Министерство науки и высшего образования Российской Федерации
Муромский институт
федерального государственного бюджетного образовательного учреждения
высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»**
(МИВлГУ)

Факультет ИТ

Кафедра ИС

ПРАКТИЧЕСКАЯ РАБОТА №11

по Специальным главам математики

Тема Методы многомерной оптимизации

Руководитель

Щаников С. А.
(фамилия, инициалы)

(подпись) (дата)

Студент ИСм-121
(группа)

Минеев Р. Р.
(фамилия, инициалы)

(подпись) (дата)

Муром 2021

Практическая работа №11.

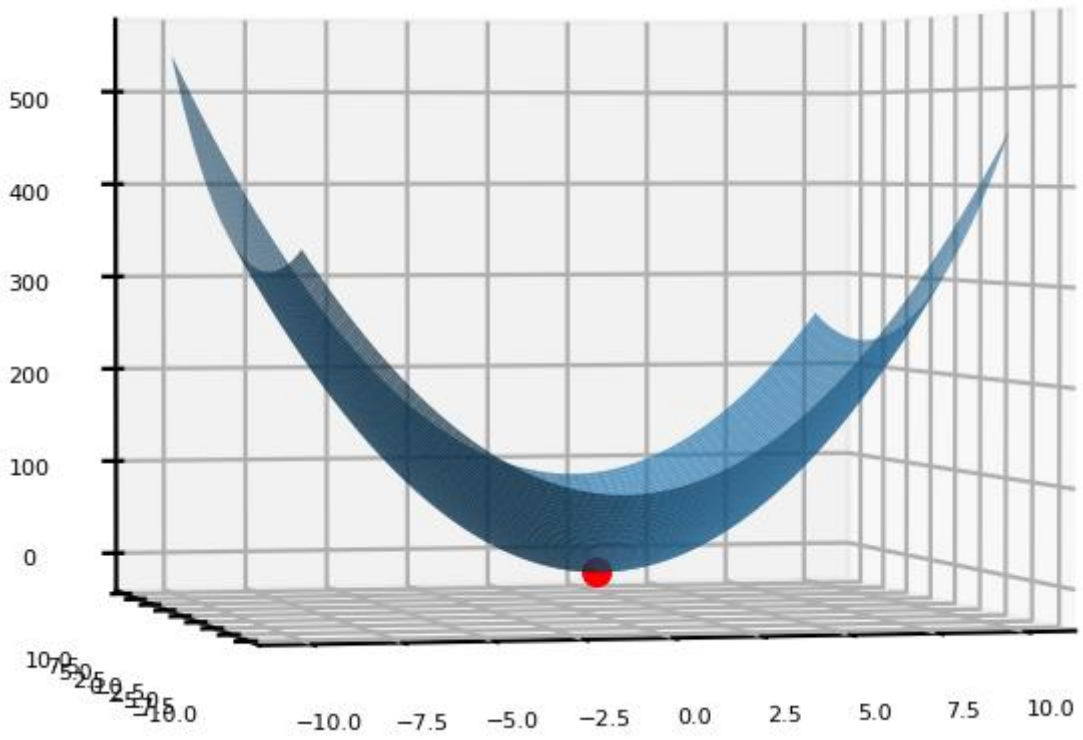
Тема: Методы многомерной оптимизации.

Задание на работу: задать функцию двух аргументов и визуализировать поиск экстремума.

```
import numpy as np
import matplotlib.pyplot as plot

def rotate_vector(length, a):
    return length * np.cos(a), length * np.sin(a)

def derivative_x(epsilon, arg):
    return (func(g_e + epsilon, arg) - func(epsilon, arg)) / g_e

def derivative_y(epsilon, arg):
    return (func(arg, epsilon + g_e) - func(arg, epsilon)) / g_e

def calculate_flip_points():
    flip_points = np.array([0, 0])
    points = np.zeros((360, arr_shape), dtype=bool)
    cx, cy = centre
    for i in range(arr_shape):
        for alpha in range(360):
            x, y = rotate_vector(step, alpha)
            x, y = x * i + cx, y * i + cy
            points[alpha][i] = derivative_x(x, y) + derivative_y(y, x) > 0
            if not points[alpha][i-1] and points[alpha][i]:
                flip_points = np.vstack((flip_points, np.array([alpha, i - 1])))
    return flip_points


def pick_estimates(positions):
    vx, vy = rotate_vector(step, positions[1][0])
    cx, cy = centre
    best_x, best_y = cx + vx * positions[1][1], cy + vy * positions[1][1]

    for index in range(2, len(positions)):
        vx, vy = rotate_vector(step, positions[index][0])
        x, y = cx + vx * positions[index][1], cy + vy * positions[index][1]
        if func(best_x, best_y) > func(x, y):
            best_x = x
            best_y = y
```

```
            for index in range(360):
                vx, vy = rotate_vector(step, index)
                x, y = cx + vx * (arr_shape - 1), cy + vy * (arr_shape - 1)
                if func(best_x, best_y) > func(x, y):
                    best_x = x
                    best_y = y

            return best_x, best_y


        def gradient_descent(best_estimates, is_x):
            derivative = derivative_x if is_x else derivative_y
            best_x, best_y = best_estimates
            value = derivative(best_y, best_x)

            descent_step = step
            while abs(value) > g_e:
                descent_step *= 0.95
                best_y = best_y - descent_step if derivative(best_y, best_x) > 0 else best_y +
descent_step
                value = derivative(best_y, best_x)

            return best_y, best_x

        def find_minimum():
            return gradient_descent(gradient_descent(pick_estimates(calculate_flip_points()), False),
True)

        def get_grid(grid_step):
            samples = np.arange(-r, r, grid_step)
            x, y = np.meshgrid(samples, samples)
            return x, y, func(x, y)

        def draw_chart(point, grid):
            point_x, point_y, point_z = point
            grid_x, grid_y, grid_z = grid
            plot.rcParams.update({
                'figure.figsize': (4, 4),
                'figure.dpi': 200,
                'xtick.labelsize': 4,
                'ytick.labelsize': 4
            })
            ax = plot.figure().add_subplot(111, projection='3d')
            ax.scatter(point_x, point_y, point_z, color='red')
            ax.plot_surface(grid_x, grid_y, grid_z, rstride=5, cstride=5, alpha=0.7)
            plot.show()


        if __name__ == '__main__':
            r = 10
            g_e = 0.00005
            centre = (g_e, g_e)
            arr_shape = 100
```

```
step = r / arr_shape
func = lambda x,y: 3*x**2 + y**2 - x*y - 4*x
min_x, min_y = find_minimum()
minimum = (min_x, min_y, func(min_x, min_y))
draw_chart(minimum, get_grid(0.05))
```



Вывод: в данной практической работе были получены навыки использования методов многомерной оптимизации для поиска оптимальных значений пользовательских функций численными методами.