Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Вычислительная математика»

**Отчет**

По лабораторной работе №2

Вариант 6

Выполнил:

*Манжиков Н.C*

Преподаватель: Рыбаков Степан Дмитриевич

Санкт-Петербург, 2023 г.

Цель работы

Решение уравнений методам хорд, методом секущих и методом простых итераций.

Исследуемое уравнение, согласно варианту: fx= 2x^3+3,41x^2-23,74x+2,95

Рассмотрим график данного уравнения f(x) = 0.

Изображение выглядит как линия, График, Параллельный, число

Автоматически созданное описание

Уравнение имеет три корня, выберем начальные приближения для корней как интервалы:

* Для крайнего левого корня => x0∈[-4.5, -4.4]
* Для центрального корня =>  x1∈[0.1, 0.2]
* Для крайнего правого корня => x2∈[2.5,  2.7]
* Найдём приближения к корням следующими методами: к крайнему левому – методом половинчатого деления, к крайнему правому – методом секущих, к центральному корню – методом простых итераций
* Далее напишем программу, реализующую поиск приближенных значений корней для ряда функций-полиномов и трансцендентных функций.

Для крайнего левого корня в начальном приближении x2∈[-4.5,  -4.4] построим приближение к корню с точностью 0,01 при помощи метода половинчатого деления:

**Уточнение корня уравнения методом половинного деления:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | an | bn | bn-an | xn1 | xn2 | F(xn1) | F(xn2) | εn |
| 1 | -4.5 | -4.4 | -4.45 | -4.4505 | -4.4495 | -0.1551 | -0.09037 | 0.02525 |
| 2 | -4.5 | -4.4505 | 0.0495 | -4.4758 | -4.4748 | -1.8049 | -1.739 | 0.00656 |
| 3 | -4.5 | -4.4758 | 0.02425 | -4.4884 | -4.4874 | -2.641 | -2.5745 | 0.00195 |
| 4 | -4.5 | -4.4884 | 0.01163 | -4.4947 | -4.4937 | -3.0619 | -2.9951 | 0.000832 |
| 5 | -4.5 | -4.4947 | 0.00531 | -4.4978 | -4.4968 | -3.273 | -3.206 | 0.000567 |
| 6 | -4.5 | -4.4978 | 0.00216 | -4.4994 | -4.4984 | -3.3787 | -3.3117 | 0.000509 |
| 7 | -4.5 | -4.4994 | 0.000578 | -4.5002 | -4.4992 | -3.4316 | -3.3646 | 0.000502 |
| 8 | -4.5 | -4.5002 | -0.000211 | -4.5006 | -4.4996 | -3.4581 | -3.391 | 0.000502 |
| 9 | -4.5 | -4.5006 | -0.000605 | -4.5008 | -4.4998 | -3.4714 | -3.4043 | 0.000502 |
| 10 | -4.5 | -4.5008 | -0.000803 | -4.5009 | -4.4999 | -3.478 | -3.4109 | 0.000501 |

x=(-4.500802734375-4.5)/2 = -4.5004013671875.

Для крайнего правого корня в начальном приближении x0∈[2.5, 2.7] построим приближение к корню с точностью 0,01 при помощи метода Ньютона:

**Уточнение корня уравнения методом Ньютона:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N | x | F(x) | dF(x) | h = f(x) / f'(x) |
| 1 | 2.7 | 3.0769 | 38.414 | 0.0801 |
| 2 | 2.6199 | 0.1248 | 35.311 | 0.00353 |

x = 2.6199 - 0.1248 / 35.311 = 2.6163677129826

Для центрального корня в начальном приближении x1∈[0.1, 0.2] построим приближение к корню с точностью 0,01 при помощи метода простых итераций

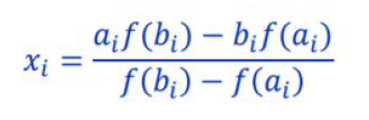
Найдем максимальное значение производной от функции f(x) = 2\*x3+3.41\*x2-23.74\*x+2.95  
max(6•x2+6.82•x-23.74) ≈ -22.136  
Значение λ = 1/(-22.136) ≈ -0.04518  
Таким образом, решаем следующее уравнение:  
x+0.0452\*(2\*x3+3.41\*x2-23.74\*x+2.95) = 0  
F(0.1)=0.612; F(0.2)=-1.646  
Поскольку F(0.1)\*F(0.2)<0 (т.е. значения функции на его концах имеют противоположные знаки), то корень лежит в пределах [0.1;0.2].

|  |  |  |
| --- | --- | --- |
| N | x | F(x) |
| 1 | 0.1 | 0.6121 |
| 2 | 0.1277 | -0.02079 |

Ответ: x = 0.127654678; F(x) = -0.0208

Метод хорд

Метод основан на замене функции на отрезке на хорду. Пересечение хорды с осью абсцисс является приближенным значением корня. Рабочая формула метода:



Особенностью метода хорд является более высокий порядок сходимости в сравнении с методом половинного деления.

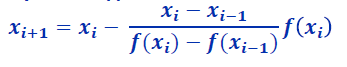
Исследование стандартной функции:

Изображение выглядит как текст, снимок экрана, черный, чек

Автоматически созданное описание

Метод секущих

Метод секущих основан на упрощении метода Ньютона путем замены производной на разностное приближение. Каждое новое приближение определяется предыдущими двумя итерациями. Рабочая формула метода:



Особенностью метода является меньший объем вычислений по сравнению с методом Ньютона, но в то же время порядок сходимости метода секущих ниже. (1.62 против 2 у Ньютона)

Исследование стандартной функции

Изображение выглядит как снимок экрана, черный, текст

Автоматически созданное описание

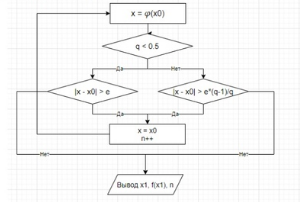
Метод простых итераций

Метод основан на нахождении по приближенному значению величины следующее приближение. Метод позволяет получить решение с заданной точностью в виде последовательности итераций. Рабочая формула метода:



Особенностью метода простых итераций является его сходимость в малой окрестности корня, в следствии чего появляется необходимость выбора изначального приближения к корню.

