МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Д. О. Василенко

МАТЕМАТИЧНІ МЕТОДИ ОПТИМІЗАЦІЇ
ПРАКТИКУМ

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Інформаційна та комунікаційна радіоінженерія»
за спеціальністю 172 «Електронні комунікації та радіотехніка»

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2023

Рецензент *Сушко І.О.,* к.т.н., доцент кафедри ПРЕ

Відповідальний редактор *Мартинюк С.Є.,* к.т.н, доцент, в.о. зав. кафедри РІ

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 07 від 27.04.2023 р.) за поданням Вченої ради Радіотехнічного факультету (протокол № 04/2023 від 27.03.2023 р.)

Навчальний посібник призначений для підготовки фахівців спеціальності 172 «Електронні комунікації та радіотехніка». Методичний посібник містить посилання на необхідні теоретичні відомості та завдання для аудиторної роботи студентів на практикумах з курсу «Математичні методи оптимізації». Метою практикумів є набуття знань про межі застосування алгоритмів, що досліджуються, залежність результатів оптимізації від параметрів алгоритмів, та набуття практичного досвіду реалізації окремих частин методів оптимізації у Matlab. Зокрема, вивчаються такі методи: метод дихотомії, метод золотого перерізу, метод покоординатного спуску, метод пошуку по зразку, алгоритм Нелдера-Міда, метод найшвидшого градієнтного спуску, метод спряжених градієнтів, метод Левенберга-Марквардта, метод ДФП, генетичний алгоритм, алгоритм бджолиного рою.

ЗМІСТ

ВСТУП

Метою практикумів з курсу «Математичні методи оптимізації» є вивчення впливу параметрів методів оптимізації на точність і швидкість знаходження оптимумів тестових функцій. Класичні методи і алгоритми оптимізації (метод дихотомії, метод золотого перерізу, метод покоординатного спуску, метод пошуку по зразку, алгоритм Нелдера-Міда, метод найшвидшого градієнтного спуску, метод спряжених градієнтів, метод Левенберга-Марквардта, метод ДФП), що розглядаються у практикумах 1 – 5, тестуються із застосуванням квадратичної функції, функції Booth та функції Розенброка. Алгоритми глобальної оптимізації (практикум 6 та 7) тестуються із використанням функції Розенброка, функції Растрігіна, Easom та Ackley. На додачу до тестових функцій додатково розглядається застосування методів оптимізації в простих інженерних задачах: оптимізація фільтру та знаходження параметрів моделі НВЧ резистора по відомим результатам вимірювань.

У даному посібнику міститься 7 практикумів. Практикуми 1 – 5 розраховані на 2 академічні години. Практикуми 6 та 7, що присвячені глобальним методам оптимізації розраховані на 4 академічні години кожний.

ПРАКТИЧНЕ ЗАНЯТТЯ 1. СКЛАДНІ ФУНКЦІЇ ДВОХ ЗМІННИХ ТА ЇХ ВІЗУАЛІЗАЦІЯ. ЧИСЛОВЕ ОБЧИСЛЕННЯ ПОХІДНИХ ФУНКЦІЙ

Мета

Знайомство із основними засобами візуалізації функції двох змінних у Matlab. Дослідження основних функцій, що використовуються для тестування алгоритмів оптимізації у подальших практикумах.

Домашнє завдання

1. Пригадати (вивчити) виведення графіків за допомогою функції *plot*, основні команди керування (*if*, *while*, *case*, …), основні способи визначення змінних ([1]).

Порядок виконання роботи

1. Ознайомитися із програмою Matlab *visualization.m*, в якій представлено базові функції для візуалізації функцій двох змінних у 3D. Розібратися із налаштуваннями команд: *meshgrid*, *linspace*, *shading*, *surf*, *surfc*.

2. Ознайомитися із програмою Matlab *visualization_contour.m*, в якій представлено базові функції для візуалізації функцій двох змінних у 2D. Розібратися із налаштуваннями команд: *pcolor*, *colormap*, *contour*.

3. Ознайомитися із складними функціями, які використовуються для тестування алгоритмів оптимізації: квадратична функція, функція Растрігіна, функція Розенброка, функція Ackley, Easom, Booth. Математичний запис функцій представлено в Табл.1.1. Звернути увагу на положення мінімуму функції (Табл.1.2) і поведінку функції в околі мінімуму. Повний перелік функцій, що можна використовувати для тестування алгоритмів оптимізації можна знайти у [2].

4. Ознайомитися із програмою Matlab *complex_functions.m*, яка виводить 2D і 3D зображення тестових функцій.

5. Змінити програму *complex_functions.m* так, щоб детально розглянути область мінімуму складних функції.

6. Для функції $x^3 + x^{1/2}$ обчислити значення похідної в точці $x = 3$ для значення Δx =1; 0,1; 0,01 використовуючи формули з інтерполюванням вперед (1.1) і центрально різницеву формулу (1.2).

$$\frac{df}{dx} = \frac{f(x+\Delta x)-f(x)}{\Delta x} \quad (1.1),$$

$$\frac{df}{dx} = \frac{f(x+\Delta x)-f(x-\Delta x\ )}{2\Delta x} \quad (1.2).$$

Оцінити похибку кожної з формул.

Таблиця 1.1 Математичний запис тестових функцій

Функція	Математичний запис функції
Rastrigin	$f(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i)]$
Ackley	$f(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(cx_i)\right) + a + \exp(1)$ a=20, b=0.2, $c = 2\pi$
Sum square	$f(x) = \sum_{i=1}^{n} ix_i^2$
Booth	$f(x) = \sum_{i=1}^{n}(x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
Rosenbrock	$f(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right) + (1 - x_i)^2\right]$
Easom	$f(x_1, x_2) = -\cos(x_1)\cos(x_2)\, exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$

Таблиця 1.2 Значення координат мінімуму і значення фугкції в точці мінімуму для тестових функцій

Функція	Координати мінімуму	Значення ф-ії в мінімумі
Rastrigin	(0, 0)	0
Ackley	(0, 0)	0
Sum square	(0, 0)	0
Booth	(1, 3)	0
Rosenbrock	(1, 1)	0
Easom	(pi, pi)	-1

Контрольні запитання

1. Відмітити особливості таких функцій: квадратична функція, функція Растрігіна, функція Розенброка, функція Ackley, Easom, Booth.

2. Яке мінімальне значення і положення мінімуму функцій з п.1?

Література

1. Mathworks. Get Started with MATLAB. – Режим доступу: https://www.mathworks.com/help/matlab/getting-started-with-matlab.html . – Заголовок з екрану.

2. Jamil, M., & Yang, X. (2013). A literature survey of benchmark functions for global optimisation problems. *Int. J. Math. Model. Numer. Optimisation, 4*, 150-194.

ПРАКТИЧНЕ ЗАНЯТТЯ №2. МЕТОДИ ОДНОМІРНОЇ ОПТИМІЗАЦІЇ

Мета

Метою практикуму є вивчення методів одномірного пошуку: методу дихотомії, методу золотого перерізу, методу Фібоначчі, методу з квадратичною апроксимацією функції, порівняння швидкості їх роботи, визначення особливостей роботи методів.

