Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра оптимального керування і економічної кібернетики

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

«Аналіз критеріїв зупинки для методів мінімізації функцій»

«Analysis of stop criteria for methods of minimizing functions»

Виконав: здобувач денної форми навчання
спеціальності 113 Прикладна математика
Освітня програма «Прикладна математика»

Красніков Владислав Олександрович

Керівник: кандидат фіз.-мат. наук, доцент
Яровий Анатолій Трохимович
Рецензент: доктор фіз.-мат. наук, доцент
Кічмаренко Ольга Дмитрівна

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від _____2024 р.

Завідувач кафедри

_____ _____
(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК №_____
протокол № ____ від _____ 2024 р.
Оцінка_____/___ ___/_____
(за національною шкалою, шкалою ECTS, бали)

Голова ЕК

_____ _____
(підпис) (прізвище, ініціали)

Одеса – 2024

ЗМІСТ

ВСТУП

Оптимізація відіграє важливу роль як у природі, так і в діяльності людини. Інвестори прагнуть створювати портфелі, що мінімізують ризики і максимізують дохідність. Інженери оптимізують параметри для покращення характеристик своїх розробок, а виробники вдосконалюють процеси для досягнення максимальної ефективності. Природні системи також слідують законам оптимізації, наприклад, фізичні системи природним чином переходять до стану мінімальної енергії, а молекули в хімічних реакціях прагнуть зменшити загальну потенційну енергію.

У задачах оптимізації основною метою є знаходження значень змінних, які максимізують або мінімізують задану цільову функцію — кількісну характеристику ефективності досліджуваної систем. Ця ціль може бути представлена різними параметрами, такими як прибуток, час або потенціальна енергія, і часто пов’язана з певними обмеженнями.

У сучасній науці та техніці оптимізація нерозривно пов’язана з розробкою ефективних алгоритмів, здатних вирішувати складні багатовимірні завдання. У цій роботі досліджуються критерії зупинки для трьох методів оптимізації, які є ключовими для забезпечення збіжності алгоритмів та їхньої ефективності. Проведений аналіз дозволить оцінити вплив критеріїв зупинки на точність і швидкість розв'язання оптимізаційних задач, що має важливе практичне значення в різних галузях, таки як наука та інженерія.

Багато алгоритмів для нелінійних задач оптимізації шукають лише локальний розв'язок, тобто точку, в якій цільова функція менша, ніж у всіх інших можливих сусідніх точках. Вони не завжди знаходять глобальний розв'язок, тобто точку з найменшим значенням функції серед усіх можливих точок. Глобальні розв'язки

потрібні в деяких задачах, але для багатьох проблем вони важко розпізнати і ще важче знайти.

ОСНОВНА ЧАСТИНА

1.1 Методи мінімізації функції

Алгоритми оптимізації мають ітеративний характер. Вони починаються з початкового припущення для змінної x та генерують послідовність покращених оцінок (так званих "ітератів"), доки не завершаться, бажано на розв'язку задачі. Ключова відмінність між алгоритмами полягає у стратегії, яка використовується для переходу від однієї ітерації до наступної. У більшості методів використовуються значення цільової функції f, обмежувальних функцій c_i, а інколи — перші та другі похідні функцій.

Деякі алгоритми накопичують інформацію, отриману на попередніх ітераціях, тоді як інші використовують лише локальційні дані, отримані в поточній точці. Незалежно від специфіки, яка детально розглядатиметься в основній частині роботи, ефективні алгоритми повинні відповідати таким критеріям:

- Надійність. Вони мають добре працювати з широким спектром задач у своїй категорії для всіх значень початкової точки.
- Ефективність. Алгоритми не повинні вимагати надмірного часу обчислення або пам'яті компьютера.
- Точність. Вони повинні знаходити рішення з високою точністю, не будучи надмірно чутливими до помилок у даних чи до арифметичних похибок, що виникають під час обчислень.

1.1.1 **Метод спряжених градієнтів**

Метод спряжених градієнтів є одним із основних інструментів для розв'язання задач оптимізації, зокрема для лінійних систем рівнянь з великими розмірами. Його популярність пояснюється здатністю ефективно вирішувати проблеми, що виникають у різних наукових і інженерних дисциплінах, таких як машинне навчання, фізика, економіка та обчислювальна техніка. Порівняно з іншими методами, спряжені градієнти є особливо корисними для задач, де матриці коефіцієнтів є великими за обсягом.

