**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №5**

**по дисциплине «Вычислительная Математика»**

**Тема: Решение нелинейных уравнений методом простых итераций**

| | | |
|---|---|---|
| Студент гр. 8381 | ___________________ | СергеевА.Д. |
| Преподаватель | ___________________ | Щеголева Н.В. |

Санкт-Петербург

2019

**Цель работы.**

**Изучение метода простых итераций для приближенного решения трансцендентного уравнения.**

**Краткое изложение основных теоретических понятий.**

Метод простых итераций решения уравнения $f(x)=0$ состоит в замене исходного уравнения эквивалентным ему уравнением x=φ(x) и построении последовательности $x_{n+1}=\varphi(x_n)$, сходящейся при $n\to\infty$ к точному решению.

Рассмотрим один шаг итерационного процесса. Исходя из найденного на предыдущем шаге значения $x_{n-1}$, вычисляется $y=\varphi(x_{n-1})$. Если $|y-x_{n-1}|>\varepsilon$, то полагается $x_n=y$ и выполняется очередная итерация. Если же $|y-x_{n-1}|<\varepsilon$, то вычисления заканчиваются и за приближенное значение корня принимается величина $x_n=y$.

Погрешность результата вычислений зависит от знака производной $\varphi'(x)$: при $\varphi'(x)>0$ погрешность определения корня составляет qε/1-q, а при $\varphi'(x)<0$ погрешность не превышает ε. Здесь q- число, такое, что $|\varphi'(x)|\leq q<1$ на отрезке [a,b]. Существование числа q является условием сходимости метода в соответствии с отмеченной выше теоремой.

Функцию $\varphi(x)$ необходимо подбирать так, чтобы $|\varphi'(x)|\leq q<1$. Это обусловливается тем, что если $\varphi'(x)<0$ на отрезке [a,b], то последовательные приближения $x_n=\varphi(x_{n-1})$ будут колебаться около корня c, если же $\varphi'(x)>0$, то последовательные приближения будут сходиться к корню c монотонно. Следует также помнить, что скорость сходимости последовательности $\{x_n\}$ к корню c функции $f(x)$ тем выше, чем выше число q.

Итерационный процесс прекращается при $|x_{n+1}-x_n|\leq\dfrac{(1-\alpha_n)\varepsilon}{\alpha_n}$.

**Постановка задачи с кратким описанием порядка выполнения работы.**

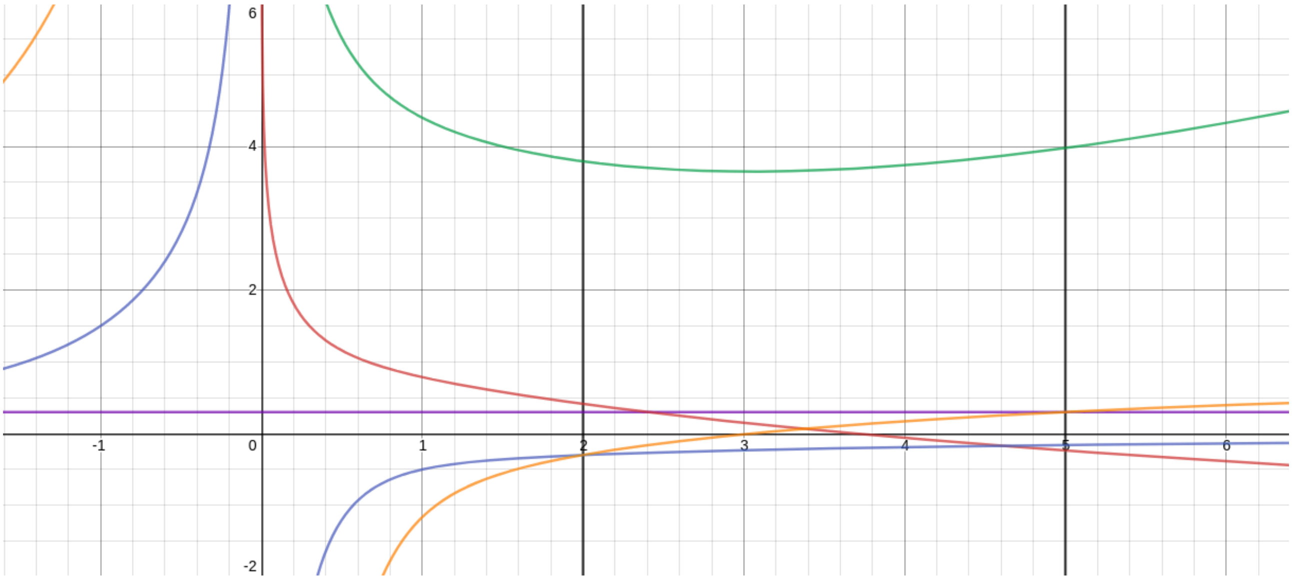
Используя метод Ньютона, требуется найти корень функции $y(x)=arctg(x)-\ln(x)$ с заданной точностью ε и проверить этот метод на скорость сходимости и обусловленность.