Домашнє завдання

1. Ознайомитися із теоретичними засадами методів одномірного пошуку: методу пасивного пошуку, методу дихотомії, методу золотого перерізу, методу Фібоначчі, методу з квадратичною апроксимацією функції ([1]: с.248-279, [2]: с.101-118).

2. Ознайомитися із реалізацією методу дихотомії у програмі Matlab *dichotomous.m*. Визначити, яка змінна відповідає за умову зупинки алгоритму по величині інтервалу, а яка відповідає за досліджуючий крок.

3. Ознайомитися із реалізацією методу золотого перерізу у програмі Matlab *goldmin.m*. Визначити, яка змінна відповідає за умову зупинки алгоритму по величині інтервалу.

4. Ознайомитися із реалізацією методу з квадратичною апроксимацією у програмі Matlab *quadr_approx.m*. Пояснити, яка ідеологія кожної наступної ітерації алгоритму.

Порядок виконання роботи

1. Використовуючи *dichotomous.m* провести оптимізацію функцій із програми *getFunction.m* із стартовим інтервалом і значенням *Epsilon*, що наведений у Табл. 2.1. Тип функції задається у програмі з *dichotomous.m* за допомогою змінної *functionNumber*. Стартове значення початку і кінця інтервалу задається змінними *xLeft_start* та *xRight_start*. Значення *delta* встановити рівним 0.001. Результати розрахунку занести у Табл. 2.2. Значення X точки мінімуму, значення функції в точці мінімуму, кількість викликів функції, кількість ітерацій виводяться у консоль MATLAB після завершення оптимізації і у змінні *x_opt*, *f_at_x_opt*, *function_eval_number_max*, *number_of_iterations*, відповідно.

Таблиця 2.1 Перелік експериментів в практикумі

Номер експерименту	Номер функції	xStart	xEnd	ε
1	1	-2	4	0.01
2	1	-2	4	0.05
3	1	-2	4	0.1
4	1	-2	4	0.5
5	2	-2	4	0.01
6	2	-2	4	0.1
7	4	2.7	7.5	0.01
8	4	2.9	7.5	0.01
9	5	3.1	20.4	0.01
10	6	-10	10	0.01
11	6	-8	10	0.01

Таблиця 2.2 Рекомендована форма запису результатів практикуму

Номер експерименту	Значення X точки мінімуму	Значення функції в точці мінімуму	Кількість викликів функції	Кількість ітерацій
1				
2				
…				

2. На прикладі функції №1 пояснити, що трапляється із методом дихотомії, якщо *delta* і *Epsilon* однакові.

3. Спостерігати, як змінюється кількість викликів функцій при зміні *Epsilon*. На прикладі функції №1 та №2 пояснити, чому при однакових межах оптимізації функцій, але різних функціях при однаковому *Epsilon* кількість викликів функцій однакова.

4. Пояснити, чому для функції №4 (експеримент 7 і 8) метод дихотомії не збігається до глобального оптимуму, а при незначній зміні лівої межі збігається в інший локальний мінімум.

5. Пояснити, чому для функції №6 (експеримент 10 і 11) метод дихотомії збігається до глобального оптимуму, а при незначній зміні лівої межі не збігається взагалі.

6. Використовуючи *goldmin.m* провести оптимізацію функцій із програми *getFunction.m* із стартовим інтервалом і значенням *Epsilon*, що наведений у Табл. 2.1. Тип функції задається у програмі з *goldmin.m* за допомогою змінної *functionNumber*. Стартове значення початку і кінця інтервалу задається змінними *xLeft_start* та *xRight_start*. Результати розрахунку занести у таблицю ідентичну Таблиці 2.2. Значення X точки мінімуму, значення функції в точці мінімуму, кількість викликів функції, кількість ітерацій виводяться у консоль Matlab після завершення оптимізації і у змінні *x_opt*, *f_at_x_opt*, *function_eval_number_max*, *number_of_iterations*, відповідно.

7. Порівняти швидкість оптимізації (кількість викликів функції і кількість ітерацій) функції №1 та №2 методом золотого перерізу і методом дихотомії. Пояснити отриманий результат.

8. Пояснити, чому для функції №4 - №6 (експеримент 7 – 11) метод золотого перерізу не збігається до глобального оптимуму, а при зміні меж збігається в інший локальний мінімум або взагалі не збігається.

9. Використовуючи *quadr_approx.m* провести оптимізацію функцій із програми *getFunction.m* із стартовим інтервалом і значенням *Epsilon*, що наведений у Табл. 2.1 для функції №1 та №2. Тип функції задається у програмі *quadr_approx.m* за допомогою змінної *functionNumber*. Стартове значення початку і кінця інтервалу задається змінними *xLeft_start* та *xRight_start*. Результати розрахунку занести у таблицю ідентичну Табл. 2.2. Значення X точки мінімуму, значення функції в точці мінімуму, кількість викликів функції, кількість ітерацій виводяться у консоль Matlab після завершення оптимізації і у змінні *x_opt*, *f_at_x_opt*, *function_eval_number_max*, *counter*, відповідно.

10. Спостерігати, що оптимізація функції №1 проходить за одну ітерацію. Пояснити чому?

11. Порівняти швидкість оптимізації методом квадратичної апроксимації і методом золотого перерізу для функції №2 (експеримент 5 і 6).

12. Провести оптимізацію функції №2 в межах -5..4. Пояснити, чому алгоритм квадратичної апроксимації помиляється.

13. Модифікувати *goldmin.m* так, щоб проводити оптимізацію методом Фібоначчі.

Контрольні запитання

1. Алгоритм роботи методу дихотомії.
2. Переваги і недоліки методу дихотомії.
3. Чим визначається в методі дихотомії кількість ітерацій для досягнення оптимальної точки?
4. Алгоритм роботи методу золотого перерізу.
5. Переваги і недоліки методу золотого перерізу. Чим визначається кількість ітерацій для досягнення оптимальної точки?
6. Порівняти метод золотого перерізу і метод дихотомії.
7. Алгоритм роботи методу Фібоначчі.
8. Метод квадратичної оптимізації. Математичне підґрунтя методу. Алгоритм роботи. Переваги і недоліки методу.

Література

1. Rao Singiresu S. Engineering Optimization: Theory and Practice / Singiresu S. Rao. - New Jersey, USA: John Wiley & Sons, 2009 – 813 p.
2. Mohamed Bakr. 2013. Nonlinear Optimization in Electrical Engineering with Applications in MATLAB. Institution of Engineering and Technology.

ПРАКТИЧНЕ ЗАНЯТТЯ №3. МЕТОДИ ОПТИМІЗАЦІЇ НУЛЬОВОГО ПОРЯДКУ

Мета

Метою практикуму є вивчення методів нульового порядку пошуку мінімуму функції багатьох змінних: методу покоординатного спуску, методу пошуку по зразку, алгоритму Нелдера-Міда, порівняння швидкості їх роботи на прикладі квадратичної функції, Booth function та функції Розенброка, визначення особливостей роботи методів.

