****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Raport**

**pentru lucrarea de laborator Nr.2**

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

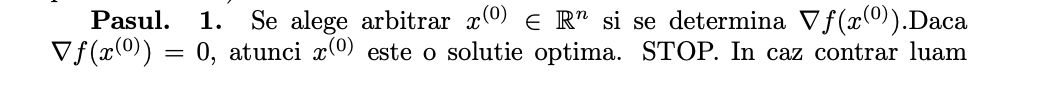
Efectuat: Studentul gr. TI-207 **Ursu Rubin**

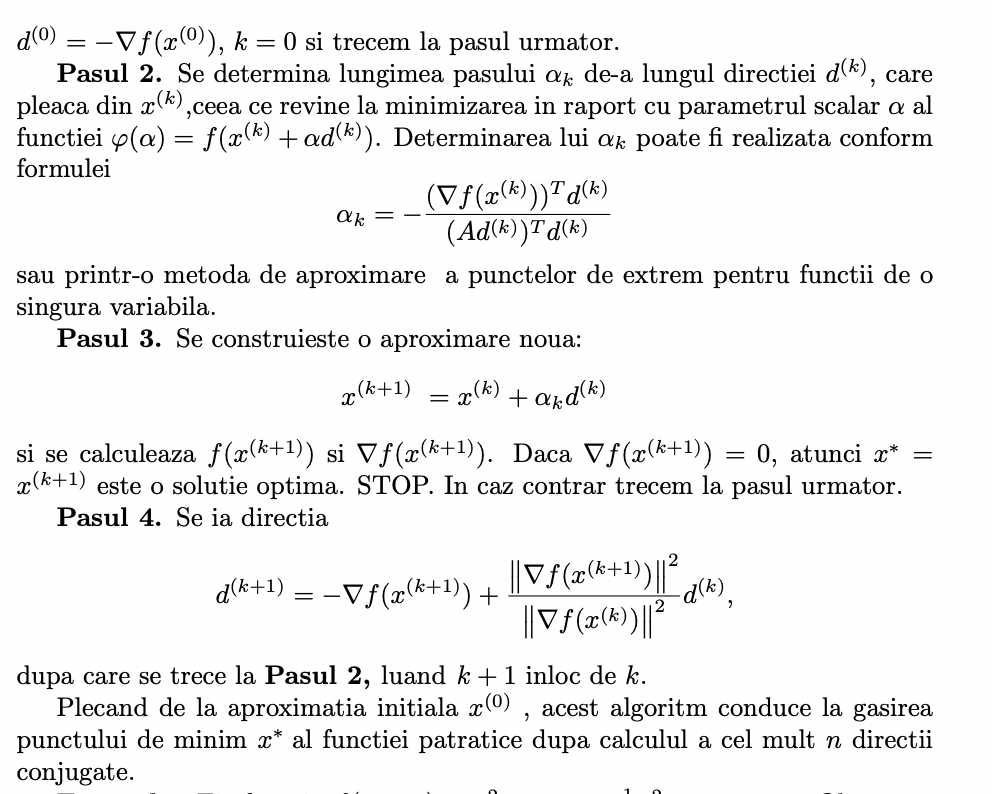
Verificat:  *ast.univ*  **Buldumac Oleg**

