ФГБОУ ВО

«Уфимский государственный авиационный технический университет»

Кафедра ТК

**ОТЧЕТ**

**по лабораторной работе № 4**

по дисциплине «Методы оптимизации»

**Тема: «Методы безусловной многомерной оптимизации нулевого порядка»**

**Вариант № 17**

Выполнила: студентка гр. ИВТ-221

Самсонова Е.О.

Проверил: доцент каф. ТК

Хасанов А.Ю.

Уфа 2021

**Содержание**

[Функция 3](#_Toc70948146)

[Задание 3](#_Toc70948147)

[Расчетные таблицы при : 3](#_Toc70948148)

[Расчетные таблицы при : 7](#_Toc70948149)

[Траектория движения точек при : 9](#_Toc70948150)

[Траектория движения точек при : 11](#_Toc70948151)

[Код программы 13](#_Toc70948152)

[Вывод 19](#_Toc70948154)

Функция

Задание

Построить таблицу для и методами:

1. Поиск по образцу;
2. Симплекс-регулярный;
3. Метод конфигураций;
4. Циклический покоординатный спуск.

**Найти точное решение** (на основе необходимых условий минимума).

В точке минимума функции должны выполняться **необходимые условия первого порядка**: в точке минимума частные производные первого порядка должны быть равны нулю.

Вычислим частные производные первого порядка целевой функции по :

Из первого уравнения выражаем

И подставим во второе уравнение:

Отсюда:

Ответ:

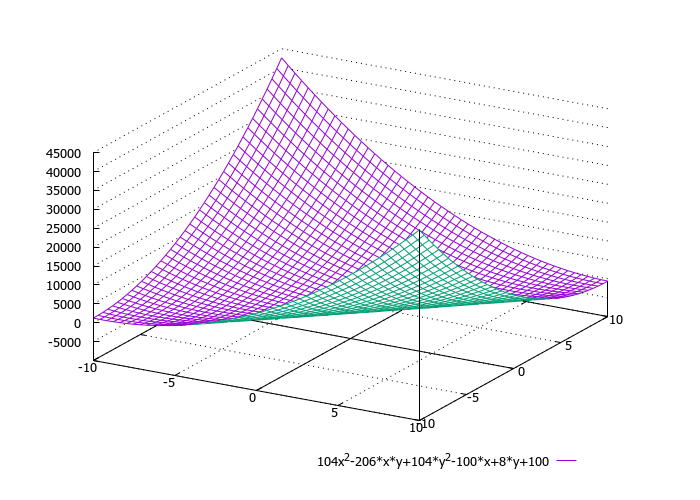


Рисунок 1− График функции

Расчетные таблицы при :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Поиск по образцу | | | | | | |
|  |  |  |  |  |  |  |
| 0.1 | 23,125 | 22,875 | -965,031 | 117 | 28 | 0.00767918 |
| 0.01 | 23,1289 | 22,8711 | -965,043 | 141 | 34 | 0.00215491 |
| 0.001 | 23,1304 | 22,8696 | -965,043 | 161 | 39 | 8,3306e-05 |
| 0.0001 | 23,1304 | 22,8696 | -965,043 | 181 | 44 | 3,01073e-06 |
| 0.00001 | 23,1304 | 22,8696 | -965,043 | 201 | 49 | 8,40552e-06 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регулярный симплекс | | | | | | |
|  |  |  |  |  |  |  |
| 0.1 | 3,53125 | 3,25 | -195,945 | 85 | 66 | 27,7318 |
| 0.01 | 20,7051 | 20,4414 | -953,264 | 1602 | 1577 | 3,43195 |
| 0.001 | 23,0111 | 22,7507 | -965,015 | 3740 | 3707 | 0,168404 |
| 0.0001 | 23,1151 | 22,8541 | -965,043 | 5279 | 5240 | 0,0217734 |
| 0.00001 | 23,1282 | 22,8673 | -965,043 | 6668 | 6623 | 0,00314114 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод конфигураций | | | | | | |
|  |  |  |  |  |  |  |
| 0.1 | 23,0117 | 22,7812 | -964,926 | 801 | 500 | 0,147962 |
| 0.01 | 23,126 | 22,8671 | -965,043 | 1560 | 962 | 0.00508282 |
| 0.001 | 23,1301 | 22,8694 | -965,043 | 25295 | 13972 | 0,000384011 |
| 0.0001 | 23,1304 | 22,8695 | -965,043 | 92833 | 50836 | 4,58331e-05 |
| 0.00001 | 23,1304 | 22,8696 | -965,043 | 254009 | 136548 | 8,1713e-06 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод покоординатного циклического спуска | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| 0.1 | 21,4102 | 21,1171 | -958,907 | 5202 | 5202 | 0 | 236 | 2,45564 |
| 0.01 | 22,6563 | 22,989 | -964,594 | 12581 | 12581 | 0 | 396 | 0,66879 |
| 0.001 | 23,1054 | 22,8449 | -965,042 | 29274 | 29274 | 0 | 898 | 0,035164 |
| 0.0001 | 23,1296 | 22,8688 | -965,043 | 45889 | 45889 | 0 | 978 | 0,00110883 |
| 0.00001 | 23,1299 | 22,8691 | -965,043 | 62857 | 62857 | 0 | 1102 | 0,00072932 |

