**Министерство образования и науки Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«Университет ИТМО»**

**Мегафакультет:** Компьютерных технологий и управления

**Направление:** 09.03.04 «Программная инженерия»

**Лабораторная работа 3**

По дисциплине:

«Вычислительная математика»

Вариант 10

На тему:

«Численное решение нелинейных уравнений»

**Выполнила:**

студентка группы **P3214**

Маньшина Елена

Витальевна

**Преподаватель:**

Малышева Татьяна

Алексеевна

Санкт-Петербург

2020

**Численное решение нелинейных уравнений**

**Цель:** изучить методы уточнения приближенных значений действительных корней. Создать программную реализацию методов.

1. Отделение корней заданного нелинейного уравнения графически.
2. Определение интервалов изоляции корней.

Для первого корня:

Для второго корня:

Для третьего корня:

1. Уточнить *крайний правый корень* нелинейного уравнения методом половинного деления с точностью ε=10-2. Вычисления оформить в виде таблицы, удержать 3 знака после запятой (см. табл. 1).

Таблица 1

Уточнение корня уравнения методом половинного деления

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| 0 | 3,700 | 4,000 | 3,850 | -2,620 | 2,458 | -0,271 | 0,300 |
| 1 | 3,850 | 4,000 | 3,925 | -0,271 | 2,458 | 1,045 | 0,150 |
| 2 | 3,850 | 3,925 | 3,888 | -0,271 | 1,045 | 0,375 | 0,075 |
| 3 | 3,850 | 3,888 | 3,869 | -0,271 | 0,384 | 0,054 | 0,038 |
| 4 | 3,850 | 3,869 | 3,860 | -0,271 | 0,054 | -0,109 | 0,019 |
| 5 | 3,860 | 3,869 | 3,865 | -0,101 | 0,054 | -0,024 | 0,009 |

1. Уточнить *крайний левый корень* нелинейного уравнения методом секущих с точностью ε=10-2. Вычисления оформить в виде таблицы, удержать 3 знака после запятой (табл.3).

Интервал изоляции корня

,

*, ->*

*Используемая формула*

Таблица 3

Уточнение корня уравнения методом секущих

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № итерации | *xk-1* | *f*(*xk-1* ) | *xk* | *f*(*xk*) |  | *f*(*xk+1*) | │*xk* − *xk*+1│ |
| *xk*+1 |
|  |
| 1 | -1,600 | -4,038 | -1,500 | -2,698 | -1,299 | -0,457 | 0,201 |
| 2 | -1,500 | -2,698 | -1,300 | -0,469 | -1,258 | -0,073 | 0,042 |
| 3 | -1,300 | -0,470 | -1,258 | -0,075 | -1,250 | -0,003 | 0,008 |

1. Уточнение *центрального корня* нелинейного уравнения методом простой итерации с точностью ε=10-2. Вычисления оформить в виде таблицы, удержать 3 знака после запятой (см. табл. 4).

*,*

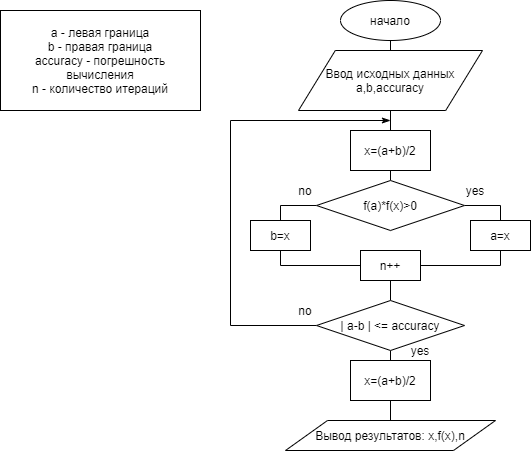
Таблица 4

Уточнение корня уравнения методом простой итерации

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации | *xk* | *f*(*xk* ) | *xk*+1 |  | │*xk* − *xk*+1│ |
| 1 | 0,400 | 0,622 | 0,512 | 0,508 | 0,112 |
| 2 | 0,512 | -0,019 | 0,508 | 0,508 | 0,004 |

Блок-схемы используемых методов, их рабочие формулы.

