|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Калужский филиал федерального государственного автономного образовательного учреждения высшего образования**  **«Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»**  **(КФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИУК Информатика и управление

КАФЕДРА ИУК4 Программное обеспечение ЭВМ, информационные технологии

**ПРАКТИЧЕСКАЯ РАБОТА**

**«Методы минимизации»**

**по дисциплине: «Методы машинного обучения»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил: студент группы ИУК4-72Б | |  |  | Губин Е.В. | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Проверил: | |  |  | Семененко М.Г. | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | | | |
|  | - Балльная оценка:  - Оценка: | | | |

Калуга, 2025

**Цель:** изучить и реализовать метод половинного деления для нахождения минимума функции одной переменной и сравнить его эффективность с встроенными методами оптимизации.

**Задачи:**

* Реализовать алгоритм метода половинного деления и метода градиентного спуска для поиска минимума функции.
* Применить методы к заданной функции
* Построить график функции с отмеченной точкой минимума и сравнить результат с встроенной функцией Python.

**Вариант №4**

**Формулировка задания №1:**

1. Написать блок-схему алгоритма нахождения минимума функции методом половинного деления.
2. Методом половинного деления найти минимум функции J(u) на отрезке [-10; 10] (по вариантам). Построить график функции и показать на нем точку минимума. Функция: ; a = 4; b = -0.25.
3. Сравнить результат с результатом использования встроенной функции.

