МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Липецкий государственный технический университет
Факультет автоматизации и информатики

Домашняя работа №4
по математическому программированию

Студент Станиславчук С. М.
Группа АС-21-1

Руководитель Качановский Ю. П.

Липецк 2023 г.

Содержание

1. Задание

Вариант: 14

Метод Нелдера-Мида для функции:

f(x)=((x_1-5)^2)/2 + ((x_2-3)^2)/3+4,

x_1=(-2, +7)^T,

x_2=(-2, +7)^T

При lamda = 2, alpha = 1, beta = 0.5, gamma = 2

Минимальное число отражений: 4

2. Решение на языке Python с комментариями

```
import numpy as np


def nelder_mead(obj_func, initial_simplex, max_iterations=10, tol=1e-6, _alpha=1, _beta=0.5,
_gamma=2, _lambda=2):
    N = len(initial_simplex[0])  # Количество переменных
    iter_count = 0


    def print_iteration(iter_count, simplex_values, initial_simplex):
        print(f"Iteration {iter_count}:")
        print("Simplex values:", simplex_values)
        print("Current simplex:")
        print(initial_simplex)


    # Оценка значений целевой функции в начальных точках симплекса
    simplex_values = np.array([obj_func(point) for point in initial_simplex])


    # Основной цикл метода Нелдера-Мида
    for iteration in range(max_iterations):
        # Сортировка симплекса по значениям функции
        order = np.argsort(simplex_values)
        simplex_values = simplex_values[order]
        initial_simplex = initial_simplex[order]


        # Вывод промежуточных результатов
        print_iteration(iter_count, simplex_values, initial_simplex)


        # Вычисление центра масс (кроме самой худшей точки)
        centroid = np.mean(initial_simplex[:-1], axis=0)
```

```
    # Отражение худшей точки
    reflection = centroid + _alpha * (centroid - initial_simplex[-1]) # [ALPHA]
    reflection_value = obj_func(reflection)

    # Проверка условия отражения
    if simplex_values[0] <= reflection_value < simplex_values[-2]:
        initial_simplex[-1] = reflection
        simplex_values[-1] = reflection_value
    elif reflection_value < simplex_values[0]:
        # Экспансия
        expansion = centroid + _gamma * (reflection - centroid) # [GAMMa]
        expansion_value = obj_func(expansion)

        if expansion_value < reflection_value:
            initial_simplex[-1] = expansion
            simplex_values[-1] = expansion_value
        else:
            initial_simplex[-1] = reflection
            simplex_values[-1] = reflection_value
    else:
        # Консолидация
        contraction = centroid + _beta * (initial_simplex[-1] - centroid) # [BETA]
        contraction_value = obj_func(contraction)

        if contraction_value < simplex_values[-1]:
            initial_simplex[-1] = contraction
            simplex_values[-1] = contraction_value
        else:
            # Сжатие
            for i in range(1, N + 1):
                initial_simplex[i] = _lambda * (initial_simplex[i] + initial_simplex[0])
                simplex_values[i] = obj_func(initial_simplex[i])

    # Проверка критерия сходимости
    if np.max(np.abs(initial_simplex - initial_simplex[0])) < tol:
        break

    iter_count += 1

# Вывод результатов после последней итерации
```

```
    print_iteration(iter_count, simplex_values, initial_simplex)

    # Возвращение оптимального значения и точки
    return simplex_values[0], initial_simplex[0]

# Пример использования
def objective_function(x):
    return ((x[0] - 5) ** 2) / 2 + ((x[1] - 3) ** 2) / 3 + 4

initial_simplex = np.array([[-2, 7], [-2, 7], [1, 5]])

min_value, optimal_point = nelder_mead(objective_function, initial_simplex)

print("\nМинимум функции:", min_value)
print("Оптимальные значения x:", optimal_point)
```

