Содержание

ВВЕДЕНИЕ

Задачей оптимизации в математике, информатике и исследовании операций называется задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств.

В данной работе разбирается оптимизация функции от двух переменных методом наискорейшего спуска, а также разработана программа, которая находит минимум функции, задаваемой пользователем в явном виде. Отчёт содержит полный текст программы на языке Python, описание всех функций, инструкцию пользователю и тестовый пример.

1. МИНИМИЗАЦИЯ ФУНКЦИИ НЕСКОЛЬКИХ ПЕРЕМЕННЫХ

1.1. Содержательное описание задачи

Задана некоторая функция от двух переменных: f(x,y). Так же задано начальное приближение x0 и y0 и точность вычислений eps. Задача состоит в том, чтобы найти минимум заданной функции (провести оптимизацию функции). Результатом должно служить минимальное значение функции, а так же соответствующие значения переменных x и y. Функция должна в явном виде вводиться пользователем.

1.2. Формальная постановка задачи

Задана некоторая функция от двух переменных: f(x,y). От пользователя требуется ввести следующие данные: начальное приближение x0 и y0 и точность вычислений eps. Также для надежности следует предусмотреть ограничение количества итераций алгоритма. Вводимые величины являются числами, точность вычислений eps должна быть больше 0. Функция f(x,y) задаётся пользователем в явном виде. Требуется найти минимум функции f(x,y) с помощью метода наискорейшего спуска.

Рассмотрим алгоритм метода наискорейшего спуска Данный метод использует понятие и свойства градиента.

Градиент (от лат. gradiens, род. падеж gradientis — шагающий, растущий) — вектор, своим направлением указывающий направление наискорейшего возрастания некоторой величины , значение которой меняется от одной точки пространства к другой (скалярного поля), а по величине (модулю) равный быстроте роста этой величины в этом направлении.

Например, если взять в качестве высоту поверхности Земли над уровнем моря, то её градиент в каждой точке поверхности будет показывать «направление самого крутого подъёма», и своей величиной характеризовать крутизну склона.

С математической точки зрения градиент — это производная скалярной функ-

ции, определенной на векторном пространстве.

Пространство, на котором определена функция и её градиент может быть вообще говоря как обычным трехмерным пространством, так и пространством любой другой размерности любой физической природы или чисто абстрактным.

Термин впервые появился в метеорологии, а в математику был введен Максвеллом в 1873 г. Обозначение $grad$ тоже предложил Максвелл.

Т.е. если градиент показывает направление наискорейшего роста функции, то антиградиент - градиент с противоположным знаком - показывает направление наискорейшего убывания функции (в данной точке). Это свойство антиградиента лежит в основе градиентных методов, в частности, метода наискорейшего спуска.

Для нахождения минимума F задаем некоторое начальное приближение $x_i^{(0)}(i = 1, ..., n)$ и строим последующие приближения по формуле:

$$x_i^{(j+1)} = x_i^{(j)} + \lambda_i^{(j)} v_i^{(j)} (i = 1, ..., n; j = 0, 1, 2..)$$

где направления $v_i^{(j)}$ и величина шага на j-м шаге соответственно равны:

$$v_i^{(j)} = -\frac{\partial F}{\partial x_i}$$

$$\lambda_i^{(j)} = \sum_i \left(\frac{\partial F}{\partial x_i} \right)^2 \left[\sum_{i,j} \frac{\partial^2 F}{\partial x_i \partial x_j} \frac{\partial F}{\partial x_i} \frac{\partial F}{\partial x_j} \right]^{-1}$$

Все производные вычисляются при $x_i = x_i^{(j)}$.

Итерационный процесс продолжается до тех пор, пока не будет удовлетворяться условие

$$|x_i^{(j+1)} - x_i^{(j)}| \leq e; (i = 1, ..., n)$$

или все производные $\frac{\partial F}{\partial x_k}$ не станут равны нулю.

