МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

“ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА”

Вариант 2бв

Студент:
Чернова Анна Ивановна

Группа P32301

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург, 2023

Метод хорд

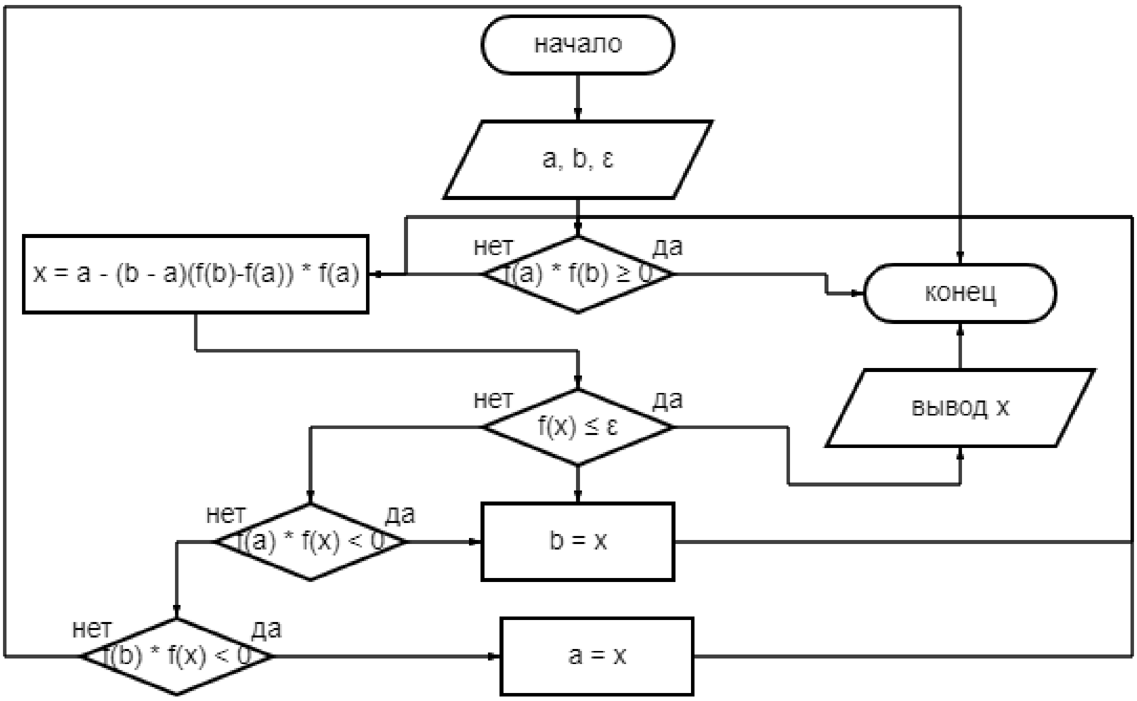
1. Описание реализованного метода

Функция на выбранном отрезке заменяется хордой, в качестве приближенного значения корня принимается точка пересечения хорды с осью абсцисс. В качестве начального приближения выбирается тот конец отрезка, для которого выполняется условие: $x_0 \in [a, b]: f(x_0) * f''(x_0) > 0$.

Рабочая формула метода: $x_i = \frac{a_i f(b_i) - b_i f(a_i)}{f(b_i) - f(a_i)}$

Критерий окончания итерационного процесса: $|x_n - x_{n-1}| \leq \varepsilon$ или $|f(x_n)| \leq \varepsilon$

2. Блок-схема численного метода



3. Листинг реализованного метода

```
private boolean checkNecessity() {
    return equation.calcEquation(leftBound) *
equation.calcEquation(rightBound) < 0;
}
private double getNextXForChordsMethod(double a, double b) {
    double result;
    result = a - (b - a) / (equation.calcEquation(b) -
equation.calcEquation(a)) * equation.calcEquation(a);
    return result;
}
public double calcRoot() throws IncorrectDataException {
```

```
    if (checkNecessity()) {
        double root = getNextXForChordsMethod(leftBound, rightBound);
        getNewInterval(leftBound, rightBound, root);
        while (Math.abs(getNextXForChordsMethod(leftBound, rightBound) -
root) > epsilon &&

Math.abs(equation.calcEquation(getNextXForChordsMethod(leftBound,
rightBound))) > epsilon) {
            root = getNextXForChordsMethod(leftBound, rightBound);
            getNewInterval(leftBound, rightBound, root);
        }
        root = getNextXForChordsMethod(leftBound, rightBound);
        return root;
    } else {
        throw new IncorrectDataException("Не выполняется одно из
достаточных условий сходимости");
    }
}
private void getNewInterval(double a, double b, double x) throws
IncorrectDataException {
    if (equation.calcEquation(a) * equation.calcEquation(x) <= 0) {
        rightBound = x;
    } else if (equation.calcEquation(x) * equation.calcEquation(b) < 0)
{
        leftBound = x;
    } else throw new IncorrectDataException("Невозможно выбрать
начальное приближение");
}
```