Блок схема алгоритма



Исследование стандартной функции

Коэффициент сходимости превышает 1: q = 1.7705290490100616

Метод Ньютона

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Вся программа:

import java.io.FileReader;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) throws IOException {  
 CheckInputInterval checkInputInterval = new CheckInputInterval();  
 Scanner inConsole = new Scanner(System.in);  
  
 double a, b, e;  
 int number;  
 boolean file\_flag;  
  
 System.out.println("""  
 Формат входного файла:  
 1. Номер выбранной функции: 1-3  
 2. Коэффициент а  
 3. Коэффициент b  
 4. Точность e  
 5. Метод решения: 2/4/5/6  
 6. Вывод результата в файл/консоль: true/false  
 """);  
  
 System.out.println("Введите: 1 - для чтения файла; 2 - для ввода с консоли");  
 int num = inConsole.nextInt();  
  
 while (!(num == 1 || num == 2)) {  
 System.out.println("Ошибка ввода!");  
 System.out.println("Введите: 1 - для чтения файла; 2 - для ввода с консоли");  
 num = inConsole.nextInt();  
 }  
  
 switch (num) {  
 case 1 -> {  
 FileReader fr = new FileReader("src/main/res/input.txt");  
 Scanner scan = new Scanner(fr);  
  
 number = Integer.parseInt(scan.nextLine().trim());  
 a = Double.parseDouble(scan.nextLine().trim().replaceAll(",", "\\."));  
 b = Double.parseDouble(scan.nextLine().trim().replaceAll(",", "\\."));  
 e = Double.parseDouble(scan.nextLine().trim().replaceAll(",", "\\."));  
  
 int num\_method = Integer.parseInt(scan.nextLine().trim());  
 file\_flag = Boolean.parseBoolean(scan.nextLine().trim());  
  
 if (Math.abs(a - b) < 0.1) {  
 System.out.println("Значения должны быть на расстоянии 0,1 и более");  
 } else {  
 if (checkInputInterval.checkInterval(a, b, number)) {  
 if (checkInputInterval.checkIntervalRoots(a, b, number) > 1) {  
 System.out.println("На интервале должен быть только один корень");  
 System.out.println("Количество корней на интервале: " + checkInputInterval.checkIntervalRoots(a, b, number));  
 } else {  
 System.out.println("Начато выполнение...");  
 switch (num\_method) {  
 case 2 -> {  
 Method2 method2 = new Method2(a, b, e, number, file\_flag);  
 method2.startCalculations();  
  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out2.txt");  
 }  
 }  
 case 4 -> {  
 Method4 method4 = new Method4(a, b, e, number, file\_flag);  
 method4.startCalculations();  
  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out4.txt");  
 }  
 }  
 case 5 -> {  
 Method5 method5 = new Method5(a, b, e, number, file\_flag);  
 method5.startCalculations();  
  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out5.txt");  
 }  
 }  
 case 6 -> {  
 MethodNewton methodNewton = new MethodNewton(a,b,e,number,file\_flag);  
 methodNewton.startCalculations();  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out5.txt");  
 }  
 }  
  
 case 1, 3 -> System.out.println("Программа не поддерживает этот метод решения");  
 }  
 }  
 } else {  
 System.out.println("В выбранном интервале нет корней");  
 }  
 }  
 }  
  
  
 case 2 -> {  
 while (true) {  
 int answer = 0;  
 while (true) {  
 System.out.println("Введите номер исследуемой функции (1-3): ");  
 try {  
 number = Integer.parseInt(inConsole.next().trim().replaceAll(",", "\\."));  
 break;  
 } catch (NumberFormatException ignored) {  
 System.out.println("Повторите ввод");  
 }  
 }  
 while (true) {  
 System.out.println("Введите a: ");  
 try {  
 a = Double.parseDouble(inConsole.next().trim().replaceAll(",", "\\."));  
 break;  
 } catch (NumberFormatException ignored) {  
 System.out.println("Повторите ввод");  
 }  
 }  
 while (true) {  
 System.out.println("Введите b: ");  
 try {  
 b = Double.parseDouble(inConsole.next().trim().replaceAll(",", "\\."));  
 break;  
 } catch (NumberFormatException ignored) {  
 System.out.println("Повторите ввод");  
 }  
 }  
 while (true) {  
 System.out.println("Введите точность e:");  
 try {  
 e = Double.parseDouble(inConsole.next().trim().replaceAll(",", "\\."));  
 break;  
 } catch (NumberFormatException ignored) {  
 System.out.println("Повторите ввод");  
 }  
 }  
  
 if (Math.abs(a - b) < 0.1) {  
 System.out.println("Значения должны быть на расстоянии 0,1 и более");  
 } else {  
 if (checkInputInterval.checkInterval(a, b, number)) {  
 if (checkInputInterval.checkIntervalRoots(a, b, number) > 1) {  
 System.out.println("На интервале должен быть только один корень");  
 System.out.println("Количество корней на интервале: " +  
 checkInputInterval.checkIntervalRoots(a, b, number));  
 } else {  
 System.out.println("""  
 Выберите метод решения:  
 2. Метод хорд  
 4. Метод секущих  
 5. Метод простой итерации""");  
 answer = inConsole.nextInt();  
 do {  
 switch (answer) {  
  
 case 2 -> {  
 System.out.println("Введите: 1 - для вывода в файл; 2 - для вывода в консоль");  
 int flag = inConsole.nextInt();  
 file\_flag = flag == 1;  
  