Расчетные таблицы при :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Поиск по образцу | | | | | | |
|  |  |  |  |  |  |  |
| 0.1 | 23,125 | 22,875 | -965,031 | 125 | 30 | 0.00767918 |
| 0.01 | 23,1289 | 22,8711 | -965,043 | 149 | 36 | 0.00215491 |
| 0.001 | 23,1304 | 22,8696 | -965,043 | 169 | 41 | 8,3306e-05 |
| 0.0001 | 23,1304 | 22,8696 | -965,043 | 189 | 46 | 3,01073e-06 |
| 0.00001 | 23,1304 | 22,8696 | -965,043 | 209 | 51 | 8,40552e-06 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регулярный симплекс | | | | | | |
|  |  |  |  |  |  |  |
| 0.1 | 3,5 | 3,21875 | -193,492 | 141 | 122 | 27,776 |
| 0.01 | 20,709 | 20,4453 | -953,302 | 1658 | 1633 | 3,42643 |
| 0.001 | 23,0117 | 22,7513 | -965,015 | 3800 | 3767 | 0,167541 |
| 0.0001 | 23,1150 | 22,8541 | -965,043 | 5331 | 5292 | 0,0218813 |
| 0.00001 | 23,1282 | 22,8674 | -965,043 | 6728 | 6683 | 0,00312765 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Метод конфигураций | | | | | | |
|  |  |  |  |  |  |  |
| 0.1 | 23,0117 | 22,7812 | -964,926 | 825 | 516 | 0,147962 |
| 0.01 | 23,126 | 22,8671 | -965,043 | 1584 | 978 | 0.00508282 |
| 0.001 | 23,1301 | 22,8694 | -965,043 | 25319 | 13988 | 0,000384011 |
| 0.0001 | 23,1304 | 22,8695 | -965,043 | 92857 | 50852 | 4,58331e-05 |
| 0.00001 | 23,1304 | 22,8696 | -965,043 | 254033 | 136564 | 8,1713e-06 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Метод покоординатного циклического спуска | | | | | | | | |
|  |  |  |  |  |  |  |  |  |
| 0.1 | 21,4609 | 21,168 | -959,255 | 5375 | 5375 | 0 | 244 | 2,38383 |
| 0.01 | 22,6511 | 22,3948 | -964,586 | 12796 | 12796 | 0 | 404 | 0,674659 |
| 0.001 | 23,1054 | 22,8449 | -965,042 | 29274 | 29274 | 0 | 898 | 0,035164 |
| 0.0001 | 23,1051 | 22,8445 | -965,042 | 29614 | 29614 | 0 | 706 | 0,0356387 |
| 0.00001 | 23,1299 | 22,8691 | -965,043 | 63542 | 63542 | 0 | 1112 | 0,0007235 |

Траектория движения точек при :

