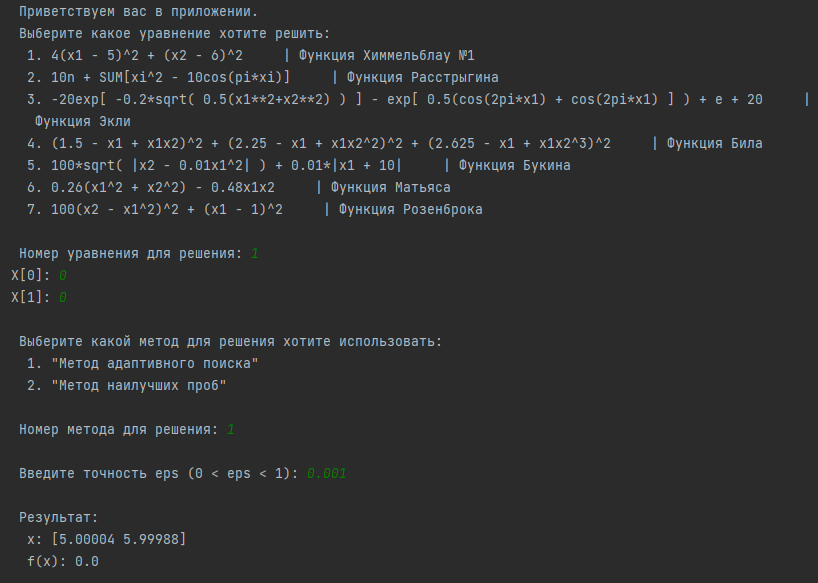
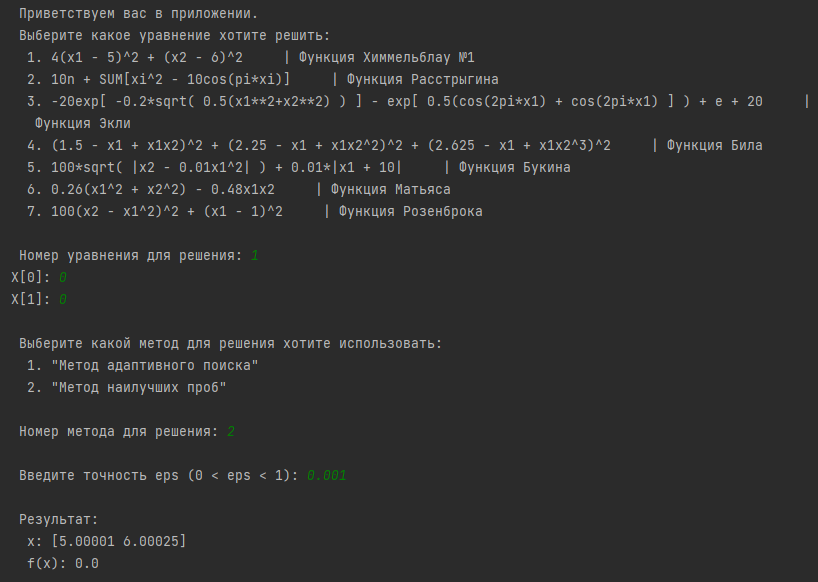
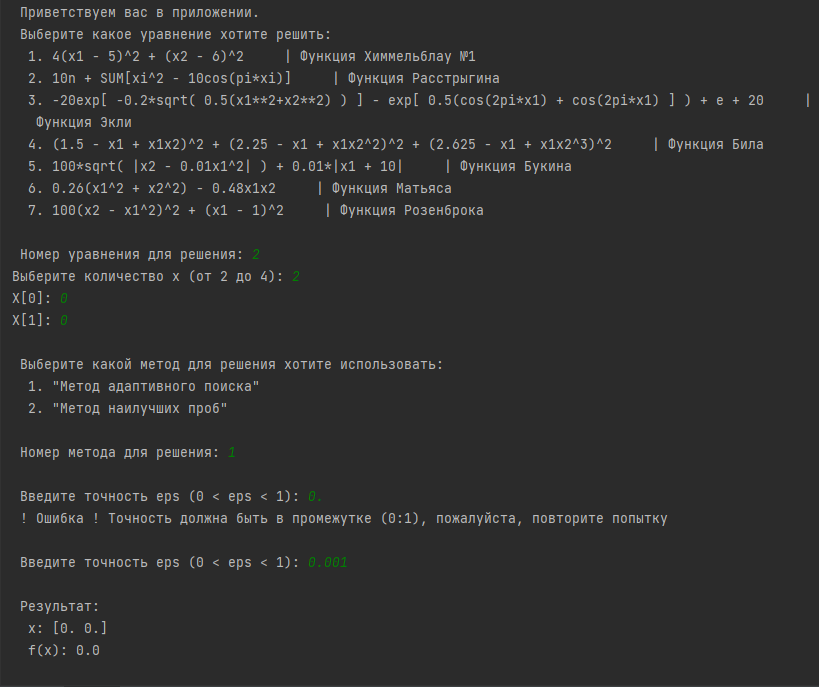
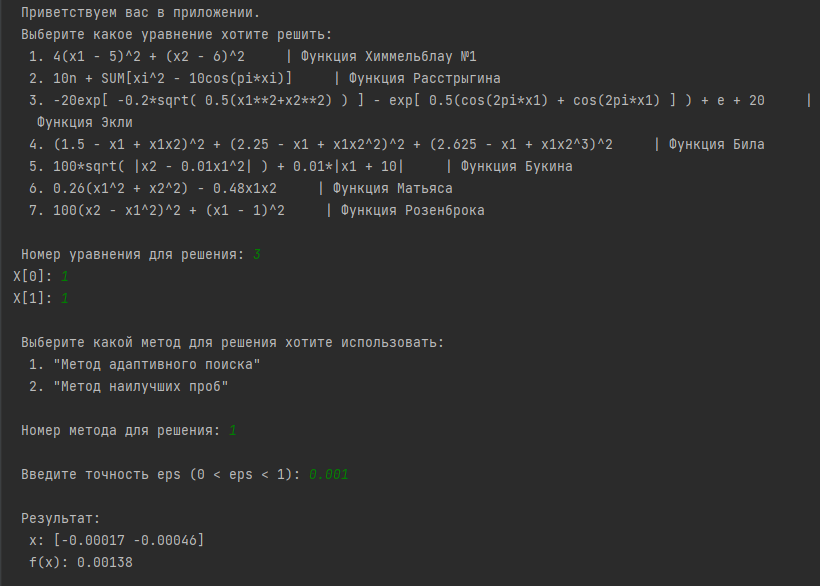
|  |  |  |
| --- | --- | --- |
| **Методы оптимизации** | | |
| Лабораторная работа №6  «Стохастическая оптимизация» | Группа студента: | ИВТ-363 |
| ФИО студента: | Бокова О.Д. |
| Вариант | 3 |
| Проверила: | Асанова Н.В. |

