БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №2

**Численное решение систем нелинейных уравнений**

**Выполнил:**

Крючков Василий

2 курс 9 группа

**Преподаватель:**

Горбачева Ю.Н.

Минск, 2022

**Постановка задачи**

Написать программу, которая решает данную систему нелинейных уравнений f (x) = 0 c точностью ε = с помощью метода Ньютона, метода секущих. Начальное приближение выбрать графически. Провести сравнительный анализ полученных результатов.

**Краткие теоретические сведения**

Метод Ньютона:

*,* где­— матрица Якоби

Условие остановки итерационного процесса:

Метод секущих:

— матрица, равная матрице Якоби, в которой заменены на

Условие остановки итерационного процесса:

Метод Гаусса-Зейделя:

*,*

**Листинг программы**

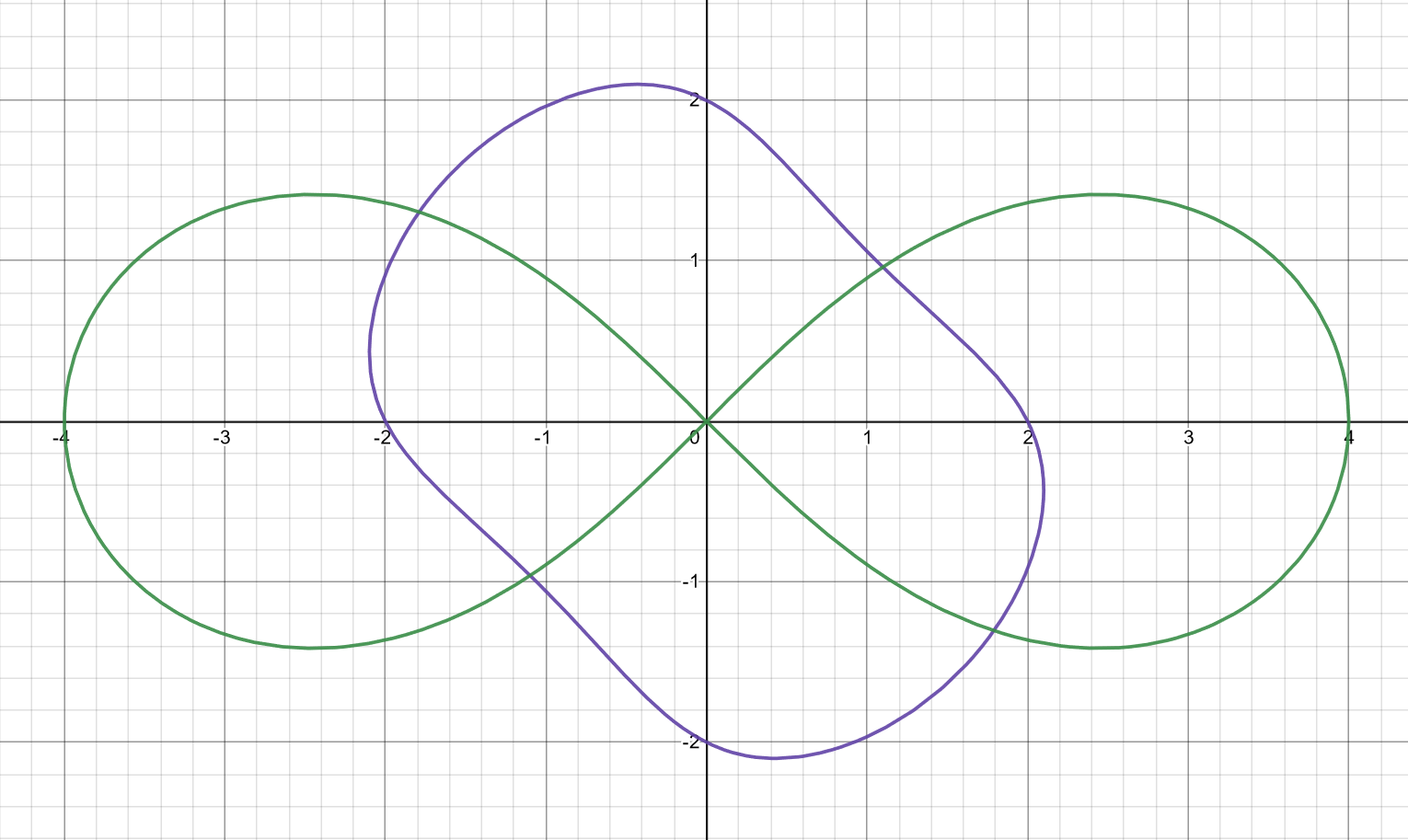
**Lab.java**

import static java.lang.Math.*pow*;  
  
