залежить від напорів.

Граничні умови для температури:

- На лівій межі

$$T(x, t)|_{x=0} = T_0(t), t \geq 0$$

- На правій межі:

$$q_T(x,t)|_{x=l} = -\lambda(h)\frac{\partial T}{\partial t}|_{x=l} = 0, \qquad t \geq 0$$

- Початкова умова для температури:

$$T(x,0) = T_0(x), \qquad x \in \Omega_1 \cup \Omega_2$$

3. Рівняння масоперенесення (конвекції-дифузії)

$$n\frac{\partial \mathrm{C}}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial \mathrm{C}}{\partial x}\right) + \frac{\partial}{\partial x}\left(D_T\frac{\partial T}{\partial x}\right) - u(\mathrm{C},h,T)\frac{\partial \mathrm{C}}{\partial x}, \qquad x \in \Omega_1 \cup \Omega_2$$

Граничні умови для температури:

- На лівій межі

$$C(x,t)|_{x=0} = \underline{C_0}(t),$$

- На правій межі

$$q_C(x,t)|_{x=l} = -D\frac{\partial C}{\partial x}|_{x=l} = 0$$

- Початкова умова масоперенесення

$$C(x,0) = C_0(x)$$

4. Умови спряження на тонкому включенні:

- Для тиску:

$$u^{\pm}|_{x=\xi} = -\frac{[h]}{\int_0^d \frac{dx}{k_\omega^*(T,I_\omega)}} + \frac{[T]}{\int_0^d \frac{dx}{\mu_\omega(h)}} + \frac{[C]}{\int_0^d \frac{dx}{\nu_\omega}}$$

- Для температури:

$$q_T^{\pm}|_{x=\xi} = -\frac{[T]}{\int_0^d \frac{dx}{\lambda_\omega(h)}}$$

- Для концентрації:

$$q_C^{\pm}|_{x=\xi} = -\frac{[C]}{\int_0^d \frac{dx}{\nu_\omega}} - \frac{[T]}{\int_0^d \frac{dx}{D_{T\omega}}}.$$

РОЗДІЛ 3
АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ ЧИСЕЛЬНОГО РОЗВ'ЯЗАННЯ

Цей розділ присвячено розробці алгоритмічного забезпечення для чисельного розв'язання складної математичної моделі, сформульованої у попередньому розділі. Детально обґрунтовується вибір чисельних методів, описуються етапи просторової та часової дискретизації, а також пропонується підхід до врахування нелінійностей та умов спряження. Метою є побудова стійкого та ефективного обчислювального алгоритму, здатного адекватно моделювати процеси нелінійного неізотермічного масоперенесення.

3.1. Обґрунтування чисельного підходу

Побудована в попередньому розділі математична модель описує нестаціонарні, нелінійні процеси масоперенесення з урахуванням температурних ефектів, порогових градієнтів та неоднорідностей у вигляді тонких включень. Складність моделі зумовлена кількома факторами:

- **Сильна зв'язність.** Рівняння для напору, температури та концентрації є взаємопов'язаними. Наприклад, коефіцієнт фільтрації залежить від температури, а швидкість фільтрації впливає на конвективний перенос маси та тепла.
- **Нелінійність.** Коефіцієнти в рівняннях залежать від самого розв'язку (напору, температури), що робить систему нелінійною.
- **Розривні коефіцієнти та умови спряження.** Наявність тонкого включення вносить розриви в параметри та вимагає спеціального врахування умов спряження.

Через ці особливості аналітичне розв’язання такої системи рівнянь є практично неможливим. Тому використання чисельного підходу є єдиним ефективним способом отримати апроксимовані рішення для практичних задач.

До чисельної схеми висуваються високі вимоги, серед яких: точність обчислень, стійкість при інтегруванні в часі, здатність враховувати складні граничні умови, а також можливість моделювання умов спряження між різнорідними підобластями.

Для реалізації цих вимог доцільним є використання методу скінченних елементів (МСЕ). Цей метод є надзвичайно гнучким інструментом для розв’язання задач з просторовою неоднорідністю. Він дозволяє враховувати зміну властивостей середовища на рівні окремих елементів, а також забезпечує гнучкість побудови сітки, дозволяючи робити її густішою в областях з очікуваними різкими змінами розв'язку (наприклад, в околі тонкого включення).

Ключовою перевагою МСЕ є перехід до слабкої (варіаційної) постановки задачі. Це дозволяє не тільки працювати з менш гладкими функціями, але й природно враховувати умови спряження та граничні умови другого роду (Неймана) без суттєвого ускладнення реалізації.

Для дискретизації за часом застосовується неявна схема, що гарантує чисельну стійкість навіть за наявності жорстких залежностей у системі (наприклад, при моделюванні процесів з дуже різними характерними часами, як-от швидка теплопровідність і повільна фільтрація).

3.2. Схема дискретизація

Щоб чисельно змоделювати складні процеси, такі як фільтрація та теплоперенос у пористих середовищах, необхідно перетворити неперервні (континуальні) рівняння на їхні дискретні аналоги. Цей перехід дозволяє замінити початкові диференціальні рівняння системою алгебраїчних рівнянь, яку можна розв'язати для обмеженої кількості точок (вузлів) у розрахунковій області.

Процес дискретизації включає поділ області моделювання на скінченну кількість елементів. У межах цих елементів невідомі величини, як-от напір і

температура, апроксимуються за допомогою базисних функцій. Основна ідея полягає в тому, щоб замінити інтегральні або диференціальні вирази їхніми кінцево-різницевими або інтегрально-дискретизованими аналогами.

Для просторової дискретизації використовується метод скінченних елементів, який забезпечує високу точність навіть у випадку нерівномірного поділу області. Для апроксимації змін у часі застосовуються різницеві схеми, наприклад, явні або неявні. Вибір конкретної схеми залежить від вимог до стійкості чисельного алгоритму.

Розіб'ємо відрізок $[0, l]$ на N кінцевих елементів із вузловими точками:

$$0 = x_0 < x_1 < \cdots < x_N = l$$

Врахуємо точку $x = \xi$ як внутрішню точку, де можлива наявність стрибку. Виберемо систему базисних функцій, $\{\varphi_i(x)\}_{i=1}^{N}$які належать до H_0 та можуть відтворювати стрибок у точці ξ. Шукані наближення $h_N(x,t)$, $C_N(x,t)$ та $T_N(x,t)$ представимо у вигляді:

$$h_N(x,t) = \sum_{i=1}^{N} h_i(t)\varphi_i(x),$$

$$T_N(x,t) = \sum_{i=1}^{N} T_i(t)\varphi_i(x),$$

$$C_N(x,t) = \sum_{i=1}^{N} C_i(t)\varphi_i(x),$$

Підставляючи формули, згадані вище, у варіаційну форму рівняння теплопереносу та обираючи $s(x) = \varphi_i(x), j = 1, \dots, N$, отримаємо системи нелінійних звичайних диференціальних рівнянь відносно векторів коефіцієнтів:

$$H(t) = \big(h_1(t), \dots, h_N(t)\big)^T$$

$$T(t) = \big(T_1(t), \dots, T_N(t)\big)^T$$

$$C(t) = \left(C_1(t), \dots, C_N(t)\right)^T$$

Для $H(t)$ маємо:

$$M_h(H, C, T)\frac{dH}{dt} + K_h(H, C, T)H = F_h(H, C, T)$$

для $\mathrm{C}(t)$:

$$M_{\mathrm{C}}(H, \mathrm{C}, T)\frac{dT}{dt} + K_{\mathrm{C}}(H, \mathrm{C}, T)H = F_{\mathrm{C}}(H, \mathrm{C}, T)$$

а для $T(t)$:

$$M_T(H, T)\frac{dT}{dt} + K_T(H, T)H = F_T(H, T)$$

де:

- M_h, M_{C}, M_T — матриці маси;
- K_h, K_{C}, K_T — матриці жорсткості;
- F_h, F_C, F_T — вектори правих частин.

Для інтегрування системи рівнянь для напору та температури у часі використаємо неявну схему, наприклад, схему Кранка–Ніколсона.. Позначимо:

$$t_j = j\tau$$

де τ — крок за часом. Тоді для рівняння напору маємо:

$$M_h\left(H^{(j+\frac{1}{2})}, T^{(j+\frac{1}{2})}\right)\frac{H^{(j+1)} - H^{(j)}}{\tau} + K_h\left(H^{(j+\frac{1}{2})}, T^{(j+\frac{1}{2})}\right)\frac{H^{(j+1)} + H^{(j)}}{2} = F_h\left(H^{(j+\frac{1}{2})}, C^{(j+\frac{1}{2})}, T^{(j+\frac{1}{2})}\right)$$

де $H^{(j)}$ — наближене значення вектора $H(t)$ у момент часу t_j.

Аналогічна формула справедлива і для рівняння часу відносно $T(t)$:

$$M_T\left(H^{\left(j+\frac{1}{2}\right)}, T^{\left(j+\frac{1}{2}\right)}\right)\frac{T^{(j+1)} - T^{(j)}}{\tau} + K_T\left(H^{\left(j+\frac{1}{2}\right)}, T^{\left(j+\frac{1}{2}\right)}\right)\frac{T^{(j+1)} + T^{(j)}}{2} = F_T\left(H^{\left(j+\frac{1}{2}\right)}, T^{\left(j+\frac{1}{2}\right)}\right)$$

Та концентрації відносно C(t):

$$M_C\left(H^{\left(j+\frac{1}{2}\right)}, C^{\left(j+\frac{1}{2}\right)}\right)\frac{C^{(j+1)}-C^{(j)}}{\tau}+K_C\left(H^{\left(j+\frac{1}{2}\right)}, C^{\left(j+\frac{1}{2}\right)}\right)\frac{C^{(j+1)}+C^{(j)}}{2}$$
$$=F_C\left(H^{\left(j+\frac{1}{2}\right)}, C^{\left(j+\frac{1}{2}\right)}, T^{\left(j+\frac{1}{2}\right)}\right)$$

3.3. Врахування умов спряження

Для реалізації умов спряження на межі включення використовується спеціальна техніка введення узагальнених елементів. У сітці вводяться додаткові умови на елементи, що містять точку , з урахуванням стрибків функцій:

$h^+ - h^- = R_h \cdot (\nabla h), \quad T^+ - T^- = R_T \cdot (\nabla T), \quad C^+ - C^- = R_C \cdot (\nabla C).$

Ці умови реалізуються у вигляді додаткових рівнянь або модифікації матриць жорсткості й маси у відповідних елементах.

3.5. Загальна структура обчислювального алгоритму

Об'єднуючи вищеописані підходи, можна сформулювати загальну покрокову структуру алгоритму для розв'язання задачі.

1. Ініціалізація:
   - Побудова скінченноелементної сітки для області Ω, з подвійними вузлами у точці включення ξ.
   - Задання початкових умов для напору, температури та концентрації у всіх вузлах сітки: X^0.
   - Встановлення параметрів моделювання: крок за часом Δt, кінцевий час t_{end}
2. Основний цикл за часом (від $n = 0$ до $t_{end}/\Delta t$):
   а) **Лінеаризація.** Оскільки система рівнянь на кроці $n+1$ є нелінійною, використовується ітераційний метод (наприклад, метод Пікара). На першій ітерації коефіцієнти матриці $K(X^{n+1})$

обчислюються на основі розв'язку з попереднього часового шару, $K(X^n)$.

b) **Формування глобальних матриць.** На кожній ітерації відбувається збирання глобальних матриць M та K з локальних (елементних) матриць. На цьому етапі враховуються умови спряження шляхом модифікації відповідних елементів матриці K.

c) **Врахування граничних умов.** Модифікація системи рівнянь для врахування граничних умов першого роду (Діріхле).

d) **Розв'язання системи.** Розв'язання отриманої лінійної алгебраїчної системи для знаходження наближення розв'язку на поточній ітерації.

e) **Перевірка збіжності.** Порівняння розв'язку поточної ітерації з попередньою. Якщо різниця менша за задану точність, перехід до наступного часового кроку. В іншому випадку — повернення до кроку (b) з оновленими коефіцієнтами.

f) Завершення та аналіз: Збереження та візуалізація отриманих результатів.

3.4. Чисельні експерименти та аналіз

З метою перевірки ефективності запропонованої математичної моделі масоперенесення при нелінійній неізотермічній фільтрації було проведено серію чисельних експериментів. У даному підрозділі подано аналіз результатів, отриманих внаслідок реалізації алгоритму на мові Python, описаного у попередньому розділі, з урахуванням впливу температури, концентрації, термодифузії та хімічного осмосу. Основна увага зосереджена на оцінці впливу тонкого включення (геобар’єру) на розподіл напору та концентрації.

Усі експерименти проводились з використанням скінченноелементної сітки, що враховує поділ області на два основні ґрунтові шари (суглинок і глина), між якими розташоване тонке напівпроникне включення. Для фіксації змін у часі застосовувалась неявна різницева схема (схема Кранка–Ніколсона), яка забезпечує чисельну стабільність. На межах області моделювання задавались граничні умови першого роду для напору, температури та концентрації.

У першому експерименті досліджено випадок без урахування температурних і концентраційних ефектів, з використанням лише класичного нелінійного закону Дарсі. Це дозволило визначити базовий розподіл напору в суглинку без впливу додаткових факторів. Результати показали відносно однорідний розподіл напору зі слабко вираженим градієнтом у центральній частині області.

Другий експеримент виконувався із включенням температурного впливу, що змінював в’язкість рідини та коефіцієнт фільтрації. Температурний градієнт створював суттєве підвищення швидкості фільтрації поблизу верхньої межі, що призводило до збільшення крутизни градієнту напору. У центральній частині області, поблизу тонкого включення, спостерігалося незначне зниження напору, що відповідає ефекту теплової деконсолідації.

У третьому моделюванні враховувався як температурний вплив, так і термодифузія, яка впливала на перерозподіл концентрації домішок. Концентрація в області моделювання початково була встановлена рівномірною, проте внаслідок термодифузійного переносу та хімічного осмосу спостерігалося накопичення речовини поблизу правої межі, де температура нижча. Це пояснюється ефектом направленого переносу речовини з гарячої зони в холодну, типового для систем із температурною неоднорідністю.

Четвертий експеримент виконувався з урахуванням геобар’єру в центральній частині області. В моделі реалізовані умови спряження на межах тонкого включення для напору, температури та концентрації. Результати засвідчили чіткий

стрибок напору на межах включення та зміну профілю концентрації у його околі. Розподіл напору демонструє значний перепад на межі глини, що пов’язано з її малою проникністю. Концентрація також зазнає стрибка, оскільки коефіцієнт термодифузії та пористість у глині інші, ніж у суглинку.