**Блок схема метода дихотомии:**

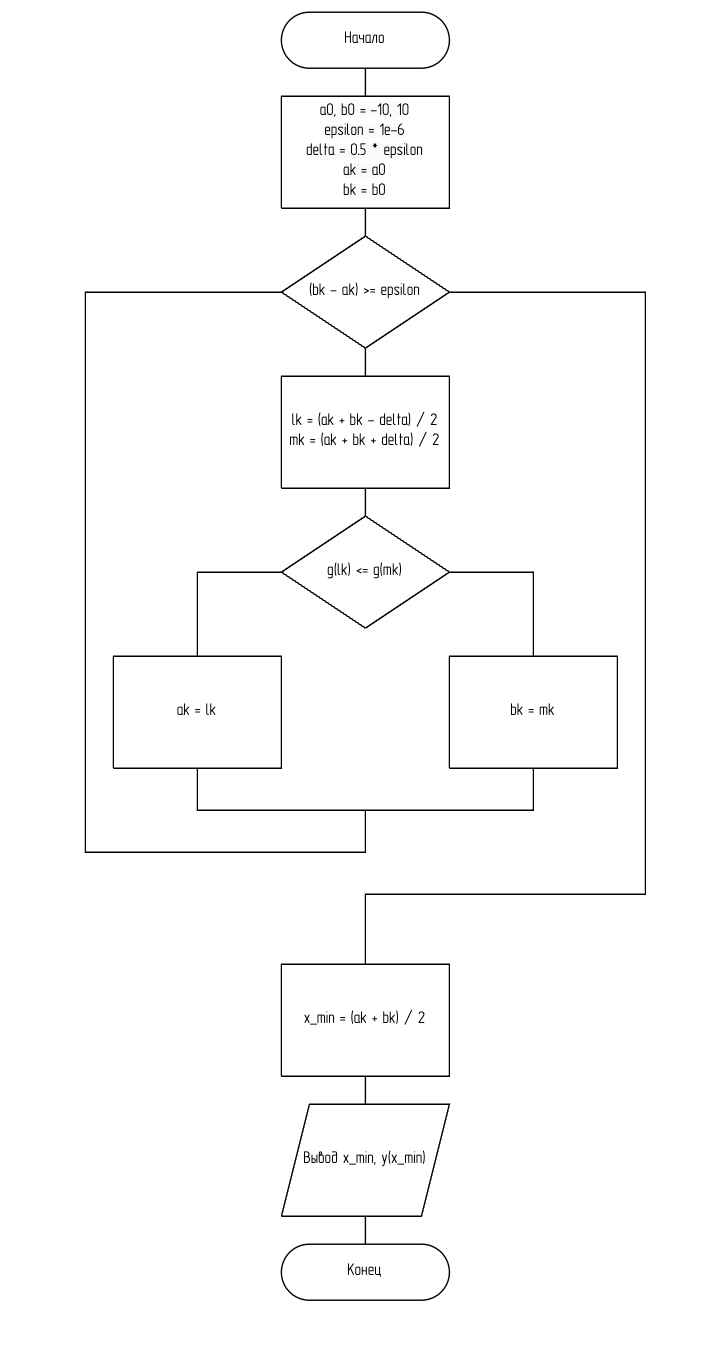


Рисунок Блок схема метода дихотомии

**Результаты выполнения программы для метода дихотомии:**

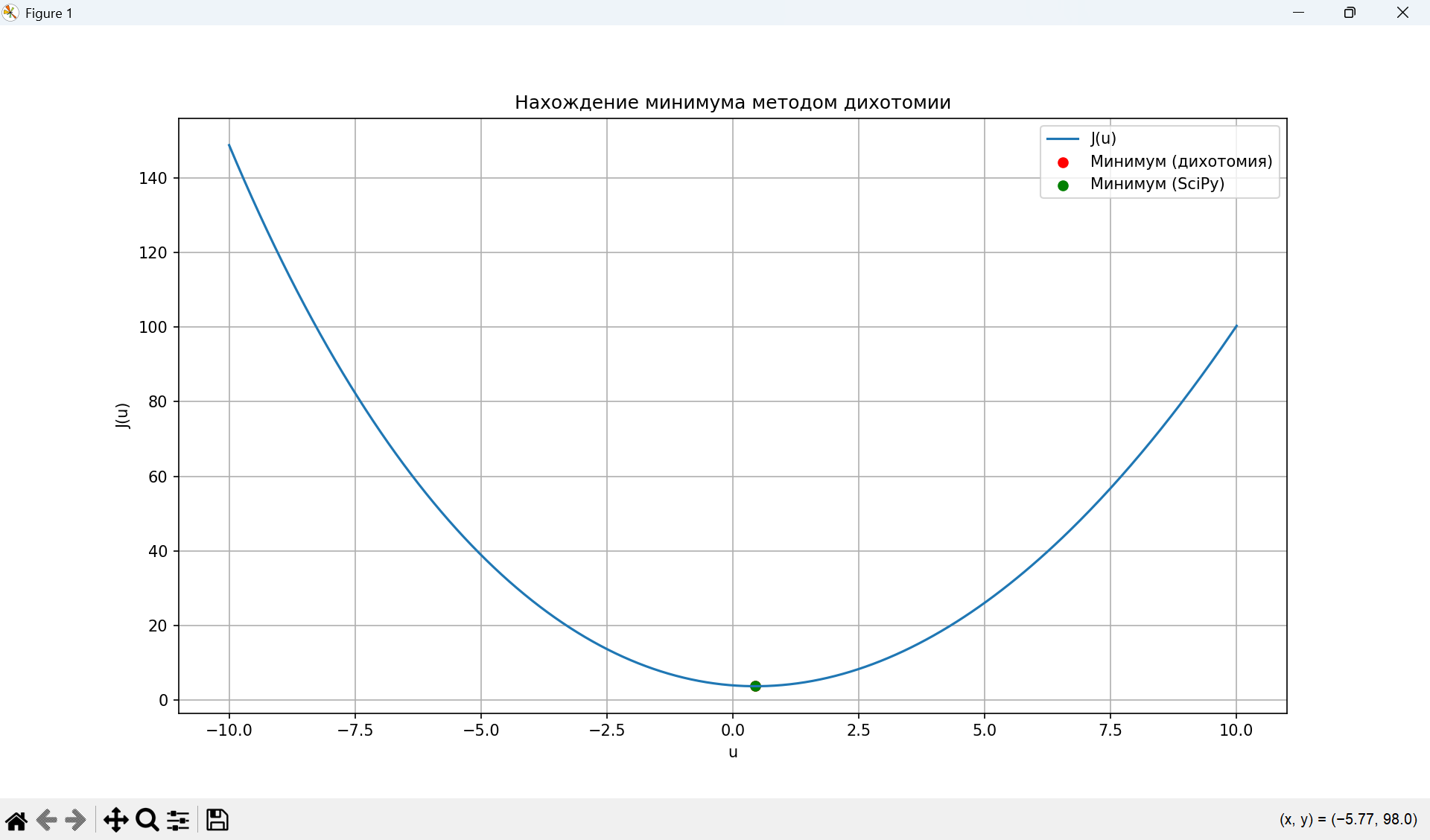


Рисунок Нахождение минимума методом дихотомии

На Рисунках 2 и 3 видно, что минимум на отрезке, найденный по методу дихотомии приблизительно равен минимуму, найденному с помощью встроенной функции Python scipy.optimize.minimize\_scalar.

