

«Київський політехнічний інститут»  
Факультет інформатики і обчислювальної техніки  
Кафедра обчислювальної техніки

Лабораторна робота №5  
З алгоритмів та методів обчислень  
Варіант 3

*Виконав:*  
Студент групи ІО-32  
Попенко Р. Л.  
*Перевірів:*  
Порєв В. М.

Київ - 2015 р.

**1. Тема завдання:**

Розв'язання систем лінійних алгебраїчних рівнянь.

Мета: Вивчити алгоритми методів розв'язання систем лінійних алгебраїчних рівнянь на ЕОМ.

**2. Завдання:**

Скласти програму розв'язання СЛАР із вказаним методом розв'язання.

Метод	Номер варіанту
Гауса з одиничною діагоналлю	3

Номер варіанту	Матриця коефіцієнтів системи	Стовпець вільних членів	Примітка
3	1 -3 2	5	$X_1=5$
	3 -4 0	7	$X_2=2$
	2 -5 3	9	$X_3=3$

**3. Лістинг програми:**

```
package lab5;
public class Matrix {
    private double[][] matrix;
    private int degree;

    public Matrix(double[][] matrix) {
        this.matrix = matrix;
        degree = matrix.length;
    }

    public double getElement(int row, int column) {
        return matrix[row][column];
    }
    @Override
    public String toString() {
        String result = "";
        for(int i = 0; i < degree; i++) {
            result += "| ";
            for(int j = 0; j < degree; j++)
                result += matrix[i][j] + "\t ";
            result += "|\n";
        }
        return result;
    }

    public void swapElements(int r1, int c1, int r2, int c2) {
        double a = matrix[r1][c1];
        matrix[r1][c1] = matrix[r2][c2];
        matrix[r2][c2] = a;
    }

    public void swapRows(int r1, int r2) {
        if(r1 == r2)
            return;
        for(int i = 0; i < degree; i++)
            swapElements(r1, i, r2, i);
    }

    public void addRows(int r1, int r2, double k) {
        for(int i = 0; i < degree; i++)
            matrix[r1][i] += matrix[r2][i] * k;
    }

    public void mulRow(int r, double m) {
        for(int i = 0; i < degree; i++)
            matrix[r][i] *= m;
    }
}
```

```

package lab5;
import java.util.Arrays;
public class Vector {
    private double[] vector;
    private int length;

    public Vector(double[] vector) {
        this.vector = vector;
        length = vector.length;
    }
    public double getElement(int index) {
        return vector[index];
    }
    public void setElement(int index, double val) {
        vector[index] = val;
    }
    public Vector setAll(Double val) {
        Arrays.fill(vector, val);
        return this;
    }
    public int getLength() {
        return length;
    }
    public void swap(int v1, int v2) {
        double c = vector[v1];
        vector[v1] = vector[v2];
        vector[v2] = c;
    }
    public void add(int v1, int v2, double k) {
        vector[v1] += vector[v2] * k;
    }
    public void multiply(int v, double m) {
        vector[v] *= m;
    }
    @Override
    public String toString() {
        return Arrays.toString(vector);
    }
    public Vector copyOf() {
        return new Vector(Arrays.copyOf(vector, length));
    }
}

```

```

package lab5;

public class GaussJordano {
    public static Vector solve(Matrix matrix, Vector free, int dac) {
        int n = free.getLength();
        Vector result = free.copyOf();
        boolean hasResult = false;
        double koef = 1.0;

        for(int i = 0; i < n - 1; i++) {
            if(matrix.getElement(i, i) == 0) {
                for(int j = i + 1; j < n; j++)
                    if(matrix.getElement(i, j) != 0) {
                        matrix.swapRows(i, j);
                        result.swap(i, j);
                        hasResult = true;
                    }
            }

            if(!hasResult)
                return result.setAll(Double.NaN);
        }
    }
}

```

```

        koef = 1.0/matrix.getElement(i, i);
        matrix.mulRow(i, koef);
        result.multiply(i, koef);
        for(int j = i + 1; j < n; j++) {
            koef = -1.0 * matrix.getElement(j, i);
            matrix.addRow(j, i, koef);
            result.add(j, i, koef);
        }
    }

    koef = 1.0/matrix.getElement(n - 1, n - 1);
    matrix.mulRow(n - 1, koef);
    result.multiply(n - 1, koef);

    for(int i = n - 1; i >= 1; i--) {
        for(int j = i - 1; j >= 0; j--) {
            koef = -1.0 * matrix.getElement(j, i);
            matrix.addRow(j, i, koef);
            result.add(j, i, koef);
        }
    }

    double pow = Math.pow(10, dac);
    for(int i = 0; i < n; i++) {
        result.setElement(i, Math.round(result.getElement(i) * pow)/pow);
    }

    return result;
}
}

```

#### Запуск програми

#### 4. Аналіз результатів:

Створена мною програма знаходить корені СЛАР методом Гауса з одиничною діагоналлю (методом Гауса-Жордана).