**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО**

**Московский авиационный институт**

**(национальный исследовательский университет)**

Кафедра 304

Вычислительные машины, системы и сети

**Отчет по лабораторной работе №1**

**по учебной дисциплине «Математическое программирование»**

**на тему**

***«Исследование сходимости численных методов задач безусловного экстремума»***

Выполнил студент группы М3О-310Б-21:

Пысларь А. И.

Принял:

Григоревский Н.В.

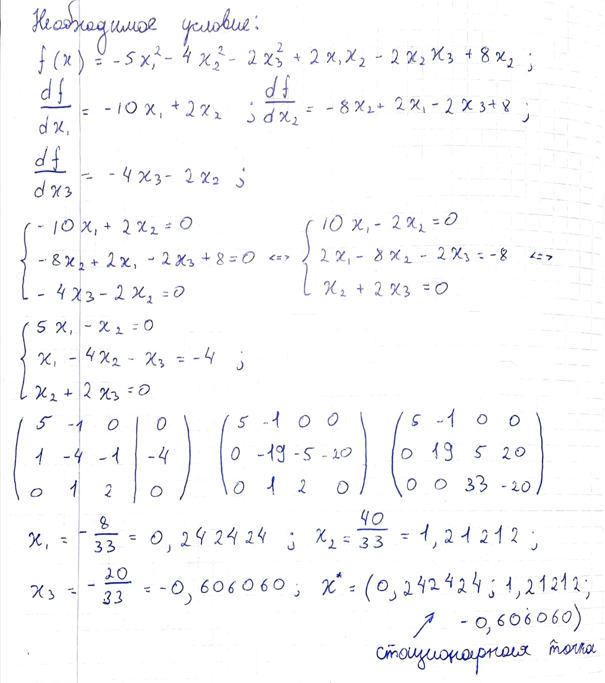
Москва 2023

**Задание**

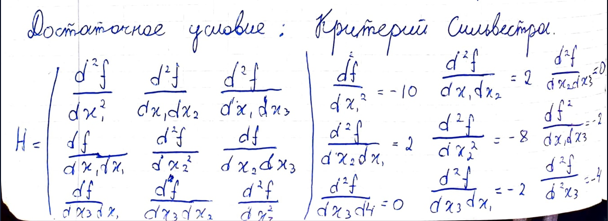
1. Найти экстремум функции с помощью необходимых и достаточных условий
2. Градиентный спуск с дроблением шага;
3. Метод наискорейшего спуска;
4. Покоординатный спуск;
5. Метод Ньютона
6. Метод Флетчера-Ривса;

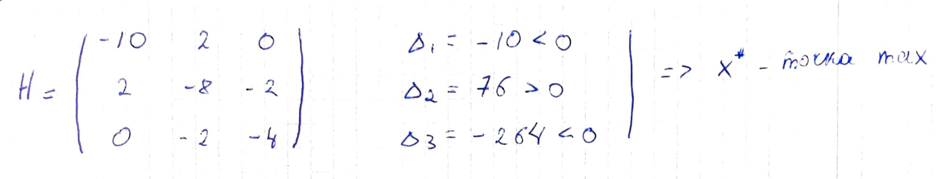


**Необходимое условие**

****

**Достаточное условие**

****

****

**Вспомогательные функции**

//Функция

double function(double x, double y, double z)

{

return (-3 \* x\*x - 2 \* y\*y - 2 \* z\*z + x \* y + y \* z - x \* z + 5 \* x);

}

//Подсчет частной производной по x

double derivativex(double x, double y, double z)

{

int h = 1;

return ((function(x + h, y, z) - function(x - h, y, z)) / 2 \* h);

}

//Подсчет частной производной по y

double derivativey(double x, double y, double z)

{

int h = 1;

return ((function(x, y + h, z) - function(x, y - h, z)) / 2 \* h);

}

//Подсчет частной производной по z

double derivativez(double x, double y, double z)

{

int h = 1;

return ((function(x, y, z + h) - function(x, y, z - h)) / 2 \* h);

}

//Подсчет градиента функции

double grad(double x, double y, double z)