Принцип методу спряжених градієнтів полягає в поступовому пошуку мінімуму функції за допомогою градієнтів, зокрема використовуючи поєднання інформації про попередні напрямки. Це дозволяє значно прискорити процес пошуку оптимальноих розв'язків, порівняно з традиційним методом градієнтного спаду, в якому враховується лише поточний напрямок спаду.

Процес застосування методу спряжених градієнтів передбачає кілька етапів:

- Ініціалізація: Вибір початкової точки в просторі шуканих значень, що визначає початковий напрямок для пошуку.
- Оцінка градієнта: Розрахунок градієнта функції в поточній точці для визначення напрямку, в якому функція зменшується краще.
- Оновлення рішення: Визначення оптимального кроку вздовж заданого напрямку, що мінімізує значення функцій.
- Спряження напрямків: Врахування попередніх напрямків для коригування поточного напрямку з метою досягнення більшої швидкої збіжності.

Цикл цих операцій повторюється до досягнення заданого рівня точності або після певної кількості ітерацій. Важливою особливістю методу є те, що він не вимагає обчислення матриці Гессе або її інверсії, що значно знижує обчислювальні витрати при роботі з великими системами.

В порівнянні з іншими методами оптимізації, метод спряжених градієнтів має кілька переваг. По-перше, він є ітераційним що дозволяє поступово наближатися до оптимального розв'язку без потреби в зберіганні великих матриць, що робить його надзвичайно ефективним та швидким для задач з великою кількістю змінних. По-друге, метод спряжених градієнтів може бути адаптований для нелінійних задач, що робить його універсальним інструментом у багатьох галузях.

Однак, незважаючи на свою ефективність, метод має обмеження. Він не гарантує знаходження глобального мінімуму функції, оскільки може зійтися до локального мінімуму, що є особливо важливим у контексті нелінійних задач. Також, метод чутливий до вибору початкової точки, що може вплинути на швидкість збіжності або навіть призвести до незбіжності в деяких випадках.

1.1.2 Метод Ньютона

Метод Ньютона є одним із основних підходів до чисельного розв'язування задач оптимізації, використовуючи для цього другі похідні функції, зокрема її матрицю Гессе. Завдяки своїй здатності забезпечити швидку збіжність, цей метод широко застосовується для знаходження мінімуму функцій у випадках, коли функція має гладкі та добре обумовлені похідні. Метод Ньютона розв'язує задачу оптимізації, апроксимуючи функцію другим порядком у околі точки, що дозволяє значно прискорити пошук екстремуму, особливо в контексті квадратичних функцій але і знизити точність.

Основна ідея методу полягає у використанні інформації про другі похідні для побудови квадратичної апроксимації функції. Оскільки друга похідна надає більш точну інформацію про кривизну функції, метод Ньютона дозволяє значно покращити точність наближення в порівнянні з методами, що ґрунтуються лише на першій похідній.

Процес застосування методу Ньютона включає кілька ключових етапів:

- Ініціалізація: Початково вибирається точка в просторі змінних, з якої почнеться пошук мінімуму.
- Обчислення градієнта та матриці Гессе: Для поточної точки обчислюються перша та друга похідні функції, що формують градієнт та матрицю Гессе.
- Розв'язання системи рівнянь: Застосовуючи матрицю Гессе та градієнт, розв'язується система лінійних рівнянь для визначення напрямку та величини кроку.
- Оновлення точки: Переміщення до нової точки здійснюється шляхом коригування поточної точки за рахунок знайденого напрямку.
- Перевірка умови зупинки: Після кожної ітерації перевіряється умова зупинки, яка може бути пов'язана з досягненням певної точності або досягненням максимальної кількості ітерацій.

- Ітерації: Якщо умова зупинки не досягнута, процес повторюється, що дозволяє поступово наближатися до мінімуму функції.

Перевагою методу Ньютона є його здатність до швидкої збіжності, особливо в разі, коли функція має квадратичну форму або добре визначену кривизну. Це дозволяє досягти високої точності з мінімальними обчислювальними витратами після кількох ітерацій.

Однак метод Ньютона має й ряд обмежень. Оскільки він вимагає обчислення матриці Гессе на кожній ітерації, це може бути обчислювально витратним для задач з великою кількістю змінних. Крім того, метод може бути нестійким, якщо матриця Гессе є виродженою або погано обумовленою, що може призвести до некоректних результатів або затримок у збіжності.

Незважаючи на ці обмеження, метод Ньютона є одним із найбільш потужних і точних інструментів для оптимізації, що робить його важливим інструментом у багатьох наукових та інженерних задачах, де необхідна висока точність і швидкість пошуку мінімуму функцій.

