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. Теория

Алгоритм заключается в формировании симплекса (simplex) и последующего его деформирования в направлении минимума, посредством трех операций:

1) Отражение (reflection);
2) Растяжение (expansion);
3) Сжатие (contract);

Инициализация симплекса: начинаем с исходной точки (вектора переменных) и создаем симплекс, который представляет собой набор точек в пространстве переменных.

Оценка значений функции: для каждой точки симплекса вычисляются значения целевой функции.

Сортировка точек: Точки симплекса сортируются по значениям функции, так что лучшая точка (с наименьшим значением) становится первой, а худшая точка - последней.

Центроид: вычисляется центроид симплекса, который представляет собой среднее значение координат всех точек симплекса, за исключением худшей.

Отражение: относительно центроида проводится отражение худшей точки. Это создает новую отраженную точку.

Оценка отраженной точки: оценивается значение функции в отраженной точке.

Выбор действия: В зависимости от результатов отражения могут выполняться следующие действия:

Если отраженная точка лучше (имеет меньшее значение функции), чем лучшая точка, то проводится растяжение.
Если отраженная точка лучше, чем вторая худшая точка, но не лучше лучшей, то отраженная точка заменяет худшую точку.
Если отраженная точка не дает улучшения, проводится сжатие.
Если ни отражение, ни растяжение, ни сжатие не улучшают ситуацию, выполняется редукция (уменьшение размера симплекса).
Проверка критериев останова: проводятся проверки на достижение максимального числа итераций, отсутствие улучшений и другие критерии останова. Если одно из условий выполняется, алгоритм завершает работу и возвращает лучшую точку.

Алгоритм продолжает эти шаги до тех пор, пока не выполняются критерии останова. В результате работы алгоритма получается набор точек, а лучшая из них считается приближенным оптимальным решением.

Основные параметры алгоритма, такие как коэффициенты отражения (alpha), растяжения (gamma), сжатия (beta) и уменьшения симплекса (_lambda), заданы по условию.

3. Решение на ЯП Python с комментариями

Описание метода Нелдера-Мида с приведенными шагами в программном коде:

3.1 Описание алгоритма на примере программы

1) Инициализация

Здесь мы объявляем такие переменные как: шаг, начальная точка, точки останова (конец, если нет улучшений, макс. итераций), размерность пространства параметров, первоначальное значение целевой функции в начальной точке, счетчик итераций без улучшений, хранение результатов в виде кортежа: [[точка, значение], ...], а также параметры alpha, beta, gamma, lamda.

А также цикл, который создает начальный симплекс вокруг начальной точки x_start. Для каждой координаты i в диапазоне от 0 до dim-1, создается новая точка x, которая отличается от x_start только в этой координате на величину step. Этот симплекс затем используется как исходный.

```
def nelder_mead(f, x_start,
                step=0.1, no_improve_thr=10e-6,
                no_improv_break=10, max_iter=0,
                alpha=1., gamma=2., beta=0.5, _lambda=2):


dim = len(x_start)            # Размерность пространства параметров
prev_best = f(x_start)        # Первоначальное значение целевой функции
not_improved = 0              # Счетчик итераций без улучшений
res = [[x_start, prev_best]]  # Хранение результатов


    # Цикл, генерирующий исходный симплекс
    for i in range(dim):
        x = copy.copy(x_start)
        x[i] = x[i] + step
        score = f(x)
        res.append([x, score])


iters = 0      # Объявление переменной для подсчета числа итераций
```

2) Главный цикл, в котором и происходят все вычисления.