Домашнє завдання

1. Ознайомитися із теоретичними засадами методів нульового порядку для оптимізації функції багатьох змінних: методу покоординатного спуску, методу пошуку по зразку, симплексного методу і алгоритму Нелдера -Міда ([1]: с.314-334, [2]: с.131-151).

2. Ознайомитися із реалізацією методу покордината спуску у програмі Matlab *gauss_sumsqu.m*. Визначити, який алгоритм використовується для одномірного пошуку і які він має параметри. Визначити, яка змінна в алгоритмі покоординатного спуску відповідає за умову зупинки алгоритму по величині інтервалу, а яка відповідає за досліджуючий крок.

3. Ознайомитися із реалізацією методу пошуку по зразку у програмі Matlab *pattern_search_sumsqu.m*. Визначити, який алгоритм використовується для одномірного пошуку і які він має параметри. Визначити, яка змінна в алгоритмі відповідає за умову зупинки алгоритму по величині інтервалу, а яка відповідає за початковий досліджуючий крок. Який алгоритм зміни досліджуючого кроку реалізовано в програмі?

4. Ознайомитися із реалізацією алгоритму Нелдера-Міда у програмі Matlab *simplex_sumsqu.m*. Визначити, яка умова в алгоритмі Нелдера-Міда відповідає за умову зупинки алгоритму. Які змінні регулюють відбиття, розтягування і редукцію симплексу.

Порядок виконання роботи

1. Використовуючи *gauss_sumsqu.m* провести оптимізацію квадратичної функції *sumsqu.m* із такими початковими умовами:

```
StartPoint=[-10  10]';
EpsilonXchange=0.01;
Epsilon=0.01;
LambdaMax=1;
Tolerance=0.01;
```

2. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні максимально можливого кроку одномірного пошуку *LambdaMax* (1…10). Необхідні значення виводяться на графіках. За результатами дослідження заповнити табл. 3.1 і зробити висновки.

3. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні досліджуючого кроку *Epsilon*, мінімального розміру відрізка оптимізації при одномірній оптимізації *Tolerance* і умови зупинки алгоритму по зміні аргументу *EpsilonXchange*. Перевірити такі значення: 0.1; 0.01; 0.001. За результатами дослідження заповнити табл. 3.1 і зробити висновки.

4. Використовуючи *pattern_search_sumsqu.m* провести оптимізацію квадратичної функції *sumsqu.m* із такими початковими умовами:

```
OldPoint=[-10 10]'
StepStart=0.1;
LambdaMax=1;
Tolerance=0.01;
MinimumDistance=0.01;
```

5. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні максимально можливого кроку одномірного пошуку *LambdaMax* (1…150). Необхідні значення виводяться на графіках. За результатами дослідження заповнити табл. 3.1 і зробити висновки.

6. Порівняти кількість викликів функції при точності 0,01 при оптимізації функції *'sumsqu'* методом покоординатного спуску і методом пошуку по зразку при оптимальному значенні *LambdaMax* для кожного із методів, отриманих в пунктах 2 і 5, відповідно.

7. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення

функції в оптимальній точці, шлях оптимізації на площині XY) при зміні досліджуючого кроку *StepStart*, мінімального розміру відрізка оптимізації при одномірній оптимізації *Tolerance* і умови зупинки алгоритму по зміні аргументу *MinimumDistance*. Перевірити такі значення: 0,1; 0,01; 0,001. За результатами дослідження заповнити таблицю 3.1 і зробити висновки.

8. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні початкової точки оптимізації ([-10 10] [-5 5] [-1 1]). За результатами дослідження заповнити табл. 3.1 і зробити висновки.

9. Використовуючи *simplex_sumsqu.m* провести оптимізацію квадратичної функції *sumsqu.m* із такими початковими умовами:

```
ReflectionCoefficient=1.0;
ExpansionCoefficient=0.5;
ContractionCoefficient=0.6;
SimplexSizeMin=0.01;
IterationMax=100;
StartPoint=[-10 10]';
SimplexStartSize=1;
```

10. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні початкового розміру симплексу *SimplexStartSize* (1…8). Необхідні значення виводяться на графіках. За результатами дослідження в даному пункті і пунктах 11 - 14 заповнити табл. 3.1 і зробити висновки.

11. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні коефіцієнта редукції *ContractionCoefficient* (0.3…0.7).

12. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні коефіцієнта експансії *ExpansionCoefficient* (0.3…0.7).

13. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні мінімально можливого розміру симплекса (0.1, 0.001).

14. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні початкової точки оптимізації. ([-10 10] [-5 5] [-1 1])

15. Порівняти швидкість оптимізації квадратичної функції і Booth function методом Гауса при таких налаштуваннях алгоритму (використовуємо програми *gauss_sumsqu.m* та *gauss_booth.m*):

```
StartPoint=[-10  10]';
EpsilonXchange=0.01;
Epsilon=0.01;
LambdaMax=10;
Tolerance=0.01;
```

16. Порівняти швидкість оптимізації квадратичної функції і Booth function методом пошуку по зразку при таких налаштуваннях алгоритму (використовуємо програми *pattern_search_sumsqu.m* та *pattern_search_booth.m*):

```
OldPoint=[-10 10]'
StepStart=1;
LambdaMax=150;
Tolerance=0.01;
MinimumDistance=0.01;
```

17. Порівняти швидкість оптимізації квадратичної функції і Booth function методом Нелдера-Міда при таких налаштуваннях алгоритму (використовуємо програми *simplex_sumsqu.m* та *simplex_booth.m*):

```
ReflectionCoefficient=1.0;
ExpansionCoefficient=0.5;
ContractionCoefficient=0.3;
SimplexSizeMin=0.01;
IterationMax=100;
StartPoint=[-10 10]';
SimplexStartSize=4;
```

18. Дослідити і зробити висновки, як проходить пошук мінімуму функції Розенброка методом Гауса, пошуку по зразку, методом Нелдера-Міда (використовуємо програми *gauss_rosen.m*, *simplex_rosen.m* та *pattern_search_rosen.m*). Використати оптимальні параметри, знайдені при дослідженні функції *sumsqu*. Перевірити такі стартові точки пошуку: [0 -

1.5], [0 1.5] [-1.5 1.5]. За результатами дослідження заповнити таблицю 3.1 і зробити висновки.

Таблиця 3.1 Рекомендована форма запису результатів практикуму

Змінна, що досліджується	Координати точки мінімуму	Значення функції в точці мінімуму	Кількість викликів функції	Кількість ітерацій
Значення змінної				
…				

Контрольні запитання

1. Алгоритм роботи методу покоординатного спуску. Умова зупинки алгоритму.
2. Переваги і недоліки методу покоординатного спуску (порівняти ефективність роботи для квадратичної функції, для функції Розенброка і функції Booth).
3. Алгоритм роботи методу пошуку по зразку. Переваги і недоліки методу.
4. Які проблеми методу покоординатного спуску вирішує метод пошуку по зразку (порівняти ефективність роботи на прикладі квадратичної функції, функції Розенброка і функції Booth)?
5. Метод пошуку по симплексу. Поняття симплексу. Основні положення алгоритму Нелдера-Міда і його відмінність від класичного методу пошуку по симплексу.
6. Послідовність роботи алгоритму Нелдера-Міда. Умова зупинки алгоритму. Порівняти ефективність роботи алгоритму Нелдера-Міда і методу пошуку по зразку та методу по координатного спуску.

