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”

Решение СНАУ. Методы хорд и Ньютона.

**Выполнил:**

Кислицин Алексей Андреевич

**Группа:**

P3231

**Преподаватель:**

Перл Ольга Вячеславовна

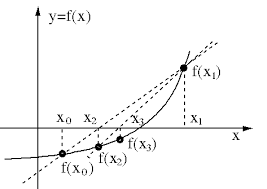


Санкт-Петербург, 2022

***Метод хорд***

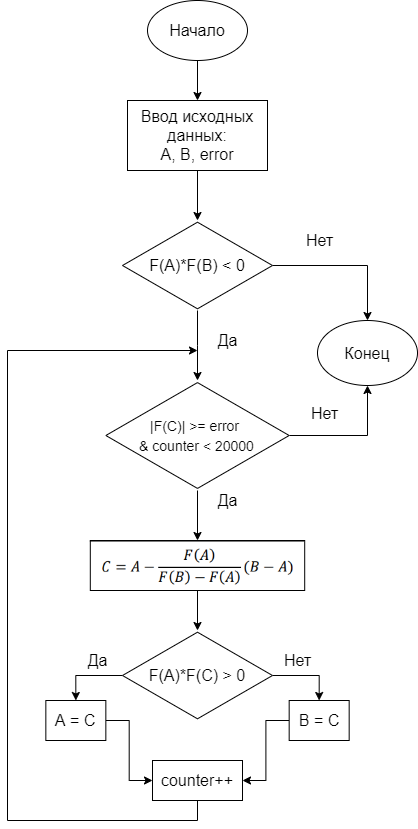
*Описание*

При решении уравнения методом хорд нелинейная функция f(x) на отделённом интервале [a, b] заменяется линейной, в качестве которой берётся хорда – прямая, стягивающая концы нелинейной функции. Вычисляются значения функции на концах отрезка, и строится прямая, соединяющая точки (a, f(a)) и (b, f(b)). При решении нелинейного уравнения методом хорд задаются интервал [a, b], на котором существует только одно решение, и точность ε. Затем через две точки с координатами (a, f(a)) и (b, f(b)) проводим хорду и определяем точку пересечения этой линии с осью абсцисс, точку c. Если при этом f(a)∙f(c)<0, то b=c, иначе a=c. Поиск решения прекращается при достижении заданной точности |f(c)|<ε.

Изображение выглядит как текст

Автоматически созданное описание

*Блок-схема*

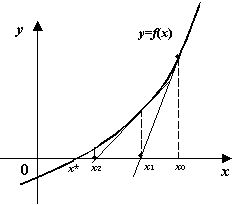


*Листинг*

public double solveBySecantMethod(int numberOfEquation) throws AbsException {  
 double[] bounds = inputObject.getBounds();  
 double error = Double.*MAX\_VALUE*;  
 double root = 0;  
 Function<Double, Double> function = functions.get(numberOfEquation - 1);  
 while (error >= inputObject.getError() && itrCounter1 < LIMIT) {  
 root = findAbscissaIntersectionBySecant(bounds, function);  
 if (function.apply(root) \* function.apply(bounds[0]) < 0)  
 bounds[1] = root;  
 else bounds[0] = root;  
 itrCounter1++;  
 error = Math.*abs*(function.apply(root));  
 }  
 if (itrCounter1 >= LIMIT)  
 throw new TooMuchIterationException("При расчете корня уравнения методом хорд был превышен лимит итераций (limit=" + LIMIT + ")");  
 return root;  
}

***Метод Ньютона (касательных)***

*Описание*

Классический метод Ньютона или касательных заключается в том, что если — некоторое приближение к корню x уравнения , то следующее приближение определяется как корень касательной к функции , проведённой в точке . Уравнение касательной к функции в точке имеет вид:



Тогда при алгоритм последовательных вычислений сводится к

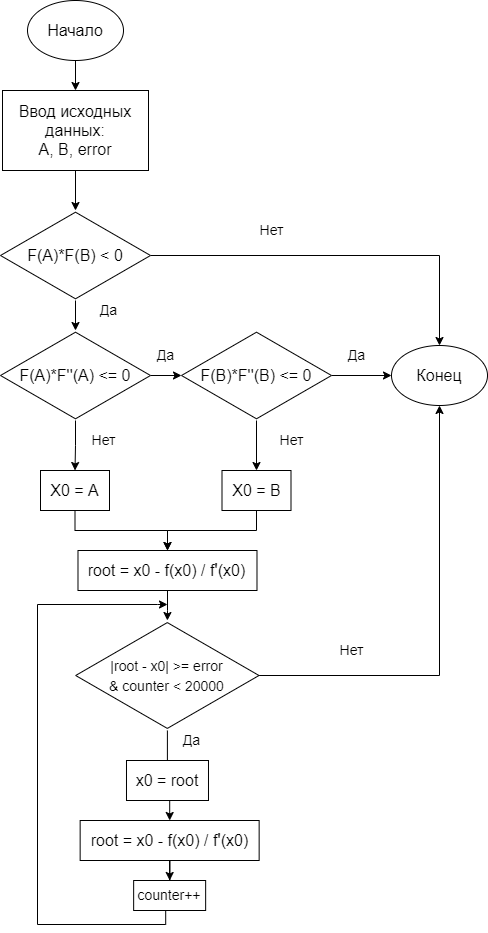


Вычисления производятся пока , где e – допустимая абсолютная погрешность, заданная для конкретного решения.