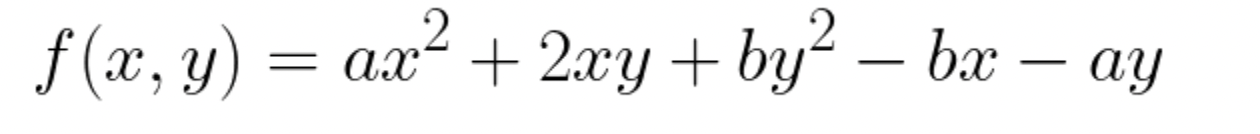
**Chișinău – 20****21**

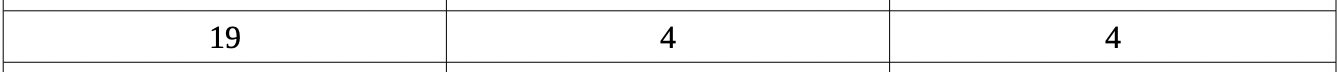
**Inforamtii despre:**

Metoda lui Hestenes-Stiefel

****

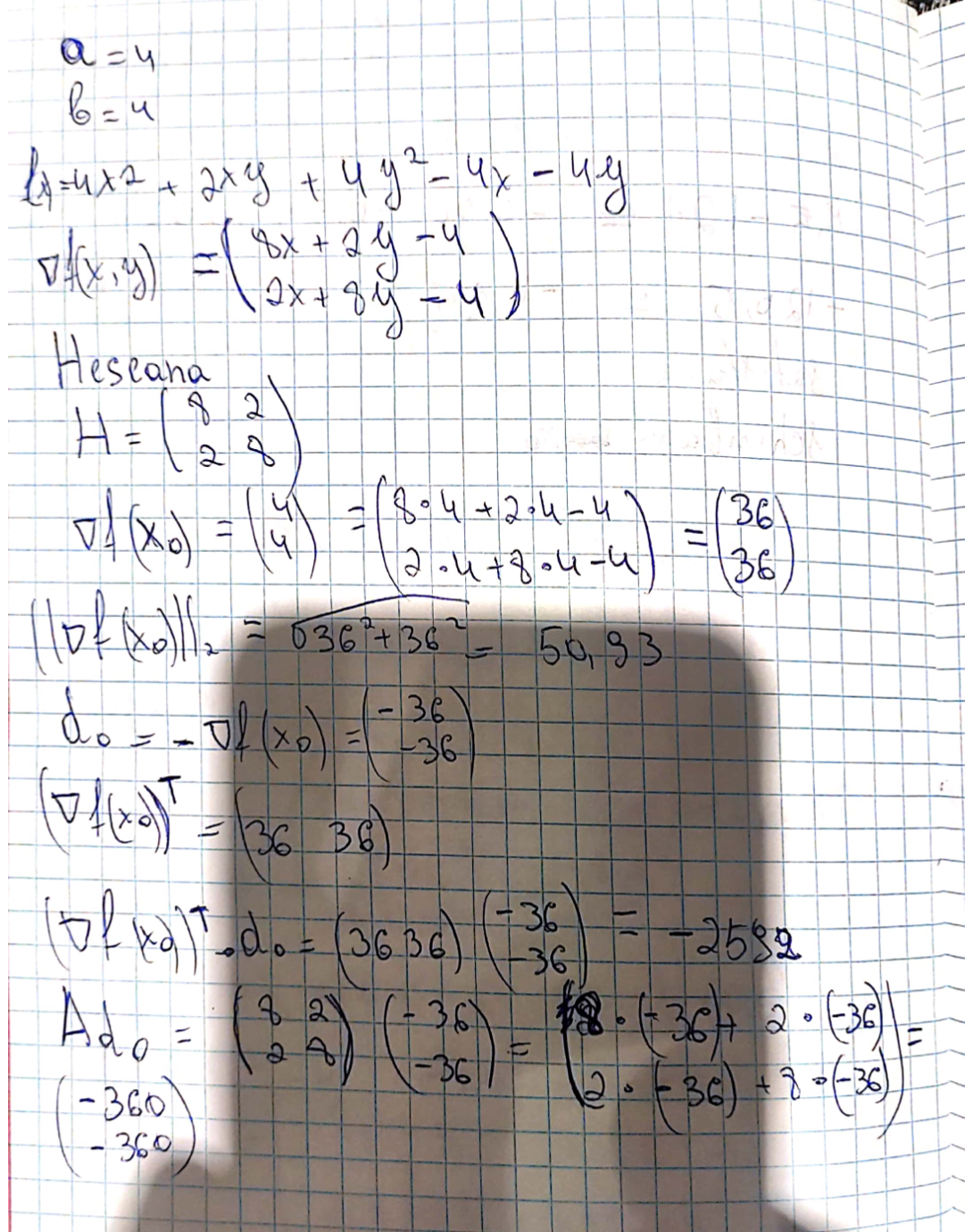
****

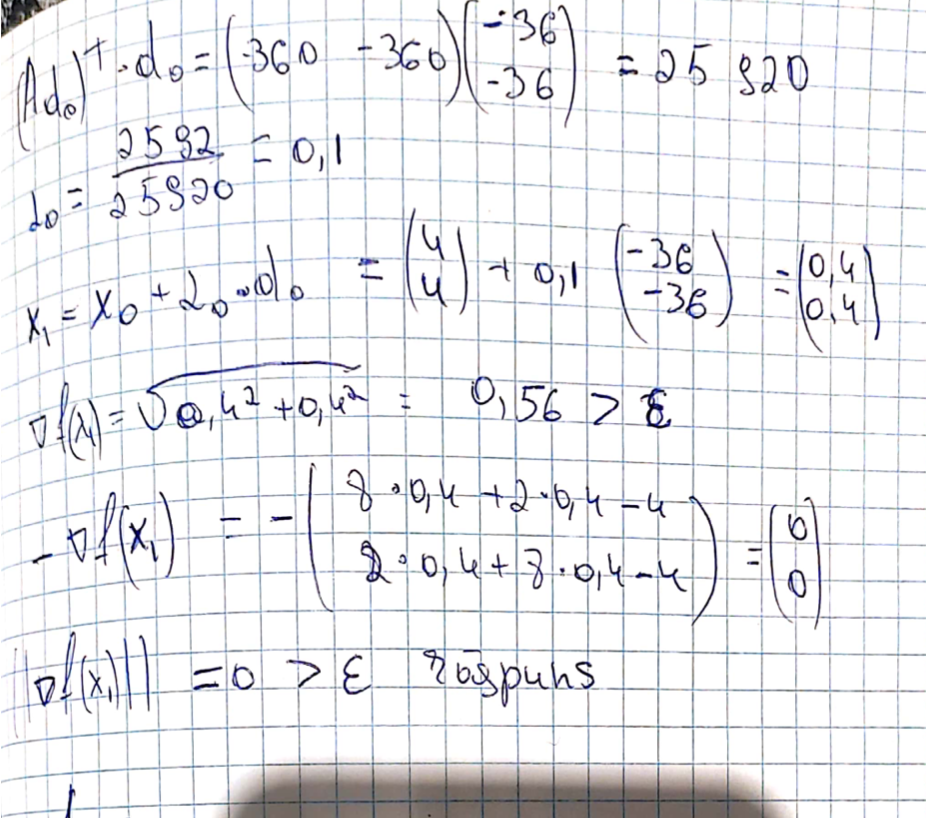


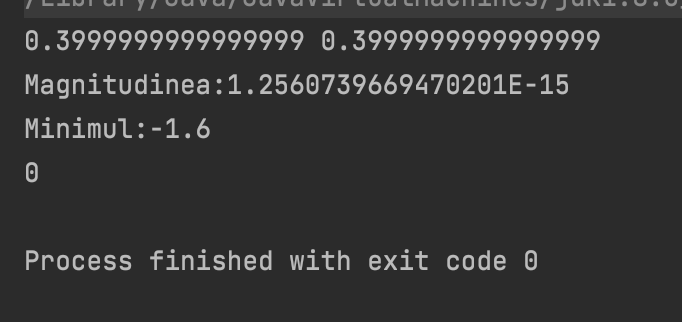
******

***Codul programului:***

