Київський академічний університет
Helmholtz-Zentrum Dresden-Rossendorf e.V.

Комп'ютерне моделювання фізичних систем

Олександр Пилиповський

Київ, Дрезден — 2022

Зміст

Вступ

Традиційними методами дослідження, які застосовуються у природничих науках, є експеримент і теоретичний аналіз математичних моделей конкретних явищ. Поява досить потужних і надійних електронно-обчислювальних машин в середині XX сторіччя доповнила інструментарій науковця методами чисельного аналізу. Особливу роль у них посідає комп'ютерний експеримент: чисельне дослідження складних математичних моделей певної системи. Для фізика-теоретика це дає можливість проаналізувати поведінку систем з великою кількістю ступенів вільності чи сильними нелінійностями, порівняти точні обрахунки зі спрощеними аналітичними моделями. Комп'ютерні моделювання у якості супроводу експериментальних досліджень дозволяють перевірити припущення про вплив різних факторів на отриманий результат (наприклад, відсутність доданка в енергії з певною симетрією призводить до зникнення ефекту, що спостерігається).

Вважається, що комп'ютерний експеримент було започатковано італо-американським фізиком Енріко Фермі (1901—1954 рр.). Разом із Джоном Уламом, Станісласом Пастою і Мері Цингу досліджувались нелінійні коливання стрижня, представленого як сукупність з 64 матеріальних точок, з'єднаних пружинами[1]. Очікувалось, що за наявності невеликого початкового збудження у системі з часом встановиться термодинамічна рівновага і можна буде зробити певні висновки про термодинамічні властивості моделі. Але замість поступового перекачування енергії до вищих мод відбувалася періодична динаміка: через певний час система поверталася до початкового стану — термалізації системи не відбулося. Згодом, модель Фермі—Пасти—Улама—Цингу зіграла важливу роль у становленні фізики нелінійних явищ, зокрема у теорії солітонів, нерівноважної термодинаміки тощо.

Як і у натурному експерименті, від результатів комп'ютерних моделювань вимагається достовірність та отримання відповіді за прийнятних витрат машинних ресурсів. Це зумовило розвиток спеціальних математичних методів (наприклад, метод скінченних елементів, *ab initio*, теорія функціоналу густини, метод Монте Карло), орієнтованих на розв'язання диференціальних рівнянь спеціального виду чи збереження прийнятної точності обрахунків за надвеликої кількості ступенів вільності. Метою даного посібника є надати читачу уявлення про розробку пакетів комп'ютерних моделювань, зокрема технічні особливості та обмеження стандартних методів, які необхідно враховувати для правильної

[1] *Fermi E., Pasta J. R., Ulam S.* Studies of Nonlinear Problems // Los Alamos Scientific Laboratory Report LA-1940. — 1955; *Dauxois T.* Fermi, Pasta, Ulam, and a mysterious lady // Physics Today. — 2008. — Vol. 61, — P. 55.

постановки чисельного експерименту та інтерпретації результатів. Для цього розглядаються:

- основні методики комп'ютерного експерименту, які базуються на розв'язанні диференціальних рівнянь у частинних похідних (це стосується широкого класу задач фізики твердого тіла, електродинаміки, гідрогазодинаміки тощо);
- базові поняття та інструменти комп'ютерної інженерії, які стають у нагоді при розробці програмного забезпечення наукового спрямування.

У даному посібнику розглянуто розв'язання деяких типових фізичних задач на прикладі наномагнетизму як області фізики, яка з одного боку близька автору, а з іншого, завдяки інтенсивному розвитку протягом останніх кількох десятків років — характеризується великою кількістю програм комп'ютерної фізики (як комерційних, так і open-source), що дозволяє порівнювати вдалість різних підходів не лише у суто фізичному, а й технічному аспекті. Поза межами теми книги залишаються більш спеціальні задачі, які на сьогодні розв'язуються невеликою кількістю спеціалізованих програм на кшталт VASP (Vienna Ab initio Simulation Package)[2]. Від читача очікується знання основ математичної фізики, алгоритмів і програмування (об'єктно-орієнтованого підходу). Приклади коду ілюструються за допомогою мови Python 3 та Wolfram Mathematica.

Примітка 0.

У примітках наводиться супутня інформація, яка може бути корисною для загального уявлення про предметну область чи конкретний приклад.

Коментар 0.

У коментарях обговорюються аспекти конкретних тем, на які варто звернути увагу.

Структура книги наступна.
****

Структура

****

Автор висловлює подяку за цінні коментарі та відгуки Михайлу Володимировичу Висоцькому, Євгену Андрійовичу Слюсарю та Артему Володимировичу Томілo.

[2] https://www.vasp.at/

Розділ 1

Інженерія програмного забезпечення

> The advantage of the computer over the brain is that at least it is being used
>
> Gabriel Laub

1.1 Архітектура програмного забезпечення наукового спрямування

Розробка програмного забезпечення (ПЗ) наукового спрямування охоплює написання користувацького інтерфейсу та, головне, реалізацію специфічних для конкретної задачі алгоритмів, що ріднить її із розробкою ПЗ для інших спеціалізованих задач. Алгоритми можуть реалізовуватись у коді власноруч у повному обсязі, або ж залучати виклики сторонніх бібліотек. Оптимізація часу виконання та вимог до обчислювальних потужностей супровідних елементів архітектури ПЗ (елементів користувацького інтерфейсу тощо), як правило, розглядається як другорядна задача у порівнянні з формуванням принципово позитивного «user experience», тобто загального враження від зручності роботи з програмою та можливості запобігти помилкам користувача у роботі з нею. Водночас швидкодія цільових алгоритмів та їх вимоги до пам'яті, особливості їх реалізації впливають на точність та принципову можливість отримати коректні чисельні результати.

Примітка 1. Nmag

Пакети мікромагнітних моделювань magnum.fe [1] та Nmag [2; 3] одними з перших (у мікромагнетизмі) поєднали користувацький інтерфейс, реалізований як бібліотека Python, з можливістю написання власних скриптів розширення функціоналу з цільовим кодом, орієнтованим на розв'язання нелінійних диференційних рівнянь у частинних похідних.

Це варто розглядати як успішний і перспективний підхід, що забезпечує високу гнучкість формулювання власних задач на базі конкретного вирішувача (бібліотеки розв'язання заданих систем рівнянь, англ. solver). Зазначимо, що підхід інтерпретації зовнішніх скриптів мовою Tcl також залучено у пакеті моделювань OOMMF [4; 5]. Бібліотека Tk цієї ж мови була використана для графічного інтерфейсу, щоправда, сама мова не є широко розповсюдженою і вимагає набуття спеціальних навичок для використання.

Варто відзначити, що Nmag реалізує збереження даних у відкритому форматі HDF5 й у вигляді вихідних даних продукується єдиний файл великого розміру. З одного боку, це спрощує аналіз послідовного набору станів системи, а з іншого ускладнює збереження даних, якщо вихідний файл сягає розміру десятків ГБ чи практичний інтерес становить лише кінцевий стан.

Надалі основна увага в обговореннях буде зосереджена на імплементації бізнес-логіки задач (див. також шаблон проектування MVC), де аналізується взаємодія розподілених по простору об'єктів. Це можуть бути окремі частинки або ефективні об'єкти, утворені дискретизацією неперервних функцій. Математично, їх формулювання часто виражено в рівняннях у частинних похідних по простору і часу. Як правило, якщо часова динаміка, аналізується стандартними алгоритмами розв'язування нелінійних диференційних рівнянь типу сімейства методів Рунге—Кутти, то інтегрування по простору потребує врахування симетрії задачі (методи скінченних різниць, об'ємів чи елементів).

Примітка 2. Бізнес-логіка та користувацький інтерфейс

Під бізнес-логікою в інженерії ПЗ розуміють все, що стосується безпосередньо предметної області задачі й описує її у термінах мови програмування. Наприклад, алгоритми обрахунку розповсюдження потоку тепла, опис геометрії області визначення, умови, за яких обирається та чи інша гілка обчислень, фізичні принципи функціонування приладу в цілому тощо. На відміну від бізнес-логіки, код користувацького інтерфейсу визначає що доступно користувачу для взаємодії та спосіб, у який елементи керування доступні для взаємодії. Наприклад, в залежності від обраного режиму обчислень, користувачу надаються різні форми введення даних: для задач гідрогазодинаміки необхідно вводити густину і в'язкість робочого тіла, а для аналізу надпровідного магніту — силу струму. Корисним є відокремлення бізнес-логіки від користувацького інтерфейсу, що дозволяє їх незалежну модифікацію і, як наслідок, більш гнучку для розвитку архітектуру програмного продукту.

1.2 Асимптотичні оцінки

Конкретну реалізацію певного алгоритму чисельних методів можна схарактеризувати тою чи іншою швидкодією і вимогами на пам'ять. Але безпосереднє порівняння різних реалізацій може не мати особливого змісту через особливості різних мов програмування, залучених бібліотек та комп'ютерних потужностей. Так, реалізація сортування виключно базовим функціоналом мови Python буде порівняно повільною (й, можливо, залежною від версії інтерпретатора), в той час, як залучення сортування зі сторонніх бібліотек (NumPy, реалізована на C++) суттєво його пришвидшить, формально залишаючись у межах мови Python для користувача продукту. Водночас швидкодія матричних операцій у NumPy буде залежати від того, з якими бібліотеками лінійної алгебри його зібрано на даному комп'ютері [6]. Тому, щоб схарактеризувати алгоритм *асимптотично*, безвідносно специфіки можливої реалізації та доступних обчислювальних потужностей, користуються так званою O-, Θ- та Ω-нотаціями, які спираються на об'єм вхідних даних.

Скалярний добуток двох векторів розмірності n потребує виконання n добутків та $n-1$ операції додавання. Нехай, множення займає комп'ютерний час Δt_1, а додавання — Δt_2. Тоді загальний час виконання скалярного добутку буде рівний

$$T_{\text{dot}}(n) = n\Delta t_1 + (n-1)\Delta t_2 + \Delta t_0 = n(\Delta t_1 + \Delta t_2) + \Delta t_0 - \Delta t_2, \tag{1.1}$$

де Δt_0 характеризує час на інші підготовчі операції, незалежні від даних (виклик функції, повернення результату тощо). Так само можна оцінити час виконання добутку матриці розміру $n \times n$ на відповідний вектор:

$$T_{\text{mat-vec}}(n) = n^2\Delta t_1 + n(n-1)\Delta t_2 + \Delta t_0' = n^2(\Delta t_1 + \Delta t_2) - n\Delta t_2 + \Delta t_0', \tag{1.2}$$

де $\Delta t_0'$ має такий самий зміст, як і Δt_0, але для іншої функції. Іншими словами, для скалярного і матрично-векторного добутків можна підібрати лінійну і квадратичну функції $f_{\text{lin}}(n; a, b) = nx + b$ та $f_{\text{quad}}(n; a, b, c) = an^2 + bn + c$ з певними коефіцієнтами, що при досить великих n (тому й мова йде про асимптотичну оцінку) будуть виконуватись нерівності

$$\begin{aligned} f_{\text{lin}}(n; a_1, b_1) \le T_{\text{dot}}(n) \le f_{\text{lin}}(n; a_2, b_2) \\ f_{\text{quad}}(n; a_1, b_1, c_1) \le T_{\text{mat-vec}}(n) \le f_{\text{quad}}(n; a_2, b_2, c_2). \end{aligned} \tag{1.3}$$

У такому випадку кажуть, що час виконання скалярного добутку є $\Theta(n)$, а матрично-векторного — $\Theta(n^2)$ (читається як «велика тета від n^2»). Тобто, час виконання конкретного алгоритму скалярного добутку завжди зростає лінійно зі збільшенням розмірності векторів, а матрично-векторного — квадратично. Звертаємо увагу, що при цьому доданки молодших ступенів ігноруються, оскільки мова йде про основний внесок у швидкість. Якби у формулу обчислення часу виконання програми входив факторіал, то на його фоні можна було б ігнорувати будь-які степеневі чи показникові функції. Легко побачити, що для великих n виконується ієрархія

$$1 \ll \log n \ll n^m \ll e^n \ll n! \tag{1.4}$$

де m — певне невід'ємне число. Строго означити цю поведінку можна наступним чином. Алгоритм характеризується складністю $\Theta(f(n))$, якщо для досить великого n_0 можна підібрати такі сталі $C_{1,2}$, що буде справедливою нерівність

$$C_1 f(n) \leq T(n) \leq C_2 f(n), \quad n > n_0. \tag{1.5}$$

Через те, що мова йде про асимптотичну поведінку, у виразі (1.4) не важливі основи логарифму та експоненти, оскільки їх зміна впливає лише на коефіцієнти $C_{1,2}$.

Для значної кількості алгоритмів може існувати верхня або нижня межа оцінки швидкодії. Наприклад, пошук числа у невпорядкованому масиві розміру n може закінчитись на першому кроці, якщо шукане число знаходиться на першій позиції, або ж виконається максимальна кількість кроків n, якщо воно знаходиться на останній позиції. Так само сортування може закінчитись за кілька кроків, якщо на вхід подано вже відсортований масив, або ж довго працювати, якщо впорядкування виявилось найменш оптимальним відносно припущень про початковий розподіл елементів. Тоді, аналогічно до (1.5), кажуть, що алгоритм поводить себе як $O(f(n))$ якщо існує верхня оцінка швидкодії, яка характеризується функцією $f(n)$, або ж $\Omega(f(n))$ за наявності нижньої оцінки. Звертаємо увагу, що Θ-оцінка має на увазі одночасну наявність O- та Ω-оцінок.

Коментар 1. Вплив розміру вхідних даних на вибір алгоритму

Слід зауважити, що конкретні коефіцієнти в оцінці швидкодій можуть бути важливими, у тих випадках, коли програма потенційно може працювати з даними суттєво різних масштабів. Наприклад, бульбашкове сортування характеризується складністю виконання $O(n^2)$, а швидке сортування — $O(n \log n)$. На практиці виявляється, що сортування бульбашкою швидше за швидке для малих n, тому в конкретній задачі може мати сенс динамічна заміна алгоритму в залежності від розміру вхідних даних (див. шаблон проектування «Стратегія»).

Асимптотичні O-, Ω- і Θ-оцінки широко застосовуються не лише для характеристик часу виконання, а й для вимог на пам'ять чи інших, споріднених з розміром вхідних даних величин. Наприклад, хеш-таблиці швидші за зв'язані списки у пошуку даних (час доступу $O(1)$ проти $O(n)$), але вимагають більшого об'єму пам'яті. Для задач, які розв'язуються методами скінченних різниць чи елементів, про які піде мова далі, характерним параметром слугуватиме крок дискретизації по простору чи часу.

Примітка 3. Зв'язані списки

У тих випадках, коли принциповою є економія пам'яті, але не швидкодія, для збереження великої кількості однотипних даних зручно використовувати так звані зв'язані списки замість масивів. Це структури даних наступного вигляду:

```
struct LinkedList {
Data item;
```

```
LinkedList nextElement
}
```

де `Data` позначає тип даних, які зберігаються. Ідея зв'язаного списку полягає у тому, що кожен його елемент містить дані `item` та посилання на наступний елемент зв'язаного списку `nextElement`. Так у пам'яті можна зберігати лише потрібні дані без резервування місця під їх оновлення. Практичні реалізації зв'язаних списків можуть мати посилання й на попередні елементи (подвійно зв'язані списки).

Примітка 4. Хеш-таблиці

Для задач, де принциповою є швидкість доступу до певного елементу даних (як правило для перевірки, чи було збережено такий елемент), альтернативою масивам слугують хеш-таблиці. У звичайному масиві необхідно виконати той чи інший алгоритм пошуку елементу. Ідея ж хеш-таблиці полягає в тому, що по шуканому елементу обчислюється певний код (хеш), який однозначно асоціюється з адресою елементу в пам'яті. У найпростішому випадку, індексом елементу в масиві може бути остача від ділення цілочисельного хешу на розмір масиву. Далі залишається лише перевірити, чи порожня ця адреса. Тип даних «словник» (dictionary) у мові Python є прикладом хеш-таблиці [7]. Зауважимо, що під місцем у пам'яті може розумітись індекс у досить великому масиві, а практична задача встановлення однозначної відповідності між довільним об'єктом і адресою в пам'яті є складною й може потребувати залучення спеціальних методів розв'язання колізій (випадків, коли різні дані дають однаковий хеш).

Наостанок підкреслимо, що вищерозглянута нотація виду $f(x) = O(g(x))$ не означає рівності у математичному сенсі, а використовується для позначення наявності специфічної властивості у функції $f(x)$, тому записи виду $O(g(x)) = \bullet$ є некоректними.

1.3 Принципи розробки програмного забезпечення

Комп'ютерне моделювання є потужним засобом дослідження фізичних, біологічних та інших систем, що застосовується як для перевірки теоретичних передбачень, так і для пояснення явищ, які спостерігаються експериментально. З урахуванням його активного застосування зростає роль якості розробки ПЗ для наукових цілей, аби забезпечити прийнятний рівень достовірності та відтворюваності отриманих чисельних результатів.

Розробка програмного забезпечення (ПЗ) для наукових цілей значною мірою схожа з процесом створення комерційних програм й завдяки окремим особливостям (наприклад, наявність математичної моделі та знайомість предметної області розробнику) дозволяє спрощення окремих кроків у проектуванні про-

дукції. Основною проблемою, з якою стикаються розробники обох типів ПЗ, є перевірка достовірності отриманих результатів. Відомі численні випадки елементарних помилок у коді або домовленостях між командами програмістів, які призводили до коштовних збитків. Застосування сучасних методик розробки дозволяє суттєво покращити якість та тривалість життя наукового ПЗ.

Коментар 2. Mars Climate Orbiter

Космічний апарат для дослідження Марсу було запущено у 1998 році. Він вийшов на заплановану орбіту червоної планети у вересні наступного року й мав виконати гальмівні маневри, але зв'язок з ним увірвався. Вважається, причиною аварії стало входження в атмосферу Марсу на 50 км нижче запланованої висоти через те, що ПЗ двигуна очікувало числа у метричній системі, а керівне ПЗ використовувало британську систему. Зворотним прикладом відмінного проектування є марсохід Curiosity, який працює суттєво довше запланованого.

Поняття інженерії ПЗ охоплює повний цикл розробки конкретного продукту, від постановки задачі до підтримки користувацької версії. На якість останньої впливають:

1. Відповідність плану розробки до фактичних вимог на продукцію.
2. Архітектура ПЗ.
3. Тестування.
4. Якість документації.
5. Допоміжні засоби розробки (системи контролю версій, KANBAN-boards тощо).

Примітка 5. Архітектура ПЗ

До архітектури ПЗ відносять широкий спектр характеристик програмного продукту. У першу чергу це структури даних, з яких формується програмний код (класи, структури тощо), формати вхідних і вихідних даних користувацький інтерфейс, набір залежностей від сторонніх продуктів. У широкому сенсі, це сукупність усіх аспектів побудови продукту включаючи з урахуванням його мети, умов експлуатації та принципів функціонування.

План розробки ПЗ визначає послідовність дій, які необхідно виконати від постановки задачі до отримання робочої версії продукту. Він визначається наявними часовими обмеженнями та вимогами на доступність функціоналу. Наприклад, ПЗ може бути доступним для використання за відсутності деяких корисних, але не принципових для основної його мети функцій.

У рамках проектування архітектури ПЗ визначається необхідний стек технологій та найбільш доцільний підхід до розробки. Так, використання високорівневих мов програмування спрощує та пришвидшує процес розробки, але може

негативно позначитись на швидкодії чи можливості розпаралелювання. Низькорівневі мови програмування (`C`, `Fortran`) можуть підтримувати спеціальні математичні бібліотеки, але разом з тим ускладнюють відстежування помилок не синтаксичного характеру. Використання сторонніх бібліотек часто спрощує процес розробки, але може поставити продукт у специфічну залежність від версій цих бібліотек та системного оточення.

Коментар 3. `MAGPAR`

Пакет мікромагнітних моделювань `MAGPAR` [8; 9] для аналізу текстур у феромагнітних наночастинках підтримувався до 2010 року. Як open-source продукт, він до сьогодні є важливим інструментом завдяки специфіці функціоналу, але інсталяція на сучасних комп'ютерах зазнає складнощів через лінкування із застарілими сторонніми бібліотеками.

Засоби автоматичного тестування та профайлінгу дозволяють спростити контроль за швидкістю і коректністю виконання обчислень за умови усталеності архітектури продукту. Розрізняють модульні тести (англ. unit tests), інтеграційні тести тощо. Для їх успішного використання програма має бути поділена на блоки, на вхід яких можна подати набір даних і порівняти відгук блоку з еталонним. Остання вимога накладає певні вимоги на проектування ПЗ у цілому.

Розробка ПЗ наукового спрямування має свої особливості. У першу чергу це стосунки замовник–розробник, у яких обидві ролі можуть збігатися принаймні у межах одного наукового колективу. Це може забезпечити глибше розуміння розробником предметної області продукту, але водночас користувачі можуть не мати достатньої кваліфікації для використання проміжних версій ПЗ. Колектив розробників може складатись як з одної (кількох) осіб (програми вузького спрямування, наприклад, Tracker [10]), так і нараховувати сотні людей різної кваліфікації, які робили свій внесок на різних етапах роботи (код аналізу даних Великого адронного колайдера налічує понад 600 учасників [11]).

1.4 Послідовна розробка

Послідовна розробка ПЗ є класичною схемою розробки програмної продукції [12]. Часто її називають водоспадною (waterfall) через чітку послідовність етапів з переходом на наступний після виконання усіх попередніх. Умовно, її можна поділити на наступні кроки, див. рис. 1.1:

1. формулювання вимог
2. проектування
3. розробка
4. тестування
5. підтримка

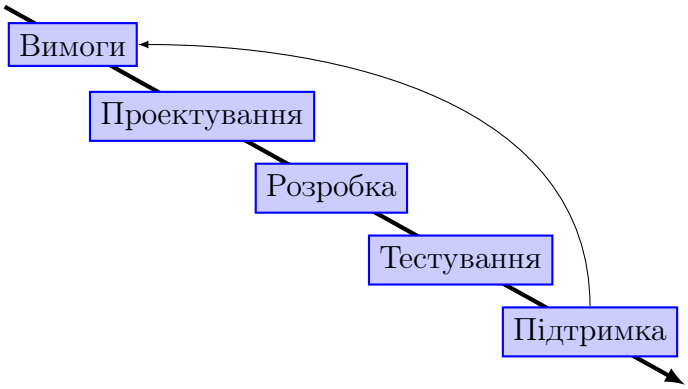


Рис. 1.1: Послідовна (waterfall) схема розробки. Перехід до наступного етапу здійснюється по завершенню усіх попередніх.

По їх виконанню цикл розробки повертається на початок: для подальшого розвитку визначаються нові вимоги, розробляється проект змін до архітектури ПЗ тощо. Дана схема добре підходить до розробки вузькоспеціалізованого ПЗ завдяки попередньому документуванню та формулюванню чітких вимок до кожного етапу роботи. Останнє також знижує поріг входження у розробку для нових учасників.

Недоліками послідовної розробки є необхідність повної уяви про кінцевий продукт з самого початку. У стадії розробки чи тестування будь-яка зміна функціоналу відкотить процес на етап проектування, що може призвести до суттєвого зростання вартості роботи.

Варто зазначити, що на сьогодні повноцінна уява замовника про кінцевий продукт зазвичай можлива за умов його вузької спеціалізації, що унеможливлює зміну функціоналу, або ж перенесення вже наявного продукту на нову платформу.

1.5 Поетапна розробка

З урахуванням специфіки використання сучасних програмних продуктів та їх ролі у бізнесі, більш зручною видається поетапна розробка ПЗ. Її головною рисою є випуск послідовності версій продукту, кожна з яких реалізує лише частину вимог, але дозволяє повноцінне тестування як концепції, так і функціоналу. Завдяки більшій гнучкості, поетапна розробка дозволяє оперативне внесення змін до архітектури продукту, але водночас вартість процесу та фінальний стан зазвичай не визначені. Поетапна розробка характеризується швидкими релізами оновленого продукту (еквівалентні кільком тижням роботи над кодом).

До поетапної розробки відносять такі відомі схеми організації роботи як Agile та екстремальне (XP) програмування. Обговорення різниці між конкретними моделями поетапної розробки виходить за межі цього посібника, але їх загальними рисами є розбиття процесу роботи над продуктом на етапи, кожен з яких складається з планування, роботи над кодом, тестування та впровадження, див.

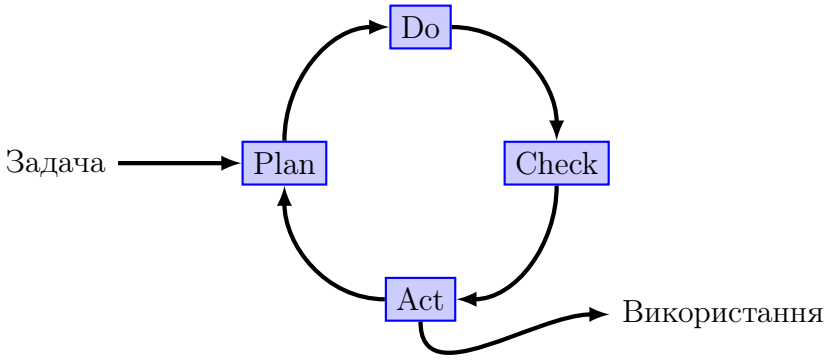


Рис. 1.2: Поетапна схема розробки дозволяє випуск продукту навіть якщо весь функціонал ще не імплементовано.

рис 1.2. Наприкінці кожного з них стає доступною нова версія продукту, на прикладі якого можна внести уточнення у його концепцію, та почати новий етап.

1.6 Проектування ПЗ

З вибором конкретної методики роботи над проектом пов'язане поняття складності в програмуванні. Так, особливості мов програмування чи фізичні можливості комп'ютерів на час використання коду вважаються менш важливими проблемами, аніж формулювання поставленої перед розробником задачі у термінах коду.

Варто зауважити наявність двох протилежних підходів до пріоритетів під час роботи над задачею: максимізація швидкості (зручності) розробки, або ж максимальна оптимізація програми[1]. У невеликих проектах із фіксованим набором вимог може бути доцільним закладати найбільш ефективні за часом виконання алгоритми та відповідні методи розробки (наприклад, використання низькорівневих мов програмування). У великих проектах із гнучким набором вимог, досить великою командою розробників, доцільнішим часто є зосередження на зручності опису задачі в термінах комп'ютера та оптимізація вузьких місць в рамках рефакторингу. У рамках категорії складності ПЗ це означає, що ідейна простота задачі дозволяє використати складніші програмні технології (наприклад, приносячи у жертву простоту коду чи зручність його модифікації під інші задачі), в той час, як у складних задачах є сенс знижувати складність проекту вибором більш гнучких методик розробки (застосування широко вживаних технологій, об'єктно-орієнтовного програмування тощо).

Процес проектування ПЗ можна розбити на декілька рівнів з різною абстракцією відносно кінцевого продукту. На верхньому рівні визначається мета продукту, середовище й умови його використання, технічні вимоги до стійкості, безпеки тощо. З технічної сторони, продукт можна представити як набір

[1] Див. також дискусію https://habr.com/ru/company/vdsina/blog/513436/.

підсистем різного ступеня взаємозалежності, які взаємодіють між собою. Підсистемою може виступати користувацький інтерфейс, модуль обрахунку конкретної фізичної взаємодії, модуль збереження даних, сторонні бібліотеки, які реалізують специфічний функціонал тощо.

У рамках підходу об'єктно-орієнтовного програмування (ООП), підсистеми реалізуються наборами класів. Класи, своєю чергою, визначаються своїми інтерфейсами (набори методів та полів даних, які належать конкретному класу, необхідно відрізняти від користувацького інтерфейсу!). На найнижчому рівні проектуються окремі методи.

Можна виділити два основних підходи до проектування ПЗ: згори–вниз і знизу–вгору в залежності від того, з чого починається проектування: загальних ідей або технічних особливостей імплементації. Підхід згори–вниз передбачає декомпозицію основної задачі на більш дрібні, що часто є легшою справою. Він дозволяє абстрагуватись від нюансів реалізації, які можуть технічно складними, та уточнювати різні аспекти архітектури продукту на кожній ітерації обговорення. Проектування згори–вниз може бути доцільним для уникнення занадто великого рівня абстракції системи та за наявності попереднього уявлення про необхідні технічні аспекти.

Примітка 6. Бажані характеристики типового проекту

1. Зрозумілі технології та стандартні методики
2. Прості нотації та наочний код
3. Масштабування
4. Слабка зв'язність модулів проекту
5. Простота супроводу

Під масштабуванням розуміють розповсюдження концепції на різні масштаби: можливість обробки даних розміром в мега- та гігабайти, запуск паралельної версії одним та десятками процесів тощо в залежності від специфіки конкретної задачі.

Деякі критичні аспекти може бути важко оцінити заздалегідь, наприклад, швидкодію тої чи іншої сторонньої бібліотеки для конкретного обсягу даних. Це вирішується експериментальним прототипуванням із написанням прототипу робочої системи.

Розглянемо потенційний стек технологій та особливості архітектури ПЗ наукового спрямування на прикладі пакету спін-ґраткових моделювань Spirit [13; 14]. Програма дозволяє аналізувати статичні стани та динаміку класичних векторів спінів $\boldsymbol{m}_i$, мінімальна модель взаємодії яких включає Гайзенберґівський обмін між найближчими сусідами та анізотропію:

$$\mathscr{H} = -J \sum_{ij} \boldsymbol{m}_i \cdot \boldsymbol{m}_j - K \sum_i (\boldsymbol{m} \cdot \boldsymbol{e}_{\text{A}})^2, \tag{1.6}$$

де J і K — обмінний інтеграл та коефіцієнт анізотропії, відповідно, $\boldsymbol{e}_{\text{A}}$ — вісь анізотропії.

Програма є крос-платформеною і може запускатись під Windows, Linux та OSX. Доступність та уніфікація графічного інтерфейсу забезпечується бібліотекою Qt. Окрім цього, доступний веб-застосунок, який дозволяє запускати моделювання з браузера. Вхідні дані програми можуть задаватись як з графічного інтерфейсу, так і за допомогою скриптів `Python`. Для збереження результатів моделювань (векторне поле) застосовується формат OVF для пакета мікромагнітних моделювань OOMMF [4; 5] («класична» програма, яка задала стандарт в обчислювальному мікромагнетизмі), що спрощує міграцію між обома програмами. Обчислювальне ядро може запускатись як на CPU (центральний процесор, бібліотека OpenMP), так і на GPU (відеокарти, бібліотека CUDA). Вибір того чи іншого варіанту ядра здійснюється на етапі компіляції за допомогою ключів до утіліти `cmake`. GPU-обрахунки здійснюються із застосуванням типу `float` замість `double`, щоб уникнути сповільнення, пов'язаного з використанням чисел із подвійною точністю на відеокартах.

Сучасний рівень розвитку комп'ютерної техніки дозволяє дещо знизити вимоги на швидкодію конкретної реалізації (наприклад, застосовуючи розпаралелювання на відеокартах) та зосередитись на якості коду та зручності його модифікації під нові вимоги. Це диктує певні рекомендації технічного характеру до архітектури продукту. Зокрема, корисно підтримувати слабку зв'язність між складовими частинами продукту та малу кількість звертань з одного класу до іншого. Проблема масштабування може бути актуальною для задач, які потребують операцій з великою кількістю змінних, оскільки, наприклад, ефективність розпаралелювання суттєво залежить від технології, на яку опирається проект. Так, оперативна пам'ять відеокарт все ще суттєво менша за доступні ресурси окремих обчислювальних вузлів (десятки ГБ проти сотень ГБ на момент написання), в той час, як для MPI ефективніше синхронізувати між процесами один великий масив даних, аніж багато менших.

1.7 Проектування класів

Технічно, клас представляє набір певних полів, у яких можуть зберігатись дані різних типів (включаючи об'єкти інших класів) та методів, які виконують операції над ними. Можна говорити, що конкретний набір значень полів становить собою конкретний внутрішній стан об'єкту цього класу, а методи дозволяють змінювати внутрішній стан у відповідь на зовнішні чинники та інформувати інші об'єкти про зміну цього внутрішнього стану. Коректно спроектований клас можна означити як математичну модель деякої сутності реального світу (тобто, не об'єкту в розумінні ООП — юніта на ігровому полі, таблиці даних, параграфу тексту, зображення тощо) разом із визначеним набором операцій над нею. Такі моделі називають абстрактними типами даних (АТД, Abstract Data Type, ADT).

Примітка 7. Приклад абстрактного типу даних

Черга — це колекція об'єктів яка дозволяє додавати та видаляти по-одному об'єкти за правилом: перший доданий у чергу об'єкт, буде видалений з неї першим (first in – first out, FIFO). Особливостями колекції з N елементів є стала складність додавання та видалення об'єктів $O(1)$ та лінійна складність пошуку $O(N)$ як в кращому, так і гіршому випадках. Для програміста-користувача черги типовий публічний інтерфейс класу дозволяє операції `push` (додати об'єкт), `pop` (вийняти об'єкт) та `search` (перевірити наявність об'єкту). Водночас внутрішня реалізація черги прихована — це може бути, наприклад, масив чи зв'язаний список, які визначать фактичну швидкодію та вимоги на оперативну пам'ять при роботі з конкретною реалізацією черги.

Коментар 4. Приклад АТД і не-АТД підходу

Не абстрактний тип

```
myFont.size = 10
myFont.style = myFont.style or bold
```

У випадку не-АТД підходу стан об'єкту задається алгоритмічно.

Абстрактний тип

```
myFont.setSize(10)
myFont.setBold()
myFont.unsetItalic()
```

АТД-підхід можна схарактеризувати як більш наближений до людської мови (human-readable) із набором синтаксично зрозумілих, самодостатніх команд.

Проектування класів можна розбити на чотири етапи:

1. Визначення співвідношень з об'єктами реального світу.
2. Проектування абстракцій.
3. Інкапсуляція.
4. Визначення областей імовірних змін.

Розглянемо кожен з них детальніше.

При розбитті задачі на класи та проектуванні інтерфейсу кожного класу доцільним є максимальне звуження області відповідальності класу та його конкретизація. Так, клас, який описує таблицю з комірками, які містять текст,

скоріш за все, не повинен відповідати за форматування тексту всередині конкретної комірки. Такий функціонал варто винести у клас комірки таблиці. Таким чином, визначається набір об'єктів, з якими варто оперувати, їх атрибути та можливі дії над об'єктами. Ілюстрацією цього підходу можуть слугувати класи `QAbstractItemModel`[2] та `QModelIndex`[3], які призначені для опису довільних табличних наборів даних та окремих елементів цих наборів, відповідно, у бібліотеці Qt.

Наступним кроком є визначення спорідненості об'єктів та характеристик, які притаманні лише одному класу. Спільні характеристики, які притаманні групі об'єктів, але не є вирішальними для означення кожного з них варто виносити у батьківські (абстрактні) класи. Так, оперуючи об'єктами «орел» і «серпокрилець», наявність пір'я, можливість польоту та полювання можна винести у батьківський клас `Bird` як абстрактні методи, у той час, як їх конкретні реалізації будуть описуватись у конкретних класах `Eagle` та `Swift`. За необхідності мати об'єкт типу «муха», може бути доцільним виділення окремого абстрактного класу (інтерфейсу) для всього, що може літати `Flyable`, що дозволить оперувати колекцію об'єктів зі здатністю літати.

