Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Программная инженерия |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Пауэлла (сопряженных направлений) |
| тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | В. В. Тынченко |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-17/1Б, 032156940 |  |  |  | Н. А. Самарин |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Задание............................................................................................................... 3

2 Вариант.............................................................................................................. 3

3 Описание метода............................................................................................... 3

4 Реализация метода............................................................................................ 3

5 Анализ результатов.......................................................................................... 4

6 Вывод................................................................................................................. 7

**1 Задание**

Разработать программу, реализующую метод сопряженных направлений.  
Найти безусловный экстремум функции, выбранной в соответствии с заданием,  
с использованием разработанной программы.

**2 Вариант**

f(x) = (x1 - 1)^2 + (x2 + x1)^2

**3 Описание метода**

В методе сопряженных направлений (методе Пауэлла ) используется  
факт, что минимум квадратичной функции может быть найден не более чем за  
n шагов при условии, что поиск ведется вдоль сопряженных относительно  
матрицы Гессе направлений. Так как достаточно большой класс целевых  
функций может быть предоставлен в окрестности точки минимума своей  
квадратичной аппроксимацией, описанная идея применяется и для  
неквадратичных функций. Задается начальная точка и направления,  
совпадающие с координатами. Находится минимум f(x) при последовательном  
движении по (n + 1) направления с помощью одного из методов одномерной  
минимизации. При этом полученная ранее точка минимума берется в качестве  
исходной для поиска по совершенно нового направлению, а направление  
используется как при первом, так и последнем поиске. Находится новое  
направление поиска, сопряженное с. Оно проходит через точки, полученные  
при последнем поиске. Заменяется на, на и т.д. Направление заменяется  
сопряженным направлением, после чего повторяется поиск по (n+1)  
направлениям, уже не содержащим старого направления.

**4 Реализация метода**

Ниже в листинге 1 представлен текст python программы, реализующей  
метод для поиска минимума функции.

Листинг 1 – Текст python программы

def powell(func, x0, e, fib\_eps=0.0001):  
 n = len(x0)  
 d\_base = {}  
 for i in range(1, n + 1):  
 lst = [0] \* n  
 lst[i - 1] = 1  
 d\_base[i] = np.array(lst)  
 d = {}  
 y = {}  
 t = {}  
 d\_base[0] = d\_base[n]  
 i = 0  
 k = 0  
 x = {0: x0}  
 y[0] = x[0]  
 while True:  
 args = y[i]

Окончание листинга 1

if d\_base[i][0]:  
 t[i] = fibonacci\_method(lambda t: func((args[0] + t, args[1])), eps=  
 fib\_eps)  
 else:  
 t[i] = fibonacci\_method(lambda t: func((args[0], args[1] + t)), eps=  
 fib\_eps)  
 y[i + 1] = y[i] + t[i] \* d\_base[i]  
 if i < (n - 1):  
 i += 1  
 continue  
 elif i == (n - 1):  
 if (y[n][0] == y[0][0]) and (y[n][1] == y[0][1]):  
 return y[n]  
 i += 1  
 continue  
 else:  
 if (y[n + 1][0] == y[0][0]) and (y[n + 1][1] == y[0][1]):  
 return y[n + 1]  
 x[k + 1] = y[n + 1]  
 if ((x[k + 1][1] - x[k][1]) \*\* 2 + (x[k + 1][0] - x[k][0]) \*\* 2) \*\* 0.5   
 < e:  
 return x[k + 1]  
 else:  
 d[n] = y[n + 1] - y[1]  
 d[0] = d[n]  
 d[1] = d\_base[2]  
 if np.linalg.matrix\_rank(np.array([d[0], d[1]])) == n:  
 for i in range(n + 1):  
 d[i] = d\_base[i]  
 i = 0  
 y[0] = x[k + 1]  
 k = k + 1  
 continue  
 else:  
 for i in range(n + 1):  
 d\_base[i] = d[i]  
 y[0] = x[k + 1]  
 k += 1  
 i = 0  
 continue

**5 Анализ результатов**

Для начала найдём реальный минимум функции, для этого произведём  
следующие действия:

Найдём частные производные 1-го порядка:

Найдём частные производные 1-го порядка:

z'y = 2x + 2y

z'y = 2x + 2y

M0: x = 1; y = -1

Найдём частные производные 2-го порядка в точке M0 и проверим  
достаточное условие экстремума:

A = z''xx(M0) = 4

B = z''xy(M0) = 2

C = z''yy(M0) = 2

AC - B^2 > 0

4\*2 - 2^2 > 0

4>0

При этом A>0, следовательно точка (1;-1) - минимум функции.

Теперь получим значения работы метода сопряженных направлений.  
Также построим графики зависимости количества вычислений целевой  
функции и отклонения от реального минимума в зависимости от изменения  
параметров метода. Результаты представлены на рисунках ниже.