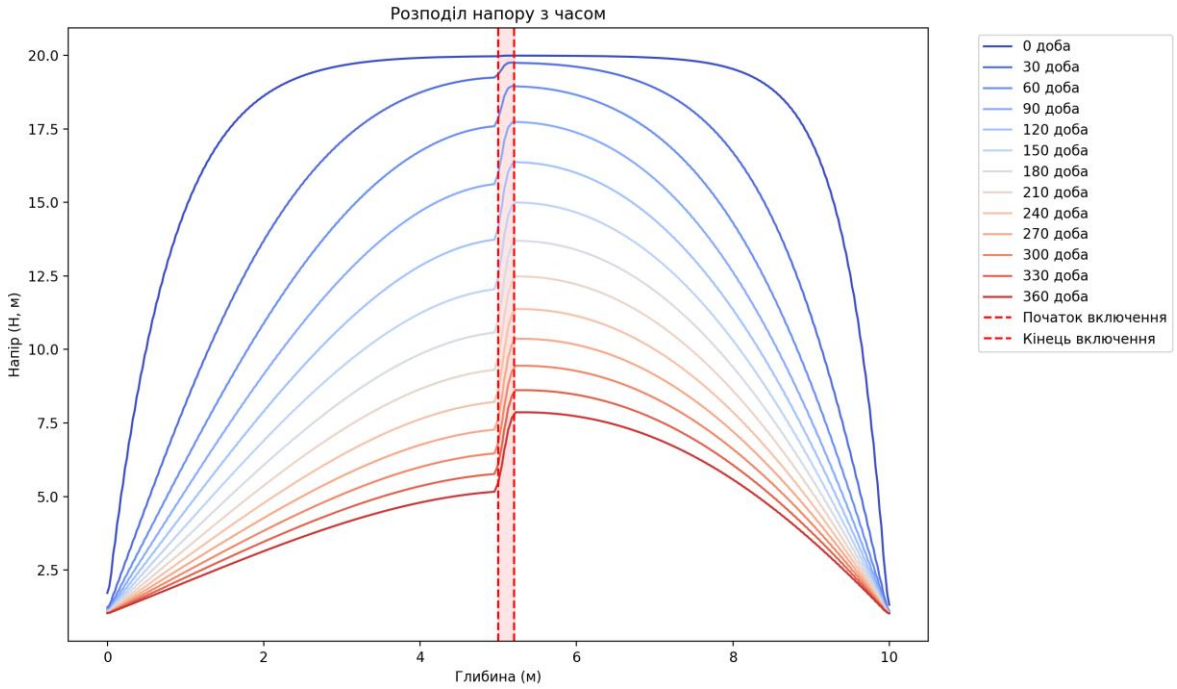

Рис. 3.1. Розподіл напорів в суглинку при заданні на межах граничної умови першого роду з тонким включенням глини

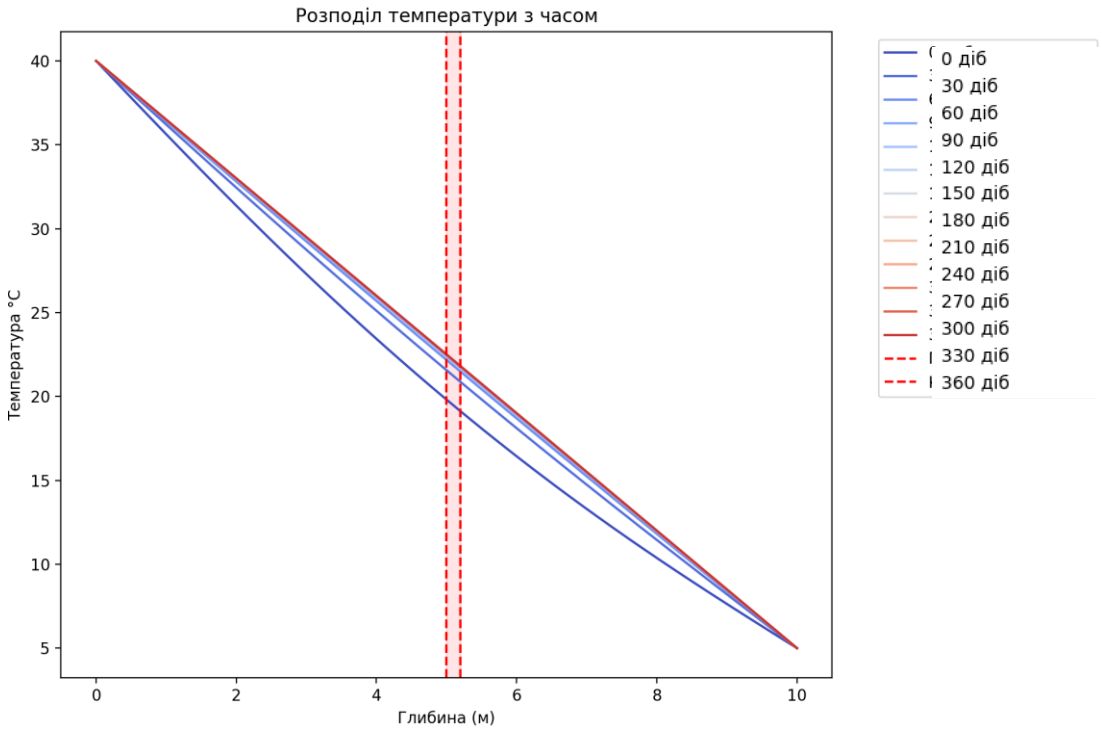

Рис. 3.2 Розподіл температури в суглинку при заданні на межах граничної умови першого роду з тонким включенням глини

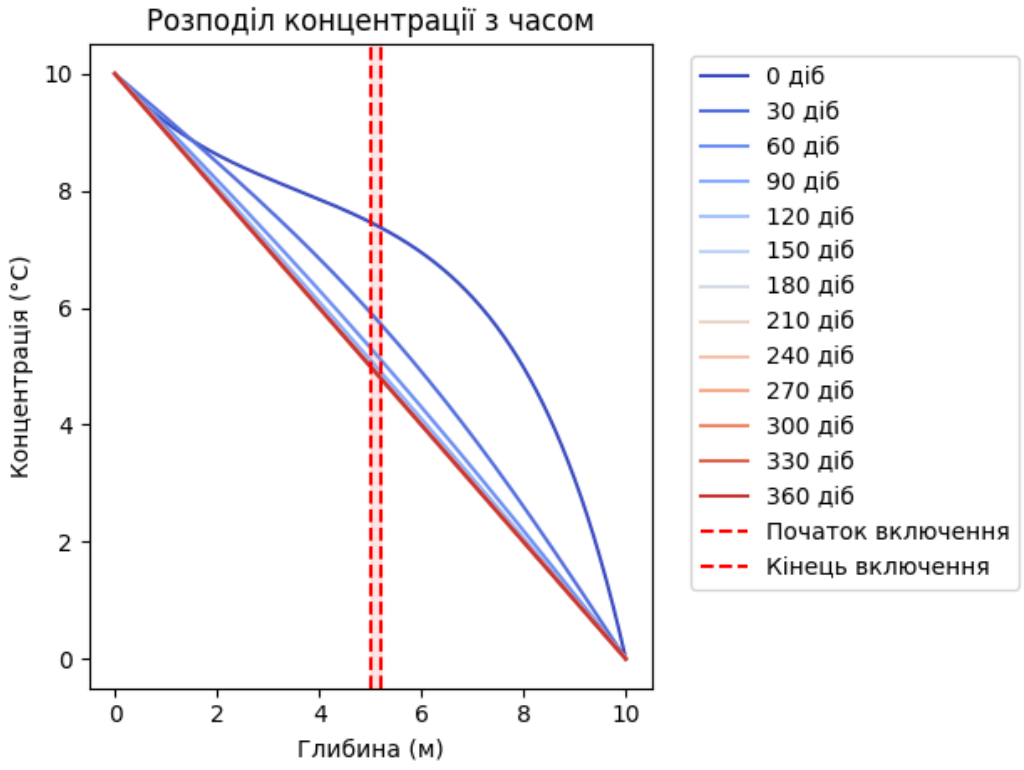

Рис. 3.2. Розподіл концентрації при врахуванні термодифузії та геобар’єру

У фінальному п’ятому моделюванні досліджено взаємний вплив усіх факторів одночасно: температури, концентрації, термодифузії, хімічного осмосу та напівпроникного включення. Такий підхід дозволяє змоделювати найбільш реалістичний сценарій, який відповідає умовам експлуатації полігонів захоронення відходів. Результати показали складну взаємодію між тепловими і концентраційними градієнтами, що формує зональність у профілях обох параметрів. Спостерігалось зростання градієнта напору поблизу гарячої межі та зміщення максимуму концентрації до області з найменшою температурою. В області геобар’єру виникала затримка переносу, що підтверджує ефективність бар’єру при фільтрації забруднень.

РОЗДІЛ 4
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Цей розділ присвячено практичній реалізації математичної моделі та чисельних алгоритмів, описаних у попередніх розділах. Розглядаються обрані інструментальні засоби, архітектура розробленого програмного комплексу та ключові аспекти імплементації алгоритму мовою програмування Python. Метою цього розділу є демонстрація перетворення теоретичних розробок на функціональний програмний інструмент для проведення чисельних експериментів.

