БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №1

**Численное решение нелинейных уравнений**

**Выполнил:**

Крючков Василий

2 курс 9 группа

**Преподаватель:**

Горбачева Ю.Н.

Минск, 2022

**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.

**Краткие теоретические сведения**

Метод половинного деления:

После того как точка перемены знака функции, задающей уравнение, найдена, дальнейшее отделение корня, т.е. уменьшение длины отрезка Δ, на котором находится корень, может быть осуществлено с помощью  
метода половинного деления. Опишем его:  
Итак, пусть мы нашли такие точки и , что. Найдем середину отрезка и вычислим Из двух половин отрезка выберем ту, для которой · 𝑓 () < 0. Затем новый отрезок опять делим пополам и выбираем ту половину, на концах которой функция принимает значения разных знаков, и т.д.

Метод простой итерации:

Теорема о сходимости метода итерации. Если:  
1) определена и непрерывна в области   
2) в этой области удовлетворяет условию Липшица:

𝑞 , для любых 𝑥′, 𝑥′′ ∈ Δ с константой 𝑞 < 1  
3) справедливо неравенство:

то последовательность приближений {} по методу итераций с исходным приближением может быть построена и {} сходится к единственному решению.

Метод Cтеффенсена:

Тот факт, что последовательность приближений метода простой итерации близка к геометрической прогрессии, позволяет применить для ускорения ее сходимости преобразование Эйткена.

Получаем итерационный процесс:

Метод Ньютона:

Теорема о сходимости метода Ньютона. Если:

1. функция определена и дважды непрерывно дифференцируема на отрезке вещественной оси с концами в точках и + 2, где , при этом на концах отрезка ;
2. выполняется неравенство , где 𝑀 = ,  
   то:

Последовательность приближений {}, 𝑛 = 1, 2, … сходится к единственному корню 𝑥\* внутри отрезка этого уравнения может быть построена по методу Ньютона с начальным приближением

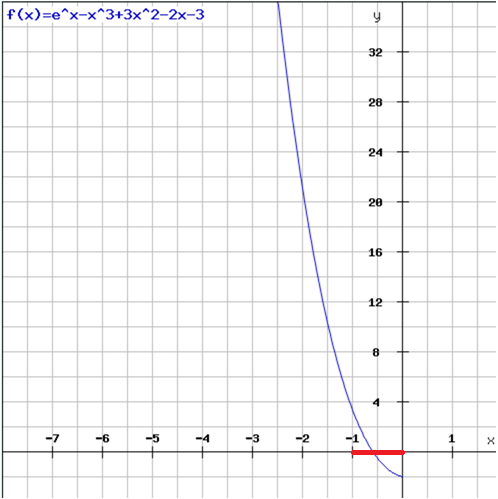
**Листинг программы**

**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-7;  
 double x = methodOfDichotomy(0.1);  
 System.*out*.println();  
 System.*out*.println();  
 methodOfSimpleIteration(e, x, true);  
 System.*out*.println();  
 System.*out*.println();  
 methodOfNewton(e, x);  
 System.*out*.println();  
 System.*out*.println();  
 methodOfStevenson(e, x);  
 }  
  
 private double f(double x) {  
 return Math.*exp*(x)-*pow*(x,3)+3\**pow*(x,2) - 2 \* x - 3;  
 }  
  
 private double df(double x) {  
 return Math.*exp*(x)-3 \* *pow*(x,2)+6\*x - 2;  
 }  
  
 private double sf(double x) {  
 return -Math.*sqrt*((-Math.*exp*(x) + *pow*(x,3) + 2 \* x + 3)/3.0);  
 }  
  
 private double kvsf(double x) {  
 return (-Math.*exp*(x) + *pow*(x,3) + 2 \* x + 3)/3.0;  
 }  
  
 private double methodOfDichotomy(double e) {  
 double a = -1;  
 double b = 0.01;  
 double x = 0;  
 int k = 0;  
 System.*out*.println("Метод дихотомии: ");  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("a(k) = %.1f" + " ", a);  
 System.*out*.printf("b(k) = %.2f" + " ", b);  
 System.*out*.printf("f(a(k)) = %.8f" + " ", f(a));  
 System.*out*.printf("f(b(k)) = %.8f" + " ", f(b));  
 System.*out*.printf("((a(k) + b(k)) / 2 = %.4f" + " ", (a+b)/2);  
 System.*out*.printf("b(k) - a(k) = %.2f\n", b - a);  
 while (Math.*abs*(a-b) > e) {  
 x = (a+b)/2;  
 if (f(a) \* f(x)<= 0)  
 b = x;  
 else  
 a = x;  
 k++;  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("a(k) = %.6f" + " ", a);  
 System.*out*.printf("b(k) = %.6f" + " ", b);  
 System.*out*.printf("f(a(k)) = %.8f" + " ", f(a));  
 System.*out*.printf("f(b(k)) = %.8f" + " ", f(b));  
 System.*out*.printf("((a(k) + b(k)) / 2 = %.6f" + " ", x);  
 System.*out*.printf("b(k) - a(k) = %.8f\n", b - a);  
 }  
 return x;  
 }  
  
