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
«Системы линейных алгебраических уравнений»

Вариант «Метод Гаусса-Зейделя»

Группа: P32312

Выполнил: Цю Тяньшэн

Проверил:
Перл Ольга Вычеславовна

Санкт-Петербург
2021г

Содержание

Описание метода, расчётные формулы

Метод Гаусса-Зейделя является итерационным методом для решения СЛАУ, имеющий один параметр ϵ. Достаточное условие для сходимость метода является то, что матрица коэффициентов A является диагонально доминантной, т.е.:

$$|a_{ii}| \geq \sum_{i!=k} |a_{ik}|$$

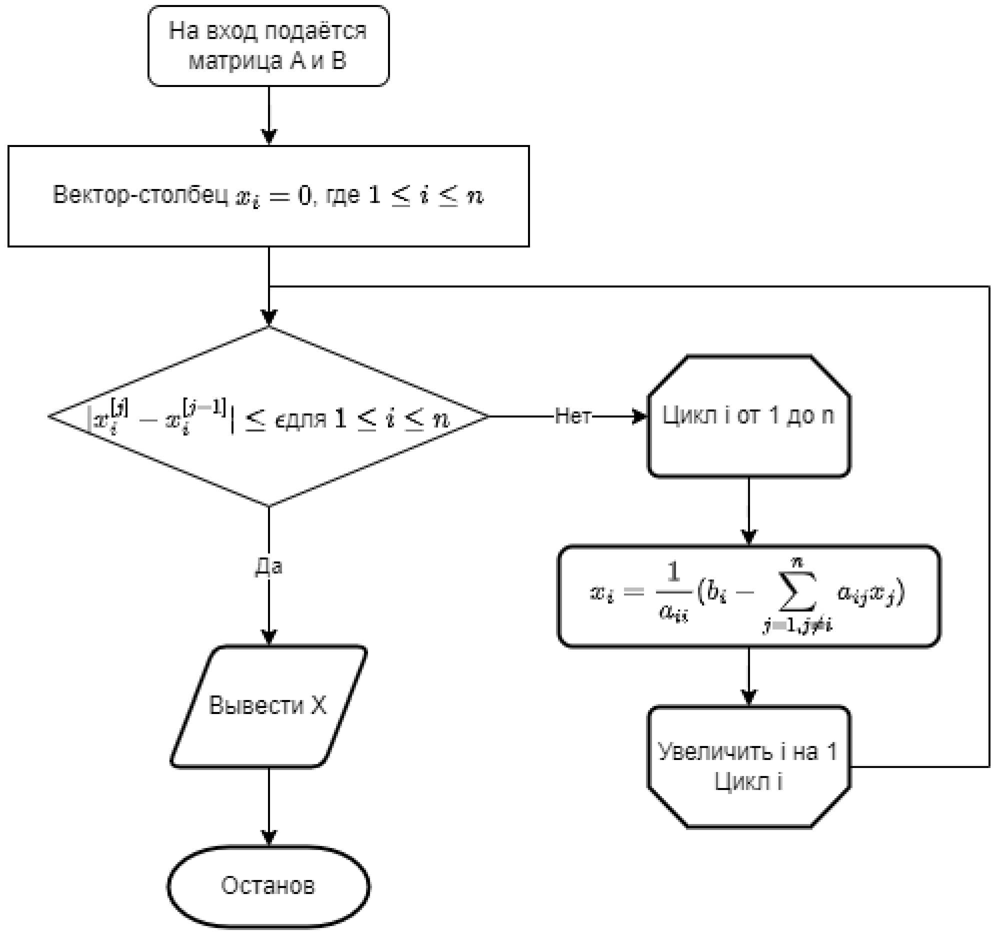
Условие для окончания итерационного процесса:

$$\left| x_i^{[j]} - x_i^{[j-1]} \right| \leq \epsilon$$

Основная формула для вычисление новых значений на каждом шаге итерации, опирающаяся на «свежие» значения:

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

Блок-схема численного метода



Листинг реализованного численного метода программы

```java
private int getDominantElementInRow(Matrix m, int row) {
    double sum = 0;
    for (int i = 0; i < m.getWidth(); i++) sum += Math.abs(m.get(row, i));

    for (int i = 0; i < m.getWidth(); i++) {
        if (sum <= 2 * Math.abs(m.get(row, i))) return i;
    }

    return -1;
}

private boolean makeDiagonallyDominant(LinearSystem system) {
    // Selection sort lol
    for (int i = 0; i < system.a().getHeight(); i++) {
        int iDom = getDominantElementInRow(system.a(), i);
        if (iDom < 0) return false;

        int minRow = i;
        int minCol = iDom;

        for (int j = i+1; j < system.a().getHeight(); j++) {
            int jDom = getDominantElementInRow(system.a(), j);
            if (jDom < 0) return false;

            if (minCol > jDom) {
                minRow = j;
                minCol = jDom;
            }
        }

        // Check if the element is on the diagonal
        if (minCol != i) return false;

        system.a().swapRows(i, minRow);
        system.b().swapRows(i, minRow);
    }

    return true;
}

@Override
public Matrix solve(LinearSystem system, Configuration config) throws
LinearSystemSolvingException {
    Matrix a = system.a();
    Matrix b = system.b();

    if (a.getWidth() != a.getHeight()) throw new
LinearSystemSolvingException("Matrix A of the system is not square. Got " + a);
    if (a.getHeight() != b.getHeight()) throw new
LinearSystemSolvingException("Matrix A does not have the same height as matrix B,
got A: " + a + ", B: " + b);

    if (!makeDiagonallyDominant(system)) throw new
LinearSystemSolvingException("Matrix A is not and can not be transformed to a
diagonally dominant form");

    boolean reachedEpsilon = false;
```

```
    Matrix x = new Matrix(a.getHeight(), 1);
    int iterations = 0;

    while (iterations++ < maxIterations && !reachedEpsilon) {
        reachedEpsilon = true;

        if (config.getFlag(Configuration.DEBUG_FLAG)) System.err.println("Iteration
No." + iterations + " error = " + system.getError(x));

        for (int i = 0; i < a.getHeight(); i++) {
            double s = b.get(i, 0);

            // Get right side sum
            for (int j = 0; j < a.getWidth(); j++) {
                // Skip diagonal elements
                if (i == j) continue;

                s -= x.get(j, 0) * a.get(i, j);
            }

            s /= a.get(i, i);
            if (Math.abs(s - x.get(i, 0)) > eps) reachedEpsilon = false;
            x.set(i, 0, s);
        }
    }

    return x;
}
```

Примеры и результаты работы программы

Пример №1

```
PS C:\Users\dmitr\Desktop\ITMO\comp-math\gauss_seidel> java -jar .\target\gauss_seidel-1.0-SNAPSHOT.jar -s 2 2 -f .\matrix.txt
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
16.00000 3.00000
7.00000 -11.00000

11.00000
13.00000

Iteration No.1 error = 12.0
Iteration No.2 error = 1.1164772727272725
Iteration No.3 error = 0.1332160382231402
Iteration No.4 error = 0.015895095469806672
Iteration No.5 error = 0.001896573891284 17
Iteration No.6 error = 2.262957483925021E-4
0.81219
-0.66497
```

Пример №2

```
PS C:\Users\dmitr\Desktop\ITMO\comp-math\gauss_seidel> java -jar .\target\gauss_seidel-1.0-SNAPSHOT.jar -s 5 5 -f .\matrix3.txt
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
0.00000 2.00000 8.00000 0.00000 0.00000
3.00000 12.00000 2.00000 0.00000 0.00000
10.00000 3.00000 0.00000 4.00000 0.00000
4.00000 0.00000 0.00000 5.00000 9.00000
0.00000 0.00000 0.00000 6.00000 5.00000

1.00000
1.00000
1.00000
1.00000
1.00000

Iteration No.1 error = 1.0
Iteration No.2 error = 0.2384259259259259
Iteration No.3 error = 0.0445358367626885 95
Iteration No.4 error = 0.03165640876644824
Iteration No.5 error = 0.015868095542943 64
Iteration No.6 error = 0.004492452178749762
Iteration No.7 error = 0.0014153365279817808
Iteration No.8 error = 0.0012570157896756085
Iteration No.9 error = 4.449445419745012 6E-4
Iteration No.10 error = 1.319565440738745 7E-4
0.02106
0.05972
0.11007
0.15252
0.01702
```