2. РАЗРАБОТКА АЛГОРИТМА

2.1. Разработка графического интерфейса пользователя

Для решения задачи требуются иметь следующие исходные данные: начальное приближение переменных x и y, а также точность вычислений eps и максимальное кол-во итераций. Для ввода этих значений необходимо предусмотреть отдельные поля. Для ввода функции f(x,y) также предусмотреть поле, причем необходимо обеспечить ввод в явном виде: например $f(x, y) = 5x + 4y^2 + 2$. Кроме того, можно предоставить пользователю возможность скомпилировать исходную функцию в код на Фортране, что позволит сильно увеличить скорость вычислений. Для переключения между режимами интерпретации и компиляции функции предусмотреть элемент ComboBox. Так как компиляция занимает довольно продолжительное время, предусмотреть отдельную кнопку для принятия функции, и продублировать её в меню "Запуск".

Для того, чтобы можно было проследить динамику работы метода, приближения, полученные на каждом шаге метода, будут выводится в таблицу. Кроме того, будет построен график исходной функции с помощью программы Gnuplot; на этом графике так же соединёнными точками будут показываться приближения, полученные с помощью метода.

Для вычисления результата создать кнопку "Расчет"и продублировать её в пункте меню Запуск->Найти минимум. В панели меню предусмотреть пункт для открытия окна “Справка” и справки по управлению программой Gnuplot, "Вид>"Показать график"и пункт меню Файл->Выход.

Итак, внешний вид разработанного интерфейса представлен на рисунке 1.