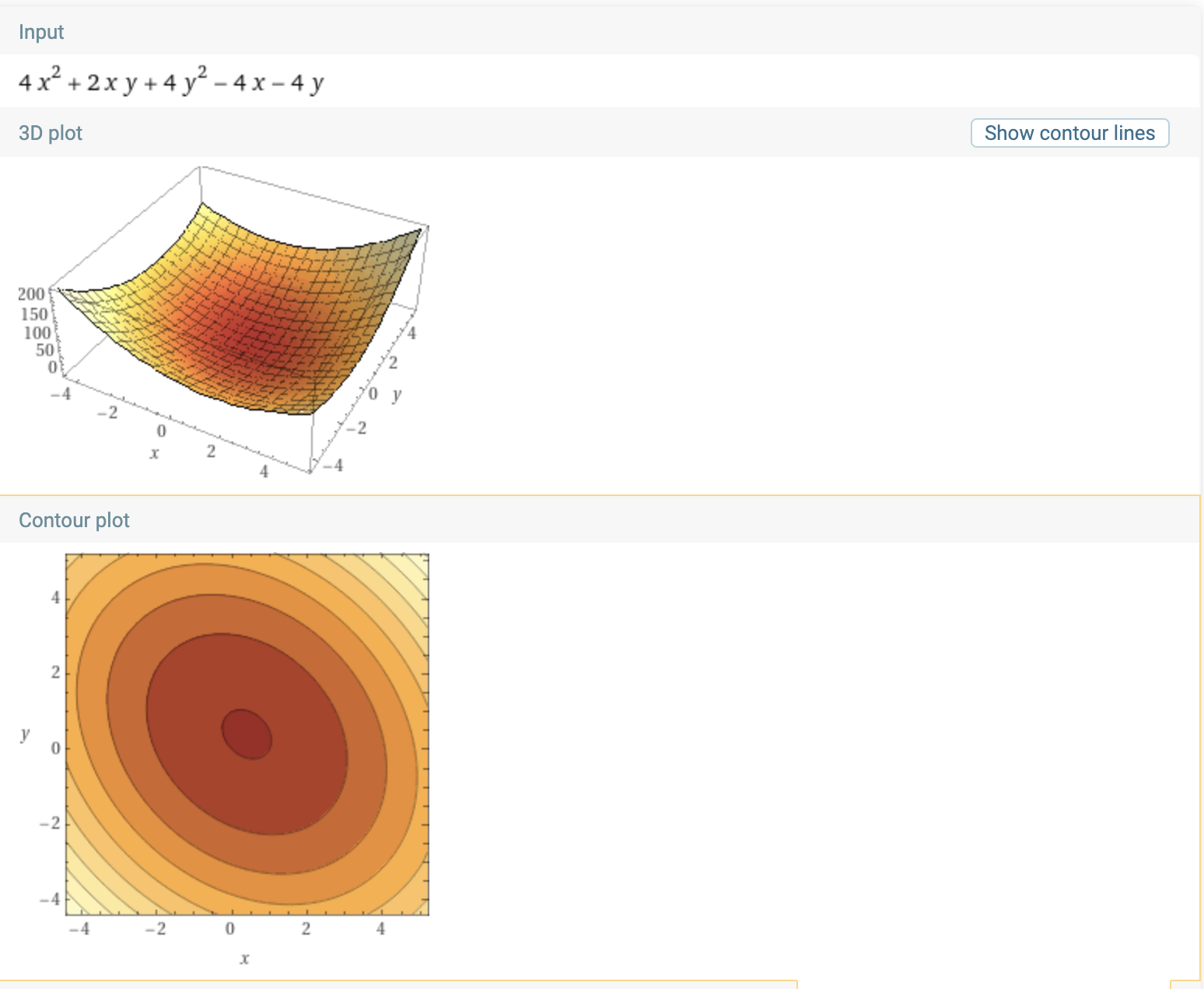
package com.ruby.se.Ruby.metodGrad;  
  
import static java.lang.Math.*pow*;  
import static java.lang.Math.*sqrt*;  
  