 Method2 method2 = new Method2(a, b, e, number, file\_flag);  
 method2.startCalculations();  
  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out2.txt");  
 }  
 }  
 case 4 -> {  
 System.out.println("Введите: 1 - для вывода в файл; 2 - для вывода в консоль");  
 int flag = inConsole.nextInt();  
 file\_flag = flag == 1;  
  
 Method4 method4 = new Method4(a, b, e, number, file\_flag);  
 method4.startCalculations();  
  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out4.txt");  
 }  
 }  
 case 5 -> {  
 System.out.println("Введите: 1 - для вывода в файл; 2 - для вывода в консоль");  
 int flag = inConsole.nextInt();  
 file\_flag = flag == 1;  
  
 Method5 method5 = new Method5(a, b, e, number, file\_flag);  
 method5.startCalculations();  
  
 if (file\_flag) {  
 System.out.println("Результаты записаны в файл out5.txt");  
 }  
 }  
 case 1, 3 -> System.out.println("Программа не поддерживает этот метод решения");  
 }  
 System.out.println("Если хотите запустить другой метод решения, введите его номер, иначе введите ноль: ");  
 answer = inConsole.nextInt();  
 } while (answer == 2 || answer == 4 || answer == 5);  
  
 }  
 } else {  
 System.out.println("В выбранном интервале нет корней");  
 break;  
 }  
 }  
 if (answer == 0)  
 break;  
 }  
 }  
 }  
 }  
}

import java.io.FileWriter;  
import java.io.IOException;  
  
public class Method2 { //Метод хорд  
 DrawChart drawChart = new DrawChart();  
 PrintTable printTable = new PrintTable();  
 Functions functions = new Functions();  
  
 double a, b, e, start\_X, end\_X;  
 int iterations = 0, number;  
 double current\_x = 1, last\_x = 1;  
 boolean flag\_file;  
  
 public Method2(double a, double b, double e, int number, boolean flag\_file) {  
 this.a = a;  
 this.b = b;  
 this.e = e;  
 this.number = number;  
 this.flag\_file = flag\_file;  
 }  
  
 public void startCalculations() throws IOException {  
 FileWriter writer = null;  
  
 if (flag\_file) {  
 writer = new FileWriter("src/main/res/out2.txt", false);  
 writer.write("Запущен метод хорд\n" +  
 "Решение уравнения на интервале [ " + a + "; " + b + " ]\nС точностью: " + e + "\n\n");  
 } else {  
 System.out.println("Запущен метод хорд\n" +  
 "Решение уравнения на интервале [ " + a + "; " + b + " ]\nС точностью: " + e + "\n\n");  
 }  
  
 start\_X = functions.setStartX(a, b, number);  
 boolean flagA = false;  
 if (start\_X == a) {  
  
 } else {  
 flagA = true;  
 }  
 if (start\_X == a)  
 end\_X = b;  
 else  
 end\_X = a;  
  
 if (!flag\_file)  
 printTable.printHeaderFor2inConsole();  
 else  
 writer.write(printTable.printHeaderFor2inFile());  
  
 while (Math.abs(functions.f(current\_x, number)) > e) {  
 current\_x = functions.x0(start\_X, end\_X, number);  
 if (iterations == 0) {  
 if (!flag\_file)  
 printTable.printLineFor2inConsole(iterations + 1, start\_X, end\_X, current\_x, functions.f(start\_X, number),  
 functions.f(end\_X, number), functions.f(current\_x, number), 0);  
 else  
 writer.write(printTable.printLineFor2inFile(iterations + 1, start\_X, end\_X, current\_x,  
 functions.f(start\_X, number), functions.f(end\_X, number), functions.f(current\_x, number), 0));  
 } else {  
 if (!flag\_file)  
 printTable.printLineFor2inConsole(iterations + 1, start\_X, end\_X, current\_x, functions.f(start\_X, number),  
 functions.f(end\_X, number), functions.f(current\_x, number), Math.abs(last\_x - current\_x));  
 else  
 writer.write(printTable.printLineFor2inFile(iterations + 1, start\_X, end\_X, current\_x,  
 functions.f(start\_X, number), functions.f(end\_X, number), functions.f(current\_x, number),  
 Math.abs(last\_x - current\_x)));  
 }  
  
 if (functions.f(start\_X, number) > 0 && functions.f(current\_x, number) < 0 || functions.f(start\_X, number) < 0  
 && functions.f(current\_x, number) > 0) {  
 end\_X = current\_x;  
 } else if (functions.f(end\_X, number) > 0 && functions.f(current\_x, number) < 0 || functions.f(end\_X, number) < 0  
 && functions.f(current\_x, number) > 0) {  
 start\_X = current\_x;  
 }  
  
 last\_x = current\_x;  
 iterations++;  
 }  
  
 if (!flag\_file) {  
 System.out.println("\nКорень уравнения:");  
 System.out.printf("x\* = %.13f%n", current\_x);  
 System.out.printf("f(x\*) = %.13f%n", functions.f(current\_x, number));  
 } else {  
 writer.write("\nКорень уравнения:\n");  
 writer.write(String.format("x\* = %.13f%n\n", current\_x));  
 writer.write(String.format("f(x\*) = %.13f%n", functions.f(current\_x, number)));  
 writer.flush();  
 }  
  