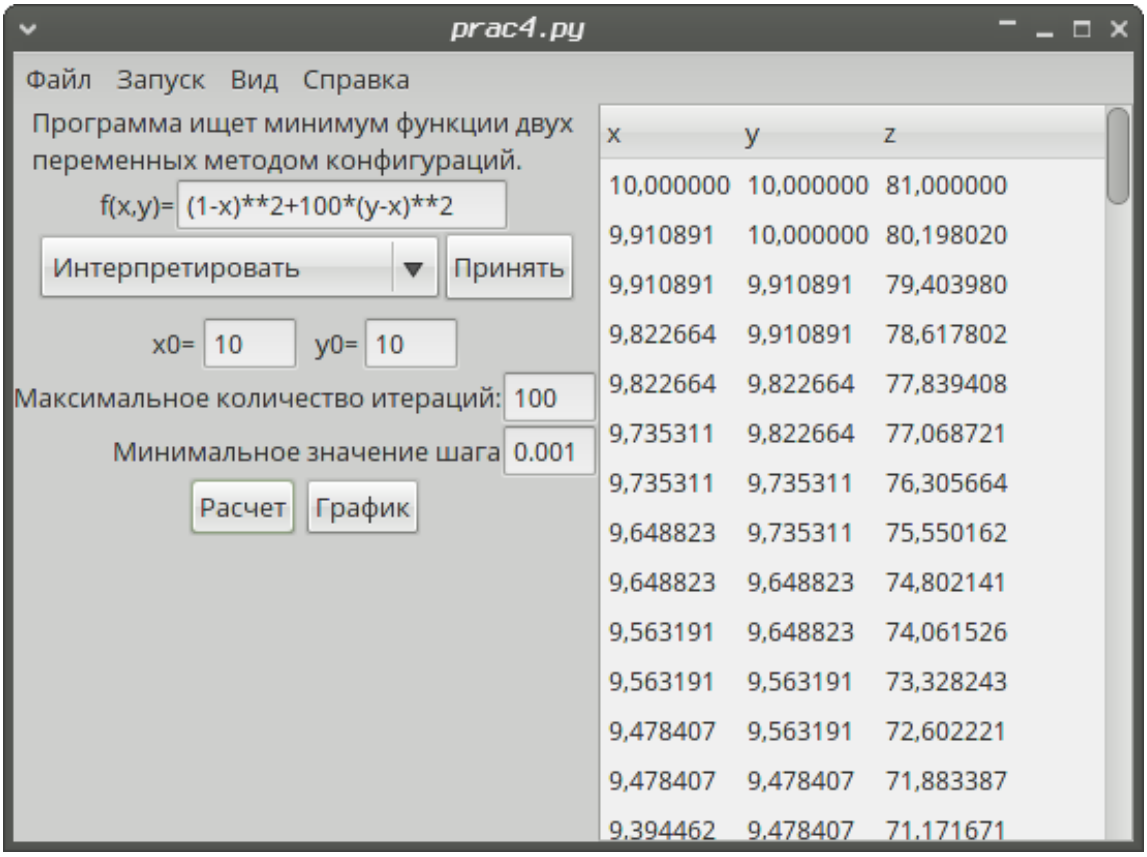

Рисунок 1 – Разработанный интерфейс программы

2.2. Разработка структур данных

Для хранения исходных данных будем использовать следующие переменные:

max_iter - максимальное количество итераций,

eps - точность вычислений,

xs - начальные приближения,

f - минимизируемая функция,

df_dx - первые производные минимизируемой функции,

d2f_dx - матрица вторых производных (матрица Гессе) функции.

2.3. Разработка структуры алгоритма

Основную программу можно разбить на три участка: считывание значений , нахождения минимума функции и вывод полученных результатов.

1) Для нахождения минимума функции будет использоваться функция gradient, принимающая в качестве параметров указанные в предыдущем разделе переменные, и возвращающая результат в виде списка приближений к минимуму. Приближения представляют собой кортежи длиной n для функции n переменных.

2) Подпрограмма ввода данных input_data.

3) Подпрограмма вывода данных output_data.

2.4. Схема алгоритма

На рисунке 2 представлена схема алгоритма минимизации функции многих переменных градиентным методом наискорейшего спуска.