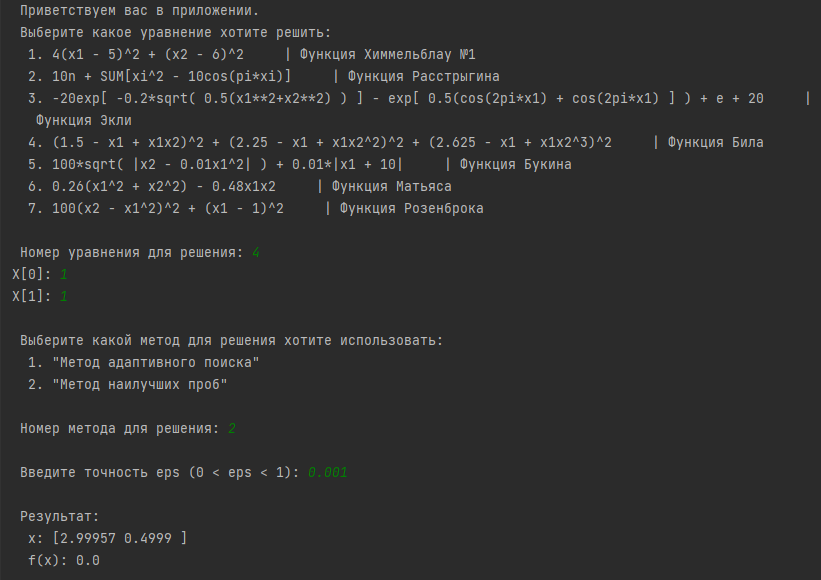
**Результаты работы программы:**

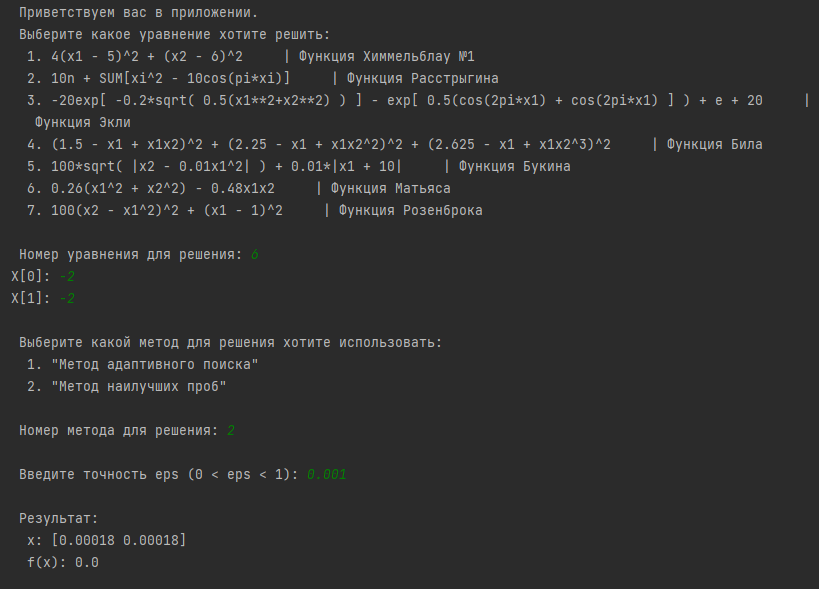


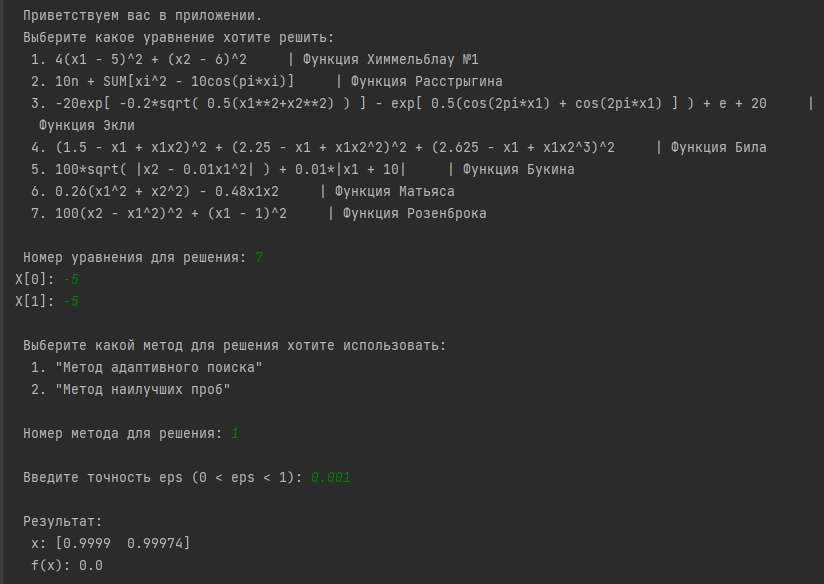












import random  
import numpy as np  
import numdifftools as nd  
from typing import Callable, List  
from scipy import optimize  
import math as mt  
  
global choice\_yp, choice\_method  
dimensionsInFunctions = [2 for i in range(7)]  
  
# ------------------------------------------------- ФУНКЦИЯ ВВОДА ЗНАЧЕНИЙ ---------------------------------------------  
def function\_input ():  
  
 def function\_is\_it\_number(number): # преобразует сроку choice\_yp в число int  
 if number.isdigit():  
 number = int(number) # проверка на число  
 else:  
 number = 1234567890  
  
 return number  
  
 start\_condition = ("\n Приветствуем вас в приложении. \n Выберите какое уравнение хотите решить: \n "  
 " 1. 4(x1 - 5)^2 + (x2 - 6)^2 | Функция Химмельблау №1 \n "  
 " 2. 10n + SUM[xi^2 - 10cos(pi\*xi)] | Функция Расстрыгина \n "  
 " 3. -20exp[ -0.2\*sqrt( 0.5(x1\*\*2+x2\*\*2) ) ] - exp[ 0.5(cos(2pi\*x1) + cos(2pi\*x1) ] ) + e + 20 | Функция Экли \n "  
 " 4. (1.5 - x1 + x1x2)^2 + (2.25 - x1 + x1x2^2)^2 + (2.625 - x1 + x1x2^3)^2 | Функция Била \n "  
 " 5. 100\*sqrt( |x2 - 0.01x1^2| ) + 0.01\*|x1 + 10| | Функция Букина \n "  
 " 6. 0.26(x1^2 + x2^2) - 0.48x1x2 | Функция Матьяса \n "  
 " 7. 100(x2 - x1^2)^2 + (x1 - 1)^2 | Функция Розенброка \n "  
 "\n Номер уравнения для решения: ")  
  