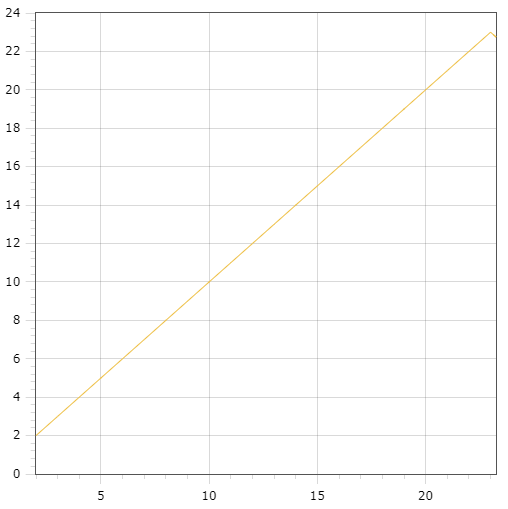


Рисунок 2 − Поиск по образцу

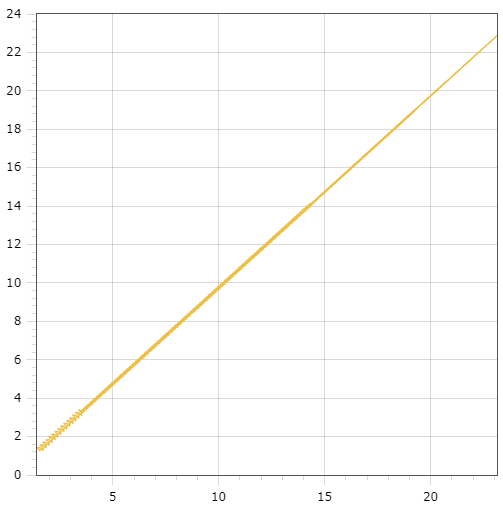


Рисунок 3 − Метод регулярного симплекса

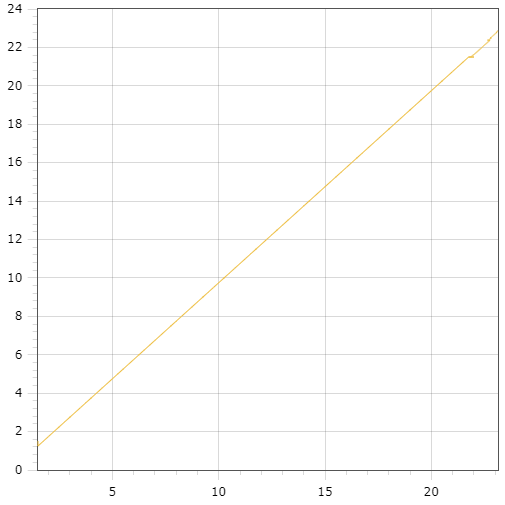


Рисунок 4 − Метод конфигураций

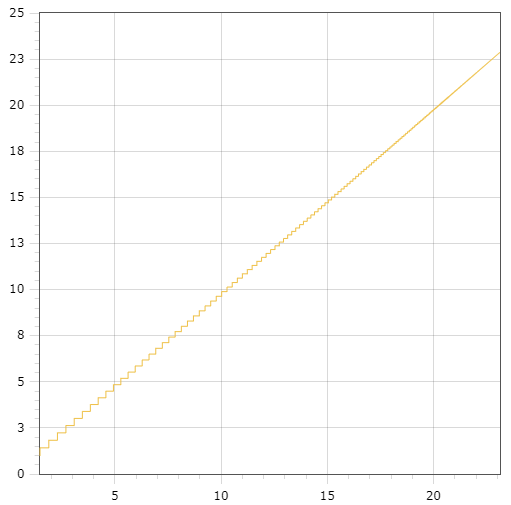


Рисунок 5 − Циклический покоординатный спуск

Траектория движения точек при :