 private void methodOfSimpleIteration(double e, double x, boolean b){  
 int k = 0;  
 double xk = Double.*MAX\_VALUE*;  
 System.*out*.println("Метод простой итерации");  
 System.*out*.printf("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.8f\n", x);  
 while ( Math.*abs*(xk-x) > e){  
 k++;  
 xk = x;  
 x = sf(x);  
 if(b){  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.16f" + " ", x);  
 System.*out*.printf("|x(k) - x(k-1)| = %.16f\n", Math.*abs*(xk-x));  
 }  
 if(k>20000)  
 throw new Error("Превышен лимит итераций, метод скорее всего не сходится");  
 }  
 }  
  
 private void methodOfNewton(double e, double x){  
 int k = 0;  
 double xk = Double.*MAX\_VALUE*;  
 System.*out*.println("Метод Ньютона");  
 System.*out*.printf("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.8f\n", x);  
 while ( Math.*abs*(xk-x) > e){  
 k++;  
 if (df(x) == 0) {  
 break;  
 }  
 xk = x;  
 x -= f(x) / df(x);  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.16f" + " ", x);  
 System.*out*.printf("|x(k) - x(k-1)| = %.16f\n", Math.*abs*(xk-x));  
 }  
 }  
  
 private void methodOfStevenson(double e, double x) {  
 int k = 0;  
 double xk = Double.*MAX\_VALUE*;  
 System.*out*.println("Метод Стеффенсена");  
 System.*out*.printf("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.8f\n", x);  
 while (Math.*abs*(xk - x) > e) {  
 k++;  
 xk = x;  
 x = (x \* sf(sf(x)) - kvsf(x)) / (sf(sf(x)) - 2 \* sf(x) + x);  
 System.*out*.print("Номер итерации: " + k + " ");  
 System.*out*.printf("x(k) = %.16f" + " ", x);  
 System.*out*.printf("|x(k) - x(k-1)| = %.16f\n", Math.*abs*(xk - x));  
 }  
 }  
}

**Результаты**

**Графики, которые использовались для отделения корня**

****

**Таблица 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k |  |  |  |  |  |  |
| 0 | -1,0 | 0,01 | 3,36787944 | -2,00965083 | -0,495000 | 1,01 |
| 1 | -1,0 | -0,495000 | 3,36787944 | -0,54406672 | -0,495000 | 0,50500000 |
| 2 | -0,747500 | -0,495000 | 1,06248799 | -0,54406672 | -0,747500 | 0,25250000 |
| 3 | -0,621250 | -0,495000 | 0,17739952 | -0,54406672 | -0,621250 | 0,12625000 |
| 4 | -0,621250 | -0,558125 | 0,17739952 | -0,20310047 | -0,558125 | 0,06312500 |