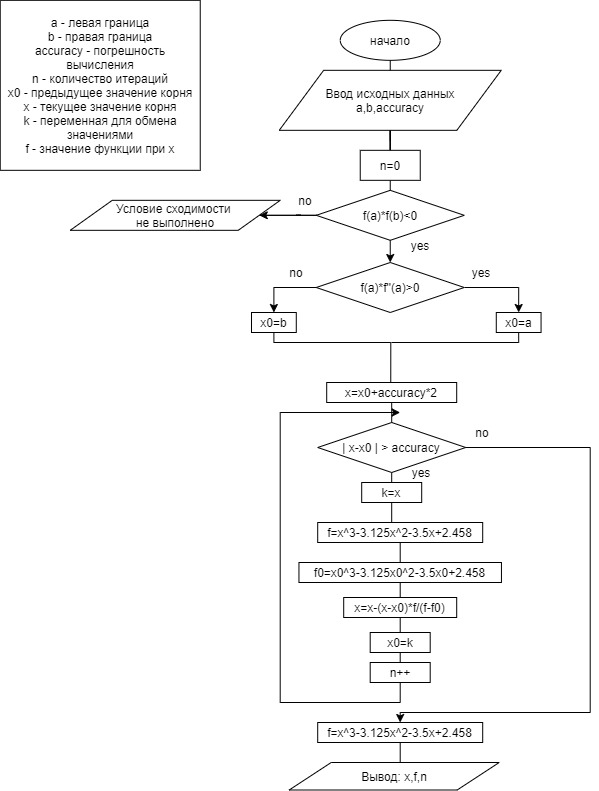
Метод половинного деления



**public class** HalfDivisionMethod {  
  
 **static** String findSolution(**double**[] parameters) **throws** MethodException {  
 **int** n = 0;  
 **double** b = parameters[0];  
 **double** a = parameters[1];  
 **double** accuracy = parameters[2];  
 **double** x;  
 Function<Double, Double> fun = ((Double x0) -> (Math.*pow*(x0, 3) - 3.125 \* Math.*pow*(x0, 2) - 3.5 \* x0 + 2.458));  
  
 **if** (fun.apply(a)\*fun.apply(b)>0){  
 **throw new** MethodException(String.*format*(**"На концах отрезка [%f,%f] функция имеет одинаковые знаки. Значит на отрезке либо нет "** +  
 **"корней, либо их четное количество.Пожалуйста, уточните интервал\n"**,a,b));  
 }  
  
 **do** {  
 x = (b + a) / 2;  
 **if** (fun.apply(x) \* fun.apply(a) > 0) {  
 a = x;  
 } **else** {  
 b = x;  
 }  
 n++;  
 } **while** (Math.*abs*(a - b) > accuracy);  
 x = (b + a) / 2;  
  
 **final** LineChart demo = **new** LineChart(**"Метод половинного деления"**, a, b);  
 demo.pack();  
 RefineryUtilities.*centerFrameOnScreen*(demo);  
 demo.setVisible(**true**);  
  
 **return** String.*format*(**"Выполнено решение уравнения x^(3)-3.125x^(2)-3.5x+2.458 методом половинного деления\n "** +  
 **"Х=%f, количество итераций n=%d, критерий окончания итерационного процесса |a-b|=%f f(%f)=%f"**, x, n, Math.*abs*(a - b), x, fun.apply(x));  
 }  
}