Пример №3 «Рандомно сгенерированные числа»

```
PS C:\Users\dmitr\Desktop\ITMO\comp-math\gauss_seidel> java -jar .\target\gauss_seidel-1.0-SNAPSHOT.jar -G 0.000001 1000
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
36.67594 -6.07183 -0.03271 -0.92220 0.10887 -1.34119 -1.42483 6.64717 -3.50500 -9.20433
2.48762 56.89725 7.05264 -7.33160 6.65844 -9.82669 -2.38429 4.25050 1.03141 -8.10494
-6.83564 -8.94585 45.44272 -7.45057 1.08268 -0.50277 9.70552 1.06677 3.24441 -0.72583
9.11785 0.47224 6.45394 62.07690 -2.94288 8.03797 -8.27373 6.89296 -9.62771 8.13161
-0.52774 -8.15390 5.95664 -5.45095 53.80639 -2.96657 -3.48513 3.66967 5.47195 -8.26177
-8.92029 4.94863 -3.22348 5.13662 0.92803 54.85387 -9.86930 -8.14618 -9.05905 -3.56133
2.59241 8.35749 -8.52783 -0.00799 -1.73052 -9.01066 49.55553 -4.51713 -3.16929 -8.07517
-0.42782 1.31804 6.40370 5.46378 2.82367 -9.54349 0.28851 47.50049 8.11369 -9.32447
9.74867 -1.27259 -2.83523 -4.38407 3.59785 -6.08394 -7.39767 7.82810 61.90068 -9.36999
-1.93497 6.46561 0.66857 -0.53448 -4.96101 4.32440 -9.73001 -9.82336 9.78090 56.62208

6.23199
-5.04031
0.63824
-2.01417
4.49003
-6.54683
-7.47507
-6.97010
-7.74984
-5.03776

Iteration No.1 error = 5.219435132717282
Iteration No.2 error = 1.3021348651376783
Iteration No.3 error = 0.312562159866395
Iteration No.4 error = 0.03646833791384553
Iteration No.5 error = 0.006228206968439165
Iteration No.6 error = 0.0012147477498136526
Iteration No.7 error = 2.483552207498074 5E-4
Iteration No.8 error = 5.140967028665067E-5
Iteration No.9 error = 7.533617067645437E-6
0.12172
-0.14675
0.04848
-0.05511
0.04657
-0.17385
-0.19402
-0.16292
-0.18380
```

Вывод

После тестирование можно выяснить, что на некоторых размеров данных прямые методы быстрее чем итерационные методы, это происходит, поскольку у прямых методов алгоритмическая сложность $O(n^3)$, а у итерационных методов $O(l * n^2)$, где l – количество итерации, так как мы не можем предсказать, сколько итерации потребует метод, время выполнения методов будет сильно различаться.

А если сравнить итерационные методы между собой, то можно сказать, что метод Гаусса-Зейделя более экономный с точки зрения памяти, так как можно в каждой итерации сразу на ходу переписывать данные в векторе столбце неизвестных, при этом метод Гаусса-Зейделя потребует меньшее количество итерации, чем метод простых итераций, поскольку сразу используются «свежие» значения. Но это одновременно обозначает, что сложнее будет реализовать параллельную обработку данных для метода Гаусса-Зейделя.

Ну и также можно отметить, что из-за строгого условия итерационных методов, они применимы для гораздо меньше случаев, чем прямые методы.