**Санкт-Петербургский Национальный Исследовательский Университет ИТМО**

**Лабораторная работа №3**

по дисциплине

*«Вычислительная математика»*

Выполнил: Анисимов М. Д.

Группа: Р3233

Преподаватель: Перл О. В.

Санкт-Петербург

2024 г.

**Цель работы**

Разработать программу, которая вычисляет СНАУ методом Ньютона. В качестве входных данных программа получает номер системы, количество начальных приближений и значения начальных приближений. Выводом является решение системы уравнений с точностью до 5 знаков после запятой.

**Описание метода**

Метод основан на нахождении частных производных уравнений нашей системы и формирование матрицы Якоби. Далее необходимо сформировать СЛАУ. Для этого используется формула , где J – матрица Якоби, – вектор дельт, – вектор значений функции в точке . При помощи данной формулы мы находим второе приближение, потом третье и так далее до момента, пока наши приближения не будут соответствовать условию

**Код программы**

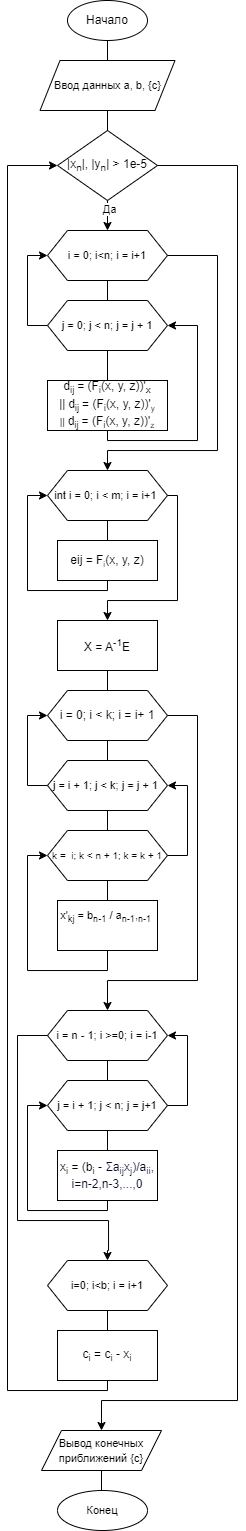
class Result {  
 private static final double *EPSILON* = 1e-5;  
  
 public static List<Double> solve\_by\_fixed\_point\_iterations(int system\_id, int number\_of\_unknowns, List<Double> initial\_approximations) {  
 List<Function<List<Double>, Double>> functions = SNAEFunctions.*get\_functions*(system\_id);  
  
 List<List<Double>> jacobianMatrix;  
 List<Double> fValues;  
 List<Double> deltaX = new ArrayList<>(Collections.*nCopies*(number\_of\_unknowns, 1.0));   
  
 while (deltaX.stream().mapToDouble(Double::doubleValue).anyMatch(val -> Math.*abs*(val) > *EPSILON*)) {  
 jacobianMatrix = *calculateJacobianMatrix*(functions, initial\_approximations);  
 fValues = *calculateFunctionValues*(functions, initial\_approximations);  
 deltaX = *solveLinearSystem*(jacobianMatrix, fValues);  
  
 for (int i = 0; i < number\_of\_unknowns; i++) {  
 initial\_approximations.set(i, initial\_approximations.get(i) - deltaX.get(i));  
 }  
 }  
  
 return initial\_approximations;  
 }

Где calculateJacobianMatrix() – метод для нахождения Якобиана, calculateFunctionValues() -–нахождение значения функции F(x, y) в точке, solveLinearSystem() -–решение СЛАУ (А именно формулы

**Пример работы программы**

|  |  |
| --- | --- |
| Пример №1 |  |
| Пример №2 |  |
| Пример №3 |  |
| Пример №4 | - забыл добавить вывод введённых же приближений в качестве результата |
| Пример №5 |  |
| Примечания: | В примерах работы программы приведены только те случаи, когда количество начальных приближений равно количеству неизвестных в уравнениях системы. Если количество начальных приближений больше или меньше, чем количество неизвестных, то программа работать не будет. Также в некоторых случаях метод Ньютона может не сходится. В таком случае программа просто зависнет |

**Блок-схема программы**

****

**Вывод**

Программа работает в том случае, когда метод Ньютона имеет сходимость. Если сходимости нет, то программа будет работать в бесконечном цикле. Также программа работает только в том случае, когда количество начальных приближений равно количеству неизвестных в системе. В случае, если пользователь вводит номер системы больше 4, то выводится ошибка о вычислении. В остальных случаях программа вычисляет приближения до тех пор, пока не обнаружит приближение, которое не удовлетворяет точности эпсилон. В таком случае программа выведет результат. В сравнении с другими методами (хорд, половинного деления и тд) метод Ньютона требует дополнительного вычисления в виде производной, а также сходимость не гарантирована. Преимуществом же является быстрая сходимость функции с хорошо выбранным начальным приближением. Алгоритмическая сложность алгоритма О(n4).