**Национальный исследовательский технологический университет «МИСиС»**

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

**Курс «Методы оптимизации»**

Лабораторная работа № 3

по теме

«Градиентные методы многомерной минимизации»

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 многомерной минимизации градиентными методами и методами второго порядка.

# Ход работы:

## Задача

Вариант задания – №23.

**Для градиентных методов:**

Рассматриваемая функция – .

Начальные условия:

**Для методов 2-го порядка:**

Рассматриваемая функция – .

Начальные условия:

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

## Листинг программ

### 2.1. Градиентные методы

Вычисление функции и производных:

from math import sqrt

def f(x1, x2):

return 100 \* (x2 - x1\*\*2)\*\*2 + (1 - x1)\*\*2

def f\_x1(x1, x2):

return 400 \* x1\*\*3 + (2 - 400\*x2)\*x1 - 2

def f\_x2(x1, x2):

return 200 \* x2 - 200 \* x1\*\*2

Вычисление 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)

Метод градиентного спуска с постоянным шагом:

def method\_of\_gradient\_descent\_with\_a\_constant\_step(x1, x2, e, M):

k = 0

while True:

grad = gradient(x1, x2)

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

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

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

gamma = 0.5

x1\_next = x1 - gamma \* grad[0]

x2\_next = x2 - gamma \* grad[1]

while (f(x1\_next, x2\_next) - f(x1, x2) >= 0):

gamma /= 2

x1\_next = x1 - gamma \* grad[0]

x2\_next = x2 - gamma \* grad[1]

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

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

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

k += 1

Метод наискорейшего спуска:

def f\_1\_gamma(x1, x2, gamma):

return x1 - gamma \* f\_x1(x1, x2)

def f\_2\_gamma(x1, x2, gamma):

return x2 - gamma \* f\_x2(x1, x2)

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

x = (a + b) / 2

if num == 1:

if (f(f\_1\_gamma(x1, x2, x - epsilon), f\_2\_gamma(x1, x2, x)) < f(f\_1\_gamma(x1, x2, x + epsilon), f\_2\_gamma(x1, x2, x))):

b = x

else:

a = x

else:

if (f(f\_1\_gamma(x1, x2, x), f\_2\_gamma(x1, x2, x - epsilon)) < f(f\_1\_gamma(x1, x2, x) , f\_2\_gamma(x1, x2, x + epsilon))):

b = x

else:

a = x

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

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

return x

def method\_of\_the\_steepest\_gradient\_descent(x1, x2, e, M):

k = 0

while True:

grad = gradient(x1, x2)

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

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

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

gamma\_x1 = dichotomy\_mehod(0, 0.1, e, x1, x2, 1)

gamma\_x2 = dichotomy\_mehod(0, 0.1, e, x1, x2, 2)

x1\_next = x1 - gamma\_x1 \* grad[0]

x2\_next = x2 - gamma\_x2 \* grad[1]

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

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

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

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

k += 1

### 2.2. Методы второго порядка

Вычисление функции и 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

def f\_x1\_x1(x1, x2):

return 36\*x1\*\*2

def f\_x2\_x2(x1, x2):

return 12\*x2\*\*2

def f\_x1\_x2(x1, x2):

return -1

def f\_x2\_x1(x1, x2):

return -1

def gradient(x1, x2):

i = f\_x1(x1, x2)

j = f\_x2(x1, x2)

return [i, j]

Вычисление градиента:

def module\_of\_gradient(grad):

i = 0; j = 1

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

Вычисление инвертированной матрицы Гессе:

def inverse\_gesse\_matrix(x1, x2):

maxtix = [[f\_x1\_x1(x1, x2), f\_x1\_x2(x1, x2)], [f\_x2\_x1(x1, x2), f\_x2\_x2(x1, x2)]]

A = maxtix[0][0] \* maxtix[1][1] - maxtix[1][0] \* maxtix[0][1]

if A == 0:

return None

A = abs(A)

inverse\_maxtix = [[maxtix[1][1] / A, maxtix[1][0] / A], [maxtix[0][1] / A, maxtix[0][0] / A]]

return inverse\_maxtix

Метод дихотомии:

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

x = (a + b) / 2

def f\_inner(xk, x, dk):

return xk + x \* dk

if num == 1:

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

b = x

else:

a = x

else:

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

b = x

else:

a = x

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

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

return x

Вычисление обратной матрицы:

def is\_matrix\_negative(a, b, c, d):

sqrt\_num = sqrt(a\*\*2 + 4\*b\*c + d\*\*2 - 2\*a\*d)

l1 = (a + b - sqrt\_num) / 2

l2 = (a + b + sqrt\_num) / 2

return ((l1 > 0) and (l2 > 0))

Метод Ньютона с регулировкой шага:

def newtons\_method\_with\_step\_adjustment(x1, x2, e, M):

k = 0

while True:

grad = gradient(x1, x2)

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

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

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

inverse\_maxtix = inverse\_gesse\_matrix(x1, x2)

d = []

if (is\_matrix\_negative(inverse\_maxtix[0][0], inverse\_maxtix[0][1], inverse\_maxtix[1][0], inverse\_maxtix[1][1]) == False):

d = [-1 \* inverse\_maxtix[0][0] \* grad[0] + -1 \* inverse\_maxtix[0][1] \* grad[1], -1 \* inverse\_maxtix[1][0] \* grad[0] + -1 \* inverse\_maxtix[1][1] \* grad[1]]

else:

d = [-1 \* grad[0], -1 \* grad[1]]

t\_1 = dichotomy\_mehod(0, 1, e, x1, x2, 1, d[0], d[1])

t\_2 = dichotomy\_mehod(0, 1, e, x1, x2, 2, d[0], d[1])

x1\_next = x1 + t\_1 \* d[0]

x2\_next = x2 + t\_2 \* d[1]

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

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

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

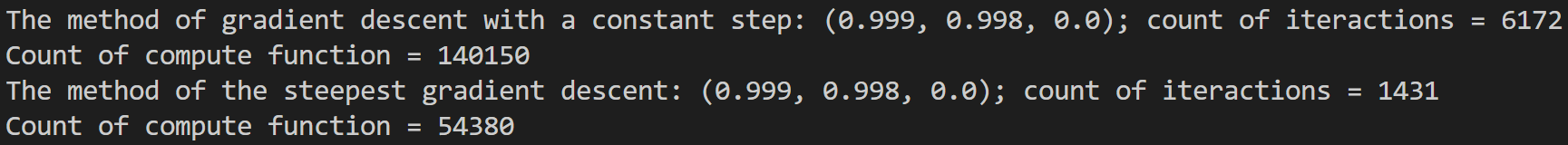
x1 = x1\_next

x2 = x2\_next

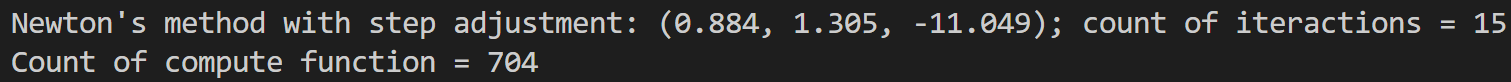
k += 1

## Результаты вычислений

### 3.1. Градиентные методы:



### 3.2. Метод второго порядка:



## Точные значения минимума и координаты точки, где он достигается

4.1. Градиентные методы

Критические точки: (1, 1) и (-1, 1)

Минимум в точке (1;1), minf(

## Траектории движения

### 6.1. Метод градиентного спуска с постоянным шагом

Начальные условия:

e1 = 0.001

e1 = 0.000001

M = 10000

= 0.5

= 0.5

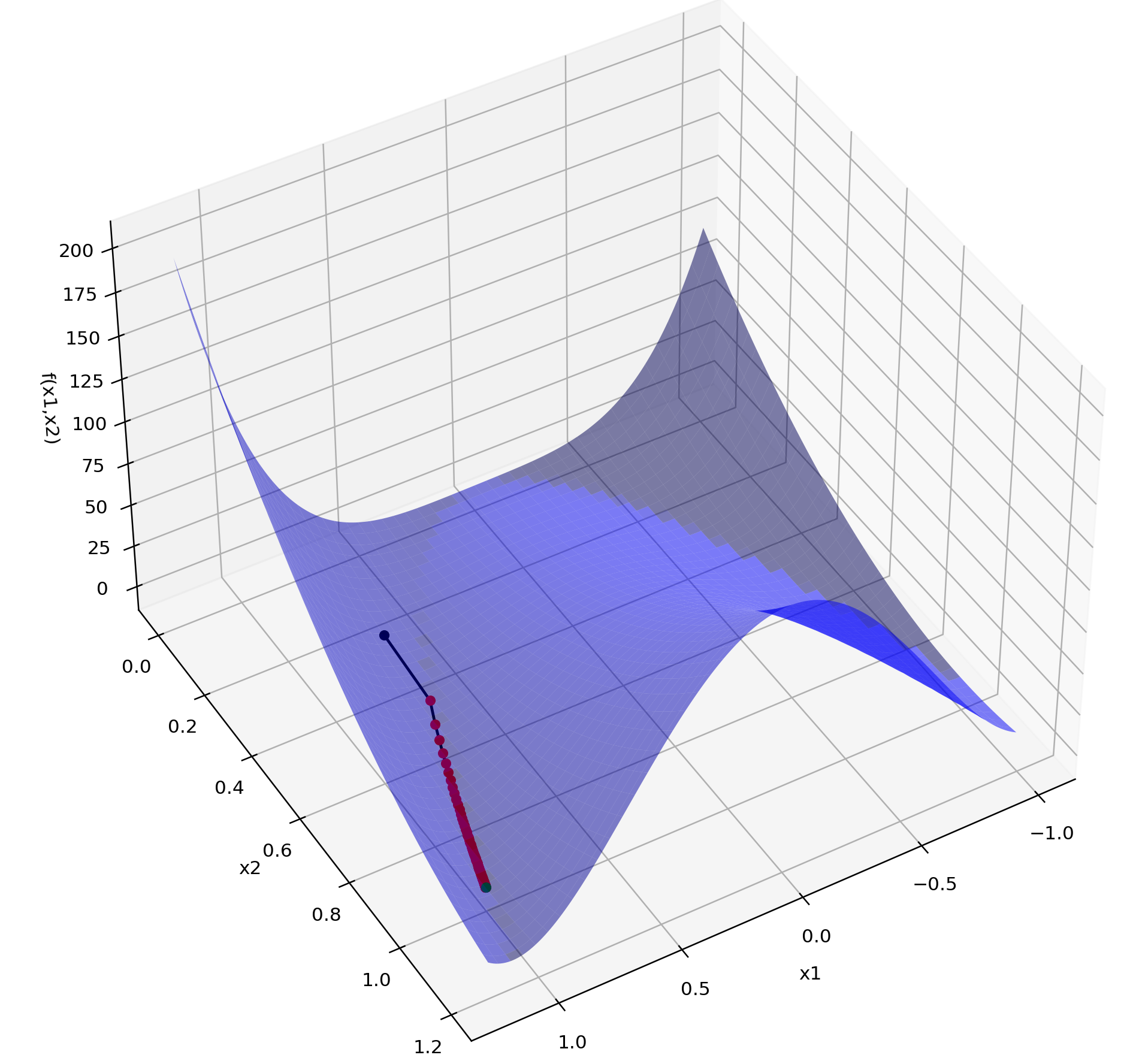


Рисунок 1 – Траектория движения 1.

### 6.2. Метод наискорейшего спуска

Начальные условия:

e1 = 0.001

e1 = 0.000001

M = 10000

= 0.5

= 0.5

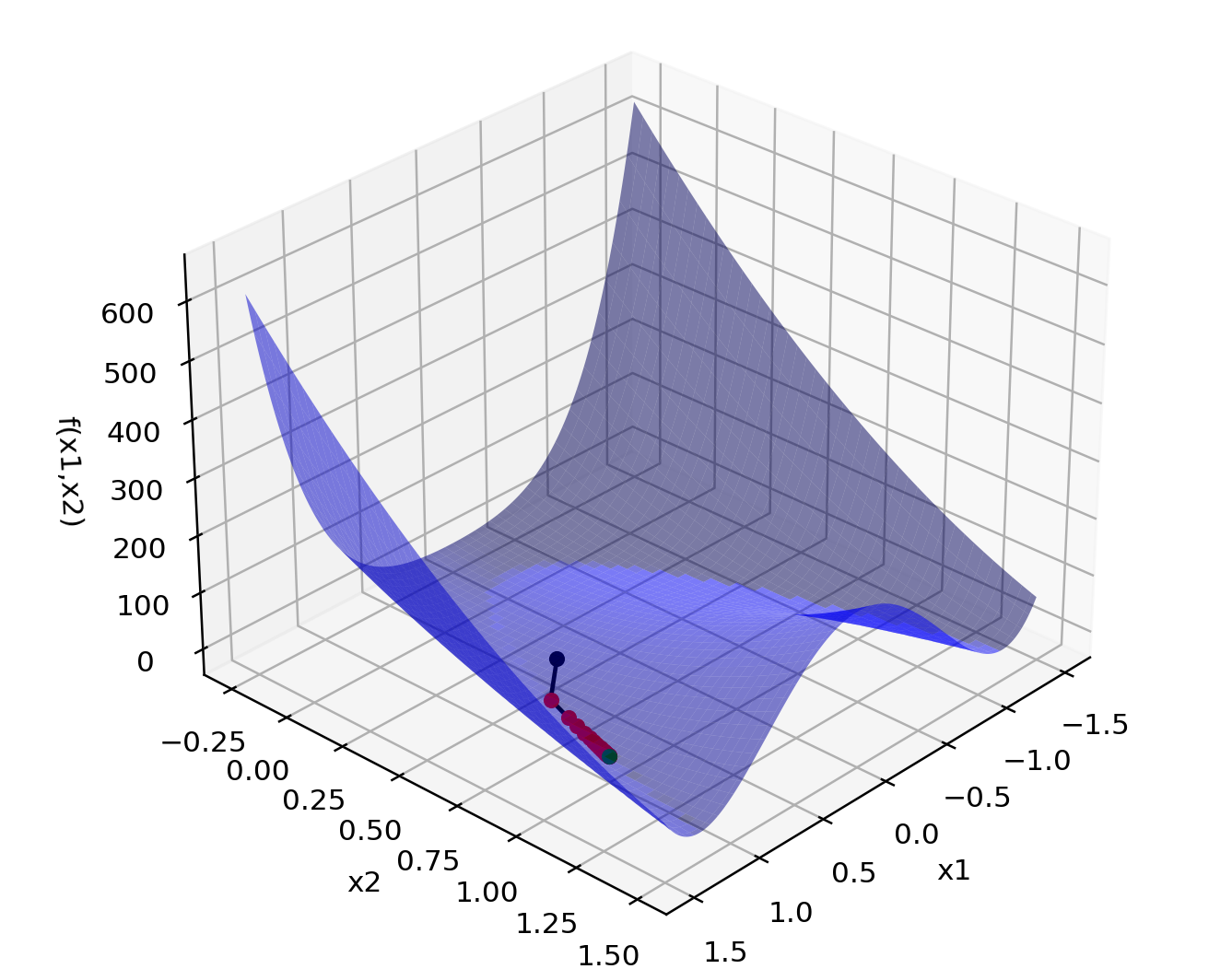


Рисунок 2 – Траектория движения 2.

