Лекція 10: Методи багатовимірної оптимізації

1. Методи багатовимірної оптимізації.
2. Порівняльна характеристика.

1. Методи багатовимірної оптимізації

Традиційно методи оптимізації в багатовимірному просторі діляться на три великі групи:

1) *прямі* методи засновані на порівнянні значень цільової функції в різних токах (покоординатного спуску, симплекс метод);

2) *градієнтні* методи засновані на точних значеннях перших похідних цільової функції;

3) методи *другого порядку* використовують також похідні другого порядку.

Стратегія прямих методів – поступове наближення до оптимуму, а при використанні непрямих методів прагнуть знайти розв'язок, не досліджуючи неоптимальні точки.

Метод покоординатного спуску. Логічним розвитком методики одновимірного пошуку є зміна кожного проектного параметра доти, поки не буде досягнуто мінімум цільової функції. По завершенню цієї процедури для всіх змінних можна повернутися до першої й подивитися, чи неможливо ще більше удосконалити рішення.

Симплекс метод. *Симплекс* в N-вимірному просторі є багатогранник, утворений $N+1$ рівновіддаленими точками-вершинами. У двовимірному випадку це трикутник, в тривимірному – тетраедр. Схеми пошуку з використанням симплексу засновані на стеженні за зміною значень цільової функції в їх вершинах. Головним в цих схемах є процес віддзеркалення – знаходження вершини нового симплексу, розташованої симетрично відносно площини, що проходить через одну із сторін вихідного симплексу. Вибір напряму пошуку вершини нового симплексу визначається положенням тієї вершини вихідного симплексу, в якій цільова функція має найгірше значення. Нова точка називається *доповненням* найгіршої точки. Якщо в тільки що отриманій вершині нового симплексу значення цільової функції виявляється гіршим, то алгоритм передбачає повернення у вихідну точку – вершину попереднього симплексу. Потім здійснюється перехід до тієї вершини попереднього симплексу, в якій цільова функція має наступне по величині значення, і відшукується точка, що є її доповненням.

Градієнтні методи. Метод оптимізації, в основу якого покладена ідея руху по найкрутішій стежці, називається методом *найшвидшого спуску*. Вектор градієнта перпендикулярний лінії рівня і вказує напрямок до нової точки в просторі проектування.

Методи II порядку. Для з'ясування, чи є дана точка мінімумом, максимумом або сідлом, доводиться досліджувати другі похідні функції.

2. Порівняльна характеристика

Рекомендації щодо вибору алгоритмів оптимізації функції багатьох змінних. Кожній складній задачі проектування властиві свої специфічні особливості, що ускладнюють використання традиційних алгоритмів оптимізації. Хоча немає жодного універсального методу, що дозволяє успішно вирішувати всі задачі, деякі методи краще пристосовані для вирішення задач певних типів. Ретельний вибір відповідного алгоритму часто дозволяє заощадити і машинний час і зусилля. Вибираючи алгоритм, слід враховувати наступні рекомендації (розташовані в довільному порядку).

1. *Проаналізувати особливості поверхні, описуваною цільовою функцією*. Якщо відомі топологічні властивості, досліджуваній поверхні, це може допомогти правильно вибрати відповідний алгоритм.

Якщо поверхня має гладкі складки, не рекомендується застосовувати методи покоординатного підйому або градієнтні методи.

Якщо ж складки явно виражені, слід віддати перевагу методам конфігурацій перед градієнтним методам .

Для поверхонь з глибокими западинами метод симплексу або метод Розенброка часто виявляються ефективнішими.

Якщо поверхня мультимодальна, правильніше буде вибрати в просторі проектування декілька початкових точок і переконатися, що у всіх випадках виходить одне і те ж рішення. При виявленні декількох локальних оптимумів конструкцію слід розробляти з урахуванням кращого з них. Нажаль, навіть найретельніший вибір початкових точок не гарантує знаходження всіх локальних оптимумів.

2. *Вивчити характер проектних параметрів*. Хоча, як правило, більшість параметрів, з якими має справу інженер, можуть набувати будь-яких значень, деякі параметри мають лише дискретні або цілі значення. У таких випадках можна користуватися звичайними алгоритмами, не звертаючи уваги на особливості змінних.

Знайшовши оптимальне рішення, слід округлити значення змінних до цілих значень. При цьому доводиться перевіряти значення цільової функції при округленні значення змінної до найближчого більшого або меншого цілого значення. Якщо задача дуже складна, такий підхід, можливо, не дозволить отримати найкраще рішення. В цьому випадку доведеться вдатися до методів, спеціально пристосованих для вирішення таких задач. Їх опис можна знайти в літературі з питань оптимізації. На жаль, вони не гарантують, що отримане рішення буде краще рішення, отриманого в результаті округлення.

3. *З'ясувати, скільки часу буде потрібно для вирішення задачі.* Час, необхідний для вирішення задачі оптимізації за допомогою даного алгоритму, складається з суми часу підготовки задачі і часу рахунку на комп'ютері. Складання додаткових підпрограм (наприклад, при введенні штрафної функції) підвищує вартість використання алгоритму. Нерідко, вибираючи той або інший алгоритм, доводиться йти на компроміс, вирішуючи питання, чи слід витратити додатковий час на підготовку рішення задачі «швидким» методом або правильніше скористатися не настільки швидким, але простішим методом. Це питання слід вирішувати кожного разу, оскільки вартість часу програмування і машинного часу залежить від того, де виконується робота.

4. *Проаналізувати особливості алгоритму*. Аналіз структури алгоритму часто дозволяє вирішити питання щодо його придатності для вирішення тієї або іншої конкретної задачі. Не рекомендується використовувати методи, що вимагають знаходження похідних в аналітичному вигляді, оскільки для багатьох цільових функцій це неможливо зробити (прикладом можуть бути залежності, отримані експериментально). Крім того, якщо структура похідних, отриманих аналітично, дуже складна, то диференціювання може різко збільшити вартість підготовки задачі і ймовірність помилок з боку програміста. Методи, що включають зміну масштабів, як правило, більш ефективні і гнучкі. Відзначимо, що гнучкіші програми для ЕОМ зазвичай не відрізняються великими розмірами або часом рахунку. Оскільки завжди бажано випробувати для рішення даної задачі декілька різних алгоритмів, то заслуговує на увагу «пакетний» хід, що дозволяє переходити з одного алгоритму на інший шляхом заміни в основній програмі звернення до підпрограми. Тим самим виключається необхідність складання нової програми, економиться час і скорочуються трудозатрати.