БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №1

Метод Гаусса решения систем линейных алгебраических уравнений

Вариант 1

Выполнил:

Черепенников Роман

2 курс 8 группа

Преподаватель:

Радкевич Елена Владимировна

Содержание

Постановка задачи

1. Методом Гаусса найти решение системы линейных алгебраических уравнений.
2. Вычислить определитель матрицы.
3. Найти обратную матрицу, сделать проверку.
4. Вычислить вектор невязки.

Алгоритм решения

Пусть есть система

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots \\ a_{p1}x_1 + a_{p2}x_2 + \ldots + a_{pn}x_n = b_p \end{cases}$$

Будем считать, что a11≠0, так как мы всегда можем этого добиться перестановкой местами уравнений системы. Исключим неизвестную переменную x_1 из всех уравнений системы, начиная со второго. Для этого ко второму уравнению системы прибавим первое, умноженное на a21a11, к третьему уравнению прибавим первое, умноженное наa31a11, и так далее, к n-ому уравнению прибавим первое, умноженное на -an1a11. Система уравнений после таких преобразований примет вид

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots + a_{3n}^{(1)}x_n = b_3^{(1)} \\ \ldots \\ \qquad a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases}$$

где aij(1)= aij+a1j-ai1a11, i=2,3,…,n, j=2,3,…,n

bi(1)=bi+ b1-ai1a11, i=2,3,…,n.

К такому же результату мы бы пришли, если бы выразили x1 через другие неизвестные переменные в первом уравнении системы и полученное выражение подставили во все остальные уравнения. Таким образом, переменная x1 исключена из всех уравнений, начиная со второго.

Далее действуем аналогично, но лишь с частью полученной системы, которая отмечена на рисунке

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots + a_{3n}^{(1)}x_n = b_3^{(1)} \\ \ldots \\ \qquad a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots + a_{nn}^{(1)}x_n = b_n^{(1)} \end{cases}$$

Будем считать, что a22(1)≠0 (в противном случае мы переставим местами вторую строку с k-ой, где a2k(1)≠0). Приступаем к исключению неизвестной переменной x2 из всех уравнений, начиная с третьего.

Для этого к третьему уравнению системы прибавим второе, умноженное на-a32(1)a22(1), к четвертому уравнению прибавим второе, умноженное на -

a42(1)a22(1), и так далее, к n-ому уравнению прибавим второе, умноженное на -an2(1)a22(1). Система уравнений после таких преобразований примет вид

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\qquad a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ \ldots \\ \qquad\qquad a_{n3}^{(2)}x_3 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)} \end{cases}$$

где aij(2)= aij(1)+a2j(1)-ai21a221, i=3, 4,…,n, j=3, 4,…,n , а
bi(2)=bi(1)+ b2(1)-ai21a221, i=3, 4,…,n. . Таким образом, переменная x2 исключена из всех уравнений, начиная с третьего.

Далее приступаем к исключению неизвестной x3, при этом действуем аналогично с отмеченной на рисунке частью системы

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\qquad a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ \ldots \\ \qquad\qquad a_{n3}^{(2)}x_3 + \ldots + a_{nn}^{(2)}x_n = b_n^{(2)} \end{cases}$$

Так продолжаем прямой ход метода Гаусса пока система не примет вид

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1 \\ \qquad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_{2n}^{(1)}x_n = b_2^{(1)} \\ \qquad\qquad a_{33}^{(2)}x_3 + \ldots + a_{3n}^{(2)}x_n = b_3^{(2)} \\ \ldots \\ \qquad\qquad\qquad a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases}$$

С этого момента начинаем обратный ход метода Гаусса: вычисляем xn из последнего уравнения как xn=bn(n-1)/ann(n-1) , с помощью полученного значения xn находим xn-1 из предпоследнего уравнения, и так далее, находим x_1 из первого уравнения.

Определитель матрицы A вычисляется следующим образом:

det A = a11*a22(1)*a33(2)*…ann(n-1), где

aii(i-1) главный элемент на i шаге.

