Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики Национального исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

ОТЧЕТ
По лабораторной работе №5
на тему
«МИНИМИЗАЦИЯ ФУНКЦИЙ»

ФИО студента	Номер группы	Дата	Вариант
Ткаченко Никита Андреевич	БПМ211	27.05.2024	9.1.21, 9.4.7, 9.5.21, 9.6.21

Москва – 2024г.

9.1. Метод Ньютона

Задача 9.1 Методом Ньютона найти минимум и максимум унимодальной на отрезке $[a,b]$ функции $f(x)$ с точностью $\varepsilon = 10^{-6}$. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

9.1.21	$x^5 - 5^x$	0.5	1.5

Метод Ньютона описывается простой итерационной формулой, использующей две производные заданной функции: $x^{(n+1)} = x^{(n)} - \frac{f'(x^{(n)})}{f''(x^{(n)})}, \quad n \geqslant 0.$

Метод Ньютона сходится квадратично, поэтому как критерий остановки можно использовать выражение: $|x^{(n)} - x^{(n-1)}| < \varepsilon.$

Реализуем:

```
def newton_method(f, df, ddf, x0, epsilon=1e-6, max_iter=1000):
    x_n = x0
    for iterations in range(max_iter):
        f_prime = df(x_n)
        f_double_prime = ddf(x_n)
        if abs(f_double_prime) < np.finfo(float).eps:
            raise ValueError("Вторая производная слишком мала, деление на ноль")
        x_n1 = x_n - f_prime / f_double_prime
        if abs(x_n1 - x_n) < epsilon:
            return x_n1, iterations
        x_n = x_n1
    raise ValueError("Метод не сошелся за максимальное количество итераций")
```

Зададим вручную функцию и ее производные:

```
f = lambda x: x**5 - 5**x
df = lambda x: 5*x**4 - 5**x*np.log(5)
ddf = lambda x: 20*x**3 - 5**x*np.log(5)*np.log(5)
```

Тогда:

```
x_min, n = newton_method(f, df, ddf, x0, epsilon)
print(f"Минимум функции достигается в точке x = {x_min}, найдет за {n} итераций.")
```

Минимум функции достигается в точке x = 1.2410554672442062, найдет за 5 итераций.

9.4 Метод Золотого Сечения

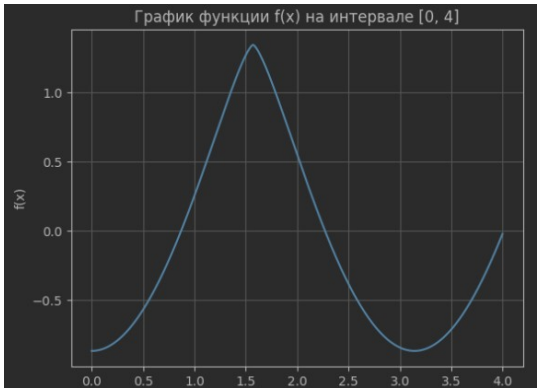
Задача 9.4. Функция $f(x)$ представлена частичной суммой ряда $f(x)=\sum_{i=1}^{n} u_i(x)$.

Построить график функции на заданном отрезке $[x_1, x_2]$ и найти ее минимумы и максимумы с указанной точностью ε.

9.4.7	$(-1)^n \frac{\cos(2nx)}{n^{5/2}}$	0	4	350	0.0001	Золотого сечения

Для начала зададим нашу функцию, просуммировав заданное выражение 350 раз:

```
def f(x, n):
    return ((-1)**n) * np.cos(2 * n * x) / n**(5/2)

def partial_sum(x, N=350):
    return sum(f(x, n) for n in range(1, N + 1))

def func_to_optimize(x):
    return partial_sum(x)
```



Рассмотрим метод золотого сечения:

Метод основан на делении заданного отрезка функции в пропорциях золотого сечения (т. е. 0.618 к 0.382).

На первой итерации заданный отрезок делится двумя симметричными относительно его центра точками и рассчитываются значения в этих точках. После чего тот из концов отрезка, к которому среди двух вновь поставленных точек ближе оказалась та, значение в которой максимально, отбрасывают. Процедура продолжается до тех пор, пока не будет достигнута заданная точность.

```
def golden_section_search(f, a, b, eps=1e-4):
    golden_ratio_inverted = 2 / (1 + np.sqrt(5))
    c = b - golden_ratio_inverted / (b - a)
    d = a + golden_ratio_inverted / (b - a)
    while abs(c - d) > eps:
        if f(c) < f(d):
            b = d
        else:
            a = c
        c = b - golden_ratio_inverted / (b - a)
        d = a + golden_ratio_inverted / (b - a)
    return (b + a) / 2
```

```
minimum = golden_section_search(func_to_optimize, a, b)
maximum = golden_section_search(lambda x: -func_to_optimize(x), a, b)

print(f"Минимум функции достигается в точке x = {minimum} со значением f(x) =
{func_to_optimize(minimum)}")
print(f"Максимум функции достигается в точке x = {maximum} со значением f(x) =
{func_to_optimize(maximum)}")
```