Метод касательных

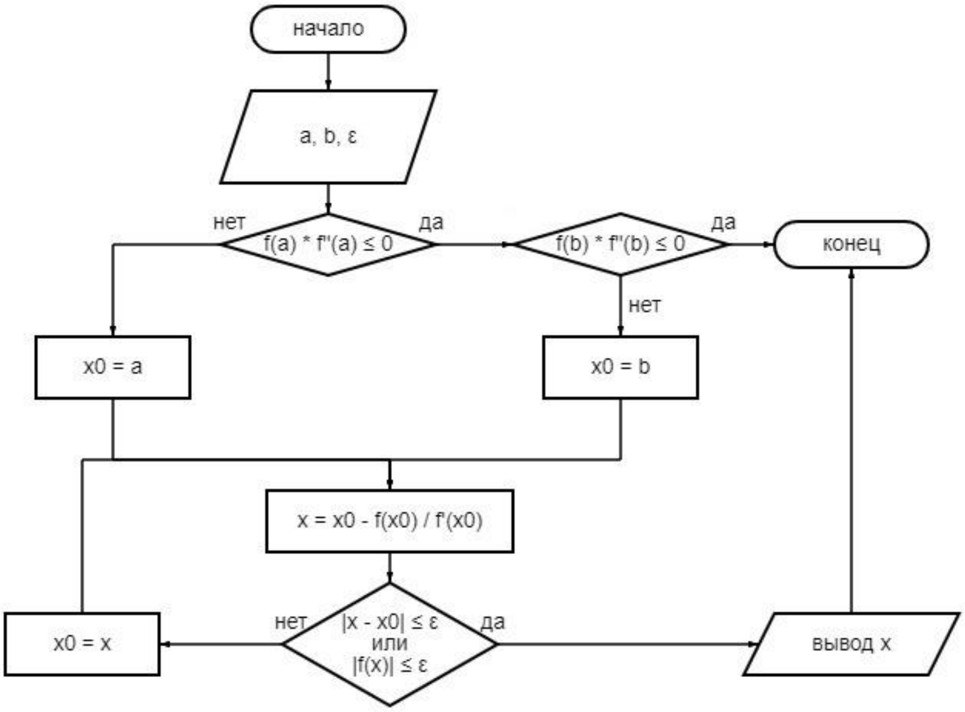
1. Описание реализованного метода

Функция на выбранном отрезке заменяется касательной, в качестве приближенного значения корня принимается точка пересечения касательной с осью абсцисс. В качестве начального приближения выбирается тот конец отрезка, для которого выполняется условие: $x_0 \in [a, b]: f(x_0) * f''(x_0) > 0$.

Рабочая формула метода: $x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$

Критерий окончания итерационного процесса: $|x_n - x_{n-1}| \leq \varepsilon$ или $|f(x_n)| \leq \varepsilon$

2. Блок-схема численного метода



3. Листинг реализованного метода

```
public double calcRoot() {
    double newRoot = root - equation.calcEquation(root) /
equation.calcFirstDerivative(root); С
    if (Math.abs(newRoot - root) <= epsilon ||
Math.abs(equation.calcEquation(newRoot)) <= epsilon) return newRoot;
    else {
        root = newRoot;
        return calcRoot();
    }
}

public void setRoot() throws IncorrectDataException {
    if (equation.calcEquation(equation.getBounds()[0]) *
equation.calcSecondDerivative(equation.getBounds()[0]) > 0) {
        root = equation.getBounds()[0];
    } else if (equation.calcEquation(equation.getBounds()[1]) *
equation.calcSecondDerivative(equation.getBounds()[1]) > 0) {
        root = equation.getBounds()[1];
    } else {
        throw new IncorrectDataException("Невозможно выбрать начальное
приближение");
    }
}
```

Метод простой итерации

1. Описание реализованного метода

Метод предназначен для решения систем вида: $\begin{cases} f_1(x_1, \dots, x_n) = 0 \\ f_2(x_2, \dots, x_n) = 0 \\ \dots \\ f_n(x_1, \dots, x_n) = 0 \end{cases}$

Привести систему уравнений к виду: $\begin{cases} x_1 = \varphi_1(x_1, \dots, x_n) \\ x_2 = \varphi_2(x_2, \dots, x_n) \\ \dots \\ x_n = \varphi_n(x_1, \dots, x_n) \end{cases}$