Результат программы:

```
Iteration 0:
Simplex values: [13.33333333 33.83333333 33.83333333]
Current simplex:
[[ 1  5]
 [-2  7]
 [-2  7]]
Iteration 1:
Simplex values: [13.33333333 13.33333333 33.83333333]
Current simplex:
[[ 1  5]
 [ 1  5]
 [-2  7]]
Iteration 2:
Simplex values: [ 4.5        13.33333333 13.33333333]
Current simplex:
[[4 3]
 [1 5]
 [1 5]]
Iteration 3:
Simplex values: [ 4.5         4.5        13.33333333]
Current simplex:
[[4 3]
 [4 3]
 [1 5]]
Iteration 4:
Simplex values: [4.5        4.5        7.45833333]
Current simplex:
[[4 3]
 [4 3]
 [2 4]]
Iteration 5:
Simplex values: [4.5        4.5        6.08333333]
Current simplex:
[[4 3]
 [4 3]
 [3 3]]
```

Рисунок 1 – Результат итераций 1-5

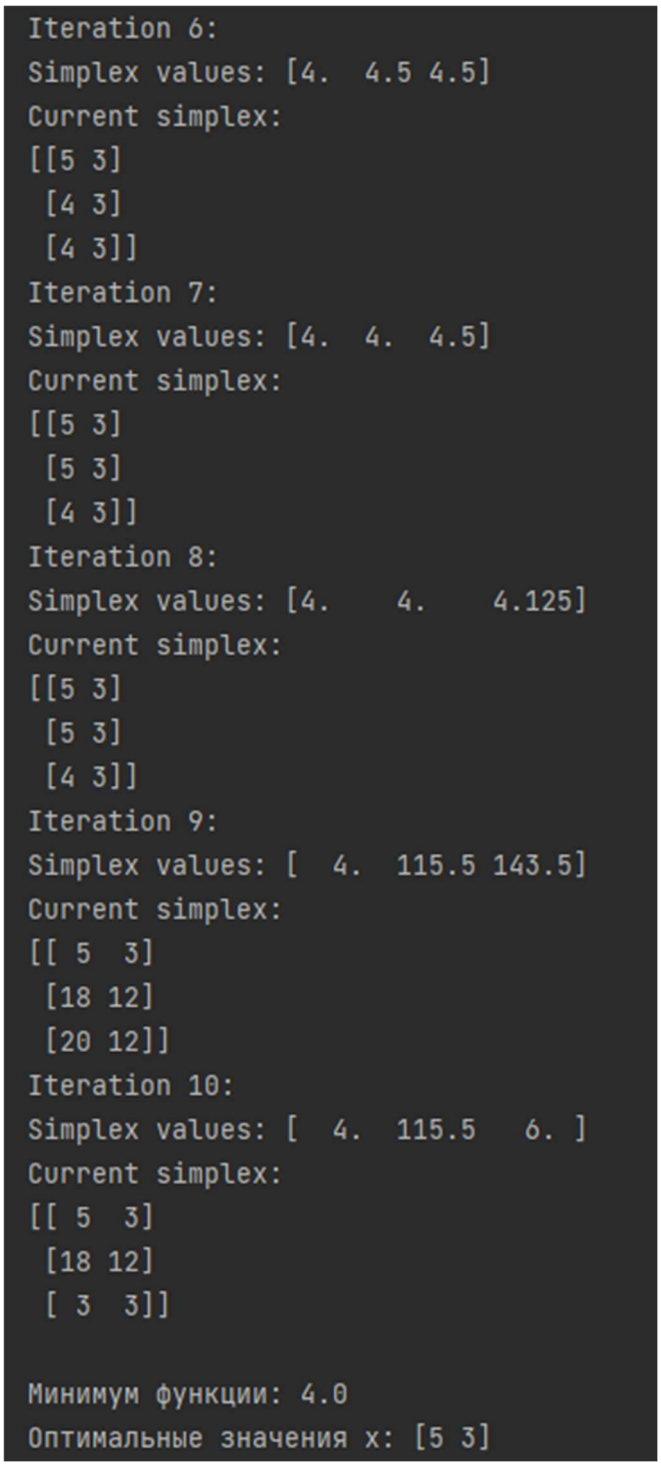

Рисунок 2 – результат итераций 6-10

3. Ответ.

Минимум функции = 4.0