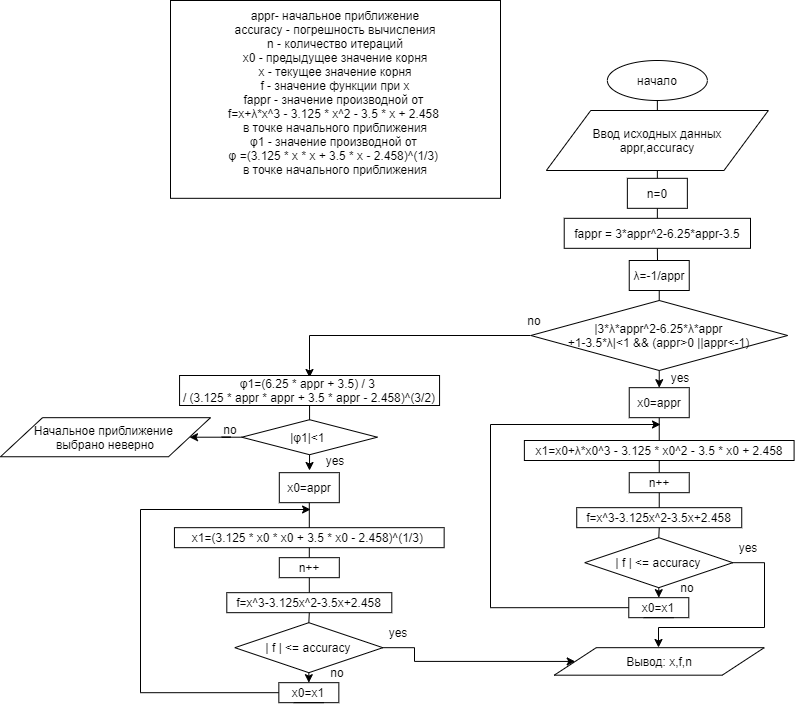
Метод секущих

*Используемая формула*



**public class** SecantMethod {  
 **static** String findSolution(**double**[] parameters) **throws** MethodException {  
 **double** x0;  
 **double** b = parameters[0];  
 **double** a = parameters[1];  
 **double** accuracy = parameters[2];  
 **int** n=0;  
 Function<Double, Double> fun = ((Double x) -> (Math.*pow*(x, 3) - 3.125 \* Math.*pow*(x, 2) - 3.5 \* x + 2.458));  
 *//условие применимости метода Ньютона* **if** (fun.apply(a)\*fun.apply(b)>0){  
 **throw new** MethodException(String.*format*(**"На концах отрезка [%f,%f] функция имеет одинаковые знаки. Значит на отрезке либо нет "** +  
 **"корней, либо их четное количество.Пожалуйста, уточните интервал\n"**,a,b));  
 }  
  
 **if** (fun.apply(a)\*(6\*a-6.25)>0) { *//выбор х0* x0=a;  
 }**else** {  
 x0=b;  
 }  
  
 **double** k;  
 **double** x = x0+accuracy\*2; *//x1 выбирается рядом с х0 самостоятельно* **while** (Math.*abs*(x-x0)>accuracy) {  
 k=x;  
 x = x - (x-x0)\*fun.apply(x)/(fun.apply(x)-fun.apply(x0));  
 x0=k;  
 n++;  
 }  
  
 **final** LineChart demo = **new** LineChart(**"Метод секущих"**, a,b);  
 demo.pack();  
 RefineryUtilities.*centerFrameOnScreen*(demo);  
 demo.setVisible(**true**);  
  
 **return** String.*format*(**"Выполнено решение уравнения x^(3)-3.125x^(2)-3.5x+2.458 методом секущих\n "** +  
 **"Х=%f, количество итераций n=%d, , критерий окончания итерационного процесса |a-b|=%f, f(%f)=%f"**,x,n,Math.*abs*(x-x0),x,fun.apply(x));  
 }  
}

Метод простых итераций



**package** main.java;  
  
**import** org.jfree.ui.RefineryUtilities;  
  
**import** java.util.function.Function;  
  
**public class** SimpleIteration {  
  
 **private double appr**;  
 **private double accuracy**;  
 **private double λ**;  
  
 SimpleIteration(**double**[] parameters) {  
 **this**.**appr** = parameters[0]; *//начальное приближение* **this**.**accuracy** = parameters[1];  
  
 }  
  
 **public** String findSolution() **throws** MethodException {  
  
   
 **double** fappr = 3\*Math.*pow*(**appr**,2)-6.25\***appr**-3.5;  
 **λ**=-1/fappr;  
  
 **if** (Math.*abs*(3\***λ**\*Math.*pow*(**appr**,2)-6.25\***λ**\***appr**+1-3.5\***λ**)<1 && (**appr**>0 ||**appr**<-1)) {  
 **return** getAnswer(((Double x) -> x+**λ**\*(Math.*pow*(x, 3) - 3.125 \* Math.*pow*(x, 2) - 3.5 \* x + 2.458) ));  
 } **else** {  
 **double** φ1 = (6.25 \* **appr** + 3.5) / 3 / Math.*cbrt*(Math.*pow*(3.125 \* **appr** \* **appr** + 3.5 \* **appr** - 2.458, 2));  
 **if** (Math.*abs*(φ1) < 1) {  
 **return** getAnswer((Double x) -> Math.*cbrt*(3.125 \* x \* x + 3.5 \* x - 2.458));  
 } **else** {  
 **throw new** MethodException(**"Начальное приближение выбрано неверно, не выполняется достаточное условие сходимости метода"**);  
 }  
 }  
  
