Національний технічний університет України «КПІ»

Кафедра автоматизованих систем обробки інформації і управління

Лабораторна робота №2

з дисципліни «[Аналіз даних в інформаційних управляючих системах](https://do.ipo.kpi.ua/course/view.php?id=5470)»

на тему: «[СТОХАСТИЧНІ МЕТОДИ ПОШУКУ](https://do.ipo.kpi.ua/mod/resource/view.php?id=149436)»

Виконав:

студент групи ІС-23

Шимків М.В.

Викладач:

Гавриленко О.В

Київ 2024

**Мета роботи**: практичне засвоєння алгоритмів програмування стохастичних

методів пошуку глобального екстремуму (мінімуму) багатомодальної цільової функції декількох змінних.

**Завдання до роботи**

1. Ознайомитися з конспектом лекцій та рекомендованою літературою, а також додатком до лабораторної роботи, що містить короткі теоретичні відомості про стохастичні методи пошуку та їх застосування в Matlab.

2. За допомогою команди >> help вивчити функції meshgrid, mesh, surface, min (див.приклади 1 і 2).

3. Побудувати трьохвимірний графік заданої функції відповідно до вашого

варіанту.

4. Реалізувати алгоритм простого стохастичного пошуку глобального мінімуму цільової функції відповідно до вашого варіанту.

5. Реалізувати алгоритм методу імітації відпалу.

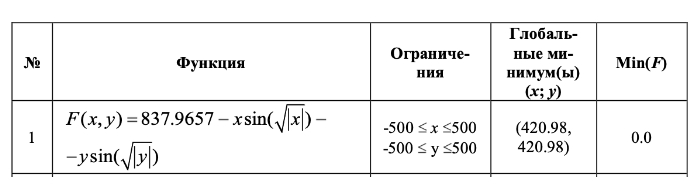
6. Порівняти точки глобального мінімуму і значення функції в мінімумі, визначені реалізованими методами стохастичного пошуку.

7. Оформити звіт з роботи.

8. Відповісти на контрольні питання.

Варіант 25

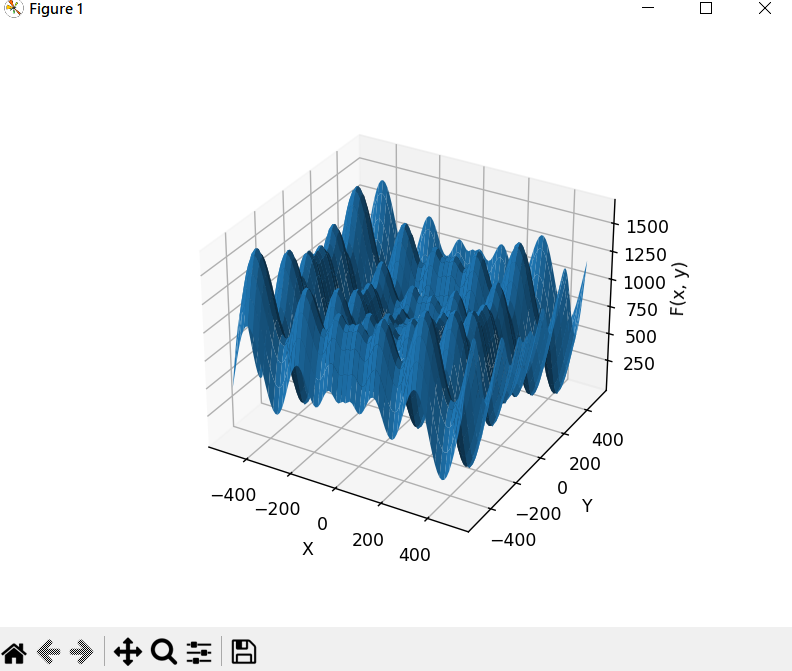
**Набір даних:**

****

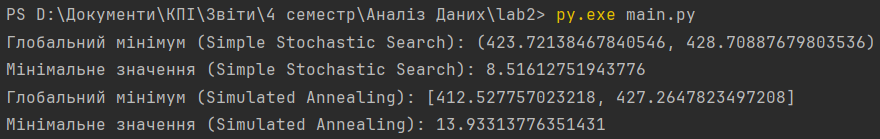
Запуск програми відбувається так:



Побудуємо трьохвимірний графік функції:



Порівняння точок глобального мінімуму і значення функції в мінімумі, визначені реалізованими методами простого стохастичного пошуку і алгоритмом методу імітації відпалу:



**Код програми:**

import numpy as np  
import matplotlib.pyplot as plt  
  
# Використаємо numpy.meshgrid для створення сітки точок  
x = np.linspace(-500, 500, 100)  
y = np.linspace(-500, 500, 100)  
X, Y = np.meshgrid(x, y)  
  
