МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Д.В. Шинтяков

Отчёт

по лабораторной работе №3

по дисциплине Численные методы и вариационное исчисление

на тему: «МЕТОД СЖИМАЮЩИХСЯ МНОГОГРАННИКОВ»

Работу выполнил

студент гр. 4941 Н.С. Горбунов

Санкт-Петербург

2021

**Цель работы**: освоить методику численной оптимизации методом сжимающихся многогранников (Нелдера-Мида).

**Вариант 7**

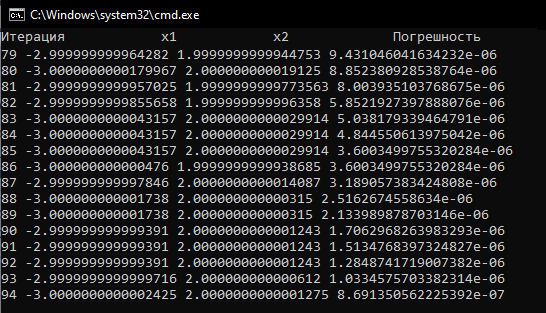
|  |  |  |  |
| --- | --- | --- | --- |
| № | Уравнение | Значения свободных параметров | |
| A | B |
| 7 |  | -3 | 2 |

**Аналитическое решение**

**Текст программы**

import numpy as np  
import math  
def max\_D(x0, x1, x2):  
 return max(math.sqrt(abs(x0[0] - x1[0]) + abs(x0[1] - x1[1])), math.sqrt(abs(x0[0] - x2[0]) + abs(x0[1] - x2[1])), math.sqrt(abs(x1[0] - x2[0]) + abs(x1[1] - x2[1])))  
  
def cycle(f0, f1, f2, x0, x1, x2):  
 tops = [(x0,f0),(x1,f1),(x2,f2)]  
 tops.sort(key = lambda xf:xf[1])  
 (x1,f1),(xg,fg),(xh,fh) = tops  
 return fh, fg, f1, xh, xg, x1  
  
def func(x):  
 return 2 \* (x[0] +3)\*\*2 + (x[1]-2)\*\*2  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 x0 = np.zeros(2, dtype=float)  
 x0[0] = 0  
 x0[1] = 0  
 x1 = np.zeros(2, dtype=float)  
 x1[0] = 1  
 x1[1] = 0  
 x2 = np.zeros(2, dtype=float)  
 x2[0] = 0  
 x2[1] = 1  
 eps = 1e-6 # Точность  
 f0 = func(x0)  
 f1 = func(x1)  
 f2 = func(x2)  
 fh, fg, f1, xh, xg, x1 = cycle(f0, f1, f2, x0, x1, x2)  
 i = 0  
 print(f'Итерация x1 x2 Погрешность')  
 while max\_D(x1, xg, xh) > eps:  
 fh, fg, f1, xh, xg, x1 = cycle(f1, fg, fh, x1, xg, xh)  
 i += 1  
 xc = 0.5 \* (x1 + xg)  
 xr = xc - (xh - xc)  
 fr = func(xr)  
 if fr < f1:  
 xe = xc + 2 \* (xr - xc)  
 fe = func(xe)  
 if fe < f1:  
 xh = xe  
 fh = func(xh)  
 else:  
 xh = xr  
 fh = func(xh)  
 elif f1 <= fr < fg:  
 xh = xr  
 fh = func(xh)  
 elif fg <= fr < fh:  
 xr, xh = xh, xr  
 fr, fh = fh, fr  
 xs = xc + 0.5 \* (xh - xc)  
 fs = func(xs)  
 if fs < fh:  
 xh = xs  
 fh = func(xh)  
 else:  
 xg = xg + (xg - x1)/2  
 fg = func(xg)  
 xh = xh + (xh - x1)/2  
 fh = func(xh)  
 elif fr >= fh:  
 xs = xc + 0.5 \* (xh - xc)  
 fs = func(xs)  
 if fs < fh:  
 xh = xs  
 fh = func(xh)  
 else:  
 xg = xg + (xg - x1)/2  
 fg = func(xg)  
 xh = xh + (xh - x1)/2  
 fh = func(xh)  
 print(f'{i} {x1[0]} {x1[1]} {max\_D(x1, xg, xh)}')

**Результат работы программы**



**Вывод:** освоил методику численной оптимизации методом сжимающихся многогранников; написал реализацию алгоритма на языке Python.