 }  
  
 **private** String getAnswer(Function<Double, Double> fun) {  
 **double** x0 = **appr**;  
 **double** x1;  
 **int** n = 0;  
 **double** f;  
 **while** (**true**) {  
 x1 = fun.apply(x0);  
 n++;  
 f = (Math.*pow*(x1, 3) - 3.125 \* Math.*pow*(x1, 2) - 3.5 \* x1 + 2.458);  
 **if** (Math.*abs*(f) <= **accuracy**) {  
 **break**;  
 }  
 x0 = x1;  
 }  
 **final** LineChart demo = **new** LineChart(**"Метод простых итераций"**, **appr**, x1);  
 demo.pack();  
 RefineryUtilities.*centerFrameOnScreen*(demo);  
 demo.setVisible(**true**);  
  
 **return** String.*format*(**"Выполнено решение уравнения x^(3)-3.125x^(2)-3.5x+2.458 методом простой итерации\n "** +  
 **"Х=%f, количество итераций n=%d, f(%f)=%f"**, x1, n, x1, f);  
  
 }  
}

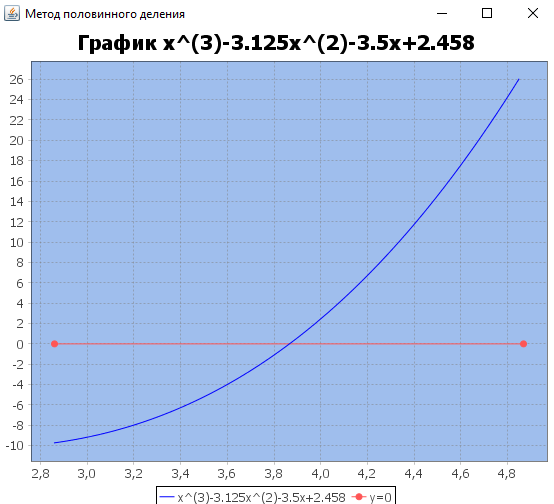
**Листинг программы.**

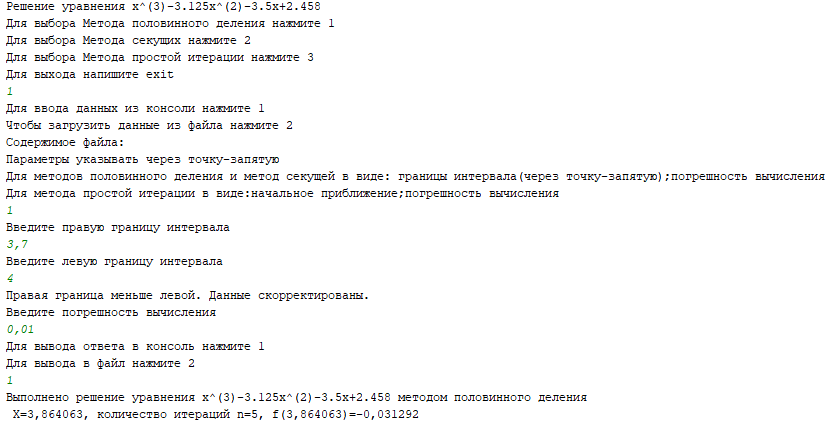
Классы, описывающие каждый из методов расположены под блок-схемами.

