Федеральное государственное автономное образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Лабораторная работа №2
Вариант: 1аб

Выполнил:
Воробьев Кирилл
P3231

Преподаватель:
Перл Ольга Вячеславовна

Санкт-Петербург, 2022г

Цель работы

Реализовать три численных метода, а именно:

- Решение нелинейных уравнений ***методом деления пополам***.
- Решение нелинейных уравнений ***методом хорд***.
- Решение систем нелинейных уравнений ***методом Ньютона***.

Описание методов

Метод деления пополам:

Метод применим для численного решения уравнения f(x) = 0 для вещественной переменной x, где f-непрерывная функция, определенная на интервале [a, b] и где f(a) и f(b) имеют противоположные знаки. В этом случае говорят, что a и b заключают в скобки корень, так как по теореме о промежуточном значении непрерывная функция f должна иметь хотя бы один корень в интервале (a, b).

На каждом шаге метод делит интервал на две половины, вычисляя среднюю точку c = (a+b) / 2 интервала и значение функции f(c) в этой точке. Если c сам по себе не является корнем (что очень маловероятно, но возможно), теперь есть только две возможности: либо f(a) и f(c) имеют противоположные знаки и скобку корня, либо f(c) и f(b) имеют противоположные знаки и скобку корня. Метод выбирает подинтервал, который гарантированно будет скобкой, в качестве нового интервала, который будет использоваться на следующем шаге. Таким образом, интервал, содержащий ноль f уменьшается в ширину на 50% на каждом шаге. Процесс продолжается до тех пор, пока интервал не станет достаточно маленьким.

Явно, если f(a) и f(c) имеют противоположные знаки, то метод устанавливает c в качестве нового значения для b, а если f(b) и f(c) имеют противоположные знаки, то метод устанавливает c в качестве нового a. (Если f(c)=0, то в качестве решения можно принять c и процесс остановится.) В обоих случаях новые f(a) и f(b) имеют противоположные знаки, поэтому метод применим к этому меньшему интервалу.

Метод хорд:

Суть метода хорд состоит в разбиении отрезка [a; b] (при условии f(a)f(b) < 0) на два отрезка с помощью хорды и выборе нового отрезка от точки пересечения хорды с осью абсцисс до неподвижной точки, на котором функция меняет знак и содержит решение, причём подвижная точка приближается к ε-окрестности решения. Построение хорд продолжается до достижения необходимой точности решения ε.

Метод хорд применим для решения уравнения вида f(x) = 0 на отрезке [a; b], если ни одна точка отрезка [a; b] не является ни стационарной, ни критической, то есть f'(x) ≠ 0 и f''(x) ≠ 0.

Условие начальной точки для метода хорд f(x)f''(x) < 0.

Условие неподвижной точки для метода хорд f(x)f''(x) > 0.

Сначала находим отрезок [a; b] такой, что функция f(x) дважды непрерывно дифференцируема и меняет знак на отрезке, то есть $f(a)f(b) < 0$. Далее применяем алгоритм решения:

Если $f(a) \cdot f''(a) > 0$, то $c = a$, иначе если $f(b) \cdot f''(b) > 0$, то $c = b$.

Если $f(a) \cdot f''(a) < 0$, то $x = a$, иначе если $f(b) \cdot f''(b) < 0$, то $x = b$.

$$\Delta x = f(x) \cdot (x - c) / (f(x) - f(c)).$$

$$x = x — \Delta x.$$

Если $|\Delta x| > \varepsilon$, то идти к 3.

Значение x является решением с заданной точностью ε нелинейного уравнения вида $f(x) = 0$. Если $f(x) = 0$, то x — точное решение.

Метод Ньютона:

Идея состоит в том, чтобы начать с первоначального предположения, затем аппроксимировать функцию ее касательной линией и, наконец, вычислить x-перехват этой касательной линии. Этот x-перехват обычно будет лучшим приближением к корню исходной функции, чем первое предположение, и метод может быть повторен.

Если касательная к кривой f(x) при $x = x_n$ пересекает ось x при x_{n+1}, то наклон равен

$$f'(x_n) = \frac{f(x_n) - 0}{x_n - x_{n+1}}$$

Решение для x_{n+1} дает

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Мы начинаем процесс с некоторого произвольного начального значения x_0 (чем ближе к нулю, тем лучше. Но в отсутствие какой-либо интуиции о том, где может находиться ноль, метод "угадай и проверь" может сузить возможности до достаточно малого интервала, обратившись к теореме о промежуточном значении). Метод обычно сходится при условии, что это начальное предположение достаточно близко к неизвестному нулю и что $f'(x_0) \neq 0$. Кроме того, для нуля кратности 1 сходимость по меньшей мере квадратична.