Література

1. Rao Singiresu S. Engineering Optimization: Theory and Practice / Singiresu S. Rao. - New Jersey, USA: John Wiley & Sons, 2009 – 813 p.
2. Mohamed Bakr. 2013. Nonlinear Optimization in Electrical Engineering with Applications in MATLAB. Institution of Engineering and Technology.

ПРАКТИЧНЕ ЗАНЯТТЯ №4. МЕТОДИ ОПТИМІЗАЦІЇ ПЕРШОГО ПОРЯДКУ

Мета

Метою практикуму є вивчення методів першого порядку пошуку мінімуму функції багатьох змінних: методу найшвидшого градієнтного спуску, методу спряжених градієнтів, порівняння швидкості їх роботи на прикладі квадратичної функції, Booth function та функції Розенброка, визначення особливостей роботи методів.

Домашнє завдання

1. Ознайомитися із теоретичними засадами методів першого порядку для оптимізації функції багатьох змінних: методу найшвидшого градієнтного спуску, методу спряжених градієнтів ([1]: с.335-345, [2]: с.153-164).
2. Ознайомитися із реалізацією методу найшвидшого градієнтного спуску в програмі Matlab *steepest_sumsqu.m*. Визначити, який алгоритм використовується для одномірного пошуку і які він має параметри. Визначити, яка змінна в програмі відповідає за умову зупинки алгоритму.
3. Ознайомитися із реалізацією алгоритму методу спряжених градієнтів в програмі Matlab *con_gradient_sumsqu.m.* Визначити, який алгоритм використовується для одномірного пошуку і які він має параметри. Визначити, яка змінна в програмі відповідає за умову зупинки алгоритму.

Порядок виконання роботи

1. Використовуючи програму *steepest_sumsqu.m* провести оптимізацію квадратичної функції *sumsqu.m* із такими початковими умовами:

```
function_name='sumsqu'
NumberOfParameters=2;
StartPoint=[-10  10]'
Tolerance=0.001;
Epsilon=0.001;
EpsilonGradient=0.1;
LambdaMax=1;
MinimumDistance=0.01;
norm_Gradient=0;
stop_on_Gradient_size=1;
```

Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні максимально можливого кроку одномірного пошуку *LambdaMax* (0.1…1…10). Необхідні значення виводяться на графіках. За результатами дослідження заповнити табл. 4.1 і зробити висновки.

2. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні досліджуючого кроку градієнту *Epsilon*, мінімального розміру відрізка оптимізації при одномірній оптимізації *Tolerance* і умови зупинки алгоритму по зміні аргументу *MinimumDistance*. Перевірити такі значення: 0.1; 0.01; 0.001; 0.0001. За результатами дослідження заповнити табл. 4.1 і зробити висновки.

3. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні умови зупинки алгоритму по величині градієнта *EpsilonGradient*. За результатами дослідження заповнити табл. 4.1 і зробити висновки.

4. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY), якщо зупинка по величині градієнта ввімкнена і якщо вимкнена (змінна stop_on_Gradient_size, 1 – ввімкнено, 0 - вимкнено).

5. Для знайдених оптимальних значень *Epsilon*, *Tolerance*, *MinimumDistance*, *EpsilonGradient* встановити у *steepest_sumsqu.m*

```
function_name='sumsqu_coeff_1'
```

і пояснити відмінність у швидкості і в шляху оптимізації.

6. Використовуючи програму *con_gradient_sumsqu.m* провести оптимізацію квадратичної функції sumsqu.m із такими початковими умовами:

```
StartPoint=[-10  10]'
Epsilon=0.001;
LambdaMax=1;
Tolerance=0.001;
function_name='sumsqu'
```

```
IterationMaxNumber=100
EpsilonGradient=0.1;
StepNormMin=0.01;
```

7. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні максимально можливого кроку одномірного пошуку *LambdaMax* (0.1…10). За результатами дослідження заповнити табл. 4.1 і зробити висновки.

8. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні досліджуючого кроку *Epsilon*, мінімального розміру відрізка оптимізації при одномірній оптимізації *Tolerance* і умови зупинки алгоритму по зміні аргументу *StepNormMin*. Перевірити такі значення: 0.1; 0.01; 0.001; 0.0001. За результатами дослідження заповнити таблицю 4.1 і зробити висновки.

9. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні умови зупинки алгоритму по величині градієнта *EpsilonGradient*. За результатами дослідження заповнити таблицю 4.1 і зробити висновки.

10. Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні початкової точки оптимізації.

11. Для знайденого оптимального значення *Epsilon*, *Tolerance*, *MinimumDistance*, *EpsilonGradient* встановити

```
function_name='sumsqu_coeff_1'
```

і пояснити відмінність у швидкості і в шляху оптимізації.

12. Порівняти кількість викликів функції при точності 0.01 при оптимізації методом найшвидшого градієнтного спуску і методом спряжених градієнтів при оптимальному значенні *LambdaMax*, *Epsilon*, *Tolerance* для кожного із методів.

13. Порівняти швидкість оптимізації квадратичної функції і Booth function методом найшвидшого градієнтного спуска при таких налаштуваннях алгоритму:

```
StartPoint=[-10  10]'
Tolerance=0.01;
Epsilon=0.01;
EpsilonGradient=0.1;
LambdaMax=10;
MinimumDistance=0.01;
```

14. Порівняти швидкість оптимізації квадратичної функції і Booth function методом спряжених градієнтів при таких налаштуваннях алгоритму:

```
NumberOfParameters=2;
StartPoint=[-10  10]'
Epsilon=0.001;
LambdaMax=1;
Tolerance=0.001;
function_name='booth'
IterationMaxNumber=100
EpsilonGradient=0.1;
StepNormMin=0.01;
```

15. Записати цільову функцію для такої задачі. Зроблені вимірювання коефіцієнта відбиття S11 на частотах 100 – 1000 МГц із кроком 100 МГц для схеми на малюнку. Необхідно розв'язати задачу оптимізації, щоб знайти, який саме резистор і ємність були в схемі під час вимірювання.

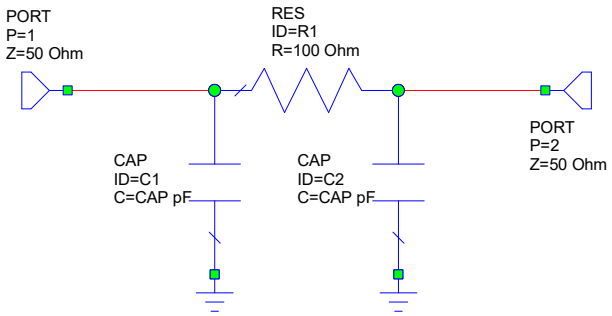

Рис.4.1. Модель резистора

16. Порівняти швидкість оптимізації поставленої задачі методом найшвидшого градієнтного спуску (файли *steepest_resistor.m* та *con_gradient_resistor.m*). Визначити значення параметрів методів оптимізації, які забезпечують найменшу кількість розрахунків цільової функції.

