****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Raport**

**Lucrarea de laborator Nr.2**

***la cursul de “Metode și modele de calcul 2”***

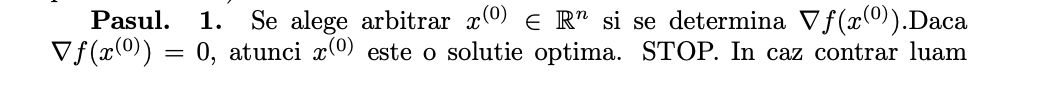
Efectuat: Studentul gr. TI-207 Bunescu Gabriel

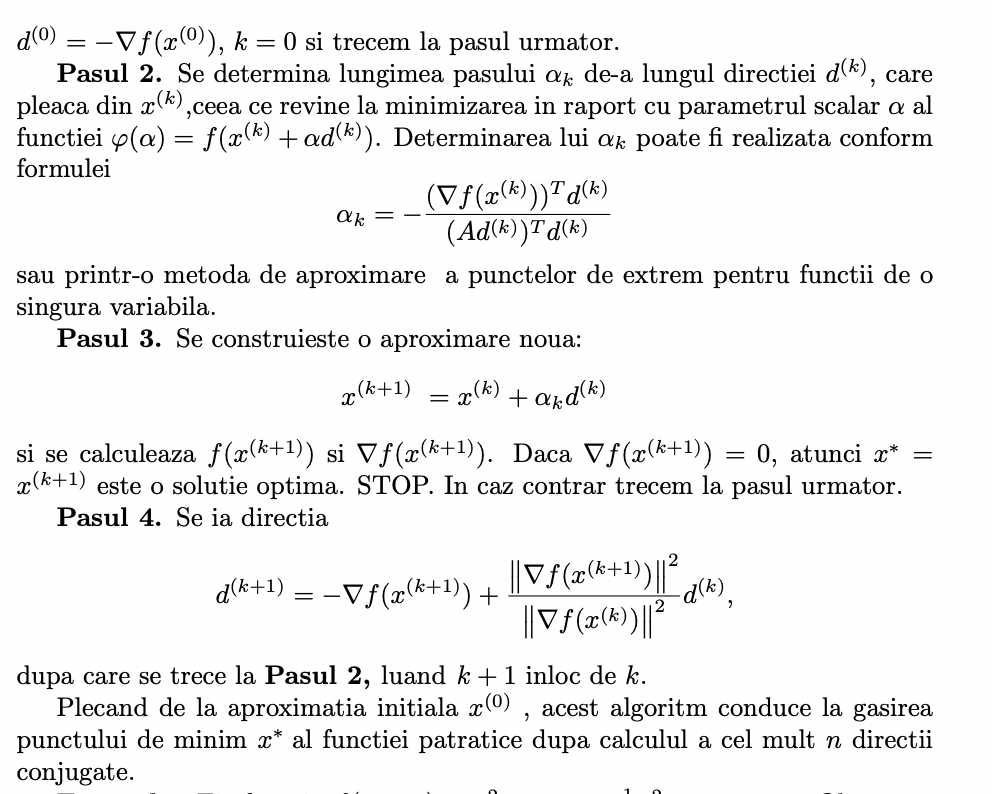
Verificat:  *ast.univ*  Buldumac Oleg

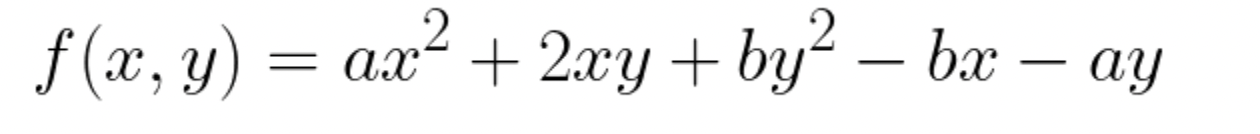
Chișinău – 2021

**Inforamtii despre:**

Metoda lui Hestenes-Stiefel

****

****



|  |  |  |
| --- | --- | --- |
| **Varianta** | **a** | **b** |
| **5** | **7** | **3** |

***Codul programului:***

#include <iostream>

#include <math.h>

#define N 5

#define EPSILON 10E-5

using namespace std;

double main\_function(double x, double y)

