**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №2

**Численное решение систем нелинейных уравнений**

**Выполнил:**

Кендысь Алексей Максимович

студент 2 курса, 9 группы,

специальность

“прикладная математика”

**Преподавательница:**

Ассистентка кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 2-4

Листинг программы ---------------------------------------------------------------- 4-8

Результаты --------------------------------------------------------------------------- 9

Выводы ------------------------------------------------------------------------------- 10

**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.

В содержание отчета должна быть включена следующая информация:

• График для нахождения начального приближения .

• Алгоритм метода Ньютона.

• Алгоритм метода секущих.

• Результаты вычислительного эксперимента в виде таблицы. Таблица должна содержать сводные данные по результатам работы метода Ньютона и метода секущих.

• Для каждого метода указать значение , где – полученное решение.

• Листинг программы с комментариями. На 9–10 баллов необходимо также решить систему методом Гаусса-Зейделя (внутренний итерационный процесс – метод Ньютона). В содержание отчета включить алгоритм метода Гаусса-Зейделя и полученные результаты.

Вариант 4.

Система нелинейных уравнений:

**Краткие теоретические сведения**

*Метод Ньютона:*

1)

2)

Условие остановки

*Метод секущих:*

1)

2)

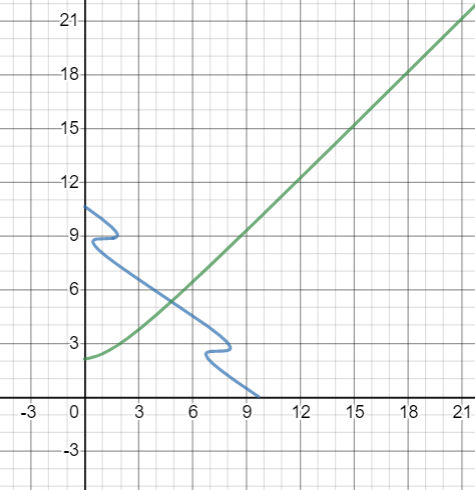
Условие остановки

*Метод Гаусса-Зейделя:*

Внутренний итерационный процесс (метод Ньютона):

Условие остановки

*Выбор начального приближения:*



Начальное приближение .

**Листинг программы**

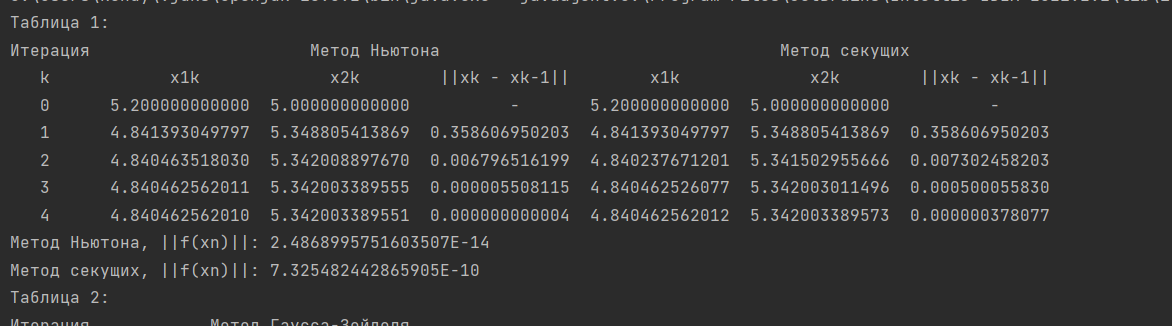
Файл Equation.java:

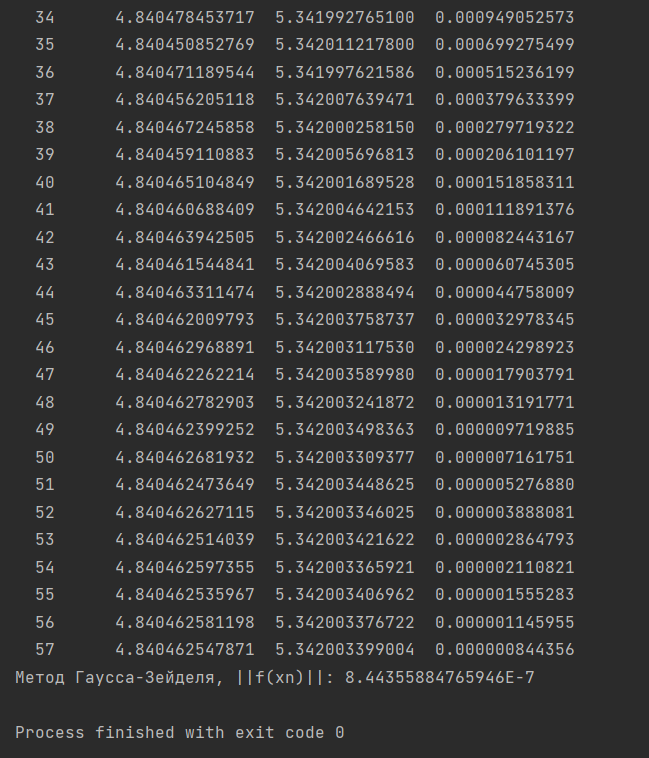
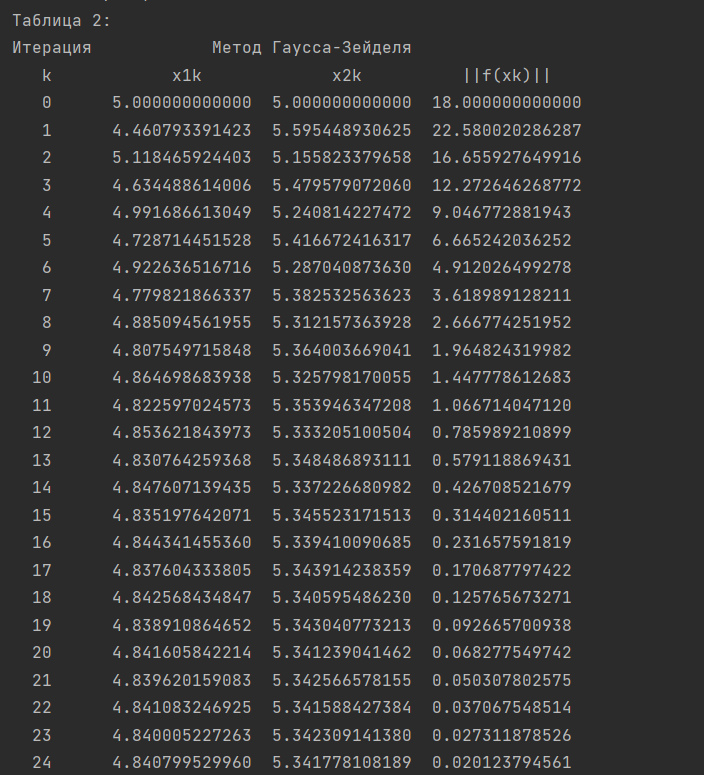
import java.util.\*;  
  
public class Equation {  
 private final static double *E* = 0.000001;  
 private final double[] x0;  
 private double[] nwtX;  
 private double[] secX;  
 private double seidelFxn;  
 private final static int *K\_MAX* = 100;  
  
 public Equation() {  
 x0 = new double[2];  
 x0[0] = 5.2;  
 x0[1] = 5.;  
 }  
  
 private double[] f(double x1, double x2) {  
 //Вектор-функция f(x)  
 double[] f = new double[2];  
 f[0] = - (Math.*sin*(x1 + 2. \* x2) - x1 - x2 + 10.);  
 f[1] = - (3. \* Math.*pow*(x1, 2.) - 4. \* Math.*pow*(x2, 2.) + x1 \* x2 + 18.);  
 return f;  
 }  
  
 private double[][] df(double[] x) {  
 //Матрица df(x)  
 double[][] df = new double[2][2];  
 df[0][0] = Math.*cos*(x[0] + 2 \* x[1]) - 1.;  
 df[0][1] = 2. \* Math.*cos*(x[0] + 2 \* x[1]) - 1.;  
 df[1][0] = 6. \* x[0] + x[1];  
 df[1][1] = x[0] - 8. \* x[1];  
 return df;  
 }  
  
 private double[][] dj(double[] x1, double[] x2) {  
 //Матрица dj(x)  
 double[][] dj = new double[2][2];  
 dj[0][0] = (- f(x2[0], x2[1])[0] + f(x1[0], x2[1])[0]) / (x2[0] - x1[0]);  
 dj[0][1] = (- f(x2[0], x2[1])[0] + f(x2[0], x1[1])[0]) / (x2[1] - x1[1]);  
 dj[1][0] = (- f(x2[0], x2[1])[1] + f(x1[0], x2[1])[1]) / (x2[0] - x1[0]);  
 dj[1][1] = (- f(x2[0], x2[1])[1] + f(x2[0], x1[1])[1]) / (x2[1] - x1[1]);  
 return dj;  
 }  
  