4.1. Вибір інструментальних засобів розробки

Для реалізації поставленої задачі було обрано мову програмування **Python** та її екосистему наукових бібліотек. Цей вибір обґрунтований низкою переваг, що роблять Python стандартом у науково-технічних обчисленнях.

1. **Мова програмування Python.** Python є високорівневою мовою з простим та чистим синтаксисом, що значно прискорює процес розробки та полегшує підтримку коду. Її динамічна типізація та інтерпретованість ідеально підходять для швидкого прототипування та тестування наукових гіпотез.
2. **Ключові бібліотеки**. Потужність Python у наукових задачах значною мірою визначається наявністю спеціалізованих бібліотек з відкритим кодом:
    - **NumPy** — фундаментальна бібліотека для роботи з багатовимірними масивами та матрицями. У розробленій програмі вона використовується для зберігання векторів вузлових значень, елементів матриць системи та для виконання всіх векторних операцій, що є значно ефективнішим за стандартні списки Python.

- **SciPy** — бібліотека, що розширює функціонал NumPy, надаючи інструменти для вирішення специфічних наукових та інженерних задач. У проєкті були використані її модулі:
  - scipy.special.gammainc — для обчислення неповної гамма-функції, яка є частиною нелінійного закону фільтрації.
  - scipy.sparse та scipy.sparse.linalg.spsolve — для ефективного зберігання розріджених матриць жорсткості та мас (що є типовим для МСЕ) та для швидкого розв'язання систем лінійних алгебраїчних рівнянь.
- **Matplotlib** — провідна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій. Вона використовується на завершальному етапі для побудови графіків розподілу напору та температури, що дозволяє аналізувати результати моделювання.

3. **Середовище розробки (IDE)**. Хоча код може виконуватися в будь-якому середовищі, що підтримує Python, розробка велася з використанням сучасних IDE, таких як Visual Studio Code або PyCharm. Вони надають необхідні інструменти для комфортної роботи: підсвічування синтаксису, автодоповнення коду, інтегровані засоби для відлагодження (дебагінгу) та керування версіями (Git).

4.2. Архітектура програмного комплексу

Для забезпечення гнучкості, масштабованості та читабельності коду було застосовано об'єктно-орієнтований підхід. Програмний комплекс складається з кількох класів, кожен з яких виконує чітко визначену роль.

- class Parameters: Цей клас реалізує патерн "Конфігурація" і слугує централізованим сховищем для всіх параметрів моделі: фізичних констант,

геометричних розмірів, параметрів сітки, часових кроків, граничних умов та налаштувань для чисельних методів. Такий підхід дозволяє легко змінювати умови експерименту, не втручаючись у логіку основних обчислювальних функцій.

- class Material та class Element: Це прості класи-структури (data classes), призначені для зберігання даних. Material містить фізичні властивості матеріалів (наприклад, суглинку та глини), такі як початковий коефіцієнт фільтрації та пористість. Element зберігає всю інформацію про один скінченний елемент: його геометричні координати, вузли, тип матеріалу та коефіцієнти базисних функцій.
- class Model: Це основний клас-контролер, який інкапсулює всю логіку моделювання. Він відповідає за повний життєвий цикл симуляції:
  1. Ініціалізація та генерація сітки.
  2. Запуск основного циклу обчислень за часом.
  3. Збереження результатів у файли.
  4. Візуалізація результатів після завершення розрахунків.

Така архітектура робить код модульним, де кожна частина відповідає за свою логічну складову, що спрощує його розуміння, тестування та подальший розвиток.

4.3. Реалізація ключових етапів алгоритму

Програмна реалізація безпосередньо відтворює чисельний алгоритм, описаний у Розділі 3. Розглянемо імплементацію його ключових етапів.

1. **Генерація сітки та ініціалізація**. Метод generate_mesh() класу Model відповідає за просторову дискретизацію області. Він створює список об'єктів Element, послідовно розбиваючи розрахункову область на елементи заданого розміру. Метод initialize() встановлює початкові умови для напору, температури та пористості, заповнюючи відповідні масиви Hn, T_profile та Void_ratio на нульовому часовому кроці.

2. **Формування матриць системи**. Процес формування глобальних матриць мас (M_head) та жорсткості (L_H) реалізований у методах MakeM_head() та MakeLH(). Ці методи ітерують по всіх скінченних елементах сітки. Для кожного елемента вони обчислюють локальні матриці за допомогою інтегрування по елементу, яке реалізовано через квадратурні формули Гаусса (методи Integral() та F_integral()). Потім локальні матриці додаються до глобальних, формуючи загальну систему рівнянь для всієї області.

3. **Врахування нелінійності та умов спряження**. Нелінійність закону фільтрації та умови спряження на тонкому включенні є найскладнішими для реалізації частинами моделі.

- **Нелінійний закон фільтрації** реалізований усередині функції integral_gamma_LH(). Ця функція обчислює коефіцієнт гідравлічного опору включення, враховуючи залежність від температури та пороговий градієнт, що розраховується через неповну гамма-функцію gammainc.
- **Умови спряження** враховуються у методі MakeLH(). Після формування базової матриці жорсткості до її відповідних елементів, що відповідають вузлам на межі включення (num_first_left та num_first_right), додається член, що моделює опір включення, розрахований функцією integral_gamma_LH().

2. **Розв'язання системи та часовий крок**. Основний обчислювальний процес відбувається в методі solve(). Це цикл, що ітерує по часових кроках від t_start до t_end. На кожному кроці:

a) Оновлюється профіль температури (метод update_temperature()).
b) Формуються матриці M_head та L_H для поточного стану системи.
c) Система рівнянь збирається відповідно до схеми Кранка-Ніколсон.
d) Застосовуються граничні умови (Діріхле або Неймана) шляхом прямої модифікації фінальної матриці системи та вектора правих частин.

e) Отримана система лінійних алгебраїчних рівнянь розв'язується за допомогою методу Gaus(), що використовує ефективний розв'язувач spsolve для розріджених матриць.

f) Результати (вектор напорів x_sol) зберігаються у масиві Hn та записуються у файл.

3. **Візуалізація результатів**. Після завершення циклу обчислень викликається метод plot_results(), який зчитує дані з текстових файлів (pressure.txt, temperature.txt) та за допомогою бібліотеки Matplotlib будує графіки, що наочно демонструють динаміку розподілу напору та температури в часі та просторі.

ВИСНОВКИ

У ході дослідження було всебічно розглянуто проблему чисельного моделювання процесів масоперенесення у неоднорідних пористих середовищах з урахуванням температурних і концентраційних ефектів. Основну увагу зосереджено на побудові математичної моделі, яка описує взаємозв'язані фізичні процеси, що включають фільтрацію, теплоперенесення та перенесення домішок. Було досліджено вплив тонких напівпроникних включень на поведінку системи, що має важливе практичне значення для проєктування захисних гідротехнічних споруд.

У математичну постановку задачі включено модифікований закон Дарсі, рівняння теплопровідності з нелінійною залежністю коефіцієнта теплопровідності від напору, а також рівняння конвекції-дифузії для перенесення домішок, яке враховує ефекти термодифузії та хімічного осмосу. Для моделювання впливу тонкого геобар'єру реалізовано спеціальні умови спряження, які враховують стрибки температури, напору та концентрації.

Було реалізовано алгоритм чисельного розв'язання задачі на мові програмування Python, із використанням таких бібліотек як NumPy, SciPy та Matplotlib. Розрахунки виконувались на скінченноелементній сітці з підвищеною точністю поблизу включення, що забезпечило деталізацію процесів у критичних зонах. Для інтегрування у часі застосовано неявну схему Кранка–Ніколсона, яка дозволила уникнути чисельної нестійкості та забезпечити збіжність обчислень навіть при суттєвих просторових градієнтах параметрів.

Виконано низку чисельних експериментів, в яких поетапно аналізувався вплив окремих факторів: температури, термодифузії, хімічного осмосу та геобар'єру. Було виявлено, що врахування температурного впливу приводить до зміни фільтраційного потоку, зокрема до його посилення у зонах з підвищеною температурою. Хімічний осмос і термодифузія змінюють розподіл концентрації розчиненої речовини, спричиняючи її накопичення у холодних областях. Геобар'єр

вносить суттєві зміни в розподіл тиску та концентрації, діючи як фільтр, що затримує перенесення речовини.

