Национальный исследовательский университет ИТМО

Факультет ПИиКТ

Лабораторная работа №1
по вычислительной математике

Работу выполнила:

Тройникова Вероника

Группа: P3233

Преподаватель:

Перл О. В.

Санкт-Петербург

2022

1. Описание метода, расчетные формулы (Метод Гаусса-Зейделя)

Метод Гайусса-Зейделя является итерационным методом (неизвестные x_i ищутся в течение некоторого числа итераций). Данный метод похож на метод простых итераций: из уравнений выражаются неизвестные (из i-того уравнения выражается i-тая неизвестная), выбираются первоначальные значения неизвестных, на каждой итерации происходит расчет новых значений x_i с использованием ранее полученных значений. Таким образом, при использовании итерационных методов выбирается начальное приближение к решению, на основе которого на каждой следующей итерации рассчитывается новое приближение. Чем больше произведено итераций, тем более точный ответ получится в итоге.

Различие этих двух методов заключается в том, что по методу Гаусса-Зейделя при расчете нового значения x_i используются уже новые значения других неизвестных, подсчитанных на текущей итерации.

В качестве первоначальных значений неизвестных можно использовать любые значения. Например, все значения можно принять равными нулю.

Условием использования метода является диагональное преобладание у матрицы коэффициентов (условие сходимости метода):

$$|a_{ii}| \geq \sum_{i!=j} |a_{ij}|$$

При этом важно, чтобы хотя бы для одного уравнения выполнялось строгое неравенство:

$$|a_{ii}| > \sum_{i!=j} |a_{ij}|$$

Это условие является достаточным, но необязательным.

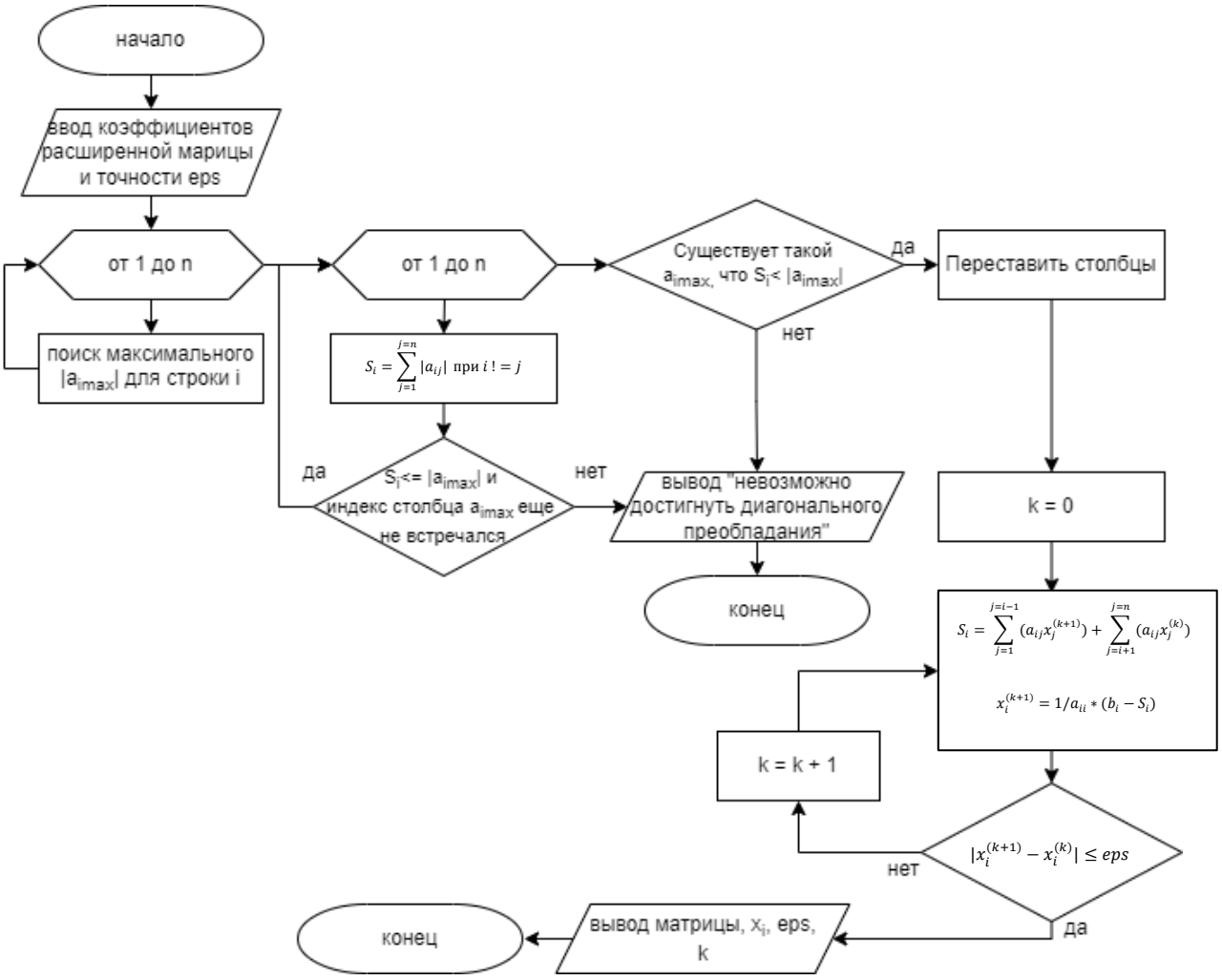
Для подсчета на каждой итерации нового значения x_i используется следующая формула (она получается выражением x_i из i-того уравнения, при этом в ней используются уже посчитанные на текущей итерации значения неизвестных):

