Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 3

Метод Хука-Дживса
Тема

Преподаватель			В. В. Тынченко
		Подпись, дата	Инициалы, Фамилия
Студент	КИ19-17/1Б, №031939174		А. К. Никитин
	Номер группы, зачетной книжки	Подпись, дата	Инициалы, Фамилия

Красноярск 2021

1 Постановка задачи

Разработать программу, реализующую метод Хука-Дживса.

Найти безусловный экстремум функции, выбранной в соответствии с заданием, с использованием разработанной программы.

Функция: $(x_2 + x_1 - 1)^2 + 2(x_1 - 2)^2 \rightarrow min$

2 Описание метода

Метод Хука-Дживса был разработан в 1961 году, но до сих пор является весьма эффективным и оригинальным. Поиск состоит из последовательности шагов исследующего поиска вокруг базисной точки, за которой в случае успеха следует поиск по образцу. Он применяется для решения задачи минимизирования функции без учета ограничений.

Описание этой процедуры представлено ниже.

Шаг 1. Выбрать начальную базисную точку b_1 и шаг длиной h_1 для каждой переменной x_j, j = 1, 2,…, n. В приведенной ниже программе для каждой переменной используется шаг h, однако указанная выше модификация тоже может оказаться полезной.

Шаг 2. Вычислить f (x) в базисной точке b_1 с целью получения сведений о локальном поведении функции f (x). Эти сведения будут использоваться для нахождения подходящего направления поиска по образцу, с помощью которого можно надеяться достичь большего убывания значения функции.

Подшаг 1. Вычисляется значение функции f (b_1) в базисной точке b_1.

Подшаг 2. Каждая переменная по очереди изменяется прибавлением длины шага. Таким образом, мы вычисляем значение функции f (b1 +h1 e1). Если это приводит к уменьшению значения функции, то b1 заменяется на b1 +h1 e1. В противном случае вычисляется значение функции f (b1 - h1e1), и если ее значение уменьшилось, то b1 заменяем на b1 - h1e1 . Если ни один из проделанных шагов не приводит к уменьшению значения функции, то точка b1 остается неизменной и рассматриваются изменения в направлении оси x2, т.

е. находится значение функции f (b1 +h2 e2) и т. д. Когда будут рассмотрены все n переменные, мы будем иметь новую базисную точку b2.

Подшаг 3. Если b2 = b1 , т. е. уменьшение функции не было достигнуто, то исследование повторяется вокруг той же базисной точки , но с уменьшенной длиной шага. На практике удовлетворительным является уменьшение шага (шагов) в десять раз от начальной длины.

Подшаг 4. Если b2 ≠ b1, то производится поиск по образцу.

Шаг 3. В. При поиске по образцу используется информация, полученная в процессе исследования, и минимизация функции завершается поиском в направлении, заданном образцом.

Подшаг 1. Разумно двигаться из базисной точки b_2 в направлении $b_2 - b_1$, поскольку поиск в этом направлении уже привел к уменьшению значения функции. Поэтому вычислим функцию в точке образца: $P_1 = b_1 + 2(b_2 - b_1)$. В общем случае: $P_i = b_i + 2(b_{i+1} - b_i)$.

Подшаг 2. Затем исследование следует продолжать вокруг точки P_i.

Подшаг 3. Если наименьшее значение на шаге В, 2 меньше значения в базисной точке b_{i+1}, то получают новую базисную точку b_{i+2}, после чего следует повторить шаг 1. В противном случае не производить поиск по образцу из точки b_{i+1}, а продолжить исследования в точке b_{i+1}.

Шаг 4. Завершить этот процесс, когда длина шага (длины шагов) будет уменьшена до заданного малого значения.

3 Исходные тексты программ

На листинге 1 представлен код программы, реализующий задание.

Листинг 1 – Метод Хука-Дживса

```
from const import f, f_arguments


def calc_func(func, arguments, values):
    arguments_dict = {argument: value for argument, value in zip(arguments,
values)}
    return func.subs(arguments_dict)


def hooke_jeeves_method(func, arguments, x0, steps=None, epsilon=0.099,
lambda_parameter=1.5, decreasing_coef=1.1):
    if steps is None:
        steps = [0.5] * len(x0)
    xk = x0

    y = xk.copy()
    iter_num = 0
    while all(map(lambda step: step > epsilon, steps)):
        iter_num += 1
        for i in range(len(arguments)):
            if calc_func(func, arguments, y[:i] + [y[i] + steps[i]] + y[i+1:]) <
\
                    calc_func(func, arguments, y):
                y[i] += steps[i]
            elif calc_func(func, arguments, y[:i] + [y[i] - steps[i]] + y[i+1:])
< \
                    calc_func(func, arguments, y):
                y[i] -= steps[i]

        if func.subs({'x1': y[0], 'x2': y[1]}) < func.subs({'x1': xk[0], 'x2':
xk[1]}):
            gap = [a-b for a, b in zip(y, xk)]
            increasing = list(map(lambda x: lambda_parameter * x, gap))

            xk = y.copy()
            y = [a+b for a, b in zip(y, increasing)]
            continue
```

```
        else:
            steps = list(map(lambda step: step / decreasing_coef, steps))
            y = xk.copy()

    return xk, iter_num


def main():
    print(hooke_jeeves_method(f, f_arguments, [0, 0], epsilon=0.3,
lambda_parameter=1, decreasing_coef=2, steps=[0.1, 0.1]))


if __name__ == '__main__':
    main()
```

4 Исследование влияния параметров метода на точность и скорость нахождения решения

Из рисунка 1 можно заключить, что параметр величины шага слабо влияет на точность решения, однако имеется прямая линейная зависимость между количеством шагов алгоритма и параметра величины шага.