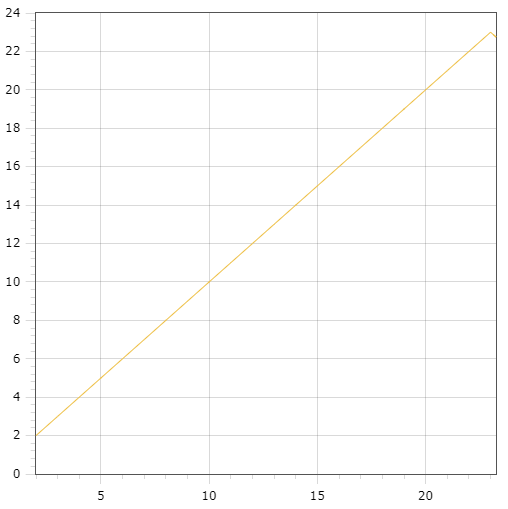


Рисунок 2 − Поиск по образцу

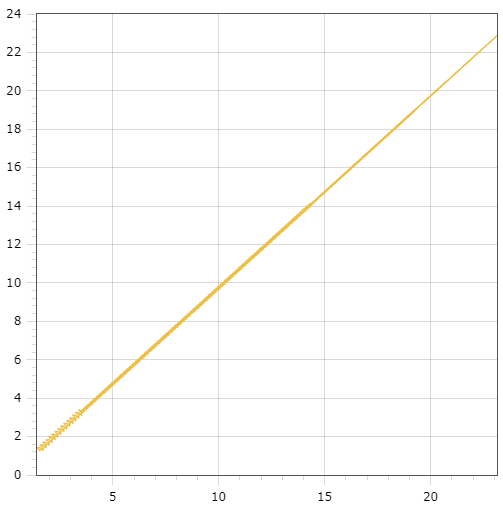


Рисунок 3 − Метод регулярного симплекса

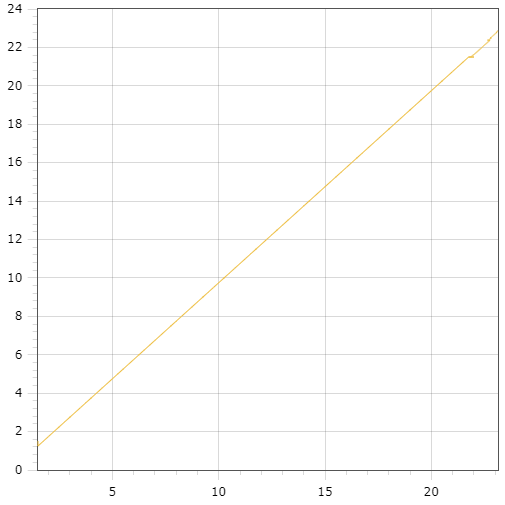


Рисунок 4 − Метод конфигураций

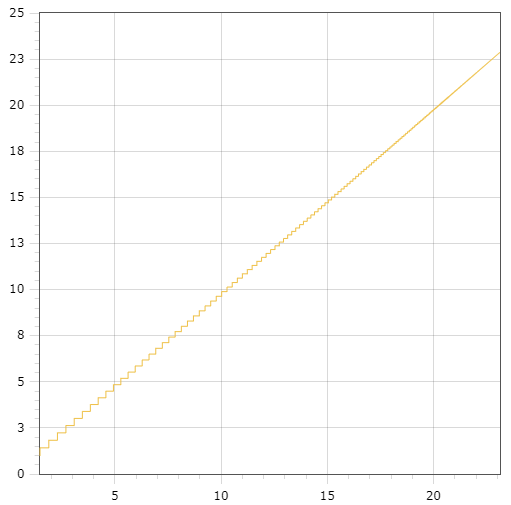


Рисунок 5 − Циклический покоординатный спуск

Код программы

#include <iostream> //Организация ввода-вывода

#include <conio.h> //Работа с текстовым интерфейсом пользователя (задержка getch())

#include <Windows.h> // Для использования SetConsoleCP(1251) и

#include <iomanip> //Манипуляторы для потоков ввода/вывода

#include <math.h>

#include <fstream>

using namespace std;

void Obraz(double x1, double x2);

void Simpl(double x1, double x2);

void Config(double x1, double x2);

void Cicl(double x1, double x2);

void find\_por1(double h, double x1, double x2, double E, double& L, int& N0);

void find\_por2(double h, double x1, double x2, double E, double& L, int& N0);

double f(double x1, double x2)

{

double f;

f = 104 \* pow(x1, 2) - 206 \* x1 \* x2 + 104 \* pow(x2, 2) - 100 \* x1 + 8 \* x2 + 100;

return f;