Для выполнения работы было принято решение использовать язык программирования *java*. Был модифицирован метод *ITER*, предоставленный на сайте *moevm*, т. к. из-за особенностей выбранного языка перестало быть возможным получение информации о количестве итераций метода, а также появилась необходимость расчёта корня с моделированием помех во входных данных и записи результата в файл для удобства последующей обработки. В его сигнатуру были добавлены *boolean isWrong*, *PrintStream PS* и *double delta*. Внутри функции был добавлен выбор между вызовами метода *f(x)* и *round(f(x), delta)* в зависимости от параметра *isWrong*, который означает, производится вычисление с моделированием помех или без него. Информация о количестве итераций печатается в файл при помощи потока *PrintStream*.

Также мной были написаны методы *collectVariabiles* и *executeResearch*, первый из которых собирает данные об условиях задачи, введённые пользователем или содержащиеся в файле *input.txt*, а второй производит измерения и расчёты для этих условий с разными параметрами, обращаясь к методу *horda*, и записывает их в файл *out.txt*. Исходный код программы содержится в Приложении А.

**Графическое или аналитическое решение уравнения.**

При помощи метода бисекции, описанного ранее, взятого с высокой точностью (ε < $10^{-6}$), был вычислен корень функции $f(x)=0$ при $x=3.6925856854554855\ldots$.




Затем уравнение $y(x)=arctg(x)-\ln(x)$ я также решил графически, используя онлайн-калькулятор *desmos*.

Таким образом был отделён корень уравнения $f(x)=0$, т. е. был найден отрезок $[2,5]$, на котором функция $f(x)$ удовлетворяет условиям теоремы Больцано-Коши.

Уравнение $f(x)=0$ было преобразовано к виду $x=\varphi(x)$ так, чтобы в окрестности $[2,5]$ производная $\varphi'(x)$ удовлетворяла условию $|\varphi'(x)|\leqslant q<1$, где q = $\frac{(M-m)}{(M+m)}$, а $\varphi(x)=x-\frac{2}{(m+M)}f(x)$, где M = -0.162 — максимальное значение производной на отрезке $[2,5]$, а m = -0.3 — минимальное значение производной на отрезке $[2,5]$.

На отрезке $[2,5]$ выбрана точка начального приближения — точка x = 2.

**Необходимые графики и таблицы с краткими выводами.**

По результатам работы программы была построена таблица. В ней представлены зависимость числа итераций функции *ITER* от заданной точности ε, а также от выбранного отрезка, на котором производятся вычисления. Также в таблице представлены сведения об устойчивости метода к ошибкам входных данных, которые были смоделированы с использованием функции *round*, округляющей значения функции с заданной точностью δ.

Для проверки обусловленности задачи было найдено число обусловленности $V_\delta = \frac{|x - x'|}{\delta}$, где $x$ — найденный ранее корень уравнения, $x = 3.6925856854554855\ldots$, а $x'$ — корень, найденный с использованием моделирования погрешностей во входных данных. Будем считать, что задача плохо обусловлена, когда число обусловленности в пять или более раз больше, чем $\frac{\varepsilon}{\delta}$.

| Корень | $x_0$ | ε | Итерации | δ | Корень с помехами | $\frac{\varepsilon}{\delta}$ | $V_\delta$ |
|---|---|---|---|---|---|---|---|
| 3.70571 | 2 | 0.1 | 2 | 0.0001 | 3.79220 | 1000 | 996.1432, хорошо |
| 3.70571 | 2 | 0.01 | 2 | 0.0001 | 3.79220 | 100 | 996.1432, плохо |
| 3.69278 | 2 | 0.001 | 4 | 0.0001 | 3.79220 | 10 | 996.1432, плохо |
| 3.69261 | 2 | 0.0001 | 5 | 0.0001 | 3.79220 | 1 | 996.1432, плохо |
| 3.69259 | 2 | 0.00001 | 6 | 0.0001 | 3.79220 | 0.1 | 996.1432, плохо |

**Скорость схождения метода, сравнение с методом бисекции и хорд.**

Из полученных данных видно, что с уменьшением коэффициента ε, число итераций и точность корня растёт, а также, что при увеличении δ уменьшается точность и обусловленность выходных данных при низкой требуемой точности ε. Аналогично из данной таблицы подтверждается вывод о том, что количество итераций зависит от требуемой точности. Таким образом, теоретические результаты совпадают с экспериментальными данными.