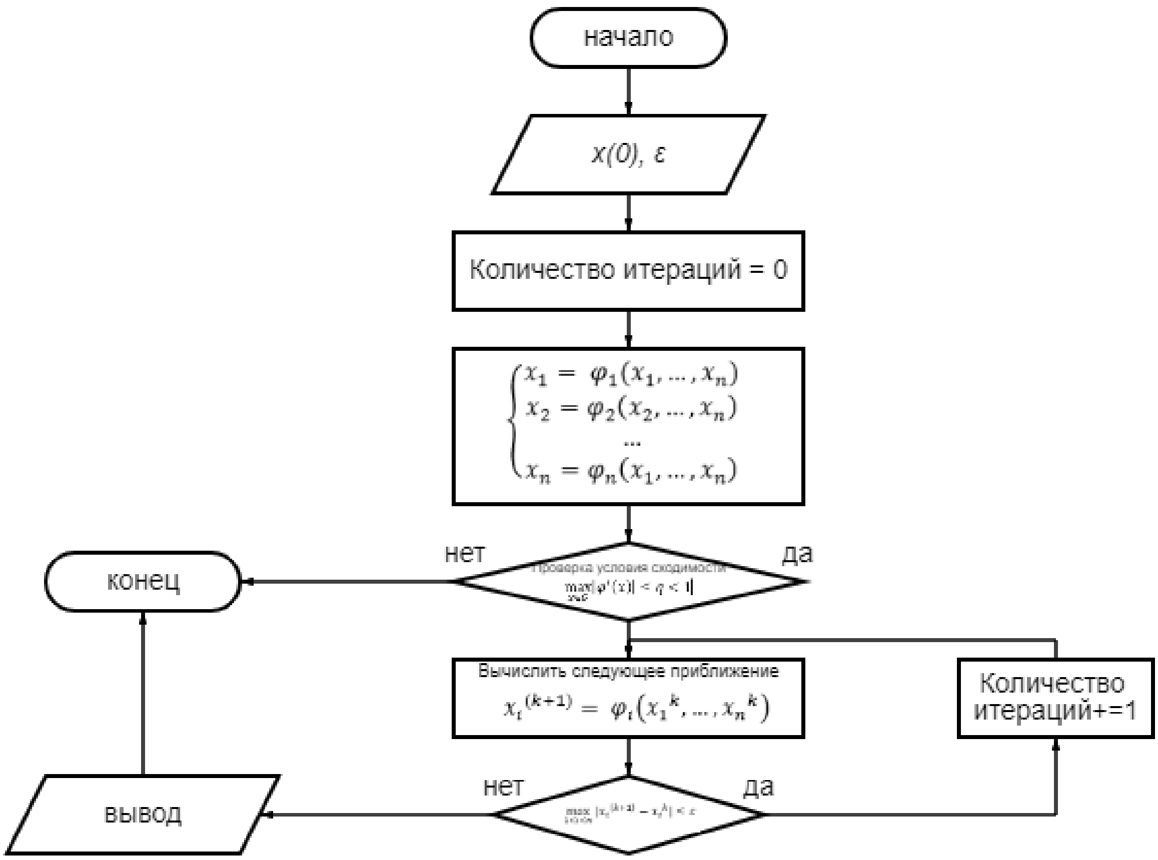
Проверить условие сходимости итерационного процесса: $\max\limits_{x \in G} |\varphi'(x)| \le q < 1$

Задать начальное приближение $x^{(0)}$.

Вычислить следующее приближение: $\begin{cases} {x_1}^{(k+1)} = \varphi_1\left({x_1}^k, \dots, {x_n}^k\right) \\ {x_2}^{(k+1)} = \varphi_2\left({x_1}^k, \dots, {x_n}^k\right) \\ \dots \\ {x_n}^{(k+1)} = \varphi_n\left({x_1}^k, \dots, {x_n}^k\right) \end{cases}$

Критерий окончания итерационного процесса: $\max\limits_{1 < i < n} |{x_i}^{(k+1)} - {x_i}^k| \le \varepsilon$