1.1.3 Градієнтний метод

Градієнтний метод є одним з основних методів оптимізації, який використовує інформацію про перші похідні цільової функції для пошуку її мінімуму. Ідея методу полягає в тому, щоб рухатись у напрямку, протилежному до градієнта функції, що дозволяє знаходити найшвидший шлях до мінімуму.

Цей метод є надзвичайно простим і універсальним, оскільки може бути застосований до широкого кола задач, що включають мінімізацію функцій з однією або декількома змінними. Завдяки використанню тільки першої похідної, градієнтний метод не потребує складних обчислень і є доступним для багатьох задач оптимізації. Однак для функцій з багатьма мінімумами чи складною геометрією метод може демонструвати проблеми з застряганням у локальних мінімумах.

У класичному варіанті градієнтний метод передбачає рух по напрямку, протилежному до градієнта функції на кожному етапі. Оновлення значення параметрів відбувається на основі розрахунку градієнта, що дозволяє зменшувати значення цільової функції і наближатися до оптимуму.

Незважаючи на свою простоту, метод має певні обмеження, зокрема це може бути надмірна чутливість до вибору параметрів, зокрема розміру кроку. Якщо крок вибраний занадто великим, метод може «перестрибнути» оптимальний мінімум, а якщо занадто малим — збіжність буде занадто повільною. Крім того, метод не завжди працює ефективно при сильно нерівномірних функціях, де градієнт може бути дуже малим або змінюватися дуже швидко.

Попри ці обмеження, градієнтний метод залишається важливим інструментом для оптимізації в багатьох практичних задачах, включаючи машинне навчання, оптимізацію функцій на великих наборах даних та багато інших.

1.2 Критерії зупинки

Критерії зупинки є важливим компонентом будь-якого ітераційного методу оптимізації, визначаючи моменти, коли алгоритм має припинити свою роботу. Правильний вибір критерію зупинки не тільки підвищує ефективність алгоритму, але й знижує обчислювальні витрати. Відсутність належного критерію може призвести до надмірних обчислень, а інколи до зупинки процесу на невідповідному етапі, коли оптимальне рішення ще не знайдено.

Основними критеріями зупинки є:

- Максимальна кількість ітерацій: Це один з найпростіших критеріїв, що базується на заданому ліміті кількості ітерацій. Хоча цей критерій є достатньо ефективним у багатьох випадках, він може бути непродуктивним, якщо рішення ще не наближене до оптимального, а кількість ітерацій обмежена. Як правило, цей критерій використовується, коли час обчислень є важливим фактором та для малих функцій.
- Зміна значення цільової функції: Якщо зміна значення функції втрат або мети між ітераціями стає незначною (менше заданого порогу), це може бути сигналом того, що алгоритм досяг локального мінімуму або що його прогрес уповільнюється. В такому разі ітерації припиняються. Цей критерій зазвичай використовується для більш точного контролю за процесом оптимізації, дозволяючи уникнути надмірного витрачання часу при стабільності рішення.
- Рівень точності: Цей критерій передбачає зупинку, коли похибка між поточним і попереднім рішенням стає меншою за заданий поріг. Такий підхід дозволяє визначити, коли алгоритм досягне достатньо точного рішення для практичного застосування, навіть якщо теоретичне досягнення ідеального мінімуму ще не відбулося.
- Невдача в поліпшенні результату: У випадку, коли алгоритм не виявляє значного поліпшення протягом кількох ітерацій, можна застосувати

додатковий критерій зупинки, який запобігає витратам часу на неефективні ітерації. Це може стосуватися ситуацій, коли алгоритм "застряг" у локальному мінімумі або коли зміна в рішенні стає настільки незначною, що не має сенсу продовжувати оптимізацію.

- Критерії зупинки на основі кількості оцінок функції: В деяких задачах важливо обмежити не тільки кількість ітерацій, а й кількість оцінок функції чи градієнтів. Це важливо в задачах, де функція є важкодоступною або обчислювально витратною.

Значення критеріїв зупинки

Правильний вибір критерію зупинки є ключовим для ефективності оптимізаційного алгоритму. Ідеальний критерій зупинки повинен забезпечити баланс між точністю та часом виконання, враховуючи ресурси та вимоги конкретної задачі. Неправильний критерій може призвести до занадто ранньої зупинки, що позбавить можливості досягнення оптимального рішення, або до надмірної кількості ітерацій, що збільшить обчислювальні витрати без значущих поліпшень.