}

void find\_por1(double h, double x1, double x2, double E, double& L, int& N0)

{

double L0, L1, y1, y0;

L0 = 0;

y0 = f(x1, x2);

N0++;

label:

L1 = L0 + h;

y1 = f(x1 + L1, x2);

N0 = N0 + 1;

if (y0 > y1)

{

L0 = L1;

y0 = y1;

goto label;

}

else if (fabs(h) <= E)

{

L = L0;

return;

}

else

{

h = (-h) / 4;

L0 = L1;

y0 = y1;

goto label;

}

}

void find\_por2(double h, double x1, double x2, double E, double& L, int& N0)

{

double L0, L1, y1, y0;

L0 = 0;

y0 = f(x1, x2);

N0++;

label:

L1 = L0 + h;

y1 = f(x1, x2 + L1);

N0 = N0 + 1;

if (y0 > y1)

{

L0 = L1;

y0 = y1;

goto label;

}

else if (fabs(h) <= E)

{

L = L0;

return;

}

else

{

h = (-h) / 4;

L0 = L1;

y0 = y1;

goto label;

}

}

void Obraz(double x1, double x2)

{

ofstream fout;

const int P = 10;

char file[P];

fout.open("obrazec.txt");

double E, xr1, xr2, yr, Ef, h, x1m[5], x2m[5], y[5], min;

int N, l, i, k;

cout << "Введите E: E="; cin >> E;

cout << "Введите h: h="; cin >> h;

x1m[0] = x1; x2m[0] = x2;

cout << "(" << x1m[0] << ";" << x2m[0] << ") ";

y[0] = f(x1m[0], x2m[0]);

l = 0; N = 1;

label:

x1m[1] = x1m[0] + h; x2m[1] = x2m[0] + h;

x1m[2] = x1m[0] + h; x2m[2] = x2m[0] - h;

x1m[3] = x1m[0] - h; x2m[3] = x2m[0] - h;

x1m[4] = x1m[0] - h; x2m[4] = x2m[0] + h;

y[1] = f(x1m[1], x2m[1]); y[2] = f(x1m[2], x2m[2]);

y[3] = f(x1m[3], x2m[3]); y[4] = f(x1m[4], x2m[4]);

N = N + 4;

min = y[1];

k = 1;

for (i = 2; i < 5; i++)

if (y[i] < min) {

min = y[i];

k = i;

}

if (y[k] < y[0])

{

x1m[0] = x1m[k];

x2m[0] = x2m[k];

cout << "(" << x1m[0] << ";" << x2m[0] << ") ";

fout << x1m[0] << ";" << setw(20) << x2m[0] << endl;

y[0] = y[k];

l = l + 1;

goto label;

}

else if (sqrt(2) \* h > E)

{

h = h / 2;

l = l + 1;

goto label;

}

else

{

xr1 = x1m[0];

xr2 = x2m[0];

yr = y[0];

}

Ef = sqrt(pow(abs(xr1 - 23.13043), 2) + pow(abs(xr2 - 22.86957), 2));

cout << "Метод поиска по образцу дал следующее лучшее решение: x1=" << xr1 << ",x2 = " << xr2 << ", y = " << yr << " за N = " << N << " за " << l << " итераций. При этом фактическая точность Eфакт = " << Ef << endl;

\_getch();

}

void Simpl(double x1, double x2)

{

ofstream fout;

const int P = 10;

char file[P];

fout.open("simplex.txt");

double E, xr1, xr2, yr, Ef, r, x1m[3], x2m[3], ym[3], min, max, c1, c2, cs1 = 0,

cs2 = 0, u1, u2, y;

int N, l, i, k1, k2;

cout << "Введите E: E="; cin >> E;

cout << "Введите r: r="; cin >> r;

x1m[0] = x1; x2m[0] = x2;

cout << "(" << x1m[0] << ";" << x2m[0] << ") ";

x1m[1] = x1m[0] + r; x2m[1] = x2m[0];

x1m[2] = x1m[0]; x2m[2] = x2m[0] + r;

cout << "(" << x1m[1] << ";" << x2m[1] << ") ";

cout << "(" << x1m[2] << ";" << x2m[2] << ") ";

l = 0;

ym[0] = f(x1m[0], x2m[0]);

ym[1] = f(x1m[1], x2m[1]);

ym[2] = f(x1m[2], x2m[2]);