public class Lab {  
 public static void main(String[] args) {  
 Program pr = new Program();  
 try {  
 pr.base();  
 }catch (Error e){  
 System.*out*.println(e.getMessage());  
 }  
 }  
}  
class Program {  
 void base() {  
 double e = 1.0E-6;  
 System.*out*.println();  
 System.*out*.println();  
 System.*out*.println();  
 System.*out*.println();  
 double[] x = new double[2];  
 x[0] = -2;  
 x[1] = 1;  
 methodOfNewton(e, x);  
 System.*out*.println();  
 System.*out*.println();  
 x[0] = -2.5;  
 x[1] = 1;  
 double [] xk = new double[2];  
 xk[0] = -1.9;  
 xk[1] = 0.9;  
 methodOfSecant(e, x, xk);  
 System.*out*.println();  
 System.*out*.println();  
 x[0] = -2;  
 x[1] = 1;  
 methodOfGaussSeidel(e, x);  
 }  
  
 private double f1(double[] x) {return Math.*exp*(x[0]\*x[1])+*pow*(x[0],2)+*pow*(x[1],2)-5;}  
 private double f2(double[] x) {  
 return *pow*(*pow*(x[0],2)+*pow*(x[1],2),2)-16\*(*pow*(x[0],2)-*pow*(x[1],2));  
 }  
 private double f1(double x1, double x2) {return Math.*exp*(x1\*x2)+*pow*(x1,2)+*pow*(x2,2)-5;}  
 private double f2(double x1, double x2) {  
 return *pow*(*pow*(x1,2)+*pow*(x2,2),2)-16\*(*pow*(x1,2)-*pow*(x2,2));  
 }  
 private double df1dx1(double[] x) {return x[1]\*Math.*exp*(x[0]\*x[1])+2\*x[0];}  
 private double df1dx1(double x1, double x2) {return x2\*Math.*exp*(x1\*x2)+2\*x1;}  
 private double df1dx2(double[] x) {  
 return x[0]\*Math.*exp*(x[0]\*x[1])+2\*x[1];  
 }  
 private double df2dx1(double[] x) {  
 return 4\**pow*(x[0],3)+4\*x[0]\**pow*(x[1],2)-32\*x[0];  
 }  
 private double df2dx2(double[] x) {  
 return 4\**pow*(x[1],3)+4\*x[1]\**pow*(x[0],2)+32\*x[1];  
 }  
 private double df2dx2(double x1, double x2) {  
 return 4\**pow*(x2,3)+4\*x2\**pow*(x1,2)+32\*x2;  
 }  
  
 private double norm(double []columnX){  
 double dTemp =0.;  
 for (double x : columnX)  
 if(Math.*abs*(x)>dTemp)  
 dTemp = Math.*abs*(x);  
 return dTemp;  
 }  
  
  
 private void methodOfNewton(double e, double[] x){  
 int k = 0;  
 double[] delX = new double[2];  
 System.*out*.println("Метод Ньютона");  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.1f %.1f\n", x[0], x[1]);  
 do{  
 k++;  
 delX[0] = ((-f1(x))\*df2dx2(x)+df1dx2(x)\*f2(x))/(df1dx1(x)\*df2dx2(x)-df1dx2(x)\*df2dx1(x));  
 delX[1] = ((-df1dx1(x))\*f2(x)+f1(x)\*df2dx1(x))/(df1dx1(x)\*df2dx2(x)-df1dx2(x)\*df2dx1(x));  
 x[0] += delX[0];  
 x[1] += delX[1];  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.16f" + " %.16f ", x[0], x[1]);  
 System.*out*.printf("||x(k) - x(k-1)|| = %.16f\n", norm(delX));  
 }while (norm(delX) > e);  
 double[] fxn = new double[2];  
 fxn[0] = f1(x);  
 fxn[1] = f2(x);  
 System.*out*.printf("||f(x(n))|| = %.16f\n", norm(fxn));  
 }  
  
 private void methodOfSecant(double e, double[] x, double[] xk){  
 int k = 0;  
 double[] delX = new double[2];  
 System.*out*.println("Метод Секущих");  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.1f %.1f\n", x[0], x[1]);  
 k++;  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.1f %.1f\n", xk[0], xk[1]);  
 do{  
 k++;  
 double a11 = (f1(x) - f1(xk[0],x[1]))/(x[0]-xk[0]);  
 double a12 = (f1(x) - f1(x[0],xk[1]))/(x[1]-xk[1]);  
 double a21 = (f2(x) - f2(xk[0],x[1]))/(x[0]-xk[0]);  
 double a22 = (f2(x) - f2(x[0],xk[1]))/(x[1]-xk[1]);  
 delX[0] = ((-f1(x))\*a22+a12\*f2(x))/(a11\*a22-a12\*a21);  
 delX[1] = ((-a11)\*f2(x)+f1(x)\*a21)/(a11\*a22-a12\*a21);  
 xk[0] = x[0];  
 xk[1] = x[1];  
 x[0] += delX[0];  
 x[1] += delX[1];  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.16f" + " %.16f ", x[0], x[1]);  
 System.*out*.printf("||x(k) - x(k-1)|| = %.16f\n", norm(delX));  
 }while (norm(delX) > e);  
 double[] fxn = new double[2];  
 fxn[0] = f1(x);  
 fxn[1] = f2(x);  
 System.*out*.printf("||f(x(n))|| = %.16f\n", norm(fxn));  
 }  
  