Бесконечный цикл while True:

```
while True:
```

2.0) Сортировка

```
res.sort(key=lambda x: x[1])  # Лучшая точка (минимум) будет в начале списка.
best = res[0][1]              # Текущее лучшее значение целевой функции
```

2.1) Проверка на точку останова

```
if max_iter and iters >= max_iter:
    return res[0]
iters += 1                # Увеличиваем счетчик итераций.
```

2.2) Вывод в консоль результатов

```
print (f'Лучшее значение среди всех минимумов на итерации [{iters}]:', best)
print("Simplex:")
for point in res:
    print(point[0], point[1])
```

2.3) Проверка на улучшение

```
if best < prev_best - no_improve_thr:
    # произошло улучшение
    not_improved = 0
    prev_best = best
else:
    # улучшение не произошло
    not_improved += 1

if not_improved >= no_improv_break:
    # возвращаем текущий лучший результат
    return res[0]
```

2.4) Центроид

В этом блоке кода вычисляется центроид, который представляет собой среднее значение координат точек симплекса, за исключением худшей точки. Центроид используется для вычисления остальных точек

$$x_0 = \frac{1}{N-1} \sum_{i=1}^{N-1} x_i$$

```
x0 = [0.] * dim           # Инициализация координат центроида
# Вычисление суммы координат точек, за исключением худшей точки
```

```
for tup in res[:-1]:
    for i, c in enumerate(tup[0]):
        x0[i] += c / (len(res)-1)
```

2.5) Отражение

Отражение выполняется относительно центроида симплекса в направлении худшей точки. Затем проверяется, улучшилось ли значение функции в отраженной точке по сравнению с худшей точкой и второй лучшей точкой.

$$x_r = x_0 + \alpha \cdot (x_0 - x_w)$$

```
xr = x0 + alpha*(x0 - res[-1][0])     # Вычисление отраженной точки
rscore = f(xr)                         # Оценка значения целевой функции в отраженной точке
if res[0][1] <= rscore < res[-2][1]: # Проверка условия отражения
    refl_number += 1                # Подсчет числа отражений(по условию должно быть > 4)
    # Замена худшей точки отраженной точкой
    del res[-1]
    res.append([xr, rscore])
    continue
```

2.6) Растяжение

Если значение функции в отраженной точке меньше, чем значение функции в лучшей точке симплекса, то выполняется экспансия. Растяжение происходит в направлении центроида симплекса.

$$x_e = x_0 + \gamma \cdot (x_0 - x_w)$$

```
    if rscore < res[0][1]:                  # Проверка условия экспансии
    # Вычисление экспансии
    xe = x0 + gamma*(x0 - res[-1][0])
    escore = f(xe)
    if escore < rscore:                  # Проверка условия улучшения
        # Замена худшей точки экспансией
        del res[-1]
        res.append([xe, escore])
        continue
    else:
        # В случае неулучшения замена худшей точки отраженной точкой
        del res[-1]
        res.append([xr, rscore])
        continue
```

2.7) Сжатие

Если значение функции в отраженной точке больше или равно значению функции в худшей точке симплекса, то выполняется сжатие. Сжатие происходит в направлении центроида симплекса.

$$x_c = x_0 + \beta \cdot (x_0 - x_w)$$

```
xc = x0 + beta*(x0 - res[-1][0])        # Вычисление сжатия
cscore = f(xc)
if cscore < res[-1][1]:                 # Проверка условия сжатия
        # Замена худшей точки сжатием
        del res[-1]
        res.append([xc, cscore])
        continue
```

2.8) Редукция

В случае, если ни один из предыдущих шагов не привел к улучшению значения целевой функции, происходит уменьшение симплекса. Каждая точка симплекса сжимается в направлении лучшей точки симплекса. Уменьшение симплекса направлено на уменьшение размера симплекса в случае, если предыдущие шаги не привели к улучшению значения целевой функции. Каждая точка симплекса сжимается в направлении лучшей точки, и процесс повторяется.

$$x_{\text{red}} = x_1 + \lambda \cdot (x_{\text{tup}} - x_1)$$

```
x1 = res[0][0]          # Запоминание координат лучшей точки
nres = []               # Инициализация нового списка для хранения новых точек симплекса
# Применение уменьшения симплекса к каждой точке
for tup in res:
    redx = x1 + _lambda*(tup[0] - x1)
    score = f(redx)
    nres.append([redx, score])
# Обновление списка точек симплекса
res = nres
```

Конец алгоритма.