### 6.3. Метод Ньютона с регулировкой шага

e1 = 0.001

e1 = 0.000001

M = 10000

= 0

= 0

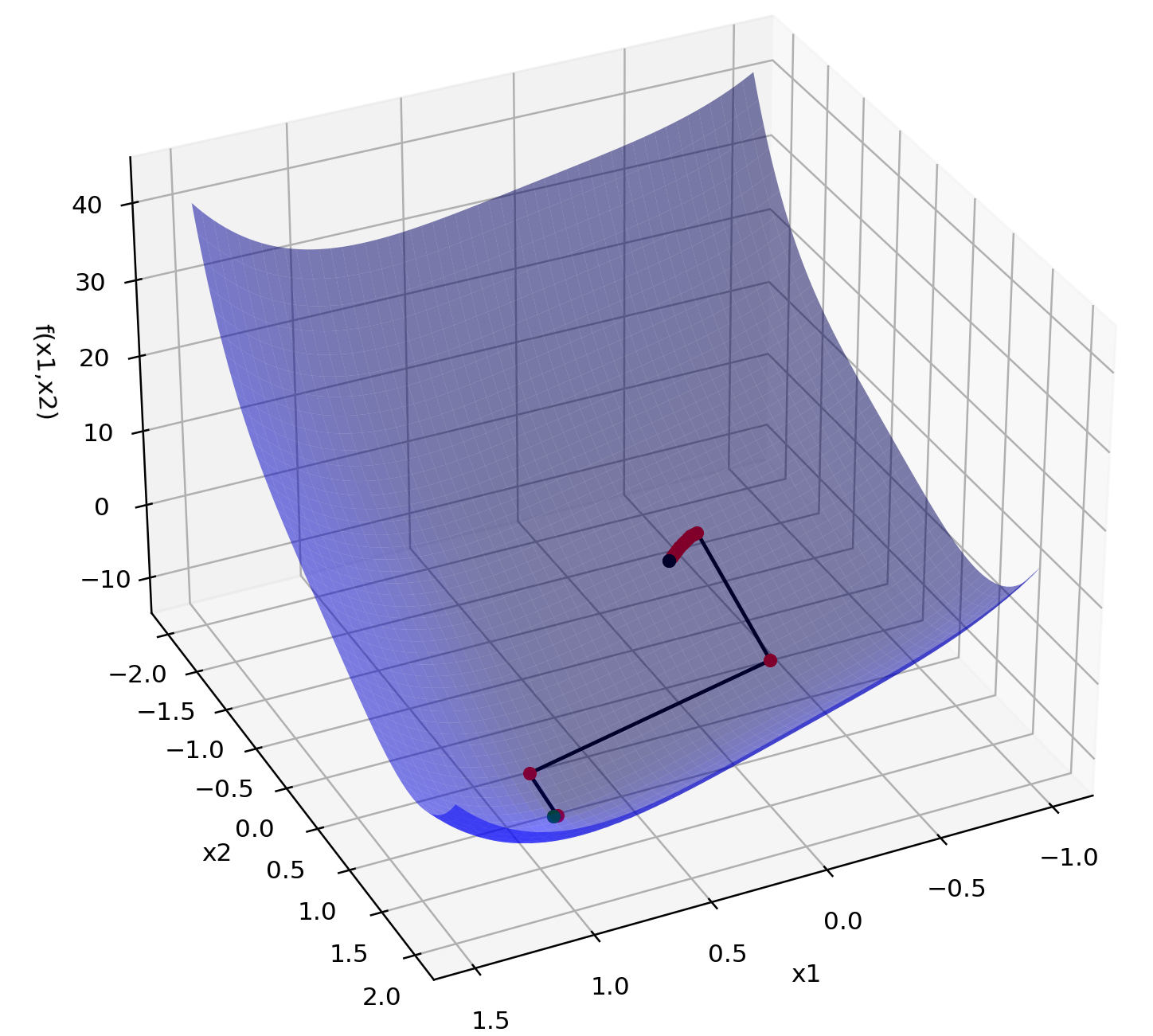


Рисунок 3 – Траектория движения 3.

## Сравнительная характеристика

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | e1 | e2 | M |  |  | k | fk |  |  |  |
| 1 | 0.001 | 0.000001 | 10000 | 0.5 | 0.5 | 6172 | 140150 | 0.999 | 0 .998 | 0.000 |
| 0.001 | 0.00001 | 10000 | 0.5 | 0.5 | 4219 | 95427 | 0.996 | 0 .992 | 0.000 |
| 0.01 | 0.0001 | 10000 | 0.5 | 0.5 | 1631 | 36321 | 0.963 | 0 .928 | 0.001 |
| 2 | 0.001 | 0.000001 | 10000 | 0.5 | 0.5 | 1431 | 54380 | 0.999 | 0 .998 | 0.000 |
| 0.001 | 0.00001 | 10000 | 0.5 | 0.5 | 968 | 36822 | 0.996 | 0 .991 | 0.000 |
| 0.001 | 0.0001 | 10000 | 0.5 | 0.5 | 414 | 15770 | 0.970 | 0 .941 | 0.001 |
| 3 | 0.001 | 0.001 | 100 | 0 | 0 | 15 | 704 | 0.884 | 1.305 | -11.049 |
| 0.001 | 0.001 | 100 | 10 | 10 | 8 | 396 | 0.885 | 1.305 | -11.049 |
| 0.001 | 0.001 | 100 | 100 | 100 | 14 | 660 | 0.885 | 1.305 | -11.049 |

Вывод: в результате выполнение лабораторной работы я приобрел практические навыки для решения задач многомерной минимизации градиентными методами и методами второго порядка. Для заданных функции нашел минимум тремя способами: метод градиентного спуска с постоянным шагом, метод наискорейшего спуска, метод ньютона с регулировкой шага.

Метод наискорейшего спуска потребовал в разы меньше итераций и вычислений функции и производных, чем метод градиентного спуска с постоянным шагом. Однако оба этих метода показали свою неэффективность на овражной функции.

Метод ньютона с регулировкой шага показал лучшие результаты и самую быструю сходимость.