Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

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

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ**

|  |
| --- |
| Метод Хука-Дживса |
| 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 задания................................................................................................ 3

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

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

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

6 Вывод................................................................................................................. 9

**1 Задание**

Разработать программу, реализующую метод Хука-Дживса. Найти  
безусловный экстремум функции, выбранной в соответствии с заданием, с  
использованием разработанной программы.

**2 Вариант задания**

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

**3 Описание метода**

Метод конфигураций, или метод Хука–Дживса (Hooke–Jeeves),  
представляет собой комбинацию исследующего поиска с циклическим  
изменением переменных и ускоряющего поиска по образцу. Исследующий  
поиск ориентирован на выявление локального поведения целевой функции и  
определение направления ее убывания вдоль "оврагов". Полученная  
информация используется при поиске по образцу при движении вдоль  
"оврагов". Исследующий поиск начинается в некоторой начальной точке ,  
называемой старым базисом. В качестве множества направлений поиска  
выбирается множество координатных направлений. Задается величина шага,  
которая может быть различной для разных координатных направлений и  
переменной в процессе поиска. Фиксируется первое координатное направление  
и делается шаг в сторону увеличения соответствующей переменной. Если  
значение функции в пробной точке меньше значения функции в исходной  
точке, шаг считается удачным. В противном случае необходимо вернуться в  
предыдущую точку и сделать шаг в противоположном направлении с  
последующей проверкой поведения функции. После перебора всех координат  
исследующий поиск завершается. Полученная точка называется новым  
базисом. Если исследующий поиск с данной величиной шага неудачен, то она  
уменьшается и процедура продолжается. Поиск заканчивается, когда текущая  
величина шага станет меньше некоторой величины. Поиск по образцу  
заключается в движении по направлению от старого базиса к новому. Величина  
ускоряющего шага задается ускоряющим множителем lambda. Успех поиска по  
образцу определяется с помощью исследующего поиска из полученной точки.  
Если при этом значение в наилучшей точке меньше, чем в точке предыдущего  
базиса, то поиск по образцу удачен. Если поиск по образцу неудачен,  
происходит возврат в новый базис, где продолжается исследующий поиск с  
уменьшенным шагом.

**4 Реализация метода**

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

Листинг 1 – Текст python программы

def hook\_jeeves(f, x0, eps=0.0000001, d=0.3, h=1, alpha=2):

Продолжение листинга 1

n = len(x0)  
 delta = d \* np.eye(n)  
 calc\_num = 0  
 k = 0  
 x = [x0]  
 y = [None, x0]  
 while True:  
 for i in range(1, n + 1):  
 fp = f(y[i] + delta[i - 1])  
 calc\_num+=2  
 if fp < f(y[i]):  
 if i + 1 >= len(y):  
 y.append(None)  
 y[i + 1] = y[i] + delta[i - 1]  
 else:  
 fm = f(y[i] - delta[i - 1])  
 calc\_num+=2  
 if fm < f(y[i]):  
 if i + 1 >= len(y):  
 y.append(None)  
 y[i + 1] = y[i] - delta[i - 1]  
 else:  
 if i + 1 >= len(y):  
 y.append(None)  
 y[i + 1] = y[i]  
 calc\_num += 2  
 if f(y[n+1]) < f(x[k]):  
 if k + 1 >= len(x):  
 x.append(None)  
 x[k + 1] = y[n + 1]  
 y[1] = x[k + 1] + h \* (x[k + 1] - x[k])  
 k = k + 1  
 else:  
 check = True  
 for i in range(0, n):  
 if delta[i][i] > eps:  
 delta[i][i] = delta[i][i] / alpha  
 check = False  
 if check:  
 break  
 else:  
 y[1] = x[k]  
 if k + 1 >= len(x):  
 x.append(None)  
 x[k + 1] = x[k]  
 k = k + 1  
 return x[k], calc\_num  
def func\_0(x):  
 # 1 -1  
 return (x[0] - 1)\*\*2 + (x[1] + x[0])\*\*2  
x0 = [0,0]  
eps=0.0000001  
d=0.3  
h=1  
alpha=2  
func = func\_0  
real\_x = [1, -1]  
minimum, calc\_num = hook\_jeeves(func, x0, eps, d, h, alpha)

