Федеральное агентство по образованию РФ

Государственное образовательное учреждение высшего профессионального образования

Белгородский Государственный Технологический Университет

им. В.Г. Шухова

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №7**

**по дисциплине «Вычислительная математика»**

# *Минимизация функции многих переменных методом градиента с дроблением шага*

Выполнил: студент группы ПВ-21

Браткова И.О.

Проверил: Бондаренко Т.В.

Белгород 2017г.

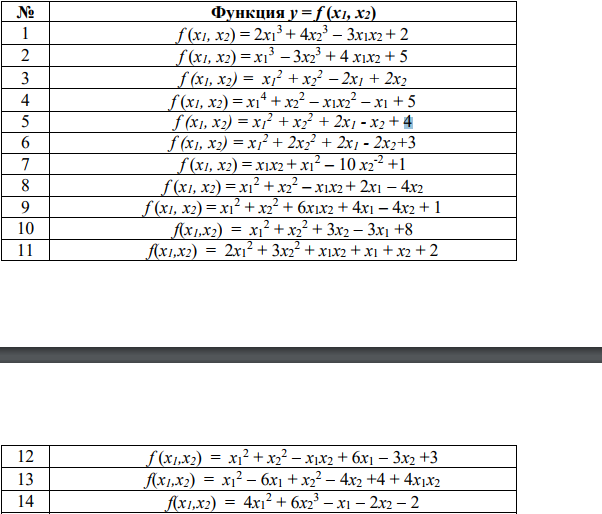
**Цель работы**: изучить метод градиента с дроблением шага для решения задачи минимизации функции многих переменных и получить практические навыки его применения.

**Задания к работе:**

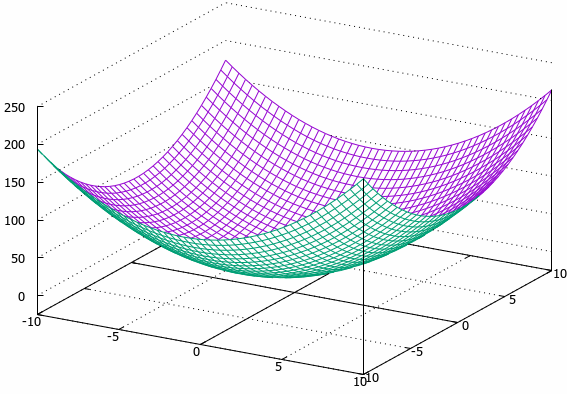
* 1. Найти точные значения координат точки минимума и минимальное значение функции *у* = *f*(*x1, х2*) для функции соответствующего варианта задания, используя необходимые и достаточные условия локального минимума.
  2. Выполнить вручную один шаг метода градиента с дроблением шага, начиная с произвольно выбранного начального приближения *М*0 (*x10, x20*)*.*
  3. Реализовать логическую функцию для нахождения точки локального минимума и минимального значения целевой функции.

Входными данными для логической функции являются:

* целевая функция *у* = *f*(*x1, х2)*;
* градиент целевой функции *grad*(*f*(*x1,х2*));
* начальное приближение *М*0(*x10, x20*) к точке локального минимума;
* точность решения *ɛ*;
* ограничение на максимальное число итераций *n*;
* параметры метода градиента с дроблением шага *α*, *β*, *γ*.

Функция возвращает значение «истина», если приближенное решение с заданной точностью получено за число итераций, не превышающее*n*, и «ложь» − в противном случае. 

Нарисуем график данной функции: x1 = x, x2 = y.



Выполнение 1-2 задания см. тетрадь 2.

#include <stdio.h> #include <stdlib.h> #include <math.h>

*//Описание типов*

typedef double tochka[2]; *//точка*

typedef double(\*t\_f)(tochka); *//функциональный тип*

typedef t\_f t\_grad[2]; *//массив, где элементы - функции высчитыания составляющих градиента*

double U(tochka x) *//функция варианта*

{ return (x[0]\*x[0]+x[1]\*x[1]+2\*x[0]-x[1]+4); }

double dUx1(tochka x) *//производная по x1*

{ return 2\*x[0]+2; }

double dUx2(tochka x*)//производная по x2*

{ return 2\*x[1]-1; }

int gradient(t\_f U, t\_grad G, tochka\* M, double al, double be, double om, double eps, int n)

*/\*t\_f - функция U = f(x1;x2), G = градиент, массив из двух функций для подсчета,*

*M - начальная точка, al, be, om = альфа, бета, омега*

*eps = точность, кол-во швгов*

*Функция возвращает значение «истина», если приближенное решение с заданной точностью*

*получено за число итераций, не превышающее n, и «ложь» в противном случае. \*/*

{

double omi = om; *//для возведения в степерь омеги перезапишем ее*

tochka gr; *//массив из двух элементов в котором будет храниться значение градиента*

gr[0] = (G[0])(\*M); *//записываем в grad1[0] значение производной в точке М по x1*

gr[1] = (G[1])(\*M); *//записываем в grad1[1] значение производной в точке М по x2*

tochka x0*;//сейчас ¬х0 = М (x1, x2)*

x0[0] = (\*M)[0]; x0[1] = (\*M)[1];

tochka x1; *//¬x1 = ¬x0 - akpga\*gradU(M)*

x1[0] = x0[0] - al\*gr[0]; x1[1] = x0[1] - al\*gr[1];

tochka algrad; *//при умножении в цикле данная величина не меняется и ее можно вынести*

algrad[0] = al\*gr[0]; algrad[1] = al\*gr[1];

*//высчитаем для сравнения правую часть и будем домножать ее в цикле на om, тюе увеличивать*

*//double ri = omi\*al\*(-be)\*(G[0](\*M) \* G[0](\*M) + G[1](\*M) \* G[1](\*M));*

double ri = -al\*be\*omi\*(pow(gr[0], 2) + pow(gr[1], 2));

double Ux0 = U(x0); *// значение функции необходимо для сравнения*

FILE \*F = fopen("Rez.txt", "w"); *//открытие файла для записи результатов*

int i=n, flag = 0; *//счетчик и флаг для выхода из цикла. Когда он = 1, решение получено*

while (i>0 && flag == 0)

{

//высчитываем две новые точки

x1[0] = x0[0] - omi \* algrad[0]; fprintf(F, "(%.3f; ", x1[0]);

x1[1] = x0[1] - omi \* algrad[0]; fprintf(F, "%.3f) \n", x1[1]);

flag = (U(x1) - Ux0) < ri; //пересчитываем значение флага в новых точках

ri \*= om; //величиваем правую часть неравенства на омега

omi \*= om; //возводим омега в следующую степень

i--;

}

fclose(F);

(\*M)[0] = x1[0]; //перезаписываем полученные точки

(\*M)[1] = x1[1];

flag = (abs(U(x0) - U(\*M))<eps); //проверяем флаг на соответствие точности

return (flag && (i <= n)); //возвращаем 1, если точность достигнута за число шагов меньше n

}

void main()

{

t\_grad G;

tochka M0;

M0[0] = -0.8;

M0[1] = 0.4;

G[0] = dUx1;

G[1] = dUx2;

int n = 5;

printf("True OR False >> %d\n", gradient(U, G, &M0, 0.5, 0.5, 0.9, 0.001, n));

printf("M (%.2f; %.2f)\n", M0[0], M0[1]);

}

Результат выполнения и часть файла:

