МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Вычислительная математика»

Тема: Метод бисекции.

Студент гр. 8381 _____ Сергеев А.Д.

Преподаватель _____ Щеголева Н.В.

Санкт-Петербург

2019

Цель работы.

Изучить методы решения уравнений итерационным методом бисекции, его точности и быстроты.

Краткое изложение основных теоретических понятий.

Если найден отрезок [a,b], такой, что $f(a)f(b)<0$, существует точка c, в которой значение функции равно нулю, т. е. $f(c)=0$, $c\in(a,b)$. Метод бисекции состоит в построении последовательности вложенных друг в друга отрезков $\{[an,bn]|[an,bn]\subset[an-1,bn-1]\subset[a,b]\}$, на концах которых функция имеет разные знаки. Каждый последующий отрезок получается делением пополам предыдущего. Процесс построения последовательности отрезков позволяет найти нуль функции f(x) (корень уравнения f(x)=0) с любой заданной точностью.

Рассмотрим один шаг итерационного процесса. Пусть на (n−1)-м шаге найден отрезок $[an-1,bn-1]\subset[a,b]$, такой, что $f(an-1)f(bn-1)<0$. Разделим его пополам точкой $\xi=an-1+bn-12$ и вычислим $f(\xi)$. Если $f(\xi)=0$, то $\xi=an-1+bn-12$ – корень уравнения. Если $f(\xi)\neq0$, то из двух половин отрезка выбирается та, на концах которой функция имеет противоположные знаки, поскольку искомый корень лежит на этой половине, т.е.$an=an-1,bn=\xi$, если $f(\xi)f(an-1)<0,an=\xi,bn=bn-1$, если $f(\xi)f(an-1)>0$,Если требуется найти корень с точностью ε, то деление пополам продолжается до тех пор, пока длина отрезка не станет меньше 2ε. Тогда координата середины отрезка есть значение корня с требуемой точностью ε.

Метод бисекции является простым и надежным методом поиска простого корня уравнения f(x)=0 (простым называется корень x=c дифференцируемой функции f(x), если $f(c)=0$ и $f'(c)\neq0$). Этот метод сходится для любых непрерывных функций

f(x), в том числе недифференцируемых. Скорость его сходимости невысока. Для достижения точности ε необходимо совершить $N \approx \log_2(b-a\varepsilon)$ итераций. Это означает, что для получения каждых трех верных десятичных знаков необходимо совершить около 10 итераций.

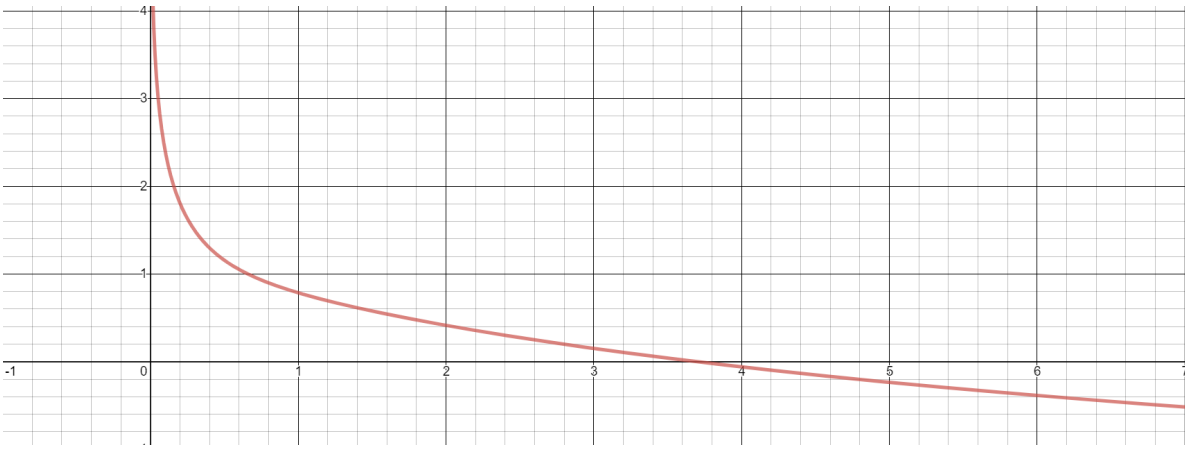
Постановка задачи с кратким описанием порядка выполнения работы.

Необходимо найти единственный корень уравнения, определив отрезок, на котором значение функции принимает разные знаки, после чего сокращать его, используя метод бисекции до тех пор, пока он не станет меньше определённого значения. Затем необходимо проверить, насколько устойчив алгоритм к помехам во входных данных.

Для этого написана программа, находящая корень на определённом отрезке и считающая итерации, после чего повторяющая то же самое действие для функции, искажённой при помощи специального метода round и введённого пользователем параметра delta.

Графическое или аналитическое решение уравнения.

Уравнение *f(x) = arctg(x) — ln(x)* я решил графически, используя онлайн калькулятор desmos для построения графика:



Необходимые графики и таблицы с краткими выводами.

По результатам работы программы была построена такая таблица:

Корень	Отрезок	Эпсилон	Итерации	Эпсилон \ Дельта	V(Дельта)	Дельта	Корень с помехами
3.625	[3;4]	0.1	3	1000	675.811767578, хорошо	0.0001	3.625
3.703125	[3;4]	0.01	6	100	105.438232422, хорошо	0.0001	3.703125
3.693359375	[3;4]	0.001	9	10	7.781982422,	0.0001	3.693359375

					хорошо		
3.6925048828125	[3;4]	0.0001	13	1	1.983642578, нормально	0.0001	3.6923828125
3.692581176757812 5	[3;4]	0.00001	16	0.1	1.983642578, плохо	0.0001	3.6923828125
3.65625	[1;6]	0.1	5	1000	363.311767578, хорошо	0.0001	3.65625
3.675781 25	[1;6]	0.01	8	100	167.999267578, хорошо	0.0001	3.675781 25
3.691650 390625	[1;6]	0.001	12	10	9.307861328, хорошо	0.0001	3.691650 390625
3.692718 505859375	[1;6]	0.0001	15	1	0.152587891, отлично	0.0001	3.692565 91796875
3.692584 991455078	[1;6]	0.00001	18	0.1	0.152587891, хорошо	0.0001	3.692565 91796875

Общий вывод по проделанной работе.

В результате работы была исследована точность и скорость метода бисекции. Как и ожидалось, они возрастают пропорционально. Интересно, что с увеличением отрезка, к которому изначально применяется метод бисекции, незначительно увеличивается его точность. При внесении погрешности во входные данные уменьшается максимальная точность алгоритма: максимальная точность становится примерно равна внесённой погрешности.

Исходный код программы.

