**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №1

**Численное решение 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 ------------------------------------------------ 2-6

Листинг программы ---------------------------------------------------------------- 6-8

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

Выводы ------------------------------------------------------------------------------- 9

**Постановка задачи**

Написать программу, которая находит решение уравнения f (x) = 0 c точностью ε = методами, указанными в варианте задания. Корень отделяем сначала графически, затем с помощью метода половинного деления с точностью ε = 0.1. Провести сравнительный анализ полученных результатов. В содержание отчета должна быть включена следующая информация:

• Графики, которые использовались для отделения корня. Отрезок отделенного корня.

• Алгоритм метода половинного деления. Сводные данные по результатам работы метода половинного деления, оформленные в виде таблицы.

• Алгоритмы методов, применяемые для нахождения корня уравнения с заданной точностью ε. Использовать в качестве отрезка отделенного корня суженный отрезок, полученный с помощью метода половинного деления.

• Проверка условий теоремы о сходимости метода простой итерации. Проверка условий теоремы о сходимости метода Ньютона.

• Сводные данные по результатам работы методов, оформленные в виде таблицы.

• Листинг программы с комментариями.

Вариант 4.

Нелинейное уравнение:

Методы: метод простой итерации, метод Ньютона, метод секущих.

**Краткие теоретические сведения**

*Метод половинного деления:*

Если , то . Иначе .

Продолжаем процесс, пока .

*Метод простой итерации:*

Условие остановки

Теорема (о сходимости метода простой итерации). Если:

1) определена на Δ и , где

2)

3)

Тогда:

1)

2) для

3)

4)

*Метод Ньютона:*

Условие остановки

Теорема (о сходимости метода Ньютона). Если:

1)

2)

Тогда:

1)

2)

3)

4)

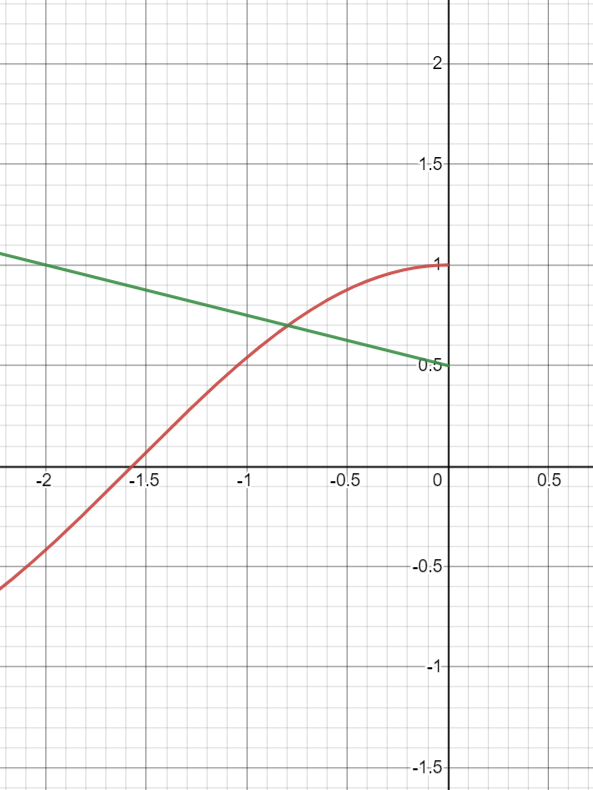
*Метод секущих:*

Условие остановки

*Отделение корня:*

Преобразуем уравнение,

Нарисуем графики левой и правой части.



Отрезок отделённого корня:

Отрезок, полученный методом половинного деления: .

*Построение сходящегося МПИ:*

Т.к. , то, используя , гарантировано получим корень с точностью .

*Проверка условий сходимости метода Ньютона:*

Проверяем

**Листинг программы**

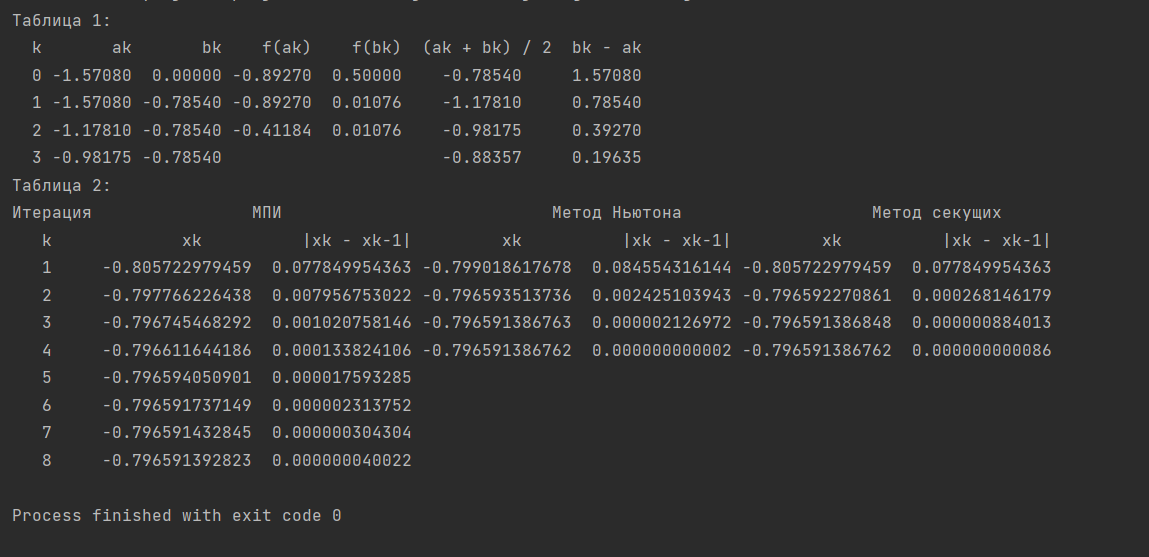
Файл Equation.java:

import java.util.\*;  
  
public class Equation {  
 private double a = - Math.*PI* / 2;  
 private double b = 0;  
 private final static double *E1* = 0.0000001;  
 private final static double *E2* = 0.1;  
 private double x0;  
  
 public void bisection() {  
 //Метод половинного деления  
 Formatter fmt = new Formatter();  
 fmt.format("%3s %8s %8s %8s %8s %14s %8s\n", "k", "ak", "bk", "f(ak)", "f(bk)", "(ak + bk) / 2", "bk - ak");  
 double xK;  
 int k = 0;  
 while(Math.*abs*(b - a) > 2 \* *E2*) {  
 xK = (a + b) / 2;  
 fmt.format("%3d % 8.5f % 8.5f % 8.5f % 8.5f % 11.5f % 8.5f\n", k, a, b, f(a), f(b), xK, b - a);  
 if(f(xK) \* f(a) < 0) {  
 b = xK;  
 } else {  
 a = xK;  
 }  
 k++;  
 }  
 x0 = (a + b) / 2;  
 fmt.format("%3d % 8.5f % 8.5f %8c %8c % 11.5f % 8.5f\n", k, a, b, ' ', ' ', x0, b - a);  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
  
 private double f(double x) {  
 //Функция f(x)  
 return Math.*cos*(x) + 0.25 \* x - 0.5;  
 }  
  
 public void methods() {  
 Formatter fmt = new Formatter();  
 fmt.format("%8s %18s %26s %26s \n", "Итерация", "МПИ", "Метод Ньютона", "Метод секущих");  
 fmt.format("%4s %10s %15s %10s %15s %10s %15s\n", "k", "xk", "|xk - xk-1|", "xk", "|xk - xk-1|" , "xk", "|xk - xk-1|");  
 double fpi1 = x0;  
 double fpi2 = phiFpi(x0);  
 double nwt1 = x0;  
 double nwt2 = phiNewton(x0);  
 double sec1 = x0;  
 double sec2 = fpi2;  
 double sec3 = phiSecant(sec2, sec1);  
 int k = 1;  
 fmt.format("%4d % 15.12f % 15.12f % 15.12f % 15.12f % 15.12f % 15.12f\n", k, fpi2, Math.*abs*(fpi2 - fpi1), nwt2, Math.*abs*(nwt2 - nwt1), sec2, Math.*abs*(sec2 - sec1));  
 k++;  
 while(Math.*abs*(fpi2 - fpi1) >= *E1* || Math.*abs*(nwt2 - nwt1) >= *E1* || Math.*abs*(sec3 - sec2) >= *E1*) {  
 //МПИ  
 fmt.format("%4d ", k);  
 if(Math.*abs*(fpi2 - fpi1) >= *E1*) {  
 fpi1 = fpi2;  
 fpi2 = phiFpi(fpi2);  
 fmt.format("% 15.12f % 15.12f ", fpi2, Math.*abs*(fpi2 - fpi1));  
 } else {  
 fmt.format("%32c", ' ');  
 }  
 //Метод Ньютона  
 if (Math.*abs*(nwt2 - nwt1) >= *E1*) {  
 nwt1 = nwt2;  
 nwt2 = phiNewton(nwt2);  
 fmt.format("% 15.12f % 15.12f ", nwt2, Math.*abs*(nwt2 - nwt1));  
 } else {  
 fmt.format("%32c", ' ');  
 }  
 //Метод секущих  
 if(Math.*abs*(sec3 - sec2) >= *E1*) {  
 sec1 = sec2;  
 sec2 = sec3;  
 sec3 = phiSecant(sec3, sec1);  
 fmt.format("% 15.12f % 15.12f\n", sec3, Math.*abs*(sec3 - sec2));  
 } else {  
 fmt.format("%31c\n", ' ');  
 }  
 k++;  
 }  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
  
 private double phiFpi(double x) {  
 //Функция фи МПИ  
 return x - 0.9 \* f(x);  
 }  
 private double phiNewton(double x) {  
 //Функция фи метода Ньютона  
 return x - f(x) / (- Math.*sin*(x) + 0.25);  
 }  
 private double phiSecant(double x2, double x1) {  
 //Функция фи метода секущих  
 return x2 - f(x2) \* (x2 - x1) / (f(x2) - f(x1));  
 }  
}

Файл Main.java:

public class Main {  
  
 public static void main(String[] args) {  
 Equation myEquation = new Equation();  
 System.*out*.println("Таблица 1:");  
 myEquation.bisection();  
 System.*out*.println("Таблица 2:");  
 myEquation.methods();  
 }  
}

**Результаты**

****

**Выводы**

1) Т.к. в построенном МПИ , то, используя , гарантировано получаем корень с точностью , что видно из результатов. МПИ имеет линейную сходимость.

2) Метод Ньютона является самых точным и самым быстрым методом из трёх при схождении, что видно из полученной невязки. Он обладает квадратичной сходимостью.

3) Метод секущих имеет сверхлинейную сходимость. Также является двухшаговым методом, он менее экономичен, чем метод Ньютона.