|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ | |
| Федеральное государственное автономное образовательное учреждение высшего образования | |
| **«Дальневосточный федеральный университет»** (ДВФУ) | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | |
| **Департамент математического и компьютерного моделирования** | |
| **ЛАБОРАТОРНАЯ РАБОТА №2** | |
| По основной образовательной программе подготовки бакалавров  направлению 02.03.01 Математика и компьютерные науки  профиль «Сквозные цифровые технологии» | |
|  | Студент группы  Б9120-02.03.01сцт - Пограничный Кирилл  (подпись)  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  | Преподаватель: Яковлев Анатолий Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |
| г. Владивосток  2023 | |
|  | |

**Постановка задачи:**

Найти минимум функции :



с условием .

**Решение:**

Для начала определим матрицу А и вектор b. А также сгенерируем начальное приближение.

Воспользуемся библиотекой numpy.

Данные генерируются случайным образом, а потом записываются в текстовые файлы.

Рассмотрим решение на примере сгенерированной матрицы А:

Вектора b:

И x0:

Далее нужно найти функцию Лагранжа, взять ее частную производную по x и приравнять ее к нулю.

Функция Лагранжа будет иметь вид:



Частная производная по x:



Есть два случая

1) y = 0, тогда:

, тогда , где  – «подозрительная» на минимум точка.

В нашем случае:

x\* =

Функция в точке x\* = 42.260250926105755

||x\*-x\_0|| = 2396.375962306893

В нашем случае r = 5 => проверка не выполняется.

Найденная точка не будет рассматриваться при выборе итогового ответа.

2) y > 0, тогда:

Преобразуем  и получим следующую систему уравнений из пяти уравнений:

.

Для нахождения точек, подозрительных на оптимум, воспользуемся методом Ньютона:



Метод Ньютона будем запускать на нескольких начальных приближениях, т.к. функция может иметь несколько оптимальных точек. За начальное приближение берётся восемь точек. В нашем случае:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i | Начальное приближение |  |  |  |
| 1 | [-2.1992554]  [1.9682616]  [1.3134242]  [1.6923226] | [-0.85216837]  [-1.0101238]  [-0.47545267]  [-0.73485533] | 8.670976631311639 | 0.7507624695659647 |
| 2 | [5.8007446]  [1.9682616]  [1.3134242]  [1.6923226] | [-0.85216829]  [-1.01012385]  [-0.47545268]  [-0.73485535] | 1.553410752569705 | 0.7507624793875429 |
| 3 | [1.8007446]  [-2.0317384]  [1.3134242]  [1.6923226] | [-0.85216846]  [-1.0101237]  [-0.47545269]  [-0.73485534] | 9.134503201382676 | 0.7507625143343724 |
| 4 | [1.8007446]  [5.9682616]  [1.3134242]  [1.6923226] | [-0.8521684]  [-1.0101238]  [-0.47545268]  [-0.73485529] | 1.5422279886457129 | 0.7507624526092984 |
| 5 | [1.8007446]  [1.9682616]  [-2.6865758]  [1.6923226] | [-0.85216847]  [-1.0101237]  [-0.47545259]  [-0.73485541] | 6.654997554074009 | 0.7507624793709873 |
| 6 | [1.8007446]  [1.9682616]  [5.3134242]  [1.6923226] | [-0.85216842]  [-1.01012368]  [-0.47545277]  [-0.73485536] | 2.6002792936986285 | 0.7507625972805201 |
| 7 | [1.8007446]  [1.9682616]  [1.3134242]  [-2.3076774] | [-0.85216844]  [-1.01012379]  [-0.47545267]  [-0.73485527] | 8.296242656048745 | 0.7507624323576554 |
| 8 | [1.8007446]  [1.9682616]  [1.3134242]  [5.6923226] | [-0.85216846]  [-1.0101238]  [-0.47545268]  [-0.73485523] | 2.003297269959546 | 0.7507624101831167 |

Все значения y > 0, поэтому находим функцию, в которой достигается минимальное значение. В нашем случае это точка:

x =

fmin = 0.7507624526092984

**Листинг программы:**

import numpy as np  
  
  
class SecondLab:  
  