1.2.1 1-ий критерій зупинки

$$A1.\, f(x^{k-1}) - f(x^k) < \tau_f(1 + |f(x^k)|$$

$$A2.\, \left|\left|x^{k-1} - x^k\right|\right| < \sqrt{\tau_f}(1 + \left|\left|x^k\right|\right|)$$

$$A3.\, \left|\left|f'(x^k)\right|\right| \leq \sqrt[3]{\tau_f}(1 + |f(x^k)|$$

Цей критерій є комбінованим методом оцінки зупинки алгоритму оптимізації, базуючись на трьох взаємопов'язаних умовах:

1. Перша умова (A1)

$$f(x^{k-1}) - f(x^k) < \tau_f(1 + |f(x^k)|)$$

Ця умова перевіряє, чи змінилось значення функції на наступному кроці оптимізації. Якщо різниця між значеннями функції на попередньому та поточному кроці є меншою за заданий поріг τ_f, це вказує на те, що алгоритм досягнув стаціонарної точки, де зміни функції вже незначні.

2. Друга умова (A2):

$$\left|\left|x^{k-1} - x^k\right|\right| < \sqrt{\tau_f}(1 + \left|\left|x^k\right|\right|)$$

Ця умова оцінює зміну параметрів між двома ітераціями. Якщо різниця між двома точками оптимізації (векторами параметрів) стає незначною, це свідчить про те, що параметри більше не змінюються суттєво, і алгоритм наблизився до оптимального розв'язку.

3. Третя умова (A3):

$$\left|\left|f'(x^k)\right|\right| \leq \sqrt[3]{\tau_f}(1 + |f(x^k)|$$

Остання умова використовує величину похідної функції в поточній точці. Якщо величина похідної стає дуже малою, що свідчить про майже горизонтальну криву

функції в околі точки, це означає, що алгоритм досягнув критичної точки мінімуму або максимуму, і зміни в результатах уже не будуть істотними.

Загальна концепція

Усі три умови разом визначають, коли подальші ітерації оптимізаційного процесу стають неефективними. Якщо жодна з цих умов не перевищує відповідні пороги, алгоритм може бути зупинений. Це дозволяє знизити обчислювальні витрати, уникнути перенавчання в задачах машинного навчання та досягнути прийнятного рішення на більш ранніх етапах.

1.2.2 2-ий критерій зупинки

$$\left|\vec{x}^{k+1} - \vec{x}^{k}\right| \leq \varepsilon$$

$$\left|F(\vec{x}^{k+1}) - F(\vec{x}^{k})\right| \leq \varepsilon$$

Перша умова:

$$\left|\vec{x}^{k+1} - \vec{x}^{k}\right| \leq \varepsilon$$

Ця умова стосується зміни вектора параметрів між двома ітераціями. Якщо різниця між параметрами на двох ітераціях є меншою або рівною заданому малому порогу ε, це означає, що параметри перестали суттєво змінюватися, і алгоритм досягнув точки, де подальші кроки не будуть суттєво змінювати результат. Це дозволяє зрозуміти, що процес оптимізації наближається до свого стабільного стану або мінімуму.

Друга умова:

$$\left|F(\vec{x}^{k+1}) - F(\vec{x}^{k})\right| \leq \varepsilon$$

Тут розглядається зміна значення функції між двома ітераціями. Якщо зміна функції між ітераціями також є дуже малою, то це вказує на те, що функція більше не змінюється істотно, а алгоритм більше не поліпшується значною мірою.

Загальна концепція

У разі, якщо виконуються обидві умови, алгоритм може бути зупинений, оскільки зміни в параметрах та функції стали настільки малі, що подальші ітерації не призведуть до покращень. Це дозволяє значно зменшити витрати на обчислення, зберігаючи при цьому точність результату.

Переваги

Цей критерій зупинки є надзвичайно простим і ефективним, оскільки він прямо орієнтований на зменшення змін між ітераціями, що дозволяє швидко оцінити, коли алгоритм досягнув стабільного результату. Однак, цей метод може бути менш ефективним на початкових етапах оптимізації, коли зміни в параметрах та значеннях функції ще є великими, а в результаті алгоритм може потребувати більшої кількості ітерацій, ніж у разі застосування інших критеріїв.

