МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ
Кафедра комп’ютерних систем та мереж

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни
„Алгоритми та методи обчислень”
для студентів денної форми навчання
за напрямом
6.050102 „Комп’ютерна інженерія”

Тернопіль, 2015

Конспект лекцій розроблений у відповідності з навчальним планом напрямку 6.050102 „Комп’ютерна інженерія ”

Укладач: к.т.н. Тиш Є.В.

Рецензент:

Відповідальний за випуск: в.о.зав. каф. КС Осухівська Г.М.

Затверджено на засіданні кафедри комп’ютерних систем та мереж, протокол №____ від «___» _____2015 р.

Схвалено та рекомендовано до друку методичною комісією факультету комп’ютерно-інформаційних систем і програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя, протокол №____ від «___» _____201_ р.

Конспект лекцій складений з врахуванням методичних розробок інших вищих закладів освіти, а також матеріалів літературних джерел, перелічених в списку.

ЗМІСТ

Лекція № 1
Основи теорії алгоритмів 4

Лекція № 2
Елементи теорії похибок 9

Лекція № 3
Методи розв’язку систем лінійних алгебраїчних рівнянь 14

Лекція № 4
Наближені методи розв’язання нелінійних алгебраїчних рівнянь 19

Лекція № 5
Інтерполювання функцій 23

Лекція № 6
Оцінювання параметрів лінійної регресії методом найменших квадратів 26

Лекція № 7
Чисельне інтегрування 28

Лекція № 8
Чисельне диференціювання 31

Лекція № 9
Чисельне інтегрування звичайних диференціальних рівнянь першого порядку, розв’язаних відносно похідної однокроковими методами.
Розв’язок задачі Коші 35

Лекція № 10
Лінійне програмування. Постановка задачі. Транспортна задача 38

Лекція № 11
Розв’язання задачі лінійного програмування графічним методом 43

Лекція № 12
Пошук початкового опорного плану 48

Лекція № 13
Симплексний метод розв’язання задачі лінійного програмування 54

Перелік рекомендованої літератури 58

Лекція №1

ОСНОВИ ТЕОРІЇ АЛГОРИТМІВ

1.1 Визначення алгоритму

Поняття теорії алгоритмів завжди традиційно відносилося до «високої» науки, рахувалося слабо пов'язаною з практикою та важким для розуміння. Але життя заперечило це уявлення, й це доводиться тим, що виникли прикладні відгалуження цієї теорії, а саме, алгоритмічні мови програмування та теорія формальних обчислень, знання яких стало зовсім необхідним для будь-якого дослідника, що має відношення до алгоритмізації процесів керування.

Алгоритм може бути визначений як ефективна процедура, що однозначно призводить до результату. При розробці алгоритму необхідно формалізувати процес розв'язку задачі, звівши його до застосування скінченної послідовності достатньо простих правил. Так, ми регулярно користуємося алгоритмами виконання основних арифметичних операцій над багатозначними числами, що були розроблені ще в IX ст. математиком Аль-Хорезмі (термін «алгоритм» походить з імені цього вченого). До середини XIX ст. всі алгоритми були обчислювальними та мали діло, переважно, з числами. Тому вся різноманітність обчислювань комбінувалася з 10-15 чітко визначених операцій арифметики, тригонометрії та аналізу. Все було очевидним та не вимагало спеціальних досліджень.

Поява в другій половині XIX ст. математики, що розглядала нечислові об'єкти (неевклідова геометрія, теорія груп, теорія множин) показала, що все не так просто та очевидно. Виявилось, що звичні розмірковування, які застосовуються до цих теорій, засновані на змістовному описі алгоритму, призводять до нерозв'язуваних протиріч (приклад – так звані протиріччя теорії множин). Зовсім особливим чином та вкрай обережно необхідно поводитись з таким поняттям, як «нескінченність». Були створені так звані фінітні методи досліджень, сутність яких полягала в тому, що вони допускають тільки скінченні комплекси дій над скінченною кількістю об'єктів. Все це потребувало уточнення поняття «алгоритм».

Нез'ясованість змістовного опису алгоритму стала причиною уточнення поняття алгоритму, в результаті чого був зроблений висновок про існування так званої алгоритмічної нерозв'язуваності, тобто про неіснування єдиного алгоритму розв'язку задач деякого класу при можливості розв'язку окремих задач цього класу.

Тому предметом теорії алгоритмів, що розв'язує задачу уточнення поняття «алгоритм», є в першу чергу уточнення таких понять, як «об'єкт», який підлягає перетворенню та «дія», що здійснюється над цим об'єктом. Тому, розробляючи алгоритм, необхідно попередньо вияснити:

- які об'єкти необхідно вважати точно визначеними;
- які елементарні дії над ними необхідно вважати точно визначеними;
- якими властивостями та можливостями володіють комбінації елементарних дій, тобто що можливо чи неможливо зробити за допомогою цих

дій;

- яким вимогам повинна задовольняти послідовність дій, щоб вважатися конструктивно заданою, тобто мати право називатися алгоритмом.

Враховуючи вище сказане та застосовуючи до обчислювальної техніки, можна сформулювати наступне визначення алгоритму. *Алгоритм* – це точний опис про виконання у визначеному порядку деякої системи операцій для розв'язання всіх задач деякого заданого типу.

Тим не менше, не можна вважати алгоритмом будь-яку інструкцію, розбиту на кроки. Тому необхідно визначити основні вимоги, що висуваються до алгоритмів:

1. Алгоритм застосовується до вхідних даних та видає результати. В звичних технічних термінах це значить, що алгоритм має входи та виходи. Окрім того, в ході роботи алгоритму проявляються проміжні результати, які використовуються в подальшому. Таким чином, кожний алгоритм працює з даними: вхідними, проміжними та вихідними.

2. Необхідно уточнити поняття «дані», тобто вказати, яким вимогам повинні задовольняти об'єкти, щоб алгоритм міг з ними працювати. Для цього об'єкти повинні бути чітко визначені та відрізнятися як один від іншого, так й від «необ'єктів».

Оскільки точне та повне словесне визначення об'єкта дати достатньо важко, в теорії алгоритмів фіксують конкретні скінченні набори вхідних об'єктів, які називаються елементарними та конкретний набір засобів побудови інших об'єктів з елементарних об'єктів. Набір елементарних об'єктів утворює скінченний алфавіт вхідних символів (цифр, букв, тощо), з яких будуються інші об'єкти.

Найпоширеніший тип алгоритмічних даних – слова скінченної довжини в скінченних алфавітах (наприклад, числа), причому кількість символів в словах (довжина слова) – одиниця виміру інформації, що обробляється. Більш складніший випадок алгоритмічних об'єктів – формули. Вони також є словами скінченної довжини в даному скінченному алфавіті, але не кожне слово є формула.

3. Данні для розміщення вимагають пам'яті. Пам'ять зазвичай вважається однорідною та дискретною, тобто складається з однакових комірок, кожна комірка може мати в собі один символ алфавіту. При цьому пам'ять може бути нескінченною.

4. Алгоритм складається з окремих елементарних кроків (дій), причому кількість їх скінченна. Прикладом множини елементарних дій є система команд комп'ютера. Зазвичай елементарний крок має справу з фіксованою кількістю символів, але ця вимога не завжди виконується (наприклад, динамічна пам'ять, що використовується в програмуванні.

Послідовність кроків детермінована, тобто після кожного кроку вказується, що робити далі.

5. Алгоритм повинен бути результативним, тобто повинен зупинитися після скінченної кількості кроків з одночасним вказуванням того, що рахувати результатом.

Зокрема, будь хто, хто пред'являє алгоритм рішень задачі, наприклад

знаходження деякої функції $f(x)$, зобов'язаний показати, що алгоритм зупиняється після скінченної кількості кроків (збігається) для будь-якого x з області визначення функції f.

6. Алгоритм повинен задовольняти вимозі масовості, тобто алгоритм повинен бути таким, щоб його можна було застосувати для класу задач, які відрізняються лише вхідними даними.

7. В процесі створення алгоритму слід розрізняти поняття:

- опис алгоритму (наприклад, інструкцію чи програму);
- механізм реалізації алгоритму, який включає в себе засоби управління процесом обчислення, а особливо: засоби запуску, засоби зупинки, засоби реалізації елементарних кроків, засоби видачі результатів, забезпечення детермінованості;
- процес реалізації алгоритму, який представляє собою послідовність кроків, яка буде породжуватись при застосуванні алгоритму до конкретних даних.

1.2 Блок-схеми алгоритмів, композиція алгоритмів

Для представлення алгоритму та організації зв'язку між його кроками використовуються блок-схеми. Блок-схема алгоритму зображується у вигляді графа, в якому вершинам відповідають кроки алгоритму, а ребрам – переходи між кроками. Вершини можуть бути операторними або операторами (виходить одне ребро), умовними або предикатами (виходить два ребра), перемикальними (виходить декілька ребер), єдиний оператор початку (виходить одне ребро), єдиний оператор кінця (не виходить жодного ребра).

Важливою особливістю алгоритму є те, що зв'язки, які описує блок-схема, не залежать від того, чи являються кроки алгоритму елементарними чи представляють собою самостійні алгоритми (блоки).

В програмуванні відоме поняття «розблочування» складного алгоритму, коли окремі його блоки програмуються різними особами. І навпаки, за допомогою блок-схем можна декілька окремих алгоритмів розглядати як блоки, які можуть бути пов'язані в один великий алгоритм.

Наприклад, блок-схема, яка обчислює функцію $f = f_2(f_1(x))$, в якій в якості аргументу використовується інша функція та яка тому й називається композитною функцією, буде мати вигляд, який зображено на рисунку 1.1.

В такій блок-схемі вхідні дані (входи) алгоритму A_2 є результати (виходи) алгоритму A_1. Таке поєднання алгоритмів називається композицією алгоритмів.

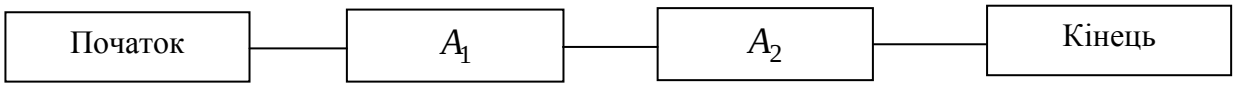


Рисунок 1.1 – A_1 – алгоритм обчислення функцій $f_1(x)$, A_2 – алгоритм обчислення функцій $f_2(x)$.

На блок-схемі алгоритму добре видно різницю між описом алгоритму та процесом його реалізації. Опис – це граф, а процес реалізації – це шляхи в графі, які визначаються логічними умовами. Якщо в процесі реалізації обчислень не з'являються умови, які ведуть до кінця, та процес зациклюється, то це означає відсутність збіжності алгоритму.

1.3 Алгоритмічні моделі

В класичному варіанті блок-схеми алгоритмів, при всій своїй наочності, відображають зв'язки лише по керуванню (що робити в наступний момент, якому блоку передати керування), а не по інформації (не містять інформації ні про дані, ні про пам'ять, ні про набір елементарних кроків, які використовуються). Таким чином, представлення алгоритму у вигляді блок-схеми не задовольняє перераховані вище вимоги.

Тому в теорії алгоритмів приймається й інший підхід до представлення алгоритму, який дозволяє побудувати алгоритмічну модель процесу, коли вдається задовольнити всі вимоги, які висуваються для алгоритму. При побудові алгоритмічної моделі проводиться уточнення детермінізму алгоритму:

- вибирається скінченний набір вхідних об'єктів, які називаються елементарними;
- вибирається скінченний набір способів побудови із них нових об'єктів;
- фіксується набір елементарних кроків;
- розробляється спосіб створення пам'яті та вибірки інформації з пам'яті.

В результаті цих дій створюється конкретна алгоритмічна модель.

Можна виділити три основних типа алгоритмічних моделей, які відрізняються евристичними міркуваннями відносно того, що таке алгоритм.

В *першій моделі* поняття алгоритму пов'язується з найбільш традиційними поняттями математики – обчисленнями та числовими функціями. Модель цього типу, яка найбільш широко використовується – є *рекурсивні функції*. Рекурсивна функція описується за допомогою так званих індуктивних (або рекурентних) визначень, які основані на переході від n до $n+1$. Слово «рекурентний» означає «зворотний» від латинського слова *recurso* – «повертаюсь, біжу назад». І якщо алгоритм носить зворотний характер, то він й називається рекурсивним. Прикладом може бути алгоритм отримання натурального ряду чисел: щоб визначити число 4, потрібно спочатку визначити число 3, а щоб визначити число 3, треба спочатку визначити число 2 і так далі до 1.

Рекурсивні функції можуть бути визначені як деякі нові функції, які отримуються із вже існуючих функцій за допомогою операцій суперпозицій.

Оператором суперпозицій S_m^n називається підстановка у функцію від m змінних m функцій від n одних й тих самих змінних. Наприклад, якщо маємо функції $h(x_1, x_2, \ldots, x_m)$, $g_1(x_1, x_2, \ldots, x_n)$, $g_2(x_1, x_2, \ldots, x_n), \ldots$, $g_m(x_1, x_2, \ldots, x_n)$, то

$$S_m^n = (h, g_1, g_2, \ldots, g_m) = h\left(g_1\left(x_q, x_2, \ldots, x_n\right)\right), g_2(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n)) =$$
$$= f(x_1, x_2, \ldots, x_n).$$

Другий тип алгоритмічної моделі базується на уявлені про алгоритм як про деякий детермінований пристрій, який здатен виконувати в кожний окремий момент часу лише дуже примітивні операції.

Евристика цих моделей дуже близька до обчислювальних машин та, відповідно, близька до інженерної інтуїції. Основною теоретичною моделлю цього типу, створеною у 30-их роках, є *машина Тьюринга*.

Третій тип алгоритмічних моделей – це перетворення слів в довільних алфавітах, в яких елементарними операціями є операції підстановки, тобто заміни частини слова (підслова) іншим словом. Прикладом таких моделей є *скінченні автомати*, створені за алфавітним відображенням, *канонічні системи Поста* та *нормальні алгоритми Маркова*.

Контрольні питання

1. Дайте визначення алгоритму.
2. Наведіть основні вимоги, що висуваються до алгоритмів.
3. Блок-схема алгоритму.
4. Композиція алгоритмів.
5. Наведіть три основні типи алгоритмічних моделей.

Лекція № 2

ЕЛЕМЕНТИ ТЕОРІЇ ПОХИБОК

2.1 Абсолютна та відносна похибки

Наближеним числом a називається число, яке незначно відрізняється від точного числа A та може замінити його у розрахунках.

Якщо $a < A$, то a називається наближеним значенням числа A з недостачею, якщо $a > A$, то a називається наближеним значенням числа A з надлишком. Якщо a є наближеним значенням числа A, то пишуть $a \approx A$.

В більшості випадків точне значення A не відоме і відповідно похибка наближеного числа a не може бути визначена.

Однак, як правило, можна вказати число, яке оцінює зверху абсолютну величину похибки наближеного числа a. Це число і буде абсолютною похибкою розглядуваного наближеного числа a.

ОЗНАЧЕННЯ. Число, яке оцінює зверху абсолютну величину похибки наближеного числа a називають **абсолютною похибкою.**

Символічно це позначають так

$$|A - a| < \Delta(a), \tag{2.1}$$

де A- точне (дійсне) значення величини, a- наближене значення величини, $\Delta(a)$- абсолютна похибка величини (число, яке оцінює зверху абсолютну величину похибки наближеного числа a).

Нерівність (1.1.1) можна записати у відповідності до означення абсолютної величини числа, наступним чином

$$-\Delta(a) < A - a < \Delta(a),$$

звідки

$$a - \Delta(a) < A < a + \Delta(a).$$

ОЗНАЧЕННЯ. **Відносною похибкою** наближеного числа a називають відношення абсолютної похибки наближеного числа a до модульного значення наближеного числа a, яку символічно позначають так

$$\delta(a) = \frac{\Delta(a)}{|a|}. \tag{2.2}$$

2.2 Правила заокруглення

Заокруглення числа полягає у відкиданні одного або декількох останніх десяткових знаків, а якщо їх немає, то в заміні однієї або декількох останніх цифр цілої частини числами нулями.

При цьому використовують наступні правила:

а) останнє число не змінюється, якщо перше число, яке відкидається менше < 5, наприклад, $3{,}53 \approx 3{,}5$;

б) останнє число збільшується на одиницю, якщо перше число, яке відкидається більше > 5, наприклад, $2{,}78 \approx 2{,}8$;

в) при відкиданні або заміні лише одного числа 5 прийнято виконувати

правило парного числа: останнє число, яке зберігається не змінюється, якщо воно парне та збільшується на одиницю, якщо воно непарне, наприклад, $2{,}8025 \approx 2{,}802$, $25{,}35 \approx 25{,}4$.

2.3 Абсолютні та відносні похибки суми та різниці чисел

Нехай задано точні значення двох чисел A_1 і A_2, а також їх наближені значення відповідно a_1 і a_2 з абсолютними похибками $\Delta(a_1)$ і $\Delta(a_2)$.

