НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи
з дисципліни «АЛГОРИТМИ І СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 1.МАТЕМАТИЧНІ АЛГОРИТМИ»
на тему
"Чисельні методи оптимізації"

Виконав:
студент групи КМ-03
Шаповалов Г. Г.

Перевірила:
Асистент кафедри ПМА
Ковальчук-Химюк Л. О.

Київ — 2023

Зміст

Вступ

Метою даної роботи є вивчити методи нульового, першого й другого порядку, практичну мінімізацію функцій багатьох змінних з використанням СКМ.

Основна частина

Варіант 28

Функція	Початковий вектор	Точка мінімуму	Значення
$x_1{}^2 + 16x_2^2$	$[2;\ 2]$	[0; 0]	0

Метод мінімізації функції – метод Нелдера-Міда [4]

Теоретичні відомості

Метод Нелдера-Міда [4] – це чисельний метод для пошуку мінімуму або максимуму функції в багатовимірному просторі. Він є корисним для вирішення нелінійних задач оптимізації, коли немає можливості використовувати похідні функції. Метод Нелдера-Міда використовує симплекс (політоп з вершинами) у просторі n+1 вимірів для апроксимації локального оптимуму цільової функції з p змінними. Важливо, що функція повинна бути плавною та унімодальною.

Метод включає в себе набір тестових точок, які розташовані у формі симплексу. Процес полягає у використанні значень функції на цих точках для екстраполяції поведінки функції та знаходження нових тестових точок. Найпростіший спосіб - замінити найгіршу точку точкою, відображеною відносно центру решти точок. Якщо нова точка краща за поточну найкращу, то проводиться експоненційне розтягнення симплексу в цьому напрямку. В іншому випадку, якщо нова точка не поліпшує значення функції значно, то симплекс стискається до кращої точки.

Хоча цей метод вимагає менше оцінок цільової функції порівняно з іншими методами оптимізації, загальна кількість ітерацій може бути великою. Метод Нелдера-Міда є евристичним і може збігати до нестаціонарних точок, тому існують альтернативні методи, які можуть вирішити цю проблему.

Похибка результатів

Похибка буде рахуватись за формулою : $\varepsilon = |f(x_k) - f(x^*)|$

Чисельне представлення похибок наведено у висновках

Алгоритми розв’язання

Оптимізація функції $x_1{}^2 + 16x_2^2$ реалізована на мові Python та Octave.

Імпортуємо бібліотеки Oct2py [1], numpy [2], scipy [3]

1. Python-реалізація:
    a. Використовується бібліотека `oct2py`, яка дозволяє взаємодіяти між Python та Octave.
    b. Оптимізація функції, заданої у вигляді `objective_function`, використовуючи метод `fmin` з бібліотеки `scipy.optimize`.
    c. Перевірка розмірності початкового вектора та обчислення оптимального вектора та значення функції.
    d. Розрахунок похибки порівняння отриманого значення функції з реальним значенням.
2. Octave-реалізація:
    - Оголошення функції `objective_function` в мові Octave.
    - Передача початкового вектора з Python до Octave.
    - Використання функції `fminunc` (нелінійна оптимізація) для пошуку оптимального вектора та значення функції в Octave.
    - Повернення результатів оптимізації до Python.
    - Розрахунок похибки порівняння отриманого значення функції з реальним значенням.

Обидва блоки коду розділені та виводять результати для порівняння між Python та Octave. Отримані значення, оптимальні вектори та похибка виводяться на екран.

У Python допоміжною функцією є fmin із бібліотеки scipy.

У Octave вбудована функція fminunc.

Обидві функції пошуку мінімуму використовують метод Нелдера-Міда.

Висновки

У результаті пошуку мінімуму функції $x_1{}^2 + 16x_2^2$ методом Нелдера-Міда було досягнуто таких результатів:

```
Результати Python
Оптимальний вектор: [-1.9104056e-05  1.1236954e-05]
Мінімальне значення функції: 2.3852711151642794e-09
Оцінка похибки: 2.3852711151642794e-09

Результати Octave
Оптимальний вектор: [[-4.79936554e-09 -4.74280417e-09]]
Мінімальне значення функції: 2.2814110271052257e-18
Оцінка похибки: 2.2814110271052257e-18
```

За результатами використання функцій бачимо, що Octave справився набагато краще. Оскільки отримано похибку меншу, ніж на Python.