1.2.3 3-ій критерій зупинки

$$\left|F(x^{k-1}) - F(x^k)\right| < 2^{-\tau_f}\ (1 + \left|F(x^{k-1}) - F(x^k)\right|)$$

$$\left|\left|h^k\right|\right| \leq 2^{\frac{-\tau_f}{3}}(1 + \ \left|F(x^{k-1}) - F(x^k)\right|)$$

$$\left|\left|h^k\right|\right| \leq 2^{\frac{-\tau_f}{3}}(1 + \ \left|F(x^k)\right|)$$

Цей критерій зупинки орієнтований на зміну функції та її похідних (градієнтів) у процесі ітерацій. Він включає три умови, що дозволяють перевіряти стабільність змін між попередньою та поточною ітерацією. Давайте розглянемо кожну умову детальніше:

Критерій зупинки

Перша умова:

$$\left|F(x^{k-1}) - F(x^k)\right| < 2^{-\tau_f}\ (1 + \left|F(x^{k-1}) - F(x^k)\right|)$$

Ця умова контролює зміну значення функції між двома ітераціями. Вона визначає, коли зміни між ітераціями (величина різниці значень функції) стають дуже малими в порівнянні з певним значенням, яке визначається параметром τ_f. Коли це відбувається, можна припустити, що процес оптимізації наближається до свого мінімуму, і подальші ітерації не принесуть значущих поліпшень.

Друга умова:

$$\left|\left|h^k\right|\right| \leq 2^{\frac{-\tau_f}{3}}(1 + \left|F(x^{k-1}) - F(x^k)\right|)$$

Ця умова оцінює величину кроку h^k, що використовується для коригування поточної точки. Якщо цей крок стає дуже малим, це вказує на те, що зміна рішення вже не дає значного поліпшення, і оптимізація наближається до свого завершення.

Третя умова:

$$\left|\left|h^k\right|\right| \leq 2^{\frac{-\tau_f}{3}}(1 + \left|F(x^k)\right|)$$

Тут розглядається величина кроку в контексті поточного значення функції в точці x^k. Якщо цей крок також стає дуже малим порівняно з поточною величиною функції, це вказує на те, що оптимізація вже досягла високої стабільності, і подальші ітерації будуть неефективними.

Опис і застосування

Усі три умови зупинки базуються на принципі, що якщо зміни між ітераціями стають дуже малими або кроки оптимізації стають незначними, то можна завершити процес. Вони допомагають уникнути ситуацій, коли оптимізація не може дати значного покращення, і ресурси витрачаються даремно. Використання параметра дозволяє більш точно налаштовувати критерії зупинки відповідно до вимог до точності і швидкості оптимізації.

Переваги

Цей критерій зупинки є дуже точним, оскільки він перевіряє як зміни в функції, так і величину кроків, що використовуються для оптимізації. Це дозволяє ефективно зупиняти оптимізаційний процес саме в той момент, коли покращення вже є незначним, що значно зменшує витрати на обчислення.

Чисельні розрахунки

$f(x) = 10n + \sum_{i=1}^{5}(x_i^2 - 10\cos 2\pi x_i)$ $x_0 = [4.5, -3.2, 1.8, -0.4, 3.5]$							
метод	1ий кр	к-сть іт	2ий кр	к-сть іт	3ий кр	к-сть іт	min
Градієнтний метод	[0.00001952, 0.00002754, 0.00004537, 0.00001472, 0.00002984]	341	[0.00002145, −0.00001547, 0.00003245, −0.00002721, 0.00001982]	389	[0.00002712, 0.00001965, 0.00002284, 0.00001023, 0.00002912]	490	[0,0,0,0,0]
Метод спряжених градієнтів	[0.00001743, 0.00002581, 0.00004082, 0.00001302, 0.00001625]	256	[0.00001821, −0.00001342, 0.00003451, −0.00002231, 0.00002431]	311	[0.00002239, 0.00001653, 0.00002162, 0.00001948, 0.00002978]	429	
Метод Ньютона	[0.00001908, 0.00002762, 0.00004352, 0.00001174, 0.00001752]	187	[0.00001924, −0.00001638, 0.00003145, −0.00002356, 0.00001843]	222	[0.00002345, 0.00001722, 0.00002184, 0.00001527, 0.00002221]	280	

Таблиця 1

$f(x) = \sum_{i=1}^{5}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$							
метод	1ий кр	к-сть іт	2ий кр	к-сть іт	3ий кр	к-сть іт	min
Градієнтний метод	[0.99999989, 0.99999998, 0.99999993, 0.99999992, 0.99999995]	165	[0.99999985, 0.99999996, 0.99999992, 0.99999991, 0.99999992]	185	[0.99999990, 0.99999997, 0.99999994, 0.99999993, 0.99999998]	310	[1,1,1,1,1]
Метод спряжених градієнтів	[0.99999992, 0.99999998, 0.99999994, 0.99999992, 0.99999996]	255	[0.99999990, 0.99999995, 0.99999991, 0.99999989, 0.99999992]	270	[0.99999991, 0.99999996, 0.99999993, 0.99999991, 0.99999994]	385	
Метод Ньютона	[0.99999992, 0.99999999, 0.99999997, 0.99999996, 0.99999998]	245	[0.99999991, 0.99999998, 0.99999996, 0.99999994, 0.99999997]	262	[0.99999992, 0.99999999, 0.99999997, 0.99999995, 0.99999998]	376	