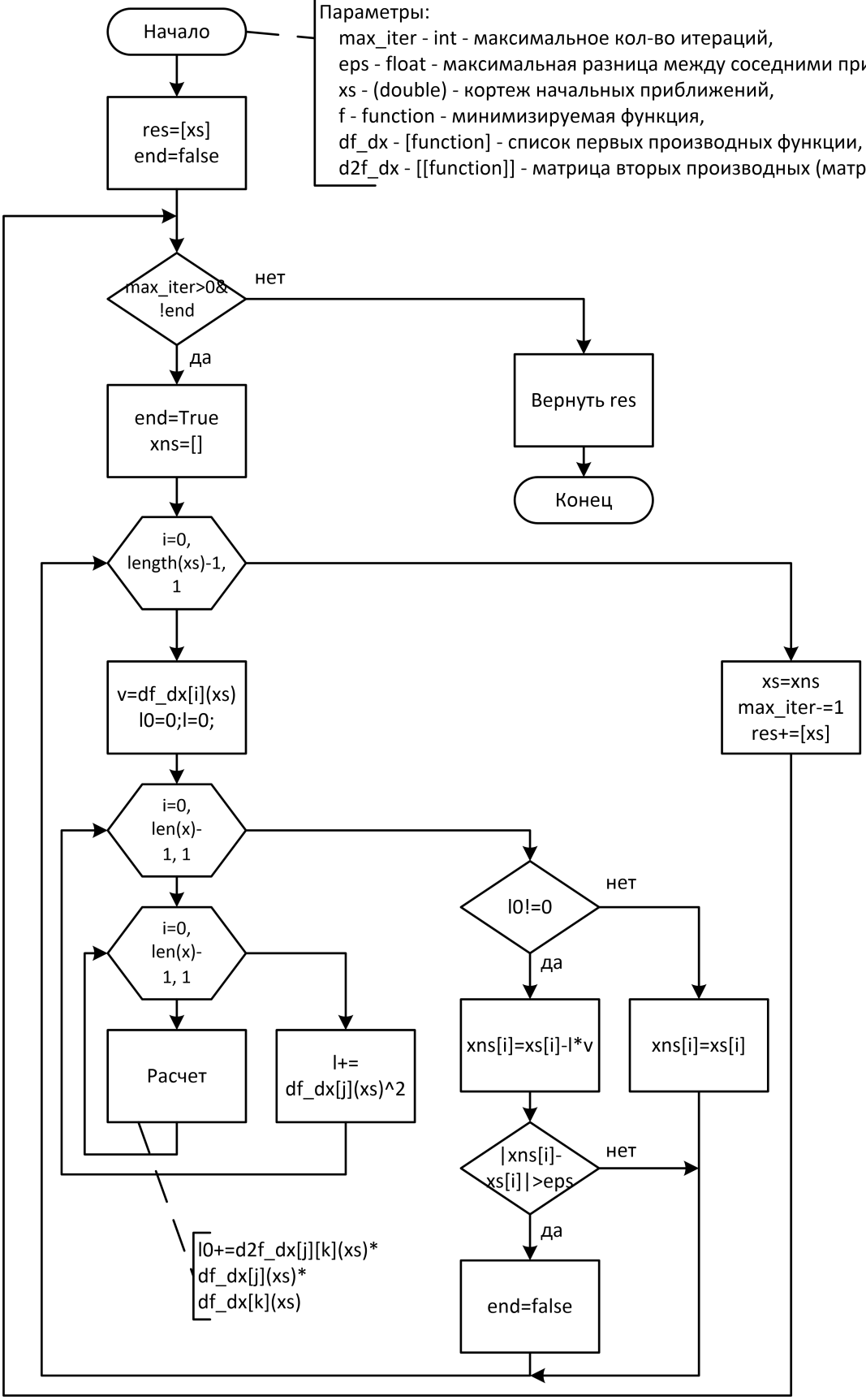

Рисунок 2 – Схема алгоритма метода наискорейшего спуска

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Описание переменных и структур данных

Для хранения исходных данных будем использовать следующие переменные:

max_iter - float - максимальное количество итераций,

eps - float - точность вычислений,

xs - (float) - начальные приближения,

f - UserFunc - минимизируемая функция,

df_dx - [UserFunc] - первые производные минимизируемой функции,

d2f_dx - [[UserFunc]] - матрица вторых производных (матрица Гессе) функции.

Класс UserFunc представляет функцию, задаваемую с помощью строки.

3.2. Описание функций

1. gradient(max_iter,eps,xs,f,df_dx,d2f_dx)

Функция gradient проводит минимизацию функции f, используя начальное приближение xs, первые производные df_dx, вторые производные d2f_dx, при максимальном кол-ве итераций max_iter и точностью eps.

Возвращает список, содержащий приближения к минимуму функции.

Параметры функции представлены в таблице 1 :

Таблица 1 – Параметры функции минимизации

имя	тип	предназначение
max_iter	int	максимальное кол-во итераций,
eps	float	максимальная разница между соседними приближениями,
xs	(double)	кортеж начальных приближений,
f	function	минимизируемая функция,
df_dx	[function]	список первых производных функции,
d2f_dx	[[function]]	матрица вторых производных (матрица Гессе)

2. input_data(self)

Подпрограмма ввода исходных данных.

3. output_data(self,res)

Подпрограмма вывода результатов.

4. on_run_click(self,button,data=None)

Подпрограмма считвания данных, минимизации функции и вывода результатов.

5. show_chart(self):

Подпрограмма построения графика.

4. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ

Данная программа находит минимум функции f(x,y). Программа не требует установки. Для её запуска необходимо открыть файл prac4.py. Внимание: для работы приложения на компьютере должен быть установлен Python 3, GTK+3, GObject-introspection, Gnuplot и SymPy.