{

double h = 1; //Шаг для вычисления частной производной

double derivativex = ((function(x + h, y, z) - function(x - h, y, z)) / 2 \* h);

double derivativey = ((function(x, y + h, z) - function(x, y - h, z)) / 2 \* h);

double derivativez = ((function(x, y, z + h) - function(x, y, z - h)) / 2 \* h);

return (sqrt(derivativex\*derivativex + derivativey \* derivativey + derivativez \* derivativez));

}

**Основная программа**

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

double x = 0;

double y = 0;

double z = 0;

int choice = 0;

cout << "Введите начальные значения\nx1 = ";

cin >> x;

cout << "x2 = ";

cin >> y;

cout << "x3 = ";

cin >> z;

cout << "Введите номер метода: \n1) Метод градиентного спуска с дроблением шага;\n";

cout << "2) Метод наискорейшего спуска;\n";

cout << "3) Метод покординатного спуска;\n";

cout << "4) Метод Флетчера-Ривса;\n";

cin >> choice;

switch (choice)

{

case 1:

{

//Метод градиентного спуска с дроблением шага

GradStepSplit(x, y, z);

break;

}

case 2:

{

//Метод наискорейшего спуска

FastStep(x, y, z);

break;

}

case 3:

{

//Метод покоординатного спуска

Coordescent(x, y, z);

break;

}

case 4:

{

//Метод Флетчера-Ривса

FletRivs(x, y, z);

break;

}

}

system("pause");

return 0;

}

**Метод градиентного спуска с дроблением шага**

**Программа**

//Метод градиентного спуска с дроблением шага

void GradStepSplit(double currentx, double currenty, double currentz)

{

double nextx = currentx;

double nexty = currenty;

double nextz = currentz;

double step = 0.1; //шаг

int count = 0; //счетчик циклов

double eps = 0.001;

double f = function(currentx, currenty, currentz);

double fg = 0;

cout << "x1 " << setw(18) << "x2" << setw(18) << "x3" << setw(20) << "f(x1,x2,x3)" << setw(12) << "grad" << endl;

cout << currentx << setw(16) << currenty << setw(16) << currentz << setw(16) << f << setw(16) << grad(currentx, currenty, currentz) << endl;

do

{

//переход на новую итерацию

currentx = nextx;

currenty = nexty;

currentz = nextz;

//Текущее значение функции

f = function(currentx, currenty, currentz);

//Подсчитываем новые значения аргументов

nextx = currentx + step \* derivativex(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

nexty = currenty + step \* derivativey(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

nextz = currentz + step \* derivativez(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

//Подсчитываем новое значение функции

fg = function(nextx, nexty, nextz);

if (fg <= f)

{

//Дробим шаг

step = step \* 0.5;

}

count++;

cout << nextx << setw(16) << nexty << setw(16) << nextz << setw(16) << fg << setw(16) << grad(nextx, nexty, nextz) << endl;

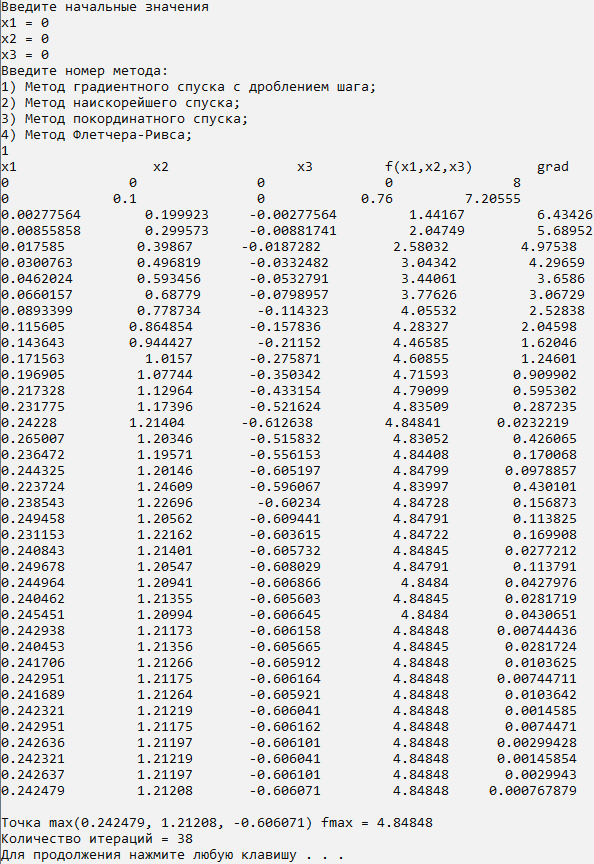
} while (grad(nextx, nexty, nextz) > eps);

cout << "\nТочка max(" << nextx << ", " << nexty << ", " << nextz << ") " << "fmax = " << fg << endl;

cout << "Количество итераций = " << count << endl;