Чисельні результати підтвердили ефективність запропонованої моделі та її придатність для використання в інженерній практиці. Модель дозволяє оцінити поведінку фільтраційної системи при змінних умовах, а також визначити потенційно небезпечні зони накопичення забруднень. Це відкриває можливість подальшого удосконалення геотехнічного захисту територій, що зазнають впливу фільтраційних процесів із термогідродинамічною складовою.

Загалом, виконана робота демонструє можливість застосування сучасного обчислювального підходу до розв’язання складних задач, пов’язаних із нестаціонарними процесами масоперенесення, у тому числі в умовах температурної нестабільності та структурної неоднорідності середовища. Отримані результати можуть бути покладені в основу подальших досліджень, спрямованих на оптимізацію протифільтраційного захисту й моделювання екологічних ризиків у зоні впливу техногенних об’єктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глушко В.П. Основи чисельного моделювання процесів переносу в пористих середовищах. — Львів: Видавництво Львівської політехніки, 2015. — 312 с.
2. Збірник тез студентської науково-практичної конференції навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки. Збірних тез. *Рівне: НУВГП*. 2023. С. 38. URL: https://ep3.nuwm.edu.ua/29343/ (дата звернення: 09.12.2024).
3. Мічута О. Р., Мартинюк П. М. Нелінійна еволюційна задача фільтраційної консолідації з некласичною умовою спряження. Journal of Optimization, Differential Equations and Their Applications. 2022. С. 71–87.
4. Самойленко Н.І., Ващенко В.М. Математичне моделювання процесів фільтрації в неоднорідних середовищах. — Київ: Наукова думка, 2012. — 224 с
5. Bear J. Dynamics of Fluids in Porous Media. — New York: Dover Publications, 1988. — 784 p.
6. Bear J. Dynamics of Fluids in Porous Media. — New York: Dover Publications, 1988. — 784 p.
7. Wang S., Wang Q., Lu B., Zhu W. Thermal effects of the hydraulic conductivity and threshold gradient of sand–clay liners in municipal solid waste landfills. Waste Management. 2022. P. 217–222.

