БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №1
Метод Гаусса для решения систем линейных уравнений

Преподаватель: Полевиков Виктор Кузьмич
Студент: Шиляев Иван
2 курс 9 группа

Постановка задачи

Сгенерировать случайным образом вещественные элементы квадратной матрицы A размера (9x9) с точностью до двух знаков после запятой. Задать вектор X и вычислить f = AX. Далее:

1. Найти решение СЛАУ, используя метод Гаусса;
2. Найти обратную матрицу для A, используя метод Гаусса;
3. Найти определитель матрицы A, используя метод Гаусса;
4. Найти меру обусловленности матрицы A.

Сгенерированная матрица:

-57,36	-87,81	88,46	29,80	-48,86	84,75	21,22	41,06	-11,62
-63,18	-28,19	-22,52	-69,48	43,95	90,18	32,05	-28,23	50,80
0,55	75,41	-82,34	-84,59	-24,43	77,08	91,98	91,67	-56,10
-10,40	79,29	81,23	-21,24	-16,45	90,60	-16,63	43,77	69,23
28,59	23,56	-12,76	50,99	-43,19	-7,99	-58,22	64,48	8,15
-7,11	-13,96	39,11	90,26	71,73	-60,29	72,99	-90,56	98,73
-80,46	-28,64	30,17	2,42	-17,07	6,02	-61,28	41,63	-10,48
-14,21	82,65	80,51	-35,07	84,48	22,85	-30,69	-61,91	76,48
79,25	-14,92	26,34	-50,32	73,19	4,41	93,13	21,83	96,31

Краткая теория

Метод Гаусса – метод решения системы линейных алгебраических уравнений (СЛАУ). Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные системы.

Решение системы методом Гаусса

Матрицу А с помощью эквивалентных преобразований приводим к треугольному виду. Элементы исходной системы находятся по следующим формулам — формулам прямого хода метода Гаусса:

$$\begin{cases} a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{kj}^{(k-1)} a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad k = \overline{1,n-1} \\ f_i^{(k)} = f_i^{(k-1)} - \frac{f_k^{(k-1)} a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = \overline{k+1,n}, \quad j = \overline{k+1,n} \\ a_{ij}^{(0)} = a_{ij} \\ f_i^{(0)} = f_i \end{cases}$$

Для нахождения решения системы выполняется обратный ход метода Гаусса:

$$\begin{cases} x_n = \frac{f_n^{(n-1)}}{a_{nn}^{(n-1)}} \\ \dots \\ x_i = \frac{f_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)} x_j}{a_{ii}^{(i-1)}} \qquad i = \overline{n-1,1} \end{cases}$$

Нахождения определителя матрицы, используя метод Гаусса

Определитель матрицы после приведения ее к диагональному виду по методу Гаусса равен:

$$detA = \prod_{i=1}^{n} a_{ii}^{(i-1)}$$

где $a_{ii}^{(i-1)}$ – диагональные элементы на *(i – 1)*-м шаге преобразования.

Нахождения обратной матрицы, используя метод Гаусса

Обратная матрица находится при решении *n* систем из *n* неизвестных. Данные системы выглядят так:

$$\sum_{j=1}^{n} a_{ij} * x_{jk} = \delta_{ik}, \quad i = \overline{1,n}, \quad где\ \delta_{ik} = \begin{cases} 1, i = k \\ 0, i \neq k \end{cases}$$

Решение каждой из систем есть один из столбцов обратной матрицы. Проверить правильность нахождения обратной матрицы можно по формуле:

$A * A^{-1} = E.$

Мера обусловленности матрицы

Мера обусловленности исходной матрицы $\nu(A)$ находится в результате умножения нормы матрицы А на норму обратной матрицы A^{-1}:

$$\nu(A) = \left\|A\right\| \bullet \left\|A^{-1}\right\|$$

Для нахождения нормы используется формула кубической нормы матрицы (т.е. норма матрицы, подчиненная кубической норме векторов). Она равна максимальной сумме модулей элементов в каждой строке:

$$\left\|A\right\| = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$$

Листинг программы
(на языке программирования Java)

```
import java.text.DecimalFormat;

public class Solution {
    public static final int SIZE = 9;
    public static double[][] A = new double[SIZE][SIZE];
    public static double[][] A_INVERSE = new double[SIZE][SIZE];
    public static double[] x = new double[SIZE];
    public static double[] myX = new double[SIZE];
    public static double[] f = new double[SIZE];
    public static boolean isDetZero = false;

    public static double round(double value, int places) {
        if (places < 0) throw new IllegalArgumentException();

        long factor = (long) Math.pow(10, places);
        value = value * factor;
        long tmp = Math.round(value);
        return (double) tmp / factor;
    }

    public static void printMatrix(double[][] A) {
        for (int i=0; i<SIZE; ++i) {
            for (int j=0; j<SIZE; ++j) {
                System.out.printf("%10.2f ", A[i][j]);
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void printMatrix2(double[][] A) {
        DecimalFormat format = new DecimalFormat("#.##E00");
        for (int i=0; i<SIZE; ++i) {
            for (int j=0; j<SIZE; ++j) {
                System.out.printf("%10s ", format.format(A[i][j]));
            }
            System.out.println();
        }
        System.out.println();
    }

    public static void generateMatrix(double[][] A) {
        for (int i=0; i<SIZE; ++i) {
            for (int j=0; j<SIZE; ++j) {
                A[i][j] = round(-99 + Math.random()*199, 2);
            }
        }
    }

    public static void generateX(double[] x) {
        for (int i=0; i<SIZE; ++i) {
            x[i] = round(-99 + Math.random()*199, 2);
        }
    }

    public static void countF(double[][] A, double[] x) {
        for (int i=0; i<SIZE; ++i) {
            f[i]=0;
        }
        for (int i=0; i<SIZE; ++i) {
```

```
            for (int j=0; j<SIZE; ++j) {
                f[i]+=A[i][j]*x[j];
            }
        }
    }

    public static void solve(double[][] myA, double[] myX, double[] f) {
        double[][] A = new double[SIZE][SIZE];
        for (int i=0; i<SIZE; ++i) {
            System.arraycopy(myA[i], 0, A[i], 0, SIZE);
        }

        for (int k=1; k<SIZE; ++k) {
            for (int i=k; i<SIZE; ++i) {
                f[i] -= f[k-1]*A[i][k-1]/A[k-1][k-1];
                double tmp = A[i][k-1];
                for (int j = 0; j < SIZE; ++j) {
                    A[i][j] -= A[k-1][j] * tmp / A[k-1][k-1];
                }
            }
        }

        myX[SIZE-1] = f[SIZE-1]/A[SIZE-1][SIZE-1];
        for (int i=SIZE-2; i>=0; --i) {
            double sum = 0;
            for (int j=i+1; j<SIZE; ++j)
                sum += A[i][j]*myX[j];
            myX[i] = (f[i]-sum)/A[i][i];
        }
    }

    public static double det(double[][] myA) {
        double det = 1;
        double[][] A = new double[SIZE][SIZE];
        for (int i=0; i<SIZE; ++i) {
            System.arraycopy(myA[i], 0, A[i], 0, SIZE);
        }
        for (int k=1; k<SIZE; ++k) {
            for (int i=k; i<SIZE; ++i) {
                double tmp = A[i][k-1];
                for (int j = 0; j < SIZE; ++j) {
                    A[i][j] -= A[k-1][j] * tmp / A[k-1][k-1];
                }
            }
        }
        for (int i=0; i<SIZE; ++i) {
            det *= A[i][i];
        }
        return det;
    }

    public static void countInverseMatrix(double[][] A, double[][] A_INVERSE) {
        if (det(A) == 0) {
            isDetZero = true;
            return;
        }
        double[] myf = new double[SIZE];
        double[] myX = new double[SIZE];
        for (int l=0; l<SIZE; ++l) {
            for (int i=0; i<SIZE; ++i)
                myf[i] = 0;
            myf[l] = 1;
            solve(A, myX, myf);
            for (int i=0; i<SIZE; ++i) {
                A_INVERSE[i][l] = myX[i];
```

```
            }
        }
    }

    public static double[][] multiply(double[][] A, double[][] B) {
        double[][] C = new double[SIZE][SIZE];
        for (int i=0; i<SIZE; ++i) {
            for (int j=0; j<SIZE; ++j) {
                for (int k=0; k<SIZE; ++k) {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }
        return C;
    }

    public static double norm_1(double[][] A) {
        double max = 0;
        for (int j=0; j<SIZE; ++j) {
            double sum = 0;
            for (int i=0; i<SIZE; ++i) {
                sum += Math.abs(A[i][j]);
            }
            if (sum>max)
                max = sum;
        }
        return max;
    }

    public static double norm_2(double[][] A) {
        double max = 0;
        for (int i=0; i<SIZE; ++i) {
            double sum = 0;
            for (int j=0; j<SIZE; ++j) {
                sum += Math.abs(A[i][j]);
            }
            if (sum>max)
                max = sum;
        }
        return max;
    }

    public static void main(String[] args) {
        generateMatrix(A);
        System.out.println("Original matrix:");
        printMatrix(A);
        generateX(x);
        countF(A, x);

        // step 1
        solve(A, myX, f);

        System.out.println("Original solution:");
        for (int i = 0; i < SIZE; ++i) {
            System.out.printf("%10.2f ", x[i]);
        }
        System.out.println();

        System.out.println("My solution:");
        for (int i = 0; i < SIZE; ++i) {
            System.out.printf("%10.2f ", myX[i]);
        }
        System.out.println();

        // step 2
```

```
        countInverseMatrix(A, A_INVERSE);
        System.out.println("Det(A) = " + det(A) + "\n");
        if (isDetZero) {
            System.out.println("Inverse matrix doesn't exist, because det(A) =
0");
        }
        else {
            System.out.println("Inverse matrix:");
            printMatrix2(A_INVERSE);
            System.out.println("Check");
            System.out.println("Unit matrix E=A*A^(-1):");
            double[][] E = multiply(A, A_INVERSE);
            printMatrix2(E);
        }

        // step 3
        double n1 = norm_1(A);
        double n2 = norm_2(A_INVERSE);
        double V = n1*n2;
        System.out.println("V(A) = ||A|| * ||A^(-1)|| = " + V);
    }
}
```