 drawChart.draw(Math.round(a), Math.round(b), number);  
 }  
}

import java.io.FileWriter;  
import java.io.IOException;  
  
public class Method4 { //Метод секущих  
 DrawChart drawChart = new DrawChart();  
 PrintTable printTable = new PrintTable();  
 Functions functions = new Functions();  
  
 double a, b, e, start\_X, next\_X, last\_x;  
 int iterations = 0, number;  
 boolean flag\_file;  
  
  
 public Method4(double a, double b, double e, int number, boolean flag\_file) {  
 this.a = a;  
 this.b = b;  
 this.e = e;  
 this.number = number;  
 this.flag\_file = flag\_file;  
 }  
  
  
 public void startCalculations() throws IOException {  
 FileWriter writer = null;  
  
 if (flag\_file) {  
 writer = new FileWriter("src/main/res/out4.txt", false);  
 writer.write("Запущен метод секущих\n" +  
 "Решение уравнения на интервале [ " + a + "; " + b + " ]\nС точностью: " + e + "\n\n");  
 } else {  
 System.*out*.println("Запущен метод секущих\n" +  
 "Решение уравнения на интервале [ " + a + "; " + b + " ]\nС точностью: " + e + "\n\n");  
 }  
  
 next\_X = functions.setStartX(a, b, number);  
 last\_x = next\_X;  
  
 if (!flag\_file)  
 printTable.printHeaderFor4inConsole();  
 else  
 writer.write(printTable.printHeaderFor4inFile());  
  
  
 while (Math.*abs*(functions.f(start\_X, number)) > e) {  
 start\_X = next\_X;  
 if (iterations == 0)  
 next\_X = start\_X - functions.f(start\_X, number) / functions.f\_(start\_X, number); // вот sus  
 else  
 next\_X = functions.setNextX(start\_X, last\_x, number);  
  
 if (iterations == 0) {  
 if (!flag\_file)  
 printTable.printLineFor4InConsole(iterations + 1, 0, 0, start\_X,  
 functions.f(start\_X, number), next\_X, functions.f(next\_X, number), 0);  
 else  
 writer.write(printTable.printLineFor4inFile(iterations + 1, 0, 0, start\_X,  
 functions.f(start\_X, number), next\_X, functions.f(next\_X, number), 0));  
 } else {  
 if (!flag\_file)  
 printTable.printLineFor4InConsole(iterations + 1, last\_x, functions.f(last\_x, number), start\_X,  
 functions.f(start\_X, number), next\_X, functions.f(next\_X, number), Math.*abs*(next\_X - start\_X));  
 else  
 writer.write(printTable.printLineFor4inFile(iterations + 1, last\_x, functions.f(last\_x, number),  
 start\_X, functions.f(start\_X, number), next\_X, functions.f(next\_X, number), Math.*abs*(next\_X - start\_X)));  
 }  
 last\_x = start\_X;  
 iterations++;  
 }  
  
 if (!flag\_file) {  
 System.*out*.println("\nКорень уравнения:");  
 System.*out*.printf("x\* = %.10f%n", start\_X);  
 System.*out*.printf("f(x\*) = %.10f%n", functions.f(start\_X, number));  
 } else {  
 writer.write("\nКорень уравнения:\n");  
 writer.write(String.*format*("x\* = %.10f%n", start\_X));  
 writer.write(String.*format*("f(x\*) = %.10f%n", functions.f(start\_X, number)));  
 writer.flush();  
 }  
 drawChart.draw(Math.*round*(a), Math.*round*(b), number);  
 }  
}

import java.io.FileWriter;  
import java.io.IOException;  
  
public class Method5 { //Метод простых итераций  
 DrawChart drawChart = new DrawChart();  
 PrintTable printTable = new PrintTable();  
 Functions functions = new Functions();  
  
 double a, b, e, start\_X, last\_X, q, abs;  
 double lambda = 1;  
 int iterations = 0, number;  
 boolean flag\_file;  
  
 public Method5(double a, double b, double e, int number, boolean flag\_file) {  
 this.a = a;  
 this.b = b;  
 this.e = e;  
 this.number = number;  
 this.flag\_file = flag\_file;  
 }  
  
 public void startCalculations() throws IOException {  
 FileWriter writer = null;  
  
 if (flag\_file) {  
 writer = new FileWriter("src/main/res/out5.txt", false);  
 writer.write("Запущен метод простых итераций\n" +  
 "Решение уравнения на интервале [ " + a + "; " + b + " ]\nС точностью: " + e + "\n\n");  
 writer.flush();  
 }else{  
 System.*out*.println("Запущен метод простых итераций\n" +  
 "Решение уравнения на интервале [ " + a + "; " + b + " ]\nС точностью: " + e + "\n\n");  
 }  
  
  
 lambda = -1 / functions.getMaxF\_(a, b, number);  
 last\_X = functions.setStartX(a, b, number);  
 start\_X = functions.Fi(last\_X, lambda, number);  
  
 q = functions.getMaxQ(a,b,lambda, number);  
  
 if (q<1){  
 if(q>0.5 && q < 1){  
 e = (1-q) \* e / q;  
 if (!flag\_file) {  
 System.*out*.println("Критерий окончания: |Xn - Xn-1| > (1-q)\*e/q ");  
 } else {  
 writer.write("Критерий окончания: |Xn - Xn-1| > (1-q)\*e/q \n\n");  
 writer.flush();  
 }  
 }  
 else if (q >0 && q <= 0.5) {  
 if (!flag\_file) {  
 System.out.println("Критерий окончания: |Xn - Xn-1| > e ");  
 } else {  
 writer.write("Критерий окончания: |Xn - Xn-1| > e ");  
 }  
 }  
  