**Проверка условий теоремы о сходимости метода простой итерации**

определена и непрерывна на

*<0,*

*>0,*

0.64

q =

=

Теорема выполняется

**Проверка условий теоремы о сходимости метода Ньютона**

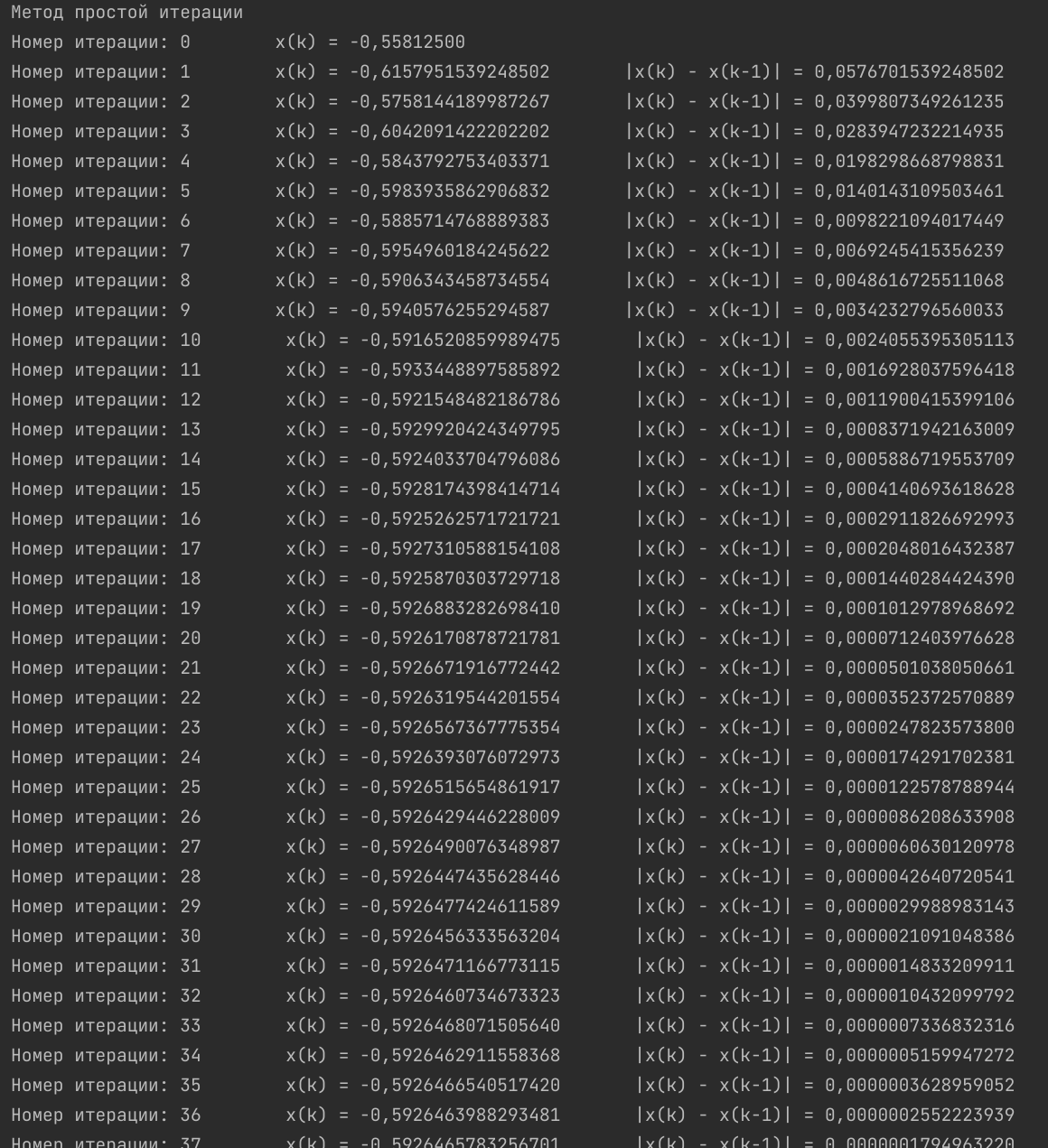
определена и дважды непрерывно дифференцируема на отрезке

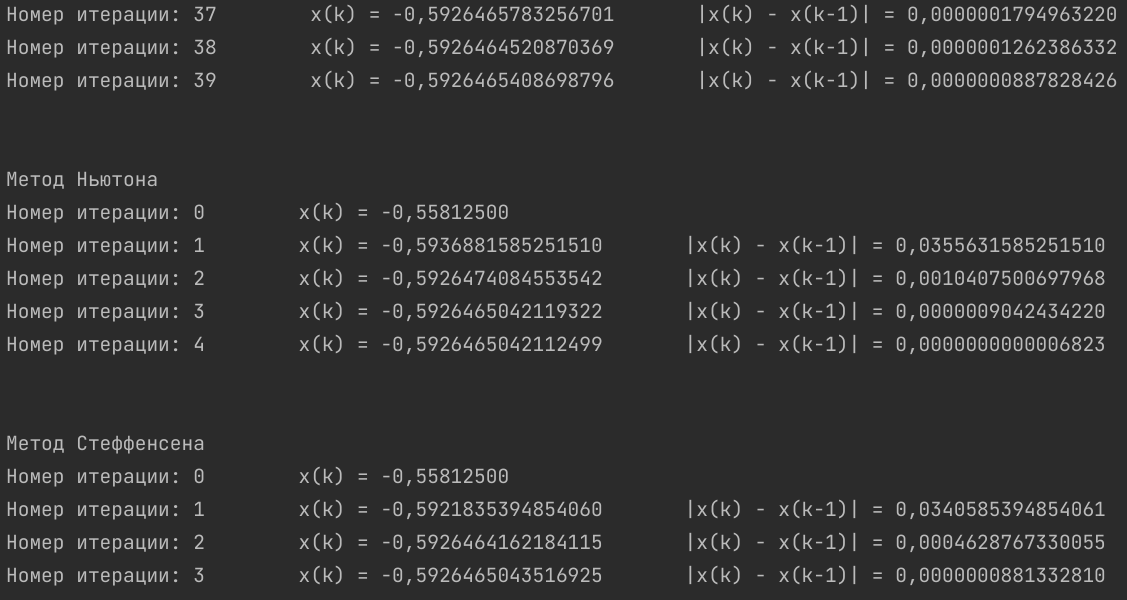
вещественной оси с концами в точках ,

𝑀 =

Теорема выполняется

**Таблица 2**

****

****

**Выводы**

Метод простой итерации является один из базовых итерационных методов нахождения приближенного решения одного численного уравнения вида с заданной точностью, однако не особо эффективен в силу своей линейной скорости сходимости. Для достижения наиболее быстрой скорости сходимости к методу простой итерации могут быть применены преобразования Эйткена, в результате чего и получается метод Стеффенсена, способный выдавать квадратичную сходимость, или еще использовать метод Ньютона, который можно трактовать как метод простой итерации с , дающий квадратичную скорость сходимости вблизи корня.