Ссылка на гит <https://github.com/Lenika2000/Lab3Math>

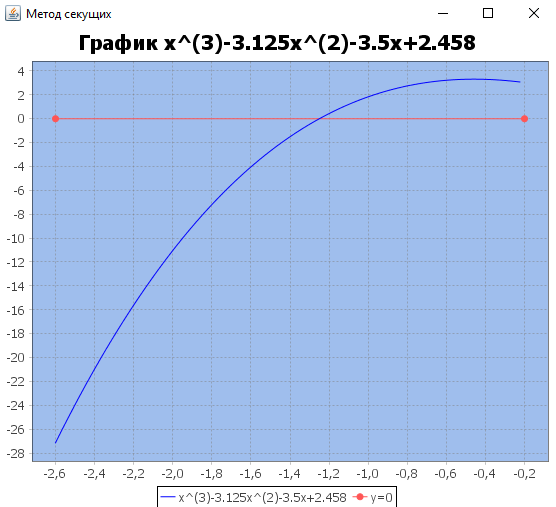
**Результаты выполнения программы.**

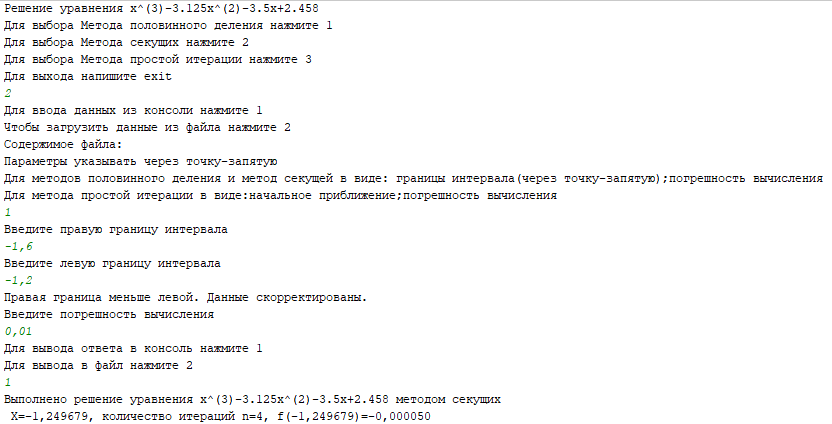
Метод половинного деления:



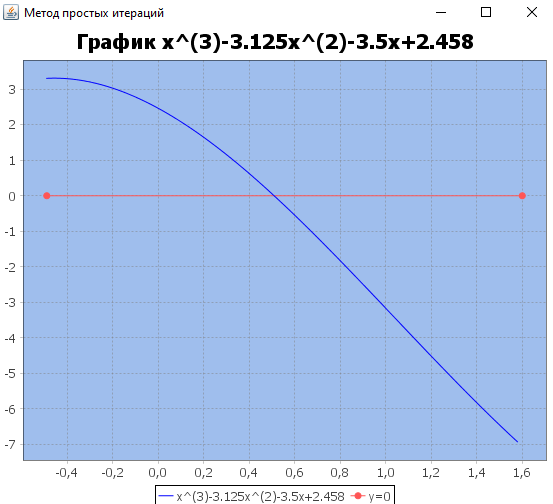


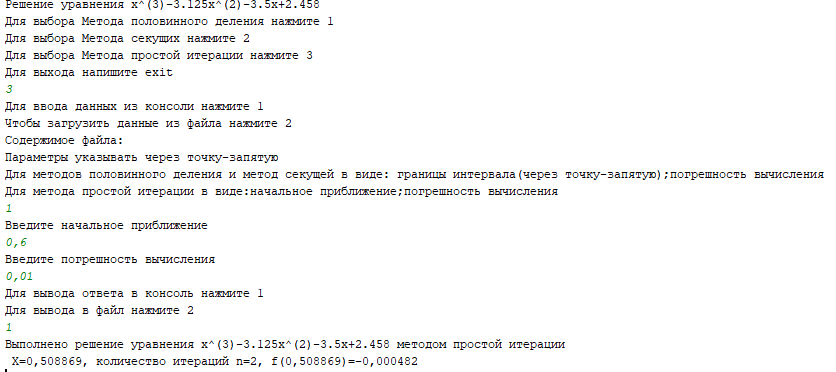
Метод секущих:





Метод простой итерации:





Выводы: В ходе лабораторной работы я познакомилась со способами уточнения приближенных значений действительных корней и создала программную реализацию трех из них