Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики і обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №5
З алгоритмів та методів обчислень

Виконав:
Студент групи ІО-22
Бас А. В.

м. Київ
2014 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 за методом Гауса.

3. Лістинг програми:

```
public class GausSolver {

    private static final double EPS = 0.0000000000001D;
    double[][] a;
    int[] bIndexes;

    public GausSolver(double[][] data) {
        this.a = cloneMatrix(data);

        bIndexes = new int[a.length];
        for (int i = 0; i < a.length; i++) {
            bIndexes[i] = i;
        }


    }

    public double[] solve() {
        goDown();
        goUp();

        int n = a.length;
        double[] res = new double[n];
        for (int i = 0; i < n; i++) {
            res[i] = a[bIndexes[i]][n];
        }

        return res;
    }

    private void goDown() {

        for (int i = 0; i < a.length; i++) {
            if (Math.abs(a[i][i]) < EPS) {
                int rowIndex = indexOfRowWithMaxValueInColumn(i, i);
                swapRows(i, rowIndex);
            }

            for (int j = i + 1; j < a.length; j++) {
                addToRowAnotherMultipliedBy(j, i, -a[j][i] / a[i][i]);
            }

            multiplyRow(i, 1.0 / a[i][i]);
        }
    }


    private void goUp() {

        for (int i = a.length - 1; i > 0; i--) {
            for (int j = i - 1; j >= 0; j--) {
                addToRowAnotherMultipliedBy(j, i, -a[j][i]);
            }
        }
```

```
    }

    private void addToRowAnotherMultipliedBy(int destRow, int srcRow, double value) {

        int n = a[srcRow].length;

        for (int i = 0; i < n; i++) {
            a[destRow][i] += a[srcRow][i] * value;
        }
    }

    private void multiplyRow(int row, double value) {
        for (int i = 0; i < a[row].length; i++) {
            a[row][i] *= value;
        }
    }

    private int indexOfRowWithMaxValueInColumn(int startRow, int col) {
        double max = a[startRow][col];
        int index = startRow;
        for (int i = startRow + 1; i < a.length; i++) {
            if (a[i][col] > max) {
                max = a[i][col];
                index = i;
            }
        }
        return index;
    }

    private void swapRows(int i, int j) {

        double[] temp = a[i];
        a[i] = a[j];
        a[j] = temp;

        int t = bIndexes[i];
        bIndexes[i] = bIndexes[j];
        bIndexes[j] = t;
    }

    private double[][] cloneMatrix(double[][] x) {
        double[][] b = new double[x.length][x[0].length];
        for (int i = 0; i < x.length; i++) {
            System.arraycopy(x[i], 0, b[i], 0, x[i].length);
        }
        return b;
    }

}


public class GausFrame extends ApplicationFrame {
    private JPanel rootPanel;
    private JComboBox mComboBox1;
    private JButton mSolveButton;

    private JScrollPane mScrollPane;
    private JTable mTable;
    private AbstractTableModel mTableModel;

    private Object[][] data;
    private ArrayList<Object[][]> cache = new ArrayList<>();
```

```
public GausFrame(final String title) {
    super(title);

    setContentPane(rootPanel);

    setPreferredSize(new Dimension(500, 400));

    setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

    mSolveButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {

            int n = mComboBox1.getSelectedIndex() + 2;

            double[][] a = new double[n][n + 1];
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    a[i][j] = Double.parseDouble(mTableModel.getValueAt(i, j).toString());
                }
                a[i][n] = Double.parseDouble(mTableModel.getValueAt(i, n + 1).toString());
            }


            GausSolver solver = new GausSolver(a);


            double[] res = solver.solve();
            for (int i = 0; i < res.length; i++) {
                mTableModel.setValueAt(res[i], i, n + 3);
            }

            mTableModel.fireTableDataChanged();


        }
    });

    for (int i = 0; i < 6; i++) {
        cache.add(new Object[i][i + 4]);
    }

    mComboBox1.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            drawGraphic(mComboBox1.getSelectedIndex() + 2);
        }
    });

    mComboBox1.setSelectedIndex(1);
    drawGraphic(3);


    pack();
    setVisible(true);
}

private void drawGraphic(int n) {

    if (mScrollPane != null) {
        rootPanel.remove(mScrollPane);
    }
```

```
        mTableModel = new MyTableModel(n);
        mTable = new JTable(mTableModel);
        mScrollPane = new JScrollPane(mTable);

        rootPanel.add(mScrollPane, BorderLayout.CENTER);

        pack();

    }

    class MyTableModel extends AbstractTableModel {

        ArrayList<String> columnName = new ArrayList<String>();
        Object[][] data;


        public MyTableModel(int n) {
            for (int i = 0; i < n; i++) {
                columnName.add("a_" + (i + 1));
            }
            columnName.add("    ");
            columnName.add(" b ");
            columnName.add("    ");
            columnName.add("x");

            data = cache.get(n);
        }

        @Override
        public int getRowCount() {
            return data.length;
        }

        @Override
        public int getColumnCount() {
            return columnName.size();
        }

        @Override
        public Object getValueAt(int rowIndex, int columnIndex) {
            return data[rowIndex][columnIndex];
        }

        @Override
        public String getColumnName(int column) {
            return columnName.get(column);
        }

        @Override
        public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
            data[rowIndex][columnIndex] = aValue;
        }

        @Override
        public boolean isCellEditable(int row, int col) {
            if (col < data.length || col == data.length + 1) {
                return true;
            } else {
                return false;
            }
        }
    }
```