Исходные данные:

Сгенерированная матрица:

-57,36	-87,81	88,46	29,80	-48,86	84,75	21,22	41,06	-11,62
-63,18	-28,19	-22,52	-69,48	43,95	90,18	32,05	-28,23	50,80
0,55	75,41	-82,34	-84,59	-24,43	77,08	91,98	91,67	-56,10
-10,40	79,29	81,23	-21,24	-16,45	90,60	-16,63	43,77	69,23
28,59	23,56	-12,76	50,99	-43,19	-7,99	-58,22	64,48	8,15
-7,11	-13,96	39,11	90,26	71,73	-60,29	72,99	-90,56	98,73
-80,46	-28,64	30,17	2,42	-17,07	6,02	-61,28	41,63	-10,48
-14,21	82,65	80,51	-35,07	84,48	22,85	-30,69	-61,91	76,48
79,25	-14,92	26,34	-50,32	73,19	4,41	93,13	21,83	96,31

Определитель: 2.9498064905486735E18

Решение:

58,03	-31,00	-6,88	91,73	21,03	38,22	44,60	54,48	-66,08

Мера обусловленности: $||A|| * ||A^{(-1)}|| = 601.3 * 0.07363501522454746 =$
$= 42.280404785122506;$

Матрица, полученная от перемножения исходной и обратной матриц:

1E00	-6,94E-18	1,39E-17	-5,55E-17	-1,67E-16	-4,86E-17	2,78E-17	-5,55E-17	6,94E-17
5,55E-17	1E00	-5,55E-17	0E00	-8,33E-17	0E00	-5,55E-17	0E00	0E00
5E-16	-8,33E-17	1E00	0E00	3,33E-16	8,33E-17	-2,5E-16	3,33E-16	-1,25E-16
2,22E-15	-7,22E-16	2,22E-15	1E00	2,83E-15	-2,22E-16	-1,22E-15	2,22E-15	3,05E-16
2,55E-15	3,33E-15	-3,47E-15	9,85E-16	1E00	5,9E-17	-1,75E-15	8,04E-15	-1,37E-15
-4,44E-16	3,83E-15	-3,89E-15	1,31E-14	4,61E-15	1E00	4,33E-15	4,88E-15	1,86E-15
4,7E-15	1,99E-15	3,05E-16	-8,6E-16	7,78E-15	7,74E-16	1E00	1,52E-14	8,71E-16
-1,22E-15	-2,55E-15	9,44E-16	-1,11E-16	1,33E-15	-3,89E-16	-7,22E-16	1E00	1,11E-16
-3,22E-15	-1,11E-14	2,92E-14	-2,46E-14	8,72E-15	6E-15	1,15E-14	-1,51E-14	1E00