ДОДАТКИ

Додаток А

Код програми

```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.special import gammainc
from scipy.sparse import diags
from scipy.sparse.linalg import spsolve
from numpy.linalg import LinAlgError
from scipy.ndimage import uniform_filter1d

# ============================================================
# Класи параметрів, матеріалів та сітки
# ============================================================

class Material:
    """
    Клас для зберігання властивостей матеріалу.
    """
    def __init__(self, name, k_base, porosity):
        self.name = name
        self.k_base = k_base
        self.porosity = porosity
        self.e_porosity = porosity / (1 - porosity)

class Parameters:
    """
    Клас для зберігання глобальних параметрів задачі.
    """
    def __init__(self):
        # Параметри точності
        self.poxibka = 1e-6
        self.epsilon = 1e-10

        # Параметри ґрунтових шарів та матеріалів
        self.loam = Material('loam', k_base=0.05, porosity=0.45)
        self.clay = Material('clay', k_base=0.00048, porosity=0.36)

        # Інші константи
        self.gama_water = 9810       # Н/м³ (питома вага води)
        self.miu_water = 8.90e-4     # Па·с (в'язкість води при 25°C)
        self.a_soil = 5.12e-4        # Стисливість ґрунту (1/Па)
        self.a_gamma = 5.12e-7
        self.max_vidx = 6

        # Параметри нелінійного закону
        self.alfa_soil = 2
        self.alfa_soil_inverse = 1 / self.alfa_soil
        self.eps_Darcy = self.poxibka
        self.a_threshold = 12.0e-12
        self.b_threshold = -0.78
        self.Threshold_min = 0.05

        self.stab_a = 3/2
        self.m_Pe = 1/3
```

```
# Напівпроникне включення (глина)
self.alfa_gamma = 5
self.alfa_gamma_inverse = 1 / self.alfa_gamma
self.gamma_complete_rez = math.gamma(self.alfa_soil_inverse)

# Геометричні параметри
self.yp = 0
self.yk12 = 5   # Розділення шарів
self.yk = 10
self.h = 0.05
self.d_gamma = 0.2

# Часові параметри
self.tau = 10.0
self.t_start = 0
self.t_end = self.tau * 39
self.m_start = round(self.t_start / self.tau)
self.m_end = round(self.t_end / self.tau)
self.m_all = round((self.t_end - self.t_start) / self.tau)

# Граничні умови за напором
self.H0 = 20
self.H_top = 1
self.H_bottom = 1

# Температурні параметри
self.T_left = 40.0
self.T_right = 5.0
self.alpha_T = 1e-6
self.dx_T = self.h
self.dt_T = self.tau * 86400
self.use_temperature = True

self.use_concentration = True
self.C_left = 10  # ліва гранична концентрація
self.C_right = 0.0  # права гранична концентрація
self.C0 = 8  # початкова концентрація
self.alpha_C = 1e-6  # коефіцієнт дифузії концентрації
self.dx_C = self.h
self.dt_C = self.tau * 86400
self.v_C = 0.000065    # Ню хімічний осмос
self.alpha_C_T = 0.000065  # коефіцієнт термодифузії
self.n_C = 0.4         # пористість піску

# Умови задачі
self.left_boundary_condition = 'Dirichlet'
self.right_boundary_condition = 'Dirichlet'
self.H_left = self.H_top
self.H_right = self.H_bottom
self.q_left = 0.0
self.q_right = 0.0
self.pok_Darcy = 0

# Файл для збереження
self.save_to_file = True
self.pressure_file = "pressure.txt"
self.temperature_file = "temperature.txt"
self.concentration_file = "concentration.txt"
```

```
        # Квадратурна формула Гаусса
        self.n_Gauss = 5
        self.x_Gauss = np.array([0, 0.0985350860, 0.304535727, 0.562025190,
0.801986582, 0.960190143])
        self.A_Gauss = np.array([0.027777778, 0.159820377, 0.242693594,
0.260463392, 0.208450667, 0.100794193])

    @property
    def my_poch(self):
        return round((self.yk - self.yp) / self.h)

    @property
    def Nx_T(self):
        return int((self.yk - self.yp) / self.dx_T) + 1

    @property
    def Nt_T(self):
        return int((self.t_end - self.t_start) * 86400 / self.dt_T) + 1

    @property
    def Nx_C(self):
        return int((self.yk - self.yp) / self.dx_C) + 1

    @property
    def Nt_C(self):
        return int((self.t_end - self.t_start) * 86400 / self.dt_C) + 1

class Element:
    """
    Клас для зберігання інформації про скінченний елемент.
    """
    def __init__(self):
        self.ne = 0
        self.xv_l = np.zeros(2)  # Лінійні вузли
        self.xv_s = np.zeros(3)  # Квадратичні вузли
        self.alfa = np.zeros(2)
        self.beta = np.zeros(2)
        self.nv_l = np.zeros(2, dtype=int)
        self.nv_s = np.zeros(3, dtype=int)
        self.material = 'loam'
        self.h = 0

# ============================================================
# Клас моделі для реалізації обчислень
# ============================================================

class Model:
    def __init__(self, params: Parameters):
        self.params = params
        self.mas_el = []
        self.ke = 0
        self.kv = None
        self.num_first_left = None
        self.num_first_right = None
        self.Hn = None
        self.Void_ratio = None
        self.Void_ratio_gamma = None
        self.T_profile = None
        self.C_profile = None
        self.generate_mesh()
```

```
        self.apply_break()

    def apply_break(self):
        """
        Формування розриву між шарами.
        """
        self.my_zag = 2 * self.params.my_poch + 1
        self.num_first_left = 2 * round((self.params.yk12 - self.params.yp) /
self.params.h)
        self.num_first_right = self.num_first_left + 1


    def generate_mesh(self):
        """
        Генерація сітки скінченних елементів.
        """
        ke = 0
        kv_l = 0
        kv_s = 0
        xv_1 = self.params.yp
        while xv_1 <= self.params.yk - self.params.poxibka:
            ke += 1
            element = Element()
            element.ne = ke
            element.xv_l[0] = xv_1
            element.xv_l[1] = xv_1 + self.params.h
            element.h = self.params.h
            element.xv_s[0] = xv_1
            element.xv_s[2] = xv_1 + self.params.h
            element.xv_s[1] = xv_1 + self.params.h / 2
            element.alfa[0] = (xv_1 + self.params.h) / self.params.h
            element.beta[0] = -1 / self.params.h
            element.alfa[1] = -xv_1 / self.params.h
            element.beta[1] = 1 / self.params.h

            element_midpoint = (element.xv_l[0] + element.xv_l[1]) / 2
            if self.params.yk12 <= element_midpoint <= self.params.yk12 +
self.params.d_gamma:
                element.material = 'clay'
            else:
                element.material = 'loam'

            element.nv_l[0] = kv_l
            element.nv_l[1] = kv_l + 1
            element.nv_s[0] = kv_s
            element.nv_s[1] = kv_s + 1
            element.nv_s[2] = kv_s + 2

            self.mas_el.append(element)
            kv_l += 1
            kv_s += 2
            xv_1 += self.params.h

        self.ke = ke
        self.kv = self.mas_el[-1].nv_s[-1]

    def get_material_params(self, material):
        """
        Отримання базової гідравлічної провідності для матеріалу.
        """
```

```
        if material == 'clay':
            return self.params.clay.k_base
        elif material == 'loam':
            return self.params.loam.k_base
        # За замовчуванням суглинок
        return self.params.loam.k_base

    def viscosity_water(self, T):
        """
        В'язкість води як функція температури (°C).
        """
        T_Kelvin = T + 273.15
        mu = 2.414e-5 * 10**(247.8 / (T_Kelvin - 140))
        return mu

    def K_filtration_T(self, K0, T):
        """
        Регулювання гідравлічної провідності за температурою.
        """
        mu_T = self.viscosity_water(T)
        mu_ref = self.viscosity_water(25)
        return K0 * mu_ref / mu_T

    def integral_gamma_LH(self, void_rat, h_plus, h_minus, leng, T_avg, material):
        """
        Обчислення інтегралу для напівпроникного включення.
        """
        p = self.params
        K0 = self.get_material_params(material)

        if not p.use_temperature:
            T_avg = 25
        K_filtration = self.K_filtration_T(K0, T_avg)

        k_gamma_filtration_denominator = (1 + void_rat + p.epsilon)
        k_gamma_filtration = (K_filtration * (1 + p.clay.e_porosity) /
k_gamma_filtration_denominator *
                              (void_rat**3 + p.epsilon) / (p.clay.e_porosity**3 +
p.epsilon))
        k_gamma_filtration = max(k_gamma_filtration, p.epsilon)

        if p.pok_Darcy != 0:
            K_ptrmeability_gamma_z = k_gamma_filtration * p.miu_water /
p.gama_water
            Threshold_gamma_z = p.a_threshold *
(K_ptrmeability_gamma_z**p.b_threshold)
            threshold_gamma_star = p.alfa_gamma / p.gamma_complete_rez *
Threshold_gamma_z
            h_diff_abs = abs((h_plus - h_minus) / leng)
            x_dop2 = h_diff_abs / threshold_gamma_star
            x_dop4 = Threshold_gamma_z / p.gamma_complete_rez *
gammainc(p.alfa_gamma_inverse, x_dop2**p.alfa_gamma)
            k_gamma_filtration *= (1 - x_dop4 / (h_diff_abs + p.eps_Darcy))

        return leng / k_gamma_filtration

    def compute_rhs_concentration_temperature(self, i, kv, ke, mas_el, C_profile,
T_profile):
        """
        Обчислення вектора правої частини, що відповідає доданкам ν∂C/∂x + μ∂T/∂x
```

```
        """
        p = self.params
        rhs = np.zeros(kv + 1)
        for el in mas_el:
            C_grad = np.zeros(3)
            T_grad = np.zeros(3)

            for j in range(3):
                node = el.nv_s[j]
                C_grad[j] = C_profile[node]
