МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Методы поиска безусловного экстремума

Отчет по лабораторной работе №2

02.03.02 Фундаментальная информатика   
и информационные технологии

Выполнил: студент 3 курса \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Решетников Р.С.

Проверил: к. б. н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чупраков П. Г.

Киров

2020 г.

**Цель лабораторной работы:**

Научится находить минимум произвольной функции методами поиска безусловного экстремума.

**Задание:**

1. Найти локальные минимумы трех функций с использованием следующих методов: метод нулевого порядка (любой) и метод ненулевого порядка (по вариантам, см. таблицу 2).

2. Дать геометрическую интерпретацию решения для двумерных функций.

3. Проверить решения с помощью математических пакетов

Функции:

а) 𝑓(𝑥1,𝑥2)= 4,5\*x12-4,5\*x1-3\*x1\*x2+2\*x22-3\*x2-4

Начальная точка 𝑥0=(2,4)

б) 𝑓(𝑥1,𝑥2,𝑥3)=(𝑥1−9)2+4(𝑥2−4)2+2(𝑥3−5)2

Начальная точка 𝑥0=(10,4,7)

в) Функция Вуда

𝑓(𝑥1,𝑥2,𝑥3,𝑥4)= 100\*(x12-x2)2+(x1-1)2+(x3-1)2+90(x32-x4)2+10.1((x1-1)2+(x4-1)2)+19.8(x2-1)\*(x4-1)

Начальная точка 𝑥0=(0,0,0,0)

**Краткое описание алгоритмов**

1. Метод конфигурации представляет собой комбинацию исследующего поиска с циклическим изменением переменных и ускоряющего поиска по образцу. Выбирается шаг и для каждой переменной выбирается в большую или меньшую сторону она будет изменятся в зависимости от поведения функции. Величина шага может быть различной для разных направлений в процессе поиска. Поиск заканчивается, когда изменение значения функции меньше введенной погрешности.
2. Метод Ньютона состоит в построении ряда точек, таких, что значение функции становится меньше с каждой следующей точкой. Изменение значений переменной разное для направлений и на каждом шаге, и зависит от градиента и гессиана. Поиск заканчивается когда изменение функции и переменных становится слишком мало, либо когда градиент становится равным нулю, то есть функция достигла экстремума. Также алгоритм прерывается когда количество итераций превышает введённый лимит.

**Листинг программы с комментариями**

Класс для работы с функциями:

class Thesefunctions

{

public Function function;

public void SetFunction(Function function)

{

this.function = function;

}

public double Func()

{

return function.Func();

}

public double GetX(int i)

{

return function.GetX(i);

}

public void SetX(int i,double x)

{

function.SetX(i, x);

}

public double GetH()

{

return function.GetH();

}

public int GetN()

{

return function.GetN();

}

public Matrix matF()

{

return function.matF();

}

}

Функции вычисления:

static void Method0(Thesefunctions fun, double e, double y)

{

int n = fun.GetN();

int k = 0;

k++;

double f0 = fun.Func();

while (y > e)

{

double f1 = fun.Func(), f2 = 0, f3 = 0;

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

{

k++;

fun.SetX(i, fun.GetX(i) + y);

f2 = fun.Func();

if (f2 > f1)

{

k++;

fun.SetX(i, fun.GetX(i) - y \* 2);

f3 = fun.Func();

if (f3 > f1 || f2 == f3)

{

fun.SetX(i, fun.GetX(i) + y);

}

}

k++;

}

if (f0 == f1)

y = y / 2;

f0 = f1;

}

string s = "Экстремум: " + Math.Round(fun.Func(), (int)Math.Round(Math.Log(e, 0.1))) + "в точке [" + Math.Round(fun.GetX(0), (int)Math.Round(Math.Log(e, 0.1)));

for (int i = 1; i < n; i++)

s += " ; " + Math.Round(fun.GetX(i), (int)Math.Round(Math.Log(e, 0.1)));

s += "] Кол-во итераций: " + k;

Console.WriteLine(s);

}

static void Method1(Thesefunctions fun, double e1, double e2, int M)

{

int k = 0, l = 0, n = fun.GetN();

double d, f0, f1, sum = 0;

Matrix matf = fun.matF();

bool b = false, b1 = true;

double det = fun.GetH();

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

sum += Math.Abs(matf[i, 0]);

while (Math.Abs(sum) > e1 && k < M && b1)

{

sum = 0;

f0 = fun.Func();

if (det > 0)

{

Matrix H = fun.function.GetMatrix();

H = H.CreateInvertibleMatrix();

H = H \* matf;

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

fun.SetX(i, fun.GetX(i) - H[i, 0]);

}

else

{

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

fun.SetX(i, fun.GetX(i) - matf[i, 0]);

f1 = fun.Func();

l++;

if (f0 < f1)

{

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

fun.SetX(i, fun.GetX(i) + 2 \* matf[i, 0]);

}

}

f1 = fun.Func();

l++;

if (Math.Abs(f1 - f0) < e2 && Math.Abs(sum) < e2)

if (b)

b1 = false;

else

b = true;

k++;

matf = fun.matF();

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

sum += matf[i, 0];

det = fun.GetH();

}

string s = "Экстремум: " + Math.Round(fun.Func(), (int)Math.Round(Math.Log(e2, 0.1))) + "в точке [" + Math.Round(fun.GetX(0), (int)Math.Round(Math.Log(e2, 0.1)));

for (int i = 1; i < n; i++)

s += " ; " + Math.Round(fun.GetX(i), (int)Math.Round(Math.Log(e2, 0.1)));

s += "] Кол-во итераций: " + l;

Console.WriteLine(s);

}

Основная программа:

static void Main(string[] args)

{

//погрешность 0,001

Console.WriteLine("Метод конфигурации: ");//шаг 2

double[] a1 = { 2, 4 };

Thesefunctions funct1 = new Thesefunctions();

funct1.SetFunction(new Func1(a1));

Method0(funct1, 0.001, 2);

double[] a2 = { 10, 4, 7 };

Thesefunctions funct2 = new Thesefunctions();

funct1.SetFunction(new Func2(a2));

Method0(funct2, 0.001, 2);

double[] a3 = { 0, 0, 0, 0 };

Thesefunctions funct3 = new Thesefunctions();

funct3.SetFunction(new Func3(a3));

Method0(funct3, 0.001, 2);

Console.WriteLine("Метод Нютона: ");//ограничение 10000

funct1 = new Thesefunctions();

double[] b1 = { 2, 4 };

funct1.SetFunction(new Func1(b1));

Method1(funct1, 0.001, 0.001, 10000);

funct2 = new Thesefunctions();

double[] b2 = { 10, 4, 7 };

funct2.SetFunction(new Func2(b22));

Method1(funct2, 0.001, 0.001, 10000);

funct3 = new Thesefunctions();

double[] b3 = { 0, 0, 0, 0 };

funct3.SetFunction(new Func3(b3));

Method1(funct3, 0.001, 0.001, 10000);

}

Функция 1:

public class Func1 : Function

{

public double[] x;

public Matrix H;

public int N;

public int GetN()

{

return N;

}

public double GetX(int i)

{

return x[i];

}

public void SetX(int i, double x)

{

this.x[i] = x;

}

public Func1(double[] x)

{

this.x = x;

N = x.Length;

H = new Matrix(N, N);

BuildH();

}

public double Func()

{

return 4.5 \* x[0] \* x[0] - 4.5 \* x[0] - 3 \* x[0] \* x[1] + 2 \* x[1] \* x[1] - 3 \* x[1] - 4;

}

public double GetH()

{

BuildH();

return H.CalculateDeterminant();

}

void BuildH()

{

H = new Matrix(N, N);

H[0, 0] = 9;

H[0, 1] = -3;

H[1, 1] = 4;

H[1, 0] = -3;

}

public Matrix ChF()

{

Matrix y = new Matrix(N, 1);

y[0, 0] = 9 \* x[0] - 3 \* x[1] - 4.5;

y[1, 0] = -3 \* x[0] + 4 \* x[1] - 3;

return y;

}

public Matrix GetMatrix()

{

BuildH();

return H;

}

}

Функция 2:

public class Func2 : Function

{

public double[] x;

public Matrix H;

int N;

public int GetN()

{

return N;

}

public double GetX(int i)

{

return x[i];

}

public void SetX(int i, double x)

{

this.x[i] = x;

}

public Func2(double[] x)

{

this.x = x;

N = x.Length;

H = new Matrix(N, N);

BuildH();

}

public double Func()

{

return (x[0] - 9) \* (x[0] - 9) + 4 \* (x[1] - 4) \* (x[1] - 4) + 2 \* (x[2] - 5) \* (x[2] - 5);

}

public double GetH()

{

BuildH();

return H.CalculateDeterminant();

}

void BuildH()

{

H = new Matrix(N, N);

H[0, 0] = 2;

H[0, 1] = 0;