 public void methods() throws NumberFormatException {  
 Formatter fmt = new Formatter();  
 fmt.format("%8s %34s %34s \n", "Итерация", "Метод Ньютона", "Метод секущих");  
 fmt.format("%4s %10s %10s %15s %10s %10s %15s \n", "k", "x1k", "x2k", "||xk - xk-1||", "x1k", "x2k", "||xk - xk-1||");  
 double[] nwt1 = Arrays.*copyOf*(x0, 2);  
 double[] nwtDx = gauss(df(nwt1), f(nwt1[0], nwt1[1]));  
 double[] nwt2 = new double[2];  
 nwt2[0] = nwt1[0] + nwtDx[0];  
 nwt2[1] = nwt1[1] + nwtDx[1];  
 double[] sec1 = Arrays.*copyOf*(x0, 2);  
 double[] secDx = Arrays.*copyOf*(nwtDx, 2);  
 double[] sec2 = Arrays.*copyOf*(nwt2, 2);  
 double[] sec3 = new double[2];  
 int k = 0;  
 fmt.format("%4d % 15.12f % 15.12f %10s % 15.12f % 15.12f %10s \n", k, nwt1[0], nwt1[1], "-", sec1[0], sec1[1], "-");  
 k++;  
 fmt.format("%4d % 15.12f % 15.12f % 15.12f % 15.12f % 15.12f % 15.12f\n", k, nwt2[0], nwt2[1], maxNorm(nwtDx), sec2[0], sec2[1], maxNorm(secDx));  
 k++;  
 while(maxNorm(nwtDx) >= *E* || maxNorm(secDx) >= *E*) {  
 fmt.format("%4d ", k);  
 //Метод Ньютона  
 if(maxNorm(nwtDx) >= *E*) {  
 nwt1 = nwt2;  
 nwtDx = gauss(df(nwt1), f(nwt1[0], nwt1[1]));  
 nwt2[0] = nwt1[0] + nwtDx[0];  
 nwt2[1] = nwt1[1] + nwtDx[1];  
 fmt.format("% 15.12f % 15.12f % 15.12f ", nwt2[0], nwt2[1], maxNorm(nwtDx));  
 } else {  
 fmt.format("%48c", ' ');  
 }  
 //Метод секущих  
 if(maxNorm(secDx) >= *E*) {  
 if(k != 2) {  
 sec1 = sec2;  
 sec2 = Arrays.*copyOf*(sec3, 2);  
 }  
 secDx = gauss(dj(sec1, sec2), f(sec2[0], sec2[1]));  
 sec3[0] = sec2[0] + secDx[0];  
 sec3[1] = sec2[1] + secDx[1];  
 fmt.format("% 15.12f % 15.12f % 15.12f\n", sec3[0], sec3[1], maxNorm(secDx));  
 } else {  
 fmt.format("%47c\n", ' ');  
 }  
 k++;  
 }  
 nwtX = nwt2;  
 secX = sec3;  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
  
 public void outNwt() {  
 System.*out*.println(maxNorm(f(nwtX[0], nwtX[1])));  
 }  
  
 public void outSec() {  
 System.*out*.println(maxNorm(f(secX[0], secX[1])));  
 }  
  
 public void gaussSeidel() {  
 //Метод Гаусса-Зейделя  
 Formatter fmt = new Formatter();  
 fmt.format("%8s %31s \n", "Итерация", "Метод Гаусса-Зейделя");  
 fmt.format("%4s %10s %10s %13s \n", "k", "x1k", "x2k", "||f(xk)||");  
 double[] gs1 = Arrays.*copyOf*(x0, 2);  
 double[] gs2 = Arrays.*copyOf*(x0, 2);  
 int k = 0;  
 fmt.format("%4d % 15.12f % 15.12f % 15.12f\n", k, gs1[0], gs1[1], maxNorm(f(gs1[0], gs1[1])));  
 k++;  
 while(maxNorm(f(gs2[0], gs2[1])) >= *E* && k < *K\_MAX*) {  
 fmt.format("%4d ", k);  
 if (k != 1) {  
 gs1 = Arrays.*copyOf*(gs2, 2);  
 }  
  
 double x1, x2;  
 x1 = gs1[0];  
 x2 = x1 - (- f(x1, gs1[1])[1] / df1(x1, gs1[1]));  
 while(Math.*abs*(x2 - x1) >= *E*) {  
 x1 = x2;  
 x2 = x1 - (- f(x1, gs1[1])[1] / df1(x1, gs1[1]));  
 }  
 gs2[0] = x2;  
  