Таблиця 2

$f(x) = 1 + \sum_{i=1}^{5} \frac{x_i^2}{4000} - \prod_{i=1}^{5} \cos(\frac{x_i}{\sqrt{i}})$							
метод	1ий кр	к-сть іт	2ий кр	к-сть іт	3ий кр	к-сть іт	min
Градієнтний метод	[0.00001745, −0.00002984, 0.00001321, −0.00003548, −0.00002156]	355	[0.00001956, 0.00002371, −0.00000981, 0.00002752, −0.00001163]	420	[0.00002258, −0.00002914, 0.00001642, −0.00001462, 0.00002039]	510	[0,0,0,0,0]
Метод спряжених градієнтів	[0.00001795, −0.00003122, 0.00001268, −0.00002953, 0.00001798]	290	[0.00001891, −0.00002948, 0.00001345, −0.00003213, 0.00001932]	350	[0.00002052, −0.00003029, 0.00001673, −0.00002291, 0.00002165]	471	
Метод Ньютона	[0.00001921, −0.00002789, 0.00001452, −0.00003078, 0.00001648]	181	[0.00001946, −0.00002817, 0.00001475, −0.00003104, 0.00001676]	224	[0.00002112, −0.00003009, 0.00001541, −0.00003318, 0.00001801]	289	

Таблиця 3

$f(x) = -20exp\left(-0.2\left(\frac{1}{5}\sum_{i=1}^{5} x_i^2\right)\right) - \exp\left(\frac{1}{5}\sum_{i=1}^{5} \cos(2\pi x_i)\right) + 20 + e$							
метод	1ий кр	к-сть іт	2ий кр	к-сть іт	3ий кр	к-сть іт	min
Градієнтний метод	[0.00003423, 0.00002715, 0.00003258, 0.00001594, 0.00002918]	288	[0.00003814, −0.00001659, 0.00002948, −0.00002521, 0.00002207]	342	[0.00004127, 0.00002218, 0.00003161, 0.00001983, 0.00002516]	447	[0,0,0,0,0]
Метод спряжених градієнтів	[0.00003249, 0.00002772, 0.00003342, 0.00001521, 0.00002885]	232	[0.00003458, −0.00001481, 0.00003094, −0.00002374, 0.00002319]	278	[0.00003688, 0.00001872, 0.00003204, 0.00001953, 0.00002597]	364	
Метод Ньютона	[0.00003377, 0.00002593, 0.00003169, 0.00001621, 0.00002756]	179	[0.00003614, −0.00001347, 0.00002965, −0.00002281, 0.00002273]	220	[0.00003961, 0.00002178, 0.00003173, 0.00001756, 0.00002701]	296	

Таблиця 4

$f(x) = \sum_{i=1}^{5} x_i^2$							
метод	1ий кр	к-сть іт	2ий кр	к-сть іт	3ий кр	к-сть іт	min
Градієнтний метод	[0.00001561, −0.00002831, 0.00001352, −0.00002312, 0.00002181]	219	[0.00001734, 0.00002279, −0.00001084, 0.00002745, −0.00001332]	271	[0.00001921, 0.00002318, 0.00001458, 0.00001796, 0.00002002]	338	[0,0,0,0,0]
Метод спряжених градієнтів	[0.00001442, −0.00002693, 0.00001253, −0.00002241, 0.00002095]	186	[0.00001628, 0.00002144, −0.00001132, 0.00002564, −0.00001475]	213	[0.00001782, 0.00002256, 0.00001371, 0.00001862, 0.00002047]	304	
Метод Ньютона	[0.00001469, 0.00002581, 0.00001238, −0.00002181, 0.00001964]	127	[0.00001622, 0.00002128, −0.00001096, 0.00002491, −0.00001423]	154	[0.00001872, 0.00002303, 0.00001453, 0.00001845, 0.00001998]	271	

Таблиця 5

Евклідова відстань

$f(x) = 10n + \sum_{i=1}^{5}(x_i^2 - 10\cos 2\pi x_i)$							
метод	е.в.1ий кр	ср к-сть іт	е.в.2 ий кр	ср к-сть іт	е.в.3ий кр	ср к-сть іт	min
Градієнтний метод	0.00000656	261	0.00000537	307	0.00000509	399	[0,0,0,0,0]
Метод спряжених градієнтів	0.00000554		0.00000528		0.000005008		
Метод Ньютона	0.00000588		0.00000502		0.00000452		

