Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Метод Нелдера-Мида |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | В. В. Тынченко |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-17/1Б, 032156940 |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Вариант.............................................................................................................. 3

3 Описание метода............................................................................................... 3

4 Реализация метода............................................................................................ 3

5 Анализ результатов.......................................................................................... 5

6 Вывод................................................................................................................. 10

**1 Задание**

Разработать программу, реализующую метод Нелдера-Мида. Найти  
безусловный экстремум функции, выбранной в соответствии с заданием, с  
использованием разработанной программы.

**2 Вариант**

f(x) = (x1 - 1)^2 + (x2 + x1)^2

**3 Описание метода**

В основу метода деформируемого многогранника, или метода  
Нелдера–Мида (Nelder–Mead), положено построение последовательности  
систем n + 1 точек xi(k), i = 1,...,n+1 , которые являются вершинами выпуклого  
многогранника. Точки системы xi(k + 1), i = 1,...,n+1, на (k + 1)-й итерации  
совпадают с точками системы xi(k), i = 1,...,n+1, кроме i = h, где точка xh(k) -  
наихудшая в системе xi(k), i = 1,...,n+1 , т.е. f(xh(k)) = max f(xi(k)). 1<=i<=n+1 .  
Точка xh(k) заменяется на другую точку по специальным правилам, описанным  
ниже. В результате многогранники деформируются в зависимости от структуры  
линий уровня целевой функции, вытягиваясь вдоль длинных наклонных  
плоскостей, изменяя направление в изогнутых впадинах и сжимаясь в  
окрестности минимума. Построение последовательности многогранников  
заканчивается, когда значения функции в вершинах текущего многогранника  
отличаются от значения функции в центре тяжести системы xi(k), i = 1,...,n+1; i  
не равно h, не более чем на величину epsilon>0.

**4 Реализация метода**

Ниже в листинге 1 представлен текст python программы, реализующей  
метод для поиска минимума функции.

Листинг 1 – Текст python программы

import matplotlib.pyplot as plt  
import matplotlib.patches as mpatches  
cache = {}  
def func\_0(x):  
 if x not in cache:  
 cache[x] = (x[0] - 1)\*\*2 + (x[1] + x[0])\*\*2  
 return cache[x]  
func = func\_0  
class Vector(object):  
 def \_\_init\_\_(self, x, y):  
 self.x = x  
 self.y = y  
 self.f = func((x, y))  
 def \_\_repr\_\_(self):  
 return "({0}, {1})".format(self.x, self.y)  
 def \_\_add\_\_(self, other):  
 x = self.x + other.x  
 y = self.y + other.y  
 return Vector(x, y)  
 def \_\_sub\_\_(self, other):

Продолжение листинга 1

x = self.x - other.x  
 y = self.y - other.y  
 return Vector(x, y)  
 def \_\_rmul\_\_(self, other):  
 x = self.x \* other  
 y = self.y \* other  
 return Vector(x, y)  
 def \_\_truediv\_\_(self, other):  
 x = self.x / other  
 y = self.y / other  
 return Vector(x, y)  
 def c(self):  
 return (self.x, self.y)  
def nelder\_mead(f, v=None, alpha=1.5, beta=0.25, gamma=2.5, epsilon=0.0001):  
 if v is None:  
 n = 3  
 v1 = Vector(-20, -20)  
 v2 = Vector(-20, 20)  
 v3 = Vector(20, 20)  
 v4 = Vector(20, -20)  
 x = [None]  
 x.extend([v1, v2, v3, v4])  
 else:  
 x = [None]  
 x.extend(v)  
 n = len(v) - 1  
 work = True  
 x.append(None)  
 x.append(None)  
 x.append(None)  
 x.append(None)  
 x.append(None)  
 while work:  
 x\_l = sorted(x[1:n + 2], key=lambda i: i.f)[0]  
 x\_s = sorted(x[1:n + 2], key=lambda i: i.f)[-2]  
 x\_h = sorted(x[1:n + 2], key=lambda i: i.f)[-1]  
 x[n + 2] = Vector(0, 0)  
 for i in sorted(x[1:n + 2], key=lambda i: i.f)[1:n + 1]:  
 x[n + 2] += i  
 x[n + 2] /= n  
 sigma = 0  
 for j in range(1, n+2):  
 sigma += (x[j].f - x[n+2].f) \*\* 2  
 sigma = (sigma / (n + 1)) \*\* 0.5  
 if sigma <= epsilon:  
 return x\_l.c(), len(cache)  
 else:  
 x[n + 3] = x[n + 2] + alpha \* (x[n + 2] - x\_h)  
 if x[n + 3].f <= x\_l.f:  
 x[n + 4] = x[n + 2] + gamma \* (x[n + 3] - x[n + 2])  
 if x[n + 4].f < x\_l.f:  
 x[x.index(x\_h)] = x[n + 4]  
 continue  
 else:  
 x[x.index(x\_h)] = x[n + 3]  
 continue  
 elif x\_s.f < x[n + 3].f and x[n + 3].f <= x\_h.f:  
 x[n + 5] = x[n + 2] + beta \* (x\_h - x[n + 2])