Запишемо нерівності

$$a_1 - \Delta(a_1) < A_1 < a_1 + \Delta(a_1),$$
$$a_2 - \Delta(a_2) < A_2 < a_2 + \Delta(a_2),$$

додаючи їх почленно, отримаємо

$$(a_1 + a_2) - (\Delta(a_1) + \Delta(a_2)) < A_1 + A_2 < (a_1 + a_2) + (\Delta(a_1) + \Delta(a_2)).$$

З останнього виразу випливає, що абсолютна похибка суми рівна сумі абсолютних похибок, що символічно позначається

$$\Delta(a_1 + a_2) = \Delta(a_1) + \Delta(a_2). \qquad (2.3)$$

Використовуючи метод математичної індукції можна показати, що для довільних додатніх a_1, a_2, a_3, ..., a_n абсолютна похибка суми рівна

$$\Delta(a_1 + a_2 + a_3 + \ldots + a_n) = \Delta(a_1) + \Delta(a_2) + \Delta(a_3) + \ldots + \Delta(a_n). \qquad (2.3')$$

Згідно (1.1.2) і (1.3.1) маємо

$$\delta(a_1 + a_2) = \frac{\Delta(a_1 + a_2)}{a_2 + a_2} = \frac{\Delta(a_1) + \Delta(a_2)}{a_1 + a_2}.$$

Таким чином, відносна похибка суми визначається

$$\delta(a_1 + a_2) = \frac{\Delta(a_1) + \Delta(a_2)}{a_1 + a_2}. \qquad (2.4)$$

Відносна похибка суми наближених чисел a_1, a_2, a_3, ..., a_n згідно (2.3′) має вид

$$\delta(a_1 + a_2 + a_3 + \ldots + a_n) = \frac{\Delta(a_1) + \Delta(a_2) + \Delta(a_3) + \ldots + \Delta(a_n)}{a_1 + a_2 + a_3 + \ldots + a_n}. \qquad (2.4')$$

Щоб знайти відносну похибку різниці двох додатніх чисел для початку запишемо нерівності

$$a_1 - \Delta(a_1) < A_1 < a_1 + \Delta(a_1),$$
$$a_2 - \Delta(a_2) < A_2 < a_2 + \Delta(a_2).$$

Переписавши ці нерівності у виді

$$a_1 - \Delta(a_1) < A_1 < a_1 + \Delta(a_1),$$
$$a_2 + \Delta(a_2) > A_2 > a_2 - \Delta(a_2)$$

і віднімаючи почленно, отримаємо

$$(a_1 - a_2) - (\Delta(a_1) + \Delta(a_2)) < A_1 - A_2 < (a_1 - a_2) + (\Delta(a_1) + \Delta(a_2)).$$

Звідси слідує, що

$$\Delta(a_1 - a_2) = \Delta(a_1) + \Delta(a_2). \qquad (2.5)$$

Із формули (2.2) та (2.5) відносна похибка різниці визначається співвідношенням

$$\delta(a_1 - a_2) = \frac{\Delta(a_1 - a_2)}{|a_1 - a_2|} = \frac{\Delta(a_1) + \Delta(a_2)}{|a_1 - a_2|}.$$

Таким чином

$$\delta(a_1 - a_2) = \frac{\Delta(a_1) + \Delta(a_2)}{|a_1 - a_2|}. \tag{2.6}$$

2.4 Абсолютні та відносні похибки добутку та відношення чисел

Для початку виразимо абсолютну похибку $\Delta(a)$ із (2.2) через відносну похибку $\delta(a)$ та модульне значення a

$$\Delta(a) = |a|\delta(a). \tag{2.7}$$

Нехай дано точні (дійсні) значення A_1 та A_2 - додатні числа a_1 і a_2 їх приблизні значення $\Delta(a_1)$, $\Delta(a_2)$, $\delta(a_1)$ та $\delta(a_2)$ - абсолютні і відносні похибки, які вважаємо достатньо малими.

Запишемо нерівності для точних значень A_1 і A_2

$$\begin{aligned} a_1 - \Delta(a_1) < A_1 < a_1 + \Delta(a_1), \\ a_2 - \Delta(a_2) < A_2 < a_2 + \Delta(a_2). \end{aligned} \tag{2.8}$$

Знайдемо для початку відносну похибку добутку $\delta(a_1 \cdot a_2)$, а після того згідно (2.7) знайдемо вираз для абсолютної похибки добутку $\Delta(a_1 \cdot a_2)$. Така послідовність обумовлена лекшим виводом виразів для $\delta(a_1 \cdot a_2)$ і $\Delta(a_1 \cdot a_2)$. Перепишемо нерівності (2.8) згідно (2.7) в наступному виді

$$\begin{aligned} a_1 - a_1 \cdot \delta(a_1) < A_1 < a_1 + a_1 \cdot \delta(a_1), \\ a_2 - a_2 \cdot \delta(a_2) < A_2 < a_2 + a_2 \cdot \delta(a_2). \end{aligned} \tag{2.9}$$

далі виконаємо наступні перетворення

$$\begin{aligned} a_1(1 - \delta(a_1)) < A_1 < a_1(1 + \delta(a_1)), \\ a_2(1 - \delta(a_2)) < A_2 < a_2(1 + \delta(a_2)) \end{aligned}$$

виконаємо почленне перемноження

$$\begin{aligned} a_1 a_2(1 - \delta(a_1))(1 - \delta(a_2)) < A_1 \cdot A_2 < a_1 a_2(1 + \delta(a_1))(1 + \delta(a_2)) \\ a_1 a_2(1 - (\sigma(a_1) + \sigma(a_2)) + (\sigma(a_1) \cdot \sigma(a_2))) < A_1 \times \\ \times A_2 < a_1 a_2(1 + (\sigma(a_1) + \sigma(a_2)) + (\sigma(a_1) \cdot \sigma(a_2))), \end{aligned} \tag{2.10}$$

як зазначалося вище відносні похибки $\delta(a_1)$ та $\delta(a_2)$ вважаються достатньо малими, тоді їх добуток $\delta(a_1) \cdot \delta(a_2)$ буде величина ще меншого порядку, тому добутком в нерівності можна знехтувати, отже

$$\begin{aligned} a_1 a_2(1 - (\delta(a_1) + \delta(a_2))) < A_1 \cdot A_2 < a_1 a_2(1 + (\delta(a_1) + \delta(a_2))), \\ a_1 a_2 - a_1 a_2(\delta(a_1) + \delta(a_2)) < A_1 \cdot A_2 < a_1 a_2 + a_1 a_2(\delta(a_1) + \delta(a_2)). \end{aligned}$$

Але згідно (2.7)

$$a_1 a_2(\delta(a_1) + \delta(a_2)) = \Delta(a_1 \cdot a_2) \tag{2.11}$$

і останню нерівність можна записати так

$$a_1 \cdot a_2 - \Delta(a_1 \cdot a_2) < A_1 \cdot A_2 < a_1 \cdot a_2 + \Delta(a_1 \cdot a_2).$$

Таким чином відносна похибка добутку наближених чисел $a_1 \cdot a_2$

рівна сумі їх відносних похибок

$$\delta(a_1 \cdot a_2) = \delta(a_1) + \delta(a_2). \qquad (2.12)$$

Відносна похибка добутку наближених чисел $a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n$ рівна

$$\delta(a_1 \cdot a_2 \cdot a_3 \cdot \ldots \cdot a_n) = \delta(a_1) + \delta(a_2) + \delta(a_3) + \ldots + \delta(a_n). \qquad (2.12')$$

Знайдемо з (2.11) вираз для абсолютної похибки добутку наближених чисел $a_1 \cdot a_2$

$$\Delta(a_1 \cdot a_2) = a_1 \cdot a_2(\delta(a_1) + \delta(a_2)) = a_1\delta(a_1) \cdot a_2 + a_2\delta(a_2) \cdot a_1 =$$
$$= \Delta(a_1) \cdot a_2 + \Delta(a_2) \cdot a_1.$$

Таким чином

$$\Delta(a_1 \cdot a_2) = \Delta(a_1) \cdot a_2 + \Delta(a_2) \cdot a_1. \qquad (2.13)$$

Розглянемо абсолютну похибку оберненого числа до a, тобто абсолютну похибку $1/a$, вважаючи, що відносна похибка $\delta(a)$ досить мала.

$$\Delta\left(\frac{1}{a}\right) = \left|\frac{1}{a} - \frac{1}{a - \Delta(a)}\right| = \frac{\Delta(a)}{a^2(1 - \Delta(a)/a)} = \frac{\Delta(a)}{a^2(1 - \delta(a))},$$

нехтуючи величиною $\delta(a)$, внаслідок її малості у порівнянні з одиницею, отримаємо

$$\Delta(1/a) = \Delta(a)/a^2,$$

завдяки (2.2)

$$\delta\left(\frac{1}{a}\right) = \frac{\Delta(1/a)}{1/a} = \frac{\Delta(a)}{a} = \delta(a). \qquad (2.14)$$

Розглядаючи тепер ділення, як множення на обернену величину, завдяки (2.12) та (2.14) отримується формула для відносної похибки ділення

$$\delta(a_1/a_2) = \delta(a_1) + \delta(a_2), \qquad (2.15)$$

відповідно до (1.4.1) абсолютна похибка відношення рівна

$$\Delta\left(\frac{a_1}{a_2}\right) = \left(\frac{a_1}{a_2}\right) \cdot (\delta(a_1) + \delta(a_2)) = \frac{1}{a_2}\Delta(a_1) + \frac{a_1}{a_2} \cdot \frac{a_2}{a_2} \cdot \delta(a_2) =$$
$$= \frac{1}{a_2}\Delta(a_1) + \frac{a_1}{a_2^2} \cdot \Delta(a_2) = \frac{1}{a_2}\left(\Delta(a_1) + \frac{a_1}{a_2}\Delta(a_2)\right).$$

Таким чином

$$\Delta\left(\frac{a_1}{a_2}\right) = \frac{1}{a_2}\left(\Delta(a_1) + \frac{a_1}{a_2}\Delta(a_2)\right).$$

1.5 Абсолютна та відносна похибки функції

Нехай задано диференційовану функцію $y = f(x)$. Нехай x_0 наближене значення аргументу, а $\Delta(x_0)$- абсолютна похибка наближеного значення аргументу x_0. Якщо $\Delta(x_0)$ достатньо мала, то має місце формула

$$\Delta(y_0) = |f'(x_0)|\Delta(x_0).$$

Відносна похибка функції відповідно рівна

$$\delta(y_0)=\frac{\Delta(y_0)}{|y_0|}=\frac{|f'(x_0)|\Delta(x_0)}{|f(x_0)|},$$

таким чином

$$\delta(y_0)=\left|\frac{f'(x_0)}{f(x_0)}\right|\Delta(x_0).$$

Нехай задано аналітичну функцію n кількості змінних $y=f(x_1,x_2,\ldots,x_n)$. Якщо задано наближені значення аргументів відповідно $\tilde{x}_1$, $\tilde{x}_2$, $\tilde{x}_3$,..., $\tilde{x}_n$ і їх абсолютні похибки $\Delta\tilde{x}_1$, $\Delta\tilde{x}_2$, $\Delta\tilde{x}_3$,..., $\Delta\tilde{x}_n$. То абсолютну похибку функції можна знайти як

$$\Delta(y(\tilde{x}_1,\tilde{x}_2,\tilde{x}_3,\ldots,\tilde{x}_n))=\sum_{i=1}^{n}\left|f'_{x_i}(\tilde{x}_1,\tilde{x}_2,\tilde{x}_3,\ldots,\tilde{x}_n)\right|\cdot\Delta(\tilde{x}_i),$$

$$\delta(y(\tilde{x}_1,\tilde{x}_2,\tilde{x}_3\ldots,\tilde{x}_n))=\sum_{i=1}^{n}\left|\frac{f'_{x_i}(\tilde{x}_1,\tilde{x}_2,\tilde{x}_3,\ldots,\tilde{x}_n)}{f(\tilde{x}_1,\tilde{x}_2,\tilde{x}_3,\ldots,\tilde{x}_n)}\right|\cdot\Delta(\tilde{x}_i).$$

Контрольні питання

1. Дайте означення абсолютної та відносної похибки.
2. Наведіть правила заокруглення.
3. Виведіть абсолютні та відносні похибки суми та різниці двох чисел.
4. Виведіть абсолютні та відносні похибки добутку та відношення чисел.
5. Виведіть абсолютну та відносну похибки функції.

Лекція № 3

МЕТОДИ РОЗВ'ЯЗКУ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

До розв'язку систем лінійних рівнянь зводиться велика кількість практичних задач. Можна стверджувати, що розвязок лінійних систем є одною з самих поширених та важливих задая обчислювальної математики.

Запишемо *систему n лінійних алгебраїчних рівнянь рівнянь з n невідомими:*

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{cases} \tag{3.1}$$

необхідно знайти такі значення $x_1, x_2, \ldots, x_n$, які перетворюють рівняння системи (3.1) в тотожності. Таким чином розв'язком системи рівнянь (3.1) є множина $\{x_i : i = \overline{1,n}\}$.

Методи розв'язку систем лінійних рівнянь поділяються на дві групи – прямі та ітераційні. *Прямі методи* використовують скінченні співвідношення (формули) для розрахунку невідомих. Вони дають розв'язок після виконання завчасно відомої кількості операцій. Ці методи порівняно прості та більш універсальні, тобто можуть застосовуватися до розв'язання широкого класу лінійних систем.

Разом з тим ці методи мають й низку недоліків. Як правило, вони потребують зберігання в оперативній пам'яті ЕОМ одразу всієї матриці, та при великих значеннях *n* витрачається багато місця в пам'яті. Далі, прямі методи зазвичай не враховують структуру матриці – при великій кількості нульових елементів в розріджених матрицях ці елементи займають місці в пам'яті машини, та над ними проводяться арифметичні дії. Суттєвим недоліком прямих методів є також накопичення похибок в процесі розв'язання, оскільки розрахунки на будь-якому етапі використовують результати попередніх операцій.

Прямі методи лінійних систем також називають *точними*, оскільки розв'язок виражається у вигляді точних формул через коефіцієнти системи. Але точний розв'язок може бути отриманий лише при виконанні розрахунків з нескінченим числом розрядів. На практиці при використанні ЕОМ розрахунки проводяться з обмеженим числом знаків, що визначаються розрядом машини. Тому не можливо уникнути похибок в кінцевому результаті.

Ітераційні методи – це методи послідовних наближень. В них необхідно задати деякий наближений розв'язок – початкове наближення. Після цього за допомогою деякого алгоритму проводиться один цикл розрахунків, що називають *ітерацією*. Ітерації проводять до отримання розв'язку з заданою точністю.

Хоча ітераційні методи зазвичай біль складні за прямі, в ряді випадків їм

надається перевага. Вони потребують зберігання в пам'яті машини не всієї матриці системи, а лише декілька векторів з n компонентами. Деколи елементи матриці можна зовсім не зберігати, а розраховувати їх по мірі необхідності. Похибки остаточних результатів при використанні ітераційних методів не накопичуються, оскільки точність розрахунків в кожній ітерації визначається лише результатами попередніх ітерацій і практично не залежать від раніше виконаних розрахунків. Ці переваги ітераційних методів роблять їх особливо корисними в випадку великої кількості рівнянь, а також погано обумовлених систем. Необхідно відмітити, що при цьому збіжність ітерацій може бути дуже повільною; тому відшукуються ефективні шляхи її прискорення.

3.1 Прямі методи

Метод Крамера. Система рівнянь (3.1) має один розв'язок, якщо визначник

$$\Delta = \begin{vmatrix} a_{11} a_{12} \dots a_{1n} \\ a_{21} a_{22} \dots a_{2n} \\ \vdots \quad \vdots \quad \vdots \\ a_{n1} a_{n2} \dots a_{nn} \end{vmatrix}$$

відмінний від нуля. Розв'язок системи (1.1) можна шукати за правилом Крамера:

$$x_1 = \frac{\Delta_1}{\Delta}, \quad x_2 = \frac{\Delta_2}{\Delta}, \quad \dots, \quad x_n = \frac{\Delta_n}{\Delta},$$

де

$$\Delta_1 = \begin{vmatrix} b_1 a_{12} \dots a_{1n} \\ b_2 a_{22} \dots a_{2n} \\ \vdots \quad \vdots \quad \vdots \\ b_n a_{n2} \dots a_{nn} \end{vmatrix}, \quad \Delta_2 = \begin{vmatrix} a_{11} b_1 \dots a_{1n} \\ a_{21} b_2 \dots a_{2n} \\ \vdots \quad \vdots \quad \vdots \\ a_{n1} b_n \dots a_{nn} \end{vmatrix}, \quad \Delta_n = \begin{vmatrix} a_{11} a_{12} \dots b_1 \\ a_{21} a_{22} \dots b_2 \\ \vdots \quad \vdots \quad \vdots \\ a_{n1} a_{n2} \dots b_n \end{vmatrix}.$$

Цей метод має недолік, що полягає в обмеженні кількості рівнянь системи (3.1).

