Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа №4-5**

**по дисциплине «Методы оптимизации»**

Вариант: **17**

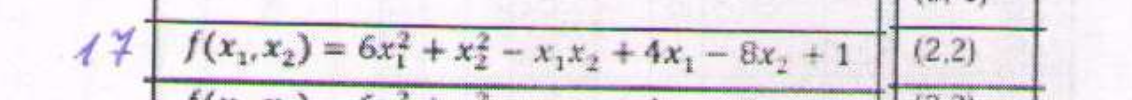
Преподаватель:   
Селина Елена Георгиевна

Выполнила: Шмидт Александра

Группа: Р3215

Санкт-Петербург, 2025 г

Задание:



Решить методом покоординатного спуска, градиентного спуска, наискорейшего спуска, сделать по 3 итерации вручную и написать программу на одном из языков программирования

Изображение выглядит как текст, рукописный текст, Шрифт, письмо

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, рукописный текст, Шрифт

Автоматически созданное описание

Задание 1

Решить графическим методом задачу линейного

программирования.

Задание 2

Решить симплекс-методом задачу линейного

программирования:

𝐹(𝑋) = 𝐶𝑋 → 𝑚𝑖𝑛, 𝐷 = {𝑥𝜖𝑅𝑛: 𝐴𝑋 = 𝑏, 𝑋 ≥ 0}

Решение:  
4 лабораторная

Метод покоординатного спуска

Изображение выглядит как текст, рукописный текст, бумага, каллиграфия

Автоматически созданное описание

Изображение выглядит как текст, рукописный текст, каллиграфия, Шрифт

Автоматически созданное описание

Метод градиентного спуска

Изображение выглядит как текст, рукописный текст, письмо, каллиграфия

Автоматически созданное описание

Изображение выглядит как текст, рукописный текст, каллиграфия, бумага

Автоматически созданное описание

Метод наискорейшего спуска

Изображение выглядит как текст, рукописный текст, письмо, каллиграфия

Автоматически созданное описание

Изображение выглядит как текст, рукописный текст, каллиграфия, документ

Автоматически созданное описание

Листинг программы

#метод покоординатного спуска  
def U(x1, x2):  
 return 6 \* x1\*\*2 + x2\*\*2 - x1 \* x2 + 4 \* x1 - 8 \* x2 + 1  
  
def minimize\_by\_x1(x2):  
 return (x2 - 4) / 12  
  
def minimize\_by\_x2(x1):  
 return (x1 + 8) / 2  
  
def coordinate\_descent(x1\_start, x2\_start, epsilon=0.0001, max\_iter=100):  
 x1, x2 = x1\_start, x2\_start  
 f\_prev = U(x1, x2)  
 print(f"Итерация 0: ({x1:.6f}, {x2:.6f}) → f = {f\_prev:.6f}")  
  
 for i in range(1, max\_iter + 1):  
 x1 = minimize\_by\_x1(x2)  
 x2 = minimize\_by\_x2(x1)  
  
 f\_curr = U(x1, x2)  
 print(f"Итерация {i}: ({x1:.6f}, {x2:.6f}) → f = {f\_curr:.6f}")  
  
 if abs(f\_prev - f\_curr) < epsilon:  
 print("\nДостигнута сходимость по ε.")  
 break  
  
 f\_prev = f\_curr  
  
 return x1, x2, f\_curr  
  
x1\_final, x2\_final, f\_final = coordinate\_descent(2, 2)  
print(f"\nМинимум найден в точке ({x1\_final:.6f}, {x2\_final:.6f}) со значением функции: {f\_final:.6f}")  
  