Реалізація принципу інкапсуляції вимагає визначити, що інші класи можуть знати про даний клас, іншими словами, які методи для зміни внутрішнього стану є публічними, та які поля можуть задаватись напряму. Розглядаючи карту міста як об'єкт, метод прокладання маршруту між точками А і Б може бути публічним, в той час, як декілька алгоритмів пошуку залишатимуться приватними методами, вибір між викликом яких робиться на основі поточної величини графу міста. Полем, яке може змінюватись напряму, може бути назва карти. Нагадаємо, що коректним підходом до зміни будь-яких полів є написання так званих сеттерів і геттерів (або їх аналогів у конкретній мові програмування) — методів, через які відбувається доступ до значення приватного поля. Зокрема, сеттер має виконувати перевірку коректності значення, яке програміст-користувач класу бажає занести у поле, та викидати виключення за неможливості виконати присвоєння.

Наостанок, корисним є інспекція архітектури в цілому на предмет можливих змін у майбутньому та об'єму змін, які при цьому знадобляться. На цьому етапі може виявитись корисним виділення додаткових абстрактних класів або реорганізація вже спроектованих.

1.8 Задачі

№1. Фур'є прямокутника. Задано прямокутний сигнал $s(t)$ з періодом T_0 і амплітудою A.

1. Аналітично розкладіть $s(t)$ на гармоніки та проілюструйте збіжність Фур'є–апроксимації $S(t, n)$ до $s(t)$ в залежності від кількості гармонік n.

[2] https://doc.qt.io/qt-5/qabstractitemmodel.html
[3] https://doc.qt.io/qt-5/qmodelindex.html

2. Виокремте програмний інтерфейс (метод, клас), який дозволить отримувати апроксимацію $S(t, n)$ для заданої кількості гармонік n та документуйте його відповідно до стандартів обраної мови програмування.

3. Проаналізуйте та опишіть технічні обмеження $S(t, n)$ (наприклад, ефект Гіббса та неоднорідність сигналу на плато).

№2. Проектування програми. Опираючись на тему вашої бакалаврської роботи визначте корисний програмний продукт і опишіть вимоги до нього. Опишіть плани роботи над ним у рамках послідовної та поетапної моделей розробки.
№3. Оцініть складність поняття архітектури ПЗ. Що у нього входить і як ця складність впливає на розробку сучасного ПЗ?
№4. Схарактеризуйте особливості послідовної розробки ПЗ.
№5. Схарактеризуйте особливості поетапної розробки ПЗ.
№6. Перелічить переваги та недоліки архітектури Spirit.

1.9 Take home message

1. Спрощення процесу розробки ПЗ завдяки плануванню та вибору правильної моделі.

2. Послідовна розробка: класична модель з кроками від формулювання вимог до підтримки готового продукту для проектів із стабільними концепцією та вимогами.

3. Поетапна розробка: сучасна модель для продуктів з нечіткою концепцією.

4. Розробка архітектури ПЗ згори–вниз і знизу–вгору як способи спрощення формулювання задачі у термінах програмування.

5. Опис архітектури ПЗ за допомогою користувацьких історій, діаграм використання варіантів, класів та послідовностей.

Посилання

1. magnum.fe: A micromagnetic finite-element simulation code based on FEniCS / C. Abert, L. Exl, F. Bruckner, A. Drews, D. Suess // Journal of Magnetism and Magnetic Materials. — 2013. — Листоп. — Т. 345. — С. 29—35. — DOI: 10.1016/j.jmmm.2013.05.051. — URL: https://doi.org/10.1016/j.jmmm.2013.05.051.
2. *Fangohr H.*, *Albert M.*, *Franchin M.* Nmag micromagnetic simulation tool // Proceedings of the International Workshop on Software Engineering for Science - SE4Science '16. — Association for Computing Machinery (ACM), 2016. — DOI: 10.1145/2897676.2897677. — URL: https://doi.org/10.1145%2F2897676.2897677.
3. Nmag micromagnetic simulation package. — URL: https://nmag-project.github.io/.

4. The Object Oriented MicroMagnetic Framework. — URL: http://math.nist.gov/oommf/ ; Developed by M. J. Donahue and D. Porter mainly, from NIST.
5. *Donahue M. J.*, *Porter D. G.* OOMMF User's Guide, Version 1.0 : тех. звіт. / Interagency Report NISTIR 6376. — 1999. — URL: http://math.nist.gov/~MDonahue/pubs/abstracts.html#Donahue199909.
6. *Beuckelmann M.* Boosting numpy: Why BLAS Matters. — 2017. — URL: http://markus-beuckelmann.de/blog/boosting-numpy-blas.html.
7. *Zadka M.* Python mailing lists. — 2000. — URL: https://mail.python.org/pipermail/python-list/2000-February/036421.html.
8. MAGPAR finite element micromagnetics package. — URL: http://www.magpar.net ; Developed by Werner Scholz.
9. Scalable parallel micromagnetic solvers for magnetic nanostructures / W. Scholz, J. Fidler, T. Schrefl, D. Suess, R. Dittrich, H. Forster, V. Tsiantos // Computational Materials Science. — 2003. — Т. 28, вип. 2, № 2. — С. 366—383. — ISSN 0927-0256. — DOI: 10.1016/S0927-0256(03)00119-8. — URL: https://doi.org/10.1016/S0927-0256(03)00119-8 ; Proceedings of the Symposium on Software Development for Process and Materials Design.
10. *Brown D.*, *Hanson R.*, *Christian W.* Tracker VIdeo Analysis and Modelling Tool for Physics Education. — URL: https://physlets.org/tracker/.
11. *Merali Z.* Computational science: ...Error // Nature. — 2010. — Т. 467, вип. 7317, № 7317. — С. 775—777. — DOI: 10.1038/467775a. — URL: https://doi.org/10.1038/467775a.
12. *Royce W. W.* Managing the development of large software systems: concepts and techniques // Proceedings of the 9th international conference on Software Engineering. — 1987. — С. 328—338.
13. Spirit: Multifunctional framework for atomistic spin simulations / G. P. Müller, M. Hoffmann, C. Dißelkamp, D. Schürhoff, S. Mavros, M. Sallermann, N. S. Kiselev, H. Jónsson, S. Blügel // Physical Review B. — 2019. — Т. 99, вип. 22, № 22. — С. 224414. — DOI: 10.1103/PhysRevB.99.224414. — URL: https://link.aps.org/doi/10.1103/PhysRevB.99.224414.
14. Spirit – Open source, cross-platform spin simulation framework. — URL: https://juspin.de/spirit/.

Література до розділу

15. *Орлов С. А.* Программная инженерия. Учебник для вузов. — 5-е вид. — СПб. : Питер, 2016. — ISBN 978-5-496-01917-0.
16. *Макконнелл С.* Совершенный код. — Русская Редакция, Microsoft Press, 2017. — ISBN 978-5-7502-0064-1.
17. Приемы объектно-ориентированного программирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. — СПб. : Питер, 2011.

18. *Фаулер М.* Рефакторинг: улучшение существующего кода. — СПб. : Символ-Плюс, 2004.

19. *Bell D.* UML Basics: Getting started using UML for visual modeling of computer programs. — IBM. — URL: https://developer.ibm.com/technologies/web-development/series/uml-basics/.

20. Refactoring.Guru. Патерни проектування. —. — URL: https://refactoring.guru/uk/design-patterns.

21. Azure Application Architecture Guide. —. — URL: https://docs.microsoft.com/en-us/azure/architecture/guide/.

Розділ 2

Шаблони проектування

2.1 Опис задач і архітектури ПЗ

Для наочного пояснення що і як повинно відбуватись, а також, з чого складається дане ПЗ використовується діаграмна техніка [1]. Одними з найбільш важливих (часто вживаних) діаграм є діаграми варіантів використання, класів та послідовностей.

Діаграма використання варіантів (Use-case diagram) оперує поняттями акторів і дій, які вони можуть виконати в рамках можливостей ПЗ. Актором може бути людина або зовнішній сервіс, які виконують певні операції. Операції можуть охоплювати кілька акторів. Схематично, на діаграмі проставляється кілька фігурок-акторів, між якими розміщуються операції. Операції з'єднуються лініями із залученими акторами, див. рис. 2.1. Наприклад, для системи моніторингу акторами можуть бути оператор, технік і віддалені датчики. Операція перегляду показів залучає оператора і датчики, в той час, як технік залучається до операції виклику датчиком за умови критичної зміни вимірюваної величини.

Примітка 8. Користувацькі історії (User stories)

Із діаграмою використання варіантів тісно пов'язані так звані «користувацькі історії»: набори коротких висловлювань про можливості користувачів ПЗ. Наприклад, для онлайн-магазину можуть бути актуальними наступні:

- Користувач хоче сортувати товари за спаданням чи зростанням вартості.
- Користувач хоче відновлювати пароль від кабінету за допомогою власної електронної пошти.
- Оператор хоче бачити список останніх покупок користувача із заданим логіном.

Діаграми класів (Class diagrams) мають на меті компактно висвітлити інтерфейс та взаємозв'язки між класами. Кожен клас (інтерфейс) представлено прямокутником, розбитим на 3 ділянки: у верхній записано назву класу, ниж-

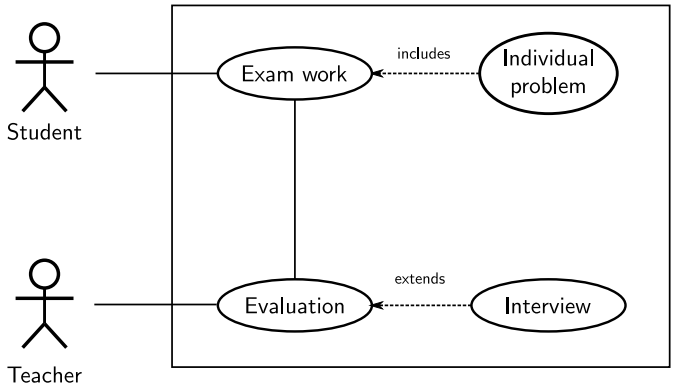
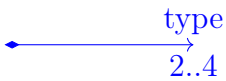


Рис. 2.1: Приклад use-case діаграми для екзамену. Студент і вчитель асоціюються з екзаменаційною роботою та оцінюванням, відповідно. Екзаменаційна робота та оцінювання пов'язані між собою. На додачу, екзаменаційна робота містить окремі індивідуальні задачі, а оцінювання може бути розширене співбесідою.

че перелічено поля класу із зазначенням типів (можливо, видимості). У нижній частині прямокутника зазначено методи класу з аргументами та типами, що повертаються. Статичні члени класу (ті, які не залежать від конкретного екземпляра й можуть бути викликані незалежно) позначаються нижнім підкресленням. Абстрактні методи позначаються курсивом, а рядок з приватним полем чи методом починається з мінуса.

Зв'язки між класами позначаються лініями зі стрілками та підписами:

- B ——▷ A Клас `B` є дочірнім до класу `A`.
- B - - - - - - ▷ A Клас `B` є імплементацією інтерфейсу `A`.
- 0..* role2 / role1 0..1 Двоспрямована асоціація, у якій позначено ролі та множинності. Перший клас (ліворуч) виконує роль `role1` для другого, при чому другий може мати від нуля до нескінченності об'єктів з цією роллю. Другий клас (праворуч) виконує роль `role2` для першого, при чому перший може мати лише один асоційований з ним об'єкт другого класу, або не мати таких взагалі.
- role2 →, 0..1 Односпрямована асоціація. Так само як для двоспрямованої, але про зв'язок відомо лише класу ліворуч.
- type ◇→, 2..4 Асоціація агрегації. Клас ліворуч агрегує від двох до чотирьох класів праворуч, приймаючи їх як деякий `type`. Загалом же, ці об'єкти цих класів існують незалежно один від одного.

- 

Асоціація композиції. Клас ліворуч агрегує від двох до чотирьох класів праворуч, при чому цикл життя класів праворуч підкорюється циклу життя головного класу.

Діаграма послідовностей пояснює послідовність викликів різних об'єктів та суть взаємодії між ними. Типово, вона представляється у вигляді горизонтального рядка з назвами об'єктів, від яких вниз розгортається часова розмірність кожного з них. Між вертикальними лініями від об'єктів позначаються виклики та дані, що повертаються. Діаграма послідовностей є ООП-адаптованим варіантом алгоритмічних блок-схем (н-д, мова блок-схем ДРАКОН [2]).

2.2 Поняття шаблону проектування

Поняття шаблону (паттерну) проектування прийшло до програмування [3; 4] з книги Крістофера Александера «Мова шаблонів. Міста. Будівлі. Будівництво» [5], у якій узагальнювались типові архітектурні рішення, які могли комбінуватись та застосовуватись у різноманітних умовах. У контексті розробки ПЗ шаблони проектування слугують типовими методами вирішення типових підзадач, які виникають при описі взаємодії чи поведінки об'єктів. Прикладом шаблону проектування може слугувати ієрархічна організація однотипних об'єктів по зразку файлової системи: каталоги є рівноправними об'єктами, але можуть бути вкладені один в інший. Так само організуються складові текстових документів тощо.

Примітка 9. Метафора шаблонів проектування

Шаблон проектування виступає, радше, не у якості будівельної плити для будинку, а ідеї як вона повинна розміщуватись. Наприклад, двері мають бути у стіні, а не стелі, та виходити у інше приміщення, а не у порожній простір на 10-му поверсі.

Зручність використання шаблонів полягає у тому, що вони надають розробникам як стандартизований спільний словник, так і перевірені зразки розв'язання типових задач. Але слід враховувати, що типова реалізація шаблону суттєво залежить від мови програмування. Так, рекомендації щодо інкапсуляції можна повноцінно виконати у C++ або Java, але не у Python, де видимість поля трактується як рекомендація, а не жорстке обмеження, яке припиняє виконання (компіляцію) при порушенні.

Розрізняють наступні шаблони проектування:

1. ідіоми,
2. породжувальні,
3. структурні,
4. поведінкові,

5. архітектурні.

Окрім них існує велика кількість шаблонів, специфічних для конкретних предметних областей (великі дані, розробка ігор тощо).

Ідіоми — це прості шаблони, які орієнтовані на розв'язання технічних задач. Типовими ідіомами є рекурсія або спосіб перемішування елементів масиву випадковим чином. Важливою ідіомою ООП є пізня ініціалізація: якщо повноцінне створення об'єкту вимагає багато ресурсів, його варто ініціалізувати лише за першого звертання. Зокрема, вона ж реалізується оператором `:=` у Wolfram Mathematica.

Породжувальні, структурні та поведінкові шаблони дозволяють розв'язувати задачі створення складних об'єктів, визначати їх взаємодію та поведінку. Надалі будуть описані деякі з них. За повним описом радимо звернутись до літератури з курсу [3; 4; 6].

Коментар 5. Качкосимулятор (за [ ])

Деяка фірма розроблює симулятор качок. У першій версії симулятору підтримуються візуалізація та озвучка різних качок: домашньої, крохаля та гумової, які описуються базовим абстрактним класом `Duck` з віртуальними методами `Display()` і `Sound()`, реалізованими у конкретних класах. У новій версії вирішили додати можливість польоту. Якщо додати новий абстрактний метод `fly()` у батьківський клас, то у гумової качки його реалізація буде просто порожнім методом. Це не дуже красиво з огляду на те, що читання викликів методів не дозволить збагнути, що метод `rubberDuck.fly()` нічого не робить. Інша проблема виникне, якщо знадобиться реалізувати летючу гумову качку-дрона створивши для неї окремий клас з потенційним дублюванням коду.

Ефективним шляхом запобігання вищеописаних проблем може бути виокремлення типових поведінок. Наприклад, інтерфейси `QuackBehavior` та `FlyBehavior` дозволяють реалізувати різні типи звуків (`Squeck`, `Mute`) та руху в повітрі (`Wings`, `Jetpack`) які відв'язані від об'єкту, який їх використовує. Таким чином, утворення конкретного типу качки перетворюється на конструювання глобальної поведінки об'єкту на льоту. Такий підхід є реалізацією поведінкового шаблону «Стратегія», див. рис. 2.2.

2.3 Породжувальні шаблони

Породжувальні шаблони надають абстракцію процесу ініціалізації чи породження об'єктів. Користь від їх використання проявляється у випадку складного алгоритму створення об'єкту, чи набору фундаментальних поведінок, з яких можна зібрати поведінку конкретного об'єкта.

Основні Породжувальні шаблони:

1. Одинак

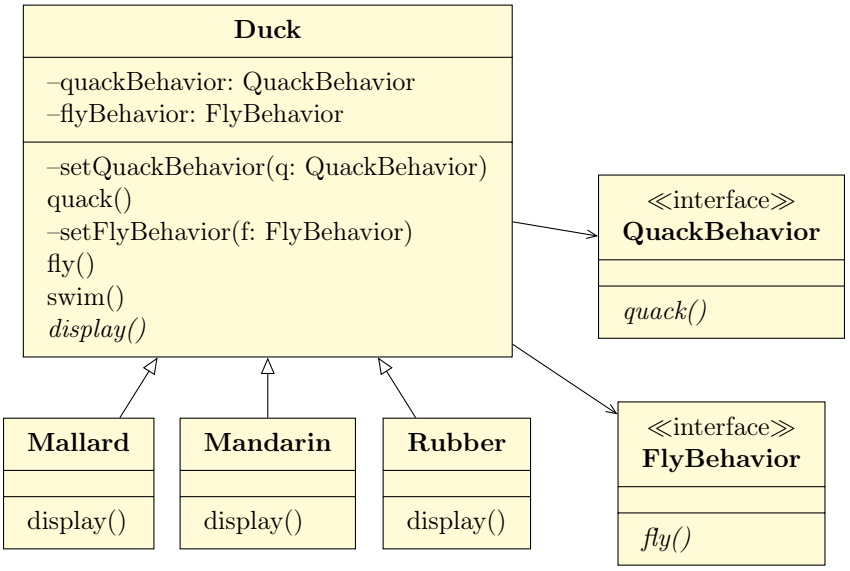


Рис. 2.2: Реалізація шаблону «Стратегія» для симулятора качок [3]. Поведінка конкретного об'єкта качки визначається у ході роботи програми в залежності від того, яка реалізація звукового супроводу `QuackBehavior` чи способу польоту `FlyBehavior` призначена цьому об'єкту.

2. Мультитон
3. Пул об'єктів
4. Прототип
5. Фабричний метод
6. Абстрактна фабрика
7. Будівельник

2.3.1 Одинак (Singleton)

Шаблон гарантує наявність у програмі виключно одного об'єкта класу і надає глобальну точку доступу до нього. Звичайна глобальна змінна не є досить гнучким рішенням (також, з урахуванням стандартних рекомендацій щодо уникнення глобальних змінних у коді), оскільки це не забороняє повторну ініціалізацію та не контролює доступ до об'єкта. Типовим прикладом сутності, що вимагає одинака, може бути база даних: множинна ініціалізація локальної бази чи сполучень з віддаленою може бути причиною суттєвої витрати ресурсів та затримок у роботі ПЗ.

Реалізація одинака полягає в інкапсуляції об'єкту класу як приватного поля свого ж класу із доступом до нього через статичний метод `getInstance()`, див. рис. 2.3. Таким чином, можна перевіряти чи відбулася ініціалізація об'єкту під

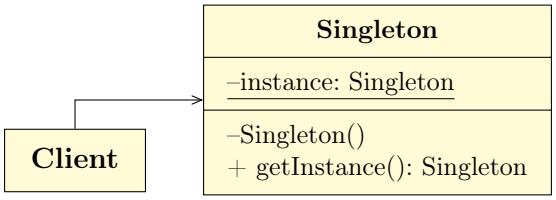


Рис. 2.3: Одинак забезпечує існування лише одного екземпляру класу та надає контроль доступу до нього.

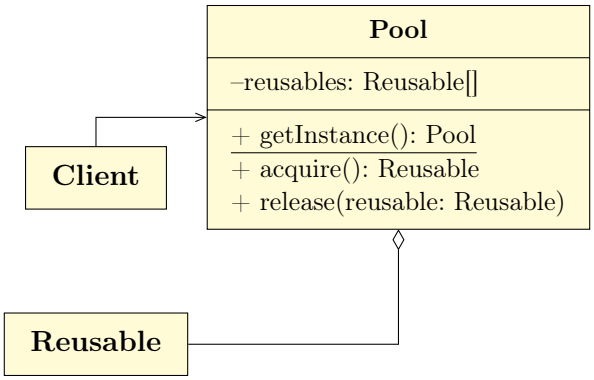


Рис. 2.4: Пул об'єктів кешує об'кти з високою вартістю створення.

час його запиту через `getInstance()` і створювати за необхідності. Останнє також дозволяє реалізовувати пізню ініціалізацію.

Серед недоліків одинака необхідно відзначити подвійну відповідальність класу як за єдиність, так і глобальність точки доступу. Специфіка реалізації шаблону суттєво залежить від мови програмування і може вимагати спеціальних трюків у випадку паралельних обчислень.

Варіацією одинака є мультитон (Multiton), в якому можна зберігати кілька екземплярів класу з вибором по ключу (номер в масиві чи ключ в словнику).

2.3.2 Пул об'єктів (Object Pool)

Наступним розвитком ідеї одинака і мультитону є пул об'єктів, що реалізує концепцію кешування об'єктів із високою вартістю створення, див. рис. 2.4. Технічно, він може реалізуватись через них. Ідеєю шаблону є зберігання об'єкту після його утворення і повертання цього екземпляру у випадку необхідності замість створення нового. Прикладом реалізації пулу об'єктів є каршеринг, кешування результатів запитів до бази даних (JDBC у Java) тощо.

На прикладі пулу об'єктів розглянемо концепцію задачі на реалізацію шаблону проектування.

```
1 from random import randint
2
3 # Цей клас репрезентує об'єкт з великою вартістю створення.
4 # Технічно, він лише зберігає якесь значення val і вміє
```

```
# його повертати, що достатньо для навчальної задачі.
class Reusable:
  def __init__(self, val):
    self.__val = val
  def __str__(self):
    return str(self.__val)

# Реалізація пулу об'єктів
class Pool:
  def __init__(self, num):
    # При створенні, пул імітує заповнення себе об'єктами, де у якості
    # високу вартість генерації репрезентує генератор випадкових чисел
    self.__reusables = [Reusable(randint(0,100)) for i in range(num)]

  def release(self):
    # Пул повертає об'єкт (тут може бути запит конкретного об'єкта за властивістю)
    return self.__reusables.pop()

  def aquire(self, r):
    # Об'єкт повертається до пулу після використання
    self.__reusables.append(r)

pool = Pool(3)
p = pool.release()
print(p)
p = pool.release()
print(p)
pool.aquire(p)
p = pool.release()
print(p)
```

Лістинг 2.1: Імплементація пулу об'єктів

2.3.3 Абстрактна фабрика (Abstract Factory)

Примітка 10. Ідіома «Фабрика»

Під фабрикою розуміють випуск типових наборів об'єктів зі спільними характеристиками. Наприклад, фабрика А може виробляти набори меблів для їдальні (столи і стільці у єдиному дизайнерському рішенні), а фабрика Б — сервізи (чашки, блюдця, чайники).

Є задачі, у яких відомі інтерфейси для створення (генерації) об'єктів або цілих сімейств об'єктів та умови початку цієї роботи. Водночас що саме і як створюється, може бути описано де-інде. Ці сімейства об'єктів повинні мати спільну суть, в той час, як реалізація може бути повністю незалежна. Відповідними прикладами виступають елементи вікон у WinApi та Qt.

Шаблон абстрактної фабрики оперує інтерфейсом фабрики, у якій перелічені методи створення конкретних об'єктів, див. рис. 2.5. Конкретні фабрики надають імплементації цих методів. Ці фабрики можуть повертати об'єкти прин-

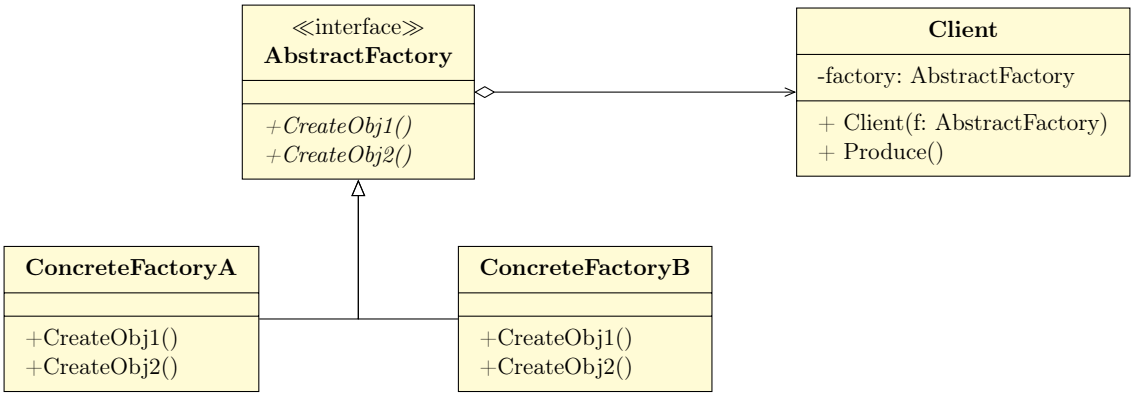


Рис. 2.5: Абстрактні фабрики дозволяють створювати сімейства об'єктів за заданим інтерфейсом без прив'язки до конкретних класів.

ципово різної суті, з єдиною спільною рисою, що полягає у наборі цих об'єктів. Клієнт фабрики ініціалізується конкретною фабрикою та через виклик методу `Produce()` починає генерацію об'єктів шляхом виклику методів `CreateObj1()`, `CreateObj2()` і так далі.

Прикладом роботи фабрики може бути генерація юнітів різних армій у грі. Перевагою абстрактної фабрики є простота розширення сімейства продуктів та абстрагування клієнту від конкретних класів продуктів — він може продукувати будь-який за заданим шаблоном. Одночасно зміна самого набору продуктів є технічно важкою задачею, що потребує зміни усіх наявних класів конкретних фабрик.

2.3.4 Будівельник (Builder)

Схожою на абстрактну фабрику, але протилежною по суті є необхідність створення поодиноких складних об'єктів за заданою процедурою. Таким складним об'єктом може бути книга, автомобіль, користувацький інтерфейс програми. Шаблон «Будівельник» дозволяє реалізувати абстракцію покрокового створення об'єктів різної природи, які мають спільний алгоритм створення. Так, обидва, об'єкт електронного мікроскопа і документація до нього складаються з великої кількості дрібніших частин зі специфічними властивостями, але процес створення віртуальної моделі обох може бути описаний однаково оскільки кожному елементу мікроскопу ставиться у відповідність свій розділ документації й різним є лише те, як імплементована генерація кожного з них. Так само може працювати експорт бази даних у різні формати. Іншими словами, разом із можливістю покрокового створення продуктів, будівельник ізолює код створення продукту від бізнес-логіки ПЗ.

Шаблон створюється за допомогою інтерфейсу `Builder`, у якому перераховані кроки побудови об'єкту, див. рис. 2.6. Конкретні будівельники імплементують ці кроки та повертають готовий продукт. Керування збиранням виконується через клас `Director`, який ініціалізується конкретним будівельником та у методі `produce()` має інформацію про послідовність виклику кроків `BuildPartA()`,

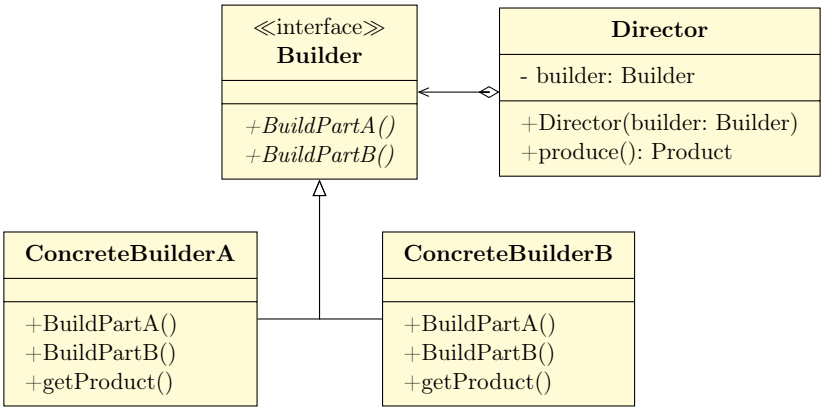


Рис. 2.6: Шаблон «Будівельник» реалізує покрокове створення об'єкту складної природи за заданим шаблоном.

`BuildPartB()` і так далі.

Коментар 6. Збірник питань з курсу «Науковий образ світу»

Викладаючи курс «Науковий образ світу», автор брав участь у підготовці кількох збірників тестових питань в колективі з п'яти співавторів. Щоб забезпечити зручну колективну роботу над виданням посібника із можливістю одночасного заповнення бази Moodle, було обрано наступний підхід.

Питання та відповіді представляли окремі текстові файли, які колекціонувалися в спільний Git-репозиторій. Вони завантажувалися Python-скриптом у вигляді об'єктів з полями «тіло питання» і списком відповідей (кожна відповідь теж являє собою об'єкт з полем, що означає правильність або неправильність цієї відповіді).

Як збірник питань у вигляді книжки (формат LaTeX), так і файли імпорту Moodle у форматі GIFT потребують спеціальної обробки, що дозволяє реалізувати шаблон «Будівельник» для реалізації стандартних кроків будівництва фінального об'єкта. Конвертація питання у необхідний текстовий формат (LaTeXчи GIFT) реалізується шаблоном «Відвідувач» (поведінковий шаблон, що обговорюватиметься далі), що дозволяє виокремити логіку представлення питання у тому чи іншому вигляді з класу питання взагалі.

2.4 Структурні шаблони

Задачею структурних шаблонів є організація сукупностей класів для отримання необхідної структури. Прикладами мети композиції класів може бути спрощення їх інтерфейсу для конкретної задачі або адаптація бібліотечного ін-

терфейсу з приховуванням функцій, які не використовуються. Основними структурними шаблонами є

1. Декоратор
2. Адаптер
3. Фасад
4. Заступник
5. Міст
6. Компонувальник
7. Полегшувач

Часто, мова йде про вже наявні класи, які неможливо (належать до скомпільованої бібліотеки), або не бажано (архітектура ПЗ вимагає їх мати саме такими з деяких причин) змінювати.

2.4.1 Декоратор

Шаблон «Декоратор» (Decorator) має на меті динамічну спеціалізацію функціоналу для уникнення породження підкласів. Це схоже на шаблон «Стратегія», але без мети принципової модифікації поведінки об'єктів. Прикладом застосування декораторів є додавання властивостей масштабування чи прокрутки у візуальному інтерфейсі програм: суть роботи класу залишається тою ж самою (показувати зображення), але змінюються деталі як саме це може бути виконано. Ту саму ідею реалізують фільтри інстаграму та інших застосунків модифікації зображень чи відео на льоту. Як і у випадку стратегії, це дозволяє значно гнучкіше модифікувати поведінку класів та робити просту композицію з необхідних елементів у ході виконання програми, але з іншого боку, складність відлагоджування програми за великої кількості декораторів може бути дуже високою. Також варто враховувати, що декоратор виступає лише функціональною обгорткою, а не самим елементом.

Проектування декораторів виконується імплементацією інтерфейсу, який визначає конкретні дії базового класу та потенційних декораторів, див. рис. 2.7. Абстрактний клас декоратора `Decorator` та основного класу `MyClass` реалізують інтерфейс `Actions`, у якому представлено те, що може бути модифіковано декоратором. Конкретні імплементації декоратору, такі як `DecoratorA`, інкапсулюють в собі об'єкт, що відповідає інтерфейсу `Actions` (об'єкт класу `MyClass` з представлених на діаграмі). Коли ж відбувається виклик дії `ActionA()`, то відгук базового класу модифікується перед тим, як бути повернутим користувачу класу. Наприклад, якщо базовий клас на виклик `ActionA()` повертає зображення, декоратор може застосувати фільтр до нього.

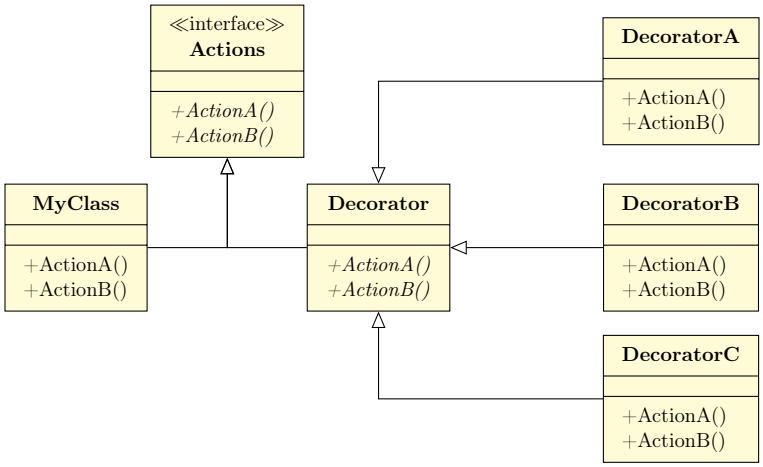


Рис. 2.7: Шаблон «Декоратор»: є базовий клас `MyClass` та декоратори `DecoratorA`, `DecoratorB`, `DecoratorC`, які реалізують конкретні особливості поведінки.

2.4.2 Адаптер, фасад, заступник і міст

Шаблони «Адаптер» (Adapter), «Фасад» (Façade), «Заступник» (Proxy) і «Міст» (Bridge) об'єднані спільною ідеєю заміни інтерфейсу цільового класу іншою, яка більш зручна для конкретного застосування. Наприклад, вони знаходять застосування в адаптації сторонніх бібліотек, legacy коду. В усіх випадках цільовий клас або класи просто приховуються за новим інтерфейсом, який керує їх взаємодією для виконання команд користувача класу.

- Ідеєю адаптеру є проста заміна інтерфейсу на більш доречний. Наприклад, якщо виклик трьох методів цільового класу завжди виконується у певній послідовності та оброблюється наперед заданим чином, адаптер може інкапсулювати цей процес за викликом лише одного методу. Він типово використовується для вже наявного коду.

- Фасад дозволяє зменшити зв'язність системи приховавши підсистему класів за певним «фасадом». Прикладом фасаду в житті може бути кол-центр магазину або банку, який отримавши запит користувача реалізує через себе його взаємодію зі складною внутрішньою структурою організації. Цей шаблон дозволяє ізолювати клієнта від складних компонент системи та, потенційно, непомітно для клієнта їх оновлювати. Але невдало спроектований фасад може навпаки, все замкнути на себе і збільшити зв'язність системи.