N = 3;

while (1)

{

min = ym[0];

k1 = 0;

for (i = 0; i < 3; i++)

if (ym[i] < min)

{

min = ym[i];

k1 = i;

}

max = ym[0];

k2 = 0;

for (i = 0; i < 3; i++)

if (ym[i] > max)

{

max = ym[i];

k2 = i;

}

if (r <= E)

{

xr1 = x1m[k1];

xr2 = x2m[k1];

cout << "(" << x1m[k1] << ";" << x2m[k1] << ") ";

fout << x1m[k1] << ";" << setw(20) << x2m[k1] << endl;

yr = ym[k1];

Ef = sqrt(pow(abs(xr1 - 23.13043), 2) + pow(abs(xr2 - 22.86957), 2));

cout << "Метод регулярного симплекса дал следующее лучшее решение: x1 = " << xr1 << ", x2 = " << xr2 << ", y = " << yr << " за N = " << N << " за " << l << " итераций.При этом фактическая точность Eфакт = " << Ef << endl;

\_getch(); return;

}

c1 = c2 = 0;

for (i = 0; i < 3; i++)

{

if (i == k2) continue;

c1 = c1 + x1m[i];

c2 = c2 + x2m[i];

}

c1 = c1 / 2;

c2 = c2 / 2;

u1 = 2 \* c1 - x1m[k2];

u2 = 2 \* c2 - x2m[k2];

y = f(u1, u2);

N = N + 1;

if (y < ym[k2])

{

x1m[k2] = u1;

x2m[k2] = u2;

cout << "(" << x1m[k2] << ";" << x2m[k2] << ") ";

fout << x1m[k2] << ";" << setw(20) << x2m[k2] << endl;

ym[k2] = y;

l = l + 1;

}

else

{

for (i = 0; i < 3; i++)

{

if (i == k1) continue;

x1m[i] = (x1m[i] + x1m[k1]) / 2;

x2m[i] = (x2m[i] + x2m[k1]) / 2;

ym[i] = f(x1m[i], x2m[i]);

}

r = r / 2;

N = N + 2;

l = l + 1;

}

}

}

void Cicl(double x1, double x2)

{

ofstream fout;

const int P = 10;

char file[P];

fout.open("cicl.txt");

double E, L, xr1, xr2, yr, Ef, d, h;

double x1l, x2l, x1l1, x2l1, x1l2, x2l2;

int l, N, N1, N0;

cout << "Введите E: E="; cin >> E;

cout << "Введите h: h="; cin >> h;

x1l = x1;

x2l = x2;

cout << "(" << x1l << ";" << x2l << ") ";

N0 = 0;

l = 0;

label:

find\_por1(h, x1l, x2l, E, L, N0);

//cout << " h= " << h << " L= " << L <<endl;

x1l1 = x1l + L;

x2l1 = x2l;

cout << "(" << x1l1 << ";" << x2l1 << ") ";

fout << x1l1 << ";" << setw(20) << x2l1 << endl;

find\_por2(h, x1l1, x2l, E, L, N0);

//cout << " h2= " << h << " L2= " << L << endl;

x1l2 = x1l1;

x2l2 = x2l1 + L;

cout << "(" << x1l2 << ";" << x2l2 << ") ";

fout << x1l2 << ";" << setw(20) << x2l2 << endl;

d = sqrt(pow(x1l2 - x1l, 2) + pow(x2l2 - x2l, 2));

l = l + 2;

x1l = x1l2;

x2l = x2l2;

if (d > E) goto label;

else

{

xr1 = x1l;

xr2 = x2l;

yr = f(xr1, xr2);

N = N0;

N1 = 0;

}

Ef = sqrt(pow(abs(xr1 - 23.13043), 2) + pow(abs(xr2 - 22.86957), 2));

cout << "Метод покоординатного циклического спуска дал следующее лучшее решение: x1 = " << xr1 << ", x2 = " << xr2 << ", y = " << yr << " за N = " << N << ", при том, что N0 = " <<N0 << ", а N1=" << N1 << " за " << l << " итераций. При этом фактическая точность Eфакт=" << Ef << endl;

