**Национальный исследовательский технологический университет «МИСиС»**

**Институт Информационных технологий и компьютерных наук (ИТКН)**

**Курс «Методы оптимизации»**

Лабораторная работа № 4

по теме

«Численные методы многомерной оптимизации с использованием производных первого порядка»

Вариант №23

Выполнил:

Студент группы БИВТ-20-1

Смирнов А.А.

Проверил:

к.ф.-м.н., доцент, Лычев А.В.

Москва, 2023

Цель: приобретение практических навыков для решения задач одномерной минимизации численными методами с использованием производной.

# Ход работы:

## Задача

Вариант задания – №23.

Рассматриваемая функция – .

Начальные условия:

Требуется найти безусловный минимум функции многих переменной y = f(, …, ), то есть найти такую точку .

## Листинг программ

### 2.1 Общие методы

Методы вычисления функции и частных производных:

def f(x1, x2):

return 3\*x1\*\*4 - x1\*x2 + x2\*\*4 - 7\*x1 - 8\*x2 + 2

def f\_x1(x1, x2):

return 12\*x1\*\*3 - x2 - 7

def f\_x2(x1, x2):

return 4\*x2\*\*3 - x1 – 8

Методы вычисления градиента и модуля градиента:

def gradient(x1, x2):

i = f\_x1(x1, x2)

j = f\_x2(x1, x2)

return [i, j]

def module\_of\_gradient(grad):

i = 0; j = 1

return sqrt(grad[i]\*\*2 + grad[j]\*\*2)

Одномерный метод минимизации (метод дихотомии):

def dichotomy\_mehod(a, b, epsilon, x1, x2, d1, d2):

x = (a + b) / 2

global counter

counter += 2

if (f(x1 + (x - epsilon)\*d1, x2 + (x - epsilon)\*d2) < f(x1 + (x + epsilon)\*d1, x2 + (x + epsilon)\*d2)):

b = x

else:

a = x

if(abs(b - a) >= 2 \* epsilon):

return dichotomy\_mehod(a, b, epsilon, x1, x2, d1, d2)

return x

### 2.2 Метод Флетчера-Ривза

def the\_fletcher\_reevse\_method(x1, x2, e1, e2, M):

global counter

k = 0

d\_prev = [0, 0]

grad\_prev = 0

while True:

counter += 2

grad = gradient(x1, x2)

module\_grad = module\_of\_gradient(grad)

if ((module\_grad < e1) | (k >= M)):

return [(round(x1, round\_num), round(x2, round\_num), round(f(x1, x2), round\_num)), k]

B = 0

if k % 2 == 1: B = module\_of\_gradient(grad)\*\*2 / module\_of\_gradient(grad\_prev)\*\*2

d = [-grad[0] + B \* d\_prev[0], -grad[1] + B \* d\_prev[1]]

t = dichotomy\_mehod(0, 0.1, e2, x1, x2, d[0], d[1])

x1\_next = x1 + t \* d[0]

x2\_next = x2 + t \* d[1]

x1\_list.append(x1); x2\_list.append(x2)

counter += 1

if ((sqrt(abs(x1\_next - x1)\*\*2 + abs(x2\_next - x2)\*\*2) <= e2)

& (abs(f(x1\_next, x2\_next) - f(x1, x2)) <= e2)):

return [(round(x1\_next, round\_num),

round(x2\_next, round\_num),

round(f(x1\_next, x2\_next), round\_num)),

k]

x1 = x1\_next; x2 = x2\_next

d\_prev = d; grad\_prev = grad

k += 1

### 2.3 Квазиньютоновский метод с симметричной формулой ранга 1

Метод для расчета матрицы:

def calculate\_matrix(H, dx, dy):

dx = np.array(dx)

dy = np.array(dy)

H = np.array(H)

a = dx - np.dot(H, dy)

aT = np.transpose(a)

numerator = np.dot(a, aT)

denominator = np.dot(aT, dy)

H\_next = H + numerator / denominator[0][0]

return H\_next

Реализация квазиньютоновского метода:

def symemetrical\_rank\_1\_formula(x1, x2, e):

global counter

k = 0

grad\_next = np.array([0, 0])

H = np.eye(2)

grad = gradient(x1, x2)

while True:

counter += 2

d = -1 \* np.dot(H, grad)

t = dichotomy\_mehod(0, 2, e\*e, x1, x2, d[0], d[1])

x1\_next = x1 + t \* d[0]

x2\_next = x2 + t \* d[1]

grad\_next = gradient(x1\_next, x2\_next)

if (module\_of\_gradient(grad\_next) < e):

return [(round(x1\_next, round\_num), round(x2\_next, round\_num), round(f(x1\_next, x2\_next), round\_num)), k]

dx1 = t \* d[0]; dx2 = t \* d[1]

dx = [dx1, dx2]

dx = np.reshape(dx, (2,1))

dy = grad\_next - grad

dy = np.reshape(dy, (2,1))

H = calculate\_matrix(H, dx, dy)

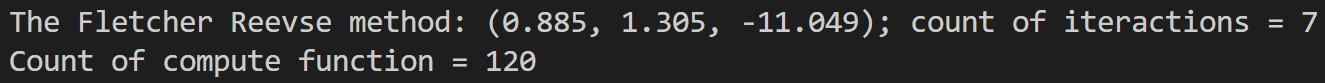
x1\_list.append(x1); x2\_list.append(x2)

x1 = x1\_next; x2 = x2\_next; grad = grad\_next

k += 1

## Результаты вычислений

### 3.1 Метод Флетчера-Ривза



### 3.2 Квазиньютоновский метод с симметричной формулой ранга 1



## Траектории движения

Для всех графиков черная точка определяет начало траектории, а зеленая конец.