3.2 Полный код программы:

```
from pprint import pprint


import matplotlib.pyplot as plt

import numpy as np


f = lambda x, y: ((x-5)**2)/2+((y-3)**2)/3+4

start_point = (-2, 7)


alpha = 1

beta = 0.5

gamma = 2

lambd = 2


n = 2


refl_number = 0


def get_h_g_l(points, func):

 points = sorted(enumerate(points), key=lambda x: func(*x[1]))

 return points[-1][0], points[-2][0], points[0][0]


def get_tetr_in(point):

 b_1 = (np.sqrt(n+1)+n-1)/(n*np.sqrt(2))*lambd

 b_2 = (np.sqrt(n+1)-1)/(n*np.sqrt(2))*lambd

 points = [point,]

 for i in range(1, n + 1):

    cur_point = [0] * n

    for j in range(n):

        if j == i - 1:

            cur_point[j] = point[j] + b_1

        else:

            cur_point[j] = point[j] + b_2

    points.append(cur_point)

 return points


def display_vertex_values(iteration, verts, h, g, l, func):

    print(f'Значение функции каждой точки на итерации {iteration}:')

    print(f'f(xh) = {func(*verts[h]):.6f}, xh = {verts[h]}')

    print(f'f(xg) = {func(*verts[g]):.6f}, xg = {verts[g]}')

    print(f'f(xl) = {func(*verts[l]):.6f}, xl = {verts[l]}')
```

```
    # Calculate the center of gravity (x_c)

    x_c = [sum(verts[j][i] for j in range(len(verts)) if j != h) / len(verts) for i in
range(len(verts[0]))]

    print(f'\nxc = {x_c}')


    # Calculate the compression point (x_r)

    x_r = [(verts[h][i] + verts[l][i]) / 2 for i in range(len(verts[0]))]

    print(f'xr = {x_r}')


    # Calculate the stretching point (x_e)

    x_e = [verts[h][i] + 2 * (verts[h][i] - x_c[i]) for i in range(len(verts[0]))]

    print(f'xe = {x_e}')


    # Calculate the reflection point (x_o)

    x_o = [verts[h][i] + 0.5 * (x_c[i] - verts[h][i]) for i in range(len(verts[0]))]

    print(f'xo = {x_o}\n')


def diff_sqad(els):

 return (els[0] - els[1])**2


if __name__=="__main__":

 verts = get_tetr_in(start_point)

 print("Точки тетраэдра:")

 pprint(verts)

 for iter in range(1000):

    print(f'\n==============================[{iter}]==============================')

    #******************

    h, g, l = get_h_g_l(verts, f)

    display_vertex_values(iter, verts, h, g, l, f)

    print(f'Лучшее значение среди всех минимумов на итерации [{iter}]: f(x) =
{f(*verts[l]):.6f}')

    print(f"Число отражений: {refl_number}")

    o = [ 0, 0, 0 ]

    #******************

    for j in range(n+1):

        if j == h:

            continue

        o = list(map(sum, zip(o, verts[j])))

    o = [i/n for i in o]

    #******************

    if alpha > 0:
```

```
        r = [(1+alpha)*o[i]-alpha*verts[h][i] for i in range(n)]
    else:
        break
 #******************
    if f(*r) < f(*verts[l]):
        e = [gamma*r[i]+(1-gamma)*o[i] for i in range(n)]
        if f(*e) < f(*verts[l]):
            verts[h] = e
            print(f"{f(*e):.6f} < {f(*verts[l]):.6f} => условие растяжения выполнено")
            print("Замена худшей точки растяженной точкой")
            print(f"f(x0 + gamma*(x0 - xh)) = {f(*e):.6f} < {f(*verts[l]):.6f}")
            print(f"На этой итерации мы провели РАСТЯЖЕНИЕ симплекса")
        else:
            verts[h] = r
            print(f"f(x0 + gamma*(x0 - xh)) = {f(*e):.6f} >= {f(*verts[l]):.6f}")
            print("Замена худшей точки отраженной точкой")
            print(f"На этой итерации мы провели ОТРАЖЕНИЕ симплекса")
            refl_number += 1
    elif f(*verts[g]) < f(*r) <= f(*verts[h]):
        c = [beta*verts[h][i]+(1-beta)*o[i] for i in range(n)]
        verts[h] = c
        print(f"f(x0 + beta*(x0 - xh)) = {f(*verts[g]):.6f} < {f(*r):.6f} -> выпонено условие
сжатия")
        print(f"На этой итерации мы провели СЖАТИЕ симплекса")
    elif f(*verts[l]) < f(*r) <= f(*verts[g]):
        verts[h] = r
        print(f"f(l) < f(r) <= f(g)")
        print(f"На этой итерации мы провели ОТРАЖЕНИЕ симплекса")
        refl_number += 1
    elif f(*r) > f(*verts[h]):
        for j in range(n+1):
            if j == l:
                continue
            verts[j] = [verts[l][i]+0.5*(verts[j][i]-verts[l][i]) for i in range(n)]
        print(f"На этой итерации мы провели РЕДУКЦИЮ симплекса\n")
    print("Симплекс:")
    pprint(verts)
    #******************
    sigma = np.sqrt(1/(n+1)*sum([(f(*x_j)-1/(n+1)*sum([f(*vert) for vert in verts]))**2 for x_j
in verts]))
    print("Сигма:", sigma)
    if sigma < 0.0001:
```

```
print("\nТочка найдена!")
print("Найденная точка:", verts[l])
print("Значение функции в этой точке:", f(*verts[l]))
break
```