2. Блок-схема численного метода



3. Листинг реализованного метода

```java
public double[] calcRoots() throws IncorrectDataException {
    result = currentApproximation;
    int cnt = 0;
    if (!checkConvergence(equationSystem.calcFirstDerivative())) {
        throw new IncorrectDataException("Достаточное условие сходимости
итерационного процесса не выполнено");
    }
    while (!isEpsilonSatisfied(getNextByIterations(equationSystem,
result), result, epsilon)) {
        double[] cur = new
double[equationSystem.getCurrentApproximation().length];
        cnt++;
        equationSystem.setCountIteration(cnt);
        result = getNextByIterations(equationSystem, result);
        for (int i = 0; i < cur.length; i++) {
            cur[i] = Math.abs(result[i] -
getNextByIterations(equationSystem, result)[i]);
        }
        equationSystem.setErrors(cur);
        if (cnt > CRITICAL_CNT) {
            throw new IncorrectDataException("MEOW");
        }
    }
    return result;
}
private static double[] getNextByIterations(EquationSystem
equationSystem, double[] previous) {
    double[] res = new
double[equationSystem.getCurrentApproximation().length];
    for (int i = 0; i < equationSystem.getCurrentApproximation().length;
i++) {
        res[i] = equationSystem.calcEquations(previous)[i];
    }
    return res;
}
private static boolean isEpsilonSatisfied(double[] cur, double[] result,
double epsilon) {
    boolean ret = true;
    for (int i = 0; i < cur.length; i++) {
        if (Math.abs(cur[i] - result[i]) > epsilon) {
            ret = false;
            break;
        }
    }
    return ret;
}
private static boolean checkConvergence(double[][] result) {
    boolean ret = true;
    for (int i = 0; i < result.length; i++) {
        double sum = 0;
        for (int j = 0; j < result[i].length; j++) {
            sum = sum + Math.abs(result[i][j]);
        }
        if (sum >= 1) {
            ret = false;
            break;
        }
    }
    return ret;
}
```

Пример работы программы:

Выберите уравнение из списка. Введите число от 1 до 3
1. 5x^3 - x^2 - 20x + 4 = 0
2. sinx * lnx + x = 0
3. x * lnx * lgx -2 = 0
2

Введите границы интервала через пробел
0.1 1

Введите точность
0.01

Ответ, полученный с помощью метода хорд: 0.35631032396493084
Ответ, полученный с помощью метода касательных: 0.368729362889062

Разница: 0.012419038924131154

Выберите систему уравнений из списка. Введите число от 1 до 3
1. 0.1x^2 + x + 0.2y^2 -0.3 = 0
   0.2x^2 + y - 0.1xy - 0.7 = 0

2. sin(x - 0.6) -y = 1.6
   3 * x - cosy = 0.9

3. 0.16y - 0.08z + 1.2 - x = 0
   0.2x - 0.424z - 1.786 - y = 0
   -0.1389x - 0.58889y - 0.0667 - z = 0
1

Введите начальное приближение. Необходимо ввести 2 корня через пробел
1 1

Введите точность
0.01

Ответ:
x = 0.20726010379801596
y = 0.679417317949696

Погрешности:
x : 0.0038773572466440642
y : 0.002090278459011219

Количество итераций: 4

Выберите уравнение из списка. Введите число от 1 до 3
1. 5x^3 - x^2 - 20x + 4 = 0
2. sinx * lnx + x = 0
3. x * lnx * lgx -2 = 0
3

Введите границы интервала через пробел

2 4

Введите точность
0.02

Ответ, полученный с помощью метода хорд: 3.271564211714425
Ответ, полученный с помощью метода касательных: 3.2747942791326685

Разница: 0.0032300674182437206

Выберите систему уравнений из списка. Введите число от 1 до 3
1. 0.1x^2 + x + 0.2y^2 -0.3 = 0
   0.2x^2 + y - 0.1xy - 0.7 = 0

2. sin(x - 0.6) -y = 1.6
   3 * x - cosy = 0.9

3. 0.16y - 0.08z + 1.2 - x = 0
   0.2x - 0.424z - 1.786 - y = 0
   -0.1389x - 0.58889y - 0.0667 - z = 0
2

Введите начальное приближение. Необходимо ввести 2 корня через пробел
0 -1.5

Введите точность
0.002

Ответ:
    x = 0.15205194312334486
    y = -2.0346541865212164

Погрешности:
    x : 0.0011858229832711065
    y : 0.00153723305747544

Количество итераций: 9

Вывод:

1. Метод деления пополам, метод хорд и метод простых итераций просты в реализации, метод касательных требует вычисления значения производной на каждой итерации, что усложняет реализацию.
2. Метод деления пополам и метод хорд имеют линейную сходимость, однако порядок сходимости метода хорд выше, чем у метода деления пополам. В сравнении с ними метод касательных имеет квадратичную сходимость.
3. Общим недостатком метода хорд, метода касательных и метода простых итераций является выбор начального приближения.
4. Метод простой итерации для решения СНАУ является менее чувствительным к выбору начального приближения, чем метод Ньютона, однако его сходимость может быть достигнута при выполнении определенных условий на матрицу системы.