17. Дослідити і зробити висновки, як проходить пошук мінімуму функції Розенброка методом найшвидшого градієнтного спуску і методом спряжених градієнтів (використовуємо програми steepest*_rosen.m*, *con_gradient_rosen.m*). Провести дослідження впливу параметрів алгоритмів на швидкість оптимізації і її кінцевий результат. Перевірити, зокрема, такі стартові точки пошуку: [0 -1.5], [0 1.5] [-1.5 1.5]. За результатами дослідження заповнити табл. 4.1 і зробити висновки.

Таблиця 4.1 Рекомендована форма запису результатів практикуму

Змінна, що досліджується	Координати точки мінімуму	Значення функції в точці мінімуму	Кількість викликів функції	Кількість ітерацій
Значення змінної				
…				

Контрольні запитання

1. Математичне підґрунтя методу градієнтного спуску. Алгоритм роботи. Вибір кроку оптимізації.

2. Переваги і недоліки методу у порівнянні із методами нульового порядку (порівняти ефективність роботи для квадратичної функції, для функції Розенброка і функції Booth). Способи обчислення градієнта функції і їх точність.

3. Математичне підґрунтя методу найшвидшого градієнтного спуску. Алгоритм роботи.

4. Переваги і недоліки методу найшвидшого градієнтного спуску у порівнянні із методами нульового порядку (порівняти ефективність роботи для квадратичної функції, для функції Розенброка і функції Booth). Способи обчислення градієнта функції і їх точність.

5. Модифіковані методи градієнтного спуску: метод градієнтного спуску з моментом, алгоритм Нестерова. Алгоритм роботи методів. Які проблеми класичних методів градієнтного спуску вирішуються в цих методах?

6. Метод спряжених напрямків. Математичне підґрунтя методу (матриця Хессіана, визначення спряженості двох векторів). Алгоритм роботи. Переваги і недоліки методу.

7. Метод спряжених градієнтів. Математичне підґрунтя методу. Метод Флетчера-Рівза. Алгоритм роботи. Переваги і недоліки методу (порівняти ефективність роботи для квадратичної функції, для функції Розенброка і функції Booth).

Література

1. Rao Singiresu S. Engineering Optimization: Theory and Practice / Singiresu S. Rao. - New Jersey, USA: John Wiley & Sons, 2009 – 813 p.

2. Mohamed Bakr. 2013. Nonlinear Optimization in Electrical Engineering with Applications in MATLAB. Institution of Engineering and Technology.

ПРАКТИЧНЕ ЗАНЯТТЯ №5. МЕТОДИ ОПТИМІЗАЦІЇ ДРУГОГО ПОРЯДКУ

Мета

Метою практикуму є вивчення методів другого порядку пошуку мінімуму функції багатьох змінних: методу Левенберга-Марквардта та квазі-ньютонівського методу ДФП, порівняння швидкості їх роботи на прикладі квадратичної функції, Booth function та функції Розенброка, визначення особливостей роботи методів. В поєднанні із методом Левенберга-Марквардта вивчається метод штрафної функції для функцій з обмеженнями.

Домашнє завдання

1. Ознайомитися із теоретичними засадами методів другого порядку для оптимізації функції багатьох змінних: методом Ньютона, методом Левенберга-Марквардта та квазі-ньютонівськими методами ([1]: с.345-365, [2]: с.175-185).

2. Ознайомитися із методом Левенберга-Марквардта у програмі Matlab *LM_sumsqu.m*. Чи використовує метод одномірний пошук по напрямку? Якщо так, то які його параметри? Яка умова зупинки методу використовується? Які параметри має метод Левенберга-Марквардта.

3. Ознайомитися із квазі-ньютонівським методом ДФП у програмі Matlab *DFP_sumsqu.m*. Чи використовує метод одномірний пошук по напрямку? Якщо так, то які його параметри? За скільки метод ДВФ оптимізує квадратичну функцію?

4. Ознайомитися із теоретичними засадами методів штрафних функцій ([1]: с.430-447).

Порядок виконання роботи

1. Використовуючи програму *LM_sumsqu.m* провести оптимізацію квадратичної функції *sumsqu.m* із такими початковими умовами:

```
NumberOfParameters=2;
StartPoint=[-10  10]'
Epsilon=1.0e-5;
GradientNormMin=1e-5;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, напрямок пошуку на кожній ітерації методу Левенберга-Марквардта (використовуючи графіки та інформацію в консолі Матлаб).

2. Використовуючи програму *LM_booth.m* провести оптимізацію функції *booth.m* із такими початковими умовами:

```
NumberOfParameters=2;
StartPoint=[-10  10]'
Epsilon=1.0e-5;
GradientNormMin=1e-5;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, напрямок пошуку на кожній ітерації метода Левенберга-Марквардта (використовуючи графіки та інформацію в консолі Матлаб).

Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні досліджуючого кроку *Epsilon* (1e-5…1e-1) та умови зупинки алгоритму по величині градієнта *GradientNormMin* (1e-5…1e-1). Заповнити табл. 5.1 і порівняти метод Левенберга-Марквардта із методом найшвидшого градієнтного спуску і методом спряжених градієнтів для цієї ж функції із практичного заняття №4.

3. Використовуючи програму *LM_rosen.m* провести оптимізацію функції rosen.m із такими початковими умовами:

```
NumberOfParameters=2;
StartPoint=[-2  2]'
Epsilon=1.0e-5;
GradientNormMin=10e-5;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, напрямок пошуку на кожній ітерації метода Левенберга-Марквардта (використовуючи графіки та інформацію в консолі Матлаб).

Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, шлях оптимізації на площині XY) при зміні досліджуючого кроку *Epsilon* (1e-5…1e-1) та умови зупинки алгоритму по величині градієнта *GradientNormMin* (1e-5…1e-1). Заповнити табл. 5.1 і порівняти метод Левенберга-Марквардта із методом найшвидшого градієнтного спуску і методом спряжених градієнтів для цієї ж функції із практичного заняття №4.

4. Провести оптимізацію квадратичної функції *sumsqu.m* методом ДФП із такими початковими умовами:

```
NumberOfParameters=2;
StartPoint=[-10 10 ]'
Epsilon=1.0e-3;
Tolerance=1.0e-3;
GradientNormMin=1e-3;
MaxIteration=100;
LambdaMax=1.0;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, напрямок пошуку на кожній ітерації метода ДФП (використовуючи графіки та інформацію в консолі Матлаб). Порівняти із швидкістю роботи методу Левенберга- Марквардта з п.1.