3.3 Результат программы

```
==============================[0]==============================
Значение функции каждой точки на итерации 0:
f(xh) = 36.739463, xh = [-1.4823619097949585, 8.931851652578136]
f(xg) = 33.833333, xg = (-2, 7)
f(xl) = 23.646082, xl = [-0.0681483474218636, 7.5176380902050415]

xc = [-0.6893827824739546, 4.8392126967350135]
xr = [-0.7752551286084111, 8.224744871391 59]
xe = [-3.068320164436966, 17.11712956426438]
xo = [-1.0858723461344566, 6.885532174656575]

Лучшее значение среди всех минимумов на итерации [0]: f(x) = 23.646082
Число отражений: 0
17.474653 < 23.646082 => условие растяжения выполнено
Замена худшей точки растяженной точкой
f(x0 + gamma*(x0 - xh)) = 17.474653 < 23.646082
На этой итерации мы провели РАСТЯЖЕНИЕ симплекса
Симплекс:
[(-2, 7),
 [-0.0681483474218636, 7.5176380902050415],
 [-0.13749870154287835, 3.912753830151291]]

==============================[1]==============================
Значение функции каждой точки на итерации 1:
f(xh) = 33.833333, xh = (-2, 7)
f(xg) = 23.646082, xg = [-0.0681483474218636, 7.5176380902050415]
f(xl) = 17.474653, xl = [-0.13749870154287835, 3.912753830151291]

xc = [-0.06854901632158066, 3.810130640118777]
xr = [-1.0687493507714392, 5.456376915075646]
xe = [-5.862901967356839, 13.3797387197624 47]
xo = [-1.0342745081607903, 5.405065320059388]

Лучшее значение среди всех минимумов на итерации [1]: f(x) = 17.474653
Число отражений: 0
4.863113 < 17.474653 => условие растяжения выполнено
Замена худшей точки растяженной точкой
f(x0 + gamma*(x0 - xh)) = 4.863113 < 17.474653
На этой итерации мы провели РАСТЯЖЕНИЕ симплекса
Симплекс:
```

```
[[3.6915294265528873, 3.1455878805344977],
 [-0.0681483474218636, 7.5176380902050415],
 [-0.13749870154287835, 3.912753830151291]]

==============================[2]==============================
Значение функции каждой точки на итерации 2:
f(xh) = 23.646082, xh = [-0.0681483474218636, 7.5176380902050415]
f(xg) = 17.474653, xg = [-0.13749870154287835, 3.912753830151291]
f(xl) = 4.863113, xl = [3.6915294265528873, 3.1455878805344977]

xc = [1.1846769083366697, 2.3527805702285964]
xr = [1.8116905395655118, 5.33161298536977]
xe = [-2.5737988589389302, 17.847353130157934]
xo = [0.5582642804574031, 4.935209330216819]

Лучшее значение среди всех минимумов на итерации [2]: f(x) = 4.863113
Число отражений: 0
f(l) < f(r) <= f(g)
На этой итерации мы провели ОТРАЖЕНИЕ симплекса
Симплекс:
[[3.6915294265528873, 3.1455878805344977],
 [3.6221790724318725, -0.4592963795192526],
 [-0.13749870154287835, 3.912753830151291]]

==============================[3]==============================
Значение функции каждой точки на итерации 3:
f(xh) = 17.474653, xh = [-0.13749870154287835, 3.912753830151291]
f(xg) = 8.938106, xg = [3.6221790724318725, -0.4592963795192526]
f(xl) = 4.863113, xl = [3.6915294265528873, 3.1455878805344977]

xc = [2.43790283299492, 0.895430500338415]
xr = [1.7770153625050045, 3.5291708553428944]
xe = [-5.2883017706184745, 9.947400489777044]
xo = [1.1502020657260208, 2.404092165244853]

Лучшее значение среди всех минимумов на итерации [3]: f(x) = 4.863113
Число отражений: 1
f(x0 + beta*(x0 - xh)) = 8.938106 < 12.958536 -> выпонено условие сжатия
На этой итерации мы провели СЖАТИЕ симплекса
Симплекс:
[[3.6915294265528873, 3.1455878805344977],
```

```
 [3.6221790724318725, -0.4592963795192526],
 [1.7596777739747507, 2.627949790329457]]

==============================[4]==============================
Значение функции каждой точки на итерации 4:
f(xh) = 9.295985, xh = [1.7596777739747507, 2.627949790329457]
f(xg) = 8.938106, xg = [3.6221790724318725, -0.4592963795192526]
f(xl) = 4.863113, xl = [3.6915294265528873, 3.1455878805344977]

xc = [2.43790283299492, 0.895430500338415]
xr = [2.725603600263819, 2.8867688354319774]
xe = [0.4032276559344121, 6.0929883703115415]
xo = [2.098790303484835, 1.761690145333936]

Лучшее значение среди всех минимумов на итерации [4]: f(x) = 4.863113
Число отражений: 1
f(l) < f(r) <= f(g)
На этой итерации мы провели ОТРАЖЕНИЕ симплекса
Симплекс:
[[3.6915294265528873, 3.1455878805344977],
 [3.6221790724318725, -0.4592963795192526],
 [5.5540307250100085, 0.05834171068578797]]

==============================[5]==============================
Значение функции каждой точки на итерации 5:
f(xh) = 8.938106, xh = [3.6221790724318725, -0.4592963795192526]
f(xg) = 7.037926, xg = [5.5540307250100085, 0.05834171068578797]
f(xl) = 4.863113, xl = [3.6915294265528873, 3.1455878805344977]

xc = [3.0818533838542983, 1.0679765304067619]
xr = [3.6568542494923797, 1.3431457505076225]
xe = [4.702830449587021, -3.5138421993712816]
xo = [3.3520162281430856, 0.30434007544375463]

Лучшее значение среди всех минимумов на итерации [5]: f(x) = 4.863113
Число отражений: 2
f(x0 + gamma*(x0 - xh)) = 7.792936 >= 4.863113
Замена худшей точки отраженной точкой
На этой итерации мы провели ОТРАЖЕНИЕ симплекса
Симплекс:
[[3.6915294265528873, 3.1455878805344977],
```

```
 [5.623381079131022, 3.6632259707395383],
 [5.5540307250100085, 0.05834171068578797]]

==============================[6]==============================
Значение функции каждой точки на итерации 6:
f(xh) = 7.037926, xh = [5.5540307250100085, 0.05834171068578797]
f(xg) = 4.863113, xg = [3.6915294265528873, 3.1455878805344977]
f(xl) = 4.340925, xl = [5.623381079131022, 3.6632259707395383]

xc = [3.104970168561303, 2.2696046170913453]