Минимум функции достигается в точке x = 3.141596527569854 со значением f(x) = -0.8671996716015662
Максимум функции достигается в точке x = 1.5708643707462782 со значением f(x) = 1.3413853281698072

9.5 Экстремумы функции двух переменных

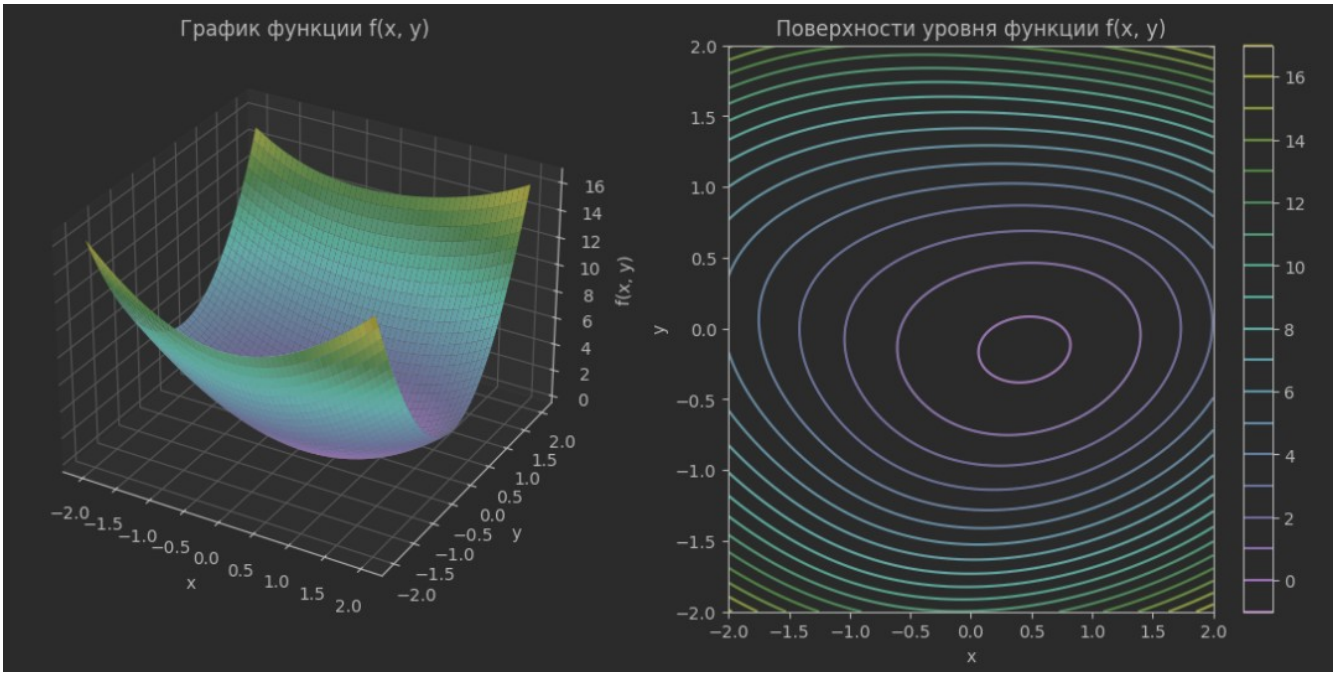
Задача 9.5. Найти минимум функции 2-х переменных $f(x, y)$ с точностью $\varepsilon = 10^{-6}$ на прямоугольнике $[x_1, x_2] \times [y_1, y_2]$.

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Задать указанную в варианте функцию $f(x, y)$.
2. Построить графики функции и поверхностей уровня $f(x, y)$.
3. По графикам найти точки начального приближения к точкам экстремума.
4. Используя встроенные функции, найти экстремумы функции с заданной точностью (см. *ПРИЛОЖЕНИЕ 9.B*).