 x1 = gs1[1];  
 x2 = x1 - (- f(gs2[0], x1)[0] / df2(gs2[0], x1));  
 while(Math.*abs*(x2 - x1) >= *E*) {  
 x1 = x2;  
 x2 = x1 - (- f(gs2[0], x1)[0] / df2(gs2[0], x1));  
 }  
 gs2[1] = x2;  
  
 fmt.format("% 15.12f % 15.12f % 15.12f\n", gs2[0], gs2[1], maxNorm(f(gs2[0], gs2[1])));  
 k++;  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 seidelFxn = maxNorm(f(gs2[0], gs2[1]));  
 if(k == *K\_MAX*) {  
 System.*out*.println("Ошибка. Метод расходится.");  
 }  
 }  
  
 public void outSeidel() {  
 System.*out*.println(seidelFxn);  
 }  
  
 private double df1(double x1, double x2) {  
 //df1 по x1  
 return 6 \* x1 + x2;  
 }  
  
 private double df2(double x1, double x2) {  
 //df2 по x2  
 return 2 \* Math.*cos*(x1 + 2 \* x2) - 1;  
 }  
  
 private double[] gauss(double[][] a, double[] f) throws NumberFormatException {  
 //Метод Гаусса  
 double max;  
 int kMax;  
 double[] temp;  
 double temp2;  
  
 //Прямой ход  
 for (int i = 0; i < a.length; i++) {  
 //Поиск максимального по модулю элемента (выбор главного элемента по столбцу)  
 kMax = i;  
 max = 0;  
 for (int k = i; k < a.length; k++) {  
 if (Math.*abs*(a[k][i]) > Math.*abs*(max)) {  
 max = a[k][i];  
 kMax = k;  
 }  
 }  
 //Проверка на невозможность применения  
 if (max == Double.*MIN\_VALUE*) {  
 System.*out*.println("Ошибка. Деление на ноль.");  
 throw new NumberFormatException();  
 }  
 //Перестановка строк  
 if (max != a[i][i]) {  
 temp = a[i];  
 a[i] = a[kMax];  
 a[kMax] = temp;  
 temp2 = f[i];  
 f[i] = f[kMax];  
 f[kMax] = temp2;  
 }  
  
 //Деление на главный элемент  
 for (int j = i + 1; j < a.length; j++) {  
 a[i][j] /= a[i][i];  
 }  
 f[i] /= a[i][i];  
  
 //Обнуление столбца  
 for (int j = i + 1; j < a.length; j++) {  
 for (int k = i + 1; k < a.length; k++) {  
 a[j][k] -= a[j][i] \* a[i][k];  
 }  
 f[j] -= a[j][i] \* f[i];  
 }  
 }  
  
 //Обратный ход  
 double[] x = new double[a.length];  
 x[a.length - 1] = f[a.length - 1];  
 for (int i = a.length - 2; i >= 0; i--) {  
 x[i] = f[i];  
 for (int j = a.length - 1; j > i; j--) {  
 x[i] -= a[i][j] \* x[j];  
 }  
 }  
  
 return x;  
 }  
  
 private double maxNorm(double[] x) {  
 double max = 0.;  
 for(double i : x) {  
 if(Math.*abs*(i) > max) {  
 max = Math.*abs*(i);  
 }  
 }  
 return max;  
 }  
}

Файл Main.java:

public class Main {  
  
 public static void main(String[] args) {  
 Equation myEquation = new Equation();  
 System.*out*.println("Таблица 1:");  
 myEquation.methods();  
 System.*out*.print("Метод Ньютона, ||f(xn)||: ");  
 myEquation.outNwt();  
 System.*out*.print("Метод секущих, ||f(xn)||: ");  
 myEquation.outSec();  
 System.*out*.println("Таблица 2:");  
 myEquation.gaussSeidel();  
 System.*out*.print("Метод Гаусса-Зейделя, ||f(xn)||: ");  
 myEquation.outSeidel();  
 }  
}

**Результаты**

****

****

**Выводы**

1) При выполнении определённых условий (теорема о сходимости метода Ньютона) метод Ньютона имеют квадратичную скорость сходимости.

2) Из таблицы видно, что метод Ньютона чуть быстрее и точнее метода секущих, но преимущество метода секущих в том, что не требуется явно искать производную функции.

3) Метод Гаусса-Зейделя заведомо сходится при диагональном доминировании в матрице Якоби, поэтому в реализации метода функции в системе считаются в обратном порядке.