$$x_i^{(k+1)} = 1/a_{ii} * (b_i - \sum_{j=1}^{j=i-1} (a_{ij} x_j^{(k+1)}) - \sum_{j=i+1}^{j=n} (a_{ij} x_j^{(k)})$$

Условием окончания выполнения итераций является достижение заданной точности (это является одним из преимуществ данного метода – можно заранее выбрать максимальную допустимую погрешность):

$$|x_i^{(k+1)} - x_i^{(k)}| \leq eps$$

2. Блок-схема численного метода



3. Листинг реализованного численного метода программы

```
private int[] makeSystemDiagonallyDominant() {
    int size = matrix.getSize();
    boolean[] flag = new boolean[size];
    boolean flagForStrictlyMore = false;
    int[] maxElementsRowInd = matrix.findAbsMaxElements();
    int[] maxElementsColumnInd = new int[size];

    for (int i = 0; i < size; i++) {
        double currentSum = 0;
        for (int j = 0; j < size; j++) {
            if (maxElementsRowInd[i] != j) currentSum += Math.abs(matrix.getArray().get(i).get(j));
        }

        if (!flag[i] && Math.abs(matrix.getArray().get(i).get(maxElementsRowInd[i])) >= currentSum) {
            flag[i] = true;
            maxElementsColumnInd[maxElementsRowInd[i]] = i;
            if (Math.abs(matrix.getArray().get(i).get(maxElementsRowInd[i])) > currentSum)
                flagForStrictlyMore = true;
        } else {
            return null;
        }
    }
    int[] permuteResult = new int[size];
    System.arraycopy (maxElementsColumnInd,  srcPos: 0, permuteResult,  destPos: 0, size);

    if (flagForStrictlyMore) {
        for (int j = 0; j < size; j++) {
            for (int i = 0; i < size; i++) {
                matrix.swap(i, maxElementsColumnInd[j], i, j);
            }
            Utils.swapForIntArraySells(maxElementsColumnInd, maxElementsColumnInd[j], j);
        }
        return permuteResult;
    }
    return null;
}
```

```
public IterationMethodAnswer findAnswer() {
    int[] maxElementsColumnInd = makeSystemDiagonallyDominant();
    if (maxElementsColumnInd == null) return null;

    ArrayList<Double> answers = new ArrayList<>(matrix.getSize());
    ArrayList<Double> inaccuracy = new ArrayList<>(matrix.getSize());

    for (int i = 0; i < matrix.getSize(); i++) {
        answers.add(0.0);
    }

    boolean endOfIterations;
    int count = 0;
    ArrayList<Double> old_answers;

    do {
        count++;
        old_answers = new ArrayList<>(answers);
        for (int i = 0; i < matrix.getSize(); i++) {
            double summand = 0.0;
            for (int j = 0; j < matrix.getSize(); j++) {
                if (j != i) summand += matrix.getArray().get(i).get(j) * answers.get(j);
            }
            answers.set(i, 1 / matrix.getArray().get(i).get(i) * (matrix.getArray().get(i).get(matrix.getSize()) - summand));
        }

        endOfIterations = true;
        for (int i = 0; i < matrix.getSize(); i++) {
            if (Math.abs(answers.get(i) - old_answers.get(i)) > eps) {
                endOfIterations = false;
                break;
```

```
            }
        }
    } while(!endOfIterations);

    for (int i = 0; i < matrix.getSize(); i++) {
        inaccuracy.add(answers.get(i) - old_answers.get(i));
    }

    permuteResults(maxElementsColumnInd, inaccuracy);
    permuteResults(maxElementsColumnInd, answers);
    return new IterationMethodAnswer(answers, inaccuracy, count);
}
```