Расчетные формулы методов

Метод деления пополам

$c = \frac{b^{n-1}+a^{n-1}}{2}$, где n-номер итерации

если $f(a^{n-1})*f(c) \le 0$, то $b^n=c$, $a^n=a^{n-1}$

если $f(b^{n-1})*f(c) \le 0$, то $a^n=c$, $b^n=b^{n-1}$

Метод хорд

$$c = a - \frac{f(a)}{f(b) - f(a)} \cdot (b - a)$$

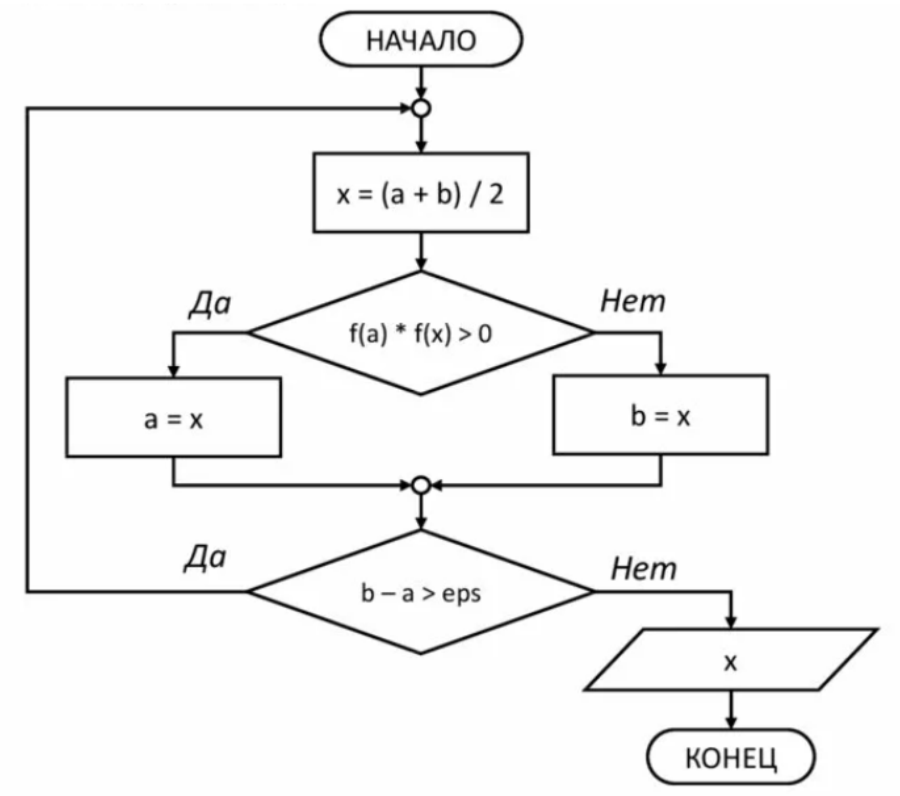
Метод Ньютона

$$J\left(\boldsymbol{x}^{(k)}\right)\Delta\boldsymbol{x}^{(k+1)} = -\boldsymbol{f}\left(\boldsymbol{x}^{(k)}\right)$$

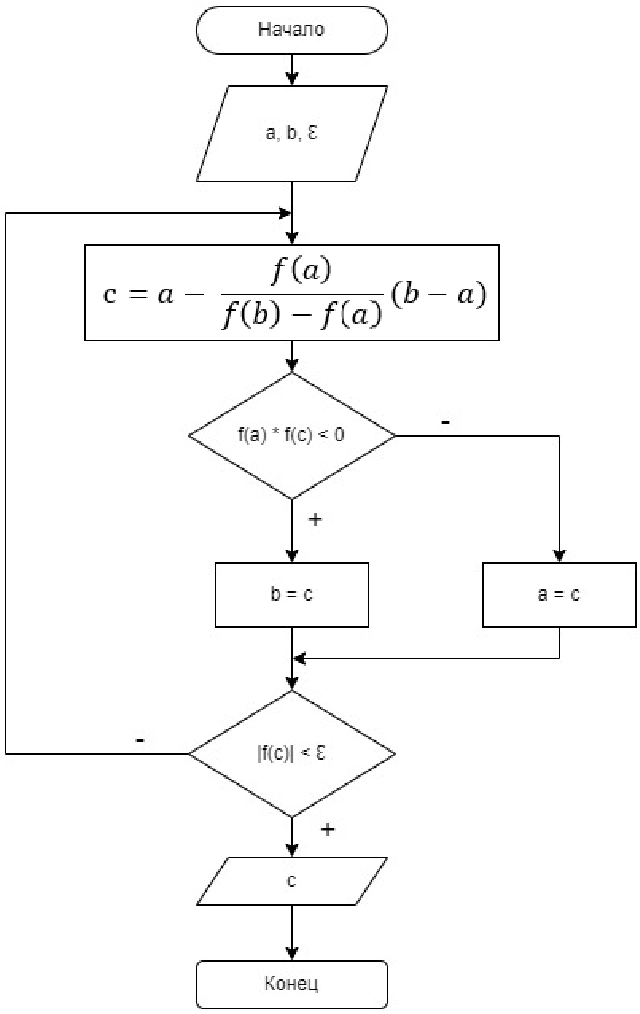
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta\boldsymbol{x}^{(k+1)}$$