}

**Работа программы**



**Метод наискорейшего спуска**

**Программа**

//подсчет шага

double minfunction(double x, double y, double z)

{

double chisl = 0;

double znam = 0;

//Частные производные функции

double s\_x = derivativex(x, y, z) / grad(x, y, z);

double s\_y = derivativey(x, y, z) / grad(x, y, z);

double s\_z = derivativez(x, y, z) / grad(x, y, z);

chisl = -(-10 \* x \* s\_x - 8 \* y \* s\_y - 4 \* z \* s\_z + 2 \* y \* s\_x + 2 \* x \* s\_y - 2 \* y \* s\_z - 2 \* z \* s\_y + 8 \* s\_y);

znam = -10 \* s\_x \* s\_x - 8 \* s\_y \* s\_y - 4 \* s\_z \* s\_z + 4 \* s\_x \* s\_y - 4 \* s\_y \* s\_z;

return (chisl / znam);

}

//Метод наискорейшего спуска

void FastStep(double currentx, double currenty, double currentz)

{

double nextx = currentx;

double nexty = currenty;

double nextz = currentz;

int count = 0; //счетчик циклов

double eps = 0.001;

double step = 0.1;

double f = function(currentx, currenty, currentz);

double fg = f;

cout << "x1 " << setw(18) << "x2" << setw(18) << "x3" << setw(20) << "f(x1,x2,x3)" << setw(12) << "grad" << endl;

cout << currentx << setw(16) << currenty << setw(16) << currentz << setw(16) << f << setw(16) << grad(currentx, currenty, currentz) << endl;

do

{

//Переход на новую итерацию

currentx = nextx;

currenty = nexty;

currentz = nextz;

//Подсчет нового шага

step = minfunction(currentx, currenty, currentz);

//Подсчитываем новые значения аргументов

nextx = currentx + step \* derivativex(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

nexty = currenty + step \* derivativey(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

nextz = currentz + step \* derivativez(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

count++;

//Подсчитываем новое значение функции

fg = function(nextx, nexty, nextz);

cout << nextx << setw(16) << nexty << setw(16) << nextz << setw(16) << fg << setw(16) << grad(nextx, nexty, nextz) << endl;

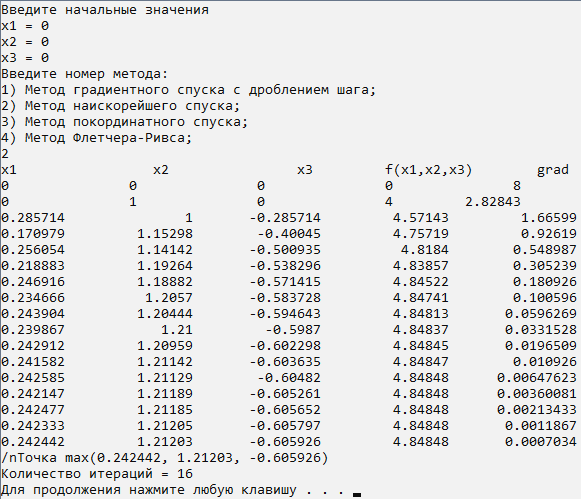
} while (grad(nextx, nexty, nextz) > eps);

cout << "/nТочка max(" << nextx << ", " << nexty << ", " << nextz << ")" << endl;

cout << "Количество итераций = " << count << endl;

}

**Работа программы**

****

**Метод покоординатного спуска**

**Программа**

//Метод покоординатного спуска

void Coordescent(double currentx, double currenty, double currentz)

