Университет ИТМО

Факультет программной инженерии и компьютерной техники

Вычислительная математика

Лабораторная работа №1

Вариант – Метод Гаусса-Зейделя

Выполнил:

Ким Даниил Кванхенович

Группа:

P3231

Преподаватель:

Перл Ольга Вячеславовна

2022

4 семестр

Описание метода, расчетные формулы:

1) Итерационные методы для решения СЛАУ – метод простых итераций, метод Гаусса-Зейделя – в отличие от прямых методов, не имеют точного количества шагов, за которое будет получен ответ. Это следствие того, что в каждой итерации методы используются значения с предыдущей / начальные значения и на их основе рассчитываются новые значения, которые будут иметь меньшую погрешность. Другими словами, количество шагов, за которое будет найдено, решение зависит от требуемой точности решения и скорости сходимости процесса. Последнее, как правило, не поддается расчетам.

2) Выведем основную формулу на примере СЛАУ с 4 неизвестными и 4 уравнениями:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 \end{cases}$$

$$x_i = \frac{b_i - \sum\limits_{i \neq k} a_{ik} \cdot x_k}{a_{ii}} = \frac{-1}{a_{ii}} \cdot \left(\sum_{i \neq k} a_{ik} \cdot x_k - b_i \right)$$

Сумму по k от 1 до n, не включая k = i ,можно расписать на две суммы:

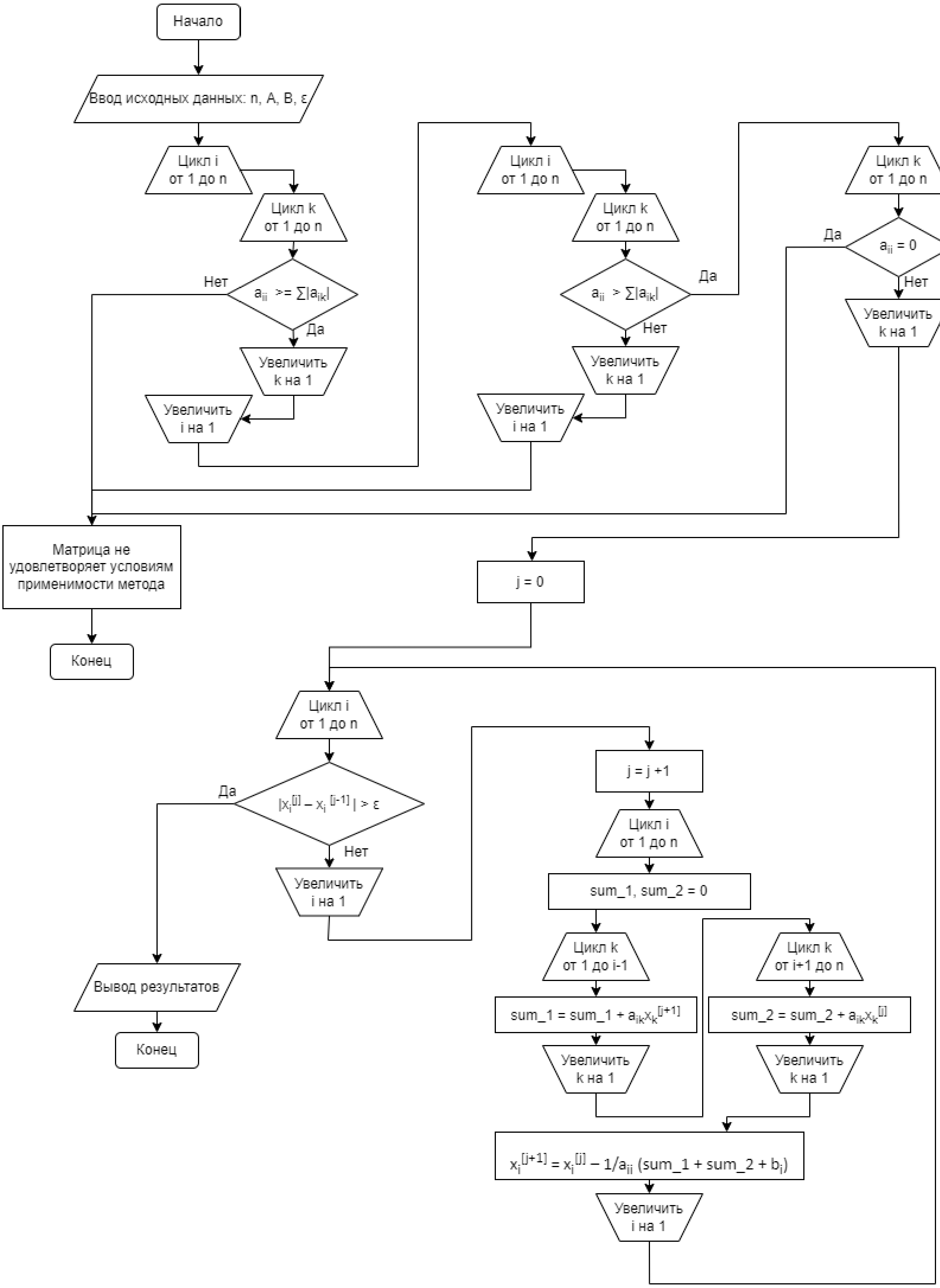
$$\sum_{k \neq i} a_{ik} \cdot x_k = \sum_{k=1}^{i-1} a_{ik} \cdot x_k + \sum_{k=i+1}^{n} a_{ik} \cdot x_k$$