Окончание листинга 1

print("Параметры метода: x0 -", x0, ", epsilon -", eps, ", delta -", d, ", lambd  
a -", h, ", alpha -", alpha)  
print("Минимум функции:", minimum)  
print("Количество вычислений функции:", calc\_num)

**5 Анализ результатов**

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

Найдём частные производные 1-го порядка:

z'x = 4x + 2y - 2

z'y = 2x + 2y

Приравняем их к нулю и решим систему:

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 Пауэлла. Также построим  
графики зависимости количества вычислений целевой функции и отклонения  
от реального минимума в зависимости от изменения параметров метода.  
Результаты представлены на рисунках ниже.

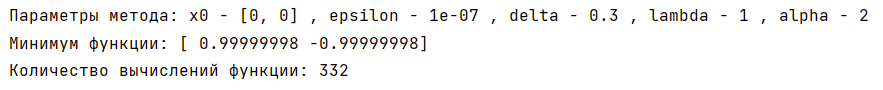


Рисунок 1 – Результат работы метода №1

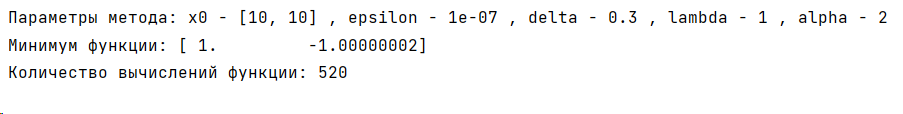


Рисунок 2 – Результат работы метода №2