{

double nextx = currentx;

double nexty = currenty;

double nextz = currentz;

int count = 0; //счетчик циклов

double eps = 0.001;

double step = 0.1; //шаг

double f = function(currentx, currenty, currentz);

double fg = f;

cout << "x1 " << setw(18) << "x2" << setw(18) << "x3" << setw(20) << "f(x1,x2,x3)" << setw(12) << "grad" << endl;

cout << currentx << setw(16) << currenty << setw(16) << currentz << setw(16) << f << setw(16) << grad(currentx, currenty, currentz) << endl;

do

{

//Переход на новую итерацию

currentx = nextx;

currenty = nexty;

currentz = nextz;

//Подсчет первого аргумента и его шага

nextx = currentx + step \* derivativex(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

step = minfunction(currentx, currenty, currentz);

//Подсчет второго аргумента и его шага

nexty = currenty + step \* derivativey(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

step = minfunction(currentx, currenty, currentz);

//Подсчет третьего аргумента и его шага

nextz = currentz + step \* derivativez(currentx, currenty, currentz) / grad(currentx, currenty, currentz);

step = minfunction(currentx, currenty, currentz);

count++;

//Подсчитываем новое значение функции

fg = function(nextx, nexty, nextz);

cout << nextx << setw(16) << nexty << setw(16) << nextz << setw(16) << fg << setw(16) << grad(nextx, nexty, nextz) << endl;

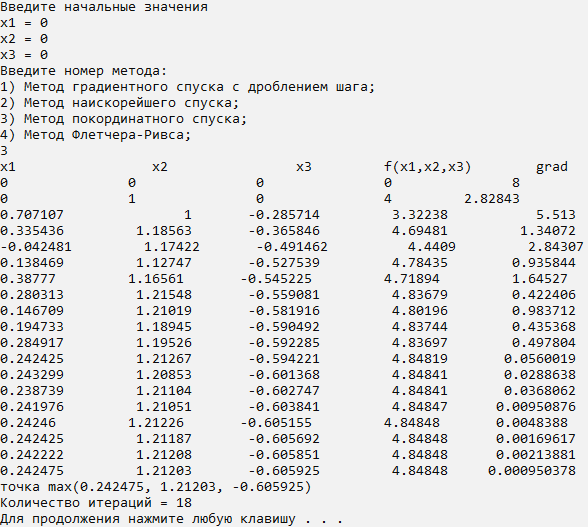
} while (grad(nextx, nexty, nextz) > eps);

cout << "точка max(" << nextx << ", " << nexty << ", " << nextz << ")" << endl;

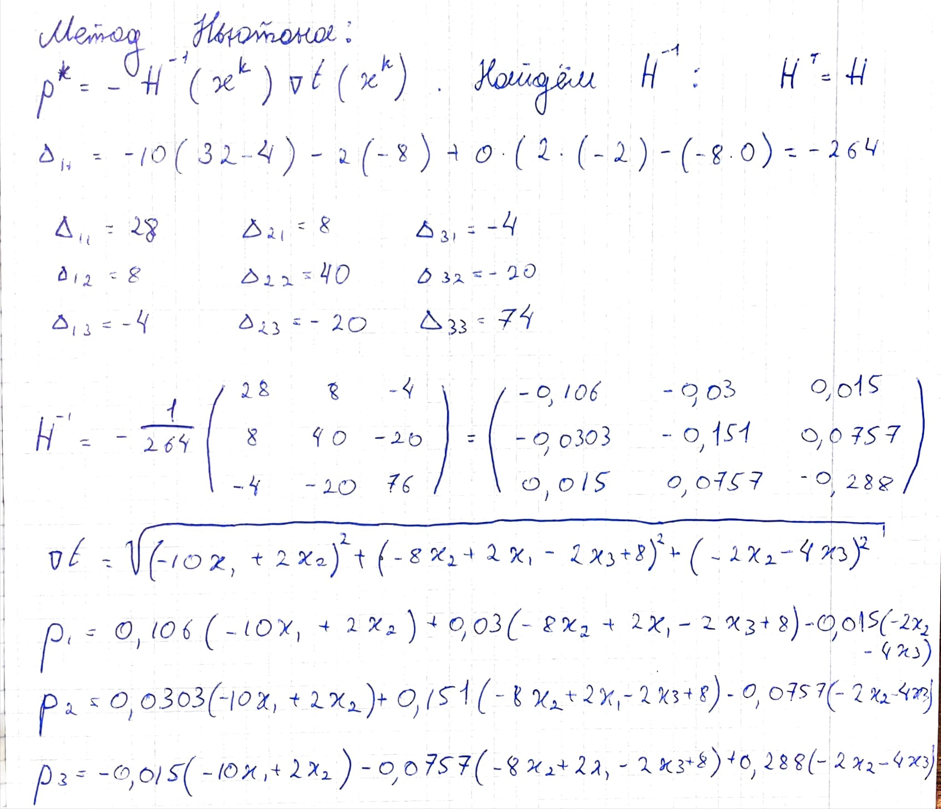
cout << "Количество итераций = " << count << endl;