Метод Гауса. Нехай дано систему лінійних алгебраїчних рівнянь

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = a_{14}, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = a_{24}, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = a_{34}, \end{cases} \tag{3.2}$$

визначник якої відмінний від нуля $\Delta \neq 0$. Вибираємо рівняння із системи таке в якому $a_{i1} \neq 0$, де $i = \overline{1,3}$. Нехай таким є перше рівняння, тобто $a_{11} \neq 0$, його будемо називати ведучим елементом.

Розділимо коефіцієнти першого рівняння системи (3.2) на a_{11}, отримаємо рівняння

$$x_1 + b_{12}x_2 + b_{13}x_3 = b_{14}, \tag{3.3}$$

де

$$b_{1k} = \frac{a_{1k}}{a_{11}}, \quad k = \overline{2,4}.$$

Помножимо рівняння (3.3) на $-a_{21}$ та додамо його до другого рівняння системи (3.2), отримаємо

$$a_{22}^{(1)} x_2 + a_{23}^{(1)} x_3 = a_{24}^{(1)}, \tag{3.4}$$

аналогічно повторимо цю ж операцію, тобто помножимо рівняння (3.3) на $-a_{31}$ і додамо до третього рівняння системи (3.2), в результаті отримаємо

$$a_{32}^{(1)} x_2 + a_{33}^{(1)} x_3 = a_{34}^{(1)}. \tag{3.5}$$

Запишемо систему з рівнянь (1.4) та (1.5)

$$\begin{cases} a_{22}^{(1)} x_2 + a_{23}^{(1)} x_3 = a_{24}^{(1)}, \\ a_{32}^{(1)} x_2 + a_{33}^{(1)} x_3 = a_{34}^{(1)}. \end{cases} \tag{3.6}$$

Таким чином загальний вигляд для коефіцієнтів $a_{ik}^{(1)}$ системи (3.6) буде наступним:

$$a_{ik}^{(1)} = a_{ik} - a_{i1} b_{1k}, \qquad \left(i = \overline{2,3}, k = \overline{2,4}\right).$$

Розділивши на $a_{22}^{(1)} \neq 0$, де $a_{22}^{(1)}$ - ведучий елемент системи (3.6), коефіцієнти першого рівняння системи (3.6), отримаємо рівняння

$$x_2 + b_{23}^{(1)} x_3 = b_{24}^{(1)}, \tag{3.7}$$

де

$$b_{2k}^{(1)} = \frac{a_{2k}^{(1)}}{a_{22}^{(1)}}, \ \left(k = \overline{3,4}\right).$$

Використовуючи рівняння (3.7) виключимо з системи (3.6) змінну x_2, що приведе до одного рівняння

$$a_{33}^{(2)} x_3 = a_{34}^{(2)},$$

де $a_{3k}^{(2)} = a_{3k}^{(1)} - a_{32}^{(1)} b_{2k}^{(1)} \ \left(k = \overline{3,4}\right)$. Якщо $a_{33}^{(2)} \neq 0$, отримаємо

$$x_3 = \frac{a_{34}^{(2)}}{a_{33}^{(2)}} = b_{34}^{(2)}. \tag{3.8}$$

Об'єднуючи рівняння (3.3), (3.7) та (3.8) отримуємо

$$\begin{cases} x_1 + b_{12} x_2 + b_{13} x_3 = b_{14}, \\ x_2 + b_{23}^{(1)} x_3 = b_{24}^{(1)}, \\ x_3 = b_{34}^{(2)}. \end{cases} \tag{3.9}$$

Процес перетворення системи (3.2) до трикутного виду (3.9) називається *прямим ходом.*

Невідомі x_1, x_2 та x_3 можна знайти із системи (3.9) за формулами *зворотнім ходом*

$$\begin{cases} x_3 = b_{34}^{(2)}, \\ x_2 = b_{24}^{(1)} - b_{23}^{(1)} x_3, \\ x_1 = b_{14} - b_{13} x_3 - b_{12} x_2. \end{cases} \tag{3.10}$$

Застосування методу Гауса для розв'язування систем лінійних рівнянь з великою кількістю невідомих досить громіздке. Крім того, кількість невідомих може бути така велика, що коефіцієнт системи не завжди можна розмістити в оперативній пам'яті ЕОМ. Тоді застосувати для її розв'язування метод гауса взагалі не можна. У цих випадках розв'язують систему ітераційними методами.

3.2 Ітераційні методи

Метод простої ітерації. Нехай задано систему лінійних алгебраїчних рівнянь

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_{1n}, \end{cases} \tag{3.11}$$

або в матричному вигляді

$$AX = B, \tag{3.12}$$

де

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots\ldots & \ldots\ldots & \ldots & \ldots\ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

Нехай діагональні елементи a_{ii} (i = 1, 2, …, n) матриці A відмінні від нуля. Тоді, розв'язавши перше рівняння системи (3.11) відносно x_1, а друге – відносно x_2 й т.д., дістанемо систему

$$\begin{cases} x_1 = a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n + \beta_1, \\ x_1 = a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n + \beta_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1} + \beta_n, \end{cases} \tag{3.13}$$

де

$$a_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}}, & i \neq j, \\ 0, & i = j, \end{cases} \qquad \beta_i = \frac{b_i}{a_{ii}}.$$

Ввівши до розгляду матриці

$$a = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots\ldots & \ldots\ldots & \ldots & \ldots\ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix},$$

систему (3.13) запишемо у вигляді

$$x = \alpha x + \beta. \tag{3.14}$$

Систему (3.14) називають системою *нормального виду*.

Розв'яжемо її методом послідовних наближень. За початкове наближення візьмемо, наприклад, стовпець вільних членів, тобто $x^{(0)} = \beta$.

Тоді послідовно знаходимо

$$x^{(k+1)} = \alpha x^{(k)} + \beta \quad (k = 0, 1, 2, \ldots) \tag{3.15}$$

або в розгорнутому вигляді

$$x_i^{(0)} = \beta_i, \tag{3.16}$$

$$x_i^{(k+1)} = \sum_{j=1}^{n} a_{ij} x_j^{(k)} + \beta_i, \qquad i = 1,2,\ldots, n; \quad k = 0,1,2,\ldots$$

Якщо послідовність наближень $x^{(0)}, x^{(1)}, \ldots, x^{(k)}, \ldots$ має границю $x^* = \lim\limits_{k\to\infty} x^{(k)}$, то ця границя і буде розв'язком системи (1.14).

Справді, перейшовши до границі, коли $k\to\infty$, у рівності (1.15), маємо

$$\lim_{k\to\infty} x^{(k+1)} = \lim_{k\to\infty} (\alpha x^{(k)} + \beta) = \alpha \lim_{k\to\infty} x^{(k)} + \lim_{k\to\infty} \beta,$$

або

$$x^* = \alpha\, x^* + \beta.$$

Таким чином, вектор

$$x^* = \begin{pmatrix} x_1^* \\ x_2^* \\ \vdots \\ x_n^* \end{pmatrix},$$

є розв'язком системи (3.14), а отже, та системи (3.11).

Метод послідовних наближень, який визначається формулами (3.15) або (3.16), називається методом *простої ітерації* або просто *методом ітерацій*.

Контрольні питання

1. Матриці: означення, квадратна, прямокутна, трикутна матриця, основні дії з матрицями, елементарні перетворення матриць, знаходження визначників.
2. Назовіть точні та наближені методи розв'язку систем рівнянь.
3. В чому суть методу Крамера?
4. В чому суть методу Гауса?
5. В чому суть методу простої ітерації?
6. Побудуйте блок-схеми зазначених методів.
7. Сформулюйте достатню умову збіжності методу ітерації для системи рівнянь.

Лекція № 4

НАБЛИЖЕНІ МЕТОДИ РОЗВ'ЯЗАННЯ НЕЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ

Наближене знаходження дійсних коренів рівняння складаються з двох етапів.

1.Визначення відрізків $[a,b]$, які містять лише один корінь рівняння. Відрізки повинні бути як найменшими. Вибір відрізку $[a,b]$ здійснюється графічно, або методом перебору.

2. Уточнення наближеного кореня шляхом застосування одного із чисельних методів (метод бісекції, метод хорд, метод дотичних, метод простої ітерації, тощо).

Метод ділення відрізка навпіл (метод бісекції). Це один з найпростіших методів знаходження коренів нелінійних рівнянь. Він полягає в наступному.

Нехай задано на відрізку $[a,b]$ неперервну функцію $y=f(x)$, яка приймає на його кінцях різні знаки (рисунок 4.1).

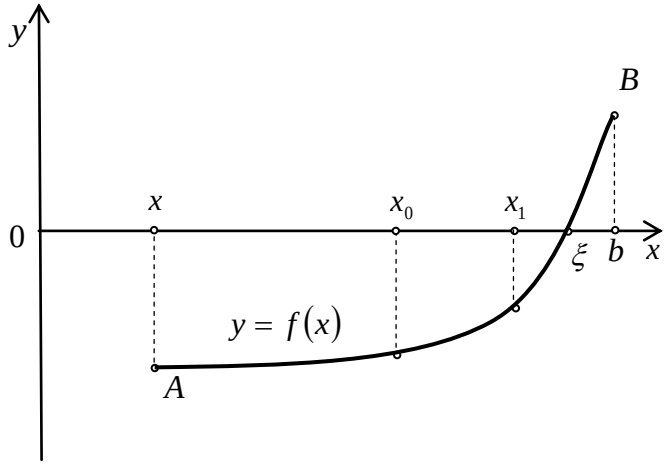


Рисунок 4.1 - Геометричний зміст методу поділу відрізку навпіл

Розіб'ємо відрізок $[a,b]$ навпіл. В наслідок чого отримаємо точку x_0:

$$x_0 = \frac{a+b}{2}.$$

Знаходимо значення $f(x_0)$ та перевіряємо чи $f(x_0)=0$. Для цього визначаємо в якому із відрізків ($[a,x_0]$ чи $[x_0,b]$) значення функції на її кінцях різного знаку. Згідно з рисунку 4.1 вибирається сегмент $[x_0,b]$, оскільки $f(b)f(x_0)\langle 0$. Після чого ділимо відрізок $[x_0,b]$ навпіл та отримуємо точку x_1 ($x_1=\frac{x_0+b}{2}$), перевіряємо знаки функцій на кінцях відрізків та вибираємо новий відрізок й т.д.. Так продовжується до тих пір, поки відрізок в якому знаходиться корінь рівняння $f(x)=0$ не стане знайденим з точністю до $\varepsilon>0$, тобто

$$\frac{b-a}{2^n}<\varepsilon,$$

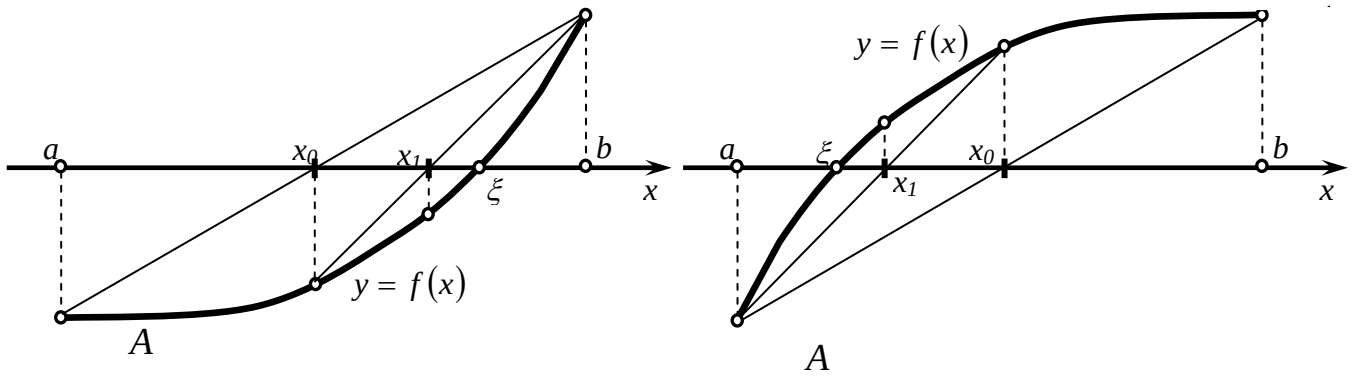
де n - кількість ітерацій.

Кінці даного відрізку й будуть давати шукане наближення кореня рівняння (лівий з недостачею, правий – з надлишком).

Можна також оцінювати довжину отриманого відрізка: якщо вона стає меншою допустимої похибки, то алгоритм зупиняється.

Метод поділу відрізка навпіл відносно повільний, але він завжди збігається.

Метод лінійної апроксимації або метод хорд. Суть методу полягає в заміні графіка функції $y = f(x)$ хордою, тобто відрізком, який з'єднує кінцеві точки графіку функції $y = f(x)$: точки $(a, f(a))$ та $(b, f(b))$ (рисунок 4.2 а,б).



а) *б)*

Рисунок 4.2 – Геометричний зміст методу хорд

Рівняння прямої, яка проходить через дві задані точки $A(a, f(a))$ та $B(b, f(b))$, має вигляд

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}. \tag{4.1}$$

Підставляючи $y = 0$ в (4.1) знаходимо абсцису x_0 точки перетину хорди з віссю абсцис:

$$x_0 = a - \frac{b - a}{f(b) - f(a)} f(a). \tag{4.2}$$

З рисунку 4.2 а,б видно, що $x_0 \in [a, b]$. Порівнюючи знаки $f(a)$, $f(x_0)$ та $f(b)$ вибираємо один з відрізків $[a, x_0]$, $[x_0, b]$. Якщо $f(a)$ та $f(x_0)$ різних знаків вибираємо $[a, x_0]$ та навпаки, якщо $f(x_0)$ та $f(b)$ різних знаків — то вибираємо інтервал $[x_0, b]$. Якщо розглядати рисунок 4.2,а, то вибирається інтервал $[x_0, b]$. Далі проводиться хорда через точки $(x_0, f(x_0))$ та $(b, f(b))$ та шукається абсциса x_1 точки перетину хорди з віссю Ox. Таким чином отримується одностороння збіжна послідовність $\{x_n; n = \overline{0, \infty}\}$, границя якої $\lim\limits_{n \to \infty} x_n$ і є коренем рівняння $f(x) = 0$.

Ітераційний процес продовжується до тих пір, поки значення $f(x_n)$ не стане за модулем менше заданого числа ε.

Алгоритми методів ділення відрізка пополам та методу хорд схожі, але другий з них в ряді випадків дає більш швидку збіжність ітераційного процесу. При цьому успіх його використання, як й методу поділу відрізку навпіл є гарантованим.

Метод дотичних або метод Ньютона. Його відмінність від попереднього методу полягає в тому, що на n –ій ітерації замість хорди проводиться дотична до кривої $y = f(x)$ при $x = z_n$ та шукається точка її перетину з віссю абсцис. При цьому не обов'язково задавати відрізок $[a,b]$, що має корінь рівняння, а достатньо знайти деяке початкове наближення $x = z_0$ (рисунок 4.3).

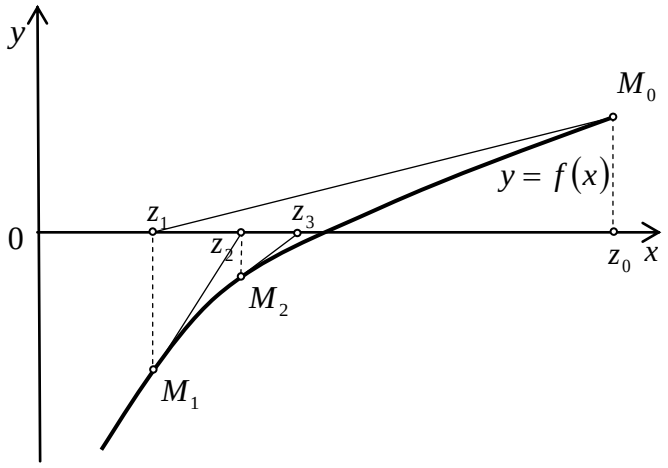


Рисунок 4.3 – Геометричний зміст методу дотичних

Рівняння дотичної, проведеної до кривої $y = f(x)$ в точці M_0 з координатами $(z_0, f(z_0))$, має вигляд:

$$y - f(z_0) = f'(z_0)(x - z_0). \tag{4.3}$$

Звідси знайдемо наступне наближення кореня z_1 як абсцису точки перетину дотичної з віссю Ox ($y = 0$):

$$z_1 = z_0 - \frac{f(z_0)}{f'(z_0)}.$$

Аналогічно можуть бути знайдені й наступні наближення як точки перетину з віссю абсцис дотичних, проведених в точках M_1, M_2, тощо. Формула для $n+1$-го наближення має вигляд

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}. \tag{4.4}$$

При цьому необхідно, щоб $f'(z_n)$ не дорівнювала нулеві. Для закінчення

ітераційного процесу може бути використане або умова $|f(z_n)|<\varepsilon$, або умова близькості двох послідовних наближень: $|z_{n+1}-z_n|<\varepsilon$.