 A = 0  
 b = 0  
 r = 0  
 x\_0 = 0  
 a = 0  
 y = 0  
 sign = 0  
  
 def \_\_init\_\_(self, r, a, y, sign):  
 np.random.seed(1)  
 self.A = np.loadtxt("a.txt", usecols=(range(4)))  
 self.b = np.loadtxt("b.txt", usecols=(range(1)), ndmin=2)  
 self.x\_0 = np.loadtxt("x\_0.txt", usecols=(range(1)), ndmin=2)  
 self.r = r  
 self.a = a  
 self.y = y  
 self.sign = sign  
 self.generate\_matrix()  
 self.generate\_vector("b")  
 self.generate\_vector("x\_0")  
  
 def f(self, x: np.ndarray) -> float:  
 res = .5 \* x.transpose() @ self.A @ x + self.b.transpose() @ x  
 return res[0][0]  
  
 def generate\_matrix(self):  
 matrix = np.random.uniform(0.4, 0.7, (4, 4))  
 np.savetxt("a.txt", matrix @ matrix, fmt='%.7f')  
  
 def generate\_vector(self, name: str):  
 matrix = np.random.uniform(1, 2, (4, 1))  
 np.savetxt(f"{name}.txt", matrix, fmt='%.7f')  
  
 def lagrange\_slae(self, x: np.ndarray) -> np.ndarray:  
 return np.append((self.A + 2 \* np.eye(4) \* self.y) @ x + (self.b + 2 \* self.y \* self.x\_0),  
 [[np.linalg.norm(x - self.x\_0)\*\*2 - self.r\*\*2]], axis=0)  
  
 def jacobian(self, x: np.ndarray) -> np.ndarray:  
 J\_1\_1 = self.A + 2 \* np.eye(4) \* self.y  
 J\_1\_2 = 2 \* (x - self.x\_0)  
 J\_2\_1 = J\_1\_2.transpose()  
 J\_2\_2 = [[0]]  
 J\_1 = np.append(J\_1\_1, J\_1\_2, axis=1)  
 J\_2 = np.append(J\_2\_1, J\_2\_2, axis=1)  
 return np.append(J\_1, J\_2, axis=0)  
  
 def newton(self, x\_k: np.ndarray, epsilon=1e-6, max\_iter=30, ):  
 x\_prev = x\_k  
 x\_cur = x\_prev - np.linalg.inv(self.jacobian(x\_prev[0:-1])) @ self.lagrange\_slae(x\_prev[0:-1])  
 it = 0  
 while np.linalg.norm(x\_cur[0:-1] - x\_prev[0:-1]) > epsilon and it < max\_iter:  
 it += 1  
 x\_prev = x\_cur  
 x\_cur = x\_prev - np.linalg.inv(self.jacobian(x\_prev[0:-1])) @ self.lagrange\_slae(x\_prev[0:-1])  
 return x\_cur  
  
 def start\_lab(self):  
 x\_ = np.append(self.x\_0, [[self.y]], axis=0)  
  
 print('')  
 x\_star = -np.linalg.inv(self.A) @ self.b  
 f\_in\_x\_star = self.f(x\_star)  
 print(f"x\*:\n{x\_star}")  
 print(f"\nФункция в точке x\* = {f\_in\_x\_star}")  
 print(f"\nx\*-x\_0:\n{x\_star - self.x\_0}")  
 print(f"\n||x\*-x\_0|| = {np.linalg.norm(x\_star - self.x\_0)}\n")  
  
 for i in range(8):  
 self.sign = -self.sign  
 x\_k = x\_.copy()  
 x\_k[i // 2][0] += self.sign \* self.a  
 print(f"\nНачальное приближение {i + 1}:\n{x\_k[0:-1]}")  
 res = self.newton(x\_k)  
 print("Значение x:")  
 print(res[0:-1])  
 print(f"Значение y = {res[4][0]}")  
 print(f"Значение функции = {self.f(res[0:-1])}\n")  
  
  
def execute\_second\_lab():  
 s\_l = SecondLab(5, 4, 3, 1)  
 s\_l.start\_lab()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 execute\_second\_lab()