Відповіді на контрольні питання

1. Узагальнена характеристика методів нульового порядку:
    A. Метод прямого пошуку: цей метод ґрунтується на послідовному випробуванні точок вздовж координатних осей або відомих напрямків. Це простий підхід, але його збіжність може бути повільною.
    B. Метод деформованого многогранника: цей метод використовує многогранник для наближення області рішення та потім деформує його, щоб зменшити область, де знаходиться мінімум. Цей метод може бути ефективним для нелінійних задач оптимізації, особливо тоді, коли область мінімуму заздалегідь невідома.
    C. Метод координат, що обертаються: цей метод працює вздовж координатних осей, обертаючи їх під час оптимізації. Він дозволяє швидко збігатися до мінімуму, але може бути менш ефективним у високовимірних просторах.
    D. Метод паралельних дотичних: цей метод використовує ітераційний процес, який включає у себе пошук та оновлення точок. Він може бути ефективним для оптимізації, особливо у випадках з обмеженнями.
2. Загальна характеристика градієнтних методів в оптимізації:
    A. Метод найшвидшого спуску: цей метод використовує градієнт (перший порядок похідної) для визначення напрямку найшвидшого спуску в поточній точці. Метою є знаходження точки, в якій градієнт функції дорівнює нулю. Однак збіжність може бути повільшою в узких долинах функції.
    B. Метод спряжених градієнтів: цей метод прагне знайти точку, в якій градієнти функції перпендикулярні один одному. Це може призводити до швидкої збіжності, особливо для квадратичних функцій.
3. Метод Ньютона та його збіжність: метод Ньютона використовує другий порядок похідної для визначення напрямку спуску. Властивість методу Ньютона полягає в тому, що збіжність може бути квадратичною, що означає, що кожна ітерація може подвоювати кількість правильних цифр у відповіді.
4. Мінімізація ярових функцій часто ускладнена через наявність різних локальних мінімумів, що можуть сповільнити швидкість збіжності методів першого порядку. Методи другого порядку можуть бути ефективнішими, оскільки враховують кривизну функції та швидше досягають глобальних мінімумів.

Перелік посилань

1. Oct2py - https://pypi.org/project/oct2py/
2. Numpy - https://numpy.org
3. Scipy - https://scipy.org
4. Метод Нелдера-Міда - https://elearn.nubip.edu.ua/mod/book/tool/print/index.php?id=45358&chapterid=4439

Додаток А – Код програми

Вміст файлу main.py :

```
from oct2py import octave
import numpy as np
from scipy.optimize import fmin


def optimize_my_function(initial_vector):
    if len(initial_vector) != 2:
        raise ValueError('Початковий вектор повинен мати розмірність 2')

    def objective_function(x):
        x1, x2 = x
        return x1**2 + 16*x2**2

    optimal_vector = fmin(objective_function, initial_vector, disp=False)

    return optimal_vector, objective_function(optimal_vector)

real_optimal_value = 0  #  Реальні значення

initial_vector = np.array([2, 2])
optimal_vector, optimal_value = optimize_my_function(initial_vector)
error = abs(optimal_value - real_optimal_value)  #  Обчислити похибку для
мінімального значення функції

print('\n\n\nШаповалов Г. Г. Варіант 28 Лаб 4\n')
print('Результати Python')
print('Оптимальний вектор:', optimal_vector)
print('Мінімальне значення функції:', optimal_value)
print('Оцінка похибки:', error)


def optimize_with_octave(initial_vector):
    if len(initial_vector) != 2:
        raise ValueError('Початковий вектор повинен мати розмірність 2')

    # Визначаємо функцію у форматі Octave
    octave.eval("function y = objective_function(x) y = 10 * (x(1) - sin(x(2)) )^2
+ 0.1 * x(2)^2; end")

    # Передаємо початковий вектор до Octave
    x0 = initial_vector.tolist()
    octave.push('initial_vector', x0)

    # Використовуємо функцію fminunc для оптимізації
    octave.eval("options = optimset('fminunc');")
    octave.eval("options.Display = 'off';")
```

```
    octave.eval("[optimal_vector, optimal_value] = fminunc(@objective_function,
initial_vector, options);")

    # Отримуємо результати оптимізації
    optimal_vector = octave.pull('optimal_vector')
    optimal_value = octave.pull('optimal_value')

    return optimal_vector, optimal_value

real_optimal_value = 0  #  Реальні значення

initial_vector = np.array([2, 2])
optimal_vector, optimal_value = optimize_with_octave(initial_vector)
error = abs(optimal_value - real_optimal_value)  #  Обчислити похибку для
мінімального значення функції

print('\nРезультати Octave')
print('Оптимальний вектор:', optimal_vector)
print('Мінімальне значення функції:', optimal_value)
print('Оцінка похибки:', error)
```