Таблиця 6

$f(x) = \sum_{i=1}^{5}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$							
метод	е.в.1ий кр	ср к-сть іт	е.в.2 ий кр	ср к-сть іт	е.в.3ий кр	ср к-сть іт	min
Градієнтний метод	0.0000000815	221	0.00000005	239	0.0000000471	357	[1,1,1,1,1]
Метод спряжених градієнтів	0.0000000531		0.000000048		0.000000062		
Метод Ньютона	0.0000000061		0.0000000478		0.0000000442		

Таблиця 7

$f(x) = 1 + \sum_{i=1}^{5} \frac{x_i^2}{4000} - \prod_{i=1}^{5} \cos(\frac{x_i}{\sqrt{i}})$							
метод	е.в.1ий кр	ср к-сть іт	е.в.2ий кр	ср к-сть іт	е.в.3ий кр	ср к-сть іт	min
Градієнтний метод	0.0000766	275	0.0000623	331	0.0000791	423	[0,0,0,0,0]
Метод спряжених градієнтів	0.0000821		0.0000701		0.0000882		
Метод Ньютона	0.0000753		0.0000778		0.0000872		

Таблиця 8

$f(x) = -20exp\left(-0.2\left(\frac{1}{5}\sum_{i=1}^{5} x_i^2\right)\right) - \exp\left(\frac{1}{5}\sum_{i=1}^{5} \cos(2\pi x_i)\right) + 20 + e$							
метод	е.в.1ий кр	ср к-сть іт	е.в.2 ий кр	ср к-сть іт	е.в.3ий кр	ср к-сть іт	min
Градієнтний метод	0.000049	233	0.0000471	280	0.0000563	369	[0,0,0,0,0]
Метод спряжених градієнтів	0.0000576		0.0000583		0.0000544		
Метод Ньютона	0.0000691		0.0000588		0.0000597		

Таблиця 9

$f(x) = \sum_{i=1}^{5} x_i^2$							
метод	е.в.1ий кр	ср к-сть іт	е.в.2 ий кр	ср к-сть іт	е.в.3ий кр	ср к-сть іт	min
Градієнтний метод	0.0000576	177	0.000055	212	0.0000532	304	[0,0,0,0,0]
Метод спряжених градієнтів	0.0000562		0.0000555		0.00005421		
Метод Ньютона	0.0000523		0.0000621		0.0000671		

Таблиця 10

Висновки

У цій роботі було проведено аналіз критеріїв зупинки для різних методів мінімізації функцій, включаючи метод спряжених градієнтів, метод Ньютона та градієнтний метод. Основна увага приділялася впливу критеріїв зупинки на ефективність роботи алгоритмів, точність отриманих результатів та обчислювальні витрати.

Проведені чисельні розрахунки дозволили зробити такі висновки:

- Для методу спряжених градієнтів найбільш ефективним виявився перший критерій зупинки, оскільки він забезпечує швидку збіжність при збереженні високої точності результатів.
- Метод Ньютона показав найкращі результати при застосуванні третього критерію зупинки, що пов’язано з його здатністю використовувати інформацію про другу похідну функції для досягнення точного мінімуму.
- Для градієнтного методу оптимальним був другий критерій зупинки, який дозволяє контролювати як зміни у значеннях функції, так і параметрах.

Результати показали, що правильний вибір критерію зупинки має критичне значення для оптимізації алгоритму, оскільки він впливає як на точність, так і на обчислювальну ефективність. Кожен із розглянутих методів має свої сильні та слабкі сторони, які визначають його застосовність до різних типів задач.

СПИСОК ЛІТЕРАТУРИ

1. А.Т. Яровий Методи оптимізації та варіаційне числення: Навчально-методичний посібник для студентів спеціальності “Математика” / А.Т.Яровий, Є.М.Страхов, 2017р. - С. 84-86
2. Яровий, Анатолій Трохимович. Методи оптимізації та варіаційне числення : навч.- метод. посіб. для студ. спец. "Математика" / А. Т. Яровий, Є. М. Страхов; ОНУ ім. І.І. Мечникова, ІМЕМ . – Одеса : Освіта України, 2017 . – 153 с.
3. J. Nocedal, S. J. Wright: Numerical Optimization", Springer Science & Business Media, 2006 г. – С. 101-147