public class metodGrad {  
 private double a,b;  
 private double delta, eps, lamda, gamma;  
 private double X, Y;  
 private double z[] = new double[2];  
 private double gradient[] = new double[2];  
 private int k;  
 private double direction[] = new double[2];  
 private double hesana[] = new double[4];  
 int check = 0;  
 private double dirCalc[] = new double[2];  
 private double lastMag;  
 public metodGrad(double i, double j) {  
 k = 0;  
 a = i;  
 b = j;  
 X = a;  
 Y = b;  
 lamda = 1;  
 gamma = 0.5;  
 delta = 0.06;  
 eps = *pow*(10,-6);  
  
 }  
 private double fx(double x, double y) {  
 return a \* *pow*(x,2) + 2 \* x \* y + b \* *pow*(y,2) - b \* x - a \* y;  
 }  
  
 private void gradFxFy() {  
 gradient[0] = a \* 2 \* X + 2 \* Y - b;  
 gradient[1] = 2 \* X + b \* 2 \* Y - a;  
 }  
 private void fZ() {  
 z[0] = X - lamda \* gradient[0];  
 z[1] = Y - lamda \* gradient[1];  
 }  
 private double normGra() {  
 return *sqrt*(*pow*(gradient[0],2) + *pow*(gradient[1],2));  
 }  
 public void result() {  
 gradFxFy();  
 while (normGra() >= eps) {  
 fZ();  
 while (fx(z[0],z[1]) - fx(X, Y) > -delta \* lamda \* *pow*(normGra(),2)) {  
 lamda \*= gamma;  
 fZ();  
 }  
  
 X = z[0];  
 Y = z[1];  
 gradFxFy();  
 k++;  
 System.*out*.println("Coordonatele sunt" + "[" + X + "]" + "[" + Y + "]");  
 System.*out*.println("Resultatul in aceste puncte:" + fx(X,Y));  
 System.*out*.println("Nr inter:" + k);  
  
 }  
 }  
 private void directionXY() {  
 if(check == 0) {  
 direction[0] = -(gradient[0]);  
 direction[1] = -(gradient[1]);  
 } else {  
 double temp = *pow*(normGra(),2) / *pow*(lastMag,2);  
 direction[0] = -(gradient[0]) + direction[0] \* temp;  
 direction[1] = -(gradient[1]) + direction[1] \* temp;  
 }  
 }  
 public void setHeseana(int a ,int b, int c, int d) {  
 hesana[0] = a;  
 hesana[1] = b;  
 hesana[2] = c;  
 hesana[3] = d;  
 }  
 private double firstPart() {  
 return direction[0] \* gradient[0] + direction[1] \* gradient[1];  
 }  
 private void secondPart() {  
 dirCalc[0] = hesana[0] \* direction[0] + hesana[1] \* direction[1];  
 dirCalc[1] = hesana[2] \* direction[0] + hesana[3] \* direction[1];  
 }  
 private double thirdPart() {  
 return dirCalc[0] \* direction[0] + dirCalc[1] \* direction[1];  
 }  
 private double alpha() {  
 return -(firstPart()/thirdPart());  
 }  
  
 public void methodHessen() {  
 double alpha;  
 int i = 0;  
 gradFxFy();  
 while(normGra() > eps) {  
  
 directionXY();  
 lastMag = normGra();  
 firstPart();  
 secondPart();  
 thirdPart();  
 alpha = alpha();  
 X = X + (alpha() \* direction[0]);  
 Y = Y + (alpha() \* direction[1]);  
 System.*out*.println(X + " " + Y);  
 gradFxFy();  
 System.*out*.println("Magnitudinea:" + normGra());  
 System.*out*.println("Minimul:" + fx(X,Y));  
 System.*out*.println(i);  
 i++;  
 check++;  
 }  
 }  
}

******

******

******

***Wolfram***



***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 dintr-o iteratie.