# Побудова поверхні за допомогою matplotlib  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
Z = 837.9657 - X \* np.sin(np.sqrt(np.abs(X))) - Y \* np.sin(np.sqrt(np.abs(Y)))  
ax.plot\_surface(X, Y, Z)  
ax.set\_xlabel('X')  
ax.set\_ylabel('Y')  
ax.set\_zlabel('F(x, y)')  
  
plt.show()  
  
def objective\_function(x, y):  
 return 837.9657 - x \* np.sin(np.sqrt(np.abs(x))) - y \* np.sin(np.sqrt(np.abs(y)))  
  
def simple\_stochastic\_search(objective\_function, bounds, n\_iter=1000):  
 best\_solution = None  
 best\_value = float('inf')  
 for \_ in range(n\_iter):  
 x = np.random.uniform(bounds[0][0], bounds[0][1])  
 y = np.random.uniform(bounds[1][0], bounds[1][1])  
 value = objective\_function(x, y)  
 if value < best\_value:  
 best\_solution = (x, y)  
 best\_value = value  
 return best\_solution, best\_value  
  
# Обмеження  
bounds = [(-500, 500), (-500, 500)]  
  
# Запускаємо алгоритм простого стохастичного пошуку  
best\_solution\_sss, best\_value\_sss = simple\_stochastic\_search(objective\_function, bounds)  
print("Глобальний мінімум (Simple Stochastic Search):", best\_solution\_sss)  
print("Мінімальне значення (Simple Stochastic Search):", best\_value\_sss)  
  
def simulated\_annealing(objective\_function, bounds, n\_iter=1000, initial\_temp=100, cooling\_rate=0.95):  
 best\_solution = None  
 best\_value = float('inf')  
 current\_solution = [np.random.uniform(bounds[i][0], bounds[i][1]) for i in range(len(bounds))]  
 current\_value = objective\_function(\*current\_solution)  
 for i in range(n\_iter):  
 new\_solution = [np.random.uniform(bounds[i][0], bounds[i][1]) for i in range(len(bounds))]  
 new\_value = objective\_function(\*new\_solution)  
 if new\_value < current\_value:  
 current\_solution = new\_solution  
 current\_value = new\_value  
 if new\_value < best\_value:  
 best\_solution = new\_solution  
 best\_value = new\_value  
 else:  
 p = np.exp(-(new\_value - current\_value) / initial\_temp)  
 if np.random.rand() < p:  
 current\_solution = new\_solution  
 current\_value = new\_value  
 initial\_temp \*= cooling\_rate  
 return best\_solution, best\_value  
  
# Запускаємо алгоритм методу імітації відпалу  
best\_solution\_sa, best\_value\_sa = simulated\_annealing(objective\_function, bounds)  
print("Глобальний мінімум (Simulated Annealing):", best\_solution\_sa)  
print("Мінімальне значення (Simulated Annealing):", best\_value\_sa)

**Контрольні питання:**

1. **Основні етапи простого стохастичного пошуку**:
   * Генерація випадкової початкової точки.
   * Оцінка значення функції цілі в початковій точці.
   * Повторення наступних кроків протягом заданої кількості ітерацій або до досягнення критерію зупинки:
     + Генерація випадкової пробної точки в допустимій області.
     + Оцінка значення функції цілі в пробній точці.
     + Порівняння значень функції в поточній та пробній точках.
     + Прийняття рішення про збереження або відкидання пробної точки на основі критерію, такого як зменшення значення функції.
2. **Основні етапи методу імітації відпалу**:
   * Вибір початкової точки.
   * Проведення ітераційного процесу, під час якого здійснюється переміщення від поточної точки до нової точки з певною ймовірністю.
   * Зниження температури з плином часу або кількості ітерацій.
   * Зупинка алгоритму за встановленим критерієм зупинки.
3. **Особливість методу відпалу, що дозволяє знаходити глобальний мінімум**:
   * Метод відпалу використовує ймовірність переходу від поточного стану до гіршого стану залежно від температури. Це дозволяє алгоритму уникати застрягання в локальних мінімумах і здійснювати дослідження інших областей простору параметрів.
4. **Мета введення параметру температури в методі відпалу і його вплив на величину відхилу пробної точки**:
   * Температура використовується для керування ймовірністю прийняття гіршого рішення. Зі зниженням температури ймовірність прийняття гіршого рішення зменшується. Величина відхилу пробної точки від поточного мінімуму може залежати від температури: при високій температурі вона може бути великою, а при низькій - меншою.
5. **Термін "заборонена область" у методі відпалу**:
   * Заборонена область - це частина простору параметрів, в якій не проводяться пошуки в рамках алгоритму. Це може бути область, яка неприпустима для розв'язку задачі або область, яку необхідно уникати з певних причин (наприклад, через низьку ефективність функції в цій області).

**Висновок:**

Зробивши лабораторну роботу, я зрозумів, що обидва методи - простий стохастичний пошук і метод імітації відпалу - мають свої переваги та недоліки. Простий стохастичний пошук простий у реалізації, але його точність залежить від кількості випадкових точок, що обираються. З іншого боку, метод імітації відпалу потребує налаштування параметрів, таких як температура, але він може забезпечити кращу збіжність до глобального мінімуму. Враховуючи контекст задачі і обмеження часу, можна вибрати метод, який найкраще відповідає потребам та можливостям.