### 4.1 Метод Флетчера-Ривза

Начальные условия:

e1 = 0.001

e2 = 0.001

= 10 = 10

M = 100

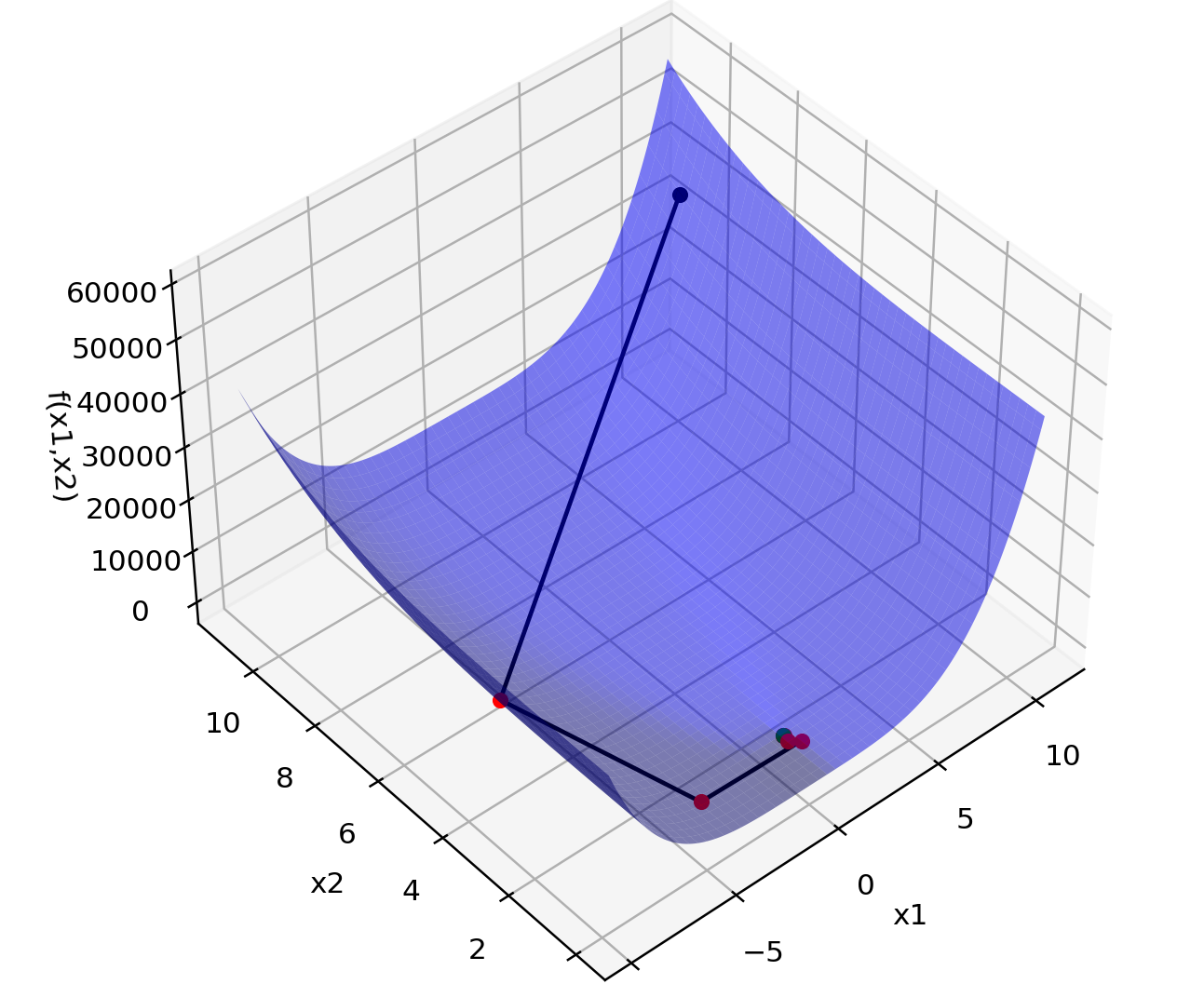


Рисунок 1 – Траектория движения 1.

