Министерство просвещения Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий
Кафедра прикладной математики

Отчет защищен с оценкой_____
Преподаватель Кантор С.А
«___» _____ 2021 г.

Отчет
по лабораторной работе № 3

«Решение СЛАУ методами Якоби и Зейделя»

по дисциплине
«Вычислительная математика»

Студент группы ПИ-81 (Б): И. А. Песняк

Преподаватель: доцент, к.ф-м.н., Кантор С. А.

Барнаул 2021

Задание:

Составить программу для решения системы линейных алгебраических уравнений методами Якоби и Зейделя. Исходные данные - матрица системы уравнений и столбец свободных членов, точность eps должна читаться из файла, а результаты расчетов помещаться в файл. Предусмотреть вывод числа итераций, необходимых для получения решения с заданной точностью eps.

Исследовать зависимость числа итераций от начального приближения, точности, выбора метода решения.

Изучить влияние сходимости величины диагонального преобладания матрицы, то есть величины отношения суммы модулей недиагональных элементов строки к модулю диагонального элемента.

Подобрать примеры, показывающие, что диагональное преобладание не является необходимым условием сходимости.

Краткое описание:

Метод Якоби и метод Зейделя используются для решения СЛАУ итерационным подходом. В методе Якоби решение ищется как предел последовательности

$$x^{(k+1)} = Bx^{(k)} + c,$$

где

$$B = -D^{-1}A_0, \text{с} = D^{-1}b,$$

$$\text{если } i = j: D_{ij} = A_{ij}, \text{иначе } D_{ij} = 0,$$

$$A_0 = A - D,$$

а в методе Зейделя:

$$x^{(k+1)} = B_1x^{(k+1)} + B_2x^{(k)} + c,$$

где элементами матрицы B_1 являются элементы матрицы B ниже главной диагонали, матрицы B_2 элементы матрицы B выше главной диагонали, остальные элементы равны нулю.

Метод Зейделя можно рассматривать как модификацию метода Якоби. Основная идея модификации состоит в том, что новые значения используются здесь сразу же по мере получения, в то время как в методе Якоби они не используются до следующей итерации.

Для обоих методов справедливо следующее: итерационный процесс сходится ⇔ все собственные числа матрицы B по модулю меньше 1.

На практике это условие сложно быстро проверить, поэтому используется достаточное условие: матрица должна обладать свойством строго диагонального преобладания:

$$\max_{i=1,\dots,n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} < 1$$

В качестве начальных приближений рассматриваются следующие три варианта:

1) $x_i = 0$
2) $x_i = b_i$
3) $x_i = \frac{b_i}{a_{ii}}$

В среднем, самым неудачным приближением оказалось второе, при третьем приближении чаще всего выходит на одну итерацию меньше, чем при первом (что, впрочем, неудивительно).

В качестве условия остановки итерационного процесса было выбрано следующее неравенство:

$$\left\| x^{(k+1)} - x^k \right\| < \varepsilon$$

В процессе выполнения лабораторной было замечено, что методы Якоби и Зейделя хоть и не всегда сходятся, когда матрицы верхне-треугольные или нижне-треугольные, но все равно дают верный результат, так как вычисления становятся похожи на обратный ход Гаусса. В случае если матрица нижне-треугольная и применяется метод Зейделя, то получается обратный ход Гаусса в чистом виде и точное решение достигается за одну итерацию, в других случаях с треугольными матрицами точное решение достигается за n итераций (см. тест №5).