5. Розрахувати матрицю Хессіана і обернену до неї для функції sumsqu.m і порівняти з матрицею А на останній ітерації методу ДФП.

6. Провести оптимізацію функції *booth.m* методом ДФП із такими початковими умовами (програма *DFP_booth.m*):

```
NumberOfParameters=2;
StartPoint=[-10 10 ]'
Epsilon=1.0e-3;
Tolerance=1.0e-3;
GradientNormMin=1e-3;
MaxIteration=100;
LambdaMax=1.0;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, напрямок пошуку на кожній ітерації метода ДФП (використовуючи графіки та інформацію в консолі Матлаб). Порівняти із швидкістю роботи методу Левенберга-Марквардта з п.2.

7. Провести оптимізацію функції *rosen.m* методом ДФП із такими початковими умовами (програма *DFP_rosen*):

```
NumberOfParameters=2;
StartPoint=[-2  2]'
Epsilon=1.0e-6;
Tolerance=1.0e-6;
GradientNormMin=10e-3;
MaxIteration=100;
LambdaMax=1.0;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, напрямок пошуку на кожній ітерації метода ДФП (використовуючи графіки та інформацію в консолі Матлаб).

Проаналізувати поведінку алгоритму (кількість ітерацій, кількість викликів функції, координати оптимальної точки, значення функції в оптимальній точці, шлях оптимізації на площині XY) при зміні досліджуючого кроку *Epsilon* (1e-5…1e-1), умови зупинки алгоритму по величині градієнта *GradientNormMin* (1e-5…1e-1) та мінімального розміру відрізка оптимізації при одномірній оптимізації *Tolerance*. Заповнити табл. 5.1 із методом найшвидшого градієнтного спуску і методом спряжених градієнтів для цієї ж функції із практичного заняття №4 та з методом Левенберга- Марквардта з п.3.

8. Порівняти метод Левенберга-Марквардта і метод ДФП при знаходженні параметрів моделі резистора. Цільова функція створювалась у практичному заняття №4, п.16.

Оптимізацію методом Левенберга-Марквардта (*LM_resistor.m*) проводити із такими параметрами:

```
NumberOfParameters=2;
StartPoint=[95  2]'
Epsilon=1.0e-6;
GradientNormMin=10e-5;
```

Оптимізацію методом ДФП (*DFPresistor.m*) проводити із такими параметрами:

```
NumberOfParameters=2;
StartPoint=[95 2]'
Epsilon=1.0e-6;
Tolerance=1.0e-6;
GradientNormMin=1e-3;
MaxIteration=100;
LambdaMax=1.0;
```

Знайти оптимальні значення параметрів методу Левенберга-Марквардта і методу ДФП при знаходженні параметрів моделі резистора.

9. Як модифікувати квадратичну функцію sumsqu.m для умовної оптимізації при $x_2<-2$.

Таблиця 5.1 Рекомендована форма запису результатів практикуму

Змінна, що досліджується	Координати точки мінімуму	Значення функції в точці мінімуму	Кількість викликів функції	Кількість ітерацій
Значення змінної				
…				

Контрольні запитання

1. Метод Ньютона. Математичне підґрунтя методу. Які вимоги до матриці Гессе. Алгоритм роботи. Переваги і недоліки методу.

2. Метод Левенберга-Марквардта. Математичне підґрунтя методу. Алгоритм роботи. Переваги і недоліки методу. Які проблеми методу Ньютона вирішує метод Левенберга-Марквардта.

3. Квазіньютонівські методи. Метод Бройдена. Математичне підґрунтя методу. Алгоритм роботи. Переваги і недоліки методу. Переваги і недоліки у порівнянні із методом Ньютона.

4. Квазіньютонівські методи. Метод Девідона-Флетчера-Пауела. Математичне підґрунтя методу. Алгоритм роботи. Переваги і недоліки методу (порівняти ефективність роботи для квадратичної функції, для функції Розенброка і функції Booth).

5. Метод зовнішніх штрафів. Вимоги до штрафної функції. Показати, що при використанні штрафної функції можна використовувати класичні методи безумовної оптимізації. Алгоритм оптимізації класичними методами безумовної оптимізації при використанні штрафної функції. Навести приклади штрафної функції для задач з обмеженнями у вигляді рівностей і нерівностей. Переваги і недоліки методу зовнішніх штрафів.

6. Метод внутрішніх штрафів. Вимоги до бар’єрної функції. Алгоритм оптимізації класичними методами безумовної оптимізації при використанні бар’єрної функції. Навести приклади бар’єрної функції для задач з обмеженнями у вигляді рівностей і нерівностей. Переваги і недоліки методу внутрішніх штрафів. Записати бар’єрну і штрафну функцію для такої задачі оптимізації:

Література

1. Rao Singiresu S. Engineering Optimization: Theory and Practice / Singiresu S. Rao. - New Jersey, USA: John Wiley & Sons, 2009 – 813 p.

2. Mohamed Bakr. 2013. Nonlinear Optimization in Electrical Engineering with Applications in MATLAB. Institution of Engineering and Technology.

ПРАКТИЧНЕ ЗАНЯТТЯ №6. ГЛОБАЛЬНІ МЕТОДИ ОПТИМІЗАЦІЇ. ГЕНЕТИЧНИЙ АЛГОРИТМ

Мета

Метою роботи є дослідження ефективності генетичного алгоритму в задачах оптимізації складних функцій одного і багатьох мінімумів і дослідження впливу параметрів алгоритму на точність і повторюваність знаходження мінімуму.

Домашнє завдання

1. Ознайомитися з теоретичними засадами генетичного алгоритму ([1]:с.694-702, [2]:с.1-93)

2. Ознайомитися із реалізацією генетичного алгоритму у програмі Matlab *bin_GA.m*. Визначити, як реалізовується зупинка роботи алгоритму, які змінні відповідають за налаштування операцій добору, кросоверу та мутації, чи використовується елітизм у алгоритмі.