От пользователя требуется ввести следующие исходные данные:

1) Функция f(x,y). После её ввода необходимо выбрать режим вычисления (интерпретация или компиляция) и нажать кнопку "Принять".

2) Начальное приближение x0.

3) Начальное приближение y0.

4) Точность вычислений eps.

5) Максимальное количество итераций max_iter.

Примечание: Функция вводится в явном виде, т.е. в поле f(x,y) можно ввести "(x-1)**2+100*(y-x**2)**2". Учтите, точность вычислений доложена быть больше 0. После ввода значений для получения результата требуется либо открыть пункт меню Запуск->Минимизировать функцию, либо нажать на кнопку «Расчет!». После этого

на экран будут выведен найденные приближения к минимуму функции. Учтите, нахождение минимума функции может занять несколько минут.

5. ТЕСТОВАЯ ЗАДАЧА

5.1. Аналитическое решение и умозрительные результаты

Введём функцию Розенброка:

$$f(x, y) = (x - 1)^2 + 100(y - x^2)^2$$

, стандартно используемую для тестирования методов оптимизации.

Её минимум находится в точке $(1, 1)$ и равен 0.

5.2. Решение, полученное с использованием разработанного ПО

Ниже на рисунке 3 представлен пример работы программы минимимизации функции двух переменных методом наискорейшего спуска.

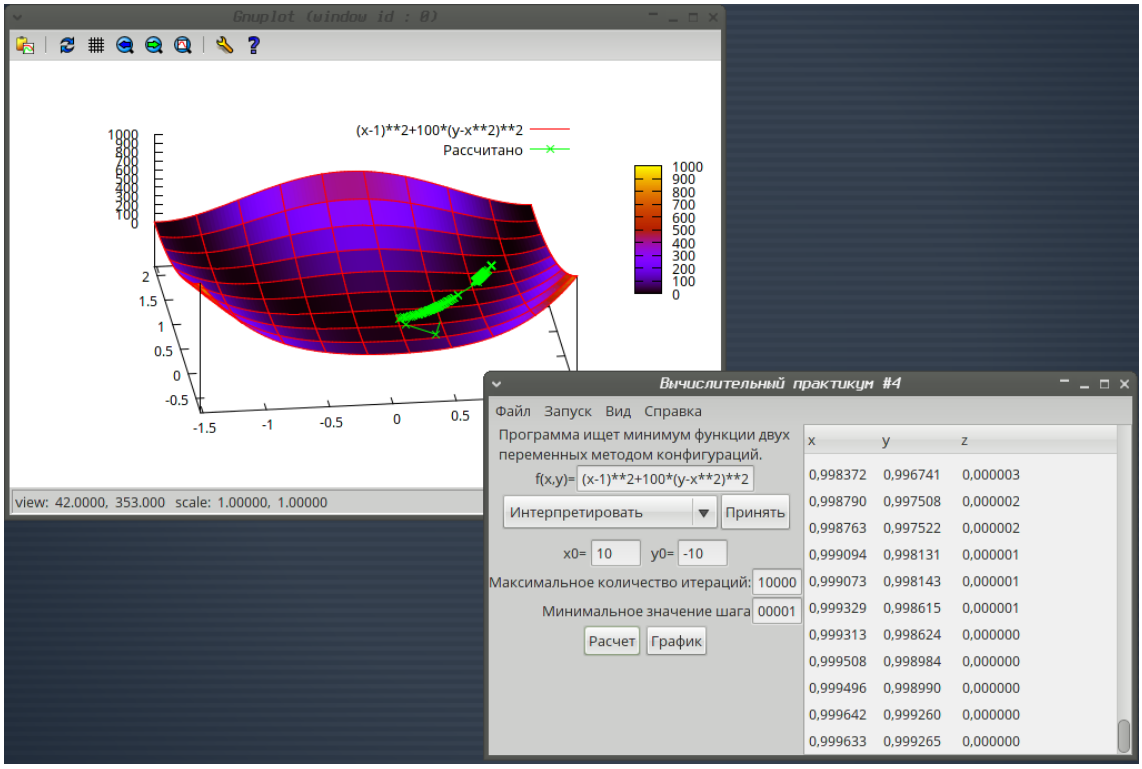

Рисунок 3 – Пример работы программы минимизации функции

5.3. Выводы

Данная программа находит минимум функции от двух переменных методом наискорейшего спуска.