H[0, 2] = 0;

H[1, 1] = 8;

H[1, 0] = 0;

H[1, 2] = 0;

H[2, 0] = 0;

H[2, 1] = 0;

H[2, 2] = 4;

}

public Matrix ChF()

{

Matrix y = new Matrix(N, 1);

y[0, 0] = 2 \* (x[0] - 9);

y[1, 0] = 8 \* (x[1] - 4);

y[2, 0] = 4 \* (x[2] - 5);

return y;

}

public Matrix GetMatrix()

{

BuildH();

return H;

}

}

Функция 3:

public class Func3 : Function

{

public double[] x;

public Matrix H;

int N;

public int GetN()

{

return N;

}

public double GetX(int i)

{

return x[i];

}

public void SetX(int i, double x)

{

this.x[i] = x;

}

public Func3(double[] x)

{

this.x = x;

N = x.Length;

H = new Matrix(N, N);

BuildH();

}

public double Func()

{

return 100 \* Math.Pow(x[0] \* x[0] - x[1], 2) + Math.Pow(x[0] - 1, 2) + Math.Pow(x[2] - 1, 2) + 90 \* Math.Pow(x[2] \* x[2] - x[3], 2) + 10.1 \* (Math.Pow(x[1] - 1, 2) + Math.Pow(x[3] - 1, 2)) + 19.8 \* (x[1] - 1) \* (x[3] - 1);

}

public double GetH()

{

BuildH();

return H.CalculateDeterminant();

}

void BuildH()

{

H[0, 0] = 1200 \* x[0] \* x[0] - 400 \* x[1] + 2;

H[0, 1] = -400 \* x[0];

H[0, 2] = 0;

H[0, 3] = 0;

H[1, 1] = 220.2;

H[1, 0] = -400 \* x[0];

H[1, 2] = 0;

H[1, 3] = 19.8;

H[2, 0] = 0;

H[2, 1] = 0;

H[2, 2] = 1080 \* x[2] \* x[2] - 360 \* x[3] + 2;

H[2, 3] = -360 \* x[2];

H[3, 0] = 0;

H[3, 1] = 19.8;

H[3, 2] = -360 \* x[2];

H[3, 3] = 200.2;

}

public Matrix ChF()

{

Matrix y = new Matrix(N, 1);

y[0, 0] = 2 \* (200 \* x[0] \* (x[0] \* x[0] - x[1]) + x[0] - 1);

y[1, 0] = -200 \* (x[0] \* x[0] - 1.101 \* x[1] - 0.099 \* x[3] + 0.2);

y[2, 0] = 2 \* (180 \* x[2] \* (x[2] \* x[2] - x[3]) + x[2] - 1);

y[3, 0] = -180 \* x[2] \* x[2] + 19.8 \* x[1] + 200.2 \* x[3] - 40;

return y;

}

public Matrix GetMatrix()

{

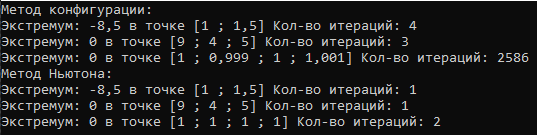
BuildH();

return H;

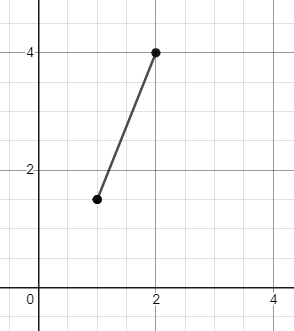
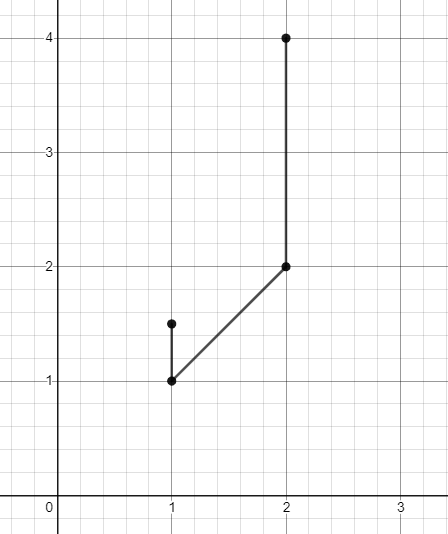
}