xr = [5.588705902070515, 1.8607838407126631]
xe = [10.452151837907419, -4.364184102125327]
xo = [4.329500446785656, 1.1639731638885666]

Лучшее значение среди всех минимумов на итерации [6]: f(x) = 4.340925
Число отражений: 3
На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:
[[4.657455252841954, 3.404406925637018],
 [5.623381079131022, 3.6632259707395383],
 [5.588705902070515, 1.8607838407126631]]

==============================[7]==============================
Значение функции каждой точки на итерации 7:
f(xh) = 4.605892, xh = [5.588705902070515, 1.8607838407126631]
f(xg) = 4.340925, xg = [5.623381079131022, 3.6632259707395383]
f(xl) = 4.113183, xl = [4.657455252841954, 3.404406925637018]

xc = [3.426945443990992, 2.355877632125519]
xr = [5.123080577456235, 2.6325953831748405]
xe = [9.912226818229563, 0.8705962578869513]
xo = [4.507825673030753, 2.1083307364190911]

Лучшее значение среди всех минимумов на итерации [7]: f(x) = 4.113183
Число отражений: 3
На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:
[[4.657455252841954, 3.404406925637018],
 [5.140418165986488, 3.5338164481882783],
```

```
 [5.123080577456235, 2.6325953831748405]]

==============================[8]==============================
Значение функции каждой точки на итерации 8:
f(xh) = 4.113183, xh = [4.657455252841954, 3.404406925637018]
f(xg) = 4.104845, xg = [5.140418165986488, 3.5338164481882783]
f(xl) = 4.052570, xl = [5.123080577456235, 2.6325953831748405]

xc = [3.421166247814241, 2.055470610454373]
xr = [4.890267915149094, 3.0185011544059295]
xe = [7.13003326289738, 6.102279556002308]
xo = [4.039310750328098, 2.7299387680456952]

Лучшее значение среди всех минимумов на итерации [8]: f(x) = 4.052570
Число отражений: 3
На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:
[[4.890267915149094, 3.0185011544059295],
 [5.131749371721361, 3.0832059156815594],
 [5.123080577456235, 2.6325953831748405]]

==============================[9]==============================
Значение функции каждой точки на итерации 9:
f(xh) = 4.052570, xh = [5.123080577456235, 2.6325953831748405]
f(xg) = 4.010987, xg = [5.131749371721361, 3.0832059156815594]
f(xl) = 4.006135, xl = [4.890267915149094, 3.0185011544059295]

xc = [3.340672428956818, 2.0339023566958296]
xr = [5.006674246302664, 2.825548268790385]
xe = [8.687896874455067, 3.8299814361328623]
xo = [4.231876503206527, 2.333248869935335]

Лучшее значение среди всех минимумов на итерации [9]: f(x) = 4.006135
Число отражений: 3
На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:
[[4.890267915149094, 3.0185011544059295],
 [5.011008643435227, 3.0508535350437445],
 [5.006674246302664, 2.825548268790385]]
```

