Министерство науки и высшего образования РФ ФГАОУ ВПО

Национальный исследовательский технологический университет «МИСИС»

Институт Информационных технологий и компьютерных наук (ИТКН)

Кафедра автоматизированных систем управления (АСУ)

Отчет по лабораторной работе №3

по дисциплине «Методы оптимизаций»

Вариант 5

Выполнил:

Студент: Безыкорнов Н.Б.

         Группа: БИВТ-20-4

Проверил:

Лычев А.В.

Москва, 2023

**Цель работы:**

Приобретение практических навыков для решения задач многомерной минимизации различными численными методами первого порядка.

**Постановка задачи:**

Вариант 5

Требуется найти безусловный минимум функции многих переменных y = f(x1, . . . , xn), то есть такую точку x ∗ ∈ R n , что f(x ∗ ) = min x∈Rn f(x).

По условию использовались методы под номерами 1,2. 2

Аналитическим методом был получен минимум F1 в точке [1;1] минимум равен -1.

Выполнение работы:

1. Метод градиентного спуска с постоянным шагом

def grads(x, y):  
 gx = 3 \* x \*\* 2 - 3  
 gy = 2 \* y - 2  
 return gx, gy  
  
  
def f(x, y):  
 return x \*\* 3 + y \*\* 2 - x \* 3 - y \* 2 + 2  
  
  
alpha, eps, x, y, gradX, gradY = 0.1, 0.0001, 0, 0, 0, 0  
iterations, calcs = 0, 0  
  
while True:  
 iterations += 1  
 calcs += 2  
 gradX, gradY = grads(x, y)  
 x -= alpha \* gradX  
 y -= alpha \* gradY  
 if (gradX \*\* 2 + gradY \*\* 2) \*\* (1 / 2) <= eps:  
 break  
  
result = f(x, y)  
  
print("Минимум функции в точке ({}, {}):".format(x, round(y, 5)), round(result, 5))  
print("Итерации: {}. Вычисления функций: {}".format(iterations, calcs))

Листинг 1 – Метод градиентного спуска с постоянным шагом

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 – результат работы программы при eps = 0.001

Изображение выглядит как текст

Автоматически созданное описание

Рис. 3 – результат работы программы при eps = 0.01

1. Метод наискорейшего градиентного спуска

import numpy as np  
  
  
def f(x):  
 global q  
 q += 1  
 return x[0] \*\* 3 + x[1] \*\* 2 - x[0] \* 3 - x[1] \* 2 + 2  
  
  
def grad\_f(x):  
 global k  
 k += 1  
 return np.array([3 \* x[0] \*\* 2 - 3, 2 \* x[1] - 2])  
  
  
def golden\_section\_line\_search(x, grad, a=0, b=1, eps=1e-6):k = 2 - ((1 + 5 \*\* 0.5) / 2)  
 x1 = a + k \* (b - a)  
 x2 = b - k \* (b - a)  
 f1 = f(x - x1 \* grad)  
 f2 = f(x - x2 \* grad)  
 while abs(b - a) > eps:  
 if f1 > f2:  
 a = x1  
 x1 = x2  
 x2 = b - k \* (b - a)  
 f1 = f2  
 f2 = f(x - x2 \* grad)  
 else:  
 b = x2  
 x2 = x1  
 x1 = a + k \* (b - a)  
 f2 = f1  
 f1 = f(x - x1 \* grad)  
 return (a + b) / 2  
  
  
def minimize(x0, eps1, eps2, M):  
 global j  
 """Функция минимизации"""  
 x = x0  
 k = 0  
 while True:  
 j += 1  
 grad = grad\_f(x)  
 if np.linalg.norm(grad) < eps1:  
 return x  
 if k >= M:  
 return x  
 gamma = golden\_section\_line\_search(x, grad)  
 x\_new = x - gamma \* grad  
 if np.linalg.norm(x\_new - x) < eps2 and np.abs(f(x\_new) - f(x)) < eps2:  
 return x\_new  
 k += 1  
 x = x\_new  
  
  
j, q = 0, 0  
k = 0  
  
answer = minimize([0, 0], 0.001, 0.001, 100)  
print("Итерации: {}, Вычисления f: {}, вычисления градиента {}".format(j, q, k))  
print("Минимум функции находится в [{};{}]. Min = {}".format(round(answer[0], 4), round(answer[1], 4),  
 round(f(answer), 4)))