\_getch();

}

void Config(double x1, double x2)

{

ofstream fout;

const int L = 10;

char file[L];

fout.open("congif.txt");

double E, h = 2, lam = 1, Egar, x1l, x2l, y[4], y1, x1l1, x2l1, X1, X2, Y, x1l2, x2l2;

int N, l;

cout << "Введите E: E="; cin >> E;

l = 0;

x1l = x1; x2l = x2;

y[0] = f(x1l, x2l);

N = 0;

label1:

y[1] = f(x1l + h, x2l);

N++;

if (y[0] > y[1])

{

x1l1 = x1l + h;

goto label2;

}

y[1] = f(x1l - h, x2l);

N++;

if (y[0] > y[1])

{

x1l1 = x1l - h;

goto label2;

}

y[1] = y[0];

x1l1 = x1l;

label2:

y[2] = f(x1l1, x2l + h);

N++;

if (y[1] > y[2])

{

x2l1 = x2l + h;

goto label3;

}

y[2] = f(x1l1, x2l - h);

N++;

if (y[1] > y[2])

{

x2l1 = x2l - h;

goto label3;

}

y[2] = y[1];

x2l1 = x2l;

label3:

if ((x1l1 == x1l) && (x2l1 == x2l))

{

if (sqrt(2) \* h <= E)

{

X1 = x1l;

X2 = x2l;

Y = y[0];

}

else

{

h = h / 2;

goto label1;

}

}

else

{

label4:

x1l2 = x1l + lam \* (x1l1 - x1l);

x2l2 = x2l + lam \* (x2l1 - x2l);

fout << x1l1 << ";" << setw(20) << x2l2 << endl;

y[3] = f(x1l2, x2l2);

N++;

if (y[3] > y[2])

{

y[0] = y[3];

l = l + 2;

x1l = x1l2;

x2l = x2l2;

goto label1;

}

else

{

lam = lam / 2;

goto label4;

}

}

Egar = sqrt(pow(abs(X1 - 23.13043), 2) + pow(abs(X2 - 22.86957), 2));

cout << "Метод конфигураций дал следующее лучшее решение: x1=" << X1 << ", x2="

<< X2 << ", y=" << Y << " за N=" << N << " за " << l << " итераций. При этом фактическая точность Eфакт = " << Egar << endl;

\_getch();

/\*cout << setw(10) << "X1" << "|" << setw(10) << "X2" << "|" << setw(10) << "f(x)" << "|" << setw(10) << "l" << "|" << setw(10) << "N" << "|" << setw(11) << "Eф" << "|" << endl;

cout << setw(10) << X1 << "|" << setw(10) << X2 << "|" << setw(10) << Y << "|" << setw(10) << l << "|" << setw(10) << N << "|" << setw(11) << Egar << "|" << endl;

Config(E \* 0.1);

return;\*/

}

int main()

{

setlocale(LC\_ALL, "Russian");

int j;

double a, b;

cout << "Введите x1[0]=";

cin >> a;

cout << "Введите x2[0]=";

cin >> b;

while (1)

{

system("cls");

cout << "1. Поиск по образцу\n";

cout << "2. Регулярный симплекс\n";

cout << "3. Метод конфигураций\n";

cout << "4. Метод покоординатного циклического спуска\n";

cout << "5. Конец работы\n";

cout << "Ваш выбор (1-5): ";

cin >> j;

switch (j)

{

case 1: Obraz(a, b); break;

case 2: Simpl(a, b); break;

case 3: Config(a, b); break;

case 4: Cicl(a, b); break;

case 5: cout << "Конец работы\n"; \_getch();

default: cout << "Ошибка, нет такого пункта в меню\n"; break;

\_getch();

}

}

}

Вывод

В ходе лабораторной работы были сравнены различные методы

поиска нулевого порядка. Лучший результат по числу экспериментов для

заданной целевой функции для поиска оптимального решения с точностями

0.1, 0.01, 0.001, 0.0001, 0.00001 показал метод поиска по образцу.