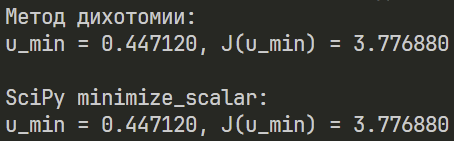


Рисунок Сравнение результата нахождения минимума по методу дихотомии с встроенной функцией поиска минимума

**Формулировка задания №2:**

1. Написать блок-схему алгоритма нахождения минимума функции двух переменных методом градиентного спуска.
2. Допустим, что задана решающая функция линейного классификатора в упрощенном виде (по вариантам).

Найти координаты и значение функции в точке минимума методом градиентного спуска.

1. Сравнить результат с результатом использования встроенной функции (показать график).

Решающая функция:



**Блок-схема алгоритма метода градиентного спуска:**

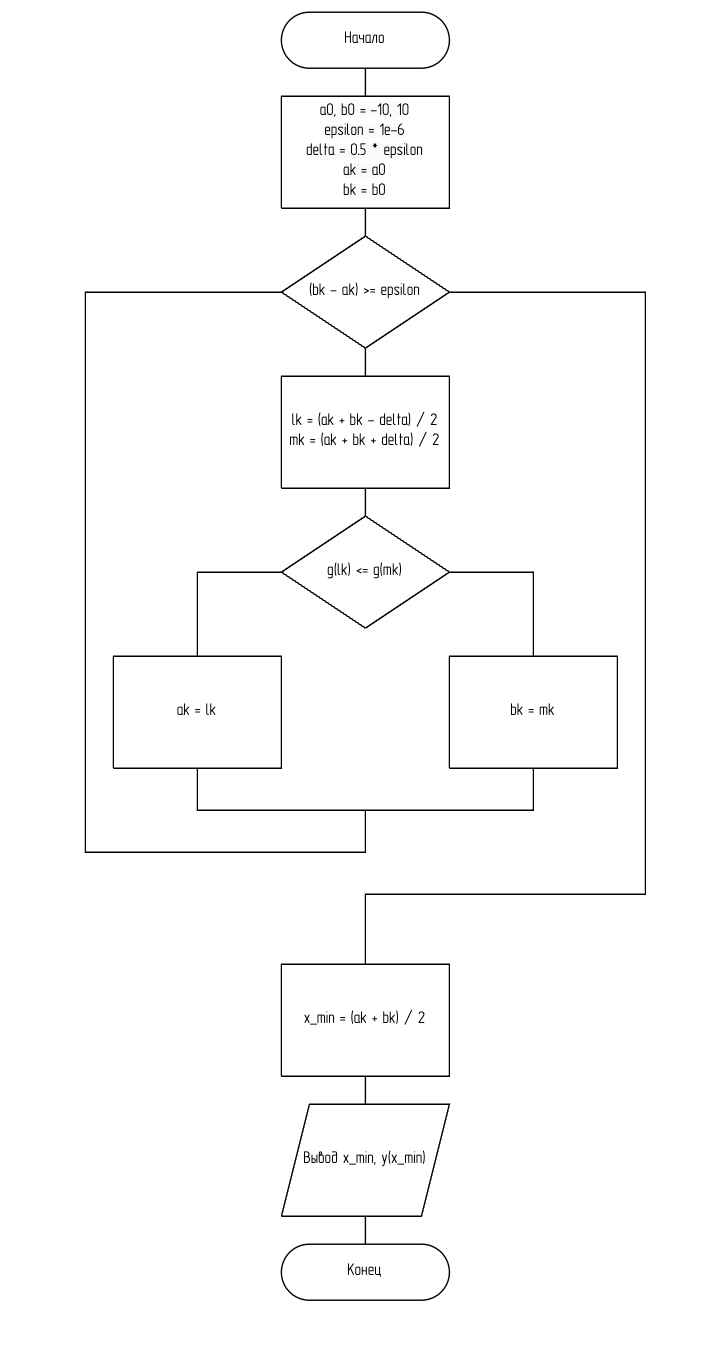


Рисунок 4 Блок-схема алгоритма минимизации функции по методу градиентного спуска\