Метод Ньютона применим только если соблюдаются несколько условий:

1. функция *y= f(x)* определена и непрерывна при
2. *f(a)·f(b) < 0* (функция принимает значения разных знаков на концах отрезка [*a, b*]);
3. производные *f'(x)* и *f''(x)* сохраняют знак на отрезке [*a, b*] (т.е. функция *f(x)* либо возрастает, либо убывает на отрезке [*a, b*], сохраняя при этом направление выпуклости);
4. 

*Блок-схема*

**

*Листинг*

public double solveByNewtonMethod(int numberOfEquation) throws AbsException {  
 double[] bounds = inputObject.getBounds();  
 double a = bounds[0];  
 double b = bounds[1];  
 double x0;  
 double root;  
 Function<Double, Double> function = functions.get(numberOfEquation - 1);  
 if (function.apply(a) \* derivative(function, 2, a) <= 0) {  
 x0 = b;  
 } else x0 = a;  
 root = findAbscissaIntersectionByTangent(numberOfEquation, x0);  
 while (Math.*abs*(root - x0) >= inputObject.getError() && itrCounter2 < LIMIT) {  
 x0 = root;  
 root = findAbscissaIntersectionByTangent(numberOfEquation, x0);  
 itrCounter2++;  
 }  
 if (itrCounter2 >= LIMIT)  
 throw new TooMuchIterationException("При расчете корня уравнения методом касательных был превышен лимит итераций (limit=" + LIMIT + ")");  
 return root;  
}

***Метод простых итераций***

*Описание*

Для того, чтобы решить СНАУ методом простых итераций, для начала нужно преобразовать каждое уравнение в системе к такому виду: – то есть выразить из каждого уравнение неизвестную переменную. После этого нужно выбрать вектор начального приближения . Алгоритм для расчёта каждого последующего приближения: .

*Листинг*

public Vector findRootsBySimpleIterationMethod(double[] point) throws AbsException {  
 Matrix lambdas = findLambdasMatrix(point);  
 Vector pointVector = new Vector(point);  
 ArrayList<Function<Vector, Double>> functions = system.getSystem();  
 double delta;  
 do {  
 delta = 0;  
 double[] funcValues = new double[system.size()];  
 for (int i = 0; i < point.length; ++i) {  
 funcValues[i] = functions.get(i).apply(pointVector);  
 }  
 Vector newPointVector = pointVector.subtract(lambdas.multiply(new Vector(funcValues)));  
 for (int i = 0; i < point.length; ++i) {  
 if (Double.*isNaN*(newPointVector.get(i))) {  
 throw new MethodDivergesException("Нельзя решить СНАУ для этого начального приближения "  
 + Arrays.*toString*(point) +  
 "\nВы можете попробовать указать другое приближение, "  
 + "которое ближе к реальному ответу");  
 }  
 if (Math.*abs*(newPointVector.get(i) - pointVector.get(i)) > delta) {  
 delta = Math.*abs*(newPointVector.get(i) - pointVector.get(i));  
 }  
 }  
 pointVector = new Vector(newPointVector.getVector().clone());  
 itrCounter3++;  
 } while (delta > inputObject.getError() && itrCounter3 < LIMIT);  
 if (itrCounter3 >= LIMIT)  
 throw new TooMuchIterationException("При расчете корней СНАУ методом простых итераций был превышен лимит итераций (limit=" + LIMIT + ")");  
 return pointVector;  
}

***Вывод:***

Реализованный метод простых итераций для решения нелинейных

уравнений находит более точное решение, в отличие от метода Ньютона.

Тем не менее он сходится к решению в малой окрестности корня, поэтому

приходится вычислять очень близкие к истинному корню границы изоляции.

Помимо этого, сходимость этого метода может быть очень медленной.

Метод Ньютона же обладает более быстрой сходимостью, но требует

вычисления производной на каждой итерации. Метод простых итераций для

решения СНАУ прост в реализации, но это только в случае, когда

неизвестные уже явно выражены в каждой функции. В случае, когда функции

заданы в общем виде необходимо вычислять матрицу значений λ и

выполнять операцию перемножения матриц на каждой итерации.