Файл Support.java:

```
package classes;



public class Support {

    public static final double MINIMAL_DELTA = 1E-9;



    public static class WrongParameterException extends
Exception {

        private WrongParameterException(String message)
{

            super(message);

        }

    }
```

```
        public static double round(double X, double Delta)
throws WrongParameterException {

            if (Delta <= MINIMAL_DELTA) {

                throw new WrongParameterException("Точность
округления слишком мала: " + Delta + "\n");

            }



            if (X > 0.0) {

                return Delta * (long) (X / Delta + 0.5);

            } else {

                return Delta * (long) (X / Delta - 0.5);

            }

        }



        public static double f(double x) {

            return Math.atan(x) - Math.log(x);
```

```
        }



        public static double f1(double x) {

            return 1/(x*x + 1) - 1/x;

        }



        public static double phi(double x, double signum) {

            return x + signum * f(x);

        }



          public static double bisect(double left, double
right, double eps, boolean isCorrect, double delta) throws
WrongParameterException {

            double E = Math.abs(eps) * 2.0;

                  double  fLeft  =  isCorrect  ?  f(left)  :
round(f(left), delta);

                 double  fRight  =  isCorrect  ?  f(right)  :
round(f(right), delta);
```

```
            double X = 0.5 * (left + right);

            double Y;



            if (fLeft * fRight > 0.0) {

                    throw new WrongParameterException("The
limits are incorrect: " + fLeft + " " + fRight + "\n");

            }



            if (fLeft == 0.0) {

                return left;

            }



            if (fRight == 0.0) {

                return right;

            }
```

```
            int N = 0;

            for (N = 0; right - left >= E; N++) {

                X = 0.5 * (right + left); //     вычисление
середины отрезка

                 //System.out.println("[ " + right + " ; "+
left + " ]");

                Y = isCorrect ? f(X) : round(f(X), delta);

                if (Y == 0.0) {

                                System.out.println("Number  of
iterations: " + N);

                    return X;

                }



                if (Y * fLeft < 0.0) {

                    right = X;

                } else {

                    left = X;

                    fLeft = Y;
```

```
                }
            }

             System.out.println("Number of iterations: " +
N);

            return X;
        }
    }
```

Файл Lab3.java:

```
package classes;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
    /**
     * Function number 24 is:
     *      f(x) = arctg(x) - ln(x)
     *      f'(x) = 1/(x^2 + 1) - 1/x
     *
     */


    public class Lab3 {

          private static final double [] HIGHEST_LIMIT = {4.0, 5.0, 6.0};

          private static final double [] LOWEST_LIMIT  = {3.0, 2.0, 1.0};


        public static void main(String [] args) {

                BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
```

```
                System.out.println("The function is: f(x) =
arctg(x) - ln(x)");

            System.out.println("It has the only root\n");

             System.out.println("Choose the Epsilon between
0.1 and 0.000001:");



            double epsilon;

            try {

                String input1 = in.readLine();

                epsilon = Double.parseDouble(input1);

            } catch (NumberFormatException e) {

                   System.out.println("You've written not a
number");

                return;

            } catch (IOException e) {

                System.out.println("Some error occurs");

                return;

            }
```

```
            if ((epsilon < 0.000001) || (epsilon > 0.1)) {

                 System.out.println("Epsilon is not between
0.1 and 0.000001");

                return;

            }



                System.out.println("\nChoose the scale of
research from 1 to 3:");



            int scale;

            try {

                String input2 = in.readLine();

                scale = Integer.parseInt(input2);

            } catch (NumberFormatException e) {

                  System.out.println("You've written not a
number");

                return;

            } catch (IOException e) {
```

```
                System.out.println("Some error occurs");
                return;
            }
            if ((scale < 1) || (scale > 3)) {
                 System.out.println("Epsilon is not between
1 and 3");
                return;
            }


                System.out.println("\nChoose the delta for
input correction:");


            double delta;
            try {
                String input3 = in.readLine();
                delta = Double.parseDouble(input3);
            } catch (NumberFormatException e) {
```

```
                    System.out.println("You've written not a
number");

                return;

            } catch (IOException e) {

                System.out.println("Some error occurs");

                return;

            }



                        System.out.println("\nResearch  in
progress...");



            double answer;

            try {

                 answer = Support.bisect(LOWEST_LIMIT[scale
- 1], HIGHEST_LIMIT[scale - 1], epsilon, true, delta);

            } catch (Support.WrongParameterException e) {

                   System.out.println("Research terminated
with following exception.");
```

```
                return;

            }

             System.out.println("Research ended! The answer
is: " + answer);



            double wrongAnswer;

            try {

                                                wrongAnswer   =
Support.bisect(LOWEST_LIMIT[scale - 1], HIGHEST_LIMIT[scale
- 1], epsilon, false, delta);

            } catch (Support.WrongParameterException e) {

                    System.out.println("Research terminated
with following exception.");

                return;

            }

              System.out.println("The wrong answer is: " +
wrongAnswer);

        }

    }
```