Практические исследования показали, что для выбранной точности метод простых итераций требует в среднем столько же итераций, сколько и метод бисекций и  метод хорд.

Если сравнить обусловленность этих методов, окажется, что, исходя из экспериментальных данных, метод простых итераций обусловлен гораздо хуже, чем метод бисекции, метод хорд и метод Ньютона.

**Общий вывод по проделанной работе.**

В результате проделанной работы, был сделан вывод о том, что число итераций метода простых итераций возрастает с увеличением точности выходных данных, а обусловленность этого метода прямо пропорциональна точности исходных данных и обработано пропорциональна точности вычисления корня.

**Приложение А: исходный код программы.**

**Файл Lab3.java:**

```
package classes;

import java.io.*;

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

public class Lab3 {
    private static final String OVERPATH = "/home/alex/IdeaProjects/labs";

    public static void main(String [] args) {
        try {
            collectVariables(false);
        } catch (FileNotFoundException fnfe) {
            System.out.println("NO INPUT FILE FOUND!");
            fnfe.printStackTrace();
        } catch (Exception e) {
            System.out.println(":/");
            e.printStackTrace();
        }
    }

    public static void collectVariables(boolean isUI) throws Exception {
         BufferedReader in = new BufferedReader(isUI ? new InputStreamReader(System.in) :
new FileReader(OVERPATH + "/buffer/input.txt"));

        int scale = 1;
        double epsilon = 0.000001, trueAnswer = 0;

        if (isUI) {
            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:");

            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:");

        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("\nType the correct answer for wrong answer checking:");

        try {
            String input3 = in.readLine();
            trueAnswer = 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;
        }
    } else {
            PrintStream out = new PrintStream(new FileOutputStream(OVERPATH +
"/buffer/out.txt"));
        trueAnswer = Double.parseDouble(in.readLine());
        for (int i = 0; i < 5; i++) {
            String input = in.readLine();
            int delimPos = input.indexOf(':');
            scale = Integer.parseInt(input.substring(0, delimPos));
            epsilon = Double.parseDouble(input.substring(delimPos + 1));
            executeResearch(scale, epsilon, trueAnswer, 0.0001, out);
        }
        out.flush();
        out.close();
        return;
    }

    executeResearch(scale, epsilon, trueAnswer, 0.0001, System.out);
    in.close();
```

```
    }

    public static void executeResearch(int scale, double epsilon, double trueAnswer, double delta, PrintStream PS) {
        PS.println("\nINITIALIZED RESEARCH FOR:\nScale = " + scale + "\nEpsilon = " + epsilon + "\nTrue answer = " + trueAnswer + "\nDelta = " + delta + "\n");
        PS.println("Research in progress...");

        double answer;
        try {
            answer = Support.ITER(scale, epsilon, PS, false, delta);
        } catch (Support.WrongParameterException e) {
            PS.println("Research terminated with following exception.");
            return;
        }
        PS.println("Research ended! The answer is: " + answer + "\n");

        double wrongAnswer;
        try {
            wrongAnswer = Support.ITER(scale, epsilon, PS, true, delta);
        } catch (Support.WrongParameterException e) {
            PS.println("Research terminated with following exception.");
            return;
        }
        PS.println("The wrong answer is: " + wrongAnswer + "\n");

        double ED = epsilon / delta;
        double Vdel = Math.abs(trueAnswer - wrongAnswer) / delta;
        PS.println("Epsilon/Delta is: " + ED);
        PS.println("V(delta) is: " + Vdel);
        PS.println("Decision: " + ((Vdel < ED * 5) ? ("GOOD") : ("BAD")) + "\n");
        PS.println("\n\n\n");
    }
}
```

### Файл Support.java:

```
package classes;

import java.io.PrintStream;

public class Support {
    public static final double MINIMAL_DELTA = 1E-9;

    private static final double M = -0.162;
    private static final double m = -0.3;

    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 phi(double x) {
        return x - (2/(m+M)) * f(x);
    }

    public static double ITER(double X0, double Eps, PrintStream PS, boolean isWrong,
double delta) throws WrongParameterException {
        int N;

        if (Eps <= 0.0) {
            throw new WrongParameterException("Неверное задание точности\n");
        }

        double X1 = isWrong ? round(phi(X0), delta) : phi(X0);
        double X2 = isWrong ? round(phi(X0), delta) : phi(X1);

        for (N = 2; (X1 - X2) * (X1 - X2) > Math.abs((2 * X1 - X0 - X2) * Eps); N++) {
            X0 = X1;
            X1 = X2;
            X2 = isWrong ? round(phi(X0), delta) : phi(X1);
        }

        PS.println("Iterations number: " + N);
        return X2;
    }
}
```