З (4.4) випливає, що на кожній ітерації об'єм розрахунків в методі дотичних більший, ніж в розглянутих раніше методах, оскільки необхідно знаходити не тільки функції $f(x)$, але й її похідні. Але швидкість збіжності тут значно вища, ніж в інших методах.

Контрольні запитання

1. Які методи розв'язку нелінійних рівнянь вам відомі?
2. В яких випадках необхідно використовувати ітераційні методи?
3. Яким умовам повинна відповідати функція $f(x)$?
4. Назвіть способи відокремлення коренів?
5. В чому сутність ітераційного процесу?
6. В чому сутність методу половинного поділу?
7. В чому сутність методу хорд?
8. Який з кінців відрізку [*a, b*] в методі хорд рахується нерухомим?
9. Умова закінчення ітераційного процесу в методі хорд?
10. В чому сутність методу Ньютона?
11. Як вибрати початкове наближення для методу Ньютона?

Лекція № 5

ІНТЕРПОЛЮВАННЯ ФУНКЦІЙ

Задача інтерполяції функції формулюється наступним чином. Нехай на сегменті $[a,b]$ задані значення функції $f(x)$ в точках x_j, $j=\overline{0,n+1}$:

$$a = x_0 < x_1 < x_2 < ... < x_n < x_{n+1} = b.$$

Необхідно відшукати многочлен (поліном) $P(x)$ не вище заданого степеня m, який при значеннях аргументу $x = x_j$, $j=\overline{0,n+1}$, що називаються **вузлами інтерполяції**, приймав тіж значення, що і дана функція $f(x)$, тобто

$$f(x_j) = P(x_j), \quad j=\overline{0,n+1}.$$

Многочлен (поліном) $P(x)$ називається **інтерполяційним многочленом** (**інтерполяційним поліномом**), інтерполюючий функцію $f(x)$ в даних вузлах інтерполяції.

5.1 Лінійна інтерполяція

В якості інтерполяційного полінома використовується інтерполяційний многочлен першої степені $P(x) = ax + b$.

Розглянемо інтерполюючу функцію $y = f(x)$ в інтервалі $(x_j < x < x_{j+1})$ Нехай її можна замінити в даному інтервалі (x_j, x_{j+1}), з достатньою точністю хордою. Рівняння хорди, яка проходить через точки (x_j, y_j) і (x_{j+1}, y_{j+1}) має вигляд

$$\frac{y - y_j}{y_{j+1} - y_j} = \frac{x - x_j}{x_{j+1} - x_j}. \qquad (5.1)$$

За формулою (5.1) можна розрахувати значення хорди в точці x у інтервалі (x_j, x_{j+1}), тобто розрахувати наближені значення функції $f(x)$ в інтервалі $[x_j, x_{j+1}]$.

5.2 Параболічна інтерполяція

За інтерполяційну функцію на відрізку $[x_{j-1}, x_{j+1}]$ приймається квадратний тричлен. Рівняння квадратного тричлена $P(x) = ax_j^2 + bx_j + c$, $x_j - 1 \le x \le x_{j+1}$ має три невідомих коефіцієнти a, b, c. Їх знаходять з системи рівнянь

$$\begin{cases} ax_{j-1}^2 + bx_{j-1} + c = y_{j-1} \\ ax_j^2 + bx_j + c = y_j \\ ax_{j+1}^2 + bx_{j+1} + c = y_{j+1} \end{cases}$$

Таким чином, інтерполяція для будь-якої точки проводиться за трьома ближніми до неї вузлами.

5.3 Інтерполяційний поліном Лагранжа

Якщо задано значення функції $f(x)$ у вузлах інтерполяції $\{x_j : \quad j=\overline{0,n}\}$, то інтерполяційний поліном Лагранжа для відновлення даної функції $f(x)$ буде мати вид

$$L_n(x)=\sum_{j=0}^{n} f(x_j)\frac{(x-x_0)\cdot(x-x_1)\cdot(x-x_2)...(x-x_{j-1})\cdot(x-x_{j+1})...(x-x_n)}{(x_j-x_0)\cdot(x_j-x_1)\cdot(x_j-x_2)...(x_j-x_{j-1})\cdot(x_j-x_{j+1})...(x_j-x_n)}.$$

Якщо продовжити інтерполяцію відомої функції $f(x)$ інтерполяційним поліномом Лагранжа, то можна ввести поняття оцінки похибки інтерполяції. Величина

$$R_n(x)=f(x)-L_n(x)$$

називається залишковим членом інтерполяції. При умові, що $f(x)$ диференційовна $n+1$ роду, модульне значення $R_n(x)$ є обмеженим, а саме

$$|R_n(x)|\le M_{n+1}(\alpha;\beta)\frac{(x-x_0)(x-x_1)...(x-x_n)}{(n+1)!},$$

де

$$M_{n+1}(\alpha;\beta)=\max_{\alpha\le x\le\beta}\left|f^{(n+1)}(x)\right|,$$

а α та β найменше й найбільше з $n+2$ чисел

$$x_j \quad (j=0,1,2,...n) \text{ і } x.$$

Якщо розглядати значення функції інтерполяційного поліному Лагранжа в точці x, що не належить ні одному з відрізків $[x_0,x_1)$, $[x_1,x_2)$, ..., $[x_{n-1},x_n]$, то таку задачу називають задачею екстраполяції.

5.4 Інтерполяційний поліном Ньютона

Нехай задано значення функції $f(x)$ у вузлових точках $\{x_j : \quad j=\overline{0,n}\}$. Будемо розглядати таке сімейство вузлових точок $\{x_j : \quad j=\overline{0,n}\}$ віддаль між якими однакова та рівна h тобто

$$x_{j+1}-x_j=h, \quad j=\overline{0,n}.$$

Інтерполяційним поліном Ньютона записують у вигляді

$$P_n(x)=f(x_0)+\frac{\Delta f(x_0)}{h}(x-x_0)+$$

$$\frac{\Delta^2 f(x_0)}{2!h^2}(x-x_0)(x-x_1)+\frac{\Delta^3 f(x_0)}{3!h^3}(x-x_0)(x-x_1)(x-x_2)+....+$$

$$+\frac{\Delta^n f(x_0)}{n!h^n}(x-x_0)(x-x_1)...(x-x_{n-1}), \qquad (5.2)$$

Формулу (5.2) називають інтерполяційним поліномом Ньютона для інтерполяції вперед, оскільки кінцеві різниці знаходяться із значень функції в лівому вузлі. Формула, де кінцеві різниці визначаються за допомогою правих

вузлів називається формулою Ньютона для інтерполяції назад:

$$P_n(x) = f(x_n) + \frac{\Delta f(x_{n-1})}{h}(x - x_n) +$$
$$+ \frac{\Delta^2 f(x_{n-2})}{2!\,h^2}(x - x_n)(x - x_{n-1}) + \ldots + \frac{\Delta^n f(x_0)}{n!\,h^n}(x - x_n)(x - x_{n-1})\ldots(x - x_1). \quad (5.3)$$

5.5 Інтерполяція сплайнами

Нехай задані вузли інтерполяції $\{x_j : \quad j = \overline{0,n}\}$, і в інтервалах $[x_j ; x_{j+1}]$, де $j = \overline{0,n-1}$ задані алгебраїчні поліноми.

Сплайном називається функція, яка разом із своїми декількома похідними неперервна, а в кожному конкретному інтервалі $[x_j ; x_{j+1}]$, $j = \overline{0,n-1}$ дорівнює відповідному алгебраїчному поліному.

Степенем сплайна називається найвища степінь полінома із заданої сукупності поліномів.

Дефектом сплайна називається різниця між степенем сплайна і його найвищою неперервною похідною на інтервалі інтерполяції $[a,b]$.

Позначимо сплайн символічно $S_n(x)$, де n - степінь сплайна, тоді величина $S_n'(x_j)$ називається *нахилом сплайна*.

Найчастіше на практиці використовують кубічні сплайни, хоча окремі види сплайнів можуть бути більш ’’зручними’’ для деякого класу задач.

Аналітичний вигляд кубічного сплайна $S_3(x)$ в інтервалі $[x_j ; x_{j+1}]$, який у вузлах x_j та x_{j+1} дорівнює відповідно $f(x_j)$ і $f(x_{j+1})$ є таким:

$$S_3(x) = f(x_j)\frac{(x_{j+1} - x)^2(2(x - x_j) + h)}{h^3} + f(x_{j+1})\frac{(x - x_j)^2(2(x_{j+1} - x) + h)}{h^3} +$$
$$+ f'(x_j)\frac{(x_{j+1} - x)^2(x - x_j)}{h^2} + f'(x_{j+1})\frac{(x - x_j)^2(x - x_{j+1})}{h^2}.$$

Значення першої похідної сплайна $S_3(x)$ у вузлах x_j та x_{j+1} дорівнює відповідно $S_3'(x_j) = f'(x_j)$ і $S_3'(x_{j+1}) = f'(x_{j+1})$.

Контрольні питання

1. Що розуміють під "інтерполюванням" функції? Постановка задачі інтерполяції.
2. Лінійна та параболічна інтерполяція.
3. Інтерполяційний поліном Лагранжа та Ньютона.
4. Похибка інтерполяції.
5. Інтерполювання сплайнами.

Лекція № 6

АПРОКСИМАЦІЯ ФУНКЦІЙ

Предметом регресійного аналізу є математичне моделювання залежності результативної ознаки від факторних ознак з наступним статистичним аналізом моделі. Математична модель регресії – це аналітичний вираз для рівняння регресії. Вибір математичної моделі регресії визначається формою емпіричної лінії регресії та попередніми дослідженнями, професійною інтуїцією, що базується на знаннях фізичної сутності досліджуваного явища.

Нехай кореляційний зв'язок між досліджуваною ознакою та фактором X описується лінійним рівнянням регресії. Для кожного запланованого вимірювання на певному рівні фактора можна записати рівняння

$$Y_i = \beta_0 + \beta_1 x_i + E_i, \qquad (6.1)$$

де E_i – випадкова похибка i-го вимірювання.

На випадкові похибки накладаються такі умови: $E(E_i)=0$ – випадкова похибка є центрованою випадковою величиною; $D(E_i)=\sigma^2$ – дисперсія не залежить від значень факторної ознаки; послідовні значення незалежні (автокореляція відсутня).

Математичне сподівання запланованого вимірювання

$$E(Y_i) = E(Y \mid X = x_i) = \beta_0 + \beta_1 x_i .$$

Дисперсія запланованого вимірювання дорівнює

$$D(Y_i) = D(Y \mid X = x_i) = D(E_i) = \sigma^2 .$$

Дисперсія є однаковою на всіх рівнях факторної ознаки.

Лінійне рівняння регресії визначають за коефіцієнтами β_0 та β_1, які є невідомими. Оцінювання цих коефіцієнтів здійснюють на основі даних вибірки. Оцінку рівняння регресії представимо таким виразом:

$$\hat{y} = b_0 + b_1 x . \qquad (6.2)$$

Відхилення фактичного значення результативної ознаки від визначеного рівнянням називають *залишковим відхиленням*

$$e_i = y_i - \hat{y}_i = y_i - b_0 - b_1 x_s . \qquad (6.3)$$

Залишкові відхилення чисельно дорівнюють довжинам вертикальних відрізків, які з'єднують точки з лінією регресії. Величину

$$SS_e = \sum_{i=1}^{n} \left(y_i - b_0 - b_1 x_i \right)^2 \qquad (6.4)$$

називають *сумою квадратів залишкових відхилень.*

Величини b_0 та b_1 визначають методом найменших квадратів (МНК). Суть МНК полягає в тому, що обирають лінію з такими значеннями коефіцієнтів b_0 та b_1, щоб сума квадратів залишкових відхилень була мінімальною.

Лінію регресії $\hat{y} = b_0 + b_1 x$ називають ще *лінією найменших квадратів.* Умова мінімальності визначається системою рівнянь:

$$\begin{cases} \dfrac{\partial SS_e}{\partial b_0} = \sum\limits_{i=1}^{n} 2\left(y_i - b_0 - b_1 x_i\right)(-1) = 0, \\ \dfrac{\partial SS_e}{\partial b_1} = \sum\limits_{i=1}^{n} 2\left(y_i - b_0 - b_1 x_i\right)(-x_i) = 0. \end{cases} \tag{6.5}$$

Якщо розкрити дужки й спростити, то одержимо систему нормальних рівнянь для визначення параметрів b_0 та b_1:

$$\begin{cases} b_0 n + b_1 \sum\limits_{i=1}^{n} x_i = \sum\limits_{i=1}^{n} y_i, \\ b_0 \sum\limits_{i=1}^{n} x_i + b_1 \sum\limits_{i=1}^{n} x_i^2 = \sum\limits_{i=1}^{n} x_i y_i. \end{cases} \tag{6.6}$$

Розв'язок системи такий

$$\begin{aligned} b_1 &= \frac{\overline{xy} - \bar{x} \cdot \bar{y}}{\overline{x^2} - \bar{x}^2}; \\ b_0 &= \bar{y} - b_1 \bar{x}. \end{aligned} \tag{6.7}$$

Лінію найменших квадратів, можна записати у вигляді

$$y = \bar{y} + b_1\left(x - \bar{x}\right). \tag{6.8}$$

Доведено, що параметри b_0 та b_1 є оптимальними точковими оцінками коефіцієнтів регресії.

В рівнянні (6.2) коефіцієнт b_1 називають *коефіцієнтом регресії.* Він показує середню зміну залежностої величини при зміні на одиницю незалежної величини. Коефіцієнт регресії завжди число імовірне. Якщо $b_1>0$, то зв'язок між y та x прямий, якщо $b_1<0$, то зв'язок обернений, якщо $b_1=0$, то зв'язку немає, тобто залежність між y та x функціональна. Коефіцієнт b_0 – *початок відліку*, тобто є значенням y, коли $x=0$.

Контрольні питання

1. Що розуміють під "апроксимуванням" функції? Коли його застосовують?
2. Метод найменших квадратів. Загальна постановка.
3. Метод найменших квадратів з лінійним базисом.
4. Метод найменших квадратів з поліноміальним базисом. Вигляд основної системи рівнянь.
5. Від яких чинників залежить точність апроксимування заданої (таблично) функції?

Лекція № 7

ЧИСЕЛЬНЕ ІНТЕГРУВАННЯ

Дамо означення інтегралу. Нехай на відрізку $[a,b]$ задана функція $y=f(x)$. За допомогою точок $x_0, x_1, ..., x_n$ розіб'ємо відрізок $[a,b]$ на n елементарних відрізків $[x_{i-1}, x_i]$ $(i=1,2,...,n)$, причому $x_0=a$, $x_n=b$. На кожному з цих відрізків виберемо довільну точку ξ_i $(x_{i-1} \le \xi_i \le x_i)$ та знайдемо добуток s_i значення функції в цій точці $f(\xi_i)$ на довжину елементарного відрізка $\Delta x_i = x_i - x_{i-1}$:

$$s_i = f(\xi_i)\Delta x_i. \tag{7.1}$$

Складемо суму всіх таких добутків:

$$S_n = s_1 + s_2 + ... + s_n = \sum_{i=1}^{n} f(\xi_i)\Delta x_i. \tag{7.2}$$

Сума S_n називається інтегральною сумою. Означеним інтегралом від функції $f(x)$ на відрізку $[a,b]$ називається границя інтегральної суми при необмеженому збільшенні кількості точок розбиття; при цьому довжина найбільшого з елементарних відрізків прямує до нуля:

$$\int_a^b f(x)dx = \lim_{max\Delta x_i \to 0} \sum_{i=1}^{n} f(\xi_i)\Delta x_i. \tag{7.3}$$

В багатьох випадках, коли підінтегральна функція задана в аналітичному вигляді, означений інтеграл вдається визначити безпосередньо за допомогою невизначеного інтеграла (вірніше, первісної) за формулою Ньютона-Лейбніца. Вона полягає в тому, що означений інтеграл дорівнює приросту первісної $F(x)$ на відрізку інтегрування:

$$\int_a^b f(x)dx = F(x)\Big|_a^b = F(b) - F(a). \tag{7.4}$$

Але на практиці цією формулою часто не можна скористатися за двома основними причинами:

1. вигляд функції $f(x)$ не дозволяє безпосереднього інтегрування, тобто первісну неможливо виразити через елементарні функції.
2. значення функції $f(x)$ задані лише на фіксованій скінченій множині точок x_i, тобто функція задана у вигляді таблиці.

У цих випадках використовують методи чисельного інтегрування. Вони основані на апроксимації підінтегральної функції деяким більш простішим виразом, наприклад многочленами.