 private void methodOfGaussSeidel(double e, double[] x){  
 int k = 0;  
 System.*out*.println("Метод Гаусса Зайделя");  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.1f %.1f\n", x[0], x[1]);  
 double[] fxn = new double[2];  
 do{  
 k++;  
 double xx = x[0];  
 double xs;  
 do{  
 xs = xx;  
 xx -= f1(xx,x[1]) / df1dx1(xx,x[1]);  
 }while(Math.*abs*(xs-xx) > e);  
 x[0] = xx;  
 xx = x[1];  
 do{  
 xs = xx;  
 xx -= f2(x[0],xx) / df2dx2(x[0],xx);  
 }while(Math.*abs*(xs-xx) > e);  
 x[1] = xx;  
 fxn[0] = f1(x);  
 fxn[1] = f2(x);  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.16f" + " %.16f ", x[0], x[1]);  
 System.*out*.printf("||f(x(n))|| = %.16f\n", norm(fxn));  
 }while (norm(fxn) > e);  
 }  
}

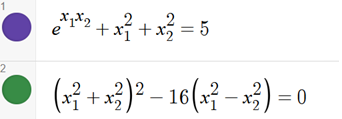
**Результаты**

**Графики, которые использовались для отделения корня**

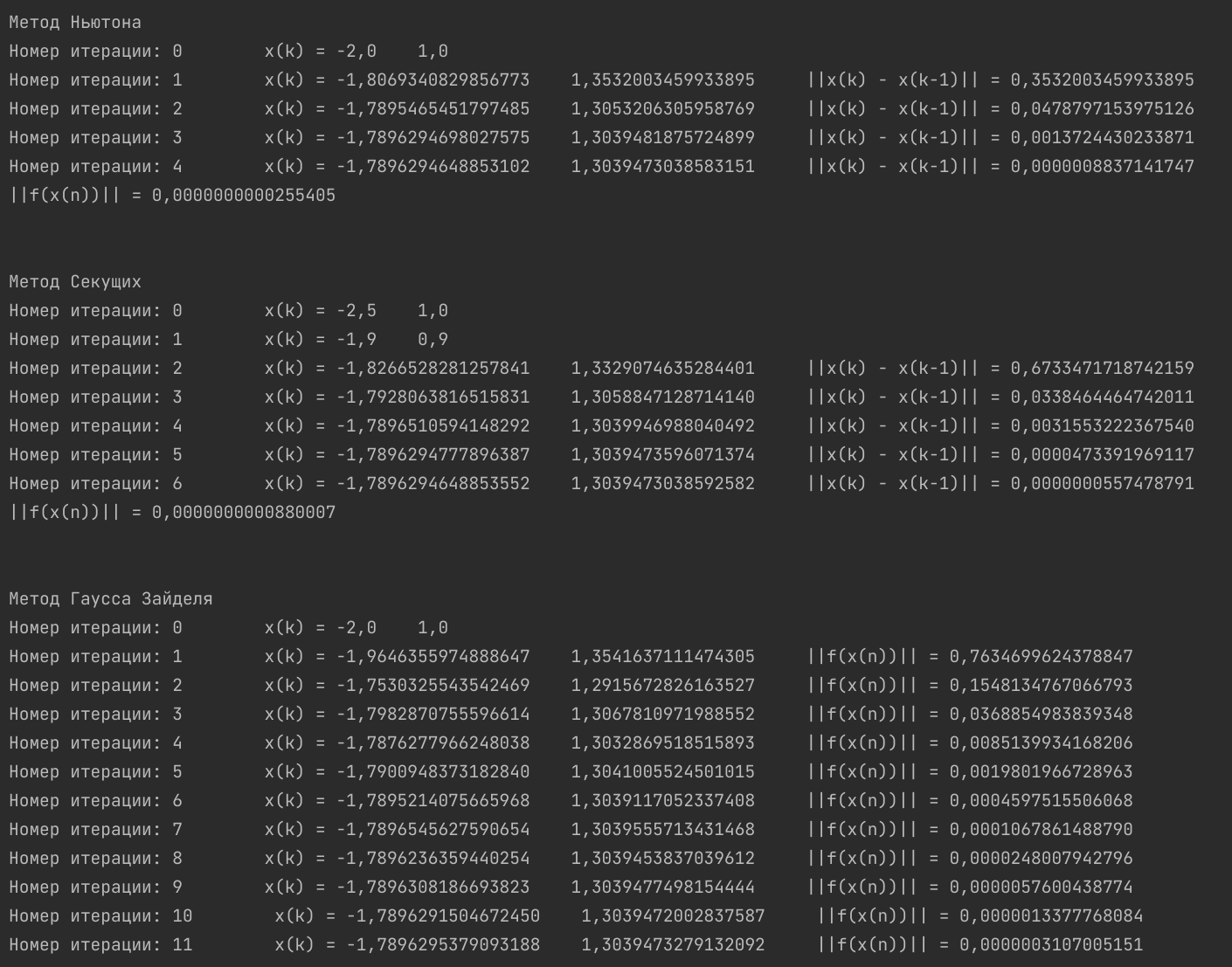
X2

****

X1

****

**Результаты вычисления**

****