 if (!flag\_file)  
 printTable.printHeaderFor5inConsole();  
 else  
 writer.write(printTable.printHeaderFor5inFile());  
  
  
 abs = Math.abs(start\_X-last\_X);  
  
 while(abs > e)  
 {  
 if (!flag\_file) {  
 printTable.printLineFor5inConsole(iterations+1, last\_X, start\_X,  
 functions.Fi(start\_X, lambda, number), functions.f(start\_X, number), Math.*abs*(start\_X - last\_X));  
 } else {  
 writer.write(printTable.printLineFor5inFile(iterations+1, last\_X, start\_X,  
 functions.Fi(start\_X, lambda, number), functions.f(start\_X, number), Math.*abs*(start\_X - last\_X)));  
 }  
  
 last\_X = start\_X;  
 start\_X = functions.Fi(last\_X, lambda, number);  
 abs = Math.*abs*(start\_X - last\_X);  
 iterations++;  
 }  
  
 if (!flag\_file) {  
 printTable.printLineFor5inConsole(iterations+1, last\_X, start\_X, functions.Fi(start\_X, lambda, number),  
 functions.f(start\_X, number), Math.*abs*(start\_X - last\_X));  
 } else {  
 writer.write(printTable.printLineFor5inFile(iterations+1, last\_X, start\_X,  
 functions.Fi(start\_X, lambda, number), functions.f(start\_X, number), Math.*abs*(start\_X - last\_X)));  
 }  
  
 if (flag\_file) {  
 writer.write("\nКорень уравнения:\n");  
 writer.write(String.*format*("x\* = %.10f%n", last\_X));  
 writer.write(String.*format*("f(x\*) = %.10f%n", functions.f(last\_X, number)));  
 writer.flush();  
 } else {  
 System.*out*.println("\nКорень уравнения:");  
 System.*out*.printf("x\* = %.10f%n", last\_X);  
 System.*out*.printf("f(x\*) = %.10f%n", functions.f(last\_X, number));  
 }  
 }  
 else {  
 if (!flag\_file)  
 System.*out*.println("\nКоэффициент сходимости превышает 1: q = " + q);  
 else {  
 writer.write("\nКоэффициент сходимости превышает 1: q = " + q);  
 writer.flush();  
 }  
 }  
 drawChart.drawForIteration(Math.round(a), Math.round(b), lambda, number);  
 }  
  
}

import java.io.FileWriter;  
import java.io.IOException;  
  
public class MethodNewton {  
 DrawChart drawChart = new DrawChart();  
 PrintTable printTable = new PrintTable();  
 Functions functions = new Functions();  
  
 double x1, x2, e;  
 int iterations = 0, systemNumber;  
 boolean flagFile;  
  
 public MethodNewton(double x1, double x2, double e, int systemNumber, boolean flagFile) {  
 this.x1 = x1;  
 this.x2 = x2;  
 this.e = e;  
 this.systemNumber = systemNumber;  
 this.flagFile = flagFile;  
 }  
  
 public void startCalculations() throws IOException {  
 FileWriter writer = null;  
  
 if (flagFile) {  
 writer = new FileWriter("src/main/res/outNewton.txt", false);  
 writer.write("Запущен метод Ньютона\n" +  
 "Решение системы уравнений\n\n");  
 } else {  
 System.*out*.println("Запущен метод Ньютона\n" +  
 "Решение системы уравнений\n\n");  
 }  
  
 if (!flagFile)  
 printTable.printHeaderForNewtonInConsole();  
 else  
 writer.write(printTable.printHeaderForNewtonInFile());  
  
 double dx1, dx2;  
  
 do {  
 double f1 = functions.systemF1(x1, x2, systemNumber);  
 double f2 = functions.systemF2(x1, x2, systemNumber);  
  
 double j11 = functions.systemJ11(x1, x2, systemNumber);  
 double j12 = functions.systemJ12(x1, x2, systemNumber);  
 double j21 = functions.systemJ21(x1, x2, systemNumber);  
 double j22 = functions.systemJ22(x1, x2, systemNumber);  
  
 double detJ = j11 \* j22 - j12 \* j21;  
  
 dx1 = (-j22 \* f1 + j12 \* f2) / detJ;  
 dx2 = (j21 \* f1 - j11 \* f2) / detJ;  
  
 x1 += dx1;  
 x2 += dx2;  
  
 iterations++;  
  
 if (!flagFile)  
 printTable.printLineForNewtonInConsole(iterations, x1, x2, dx1, dx2);  
 else  
 writer.write(printTable.printLineForNewtonInFile(iterations, x1, x2, dx1, dx2));  
  
 } while (Math.*abs*(dx1) > e || Math.*abs*(dx2) > e);  
  
 if (!flagFile) {  
 System.*out*.println("\nРешение системы:");  
 System.*out*.printf("x1 = %.10f%n", x1);  
 System.*out*.printf("x2 = %.10f%n", x2);  
 } else {  
 writer.write("\nРешение системы:\n");  
 writer.write(String.*format*("x1 = %.10f%n", x1));  
 writer.write(String.*format*("x2 = %.10f%n", x2));  
 writer.flush();  
 }  
  
 drawChart.drawSystem(systemNumber);  
 }  
}

public class PrintTable {  
  
 //Таблица для второго метода  
 public void printHeaderFor2inConsole() {  
 System.*out*.println("+-------------------------------------------------" +  
 "------------------------------------------------------------------------------+");  
 System.*out*.printf("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 "№", "a", "b", "x", "F(a)", "F(b)", "F(x)", " |xn+1 - xn| ");  
 System.*out*.println("+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------+");  
 }  
  