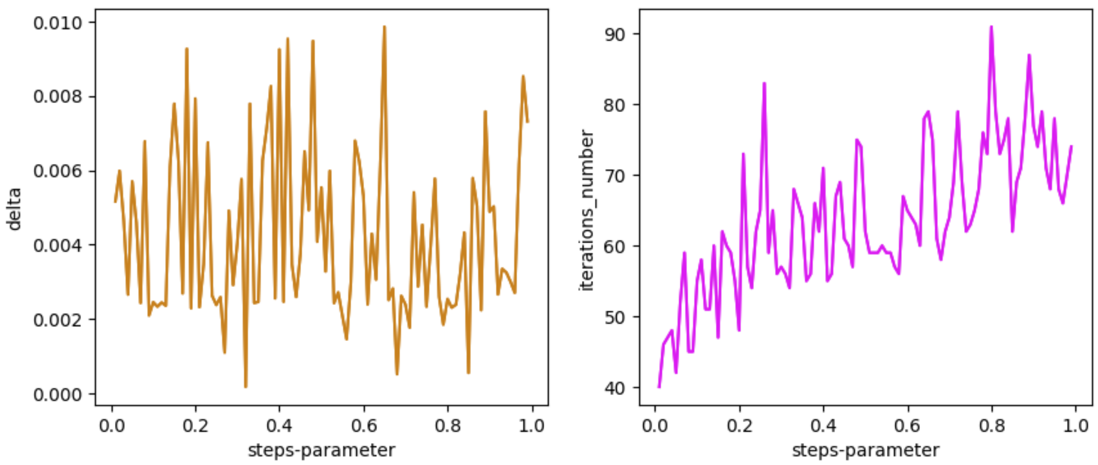

Рисунок 1 – Влияние величины шага на точность и производительность

Из рисунка 2 можно заключить, что чем меньше параметр эпсилон, тем точнее решение, однако тем больше шагов необходимо алгоритму для нахождения ответа.

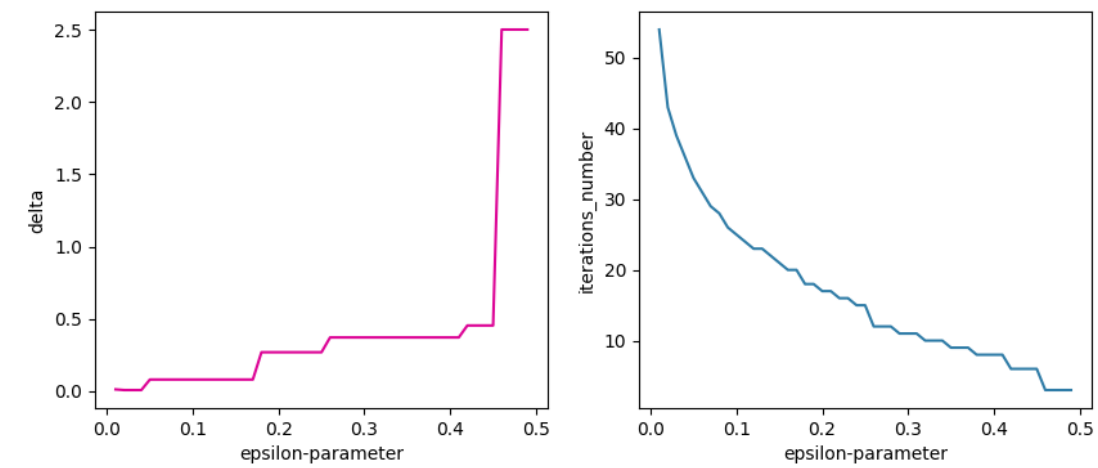

Рисунок 2 - Влияние параметра ε на точность и производительность

Из рисунка 3 можно заключить, что ускоряющий множитель не имеет явной зависимости с точностью решения алгоритма, однако способен значительно ускорить его.

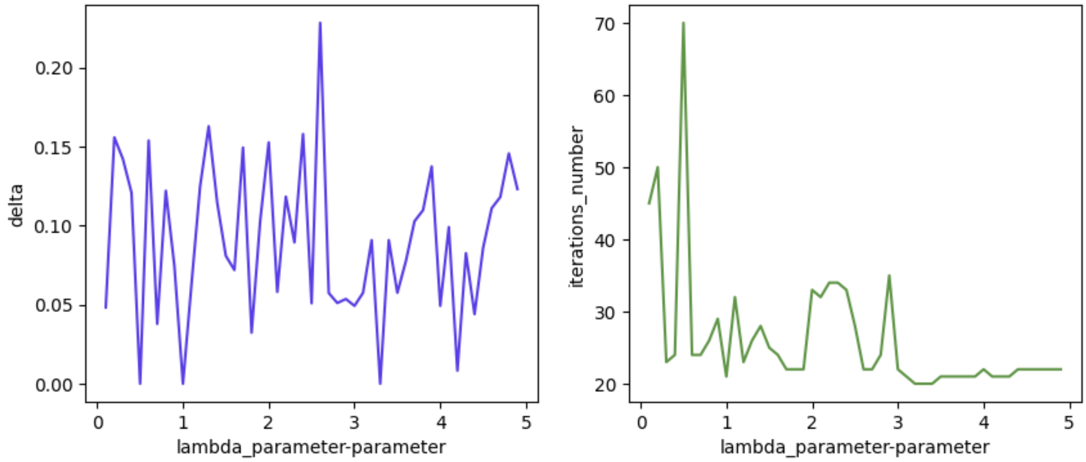

Рисунок 3 - Влияние ускоряющего множителя на точность и производительность

5 Вывод

В результате данной работы был реализован и проанализирован метод Хука-Дживса для поиска локального минимума многомерной функции. Также были проанализированы гиперпараметры метода и их влияние на работу функции.