9.5.21	$x^2 + 3y^2 + \sin(x - y + 3)$	-2	2	-2	2

```
def f(xy):
    x, y = xy
    return x**2 + 3*y**2 + np.sin(x - y + 3)

x = np.linspace(-2, 2, 400)
y = np.linspace(-2, 2, 400)
X, Y = np.meshgrid(x, y)
Z = f([X, Y])
```




```
initial_guess = [0, 0]
result = minimize(f, initial_guess, bounds=[(-2, 2), (-2, 2)], tol=1e-6)
```

Минимум функции достигается в точке x = 0.448742081438306, y = -0.14958301051635287 со значением f(x, y) = -0.17252314864696905

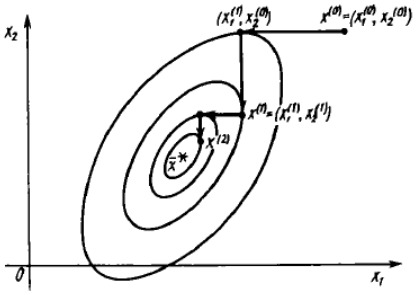
9.6 Метод Покоординатного Спуска

Задача 9.6. Указанным в индивидуальном варианте методом найти минимум квадратичной функции $f(x,y)=a_{11}x^2+2a_{12}xy+a_{22}y^2+2a_{13}x+2a_{23}y$ с точностью $\varepsilon=10^{-6}$. Для решения задачи одномерной минимизации использовать метод Ньютона. Построить график функции f. Предусмотреть подсчет числа итераций, потребовавшихся для достижения заданной точности.

9.6.21	4	0.5	0.5	6.5	-2.5	Покоординатный спуск

Итоговая функция: `4*x1**2 + 0.5*x1*x2 + 0.5*x2**2 + 6.5*x1 - 2.5*x2`

Метод покоординатного спуска, как и любой метод спуска, основывается на выборе направления спуска и шага спуска. В данном методе направление спуска — поочередно координатные оси, дял каждого направления остальные направления фиксируются, и методом одномерной оптимизации находится значение k-й координаты (учитывая значения уже найденных k-1 координат).



Критерий остановки можно выбрать какой угодно, из списка:

$$|x^{(n+1)} - x^{(n)}| < \varepsilon_1,$$
$$|f(x^{(n+1)}) - f(x^{(n)})| < \varepsilon_2,$$
$$|f'(x^{(n)})| < \varepsilon_3,$$

- я выбрал первый.

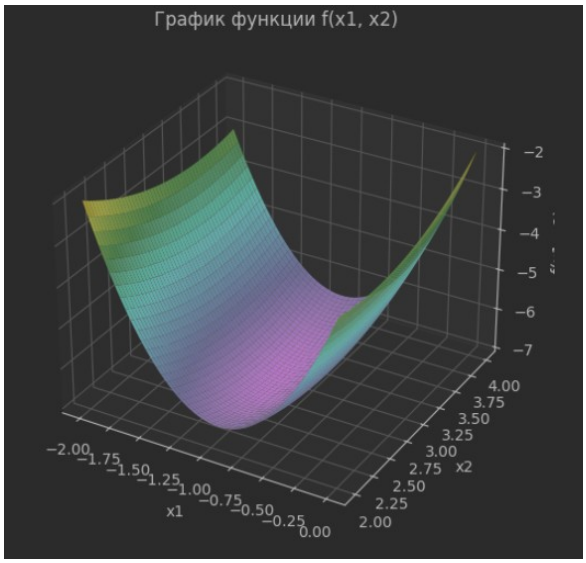
Зададим вручную функцию и ее производные по каждой из координат (для метода Ньютона):

```
def f(x):
    x1, x2 = x
    return 4*x1**2 + 0.5*x1*x2 + 0.5*x2**2 + 6.5*x1 - 2.5*x2

def df_dx1(x):
    x1, x2 = x
    return 8*x1 + 0.5*x2 + 6.5

def df_dx2(x):
    x1, x2 = x
    return 0.5*x1 + x2 - 2.5

def d2f_dx1(x):
    return 8

def d2f_dx2(x):
    return 1
```

Покоординатный спуск (видна одномерная оптимизация для каждой переменной):

```
def coordinate_descent(f, x0, tol=1e-6, max_iter=1000):
    x = np.array(x0, dtype=float)
    iterations = 0
    for iterations in range(max_iter):
        x_old = np.copy(x)

        # Минимизация по x1
        x1_minimization_function = lambda x1: f([x1, x[1]])
        df_x1_minimization = lambda x1: df_dx1([x1, x[1]])
        x[0], iterations_1 = newton_method(x1_minimization_function, df_x1_minimization, d2f_dx1, x[0])

        # Минимизация по x2
        x2_minimization_function = lambda x2: f([x[0], x2])
        df_x2_minimization = lambda x2: df_dx2([x[0], x2])
        x[1], iter2 = newton_method(x2_minimization_function, df_x2_minimization, d2f_dx2, x[1])

        if np.linalg.norm(x - x_old) < tol:
            break

    return x, iterations
```

График функции:



Минимум функции достигается в точке x1 = -0.9999999944120646, x2 = 2.9999999972060323 со значением f(x) = -7.0, Количество итераций: 5