 public String printHeaderFor2inFile() {  
 String str = "+-------------------------------------------------" +  
 "------------------------------------------------------------------------------+\n";  
 str += String.*format*("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 "№", "a", "b", "x", "F(a)", "F(b)", "F(x)", " |xn+1 - xn| ");  
 str += "+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------+\n";  
 return str;  
 }  
  
  
 public void printLineFor2inConsole(int i, double a, double b, double x, double f\_a, double f\_b,  
 double f\_x, double abs) {  
 System.*out*.printf("|%-15d|%-15f|%-15f|%-15f|%-15f|%-15f|%-15f|%-15f|\n",  
 i, a, b, x, f\_a, f\_b, f\_x, abs);  
 System.*out*.println("|---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------|");  
 }  
  
 public String printLineFor2inFile(int i, double a, double b, double x, double f\_a, double f\_b,  
 double f\_x, double abs) {  
 String str = String.*format*("|%-15d|%-15f|%-15f|%-15f|%-15f|%-15f|%-15f|%-15f|\n",  
 i, a, b, x, f\_a, f\_b, f\_x, abs);  
 str += "|---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------|\n";  
 return str;  
 }  
  
 //Таблица для четвертого метода  
 public void printHeaderFor4inConsole() {  
 System.*out*.println("+-----------------------------------------------------" +  
 "--------------------------------------------------------------------------+");  
 System.*out*.printf("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 "№", " xn-1", " f(xn-1)", " xn", " f(xn)", " xn+1", " f(xn+1)", " |xn+1 - xn| ");  
 System.*out*.println("+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------+");  
 }  
  
 public String printHeaderFor4inFile() {  
 String str = "+-----------------------------------------------------" +  
 "--------------------------------------------------------------------------+\n";  
 str += String.*format*("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 "№", " xn-1", " f(xn-1)", " xn", " f(xn)", " xn+1", " f(xn+1)", " |xn+1 - xn| ");  
 str += "+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------+\n";  
 return str;  
 }  
  
 public void printLineFor4InConsole(int i, double xn\_1, double f\_xn\_1, double xn, double f\_xn, double xn\_\_1, double f\_xn\_\_1, double abs) {  
  
 System.*out*.printf("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 i, Math.*round*(xn\_1 \* 100.0) / 100.0, Math.*round*(f\_xn\_1 \* 100.0) / 100.0, Math.*round*(xn \* 100.0) / 100.0,  
 Math.*round*(f\_xn \* 100.0) / 100.0, Math.*round*(xn\_\_1 \* 100.0) / 100.0,  
 Math.*round*(f\_xn\_\_1 \* 100.0) / 100.0, Math.*round*(abs \* 100.0) / 100.0);  
 System.*out*.println("+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------+");  
 }  
  
 public String printLineFor4inFile(int i, double xn\_1, double f\_xn\_1, double xn, double f\_xn, double xn\_\_1, double f\_xn\_\_1, double abs) {  
 String str = String.*format*("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 i, Math.*round*(xn\_1 \* 100.0) / 100.0, Math.*round*(f\_xn\_1 \* 100.0) / 100.0, Math.*round*(xn \* 100.0) / 100.0,  
 Math.*round*(f\_xn \* 100.0) / 100.0, Math.*round*(xn\_\_1 \* 100.0) / 100.0,  
 Math.*round*(f\_xn\_\_1 \* 100.0) / 100.0, Math.*round*(abs \* 100.0) / 100.0);  
 str += "+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+---------------+---------------+\n";  
 return str;  
 }  
  
 //Таблица для пятого метода  
 public void printHeaderFor5inConsole() {  
 System.out.println("+-------------------------------------------------" +  
 "----------------------------------------------+");  
 System.out.printf("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 "№", "xi", "xi+1", "Fi(xi+1)", "f(xi+1)", " |xi+1 - xi| ");  
 System.out.println("+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+");  
 }  
  
 public String printHeaderFor5inFile() {  
 String str = "+-------------------------------------------------" +  
 "----------------------------------------------+\n";  
 str += String.format("|%-15s|%-15s|%-15s|%-15s|%-15s|%-15s|\n",  
 "№", "xi", "xi+1", "Fi(xi+1)", "f(xi+1)", " |xi+1 - xi| ");  
 str += "+---------------+---------------+---------------+" +  
 "---------------+---------------+---------------+\n";  
 return str;  
 }  
  
 public void printLineFor5inConsole(int i, double xi, double xi\_1, double Fi, double f, double abs) {  
 System.out.printf("|%-15d|%-15f|%-15f|%-15f|%-15f|%-15f|\n",  
 i, xi, xi\_1, Fi, f, abs);  
 System.out.println("|---------------+---------------+---------------+" +  
 "---------------+---------------+---------------|");  
 }  
 public void printHeaderForNewtonInConsole() {  
 System.out.println("-------------------------------------------------------");  
 System.out.println("| Iteration | x1 | x2 | |Δx1| |Δx2| |");  
 System.out.println("-------------------------------------------------------");  
 }  
  
 public String printHeaderForNewtonInFile() {  
 StringBuilder sb = new StringBuilder();  
 sb.append("-------------------------------------------------------\n");  
 sb.append("| Iteration | x1 | x2 | |Δx1| |Δx2| |\n");  
 sb.append("-------------------------------------------------------\n");  
 return sb.toString();  
 }  
 public void printLineForNewtonInConsole(int iteration, double x1, double x2, double deltaX1, double deltaX2) {  
 System.out.printf("| %-7d| %-10.6f| %-10.6f| %-6.6f| %-6.6f|\n", iteration, x1, x2, Math.abs(deltaX1), Math.abs(deltaX2));  
 }  
  
