Санкт-Петербургский национальный
исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 «Программная инженерия»
Дисциплина «Вычислительная математика»

Отчет
По лабораторной работе №3
«Метод Ньютона»

Выполнил студент:
Бабушкин А.М. (P3221)
Преподаватель:
Перл О.В.

Санкт-Петербург
2024

Описание численного метода:

Решение нелинейных уравнений методом Ньютона основывается на построении матрицы Якоби путем нахождения частных производных.

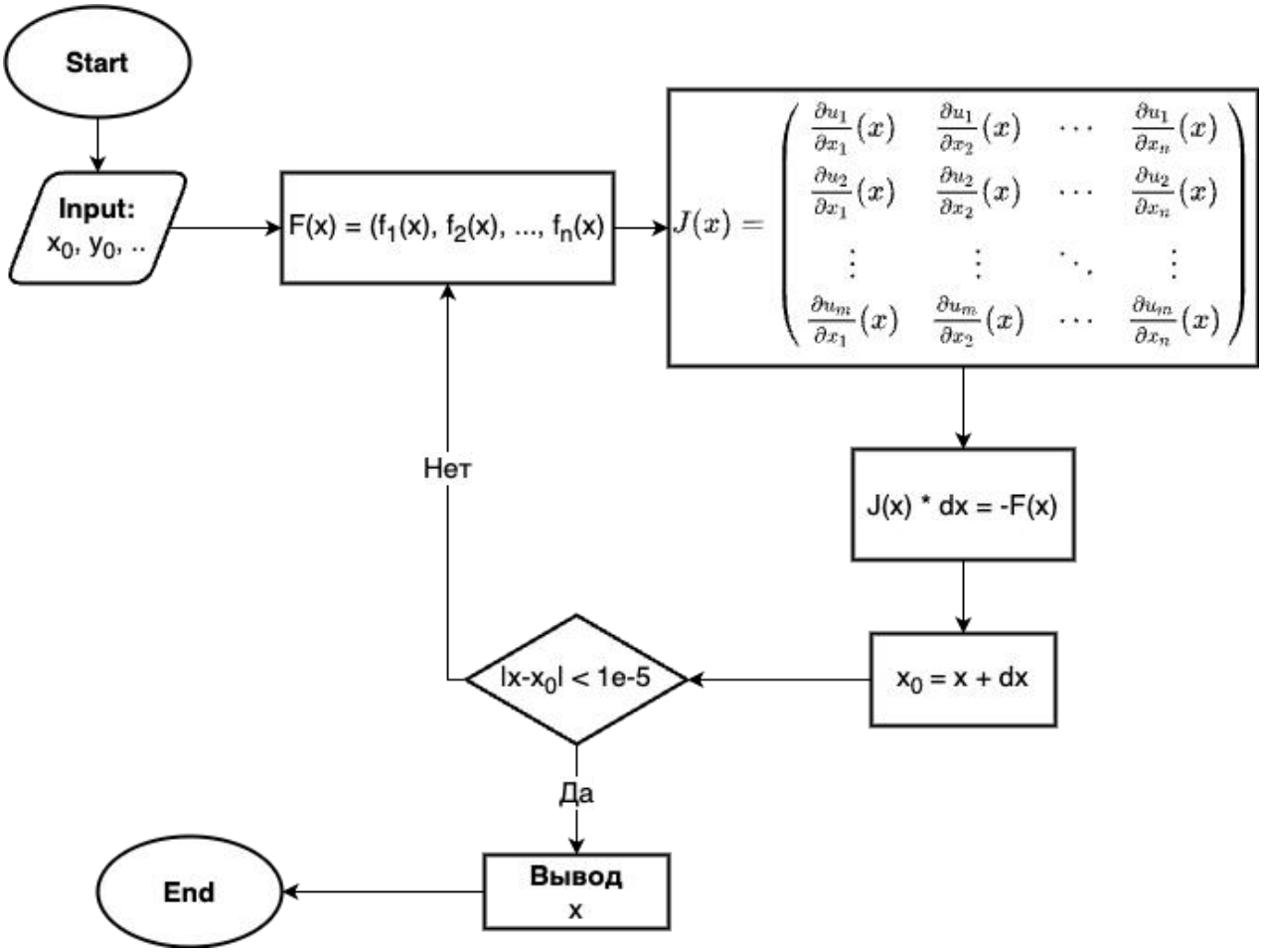
- Матрица Якоби:

$$J(x) = \begin{pmatrix} \frac{\partial u_1}{\partial x_1}(x) & \frac{\partial u_1}{\partial x_2}(x) & \cdots & \frac{\partial u_1}{\partial x_n}(x) \\ \frac{\partial u_2}{\partial x_1}(x) & \frac{\partial u_2}{\partial x_2}(x) & \cdots & \frac{\partial u_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial u_m}{\partial x_1}(x) & \frac{\partial u_m}{\partial x_2}(x) & \cdots & \frac{\partial u_m}{\partial x_n}(x) \end{pmatrix}$$

- Затем нам нужно сформировать СЛАУ: $J(x) * dx = -F(x)$

Решив эту СЛАУ мы получим второе приближение, потом третье и т.д. пока оно не будет удовлетворять нашему условию.

Блок-схема:


Start
Input:
x0, y0, ..
F(x) = (f1(x), f2(x), ..., fn(x)
J(x) =
J(x) * dx = -F(x)
x0 = x + dx
|x-x0| < 1e-5
Нет
Да
Вывод
x
End

Метод реализованный на языке Java:

```java
class Result {
    public static List<Double> solve_by_fixed_point_iterations(int system_id, int number_of_unknowns, List<Double> initial_approximations) {
        List<Function<List<Double>, Double>> functions = SNAEFunctions.get_functions(system_id);

        List<Double> currentApproximations = initial_approximations;
        double epsilon = 1e-5;

        while (true) {
            List<Double> nextApproximations = new ArrayList<>();

            for (int i = 0; i < number_of_unknowns; i++) {
                double sum = 0.0;
                for (Function<List<Double>, Double> function : functions) {
                    List<Double> updatedArgs = new ArrayList<>(currentApproximations);
                    updatedArgs.set(i, function.apply(updatedArgs));
                    sum += updatedArgs.get(i);
                }
                nextApproximations.add(currentApproximations.get(i) - sum / functions.size());
            }

            boolean isConverged = true;
            for (int i = 0; i < number_of_unknowns; i++) {
                if (Math.abs(currentApproximations.get(i) - nextApproximations.get(i)) > epsilon) {
                    isConverged = false;
                    break;
                }
            }

            if (isConverged) {
                return nextApproximations;
            } else {
                currentApproximations = nextApproximations;
            }
        }
    }
}
```

Тесты:

Тест 1

Ввод	**Вывод**
1	-1.1839731744105369E-6
2	0.9999988160268257
3	
4	

Тест 2

Ввод	**Вывод**
2	0.7002303562721797
2	0.3702303562721795
0.66	
0.33	

Тест 3

Ввод	**Вывод**
5	4.0
3	2.0
4	0.0
2	
0	

Тест 4

Ввод	**Вывод**
3	0.17152339140053463
3	0.6705233914005344
0.001	1.1605233914005346
0.5	
0.99	

Тест 5

Ввод	**Вывод**
4	-Infinity
3	-Infinity
2	-Infinity
5	
2	

Вывод:

В ходе лабораторной работы был реализован метод Ньютона для решения нелинейных уравнений. Метод основывается на построении матрицы Якоби и последующем решении системы линейных алгебраических уравнений (СЛАУ).

Сравнение с другими методами:
Метод Ньютона имеет более быструю сходимость, чем метод итераций по простой точке, но он может быть менее устойчивым.

Анализ применимости метода:
- Метод Ньютона применим для решения нелинейных уравнений и систем нелинейных уравнений.
- Метод может не сходиться, если начальное приближение слишком далеко от решения.

Алгоритмическая сложность:

Алгоритмическая сложность метода Ньютона составляет $O(n^4)$, где n - размер системы уравнений.

Общие выводы:
Метод Ньютона является мощным инструментом для решения нелинейных уравнений.
Метод имеет ряд ограничений, которые необходимо учитывать при его использовании.
Для достижения наилучших результатов важно выбрать подходящее начальное приближение и использовать методы контроля численной ошибки.