- Заступник, використовується для контролю над зверненнями до об'єкта, наприклад, для гарантії виконання певних дій до чи після звернення, примусової ініціалізації тощо. Прикладами заступників є «розумні» вказівники у C++ та плейсхолдери зображень, які їх заміняють під час завантаження. Заступник дозволяє реалізовувати контроль, незалежний від циклу

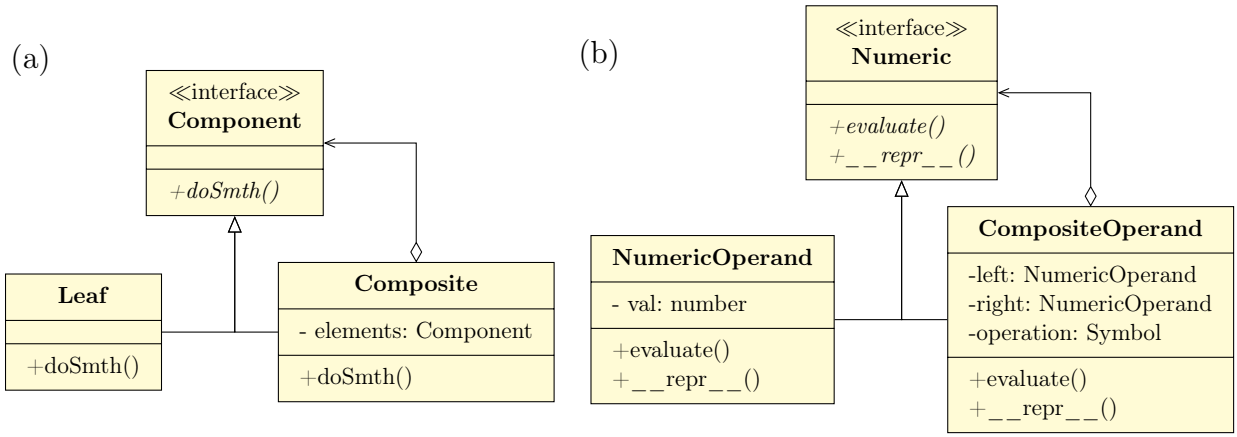


Рис. 2.8: Шаблон «Компонувальник». (a) Загальна схема. (b) «Python-like» приклад ієрархії класів для обчислення арифметичного виразу.

життя об'єкту, але збільшення часу відгуку може бути критичним для деяких практичних задач.

- Задачею мосту є розділення абстракції від реалізації, щоб їх можна було змінювати незалежно одну від одної. В певному сенсі це є альтернативою спадкуванню, що забезпечує жорсткий зв'язок між ними. Прикладом мосту в реальному житті є пульт дистанційного керування і приставка, віконний інтерфейс крос-платформенної програми, що виконує певні дії, змістовні для користувача, та його конкретна реалізація на заданій платформі (WinApi, KDE, Gnome).

2.4.3 Компонувальник

Шаблон «Компонувальник» (Composite) використовується для утворення деревоподібних структур даних у випадках, коли складний і елементарний об'єкти мають поводитись однаково. Типовими прикладами використання шаблону є репрезентування алгебраїчних виразів та файлової системи.

Приклад UML-діаграми шаблону показано на рис. 2.8. У цьому випадку композитний і елементарний об'єкти реалізують інтерфейс `Component` з операцією `doSmth()` із тою різницею, що композит виступає в ролі утворювача деревоподібної структури. Для деяких задач може бути зручним не виділяти окремий клас для елементарних об'єктів обриваючи ієрархію порожнім об'єктом або вказівником (такі як `None` у Python або `nullptr` у C++). Слід враховувати, що пряма ООП-реалізація може мати невелику швидкодію, що вирішується компонуванням через хеш-таблиці.

2.4.4 Легковаговик

Шаблон «Легковаговик» (Flyweight) є актуальним у тих випадках, коли необхідно адмініструвати екстремально велику кількість об'єктів. Оптимізація ресурсів на їх підтримку може бути занадто великою, якщо поводитись з ними

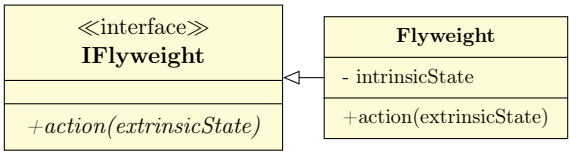


Рис. 2.9: Шаблон «Легковаговик»: спільний внутрішній стан виділено у поле `intrinsicState`, у той час, як поведінка згідно з індивідуальними особливостями визначається параметром `extrinsicState`.

звичайним чином. Прикладом можуть бути літери у rich-text форматі. (Звертаємо увагу на неочевидні аспекти написання якісного текстового редактору [7].)

Розв'язання проблеми ресурсів може полягати в тому, що ідея об'єкту зазнає декомпозиції на ресурсоємний або просто спільний для великої кількості об'єктів внутрішній стан, `intrinsicState` та індивідуальні зовнішні стани `extrinsicState`, див. рис. 2.9. Тоді об'єкти з однаковим `intrinsicState` виражаються одним екземпляром класу `Flyweight`. Індивідуальність поведінки визначається передачею в методи зовнішнього стану.

2.5 Шаблони поведінки

Задачею шаблонів поведінки є організація взаємодії між класами для досягнення бажаної поведінки цільового класу. Основними поведінковими шаблонами є

1. Спостерігач
2. Стратегія
3. Стан
4. Шаблонний метод
5. Ітератор
6. Порожній об'єкт
7. Команда
8. Ланцюжок відповідальності
9. Посередник
10. Відбиток
11. Відвідувач
12. Інтерпретатор

Нижче буде розглянуто деякі з них. Нагадаємо, що принцип шаблону «стратегія» обговорювався раніше.

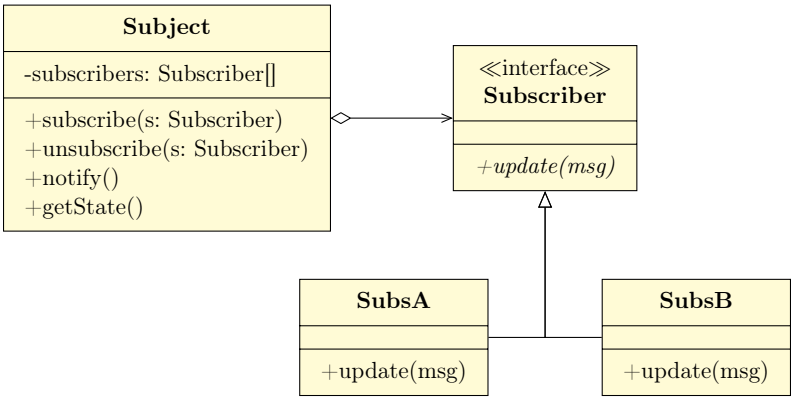


Рис. 2.10: Шаблон «Спостерігач»: підписники `SubsA` і `SubsB` отримують оновлення про зміни у `Subject`.

2.5.1 Стратегія, стан

Шаблони «Стратегія» (Strategy) і «Стан» (State) модифікують поведінку об'єкту шляхом виокремлення певного його функціоналу в окремі класи, екземпляри яких викликаються з цільового. Різниця у застосуванні проявляється в інтерпретації поведінки. Так, шаблон «Стан» є доцільним у випадках, коли поведінка об'єкту визначається його внутрішнім станом і цей стан може змінюватись. Наприклад, об'єкт передачі даних мережею має різні реакції в залежності від стану з'єднання, або запис у блозі демонструє різну поведінку в залежності від того, чи він знаходиться на етапі чернетки, модерації, опублікованості чи в архіві. Множина станів є такою, що її легко змінювати в майбутньому. Це також позбавляє від великої кількості умовних операторів і концентрує код, що пов'язаний зі станом, в одному місці. Шаблон «Стратегія» доцільний для імплементації сімейств алгоритмів. Слід пам'ятати, що клас-клієнт все одно має щось знати про алгоритми, що робить інкапсуляцію неповною. Обидва шаблони є альтернативою створенню підкласів і дозволяють делегувати відповідальність в інші класи замість збільшення ієрархії спадкування.

2.5.2 Спостерігач

Шаблон «Спостерігач» (Observer) дозволяє оповіщувати набори об'єктів про зміну стану одного з них, дозволяючи автоматичне оновлення станів інших. Прикладом може бути зміна діаграми в Excel, коли змінюються дані у колонках.

Приклад реалізації спостерігача наведено на рис. 2.10. Є об'єкт класу `Subject`, за станом якого ведеться спостереження. Цей інформатор веде облік своїх підписників, які реалізують інтерфейс `Subscriber`. Коли стан змінюється, вони отримують повідомлення через виклик методу `update(msg)`.

Перевагами шаблону є можливість гнучкого керування підписниками та незалежність між підписниками та інформаторами. Якщо метод інформування підписника також приймає ідентифікатор спостерігача, можна реалізовувати спостереження за множиною інформаторів. Слід враховувати, що класична ре-

алізація шаблону не враховує порядок інформування підписників та допускає видалення об'єкту підписника з утворенням некоректного посилання у списку інформатора. Прикладом шаблону є механізм подій у Qt.

2.5.3 Ітератор

Ітератори часто входять у склад високорівневих мов програмування як узагальнення поняття масиву і можуть бути застосовані напряму. Їх метою є реалізація доступу до піделементів агрегованого об'єкта у певній послідовності та без розкриття його внутрішнього представлення. Перевагою ітератора є можливість визначення різних типів обходу, але з урахуванням накладних витрат слід перевіряти, чи не досить простого циклу як альтернативи. Конкретна реалізація ітератора також може бути нестійкою відносно змін колекції під час обходу.

Коментар 7. Дерева

Нагадаємо, що бінарним деревом пошуку називається ієрархічна структура даних, у якій кожен елемент має не більше двох нащадків, ключі лівого піддерева не більше аніж ключ батьківського елементу, а ключі правого — не менші за нього. Також усі піддерева також мають бути бінарними деревами пошуку. Для таких структур вводять три порядки їх обходу, які коротко можна сформулювати так:

- **Прямий:** якщо поточний вузол не порожній, показуємо його, рекурсивно обходимо ліве піддерево, потім праве.
- **Зворотній:** якщо поточний вузол не порожній, обходимо ліве піддерево, потім праве й показуємо поточний вузол.
- **Центрований:** якщо поточний вузол не порожній, обходимо ліве піддерево, показуємо поточний вузол і обходимо праве піддерево.

2.5.4 Команда

Шаблон «Команда» (Command) дозволяє інкапсулювати дію як об'єкт із власними параметрами. Іншими словами, конкретні дії (зберегти файл, надіслати повідомлення, програти звук) представляються не методами основного класу, а об'єктами. Таким чином можна параметризувати дії, створювати черги команд, спрощувати логування та їх скасування[1]. Прикладом шаблону може бути дія, що динамічно призначається на кнопку графічного інтерфейсу програми. Команда є спорідненою до стратегії.

Ідею шаблону та приклад використання показано на рис. 2.11. Є головна дія, яку можна виконати, представлена методом `execute()` в інтерфейсі `ICommand`. Тому виконувач команди завжди може її викликати з екземпляра класу, який

[1]Багаторазове повторення та скасування команд можуть й не повернути об'єкт до початкового стану. Для цього слугує шаблон «Відбиток».

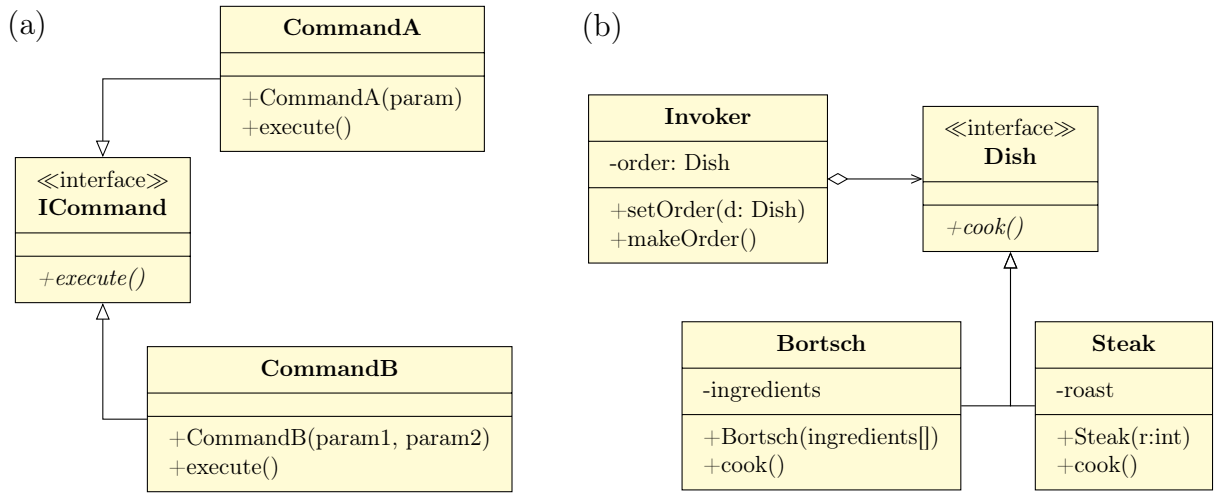


Рис. 2.11: Шаблон «Команда». (a) UML шаблону. (b) Реалізація шаблону на прикладі замовлення в ресторані, де стравою `Dish` може виступати відбивна або борщ з індивідуальними параметрами.

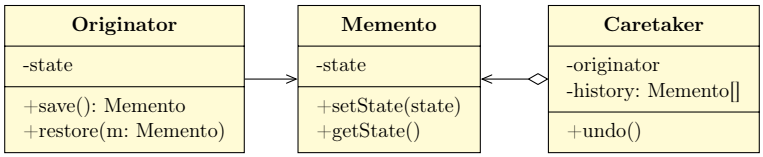


Рис. 2.12: Шаблон «Відбиток». Знімок стану класу `Originator` зберігається у класі `Memento` і заповнюється ним же без порушення інкапсуляції. Клас для оперування знімками `Caretaker` має доступ лише до інтерфейсу `Memento`, а не до даних.

реалізує даний інтерфейс. Водночас що саме буде виконано, визначається змістом цього класу. Якщо командою реалізовано замовлення страви у ресторані, то для кухаря головною командою буде `cook()`, яка визначається в інтерфейсі `Dish` та реалізується у двох варіантах — борщу `Bortsch` та стейку `Steak`. Результат приготування ж буде визначатись параметрами конструктора: інгредієнтами борщу чи ступенем прожарювання стейку. Зручністю цього підходу є можливість утворення черги, по якій можна ітерувати, динамічної зміни команд та відкладеність запуску операції. Зворотною стороною підходу є необхідність створювати велику кількість додаткових класів.

2.5.5 Відбиток

Шаблон «Відбиток» (Memento) слугує для збереження стану об'єкту зі збереженням принципів ООП, тобто без порушення інкапсуляції при створенні резервної копії.

UML-діаграму цього шаблону показано на рис. 2.12. Позбутись порушення інкапсуляції можна перекладанням відповідальності за підготовку власної копії на клас `Originator`, за станом якого необхідно слідкувати. Йому досту-

пні екземпляри класу `Memento`, який повинен мати поля для збереження усіх необхідних даних. Оперування снапшотами відбувається через окремий клас `Caretaker`, який дозволяє утворювати чергу зі снапшотів та обирати потрібний з них при відновленні стану.

Перевагою шаблону є зменшення зв'язності системи та централізованість керування станами класів, а недоліком — необхідність класу-посередника. Звертаємо увагу, що в C++ можливе використання дружніх класів (friend classes) в обхід класичної схеми шаблону.

2.5.6 Відвідувач

Шаблон «Відвідувач» (Visitor) реалізує так звану множинну диспетчеризацію, що є зручним у мовах, де її пряма підтримка відсутня. Також, це дозволяє мінімізувати зміни у класах, для яких згодом доводиться створювати специфічні обробники, наприклад, розширення функціоналу програми, яка оперує вузлами бінарного дерева.

Коментар 8. Диспетчеризація

У сучасних мовах програмування досить часто застосовується механізм вибору конкретної реалізації методу в залежності від динамічних типів об'єкту. Наприклад, якщо у C# створено масив об'єктів класів, що реалізують певний інтерфейс, то при звертанні до конкретного елементу масиву буде виконуватись код, який прописано у конкретному класі. Це приклад одинарної диспетчеризації. У випадку множинної диспетчеризації на вибір реалізації впливає більше, аніж один динамічний тип. Наприклад, реалізація функції зіткнення двох об'єктів визначається обома типами: ракета–астероїд, ракета–ракета, астероїд–планета тощо. Зокрема, множинна диспетчеризація присутня у CLOS (Common Lisp Object System) та R.

На рис. 2.13 показано реалізацію шаблону для випадку обходу дерева, в якому вузли можуть бути різних типів, `NodeA` і `NodeB`. Конкретні відвідувачі реалізують інтерфейс `IVisitor`, в якому прописано абстрактні методи для виконання операцій над усіма можливими типами вузлів. Кожен з цих методів приймає екземпляр вузла, над яким необхідно виконати операцію. Різні відвідувачі будуть виконувати різні операції над одним і тим самим вузлом.

Кожен з класів вузлів імплементує інтерфейс `INode`, який дозволяє зв'язати екземпляр конкретного відвідувача методом `accept()`. Отримавши відвідувача, екземпляр вузла знає, який метод з нього треба викликати (`VisitA()` для `NodeA`) і передає себе у нього.

Перевагами відвідувача є можливість виокремлення операцій, що залежать від конкретних класів із мінімальною зміною наявного коду. Якщо оперування з відвідувачем завчасно передбачено у класі, то зміна функціоналу програми можлива навіть за наявності лише двійкової версії класу в бібліотеці. На відміну від ітератора, тут можна працювати з об'єктами різних класів. З іншого боку, підхід не повністю відповідає принципам інкапсуляції та навантажує задачами

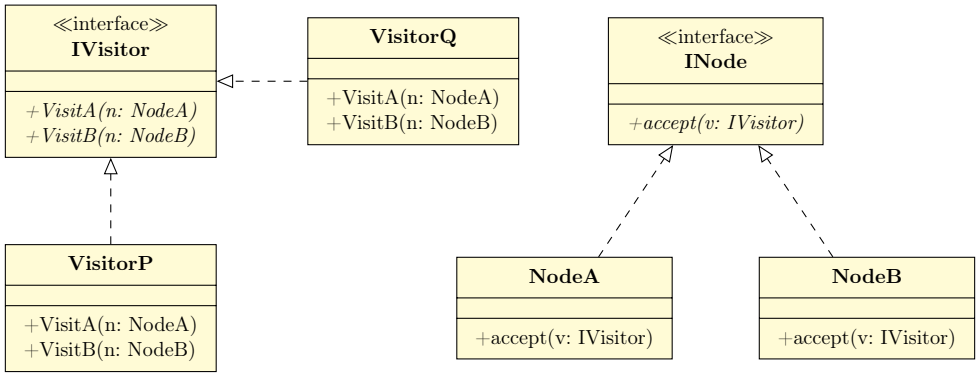


Рис. 2.13: Шаблон «Відвідувач» на прикладі операцій з деревом. Інтерфейс `IVisitor` визначає абстрактні методи, які можна виконувати над вузлами дерева різних типів. Конкретні відвідувачі реалізують їх. Вузли дерева приймають об'єкт відвідувача та викликають правильний для даного класу метод.

клас-посередник.

2.6 Архітектурні шаблони

Архітектура ПЗ у цілому осмислює взаємодію компонент системи та представлення даних. Існує велика кількість шаблонних архітектур у залежності від області застосування ПЗ [8]. Класичною архітектурою розподілених застосунків є багаторівнева (N-tier), де вибудовується система ієрархії рівнів клієнтського, веб-інтерфейсу, проміжних рівнів обробки та рівня збереження даних, де під окремим рівнем часто розуміється окрема фізична або віртуальна машина із власною зоною відповідальності (фаєрвол, база даних тощо). Тут зупинимось на описі класичної архітектури MVC для проектування stand-alone застосунків.

Ідеєю архітектури MVC (Model–View–Controller) є розділення системи на три компоненти, див. рис. 2.14:

1. Model (модель) визначає рівень даних та взаємозв'язків між ними. Це може бути база даних із класами для операцій з нею. Це те, що називають бізнес-логікою: модель предметної області з тим, що оброблюється, і правилами як оброблюється. Це головний і єдиний незалежний ні від чого компонент архітектури.

2. View (представлення) — це інтерфейс роботи користувача, наприклад, група класів GUI (Graphical User Interface) або CLI (Command Line Interface), з якими взаємодіє користувач ПЗ. Представлення власне й репрезентує дані користувачу та переадресовує запити на операції з ними.

3. Controller (контролер) визначає взаємодію між конкретним представленням та моделлю. Він отримує сигнали від представлення, може здійснювати їх проміжний аналіз та переадресовує до моделі.

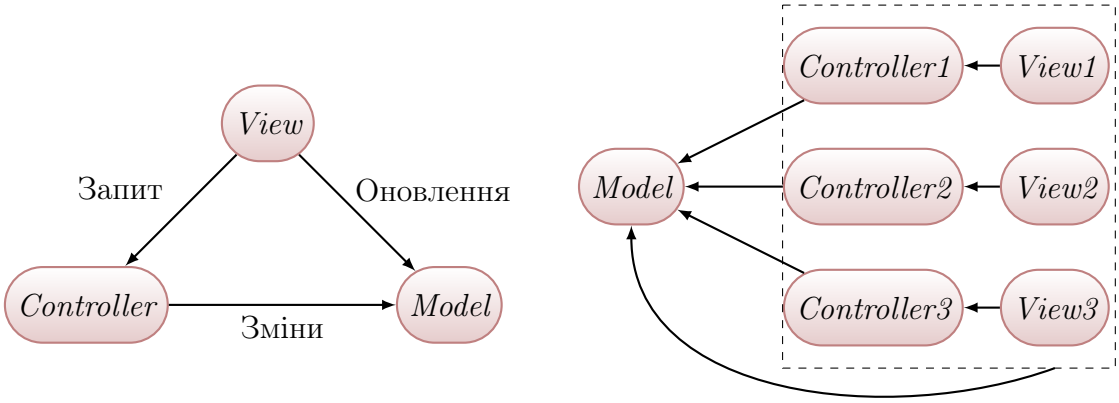


Рис. 2.14: Шаблон MVC. (a) Концепція. Стрілками вказано напрямки від залежного до незалежного елементу. (b) Множинні представлення із власними контролерами залежні від одної моделі.

Модель може бути виключно в єдиному екземплярі для конкретного ПЗ, в той час, як пари представлень і контролерів можуть бути наявні у необхідній кількості, реалізуючи необхідні шляхи взаємодії з користувачами (графічний інтерфейс, інтерфейс командного рядка, доступ через мережевий порт тощо). Коли користувач взаємодіє із представленням, запит на зміни передається через контролер до моделі. Модель інформує представлення про зміни для оновлення інформації, яка надається користувачу.

Коментар 9. MVC у фреймворках

Необхідно зауважити, що у реалізаціях деяких фреймворків (зокрема, для проектування веб-застосунків) під моделлю розуміють базу даних, під контролером — бізнес-логіку, а під представленням — логіку виведення даних.

Модель MVC зручно будувати за допомогою шаблонів, розглянутих раніше. Так усі представлення зручно підписати на оновлення моделі за допомогою спостерігача. Також, представлення може реалізувати компонувальник для композиції форм та елементів форм, з якими взаємодіє користувач, та генеруватись фабрикою або будівельником. Вибір конкретного контролера та пов'язаного з ним представлення може бути покладено на стратегію.

2.7 Задачі

№7. Опишіть 5–8 користувацьких історій для систем

1. управління аптекою
2. керування готелем
3. бронювання квитків

4. онлайн-бібліотеки

№8. Зобразіть діаграми використання варіантів для вищезгаданих систем.
№9. Зобразіть діаграму послідовностей для однієї з користувацьких історій.
№10. Спроектуйте систему та створіть демо-програму для представлення описаної в задачі структури та обґрунтуйте вибір використаних шаблонів.

1. Представлення веб-сайтів, які генеруються за визначеними шаблонами (лендінг, візитка фірми, особистий сайт, блог тощо).

2. Класи представлення білкової молекули.

3. Інтернаціональна поліграфічна продукція на основі календаря (календар, шкільний чи діловий щоденник). Врахуйте, що свята і вихідні дні в різних країнах різні.

4. Представлення «помешкань» для тварин у зоопарку (для риб потрібні скляні стіни, в той час як жирафу високі решітки у яких можна кріпити гілки дерев).

5. Елементи графічного інтерфейсу з можливістю локалізації.

6. Класи композитної структури алгебраїчного виразу.

7. База даних статей, з якої користувачі можуть отримати PDF.

№11. Спроектуйте систему та створіть демо-програму для представлення описаної в задачі структури та обґрунтуйте вибір використаних шаблонів.

1. Великий набір примітивів векторної графіки (лінія, дуга кола) та їх композицій.

2. Елементи векторної графіки, до яких можна застосувати ефекти (заливка, модифікація контуру тощо).

3. Мобільні телефони з різною функціональністю (фото, будильник, календар тощо) для персонажів комп'ютерної гри.

4. Елементи графічного інтерфейсу (кнопки, вікна, панелі тощо) та їх реалізації під різні API.

5. Елементи графічного інтерфейсу користувача мобільної ОС, які залежать від регіональних налаштувань і перемикаються за сигналом GPS без додаткових дій з боку користувача.

6. Іконки для елементів файлової системи.

7. Елементи умовного географічного простору з ієрархічною структурою із можливістю обчислення площі.

8. Класи для маніпулювання зображенням, у якому доступ дозволено лише до пікселів у заданому прямокутнику.

9. Класи для маніпулювання зображенням великого розміру з можливістю прозорого кешування (користувач не знає про кеш).

№12. Спроектуйте систему та створіть демо-програму для представлення описаної в задачі структури та обґрунтуйте вибір використаних шаблонів.

1. Класи обробки HTTP-запитів різних типів (GET, POST тощо) із можливістю динамічної зміни кількості обробників.
2. Елементи графічного інтерфейсу з кнопками, функціонал яких можна динамічно змінювати.
3. Персонаж в ігровому просторі, стан якого може змінюватись та бути відкоченим назад.
4. Послідовний обхід дерева у прямому та зворотному напрямках із фільтрацією вузлів по заданому критерію.

№13. MVC. Створіть демо-програму сервісного центру з двома приймальнями (View). Працівник приймальні (Controller) оброблює замовлення занотовуючи опис проблем, присвоює замовленню номер та віддає прилад до майстерні (Model). З майстерні прилад повертається до приймальні, звідки його забирає замовник.

2.8 Take home message

1. Ідіоми і шаблони як стандартні методи вирішення задач алгоритміки та побудови архітектури ПЗ.
2. Породжувальні шаблони як абстракція створення об'єктів складної структури чи наборів об'єктів спільної природи.
3. Структурні шаблони як абстракція взаємодії різних класів.
4. Шаблони поведінки як абстракція розподілу обов'язків між класами.
5. Архітектурний шаблон MVC як спосіб декомпозиції архітектури ПЗ на складові, які можуть незалежно одна від одної модифікуватись.

Посилання

2. *Пароннджанов В. Д.* Учебное пособие по языку ДРАКОН для вузов. — М. : ДМК Пресс, 2012. — ISBN 978-5-94074-800-7.

5. *Alexander C.*, *Ishikawa S.*, *Silverstein M.* A Pattern Language: Towns, Buildings, Construction. — Oxford University Press; Illustrated edition, 1977. — ISBN 0-19-501919-9.

7. *Хохта В.* Текстовый редактор — это вам не высшая математика, тут думать надо. — 2018. — URL: https://habr.com/ru/company/jugru/blog/424763/.

9. *Merali Z.* Computational science: ...Error // Nature. — 2010. — Т. 467, вип. 7317, № 7317. — С. 775—777. — DOI: 10.1038/467775a. — URL: https://doi.org/10.1038/467775a.

Література до розділу

1. *Bell D.* UML Basics: Getting started using UML for visual modeling of computer programs. — IBM. — URL: https://developer.ibm.com/technologies/web-development/series/uml-basics/.

3. Паттерны проектирования / Э. Фримен, Э. Фримен, К. Сьерра, Б. Бейтс. — СПб. : Питер, 2011. — ISBN 978-5-459-00435-9.

4. Приемы объектно-ориентированного программирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. — СПб. : Питер, 2011.

6. Refactoring.Guru. Патерни проектування. —. — URL: https://refactoring.guru/uk/design-patterns.

8. Azure Application Architecture Guide. —. — URL: https://docs.microsoft.com/en-us/azure/architecture/guide/.

10. *Орлов С. А.* Программная инженерия. Учебник для вузов. — 5-е вид. — СПб. : Питер, 2016. — ISBN 978-5-496-01917-0.

11. *Макконнелл С.* Совершенный код. — Русская Редакция, Microsoft Press, 2017. — ISBN 978-5-7502-0064-1.

12. *Фаулер М.* Рефакторинг: улучшение существующего кода. — СПб. : Символ-Плюс, 2004.

Розділ 3

Метод скінченних різниць

Почнемо з обговорення задач, які описуються у термінах виразів із частинними похідними різних порядків. Зокрема, це можуть бути функціонали енергії чи диференційні рівняння (у тому числі й нелінійні), що описують рівноважні розподіли функції (параметру порядку) u (будемо вважати u досить гладкою функцією).

Примітка 11. Параметр порядку

Під параметром порядку розуміють певну величину, яка є основною характеристикою задачі та виступає як аргумент функціоналу енергії. Прикладами параметрів порядку можуть бути скалярна функція ψ, що характеризує густину топологічних дефектів у надпровідниках, намагніченість $\boldsymbol{M}$ у задачах мікромагнетизму (векторний параметр порядку), яка виникає за температури нижче температури Кюрі, орієнтація молекул в нематику тощо.

3.1 Дискретизація простору. Сітки

Технічно найпростішим методом аналізу задач, сформульованих як системи рівнянь у частинних похідних всередині простору досить симетричної геометрії, є дискретизація простору, на якому ці вирази задані, сіткою із певним кроком та заміна похідних на їх різницеві апроксимації. Наприклад, якщо задача задана на прямокутнику $\Omega = [0, a] \times [0, b]$, то сітка з рівномірним кроком може бути описана як

$$\varpi = \left\{(x_i, y_j) = (ih_x, jh_y),\ i = \overline{0, N},\ j = \overline{0, M}\right\}, \tag{3.1}$$

див. 3.1. Тут кроки дискретизації вздовж абсциси та ординати рівні $h_x = a/N$ і $h_y = b/M$ відповідно. Тепер область Ω фігурує в задачі не як неперервна множина з нескінченної кількості точок, а як набір з $(N+1)(M+1)$ вузлів у вершинах прямокутників зі сторонами h_x і h_y. Кількість цих вузлів визначає просторову розмірність задачі: для скалярного параметра порядку необхідно зберігати стільки ж значень u, скільки наявно вузлів у сітці. Для векторних параметрів порядку кількість змінних збільшується відповідно до розмірності вектору. Чисельне розв'язання задач математичної фізики з інтерпретацією просторових

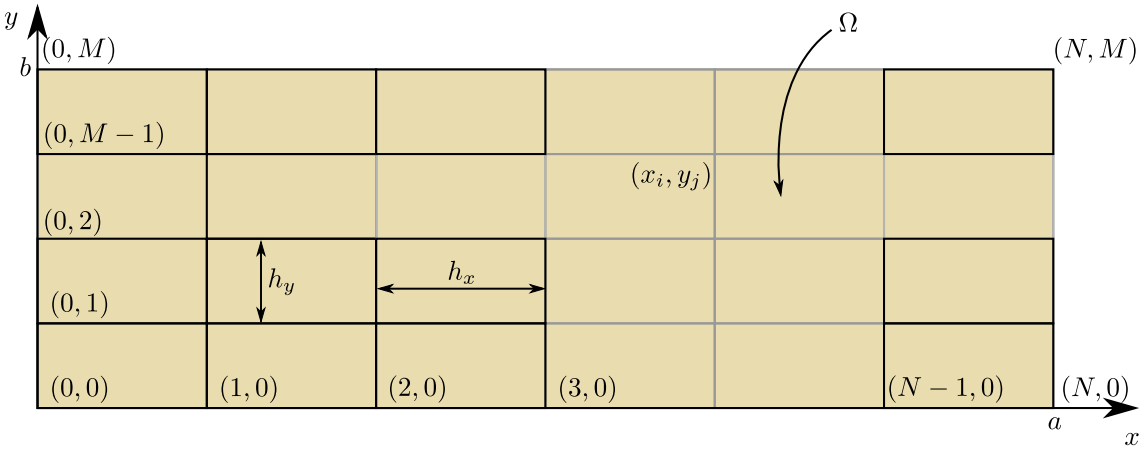


Рис. 3.1: Рівномірна прямокутна сітка на області Ω з кроками h_x і h_y вздовж перпендикулярних осей.

похідних як різницевих схем (наборів різницевих рівнянь) на заданій сітці має назву методу скінченних різниць.

3.1.1 Різницеві похідні

Якість наближення неперервної області сіткою визначається тим, наскільки швидко змінюється параметр порядку у просторі, тобто, який масштаб його похідних у порівнянні з кроком дискретизації сітки. Якщо задача одновимірна ($b = 0$ й індекс j у (3.1) відсутній), то всередині сітки похідна від u по координаті x на i-му вузлі може бути виражена трьома способами:

$$(\partial_x u)^i = \frac{u^{i+1} - u^i}{h_x}, \quad \text{[правостороння різниця (forward difference)]} \tag{3.2а}$$

$$(\partial_x u)^i = \frac{u^i - u^{i-1}}{h_x}, \quad \text{[лівостороння різниця (backward difference)]} \tag{3.2б}$$

$$(\partial_x u)^i = \frac{u^{i+1} - u^{i-1}}{2h_x}, \quad \text{[центральна різниця (central difference)]} \tag{3.2в}$$

де $u^i \equiv u(x_i)$. Тут і далі значення координат на сітці будуть позначатись нижніми індексами, а значення функцій — верхніми. Легко побачити, що формули (3.2) у границі $h_x \to 0$ збігаються до означення похідної у математичному аналізі. Іншими словами, якби можна було використовувати нескінченно малий крок дискретизації, то результат обчислень за будь-якою з них був би тотожній. Очевидно, що на практиці ця вимога є недосяжною, що призводить до специфічних особливостей їх використання. Почнемо з оцінки точності виразів (3.2). Виходячи з початкового формулювання задачі у термінах континуальної функції $u(x)$, значення параметру порядку на сусідньому вузлі можна наближено

записати як розклад у ряд Тейлора:

$$\begin{aligned} u^{i+1} &= u^i + h_x(\partial_x u)^i + \frac{h_x^2}{2}\left[(\partial_{xx}u)^i\right]^2 + \dots, \\ u^{i-1} &= u^i - h_x(\partial_x u)^i + \frac{h_x^2}{2}\left[(\partial_{xx}u)^i\right]^2 + \dots \end{aligned} \tag{3.3}$$

Підставивши цей розклад у (3.2) легко побачити, що права й ліва різниці у найменшому порядку за h_x мають доданки, які пропорційні h_x (тобто, похибка відкидання (truncation error) становить $O(h_x)$). Водночас для центральної різниці похибка відкидання містить наступний порядок, $O(h_x^2)$. Формально, це означає, що задана точність обчислення похідної, розрахованої методом центральних різниць, досягається за суттєво більших кроків h_x, аніж за інших методів, тобто центрально-різницеві схеми допускають менш щільні сітки й, як наслідок, менші розмірності задач. Однак, центральна різницева схема може призводити до фіктивних осциляцій u у просторі внаслідок того, що вона зв'язує значення функції через вузол, а не на сусідніх вузлах. Боротьба з цим потребує спеціальних методів.