7.1 Метод прямокутників

Найпростішим методом чисельного інтегрування є метод прямокутників. В ньому здійснюється заміна означеного інтеграла інтегральною сумою (7.2). За

точки ξ_i можуть вибиратися ліві ($\xi_i = x_{i-1}$) або праві ($\xi_i = x_i$) границі елементарних відрізків. Позначивши $f(x_i) = y_i$, $\Delta x_i = h_i$, отримаємо наступні формули метода прямокутників відповідно до цих двох випадків:

$$\int_a^b f(x)dx = h_1 y_0 + h_2 y_1 + ... + h_n y_{n-1}, \tag{7.5}$$

$$\int_a^b f(x)dx = h_1 y_1 + h_2 y_2 + ... + h_n y_n. \tag{7.6}$$

Широко поширеним та найбільш точним є вигляд формули прямокутників, що використовує значення функції в середніх точках елементарних відрізків (метод середніх прямокутників):

$$\int_a^b f(x)dx = \sum_{i=1}^{n} h_i f\left(x_{i-\frac{1}{2}}\right), \tag{7.7}$$

$$x_{i-\frac{1}{2}} = \frac{x_{i-1} - x_i}{2} = x_{i-1} + \frac{h_i}{2}, \quad i = 1,2,...,n.$$

Важливим частинним випадком розглянутих формул є їх застосування при чисельному інтегруванні з постійним кроком $h_i = h = const$ $(i = 1,2,...,n)$. Формула середніх прямокутників в цьому випадку приймають відповідно вигляд

$$\int_a^b f(x)dx = h\sum_{i=1}^{n} f(x_{i-\frac{1}{2}}). \tag{7.8}$$

7.2 Метод трапецій

Метод трапецій використовує лінійну інтерполяцію, тобто графік функції $y = f(x)$ подається у вигляді ламаної, що з'єднує точки (x_i, y_i). В цьому випадку площа всієї фігури (криволінійної трапеції) складається з площ елементарних прямолінійних трапецій. Площа кожної такої трапеції рівна:

$$s_i = \frac{y_{i-1} - y_i}{2} h_i, \quad i = 1,2,...,n.$$

Додаючи всі ці рівності, отримуємо формулу трапецій для чисельного інтегрування:

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=1}^{n} h_i (y_{i-1} + y_i). \tag{7.9}$$

Для $h_i = h = const$ $(i = 1,2,...,n)$ формула (5.9) набуде вигляду:

$$\int_a^b f(x)dx = h\left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i\right). \qquad (7.10)$$

7.3 Метод парабол (метод Сімпсона)

Розіб'ємо відрізок інтегрування $[a,b]$ на парне число рівних частин з кроком h. На кожному відрізку $[x_0, x_2]$, $[x_2, x_4]$, ... , $[x_{i-1}, x_{i+1}]$, ... , $[x_{n-2}, x_n]$ підінтегральну функцію $f(x)$ замінимо інтерполяційним многочленом другого степеня:

$$f(x) \approx \phi_i(x) = a_i x^2 + b_i x + c_i,$$
$$x_{i-1} \le x \le x_{i+1}.$$

Коефіцієнти цих квадратних тричленів можуть бути знайдені з умов рівності многочлена в точках x_i відповідним табличним даним y_i. За $\phi_i(x)$ можна прийняти інтерполяційний многочлен Лагранжа другого степеня, що проходить через точки (x_{i-1}, y_{i-1}), (x_i, y_i), (x_{i+1}, y_{i+1}):

$$\phi_i(x) = \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} y_{i-1} + \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} y_i + \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} y_{i+1}.$$

Елементарна площа s_i може бути знайдена за допомогою означеного інтеграла. Враховуючи рівність $x_{i+1} - x_i = x_i - x_{i-1} = h$, отримаємо

$$s_i = \int_{x_{i-1}}^{x_{i+1}} \phi_i(x)dx = \frac{1}{2h^2} \int_{x_{i-1}}^{x_{i+1}} [(x-x_i)(x-x_{i+1})y_{i-1} - 2(x-x_{i-1})(x-x_{i+1})y_i + (x-x_{i-1})(x-x_i)y_{i+1}]dx =$$

$$= \frac{h}{3}(y_{i-1} + 4y_i + y_{i+1}).$$

Провівши такі розрахунки для кожного елементарного відрізку $[x_{i-1}, x_{i+1}]$, просумуємо отримані вирази:

$$S = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + ... + 2y_{n-2} + 4y_{n-1} + y_n).$$

Дані вирази для S приймається за значення означеного інтегралу:

$$\int_a^b f(x)dx \approx \frac{h}{3}[y_0 + 4(y_1 + y_3 + ... + y_{n-1}) + 2(y_2 + y_4 + ... + y_{n-2}) + y_n]. \qquad (7.11)$$

Отримане співвідношення називається формулою Сімпсона.

Контрольні питання

1. Суть методу прямокутників. Геометрична інтерпретація. Порядок похибки.
2. Суть методу трапецій. Геометрична інтерпретація. Порядок похибки.
3. Суть методу парабол. Геометрична інтерпретація. Порядок похибки.
4. Побудуйте блок-схеми методів прямокутників та трапецій.

Лекція № 8

ЧИСЕЛЬНЕ ДИФЕРЕНЦІЮВАННЯ

Задачі чисельного диференціювання виникають тоді, коли необхідно знайти похідну функції $f(x)$, яка задана на гратці $\{x_j : j=\overline{0,N}\}$ (наприклад, таблично). Тому функцію $f(x)$ на розглядуваному проміжку $[a;b]$ замінюють інтерполяційним поліномом $P_n(x)$ та вважають, що

$$f'(x)\approx P_n'(x), \quad x\in[a;b].$$

Аналогічно знаходять похідні вищих порядків

$$f^{(k)}(x)\approx P_n^{(k)}(x), \quad x\in[a;b].$$

Якщо для інтерполяційного многочлену (поліному) відомий залишковий член

$$R_n(x)=f(x)-P_n(x),$$

то можна знайти залишковий член $r_n(x)$ похідної $P_n^{(k)}(x)$:

$$r_n(x)=f^{(k)}(x)-P_n^{(k)}(x)=R_n^{(k)}(x).$$

Розглянемо випадок, коли задана функція $f(x)$ на гратці $\{x_j : \ j=\overline{-N,N}\}$, де $x_{j+1}-x_j$ є достатньо малою величиною (рисунок 8.1).

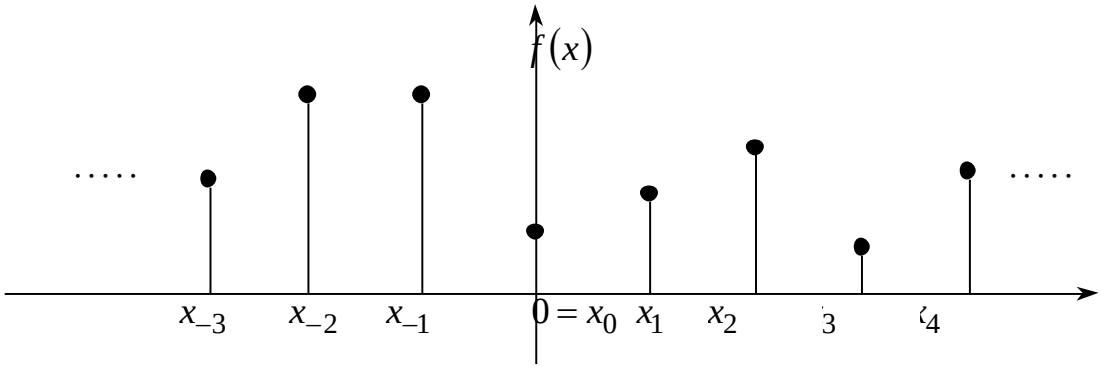


Рисунок 8.1

Запишемо ряд Маклорена для заданої функції $f(x)$:

$$f(x)=f(x_0)+\frac{f'(x_0)}{1!}x+\frac{f''(x_0)}{2!}x^2+\cdots+\frac{f^{(n)}(x_0)}{n!}x^n+\cdots. \qquad (8.1)$$

Якщо величина $x_{j+1}-x_j$, як зазначалося вище є достатньо малою, то тоді значення функції $f(x)$ в точці x_1 можна визначити з допомогою ряду (5.1.1)

$$f(x_1)=f(x_0)+\frac{f'(x_0)}{1!}x_1+\frac{f''(x_0)}{2!}x_1^{\,2}+\cdots+\frac{f^{(n)}(x_0)}{n!}x_1^{\,n}+\cdots. \qquad (8.2)$$

Величина x_1-x_0 дорівнює x_1. Очевидно якщо x_1 достатньо мала величина, то величини $x_1^{\,2}$, $x_1^{\,3}$, ..., x_1^n ще менші і таким чином доданками ряду (8.2) при $x_1^{\,2}$, $x_1^{\,3}$, ..., x_1^n можна знехтувати. Отже відкинувши в (8.2) всі доданки крім перших двох - отримаємо

$$f(x_1) \approx f(x_0) + \frac{f'(x_0)}{1!} x_1.$$

З останнього співвідношення виразимо, чому наближено рівне $f'(x_0)$, а саме

$$f'(x_0) \approx \frac{f(x_1) - f(x_0)}{x_1}. \tag{8.3}$$

Вираз (8.3) є правою розділеною різницею в точці $x = x_0$.

Для того, щоб одержати ліву розділену різницю для точки $x = x_0$ для початку запишемо чому наближено рівне значення функції $f(x)$ в точці $x = x_{-1}$ з допомогою ряду Маклорена (8.1):

$$f(x_{-1}) \approx f(x_0) + \frac{f'(x_0)}{1!} x_{-1} + \frac{f''(x_0)}{2!} x_{-1}^2 + \cdots + \frac{f^{(n)}(x_0)}{n!} x_{-1}^n + \cdots. \tag{8.4}$$

Поступивши аналогічним чином, як і в (8.2), отримаємо

$$f(x_{-1}) \approx f(x_0) + \frac{f'(x_0)}{1!} x_{-1}.$$

З останнього співвідношення випливає, що

$$f'(x_0) \approx \frac{f(x_{-1}) - f(x_0)}{x_{-1}}.$$

Число x_{-1} від'ємне. Знак мінус числа x_{-1} винесемо перед дробом, після чого внесемо у чисельник, таким чином отримаємо

$$f'(x_0) \approx \frac{f(x_0) - f(x_{-1})}{|x_{-1}|}. \tag{8.5}$$

Вираз (8.5) є лівою розділеною різницею в точці $x = x_0$.

Для того, щоб знайти центральну розділену різницю віднімемо від (8.2) вираз (8.4), отримаємо

$$f(x_1) - f(x_{-1}) \approx \frac{f'(x_0)}{1!} x_1 - \frac{f'(x_0)}{1!} x_{-1} +$$

$$+ \frac{f''(x_0)}{2!} x_1^2 - \frac{f''(x_0)}{2!} x_{-1}^2 + \cdots + \frac{f^{(n)}(x_0)}{n!} x_1^n - \frac{f^{(n)}(x_0)}{n!} x_{-1}^n + \cdots. \tag{8.6}$$

Нехтуючи доданками в (8.6) при x_1^2, x_{-1}^2, x_1^3, x_{-1}^3, ..., x_1^n, x_{-1}^n, ... отримаємо

$$f(x_1) - f(x_{-1}) \approx \frac{f'(x_0)}{1!} x_1 - \frac{f'(x_0)}{1!} x_{-1}.$$

З останнього виразу запишемо чому рівна $f'(x_0)$

$$f'(x_0) \approx \frac{f(x_1) - f(x_{-1})}{x_1 - x_{-1}}. \tag{8.7}$$

Вираз (8.7) називається центральною розділеною різницею в точці $x = x_0$.

Для того, щоб визначити вираз для розділеної різниці другого порядку, при умові, що відрізки x_j рівновіддалені в точці $x = x_0$, додамо вирази (8.2) та

(8.4) між собою. Внаслідок чого отримаємо

$$f(x_1)+f(x_{-1})\approx 2f(x_0)+\frac{f''(x_0)}{2!}x_1^2+\frac{f''(x_0)}{2!}x_{-1}^2+$$
$$\frac{f^{IV}(x_0)}{4!}x_1^4+\frac{f^{IV}(x_0)}{4!}x_{-1}^4+\cdots . \qquad (8.8)$$

Нехтуючи в (8.8) доданками при $x_1^4,\ x_{-1}^4,\ x_1^6,\ x_{-1}^6,\ x_1^8,\ x_{-1}^8, \ldots$, отримаємо

$$f(x_1)+f(x_{-1})\approx 2f(x_0)+\frac{f''(x_0)}{2!}x_1^2+\frac{f''(x_0)}{2!}x_{-1}^2. \qquad (8.9)$$

З (8.9) слідує

$$f''(x_0)\approx\frac{(f(x_{-1})-2f(x_0)+f(x_1))\cdot 2}{x_1^2+x_{-1}^2},$$

але, якщо відліки x_j рівновіддалені, то $x_1^2=x_{-1}^2$ і тоді останній вираз буде рівним

$$f''(x_0)\approx\frac{f(x_{-1})-2f(x_0)+f(x_1)}{x_1^2}. \qquad (8.10)$$

Вираз (8.10) називається розділеною різницею другого порядку в точці $x=x_0$.

Отже значення $\left.\frac{df(x)}{dx}\right|_{x=x_j}$ при рівновіддалених x_j можна наближено обчислювати по наступних виразах

$$\frac{f(x_{j+1})-f(x_j)}{h},\qquad \frac{f(x_j)-f(x_{j-1})}{h},\qquad \frac{f(x_{j+1})-f(x_{j-1})}{2h},$$

де $h=x_{j+1}-x_j$. А значення $\left.\frac{d^2f(x)}{dx^2}\right|_{x=x_j}$ по наближеному виразу

$$\frac{f(x_{j-1})-2f(x_j)+f(x_{j+1})}{h^2}.$$

Очевидно таким чином можна отримати наближені вирази для обчислення диференціальних операторів набла ∇ та Лапласа Δ для функції $f(x,y)$

$$\nabla f(x,y)\approx\frac{f(x_{j+1},y)-f(x_j,y)}{h}+\frac{f(x,y_{j+1})-f(x,y_j)}{d},$$
$$\nabla f(x,y)\approx\frac{f(x_j,y)-f(x_{j-1},y)}{h}+\frac{f(x,y_j)-f(x,y_{j-1})}{d},$$
$$\nabla f(x,y)\approx\frac{f(x_{j+1},y)-f(x_{j-1},y)}{2h}+\frac{f(x,y_{j+1})-f(x,y_{j-1})}{2d},$$
$$\Delta f(x,y)\approx\frac{f(x_{j-1},y)-2f(x_j,y)+f(x_{j+1},y)}{h^2}+\frac{f(x,y_{j-1})-2f(x,y_j)+f(x,y_{j+1})}{h^2},$$

де h і d кроки дискретизації по змінним відповідно x та y.

Контрольні запитання

1. Чисельне диференціювання. Перша різницева похідна "вперед".
2. Чисельне диференціювання. Перша різницева похідна "назад".
3. Чисельне диференціювання. Перша центральна різницева похідна.
4. Чисельне диференціювання. Друга різницева похідна.

Лекція № 8

ЧИСЕЛЬНЕ ІНТЕГРУВАННЯ ЗВИЧАЙНИХ ДИФЕРЕНЦІАЛЬНИХ РІВНЯНЬ ПЕРШОГО ПОРЯДКУ, РОЗВ'ЯЗАНИХ ВІДНОСНО ПОХІДНОЇ ОДНОКРОКОВИМИ МЕТОДАМИ

Часто інженерні задачі зводяться до відшукання розв'язку певного диференціального рівняння (або системи таких рівнянь), який задовольняє певні початкові умови (задача Коші). Проінтегрувати таке рівняння аналітичними методами вдається не завжди. При цьому дістають здебільшого такий вираз, до якого шукана функція входить неявно, а тому користуватися ним незручно.

На практиці застосовують здебільшого наближене інтегрування диференціальних рівнянь. Воно дає змогу знайти наближений розв'язок задачі Коші або у вигляді певного аналітичного виразу (наприклад, ряду Тейлора), або у вигляді деякої таблиці значень.

ЗАДАЧА КОШІ. Нехай задано диференціальне рівняння першого порядку з відокремленою похідною

$$y' = f(x, y). \tag{9.1}$$

Необхідно знайти наближений розв'язок даного диференціального рівняння, який задовольняє початковим умовам, тобто при $x = x_0$ значення наближеного розв'язку $y(x_0) = y_0$, де x_0 та y_0 задані числа.

Геометрично це означає, що треба знайти ту інтегральну криву $y(x)$ рівняння (9.1), яка проходить через точки (x_0, y_0).

9.1 Метод Ейлера

Найпростішим чисельним методом розв'язку задачі Коші для звичайного диференціального рівняння є метод Ейлера. Суть методу в тому, що на малому проміжку зміни аргументу

$$x_0 \le x \le x_0 + h = x_1$$

інтегральна крива диференціального рівняння

$$y' = f(x, y)$$

заміняється відрізком прямої (дотичної)

$$y - y_0 = f(x_0, y_0)(x - x_0).$$

Звідси $y_1 = y_0 + f(x_0, y_0)h$ та такий процес можна повторювати так далі, тобто

$$y_{k+1} = y_k + hf(x_k, y_k), \tag{9.2}$$

де $x_k = x_0 + kh$, а h – крок дискретизації аргументу x.