{

return 7 \* pow(x, 2) + 2 \* x \* y + 3 \*pow(y, 2) - 3 \* x - 7 \* 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] = {{14, 2},

{2, 6}};

if(r == 1)

return matrix[0][0] \* x + matrix[0][1] \* y;

if(r == 2) {

return matrix[1][0] \* x + matrix[1][1] \* y;

} else

return 0;

}

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 = 7;

y0 = 3;

gradientulX = 14 \* x0 + 2 \* y0 - 3;

gradientulY = 2 \* x0 + 6 \* y0 - 7;

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 = 14 \* Z1 + 2 \* Z2 - 3;

g2 = 2 \* Z1 + 6 \* Z2 - 7;;

cout << "Gradientui X = " << gradientulX << endl;

cout << "Gradientul Y = " << gradientulY << endl ;

cout << "Initial magnitude = " << magnitude(gradientulX, gradientulY) << endl << endl;

while(magnitude(g1, g2) > EPSILON)

{

++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 = 14 \* Z1 + 2 \* Z2 - 3;

g2 = 2 \* Z1 + 6 \* Z2 - 7;;

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;

}

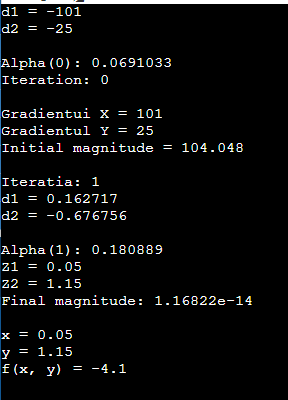


Fig.1.1. Rezultatele compilari

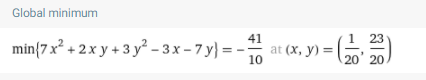


Fig.2.1. Minimul global cu ajutorul wolfram-online

X=0.05

Y=1.15

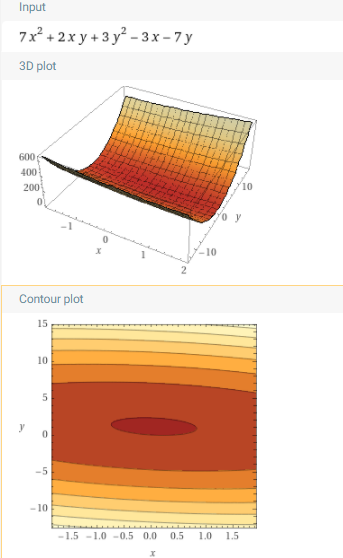
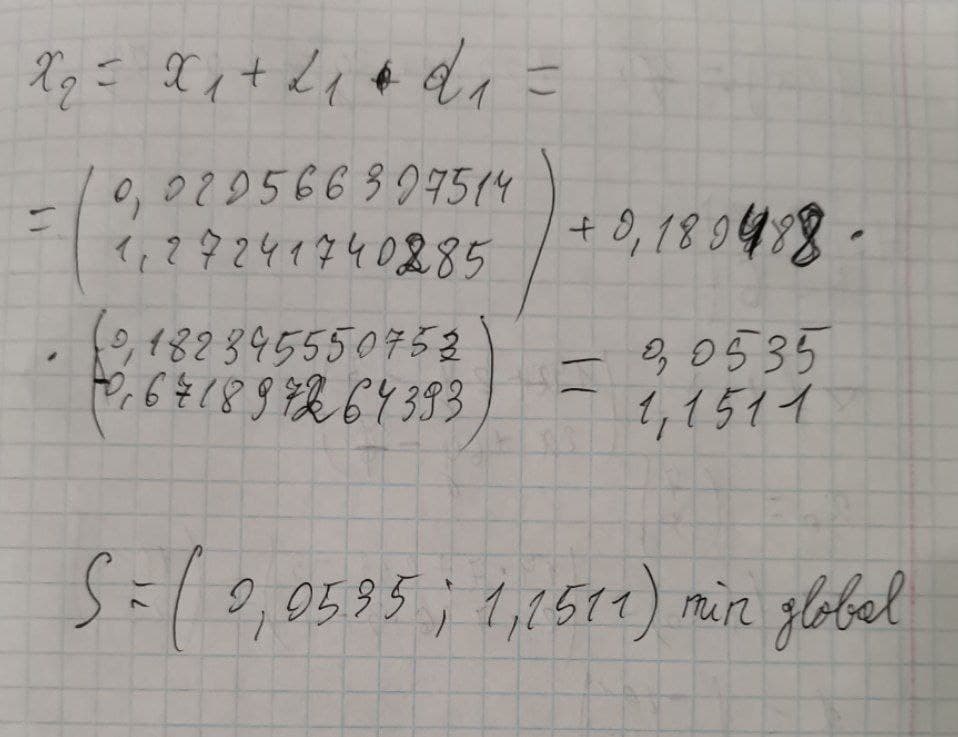
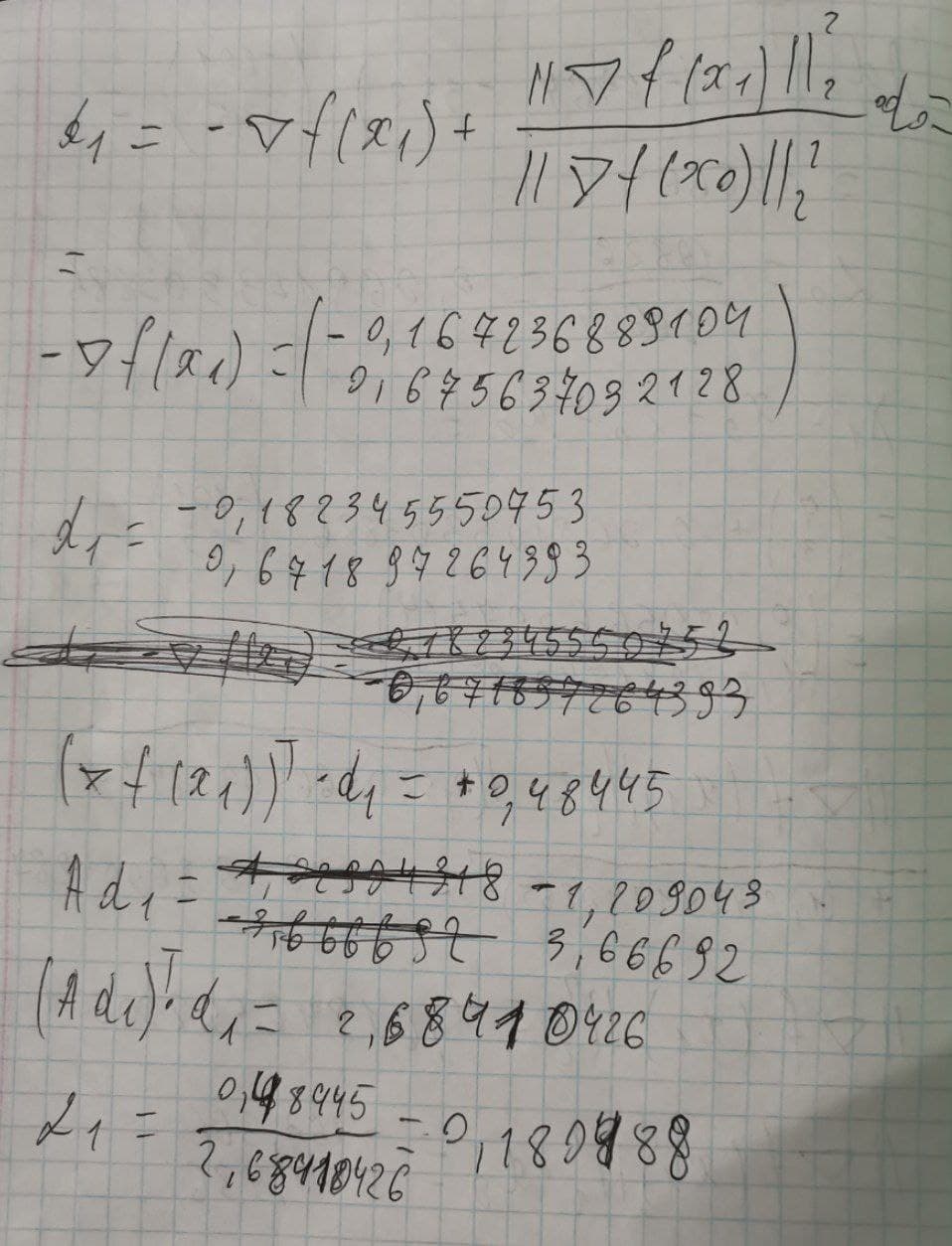