 choice\_yp = input(start\_condition)  
  
 choice\_yp = function\_is\_it\_number(choice\_yp)  
  
 iter = 0  
  
 if choice\_yp not in [1,2,3,4,5,6,7]:  
 while True:  
 choice\_yp = input(" Вы ввели неверный номер уравнения, пожалуйста, введите номер от 1 до 7,"  
 " чтобы выбрать уравнение для решения из списка: ")  
  
 choice\_yp = function\_is\_it\_number(choice\_yp)  
  
 iter += 1  
 if (iter % 4 == 0 and choice\_yp not in [1,2,3,4,5,6,7]):  
 print(start\_condition)  
 elif (choice\_yp in [1,2,3,4,5,6,7]):  
 break  
  
 if (choice\_yp == 2):  
 while True:  
 try:  
 number\_x = int(input('Выберите количество x (от 2 до 4): '))  
 except:  
 print("Данные некорректны")  
 continue  
 if 1 < number\_x < 5:  
 break  
 print('Такого номера не существует')  
 dimensionsInFunctions[choice\_yp - 1] = number\_x  
  
 if choice\_yp in [1,3,4,5,6,7]: # создает пустые размерные матрицы !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
 x = np.zeros(2, float)  
 elif choice\_yp == 2:  
 x = np.zeros(number\_x, float)  
  
 for i in range(len(x)): # ввод данных  
 x[i] = float(input(f"X[{i}]: "))  
  
 if choice\_yp == 2 and number\_x == 3:  
 x[0] = random.uniform(0.41, 0.72)  
 x[1] = random.uniform(0.41, 0.72)  
 x[2] = random.uniform(0.41, 0.72)  
  
 elif choice\_yp == 2 and number\_x == 4:  
 x[0] = random.uniform(0.41, 0.72)  
 x[1] = random.uniform(0.41, 0.72)  
 x[2] = random.uniform(0.41, 0.72)  
 x[3] = random.uniform(0.41, 0.72)  
  
 method\_choice\_condition = ("\n Выберите какой метод для решения хотите использовать: \n "  
 " 1. \"Метод адаптивного поиска\" \n "  
 " 2. \"Метод наилучших проб\" \n "  
 "\n Номер метода для решения: ")  
  
 choice\_method = input(method\_choice\_condition) # выбор метода решения  
  
 choice\_method = function\_is\_it\_number(choice\_method)  
  
 iter = 0  
  
 if choice\_method not in [1,2]:  
 while True:  
 choice\_method = input(" Вы ввели неверный номер метода, пожалуйста, введите номер от 1 до 2,"  
 " чтобы выбрать способ решения из списка ниже: ")  
  
 choice\_method = function\_is\_it\_number(choice\_method)  
  
 iter += 1  
 if (iter % 4 == 0 and choice\_method not in [1,2]):  
 print(method\_choice\_condition)  
 elif (choice\_method in [1,2]):  
 break  
  
 while True:  
 eps = float(input("\n Введите точность eps (0 < eps < 1): "))  
  
 if 1 > eps > 0:  
 break  
 elif eps <= 0:  
 print(" ! Ошибка ! Точность должна быть в промежутке (0:1), пожалуйста, повторите попытку")  
 elif eps > 1:  
 print(" ! Ошибка ! Точность должна быть в промежутке (0:1), пожалуйста, повторите попытку")  
  
  
 return choice\_yp, choice\_method, x, choice\_method, eps  
# ----------------------------------------------------------------------------------------------------------------------  
  