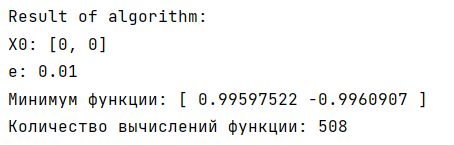


Рисунок 1 – Результат работы метода

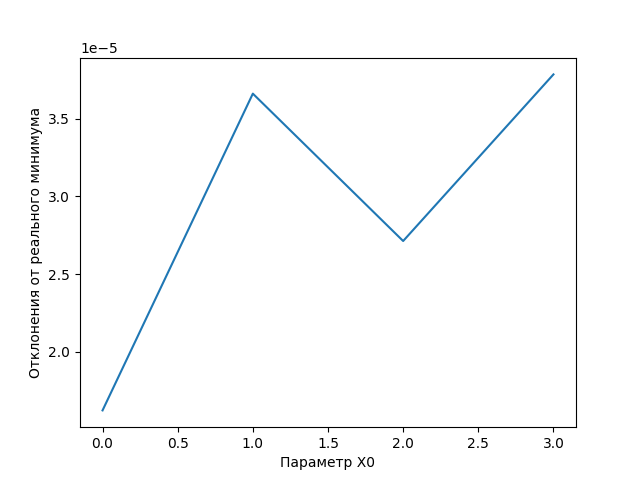


Рисунок 2 – График отклонения от реального минимума для X0

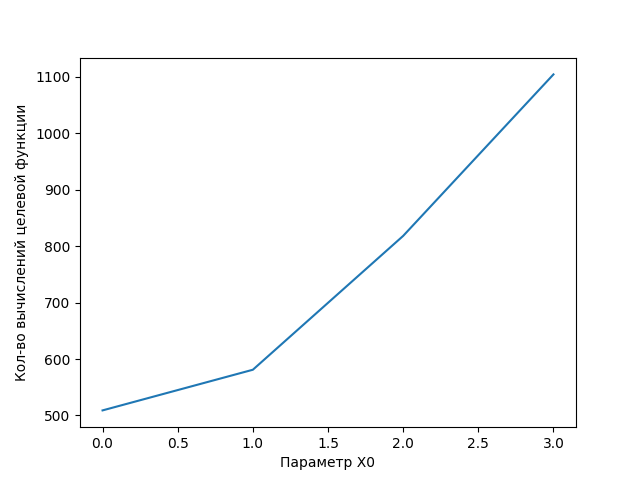


Рисунок 3 – График количества вычислений целевой функции для X0

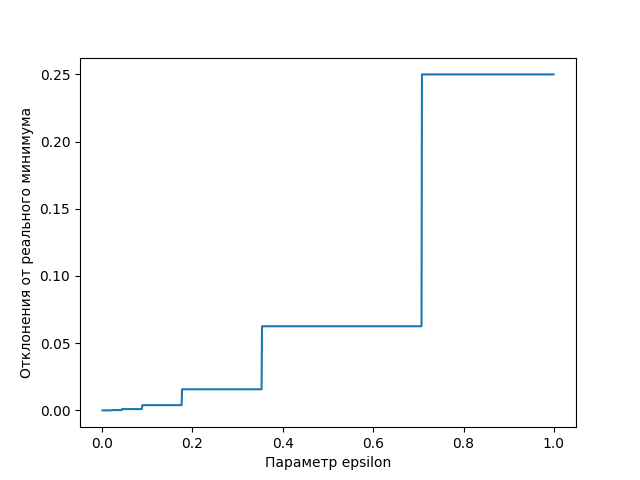


Рисунок 4 – График отклонения от реального минимума для epsilon

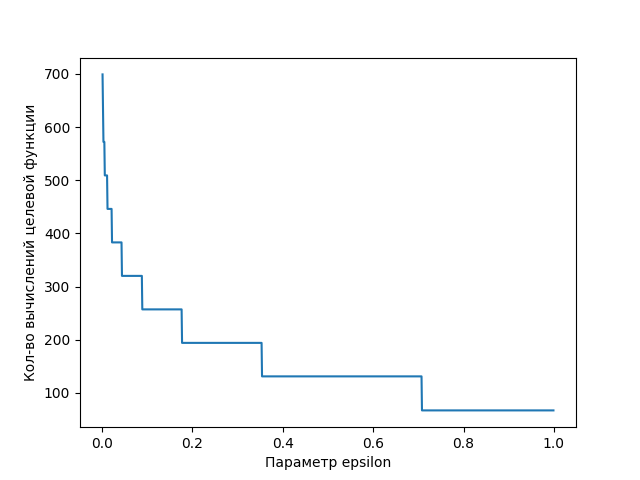


Рисунок 5 – График количества вычислений целевой функции для epsilon

В результате можно сказать, что найденный минимум функции  
соответствует реальному. При этом можно определить следующее влияние  
параметров на результат: чем больше эпсилон тем больше отклонение, но  
меньше кол-во вычислений. Чем больше x0 отличается от реального минимума  
тем больше вычислений функции.

**6 Вывод**

При выполнении задания был успешно реализован метод сопряженных  
направлений, результаты работы метода сравнены с реальным и близки к нему,  
исследована зависимость работы метода от значений его параметров.