9.2 Удосконалений метод Ейлера

В удосконаленому методі Ейлера спочатку за методом Ейлера

обчислюють наближений розв'язок $y_{k+\frac{1}{2}}$ задачі (9.1) в точці $x_{k+\frac{1}{2}} = x_k + \frac{1}{2}h$:

$$y_{k+\frac{1}{2}} = y_k + \frac{1}{2}hf(x_k, y_k), \tag{9.3}$$

а потім за формулою

$$y_{k+1} = y_k + hf\left(x_{k+\frac{1}{2}}, y_{k+\frac{1}{2}}\right) \tag{9.4}$$

наближений розв'язок y_{k+1} у точці x_{k+1}; на кожному кроці інтегрування праву частину рівняння (9.1) обчислюють двічі (у точках (x_k, y_k) та $(x_{k+\frac{1}{2}}, y_{k+\frac{1}{2}})$).

9.3 Удосконалений метод Ейлера-Коші

В цьому методі на кожному кроці інтегрування праву частину рівняння (9.1) обчислюють двічі: спочатку за методом Ейлера обчислюється наближене значення шуканого розв'язку $\tilde{y}_{k+1}$ у точці x_{k+1}

$$\tilde{y}_{k+1} = y_k + hf(x_k, y_k), \tag{9.5}$$

яке потім уточнюється за формулою

$$y_{k+1} = y_k + \frac{h}{2}(f(x_k, y_k) + f(x_{k+1}, \tilde{y}_{k+1})). \tag{9.6}$$

9.4 Оцінка похибки наближеного розв'язку задачі Коші

Для оцінки похибки наближений розв'язок задачі Коші (9.1) у кожній вузловій точці обчислюють двічі: з кроком *h* та *h/2*. Позначають їх відповідно y_k та y_k. Десяткові розряди наближень y_k та y_k, які збігаються між собою, вважають точними цифрами наближеного розв'язку в точці x_k.

Оцінка похибки методу визначається за формулою

$$\varepsilon_k = y_k - y(x_k), \tag{9.7}$$

де $y(x_k)$ – точний розв'язок задачі у вузловій точці x_k.

Щоб вивести таку оцінку похибки, припустимо, що виконуються такі умови:

1. на кожному кроці інтегрування *h* похибка методу приблизно пропорційна $h^{s+1}(s \geq 1)$, де *s* – порядок точності методу;
2. похибка методу на кожному кроці інтегрування однакова;
3. на кожному наступному кроці інтегрування сумарна похибка методу включає також усі похибки, зроблені на попередніх кроках. Тому, якщо $y_1 - y(x_1) = Mh^{s+1}$, де *M* – невідомий коефіцієнт пропорційності, то

$$y_2 - y(x_2) = 2Mh^{s+1},$$
$$y_3 - y(x_3) = 3Mh^{s+1},$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$y_n - y(x_n) = nMh^{s+1}.$$

Отже, для похибки в точці x_k при інтегруванні з кроком h маємо рівність

$$y_k - y(x_k) = kMh^{s+1}, \tag{9.8}$$

а при інтегруванні з кроком $h/2$ – рівність:

$$y_k - y(x_k) = 2kM\left(\frac{h}{2}\right)^{s+1}. \tag{9.9}$$

Віднявши почленно (9.9) від рівності (9.8) та розв'язавши отриману рівність відносно невідомого коефіцієнта M, знайдемо

$$M = \frac{2^s(\tilde{y}_k - y_k)}{kh^{s+1}(2^s - 1)}.$$

Підставивши ці значення (9.9), маємо

$$y_k - y(x_k) = \frac{\tilde{y}_k - y_k}{2^s - 1}.$$

Звідси для абсолютної похибки в точці остаточно дістанемо таку рівність:

$$|\varepsilon_k| = |y_k - y(x_k)| = \frac{|\tilde{y}_k - y_k|}{2^s - 1}. \tag{9.10}$$

Контрольні запитання

1. Постановка задачі Коші.
2. Метод Ейлера (метод Ейлера, удосконалений метод Ейлера, удосконалений метод Ейлера-Коші). Геометрична інтерпретація методу.
3. Метод складання початку таблиці.
4. Метод Бубнова-Гальоркіна
5. Формула Адамса.

Лекція № 9

Загальна характеристика задач математичного програмування

В загальному задачу математичного програмування можна сформулювати так: необхідно знайти екстремум (максимум чи мінімум) функції:

$$Z = f(x_1, x_2,..., x_n) \qquad (10.1)$$

за умов

$$g_i(x_1, x_2,..., x_n) \quad R \quad b_i \quad (i = 1,2..., m) \qquad (10.2)$$

$$X = (x_1, x_2,..., x_n) \in Q\ , \qquad (10.3)$$

де Q – деяка множина n-вимірного евклідового простору R^n.

Умови (10.2) називають обмеженнями, а функцію (10.1) – цільовою функцією. При цьому функція $\{g_i\}$ та f, а також числа b_i ($i=1, m$) вважаються заданими.

Використати класичні методи пошуку умовного екстремуму функцій для розв'язання задачі (10.1) - (10.3) практично неможливо, бо, як правило, екстремум у такого класу задачах досягається на межі допустимої множини.

Математичне програмування поділяється на такі основні задачі: лінійне програмування, нелінійне програмування, стохастичне програмування та динамічне програмування.

Лінійним програмуванням називається розділ математичного програмування, що вивчає задачі типу (10.1) - (10.3), в яких функції $\{g_i,\ i=1,m\}$ та f є лінійними.

Підрозділ лінійного програмування, в якому визначається, щоб на всій або частину змінних величин $\{x_i,\ i=1,n\}$ було накладено допоміжну вимогу цілочисельності, називають лінійною цілочисловою програмування.

Якщо накладається вимога дискретності області Q, то такі задачі вивчаються в підрозділі математичного програмування, що носить назву дискретного програмування. Як бачимо, лінійна цілочислова програмування є частковим випадком дискретного програмування.

10.2 Транспортна задача

У пунктах постачання *A1,A2, ... ,Am* міститься однаковий товар, який треба перевезти в пункти споживання *B1,B2, ... ,Bn*. Відомо, скільки одиниць товару є в кожному пункті постачання, скільки одиниць товару потребує кожен пункт споживання, а також, вартість перевезення одиниці товару з кожного пункту постачання у кожен пункт споживання.

Припустимо, що виконується умова балансу, тобто загальна кількість одиниць товару, який є в пунктах постачання, збігається з загальною кількістю одиниць товару, що потребують пункти споживання. У такому випадку задачу можна сформулював так: треба так спланувати перевезення товару з пунктів постачання в пункти споживання, щоб весь товар з пунктів постачання був

вивезений, потреби всіх пунктів були задоволені та одночасно загальна вартість усіх перевезень була б мінімальною.

Складемо математичну модель задачі. Нехай:

a – кількість одиниць товару, що міститься в i-му пункті постачання A;

b – кількість одиниць товару, що потребує j-й пункт споживання B;

c – вартість перевезення одиниці товару з i-го пункту постачання A в j-й пункт споживання B;

x – кількість одиниць товару, що планується перевезти з i-го пункту постачання в j-й пункт споживання (шукані величини).

Об'єм товару, що планується перевезти з першого, другого, m-го пунктів постачання, задовольняють умови:

$$\begin{cases} x_{11} + x_{21} + ... + x_{1n} = a_1, \\ x_{12} + x_{22} + ... + x_{m2} = a_2, \\ ..... \\ x_{1n} + x_{2n} + ... + x_{mn} = a_m; \end{cases}$$

А для об'ємів товару, що планується ввезти у перший, другий, n-й пункти споживання повинні виконуватися умови:

$$\begin{cases} x_{11} + x_{21} + ... + x_{1n} = b_1, \\ x_{12} + x_{22} + ... + x_{m2} = b_2, \\ ..... \\ x_{1n} + x_{2n} + ... + x_{mn} = b_m; \end{cases}$$

Очевидно, що $x_{ij} \geq 0$ (i=1,2,…,m; j=1,2,…,n).

Нехай вартість перевезення x одиниць товару становить $c_{ij}x_{ij}$. Тоді загальна вартість усіх перевезень обчислюється за формулою:

$$L = c_{11}x_{11} + c_{12}x_{12} + ... + c_{1n}x_{1n} + c_{21}x_{21} + c_{22}x_{22} + ... + c_{2n}x_{2n} + ... + c_{m1}x_{m1} + ... + c_{mn} + c_{mn}$$

Отже, математична модель задачі є такою:

Знайти мінімум лінійної форми

$$L = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

за умов

$$\sum_{j=1}^{n} x_{ij} = a_i \quad (i = 1,2,,,,m)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad (j = 1,2...,n)$$

$$x_{ij} \geq 0 \quad (i = 1,2...,m;\ j = 1,2,...,n)$$

10.3 Постановка задачі лінійного програмування

Математично в загальному вигляді задачі лінійного програмування формують так.

Задано систему лінійних рівнянь

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + a_{1n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + a_{mn}x_n = b_m \end{cases} \tag{10.4}$$

та лінійну функцію (цільову функцію)

$$Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n. \tag{10.5}$$

Треба знайти такий невід'ємний розв'язок системи (10.4), при якому лінійна функція (10.5) набуває найбільшого (найменшого) значення.

Система (10.4) може мати один невід'ємний розв'язок, не мати жодного невід'ємного розв'язку, або мати нескінченну множину невід'ємних розв'язків. Для останнього випадку задача лінійного програмування полягає в тому, щоб з цієї множини знайти той розв'язок, при якому цільова функція набуває максимуму (мінімуму).

Загальна постановка задачі лінійного програмування:

Необхідно знайти такі значення $x_1^0, x_2^0, \ldots, x_n^0$ змінних $x_1, x_2, \ldots x_n$, які задовольняють системі співвідношень:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n \; R_1 \; a_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n \; R_2 \; a_2. \\ \ldots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n \; R_m \; a_m \end{cases} \tag{10.6}$$

та при цьому надають найбільшого чи найменшого значення функції :

$$f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{10.7}$$

де $R_i \left(i = \overline{1,m}\right)$ – один із знаків $=, \leq, \geq$;

$a_i, a_{ij}, c_i \left(i = \overline{1,m};\ j = \overline{1,n}\right)$ – задані дійсні числа.

Співвідношення (10.6) називають обмеженням даної задачі лінійного програмування. Функцію (10.7) – цільовою (функцією цілі) функцією даної задачі.

Будь-який впорядкований набір $\left(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}\right)$ значень змінних, що задовольняє (10.6) називається *допустимим рішенням задачі* або планом задачі. Множина всіх допустимих рішень називається *допустимою множиною задачі*. Допустимий розв'язок $\left(x_1^0, x_2^0, \ldots, x_n^0\right)$, що надає цільовій функції максимальне (чи мінімальне) значення, називається *оптимальним розв'язком*, оптимальним планом або просто розв'язком задачі лінійного програмування.

Значення цільової функції (10.7) при будь-якому із оптимальних розв'язків задачі називається оптимумом цієї задачі.

Задачу лінійного програмування, яка має допустимі розв'язки (тобто, система обмежень якої є сумісною) будемо називати допустимою, а задачу із несумісною системою обмежень – недопустимою.

В більшості задач лінійного програмування серед обмежень (10.6) є обмеження типу:

$$x_j \geq 0 \quad \text{, або } x_j \leq 0, \text{ або } b_j \leq x \leq d_j$$

для всіх, або декількох j. Такі обмеження називають прямими обмеженнями на змінні.

Задачу лінійного програмування типу:

$$\begin{cases} c_1x_1 + c_2x_2 + ... + c_nx_n \to \max(\min) \\ a_{11} \cdot x_1 + a_{12} \cdot x_2 + ... + a_{1n} \cdot x_n = a_1; \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + ... + a_{2n} \cdot x_n = a_{2.}; \\ ... \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + ... + a_{mn} \cdot x_n = a_m; \\ x_j \geq 0 \left(j = \overline{1,n}\right) \end{cases} \tag{10.8}$$

називають канонічною задачею лінійного програмування.

(10.8) – можна подати й в матричній формі. Якщо ввести

$$C = (c_1, c_2, ..., c_n); \quad A = \begin{pmatrix} a_{11}\, a_{12} \ldots a_{1n} \\ a_{21}\, a_{22} \ldots a_{2n} \\ \ldots \\ a_{m1}\, a_{m2} \ldots a_{mn} \end{pmatrix}; \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}; \quad P_0 = \begin{pmatrix} a_1 \\ a_2 \\ \ldots \\ a_m \end{pmatrix};$$

то канонічну задачу лінійного програмування можна записати у матричній формі:

$$\begin{cases} C \cdot X \to \max(\min) \\ A \cdot X = P_0 \\ X \geq 0 \end{cases} \tag{10.9}$$

Також використовують векторну форму запису канонічної задачі лінійного програмування. Введемо для цього такі позначення стовпців матриці *A*:

$$P_1 = \begin{pmatrix} a_{11} \\ a_{21} \\ \ldots \\ a_{m1} \end{pmatrix}, \; P_2 = \begin{pmatrix} a_{12} \\ a_{22} \\ \ldots \\ a_{m2} \end{pmatrix}, \quad \ldots \quad , \; P_n = \begin{pmatrix} a_{1n} \\ a_{2n} \\ \ldots \\ a_{mn} \end{pmatrix};$$

тоді задача (10.9) буде мати вигляд:

$$\begin{gathered} (C, X) = c_1x_1 + c_2x_2 + ... + c_nx_n \to \max(\min) \\ P_1x_1 + P_2x_2 + ... + P_nx_n = P_0 \\ x_1 \geq 0, x_2 \geq 0, ..., x_n \geq 0\,. \end{gathered} \tag{10.10}$$

Якщо при постановці канонічної задачі використовуються двосторонні обмеження на можливі значення змінних $X = (x_1, ..., x_n)$, тобто:

$$b_j \leq x_j \leq d_j, \; j = 1, 2, ..., n,$$

то таку задачу називають канонічною задачею лінійного програмування з двосторонніми обмеженнями на змінні.

Будь-які інші задачі лінійного програмування необхідно розглядати як різні форми запису загальної задачі лінійного програмування. Кожну із задач

можна звести до будь-якої іншої форми.

Контрольні питання

1. Що таке математичне програмування? На які розділи воно поділяється?
2. Наведіть загальну постановку задачі лінійного програмування.
3. Сформулюйте транспортну задачу.
4. Дайте визначення таким поняттям як: цільова функція, допустиме рішення задачі, допустима множина задачі, оптимальний розв’язок задачі лінійного програмування.
5. Яким чином записується канонічна задача лінійного програмування?
6. Запишіть канонічну задачу лінійного програмування у матричній та векторній формі.

Лекція № 11

РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ ГРАФІЧНИМ МЕТОДОМ

11.1 Опуклі множини

Точка $X' = \left(x_1', x_2', ..., x_n'\right) \in M$ називається внутрішньою точкою множини M, якщо вона належить множині M разом з деяким малим околом.

Точка $X^0 = \left(x_1^0, ..., x_n^0\right) \in M$ називається точкою згущення множини M, якщо в кожному її околі міститься хоча б одна точка множини M, відмінна від X^0. Множина M, яка містить всі свої точки згущення, називається замкнутою.

Опуклою комбінацією точок $X_1, X_2, ..., X_k$ простору R^n називається точка:

$$x = a_1 x_1 + a_2 x_2 + ... + a_k x_k, \tag{11.1}$$

де $a_i \geq 0, i = \overline{1,k}$, $\sum_{i=1}^{k} a_i = 1$.

Множина $C \subset R^n$ називається *опуклою*, якщо для будь-яких двох точок X_1 та X_2, що належить до C, до C належить й будь-яка опукла комбінація цих точок, тобто:

$$X = aX_1 + (1-a)X_2, \; X \in C, \; 0 \leq a \leq 1 \tag{11.2}$$

Геометрично це означає, що опукла множина разом із своїми будь-якими двома точками завжди містить усі точки прямолінійного відрізку, який їх з'єднує.

Приклади опуклих множин: прямолінійний відрізок, пряма, круг, куб, куля. Множина точок кола або кільця – не опуклі множини.

Запишемо задачу лінійного програмування у векторній формі:

$$\begin{gathered} L = cX \to \min \\ x_1 P_1 + x_2 P_2 + ... + x_n P_n = P_0 \\ X \geq 0 \end{gathered} \tag{11.3}$$

Дамо наступні означення.

Означення 1. Розв'язок $X = \left(x_1, x_2, ..., x_n\right)$ називається *опорним*, якщо вектори P_j, які відповідають додатнім компонентам x_j розв'язку X, утворюють лінійно незалежну систему.

Означення 2. Опорний розв'язок називається *невиродженим*, якщо він містить рівно m додатних компонентів. Якщо опорний розв'язок містить менше за m додатних компонентів, то такий опорний розв'язок називається *виродженим*.

Властивості розв'язків задач лінійного програмування. Запишемо задачу у вигляді:

$$\begin{cases} L(X) = cX \to \min \\ AX = B \\ X \geq 0 \end{cases} \quad (11.4)$$

Теорема 1. Допустима множина задачі (11.4) опукла.