Аналогічно можна наближено обчислювати похідні вищих порядків та змішані похідні. Так, похідну $\partial_{xx}u$ можна обчислити із суми виразів (3.3), або як різницеву похідну від функції, що сама є похідною, що приводить до виразу

$$(\partial_{xx}u)^i = \frac{u^{i+1} - 2u^i + u^{i-1}}{h_x^2} + O(h_x^2), \tag{3.4}$$

де використано означення за центральною різницевою схемою. Змішана похідна характеризується властивістю $\partial_{xy}u(x,y) \equiv \partial_x(\partial_y u) = \partial_y(\partial_x u)$. Зовнішня похідна має вигляд

$$(\partial_y u)^{i,j} = \frac{(\partial_x u)^{j+1} - (\partial_x u)^{j-1}}{2h_y} + O(h_y^2). \tag{3.5}$$

З урахуванням (3.2в), отримаємо

$$(\partial_y u)^{i,j} = \frac{u^{i+1,j+1} - u^{i-1,j+1} + u^{i+1,j-1} - u^{i-1,j-1}}{4h_x h_y} + O(h_x^2, h_y^2). \tag{3.6}$$

Варто окремо зазначити випадок похідної від добутку функції на першу похідну, $\partial_x f(x)$ із $f(x) = p(x)\partial_x u$. Як правило, множник $p(x)$ є наперед визначеним на всій області Ω, що дозволяє використати для нього формально більш щільну сітку:

$$\begin{aligned} (\partial_x f)^i &\approx \frac{f^{i+1/2} - f^{i-1/2}}{h_x}, \\ f^{i+1/2} &\approx p^{i+1/2} \times \frac{u^{i+1} - u^i}{h_x}, \quad f^{i-1/2} \approx p^{i-1/2} \times \frac{u^i - u^{i-1}}{h_x}, \\ (\partial_x f)^i &\approx \frac{p^{i+1/2}u^{i+1} - (p^{i+1/2} + p^{i-1/2})u^i + p^{i-1/2}u^{i-1}}{h_x^2} + O(h_x^2). \end{aligned} \tag{3.7}$$

Вищенаведені вирази справедливі для внутрішньої області сітки ϖ, тобто для $0 < i < N$ і $0 < j < M$. На межі ϖ центральна похідна не існує, тому можливе обрахування лише правої або лівої різниць. Для відповідних вузлів має сенс

використовувати односторонні різницеві похідні (one-sided difference). Покажемо спосіб їх введення на прикладі лівої межі одновимірної області $i = 0, 1, \ldots$ із $x_i = ih_x$. Щоб збільшити точність обчислення (наприклад, для забезпечення порядку $O(h_x^2)$ на всій сітці включаючи її межу), можна залучити додаткові вузли:

$$(\partial_x u)^0 \approx \frac{au^0 + bu^1 + cu^2}{h_x}, \tag{3.8}$$

де a, b і c — коефіцієнти, які треба знайти. Виразимо значення параметру порядку в старших вузлах за допомогою розкладу в ряд Тейлора у точці $x = 0$ і підставимо його у (3.8) для знаходження похибки відкидання:

$$\begin{aligned} u^1 &= u^0 + h_x \partial_x u(0) + \frac{h_x^2}{2} \partial_{xx} u(0) + \ldots, \\ u^2 &= u^0 + 2h_x \partial_x u(0) + 2h_x^2 \partial_{xx} u(0) + \ldots, \\ \frac{au^0 + bu^1 + cu^2}{h_x} &\approx \frac{a+b+c}{h_x} u^0 + (b + 2c)(\partial_x u)^0 + \frac{h_x}{2}(b + 4c)(\partial_{xx} u)^0. \end{aligned} \tag{3.9}$$

Шуканий вираз дасть першу похідну з точністю $O(h_x^2)$ за умови

$$\left.\begin{aligned} a + b + c &= 0, \\ b + 2c &= 1, \\ b + 4c &= 0 \end{aligned}\right\} \Rightarrow \begin{cases} a = -\dfrac{3}{2}, \\ b = 2, \\ c = -\dfrac{1}{2}. \end{cases} \tag{3.10}$$

Тоді права одностороння різницева схема для першої похідної має вигляд

$$(\partial_x u)^0 = \frac{-3u^0 + 4u^1 - u^2}{2h_x} + O(h_x^2). \tag{3.11}$$

Аналогічно отримуються вирази для лівих різниць та похідних вищих порядків. Таблиці коефіцієнтів лівих, центральних та правих різницевих схем для похідних різних степенів наведено в [1].

3.1.2 Складні геометрії

Практична реалізація методу скінченних різниць проста і наочна в одновимірних задачах з рівномірною сіткою та багатовимірних задачах, де область інтегрування є прямокутником, прямокутним паралелепіпедом тощо. Іншими словами, запис різницевих схем є простим, якщо різницеві похідні можна записати вздовж ортогональних напрямків. На практиці такі випадки є екзотичними й необхідно аналізувати складніші геометрії. Для цього можна застосовувати наступні підходи.

Залежно від типу задачі та вимог на регулярність сітки, різницеву схему можна записати для системи координат, яка має симетрію геометрії. Наприклад, розглядаючи задачу на поверхні сфери, геометрично нерівномірна сітка може бути введена з використанням сталих кроків по азимутальному і полярному кутах. Потенційним недоліком такого підходу буде її велика густина в азимутальному вимірі поблизу полюсів.

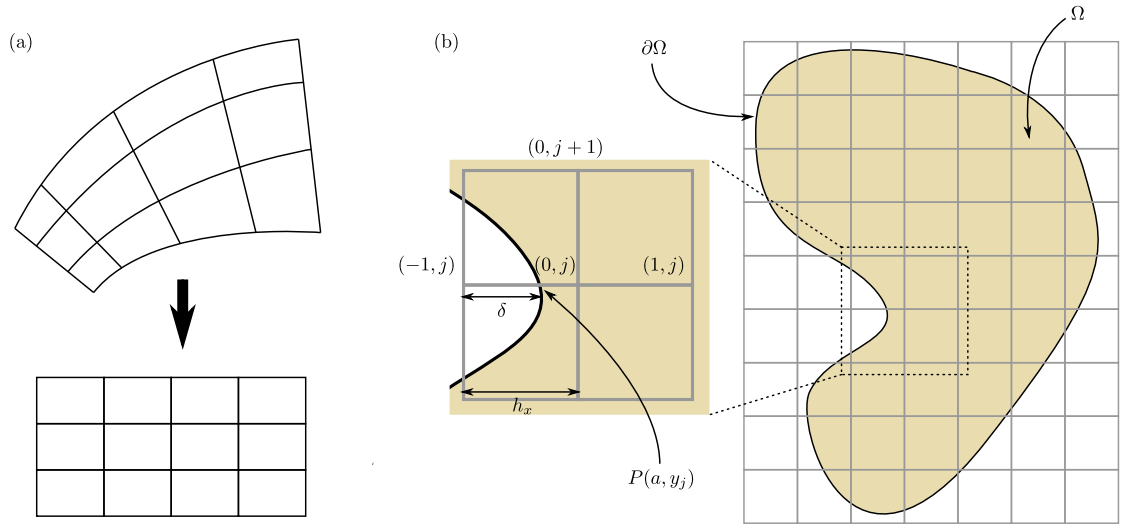


Рис. 3.2: (a) Формальне перетворення ортогональної сітки на складній геометрії у прямокутну. (b) Схема врахування криволінійної межі $\partial\Omega$ на рівномірній квадратній сітці. Точка P межі з координатами (a, y_j) потрапляє у міжвузля сітки на відстані $h_x - \delta$ від вузла з індексами $(0, j)$. Це дозволяє ввести фіктивний вузол з від'ємною координатою ліворуч, щоб інтерполювати значення параметру порядку в точці P.

Узагальненням вищеописаного підходу може бути перехід до нових ортогональних координат, для яких геометрія формально набуде прямокутної форми знову, див. рис. 3.2(a). Так, у двовимірному випадку нові координати $\eta = \eta(x, y)$ і $\xi = \xi(x, y)$ дають наступні диференціальні співвідношення параметру порядку:

$$\begin{aligned} \partial_x u &= \partial_\eta u(\eta, \xi)\partial_x \eta + \partial_\xi u(\eta, \xi)\partial_x \xi, \\ \partial_y u &= \partial_\eta u(\eta, \xi)\partial_y \eta + \partial_\xi u(\eta, \xi)\partial_y \xi. \end{aligned} \tag{3.12}$$

Спрощення межових умов у цьому випадку призводить до більш громіздких виразів самих похідних та необхідності обчислювати метрику простору (якобіан перетворення) при розрахунку інтегральних значень.

Третій підхід полягає у наближеному обчисленні значень функцій на межі із застосуванням прямокутної (для двовимірного випадку) сітки, яка не збігається із межею області інтегрування. Розглянемо випадок, показаний на рис. 3.2(b). Нехай на межі області Ω вимагається значення $u(\partial\Omega) = f(x, y)$ (межові умови Діріхле). Оскільки межа проходить між сусідніми вузлами сітки, використаємо лінійну інтерполяцію для оцінки значення параметру порядку в міжвузлі:

$$u(a, y_j) = f(a, y_j) \approx \frac{u^{-1,j} \times (h_x - \delta) + u^{0,j} \times \delta}{h_x}. \tag{3.13}$$

Тоді

$$u^{-1,j} = \frac{h_x}{h_x - \delta} f(a, y_j) - \frac{\delta}{h_x - \delta} u^{0,j}. \tag{3.14}$$

Значення функції u у фіктивному вузлі з індексом абсциси $i = -1$ може бути використано для обчислення відповідних різницевих похідних. Недоліком цього підходу є потенційна складність імплементації автоматичного врахування всіх межових точок та громіздкість виразів для інших типів межових умов.

3.2 Рівняння Пуассона

Розглянемо рівняння Пуассона (див. вираз (Б.8) у додатку Б) на відрізку для прикладу імплементації різницевої схеми. Нехай крайова задача має вигляд

$$\partial_{xx} u = f(x), \quad x \in [0, L], \quad u(0) = \mu_0, \quad u(L) = \mu_1. \tag{3.15}$$

Введемо рівномірну сітку

$$\varpi = \left\{ x_i = ih,\ i = \overline{0, N},\ h = \frac{L}{N} \right\} \tag{3.16}$$

і перепишемо (3.15) у різницевих термінах. Ліва частина рівняння визначається виразом (3.4). Права частина може бути виражена як значення функції $f^i = f(x_i)$, усередненням по інтервалу виду $f^i \to (1/h) \int_{x_i - h/2}^{x_i + h/2} f(x) \mathrm{d}x$ тощо. Таким чином, отримується система рівнянь

$$\begin{cases} u^{i+1} - 2u^i + u^{i-1} = h^2 f^i, \quad i = \overline{1, N-1}, \\ u^0 = \mu_0, \quad u^N = \mu_1. \end{cases} \tag{3.17}$$

Для подальшого чисельного розв'язання її зручно представити у матричному вигляді

$$A\boldsymbol{u} = \boldsymbol{f}, \tag{3.18а}$$

де

$$A = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & & \dots & \\ & & & 1 & -2 \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} u^1 \\ u^2 \\ \dots \\ u^{N-2} \\ u^{N-1} \end{bmatrix}, \quad \boldsymbol{f} = h^2 \begin{bmatrix} f^1 - \mu_0/h^2 \\ f^2 \\ \dots \\ f^{N-2} \\ f^{N-1} - \mu_1/h^2 \end{bmatrix}. \tag{3.18б}$$

Звертаємо увагу, що розмірність матриці менша, аніж кількість вузлів, а межові умови увійшли в задачу як поправки до двох елементів вектора $\boldsymbol{f}$.

У випадку великих розмірностей тридіагональну матрицю A зручно зберігати як розріджену (sparse). Тут враховано, що за межових умов Діріхле значення u у першому і останньому вузлах відомі й не потребують додаткових обчислень. Таким чином, розв'язання межової задачі зводиться до знаходження розв'язку (3.18а) відомими методами. Результат обчислень подібної задачі показано на рис. 3.3. Аналогічним чином записуються різницеві схеми для межових умов інших типів.

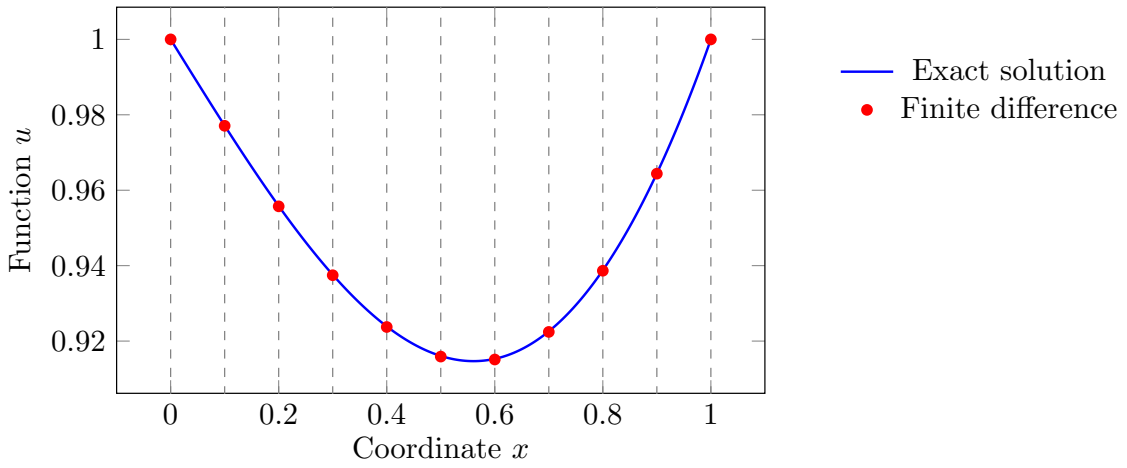


Рис. 3.3: Аналітичне і чисельне знаходження розв'язку рівняння $\partial_{xx} u = \sin \frac{\pi}{2} x$ із межовими умовами $u(0) = u(1) = 1$ на відрізку $x \in [0, 1]$.

3.3 Рівняння параболічного типу

Розширимо попередній приклад на двовимірні задачі. У якості другої змінної візьмемо час на прикладі рівняння дифузії (теплопровідності, див. вираз (Б.7) у додатку Б)

$$\begin{aligned} \partial_t u &= \partial_{xx} u + f(x,t), \\ u(0,t) &= \mu_0(t), \quad u(L,t) = \mu_1(t), \\ u(x,0) &= g(x), \end{aligned} \tag{3.19}$$

де $x \in [0, L]$ і $t \in [0, T]$. Відповідна сітка може бути записана як

$$\varpi = \left\{ (x_i, t_j) = (ih, j\tau),\ i = \overline{0, N},\ j = \overline{0, M} \right\} \tag{3.20}$$

із просторовим і часовим кроками дискретизації $h = L/N$ і $\tau = T/M$ відповідно. У задачах із часовою динамікою множина вузлів з однаковим значенням часу називається шаром, а стан системи — знімком (snapshot). Як і у попередній задачі, просторову похідну перепишемо за допомогою центральної різницевої схеми, а похідну по часу виразимо як праву різницеву похідну. Таким чином, задача (3.19) набуде вигляду

$$\begin{cases} \dfrac{u^{i,j+1} - u^{i,j}}{\tau} = \dfrac{u^{i+1,j} - 2u^{i,j} + u^{i-1,j}}{h^2} + f^{i,j}, & i = \overline{1, N-1},\ j = \overline{0, M-1}, \\ u^{0,j} = \mu_0(t_j), \quad u^{N,j} = \mu_1(t_j), & j = \overline{0, M}, \\ u^{i,0} = g(x_i), & i = \overline{0, N}. \end{cases} \tag{3.21}$$

Звертаємо увагу, що чисельна задача повинна бути узгоджена у межових і початкових умовах, $\mu_0(0) = g(0)$ і $\mu_1(0) = g(L)$, інакше (3.21) не матиме розв'язку.

Відмінністю даної задачі (3.21) від розглянутого раніше рівняння Пуассона є те, що безпосереднє інтегрування по просторовій змінній не виконується

завдяки наявності початкової умови $g(x)$. З різницевого рівняння можна безпосередньо виразити значення різницевої функції у наступний момент часу t_{j+1}:

$$u^{i,j+1} = u^{i,j} + \tau f^{i,j} + \frac{\tau}{h^2}(u^{i+1,j} - 2u^{i,j} + u^{i-1,j}), \quad i = \overline{1, N-1},\ j = \overline{0, M-1}. \tag{3.22}$$

Це так звана *явна різницева схема*. Можна показати, що вона працює лише за умови $\tau \leq 0.5h^2$ [2], інакше похибка обчислень буде швидко зростати замість того, щоб залишатись малою. Таку поведінку називають нестійкою. Схема (3.21) є умовно стійкою через те, що крок по часу обмежується просторовою дискретизацією. Результат інтегрування цією схемою показано на рис. 3.4(b–d).

Коментар 10. Стійкість чисельних схем

Під стійкістю конкретної чисельної схеми розуміють рівномірну й неперервну залежність її розв'язку від параметрів задачі. Наприклад, для рівняння теплопровідності такими параметрами є джерело $f(x,t)$, початкова і межові умови $g(x)$, $\mu_{0,1}(t)$. Іншими словами, якщо задана чисельна схема стійка, то малі відхилення від початкових параметрів (а також, параметри інтегрування) призводять до малих змін у результаті обчислень. Якщо це не так, то можуть проявлятися випадкові флуктуації числових значень необмежено посилюючись і спотворюючи результат обчислень, або ж виникнуть артефакти, пов'язані зі способом дискретизації.

Така різницева схема обмежено використовується на практиці через те, що просторова дискретизація часто вимагається малою, що може зробити крок по часу неприйнятно малим. Відмітимо, що, попри це, явні різницеві схеми мають значну перевагу перед іншими у питанні прототипування і тестування окремих модулів складних обчислювальних систем завдяки простоті імплементації у коді. Якщо мализна кроку по часу не є принциповою (це може бути справедливо при обчисленнях на відеокартах), явна різницева схема може використовуватись у якості основної.

Замість правої різницевої похідної по часу у (3.21) можна записати ліву різницеву похідну:

$$\begin{cases} \dfrac{u^{i,j} - u^{i,j-1}}{\tau} = \dfrac{u^{i+1,j} - 2u^{i,j} + u^{i-1,j}}{h^2} + f^{i,j}, & i = \overline{1, N-1},\ j = \overline{1, M}, \\ u^{0,j} = \mu_0(t_j), \quad u^{N,j} = \mu_1(t_j), & j = \overline{0, M}, \\ u^{i,0} = g(x_i), & i = \overline{0, N}. \end{cases} \tag{3.23}$$

У цьому випадку знімок системи у поточний момент часу є невідомим і повинен бути виражений через стан на попередньому кроці способом, схожим на розв'язання задачі Пуассона. Це одна з можливих *неявних схем*. Відповідне матричне рівняння на j-му шарі сітки має вигляд

$$A^j \boldsymbol{u}^j = \boldsymbol{f}^j, \tag{3.24a}$$

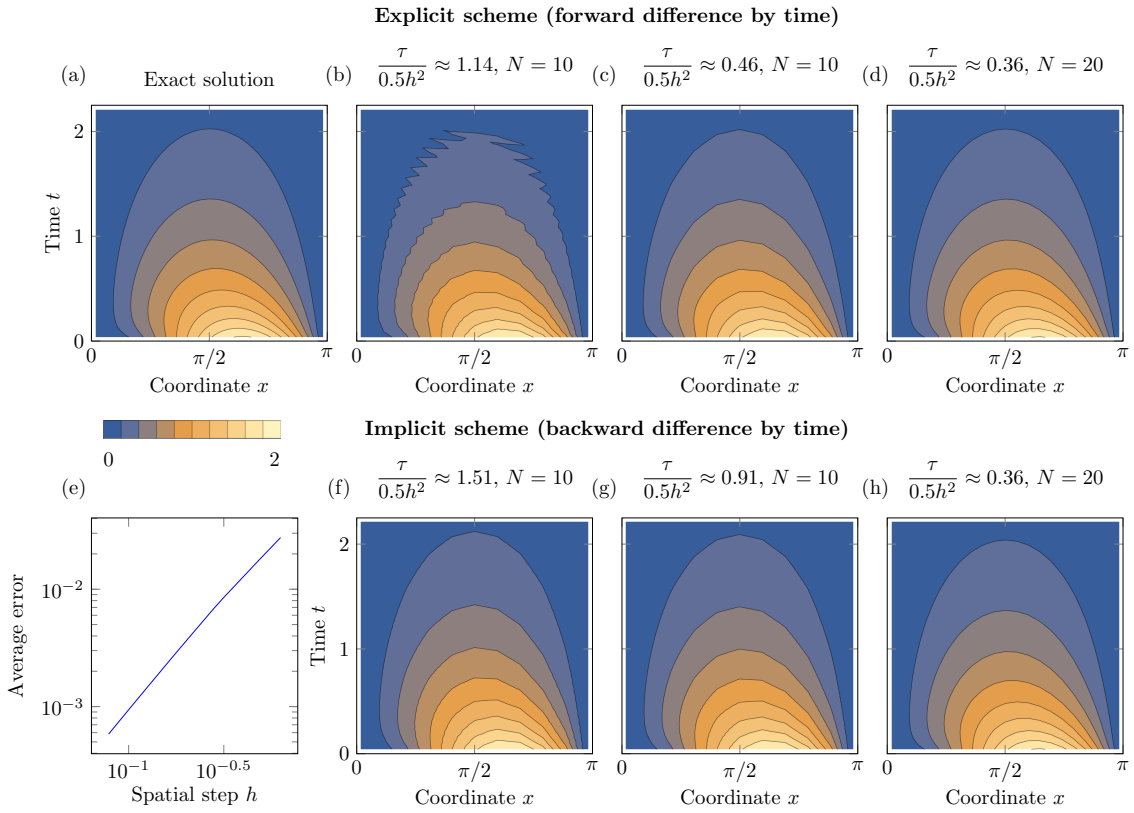


Рис. 3.4: Аналітичне і чисельне знаходження розв'язку рівняння $\partial_t u = \partial_{xx} u$ із межовими умовами $u(0) = u(\pi) = 0$ і початковою умовою $u(x, 0) = x \sin x$ на відрізку $x \in [0, \pi]$. (a) Точний розв'язок. (b–d) Розв'язок рівняння явною різницевою схемою (3.22) із різними кроками по простору і часу. Для (b) умова стійкості не виконується, що призводить до фіктивних осциляцій. (e) Середнє значення похибки обчислень неявної схеми у порівнянні з точним розв'язком на вузол сітки якщо $\tau = 0.5h^2$. (f–h) Розв'язок рівняння неявною схемою за (3.24). За однакових просторових дискретизацій та менших кроків по часу (більших співвідношень $\tau/(0.5h^2)$) розв'язок близький до аналітичного.

де

$$A^j = \begin{bmatrix} 1+2\alpha & -\alpha & & & \\ -\alpha & 1+2\alpha & -\alpha & & \\ & -\alpha & 1+2\alpha & -\alpha & \\ & & & \dots & \\ & & & -\alpha & 1+2\alpha \end{bmatrix}, \quad \boldsymbol{u}^j = \begin{bmatrix} u^{1,j} \\ u^{2,j} \\ \dots \\ u^{N-2,j} \\ u^{N-1,j} \end{bmatrix},$$
$$\boldsymbol{f}^j = \begin{bmatrix} u^{1,j-1} + \tau f^{1,j} + \alpha\mu_0(t_j) \\ u^{2,j-1} + \tau f^{2,j} \\ \dots \\ u^{N-2,j-1} + \tau f^{N-2,j} \\ u^{N-1,j-1} + \tau f^{N-1,j} + \alpha\mu_1(t_j) \end{bmatrix}. \tag{3.24б}$$

Тут використано позначення $\alpha = \tau/h^2$. Дана неявна схема є стійкою для будь-

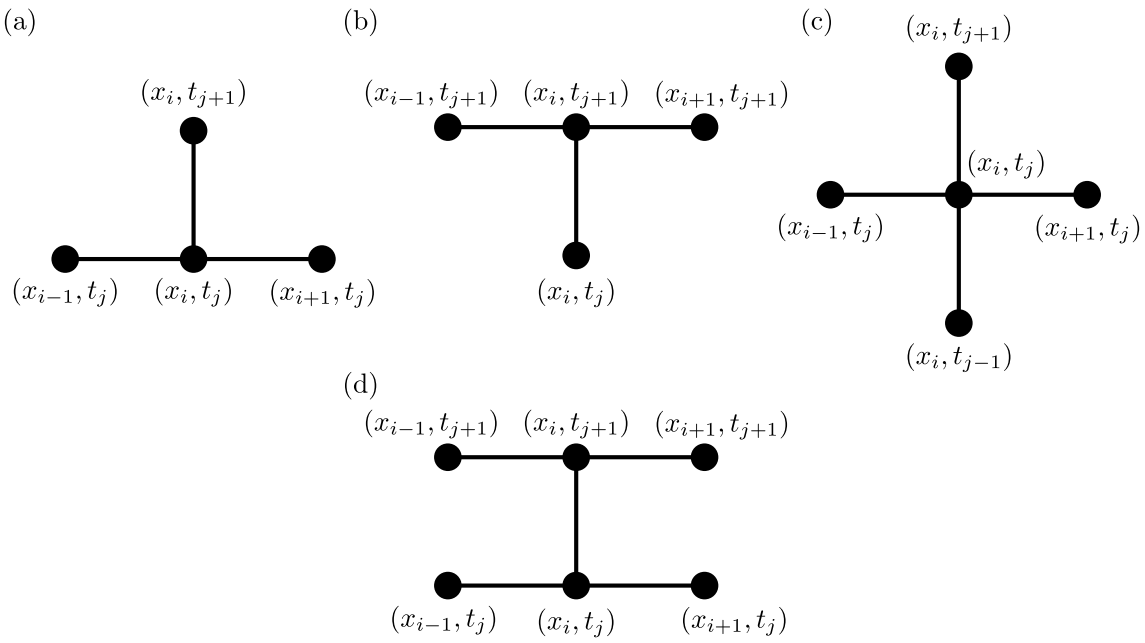


Рис. 3.5: Приклади шаблонів різницевих схем. (a) Явна схема. (b) Неявна схема. (c) Трьохшарова схема. (d) Схема Кранка—Ніколсон.

яких кроків просторової і часової дискретизації. Результат інтегрування цією схемою показано на рис. 3.4(e–h).

Наочну характеристику конкретної різницевої схеми дає так званий шаблон: множина точок, які використовуються для побудови різницевого виразу. Кілька найпростіших прикладів шаблонів показано на рис. 3.5.

Вищенаведені явна і неявна схеми є схемами першого порядку точності по часу. Прикладом реалізації другого порядку точності є *схема Кранка—Ніколсон* (Cranck–Nicolson, за британськими математиками Джоном Кренком і Філіс Ніколсон), яка також забезпечує другий порядок точності по координаті. Для цього розпишемо центральну похідну за часом відповідно до (3.2в) для i-го вузла в $(j+1/2)$-му шарі:

$$(\partial_t u)^{i,j+1/2} \approx \frac{u^{i,j+1} - u^{i,j}}{\tau}. \tag{3.25}$$

Відповідну праву частину оцінимо як арифметичне середнє явної (3.21) та неявної (3.23) схем, тобто

$$\frac{u^{i,j+1} - u^{i,j}}{\tau} = \frac{[u^{i+1,j+1} - 2u^{i,j+1} + u^{i-1,j+1}] + [u^{i+1,j} - 2u^{i,j} + u^{i-1,j}]}{2h^2} + \\ + \frac{f^{i,j+1} + f^{i,j}}{2}. \tag{3.26}$$

Схема (3.26) також є неявною і стійкою безвідносно до кроків сітки. Зауважимо, що даному підходу властиві й вади центральної різницевої схеми, зокрема поява фіктивних осциляцій якщо крок сітки занадто великий. Визначення невідомих $u^{i,j+1}$ відбувається так само, як і для вищерозглянутої неявної схеми

першого порядку за часом шляхом розв'язання матричного рівняння. Якщо диференціальне рівняння є нелінійним (що типово для задач магнетизму чи надпровідності), то визначення $u^{i,j+1}$ зводиться до розв'язання нелінійної системи алгебраїчних рівнянь. У деяких випадках, спростити задачу можна лінеаризацією такої системи.

3.4 Методи Рунге—Кутти вищих порядків

Вищерозглянуті методи інтегрування по часу фактично реалізують так званий метод Ейлера. Значення невідомої функції на наступному кроці по часу оцінюється за значенням похідної по часу, по аналогії з одновимірним випадком $u(t+\tau) \approx u(t) + \partial_t u(t) \times \tau$ із відповідною похибкою $O(\tau)$. Метод Ейлера (явний і неявний варіанти) є найпростішим з сімейства методів Рунге—Кутти, які дозволяють підвищити точність інтегрування часової динаміки із прийнятною кількістю обрахунків.

Для застосування методу Рунге—Кутти (Runge–Kutta) задача повинна бути сформульована у вигляді

$$\partial_t \boldsymbol{u} = \boldsymbol{f}(t, \boldsymbol{u}, \partial_x \boldsymbol{u}, \ldots), \quad \boldsymbol{u}(t_0) = \boldsymbol{u}_0. \tag{3.27}$$

Тут невідома функція $\boldsymbol{u}$ може бути як векторною, так і скалярною, а у початковий момент часу t_0 вона набуває значення $\boldsymbol{u}_0$. Вектор-функція $\boldsymbol{f}$ може включати нелінійні залежності як від $\boldsymbol{u}$, так і її похідних по простору.

Коментар 11. Представлення диференціального рівняння у векторному вигляді

Зауважимо, що рівняння, яке включає вищі похідні по часу може бути зведене до векторного вигляду (3.27). Наприклад, маючи рівняння нелінійного маятника

$$\partial_{tt} u + 2r\partial_t u + \omega^2 u = f(t, u)$$

з коефіцієнтом дисипації r, власною частотою ω та вимушуючою силою $f(t,u)$, можна ввести перепозначення $u_1(t) = u$, $u_2(t) = \partial_t u$. Тоді вираз (3.27) набуде вигляду

$$\partial_t \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u_2 \\ -2ru_2 - \omega^2 u_1 + f(t, u_1) \end{bmatrix}.$$

Ідея методів Рунге—Кутти полягає в уточненні напрямку руху за похідною. Проілюструємо це на прикладі *методу Гойна* (за Карлом Гойном, нім. Karl Heun). Значення функції на кроці $t+\tau$ може бути подане у вигляді ряду Тейлора

$$\boldsymbol{u}(t+\tau) = \boldsymbol{u}(t) + \tau\partial_t \boldsymbol{u}(t) + \frac{\tau^2}{2}\partial_{tt}\boldsymbol{u}(t) + O(\tau^3), \tag{3.28}$$

де другу похідну можна розписати як

$$\partial_{tt}\boldsymbol{u} \approx \partial_t \boldsymbol{f} + \partial_{\boldsymbol{u}} \boldsymbol{f}\, \boldsymbol{f}, \tag{3.29}$$

де $\partial_{\boldsymbol{u}}\boldsymbol{f}$ — якобіан. Саму функцію $\boldsymbol{f}$ теж можна представити у вигляді ряду Тейлора

$$\boldsymbol{f}(t+\tau, \boldsymbol{u}+\boldsymbol{k}) \approx \boldsymbol{f}(t, \boldsymbol{u}) + \tau\partial_t \boldsymbol{f}(t, \boldsymbol{u}) + \partial_{\boldsymbol{u}}\boldsymbol{f}\boldsymbol{k}. \tag{3.30}$$

Підставляючи (3.29) і (3.30) до (3.28) отримаємо вираз для оцінки значення функції на кроці $t+\tau$:

$$\begin{aligned} \boldsymbol{u}(t+\tau) &= \boldsymbol{u}(t) + \tau\left(\frac{1}{2}\boldsymbol{k}_1 + \frac{1}{2}\boldsymbol{k}_2\right), \\ \boldsymbol{k}_1 &= \boldsymbol{f}(t, \boldsymbol{u}(t)), \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t+\tau, \boldsymbol{u}(t) + \tau\boldsymbol{k}_1). \end{aligned} \tag{3.31}$$

Локальна похибка методу Гойна становить $O(\tau^3)$, що дає глобальну похибку $O(\tau^2)$. Метод Гойна також відомий як покращений метод Ейлера та метод Рунге—Кутти другого порядку (RK2).

Будь-який метод Рунге—Кутти для кроку $t_{n+1} = t_0 + \tau n$ можна подати як

$$\begin{aligned} \boldsymbol{u}^{n+1} &= \boldsymbol{u}^n + \sum_{i=1}^{p} b_i \boldsymbol{k}_i, \\ \boldsymbol{k}_i &= \tau\boldsymbol{f}\left(t_n + c_i\tau, \boldsymbol{u}^j + \sum_{j=1}^{p} a_{ij}\boldsymbol{k}_j\right). \end{aligned} \tag{3.32}$$

де число p називають кількістю кроків методу, а так звані ваги $\boldsymbol{b} = \{b_1, b_2, \ldots, b_p\}$, вузли $\boldsymbol{c} = \{c_1, c_2, \ldots, c_p\}$ і матриця Рунге—Кутти $A = \{a_{ij}\}$ пов'язані співвідношеннями

$$\sum_{i=1}^{p} b_i = 1, \quad \sum_{i=1}^{p} a_{ij} = c_i. \tag{3.33}$$

У схематичному вигляді конкретна схема Рунге—Кутти записується у вигляді так званих таблиць Батчера (за Джоном Батчером, англ. John C. Butcher):

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^{\mathrm{T}} \end{array}. \tag{3.34}$$

Для явних методів Рунге—Кутти матриця A є нижньотрикутною з нулями на головній діагоналі. Для кожної кількості кроків s можливі різні значення $\boldsymbol{b}$, $\boldsymbol{c}$ і A, що має певний вплив на конкретні значення похибок і кількість обчислень функції $\boldsymbol{f}$. Отримання формул для явних методів Рунге—Кутти до 4 порядку може базуватись на вищепродемонстрованому розкладі у ряд Тейлора. Методи вищих порядків вимагають застосування теорії графів.

Зауважимо, що порядок p явних методів Рунге—Кутти не більший за кількість кроків, при чому для відомих схем $p = s$ для $s \leq 4$. Це обмежує їх практичне використання методами до 4 і 5 порядків включно. У таблиці 3.4(a,b) наведено приклади таблиць Батчера для явних методів середньої точки та Рунге—Кутти 4 порядку (RK4).

Табл. 3.1: Таблиці Батчера для явних методі середньої точки, Рунге—Кутти 4 порядку (RK4) та Рунге—Кутти—Фельберга 4-5 порядків (RKF45, версія Фельберга). Перший рядок із нулями опущено.

