**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Дисциплина: Вычислительная математика

Лабораторная работа № 7

Тема: «Минимизация функции многих переменных методом градиента с дроблением шага»

Выполнила:

Студент группы ВТ-21

Сидорова Ангелина Сергеевна

Проверила: Бондаренко Т. В.

Белгород 2018

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 вычисление приближенного значения точки минимума целевой функции методом градиента с дроблением шага, начиная с произвольно выбранного начального приближения М0 (x10, x20). Точность решения ε = 0,01. Вручную подробно достаточно выполнить первый шаг метода градиента с дроблением шага. Параметры метода выбрать самостоятельно.

3. Реализовать логическую функцию для нахождения приближенного значения точки локального минимума и минимального значения целевой функции у = f(x1, х2) методом градиента с дроблением шага. Входными данными для логической функции являются:

− целевая функция у = f (x1, х2);

− градиент целевой функции grad (f (x1, х2));

− начальное приближение М0(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.

Результат работы программы: приближенное значение точки локального минимума и минимальное значение целевой функции у = f (x1, х2) с заданной точностью, количество выполненных итераций.

Предусмотреть возможность сохранения пошаговых результатов реализации метода градиента с дроблением шага в файл.

Задание 3

vm.h

#ifndef VM\_H\_INCLUDED

#define VM\_H\_INCLUDED

typedef struct mnog{

float k;

int n;

float x[20];

int p[20];

} mnog;

float func(mnog \*F, float \*M, int n);// возвращает значение функции F в точке M

int MethodGrad(mnog \*F, float \*M, int n, int k, float e, int N);

#endif // VM\_H\_INCLUDED

vm.c

#include <stdio.h>

#include <math.h>

#include "vm.h"

float func(mnog \*F, float \*M, int n)

{

float s = 0;

for (int i = 0; i < n; i++)

{

float s1 = 0;

for (int j = 0; j < F[i].n; j++)

{

int x = F[i].x[j];

s1 += pow(M[x], F[i].p[j]);

}

if (F[i].n == 0)

s += F[i].k;

s += F[i].k \* s1;

}

return s;

}

void copy\_mnog(mnog \*W, mnog \*F, int n)// копирует многочлен W в F

{

for (int i = 0; i < n; i++)

{

F[i].k = W[i].k;

F[i].n = W[i].n;

for (int j = 0; j < F[i].n; j++)

{

F[i].x[j] = W[i].x[j];

F[i].p[j] = W[i].p[j];

}

}

}

float dif(mnog \*W, float \*M, int n, int t)// дифференцирует многочлен W  
и возвращает его значение в точке M

{

mnog \*F = calloc(n,sizeof(mnog));

copy\_mnog(W,F,n);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < F[i].n; j++)

if (F[i].x[j] == t && F[i].p[j] > 0)

{

F[i].k = F[i].k \* F[i].p[j];

F[i].p[j]--;

break;

}

else

F[i].k = 0;

}

float s = 0;

for (int i = 0; i < n; i++)

{

float s1 = 0;

for (int j = 0; j < F[i].n; j++)

{

int x = F[i].x[j];

s1 += pow(M[x], F[i].p[j]);

}

s += F[i].k \* s1;

}

return s;

}

int MethodGrad(mnog \*F, float \*M1, int n, int k, float e, int N)// присваивает M1 значения\_глобального\_минимума\_функции\_F возвращает 1, если удалось за N шагов достичь точность e, иначе 0

{

float a = 1, b = 0.5, c = 0.5, p = 0;

float \*M0 = calloc(k,sizeof(float));

float (\*grad)(mnog \*F, float \*M, int n, int t) = dif;

int kol = 0;

do

{

for (int j = 0; j < k; j++)

M0[j] = M1[j];

int i = 0, f = 0;

p = 0;

for (int j = 0; j < k; j++)

p += grad(F,M0,n,j)\*grad(F,M0,n,j);

while (!f)

{

float ci = pow(c,i);

for (int j = 0; j < k; j++)

M1[j] = M0[j] - a\*ci\*grad(F,M0,n,j);

f = func(F,M1,n) - func(F,M0,n) < - b\*a\*ci\*p;

i++;

}

kol++;

if (kol > N)

return 0;

}

while (kol < N && fabs(func(F,M0,n)-func(F,M1,n)) > e);

return 1;

}

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include "vm.h"

void Init(mnog \*F, float \*M)

{

M[0] = 0; M[1] = 0;

F[0].k = 1; F[1].k = 1; F[2].k = 3; F[3].k = -3; F[4].k = 8;

F[0].x[0] = 0; F[1].x[0] = 1; F[2].x[0] = 0; F[3].x[0] = 1; F[4].x[0] = 0;

F[0].p[0] = 2; F[1].p[0] = 2; F[2].p[0] = 1; F[3].p[0] = 1; F[4].p[0] = 0;

F[0].n = 1; F[1].n = 1; F[2].n = 1; F[3].n = 1; F[4].n = 0;

}

main.c

int main()

{

int k = 2, n = 5, N = 5;

float e = 0.01;

float \*M = calloc(k,sizeof(float));

mnog \*F = calloc(n,sizeof(mnog));

Init(F,M);

int check = MethodGrad(F,M,n,k,e,N);

if (check)

{

printf("Reshenie :\nM = (");

for (int i = 0; i < k - 1; i++)

printf("%f; ",M[i]);

printf("%f)\n",M[k-1]);

printf("F(M) = %f",func(F,M,n));

}

else

printf("Za %i shagov ne udalos` dostich` zadannoy` tochnosti.",N);

free(M);

free(F);

}

r1jAKMcYPyM.jpg