Теорема 2. Цільова функція задачі лінійного програмування (11.4) досягає мінімального значення у крайній точці опуклої множини *M* задачі. Якщо цільова функція набуває мінімального значення більш ніж в одній крайній точці, то вона набуває цього ж значення в будь-якій точці, яка є опуклою комбінацією цих точок.

Згідно з Теоремою 2, пошук оптимального плану задачі лінійного програмування можна обмежити перебором крайніх точок опуклої множини допустимих розв'язків задачі.

Теорема 3. (Критерій крайності точок опуклої множини розв'язків). Для того, щоб точка $X = (x_1, x_2, ..., x_n)$ опуклої множини планів задачі лінійного програмування була крайньою, необхідно й достатньо, щоб вектори P_j, які відповідають додатнім компонентам x_j, утворювали лінійно незалежну систему.

Наслідок теореми. Кожній крайній точці множини планів *M* задачі лінійного програмування відповідає *m* лінійно незалежних векторів із системи $P_1, P_2, ..., P_n$.

Означення. Точка *X* опуклої множини *C* називається *крайньою*, якщо не існує таких двох точок множини *C,* відмінних від *X*, що точка *X* є їх опуклою комбінацією.

Наприклад, крайні точки прямолінійного відрізка – це точки кінців відрізка, крайні точки круга – це точки кола, що його обмежує.

Із теореми 3 та означення опорного плану випливає, що кожний опорний план задачі лінійного програмування є крайньою точкою множини планів задачі.

11.2 Графічний метод розв'язання задачі лінійного програмування

Якщо кількість змінних цільової функції не перевищує трьох, то можна дати геометричну інтерпретацію задачі лінійного програмування.

Нехай необхідно знайти мінімум або максимум лінійної функції:

$$L = c_1 x_1 + c_2 x_2 \quad (11.5)$$

за умов:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 \leq b_1, \\ a_{21}x_1 + a_{22}x_2 \leq b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 \leq b_m; \end{cases} \quad (11.6)$$

$$x_1 \geq 0, \quad x_2 \geq 0. \quad (11.7)$$

Введемо на площині декартову прямокутну систему координат та

співставимо кожній парі чисел (x_1, x_2) точку площини з координатами. Обмежуючі співвідношення (11.5) - (11.7) визначають на площині півплощини, що породжуються внаслідок розбиття площини лініями типу $a_{i1}x_1 + a_{i2}x_2 = b_i, \quad i = \overline{i,m}.$

Щоб знайти розв'язок задачі лінійного програмування, треба спочатку знайти множину точок на площині, що задовольняють умови (11.6) та (11.7). Для цього в площині x_1Ox_2 проведемо прямі:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1, \\ a_{21}x_1 + a_{22}x_2 = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 = b_m; \end{cases} \qquad (11.8)$$

Кожна пряма поділяє площину на дві півплощини. Щоб дізнатися, точки якої півплощини задовольняють нерівність $a_{i1}x_1 + a_{i2}x_2 \le b_i$, необхідно в цю нерівність підставити координати точки *O*, тобто початку координат ($x_1=0$, $x_2=0$). Якщо координати цієї точки задовольняють нерівність, то нерівність будуть задовольняти всі точки півплощини, яка містить початок координат. Якщо координати цієї точки не задовольняють нерівність, то нерівність будуть задовольняти всі точки півплощини, яка не містить початку координат.

Нехай першу нерівність системи (11.6) задовольняють точки півплощини P_1, другу-точки півплощини P_2 й т. д., останню – точки півплощини P_m. Перетином півплощин $P_1, P_2, \ldots, P_m$ є деяка опукла многокутна область К, множина точок якої є множиною розв'язків системи (11.6). Нехай К не є порожньою множиною. Множина точок, які задовільняють умові (11.7) утворює першу четверть площини x_1Ox_2. Тому множиною точок, які одночасно задовільняють умовам (11.6) та (11.7), є та частина області К, яка лежить у першій чверті площини x_1Ox_2. Позначимо цю площину буквою М. В загальному випадку М може бути або порожньою множиною, або опуклим многокутником, або необмеженою опуклою многокутною областю. В першому випадку умови задачі є суперечливими, задача не має розв'язку; в другому-завжди існують точки, в яких функція набуває мінімального та максимального значень; у третьому-лінійна функція на множині М, як правило є необмеженою або знизу, або зверху та знизу, тому ця функція може не досягати або максимуму, або мінімуму, або максимуму та мінімуму.

Геометричне місце точок, в яких лінійна функція (11.5) набуває фіксованого значення *c*, є прямою лінією, перпендикулярною до вектора $N=(c_1,c_2)$, та визначається рівнянням: $c_2x_1 + c_2x_2 + c$

Надавши параметру с всіх можливих значень від $-\infty$ до ∞, одержимо сімейство паралельних прямих, які займуть всю площину. Перехід віл однієї прямої до іншої у напрямку вектора N веде до збільшення значення лінійної функції, у протилежному напрямку – до його зменшення.

Відкладемо в площині x_1Ox_2 вектор N з координатами $x_1=c_1$, $x_2=c_2$ та перетнемо область прямою l, перпендикулярно до вектора *N* (рисунок 11.1).

Щоб знайти точку області М, в якій лінійна функція досягає максимуму,

зміщуємо пряму l паралельно самій собі у напрямку вектора N доти, доки пряма l не стане опорною до M (пряма називається опорною до опуклої області, якщо область лежить по один бік прямої та пряма має з нею хоча б одну спільну точку). Спільна точка l та M буде тією точкою, в якій досягається максимум лінійної функції. Для знаходження точки області M, де лінійна функція (11.5) досягає мінімуму, зміщуємо пряму l паралельно самій собі у протилежному напрямку доти, доки пряма l не стане опорною до M. Спільна точка l та M буде тією точкою, в якій досягається мінімум лінійної функції. Так на рисунку 11.1 лінійна функція досягає максимуму в точці A, а мінімуму – в точці B.

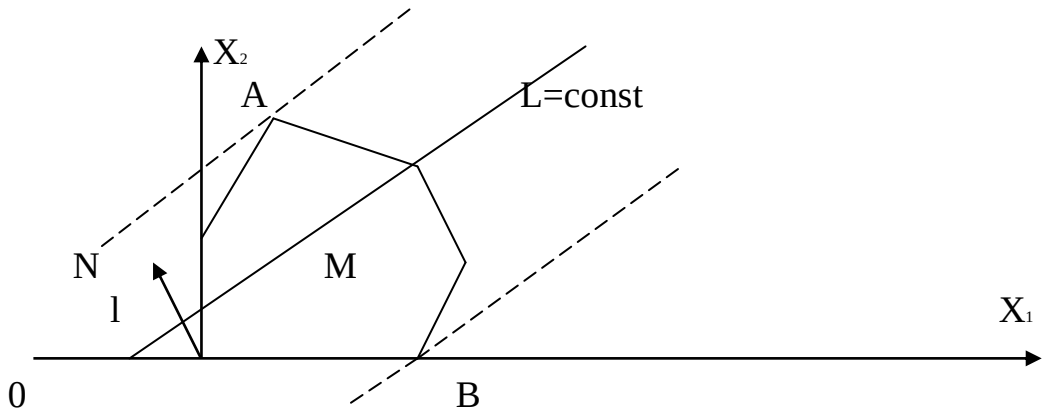


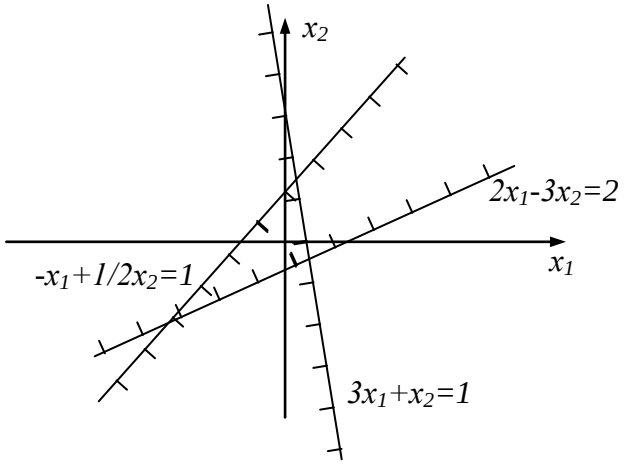
Рисунок 11.1 – Розв’язання задачі лінійного програмування графічним методом

Якщо M – опуклий многокутник, то оптимальне значення лінійної функції завжди досягається в його вершині. Якщо оптимальне значення досягається в двох вершинах, то L набуває його в будь-якій точці, що лежить на відрізку, що з’єднує їх.

Якщо область M необмежена, то мінімум чи максимум лінійної функції може не досягатися, тобто при зміщенні прямої l у тому чи іншому напрямку весь час будуть траплятися точки області M.

Наприклад. Необхідно знайти допустимі області задачі лінійного програмування з обмеженнями:

$$\begin{cases} 2x_1 - 3x_2 \le 2 \\ 3x_1 + x_2 \le 1 \\ -x_1 + \dfrac{1}{2}x_2 \le 1 \end{cases}$$



Контрольні питання

1. Зробіть постановку задачі лінійного програмування в канонічній формі.
2. Дайте визначення опуклої множини.
3. Наведіть теореми про опуклі множини.
4. Який розв’язок задачі лінійного програмування називається опорним, а який оптимальним?
5. Допустима область розв’язків задачі лінійного програмування (графічна інтерпретація).
6. Суть графічного методу розв’язку задачі лінійного програмування.

Лекція № 12

ПОШУК ПОЧАТКОВОГО ОПОРНОГО ПЛАНУ

12.1 Зведення задач лінійного програмування до канонічної форми

При розгляді симплексного методу ми вважали, що задача лінійного програмування задача в канонічній формі. Покажемо, як знайти розв'язок задачі лінійного програмування, якщо наприклад, обмеження задачі задана у вигляді системи лінійних нерівностей або якщо треба знайти максимум лінійної форми.

1. Нехай задача лінійного програмування матиме вигляд

$$L = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \to \min$$

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n \le b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n \le b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n \le b_m \end{cases} \qquad (12.1)$$

$$x_j \ge 0,\ j = 1,2,\ldots,n \quad b_i \ge 0 (i = 1,2,\cdots,m)$$

Щоб знайти розв'язок задачі (12.1) треба розв'язати таку задачу

$$L = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \to \min$$

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n + x_{n+1} = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n + x_{n+2} = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n + x_{n+m} = b_m \end{cases} \qquad (12.2)$$

$$x_j \ge 0,\ j = 1,2,\ldots,n+m$$

Задача (12.2) є зведенням до канонічної форми задачі (12.1).

Якщо $\overline{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \bar{x}_{n+1}, \ldots, \bar{x}_{m+n})$ є оптимальним планом задачі (12.2) то вектор $x = (\bar{x}_1, \bar{x}_{2,} \cdots, \bar{x}_{n,})$ буде оптимальним планом задачі (12.1). Зауважимо, що початковий опорний план задачі (12.2) очевидний

$$\bar{x}_1 = (0,0,\ldots,0,b_1,b_2,\ldots,b_m)$$

Базис який йому відповідає, складається з одиничних векторів $P_{n+1}, P_{n+2}, \ldots P_{n+m}$.

2. Припустимо , що задача лінійного програмування має вигляд:

$$L = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \to \min$$
$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n \geq b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n \geq b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n \geq b_m \end{cases} \qquad (12.3)$$
$$x_j \geq 0\,,\ j = 1,2,\ldots,n$$

Щоб знайти розв'язок задачі (12.3), треба розв'язати таку задачу:

$$L = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \to \min$$
$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n - x_{n+1} = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \ldots + a_{2n} \cdot x_n - x_{n+2} = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n - x_{n+m} = b_m \end{cases} \qquad (12.4)$$
$$x_j \geq 0\,,\ j = 1,2,\ldots,n+m$$

Задача (12.4) є зведенням до канонічної форми задачі (12.3). Якщо $\overline{x} = \left(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \bar{x}_{n+1}, \ldots, \bar{x}_{m+n}\right)$ є оптимальним планом задачі (12.4) то вектор $x = \left(\bar{x}_1, \bar{x}_{2,} \cdots, \bar{x}_{n,}\right)$ буде оптимальним планом задачі (12.3). Початковий опорний план задачі (12.4), на відміну від задачі (12.2), не є очевидним. Тому для розв'язання цієї задачі треба спочатку знайти її початковий опорний план.

Покладемо, що для того, щоб знайти початковий опорний план задачі (12.4), досить розв'язати допоміжну задачу з однією штучною змінною.

Нехай

$$\max b_i = b_s$$
$$1 \leq i \leq m$$

Віднімемо кожне і-те ($i \neq s$) рівняння системи рівнянь задачі (12.4)

$$\sum_{j=1}^{n} a_{i \cdot j} \cdot x_j - x_{n+i} = b_i\,.$$

від *s*-го рівняння цієї системи

$$\sum_{j=1}^{n} a_{s \cdot j} \cdot x_j - x_{n+s} = b_s\,,$$

Та одержимо систему рівнянь

$$\sum_{j=1}^{n} (a_{s \cdot j} - a_{s \cdot j}) \cdot x_j - x_{n+s} + x_{n+i} = b_s - b_i\,.$$

Тепер від задачі (12.4) перейдемо до такої еквівалентної задачі:

$$L = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n \to \min\,. \qquad (12.5)$$

В системі рівнянь якої коефіцієнти при невідомих x_{n+i} ($L \neq s$) дорівнюють одиниці, а коефіцієнт при невідомій x_{n+s} становить -1. У задачі (12.5) серед векторів $P_{n+1}, P_{n+2}, \ldots P_{n+m}$ є *m*-1 одиничних векторів $P_{n+1}, P_{n+2}, \ldots P_{n+s-1,} P_{n+s+1,} \ldots$ Тому, щоб знайти початковий опорний план цієї

задачі, досить розв'язати таку допоміжну задачу з однією штучною змінною:

$$\overline{L} = x_{n+m+1} \to \min$$

$$\begin{cases} \sum_{j=1}^{n} \left(a_{si} - a_{ij}\right) \cdot x_j - x_{n+s} + x_{n+i} = b_s - b_i \quad (i \neq s) \\ \sum_{j=1}^{n} a_{sj} \cdot x_j - x_{n+s} + x_{n+m+1} = b_s \end{cases}$$

$$x_j \geq 0 \quad j = 1,2,\cdots,n+m+1$$

3. Якщо в задачі лінійного програмування треба знаходити не мінімум лінійної форми, а максимум, то не обов'язково для розв'язування задачі треба переходити до задачі на мінімум. Процес розв'язування задачі симплексним методом на максимум буде таким, як і процес розв'язування задачі на мінімум за винятком критеріїв: вибору вектора, що треба ввести в новий базис; оптимальності опорного плану; необмеженості лінійної форми на множині планів. У задачах на максимум у новий базис вводять той вектор P_j, який відповідає мінімальній різниці Δj та опорний план задачі буде оптимальним, якщо для всіх j виконується умова $\Delta j \geq 0$; лінійна форма на множині планів є необмеженою, якщо $\Delta j \prec 0$ та всі $x_{ij} \leq 0$.

12.2 Знаходження початкового опорного плану

Щоб застосувати симплекс-метод для знаходження оптимального розв'язку задачі лінійного програмування, треба знайти відправну точку – початковий опорний план. Якщо обмеження задачі лінійного програмування задано в канонічному вигляді

$$AX = A_0,\ A_0 \geq 0,\ X \geq 0 \qquad (12.6)$$

та серед векторів $A_1, A_2, \ldots, A_n$ є одиничний базис, то початковим опорним планом буде вектор $X = (0,0,\ldots,0,b_1,b_2,\ldots,b_m)$. У деяких випадках одиничний базис у системі обмежень (12.6) легко виділити. Наприклад, нехай маємо систему обмежень

$$\begin{cases} 3x_1 - x_4 - 2x_6 = 6, \\ 2x_2 + x_4 + 3x_5 + x_6 = 8 \\ x_3 - 2x_4 + 5x_5 + 6x_6 = 5 \\ x_j \geq 0, \quad (j = 1,2,...,6). \end{cases}$$

Якщо перше рівняння системи поділити на 3, а друге – на 2, то дістанемо систему

$$\begin{cases} x_1 - \frac{1}{3}x_4 - \frac{2}{3}x_6 = 2, \\ x_2 + \frac{1}{2}x_4 + \frac{3}{2}x_5 + \frac{1}{2}x_6 = 4 \\ x_3 - 2x_4 + 5x_5 + 6x_6 = 5 \\ x_j \geq 0, \quad (j = 1,2,...,6). \end{cases}$$

У ній вектори A_1, A_2, A_3 утворюють одиничний базис та всі вільні члени додатні. Поклавши $x_4 = x_5 = x_6$, знайдемо опорний план $X = (2,4,5,0,0,0)$.

Якщо система (12.6) не містить у явному вигляді одиничного базису, то його у деяких випадках можна виділити методом повного виключення Гауса.

Якщо обмеження задачі лінійного програмування задано у вигляді нерівностей

$$A_1x_1 + A_2x_2 + \ldots + A_nx_n \leq A_0,\ A_0 \geq 0,\ X \geq 0 \qquad (12.7)$$

то, додавши до лівої частини кожної нерівності системи (12.7) по невід'ємній змінній $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$, дістанемо розширену систему лінійних обмежень

$$A_1x_1 + \ldots + A_nx_n + A_{n+1}x_{n+1} + \ldots + A_{n+m}x_{n+m} = A_0,\ A_0 \geq 0,\ X \geq 0 \qquad (12.8)$$

Змінні $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ називають *додатковими змінними*. В лінійну функцію вони входять з нульовими коефіцієнтами. Отже, початкова задача лінійного програмування перетворилася у розширену: знайти оптимальне значення лінійної функції

$$L = c_1 \cdot x_1 + c_2 \cdot x_2 + \ldots + c_n \cdot x_n + 0 \cdot x_{n+1} + \ldots + 0 \cdot x_{n+m}$$

за обмежень (12.8). Додаткові вектори $A_{n+1}, A_{n+2}, \ldots, A_{n+m}$ утворюють одиничний базис m-вимірного векторного простору, а вектор $X = (x_1 = 0, \ldots, x_m = 0, x_{m+1} = b_1, \ldots, x_{n+m} = b_{n+m})$ є опорним планом розширеної задачі.

Зазначимо, що розв'язок розширеної задачі (якщо він існує) буде також розв'язком початкової задачі. Якщо розширена задача розв'язку не має, то не має розв'язку й початкова задача.

Застосовуючи симплексний метод до розширеної задачі, поступово замінюють у системі базисних векторів додаткові вектори $A_{n+1}, A_{n+2}, \ldots, A_{n+m}$ векторами початкової системи обмежень. Якщо лінійна функція досягла свого оптимального значення, а в системі базисних векторів є хоча б один додатковий вектор, наприклад A_{n+i}, то це означає, що i-те обмеження в початковій задачі має смисл строгої нерівності. Отже, початкова задача матиме оптимальний розв'язок (якщо його має розширена задача) й тоді, коли не всі додаткові вектори виведено з базису.

Часто, виділивши з системи обмежень одиничний базис та базисний розв'язок, який йому відповідає, можна впевнитися, що знайдений розв'язок не є опорним планом системи обмежень, бо серед базисних змінних є й від'ємні. З метою цілеспрямованого пошуку опорного плану треба від знайденого одиничного базису перейти до іншого. Для цього застосовують метод *симплексного перетворення*. Оскільки симплексні перетворення виконують над векторами $A_1, A_2, \ldots, A_n$, A_0 системи обмежень, то ці вектори зведемо в

таблицю 12.1.

Нехай вектор $A_0 \geq 0$, тобто всі $b_i \geq 0$ $(i = 1,2,\ldots,m)$. Цього можна досягти, помноживши рівняння з від'ємними членами на -1. Тепер задача полягає в тому, щоб у таблиці 12.1 виділити одиничний базис, не порушивши невід'ємність компонент вектора A_0. Перетворюють таблицю за таким алгоритмом:

1. З неодиничних векторів $A_1, A_2, \ldots, A_n$ взяти той, у якого є хоча б один додатний елемент. Нехай таким вектором буде A_j. Якщо такого вектора в таблиці немає, то це означає, що система обмежень несумісна та процес симплексного перетворення завершено.

2. Знайти відношення θ_i елементів вектора A_0 до відповідних додатних елементів вектора A_j, записати їх у відповідному рядку стовпця θ та взяти з цих відношень найменше. Нехай таким буде відношення $\theta = \frac{b_i}{a_{ij}}$. Тоді елемент a_{ij} - ведучий, а рядок та стовпець таблиці, на перетині яких лежить a_{ij}, відповідно ведучі рядок та стовпець.

3. Коефіцієнти ведучого рядка (крім θ_i) таблиці поділити на ведучий елемент та записати у відповідному рядку нової таблиці.

4. Елементи кожного наступного рядка нової таблиці дістають шляхом додавання відповідного рядка початкової таблиці та рядка, записаного в п.3, помноженого на таке число, щоб у ведучому стовпці при додаванні дістати нулі. На цьому заповнення нової таблиці завершується та перетворення нової таблиці починають з п.1.

Таблиця 12.1 – Таблиця симплексного перетворення

№ рядка	A_0	A_1	A_2	…	A_j	…	A_n	$\sum$	θ
1	b_1	a_{11}	a_{12}	…	a_{1j}	…	a_{1n}		
2	b_2	a_{21}	a_{22}	…	a_{2j}	…	a_{2n}		
…	…	…	…	…	…	…	…	…	…
i	b_i	a_{i1}	a_{i2}	…	a_{ij}	…	a_{in}		
…	…	…	…	…	…	…	…	…	…
m	b_m	a_{m1}	a_{m2}	…	a_{mj}	…	a_{mn}		

Такі перетворення продовжують доти, поки в таблиці не буде виділено одиничний базис, якому відповідає опорний план, або ж не буде встановлено, що система обмежень несумісна. Стовпець $\sum$ в таблиці призначений для контролю обчислень, його елементи обчислюють так, як і за методом повного виключення Гауса.

Зазначимо, що при виборі вектора A_j треба стежити за тим, щоб не повернутися до попереднього базису. Ця умова додатково обмежує вибір

ведучого елемента.

Контрольні питання

1. Сформулюйте загальну постановку задачі математичного програмування.
2. Класифікація задач математичного програмування.
3. Наведіть приклади задач математичного програмування.
4. Дайте постановку задачі лінійного програмування в канонічній формі.
5. Наведіть принципи зведення довільної задачі лінійного програмування до канонічної.
6. Множина допустимих розв’язків та оптимальний розв’язок задачі лінійного програмування.
7. Методи пошуку початкового опорного плану задачі лінійного програмування.

Лекція № 13

СИМПЛЕКСНИЙ МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

13.1 Теоретичні основи симплексного методу

Суть симплексного методу (або методу послідовного поліпшення розв'язку задачі лінійного програмування) полягає в послідовному переході від одного опорного розв'язку до іншого так, що значення цільової функції весь час зменшується.

Сформулюємо теореми, які лежать в основі симплексного методу.

Нехай задано задачу лінійного програмування, що записана в канонічній формі. Нехай X - деякий невироджений опорний розв'язок задачі. Не зменшуючи загальності, будемо вважати, що $X = (x_1, x_2, ..., x_m, 0, ..., 0)$ де $x_i > 0 (i = 1,2,..., m)$. Тоді

$$x_1 \cdot P_1 + x_2 \cdot P_2 + \ldots + x_m \cdot P_m = P_0 .$$

Значення цільової функції для розв'язку X буде дорівнювати:

$$Z_0 = c_1 x_1 + c_2 x_2 + \ldots + c_m x_m .$$

Опорному розв'язку X відповідає система лінійно незалежних векторів $P_1, P_2, \ldots, P_m$, яка утворює базис в m-вимірному векторному просторі. Нехай розклад векторів $P_j (j = \overline{1,n})$ через вектори такого базису має вигляд:

$$P_j = x_{1j} P_1 + x_{2j} P_2 + \ldots + x_{mj} P_m (j = \overline{1,n}) .$$

Знайдемо величини: $Z_j = c_1 x_{1j} + c_2 x_{2j} + \ldots + c_m x_{mj}$, та обчислимо різниці: $\Delta_j = Z_j - C_j (j = \overline{1,n})$.

Теорема 1. (Ознака переходу до нового опорного плану). Якщо існує такий індекс $j (j = \overline{1,n})$, для якого $\Delta_j > 0$ та серед коефіцієнтів $x_{ij} (i = \overline{1,m})$ розкладу вектора P_j через вектори базису є хоча б один додатний, то від опорного розв'язку X можна перейти до нового опорного розв'язку X, такого, що $L(X') < L(X)$.

Наслідок. Якщо існує такий індекс $j = k$, для якого $\Delta_k > 0$ та серед коефіцієнтів $x_{ik} (i = \overline{1,m})$ є хоча б один додатний, то введення в базис вектора P_k веде до зменшення цільової функції.

Теорема 2. (Ознака необмеженості цільової функції на множині планів). Якщо існує такий індекс $j (j = \overline{1,n})$, для якого $\Delta_j > 0$ та всі коефіцієнти розкладу вектора P_j через вектори базису $x_{ij} < 0 (i = \overline{1,m})$, то цільова функція на множині планів є необмеженою, задача лінійного програмування не має розв'язку.

Теорема 3. (Ознака оптимальності опорного розв'язку). Якщо для всіх індексів $j (j = \overline{1,n})$ виконується умова $\Delta_j \le 0$, то опорний розв'язок X є

оптимальним.

13.2 Алгоритм симплексного методу

Нехай $X=(x_1,x_2,...,x_m,0,...0)$ - початковий опорний розв'язок канонічної задачі лінійного програмування, а базис, що йому відповідає, складається з векторів $p_1, p_2,..., p_m$. Знайдемо розклад векторів $p_j(j=1,n)$ через вектори базису: $p_j=x_{1j}p_1+x_{2j}p_2+...+x_{mj}p_m(j=1,2,...,n)$.

Обчислимо різниці $\Delta_j=z_j-c_j$ для всіх j, де $z_j=c_1x_{1j}+c_2x_{2j}+...+c_mx_{mj}$ та значення цільової функції Z_0 для плану X. $Z_0=c_1x_1+...+c_mx_m$

Можливі три такі випадки:

1. Для всіх індексів $j(j=\overline{1,n})$ виконується умова $\Delta_j=Z_j-c_j\le 0$.
2. Існує такий індекс j, для якого $\Delta_j=Z_j-c_j>0$ і всі коефіцієнти розкладу вектора P_j через вектори базису $x_{ij}\le 0(i=1,..,m)$
3. Для кожного індексу j, для якого $\Delta_j=z_j-c_j>0$, серед коефіцієнтів $x_{ij}(i=\overline{1,m})$ розкладу вектора P_j через вектори базису є хоч би один додатній.

У першому випадку розв'язування задачі припиняється, X - опорно оптимальний розв'язок. У другому випадку цільова функція на множині планів є необмеженою, і, відповідно, задача не має розв'язку.

У третьому випадку, якщо X - невироджений опорний розв'язок, можна перейти до нового опорного розв'язку з меншим значенням цільової функції.

Нехай виконується умова третього випадку. Перейдемо до нового опорного розв'язку. Для цього виключимо один вектор, який не належить до базису. На практиці у новий базис вводять той вектор P_j, який відповідає максимальній різниці $\Delta_j=Z_j-C_j$. Якщо $\max\Delta_j=\Delta_k$, то вектор P_k будемо вводити в новий базис.

Якщо $\min\dfrac{x_i}{x_k}=\dfrac{x_l}{x_{lk}}$, де мінімум береться по тих i, для яких $x_{ik}>0$, то вектор P_l необхідно ввести із базису. Елемент x_{lk} прийнято називати розв'язком.

Отже, у третьому випадку переходимо до нового опорного розв'язку $X'=(x_1',x_2',\ldots,x_n')$, де $x_i'=0(i\ne 1,2,\ldots,l-1,l+1,\ldots,m,k)$, якому відповідатиму базис, що складається з векторів $P_1,P_2,\ldots,P_{l-1},P_{l+1},\ldots,P_m,P_k$.

Обчислимо компоненти нового опорного плану X', коефіцієнти розкладу векторів P_j через вектори нового базису, значення цільової функції Z_0' і різниці $\Delta_j'=Z_j'-C_j$ для нового опорного плану. Оскільки:

$$P_k=x_{1k}P_1+x_{2k}P_2+\ldots+x_{l-1,k}P_{l-1}+x_{lk}P_l+x_{l+1,k}P_{l+1}+\ldots+x_{mk}P_m$$

та $x_{lk}\ne 0$, то з цієї рівності знаходимо розклад вектора P_l через вектори нового базису:

$$\mathrm{P}_l = -\frac{x_{1k}}{x_{lk}} \cdot \mathrm{P}_1 - \frac{x_{2k}}{x_{lk}} \cdot \mathrm{P}_2 - \ldots - \frac{x_{l-1,k}}{x_{lk}} \cdot \mathrm{P}_{l-1} - \frac{x_{l+1,k}}{x_{lk}} \cdot \mathrm{P}_{l+1} - \frac{x_{mk}}{x_{lk}} \cdot \mathrm{P}_m + \frac{1}{x_{lk}} \cdot \mathrm{P}_k .$$

Підставляючи замість P_l його розклад через вектори нового базису в рівність:

$x_1\mathrm{P}_1 + x_2\mathrm{P}_2 + \ldots + x_{l-1}\mathrm{P}_{l-1} + x_l\mathrm{P}_l + x_{l+1}\mathrm{P}_{l+1} + \ldots + x_m\mathrm{P}_m = \mathrm{P}_0$ одержуємо:

$$x'_1\mathrm{P}_1 + x'_2\mathrm{P}_2 + \ldots + x'_{l-1}\mathrm{P}_{l-1} + x'_l\mathrm{P}_l + x'_{l+1}\mathrm{P}_{l+1} + \ldots + x'_m\mathrm{P}_m = \mathrm{P}_0 \text{ де}$$

$$x'_i = x_i - \frac{x_l}{x_{lk}} \cdot x_{ik} \, (i = 1,2,\ldots,l-1,l+1,\ldots,m)$$

$$x'_k = \frac{x_l}{x_{lk}}$$

Підставляючи замість P_l його розклад через вектори нового базису в рівність:

$$\mathrm{P}_j = x_{1j}\mathrm{P}_1 + x_{2j}\mathrm{P}_2 + \ldots + x_{l-1,j}\mathrm{P}_{l-1} + x_{lj}\mathrm{P}_l + x_{l+1,j}\mathrm{P}_{l+1} + \ldots + x_{mj}\mathrm{P}_m ,$$

знаходимо розклад векторів P_j через вектори нового базису:

$$\mathrm{P}_j = x'_{1j}\mathrm{P}_1 + x'_{2j}\mathrm{P}_2 + \ldots + x'_{l-1,j}\mathrm{P}_{l-1} + x'_{l,j}\mathrm{P}_l + x'_{l+1,j}\mathrm{P}_{l+1} + \ldots + x'_{m,j}\mathrm{P}_m + x'_{k,j}\mathrm{P}_k$$

де

$$x'_{ij} = x_{ij} - \frac{x_{lj}}{x_{lk}} \cdot x_{ik} \, (i = 1,2,\ldots,l-1,l+1,\ldots,m)$$

$$x'_{kj} = \frac{x_{lj}}{x_{ik}}$$

Оскільки

$$\Delta'_j = Z'_j - C_j = C_1x'_{1j} + C_2x'_{2j} + \ldots + C_{i-1}x'_{l-1,j} + C_{l+1}x'_{l+1,j} + \ldots + C_mx'_{mj} + C_kx'_{kj} - C_j \, (j = 1,2,\ldots,n) ,$$

то підставляючи замість x_{ij} їх значення з формул, одержимо:

$$Z'_0 = Z_0 - \frac{x_l}{x_{ik}} \cdot \Delta_k .$$

Знайшовши компоненти нового опорного плану X', коефіцієнти розкладу векторів P_j через вектори нового базису, значення цільової функції Z'_0 і різниці $\Delta_j = Z_j - c_j$ для нового опорного плану, складено нову сиплекс таблицю.

Контрольні питання

1. Зробіть постановку задачі лінійного програмування в канонічній формі.
2. Методи зведення задачі лінійного програмування в неканонічній формі до канонічної форми.
3. Допустима область розв'язків задачі лінійного програмування.
4. Сформулюйте теорему про ознаку переходу до нового опорного плану.
5. Сформулюйте теорему про ознаку необмеженості цільової функції на множині планів.

6. Сформулюйте теорему про ознаку оптимальності опорного розв’язку.
7. Алгоритм симплекс-методу розв’язання задач лінійного програмування.

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна:

1. Лященко М.Я., Голованъ М.С. Чисельні методи. Київ: Либідь, 1996.
2. Фельдман Л.П., Петренко А.І., Дмитрієва О.А. Чисельні методи в інформатиці. – К.: Видавнича группа BHV, 2006. – 480 с.

Додаткова:

3. Демидович Б.П., Марон И.А. Основы вычислительной математики. Госуд-ное издательство физико-математической лит. 1960.
4. Волков Е.А. Численные методы: Учебное пособие. – М.: Наука. Главная редакция физико-математической литературы, 1982.
5. Мэтьюз Джон Г., Финк Куртис Д. Численные методы. Использование MATLAB, 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001.
6. Турчак Л.И. Основы численных методов: Учеб. Пособие. – М.: Наука. Главная редакция физико-математической литературы, 1987.
7. Банди Б. Основы линейного программирования / Пер. с англ. – М.: Радио и связь, 1989.
8. Кузнецов А.В., Сакович В.А., Холод Н.И. Высшая математика: Математическое программирование: учебник / под общ. Ред. А.В. Кузнецова. – Мн.: Выш. школа, 2001.