4. Примеры и результаты работы программы на разных данных

```
----Решение СЛАУ методом Гаусса-Зейделя----
Вводить коэфициенты следует построчно, через пробел.
Пример ввода: a1 a2 a3 ... an b1
Введите способ ввода данных (0 - ввод с клавиатуры, 1 - ввод из файла, 2 - рандомные числа, 3 - выход):
1
Введите имя файла:
1.txt
Матрица:
1.0  7.0  3.0  -2.0  6.0
9.0  2.0  2.0  1.0  23.0
2.0  -1.0  3.0  11.0  10.0
3.0  2.0  20.0  -2.0  -8.0
Введите точность (больше 0 и меньше 1):
0.0001
Получены ответы (в порядке X1..Xn):
x1 = 2.404622115495354
x2 = 1.0764485187381394
x3 = -0.7898229658824719
x4 = 0.7851520331564521

Количество итераций: 7
Погрешности (в порядке для X1..Xn):
для x1: -4.945740335493554E-6
для x2: 1.2879631977380512E-5
для x3: -2.735579586188308E-7
для x4: 2.725441889506186E-6
```

```
----Решение СЛАУ методом Гаусса-Зейделя----
Вводить коэфициенты следует построчно, через пробел.
Пример ввода: a1 a2 a3 ... an b1
Введите способ ввода данных (0 - ввод с клавиатуры, 1 - ввод из файла, 2 - рандомные числа, 3 - выход):
0
Введите размер системы (больше 0 и меньше 21):
3
Введите коэффициенты (1 строка):
0 0 0 1
Введите коэффициенты (2 строка):
0 0 0 4
Введите коэффициенты (3 строка):
0 0 0 6
Введите точность (больше 0 и меньше 1):
0.0001
Невозможно достигнуть диагонального преобладания!
```

```
----Решение СЛАУ методом Гаусса-Зейделя----
Вводить коэфициенты следует построчно, через пробел.
Пример ввода: a1 a2 a3 ... an b1
Введите способ ввода данных (0 - ввод с клавиатуры, 1 - ввод из файла, 2 - рандомные числа, 3 - выход):
2
Введите размер системы (больше 0 и меньше 21):
5
Матрица:
431.38  135.62  64.62  -21.94  97.01  -84.39
-142.7  690.9  179.24  -178.47  45.02  75.04
66.44  54.72  670.29  152.12  -120.38  -124.52
11.72  111.06  93.5  544.77  -104.77  -74.51
-75.91  170.46  -21.82  130.2  485.49  -10.15
Введите точность (больше 0 и меньше 1):
0.001
Получены ответы (в порядке X1..Xn):
x1 = -0.19270031225886558
x2 = 0.077370928133579
x3 = -0.15195778993086176
x4 = -0.13186368042269458
x5 = -0.04966857586866245

Количество итераций: 5
Погрешности (в порядке для X1..Xn):
для x1: 1.925126079534989E-4
для x2: 1.573500384525195E-4
для x3: -6.791319354493286E-5
для x4: 6.261790884190543E-5
для x5: -4.499160250310413E-5
```

5. Вывод

Метод Гаусса-Зейделя – итерационный метод, схожий с методом простых итераций. Количество итераций заранее неизвестно, однако данный метод может работать быстрее МПИ, так как на каждом своем шаге использует уже полученные на текущей итерации значения. На скорость работы метода влияют начальные значения; алгоритмическая сложность метода (как и МПИ) o(mn^2). Однако в сравнении с МПИ метод Гаусса-Зейделя имеет свои минусы: труднее подобрать подходящую матрицу, так как условие сходимости более строгое; сложнее параллелизовать, так как значения с текущей итерации уже используются при вычислениях.

Сравнивая итерационные и прямые методы между собой, можно сделать следующие выводы:

- итерационные методы позволяют вычислять решение системы с заданной точностью (в отличие от прямых методов). Погрешность прямых методов зависит от количества промежуточных вычислений (если их достаточно много, то итоговая погрешность может быть значительной); теоретически, погрешность у прямых методов должна отсутствовать.
- прямые методы позволяют найти решение за конечное, известное заранее число операций; количество итераций при решении итерационным методом заранее не известно.
- при использовании прямого метода требуется хранение полной матрицы в памяти, а при использовании итерационных – нет. Это обусловлено тем, что в случае прямого метода исходная матрица меняется в процессе расчетов, а в случае итерационного метода – нет.
- итерационные методы больше подходят для решения больших СЛАУ, а прямые – маленьких/средних (так как итерационные методы позволяют не хранить всю матрицу в памяти).