**Результаты выполнения программы для метода градиентного спуска:**

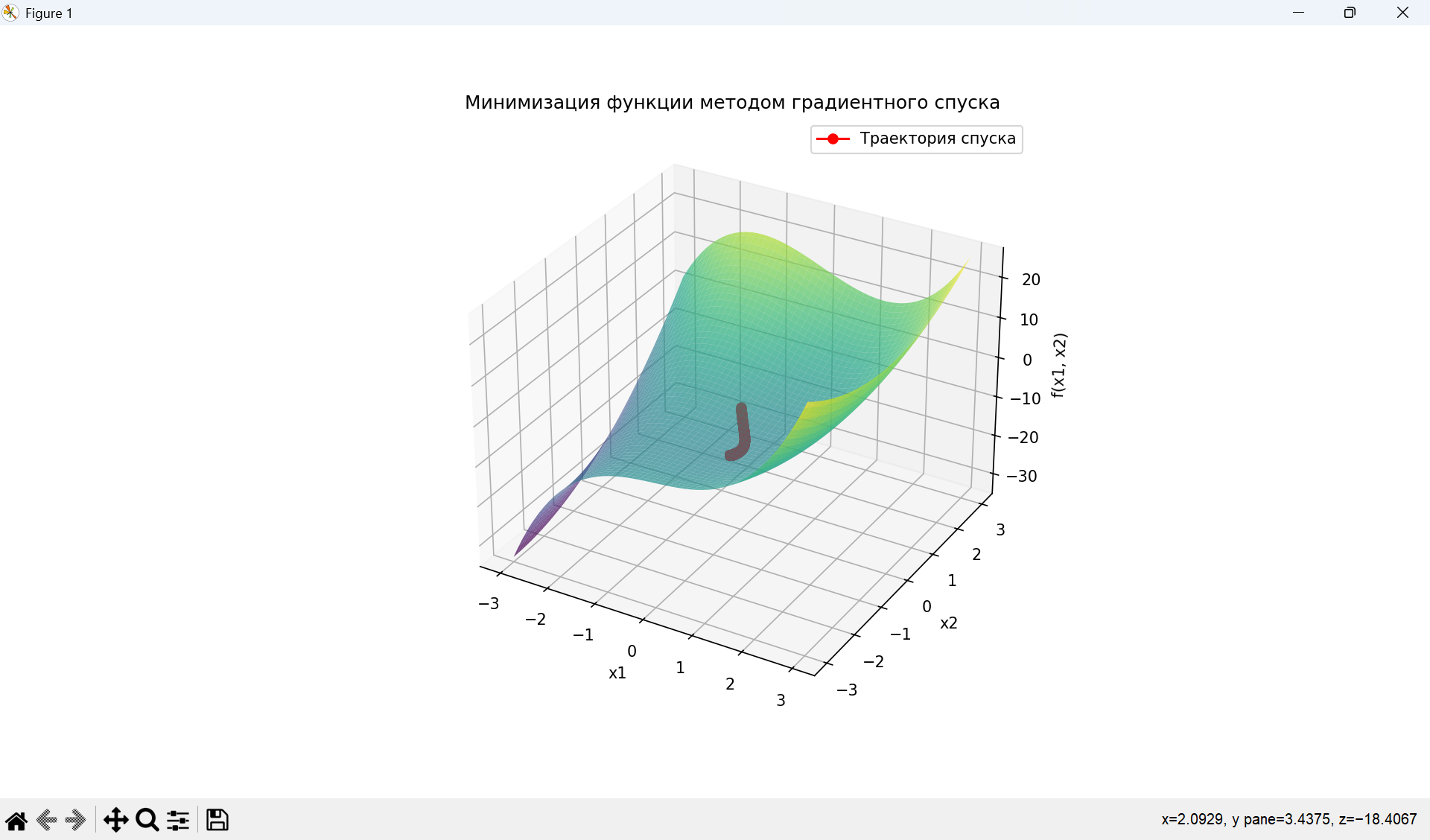


Рисунок 5 Нахождение минимума методом градиентного спуска

На Рисунках 5 и 6 видно, что минимум на отрезке, найденный по методу дихотомии приблизительно равен минимуму, найденному с помощью встроенной функции Python scipy.optimize.minimize\_scalar.