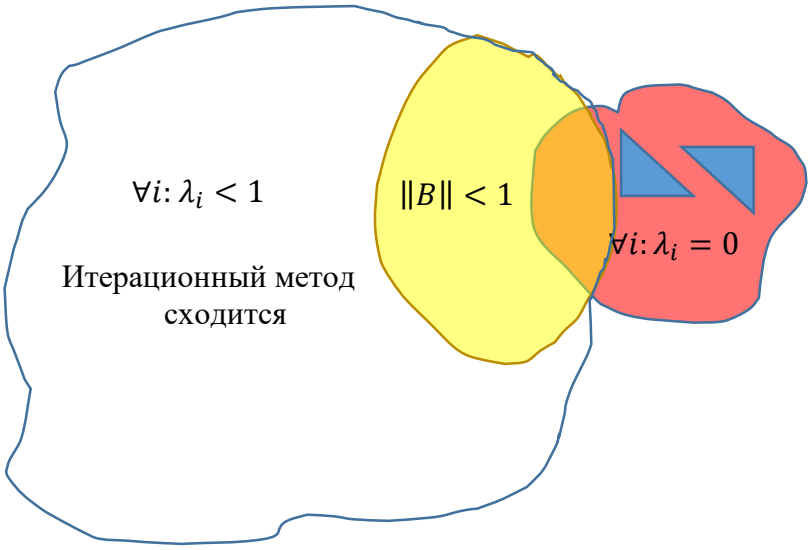

Рис 1. Область применимости методов.

Тестовые примеры:

#	n	A b	ε	x_0	Метод Якоби		Метод Зейделя	
					Итераций	Норма невзяки	Итераций	Норма невзяки
1	3	10 1 3 4 4 33 2 5 4 2 55 6	0,1E-6	1	10	3.07E-6	5	3.98E-8
			0,1E-12	1	19	2.57E-12	9	1.02E-14
			0,1E-6	2	12	4.14E-6	6	1.252E-7
			0,1E-12	2	21	3.40E-12	10	1.02E-14
			0,1E-6	3	9	3.07E-6	4	1.64E-8
			0,1E-12	3	18	2.57E-12	9	4.44E-15
2	5	100 3 2 4 90 78 30 544 54 3 450 56 54 54 345 165 65 543 345 534 554 4637 3133 5567 543 534 43 4953 6678 53468	0,1E-6	1	356	0.0022	32	5.22E-4
			0,1E-12	1	647	2.25E-9	56	5.78E-10
			0,1E-6	2	554	0.0022	46	5.70E-4
			0,1E-12	2	845	2.27E-9	71	3.765E-10
3	5	100 3 2 4 90 78 30 544 54 3 450 56 54 54 345 165 65 543 345 534 554 4537 3133 5567 543 534 43 4953 6678 53468	0,1E-6	1	423	0.0023	33	5.08E-4
			0,1E-12	1	778	2.21E-9	58	4.88E-10
			0,1E-6	2	666	0.0022	48	3.97E-4
			0,1E-12	2	1015	2.21E-9	73	3.82E-10
4	3	25 18 2,5 3 4 12 1 4 2 3 1 67	0,1E-6	1	413	5.85E-6	19	4.26E-7
			0,1E-12	1	742	5.96E-12	32	1.46E-12
5	4	2 0 0 0 5 4 3 0 0 4 67 3 9 0 53 3 4 5 2 6	0,1E-12	1	5	1.42E-14	2	1.42E-14
			0,1E-12	2	5	1.42E-14	2	1.42E-14
5	1000	Матрица со строгим диагональным преобладанием	0,1E-6	1	6	7.47E-6	5	6.40E-8
			0,1E-12	1	8	4.14E-8	7	4.14E-8

Пояснения к тестам:

1) $\max\limits_{i=1,\dots,n} \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = 0.4$

2) $\max\limits_{i=1,\dots,n} \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = 0.99$, условие диагонального преобладания выполняется

3) $\max\limits_{i=1,\dots,n} \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = 1.006$, но все собственные числа матрицы **B** по модулю меньше единицы, поэтому метод сходится.

4) $\max\limits_{i=1,\dots,n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = 5$, но все собственные числа матрицы **B** по модулю меньше единицы, поэтому метод сходится.

1. $\lambda_1 = -0{,}5$
2. $\lambda_2 = -0{,}45887$
3. $\lambda_3 = 0{,}95887$

5) $\max\limits_{i=1,\dots,n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{|a_{ij}|}{|a_{ii}|} = 7.77$, $\lambda = 0$, нижне-треугольная матрица, случай описан выше.

