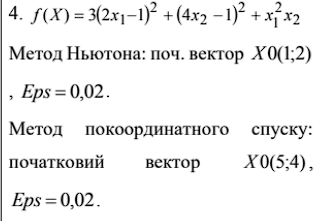
**Практична робота № 6**

**Тема. Нелінійна багатовимірна безумовна оптимізація**



**Метод Ньютона:**

1. Обчислити градієнт функції: df/dx1 = 12x1 - 6 + 2x2 df/dx2 = 8x2 - 4 + x1^2
2. Обчислити гессіан функції: d^2f/dx1^2 = 12 + 2x2 d^2f/dx1dx2 = 2x1 d^2f/dx2dx1 = 2x1 d^2f/dx2^2 = 8
3. Задати початкову точку X0 = (1, 2)
4. Обчислити значення функції F(X0)
5. Повторювати наступні кроки, доки не досягнута необхідна точність Eps: a. Обчислити обернений гессіан: H^-1 = [[d^2f/dx1^2, d^2f/dx1dx2], [d^2f/dx2dx1, d^2f/dx2^2]]^-1 b. Обчислити вектор зміни X: delta\_X = -H^-1 \* gradient(X) c. Оновити значення X: X = X + delta\_X d. Обчислити значення функції F(X)
6. Вивести оптимальну точку X та мінімальне значення функції F(X)
7. import numpy as np
8. from scipy.optimize import minimize
9. # Функція
10. def objective(X):
11. x1, x2 = X
12. return 3\*(2\*x1 - 1)\*\*2 + (4\*x2 - 1)\*\*2 + x1\*\*2 \* x2
13. # Градієнт функції
14. def gradient(X):
15. x1, x2 = X
16. df\_dx1 = 12\*x1 - 6 + 2\*x2
17. df\_dx2 = 8\*x2 - 4 + x1\*\*2
18. return np.array([df\_dx1, df\_dx2])
19. # Гессіан функції
20. def hessian(X):
21. x1, x2 = X
22. h11 = 12 + 2\*x2
23. h12 = 2\*x1
24. h21 = 2\*x1
25. h22 = 8
26. return np.array([[h11, h12], [h21, h22]])
27. # Початковий вектор
28. X0 = np.array([1, 2])
29. # Розв'язання задачі мінімізації
30. result = minimize(objective, X0, method='Newton-CG', jac=gradient, hess=hessian, tol=0.02)
31. # Виведення результату
32. print('Метод Ньютона:')
33. print('Оптимальний вектор X:', result.x)

print('Мінімальне значення функції:', result.fun)

**Метод покоординатного спуску:**

1. Задати початкову точку X0 = (5, 4)
2. Обчислити значення функції F(X0)
3. Повторювати наступні кроки, доки не досягнута необхідна точність Eps: a. Оновити значення x1: x1 = x1 - learning\_rate \* (dF/dx1) b. Оновити значення x2: x2 = x2 - learning\_rate \* (dF/dx2) c. Обчислити значення функції F(X)
4. Вивести оптимальну точку X та мінімальне значення функції F(X)
5. from scipy.optimize import minimize
6. # Функція
7. def objective(X):
8. x1, x2 = X
9. return 3\*(2\*x1 - 1)\*\*2 + (4\*x2 - 1)\*\*2 + x1\*\*2 \* x2
10. # Початковий вектор
11. X0 = np.array([5, 4])
12. # Розв'язання задачі мінімізації
13. result = minimize(objective, X0, method='BFGS', tol=0.02)
14. # Виведення результату
15. print('Метод покоординатного спуску:')
16. print('Оптимальний вектор X:', result.x)

print('Мінімальне значення функції:', result.fun)

В обох методах Eps використовується для визначення точності досягнення оптимальної точки. Можна обчислити значення функції та оновлювати значення X за допомогою цих алгоритмів до тих пір, поки не досягнете заданої точності Eps.