МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

**по дисциплине Методы поисковой оптимизации**

Работу выполнила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ромашкина А.А.

Факультет Компьютерных технологий и прикладной математики

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии курс 4

Краснодар

2022

СОДЕРЖАНИЕ

[Задание 3](#_Toc116326542)

[Алгоритм 3](#_Toc116326543)

[Результат работы программы 4](#_Toc116326544)

[Листинг 5](#_Toc116326545)

Тема работы: Метод градиентного спуска с постоянным шагом.

# Задание

Найти локальный минимум функции методом наискорейшего спуска (*x*0= (0,5; 1), ; *М*=10).

# Алгоритм

*Шаг 1*. Задать *х ,* 0 < ε < 1, ε 1 > 0, ε 2 > 0, *М* –предельное число итераций. Найти градиент функции в произвольной точке .

*Шаг 2.* Положить *k* = 0.

*Шаг 3.* Вычислить *f(xk).*

*Шаг 4.* Проверить выполнение критерия окончания *|f(x\*)| <* ε1*:*

а) если критерий выполнен, то расчёт закончен и *х\* = xk*;

б) если критерий не выполнен, то перейти к шагу 5.

*Шаг 5.* Проверить выполнение неравенства *k ≥ M:*

а) если неравенство выполнено, то расчет окончен: *х\* = xk*;

б) если нет, то перейти к шагу 6.

*Шаг 6*. Задать величину шага *tk.*

*Шаг 7*. Вычислить *xk+1 = xk - tkf(xk)*.

*Шаг 8*. Проверить выполнение условия

*f*(*xk+1*) *- f*(*xk*) *< 0* (или *|f*(*xk+1*) *- f*(*xk*) *|<*  ε *||f(xk)||2*);

а) если условие выполнено, то перейти к шагу 9;

б) если условие не выполнено, положить  иперейти к шагу 7.  
*Шаг 9*. Проверить выполнение условий

*||xk+1 - xk|| <* ε2*, ||f(xk+1) - f(xk)|| <* ε2:

а) если оба условия выполнены при текущем значении *k* и *k = k -*1, то расчет окончен и *x\* = xk+1*;

б) если хотя бы одно из условий не выполнено, положить *k = k* +1 и перейти к шагу 3.

# Результат работы программы

Вся работа программы представлена на рисунках 1-2 ниже.