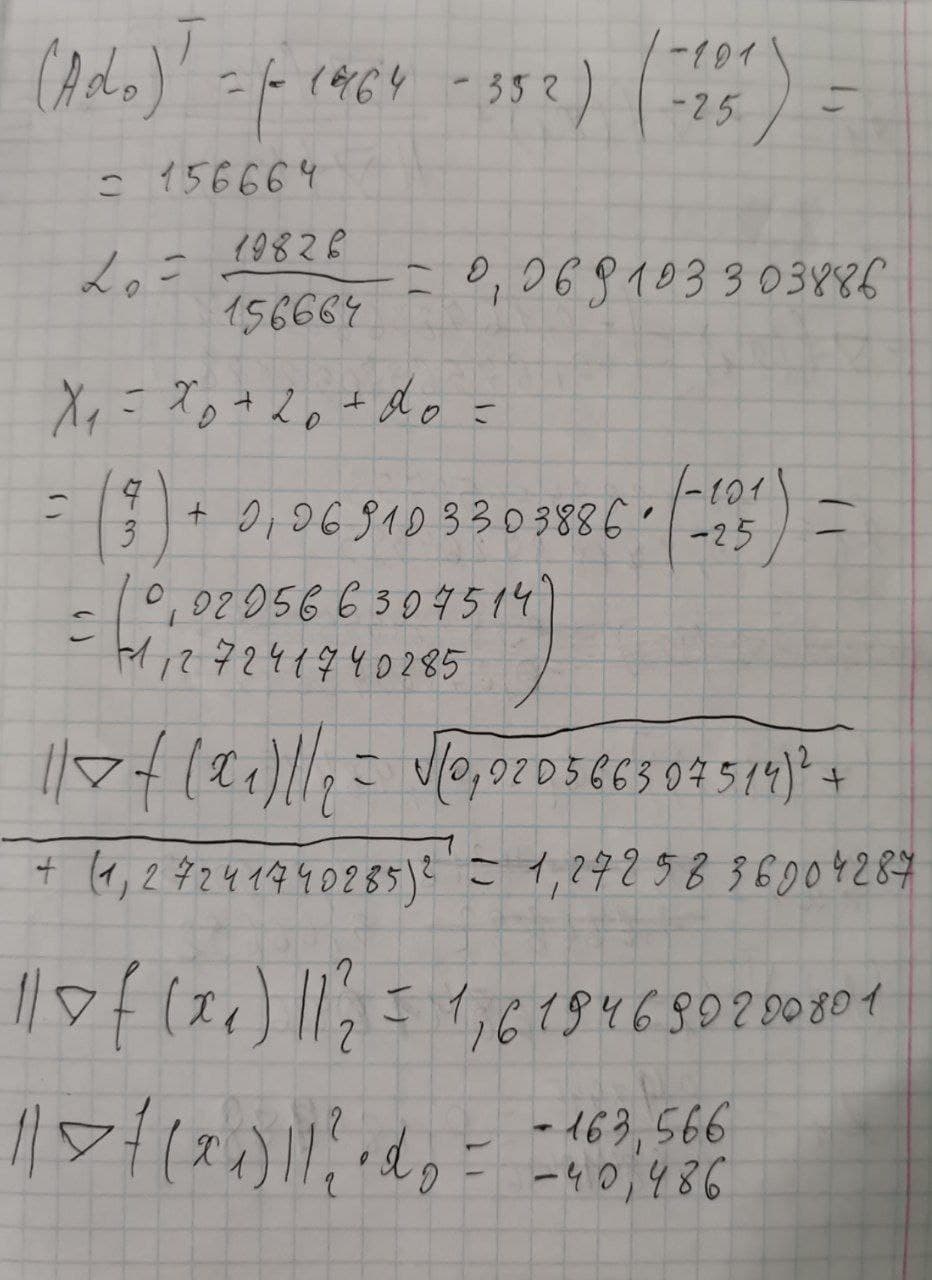
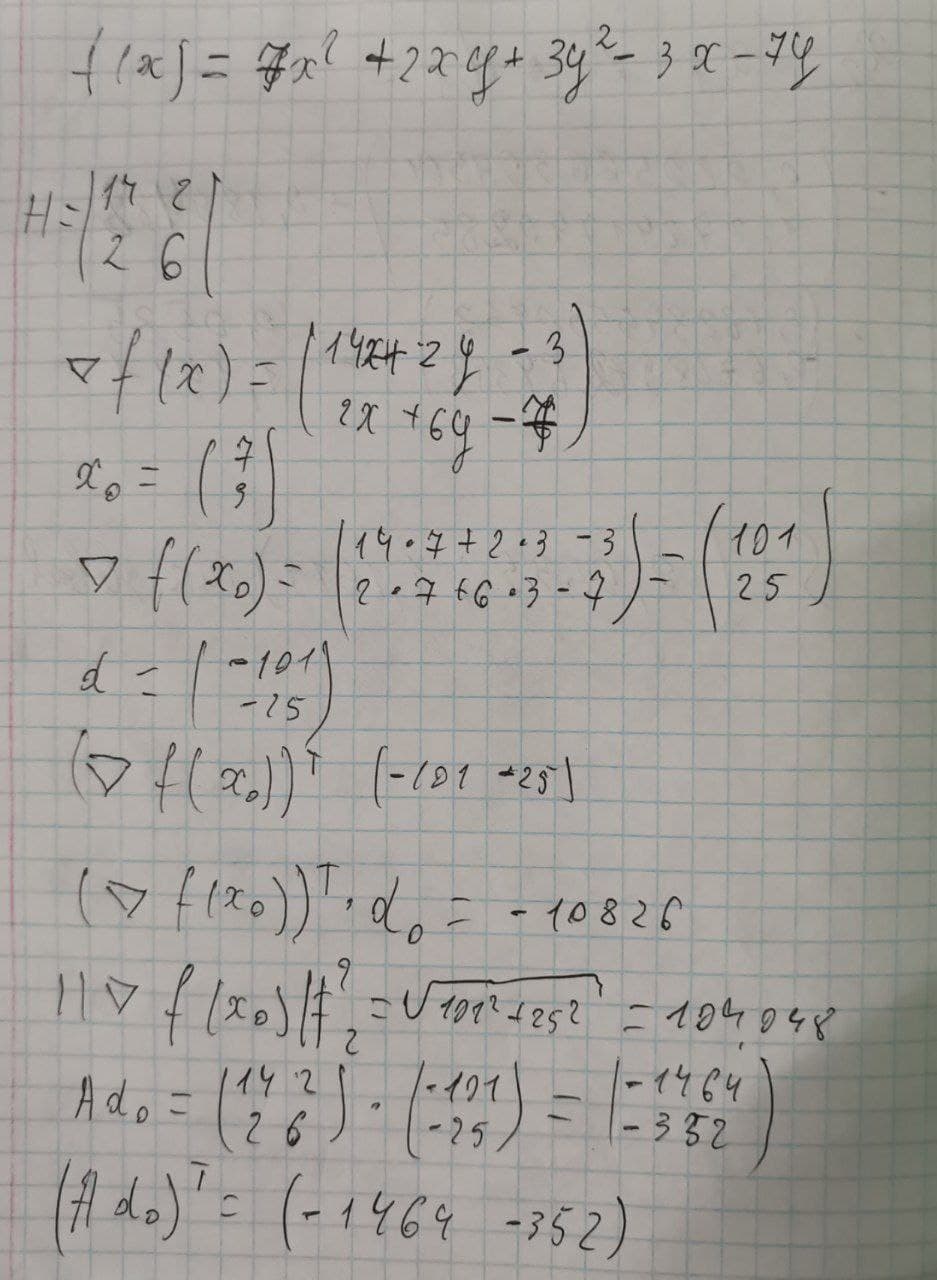


Fig.3.1. Graficele din wolfram-online

**Calculele în caiet:**



**Concluzie:**

Efectuând această lucrare de laborator am însușit metoda de căutare a minimului funcției, și anume “gradientului cu fracționarea pasului". Si la fel am implimentat o metoda noua „Metoda lui Hestenes-Stiefel” care rezolva aceasta problema mai rapid, prin două iteratie.