Листинг 2 – Метод наискорейшего градиентного спуска

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4 – результат работы программы с точностью 0.1

Изображение выглядит как текст

Автоматически созданное описание

Рис. 5 – результат работы программы с точностью 0.001

1. Метод Ньютона с регулировкой шага

import numpy as np  
  
  
def f(x, y):  
 global q  
 q += 1  
 return ((x - 5) \*\* 2) \* ((y - 4) \*\* 2) + (x - 5) \*\* 2 + (y - 4) \*\* 2 + 1  
  
  
def grad\_f(x, y):  
 global q  
 q += 1  
 return np.array([2 \* (x - 5) \* ((y - 4) \*\* 2) + 2 \* (x - 5), 2 \* (y - 4) \* ((x - 5) \*\* 2) + 2 \* (y - 4)])  
  
  
def hessian\_f(x, y):  
 global q  
 q += 1  
 return np.array([[2 \* ((y - 4) \*\* 2) + 2 + 2 \* (x - 5) \*\* 2, 4 \* (x - 5) \* (y - 4)],  
 [4 \* (x - 5) \* (y - 4), 2 \* ((x - 5) \*\* 2) + 2 \* (y - 4) \*\* 2]])  
  
  
def newton\_method(x0, eps1, eps2, M, alpha):  
 global j  
 x = x0  
 k = 0  
 while True:  
 j += 1  
 grad = grad\_f(x[0], x[1])  
 if np.linalg.norm(grad) < eps1:  
 return x  
 if k >= M:  
 return x  
 hessian\_inv = np.linalg.inv(hessian\_f(x[0], x[1]))  
 if np.all(np.linalg.eigvals(hessian\_inv) > 0):  
 d = -hessian\_inv.dot(grad)  
 else:  
 d = -grad  
 tk = 1 if np.array\_equal(d, -hessian\_inv.dot(grad)) else 0.5  
 x\_new = x + tk \* d  
 while f(\*x\_new) > f(\*x) + alpha \* np.dot(grad, x\_new - x):  
 tk /= 2  
 x\_new = x + tk \* d  
 if np.linalg.norm(x\_new - x) <= eps2 and abs(f(\*x\_new) - f(\*x)) <= eps2:  
 return x\_new  
 k += 1  
 x = x\_new  
  
  
x0 = np.array([1, 1])  
q, j = 0, 0  
eps1 = 0.0001  
eps2 = 0.0001  
M = 100  
alpha = 0.5  
  
x\_min = newton\_method(x0, eps1, eps2, M, alpha)  
print("Кол-во итераций:", j, "Кол-во вычислений функции:", q)  
print("Минимум в точке ({}, {}), f(min): {}".format(x\_min[0], x\_min[1], f(x\_min[0], x\_min[1])))

Листинг 3 – Метод Ньютона с регулировкой шага

Изображение выглядит как текст

Автоматически созданное описание

Рис. 6 – результат работы программы c начальной точкой 0 0

Изображение выглядит как текст

Автоматически созданное описание

Рис. 7 – результат работы программы c начальной точкой 0 1

Изображение выглядит как текст

Автоматически созданное описание

Рис. 8 – результат работы программы c начальной точкой 1 1

Вывод:

В данной лабораторной работе были изучены градиентные методы многомерной минимизации и методы второго порядка. Были написаны программы поиска минимума функции различными методами, найдены координаты и значение функции в точке минимума. Была проведена сходимость алгоритмов и проанализированы полученные результаты.