ИТМО Кафедра Информатики и прикладной математики

Отчет по лабораторной работе №1 «Решение системы линейных алгебраических уравнений СЛАУ» Вариант : метод простых итераций

Выполнил: студент группы P3217

Плюхин Дмитрий

Преподаватель: Калёнова О. В.

2016 год

1. Описание метода

Метод простых итераций – приближенный численный метод решения СЛАУ, эффективный в тех случаях, когда имеем дело с большим числом неизвестных и решение методом Гаусса становится весьма сложным.

Так, если дана линейная система

$$\begin{cases} \boldsymbol{a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1} \\ \boldsymbol{a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_1} \\ \dots \\ \boldsymbol{a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n} \end{cases}$$

То введя в рассмотрение матрицы, состоящие из коэффициентов при переменных (A), самих переменных (x) и свободных членов (b) соответственно, можно записать систему в следующем виде:

$$Ax = b$$

При использовании метода итераций сначала производится преобразование матрицы коэффициентов таким образом, чтобы модули диагональных коэффициентов были велики по сравнению с модулями недиагональных коэффициентов, в этом случае обеспечивается хорошее схождение дальнейшего процесса итераций.

Далее производится разрешение первого уравнения системы относительно x_1, второго – относительно x_2 и так далее по соответствующим формулам:

$$\begin{cases} \boldsymbol{\beta_1 + \alpha_{12}x_2 + \cdots + \alpha_{1n}x_n = x_1} \\ \boldsymbol{\alpha_{21}x_1 + \beta_2 \dots + \alpha_{2n}x_n = x_2} \\ \dots \\ \boldsymbol{\alpha_{n1}x_1 + \alpha_{n2}x_2 + \cdots + \beta_n = x_n} \end{cases}$$

где $\beta_i = \frac{b_i}{a_{ii}}$ $\quad \alpha_i = -\frac{a_{ij}}{a_{ii}}$ $\quad$ при $i \neq j$

Обозначим α матрицу новых коэффициентов, а β – матрицу новых свободных членов

Новая система решается методом последовательных приближений. За нулевое приближение, например, принимается столбец свободных членов ($x^{(0)} = \beta$), далее любое k+1 – е приближение вычисляют по формуле

$$x^{(k+1)} = \beta + \alpha x^{(k)}$$

Приближения считаются до тех пор, пока не будет достигнута необходимая точность, в этом и заключается итерационный процесс.

Формулы приближений в развернутой форме имеют следующий вид:

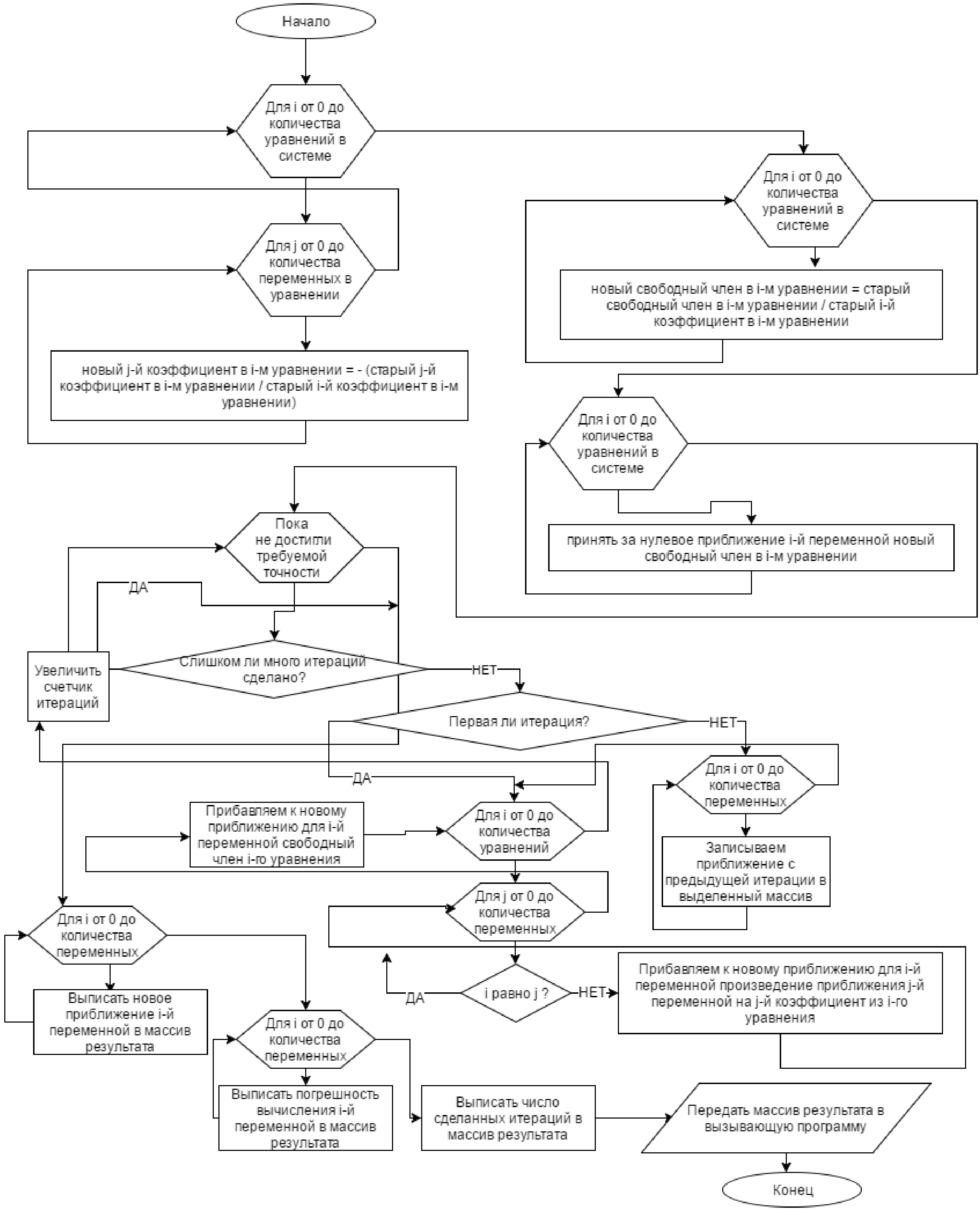
$$\begin{cases} \boldsymbol{x_i^{(0)} = \beta_i} \\ \boldsymbol{x_i^{(k+1)} = \beta_i + \sum_{j=1}^{n} \alpha_{ij}x_j^{(k)}} \\ \dots \\ \boldsymbol{(\alpha_{ii} = 0; i = 1, \dots, n; k = 0, 1, 2, \dots)} \end{cases}$$

2. Листинг основной части программы

```
public static double[][] resolveSystem(double[][] koefficients, double[] freeMembers, double accuration){

        double[][] answer = new double[3][koefficients.length];
        double[][] koefficientsModified = new double[koefficients.length][koefficients[0].length];
        double[] freeMembersModified = new double[freeMembers.length];
        double[] approach = new double[freeMembers.length];
        double[] approachTmp = new double[freeMembers.length];
        int numOfIteration = 0;

        // Modifying koefficients
        for (int i = 0; i < koefficients.length; i++){
            for (int j = 0; j < koefficients[0].length; j++){
                koefficientsModified[i][j] = - koefficients[i][j]/koefficients[i][i];
            }
        }
```

```java
        //Modifying free members
        for (int i=0; i < freeMembers.length; i++){
            freeMembersModified[i] = freeMembers[i]/koefficients[i][i];
        }

        // Zero's approach
        for (int i=0; i<approach.length; i++){
            approach[i] = freeMembersModified[i];
        }

        //Iterative process
        do {
            if (numOfIteration > NumberConstants.MAX_NUM_OF_ITERATIONS){
                break;
            }

            if (numOfIteration > 0){
                for (int i = 0; i < koefficients.length; i++){
                    approach[i] = approachTmp[i];
                }
            }

            for (int i = 0; i < koefficients.length; i++){
                approachTmp[i] = 0;
                for (int j = 0; j < koefficients[0].length; j++){
                    if (i!=j){
                        approachTmp[i] += approach[j]*koefficientsModified[i][j];
                    }
                }
                approachTmp[i] += freeMembersModified[i];
            }

            numOfIteration ++;

        } while (countMediumAccuration(approach, approachTmp) >= accuration);

        // Returning result
        for (int i = 0; i < answer[0].length; i++){
            answer[0][i] = approachTmp[i];
        }


        for (int i = 0; i < answer[1].length; i++){
            answer[1][i] = Math.abs(approachTmp[i]-approach[i]);
        }

        answer[2][0] = numOfIteration;

        return answer;
    }
public static boolean diagonalDomination(double[][] koefficients, int[] numXs){

        boolean isDiagonalDomination = true;

        check:
        for (int i = 0; i < koefficients.length; i++){
            for (int j = 0; j < koefficients[0].length; j++){
                if (koefficients[i][j] > koefficients[i][i]){
                    isDiagonalDomination = false;
                    break check;
                }
            }
        }

        if ((!isDiagonalDomination) && (!tryDominate(koefficients, numXs)) || (hasDiagonalZeros(koefficients))){
            throw new IllegalArgumentException();
        }

        return isDiagonalDomination;
    }
public static boolean isSystemCorrect(double[][] matrix){
        if (getDeterminant(matrix, new int[matrix.length], 0) != 0){
            return true;
        }
        return false;
    }
```