(a)

1/2	1/2	0
	0	1

(b)

1/2	1/2			
1/2	0	1/2		
1	0	0	1	
	1/6	1/3	1/3	1/6

(c)

1/4	1/4					
3/8	3/32	9/32				
12/13	1932/2197	−7200/2197	7296/2197			
1	439/216	−8	3680/513	−845/4104		
1/2	−8/27	2	−3544/2565	1859/4104	−11/40	
	25/216	0	1408/2565	2197/4104	−1/5	0
	16/135	0	6656/12 825	28 561/56 430	−9/50	2/55

Підбір оптимального кроку по часу може виконуватись із різних міркувань. Так, для методу заданого порядку p можна скористатись правилом Рунге, згідно якого різниця між точним розв'язком і його чисельним наближенням за порядком величини рівна

$$\epsilon = \frac{\left\| \boldsymbol{u}^{n+1}|_p - \boldsymbol{u}^{n+1}|_{p+1} \right\|}{2^p - 1}. \tag{3.35}$$

Порівнюючи фактичну похибку обчислень ϵ із певною бажаною ϵ_0 можна підібрати оптимальний крок повторивши обрахунки на усьому цільовому інтервалі часу. Але для практичних задач такий підхід може виявитись недоцільним через велику часову вартість кожного прогону із заданим кроком τ. Оптимізацію можна виконати динамічно підбираючи τ на кожній ітерації (так звані адаптивні методи). Ефективним методом оцінки є

$$\tau_{n+1} = \alpha_{\text{safe}} \tau_n \left(\frac{\epsilon_0}{\epsilon} \right)^{1+1/p}, \quad \text{якщо} \quad \epsilon \geq \epsilon_0, \tag{3.36}$$

або перерахунок $\boldsymbol{u}^{n+1}$ зі зменшеним τ_n

$$\tau_n \leftarrow \alpha_{\text{safe}} \tau_n \left(\frac{\epsilon_0}{\epsilon} \right)^{1/p}, \quad \text{якщо} \quad \epsilon < \epsilon_0. \tag{3.37}$$

Тут $\alpha_{\text{safe}} \sim [0.8;\, 0.9]$ обирається для «безпеки», оскільки оцінка точності завжди наближена.

Інший підхід полягає у порівнянні значення $\boldsymbol{u}^{n+1}$, обчисленного одним і тим самим методом різного порядку, прикладом чого слугує метод Рунге—Кутти—Фельберга 4–5 порядків (RKF45), див. таблицю Батчера 3.4(c). Таблиця містить

два рядки коефіцієнтів $\boldsymbol{b}$ і дозволяє розрахувати значення функції методом 4 і 5 порядків використавши одні й ті самі $\boldsymbol{k}_{1,\dots,5}$ та $\boldsymbol{k}_6$:

$$\begin{aligned}\boldsymbol{u}^{n+1}|_{p=4} &= \boldsymbol{u}^n + \frac{24}{216}\boldsymbol{k}_1 + \frac{1408}{2565}\boldsymbol{k}_3 + \frac{2197}{4101}\boldsymbol{k}_4 - \frac{1}{5}\boldsymbol{k}_5,\\ \boldsymbol{u}^{n+1}|_{p=5} &= \boldsymbol{u}^n + \frac{16}{135}\boldsymbol{k}_1 + \frac{6656}{12\,825}\boldsymbol{k}_3 + \frac{28\,561}{56\,430}\boldsymbol{k}_4 - \frac{9}{50}\boldsymbol{k}_5 + \frac{2}{50}\boldsymbol{k}_6.\end{aligned} \tag{3.38}$$

Відповідні оптимальні кроки для наступної або поточної ітерацій рівні

$$\tau_{n+1} = \alpha_{\text{safe}}\tau_n \left(\frac{\epsilon_0}{\epsilon}\right)^{1/4}, \quad \text{якщо} \quad \epsilon \geq \epsilon_0, \tag{3.39}$$

та

$$\tau_n \leftarrow \alpha_{\text{safe}}\tau_n \left(\frac{\epsilon_0}{\epsilon}\right)^{1/5}, \quad \text{якщо} \quad \epsilon < \epsilon_0. \tag{3.40}$$

На практиці доцільно також обмежувати мінімальний і максимальний кроки виходячи з фізичного контексту. Також варто враховувати, що адаптивний крок по часу вносить числові шуми в оцінку енергії системи, що може виявитись критичним для певних типів задач.

Розгляд інших схем виходить за межі даного посібника. Зауважимо лише, що неявні методи Рунге—Кутти"хоча й більш громіздкі в реалізації, є більш придатними для розв'язання так званих жорстких рівнянь, тобто тих, які вимагають екстремально малих кроків при розв'язанні явними методами. Такі задачі можуть виникати, наприклад, через наявність коефіцієнтів із суттєво різними порядками, наприклад 1 проти 10^5. Також для рівнянь, що містять похідну по часу другого порядку існують спеціалізовані схеми Рунге—Кутти—Ністрома (Runge–Kutta–Nyström), в яких можна забезпечити мінімізацію фазових похибок тощо [3—6].

3.5 Рівняння гіперболічного типу

3.5.1 Чисельна дисипація

3.6 Знаходження рівноважних енергетичних станів

Для практичної ілюстрації як знаходити рівноважні стани складних, у тому числі багаточастинкових систем, розглянемо магнітну текстуру у феромагнітному нанодроті. Безпосереднє розв'язування варіаційної задачі може потребувати суттєвих обчислювальних або людських ресурсів (у сенсі складності імплементації математичних виразів у коді та їх тестування). Альтернативним методом є задача мінімізації енергії, асоційованої з відповідними варіаційними рівняннями. Її реалізація у коді може бути суттєво простіша технічно (інколи, вона й швидше обчислюється), але серед результатів можуть з'являтись фіктивні розв'язки, пов'язані з особливостями ітераційного процесу мінімізації функції багатьох змінних.

Примітка 12. Мікромагнетизм

Магнетизм є одним з характерних розділів фізики, у якому є принциповою нелінійність рівнянь, які описують статичні та динамічні процеси на мікро- та наномасштабах. Це зумовило широке використання цієї галузі фізики для опрацювання та тестування математичних і чисельних методів аналізу солітонних рівнянь і динамічного хаосу разом з іншими розділами фізики, які базуються на функціоналі Гінзбурга—Ландау.

Феноменологічна теорія магнетизму оперує поняттям вектора намагніченості $\boldsymbol{M}(\boldsymbol{r})$ — магнітного моменту одиниці об'єму. За температур, значно нижчих температури Кюрі, зручно працювати з одиничним вектором $\boldsymbol{m}(\boldsymbol{r}) = \boldsymbol{M}/|\boldsymbol{M}|$, що є характерною особливістю задач феромагнетизму (на відміну від надпровідності, де параметр порядку є скалярним). У найпростішому випадку можна обмежитись двома доданками у функціоналі енергії: обмінній енергії (нуль для однорідної магнітної текстури та більше нуля за наявності будь-якої неоднорідності; вона визначається перекриттям хвильових функцій електронів сусідніх атомів), та анізотропії (мікроскопічно визначається спін-орбітальною взаємодією й визначає енергетично вигідний напрямок $\boldsymbol{m}$).

Обмежимось випадком тонкого феромагнітного дроту круглого перетину площею S, який описується функціоналом енергії

$$\begin{aligned} E[\boldsymbol{m}] &= E_{\text{x}}[\boldsymbol{m}] + E_{\text{a}}[\boldsymbol{m}], \\ E_{\text{x}}[\boldsymbol{m}] &= SA \int \left[(\nabla m_x)^2 + (\nabla m_y)^2 + (\nabla m_z)^2\right] \mathrm{d}s — \text{ обмінна енергія}, \\ E_{\text{a}}[\boldsymbol{m}] &= -SK \int (\boldsymbol{m} \cdot \boldsymbol{e}_{\text{a}})^2 \mathrm{d}s — \text{ анізотропія}, \end{aligned} \tag{3.41}$$

де $A > 0$ (Дж/м) і K (Дж/м^3) константи обміну та анізотропії, ∇ — градієнт, $\boldsymbol{e}_{\text{a}}$ — одиничний вектор, який вказує напрямок осі анізотропії, а інтегрування виконується по довжині дроту (умова малої товщини дозволяє вважати розподіл $\boldsymbol{m}$ по товщині однорідним, тобто функцією лише від координати вздовж дроту $\boldsymbol{m}(s)$). Технічно, задача зводиться до дискретизації виразу (3.41) та пошуку мінімуму отриманої суми по значенням змінної $\boldsymbol{m}$ у вузлах сітки. Також звертаємо увагу, що тут площа перетину дроту завжди входить у вираз як множник до коефіцієнта взаємодії, тому для чисельної імплементації є доцільним перенормування $SA \to \widetilde{A}$ та $SK \to \widetilde{K}$.

Примітка 13. Неколінеарні магнітні текстури

З фундаментальної та практичної точки зору особливу цікавість складають такі розподіли векторного поля $\boldsymbol{m}$, які містять локальну неоднорідність. Так, для випадку $K > 0$ та $\boldsymbol{e}_{\text{a}} = \boldsymbol{e}_x$ у дроті існує розв'язок типу доменної стінки $m_x = \operatorname{th} x/\ell$, де так звана магнітна довжина $\ell = \sqrt{A/K}$.

Коментар 12. Властивості функціоналу енергії

Варто занотувати властивості функціоналу (3.41), що дозволить спростити тестування фінального коду.

- Інтегранд у E_{x} є сумою квадратів, тому невід'ємний для будь-якої магнітної текстури. Оскільки він є сумою градієнтів, то мінімальне значення, нуль, досягається для однорідної текстури, тобто $m_x = \text{const}$, $m_y = \text{const}$, $m_z = \text{const}$, при чому самі по собі ці три константи можуть бути довільними за умови $m_x^2 + m_y^2 + m_z^2 = 1$. Тобто для довільної, але однорідної у просторі орієнтації вектора $\boldsymbol{m}$ обмінна енергія має бути нульовою.

- Знак константи анізотропії K може бути довільним. Якщо $K < 0$, то означення (3.41) вимагатиме невід'ємного значення E_{a} (зауважимо, що за таких K розв'язок у вигляді доменної стінки не існує). Це досягається при $\boldsymbol{m} \perp \boldsymbol{e}_{\text{a}}$. Тобто тест енергії анізотропії з $\boldsymbol{e}_{\text{a}} = \boldsymbol{e}_x$ повинен призвести до $m_x = 0$. Якщо ж при цьому змінити знак K на додатній, то інтегранд навпаки, буде максимізуватись при $\boldsymbol{m} || \boldsymbol{e}_{\text{a}}$, що еквівалентно $m_x = \pm 1$. Звертаємо увагу, що через квадратичність інтегранду знак m_x не визначений.

Ці кілька тестів є доречними при перевірці коректності коду, який має справу з обчисленням значень енергії (3.41), або виразів, з нею пов'язаних. При модульному підході написання коду, коли за обчислення кожного з доданків у повній енергії відповідає свій модуль (це може бути окрема функція, файл з набором функцій, бібліотека тощо), можна перевіряти код для кожної з взаємодій незалежно одна від одної.

3.6.1 Прямий феромагнітний дріт

Вважаючи, що дріт довжиною L напрямлено вдовж осі абсцис ($s \equiv x$), введемо одновимірну сітку з N вузлів та рівномірним кроком[1]:

$$\varpi = \left\{ x_i = (i-1)h, \quad i = \overline{1, N}, \quad h = \frac{L}{N-1} \right\}. \tag{3.42}$$

Для феромагнітного дроту це означає наявність N вузлів, у яких обраховуються компоненти вектору $\boldsymbol{m}$

$$\boldsymbol{m}(x_i) \xrightarrow{\varpi} \boldsymbol{m}^i = \left\{ m_x^i, m_y^i, m_z^i \right\}, \tag{3.43}$$

та $N-1$ циліндричних ділянок довжиною h, на яких обчислюватиметься середнє значення енергії. Таким чином наявно $3N$ змінних у чисельній задачі. Чисельне інтегрування дозволить представити повну енергію як вираз, залежний

[1] На практиці, крок не зобов'язаний бути рівномірним, але для обговорюваного прикладу це спрощує ілюстрацію, й є корисним у випадках, коли положення неоднорідності у шуканому розв'язку не є відомим наперед.

від значень намагніченості на вузлах. Його мінімум за $3N$ змінними можна знайти методом спряжених градієнтів чи аналогічним, придатним для задачі оптимізації великої розмірності.

Необхідно контролювати сталість довжини $\boldsymbol{m}$ протягом мінімізації енергії. Одним з варіантів реалізації контролю є введення штрафної енергетичної функції, яка додається до (3.41)

$$E^{\varpi}_{\text{pen}} = \Lambda \sum_{i=1}^{N} \left[1 - (m^i_x)^2 - (m^i_y)^2 - (m^i_z)^2\right]^2. \tag{3.44}$$

Тут $\Lambda > 0$ є штрафним множником, який визначає втрату цільової функції при відхиленні $\boldsymbol{m}$ від дозволеного значення на вузлах: якщо довжина $\boldsymbol{m}^i$ відрізняється від одиниці, то відповідний доданок у сумі буде додатнім і змушуватиме оптимізувати його значення разом з енергією. Його оптимальна величина залежить від кількості змінних у задачі й може бути підібрана експериментально. Для кількох десятків тисяч змінних можна починати з $\Lambda \sim 100$. Штрафні функції можуть визначатись не квадратичними, а через модулі, що впливає на збіжність обраного методу пошуку мінімуму функції. Для методу спряжених градієнтів чи Ньютона гладкі штрафні функції є оптимальнішими.

Іншим способом врахувати умову $|\boldsymbol{m}| \equiv 1$ є формулювання задачі у кутових змінних: $\boldsymbol{m} = \{\cos\theta, \sin\theta\cos\phi, \sin\theta\sin\phi\}$, де для прикладу обрано осі відрахунку азимутального кута $\phi(x)$ та полярного кута $\theta(x)$ від осей $\hat{\boldsymbol{y}}$ та $\hat{\boldsymbol{x}}$, відповідно. На додачу, це зменшує кількість змінних (два кути локальної сферичної системи координат замість трьох декартових компонент вектора), але змінює «ландшафт» цільової функції мінімізації роблячи його періодичним і потребуючи врахування області значень полярного кута $\theta = [0, \pi]$. Це може потребувати обмежень по кроку в методі мінімізації через те, що типова його реалізація формально змінюючи кути не враховує обертання вектора й відповідну циклічність значень.

Для обчислення енергії анізотропії скористаємось інтегруванням методом трапецій:

$$E^{\varpi}_{\text{a}} = -SK \times \frac{h}{2} \sum_{i=1}^{N-1} \left[(\boldsymbol{m}^i \cdot \boldsymbol{e}^i_{\text{a}})^2 + (\boldsymbol{m}^{i+1} \cdot \boldsymbol{e}^{i+1}_{\text{a}})^2\right]. \tag{3.45}$$

Похибка обчислення E^{ϖ}_{a} має порядок $O(h^3)$. Звертаємо увагу, що хоча шукані значення $\boldsymbol{m}$ локалізовані на вузлах сітки, обчислення виконуються усередненням по циліндричному елементу об'єму дроту з площею перерізу S та довжиною h.

Коментар 13. Точки нестійкої рівноваги функціоналу енергії

Однією з основних проблем пошуку екстремальних значень (мінімумів або максимумів) певної функції є необхідність уникати сідлових точок та точок нестійкої рівноваги у просторі її змінних. Розглянемо випадок, коли у енергії анізотропії $K > 0$ та $\boldsymbol{e}_{\text{a}} = \{1, 0, 0\}$ (анізотропія типу легка вісь):

$$E_a = -SK \int m_x^2 \mathrm{d}S. \tag{3.46}$$

Тоді мінімум енергії (3.45) досягатиметься при $\boldsymbol{m}^i = \{\pm 1, 0, 0\}$. Але будь-який розподіл виду $\boldsymbol{m}^i = \{0, m_y^i, m_z^i\}$ даватиме нульовий градієнт від E_a по компонентах намагніченості. Позбавитись ризику потрапити у таку точку нестійкої рівноваги можна переписавши підінтегральний вираз на еквівалентний, скориставшись сталою довжиною $\boldsymbol{m}$:

$$E_\mathrm{a}^\mathrm{equiv} = -SK \int (1 - m_y^2 - m_z^2)\mathrm{d}S. \tag{3.47}$$

У такому випадку градієнт дискретизованої енергії по компонентах намагніченості буде ненульовим для будь-якої текстури, що не мінімізує енергію анізотропії й можна уникнути зупинки у стані нестійкої рівноваги. У загальному випадку доцільно робити декомпозицію подібних виразів на доданки, хоча б один з яких даватиме ненульовий градієнт. Зауважимо, що у цих випадках зупинка ітераційного процесу відбуватиметься завдяки штрафній енергетичній функції. Якщо її введення у конкретній задачі неможливо, у користувацькому інтерфейсі програми слід враховувати обмеження на початкові стани. Інколи доцільно вносити мале випадкове збурення на вузлах для уникнення подібних ситуацій.

Зауважимо, що на відміну від типових задач знаходження значення інтегралу чисельними методами, де крок інтегрування може підбиратись виходячи з вимоги на задану точність (наприклад, оцінкою методом Рунге), у задачах, подібних до обговорюваної, розмір сітки інтегрування визначається наперед виходячи з фізичних міркувань. Так для анізотропних феромагнетиків крок сітки h повинен бути менший за магнітну довжину ℓ (хоча б у кілька разів), що визначає характерний розмір неоднорідностей магнітної текстури. Це пов'язано з високою вартістю зміни просторової дискретизації системи у термінах комп'ютерного часу та необхідної пам'яті. Водночас, при аналізі часової динаміки крок інтегрування по часу може підбиратись динамічно.

Розрахунок обмінної енергії E_x потребує обчислення градієнту в кожному вузлі. Для одновимірної задачі із дротом, спрямованим вздовж осі x, яку розглядаємо тут, $\nabla \to \partial_x$. Скористаємось правою різницевою похідною. Похідна на i-му вузлі рівна

$$(\partial_x m_\nu)^i \approx \frac{m_\nu^{i+1} - m_\nu^i}{h}, \quad i = \overline{1, N-1}, \quad \nu = x, y, z. \tag{3.48}$$

де індекс фактично нумерує ділянки дроту, на яких обчислюється похідна. Тоді апроксимація інтегралу обмінної енергії методом трапецій запишеться як

$$E_\mathrm{x}^\varpi = SA \times h \sum_{i=1}^{N-1} \sum_{\nu=x,y,z} \left[(\partial_x m_\nu)^i\right]^2. \tag{3.49}$$

Звертаємо увагу, що у різницевих сумах (3.45) та (3.49) підсумовування ведеться по елементах об'єму, в той час, як штраф (3.44) розраховується для кожного вузла.

Таким чином, знаходження рівноважних станів намагніченості у прямому дроті зводиться до мінімізації виразу

$$E^{\varpi} = E^{\varpi}_{\text{x}} + E^{\varpi}_{\text{a}} + E^{\varpi}_{\text{pen}} \tag{3.50}$$

за набором змінних m^i_ν, $\nu = x, y, z$, $i = \overline{1, N}$, що на практиці може бути здійснено методом спряжених градієнтів або одним з його різновидів. Зауважимо, що у задачах, які становлять практичний інтерес, кількість вузлів на об'єм зразка може склади сотні тисяч. Тому застосування методів, які базуються на знаходженні обернених матриць, рідко буває доцільним.

Якщо розглядати феромагнетик з $K > 0$ та $\boldsymbol{e}_{\text{a}} = \{1, 0, 0\}$ (анізотропія типу легка вісь), найменшу енергію матиме однорідний стан вздовж дроту з

$$\boldsymbol{m} = \{\pm 1, 0, 0\}. \tag{3.51}$$

Найнижчим енергетичним збудженням буде доменна стінка із профілем

$$m_x(x) = \operatorname{th} \frac{x - x_0}{w}, \tag{3.52}$$

де $w = \ell = \sqrt{A/K}$ — її ширина, а x_0 — положення центру, яке може бути довільним у даній моделі. Це проілюстровано на рис. 3.6. У якості початкової магнітної текстури задано доменну стінку із шириною $w = 0.5\ell$. Процес мінімізації енергії на певній ітерації дозволяє побачити метастабільний стан у вигляді звичайної доменної стінки з $w = \ell$, але через малий розмір зразка у порівнянні з неоднорідністю текстури (зверніть увагу на те, як близько кінчики червоної лінії рівноважного розподілу m_y на рис. 3.6(c) підходять до меж зразка) далі вона деформується і система приходить до основного, однорідного стану. Для довшого зразка доменна стінка залишиться стабільною.

Зауважимо, що природні значення параметрів зразка є числами різних порядків: $A \sim 1\,\text{пДж/м}$, $K \sim 1\,\text{МДж/м}^3$ що дає $\ell \sim 3 \ldots 50\,\text{нм}$. Характерні діаметри та довжини феромагнітних дротів можуть становити величини порядку десятків та сотень–тисяч ℓ, відповідно. Оперування числами різних порядків є джерелом чисельного шуму, пов'язаного з обмеженістю мантиси у двійковому представленні чисел, тому виконувати обчислення рекомендується із параметрами, попередньо віднормованими так, щоб мати коефіцієнти порядку 1.

3.6.2 Феромагнітне кільце

Особливості методу скінченних різниць на складніших геометріях можна розглянути за допомогою модифікації задачі із попереднього розділу на випадок дроту радіусом R, замкненого у кільце (s — натуральний параметр вздовж кільця). Сітку можна визначити як

$$\varpi = \left\{ (x_i = R\cos(i-1)\phi, y_i = R\sin(i-1)\phi), \quad i = \overline{1, N}, \quad \phi = \frac{2\pi}{N} \right\}, \tag{3.53}$$

де крок вздовж кільця рівний $h = R\phi$. Звертаємо увагу, що порівнювати з характерними просторовими масштабами необхідно саме h, а не ϕ ($h \ll \ell$).

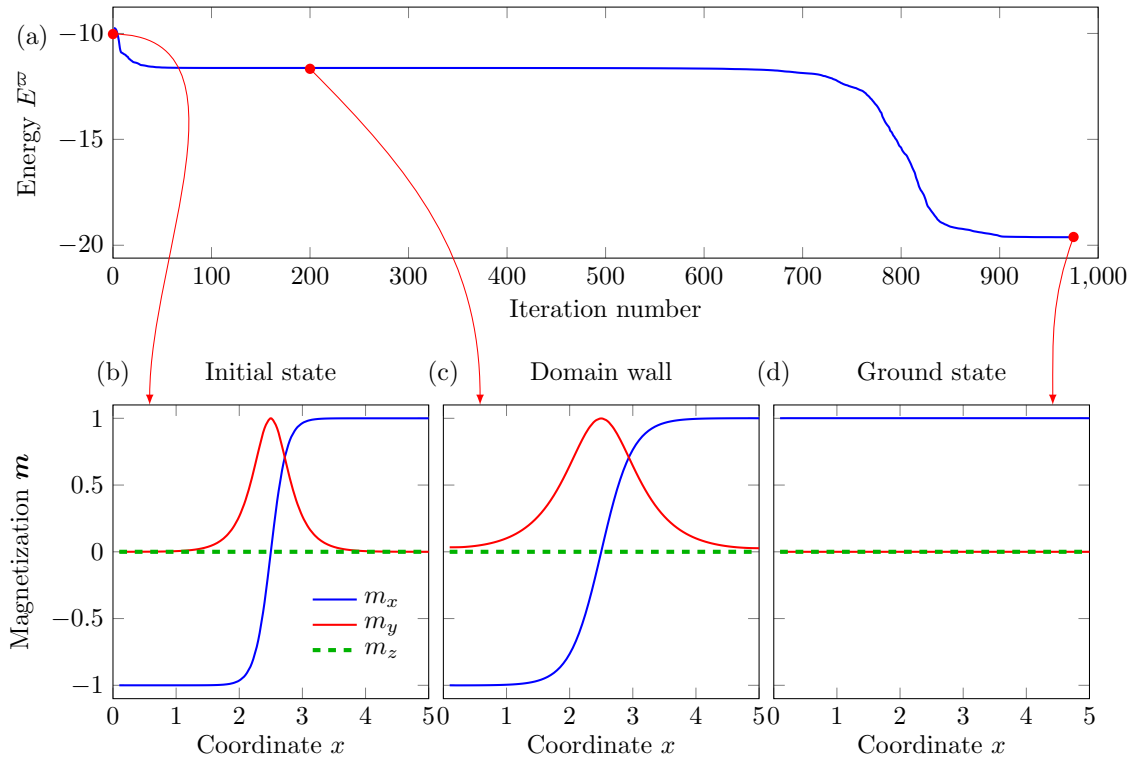


Рис. 3.6: Знаходження рівноважних станів прямого феромагнітного дроту ($\widetilde{A} = 1$, $\widetilde{K} = 4$, $N = 50$, $h = 0.1$). (a) Зміна повної енергії сітки E^{ϖ} з номером ітерації. (b) Початковий стан: доменна стінка в центрі зразка із шириною вдвічі меншою за рівноважну. (c) Зрелаксована доменна стінка як метастабільний стан. (d) Основний стан з найменшою енергією.

Відомо, що для криволінійних дротів характерною є анізотропія типу легка вісь з напрямком, дотичним у кожній точці вздовж дроту [7]. Тоді вектор осі анізотропії буде задаватись як

$$\boldsymbol{e}_{\mathrm{a}}^{i} = \{-\sin(i-1)\phi, \cos(i-1)\phi, 0\}. \tag{3.54}$$

Вираз енергії анізотропії на сітці ϖ (3.45) набуде виду

$$\begin{aligned} E_{\mathrm{a}}^{\varpi} = &- SK \times \frac{h}{2} \sum_{i=1}^{N-1} \left[(\boldsymbol{m}^{i} \cdot \boldsymbol{e}_{\mathrm{a}}^{i})^2 + (\boldsymbol{m}^{i+1} \cdot \boldsymbol{e}_{\mathrm{a}}^{i+1})^2 \right] - \\ &- SK \times \frac{h}{2} \left[(\boldsymbol{m}^{N} \cdot \boldsymbol{e}_{\mathrm{a}}^{N})^2 + (\boldsymbol{m}^{1} \cdot \boldsymbol{e}_{\mathrm{a}}^{1})^2 \right], \end{aligned} \tag{3.55}$$

де новий доданок відповідає періодичним межовим умовам.

Більш суттєвими є зміни до виразу обмінної енергії. По-перше, змінюється вираз для градієнта, $\nabla \to (1/R)\partial/\partial\phi$, індекси стають циклічними для врахування періодичності:

$$\begin{aligned} &(\nabla m_\nu)^i = \frac{1}{R}(\partial_\phi m_\nu)^i \approx \frac{m_\nu^{i+1} - m_\nu^i}{R\phi}, \quad i = \overline{1, N}, \\ &m_\nu^{N+1} \equiv m_\nu^1, \quad \nu = x, y, z. \end{aligned} \tag{3.56}$$

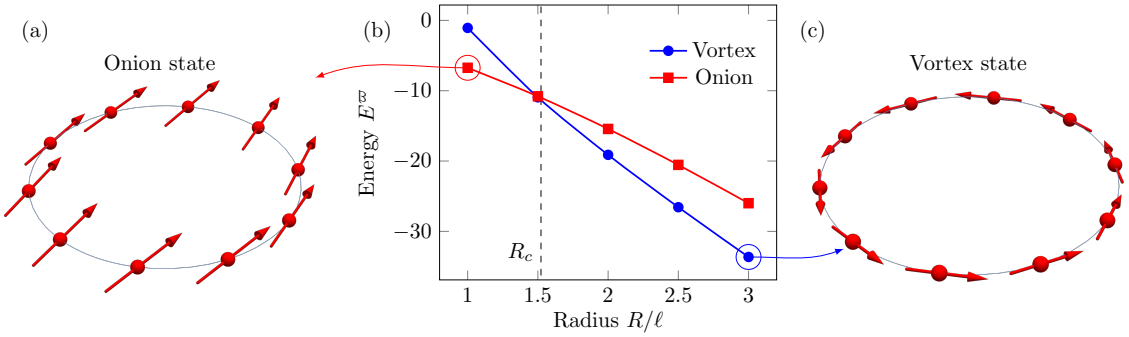


Рис. 3.7: Основні стани феромагнітного кільця ($\widetilde{A} = 1$, $\widetilde{K} = 4$, $h \approx 0.05$). На графіку показано зміну енергії стану, який зрелаксовано або з вихрового (дає вихровий стан), або з однорідного вздовж $\boldsymbol{e}_x$ (дає «onion»). Енергії зрівнюються при $R = R_c$: ліворуч від пунктирної лінії основним станом є «onion», праворуч — вихровий. На виносках показано відповідні стани, напрямок намагніченості позначено червоними стрілками.

Енергія обмінної взаємодії при цьому набуде вигляду

$$E_{\mathrm{x}}^{\varpi} = SA \times \frac{h}{R^2} \sum_{i=1}^{N} \sum_{\nu=x,y,z} \left[(\partial_\phi m_\nu)^i \right]^2 \tag{3.57}$$

У такій системі реалізується два різних основних стани в залежності від радіуса кільця [8], див. рис. 3.7. Якщо $R > R_c \approx \ell/0.657$, то найменшу енергію має стан «вихор» із $\boldsymbol{m} = \pm \boldsymbol{e}_{\mathrm{a}}$, тобто намагніченість утворює замкнений потік за або проти годинникової стрілки. Для малих радіусів енергетично вигіднішим є так званий «onion»-стан, який є майже однорідним.

Нехай певна фізична система може характеризуватись кількома стійкими станами. Навіть якщо енергія одного зі станів є меншою, інший за тих самих умов може бути метастабільним, тобто стійким до малих спотворень форми. Жоден з числових методів мінімізації значення функції багатьох змінних не гарантує знаходження глобального мінімуму енергії. Тому за відсутності апріорного знання про енергетичні стани конкретної системи можна сформулювати наступну стратегію пошуку стану з найменшою енергією (метод мультистарту):

1. Знайти локальні мінімуми енергії, запускаючи числовий процес із різних початкових станів:
   - Випадковий розподіл (таких запусків варто робити декілька).
   - Стани із наперед відомими симетріями (наприклад, для феромагнітного прямого дроту це означає напрямок $\boldsymbol{m}$ вздовж дроту, або для кільця — вихровий стан).
   - Стани вздовж осей лабораторної системи координат для уникнення можливих точок нестійкої рівноваги.
2. Зі знайдених станів відкинути однакові.

3. Обрати стан з найменшою енергією і проаналізувати його. Зауважимо, що кінцевий стан може бути принципово іншої симетрії, аніж початковий, і робити припущення про кінцевий стан виключно зі значення енергії в загальному випадку не можна.

Хоча чисельні методи не дозволяють гарантувати глобальність знайденого мінімуму енергії, це може випливати з властивостей конкретної фізичної задачі. Слід також враховувати, що одному значенню енергії може відповідати кілька різних станів. Наприклад, зміна напрямку стрілок у вихровому стані на протилежний (рис. 3.7(c)), або поворот стану «onion» (рис. 3.7(a)) як єдиного цілого не вплине на енергію. Такі стани називають виродженими. Зокрема, поворот «onion» відповідає так званій Голдстоунівській моді. Великий ступінь виродження мають фрустровані системи (спіновий лід, системи спінів, що дипольно взаємодіють, тощо).

3.7 Задачі

№14. Запишіть $O(\bullet)$ для таких виразів:

1. $1 + 0.01n^4 + 0.1n$
2. $n^3 + 1000n^2$
3. $n \log n + \log^2 n$
4. $10n \log n + 0.003n^2$

№15. Оцініть час виконання та складність наступних кодів в O-нотації:

```
# cities = ['Kyiv', 'Berlin', 'Shanghai', ...]
hash = 0
for i, city in enumerate(cities):
  hash += i**2 + len(city)
```

Лістинг 3.1: Код 1

```
hash = n**2
for i in range(n):
  for j in range(n):
    hash = hash ^ (3*(i + j))
```

Лістинг 3.2: Код 2

№16. Розв'яжіть рівняння Пуассона(3.15) із такими джерелами і межовими умовами Діріхле на відрізку $x \in [0, 1]$:

1. $f(x) = \cos \pi x$, $\mu_0 = \mu_1 = 1$
2. $f(x) = x^2 \sin \pi x/2$, $\mu_0 = 0$, $\mu_1 = 1$
3. $f(x) = x(1 - x)^2$, $\mu_0 = \mu_1 = 1$
4. $f(x) = (1 - x^2)e^x$, $\mu_0 = 1$, $\mu_1 = 0$

№17. Розв'яжіть рівняння теплопровідності (3.19) без джерела на відрізку $x \in [0, 1]$ із такими параметрами:

1. $g(x) = \sin 3\pi x,\ \mu_0(t) = \mu_1(t) = 0$
2. $g(x) = \cos^2 \pi x,\ \mu_0(t) = 1 - e^{-t},\ \mu_1(t) = (1 - 2e^{-t})$
3. $g(x) = x^2 \sin \pi x/2,\ \mu_0(t) = 0.1t,\ \mu_1(t) = e^{-t}$
4. $g(x) = (x^2 - 1) \sin \pi x,\ \mu_0(t) = t,\ \mu_1(t) = 0.1t$

№18. Перепишіть вираз (3.45) для анізотропії типу легка вісь $\boldsymbol{e}_\mathrm{a} \equiv \boldsymbol{e}_x$, аби уникати точок нестійкої рівноваги.

№19. Енергія анізотропії. Напишіть код, який реалізує обчислення енергії анізотропії (3.45) та обчислює її градієнт на сітці з N вузлів. Передбачте можливість задати власний напрямок осі анізотропії в кожному вузлі (вектор осі анізотропії має бути представлений масивом так само, як і $\boldsymbol{m}$).

1. Покажіть, що вираз сягає мінімуму при $\boldsymbol{m} = \pm\boldsymbol{e}_\mathrm{a}$ і $K > 0$.
2. Покажіть, що енергія анізотропії рівна нулю, якщо $K < 0$, а $\boldsymbol{m} \perp \boldsymbol{e}_\mathrm{a}$.
3. Модифікуйте код на випадок періодичних межових умов.

№20. Обмінна енергія. Напишіть код, який реалізує обчислення енергії анізотропії (3.49) та обчислює її градієнт на сітці з N вузлів.

1. Покажіть, що вираз сягає мінімуму при $\boldsymbol{m} = const$.
2. Модифікуйте код на випадок кільця із періодичними межовими умовами.

№21. Прямий дріт. Напишіть код, який дозволить знайти рівноважні енергетичні стани прямого феромагнітного дроту з $\widetilde{A} \sim 1$ і $\widetilde{K} \sim 4$ із $\boldsymbol{e}_\mathrm{a} = \{1, 0, 0\}$ якщо дріт напрямлено вздовж осі x. Встановіть залежність ширини зрелаксованої доменної стінки від кроку сітки h при сталій довжині зразка, підбираючи параметри пробної функції $f(x) = \pm \operatorname{th}(x - x_0)/w$.

№22. Кільце. Використовуючи параметри $\widetilde{A} \sim 1$ і $\widetilde{K} \sim 4$ з анізотропією (3.54) обчисліть енергію стану «onion» для кількох значень радіусу в діапазоні $R \in [0.1R_c, 3R_c]$ використавши однорідний стан вздовж осі $\boldsymbol{e}_y$ як початковий. Повторіть те саме з вихровим початковим станом ($\boldsymbol{m} = \pm\boldsymbol{e}_\mathrm{a}$) й знайдіть радіус кільця, за якого енергії стають рівними. Виразіть цей радіус в одиницях ℓ.