Окончание листинга 1

x[x.index(x\_h)] = x[n + 5]  
 continue  
 elif x\_l.f < x[n + 3].f and x[n + 3].f <= x\_s.f:  
 x[x.index(x\_h)] = x[n + 3]  
 continue  
 elif x[n + 3].f > x\_h.f:  
 for j in range(1, n + 2):  
 x[j] = x\_l + 0.5 \* (x[j] - x\_l)  
 continue  
 return 0  
real\_x = (1, -1)  
alpha=2  
beta=0.25  
gamma=2.5  
epsilon=0.0001  
v = [Vector(0, 1), Vector(1, -1), Vector(-1, -1)]  
print("Result of Nelder-Mead algorithm: ")  
minimum, calc\_num = nelder\_mead(func, v, alpha =alpha, beta=beta, gamma=gamma, e  
psilon=epsilon)  
cache = {}  
print("Минимум функции:", minimum)  
print("Количество вычислений функции:", calc\_num)

**5 Анализ результатов**

Для начала найдём реальный минимум функции, для этого произведём  
следующие действия:

Найдём частные производные 1-го порядка:

z'x = 4x + 2y - 2

z'y = 2x + 2y

Приравняем их к нулю и решим систему:

M0: x = 1; y = -1

Найдём частные производные 2-го порядка в точке M0 и проверим  
достаточное условие экстремума:

A = z''xx(M0) = 4

B = z''xy(M0) = 2

C = z''yy(M0) = 2

AC - B^2 > 0

4\*2 - 2^2 > 0

4>0

При этом A>0, следовательно точка (1;-1) - минимум функции.

Теперь получим значения работы метода Нелдера–Мида. Также построим  
графики зависимости количества вычислений целевой функции и отклонения  
от реального минимума в зависимости от изменения параметров метода.  
Результаты представлены на рисунках ниже.

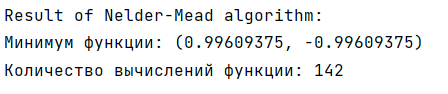


Рисунок 1 – Результат работы метода

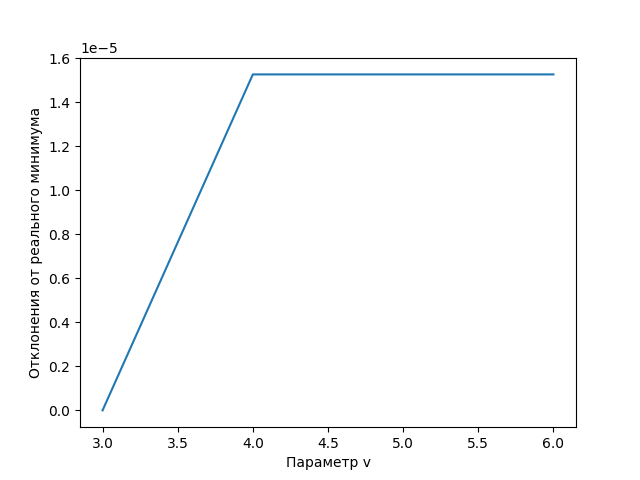


Рисунок 2 – График отклонения от реального минимума при изменении  
количества вершин

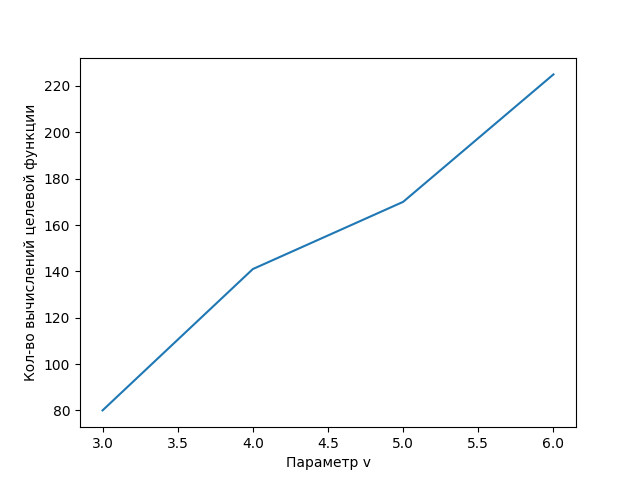


Рисунок 3 – График количества вычислений целевой функции при  
изменении количества вершин