3. Блок – схема численного метода



4. Примеры и результаты работы программы

Пример 1. Уравнение из 6 переменных:

Коэффициенты:
0.1 0.2 0.3 0.4 0.5 6
0.1 0.2 0.3 0.4 5 0.6
0.1 0.2 0.3 4 0.5 0.6
0.1 0.2 3 0.4 0.5 0.6
0.1 2 0.3 0.4 0.5 0.6
1 0.2 0.3 0.4 0.5 0.6
Свободные члены:
1 1 1 1 1 1
Точность:
0.001

```
Here is answer for your question:
Column of variables:
x1 = 0.666015625
x2 = 0.3330078125
x3 = 0.22200520833333331
x4 = 0.16650390625
x5 = 0.133203125
x6 = 0.11100260416666666
Column of accurations:
x1 = 0.001953125
x2 = 9.765625E-4
x3 = 6.510416666666574E-4
x4 = 4.8828125E-4
x5 = 3.9062500000000555E-4
x6 = 3.255208333333426E-4
Resolve founded after 9.0 iterations.
```

Пример 2. Уравнение из 5 переменных:
Коэффициенты:
-1 3 5 -7 9
12 4 -8 1 2
2 3 -5 -11 2
9 19 -1 -2 3
3 13 16 11 -3
Свободные члены:
1 2 3 4 5
Точность:
0.01

```
Here is answer for your question:
Column of variables:
x1 = 0.559055218378809
x2 = -0.005849639658775302
x3 = 0.4373961535151804
x4 = -0.44474409264980375
x5 = -0.40251454749360677
Column of accurations:
x1 = 0.01582044021515583
x2 = 8.863076703419237E-5
x3 = 0.007095659413322852
x4 = 0.011575800310624662
x5 = 0.0021540993506091732
Resolve founded after 8.0 iterations.
```

Пример 3. Уравнение из 3 переменных:
Коэффициенты:
4 0.24 -0.08
0.09 3 -0.15
0.04 -0.08 4
Свободные члены:
8 9 20
Точность:
0.001

```
Here is answer for your question:
Column of variables:
x1 = 1.909228
x2 = 3.194948
x3 = 5.044794
Column of accurations:
x1 = 1.72000000000061E-4
x2 = 5.48000000000215E-4
x3 = 1.939999999995834E-4
Resolve founded after 3.0 iterations.
```

5. Вывод

В результате лабораторной работы был изучен новый способ решения СЛАУ, который показался мне более простым, чем метод Гаусса, но требующим проведение большего количества вычислений. Также я освежил в памяти принципы построения блок-схем, описывающих действие алгоритмов. Первая лабораторная работа потребовала достаточно большого количества времени и усилий, однако все они будут оправданы в том случае, если в будущем у меня появится необходимость реализовать приложение, решающее СЛАУ автоматически. Полученные знания можно применить и в математике при дальнейшем изучении темы решения СЛАУ.