3.8 Take home message

1. Метод скінченних різниць для задач, де можна ввести ортогональну, однорідну сітку або зробити відповідне перетворення геометрії для її введення.
2. Явні різницеві схеми суттєво простіші за неявні в своєму технічному виконанні, але вимагають дрібних кроків для забезпечення стійкості розв'язку.
3. Знаходження рівноважних станів певної фізичної системи шляхом дискретизації інтегралу енергії на заданій сітці з подальшою мінімізацією функції багатьох змінних.
4. Метод мультистарту для пошуку стану з найменшою енергією як найбільш імовірного на глобальний мінімум.

Посилання

3. *Fehlberg E.* Klassische Runge-Kutta-Nyström-Formeln mit Schrittweiten-Kontrolle für Differentialgleichungen $\ddot{x} = f(t, x)$ // Computing. — 1972. — Груд. — Т. 10, вип. 4, № 4. — С. 305—315. — ISSN 0010-485X. — DOI: 10.1007/bf02242243. — URL: http://dx.doi.org/10.1007/bf02242243.
4. *Simos T.*, *Dimas E.*, *Sideridis A.* A Runge–Kutta–Nyström method for the numerical integration of special second-order periodic initial-value problems // Journal of Computational and Applied Mathematics. — 1994. — Т. 51, вип. 3, № 3. — С. 317—326. — ISSN 0377-0427. — DOI: 10.1016/0377-0427(92)00114-o. — URL: http://dx.doi.org/10.1016/0377-0427(92)00114-o.
5. Improving the accuracy of simulation of radiation-reaction effects with implicit Runge-Kutta-Nyström methods / N. V. Elkina, A. M. Fedotov, C. Herzing, H. Ruhl // Physical Review E. — 2014. — Т. 89, вип. 5, № 5. — С. 053315. — ISSN 1539-3755. — DOI: 10.1103/physreve.89.053315. — URL: http://dx.doi.org/10.1103/physreve.89.053315.
6. Efficient Two-Derivative Runge-Kutta-Nyström Methods for Solving General Second-Order Ordinary Differential Equations $y''(x) = f(x, y, y')$ / T. S. Mohamed, N. Senu, Z. B. Ibrahim, N. M. A. Nik Long // Discrete Dynamics in Nature and Society. — 2018. — Т. 2018. — С. 1—10. — ISSN 1026-0226. — DOI: 10.1155/2018/2393015. — URL: http://dx.doi.org/10.1155/2018/2393015.
7. *Slastikov V. V.*, *Sonnenberg C.* Reduced models for ferromagnetic nanowires // IMA Journal of Applied Mathematics. — 2012. — Квіт. — Т. 77, вип. 2, № 2. — С. 220—235. — ISSN 1464-3634. — DOI: 10.1093/imamat/hxr019. — URL: http://dx.doi.org/10.1093/imamat/hxr019.
8. *Sheka D. D.*, *Kravchuk V. P.*, *Gaididei Y.* Curvature effects in statics and dynamics of low dimensional magnets // Journal of Physics A: Mathematical and Theoretical. — 2015. — Т. 48, вип. 12, № 12. — С. 125202. — DOI: 10.1088/1751-8113/48/12/125202. — URL: http://stacks.iop.org/1751-8121/48/i=12/a=125202.

Література до розділу

1. *Fornberg B.* Generation of finite difference formulas on arbitrarily spaced grids // Mathematics of Computation. — 1988. — Т. 51, вип. 184, № 184. — С. 699—699. — DOI: 10.1090/s0025-5718-1988-0935077-0. — URL: https://doi.org/10.1090/s0025-5718-1988-0935077-0.
2. *Самарский А. А.*, *Гулин А. В.* Численные методы: Учебное пособие для вузов. — М. : Наука. Гл. ред. физ-мат. лит., 1989. — С. 432. — ISBN 5-02-013996-3.
9. *Hutchinson I. H.* A Student's Guide to Numerical Methods. — Cambridge University Press, 05.11.2018. — ISBN 1107095670. — URL: https://www.ebook.de/de/product/23636703/ian_h_hutchinson_a_student_s_guide_to_numerical_methods.html.

10. *Harmuth H. F.*, *(Eds.) B. M.* Calculus of Finite Differences in Quantum Electrodynamics. — Academic Press, 2003. — (Advances in Imaging and Electron Physics 129). — ISBN 978-0-12-014771-7.

11. *Boole G.* A Treatise on the Calculus of Finite Differences. — Cambridge University Press, 2009. — DOI: 10.1017/cbo9780511693014. — URL: https://doi.org/10.1017/cbo9780511693014.

12. *Iserles A.* A First Course in the Numerical Analysis of Differential Equations. — Cambridge : Cambridge University Press, 2008. — DOI: 10.1017/cbo9780511995569. — URL: http://dx.doi.org/10.1017/cbo9780511995569.

13. *Süli E.*, *Mayers D. F.* An Introduction to Numerical Analysis. — Cambridge University Press, 2003. — DOI: 10.1017/cbo9780511801181. — URL: http://dx.doi.org/10.1017/cbo9780511801181.

Розділ 4

Консервативні методи обчислень

4.1 Консервативні та неконсервативні методи обчислень

Як було зазначено у попередньому розділі, основними перевагами методу скінченних різниць є простота ідеї та технічної імплементації. Водночас це твердження стосується лише простих геометрій області Ω, поведінка параметру порядку на яких досліджується: наприклад, відмінність Ω від прямокутної у двовимірному випадку суттєво ускладнює імплементацію межових умов і підвищує вартість тестування фінального програмного продукту. Особливу незручність це становить у тих випадках, коли не лише параметри задачі (наприклад, конкретний вигляд функції початкових умов у рівнянні дифузії), а й область аналізу можуть бути різні в рамках одного дослідження й тому має сенс використовувати один і той самий код.

Варто зазначити, що формулювання задачі у термінах різницевих схем має ще одну неочевидну проблему: закони збереження, справедливі для неперервної задачі можуть порушуватися (і, як правило, порушуються) для їх різницевих аналогів. Так, врахування цього є важливим у задачах зі збереженням енергії чи потоку речовини.

Примітка 14. Моделювання неконсервативних систем

Задачі мікромагнетизму, як правило, базуються на рівнянні Ландау—Ліфшиця із релаксаційним доданком $\boldsymbol{R}(\boldsymbol{m})$:

$$\partial_t \boldsymbol{m} = \frac{\gamma_0}{M_\mathrm{s}} \boldsymbol{m} \times \frac{\delta E}{\delta \boldsymbol{m}} + \boldsymbol{R}(\boldsymbol{m}), \tag{4.1}$$

де γ_0 — гіромагнітне відношення і M_s — намагніченість насичення. Наявність релаксаційного доданка робить систему неконсервативною, тобто для підтримки магнітної динаміки необхідне накачування енергією зовні, наприклад, зовнішнім магнітним полем. Як правило, релаксацію завжди враховують у практичних задачах. Тому з одного боку, це дає можливість не ускладнювати технічну реалізацію вимогою на збереження енергії та моделюванням динаміки отримувати рівноважні магнітні текстури (дина-

міка обов'язково зупиняється після дисипації надлишку енергії у системі), але з іншого, накладає обмеження на дослідження власних коливальних мод. З технічної точки зору, наявність релаксації у формулюванні задачі в першу чергу стабілізує її відносно флуктуацій чисельного характеру.

Вищевказані аспекти обмежують універсальність відповідних різницевих схем і ставлять задачу пошуку альтернативних методів розв'язання диференційних рівнянь у частинних похідних. Особливо важливим це є для задач з жорсткими вимогами на збереження потоку чи кількості речовини.

Методи обчислень, в яких виконуються закони збереження, називають *консервативними* або *дивергентними*. Часто їх будують формулюючи задачі у так званій дивергентній формі, тобто рівняннями в яких похідна по просторовій координаті входить з коефіцієнтом 1. Такі рівняння зазвичай відповідають тим чи іншим законам збереження в диференціальній формі, наприклад рівняння неперервності:

$$\partial_t u + \nabla \cdot \boldsymbol{j}(u) = 0, \tag{4.2}$$

де $\boldsymbol{j}$ характеризує густину потоку певної величини із густиною u. Якщо у якості цієї величини виступає енергія, то вираз (4.2) каже про її незмінність. Застосування теореми Остроградського—Гауса до (4.2) по деякому невеликому об'єму дозволяє отримати вирази на потоки, які втікають або витікають з нього. Тому дискретизація дивергентних рівнянь дозволяє зберегти баланс потоку величини u.

У рамках неперервної моделі дивергентну та недивергентну форму рівняння завжди можна перетворити одну в одну як тотожність. Різниця між ними для чисельних обчислень полягає в тому, що дискретні форми дивергентних і дивергентних рівнянь не обов'язково перетворюються одна в іншу, як неперервні. Наслідком цього є збереження інтегральних характеристик (типово, енергії) у дискретній моделі дивергентних рівнянь (можливо, із порушенням балансу внутрішньої енергії) і навпаки в дискретній моделі недивергентних виразів [1]. Принципово, можна записати методи чисельного розв'язання задач, у яких обидві форми вихідних співвідношень будуть задовольнятись. Їх називають *повністю консервативними методами*.

Виконання законів збереження у дискретизованих моделях фізичних систем та довільність досліджуваної геометрії можна забезпечити у так званому *методі контрольних об'ємів* (англ. finite volume method, FVM, також інколи вживається термін «метод скінченних об'ємів»). Ця вимога є актуальною в задачах на дослідження потоків речовини й енергії (зокрема у гідрогазодинаміці).

4.2 Розривні коефіцієнти у фізичних задачах

У попередньому розділі розглядалися приклади зі сталими коефіцієнтами: як рівняння Пуассона та дифузії, так і задачі на мінімізації енергії магнітних нанодротів. Практичні задачі часто формулюються у термінах коефіцієнтів, які залежать від координати. Зокрема, ця зміна може супроводжуватися розривом

функції чи її похідних. Це можуть бути змінні коефіцієнти дифузії чи теплопровідності, варіація матеріальних параметрів у гранульованих матеріалах тощо. Математичне формулювання відповідних рівнянь вимагає явного врахування умов зшивання на межах відповідних областей, або формулювання відповіді у термінах узагальнених функцій. Розглянемо, як відсутність неперервної диференційовності може позначитись на результаті інтегрування диференційного рівняння різницевими методами.

Для прикладу розглянемо одновимірну стаціонарну задачу теплопровідності (дифузії). Якщо потік величини u рівний $W = -k(x)\partial_x u(x)$, то рівняння балансу на нескінченно малому проміжку $\mathrm{d}x$ рівне

$$\mathrm{d}W = -\mathrm{d}\left(k(x)\partial_x u\right) = f(x)\mathrm{d}x, \tag{4.3}$$

де $k(x)$ — коефіцієнт теплопровідності, а $f(x)$ — функція розподілу джерел. У випадку межових умов Діріхле на відрізку це дає крайову задачу

$$\begin{gathered}\partial_x\left(k\partial_x u\right) = -f(x), \quad x \in [0, L], \\ u(0) = \mu_1, \quad u(L) = \mu_2.\end{gathered} \tag{4.4а}$$

Загальний підхід до чисельного розв'язання такої задачі описано у попередньому розділі. Обмежимось випадком $f(x) = 0$ і

$$k(x) = \begin{cases} k_1, & x < L/2, \\ k_2, & x > L/2 \end{cases} \tag{4.4б}$$

із довільними додатними сталими $k_{1,2}$ та точкою зшивання $x_c = L/2$. Аналітичний розв'язок має вигляд

$$u(x) = \begin{cases} \mu_1 + \dfrac{2k_2(\mu_2 - \mu_1)}{(k_1 + k_2)L}x, & x \in [0, L/2], \\ \dfrac{2k_1\mu_1 - k_1\mu_2 + k_2\mu_2}{k_1 + k_2} + \dfrac{2k_1(\mu_2 - \mu_1)}{(k_1 + k_2)L}x, & x \in (L/2, L]. \end{cases} \tag{4.5}$$

Легко пересвідчитись, що він узгоджується із розв'язком неперервно диференційовної задачі при $k_1 = k_2$.

Примітка 15. Диференціальні рівняння, розв'язки яких вимагають зшивання

Наслідком недиференційовності функції $k(x)$ у точці x_c є необхідність врахувати умови зшивання функції $u(x)$ та її похідної. Оскільки розв'язок шукається у класі неперервних функцій, повинна виконуватись умова

$$[u(x)]|_{x=x_c} \equiv u(x_c - 0) - u(x_c + 0) = 0. \tag{4.6а}$$

Інтегрування рівняння (4.4а) в нескінченно малому околі точки недиференційовності дає умову

$$[k\partial_x u]|_{x=x_c} \equiv (k\partial_x u)|_{x=x_c+0} - (k\partial_x u)|_{x=x_c-0} = 0. \tag{4.6б}$$

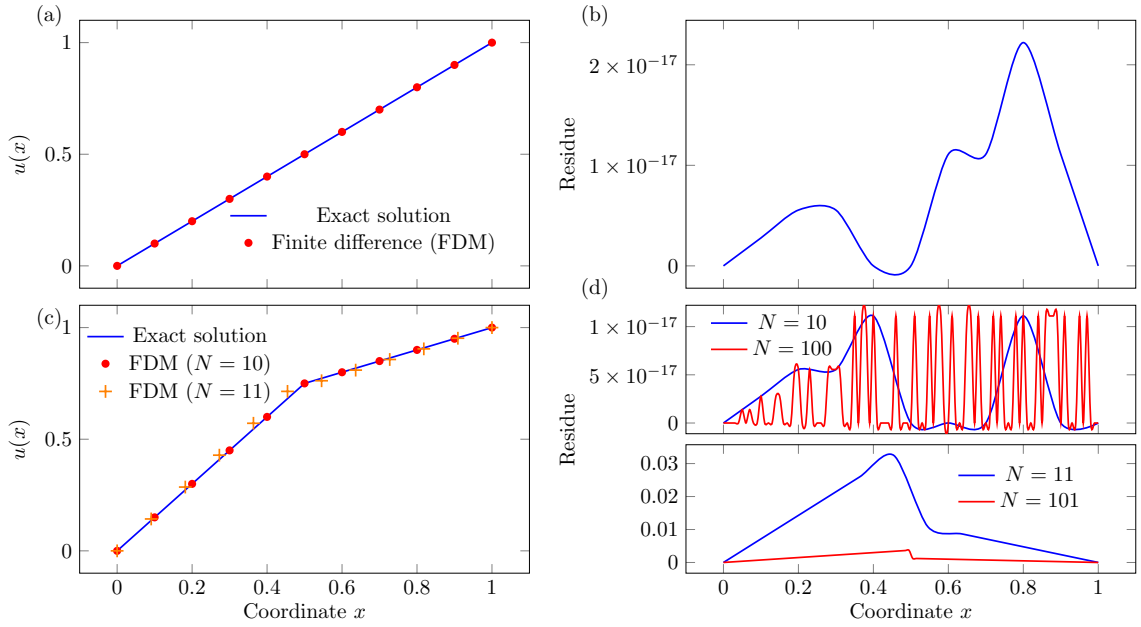


Рис. 4.1: (a) Розв'язок задачі (4.4a) із неперервно диференційовним коефіцієнтом $k_1 = k_2 = 1$ та межовими умовами $\mu_1 = 0$, $\mu_2 = 1$ на відрізку $x \in [0, 1]$ за допомогою методу скінченних різниць. (b) Абсолютне значення нев'язки $|u_{\text{numerical}}(x) - u_{\text{exact}}(x)|$ чисельного розв'язку на сітці з 11 вузлів. (c,d) Те саме, але за відсутності неперервної диференційовності із $k_1 = 1$, $k_2 = 3$ та різних кроків дискретизації. Результатом є суттєва немонотонність похибки обчислень як функції від кроку дискретизації: амплітуда нев'язки змінюється більше, аніж на 10 порядків для парної та непарної кількості вузлів.

Відповідну різницеву схему запишемо скориставшись виразом різницевої похідної від добутку відомої функції на похідну шуканої (3.7):

$$\begin{gathered} k^{i+1/2}u^{i+1} - (k^{i+1/2} + k^{i-1/2})u^i + k^{i-1/2}u^{i-1} = 0, \quad i = \overline{1, N-1} \\ u^0 = \mu_1, \quad u^N = \mu_2. \end{gathered} \tag{4.7}$$

При $k_1 = k_2$ (неперервно диференційовна залежність) різницева схема працює із високою точністю, див. рис. 4.1(a,b), де похибка чисельного обрахунку знаходиться на межі числового шуму. Внесемо неаналітичність взявши $k_2 = 3$. Як видно із порівняння на рис. 4.1(c,d), точність різницевої схеми починає немонотонно залежати від кількості вузлів: для непарної кількості вузлів (парне N) точність висока, у той час, як для парної кількості вузлів вона погіршується на 15 порядків. З постановки задачі видно, що це пов'язано із тим, чи потрапляє точка зшивання $x_c = L/2$ на вузол, чи ні. Якщо x_c припадає на вузол, то отримується коректний результат.

Коментар 14. Сітка для областей з кусково-диференційовними параметрами

Якщо область, на якій розв'язується диференціальне рівняння, характеризується кусково-диференційовними функціями, то при реалізації звичайної різницевої схеми вузли сітки повинні збігатися з областями спряження областей диференційовності (наприклад, точки для одновимірних задач і криві для двовимірних). Якщо така область потрапляє у міжвузля, різницева схема стає еквівалентною до задачі, в якій умови узгодження (4.6) містять фіктивні доданки (джерела), величина яких визначається зсувом області спряження відносно вузлів.

Варто зауважити, що абсолютна похибка у 3%, показана на рис. 4.1(d) може виглядати порівняно прийнятною, але на практиці виявитись визначальною для фізичних висновків. По-перше, при пошуку енергетичних мінімумів за порівняння різних рівноважних станів, різниця між станами може виявитись суттєво меншою через визначеність енергії з точністю до певного сталого доданку. По-друге, не завжди вдається забезпечити однакові сітки (або принаймні кроки дискретизації) при моделюванні різних зразків у межах одної задачі. У таких ситуаціях стійкість чисельного методу є принциповою для оцінки систематичної похибки. Окрім цього, зменшення похибки при збільшенні щільності сітки хоча й може використовуватись для аналізу стійкості конкретної схеми у тих випадках, коли її важко довести аналітично, в загальному випадку не гарантує асимптотичного зменшення до нуля і можна підібрати приклади, коли похибка виявиться як завгодно великою.

Примітка 16. Фазові діаграми

Ілюстрацією до останнього можуть слугувати побудови фазових діаграм методом скінченних елементів, який буде обговорюватися далі. Типова двовимірна фазова діаграма у фізиці твердого тіла показує рівноважні стани речовини або об'єктів при зміні певних двох параметрів за сталих інших (наприклад, зміна температури і зовнішнього магнітного поля, або товщина і ширина нанооб'єкту). Чисельна побудова фазових діаграм передбачає визначення кроку по змінним параметрам (наприклад, зміна температури і поля з кроками 10 K і 200 мТ відповідно), обрахунок різних станів у кожній точці діаграми з однаковою сіткою та вибір з них оптимального. Нестійкість чисельного методу відносно сітки означає, що безпосереднє порівняння моделювань у різних точках діаграми (різних сітках дискретизації) є неможливим. Так, відносні й абсолютні зміни енергії системи, отримані на різних нерегулярних сітках можуть виявитись немонотонними залежностями параметрів фазової діаграми навіть якщо для цього нема ніяких фізичних підстав.

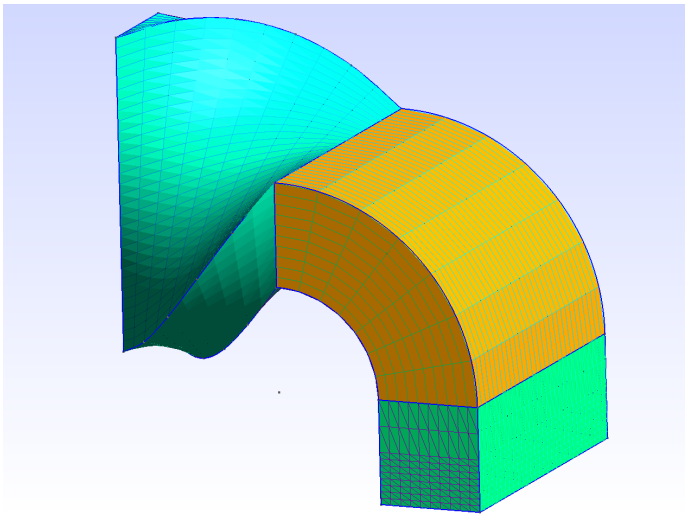


Рис. 4.2: Приклад регулярної сітки на викривленій геометрії, згенерований у Gmsh [2]. Частини деталі, які було згенеровано як окремі області з власною дискретизацією, зафарбовано різними кольорами. Сині та зелені лінії на них вказують на форму елементів сітки на поверхні.

4.3 Метод контрольних об'ємів (МКО)

Позбутись обмежень, пов'язаних із вимогами на дискретизацію області інтегрування, проілюстрованих вище, можна за допомогою *методу контрольних об'ємів*, який також зручний для проектування консервативних методів обчислень. Якщо конкретна реалізація МКО консервативна, у вітчизняній літературі її часто називають методом балансу або інтегро-інтерполяційним. Суть методу полягає у розбитті області інтегрування на невеликі ділянки (контрольні об'єми), для кожної з яких можна вимагати виконання заданих законів збереження. У кожній ділянці є лише одна точка, до якої прив'язується значення шуканої величини. Варто зазначити, що у літературі під МКО часто розуміють виключно консервативні схеми, але сама процедура розбиття області інтегрування на контрольні об'єми не гарантує консервативності сама по собі і може застосовуватись для будь-якого диференціального рівняння.

Перед тим, як обговорювати побудову МКО, розглянемо особливості дискретизації та опису простору, на якому задані диференціальні рівняння.

4.3.1 Дискретизація області аналізу

Довільність області аналізу призводить до появи двох типів сіток, які можуть бути використані: регулярних та нерегулярних. Регулярні сітки можуть бути побудовані із використанням симетрії області. Сітки із використанням криволінійних паралелепіпедів споріднені зі способом дискретизації для методу скінченних різниць, див. рис 4.2. Часто вживаними, особливо для випадків, які вимагають рівномірної *у середньому* дискретизації, є сітки на основі трикутників для двовимірних областей і тетраедрів для тривимірних.

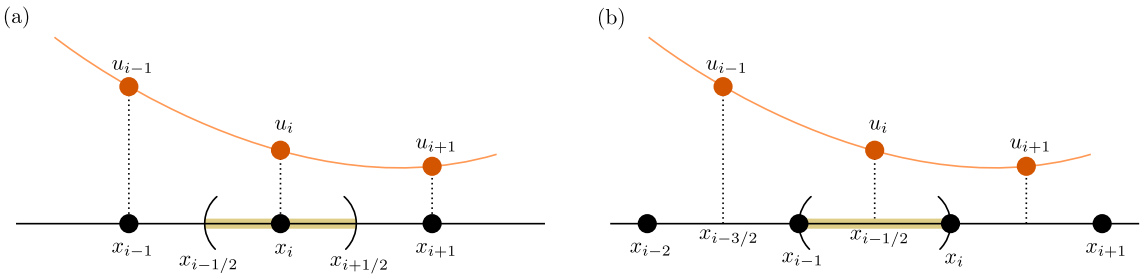


Рис. 4.3: Контрольні об'єми у одновимірній задачі. Значення функції позначено помаранчевим кольором, а область контрольного об'єму — жовтим поміж дужок. (a) Контрольний об'єм центрований на вузлі. (b) Контрольний об'єм центрований на комірці.

Є кілька способів введення контрольних об'ємів на сітці. У одновимірному випадку, очевидно, елементами сітки будуть відрізки, які сполучають сусідні вузли. У такому випадку контрольний об'єм центрований на вузлі буде обмежено центрами сусідніх відрізків, див. рис. 4.3(a) (контрольні об'єми центровані на вузлах). Альтернативно, можна вибрати за контрольний об'єм сам відрізок, див. рис. 4.3(b) (контрольні об'єми центровані на комірках). Різниця між цими способами полягає у інтерпретації країв області. Так при центруванні на вузлі межові об'єми будуть відрізнятись від об'ємів всередині області. Інколи специфіка межових умов конкретної задачі може робити один з цих варіантів більш зручним з технічної точки зору. Зауважимо, що як буде показано далі, за будь-якого випадку самі значення функції треба розуміти як усереднені по контрольному об'єму і тому зручно асоціювати значення шуканих величин із його центром мас.

Двовимірні та тривимірні задачі залишають більше можливостей для визначення контрольних об'ємів. Перший спосіб аналогічний центруванню на комірці з одновимірного випадку: за контрольний об'єм обирається сама комірка, див. рис. 4.4(a). За точку обчислення функції доцільно брати центр мас комірки, який повинен лежати всередині неї. Ці вимоги не завжди легко виконати на межі складних геометричних областей. Для двовимірного і тривимірного випадків відповідно це означає, що оптимальною формою елементів сітки будуть правильні трикутники та тетраедри.

Коментар 15. Інтегро-диференціальні задачі

Регулярність сітки може грати визначальну роль у задачах, які описуються інтегро-диференціальними рівняннями. Зокрема, у магнетизмі до них призводить врахування магнетостатичної взаємодії. У загальному випадку це означає наявність будь-якої нелокальної у просторі чи часі взаємодії, що означає ускладнення задачі від $O(N)$ до $O(N^2)$ за кількістю точок, в яких відбувається обчислення параметру порядку. Для пришвидшення таких обрахунків використовуються спеціальні математичні методи (наприклад, за допомогою перетворення Фур'є [3] чи із використанням

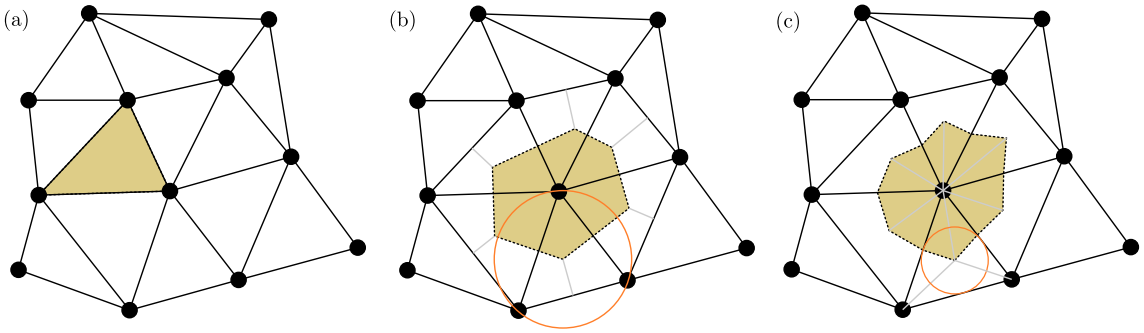


Рис. 4.4: Контрольні об'єми у двовимірній задачі із трикутною сіткою. Вузли сітки позначені жирними точками (a) Контрольний об'єм (жовтий) центрований на комірці. (b) Контрольний об'єм центрований на вузлі та побудований на центрах описаних кіл (одне з них позначено помаранчевим). Перпендикуляри до сторін трикутників позначено сірим. (c) Контрольний об'єм центрований на вузлі та побудований на центрах вписаних кіл (одне з них позначено помаранчевим). Бісектриси кутів трикутників позначено сірим. Неопуклість даного елементу може негативно позначитись на точності обчислень.

$\mathcal{H}^2$-матриць [4]). При цьому більша регулярність сітки дозволяє зменшити кількість ненульових елементів допоміжних матриць, що економить пам'ять і час обрахунку.

За центрування на вузлі, контрольний об'єм навколо вузла сітки можна виділити побудувавши діаграму Вороного, як показано для трикутників на рис. 4.4(b). У цьому випадку для кожного трикутника визначається центр описаного кола (перетин перпендикулярів до центрів сторін) і контрольний об'єм утворюється ламаною, що з'єднує ці центри. Так само можна з'єднувати центри сторін із центрами вписаних кіл, див. рис. 4.4(c).

Як і в одновимірному випадку, жоден з вищеописаних методів не має однозначної переваги й зручність використання може визначатись особливостями конкретної задачі. Так, у задачах гідрогазодинаміки, в яких відомі компоненти швидкостей на межі області, а тиск є невідомим, зручніше використовувати контрольні об'єми, побудовані на центрах описаних кіл (центрах мас), оскільки у такому випадку кількість точок для обчислення тиску збігатиметься із кількістю точок для обчислення швидкостей. У інших випадках системи рівнянь можуть виявитись недо- або перевизначеними.

4.3.2 Обчислення геометричних параметрів

Визначення усереднених значень тих чи інших параметрів по контрольному об'єму може здійснюватися аналітично (за можливості), або чисельно із використанням відповідних квадратурних формул. Типова квадратурна формула

для обчислення інтегралу функції $f(\boldsymbol{r})$ по контрольному об'єму Ω_0 має вигляд

$$\int f(\boldsymbol{r})d\Omega_0 \approx \sum_i w_i(\boldsymbol{r}_i)f(\boldsymbol{r}_i), \tag{4.8}$$

де $w_i(\boldsymbol{r}_i)$ — вагові коефіцієнти, які можуть бути сталими або функціями координати, а $\boldsymbol{r}_i \in \Omega_0$ — квадратурні вузли. Найпростішими квадратурними формулами є формули Ньютона—Котеса, які придатні для випадку рівномірного розподілу квадратурних вузлів по Ω_0. В загальному випадку більш доцільним може бути використання інших квадратурних формул, наприклад Гауса. Також зауважимо, що формули Ньютона—Котеса не придатні для великої кількості квадратурних вузлів, оскільки швидко втрачають стійкість. Окрім цього слід враховувати, що типова точність методів контрольних об'ємів є квадратичною по кроку дискретизації, оскільки реалізація вищих порядків точності є занадто громіздкою і рідко виправданою для практичних застосунків.

У одновимірних задачах формули Ньютона—Котеса для двох і трьох квадратурних вузлів відповідають відомим правилам інтегрування трапецією та Сімпсона. Для відрізка $x \in [x_1, x_2]$ вони мають вигляд [5]

$$\int_{x_1}^{x_2} f(x)dx \approx \frac{x_2 - x_1}{2}\left(f^1 + f^2\right) + O(h^3) \quad \text{(формула трапецій)}, \tag{4.9}$$

$$\int_{x_1}^{x_2} f(x)dx \approx \frac{x_2 - x_1}{6}\left(f^1 + 4f^{12} + f^2\right) + O(h^5) \quad \text{(формула Сімпсона)}, \tag{4.10}$$

де f^{12} обчислюється у центрі відрізка $x_{12} = (x_1+x_2)/2$, див. рис. 4.5(а). Ці вирази відносяться до так званих закритих форм, оскільки враховують значення на межах інтервалів. Нагадаємо, що формула трапецій дає точну відповідь якщо $f(x)$ є лінійною функцією, а Сімпсона — кубічним поліномом.

Гаусові квадратурні формули призначені для високоточного обчислення інтегралів за допомогою спеціальних інтерполяційних поліномів, що дозволяють зменшити кількість обчислень функції. Їх зручно записати для відрізка $\Omega_0^{\text{ref}} = [-1, 1]$, див. рис. 4.5(а). Довільний відрізок $\Omega_0 = [x_1, x_2]$ можна відобразити у нього як

$$\Omega_0 \to \Omega_0^{\text{ref}}: \ \left\{ y = \frac{2}{x_2 - x_1}x + \frac{x_1 + x_2}{x_1 - x_2}, \quad x \in [x_1, x_2],\ y \in [-1, 1] \right\}. \tag{4.11}$$

Тоді відповідні квадратурні формули по двох і трьом точкам матимуть вигляд

$$\int_{x_1}^{x_2} f(x)dx = \int_{-1}^{1} \widetilde{f}(y)dy \approx \widetilde{f}\left(-\frac{1}{\sqrt{3}}\right) + \widetilde{f}\left(\frac{1}{\sqrt{3}}\right), \tag{4.12}$$

$$\int_{x_1}^{x_2} f(x)dx = \int_{-1}^{1} \widetilde{f}(y)dy \approx \frac{5}{9}\widetilde{f}\left(-\sqrt{3/5}\right) + \frac{8}{9}\widetilde{f}\left(0\right) + \frac{5}{9}\widetilde{f}\left(\sqrt{3/5}\right), \tag{4.13}$$

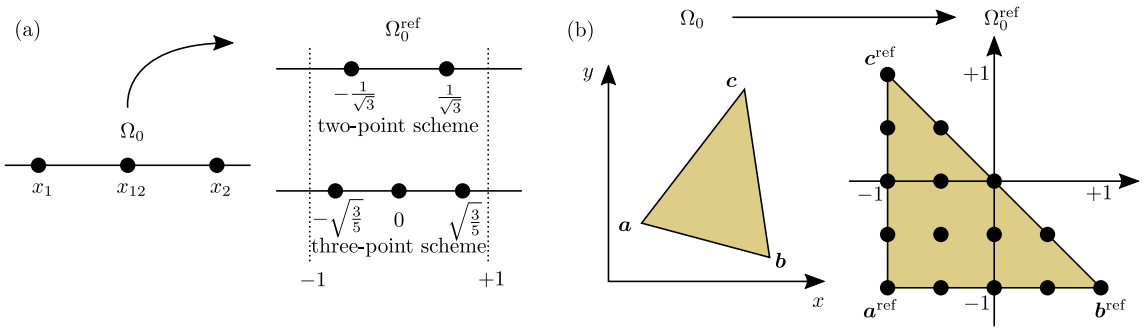


Рис. 4.5: Схема розподілу квадратурних вузлів у контрольному об'ємі. (a) Одновимірна задача: для інтегрування методом трапецій використовуються вузли x_1 і x_2, а у методі Сімпсона до них додається середній вузол x_{12}. Інтегрування квадратурними формулами Гауса виконується при переході до референсного простору Ω_0^{ref}, що представлено відрізком $[-1, 1]$. (b) Схема чисельного інтегрування на трикутній сітці. Трикутник з вершинами у точках $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ відображається на прямокутний рівнобедрений у референсному просторі, для якого й записуються відповідні квадратурні формули. На рисунку показано приклад розміщення 15 квадратурних вузлів для формул Ньютона—Котеса 4 порядку точності.

та $\widetilde{f}(y) \equiv \frac{x_2-x_1}{2} f\left(\frac{1}{2}(x_1 + x_2 - x_1 y + x_2 y)\right)$. Для обчислення інтегралу по n точкам на інтервалі квадратурні формули Гауса є точними при інтегруванні поліномів степеня $2n-1$, тобто двохточкова формула Гауса еквівалентна трьохточковій формулі Сімпсона. Порівняння обрахунків наведеними виразами показано у таблиці 4.3.2. Звертаємо увагу, що формула Сімпсона і квадратурна формула Гауса за двома точками мають однакову похибку, але друга потребує меншої кількості обчислень значення функції.