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

В следствие того, что в каждой итерации мы считаем неизвестные один за другим, при подсчете x_i неизвестные x_k, где $1 <= k < i$, будут уже подсчитаны. Их можно поставить в левую сумму. Таким образом, метод повышает скорость схождения процесса используя свежеполученные данные.

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

Блок-схема численного метода:



Листинг реализованного численного метода программы:

1) Численный метод:

```
do {
    iterationCounter += 1;
    System.arraycopy(xCurIter, srcPos: 0, xPrevIter, destPos: 0, dimension);

    for (int i = 0; i<dimension; i++) {
        double sum_1 = 0;
        double sum_2 = 0;
        for (int k = 0; k< i; k++) {
            sum_1 += matrixA[i][k] * xCurIter[k];
        }
        for (int k = i+1; k < dimension; k++) {
            sum_2 += matrixA[i][k] * xPrevIter[k];
        }
        xCurIter[i] =  - (sum_1 + sum_2 - matrixB[i])/matrixA[i][i];
    }
} while ( notPrecise(xPrevIter, xCurIter, squareMatrixWrapper.getEpsilon()) );
```

2) Вспомогательный метод notPrecise () – для проверки факта схождения:

```
private boolean notPrecise (double[] xPrevIter, double[] xCurIter, double epsilon) {
    for (int i = 0; i < xCurIter.length; i++) {
        if (Math.abs((xCurIter[i] - xPrevIter[i])) > epsilon)
            return true;
    }
    return false;
}
```

Примеры работы программы:

1) Все элементы матрицы A равны 0, а значения столбца B – любые не одинаковые числа:

```
Выберите режим ввода данных:
    -пользовательский ввод[0]
    -ввод из файла[1]
    -генерация случайных матриц [2]
0
Введите порядок матрицы [n] :
3
Введите элементы 1 ряда матрицы:
0 0 0 407.323
Введите элементы 2 ряда матрицы:
0 0 0 -3.42
Введите элементы 3 ряда матрицы:
0 0 0 1.2
Введите требуемую погрешность [ε]:
0,0003
Ошибка! Не выполняется строгое условие сходимости метода.
     aii > S( aik ) , k != i
```

