Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа № 1

по дисциплине «Вычислительная математика»

Системы линейный алгебраических уравнений

Выполнил:

Гурьянов Кирилл Алексеевич

Группа: P32302

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург

2023

1. **Описание метода, расчетные формулы. Метод Гаусса-Зейделя**

Метод Гаусса-Зейделя — итерационный метод, используемый для решения систем линейных уравнений. Неизвестные системы ищутся в течение некоторого числа итераций. Основан на методе простых итераций, однако имеет с ним несколько различий. Главное различие: в методе Гаусса-Зейделя при расчете нового значения x_i используются старые значения других переменных, при условии, что номера других переменных больше чем i. Если же номера других переменных меньше i, то используются значения этих неизвестных, подсчитанные на текущей итерации, а не на предыдущей.

Суть метода заключается в следующем: система приводится к диагональному преобладанию. Затем из каждого уравнения выражается неизвестная, причем из i-го уравнения выражается i-ая неизвестная. После этого выбираются начальные значения неизвестных для выполнения первой итерации. Существует несколько стратегий выбора начальных значений:

- 0
- b_i
- некоторое предварительно рассчитанное значения, чтобы повысить точность результата
- любое другое значение

Таким образом, при использовании метода Гаусса-Зейделя на каждый следующей итерации рассчитывается новое приближение неизвестных. Чем больше будет произведено итераций, тем более точный ответ будет получен.

Необходимым условием использования данного метода (условие сходимости) является возможность приведения матрицы

коэффициентов системы линейных уравнений к диагональному преобладанию:

$$|a_{ii}| \geq \sum_{i \neq k} |a_{ik}| \ (i,\ k = 1,\ 2,\ \ldots,\ n)$$

Но хотя бы для одного уравнения неравенство должно выполняться строго:

$$|a_{ii}| > \sum_{i \neq k} |a_{ik}| \ (i,\ k = 1,\ 2,\ \ldots,\ n)$$

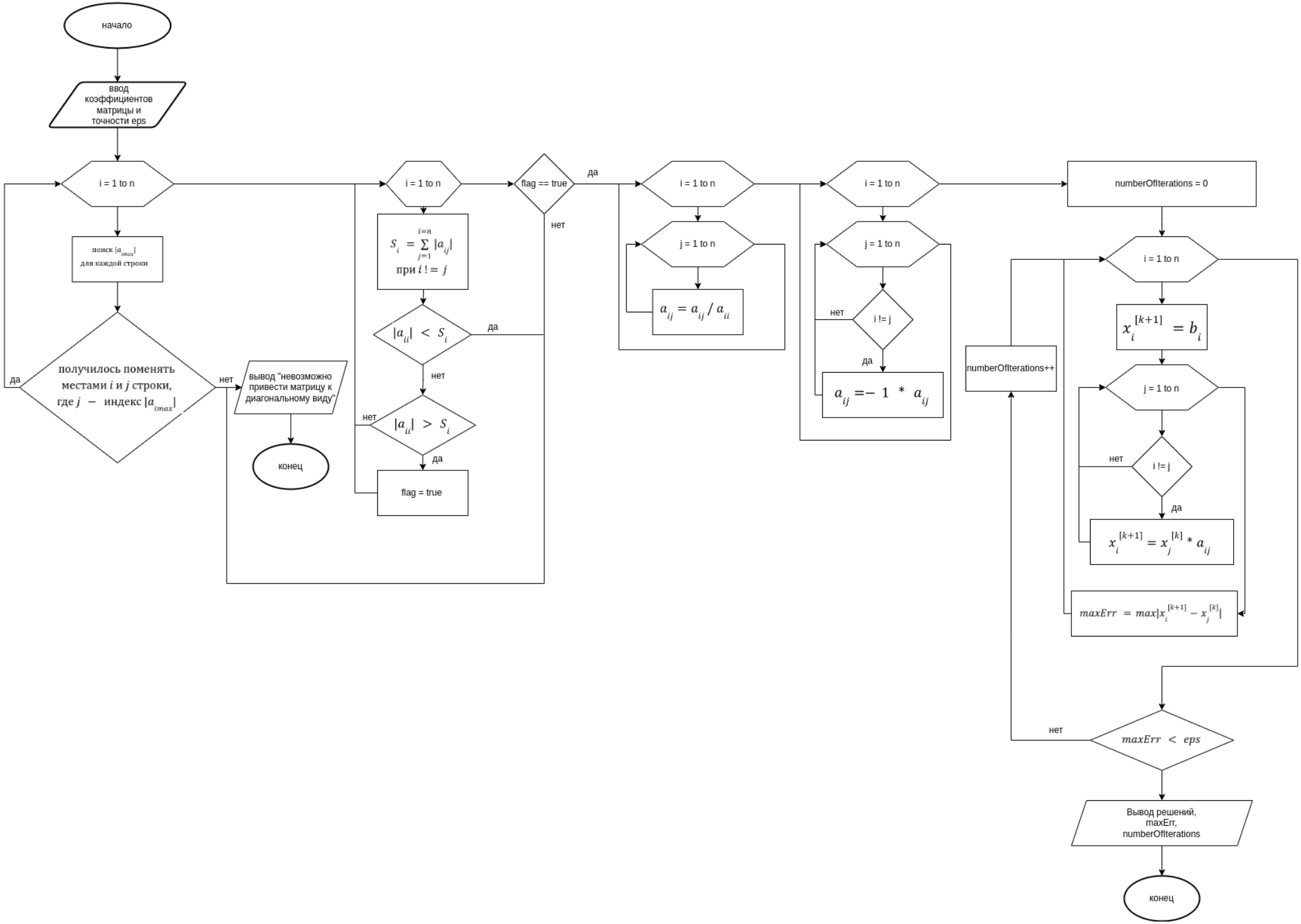
Для подсчета на каждой итерации значения неизвестной используется следующая формула:

$$x_i^{[j+1]} = x_i^{[j]} - \frac{1}{a_{ii}} \left(\sum_{k=1}^{i-1} a_{ik} x_k^{[j+1]} + \sum_{k=i+1}^{n} a_{ik} x_k^{[j]} - b_i \right)$$

Итерации прекращаются при достижении заданной точности:

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

2. Блок-схема численного метода


начало
ввод коэффициентов матрицы и точности eps
i = 1 to n
поиск $|a_{imax}|$ для каждой строки
получилось поменять местами i и j строки, где j – индекс $|a_{imax}|$
да
нет
вывод "невозможно привести матрицу к диагональному виду"
конец
i = 1 to n
$S_i = \sum_{j=1}^{i=n} |a_{ij}|$ при $i != j$
$|a_{ii}| < S_i$
да
нет
$|a_{ii}| > S_i$
нет
да
flag = true
flag == true
да
нет
i = 1 to n
j = 1 to n
$a_{ij} = a_{ij} / a_{ii}$
i = 1 to n
j = 1 to n
нет
i != j
да
$a_{ij} = -1 * a_{ij}$
numberOfIterations = 0
i = 1 to n
$x_i^{[k+1]} = b_i$
j = 1 to n
нет
i != j
да
$x_i^{[k+1]} = x_j^{[k]} * a_{ij}$
$maxErr = max|x_i^{[k+1]} - x_j^{[k]}|$
numberOfIterations++
нет
$maxErr < eps$
Вывод решений, maxErr, numberOfIterations
конец