Так как функция задается строкой, а используемый метод требует задания производных первого и второго порядка, для их расчета используется библиотека символьной алгебры SymPy.

ЗАКЛЮЧЕНИЕ

В данной работе рассматривалась проблема оптимизации функции двух переменных. Для решения проблемы был использован метод наискорейшего спуска. Была написана программа на языке Python, реализующий данный метод. Примеча-

тельно то, что исходная функция в явном виде вводится пользователем, что делает программу намного универсальнее.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://python.org
2. http://www.gtk.org
3. http://ru.wikipedia.org
4. http://en,wikipedia.org

ПРИЛОЖЕНИЕ

Далее приводится текст программы оптимизации функции двух переменных методом наискорейшего спуска, написанной на Python 3.

```
#!/usr/bin/env python
from math import *
import subprocess as subp
from gi.repository import Gtk,Gdk
from tempfile import NamedTemporaryFile
from os import remove
from sympy import sympify,diff
from sys import stderr
from signal import signal,SIG_IGN,SIGCHLD
from sympy.utilities.autowrap import autowrap
from time import time

class UserFunc:
    allow_func={"sin":sin,
                "cos":cos,
                "exp":exp}
    def __init__(self,expr="",compile_to=''):
        self.expr=expr
        if compile_to:

            var=sympify('x'),sympify('y')

            self.compiled=autowrap(sympify(expr),language=compile_to,args=var)
            print(expr,'compiled!')
        else:
```

```
            self.compiled=None
            pass

    def __call__(self,x,y):

        if self.compiled:
            return self.compiled(x,y)
        elif self.expr:
            self.allow_func["x"]=x
            self.allow_func["y"]=y
            try:
                z=eval(self.expr,{"__builtins__":None},self.allow_func)
            except:
                raise TypeError
            else:
                try:
                    z=float(z)
                except:
                    raise TypeError
            return z
    def __str__(self):
        return str(self.expr)
    def __repr__(self):
        return str(self.expr)
def gradient(max_iter,eps,xs,f,df_dx,d2f_dx):
    '''
    Функция gradient проводит минимизацию функции f,
    используя начальное приближение xs, первые производные df_dx,
    вторые производные d2f_dx, при максимальном кол—ве итераций max_iter
    и точностью eps.
    Возвращает список, содержащий приближения к минимуму функции.
    Параметры:
    max_iter — int — максимальное кол—во итераций,
    eps — float — максимальная разница между соседними приближениями,
    xs — (double) — кортеж начальных приближений,
    f — function — минимизируемая функция,
    df_dx — [function] — список первых производных функции,
    d2f_dx — [[function]] — матрица вторых производных (матрица Гессе)
    '''
    res=[xs]

    end=False
    t=time()
    while max_iter>0 and not end:
        end=True
        xns=[]
        for i in range(0,len(xs)):
            v=df_dx[i](*xs)
            l0=0;l=0;
            for j in range(0,len(xs)):
                for k in range(0,len(xs)):
                    #print('a',end='')
                    a=d2f_dx[j][k](*xs)
                    #print(a)
                    a*=df_dx[j](*xs)
                    a*=df_dx[k](*xs)
                    l0+=a
                l+=df_dx[j](*xs)**2
            if l0 and l:
```

```
                l=l/l0;
                xns.append(xs[i]-l*v)
                if abs(xns[-1]-xs[i])>eps:
                    end=False
            else:
                xns.append(xs[i])
                print("Extremum!")
        xs=xns
        max_iter-=1
        res.append(xs)
    print(time()-t)