Обчислення двовимірних інтегралів проілюструємо на прикладі трикутних елементів сітки. Нехай квадратурні вузли $\boldsymbol{r}_{1,2,3}$ відповідають вершинам заданого трикутного контрольного об'єму. Подібно до інтегрування квадратурними формулами Гауса, його можна звести до прямокутного трикутника у референ-

Табл. 4.1: Порівняння точності квадратурних формул на прикладі функції $f(x) = x \sin x$, яка інтегрується на інтервалі $x \in [0.1, 1.3]$. У останній колонці півжирним виділені знаки, які збігатися з аналітичним обрахунком $\int_{x_1}^{x_2} f(x)dx = \sin x - x\cos x\big|_{x_1}^{x_2} \approx 0.615477$.

Формула	К-ть точок	Значення	Точність
трапецій (4.9)	2	**0.**757565	23%
Сімпсона (4.10)	3	**0.61**3284	0.36%
квадратура Гауса (4.12)	2	**0.61**6945	0.23%
квадратура Гауса (4.13)	3	**0.6154**69	0.001%

сній системі координат Ω_0^{ref} у з вершинами у точках $\boldsymbol{a}^{\text{ref}}(-1,-1)$, $\boldsymbol{b}^{\text{ref}}(1,-1)$ та $\boldsymbol{c}^{\text{ref}}(-1,1)$, див. рис. 4.5(b). Якщо позначити точки початкового трикутника як $\boldsymbol{a} = (a_x, a_y)$, $\boldsymbol{b} = (b_x, b_y)$ і $\boldsymbol{c} = (c_x, c_y)$, то лабораторна система координат буде виражатись через референсну як

$$\begin{cases} x(\widetilde{x},\widetilde{y}) = a_x + (\widetilde{x}+1)\dfrac{b_x - a_x}{2} + (\widetilde{y}+1)\dfrac{c_x - a_x}{2}, \\ y(\widetilde{x},\widetilde{y}) = a_y + (\widetilde{x}+1)\dfrac{b_y - a_y}{2} + (\widetilde{y}+1)\dfrac{c_y - a_y}{2}. \end{cases} \tag{4.14}$$

Відповідний якобіан переходу між системами координат рівний $V_{\text{tri}} = |(b_x - a_x)(c_y - a_y) - (b_y - a_y)(c_x - a_x)|/4$.

Формули Ньютона—Котеса легко вивести для покриття референсного трикутника квадратною сіткою вузлів, які потрапляють як всередину трикутника, так і на його сторони. Якщо на основу трикутника припадає n вузлів, то усього у сітці їх буде $n(n+1)/2$, а порядок точності відповідної схеми складатиме $n-1$. Так, формула (4.8) набуває вигляду

$$\int_{-1}^{1} dx \int_{-1}^{-x} dy \varphi(x,y) \approx \sum_k w_k \varphi(x_k, y_k), \tag{4.15}$$

де індекс k пробігає по усім квадратурним вузлам (див. рис. 4.5(b)), а замість $\phi(x,y)$ необхідно підставити $n(n+1)/2$ лінійно незалежних інтерполяційних поліномів. Вдалість їх вибору впливає на стійкість квадратурної схеми та фактичну величину похибки обчислень. Існують спеціальні способи побудови наборів таких поліномів, які забезпечують максимальну симетрію квадратурної формули та дозволяють врахувати особливості конкретної фізичної задачі, наприклад, у випадку інтегрування векторних величин зберігати рівність нулю ротора чи дивергенції векторного поля у межах контрольного об'єму. Зокрема, можна будувати поліноми різних степенів на основі базисних

$$\varphi_1(x,y) = \frac{1+y}{2}, \quad \varphi_2(x,y) = \frac{1+x}{2}, \quad \varphi_3(x,y) = -\frac{x+y}{2}. \tag{4.16}$$

Таким чином, схему першого порядку можна визначити як

$$\int_{\triangle} f(x,y)dS \approx \frac{2}{3}\left[\widetilde{f}(-1,-1) + \widetilde{f}(1,-1) + \widetilde{f}(-1,1)\right], \tag{4.17}$$

а другого

$$\int_{\triangle} f(x,y)dS \approx \frac{2}{3}\left[\widetilde{f}(-1/3,-1) + \widetilde{f}(1,-1) + \widetilde{f}(-1,-1/3)\right], \tag{4.18}$$

де $\widetilde{f}(\widetilde{x},\widetilde{y}) \equiv V_{\text{tri}} f(x(\widetilde{x},\widetilde{y}), y(\widetilde{x},\widetilde{y}))$. Для прикладу наведемо розрахункові формули квадратур Гауса від першого до третього порядків:

$$\int_{\triangle} f(x,y)dS \approx 2\widetilde{f}(-1/3,-1/3), \tag{4.19}$$

$$\int_{\triangle} f(x,y)dS \approx \frac{2}{3}\left[\widetilde{f}(-2/3,-1/3)+\widetilde{f}(-2/3,1/3)+\widetilde{f}(1/3,-2/3)\right] \tag{4.20}$$

$$\begin{aligned}\int_{\triangle} f(x,y)dS \approx -1.125\widetilde{f}(-1/3,-1/3)+\\ +1.0416(6)\left[\widetilde{f}(-0.6,-0.6)+\widetilde{f}(-0.6,0.2)+\widetilde{f}(0.2,-0.6)\right].\end{aligned} \tag{4.21}$$

Таблиці для схем різних порядків наведено у [6].

Наостанок наведемо квадратурні формули Гауса першого—третього порядків для інтегрування по об'єму тетраедру з вершинами у точках $\boldsymbol{a}(-1,-1,-1)$, $\boldsymbol{b}(1,-1,-1)$, $\boldsymbol{c}(-1,1,-1)$ і $\boldsymbol{d}(-1,-1,1)$ (порівняйте зі схемою інтегрування по трикутнику):

$$\int f(\boldsymbol{r})d\boldsymbol{r} \approx \frac{4}{3}f(-1/2,-1/2,-1/2), \tag{4.22}$$

$$\begin{aligned}\int f(\boldsymbol{r})d\boldsymbol{r} \approx \frac{1}{3}\left[f(-0.724,-0.724,-0.724)+f(0.171,-0.724,-0.724)+\right.\\ \left.+f(-0.724,0.171,-0.724)+f(-0.724,-0.724,0.171)\right],\end{aligned} \tag{4.23}$$

$$\begin{aligned}\int f(\boldsymbol{r})d\boldsymbol{r} \approx -1.06(6)f(-0.5,-0.5,-0.5)+0.6\left[f(-2/3,-2/3,-2/3)+\right.\\ \left.f(-2/3,-2/3,0)+f(-2/3,0,-2/3)+f(0,-2/3,-2/3)\right].\end{aligned} \tag{4.24}$$

Таблиці для схем вищих порядків та елементів іншої форми наведено у [6].

4.3.3 Реалізація МКО для рівняння неперервності

Почнемо зі скалярної задачі, у якій можна записати рівняння неперервності (4.2) для функції $u(\boldsymbol{r},t)$ із заданими межовими умовами:

$$\begin{cases}\partial_t u+\nabla\cdot\boldsymbol{j}=f(\boldsymbol{r},t), & \boldsymbol{r}\in\Omega,\\ u(\boldsymbol{r},0)=u_0(\boldsymbol{r}),\end{cases} \tag{4.25}$$

де $\boldsymbol{j}=\boldsymbol{v}(\boldsymbol{r},t)u$ і $f(\boldsymbol{r},t)$ є потоком і густиною джерел, відповідно. Розіб'ємо область Ω на N контрольних об'ємів Ω_i, $i=\overline{1,N}$ і перейдемо до інтегральної форми (4.25). Усереднення (4.26) на області Ω_i об'ємом V_i має вигляд

$$\begin{cases}\dfrac{1}{V_i}\displaystyle\int_{\Omega_i}\partial_t u d\boldsymbol{r}+\frac{1}{V_i}\int_{\Omega_i}\nabla\cdot(\boldsymbol{v}u)d\boldsymbol{r}=\frac{1}{V_i}\int_{\Omega_i}f(\boldsymbol{r},t)d\boldsymbol{r},\\ \dfrac{1}{V_i}\displaystyle\int_{\Omega_i}u(\boldsymbol{r},0)d\boldsymbol{r}=\frac{1}{V_i}\int_{\Omega_i}u_0(\boldsymbol{r})d\boldsymbol{r}.\end{cases} \tag{4.26}$$

Інтегрування часової динаміки може відбуватись стандартними різницевими схемами. Так, для схеми Ейлера (правостороння різниця по часу) можна записати

$$\frac{1}{V_i}\int_{\Omega_0}\partial_t u d\boldsymbol{r}\approx\frac{1}{V_i\Delta t}\int_{\Omega_i}(u^{n+1}-u^n)d\boldsymbol{r}\approx\frac{\overline{u}^{i,n+1}-\overline{u}^{i,n}}{\Delta t}, \tag{4.27}$$

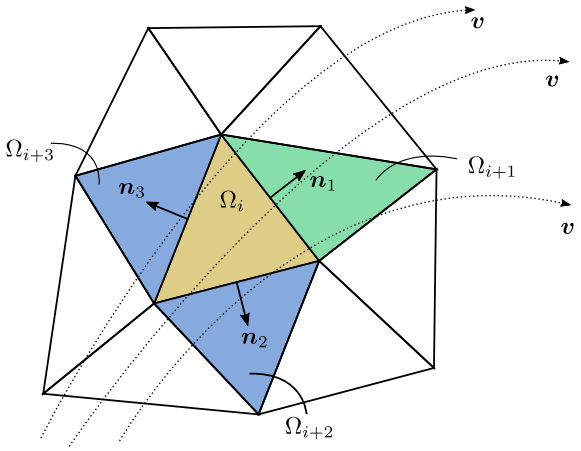


Рис. 4.6: Схема обчислення потоку крізь сторони трикутного контрольного об'єму Ω_i із гранями, пронумерованими від 1 до 3. Обчислення середнього значення потоку крізь сторону трикутника Ω_i з нормаллю $\boldsymbol{n}_1$ (до Ω_{i+1}, позначеного зеленим) по формулі (4.29) виконується для $\alpha(1) = i$, оскільки потік $\boldsymbol{v}$ витікає назовні Ω_i. Для двох інших граней із контрольними об'ємами, позначеними як Ω_{i+2} та Ω_{i+3} (сині), потік напрямлено всередину Ω_i, тому $\alpha(2) = i + 2$ та $\alpha(3) = i + 3$, відповідно.

де Δt — крок інтегрування по часу, а верхній індекс n позначає значення функції на n-му шарі. Тут і далі усереднення по контрольним об'ємам чи відповідним по змісту областям будемо позначати рискою зверху. Праву частину (4.26) можна обчислити за відповідними квадратурними формулами:

$$\frac{1}{V_i}\int_{\Omega_i} f(\boldsymbol{r},t)d\boldsymbol{r} \approx \overline{f}^i. \tag{4.28}$$

Варто підкреслити, що значення відомих та невідомих величин обраховуються всередині заданого контрольного об'єму та асоціюються з ним як набір чисел, в той час, як для сітки, якою дискретизовано простір розв'язання задачі це може означати як асоціацію з коміркою сітки, так і з її вузлом.

Для опрацювання доданку в (4.26) із дивергенцією під інтегралом, скористаємось формулою Остроградського—Гауса для перетворення інтегралу по об'єму на інтеграл по поверхні, та теоремою про середнє значення. Він набуде вигляду

$$\frac{1}{V_i}\int_{\Omega_i} \nabla\cdot(\boldsymbol{v}u)d\boldsymbol{r} \approx F^{i,n}, \quad F^{i,n} = \frac{1}{V_i}\sum_j \overline{u}^{\alpha(j),n}\int_{\partial\Omega_i^j}(\boldsymbol{v}\cdot\boldsymbol{n}_j)dS. \tag{4.29}$$

Тут підсумовування виконується по гранях $\partial\Omega_i^j$ контрольного об'єму Ω_i із нормалями $\boldsymbol{n}_j$[1]. Індекс $\alpha(j)$[2] рівний i якщо $\boldsymbol{v}\cdot\boldsymbol{n} > 0$ (потік напрямлено назовні), або індексу сусіднього контрольного об'єму, який має спільну грань із i-м,

[1]Індекс i для нормалі опущено для спрощення виразів.

[2]Строго кажучи, функція $\alpha(j) \equiv \alpha_i(j)$, оскільки вона своя для кожного Ω_i, але для спрощення записів нижній індекс i буде опущено.

до якої обчислюється нормаль, якщо $\boldsymbol{v} \cdot \boldsymbol{n} < 0$ (потік напрямлено всередину), див. рис. 4.6. Зауважимо, що такий вибір обчислення «у напрямку потоку» («upwind» or «upstream» scheme) визначає стійкість схеми. Обчислення інтегралу по поверхні може виконуватись за відомими квадратурними формулами для простору менших розмірностей за допомогою відповідного проектування. Так грані тетраедрів можна проектувати на площину і застосовувати для них відповідні квадратурні формули Гауса для трикутників (див. також [7]).

Примітка 17. Теорема про середнє значення

Для монотонних функцій $f(\boldsymbol{r})$ і $g(\boldsymbol{r})$ справедлива рівність

$$\int f(\boldsymbol{r})g(\boldsymbol{r})d\boldsymbol{r} = \overline{f} \int g(\boldsymbol{r})d\boldsymbol{r}, \tag{4.30}$$

де під $\overline{f}$ можна розуміти усереднене значення $f(\boldsymbol{r})$ із вагою $g(\boldsymbol{r})$.

Таким чином, комбінуючи (4.27), (4.28) і (4.29), початкова задача (4.25) зводиться до

$$\begin{cases} \overline{u}^{i,n+1} = \overline{u}^{i,n} + \Delta t \left(\overline{f}^{i,n} - F^{i,n} \right), \quad i = \overline{1,N},\ n = \overline{1,M}, \\ \overline{u}^{i,0} = \overline{u}^{i}_{0}. \end{cases} \tag{4.31}$$

що чисельно розв'язується очевидним чином.

Вищеописане означення $\overline{u}^{\alpha(j),n}$ на центрі вузла легко імплементується у коді, але формально дає перший порядок точності і має низьку точність для практичних застосувань. Очевидно, більш точною буде інтерпретація середнього значення шуканої функції u як обчисленої безпосередньо на грані $\partial\Omega_i^j$, наприклад, як середнє арифметичне між значеннями у контрольних об'ємах за напрямком потоку, наприклад, $\overline{u}^{\alpha(j),n} = 0.5(u^{i,n} + u^{j,n})$ для $\boldsymbol{v} \cdot \boldsymbol{n} > 0$, або ж лінійною інтерполяцією у напрямку, що з'єднує центри сусідніх контрольних об'ємів. Це відповідає центральній різницевій схемі другого порядку точності, але, так само як і у методі скінченних різниць, часто призводить до появи фіктивних осциляцій. Для боротьби з ними використовуються спеціальні методи зі «зваженою» протипоточністю, у яких частково залучається інформація з інших контрольних об'ємів, що «розповсюджується» проти потоку $\boldsymbol{v}$, або методи зі штучною дисипацією на контрольних об'ємах. Так чи інакше всі вони залучають додаткові точки, що ускладнює як аналітичне представлення розв'язання, так і код програми. Серед методів зі «зваженою» протипоточністю варто згадати схему QUICK Леонарда [8].

4.3.4 Різницеве МКО-рівняння теплопровідності

Розглянемо стаціонарну задачу, подібну до проаналізованої у розділі 4.2 не обмежуючись одновимірним випадком. Тоді рівняння балансу потоку речовини чи тепла матиме вигляд

$$\nabla \cdot \boldsymbol{W} = f(\boldsymbol{r}), \quad \boldsymbol{W} = -k(\boldsymbol{r})\nabla u. \tag{4.32}$$

Як і у випадку рівняння неперервності, проінтегруємо (4.32) по контрольному об'єму і перетворимо об'ємний інтеграл від дивіргенції на інтеграл по поверхні обчисливши середнє:

$$\frac{1}{V_i}\sum_j \int_{\partial\Omega_i^j} \boldsymbol{W}\cdot\boldsymbol{n}_j dS = \overline{f}^i. \tag{4.33}$$

Обчислити інтеграл потоку можна різними способами. Перший спосіб полягає у тому, щоб виразити $\nabla u = -\boldsymbol{W}/k$ і, скориставшись теоремою про середнє значення, знайти середнє значення градієнту шуканої функції у напрямку $\boldsymbol{n}_j$:

$$\boldsymbol{n}_j\cdot\nabla u \approx \overline{\nabla_j u} = -\overline{d}^{ij}\overline{\boldsymbol{W}\cdot\boldsymbol{n}_j}, \quad \overline{d}^{ij} = \frac{1}{V_i + V_{\alpha(j)}}\int\limits_{\Omega_i+\Omega_{\alpha(j)}} \frac{d\boldsymbol{r}}{k(\boldsymbol{r})}. \tag{4.34}$$

Тут $\overline{\boldsymbol{W}\cdot\boldsymbol{n}_j}$ має зміст середнього значення потоку $\boldsymbol{W}$ через j-ту грань, а інтегрування виконується по двох сусіднім контрольним об'ємам Ω_i та $\Omega_{\alpha(j)}$, які мають цю грань спільною. З іншого боку, дискретне значення градієнту можна представити як

$$\overline{\nabla_j u} \approx \frac{u^{\alpha(j)} - u^i}{h_{i,\alpha(j)}}, \tag{4.35}$$

де $h_{i,\alpha(j)}$ відстань між центрами контрольних об'ємів Ω_i та $\Omega_{\alpha(j)}$. Таким чином, дискретна форма виразу (4.33) формулюється як

$$\frac{1}{V_i}\sum_j (u^i - u^j)\frac{S_{ij}}{h_{ij}\overline{d}^{ij}} = \overline{f}^i, \quad i = \overline{1,N}, \tag{4.36}$$

де S_{ij} — площа грані між сусідніми об'ємами Ω_i та Ω_j та індекс j пробігає сусідів Ω_i. Цей підхід у одновимірному випадку розглянуто у підручнику О. О. Самарського [9]. Завдяки тому, що обчислюється середнє значення потоку через кожну з граней, він придатний як для гладких, так і розривних коефіцієнтів рівняння теплопровідності.

Інший спосіб усереднення, що споріднений методу скінченних різниць, полягає у тому, що у рівняння (4.33) підставляється вираз для потоку $\boldsymbol{W}$ (див. підручник М. Дж. Гандера і Ф. Квока [10]). Винесення похідної у напрямку $\boldsymbol{n}_j$ за теоремою про середнє значення дозволяє перетворити інтеграл по j-й грані у (4.33) як

$$\int\limits_{\partial\Omega_i^j} \boldsymbol{W}\cdot\boldsymbol{n}_j dS = -\int\limits_{\partial\Omega_i^j} k(\boldsymbol{r})\boldsymbol{n}_j\cdot\nabla u dS \approx (u^i - u^{\alpha(j)})\frac{S_{ij}\overline{k}^{ij}}{h_{i,\alpha(j)}}, \tag{4.37}$$

де $\overline{k}^{ij}$ — середнє значення $k(\boldsymbol{r})$ на грані між сусідніми Ω_i та Ω_j. Таким чином, альтернативна дискретна форма (4.33) має вигляд

$$\frac{1}{V_i}\sum_j (u^i - u^j)\frac{S_{ij}\overline{k}^{ij}}{h_{i,\alpha(j)}} = \overline{f}^i, \quad i = \overline{1,N}. \tag{4.38}$$

Цей підхід не є консервативним, тому може застосовуватись лише для досить гладких параметрів задачі. Неконсервативність методу можна побачити порівнявши (4.36) та (4.38). Вираз (4.36) отримується з оцінки значення потоку $\overline{\boldsymbol{W}\cdot\boldsymbol{n}_j}$ крізь задану грань і усереднення коефіцієнту $k(\boldsymbol{r})$, «заховане» у $\overline{d}^{ij}$, враховує його розподіл по двом сусіднім контрольним об'ємам. Оцінка ж потоку в (4.38) отримувалась з усереднення по площі грані, крізь яку проходить потік. Це еквівалентне обранню значення k на міжвузлі й не враховує його просторову варіацію в напрямку, перпендикулярному до грані.

Зазначимо, що технічну спорідненість підходу (4.38) із методом скінченних різниць можна використовувати для опрацювання областей складної форми: всередині них вводити рівномірну сітку, обчислення на якій виконуються за методом скінченних різниць, а межу апроксимувати набором контрольних об'ємів.

Звернемо увагу, що зі структури (4.36) і (4.38) видно, що висока симетрія сітки дозволяє отримати меншу кількість повністю ненульових діагоналей у матричних елементах, що є фактором пришвидшення виконання обчислень.

Глобальне виконання законів збереження зручно показати на прикладі одновимірної задачі з рівномірною сіткою із кроком h на відрізку $x \in [x_1, x_2]$. Різницеве рівняння (4.36) має вигляд

$$\frac{u^i - u^{i-1}}{h^2\overline{d}^{i-1,i}} + \frac{u^i - u^{i+1}}{h^2\overline{d}^{i,i+1}} = \overline{f}^i, \quad i = \overline{1, N}. \tag{4.39}$$

де індекс позначає номер відрізка (одновимірного контрольного об'єму), а не вузла. Підсумовування (4.39) по всім відрізкам дасть

$$\Delta W = \int_{x_1}^{x_2} f(x)dx, \tag{4.40}$$

що є інтегральною формою рівняння балансу. Таким чином, виконана дискретизація рівняння теплопровідності підтримує закон збереження потоку на всій області аналізу.

4.3.5 Межові умови для рівняння теплопровідності

Розглянемо вигляд межових умов для рівняння теплопровідності (4.32). Найбільш просто реалізуються межові умови Дириxле

$$u(\boldsymbol{r}) = g(\boldsymbol{r}), \quad \boldsymbol{r} \in \partial\Omega. \tag{4.41}$$

У такому випадку зручно обирати таку сітку, щоб її вузли (центри контрольних об'ємів) припадали на $\partial\Omega$. Тоді у системах рівнянь (4.36) замість значень u^j на межі можна використовувати значення функції $g(\boldsymbol{r})$ у відповідних координатах. Обчислення консервативною схемою МКО задачі (4.4) показано на рис. 4.7.

Межові умови типу Неймана

$$\frac{\partial u}{\partial \boldsymbol{n}} = g(\boldsymbol{r}), \quad \boldsymbol{r} \in \partial\Omega \tag{4.42}$$

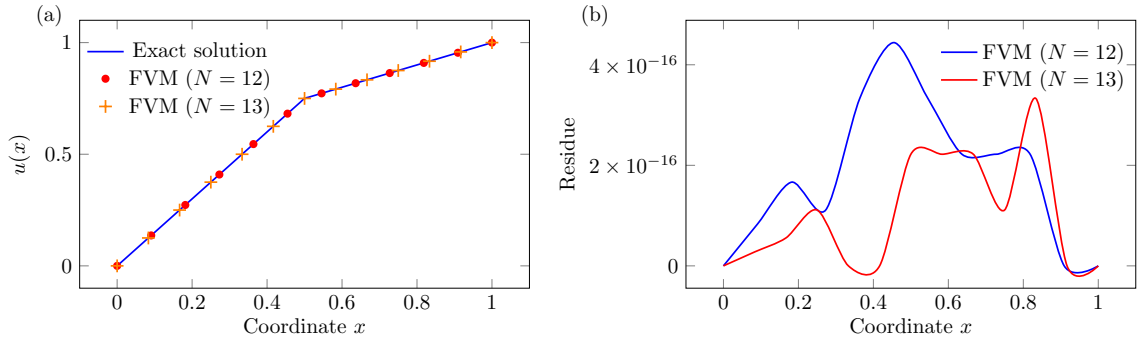


Рис. 4.7: (a) Розв'язок (4.4a) із $k_1 = 1$, $k_2 = 1.1$ та межовими умовами $\mu_1 = 0$, $\mu_2 = 1$ на відрізку $x \in [0, 1]$ консервативною реалізацією методу контрольних об'ємів з різними кроками дискретизації. (b) Абсолютне значення нев'язки $|u_{\text{numerical}}(x) - u_{\text{exact}}(x)|$ чисельного розв'язку. У порівнянні з рис. 4.1(c,d) видно, що точність обчислень находиться на межі числового шуму й мало змінюється при малій зміні розміру контрольного об'єму.

та Робіна

$$p(\boldsymbol{r})u + q(\boldsymbol{r})\frac{\partial u}{\partial \boldsymbol{n}} = g(\boldsymbol{r}), \quad \boldsymbol{r} \in \partial\Omega \tag{4.43}$$

опрацьовуються схожим чином. Для прикладу розглянемо Нейманівські (4.42). Нехай контрольний об'єм Ω_i знаходиться на межі області інтегрування та його грань з індексом $j = 1$ (нумерація граней також починатиметься з одиниці для визначеності) є межовою. Тоді інтеграл потоку по поверхні контрольного об'єму можна розписати як

$$\sum_j \int\limits_{\partial\Omega_i^j} \boldsymbol{W} \cdot \boldsymbol{n}_j dS = \int\limits_{\partial\Omega_i^1} \boldsymbol{W} \cdot \boldsymbol{n}_1 dS + \sum_{j>1} \int\limits_{\partial\Omega_i^j} \boldsymbol{W} \cdot \boldsymbol{n}_j dS. \tag{4.44}$$

Суму по $j > 1$ можна розписати за (4.37), а в інтегралі по грані $\partial\Omega_i^1$ використати межову мову (4.42). Тоді для межового контрольного об'єму з одною гранню на межі рівняння (4.36) набуде вигляду

$$\frac{1}{V_i}\sum_{j>1}(u^i - u^j)\frac{S_{ij}}{h_{i,\alpha(j)}\overline{d}^{ij}} - \frac{1}{V_i}\frac{\overline{g}^{i1}S_{i1}}{h_{i1}\overline{d}^{i1}} = \overline{f}^i, \tag{4.45}$$

де замість похідної за напрямком використано межову умову і відповідне середнє значення $\overline{g}^{i1}$ можна обчислити за відомими квадратурними формулами.

4.4 Повністю консервативні схеми

4.5 Методи стабілізації розв'язку

4.6 Задачі

№23. Знайдіть аналітично та за квадратурними формулами наступні інтеграли на відрізку. Визначіть похибку обчислень.

1. $\int\limits_0^2 \sin^2 \frac{x}{10} dx$
2. $\int\limits_{-1}^0 \frac{x}{x+2} dx$
3. $\int\limits_0^3 \frac{1}{x^2+2} dx$
4. $\int\limits_1^{1.5} x \cos x dx$

№24. Побудуйте графік і опишіть властивості базисних поліномів (4.16) відносно до референсного трикутника, зображеного на рис. 4.5(b).

№25. Використовуючи базисні поліноми (4.16), отримайте квадратурні формули (4.17) і (4.18). Як вони зміняться, якщо обрати інші базисні поліноми?

№26. Знайдіть аналітично та за квадратурними формулами наступні інтеграли від наступних функцій на референсному трикутнику. Визначить похибку обчислень.

1. $f(x, y) = x(1 - y^2)$
2. $f(x, y) = \sin \frac{x}{2} \cos \frac{y}{3}$
3. $f(x, y) = (y - 1) \sin(x + 3)$
4. $f(x, y) = \sin^2(x + y)$

№27. Отримайте перетворення тетраедру з довільною орієнтацією у просторі до такого, на якому застосовні формули (4.22)—(4.24).

№28. Виведіть схему МКО-дискретизації межових умов Робіна (4.43).

№29. Реалізуйте МКО-схеми для одновимірних еквівалентів рівнянь (4.36) і (4.38) для задачі, що була розглянута у розділі 4.2. Чисельно покажіть еквівалентність (4.38) до результату інтегрування методом скінченних різниць.

4.7 Take home message

1. Дискретна модель фізичної системи не еквівалентна континуальній.
2. Консервативні схеми обчислень як спосіб автоматичного забезпечення законів збереження.
3. Метод контрольних об'ємів для розв'язання диференціальних рівнянь у частинних похідних на довільних областях простору.
4. Обчислення складних геометрій за допомогою нерегулярних сіток на межі областей методами скінченних різниць і контрольних об'ємів.

Посилання

1. *Попов Ю. П.*, *Самарский А. А.* Полностью консервативные разностные схемы // Ж. вычисл. матем. и матем. физ. — 1969. — Т. 9, вип. 4, № 4. — С. 953—958.
2. Gmsh repository. — URL: https://gitlab.onelab.info/gmsh/gmsh/-/tree/8a57645881f02deb16993380ca57bb2b8a026e11 ; (приклад взято з посібника по Gmsh).
3. *Exl L.*, *Schrefl T.* Non-uniform FFT for the finite element computation of the micromagnetic scalar potential // Journal of Computational Physics. — 2014. — Серп. — Т. 270. — С. 490—505. — DOI: 10.1016/j.jcp.2014.04.013. — URL: https://doi.org/10.1016/j.jcp.2014.04.013.
4. *Börm S.* Efficient Numerical Methods for Non-local Operators. — European Mathematical Society Publishing House, 12.2010. — (Ems Tracts in Mathematics). — ISBN 3037190914,9783037190913. — DOI: 10.4171/091. — URL: https://www.ems-ph.org/books/book.php?proj_nr=125&srch=series%7CETM.
5. *Weisstein E. W.* Newton-Cotes Formulas. — MathWorld–A Wolfram Web Resource. — URL: https://mathworld.wolfram.com/Newton-CotesFormulas.html.
7. *Reeger J. A.*, *Fornberg B.*, *Watts M. L.* Numerical quadrature over smooth, closed surfaces // Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. — 2016. — Жовт. — Т. 472, вип. 2194, № 2194. — С. 20160401. — DOI: 10.1098/rspa.2016.0401. — URL: https://doi.org/10.1098/rspa.2016.0401.
8. *Leonard B.* A stable and accurate convective modelling procedure based on quadratic upstream interpolation // Computer Methods in Applied Mechanics and Engineering. — 1979. — Черв. — Т. 19, вип. 1, № 1. — С. 59—98. — DOI: 10.1016/0045-7825(79)90034-3. — URL: https://doi.org/10.1016/0045-7825(79)90034-3.

Література до розділу

6. *Šolín P.*, *Segeth K.*, *Doležel I.* Higher-Order Finite Element Methods. — Boca Raton London Newy York Washington, D.C. : Chapman & Hall/CRC, 2004. — ISBN 1-58488-438-X.
9. *Самарский А. А.* Введение в теорию разностных схем. — М. : Наука, 1971.
10. *Gander M. J.*, *Kwok F.* Numerical Analysis of Partial Differential Equations Using Maple and MATLAB. — Society for Industrial, Applied Mathematics, 08.2018. — DOI: 10.1137/1.9781611975314. — URL: https://doi.org/10.1137/1.9781611975314.
11. *Самарский А. А.*, *Гулин А. В.* Численные методы: Учебное пособие для вузов. — М. : Наука. Гл. ред. физ-мат. лит., 1989. — С. 432. — ISBN 5-02-013996-3.

13. *Флетчер К.* Вычислительные методы в динамике жидкостей: в 2-х томах. Т. 2. — М. : Мир, 1991. — С. 552. — ISBN 5-03-001881-4.

14. *Флетчер К.* Вычислительные методы в динамике жидкостей: в 2-х томах. Т. 1. — М. : Мир, 1991. — С. 504. — ISBN 5-03-001881-6.

Розділ 5

Метод скінченних елементів

5.1 Ідея методу

Метод скінченних елементів (МСЕ, Finite elemeths method, FEM) є одним з математично найрозвинутіших на сьогоднішній день методів розв'язання задач у частинних похідних, який можна застосувати до будь-якої геометрії простору. Його основи були незалежно закладені Вальтером Рітцем і Борисом Галеркіним на початку XX сторіччя [1]. У той час, як методи скінченних різниць та контрольних об'ємів мають справу з різними способами побудови різницевих аналогів диференціальних операторів, у МСЕ ставиться задача пошуку наближеного розв'язку задачі, який апроксимується набором відомих (базисних) функцій, вагові коефіцієнти до яких необхідно знайти. Чим ближче за своїми властивостями базисні функції до точного розв'язку (наприклад, опуклість і кількість нулів на відрізку), тим краще наближення отримується в результаті розв'язання. Як правило, у практичних задачах підібрати базисні функції на всьому просторі аналізу досить важко. Тому використовується дискретизація простору аналогічна тому, як це виконується для МКО і підбираються такі базисні функції, що вони відрізняються від нуля лише у певному околі вузлів сітки. Якщо сітка є досить дрібною, то в якості базисних функцій можна обирати лінійні функції й це все одно буде давати прийнятну точність обрахунків.

5.2 Крайова задача як задача мінімізації

Розглянемо наступну межову задачу всередині області Ω:

$$\mathcal{L}u = f(\boldsymbol{r}), \quad u|_{\partial\Omega} = 0, \quad \boldsymbol{r} \in \Omega, \tag{5.1}$$

де $\mathcal{L}$ — лінійний диференціальний оператор, функції $f(\boldsymbol{r})$ задано, а $u \equiv u(\boldsymbol{r})$ необхідно знайти. Введемо визначення внутрішнього добутку функцій комплекснозначних u і v:

$$\langle u, v \rangle = \int\limits_{\Omega} uv^* \mathrm{d}\boldsymbol{r}, \tag{5.2}$$

де символ * означає комплексне спряження. Звернемо увагу, що внутрішній добуток є лінійним оператором, тобто для довільних сталих $a, b \in \mathbb{C}$ виконується

$\langle au_1(\boldsymbol{r}) + bu_2(\boldsymbol{r}), v\rangle = a\langle u_1(\boldsymbol{r}), v\rangle + b\langle u_2(\boldsymbol{r}), v\rangle$. Будемо вважати, що виконуються умови самоспряженості

$$\langle \mathcal{L}u, u\rangle = \langle u, \mathcal{L}u\rangle \tag{5.3}$$

та позитивної визначеності оператора

$$\langle \mathcal{L}u, u\rangle \geq 0 \tag{5.4}$$

де рівність досягається лише у випадку $u \equiv 0$. Покажемо, що якщо конкретна функція u_0 є розв'язком (5.1), то наступний функціонал при $u = u_0$ сягає екстремуму:

$$F[u] = \frac{1}{2}\langle \mathcal{L}u, u\rangle - \frac{1}{2}\left(\langle u, f\rangle + \langle f, u\rangle\right). \tag{5.5}$$

Для цього обчислимо його варіацію:

$$\begin{aligned}\delta F = F[u+\delta u] - F[u] = \frac{1}{2}\left[\langle \mathcal{L}(u+\delta u), u+\delta u\rangle - \mathcal{L}u, u\rangle\right] - \\ - \frac{1}{2}\left[\left(\langle u+\delta u, f\rangle + \langle f, u+\delta u\rangle\right) - \left(\langle u, f\rangle + \langle f, u\rangle\right)\right] = \\ = \frac{1}{2}\left(\langle \mathcal{L}u - f, \delta u\rangle + \langle \delta u, \mathcal{L}u - f\rangle\right),\end{aligned} \tag{5.6}$$

де було використано лінійні властивості внутрішнього добутку та $\mathcal{L}$. Якщо скористатись властивістю $\langle u, v\rangle = (\langle v, u\rangle)^*$, то варіація F рівна

$$\delta F = \mathrm{Re}\langle \mathcal{L}u - f, \delta u\rangle. \tag{5.7}$$

Ця рівність виконується для будь-якої функції δu із $\nabla \delta u|_{\partial\Omega} = 0$ лише тоді, коли $u = u_0$. Таким чином, розв'язання межової задачі (5.1) можна звести до задачі мінімізації функціоналу (5.5).