                T_grad[j] = T_profile[node]

            for i1 in range(3):
                for i2 in range(3):
                    grad_phi_i = el.beta[0] if i1 == 0 else (el.beta[1] if i1 == 1
else el.beta[0] + el.beta[1])
                    grad_phi_j = el.beta[0] if i2 == 0 else (el.beta[1] if i2 == 1
else el.beta[0] + el.beta[1])

                    avg_dCdx = np.dot(C_grad, [1 if k == i2 else 0 for k in
range(3)])
                    avg_dTdx = np.dot(T_grad, [1 if k == i2 else 0 for k in
range(3)])

                    mu_T = self.viscosity_water(np.mean(T_grad))  # μ(T)
                    contrib = - (
                            p.v_C * avg_dCdx +
                            mu_T * avg_dTdx
                    ) * grad_phi_i * el.h / 2  # інтегруємо лінійно

                    rhs[el.nv_s[i1]] += contrib
        return rhs

    def compute_rhs_thermodiffusion(self, i, kv, ke, mas_el, T_profile):
        """
        Доданок термодифузії до рівняння концентрації.
        """
        p = self.params
        rhs = np.zeros(kv + 1)
        for el in mas_el:
            T_grad = np.zeros(3)
            for j in range(3):
                node = el.nv_s[j]
                T_grad[j] = T_profile[node]

            for i1 in range(3):
                for i2 in range(3):
                    grad_phi_i = el.beta[0] if i1 == 0 else (el.beta[1] if i1 == 1
else el.beta[0] + el.beta[1])
                    grad_phi_j = el.beta[0] if i2 == 0 else (el.beta[1] if i2 == 1
else el.beta[0] + el.beta[1])

                    avg_dTdx = np.dot(T_grad, [1 if k == i2 else 0 for k in
range(3)])

                    scale = (1 + p.n_C) / (p.n_C * p.gama_water)
                    contrib = - scale * p.alpha_C_T * avg_dTdx * grad_phi_i * el.h
/ 2

                    rhs[el.nv_s[i1]] += contrib
```

```
        return rhs

    def compute_rhs_advection(self, C_profile, H_profile, T_profile):
        """
        Обчислює праву частину рівняння концентрації: -u(C,h,T) * ∂C/∂x
        Використовується просте наближення швидкості фільтрації u ≈ -K(h,T) * ∂h/∂x
        """
        p = self.params
        rhs = np.zeros(model.kv + 1)

        for el in model.mas_el:
            h_nodes = [H_profile[node] for node in el.nv_s]
            C_nodes = [C_profile[node] for node in el.nv_s]
            T_nodes = [T_profile[node] for node in el.nv_s]

            for i1 in range(3):
                for i2 in range(3):
                    # апроксимація похідних
                    dh_dx = h_nodes[i2]  # замість ∂h/∂x
                    dC_dx = C_nodes[i2]
                    T_local = np.mean(T_nodes)
                    K0 = self.get_material_params(el.material)
                    K = self.K_filtration_T(K0, T_local)
                    u = -K * dh_dx

                    grad_phi_i = el.beta[0] if i1 == 0 else (el.beta[1] if i1 == 1
else el.beta[0] + el.beta[1])
                    contrib = - u * dC_dx * grad_phi_i * el.h / 2

                    rhs[el.nv_s[i1]] += contrib

        return rhs

    def StartH(self, yz: float) -> float:
        """
        Початкова умова для напору.
        """
        return self.params.H0

    def F_integral(self, i, el, x, number_v1, number_v2, pok, T_local, C_local):
        """
        Обчислення підінтегрального виразу для матриць системи.
        """
        p = self.params
        alfa_x_beta = [el.alfa[i1] + x * el.beta[i1] for i1 in range(2)]

        func_square = [
            alfa_x_beta[0]*(2*alfa_x_beta[0]-1),
            alfa_x_beta[1]*(2*alfa_x_beta[1]-1),
            4*alfa_x_beta[0]*alfa_x_beta[1]
        ]

        func_square_dx = [
            el.beta[0]*(2*alfa_x_beta[0]-1)+2*el.beta[0]*alfa_x_beta[0],
            el.beta[1]*(2*alfa_x_beta[1]-1)+2*el.beta[1]*alfa_x_beta[1],
            4*(alfa_x_beta[0]*el.beta[1] + alfa_x_beta[1]*el.beta[0])
        ]

        void_ratio_z = 0
        for i1 in range(3):
```

```
            void_ratio_z += self.Void_ratio[i - 1, el.nv_s[i1]] * func_square[i1]

        K0 = self.get_material_params(el.material)
        if p.use_temperature:
            K_filtration = self.K_filtration_T(K0, T_local)
        else:
            K_filtration = K0

        if not np.isfinite(K_filtration) or K_filtration <= 0:
            K_filtration = p.epsilon

        if pok == 'M_head':
            return p.gama_water * p.a_soil / (1 + void_ratio_z + p.epsilon) *
func_square[number_v1] * func_square[number_v2]
        elif pok == 'L_H':
            return K_filtration * func_square_dx[number_v1] *
func_square_dx[number_v2]

        return 0

    def Integral(self, i, el, number_v1, number_v2, pok, n_Gauss, x_Gauss, A_Gauss,
T_local, C_local):
        """
        Обчислення інтегралу по елементу за формулами Гаусса.
        """
        Yakobian = el.xv_l[1] - el.xv_l[0]
        int_include = 0
        for i1 in range(n_Gauss):
            x = el.xv_l[0] + Yakobian * x_Gauss[i1]
            a = self.F_integral(i, el, x, number_v1, number_v2, pok, T_local,
C_local)
            if not np.isfinite(a):
                a = 0
            int_include += Yakobian * A_Gauss[i1] * a
        return int_include

    def MakeM_head(self, i, kv, ke, mas_el, n_Gauss, x_Gauss, A_Gauss, T_profile,
C_profile):
        """
        Формування масової матриці для напору.
        """
        M_head = np.zeros((kv + 1, kv + 1))
        for i11 in range(ke):
            T_local = T_profile[mas_el[i11].nv_s[1]]
            C_local = C_profile[mas_el[i11].nv_s[1]]
            for i12 in range(3):
                for i13 in range(3):
                    dop = self.Integral(i, mas_el[i11], i12, i13, 'M_head',
n_Gauss, x_Gauss, A_Gauss, T_local, C_local)
                    M_head[mas_el[i11].nv_s[i12], mas_el[i11].nv_s[i13]] += dop
        return M_head

    def MakeLH(self, i, kv, ke, mas_el, n_Gauss, x_Gauss, A_Gauss,
Void_ratio_gamma, Hn, d_gamma, T_profile, C_profile):
        """
        Формування матриці L_H.
        """
        p = self.params
        L_H = np.zeros((kv + 1, kv + 2))
        for i1 in range(ke):
```

```
            T_local = T_profile[mas_el[i1].nv_s[1]]
            C_local = C_profile[mas_el[i1].nv_s[1]]
            for i2 in range(3):
                for i3 in range(3):
                    dop = self.Integral(i, mas_el[i1], i2, i3, 'L_H', n_Gauss,
x_Gauss, A_Gauss, T_local, C_local)
                    if not np.isfinite(dop):
                        dop = 0
                    L_H[mas_el[i1].nv_s[i2], mas_el[i1].nv_s[i3]] += dop

        T_avg_gamma = (T_profile[self.num_first_left] +
T_profile[self.num_first_right]) / 2
        C_avg_gamma = (C_profile[self.num_first_left] +
C_profile[self.num_first_right]) / 2
        I_gamma = self.integral_gamma_LH(Void_ratio_gamma[i - 1], Hn[i - 1,
self.num_first_right],
                                         Hn[i - 1, self.num_first_left], d_gamma,
T_avg_gamma, 'clay')
        if I_gamma == 0:
            I_gamma = p.epsilon

        L_H[self.num_first_left, self.num_first_left] += 1 / I_gamma
        L_H[self.num_first_left, self.num_first_right] -= 1 / I_gamma
        L_H[self.num_first_right, self.num_first_right] += 1 / I_gamma
        L_H[self.num_first_right, self.num_first_left] -= 1 / I_gamma

        return L_H

    def Gaus(self, A, B):
        """
        Розв'язання системи лінійних рівнянь.
        """
        p = self.params
        diagonals = []
        offsets = list(range(-p.max_vidx + 1, p.max_vidx))
        for offset in offsets:
            diag = np.diag(A, k=offset)
            diagonals.append(diag)
        C = diags(diagonals, offsets).tocsr()
        try:
            x = spsolve(C, B)
        except LinAlgError as e:
            print("Помилка лінійної алгебри:", e)
            x = np.linalg.lstsq(C.toarray(), B, rcond=None)[0]
        return x

    def update_temperature(self, T, alpha_T, dt_T, dx_T, T_left, T_right):
        """
        Оновлення температурного профілю методом скінченних елементів.
        """
        Nx = len(T)
        K = np.zeros((Nx, Nx))
        M = np.zeros((Nx, Nx))
        F = np.zeros(Nx)
        r = alpha_T / dx_T**2

        for i in range(Nx - 1):
            K_e = r * np.array([[1, -1], [-1, 1]])
            M_e = (dx_T / 6) * np.array([[2, 1],
                                         [1, 2]])
```

```
            K[i:i+2, i:i+2] += K_e
            M[i:i+2, i:i+2] += M_e

        # Граничні умови
        K[0, :] = 0
        K[0, 0] = 1
        M[0, :] = 0
        M[0, 0] = 1
        F[0] = T_left

        K[-1, :] = 0
        K[-1, -1] = 1
        M[-1, :] = 0
        M[-1, -1] = 1
        F[-1] = T_right

        A = M + dt_T * K
        b = M @ T + F * dt_T
        T_new = np.linalg.solve(A, b)
        return T_new

    def update_concentration(self, C, alpha_C, dt_C, dx_C, C_left, C_right):
        """
        Оновлення температурного профілю методом скінченних елементів.
        """
        Nx = len(C)
        K = np.zeros((Nx, Nx))
        M = np.zeros((Nx, Nx))
        F = np.zeros(Nx)
        r = alpha_C / dx_C**2

        for i in range(Nx - 1):
            K_e = r * np.array([[1, -1], [-1, 1]])
            M_e = (dx_C / 6) * np.array([[2, 1],
                                         [1, 2]])
            K[i:i+2, i:i+2] += K_e
            M[i:i+2, i:i+2] += M_e

        # Граничні умови
        K[0, :] = 0
