Ministerul Educației, Culturii și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Ingineria Software și Automatică

RAPORT

Lucrarea de laborator Nr.2
la disciplina Metode și Modele de Calcul 2
Tema: Optimizarea unei funcții neliniare dupa metoda directiilor conjugate

A efectuat: st. gr. TI-191 Gorbatenchii Catalin

A verificat : asist. Univ. Buldumac Oleg

Chişinău 2020

Sarcina lucrarii:

Optimizarea unei functii neliniare dupa metoda directiilor conjugate.

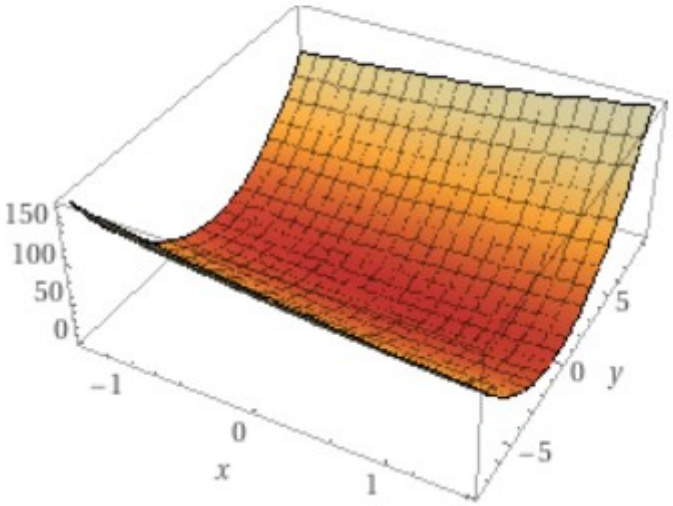
$$f(x,y) = ax^2 + 2xy + by^2 - bx - ay$$

Varianta după registru	a	b
12	3	2

F(x) = 3x^2 + 2xy + 2y^2 – 2x - 3y
Derivata partiala dupa x: 6x + 2y – 2
Derivata partiala dupa y: 2x + 4y - 3

$3x^2 + 2xy + 2y^2 - 2x - 3y$

3D plot:



Graficul functiei propuse construit cu ajutorul paginii WEB Wolfram Alpha Widgets

Global minimum:

$$\min\{3x^2 + 2xy + 2y^2 - 2x - 3y\} = -\frac{23}{20} \text{ at } (x,y) = \left(\frac{1}{10}, \frac{7}{10}\right)$$

Punctul minim calculat cu ajutorul paginii WEB Wolfram Alpha Widgets

Codul programului:

```
#include <iostream>
#include <math.h>
#define N 5
#define EPSILON 10E-7
using namespace std;

double main_function(double x, double y)
{
    return 3 * pow(x, 2) + 2 * x * y + 2 *pow(y, 2) - 2 * x - 3 * y;
```

```
}

double magnitude(double x, double y)
{
    return sqrt(pow(x, 2) + pow(y, 2));
}

double scalar_multiplication(double x1, double x2, double y1, double y2)
{
    return x1 * y1 + x2 * y2;
}

double matrix_multiplication(double x, double y, double r)
{
    double matrix[N][N] = {{6, 2}, {2, 4}};

    if(r == 1)
        return matrix[0][0] * x + matrix[0][1] * y;

    if(r == 2)
        return matrix[1][0] * x + matrix[1][1] * y;
}

double ALFA(double a, double b, double c, double d)
{
    double top = scalar_multiplication(a, b, c, d);
    double bottom = scalar_multiplication(matrix_multiplication(c, d, 1),
matrix_multiplication(c, d, 2), c, d);
    return (-1) * top / bottom;
}

int main()
{
    int k;
    double x0, y0, gradientulX, gradientulY, Z1, Z2;
    double d1, d2, g1, g2, alpha;

    x0 = 0;
    y0 = 1;

    gradientulX = 6 * x0 + 2 * y0 - 2;
    gradientulY = 2 * x0 + 4 * y0 - 3;

    if(gradientulX == 0 && gradientulY == 0)
    {
        cout << "\tx = " << x0 << endl;
        cout << "\ty = " << y0 << endl <<endl;
        return 0;
    }

    d1 = -1 * gradientulX;
    d2 = -1 * gradientulY;
    k = 0;
    cout << "d1 = " << d1 << endl;
    cout << "d2 = " << d2 << endl << endl;

    alpha = ALFA(gradientulX, gradientulY, d1, d2);
    cout << "Alpha(" << k << "): " << alpha << endl;

    cout << "Iteration: " << k << endl << endl;

    Z1 = x0 + alpha * d1;
    Z2 = y0 + alpha * d2;
    g1 = 6 * Z1 + 2 * Z2 - 2;
    g2 = 2 * Z1 + 4 * Z2 - 3;;

    cout << "Gradientui X = " << g1 << endl;
```

```
    cout << "Gradientul Y = " << g2 << endl ;


    cout << "Initial magnitude = " << magnitude(g1, g2) << endl << endl;

    while(magnitude(g1, g2) >= EPSILON || k > 1000)
    {
        ++k;
        cout << "Iteratia: " << k << endl;

        d1 = (-1 * g1) + pow(magnitude(g1, g2), 2) / pow(magnitude(gradientulX,
gradientulY), 2) * d1;
        d2 = (-1 * g2) + pow(magnitude(g1, g2), 2) / pow(magnitude(gradientulX,
gradientulY), 2) * d2;
        cout << "d1 = " << d1 << endl;
        cout << "d2 = " << d2 << endl;
        alpha = ALFA(g1, g2, d1, d2);
        cout << endl;
        cout << "Alpha(" << k << "): " << alpha << endl;

        Z1 = Z1 + alpha * d1;
        Z2 = Z2 + alpha * d2;
        cout << "Z1 = " << Z1 << endl;
        cout << "Z2 = " << Z2 << endl;

        gradientulX = g1;
        gradientulY = g2;
        g1 = 6 * Z1 + 2 * Z2 - 2;
        g2 = 2 * Z1 + 4 * Z2 - 3;;
        cout << "Final magnitude: " << magnitude(g1, g2) << endl << endl;

    }
   cout << "x = " << Z1 << endl;
   cout << "y = " << Z2 << endl;
    cout << "f(x, y) = " << main_function(Z1, Z2) << endl;

    return 0;
}
```

Rezultatul:

```
d1 = -0
d2 = -1

Alpha(0): 0.25
Iteration: 0

Gradientui X = -0.5
Gradientul Y = 0
Initial magnitude = 0.5

Iteratia: 1
d1 = 0.5
d2 = -0.25

Alpha(1): 0.2
Z1 = 0.1
Z2 = 0.7
Final magnitude: 1.75542e-16

x = 0.1
y = 0.7
f(x, y) = -1.15

Process finished with exit code 0
```

Concluzii:
In aceasta lucrare de laborator am avut drept scop optimizarea unei functii dupa metoda directiilor conjugate, cu alte cuvinte, gasirea punctului minim al unei functii neliniare dupa algoritmul lui Hestenes-Stiefel. Rezultatul (f(x,y) = - 1.15) a fost obtinut din doar 2 iteratii.