Порядок виконання роботи

1. Використовуючи *bin_GA.m* провести оптимізацію функції Розенброка із такими початковими умовами:

```
function_name='rosen'
npar=2;
Xmin=-2;
Xmax=2;
Xnbits=10;
popsize=200;
mutrate=.01;
steady_state_param=0.5;

maxit=100;
mincost=-1;

N_stag = 5;
proc_stag = 5;
stag_start = 15;
```

```
cost_stagn_parents_procent = 1;
cost_stagn_limit = 0.8;
cost_stagn_analysis_start=15;

selection_number = 2;
crossover_number = 3;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, положення точки мінімуму і значення функції в точці мінімуму, умову зупинки оптимізації, використовуючи графіки та інформацію в консолі Matlab.

Повторити запуск оптимізації 5 раз. Спостерігати, що для кожного запуску змінюється кількість ітерацій, змінюється фінальна точка оптимізації. Занести результати кожного запуску оптимізації у табл. 6.1.

2. Дослідити, як змінюється збіжність генетичного алгоритму в залежності від величини популяції *popsize*. Перевірити значення 30, 50, 100, 200, 400. Для кожного значення розміру популяції повторити запуск оптимізації 5 раз і для кожного запуску оптимізації записати кількість ітерацій і викликів функції, фінальну точку і значення цільової функції. Занести результати кожного запуску оптимізації у табл. 6.1. Для кожного значення *popsize* розрахувати по 5 запускам середнє значення та стандартне відхилення. Зробити висновки про швидкість і стабільність роботи генетичного алгоритму.

3. Встановити значення *popsize=200*. Дослідити, як змінюється збіжність генетичного алгоритму в залежності від кількості біт на аргумент функції *Xnbits*. Перевірити значення 5, 10, 20, 30. Для кожного значення *Xnbits* повторити запуск оптимізації 5 раз і для кожного запуску оптимізації записати кількість ітерацій і викликів функції, фінальну точку і значення цільової функції. Занести результати кожного запуску оптимізації у табл. 6.1. Для кожного значення *Xnbits* розрахувати по 5 запускам середнє значення та стандартне відхилення. Зробити висновки про швидкість, точність і стабільність роботи генетичного алгоритму.

4. Встановити значення *Xnbits =10*. Дослідити, як змінюється збіжність генетичного алгоритму в залежності від імовірності мутації *mutrate*. Перевірити значення 0.01, 0.1, 0.2. Для кожного значення *mutrate* повторити запуск оптимізації 5 раз і для кожного запуску оптимізації

записати кількість ітерацій і викликів функції, фінальну точку і значення цільової функції. Занести результати кожного запуску оптимізації у табл. 6.1. Для кожного значення *mutrate* розрахувати по 5 запускам середнє значення та стандартне відхилення. Зробити висновки про швидкість, точність і стабільність роботи генетичного алгоритму.

5. Встановити значення *mutrate =0.01*. Дослідити, як змінюється збіжність генетичного алгоритму в залежності від типу селекції. Перевірити рангову селекцію (*selection_number=1*) і рівномірну селекцію (*selection_number=3*). Для кожного типу селекції повторити запуск оптимізації 5 раз і для кожного запуску оптимізації записати кількість ітерацій і викликів функції, фінальну точку і значення цільової функції. Занести результати кожного запуску оптимізації у табл. 6.1. Для кожного типу селекції розрахувати по 5 запускам середнє значення та стандартне відхилення. Зробити висновки про швидкість, точність і стабільність роботи генетичного алгоритму.

6. Встановити значення *selection_number=2*. Дослідити, як змінюється збіжність генетичного алгоритму в залежності від типу кросоверу. Перевірити одноточковий кросовер (*crossover_number=1*) і порівняти його з двоточковим (*crossover_number=3*). Для кожного типу селекції повторити запуск оптимізації 5 раз і для кожного запуску оптимізації записати кількість ітерацій і викликів функції, фінальну точку і значення цільової функції. Занести результати кожного запуску оптимізації у табл. 6.1. Для кожного типу кросоверу розрахувати по 5 запускам середнє значення та стандартне відхилення. Зробити висновки про швидкість, точність і стабільність роботи генетичного алгоритму.

7. Встановити значення *crossover_number = 3*. Відкрити програму *bin_GA_simple_selection.m* і встановити параметри з п.1. У програмі *bin_GA.m* і *bin_GA_simple_selection.m* реалізовано генетичний алгоритм Steady-State . Кількість хромосом, яка проходить незмінною у наступну популяцію визначається параметром *steady_state_param*. При цьому у *bin_GA.m* хромосоми для операції кросоверу вибираються лише з тієї частини популяції, яка проходить у наступну популяцію (тобто лише з найкращих хромосом), а у *bin_GA_simple_selection.m* – з усієї популяції. Порівняти реалізацію алгоритму у *bin_GA_simple_selection.m* з *bin_GA.m* для значень *steady_state_param* 0.5, 0.3, 0.1, 0.01 і розміру популяції 50 і 200. Для кожного набору параметрів повторити запуск оптимізації 5 раз і для кожного запуску оптимізації записати кількість ітерацій і викликів функції,

фінальну точку і значення цільової функції. Занести результати кожного запуску оптимізації у табл. 6.1. Для кожного набору параметрів розрахувати по 5 запускам середнє значення та стандартне відхилення. Зробити висновки про швидкість, точність і стабільність роботи генетичного алгоритму.

8. *У bin_GA.m* з параметрами з п.1 вимкнути передчасну зупинку алгоритму

```
cost_stagn_analysis_start=100;
stag_start = 100;
```

і перевірити, чи зменшується значення цільової функції після стагнації алгоритму (~ 16-20 ітерація). Повторити запуск алгоритму 5 раз.

9. Повторити п.2 – п.6 для функції Растрігіна rastr2d.m

```
function_name='rastr2d'
Xmin=-2;
Xmax=2;
```

Зробити висновки про оптимальне значення кожного параметру генетичного алгоритму.

10. Повторити п.2– п.8 для функції Easom easom.m

```
function_name='easom'
Xmin=-10;
Xmax=10;
```

Зробити висновки про оптимальне значення кожного параметру генетичного алгоритму.

Таблиця 6.1 Рекомендована форма запису результатів практикуму

Змінна, що досліджується	Координати точки мінімуму	Значення функції в точці мінімуму	Кількість викликів функції	Кількість ітерацій
Значення змінної				
…				

Контрольні запитання

1. Структурна схема бінарного генетичного алгоритму. Основні операції генетичного алгоритму. Розкрити бінарне кодування змінних.

2. Бінарний генетичний алгоритм. Основні операції генетичного алгоритму. Розкрити бінарне кодування змінних і види основних операцій бінарного генетичного алгоритму.

3. Steady-state генетичний алгоритм. Ідеологія елітизму. Цільова функція для максимізації і для мінімізації. Модифікації цільової функції.

4. Рекомендації по вибору основних параметрів генетичного алгоритму.

5. Умови зупинки генетичного алгоритму.

Література

1. Rao Singiresu S. Engineering Optimization: Theory and Practice / Singiresu S. Rao. - New Jersey, USA: John Wiley & Sons, 2009 – 813 p.

2. Rahmat-Samii Yahya Electromagnetic optimization by genetic algorithm / Yahya Rahmat-Samii Yahya, Eric Michielssen – New York, USA: John Wiley & Sons, 1999. – 480 p.

ПРАКТИЧНЕ ЗАНЯТТЯ №7. ГЛОБАЛЬНІ МЕТОДИ ОПТИМІЗАЦІЇ. АЛГОРИТМ БДЖОЛИНОГО РОЮ

Мета

Метою роботи є дослідження ефективності алгоритму бджолиного рою в задачах оптимізації складних функцій одного і багатьох мінімумів і дослідження впливу параметрів алгоритму на точність і повторюваність знаходження мінімуму.

Домашнє завдання

1. Ознайомитися з теоретичними засадами алгоритму бджолиного рою ([1]:с.708-711, [2], [3]).

2. Ознайомитися із реалізацією алгоритму бджолиного рою у програмі Matlab *PSO_run.m* та *pso_Trelea_vectorized.m*. Визначити, як реалізовується зупинка роботи алгоритму, як реалізуються граничні умови, які змінні відповідають за налаштування розміру популяції, інерційного та соціальних коефіцієнтів.

Порядок виконання роботи

1. Провести оптимізацію функції Растрігіна із такими початковими умовами (файл *PSO_run.m*):

