Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Методы оптимизации (МОптим)

ОТЧЕТ

по лабораторной работе №4

Тема работы: Нелинейная оптимизация

Выполнил

студент: гр. 051006 Артихович Н.С.

Проверила: Петюкевич Н.С.

Минск 2022

**Вариант 1**

**Задание 2**

Метод Гаусса-Зейделя. Определить оптимальные партии поставок X1..n, такие, чтобы расходы были минимальны.

Начальные значения:

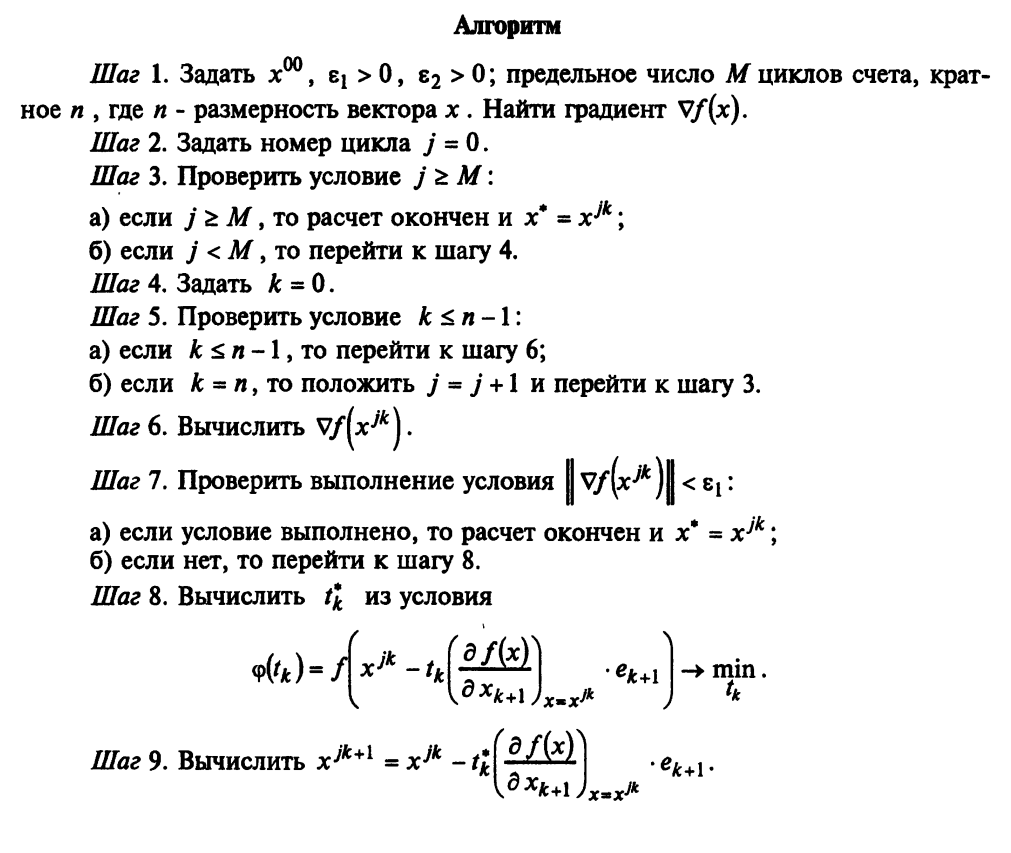
e = 0.0001

|  |  |  |  |
| --- | --- | --- | --- |
| Vi | Ki | Si | fi |
| 400 | 10 | 16 | 4 |
| 600 | 12 | 8 | 3 |
| 800 | 11 | 8 | 5 |
| 700 | 9 | 7 | 4 |
| 200 | 8 | 4 | 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I | Vi | Ki | Si | fi |
| 1 | 900 | 10 | 5 | 16 |
| 2 | 700 | 5 | 15 | 4 |
| 3 | 300 | 20 | 10 | 15 |
| 4 | 1000 | 30 | 2 | 22 |
| 5 | 200 | 6 | 3 | 10 |

X1..n= 0

Функция L = 900\*10/X1 + 0.5 \* 5 \* X1 + 700\*5/X2 + 0.5 \* 15 \* X2 + 300\*20/X3 + 0.5 \* 10 \* X3 + 1000\*30/X4 + 0.5 \* 2 \* X4 + 200\*6/X5 + 0.5 \* 3 \* X5





**Код программы**

import numpy as np  
from typing import Callable, List  
from scipy.optimize import minimize  
  
  
def f(x):  
 X1, X2, X3, X4, X5 = x  
 return 900 \* 10 / X1 \  
 + 0.5 \* 5 \* X1 \  
 + 700 \* 5 / X2 \  
 + 0.5 \* 15 \* X2 \  
 + 300 \* 20 / X3 \  
 + 0.5 \* 10 \* X3 \  
 + 1000 \* 30 / X4 \  
 + 0.5 \* 2 \* X4 \  
 + 200 \* 6 / X5 \  
 + 0.5 \* 3 \* X5  
  
  
def coordinate\_descent(func: Callable[..., float],  
 x0: List[float],  
 odm: Callable[[Callable[[float], float], float, float], float],  
 eps: float = 0.0001,  
 step\_crushing\_ratio: float = 0.99):  
 k = 0  
 N = len(x0)  
 h = np.array([1.0] \* N)  
 x\_points = [x0]  
  
 while h[0] > eps:  
 x\_points.append([0] \* N)  
 for i in range(N):  
 args = x\_points[k].copy()  
  
 def odm\_func(x):  
 nonlocal i, func, args  
 args[i] = x  
 return func(\*args)  
  
 ak = odm(odm\_func, args[i], h[i])  
  
 x\_points[k + 1][i] = ak  
  
 if np.linalg.norm(np.array(x\_points[k + 1]) - np.array(x\_points[k])) <= eps:  
 break  
  
 k += 1  
 h \*= step\_crushing\_ratio  
  
 return x\_points[len(x\_points) - 1]  
  
  
def odm(fnc, x0, h):  
 res = minimize(fnc, x0, method='nelder-mead', options={'xatol': h, 'disp': False})  
 return res.x[0]  
  
  
res = coordinate\_descent(lambda \*args: f(args), [0., 0., 0., 0., 0.], odm)  
for id, i in enumerate(res):  
 print(f"q{id}={i}")  
print("f=" + str(f(coordinate\_descent(lambda \*args: f(args), [0., 0., 0., 0., 0.], odm))))

Результат работы (решение)