        K[0, 0] = 1
        M[0, :] = 0
        M[0, 0] = 1
        F[0] = C_left

        K[-1, :] = 0
        K[-1, -1] = 1
        M[-1, :] = 0
        M[-1, -1] = 1
        F[-1] = C_right

        A = M + dt_C * K
        b = M @ C + F * dt_C
        C_new = np.linalg.solve(A, b)
        return C_new



    def initialize(self):
        """
```

```
        Ініціалізація масивів напору, пористості та температури.
        """
        p = self.params
        self.Hn = np.zeros((p.m_end + 1, self.kv + 1))
        self.Void_ratio = np.zeros((p.m_end + 1, self.kv + 1))
        self.Void_ratio_gamma = np.zeros(p.m_end + 1)

        # Початковий розподіл напору
        for el in self.mas_el:
            for j in range(3):
                self.Hn[0, el.nv_s[j]] = self.StartH(el.xv_s[j])

        # Початкова пористість
        for el in self.mas_el:
            for j in range(3):
                self.Void_ratio[0, el.nv_s[j]] = self.params.loam.e_porosity

        for i in range(p.m_end + 1):
            self.Void_ratio_gamma[i] = self.params.clay.e_porosity

        # Початковий розподіл концентрації
        if self.params.use_concentration:
            self.C_profile = np.full(self.kv + 1, self.params.C0)
        else:
            self.C_profile = np.zeros(self.kv + 1)

        # Початковий температурний профіль
        if p.use_temperature:
            depth = np.linspace(p.yp, p.yk, self.kv + 1)
            # Формула для температурного поля (демо приклад)
            T_init = 0.15 * (depth**2) - p.T_right * depth + p.T_left
        else:
            T_init = np.full(self.kv + 1, p.T_left)
        self.T_profile = T_init

    def solve(self):
        """
        Основний цикл розв'язання задачі.
        """
        p = self.params

        # Очищення файлів виводу
        if p.save_to_file:
            open(p.pressure_file, "w").close()
            open(p.temperature_file, "w").close()
            open(p.concentration_file, "w").close()

        self.initialize()
        T_history = [self.T_profile.copy()]

        # Основний часовий цикл
        for i in range(1, p.m_end + 1):
            # Оновлення температури
            if p.use_temperature:
                self.T_profile = self.update_temperature(self.T_profile, p.alpha_T,
p.dt_T, p.dx_T, p.T_left, p.T_right)
            else:
                self.T_profile = np.full(self.kv + 1, p.T_left)
            T_history.append(self.T_profile.copy())
```

```
            if p.use_concentration:
                self.C_profile = self.update_concentration(
                    self.C_profile,
                    p.alpha_C, p.dt_C, p.dx_C,
                    p.C_left, p.C_right
                )

                rhs_thermodiff = self.compute_rhs_thermodiffusion(i, self.kv,
self.ke, self.mas_el, self.T_profile)
                self.C_profile += rhs_thermodiff  # додаємо термодифузійний внесок



            M_head = self.MakeM_head(i, self.kv, self.ke, self.mas_el, p.n_Gauss,
p.x_Gauss, p.A_Gauss, self.T_profile, self.C_profile)
            L_H = self.MakeLH(i, self.kv, self.ke, self.mas_el, p.n_Gauss,
p.x_Gauss, p.A_Gauss, self.Void_ratio_gamma, self.Hn, p.d_gamma, self.T_profile,
self.C_profile)

            # Формуємо систему рівнянь
            for i1 in range(self.kv + 1):
                for j1 in range(self.kv + 1):
                    L_H[i1, self.kv + 1] += M_head[i1, j1] / p.tau * self.Hn[i - 1,
j1]
                    L_H[i1, j1] = M_head[i1, j1] / p.tau + L_H[i1, j1]



            # Граничні умови для напору
            if p.left_boundary_condition == 'Dirichlet':
                self.Hn[i, 0] = p.H_left
                for i1 in range(self.kv + 1):
                    L_H[0, i1] = 0
                L_H[0, 0] = 1
                L_H[0, self.kv + 1] = self.Hn[i, 0]
            elif p.left_boundary_condition == 'Neumann':
                L_H[0, self.kv + 1] += p.q_left

            if p.right_boundary_condition == 'Dirichlet':
                self.Hn[i, self.kv] = p.H_right
                for i1 in range(self.kv + 1):
                    L_H[self.kv, i1] = 0
                L_H[self.kv, self.kv] = 1
                L_H[self.kv, self.kv + 1] = self.Hn[i, self.kv]
            elif p.right_boundary_condition == 'Neumann':
                L_H[self.kv, self.kv + 1] += -p.q_right

            rhs_extra = self.compute_rhs_concentration_temperature(
                i, self.kv, self.ke, self.mas_el,
                self.C_profile, self.T_profile
            )

            a_Gaus = np.array(L_H)
            n, m = a_Gaus.shape
            A = a_Gaus[:n, :m - 1]
            B = a_Gaus[:n, m - 1]
            B += rhs_extra
            x_sol = self.Gaus(A, B)

            for i1 in range(self.kv + 1):
```

```
                self.Hn[i, i1] = x_sol[i1]

            self.Void_ratio[i, :] = self.Void_ratio[i - 1, :]
            self.apply_break()
            self.Void_ratio_gamma[i] = self.Void_ratio_gamma[i - 1]

            # Збереження даних у файл
            if p.save_to_file:
                with open(p.pressure_file, "a") as file:
                    file.write('\t'.join([str(round(val, 4)) for val in x_sol]) +
'\n')

                with open(p.temperature_file, "a") as file:
                    file.write('\t'.join([str(round(val, 4)) for val in
self.T_profile]) + '\n')
                with open(p.concentration_file, "a") as file:
                    file.write('\t'.join([str(round(val, 4)) for val in
self.C_profile]) + '\n')


        return T_history

    def plot_results(self):
        """
        Побудова графіків результатів.
        """
        p = self.params
        inclusion_start = p.yk12
        inclusion_end = p.yk12 + p.d_gamma

        # Напір
        data = []
        with open(p.pressure_file, "r") as file:
            lines = file.readlines()
            for line in lines:
                data.append([float(x) for x in line.split()])

        # Згладжування даних
        window_size = 5
        data = [uniform_filter1d(data_set, size=window_size) for data_set in data]

        fig, ax = plt.subplots()
        colors = plt.cm.coolwarm(np.linspace(0, 1, len(data)))
        tm = 3
        for i in range(len(data)):
            if tm == 3:
                x_values = np.linspace(p.yp, p.yk, self.kv + 1)
                ax.plot(x_values, data[i], label="{} діб".format(i*10),
color=colors[i])
                tm = 0
            tm += 1

        ax.axvline(x=inclusion_start, color='red', linestyle='--', label="Початок
включення")
        ax.axvline(x=inclusion_end, color='red', linestyle='--', label="Кінець
включення")
        ax.axvspan(inclusion_start, inclusion_end, color='red', alpha=0.1)
        ax.legend(loc="upper left", bbox_to_anchor=(1.05, 1))
        ax.set_xlabel('Глибина (м)')
        ax.set_ylabel('Напір (м)')
```

```
        ax.set_title('Розподіл напору з часом')
        plt.tight_layout()
        plt.show()

        # Температура
        data = []
        with open(p.temperature_file, "r") as file:
            lines = file.readlines()
            for line in lines:
                data.append([float(x) for x in line.split()])

        fig, ax = plt.subplots()
        colors = plt.cm.coolwarm(np.linspace(0, 1, len(data)))
        tm = 3
        for i in range(len(data)):
            if tm == 3:
                x_values = np.linspace(p.yp, p.yk, self.kv + 1)
                ax.plot(x_values, data[i], label="{} діб".format(i*10),
color=colors[i])
                tm = 0
            tm += 1

        ax.axvline(x=inclusion_start, color='red', linestyle='--', label="Початок
включення")
        ax.axvline(x=inclusion_end, color='red', linestyle='--', label="Кінець
включення")
        ax.axvspan(inclusion_start, inclusion_end, color='red', alpha=0.1)
        ax.legend(loc="upper left", bbox_to_anchor=(1.05, 1))
        ax.set_xlabel('Глибина (м)')
        ax.set_ylabel('Температура (°C)')
        ax.set_title('Розподіл температури з часом')
        plt.tight_layout()
        plt.show()

        data = []
        with open(p.concentration_file, "r") as file:
            lines = file.readlines()
            for line in lines:
                data.append([float(x) for x in line.split()])

        fig, ax = plt.subplots()
        colors = plt.cm.coolwarm(np.linspace(0, 1, len(data)))
        tm = 3
        for i in range(len(data)):
            if tm == 3:
                x_values = np.linspace(p.yp, p.yk, self.kv + 1)
                ax.plot(x_values, data[i], label="{} діб".format(i * 10),
color=colors[i])
                tm = 0
            tm += 1

        ax.axvline(x=inclusion_start, color='red', linestyle='--', label="Початок
включення")
        ax.axvline(x=inclusion_end, color='red', linestyle='--', label="Кінець
включення")
        ax.axvspan(inclusion_start, inclusion_end, color='red', alpha=0.1)
        ax.legend(loc="upper left", bbox_to_anchor=(1.05, 1))
        ax.set_xlabel('Глибина (м)')
        ax.set_ylabel('Концентрація (°C)')
        ax.set_title('Розподіл концентрації з часом')
```

```
        plt.tight_layout()
        plt.show()

# ============================================================
# Запуск моделювання
# ============================================================

if __name__ == "__main__":
    params = Parameters()
    model = Model(params)
    model.solve()
    model.plot_results()
```