### 4.2 Квазиньютоновский метод с симметричной формулой ранга 1

Начальные условия:

e1 = 0.001

= 10 = 10

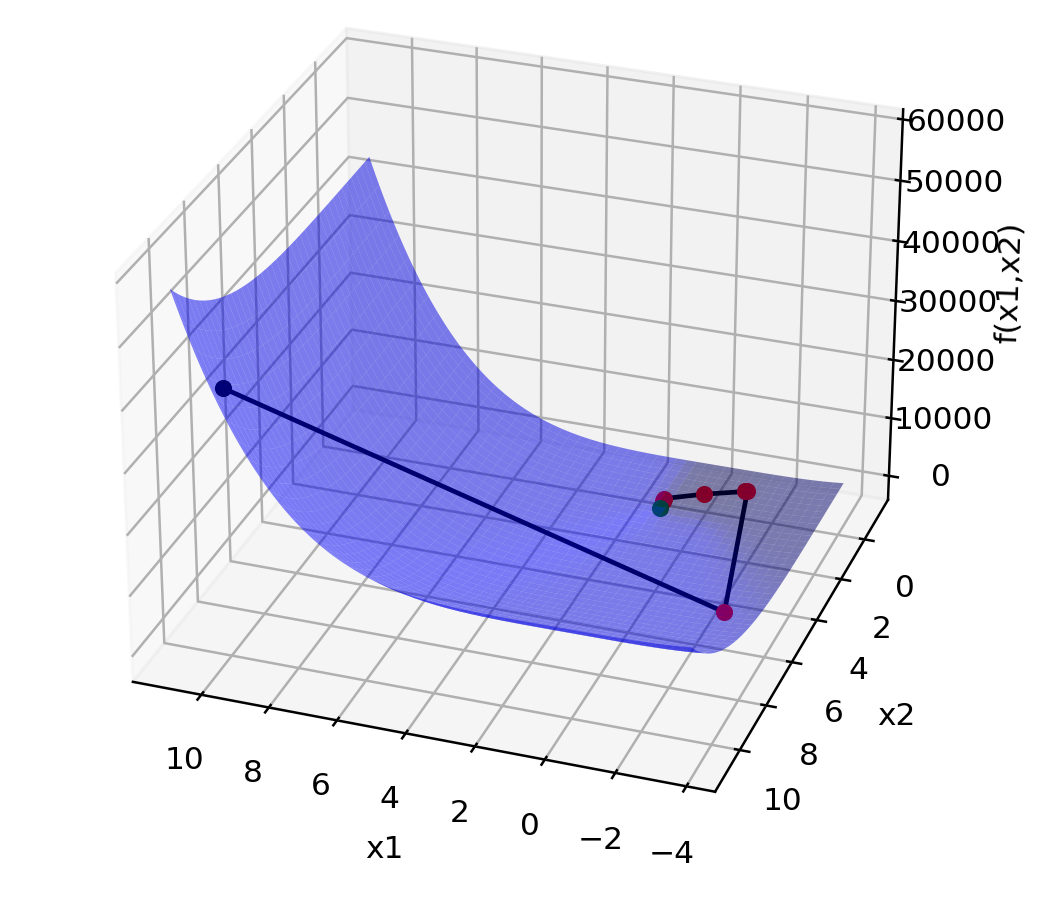


Рисунок 2 – Траектория движения 2.

## Сравнительная характеристика

1. Метод Флетчера-Ривза
2. Квазиньютоновский метод с симметричной формулой ранга 1
3. Метод Ньютона с регулировкой шага

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | e1 | e2 | M |  |  | k | fk |  |  |  |
| 1 | 0.001 | 0.001 | 100 | 0 | 0 | 5 | 90 | 0.885 | 1.305 | -11.049 |
| 0.001 | 0.001 | 100 | -5 | 3 | 6 | 105 | 0.885 | 1.305 | -11.049 |
| 0.001 | 0.001 | 100 | -10 | 17 | 7 | 120 | 0.885 | 1.305 | -11.049 |
| 2 | 0.001 |  | 100 | 0 | 0 | 3 | 88 | 0.885 | 1.305 | -11.049 |
| 0.001 |  | 100 | -5 | 3 | 8 | 198 | 0.885 | 1.305 | -11.049 |
| 0.001 |  | 100 | -10 | 17 | 13 | 308 | 0.885 | 1.305 | -11.049 |
| 3 | 0.001 | 0.001 | 100 | 0 | 0 | 3 | 112 | 0.885 | 1.305 | -11.049 |
| 0.001 | 0.001 | 100 | -5 | 3 | 9 | 260 | 0.885 | 1.305 | -11.049 |
| 0.001 | 0.001 | 100 | -10 | 17 | 10 | 286 | 0.885 | 1.305 | -11.049 |

Вывод: в результате выполнение лабораторной работы я приобрел практические навыки для решения задач многомерной минимизации с использованием производных первого порядка. Для заданных функции нашел минимум двумя способами: метод Флетчера-Ривза и квазиньютоновский метод с симметричной формулой ранга 1.

При близком значении квазиньютоновский метод показал наилучший результат. Однако, при отдалении от точки минимума, данный алгоритм проигрывает остальным. По трем наблюдениям алгоритм Флетчера-Ривза показал наилучший результат.