Задача нахождения матрицы, обратной матрице *A*, эквивалентна задаче решения
матричного уравнения

$$AX = E$$

Это уравнение можно свести к системе из n линейных уравнений следующего вида:

$$AX_j = E_j,$$ где

X_j – j столбец матрицы X, а E_j – j столбец матрицы E.

Листинг программы

```
package main.java.com.bsu;

import javafx.util.Pair;

public class Main {

    public static void main(String[] args) {
        try {
            Double[][] A = {
                    {0.6444, 0.0000, -0.1683, 0.1184, 0.1973},
                    {-0.0395, 0.4208, 0.0000, -0.0802, 0.0263},
                    {0.0132, -0.1184, 0.7627, 0.0145, 0.0460},
                    {0.0395, 0.0000, -0.0960, 0.7627, 0.0000},
                    {0.0263, -0.0395, 0.1907, -0.0158, 0.5523}};
            Double[][] b = {
                    {1.2677},
                    {1.6819},
                    {-2.3657},
                    {-6.5369},
                    {2.8351}};
            System.out.println("Исходная система уравнений: ");
            for (int i = 0; i < A.length; ++i) {
                for (int j = 0; j < A[0].length; ++j) {
                    System.out.printf("%.4f", A[i][j]);
                    System.out.printf("\t");
                }
                System.out.print("| ");
                System.out.println(b[i][0]);
            }
            Pair<Double[][], Double> res = solveSystem(A, b);
            Double[][] inv = inverseMatrix(A);
            Double[][] x = res.getKey();
            Double det = res.getValue();
            Double[][] nevyazka = (subMatrix(multiplyMatrix(A, x), b));
            System.out.println("det(A) = " + det);
            System.out.println("Результаты:");
            for(int i = 0; i<x.length;++i){
                System.out.println("x"+(i+1)+" = "+x[i][0]);
            }
            System.out.println("A^(-1)");
            printMatrix(inv);
            System.out.println("A * A^(-1): ");
            printMatrix(multiplyMatrix(A,inv));
            System.out.println("Невязка: ");
            System.out.print("[");
            for(int i = 0; i<x.length;++i){
                System.out.print(nevyazka[i][0]);
                if(i!= x.length - 1){
                    System.out.print(", ");
                }
            }
            System.out.println("]");
        } catch (Exception exception) {
            System.out.println(exception);
        }
    }

    //Действия над матрицами
    public static Double[][] multiplyMatrix(Double[][] A, Double[][] B) throws IllegalArgumentException {
        if (A[0].length != B.length) {
```

```
        throw new IllegalArgumentException("Matrix size mismatch");
    }
    Double[][] result = new Double[A.length][B[0].length];
    for (int i = 0; i < A.length; ++i) {
        for (int j = 0; j < B[0].length; ++j) {
            result[i][j] = 0.0;
            for (int k = 0; k < A[0].length; ++k) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return result;
}

public static Double[][] subMatrix(Double[][] A, Double[][] B) throws IllegalArgumentException {
    if (A.length != B.length || A[0].length != B[0].length) {
        throw new IllegalArgumentException("Matrix size mismatch");
    }
    Double[][] result = new Double[A.length][A[0].length];
    for (int i = 0; i < result.length; ++i) {
        for (int j = 0; j < result[0].length; ++j) {
            result[i][j] = A[i][j] - B[i][j];
        }
    }
    return result;
}

public static void printMatrix(Double[][] A) {
    for (Double[] doubles : A) {
        for (Double aDouble : doubles) {
            System.out.printf("%.4E", aDouble);
            System.out.printf("\t");
        }
        System.out.println();
    }
}


//поиск обратной матрицы
public static Double[][] inverseMatrix(Double[][] A) throws IllegalArgumentException {
    if (A.length != A[0].length) {
        throw new IllegalArgumentException("Matrix should be square");
    }
    Double[][] inverse = new Double[A.length][A.length];
    Double[][] systemMatrix = new Double[A.length][2 * A.length];
    //заполнение системы
    for (int i = 0; i < A.length; ++i) {
        for (int j = 0; j < A.length; ++j) {
            systemMatrix[i][j] = A[i][j];
        }
    }
    for (int i = 0; i < A.length; ++i) {
        for (int j = 0; j < A.length; ++j) {
            if (i == j) {
                systemMatrix[i][A.length + j] = 1.0;
