Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**ОПТИМИЗАЦИЯ ФУНКЦИИ МНОГИХ ПЕРЕМЕННЫХ**

**Отчёт по лабораторной работе №3**

**По дисциплине**

**«Методы оптимизации»**

Студент гр. 430-2:

А.А. Лузинсан

« » 2022 г.

Проверил:

к.т.н, доцент каф. АСУ   
(должность уч.степень, уч.звание)

А.А. Шелестов

« » 2022 г.

Томск 2022

Оглавление

[ВВЕДЕНИЕ 3](#_Toc121083572)

[1 ТЕОРИЯ 4](#_Toc121083573)

[1.1 Метод Хука-Дживса 4](#_Toc121083574)

[1.2 Симплексный метод 4](#_Toc121083575)

[2 АЛГОРИТМЫ МЕТОДОВ 5](#_Toc121083576)

[2.1 Метод Хука-Дживса 5](#_Toc121083577)

[2.1 Симплексный метод 6](#_Toc121083578)

[ЗАКЛЮЧЕНИЕ 8](#_Toc121083579)

[ЛИСТИНГ ПРОГРАММЫ 9](#_Toc121083580)

# **ВВЕДЕНИЕ**

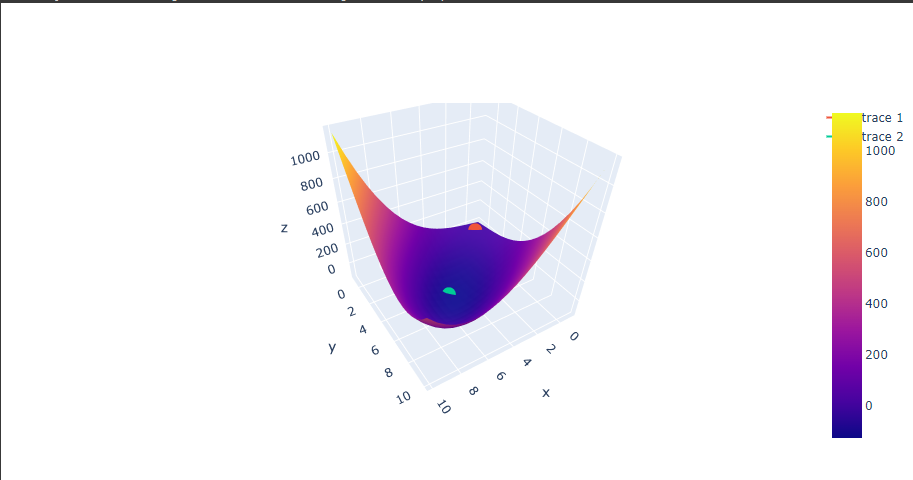
Задание: найти минимум функции двух переменных, используя два прямых метода: симплексный метод и метод Хука-Дживса.

Точность: ε = .

Вариант задания:

2)

Вид исходной функции представлен на рисунке 1.1.

Рисунок 1.1 – Изображение исходной функции

# **1** **ТЕОРИЯ**

# **1.1 Метод Хука-Дживса**

Суть метода: нахождение в окрестности текущей точки наилучшей и движение в этом направлении. Если значение в окрестных точках больше, чем в текущей, то происходит уменьшение шага.

Процедура Хука–Дживса представляет собой комбинацию двух поисков:

а) "Исследующий" поиск: с заданным шагом Δi происходит расчет функции в пробных точках вокруг некоторой исходной точки x0 (*f*(x0 ± Δj )). Если значение ЦФ в пробной точке меньше значения ЦФ в исходной точке, то шаг поиска успешный. В противном случае из исходной точки делается шаг в противоположном направлении. После перебора всех n координат исследующий поиск завершается. Полученная точка называется базовой.

б) Ускоряющий поиск по образцу: осуществляется шаг из полученной базовой точки вдоль прямой, соединяющей эту точку с предыдущей базовой. Новая точка образца определяется по формуле:

xpk+1 = xk + (xk – xk-1).

# **1.2 Симплексный метод**

Суть метода: приближение к минимальной точке с помощью изменения координат вершин симплекса. Подробнее о методе описано в алгоритме ниже.

# **2** **АЛГОРИТМЫ МЕТОДОВ**

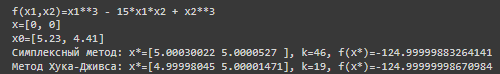
В результате применения двух методов над заданной функцией были получены результате, представленные на рисунке 2.1.

Рисунок 2.1 – Результат применения прямых методов нахождения минимума функции двух переменных

## 2.1 Метод Хука-Дживса

Введем следующие обозначения:

* xk – текущая базовая точка;
* xk-1 – предыдущая базовая точка;
* xpk+1 – точка, построенная при движении по образцу;
* xk+1 – следующая (новая) базовая точка.

Критерий останова: ||Δx|| ≤ ε.

Алгоритм:

1. Определить начальную точку x0; приращения (шаги) ΔI, i= 1, n; коэффициент уменьшения шага α > 1; параметр окончания поиска ε.
2. Провести исследующий поиск.
3. Был ли исследующий поиск удачным (найдена ли точка с меньшим значением ЦФ)? Да: переход на пункт 5.
4. Проверка на окончание поиска. Выполняется ли неравенство || Δx|| ≤ ε.? Да: окончание поиска, т.е. текущая точка аппроксимирует точку экстремума x\*. Нет: уменьшить приращение Δi/α; i=1,2,…,n. Переход на пункт 2.
5. Провести поиск по образцу: xpk+1 = xk + (xk – xk-1).
6. Провести исследующий поиск, используя точку xpk+1 в качестве временной базовой точки. Пусть в результате получена точка xk+1
7. Выполняется ли неравенство: f(xk+1) = f(xk)?. Да: положить xk-1 = xk; xk = xk+1. Переход на пункт 5. Нет: переход на пункт 4.