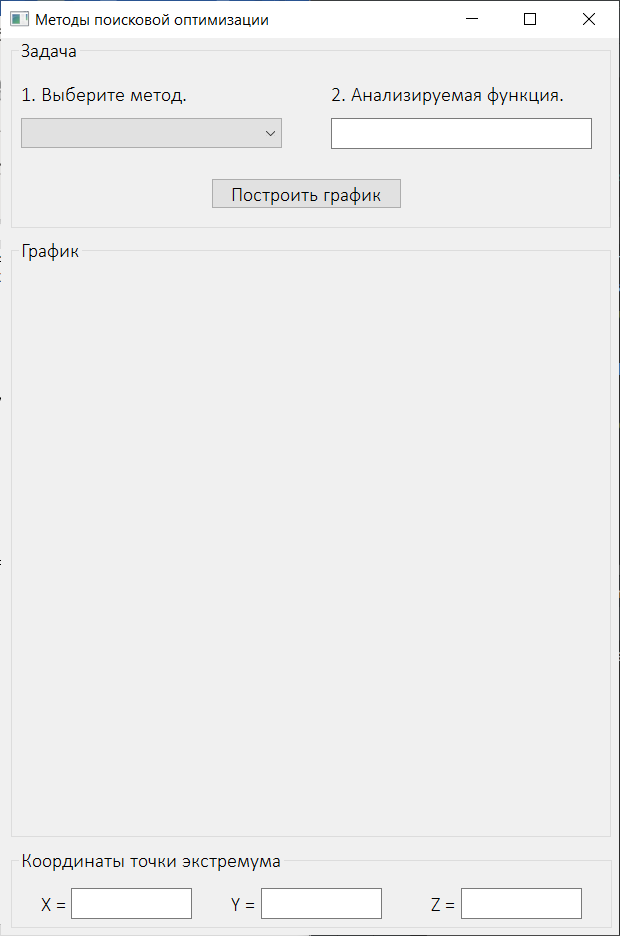


Рисунок 1 – Начальный вид программы при запуске

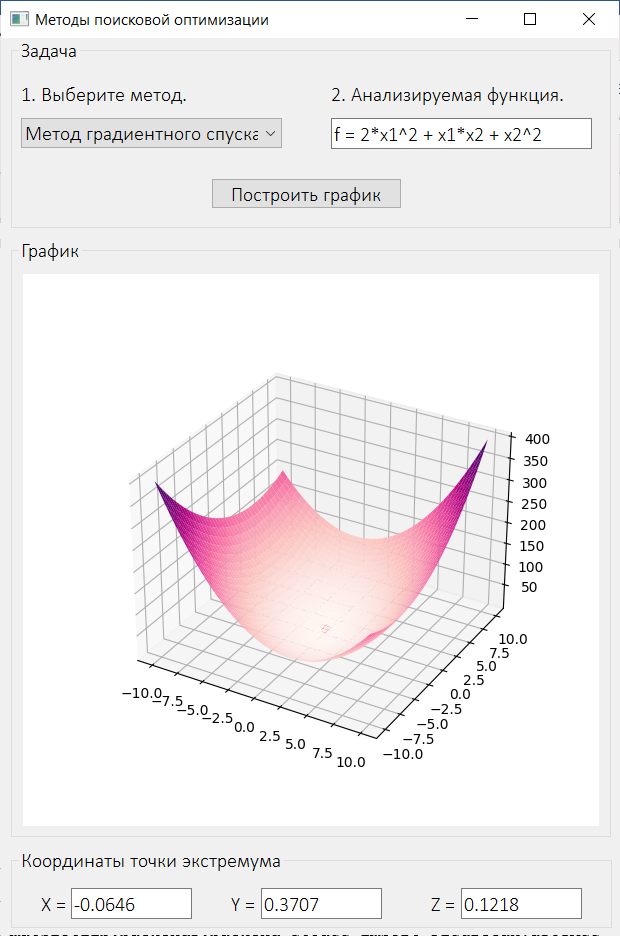


Рисунок 2 – Вид программы при выборе метода градиентного спуска с постоянным шагом

# Листинг

import math

import numpy as np

import sys

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import \*

from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg

from matplotlib.figure import Figure

from lab1 import Ui\_MainWindow

class MainWindow(QtWidgets.QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

QtWidgets.QMainWindow.\_\_init\_\_(self)

self.gridlayout = None

self.f = Function()

self.setupUi(self)

self.comboBox.activated.connect(self.on\_selected)

self.pushButton.clicked.connect(self.button1\_clicked)

self.graph = Canvas(self, width=1, height=1, dpi=100)

def on\_selected(self):

if self.comboBox.currentIndex() == 0:

self.lineEdit.setText("f = 2\*x1^2 + x1\*x2 + x2^2")

else:

self.lineEdit.setText("")

def button1\_clicked(self):

x = np.linspace(-10, 10, 42)

y = np.linspace(-10, 10, 42)

xgrid, ygrid = np.meshgrid(x,y)

zgrid = np.array([self.f.function([xgrid[i], ygrid[i]]) for i in range(42)])

self.graph.axes.plot\_surface(xgrid, ygrid, zgrid, cmap ='RdPu', linewidths = 0.2);

g = GradientDescent(self.f)

x\_y = g.gradient\_descent()

z = self.f.function(x\_y)

self.graph.axes.plot(x\_y[0], x\_y[1], z, 'ro')

self.lineEdit\_2.setText(str(round(x\_y[0], 4)))

self.lineEdit\_3.setText(str(round(x\_y[1], 4)))

self.lineEdit\_4.setText(str(round(z, 4)))

self.gridlayout = QGridLayout(self.groupBox)

self.gridlayout.addWidget(self.graph)

self.show()

class Canvas(FigureCanvasQTAgg):

def \_\_init\_\_(self, parent=None, width=1, height=1, dpi=100):

fig = Figure(figsize=(width, height), dpi=dpi)

self.axes = fig.add\_subplot(111, projection='3d')

super(Canvas, self).\_\_init\_\_(fig)

def norma(x):

return math.sqrt(np.sum(x \*\* 2))

class Function:

@staticmethod

def function(x\_k):

return 2 \* x\_k[0] \* x\_k[0] + x\_k[0] \* x\_k[1] + x\_k[1] \* x\_k[1]

@staticmethod

def gradient\_function(x\_k):

return np.array([4 \* x\_k[0] + x\_k[1], x\_k[0] + 2 \* x\_k[1]])

class GradientDescent:

def \_\_init\_\_(self, f):

self.f = f

self.x\_k = np.array([0.5, 1])

self.e1 = 0.1

self.e2 = 0.15

self.M = 10

def gradient\_descent(self):

k = 0

while True:

gradient = self.f.gradient\_function(self.x\_k)

if norma(gradient) < self.e1:

return self.x\_k

else:

if k >= self.M:

return self.x\_k

else:

t\_k = 0.1

while True:

x\_k\_plus\_1 = self.x\_k - t\_k \* self.f.gradient\_function(self.x\_k)

if self.f.function(x\_k\_plus\_1) - self.f.function(self.x\_k) < 0:

if norma(x\_k\_plus\_1 - self.x\_k) < self.e2 and abs(

self.f.function(x\_k\_plus\_1) - self.f.function(self.x\_k)) < self.e2:

return x\_k\_plus\_1

else:

k += 1

self.x\_k = x\_k\_plus\_1

break

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

window = MainWindow()

window.show()

sys.exit(app.exec\_())