Пример работы программы:

```
Стандартная погрешность: 0,000001
1. Считать матрицу из файла
2. Сгенирировать матрицу с диагональным преобладанием
3. Решение методом Якоби
4. Решение методом Зейделя
5. Задать точность
6. Выбрать начальное приближение
7. Выход
1
1. Считать матрицу из файла
2. Сгенирировать матрицу с диагональным преобладанием
3. Решение методом Якоби
4. Решение методом Зейделя
5. Задать точность
6. Выбрать начальное приближение
7. Выход
3
x1 = -12,912 x2 = -12,884 x3 = 6,367 x4 = -7,824 x5 = 15,849
Норма невзяки: 0.002240030044958985
Количество итераций: 356
1. Считать матрицу из файла
2. Сгенирировать матрицу с диагональным преобладанием
3. Решение методом Якоби
4. Решение методом Зейделя
5. Задать точность
6. Выбрать начальное приближение
7. Выход
4
x1 = -12,912 x2 = -12,884 x3 = 6,367 x4 = -7,824 x5 = 15,849
Норма невзяки: 5.227671153988922E-4
Количество итераций: 32
1. Считать матрицу из файла
2. Сгенирировать матрицу с диагональным преобладанием
3. Решение методом Якоби
4. Решение методом Зейделя
5. Задать точность
6. Выбрать начальное приближение
7. Выход
6
1. Заполнить нулями
2. Заполнить столбцом свободных членов
3. Заполнить b[i]/A[i][i]
```

Код программы:

```
package cm;

import java.io.*;
import java.util.Random;
import java.util.Scanner;


import static java.lang.Math.*;

public class Main {
    interface Expression{
        int method(double[][] A, double[] x, double e, double[] b, int n);
    }
    //норма l1
    static double vectorNorm(double[] v, int n) {
        double res = 0;
        for(int i = 0; i < n; i++){
            res += abs(v[i]);
        }
        return res;
    }

    static int jacobi(double[][] A, double[] x, double e, double[] b, int n){
        double[] new_x = new double[n];
        double[] dx = new double[n];
        int amountOfIterations = 0;
        double normOfDX = e + 1;
        while(normOfDX >= e) {
            amountOfIterations++;
            for (int i = 0; i < n; i++) {
                double sum = 0;
                for(int j = 0; j < n; j++){
                    if(i != j){
                        sum += A[i][j] * x[j];
                    }
                }
                new_x[i] = - sum / A[i][i] + b[i] / A[i][i];
            }
            //вычичсление разницы между старым и новым решением
            // и сохранение нового решения
            for (int i = 0; i < n; i++) {
                dx[i] = new_x[i] - x[i];
                x[i] = new_x[i];
            }
            normOfDX = vectorNorm(dx, n);
        }
        return amountOfIterations;
    }

    static int seidel(double[][] A, double[] x, double e, double[] b, int n){
        double[] new_x = new double[n];
        double[] dx = new double[n];
        int amountOfIterations = 0;
        double normOfDX = e + 1;
        while(normOfDX >= e) {
            amountOfIterations++;
            for (int i = 0; i < n; i++) {
                double sum = 0;
                for(int j = 0; j < n; j++){
                    if(i > j){
                        sum += A[i][j] * new_x[j];
                    }
                    else if(i < j){
                        sum += A[i][j] * x[j];
                    }
                }
                new_x[i] = - sum / A[i][i] + b[i] / A[i][i];

            }
            //вычичсление разницы между старым и новым решением
            // и сохранение нового решения
            for (int i = 0; i < n; i++) {
                dx[i] = new_x[i] - x[i];
                x[i] = new_x[i];
            }
            normOfDX = vectorNorm(dx, n);
        }
```

```
        return amountOfIterations;
    }

    //вычисление невзяки
    static double[] discrepancy(double[][] A, double[] x, double[] b, int n){
        double[] discrepancy = new double[n];
        for (int i = 0; i < n; i++){
            discrepancy[i] = 0;
            for(int j = 0; j < n; j++){
                discrepancy[i] += A[i][j] * x[j];
            }
            discrepancy[i] -= b[i];
        }
        return discrepancy;
    }

    //Выполняется ли диагональное преобладание
    static boolean checkDiagonalDomination(double[][] A, int n){
        boolean res = true;
        for(int i = 0; i < n; i++){
            double sum = 0;
            for(int j = 0; j < n; j++){
                if(i != j)
                    sum += A[i][j];
            }
            if(A[i][i] <= sum)
                res = false;
        }
        return res;
    }

    static void solve(Expression method, double[][] A, double[] x, double e, double[] b, int n,
double[] x0){
        if(!checkDiagonalDomination(A, n))
            System.out.println("Диагональное преобладание не выполняется");
        if (n >= 0) System.arraycopy(x0, 0, x, 0, n);
        int amountOfIterations = method.method(A, x, e, b, n);
        for (int i = 0; i < n; i++){
            System.out.printf("x%d = %5.3f ", i + 1,  x[i]);
        }
        System.out.println();
        double[] discrepancy = discrepancy(A, x, b, n);
        System.out.print("Норма невзяки: " + vectorNorm(discrepancy, n) + "\n");
        System.out.println("Количество итераций: " + amountOfIterations);
    }




    public static void main(String[] args) throws FileNotFoundException {
        double e = 0.000001;
        double[][] A = null;
        double[] b = null;
        double[] x = null;
        double[] x0 = null;
        int n = 1;

        int option;
        boolean isMatrixCreated = false;
        System.out.printf("Стандартная погрешность: %f\n", e);
        do {
            System.out.println("1. Считать матрицу из файла");
            System.out.println("2. Сгенирировать матрицу с диагональным преобладанием");
            System.out.println("3. Решение методом Якоби");
            System.out.println("4. Решение методом Зейделя");
            System.out.println("5. Задать точность");
            System.out.println("6. Выбрать начальное приближение");
            System.out.println("7. Выход");
            Scanner scanner = new Scanner(System.in);
            option = scanner.nextInt();

            switch (option) {
                case 1 -> {
                    Scanner fin = new Scanner(new File("in.txt"));
                    n = fin.nextInt();
                    A = new double[n][n];
                    b = new double[n];
                    x0 = new double[n];
                    x = new double[n];
                    //чтение матрицы из файла
```

```
                    for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++) {
                            A[i][j] = fin.nextDouble();
                        }
                        b[i] = fin.nextDouble();
                    }
                    isMatrixCreated = true;
                }

                case 2 -> {
                    System.out.print("Введите размерность матрицы: ");

                    n = scanner.nextInt();

                    A = new double[n][n];
                    b = new double[n];
                    x0 = new double[n];
                    x = new double[n];

                    //генерация матрицы
                    Random random = new Random();
                    double sum = 0;
                    for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++) {
                            if (i != j) {
                                A[i][j] = random.nextDouble();
                                sum += abs(A[i][j]);

                            }
                            A[i][i] = sum + abs(random.nextDouble());

                        }
                        b[i] = random.nextDouble()*100000;
                    }
                    isMatrixCreated =  true;

                    /*PrintWriter pw = new PrintWriter("generatedMatrix.txt");
                    pw.println(n);
                    for (int i = 0; i < n; i++) {
                        for (int j = 0; j < n; j++) {
                            pw.print(A[i][j] + " ");
                        }
                        pw.println(b[i]);
                    }
                    pw.close();*/
                }


                //решение методом Якоби
                case 3 -> {
                    if (isMatrixCreated) {
                        solve(Main::jacobi, A, x, e, b, n, x0);
                    }

                }
                //Решение методом Зейделя
                case 4 -> {
                    if (isMatrixCreated) {
                        solve(Main::seidel, A, x, e, b, n, x0);
                    }
                }

                case 5 -> {
                    System.out.print("Введите точность: ");
                    e = scanner.nextDouble();
                }
                case 6 -> {
                    if (isMatrixCreated) {
                        int internalOption;
                        System.out.println("1. Заполнить нулями");
                        System.out.println("2. Заполнить столбцом свободных членов");
                        System.out.println("3. Заполнить b[i]/A[i][i]");
                        internalOption = scanner.nextInt();
                        switch (internalOption) {
                            case 1: {
                                for (int i = 0; i < n; i++) {
                                    x0[i] = 0;
                                }
                            }break;
                            case 2: {
```

```
                                System.arraycopy(b, 0, x0, 0, n);
                            }break;
                            case 3:
                                for (int i = 0; i < n; i++) {
                                    x0[i] = b[i] / A[i][i];
                                }
                        }
                    }
                }


            }
        }while (option != 7);

    }
}
```