  
#градиентный метод  
def f(x1, x2):  
 return 6 \* x1\*\*2 + x2\*\*2 - x1 \* x2 + 4 \* x1 - 8 \* x2 + 1  
  
def grad\_f(x1, x2):  
 df\_dx1 = 12 \* x1 - x2 + 4  
 df\_dx2 = 2 \* x2 - x1 - 8  
 return df\_dx1, df\_dx2  
  
def gradient\_descent\_with\_step\_adjustment(x1\_0, x2\_0, lambd=0.25, eps=0.0001, max\_iter=100):  
 for i in range(max\_iter):  
 grad\_x1, grad\_x2 = grad\_f(x1\_0, x2\_0)  
  
 x1\_1 = x1\_0 - lambd \* grad\_x1  
 x2\_1 = x2\_0 - lambd \* grad\_x2  
  
 f0 = f(x1\_0, x2\_0)  
 f1 = f(x1\_1, x2\_1)  
  
 print(f"итерация {i+1}: (x1, x2) = ({x1\_1:.4f}, {x2\_1:.4f}), f = {f1:.4f}")  
  
 # проверка на слишком большой шаг  
 if f0 - f1 < 0:  
 print(f"шаг слишком большой, он был уменьшен с {lambd} до {lambd \* 0.8}")  
 lambd \*= 0.8  
 continue # пробуем снова с уменьшенным шагом  
  
 if abs(f0 - f1) < eps:  
 print(f"сошлось в (x1, x2) = ({x1\_1:.4f}, {x2\_1:.4f}), f = {f1:.4f}")  
 break  
  
 x1\_0, x2\_0 = x1\_1, x2\_1  
  
gradient\_descent\_with\_step\_adjustment(2, 2)  
  
  
#метод наискорейшего спуска  
def f(x1, x2):  
 return 6 \* x1 \*\* 2 + x2 \*\* 2 - x1 \* x2 + 4 \* x1 - 8 \* x2 + 1  
  
def grad\_f(x1, x2):  
 df\_dx1 = 12 \* x1 - x2 + 4  
 df\_dx2 = 2 \* x2 - x1 - 8  
 return np.array([df\_dx1, df\_dx2])  
  
def gradient\_descent\_with\_line\_search(x0, epsilon=1e-4, max\_iter=1000):  
 x\_k = np.array(x0, dtype=float)  
 path = [tuple(x\_k)]  
  
 print(f"{'Iter':<6}{'x1':>10}{'x2':>10}{'f(x1,x2)':>15}{'lambda':>10}")  
 print("-" \* 55)  
  
 for k in range(max\_iter):  
 grad = grad\_f(\*x\_k)  
 norm\_grad = np.linalg.norm(grad)  
  
 if norm\_grad < epsilon:  
 print(f"\nСошлось за {k} итераций.")  
 break  
  
 S\_k = grad / norm\_grad  
 lmbd\_k = 0.25 # начальный шаг  
 f\_curr = f(\*x\_k)  
  
 while True:  
 x\_new = x\_k - lmbd\_k \* S\_k  
 f\_new = f(\*x\_new)  
  
 if f\_new < f\_curr:  
 break  
 else:  
 old\_lambda = lmbd\_k  
 lmbd\_k \*= 0.8  
 print(f"f увеличилась (было {f\_curr:.6f}, стало {f\_new:.6f}) → уменьшаем шаг: {old\_lambda:.6f} → {lmbd\_k:.6f}")  
  
 x\_k = x\_new  
 path.append(tuple(x\_k))  
 print(f"{k:<6}{x\_k[0]:>10.6f}{x\_k[1]:>10.6f}{f\_new:>15.6f}{lmbd\_k:>10.6f}")  
  
 return x\_k, f(\*x\_k), path  
  
# Запуск  
minimum\_point, minimum\_value, path = gradient\_descent\_with\_line\_search((2, 2), epsilon=0.0001)  
  
# Вывод результата  
print(f"\nМинимум функции достигается в точке: ({minimum\_point[0]:.6f}, {minimum\_point[1]:.6f})")  
print(f"Значение функции в этой точке: {minimum\_value: 6f}")

Пример работы программы:  
Итерация 0: (2.000000, 2.000000) → f = 17.000000

Итерация 1: (-0.166667, 3.916667) → f = -14.840278

Итерация 2: (-0.006944, 3.996528) → f = -14.999723

Итерация 3: (-0.000289, 3.999855) → f = -15.000000

Итерация 4: (-0.000012, 3.999994) → f = -15.000000

Достигнута сходимость по ε.

Минимум найден в точке (-0.000012, 3.999994) со значением функции: -15.000000