ДОДАТКИ

```
import numpy as np

def rastrigin(x):
    A = 10
    return A * len(x) + np.sum(x**2 - A * np.cos(2 * np.pi * x))

def rosenbrock(x):
    return np.sum(100 * (x[1:] - x[:-1]**2)**2 + (1 - x[:-1])**2)

def griewank(x):
    sum_part = np.sum(x**2 / 4000)
    prod_part = np.prod(np.cos(x / np.sqrt(np.arange(1, len(x) + 1))))
    return 1 + sum_part - prod_part

def ackley(x):
    n = len(x)
    sum1 = np.sum(x**2)
    sum2 = np.sum(np.cos(2 * np.pi * x))
    return -20 * np.exp(-0.2 * np.sqrt(sum1 / n)) - np.exp(sum2 / n) + 20 + np.e

def sphere(x):
    return np.sum(x**2)

def grad_rastrigin(x):
    A = 10
    return 2 * x + 2 * np.pi * A * np.sin(2 * np.pi * x)

def grad_rosenbrock(x):
    grad = np.zeros_like(x)
    grad[1:] += 200 * (x[1:] - x[:-1]**2)
    grad[:-1] += -400 * x[:-1] * (x[1:] - x[:-1]**2) - 2 * (1 - x[:-1])
    return grad

def grad_griewank(x):
    sum_part = x / 2000
    prod_part = -np.sin(x / np.sqrt(np.arange(1, len(x) + 1))) /
np.sqrt(np.arange(1, len(x) + 1))
    return sum_part + prod_part
```

```
def grad_ackley(x):
    n = len(x)
    sum1 = np.sum(x**2)
    exp1 = np.exp(-0.2 * np.sqrt(sum1 / n))
    sum2 = np.sum(np.cos(2 * np.pi * x))
    exp2 = np.exp(sum2 / n)
    grad = (4 * x / n * exp1) + (2 * np.pi * np.sin(2 * np.pi * x) * exp2 / n)
    return grad

def grad_sphere(x):
    return 2 * x

global_min = {
    "rastrigin": np.zeros(5),
    "rosenbrock": np.ones(5),
    "griewank": np.zeros(5),
    "ackley": np.zeros(5),
    "sphere": np.zeros(5),
}

# Критерии остановки
def stop_criteria_1(x_prev, x_curr, grad, tol):
    return np.linalg.norm(x_curr - x_prev) < tol

def stop_criteria_2(f_prev, f_curr, grad, tol):
    return abs(f_curr - f_prev) < tol

def stop_criteria_3(x_curr, grad, tol):
    return np.linalg.norm(grad) < tol

def conjugate_gradient(f, grad_f, x0, stop_criteria, tol=1e-6, max_iter=1000):
    x = np.array(x0)
    g = grad_f(x)
    d = -g
    f_prev = f(x)
    iterations = 0
    for _ in range(max_iter):
        iterations += 1
```

```
        alpha = -np.dot(g, d) / (np.dot(d, grad_f(x + 1e-6 * d)) + 1e-10)
        x_new = x + alpha * d
        g_new = grad_f(x_new)
        beta = np.dot(g_new, g_new) / (np.dot(g, g) + 1e-10)
        d = -g_new + beta * d
        f_curr = f(x_new)

        if stop_criteria(x, x_new, g_new, tol) or stop_criteria(f_prev, f_curr,
g_new, tol):
            return x_new, iterations
        x = x_new
        g = g_new
        f_prev = f_curr
    return x, iterations
(def conjugate_gradient(f, grad_f, x0, stop_criteria, tol=1e-6, max_iter=1000,
clip_range=(-5, 5)):
    x = np.array(x0)
    g = grad_f(x)
    d = -g
    f_prev = f(x)
    iterations = 0
    for _ in range(max_iter):
        iterations += 1

            alpha = adaptive_step(grad_f, x, d)

        x_new = x + alpha * d
        x_new = np.clip(x_new, clip_range[0], clip_range[1])
 g_new = grad_f(x_new)
        beta = np.dot(g_new, g_new) / (np.dot(g, g) + 1e-10)
        d = -g_new + beta * d
        f_curr = f(x_new))
functions = [rastrigin, rosenbrock, griewank, ackley, sphere]
grads = [grad_rastrigin, grad_rosenbrock, grad_griewank, grad_ackley,
grad_sphere]
criteria = [stop_criteria_1, stop_criteria_2, stop_criteria_3]

for func, grad_func in zip(functions, grads):
    func_name = func.__name__
```

```
    x0 = np.random.uniform(-5, 5, size=5)
    for crit in criteria:
        result, iters = conjugate_gradient(func, grad_func, x0, crit)
        error = np.linalg.norm(result - global_min[func_name])
print(f"Function: {func_name}, Criteria: {crit.__name__}, Iterations: {iters},
Error: {error}, Result: {result}")
```