}

**Результаты выполнения программы**

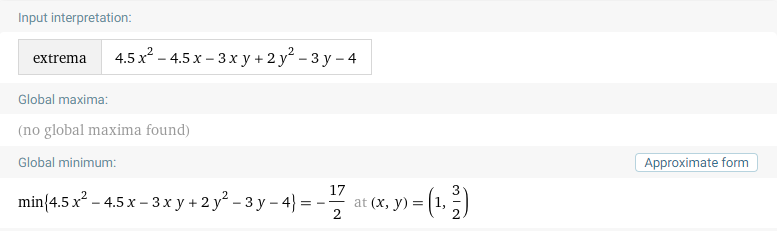


**Графическая интерпретация для двумерного случая**

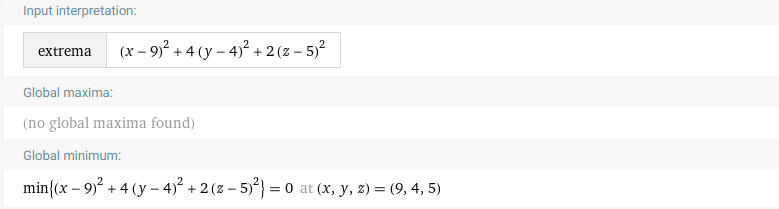
Метода конфигураций: Для метода Ньютона:

**Проверка вычислений в математическом пакете**

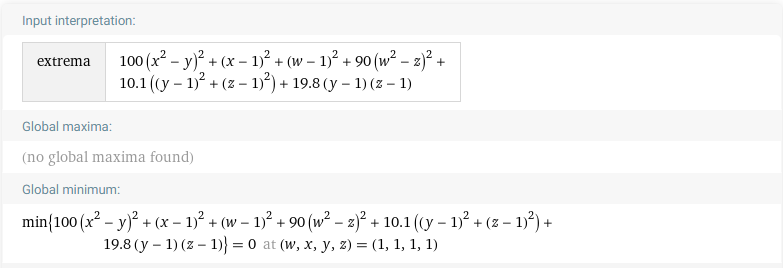
Первой функции:

****

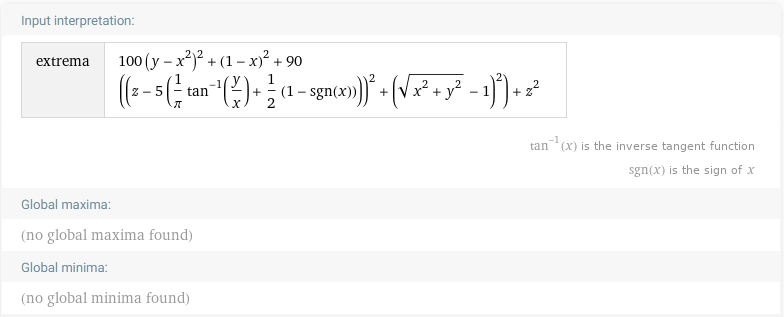
Второй функции:



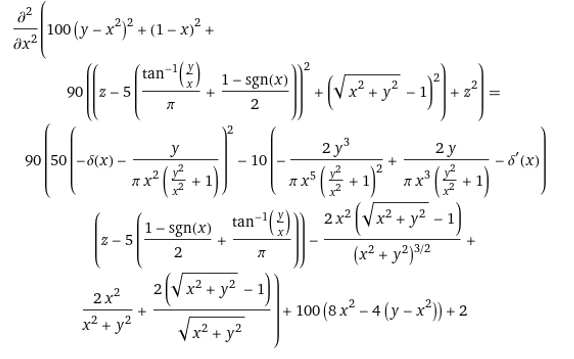
Третьей функции:

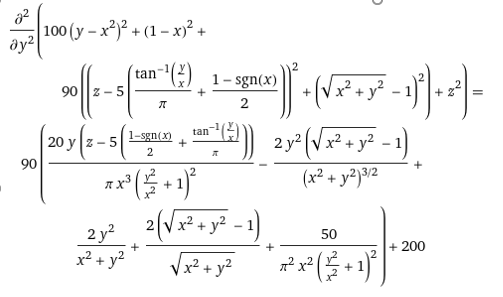
э

Для функции из задания экстремума нет:



Значение производных:





**Выводы**

Были найдены минимумы функций разными способами, методом конфигурации и методом Ньютона. Результат проверен математическим пакетом wolframalpha. Метод второго порядка оказался гораздо быстрее. Методы нулевого порядка менее эффективны на сложных функциях, т. к. сдвиг переменной изменяется постоянно уменьшаясь в какое-то количество раз, а не менялось динамически в зависимости от производный, из-за этого количество итераций очень велико.