3. Листинг реализованного метода программы

```
public class GaussSeidelMethod {
    1 usage
    public Answer solveTheSystem(Matrix matrix, double error) throws ArithmeticException, DiagonalDominanceException {
        ArrayList<ArrayList<Double>> matrixCopy = (ArrayList<ArrayList<Double>>) matrix.getMatrix().clone();
        matrixCopy = leadToDiagonalDominance(matrixCopy);
        if (!isDiagonalDominance(matrixCopy))
            throw new DiagonalDominanceException("Матрицу невозможно привести к диагональному преобладанию.");
        prepareForIterations(matrixCopy);
        double[] approximations = new double[matrixCopy.size()];
        double currentError = Double.MAX_VALUE;
        double[] oldApproximations = new double[matrixCopy.size()];
        double[] errors = new double[matrixCopy.size()];
        int numberOfIterations = 0;
        while (currentError > error) {
            for (int i = 0; i < matrixCopy.size(); i++) {
                approximations[i] = matrixCopy.get(i).get(matrixCopy.size());
                for (int j = 0; j < matrixCopy.size(); j++) {
                    if (i != j) {
                        approximations[i] += approximations[j] * matrixCopy.get(i).get(j);
                    }
                }
            }
            currentError = updateError(matrixCopy, approximations, oldApproximations, errors, currentError);
            numberOfIterations++;
        }
        return new Answer(approximations, errors, matrixCopy.size(), numberOfIterations);
    }

    1 usage
    private void prepareForIterations(ArrayList<ArrayList<Double>> matrix) {
        divByTheDiagElement(matrix);
        transferringElements(matrix);
    }
```

```
    private void divByTheDiagElement(ArrayList<ArrayList<Double>> matrix) {
        double diagonalElement;
        for (int i = 0; i < matrix.size(); i++) {
            diagonalElement = matrix.get(i).get(i);
            for (int j = 0; j < matrix.size() + 1; j++) {
                matrix.get(i).set(j, matrix.get(i).get(j) / diagonalElement);
            }
        }
    }

    1 usage
    private void transferringElements(ArrayList<ArrayList<Double>> matrix) {
        for (int i = 0; i < matrix.size(); i++) {
            for (int j = 0; j < matrix.size(); j++) {
                if (j != i) {
                    matrix.get(i).set(j, matrix.get(i).get(j) * (-1));
                }
            }
        }
    }
```

```java
private double updateError(ArrayList<ArrayList<Double>> matrixCopy, double[] approximations, double[] oldApproximations,
                           double[] errors, double currentError) {
    double maxError = 0;
    for (int i = 0; i < matrixCopy.size(); i++) {
        double err = Math.abs(approximations[i] - oldApproximations[i]);
        errors[i] = err;
        if (err > maxError) {
            maxError = err;
        }
    }
    System.arraycopy(approximations, srcPos: 0, oldApproximations, destPos: 0, matrixCopy.size());
    if (maxError < currentError) currentError = maxError;
    return currentError;
}

1 usage
private ArrayList<ArrayList<Double>> leadToDiagonalDominance(ArrayList<ArrayList<Double>> matrixCopy) throws DiagonalDominanceException {
    int matrixSize = matrixCopy.size();
    boolean[] swaps = new boolean[matrixCopy.size()];
    ArrayList<ArrayList<Double>> newMatrix = (ArrayList<ArrayList<Double>>) matrixCopy.clone();
    for (int i = 0; i < matrixSize; i++) {
        double max = 0;
        byte indexOfMax = 0;
        for (byte j = 0; j < matrixSize; j++) {
            if (Math.abs(matrixCopy.get(i).get(j)) > Math.abs(max)) {
                max = matrixCopy.get(i).get(j);
                indexOfMax = j;
            }
        }
        if (!swaps[indexOfMax]) {
            newMatrix.set(indexOfMax, matrixCopy.get(i));
            swaps[indexOfMax] = true;
        } else throw new DiagonalDominanceException("Матрицу невозможно привести к диагональному преобладанию.");
    }
    return newMatrix;
}
```

```java
    private boolean isDiagonalDominance(ArrayList<ArrayList<Double>> matrixCopy) {
        double sumOfRow;
        double diagonalElement;
        boolean isDiagonalDominance = true;
        boolean flag = false;
        int matrixSize = matrixCopy.size();
        for (int i = 0; i < matrixSize; i++) {
            sumOfRow = 0;
            for (int j = 0; j < matrixSize; j++) {
                if (i != j) sumOfRow += Math.abs(matrixCopy.get(i).get(j));
            }
            diagonalElement = matrixCopy.get(i).get(i);
            if (Math.abs(diagonalElement) < sumOfRow) isDiagonalDominance = false;
            if (Math.abs(diagonalElement) > sumOfRow) flag = true;
        }
        return isDiagonalDominance && flag;
    }

}
```