```
par_min=[-2 -2];
par_max=[2 2];
function_name = 'rastr2d';
minmax=0;
modl=1;
ps=30;
epoch   = 50;
errgrad = 0.001;
errgoal=0;
errgraditer=10;
BoundaryType=1;
```

Зробити висновки про кількість ітерацій, кількість викликів функції, положення точки мінімуму і значення функції в точці мінімуму, умову зупинки оптимізації, використовуючи графіки та інформацію в консолі Matlab.

Повторити запуск оптимізації 5 раз. Спостерігати, що для кожного запуску змінюється кількість ітерацій, змінюється фінальна точка оптимізації.

2. Дослідити, як змінюється збіжність АБР в залежності від величини популяції *ps*. Перевірити значення 10, 20, 30, 50, 100, 400. Для цього скористатись програмою *PSO_run_serie.m* із параметрами з п.1, яка виводить в консоль усереднене по 100 запускам алгоритму кількість викликів функції, кількість ітерацій, значення цільової функції, координати фінальної точки оптимізації. Занести результати кожного запуску оптимізації у табл. 7.1. Зробити висновки про швидкість, точність і стабільність роботи АБР.

3. Встановити значення *ps=30*. Дослідити, як змінюється збіжність АБР в залежності від типу граничних умов (встановлюється параметром *BoundaryType*). Для цього скористатись програмою *PSO_run_serie.m* із параметрами з п.1, яка виводить в консоль усереднене по 100 запускам алгоритму кількість викликів функції, кількість ітерацій, значення цільової функції, координати фінальної точки оптимізації. Занести результати кожного запуску оптимізації у табл. 7.1. Зробити висновки про швидкість, точність і стабільність роботи АБР.

4. Встановити значення *BoundaryType=2*. Дослідити, як змінюється збіжність АБР в залежності від параметрів АБР (інерційного w, когнітивного *c1* і соціального *c2*). Порівняти набір параметрів Трелеа (*modl=1*) і Клерка (*modl=2*). Для цього скористатись програмою *PSO_run_serie.m* із параметрами з п.1, яка виводить в консоль усереднене по 100 запускам алгоритму кількість викликів функції, кількість ітерацій, значення цільової функції, координати фінальної точки оптимізації. Занести результати кожного запуску оптимізації у табл. 7.1. Зробити висновки про швидкість, точність і стабільність роботи АБР.

5. Встановити значення *modl =1*. Розмістити оптимальне значення на межі діапазону і повторити обчислення в п.2 – п.4

```
par_min=[0 0];
par_max=[4 4];
```

6. Записати коефіцієнт відбиття для ФНЧ на рис.7.1 для умови L1=L3, C1=C2.

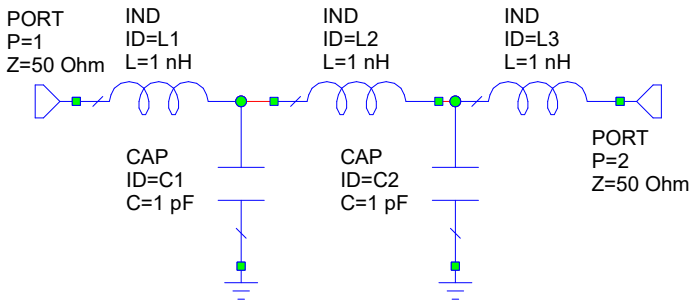

Рис.7.1 Схема фільтру низьких частот

7. Записати цільову функцію, яка забезпечить синтез фільтра з частотою зрізу 500 МГц та коефіцієнтом відбиття в робочій смузі частот менше -20 дБ.

8. Спробувати синтезувати фільтр за допомогою алгоритму Левенберга-Марквардта (*LM_filter.m*) та методом найшвидшого градієнтного спуску (*steepest_filter.m*).

Перевірити такі стартові точки [C L1 L2]: [4 10 20]; [8 16 20]; [1 1 1].

9. Ознайомитися із програмою Matlab *PSO_run_filter.m* і запустити її на виконання із такими налаштуваннями:

```
par_min=[0 0   0]; par_max=[20 40  40];
function_name = 'filter_model';
minmax=0; modl=1;  ps=30; epoch   = 100;
errgrad = 0.0001;   errgoal=0;  errgraditer=10;
BoundaryType=2;
```

Порівняти із результатами в п.8 по таким результатам критеріям:

- чи досягнуто умови оптимізації;
- чи метод Левенберга-Марквардта та метод найшвидшого градієнтного спуску досягають мінімуму при будь-якій стартовій точці;
- кількість викликів функції для досягнення мінімуму.

10. Визначити оптимальні параметри алгоритму Левенберга-Марквардта та методу найшвидшого градієнтного спуску в задачі синтезу ФНЧ. Критерій оптимальності – результат оптимізації близький до точки мінімуму функції та мінімізація кількості викликів функції.

11. Визначити оптимальні параметри АБР (кількість часток у рої, граничні умови, набір параметрів Трелеа чи Клерка) в задачі синтезу ФНЧ.

12. Повторити п.2 – п.4 практичного завдання для функції Розенброка

```
function_name='rosen'
par_min=[-2 -2];
par_max=[2 2];
```

13. Повторити п.2 – п.4 практичного завдання для функції Easom.

```
function_name='easom'
par_min=[-10 -10];
par_max=[10 10];
```

Таблиця 7.1 Рекомендована форма запису результатів практикуму

Змінна, що досліджується	Координати точки мінімуму (середнє по 100 запускам)	Значення функції в точці мінімуму (середнє по 100 запускам)	Кількість викликів функції (середнє по 100 запускам)	Кількість ітерацій (середнє по 100 запускам)
Значення змінної				
…				

Контрольні запитання

1. Структурна схема алгоритму бджолиного рою.

2. Основні види граничних умов в алгоритмі. Як вони реалізуються програмно?

3. Рекомендації по вибору основних параметрів алгоритму: кількість особин у рої, коефіцієнт інерції і соціальні коефіцієнтию

4. Які умови зупинки алгоритму реалізуються в АБР

5. Порівняти швидкість роботи АБР та генетичного алгоритму на прикладі функції Растрігіна.

Література

1. Rao Singiresu S. Engineering Optimization: Theory and Practice / Singiresu S. Rao. - New Jersey, USA: John Wiley & Sons, 2009 – 813 p.

2. Kennedy J. Particle swarm optimization / J. Kennedy, R. Eberhart // IEEE International Conference on Neural Networks, 27-30 Nov 1995. – Perth,Australia, 1995. – P. 1942-1948.

3. Robinson J. Particle swarm optimization in electromagnetics / J. Robinson, Y. Rahmat-Samii // IEEE Transactions on Antennas and Propagation. – 2004. –Vol. 52, Issue 2. – P. 397-407.