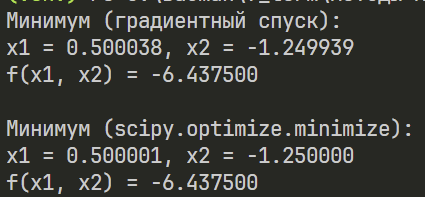


Рисунок 6 Сравнение результата нахождения минимума по методу градиентного спуска с встроенной функцией поиска минимума

**Листинг программы метода дихотомии:**

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import minimize\_scalar

a = 4

b = -0.25

*def* J(*u*):

    return *u*\*\*2 + a \* np.exp(b \* *u*)

*def* dichotomy\_method(*func*, *left*, *right*, *eps*=1e-5, *delta*=1e-4, *max\_iter*=10000):

    iteration = 0

    while abs(*right* - *left*) > *eps* and iteration < *max\_iter*:

        x1 = (*left* + *right* - *delta*) / 2

        x2 = (*left* + *right* + *delta*) / 2

        f1, f2 = *func*(x1), *func*(x2)

        if f1 < f2:

*right* = x2

        else:

*left* = x1

        iteration += 1

    return (*left* + *right*) / 2

u\_min = dichotomy\_method(J, -10, 10)

J\_min = J(u\_min)

res = minimize\_scalar(J, *bounds*=(-10, 10), *method*='bounded')

print("Метод дихотомии:")

print(*f*"u\_min = {u\_min*:.6f*}, J(u\_min) = {J\_min*:.6f*}")

print("\nSciPy minimize\_scalar:")

print(*f*"u\_min = {res.x*:.6f*}, J(u\_min) = {res.fun*:.6f*}")

u = np.linspace(-10, 10, 400)

plt.plot(u, J(u), *label*='J(u)')

plt.scatter(u\_min, J\_min, *color*='red', *label*='Минимум (дихотомия)')

plt.scatter(res.x, res.fun, *color*='green', *label*='Минимум (SciPy)')

plt.title("Нахождение минимума методом дихотомии")

plt.xlabel("u")

plt.ylabel("J(u)")

plt.legend()

plt.grid(True)

plt.show()

**Листинг программы по методу градиентного спуска:**

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

from scipy.optimize import minimize

*def* f(*x*):

    x1, x2 = *x*

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

*def* grad\_f(*x*):

    x1, x2 = *x*

    df\_dx1 = 3\*x1\*\*2 - x2 - 2

    df\_dx2 = 2\*x2 - x1 + 3

    return np.array([df\_dx1, df\_dx2])

*def* gradient\_descent(*start*, *learning\_rate*=0.01, *eps*=1e-6, *max\_iter*=10000):

    x = np.array(*start*, *dtype*=float)

    path = [x.copy()]

    for i in range(*max\_iter*):

        grad = grad\_f(x)

        x\_new = x - *learning\_rate* \* grad

        if np.linalg.norm(x\_new - x) < *eps*:

            break

        x = x\_new

        path.append(x.copy())

    return x, f(x), np.array(path)

start = [0, 0]

learning\_rate = 0.01

x\_min, f\_min, path = gradient\_descent(start, learning\_rate)

print("Минимум (градиентный спуск):")

print(*f*"x1 = {x\_min[0]*:.6f*}, x2 = {x\_min[1]*:.6f*}")

print(*f*"f(x1, x2) = {f\_min*:.6f*}")

res = minimize(f, start)

print("\nМинимум (scipy.optimize.minimize):")

print(*f*"x1 = {res.x[0]*:.6f*}, x2 = {res.x[1]*:.6f*}")

print(*f*"f(x1, x2) = {res.fun*:.6f*}")

x1\_vals = np.linspace(-3, 3, 100)

x2\_vals = np.linspace(-3, 3, 100)

X1, X2 = np.meshgrid(x1\_vals, x2\_vals)

Z = f([X1, X2])

fig = plt.figure(*figsize*=(10, 7))

ax = fig.add\_subplot(111, *projection*='3d')

ax.plot\_surface(X1, X2, Z, *cmap*='viridis', *alpha*=0.7)

ax.plot(path[:,0], path[:,1], [f(p) for p in path], *color*='red', *marker*='o', *label*='Траектория спуска')

ax.set\_title('Минимизация функции методом градиентного спуска')

ax.set\_xlabel('x1')

ax.set\_ylabel('x2')

ax.set\_zlabel('f(x1, x2)')

ax.legend()

plt.show()

**Вывод:** в ходе лабораторной работы был реализован алгоритм поиска минимума функции методом дихотомии (с заданным максимальным количеством итераций) и произведено сравнение с результатом поиска минимума встроенной библиотечной функцией.