    return res


class Application(Gtk.Builder):
    def __init__(self,ui_filename):
        #signal(SIGCHLD,SIG_IGN)
        Gtk.Builder.__init__(self)
        self.add_from_file(ui_filename)
        self.connect_signals(self)
        self.plot=None
        self.tempfile=NamedTemporaryFile(delete=False)
        self.tempfile.close()
    def show_msg(self,msg):
        md=Gtk.MessageDialog(None, Gtk.DialogFlags.MODAL,
                    Gtk.MessageType.WARNING, Gtk.ButtonsType.OK, msg);
        md.run ();
        md.destroy();
    def show(self,form_name):
        window = self.get_object(form_name)
        window.show()
        Gtk.main()
    def on_window_destroy( self,widget, data=None):
        self.get_object('window1').hide()
        if self.plot and self.plot.poll()==None:
            self.plot.stdin.close()
            self.plot.terminate()
            if self.plot.poll()==None:
                self.plot.kill()
                self.plot.wait()
        if self.tempfile:
            if not self.tempfile.closed:
                self.tempfile.close()
            remove(self.tempfile.name)
        Gtk.main_quit()
    def on_func_clicked(self,button,data=None):
        expr=self.get_object("entry1").get_text();
        if self.get_object("combobox1").get_property("active")==1:
            lang='F95'
        elif self.get_object("combobox1").get_property("active")==2:
            lang='C'
        else:
            lang=''
        self.f=UserFunc(expr,lang)
        var=sympify('x'),sympify('y')
        self.df_dx=[]
        for i in var:
            self.df_dx.append(UserFunc(str(diff(sympify(self.f.expr),i)),lang))
        self.d2f_dx=[]
```

```
        for i in range(0,len(var)):
            t=[]
            for j in range(0,i):
                t.append(self.d2f_dx[j][i])
            for j in range(i,len(var)):
                t.append(UserFunc(str(diff(sympify(self.df_dx[i].expr),var[j])),lang))
            self.d2f_dx.append(t)
        print(self.df_dx); print(self.d2f_dx);
    def show_chart(self):
        table=self.get_object("treeview1")
        model=table.get_model()
        self.tempfile=open(self.tempfile.name,'wb')
        for r in model:
            self.tempfile.write("{0} {1} {2}\n".format(r[0],r[1],
                                        self.f(r[0],r[1])).encode())
        self.tempfile.close()
        if self.plot and self.plot.poll()==None:
            self.plot.stdin.close()
            self.plot.terminate()
            if self.plot.poll()==None:
                self.plot.kill()
                self.plot.wait()
        msg=('set pm3d;splot '+self.f.expr+',"'+self.tempfile.name+
            '" title "Рассчитано" with linespoints;pause -1')
        self.plot=subp.Popen(['gnuplot','-e',msg],bufsize=4000,shell=False,
                    stdin=subp.PIPE);

    def chart_clicked(self,widget,data=None):
        self.show_chart()
    def input_data(self):
        x=float(self.get_object("entry2").get_text());
        y=float(self.get_object("entry3").get_text())
        maxiter=int(self.get_object("entry6").get_text())
        eps=float(self.get_object("entry7").get_text())
        return x,y,maxiter,eps,self.f,self.df_dx,self.d2f_dx
    def output_data(self,res):
        table=self.get_object("treeview1")
        model=table.get_model()
        model.clear()
        for r in res:
            model.append([r[0],r[1],self.f(r[0],r[1])])

    def on_run_clicked(self,button,data=None):
        x,y,maxiter,eps,f,df_dx,d2f_dx=self.input_data()
        res=gradient(maxiter,eps,(x,y),f,df_dx,d2f_dx)
        self.output_data(res)
        self.show_chart()


app=Application("prac4.ui")
app.show("window1")
```