            } else {
                systemMatrix[i][A.length + j] = 0.0;
            }
        }
    }
    //прямой ход
    for (int i = 0; i < systemMatrix.length; ++i) {
```

```
            double mainElement = systemMatrix[i][i];
            //деление строки на ведущий элемент
            for (int j = i; j < 2 * systemMatrix.length; ++j) {
                systemMatrix[i][j] /= mainElement;
            }
            for (int k = i + 1; k < systemMatrix.length; ++k) {
                for (int j = i + 1; j < 2 * systemMatrix.length; ++j) {
                    systemMatrix[k][j] -= systemMatrix[i][j] * systemMatrix[k][i];
                }
                systemMatrix[k][i] = 0.0;
            }
        }
        //обратный ход
        for (int i = systemMatrix.length - 1; i >= 0; --i) {
            for (int k = i - 1; k >= 0; --k) {
                double factor = systemMatrix[k][i];
                for (int j = 2 * systemMatrix.length - 1; j >= i; --j) {
                    systemMatrix[k][j] -= systemMatrix[i][j] * factor;
                }
            }
        }
        //заполнение обратной матрицы
        for (int i = 0; i < A.length; ++i) {
            for (int j = 0; j < A.length; ++j) {
                inverse[i][j] = systemMatrix[i][A.length + j];
            }
        }
        return inverse;
    }

    //схема с выбором наибольшего по столбцу
    public static int findMaxColumnElementIndex(Double[][] A, int col, int beginRow) throws
ArrayIndexOutOfBoundsException {
        if (col > A[0].length || beginRow > A.length) {
            throw new ArrayIndexOutOfBoundsException();
        }
        int maxElementIndex = beginRow;
        double maxElement = Math.abs(A[0][beginRow]);
        for (int i = beginRow + 1; i < A.length; ++i) {
            if (Math.abs(A[i][col]) > maxElementIndex) {
                maxElementIndex = i;
                maxElement = Math.abs(A[i][col]);
            }
        }
        return maxElementIndex;
    }

    public static void swapRows(Double[][] A, int row1, int row2) throws ArrayIndexOutOfBoundsException {
        if (row1 > A.length || row2 > A.length) {
            throw new ArrayIndexOutOfBoundsException();
        }
        for (int j = 0; j < A[0].length; ++j) {
            Double tmp = A[row1][j];
            A[row1][j] = A[row2][j];
            A[row2][j] = tmp;
        }
    }

    public static Pair<Double[][], Double> solveSystem(Double[][] A, Double[][] b) throws
IllegalArgumentException {
        if (A.length != b.length || A.length != A[0].length || b[0].length != 1) {
            throw new IllegalArgumentException("Matrix size mismatch");
```

```
        }
        Double[][] x = new Double[A.length][1];
        Double[][] systemMatrix = new Double[A.length][A.length + 1];
        Double determinant = 1.0;
        //заполняем матрицу системы с которой будем работать
        for (int i = 0; i < A.length; ++i) {
            for (int j = 0; j < A.length + 1; ++j) {
                if (j != A.length) {
                    systemMatrix[i][j] = A[i][j];
                } else {
                    systemMatrix[i][j] = b[i][0];
                }
            }
        }
        //прямой ход
        for (int i = 0; i < systemMatrix.length; ++i) {
            int row = findMaxColumnElementIndex(systemMatrix, i, i);
            if (i != row) {
                determinant *= -1;
                swapRows(systemMatrix, i, row);
            }
            double mainElement = systemMatrix[i][i];
            determinant *= mainElement;
            //деление строки на ведущий элемент
            for (int j = i; j < systemMatrix.length + 1; ++j) {
                systemMatrix[i][j] /= mainElement;
            }
            for (int k = i + 1; k < systemMatrix.length; ++k) {
                for (int j = i + 1; j < systemMatrix.length + 1; ++j) {
                    systemMatrix[k][j] -= systemMatrix[i][j] * systemMatrix[k][i];
                }
                systemMatrix[k][i] = 0.0;
            }
        }
        //обратный ход
        for (int i = systemMatrix.length - 1; i >= 0; --i) {
            x[i][0] = systemMatrix[i][systemMatrix.length];
            for (int j = systemMatrix.length - 1; j > i; --j) {
                x[i][0] -= systemMatrix[i][j] * x[j][0];
            }
        }
        return new Pair<>(x, determinant);
    }
}
```