2) Все элементы матрицы A равны 1, а значения столбца B – любые не одинаковые числа.

```
Выберите режим ввода данных:
    -пользовательский ввод[0]
    -ввод из файла[1]
    -генерация случайных матриц [2]
1
Ошибка! Не выполняется нестрогое условие сходимости метода.
     |aii| >= S( |aik| ) , k != i
```

3) Любая матрица, определитель которой для самопроверки Вы можете рассчитать при помощи метода Крамера.

```
Выберите режим ввода данных:
    -пользовательский ввод[0]
    -ввод из файла[1]
    -генерация случайных матриц [2]
0
Введите порядок матрицы [n] :
3
Введите элементы 1 ряда матрицы:
5 -3 2 4
Введите элементы 2 ряда матрицы:
0 7 5 12
Введите элементы 3 ряда матрицы:
5.5 -4 10 11.5
Введите требуемую погрешность [ε]:
0.0001
###Проверка пройдена
Количество итераций: 16
Столбец неизвестных:
1,0000213 0,9999669 0,9999750
Столбец погрешностей:
0,0000573 0,0000995 0,0000713
```

4) Матрица 20 x 20 (данная в качестве образца) с вводом из файла:

```
Выберите режим ввода данных:
    -пользовательский ввод[0]
    -ввод из файла[1]
    -генерация случайных матриц [2]
1
###Проверка пройдена
Количество итераций: 9
Столбец неизвестных:
-8,7530000 -9,9290000 8,0470000 8,1260000 0,9440001 -8,4700000 6,9110000 2,7950001 -9,8570001 5,4830000
 -0,9479999 5,4280000 0,6340000 1,4650000 -6,4020000 -0,7020000 1,6260000 9,7880000 -7,1740000 3,3789999
Столбец погрешностей:
0,0000001 0,0000001 0,0000003 0,0000004 0,0000002 0,0000002 0,0000002 0,0000000 0,0000000 0,0000001
 0,0000000 0,0000000 0,0000000 0,0000000 0,0000000 0,0000000 0,0000000 0,0000000 0,0000001 0,0000000
```

5) С генерированием матрицы:

```
Выберите режим ввода данных:
    -пользовательский ввод[0]
    -ввод из файла[1]
    -генерация случайных матриц [2]
2
Введите порядок матрицы [n] :
3
Введите требуемую погрешность [ε]:
0.0001
Стобец сгенерированных неизвестных:
    924.839 -331.922 -700.595
Матрица A
[2354.193]  -550.356    853.643
909.631     [1840.639]  179.644
-148.227    -417.748    [1111.279]

Матрица B
B[1] = 1761866.746574
B[2] = 104455.95807099999
B[3] = -776982.869802

###Проверка пройдена
Количество итераций: 8
Столбец неизвестных:
924,8390091 -331,9220052 -700,5950007
Столбец погрешностей:
0,0000971 0,0000553 0,0000078
```

Вывод:

Как следует из принципа работы метода Гаусса-Зейделя, он позволяет решать СЛАУ с требуемой погрешностью. Расплата за это – малая скорость выполнения по сравнению с прямыми методами. Однако, использование значений с последней итерации при подсчете неизвестных ускоряет процесс сходимости, что даёт ему преимущество по времени над методом простых итераций.

Использование метода накладывает определенные ограничения на входные данные. Необходимо преобразовывать их в угоду выполнения условий сходимости. С точки зрения этих ограничений метод простых итераций выглядит выигрышнее, так как условие там менее строгое.

С алгоритмической сложность итерационных методов не все так просто. На каждой итерации для всех x_i рассчитывается сумма n-1 элементов. Выходит, что общая производительность алгоритма равна O(кол-во итераций * n*(n-1)). Эффективность работы алгоритма зависит от количества итераций, которое на деле никак не ограничена. Таким образом, в худшем случае итерационные методы менее эффективны, чем прямые методы.

Говоря о численных ошибках, могу опять сослаться на принцип работы итерационных методов. Расчеты можно производить до тех пор, пока погрешность нас не будет удовлетворять.