 public String printLineForNewtonInFile(int iteration, double x1, double x2, double deltaX1, double deltaX2) {  
 return String.format("| %-7d| %-10.6f| %-10.6f| %-6.6f| %-6.6f|\n", iteration, x1, x2, Math.abs(deltaX1), Math.abs(deltaX2));  
 }  
 public String printLineFor5inFile(int i, double xi, double xi\_1, double Fi, double f, double abs) {  
 String str = String.format("|%-15d|%-15f|%-15f|%-15f|%-15f|%-15f|\n",  
 i, xi, xi\_1, Fi, f, abs);  
 str += "|---------------+---------------+---------------+" +  
 "---------------+---------------+---------------|\n";  
 return str;  
 }  
}

public class Functions {  
 public double f(double x, int number) { //Базовая функция  
 if (number == 1)  
 return 2.0 \* Math.*pow*(x, 3) + 3.41 \* Math.*pow*(x, 2) - 23.74 \* x + 2.95;  
 else if (number == 2)  
 return Math.*sin*(x) + 0.5; //корень отрицательный потому что  
 else if (number == 3)  
 return Math.*pow*(x, 3) - x + 4;  
 else  
 return 1;  
 }  
  
 public double f\_(double x, int number) { //Первая производная  
 if (number == 1)  
 return 6.0 \* Math.*pow*(x, 2) + 6.82 \* x -23.74;  
 else if (number == 2)  
 return Math.*cos*(x);  
 else if (number == 3)  
 return 3 \* Math.*pow*(x, 2) - 1;  
 else  
 return 1;  
 }  
  
 public double f\_\_(double x, int number) { //Вторая производная  
 if (number == 1)  
 return 26.7 \* x + 15.62;  
 else if (number == 2)  
 return -Math.*sin*(x);  
 else if (number == 3)  
 return 6 \* x;  
 else  
 return 1;  
 }  
 public double systemF1(double x1, double x2, int systemNumber) {  
 if (systemNumber == 1) {  
 return x1 \* x1 + x2 \* x2 - 4;  
 } else if (systemNumber == 2) {  
 return Math.*sin*(x1) - x2;  
 } else if (systemNumber == 3) {  
 return x1 \* x1 - x2 + 1;  
 } else {  
 return 1;  
 }  
 }  
  
 public double systemF2(double x1, double x2, int systemNumber) {  
 if (systemNumber == 1) {  
 return x1 - x2 \* x2 - 1;  
 } else if (systemNumber == 2) {  
 return x1 - Math.*cos*(x2);  
 } else if (systemNumber == 3) {  
 return Math.*exp*(x1) - x2;  
 } else {  
 return 1;  
 }  
 }  
 public double systemJ11(double x1, double x2, int systemNumber) { //матрица Якоби  
 if (systemNumber == 1) {  
 return 2 \* x1;  
 } else if (systemNumber == 2) {  
 return 0;  
 } else if (systemNumber == 3) {  
 return 2 \* x1;  
 } else {  
 return 1;  
 }  
 }  
  
 public double systemJ12(double x1, double x2, int systemNumber) {  
 if (systemNumber == 1) {  
 return 2 \* x2;  
 } else if (systemNumber == 2) {  
 return -1;  
 } else if (systemNumber == 3) {  
 return 0;  
 } else {  
 return 1;  
 }  
 }  
  
 public double systemJ21(double x1, double x2, int systemNumber) {  
 if (systemNumber == 1) {  
 return 1;  
 } else if (systemNumber == 2) {  
 return 1;  
 } else if (systemNumber == 3) {  
 return Math.*exp*(x1);  
 } else {  
 return 1;  
 }  
 }  
  
 public double systemJ22(double x1, double x2, int systemNumber) {  
 if (systemNumber == 1) {  
 return -2 \* x2;  
 } else if (systemNumber == 2) {  
 return Math.sin(x2);  
 } else if (systemNumber == 3) {  
 return -1;  
 } else {  
 return 1;  
 }  
 }  
  
 public double x0(double a, double b, int number) { //Начальный элемент  
 return a - ((b - a) / (f(b, number) - f(a, number))) \* f(a, number);  
 }  
  
 public double Fi(double x, double lambda, int number) {  
 return x + lambda \* f(x, number);  
 }  
  
 public double Fi\_(double x, double lambda, int number) {  
 return 1 + lambda \* f\_(x, number);  
 }  
  
 public double getMaxF\_(double a, double b, int number) {  
 return Math.max(f\_(a, number), f\_(b, number));  
 }  
  
 public double getMaxQ(double a, double b, double lambda, int number) {  
 return Math.max(Math.abs(Fi\_(a, lambda, number)), Math.abs(Fi\_(b, lambda, number)));  
 }  
  
 public double setNextX(double x1, double x2, int number) {  
 return x1 - (x1 - x2) \* f(x1, number) / (f(x1, number) - f(x2, number));  
 }  
  
 public double setStartX(double a, double b, int number) {  
 if (f(a, number) \* f\_\_(a, number) > 0) {  
 return a;  
 } else if (f(b, number) \* f\_\_(b, number) > 0) {  
 return b;  
 } else {  
 return b; //change 1 to b  
 }  
 }  
}

import org.jfree.chart.ChartFactory;  
import org.jfree.chart.ChartUtils;  
import org.jfree.chart.JFreeChart;  
import org.jfree.chart.plot.PlotOrientation;  
import org.jfree.chart.plot.ValueMarker;  
import org.jfree.chart.plot.XYPlot;  
import org.jfree.data.xy.XYSeries;  
import org.jfree.data.xy.XYSeriesCollection;  
  