4. Примеры и результаты работы программы на разных данных

```
Решение систем линейных алгебраических уравнений методом Гаусса-Зейделя
Вы уверены, что оно вам нужно? Если нет, то нажмите сочетание клавиш Ctrl+C
Выберите формат ввода данных: 0 - ввод через консоль, 1 - ввод через файл, 2 - генерация случайной матрицы
1
Введите полный путь до файла:
/home/korako2/IdeaProjects/lab_1/src/matrix_1
10.0 2.0 -1.0 -2.0 2.0
-4.0 15.0 3.0 1.0 24.0
2.0 -2.0 20.0 -1.0 4.0
1.0 2.0 -4.0 15.0 2.0
Введите погрешность вычислений от 0,000001 до 0,1:
0,0001
x1 = -0.05986762702687934; погрешность = 1.9430149122895424E-5
x2 = 1.5102128333393654; погрешность = 3.123517087200156E-6
x3 = 0.3585873454219279; погрешность = 3.677389950385024E-6
x4 = 0.03158608946905733; погрешность = 1.0176299018727075E-7
Количество итераций: 5
```

```
Решение систем линейных алгебраических уравнений методом Гаусса-Зейделя
Вы уверены, что оно вам нужно? Если нет, то нажмите сочетание клавиш Ctrl+C
Выберите формат ввода данных: 0 - ввод через консоль, 1 - ввод через файл, 2 - генерация случайной матрицы
0
Введите количество неизвестных в уравнении большее 0 и не превышающее 20: 3
Введите коэффициенты уравнений построчно через пробел.
Пример ввода:
a11 a12 a13 ... a1n b1
a21 a22 a23 ... a2n b2
......................
an1 an2 an3 ... ann bn
1: 1 1 1 1
2: 1 1 1 1
3: 1 1 1 1
Введите погрешность вычислений от 0,000001 до 0,1:
0,1
Матрицу невозможно привести к диагональному преобладанию.
```

```
Решение систем линейных алгебраических уравнений методом Гаусса-Зейделя
Вы уверены, что оно вам нужно? Если нет, то нажмите сочетание клавиш Ctrl+C
Выберите формат ввода данных: 0 - ввод через консоль, 1 - ввод через файл, 2 - генерация случайной матрицы
2
Введите количество неизвестных в уравнении большее 0 и не превышающее 20: 6
38.39185350461358 5.153343074863901 5.399073075803424 5.6813653685222505 9.715783712479158 2.449959283358061 5.183509390127137
1.593020033129846 43.834846028821744 7.233692831352755 8.964468775909378 9.187957556877786 2.966782942034696 1.757680423265665
7.781752398021517 6.634830499789514 50.453941987961215 4.4223925229238406 6.503867975114299 7.277022466694699 5.444473015533805
7.718846797595555 6.940483863329244 6.004600102543139 49.12200340626876 6.610578780093795 4.964895830439625 9.897968547496717
1.8029960584939198 6.505050097897556 0.2017033759533926 5.107884191147326 33.34851651435323 4.7769790365805385 3.6459261132724374
5.759578192740696 1.1502758439041383 6.0267043788297014 6.643448331134732 2.00866979580817 42.03784774805268 7.991821377365575
Введите погрешность вычислений от 0,000001 до 0,1:
0,000001
x1 = 0.08140044420963986; погрешность = 8.410753250809311E-7
x2 = -0.02861700351628472; погрешность = 3.3405141756231993E-7
x3 = 0.05589044422265162; погрешность = 2.184892344547773E-7
x4 = 0.16275702747217263; погрешность = 2.456936819272837E-7
x5 = 0.06477087880494528; погрешность = 1.3419026120842226E-7
x6 = 0.14291169810873663; погрешность = 4.7812227649268735E-8
Количество итераций: 8
```

5. Вывод

Метод Гаусса-Зейделя — итерационный метод, основанный на методе простых итераций. На момент начала алгоритма количество итераций, но данный метод может работать более быстро, нежели метод простых итераций, т.к. на каждой итерации используются полученные на текущей итерации значения приближений. Алгоритмическая сложность — o(mn^2). Скорость работы метода зависит от выбора начальных значений.

Метод Гаусса-Зейделя имеет ряд минусов, в сравнении с методом простой итерации, а именно: его более трудно распараллелить, т.к. значения текущей итерации сразу же используются в вычислениях, а также данный метод имеет более строго условие сходимости, отчего более трудно найти матрицу, для которой данный метод применим.

Проведя анализ итерационных и прямых методов, можно сделать следующие выводы:

а) Прямые методы находят решение системы линейных уравнений за конечное, известное заранее количество операций, в отличие от итерационных методов, где количество итераций на момент старта неизвестно.

б) Итерационные методы больше подходят решения объемных систем линейных алгебраических уравнений, т.к. позволяют при необходимости не хранить всю матрицу в памяти, в отличие от прямых, которые идеально подходят для небольших систем линейных алгебраических уравнений.

в) Итерационные методы дают возможность вычислить решение системы линейных алгебраических уравнений с заданной

точностью, в отличие от прямых методом. Погрешность прямых методов зависит от количества промежуточных вычислений.