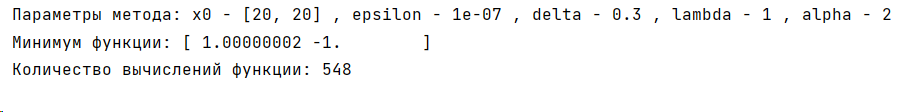


Рисунок 3 – Результат работы метода №3

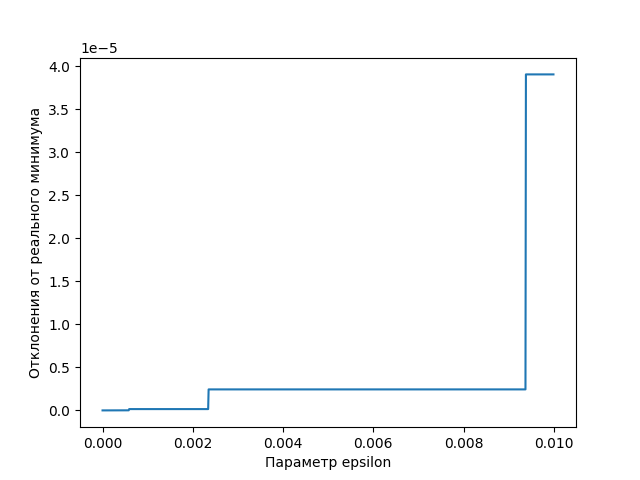


Рисунок 4 – График отклонения от реального минимума для epsilon

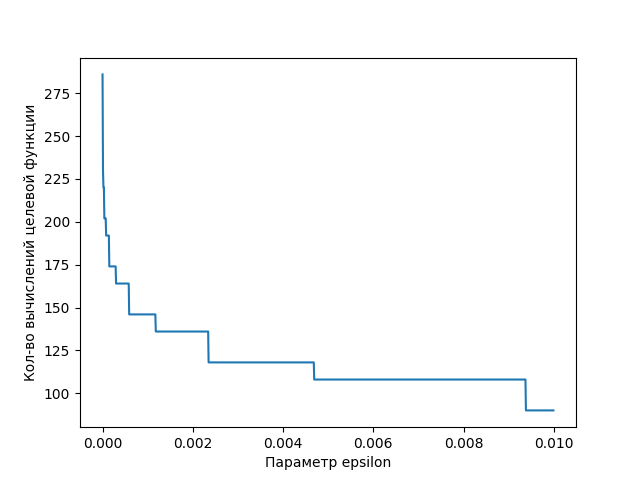


Рисунок 5 – График количества вычислений целевой функции для epsilon

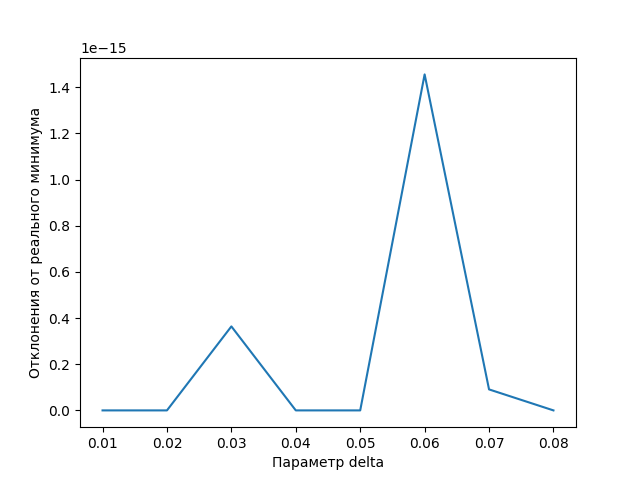


Рисунок 6 – График отклонения от реального минимума для delta

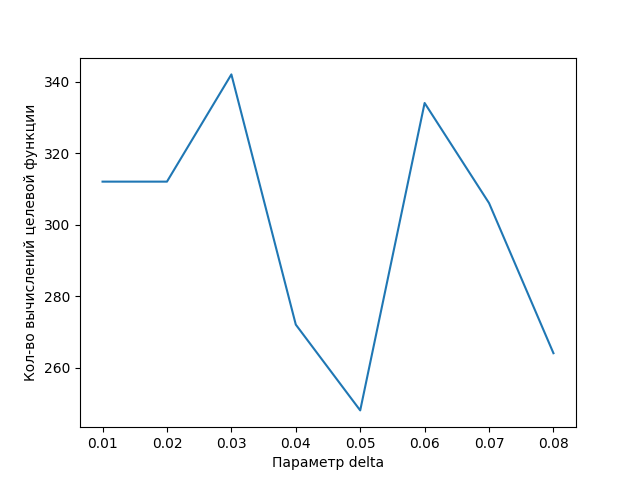


Рисунок 7 – График количества вычислений целевой функции для delta

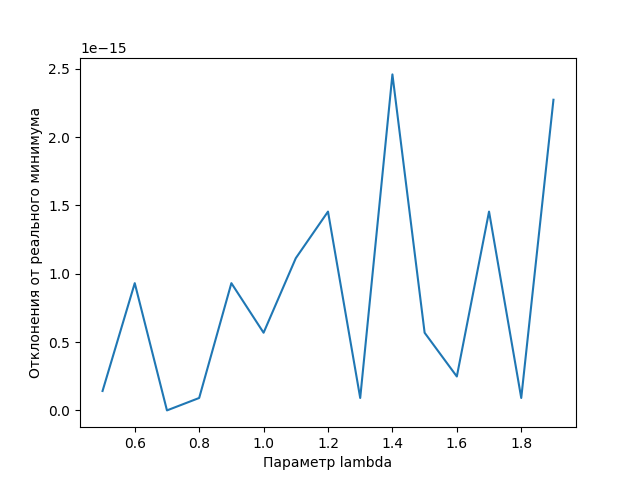


Рисунок 8 – График отклонения от реального минимума для lambda

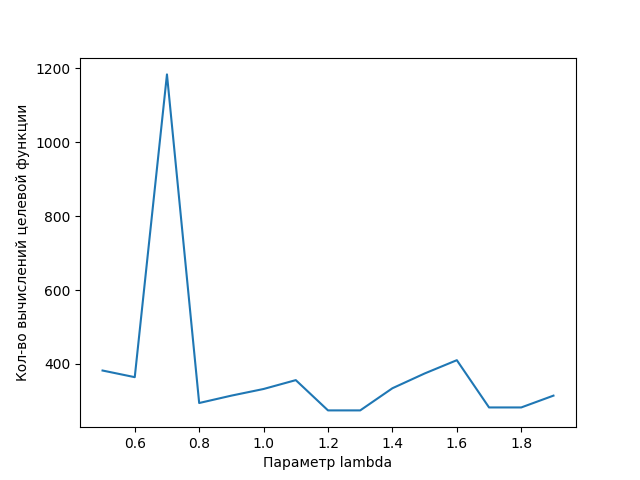


Рисунок 9 – График количества вычислений целевой функции для lambda

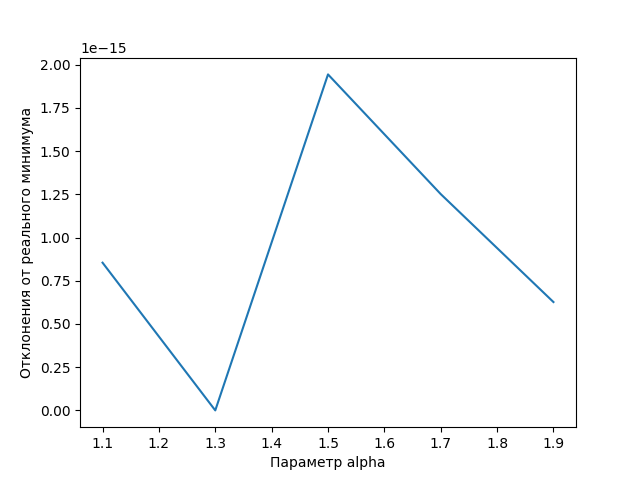


Рисунок 10 – График отклонения от реального минимума для alpha