==========[10]==========

Значение функции каждой точки на итерации 10:

f(xh) = 4.010167, xh = [5.006674246302664, 2.825548268790385]

f(xg) = 4.006135, xg = [4.890267915149094, 3.0185011544059295]

f(xl) = 4.000923, xl = [5.011008643435227, 3.0508535350437445]

xc = [3.300425519528107, 2.023118229816558]

xr = [5.008841444868946, 2.9382009019170647]

xe = [8.419171699851777, 4.430408346738039]

xo = [4.153549882915385, 2.4243332493034715]

Лучшее значение среди всех минимумов на итерации [10]: f(x) = 4.000923

Число отражений: 3

На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:

```
[[4.950638279292161, 3.034677344724837],
 [5.011008643435227, 3.0508535350437445],
 [5.008841444868946, 2.9382009019170647]]
```

==========[11]==========

Значение функции каждой точки на итерации 11:

f(xh) = 4.001619, xh = [4.950638279292161, 3.034677344724837]

f(xg) = 4.001312, xg = [5.008841444868946, 2.9382009019170647]

f(xl) = 4.000923, xl = [5.011008643435227, 3.0508535350437445]

xc = [3.3399500294347244, 1.9963514789869363]

xr = [4.980823461363694, 3.042765439884291]

xe = [8.172014779007032, 5.111329076200638]

xo = [4.145294154363443, 2.5155144118558868]

Лучшее значение среди всех минимумов на итерации [11]: f(x) = 4.000923

Число отражений: 3

На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:

```
[[4.980823461363694, 3.042765439884291],
 [5.011008643435227, 3.0508535350437445],
 [5.009925044152086, 2.9945272184804046]]
```

==========[12]==========

Значение функции каждой точки на итерации 12:

f(xh) = 4.000923, xh = [5.011008643435227, 3.0508535350437445]

f(xg) = 4.000793, xg = [4.980823461363694, 3.042765439884291]

f(xl) = 4.000059, xl = [5.009925044152086, 2.9945272184804046]

xc = [3.3302495018385936, 2.012430886121565]

xr = [5.010466843793656, 3.0226903767620747]

xe = [8.372526926628495, 5.127698832888103]

xo = [4.170629072636911, 2.531642210582655]

Лучшее значение среди всех минимумов на итерации [12]: f(x) = 4.000059

Число отражений: 3

f(l) < f(r) <= f(g)

На этой итерации мы провели ОТРАЖЕНИЕ симплекса

Симплекс:

[[4.980823461363694, 3.042765439884291],
 [4.979739862080553, 2.9864391233209506],
 [5.009925044152086, 2.9945272184804046]]

==========[13]==========

Значение функции каждой точки на итерации 13:

f(xh) = 4.000793, xh = [4.980823461363694, 3.042765439884291]

f(xg) = 4.000267, xg = [4.979739862080553, 2.9864391233209506]

f(xl) = 4.000059, xl = [5.009925044152086, 2.9945272184804046]

xc = [3.329888302077547, 1.9936554472671186]

xr = [4.99537425275789, 3.0186463291823475]

xe = [8.282693779935988, 5.140985425118636]

xo = [4.15535588172062, 2.518210443575705]

Лучшее значение среди всех минимумов на итерации [13]: f(x) = 4.000059

Число отражений: 4

На этой итерации мы провели РЕДУКЦИЮ симплекса

Симплекс:

[[4.99537425275789, 3.0186463291823475],
 [4.99483245311632, 2.990483170900678],
 [5.009925044152086, 2.9945272184804046]]

```
Точка найдена!
Найденная точка: [5.009925044152086, 2.9945272184804046]
Значение функции в этой точке: 4.000059237029897
```

* Можем заметить, что одно из условий задачи (минимальное число отражений: 4) выполнилось, т.к. на последней [13] итерации метода общее число отражений равно 4.

4. Ответ.

Минимум функции, который нашелся методом Нелдера-Мида для функции: f(x)=((x_1-5)^2)/2 + ((x_2-3)^2)/3+4 :

= 4.000059237029897