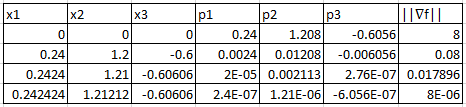
}

**Работа программы**



**Метод Ньютона**

****

****

**||∇f|| < ε**

**Т. об. max (**0.242424, 1.212121, -0.60606)

**Метод Флетчера-Ривса**

**Программа**

//подсчет шага

double lastminfunction(double x, double y, double z, double s\_x, double s\_y, double s\_z)

{

double chisl = 0;

double znam = 0;

chisl = -(-10 \* x \* s\_x - 8 \* y \* s\_y - 4 \* z \* s\_z + 2 \* y \* s\_x + 2 \* x \* s\_y - 2 \* y \* s\_z - 2 \* z \* s\_y + 8 \* s\_y);

znam = -10 \* s\_x \* s\_x - 8 \* s\_y \* s\_y - 4 \* s\_z \* s\_z + 4 \* s\_x \* s\_y - 4 \* s\_y \* s\_z;

return (chisl / znam);

}

//Метод Флетчера-Ривса

void FletRivs(double currentx, double currenty, double currentz)

{

//начальная установка

double nextx = currentx;

double nexty = currenty;

double nextz = currentz;

double beforegrad = grad(currentx, currenty, currentz);

double step = 0; //шаг

int count = 0; //количество итераций

double eps = 0.001; //точность

double b = 0;

//элементы вектора P

double p\_x = 0;

double p\_y = 0;

double p\_z = 0;

double beforep\_x = 0;

double beforep\_y = 0;

double beforep\_z = 0;

double f = function(currentx, currenty, currentz);

cout << "x1 " << setw(18) << "x2" << setw(18) << "x3" << setw(20) << "f(x1,x2,x3)" << setw(14) << "grad" << endl;

cout << currentx << setw(23) << currenty << setw(16) << currentz << setw(16) << f << setw(16) << grad(currentx, currenty, currentz) << endl;

while (grad(nextx, nexty, nextz) > eps)

{

//Переход на новую итерацию

currentx = nextx;

currenty = nexty;

currentz = nextz;

b = pow(grad(nextx, nexty, nextz), 2) / pow(beforegrad, 2);

//Обновление старого градиента

beforegrad = grad(currentx, currenty, currentz);

//Подсчет коэффициента p

p\_x = derivativex(nextx, nexty, nextz) + b \* beforep\_x;

p\_y = derivativey(nextx, nexty, nextz) + b \* beforep\_y;

p\_z = derivativez(nextx, nexty, nextz) + b \* beforep\_z;

//сохрание значений p

beforep\_x = p\_x;

beforep\_y = p\_y;

beforep\_z = p\_z;

//Получение значения шага

step = lastminfunction(nextx, nexty, nextz, p\_x, p\_y, p\_z);

//Подсчитываем новые значения аргументов

nextx = currentx + step \* p\_x;

nexty = currenty + step \* p\_y;

nextz = currentz + step \* p\_z;

count++;

f = function(nextx, nexty, nextz);

cout << nextx << setw(16) << nexty << setw(16) << nextz << setw(16) << f << setw(16) << grad(nextx, nexty, nextz) << endl;

}

cout << "/nТочка max(" << nextx << ", " << nexty << ", " << nextz << ")" << endl;

cout << "Количество итераций = " << count << endl;

}

**Работа программы**