Входные данные

A					b
0,6444	0,0000	-0,1683	0,1184	0,1973	1.2677
-0,0395	0,4208	0,0000	-0,0802	0,0263	1.6819
0,0132	-0,1184	0,7627	0,0145	0,0460	-2.3657
0,0395	0,0000	-0,0960	0,7627	0,0000	-6.5369
0,0263	-0,0395	0,1907	-0,0158	0,5523	2.8351

Вывод программы

Исходная система уравнений:

0,6444	0,0000	-0,1683	0,1184	0,1973	| 1.2677
-0,0395	0,4208	0,0000	-0,0802	0,0263	| 1.6819
0,0132	-0,1184	0,7627	0,0145	0,0460	| -2.3657
0,0395	0,0000	-0,0960	0,7627	0,0000	| -6.5369
0,0263	-0,0395	0,1907	-0,0158	0,5523	| 2.8351

det(A) = 0.08339424331325142

Результаты:

x1 = 0.9982150476244698

x2 = 1.999865282123798

x3 = -2.99975970834894

x4 = -9.000008425832783

x5 = 6.007053532763626

Невязка:

[-3.3306690738754696E-15, 2.220446049250313E-16, 0.0, 8.881784197001252E-16, -4.440892098500626E-16]

A^(-1)

1,5876E+00	7,5017E-02	4,7008E-01	-2,6014E-01	-6,0988E-01
1,3760E-01	2,3889E+00	1,0145E-01	2,2436E-01	-1,7136E-01
-4,5417E-04	3,6621E-01	1,3447E+00	1,0337E-02	-1,2927E-01
-8,2280E-02	4,2209E-02	1,4490E-01	1,3259E+00	1,5315E-02
-6,7958E-02	4,2040E-02	-4,7527E-01	6,2796E-02	1,8725E+00

A * A^(-1):

1,0000E+00	0,0000E+00	0,0000E+00	-8,6736E-18	0,0000E+00
-8,0231E-18	1,0000E+00	1,7347E-18	-5,8547E-18	-6,9389E-18
-8,6736E-19	-4,6187E-17	1,0000E+00	-3,4694E-18	-2,7756E-17
0,0000E+00	0,0000E+00	2,7756E-17	1,0000E+00	3,4694E-18
6,9389E-18	-2,0817E-17	-5,5511E-17	0,0000E+00	1,0000E+00

Вывод

Метод Гаусса является наиболее мощным и универсальным спосоом нахождения решения любой системы линейных уравнений.

Преимущества метода:

– при решении систем линейных уравнений с числом уравнений и неизвестных более трёх метод Гаусса не такой громоздкий, поскольку при решении методом Гаусса необходимо меньше вычислений;

– методом Гаусса можно решать неопределённые системы линейных уравнений, то есть, имеющие общее решение;

– метод основан на элементарных методах - методе подстановки неизвестных и методе сложения уравнений.

Метод не приводит к большим ошибкам, о чём свидетельствуют результаты вычислений в лабораторной работе.