## 2.1 Симплексный метод

Алгоритм метода:

1. Задается исходная вершина симплекса. x0= (x1, …, x0n)  
   Задается коэффициент сжатия γ∈[0,1] и размер симплекса L. Строится симплекс:  
   Здесь j-я строка – это координаты j-ой вершины Vj. (j = 1, ..., n+ 1), где n - размерность пространства (размерность вектора x ), i – номер координаты i = 1,...,n.  
   Определение координат , начиная со второй, производится по формуле: , (j=1, …, n; I = 1, …, n), где - матрица размерности (n + 1) \* n:

,  
где .

Векторы соответствующие вершинам V1, …, Vn, составят одинаковые углы с координатными осями x1,..., xn.

1. В вершинах симплекса вычисляется ЦФ f(xj), j = 0,…,n.
2. Проверяем условия: ||xj – xj-1|| ≤ ε1, |f(xj) – f(xj-1)| ≤ ε2. Если «да», то конец; если «нет», то переходим в пункт 4.
3. Находится «наихудшая» вершина симплекса (при поиске минимума «наихудшая» вершина – та, в которой значение функции максимально).

f(xp) =

1. Осуществляется расчет координат новой вершины (вершина отражения xp):

6. Если точка оказывается «хуже» всех остальных точек симплекса, то осуществляется возврат к исходному симплексу с последующим его сжатием относительно «лучшей» из вершин xk. Переход на пункт 2. Если не является «худшей» в новом симплексе, то перейти на пункт 3.

f(xk) =

# ЗАКЛЮЧЕНИЕ

Я изучила методы нахождения минимумов функций двух переменных и нашла минимум функции двух переменных заданного варианта, используя два прямых метода: симплексный метод и метод Хука-Дживса.

# ЛИСТИНГ ПРОГРАММЫ

import plotly.express as px

import plotly.graph\_objects as go

import sympy

import numpy as np

def Simplex(f, x, eps):

k = 1

gamma = np.random.rand() #коэффициент сжатия. От 0 до 1

n = len(x) #Размер измерения (двумерная)

L = n+1 #Размер симплекса

p = (L/(n\*np.sqrt(2)))\*(np.sqrt(n+1)+n-1)

q = (L/(n\*np.sqrt(2)))\*(np.sqrt(n+1)-1)

\_x\_ = np.zeros((L,n),dtype = x.dtype)

\_x\_[1:,:] = q

for i in range(n):

\_x\_[i+1,i] = p

simpl = np.zeros((L,n),dtype=np.float64)

simpl[0] = x

simpl = simpl[0] + \_x\_

res = [0]\*L

while True:

for i in range(L):

res[i] = f(\*simpl[i])

for i in range(n):

if abs(res[i+1]-res[i])<=eps and np.sqrt(((simpl[i+1]-simpl[i])\*\*2).sum())<=eps:

return (simpl[i]+simpl[i+1])/2, k, (res[i+1]+res[i])/2

maxarg = np.argmax(res)

ref = (simpl.sum(axis=0) - simpl[maxarg])\*2/n - simpl[maxarg]

if f(\*ref) > res[maxarg]:

minarg = np.argmin(res)

simpl = simpl[minarg] \* gamma + (1-gamma) \* simpl

else:

simpl[maxarg] = ref

k += 1

return simpl, k, res

def Hooke(f, x, eps):

k = 1

alpha = 1+np.random.rand()\*10

n = len(x)

delta = x.copy()

delta[:] = 10

best = x.copy()

foundbetter = False

while True:

foundbetter = False

for i in range(n):

newbest = best.copy()

newbest[i] += delta[i]

if f(\*best) > f(\*newbest):

best = newbest

foundbetter = True

continue

newbest = best.copy()

newbest[i] -= delta[i]

if f(\*best) > f(\*newbest):

best = newbest

foundbetter = True

if not(foundbetter):

if any(delta < eps):

return best, k, f(\*best)

else:

delta /= alpha

k += 1

while True:

newbest = best + (best - x)

if f(\*newbest)<f(\*best):

best = newbest

else:

break

return best, k, f(\*best)

def lab3():

x1 = sympy.Symbol('x1')

x2 = sympy.Symbol('x2')

f = x1\*\*3+x2\*\*3-15\*x1\*x2

print('f(x1,x2)={}'.format(f))

\_x = [0,0]

\_x0 = [5.23,4.41]

print('x={}\nx0={}'.format(\_x,\_x0))

eps = 1e-4

foo = sympy.lambdify([x1,x2],f,'numpy')

print("Симплексный метод: x\*={}, k={}, f(x\*)={}".format(\*Simplex(foo, np.array(\_x0,dtype=np.float64), eps)))

print("Метод Хука-Дживса: x\*={}, k={}, f(x\*)={}".format(\*Hooke(foo, np.array(\_x0,dtype=np.float64), eps)))

x = np.linspace(-1,10,20)

y = np.linspace(-1,10,20)

X, Y = np.meshgrid(x, y)

Z = foo(X, Y)

fig = go.Figure(data=[go.Surface(x=X,y=Y,z=Z)])

fig.add\_trace(go.Scatter3d(x=[\_x[0]],y=[\_x[1]],z=[foo(\*\_x)]))

fig.add\_trace(go.Scatter3d(x=[\_x0[0]],y=[\_x0[1]],z=[foo(\*\_x0)]))

fig.show()

lab3()