# ------------------------------------------------- ВЫБОР УРАВНЕНИЯ ----------------------------------------------------  
def function\_selection(choice\_yp, x\_i):  
 try:  
  
 if choice\_yp == 1:  
 return lambda x: 4 \* (x[0] - 5) \*\* 2 + (x[1] - 6) \*\* 2  
  
 if choice\_yp == 2:  
 if (x\_i == 2):  
 return lambda x: 10 \* x\_i + ((x[0] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[0])) + (x[1] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[1])))  
  
 if (x\_i == 3):  
 return lambda x: 10 \* x\_i + (  
 (x[0] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[0])) + (x[1] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[1])) + (  
 x[2] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[2])))  
  
 if (x\_i == 4):  
 return lambda x: 10 \* x\_i + (  
 (x[0] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[0])) + (x[1] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[1])) + (  
 x[2] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[2])) + (x[3] \*\* 2 - 10 \* mt.cos(2 \* 3.14 \* x[3])))  
  
 if choice\_yp == 3:  
 return lambda x: -20 \* mt.exp(-0.2 \* mt.sqrt(0.5 \* (x[0] \*\* 2 + x[1] \*\* 2))) - mt.exp(  
 0.5 \* (mt.cos(2 \* 3.14 \* x[0]) + mt.cos(2 \* 3.14 \* x[1]))) + 2.71828 + 20  
  
 if choice\_yp == 4:  
 return lambda x: (1.5 - x[0] + x[0] \* x[1]) \*\* 2 + (2.25 - x[0] + x[0] \* x[1] \*\* 2) \*\* 2 + (2.625 - x[0] + x[0] \* x[1] \*\* 3) \*\* 2  
  
 if choice\_yp == 5:  
 return lambda x: 100 \* mt.sqrt(abs(x[1] - 0.01 \* x[0] \*\* 2)) + 0.01 \* abs(x[0] + 10)  
  
 if choice\_yp == 6:  
 return lambda x: 0.26 \* (x[0] \*\* 2 + x[1] \*\* 2) - 0.48 \* x[0] \* x[1]  
  
 if choice\_yp == 7:  
 return lambda x: 100 \* (x[1] - x[0] \*\* 2) \*\* 2 + (x[0] - 1) \*\* 2  
  
 except NameError:  
 print("\n\n Введенные вами значения x0 не удовлетворяют ОДЗ данного уравнения! "  
 " \nПросьба изменить вводимые x0 при следующем вызове данного уравнения.")  
 exit()  
# ----------------------------------------------------------------------------------------------------------------------  
  
# ------------------------------------------------- СТОХАСТИЧЕСКИЕ МЕТОДЫ ----------------------------------------------  
def best\_samples(func: Callable[[np.array], float], x\_start: List[float], eps: float, N: int = 100, M: int = 300,  
 b: float = 0.5):  
 x = x\_start  
 k = 0  
 h = 1  
 while k < N:  
 y\_xlam = []  
 f = []  
  
 for \_ in range(M):  
 e = np.random.uniform(-1, 1, len(x))  
 y = x + h \* e / np.linalg.norm(e)  
 y\_xlam.append(y)  
 f.append(func(y))  
  
 min\_index = np.argmin(f)  
 f\_min = f[min\_index]  
  
 if f\_min < func(x):  
 x = y\_xlam[min\_index]  
 k += 1  
 else:  
 if h <= eps:  
 return x  
 elif h > eps:  
 h \*= b  
 return x  
  
  
def adaptive\_method(func: Callable[[np.array], float], x\_start: List[float], eps: float, N: int = 100, M: int = 300,  
 a: float = 1.5, b: float = 0.5):  
 x = x\_start  
 h = 1  
 k = 0  
 j = 1  
 while k < N:  
 e = np.random.uniform(-1, 1, len(x))  
 y = x + h \* e / np.linalg.norm(e)  
  
 if func(y) < func(x):  
 z = x + a \* (y - x)  
 if func(z) < func(x):  
 x = z  
 h \*= a  
 k += 1  
 j = 1  
 continue  
  
 if j < M:  
 j += 1  
 else:  
 if h <= eps:  
 break  
  
 h \*= b  
 j = 1  
  
 return x  
# ----------------------------------------------------------------------------------------------------------------------  
  
# ------------------------------------------------- ГЛАВНОЕ ТЕЛО -------------------------------------------------------  
if \_\_name\_\_ == '\_\_main\_\_':  
 while True:  
 choice\_yp, choice\_method, x0, choice\_method, eps = function\_input()  
 function = function\_selection(choice\_yp, len(x0))  
  
 if choice\_method == 1:  
 x\_min = adaptive\_method(function, x0, eps)  
  
 if choice\_method == 2:  
 x\_min = best\_samples(function, x0, eps)  
  
 print(f"\n Результат: \n x: {np.round(x\_min, 5)} \n f(x): {np.round(function(x\_min), 5)} \n")  
# ----------------------------------------------------------------------------------------------------------------------