Блок-схемы методов

Метод деления пополам



Метод хорд



Метод Ньютона

Методы в коде

Метод деления пополам

```
public static double doMethod(double a, double b, double eps, NumberOfEquation equation) throws EndsOfTheSegmentExce
    checkingForSegment(a, b, equation);
    double c;
    while (b - a > eps) {
        c = (b - a) / 2.0 + a;
        if (EquationStorage.getEquation(equation, a) * EquationStorage.getEquation(equation, c) < 0) b = c;
        else a = c;
    }
    return a;
}

private static void checkingForSegment(double a, double b, NumberOfEquation equation) throws EndsOfTheSegmentExcepti
    if (EquationStorage.getEquation(equation, a) * EquationStorage.getEquation(equation, b) > 0) {
        throw new EndsOfTheSegmentException("Концы отрезка имеют одинаковый знак!");
    }
}
```

Метод хорд

```java
public static double doMethod(double a, double b, double eps, NumberOfEquation equation) throws EndsOfTheSegmentException, EquationDoes
    checkingForSegment(a, b, equation);
    double buffer;
    double difference = Double.MAX_VALUE;
    double value = 0;
    while (difference > eps) {
        if (EquationStorage.getEquation(equation, a) * EquationStorage.getEquation(equation, getNextValue(a, b, equation)) < 0) {
            buffer = b;
            b = getNextValue(a, b, equation);
            difference = Math.abs(buffer - b);
            value = b;
        } else {
            buffer = a;
            a = getNextValue(a, b, equation);
            difference = Math.abs(buffer - a);
            value = a;
        }
    }
    return value;
}

private static double getNextValue(double value, double immutableValue, NumberOfEquation equation) throws EquationDoesNotExistException
    return value - ((immutableValue - value) / (EquationStorage.getEquation(equation, immutableValue) -
            EquationStorage.getEquation(equation, value))) * EquationStorage.getEquation(equation, value);
}

private static void checkingForSegment(double a, double b, NumberOfEquation equation) throws EndsOfTheSegmentException, EquationDoesNot
    if (EquationStorage.getEquation(equation, a) * EquationStorage.getEquation(equation, b) > 0) {
        throw new EndsOfTheSegmentException("Концы отрезка имеют одинаковый знак!");
    }
}
```

Метод Ньютона

```java
public static double[] doMethod(NumberOfEquation number, double[] startValues, double eps) throws EquationDoesNotExistException {
    double[] unknownsColumn = new double[2];
    EquationSystem system = getSystem(number);
    unknownsColumn[0] = system.getFirstEquation(startValues[0], startValues[1]);
    unknownsColumn[1] = system.getSecondEquation(startValues[0], startValues[1]);
    double[] buffer = {unknownsColumn[0] + 2 * eps, unknownsColumn[1] + 2 * eps}; // + eps + 1 - для того чтобы корректно зайти в цикл
    double[][] jacoby;
    double[] results;
    double[] differences;
    for (int i = 1; Math.abs(unknownsColumn[0] - buffer[0]) > eps && Math.abs(unknownsColumn[1] - buffer[1]) > eps; i++) {
        buffer = Arrays.copyOf(unknownsColumn, unknownsColumn.length);
        jacoby = calculateJacoby(buffer, system);
        results = calculateResults(buffer, system);
        differences = Gauss.getUnknownColumn(jacoby, results);
        unknownsColumn[0] += differences[0];
        unknownsColumn[1] += differences[1];
        printInformationAboutIteration(i, unknownsColumn);
    }
    return unknownsColumn;
}
```

Вычисление матрицы Якоби для метода Ньютона

```java
private static double[][] calculateJacoby(double[] unknownsColumn, EquationSystem system) {
    double[][] jacoby = new double[2][2];
    jacoby[0][0] = system.getDerivativeXOfFirstEquation(unknownsColumn[0], unknownsColumn[1]);
    jacoby[0][1] = system.getDerivativeYOfFirstEquation(unknownsColumn[0], unknownsColumn[1]);
    jacoby[1][0] = system.getDerivativeXOfSecondEquation(unknownsColumn[0], unknownsColumn[1]);
    jacoby[1][1] = system.getDerivativeYOfSecondEquation(unknownsColumn[0], unknownsColumn[1]);
    return jacoby;
}
```

Вывод

В ходе лабораторной работы я рассмотрел решение нелинейных уравнений методом деления пополам и методом хорд, а также решение систем нелинейных уравнений методом Ньютона. В методе хорд, в отличие от метода половинного деления, отрезок делится не пополам, а, что более естественно, пропорционально отношению $f(a) / f(b)$, в силу этого некоторые уравнения могут решаться быстрее данным методом, нежели методом деления пополам.