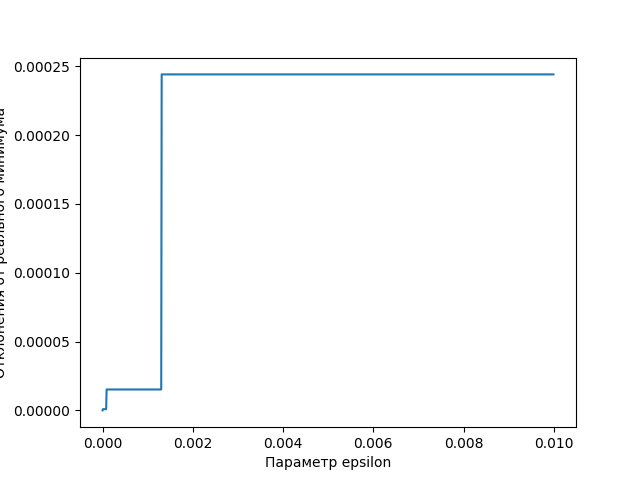


Рисунок 4 – График отклонения от реального минимума для epsilon

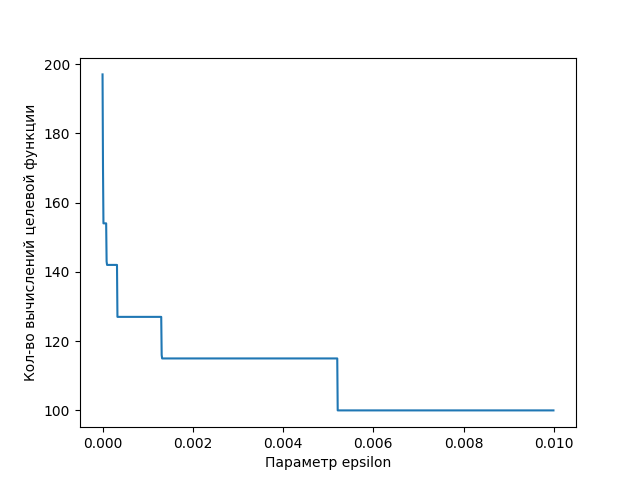


Рисунок 5 – График количества вычислений целевой функции для epsilon

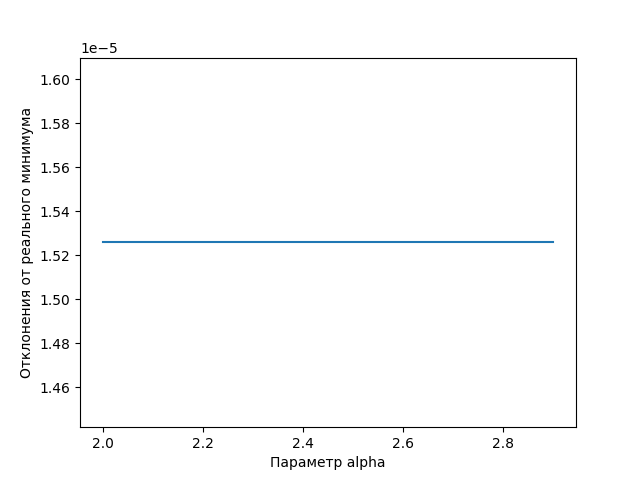


Рисунок 6 – График отклонения от реального минимума для alpha

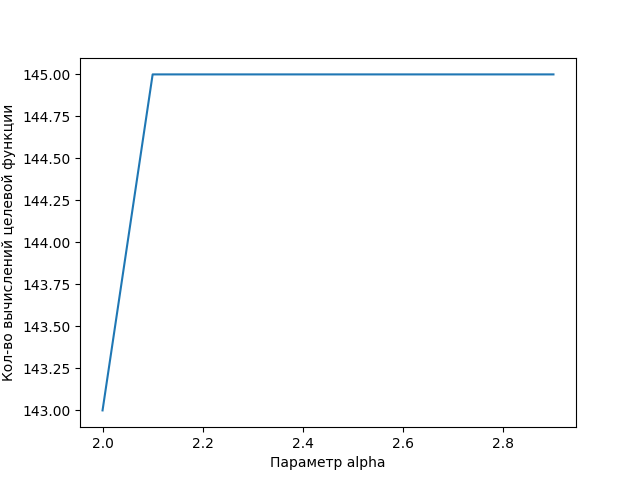


Рисунок 7 – График количества вычислений целевой функции для alpha

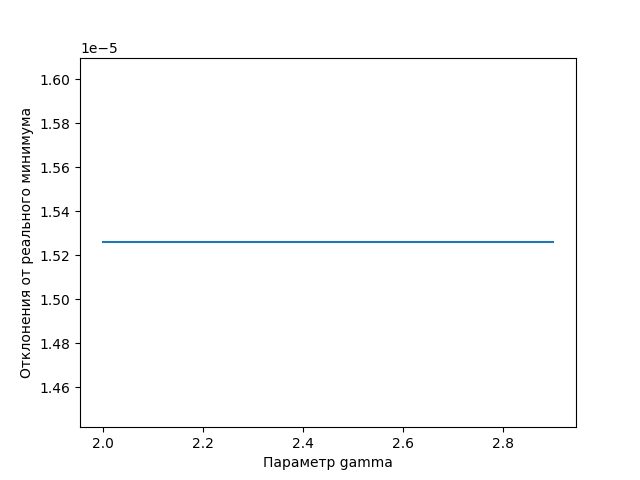


Рисунок 8 – График отклонения от реального минимума для gamma

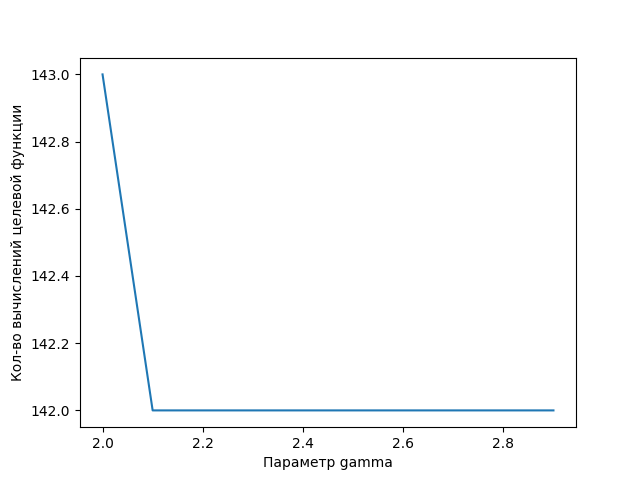


Рисунок 9 – График количества вычислений целевой функции для gamma

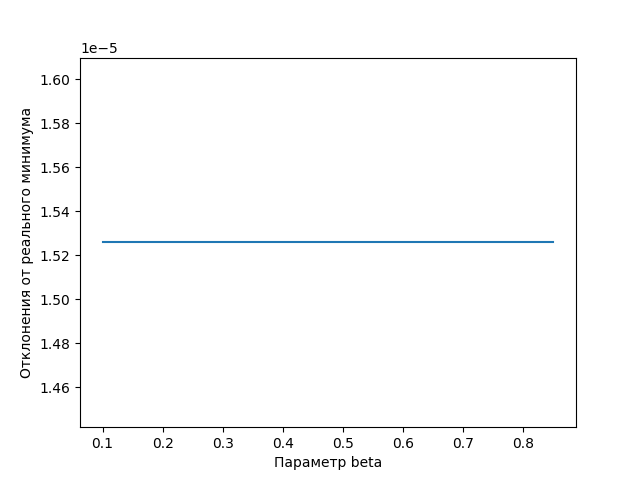


Рисунок 10 – График отклонения от реального минимума для beta