Зауважимо, що усі наведені міркування справедливі й для векторних задач із невідомим вектор-стовпчиком $\boldsymbol{u}$. У цьому випадку в означенні внутрішнього добутка під інтегралом необхідно записувати скалярний добуток $\boldsymbol{u}$ і $\boldsymbol{v}^*$. Вимога позитивної визначеності оператора (5.4) не є строго необхідною, оскільки визначає тип екстремуму F (мінімум або максимум), що не обов'язково є принциповим для конкретної фізичної задачі.

5.2.1 Метод Рітца

Щоб знайти наближений розв'язок крайової задачі (5.1) (обмежимось для прикладу дійснозначними величинами), скористаємось методом Рітца для функціоналу $F[\widetilde{u}]$, де так звана *пробна функція* має вигляд

$$\widetilde{u}(\boldsymbol{r}) = \sum_{i=1}^{N} c_i \psi_i(\boldsymbol{r}) \equiv \boldsymbol{c}\cdot\boldsymbol{\psi} \equiv \boldsymbol{c}^T\boldsymbol{\psi}. \tag{5.8}$$

Тут вектор $\boldsymbol{c} = \{c_i\}$, $i = \overline{1, N}$ містить невідомі сталі, а вектор-функція $\boldsymbol{\psi} = \{\psi_i(\boldsymbol{r})\}$, $\psi_i(\boldsymbol{r}) \in \Omega$ є набором відомих *базисних функцій*, які визначені на всій

області задачі, а символ ^{T} позначає транспонування. Тоді функціонал F набуде векторного вигляду

$$F = \frac{1}{2}\boldsymbol{c}^T \left[\int\limits_\Omega \boldsymbol{\psi}\left(\mathcal{L}\boldsymbol{\psi}^T\right)\mathrm{d}\boldsymbol{r}\right]\boldsymbol{c} - \boldsymbol{c}^T \int\limits_\Omega \boldsymbol{\psi} f \mathrm{d}\boldsymbol{r}. \tag{5.9}$$

Звернемо увагу, що у першому інтегралі фактично наявний зовнішній добуток векторів, $\boldsymbol{u}\otimes\boldsymbol{v} \equiv \{u_i v_j\}_{i,j=\overline{1,N}}$, який утворює матрицю з двох векторів. Умові екстремуму відповідає рівність нулю похідних за c_i:

$$\frac{\partial F}{\partial c_i} = \frac{1}{2}\sum_{j=1}^{N} c_j \int\limits_\Omega \left(\psi_i \mathcal{L}\psi_j + \psi_j \mathcal{L}\psi_i\right)\mathrm{d}\boldsymbol{r} - \int\limits_\Omega f\psi_i \mathrm{d}\boldsymbol{r}, \quad i = \overline{1,N}. \tag{5.10}$$

Оскільки під обома інтегралами знаходяться відомі функції, їх можна завчасно обчислити аналітично або за допомогою квадратурних формул. Тоді система (5.10) формулюється у матричному вигляді

$$A\boldsymbol{c} = \boldsymbol{b} \tag{5.11а}$$

із елементами матриці $A = \{A_{ij}\}$ та вектора $\boldsymbol{b} = \{b_i\}$

$$A_{ij} = \frac{1}{2}\int\limits_\Omega \left(\psi_i \mathcal{L}\psi_j + \psi_j \mathcal{L}\psi_i\right)\mathrm{d}\boldsymbol{r}, \quad b_i = \int\limits_\Omega f\psi_i \mathrm{d}\boldsymbol{r}, \quad i,j = \overline{1,N}. \tag{5.11б}$$

Звернемо увагу, що тривіальні межові умови Діріхле задачі (5.1) можуть бути враховані двояко. Якщо базисні функції $\psi_i(\boldsymbol{r})$ їх задовольняють, то вони будуть виконані автоматично. Інакше їх треба явно врахувати у (5.11), що зменшить розмірність цієї матричної задачі.

Розглянемо приклад наступний приклад [2] для ілюстрації застосування методу Рітца. Нехай задано рівняння

$$\begin{cases} u''(x) = 1 + x, \quad x \in [0,1], \\ u(0) = 0, \quad u(1) = 1. \end{cases} \tag{5.12}$$

Його фізичною інтерпретацією може бути розподіл електричного потенціалу u між обкладинками конденсатору з різницею напруг в 1 В і розподілом заряду в діелектрику як $-(1+x)$ Кл/м^3. Його аналітичний розв'язок

$$u(x) = \frac{x}{3} + \frac{x^2}{2} + \frac{x^3}{6}. \tag{5.13}$$

Для розв'язання методом Рітца оберемо поліноми як базисні функції (див. (5.8)):

$$\psi_i(x) = x^{i-1}, \quad i = \overline{1,4}. \tag{5.14}$$

З межових умов випливає, що $\widetilde{u}(0) = c_1 = 0$ та $\widetilde{u}(1) = c_2 + c_3 + c_4 = 1$, або ж $c_4 = 1 - c_2 - c_3$. Тоді (5.9) можна проінтегрувати і записати як

$$F[\widetilde{u}] = \frac{27}{30} - \frac{5}{12}c_2 + \frac{2}{5}c_2^2 - \frac{1}{6}c_3^2 + \frac{1}{15}c_3^2 + \frac{3}{10}c_2 c_3. \tag{5.15}$$

Мінімум цього виразу досягається при $c_2 = 1/3$ і $c_3 = 1/2$, що дає

$$\widetilde{u} = \frac{x}{3} + \frac{x^2}{2} + \frac{x^3}{6}. \tag{5.16}$$

Завдяки вдалому підбору базисних функцій цей результат співпав з точним розв'язком (5.13). За меншої кількості базисних поліномів (або інших базисних функцій) розв'язок був би лише наближеним.

5.2.2 Метод Галеркіна

Якщо пробна функція (5.8) є наближеним розв'язком крайової задачі (5.1), то оцінка нев'язки у заданій точці $\boldsymbol{r}$ має вигляд

$$\rho = \mathcal{L}\widetilde{u} - f \neq 0 \tag{5.17}$$

де конкретні властивості оператора $\mathcal{L}$ (додатня визначеність чи самоспряженість) не грають ролі. Звернемо увагу, що таким самим чином визначалася нев'язка у задачі з розривними коефіцієнтами у розділі про метод контрольних об'ємів. Якість наближеного розв'язку $\widetilde{u}$ можна оцінити за інтегральним значенням нев'язки. Для цього означимо зважену нев'язку

$$R_i = \int\limits_{\Omega} w_i(\boldsymbol{r})\rho(\boldsymbol{r})\mathrm{d}\boldsymbol{r}, \tag{5.18}$$

де $w_i(\boldsymbol{r})$ — певна вагова функція і будемо вимагати її рівність нулю, $R_i = 0$.

Можна обирати різні вагові функції, що дасть різне формулювання задачі. Так, якщо у якості вагових функцій обрати базисні, $w_i \equiv \psi_i$, $i = \overline{1, N}$, матричне рівняння на коефіцієнти $\boldsymbol{c}$ буде схожим за структурою на (5.11). Зокрема, якщо оператор $\mathcal{L}$ є самоспряженим, то задачі Рітца і Галеркіна співпадуть. Як правило, цей підхід дає найбільш акуратне наближення і тому є одним з найпопулярніших.

Можна вимагати мінімізації квадрату нев'язки

$$I = \frac{1}{2}\int\limits_{\Omega} \rho^2\mathrm{d}\boldsymbol{r}, \tag{5.19}$$

що еквівалентно методу найменших квадратів або вибору $\mathcal{L}\psi_i$ у якості вагових функцій. Якщо у якості вагових функцій обрати дельта-функцію Дірака $\delta(\boldsymbol{r} - \boldsymbol{r}_i)$, де точки $\boldsymbol{r}_i$ називаються точками колокації. У такому випадку крайова задача буде задовольнятись точно у точках колокації,

$$R_i = \left(\boldsymbol{c}^T\mathcal{L}\boldsymbol{\psi} - f\right)|_{\boldsymbol{r}=\boldsymbol{r}_i} = 0. \tag{5.20}$$

При цьому зручно обирати кількість точок колокації рівною кількості невідомих коефіцієнтів.

5.3 Сильне і слабке формулювання крайової задачі

Знову для прикладу розглянемо задачу Пуасона із $\mathcal{L} \equiv -\Delta$ у (5.1). Як правило, від розв'язку фізичної задачі очікується, що він має скінченну енергію, тобто для скалярної функції u виконується співвідношення виду

$$\int\limits_{\mathbb{R}^3} |u(\boldsymbol{r})|^2 \mathrm{d}\boldsymbol{r} < \infty. \tag{5.21}$$

Звернемо увагу, що для задачі (5.1) окрім властивості інтегровності, необхідно також існування принаймні другої похідної u.

Коментар 16. Простір функцій Соболєва

Кажуть, що функції, для яких інтеграл від модуля степені p є збіжним, належать простору Лебега (інтегровні за Лебегом), що записується як $u \in L_p$. Зокрема, у випадку $p = 2$ (5.21), це функції з Гільбертового простору. Якщо від u можна обчислити k похідних, то кажуть, що вона належить простору Соболєва[a], $u \in H_p^k$. Для більшості фізичних задач виконується $p = 2$ (наприклад, коли невідомі функції виражають напруженості полів) і необхідність обчислення оператору Лапласа вимагає мінімального значення $k = 2$.

Для спрощення часто використовується позначення $H^1 \equiv H_2^1$ для функцій будь-якої розмірності. Також вводять простори Соболєва $\boldsymbol{H}(\mathrm{curl})$ для векторних функцій $\boldsymbol{u}$, які разом зі своїм ротором $\nabla \times \boldsymbol{u}$ належать H^1, та $\boldsymbol{H}(\mathrm{div})$ для векторних функцій $\boldsymbol{u}$, які разом зі своєю дивіргенцією $\nabla \cdot \boldsymbol{u}$ належать H^1. Для двох останніх ключовою є квадратична інтегровність відповідних диференціальних операторів.

[a] Взагалі, вимагається існування похідних у «слабкому сенсі» — узагальнених похідних, розгляд теорії яких виходить за межі даного курсу.

Домножимо рівняння Лапласа на вагову функцію $v(\boldsymbol{r})$ (її називають *тестовою*) і проінтегруємо по частинам:

$$\int\limits_{\Omega} \nabla u \cdot \nabla v \mathrm{d}\boldsymbol{r} - \int\limits_{\partial\Omega} v \nabla u \cdot \boldsymbol{n} \mathrm{d}S = \int\limits_{\Omega} f v \mathrm{d}\boldsymbol{r}. \tag{5.22}$$

Якщо v — довільна, то задачі (5.1) і (5.22) еквівалентні. Але з технічної точки зору, остання не вимагає подвійної диференціовності u, $u \in H_2^1$. Тому задачу (5.22) називають *слабким формулюванням (слабкою формою)*, а її розв'язки — *слабкими*, на відміну від так званого *сильного формулювання* (5.1). При цьому, умови на тестову функцію одразу слабші, аніж на u: $v \in H_2^1$. Слабке формулювання задачі є ключовим для більшості методів пошуку наближених розв'язків фізичних задач, оскільки послаблені вимоги на диференційовність невідомої функції дозволяють спростити відповідні алгоритми. Звісно, при цьому також

розглядається конкретний набір тестових функцій замість вимоги на їх довільність, зокрема зручно обирати однаковий набір тестових та базисних функцій. У позначеннях внутрішнього добутка вираз (5.22) можна записати коротше, як

$$\langle \nabla u, \nabla v \rangle - \langle v, \nabla u \cdot \boldsymbol{n} \rangle_{\partial\Omega} = \langle f, v \rangle, \tag{5.23}$$

де під нижнім індексом $\partial\Omega$ розуміється обчислення по межі області.

Коментар 17. Методи скінченних елементів Рітца і Галеркіна

Вирази (5.5) і (5.22) математично можна представити як білінійні форми, які діють на функції u і v (при чому для функціоналу Рітца $u \equiv v$). Якщо така білінійна форма є симетричною, то кажуть про реалізацію методу скінченних елементів Рітца, інакше — про метод скінченних елементів Галеркіна або Бубнова–Галеркіна.

5.3.1 Межові умови Неймана

У випадку Нейманівських межових умов

$$\nabla u \cdot \boldsymbol{n}|_{\partial\Omega} = u_1(\boldsymbol{r}) \tag{5.24}$$

вираз (5.22) спрощується очевидним чином:

$$\int\limits_{\Omega} \nabla u \cdot \nabla v \mathrm{d}\boldsymbol{r} - \int\limits_{\partial\Omega} v u_1 \mathrm{d}S = \int\limits_{\Omega} f v \mathrm{d}\boldsymbol{r}. \tag{5.25}$$

Завдяки такому «автоматичному» врахуванню межових умов (5.24) їх називають натуральними.

5.3.2 Межові умови Діріхле

У порівнянні з Нейманівськими межовими умовами, врахування умов Діріхле

$$u(\boldsymbol{r})|_{\partial\Omega} = u_1(\boldsymbol{r}) \tag{5.26}$$

є складнішим й у МСЕ їх називають суттєвими межовими умовами. У слабкій формі (5.22) присутня лише похідна невідомої функції на межі, яка теж є невідомою. Якщо $u_1(\boldsymbol{r}) \equiv 0$, то можна обрати тестові функції $v(\boldsymbol{r})$ такі, що вони теж будуть нульовими на межі. Якщо це не так, задачу можна формально звести до тривіальних межових умов заміною $u' = u - u_1$, що вимагатиме довизначення $u_1(\boldsymbol{r})$ на всій області Ω. Тоді слабка форма набуде вигляду

$$\int\limits_{\Omega} \nabla u' \cdot \nabla v \mathrm{d}\boldsymbol{r} + \int\limits_{\partial\Omega} v \nabla u_1 \cdot \boldsymbol{n} \mathrm{d}S = \int\limits_{\Omega} f v \mathrm{d}\boldsymbol{r}, \tag{5.27}$$

де позбутись інтегралу по межі можна відповідним вибором тестових функцій.

5.4 Побудова методу скінченних елементів

За формальним визначенням, метод скінченних елементів полягає у побудові спеціальних скінченновимірних функціональних підпросторів V_h типу просторів Соболєва, які називають просторами скінченних елементів. Для цього вимагається:

1. Розбиття області Ω на набір підобластей $\{\Omega_i\}$ об'єм яких більше нуля й які лише дотикаються (але не перетинаються). . По об'єму та межі кожного скінченного елементу можна взяти визначений інтеграл.

2. На кожній підобласті Ω_i задано функціональний простір $\mathcal{P}_i$ (який, як правило, містить неперервні, кусково-диференціойвні поліноми).

3. У цьому функціональному просторі задано «канонічний» набір базисних функцій $\mathcal{N}_i$, які ненульові лише на «мінімальній» ділянці фізичного простору Ω.

Тоді трійка $\{\Omega_i, \mathcal{P}_i, \mathcal{N}_i\}$, $i = \overline{1, N}$ називається скінченним елементом. Часто скінченним елементом називають безпосередньо область Ω_i вважаючи, що набір базисних функцій вже визначено специфікою задачі й їх можна не обговорювати окремо (порівняйте це із введенням контрольного об'єму). У третьому пункті під канонічністю мається на увазі ортогональність набору базисних функцій: якщо внутрішній добуток кожної пари різних базисних функцій рівний нулю, це спрощує структуру вихідних матриць, до яких зводитиметься обчислення. Мінімальність означає, що носій кожної з них (та частина області визначення функції, на якій вона ненульова) задано якомога меншим.

Нехай задача (5.1) задана на відрізку $x \in [x_0, x_N]$, то цей відрізок можна розбити на набір послідовно заданих вузлів $x_1, \dots x_{N-1}$, при чому рівномірність їх впорядкування не обов'язкова. Задамо кусково-лінійну базисну функцію виду

$$\psi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \le x \le x_i, \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i \le x \le x_{i+1}, \\ 0, & x \notin [x_{i-1}, x_{i+1}], \end{cases} \qquad i = \overline{1, N-1} \tag{5.28}$$

Вона має «трикутну» форму починаючи зростання від нуля на $(i-1)$-му вузлі, сягаючи одиниці на i-му вузлі, спадаючи до нуля на $(i+1)$-му вузлі та будучи нульовою в інших точках відрізка. Лінійна комбінація з таких функцій рівна нулю на межі відрізка, що дозволяє автоматично врахувати тривіальні межові умови Діріхле. Якщо вузлів на відрізку задано досить багато, то з $\psi_i(x)$ можна зібрати досить точну апроксимацію точного розв'язку рівняння. Уявити побудову апроксимації можна розглянувши відрізок $\Omega_i = [x_i, x_{i+1}]$ із значеннями функції на вузлах u^i та u^{i+1}. Тоді інтерполяція $\overline{u}(x)$ на відрізку буде складатись з відповідних зважених базисних функцій

$$u_h^i(x) = u^i \psi_i(x) + u^{i+1} \psi_{i+1}(x) \equiv u^i \frac{x_{i+1} - x}{x_{i+1} - x_i} + u^{i+1} \frac{x - x_i}{x_{i+1} - x_i}. \tag{5.29}$$

Підкреслимо, що в МСЕ обчислення середніх значень функцій відбувається на Ω_i як і у випадку МКО, але значення функцій зберігаються на вузлах сітки.

Ключовою властивістю, яку варто забезпечувати при виборі базисних функцій є згадана вище ортогональність. Так, для набору (5.28) виконуються наступні співвідношення:

$$\langle \psi_i, \psi_i \rangle = \frac{x_{i+1} - x_{i-1}}{3}, \quad \langle \psi_i, \psi_{i\pm1} \rangle = \pm\frac{x_{i\pm1} - x}{6}, \quad \langle \psi_i, \psi_{i\pm2,3,\dots} \rangle = 0. \tag{5.30}$$

Іншими словами, пари зазначених базисних функцій мають ненульовий внутрішній добуток лише якщо вони є сусідами. Аналогічним чином можна будувати набори базисних функцій у двох і трьох вимірах. Так, якщо площина розбита на трикутники, базисні функції повинні бути рівні одиниці на вузлах і лінійно спадати до нуля на протилежних до вузлів сторонах.

У якості базисних функцій також зручно обирати функції, які дозволяють інтерполяцію всередині області Ω_i, як це зроблено у (5.29) (підкреслимо, що у строго математичному сенсі ця область стане скінченним елементом лише після того, як на ній буде їх означено). Їх називають функціями форми (shape functions) Наприклад, якщо Ω_i є відрізком як у прикладах вище, то доцільним може бути вибір двох лінійних функцій, кожна з яких зануляється на своєму кінці відрізку. Так легко побачити, що у (5.29) функціями форми є

$$N_i^1 = \frac{x_{i+1} - x}{x_{i+1} - x_i}, \quad N_i^2 = \frac{x - x_i}{x_{i+1} - x_i}, \tag{5.31}$$

де їх амплітуда одразу нормована на одиницю. Так само можна діяти для задач більших розмірностей. Зокрема, на тетрагональній сітці в i-му вузлі можна задавати функцію форми

$$N_i(\boldsymbol{r}) = a_i + b_i x + c_i y + d_i z \tag{5.32}$$

із умовою нормування коефіцієнтів $N_i(\boldsymbol{r}_j) = \delta_{ij}$, де $\boldsymbol{r}_j$ — радіус-вектор j-го вузла, а δ — символ Кронекера. Для спеціальних задач доцільнім може видатись використання спеціалізованих функцій, які будуть враховувати симетрію чи інші властивості задачі, наприклад з просторів $\boldsymbol{H}(\mathrm{curl})$ чи $\boldsymbol{H}(\mathrm{div})$, що часто є актуальним в задачах електродинаміки. Звернемо увагу на існування сімейства ієрархічних поліномів, які дозволяють будувати такі набори для об'ємів різної форми та розмірності. Вони також використовуються для побудови квадратурних формул, як це описувалось для МКО.

Якщо для задачі задано набір скінченних елементів, то звести диференціальне рівняння до системи алгебаїчних можна наступним чином (обмежимось випадком рівняння Пуасона із вільними межовими умовами $\nabla u|_{\partial\Omega} = 0$). Будемо шукати наближений розв'язок задачі як (5.8) і виберемо у якості тестових функцій ті ж $\psi_j(\boldsymbol{r})$. Тоді слабка форма (5.23) набуде вигляду

$$\langle \nabla\widetilde{u}, \nabla\psi_j \rangle = \langle f, \psi_j \rangle, \quad j = \overline{1, N} \tag{5.33}$$

де індекс j пробігає по всім можливим значенням, або ж, розписавши $\widetilde{u}$,

$$\left\langle \sum_{i=1}^{N} c_i \nabla\psi_i, \nabla\psi_j \right\rangle = \langle f, \psi_j \rangle, \quad j = \overline{1, N}. \tag{5.34}$$

Позначимо невідомі коефіцієнти як значення невідомої функції на вузлах $\boldsymbol{c} = \boldsymbol{u} \equiv \{u^i\}_{i=\overline{1,N}}$ і розкладемо $f(\boldsymbol{r})$ в цьому ж базисі: $f(\boldsymbol{r}) \approx \widetilde{f} = \sum_{i=1}^{N} f^i \psi_i$. Тоді останню рівність можна переписати у матричному вигляді

$$K\boldsymbol{u} = M\widetilde{\boldsymbol{f}}, \tag{5.35}$$

де $K = \{\langle \nabla\psi_i, \nabla\psi_j \rangle\}_{i,j=\overline{1,N}}$ називають матрицею жорсткості, $M = \{\langle \psi_i, \psi_j \rangle\}_{i,j=\overline{1,N}}$ — маси, а вектор $\boldsymbol{f} \equiv \{f^i\}_{i=\overline{1,N}}$. Легко побачити, що у випадку (5.28) обидві матриці є тридіагональними.

5.5 Take home message

1. Пошук наближених розв'язків диференціальних рівнянь у частинних похідних методом скінченних елементів.
2. Методи Рітца і Галеркіна для пошуку наближених розв'язків диференціальних рівнянь.
3. Скінченний елемент як невелика область простору із асоційованим набором базисних функцій.
4. Слабка форма диференціального рівняння для спрощення формалізації диференціального рівняння у алгебраїчних термінах.

Література до розділу

1. *Gander M. J.*, *Kwok F.* Numerical Analysis of Partial Differential Equations Using Maple and MATLAB. — Society for Industrial, Applied Mathematics, 08.2018. — DOI: 10.1137/1.9781611975314. — URL: https://doi.org/10.1137/1.9781611975314.
2. *Jin J.-M.* The finite element method in electromagnetics. — 3rd. — Wiley, 2014.
3. *Šolín P.*, *Segeth K.*, *Doležel I.* Higher-Order Finite Element Methods. — Boca Raton London Newy York Washington, D.C. : Chapman & Hall/CRC, 2004. — ISBN 1-58488-438-X.
4. Automated Solution of Differential Equations by the Finite Element Method / A. Logg, K.-A. Mardal, G. N. Wells [та ін.] ; за ред. A. Logg, K.-A. Mardal, G. N. Wells. — Springer, 2012. — ISBN 978-3-642-23098-1. — DOI: 10.1007/978-3-642-23099-8. — URL: https://doi.org/10.1007/978-3-642-23099-8.
5. *Öchsner A.*, *Merkel M.* One-Dimensional Finite Elements. — 2018.

Розділ 6

Задачі статистичної фізики

TBD: Metropolis algorithm

Додаток А

Об'єктно-орієнтовне програмування

А.1 Концепція

Концепція об'єктно-орієнтованого програмування (ООП) пропонує трактувати спеціальні типи даних, класи, як аналоги об'єктів «реального життя» (це також відбито в термінології «об'єкт класу `ClassName`» як змінна, яка має тип `ClassName`). На відміну від «простих» типів даних, класи[1] складаються не лише з полів, які містять безпосередньо дані, а й функцій — методів, — які призначені виконувати операції над полями.

Відповідно, поля класів — це інформація про об'єкт, а методи класів — це події, асоційовані з об'єктом. Коректне використання підходів ООП дозволяє частково перекласти на компілятор контроль за логікою роботи програми на додачу до простої перевірки синтаксису.

А.2 Принципи ООП

- **Інкапсуляція.** Не кожне поле або метод мають бути доступні для використання (виклику) зовні *області опису класу в коді*. Доступні поля або методи називають публічними. Це дозволяє контролювати операції над полями класу, наприклад, не дозволяти зміну поля без попередньої перевірки значення, на яке воно змінюється: компілятор не дозволить звернутися до поля класу напряму.

- **Спадкування.** Якщо описано клас `Bird`, який містить функціонал та дані спільні для усіх птахів, класи `Eagle` та `Kiwi` мають бути успадковані від `Bird` аби уникнути повторного написання коду та зосередитись лише на властивих орлам та ківі особливостям. Таким чином, вже написаний і протестований код може бути використаний повторно з контролем з боку компілятора (інтерпретатора) без необхідності його зміни.

[1] У межах цього обговорення ми не будемо зосереджувати увагу на реалізаціях конкретних мов програмування: так у `C++` структури теж можуть містити поля і методи, але концепція їх використання інша.

- **Поліморфізм.** Батьківський клас може передбачити наявність методу у класах-нащадках, але не зобов'язаний імплементувати його[2]. Класи, які не імплементують всі свої методи називають абстрактними. Так, батьківський клас `Flyable` передбачає наявність методу `Fly`[3], але реалізацію полишає на класи-нащадки `Airplane` і `Leaf`. Це дозволяє оперувати з об'єктами різних класів однотипно, якщо у конкретному випадку від них вимагається лише функціонал батьківського класу.

Перераховані принципи мають стосунок лише до коду, але не до інтерфейсу, з яким має справу користувач програми. Вони спрямовані на полегшення процесу програмування, зокрема контролю за помилками (наприклад, випадкова зміна значення змінної або внесення змін у вже відлагоджений код), колективну роботу, тестування та контроль версій.

A.3 S.O.L.I.D.

[2]Звертаємо увагу на відмінність поліморфізму в ООП від поліморфізму функцій взагалі. В останньому випадку під поліморфізмом розуміють можливість функції з конкретним іменем оперувати набором аргументів різних типів. Наприклад, функція `add` може обчислювати суму чисел, якщо вона викликана із цілочисельними аргументами, або виконувати конкатенацію рядків, якщо їй на вхід було передано текст.

[3]Це також може бути інтерфейс у деяких мовах програмування.

Додаток Б

Задачі математичної фізики

Б.1 Класифікація рівнянь другого порядку із частинними похідними

Значна кількість фізичних моделей, які оперують з неперервними величинами (хвильова функція, поле швидкостей, електромагнітні поля), може бути описана диференціальними рівняннями у частинних похідних другого порядку. Зокрема, у випадку невідомої функції двох змінних $u(x, y)$ рівняння називають лінійним відносно старших похідних, якщо воно має вигляд

$$a_{11}\partial_{xx}u + 2a_{12}\partial_{xy}u + a_{22}\partial_{yy}u = F(x, y, u, \partial_x u, \partial_y u), \quad (x, y) \in \mathcal{D}. \tag{Б.1}$$

Тут коефіцієнти a_{11}, a_{12}, a_{22} — функції від x та y, символ ∂ позначає частинну похідну по змінним, вказаним у нижньому індексі, а $\mathcal{D}$ — область визначення. Якщо коефіцієнти мають функціональну залежність подібну до правої частини F, то рівняння (Б.1) називають квазілінійним. Надалі усі коефіцієнти рівнянь будуть вважатись досить гладкими функціями (мається на увазі, що функцію можна диференціювати достатню кількість раз), а розв'язок u таким, що задовольняє рівняння у кожній точці області визначення, якщо не введено додаткові уточнення. Це може бути не так у значній кількості практично цікавих задач (наприклад, розповсюдження хвилі крізь ділянки простору з різними діелектричними та магнітними проникностями). У таких випадках застосовуються спеціальні аналітичні та чисельні підходи для розв'язання відповідних рівнянь, що буде обговорюватись окремо.

У залежності від співвідношень між a_{11}, a_{12} і a_{22}, рівняння (Б.1) відноситься до одної з так званих канонічних форм [1; 2][1].

1. *Еліптичний тип:* $a_{12}^2 - a_{11}a_{22} < 0$.

2. *Гіперболічний тип:* $a_{12}^2 - a_{11}a_{22} > 0$.

3. *Параболічний тип:* $a_{12}^2 - a_{11}a_{22} = 0$.

[1] Якщо у рівнянні наявні доданки з функцією u або її першими похідними, то звести його до вигляду (Б.1) можна за допомогою лінійних перетворень координат.

Важливо, що дана характеристика не змінюється при перетвореннях координат, що дозволяє робити загальні висновки про конкретне рівняння. У випадку більшої кількості змінних класифікація зберігається. Для її введення матрицю коефіцієнтів a_{ij} у багатовимірному аналогу рівняння (Б.1) діагоналізують. Якщо всі елементи діагоналі при цьому мають один знак, рівняння відносять до еліптичного типу. Гіперболічним (ультрагіперболічним) рівнянням називають таке, в якого один (декілька) з діагональних елементів має інший знак. Якщо хоча б один з діагональних елементів рівний нулю, то таке рівняння відносять до параболічного типу. Інші випадки структури діагональних елементів відносять до більш спеціальних типів.

Б.2 Рівняння гіперболічного типу

Задачі, які описують коливні процеси типово зводяться до рівнянь у частинних похідних гіперболічного типу, які можна представити у загальному вигляді

$$\rho\partial_{tt}u = \nabla\cdot(p\nabla u) - qu + F(\boldsymbol{r},t), \tag{Б.2}$$

де функція $u = u(\boldsymbol{r},t)$, а величини ρ, p і q визначаються конкретною задачею. Так, для малих коливань тонкої однорідної струни ρ і $p = \text{const}$ матимуть зміст лінійної густини і натяг, а для пружного стрижня p набуває змісту координатнозалежного модуля Юнга. Параметр q може характеризувати дисипацію енергії в системі. У задачах із $p = const$ задача (Б.2) зводиться до хвильового рівняння

$$\partial_{tt}u = \frac{1}{c^2}\Delta u - qu + F(\boldsymbol{r},t), \tag{Б.3}$$

де параметр c має зміст швидкості, а $\Delta = \nabla\cdot\nabla$ — оператор Лапласа.

До задач гіперболічного типу зводиться система рівнянь гідродинаміки

$$\partial_t\boldsymbol{v} + (\boldsymbol{v}\cdot\nabla)\boldsymbol{v} + \frac{1}{\rho}\nabla p = \boldsymbol{F}, \tag{Б.4}$$

$$\partial_t\rho + \nabla(\rho\boldsymbol{v}) = 0, \tag{Б.5}$$

$$p = f(\rho). \tag{Б.6}$$

Рівняння (Б.4) має назву рівняння Нав'є–Стокса й описує рух ідеальної рідини. Друге рівняння системи (Б.5) описує неперервність рідини й третє, (Б.6), характеризує її термодинамічні властивості.

Б.3 Рівняння параболічного типу

Типовим прикладом рівняння параболічного типу є рівняння теплопровідності

$$\rho\partial_t u = \nabla\cdot(p\nabla u) - qu + F(\boldsymbol{r},t) \tag{Б.7}$$

(порівняйте з (Б.2)). Тут $p = k(\boldsymbol{r})$ має зміст коефіцієнту теплопровідності, а q характеризує тепловий обмін з навколишнім середовищем. Таку саму структуру має рівняння, що описує дифузійні процеси з тою різницею, що p є коефіцієнтом дифузії.

Б.4 Рівняння еліптичного типу

Задачі на дослідження стаціонарних процесів часто зводяться до розв'язання рівнянь еліптичного типу, серед яких найбільш поширеним є рівняння Пуасона (Лапласа у випадку $F \equiv 0$):

$$\Delta u = F(\boldsymbol{r}) \tag{Б.8}$$

Зокрема, це задачі електростатики, де під функцією u треба розуміти скалярний потенціал, а F описує розподіл зарядів. Зазначимо, що до виду (Б.8) можна звести рівняння теплопровідності (Б.7) за певних значень його параметрів.

Б.5 Межові й початкові умови

Певне диференціальне рівняння у частинних похідних може мати безліч розв'язків. Однозначність розв'язку конкретної фізичної задачі при цьому досягається визначенням крайових: межових і початкових умов на невідому функцію. У зв'язку з цим розрізняють наступні постановки задач.

- У *задачі Коші* для рівнянь параболічного і гіперболічного типу визначені початкові умови, а область визначення $\mathcal{D} \equiv \mathbb{R}^N$, де N — кількість просторових змінних. Очевидно, межові умови при цьому відсутні.
- У *межовій задачі* рівнянь еліптичного типу визначені межові умови на межі області $\mathcal{D}$, яку позначатимемо як $\partial\mathcal{D}$.
- У *змішаній задачі* для рівнянь гіперболічного і параболічного типу наявні як початкові, так і межові умови, а область визначення $\mathcal{D}$ є обмеженою.

Розрізняють наступні основні типи межових умов, які часто зустрічаються у фізичних задачах.

- Межові умови Діріхле (перша крайова задача або задача Діріхле):

  $$u(\boldsymbol{r})|_{\boldsymbol{r}\in\partial\mathcal{D}} = g(\boldsymbol{r}). \tag{Б.9}$$

  де $g(\boldsymbol{r})$ — деяка задана функція.
- Межові умови Неймана (друга крайова задача або задача Неймана)

  $$\left.\frac{\partial u}{\partial \boldsymbol{n}}\right|_{\boldsymbol{r}\in\partial\mathcal{D}} = \boldsymbol{g}(\boldsymbol{r}), \tag{Б.10}$$

  де $\boldsymbol{n}$ — зовнішня нормаль до $\partial\mathcal{D}$, а $\partial/\partial\boldsymbol{n} \equiv (\boldsymbol{n}\cdot\nabla)$ — похідна вздовж напрямку $\boldsymbol{n}$, а під $\boldsymbol{g}(\boldsymbol{r})$ мається на увазі векторна функція.
- Межові умови Робена (або змішані, третя крайова задача)

  $$\left.\frac{\partial u}{\partial \boldsymbol{n}} + g_1(\boldsymbol{r})\left[u(\boldsymbol{r}) - g_2(\boldsymbol{r})\right]\right|_{\boldsymbol{r}\in\partial\mathcal{D}} = g(\boldsymbol{r}), \tag{Б.11}$$

  де $g_{1,2}(\boldsymbol{r})$ — деякі задані функції.

Зауважимо, що у деяких практичних задачах цікавих саме з точки зору чисельного розв'язання, межові умови можуть бути залежними від часу й мати більш складні функціональні залежності від невідомої функції u.

Б.6 Задачі

№30. Як співвідноситься між собою визначення типу диференціального рівняння за допомогою дискримінанту $a_{12}^2 - a_{11}a_{22}$ та діагоналізація у випадку довільної кількості змінних?

Література до розділу

1. *Владимиров В. С.* Уравнения математической физики. — М. : Наука, 1981. — С. 512.
2. *Тихонов А. Н.*, *Самарский А. А.* Уравнения математической физики: Учебное пособие. — М. : Изд-во МГУ, 1999. — ISBN 5-211-04138-0.