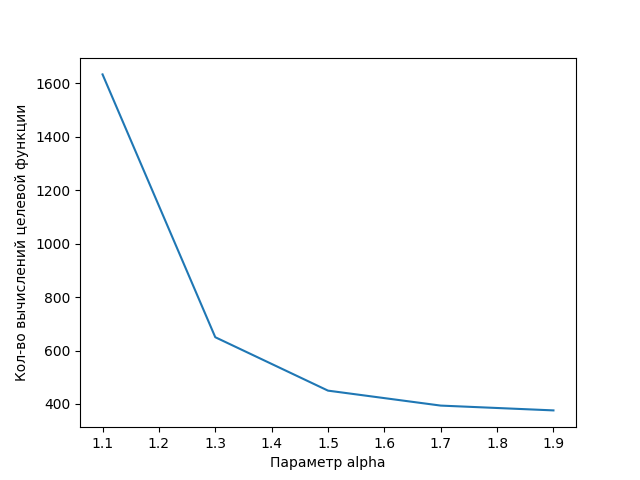


Рисунок 11 – График количества вычислений целевой функции для alpha

В результате можно сказать, что найденный минимум функции близок к  
реальному. При этом наблюдается следующее влияние параметров на  
результат: чем больше x0 отличается от реального минимума, тем больше  
вычислений функции, чем больше эпсилон тем больше отклонение, но меньше  
кол-во вычислений, у параметров delta, lambda и alpha минимальное влияние на  
точность, при этом чем больше alpha тем меньше вычислений функции, у  
параметров delta и lambda сложно определить закономерность их влияния на  
кол-во вычислений, но оно присутствует.

**6 Вывод**

При выполнении задания был успешно реализован метод Хука-Дживса,

результаты работы метода сравнены с реальным и близки к нему, исследована  
зависимость работы метода от значений его параметров.