|  |  |
| --- | --- |
| 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 №1** | |
| 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 Олегович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г. |
|  | Преподаватель  Яковлев Анатолий Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность, ученое звание)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) (ФИО)  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |
| г. Владивосток  2023 | |

**Постановка задачи:**

Минимизировать функцию , где ,

*A =* ([[125.5, 79.75, 132, 125, 102.75, 122.75],  
 [79.75, 72.5, 100.25, 76.5, 67.5, 62],  
 [132, 100.25, 199.5, 153, 130.5, 168.5],  
 [125, 76.5, 153, 164, 124, 152],  
 [102.75, 67.5, 130.5, 124, 173.25, 125.25],  
 [122.75, 62, 168.5, 152, 125.25, 189.25]])

Проверка матрицы , что она является положительно определённой:

Все собственные значения матрицы положительные, значит матрица положительно определенная.

**Метод градиента:**

*,* где

Первая производная функции: .

Приравнивая производную к нулю, получаем вектор .

Все значения округлены до тысячных.

Алгоритм отработал за 107962 шагов. Условие выхода из цикла:

*Промежуточные результаты:*

*Промежуточные значения функционала:*

Значение функционала в точке

*Погрешности метода градиента:*

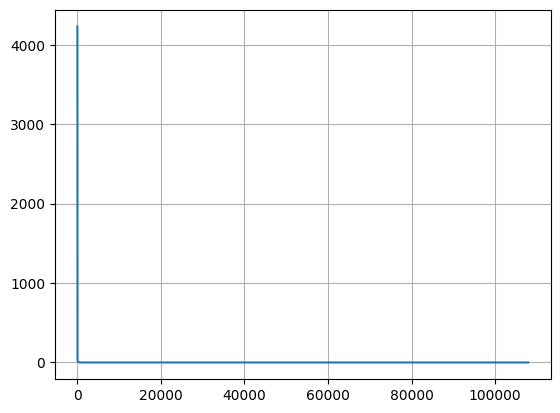


График зависимости значения функции от номера шага методом градиентного спуска

**Приложения**

**Приложение 1. Код алгоритма.**

import numpy as np  
import matplotlib.pyplot as plt  
  
# Проверяем, что все собственные значения матрицы положительны  
def is\_pos\_def(A):  
 return np.all(np.linalg.eigvals(A) > 0)  
  
  
# Умножаем обратную матрицу на вектор b, чтобы получить точное значение  
def exact\_solution(A, b):  
 return -np.linalg.inv(A).dot(b)  
  
  
# Находим значение функции  
def function(x):  
 return .5 \* np.matmul(np.matmul(x.T, A), x) + np.matmul(b.T, x)  
  
  
# Находим первую производную функции  
def derivative(x):  
 return np.matmul(.5 \* (np.add(A.T, A)), x) + b  
  
  
# Находим корень суммы квадратов элементов вектора x  
def error(x):  
 vect\_sum = 0  
 for elem in x:  
 vect\_sum += np.power(elem, 2)  
  
 return np.sqrt(vect\_sum)  
  
  
# Используем метод градиентного спуска  
class GradientMethod:  
 def \_\_init\_\_(self, A: np.array, x\_0: np.array, steps: int, eps: float):  
 self.ctr = 1  
 self.x\_k = x\_0  
 self.x\_k1 = self.x\_k - 1e-4 \* derivative(self.x\_k)  
  
 self.func\_sols = []  
  
 if steps == 0 and is\_pos\_def(A):  
 print("Матрица положительно определена")  
 print("A:\n", A)  
 print(f"lambda = {np.linalg.eigvals(A)}")  
 while error(self.x\_k1 - self.x\_k) > eps:  
 self.ctr += 1  
 self.x\_k = self.x\_k1  
 self.x\_k1 = self.x\_k - 1e-4 \* derivative(self.x\_k)  
 self.func\_sols.append(function(self.x\_k))  
 print(f"Погрешности метода: {abs(self.x\_k - exact\_solution(A, b))}")  
  
 elif is\_pos\_def(A):  
 while self.ctr < steps:  
 self.ctr += 1  
 self.x\_k = self.x\_k1  
 self.x\_k1 = self.x\_k - 1e-4 \* derivative(self.x\_k)  
 self.func\_sols.append(function(self.x\_k))  
  
 else:  
 print("Матрица не определена положительно")  
  
 print(f"\nТочка минимума: {self.x\_k1}, значение функции: {function(self.x\_k)}")  
 print(f"\nПогрешность функции в x: {abs(function(self.x\_k) - function(exact\_solution(A, b)))}")  
 print(f"Потребовалось итераций: {self.ctr}, точность: {error(self.x\_k1 - self.x\_k)}")  
  
  
 def display(self):  
 return self.func\_sols, self.ctr, self.x\_k1  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 A = np.array([[2., 3.5, 4.5, 6., 7., 2.],  
 [0., 6., 1.5, 5., 3., 0.5],  
 [6., 6., 0., 9.5, 5., 3.5],  
 [5., 2., 6., 9., 3., 3.],  
 [9., 6.5, 6., 2., 3., 1.],  
 [8., 0.5, 2., 7., 6., 6.]])  
 A = np.matmul(A, A.T)  
  
 b = np.array([1., 0.5, 0.5, 1.5, 1.5, 2.])  
 x\_0 = np.array([1.5, 0.5, 2.5, 0.5, 2., 1.5])  
  
 eps = 1e-7  
  
 func\_solves, steps, x = GradientMethod(A, x\_0, 0, eps).display()  
 plt.plot(func\_solves)  
 print(f"Результат для 1/4 итераций: {GradientMethod(A, x\_0, round(steps / 4), eps).display()[2]}\n")  
 print(f"Результат для 1/2 итераций: {GradientMethod(A, x\_0, round(steps / 2), eps).display()[2]}\n")  
 print(f"Результат для 3/4 итераций: {GradientMethod(A, x\_0, round(steps \* 3 / 4), eps).display()[2]}\n")  
  
  
 print(f"Точное решение: {exact\_solution(A, b)}")  
 print(f"Значение функции в точке x\*: {function(exact\_solution(A, b))}")  
  
 plt.grid()  
 plt.savefig('solution.png', bbox\_inches='tight')  
 plt.show()