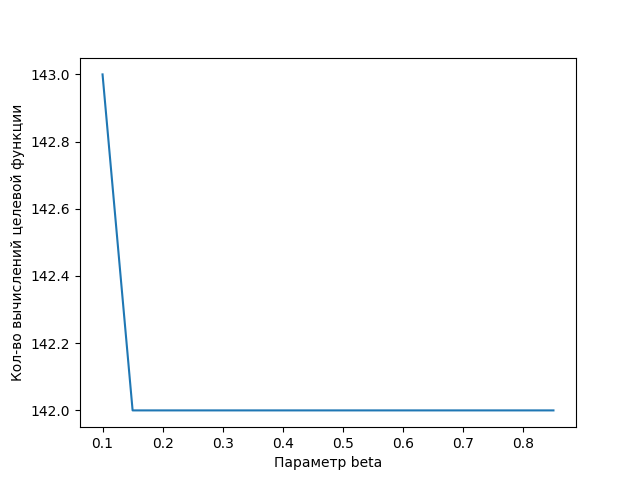


Рисунок 11 – График количества вычислений целевой функции для beta

В результате можно сказать, что найденный минимум функции  
соответствует реальному. При этом наблюдается следующее влияние  
параметров на результат: чем больше вершин многогранника, тем больше  
вычислений функции и отклонение, чем больше эпсилон тем больше  
отклонение, но меньше кол-во вычислений. Влияние параметров alpha, beta и  
gamma для данной функции практически отсутствует.

**6 Вывод**

При выполнении задания был успешно реализован метод Нелдера–Мида,  
результаты работы метода сравнены с реальным и близки к нему, исследована  
зависимость работы метода от значений его параметров.