import java.awt.\*;  
import java.io.File;  
import java.io.IOException;  
  
  
public class DrawChart {  
 Functions functions = new Functions();  
  
 private double getEquationValue(double x1, double x2, int equationNumber, int systemNumber) {  
 if (systemNumber == 1) {  
 if (equationNumber == 1) {  
 return Math.pow(x1, 2) + Math.pow(x2, 2) - 4;  
 } else if (equationNumber == 2) {  
 return Math.sin(x1) + 0.5;  
 }  
 } else if (systemNumber == 2) {  
 if (equationNumber == 1) {  
 return Math.pow(x1, 2) - x2 - 1;  
 } else if (equationNumber == 2) {  
 return Math.cos(x1) + x2;  
 }  
 }  
  
 return 0;  
 }  
  
 private void rawDraw(XYSeriesCollection lineDataset) {  
 JFreeChart lineChart = ChartFactory.createXYLineChart(  
 "f(x)", "x", "y",  
 lineDataset, PlotOrientation.VERTICAL,  
 true, true, false);  
  
 int width = 1920;  
 int height = 1080;  
  
 try {  
 ChartUtils.saveChartAsJPEG(new File("src/main/res/Chart.jpeg"), lineChart, width, height);  
 } catch (IOException e) {  
 System.out.println(e.getMessage());  
 }  
 }  
  
 public void drawSystem(int systemNumber) {  
 XYSeriesCollection lineDataset = new XYSeriesCollection();  
  
 double a = -5;  
 double b = 5;  
 double len = b - a;  
 int numPoints = 100;  
  
 if (systemNumber == 1) {  
 XYSeries equation1 = new XYSeries("Equation 1");  
  
 for (int i = 0; i <= numPoints; i++) {  
 double x1 = a + i \* (len / numPoints);  
 double x2 = a + i \* (len / numPoints);  
 double y = systemF1(x1, x2);  
  
 equation1.add(x1, y);  
 }  
  
 lineDataset.addSeries(equation1);  
 } else if (systemNumber == 2) {  
 XYSeries equation2 = new XYSeries("Equation 2");  
  
 for (int i = 0; i <= numPoints; i++) {  
 double x1 = a + i \* (len / numPoints);  
 double x2 = a + i \* (len / numPoints);  
 double y = systemF2(x1, x2);  
  
 equation2.add(x1, y);  
 }  
  
 lineDataset.addSeries(equation2);  
 }  
  
 rawDraw(lineDataset);  
 }  
  
 private double systemF1(double x1, double x2) {  
 // Define the equation for system function 1 using x1 and x2 variables  
 return x1 \* x1 + x2 \* x2 - 4;  
 }  
  
 private double systemF2(double x1, double x2) {  
 // Define the equation for system function 2 using x1 and x2 variables  
 return x1 - x2\*x2 - 1;  
 }  
  
 public void drawForIteration(double a, double b, double lambda, int number) {  
 XYSeriesCollection lineDataset = new XYSeriesCollection ();  
  
 a = a - 1;  
 b = b + 1;  
  
 XYSeries function = new XYSeries("function");  
  
 double len = b-a;  
 for (double i = a; i <= b; i+=len/12) {  
 function.add(i,i - lambda \* functions.f(i, number));  
 }  
  
 XYSeries zero = new XYSeries("zero");  
 zero.add(a,0);  
 zero.add(b,0);  
  
  
 lineDataset.addSeries(function);  
 lineDataset.addSeries(zero);  
  
 rawDraw(lineDataset);  
  
 }  
  
 public void draw(double a, double b, int number){  
 XYSeriesCollection lineDataset = new XYSeriesCollection ();  
  
 a = a - 1;  
 b = b + 1;  
  
  
 XYSeries function = new XYSeries("function");  
 double len = b-a;  
 for (double i = a; i <= b; i+=len/12) {  
 function.add(i,functions.f(i, number));  
 }  
  
 XYSeries zero = new XYSeries("zero");  
 zero.add(a,0);  
 zero.add(b,0);  
  
  
 lineDataset.addSeries(function);  
 lineDataset.addSeries(zero);  
  
 rawDraw(lineDataset);  
  
 }  
}

public class CheckInputInterval {  
 Functions functions = new Functions();  
  
  
 public boolean checkInterval(double a, double b, int number) { //Проверка на наличие корней  
 boolean rootFlag = false;  
  
 for (double i = a; i <= b - 0.1; i = i + 0.1)  
 if (functions.f(i, number) > 0 && functions.f(i + 0.1, number) < 0 || functions.f(i, number) < 0 &&  
 functions.f(i + 0.1, number) > 0) {  
 rootFlag = true;  
 break;  
 }  
  
 if (functions.f(a, number) \* functions.f(b, number) < 0 && !rootFlag)  
 rootFlag = true;  
  
 return rootFlag;  
 }  
  
 public int checkIntervalRoots(double a, double b, int number) { //Подсчет количества корней  
 int roots = 0;  
 for (double i = a; i <= b; i += 0.1) {  
 if (functions.f(i, number) > 0 && functions.f(i + 0.1, number) < 0 || functions.f(i, number) < 0 &&  
 functions.f(i + 0.1, number) > 0) {  
 roots++;  
 }  
 }  
 return roots;  
 }  
}

Вывод

Во время выполнения лабораторной работы я изучил работу метода хорд, метода секущих и метода прямых итераций. Основные особенности каждого метода расписаны в описании работы каждого их них.