Київський національний університет
імені Т.Шевченка

Звіт

до лабораторної роботи №1
на тему:
«Чисельні методи дослідження перехідних процесів у лінійних стаціонарних системах»

Студента третього курсу
Групи САТР-3
Факультету комп'ютерних наук
та кібернетики
Арзамасцева Владислава

Київ
2020

Мета

Метою даної лабораторної роботи є розробка програми, яка реалізує ітераційний алгоритм обчислення значень виходів дискретизованої моделі лінійної стаціонарної системи під дією кусково-постійного вхідного процесу.

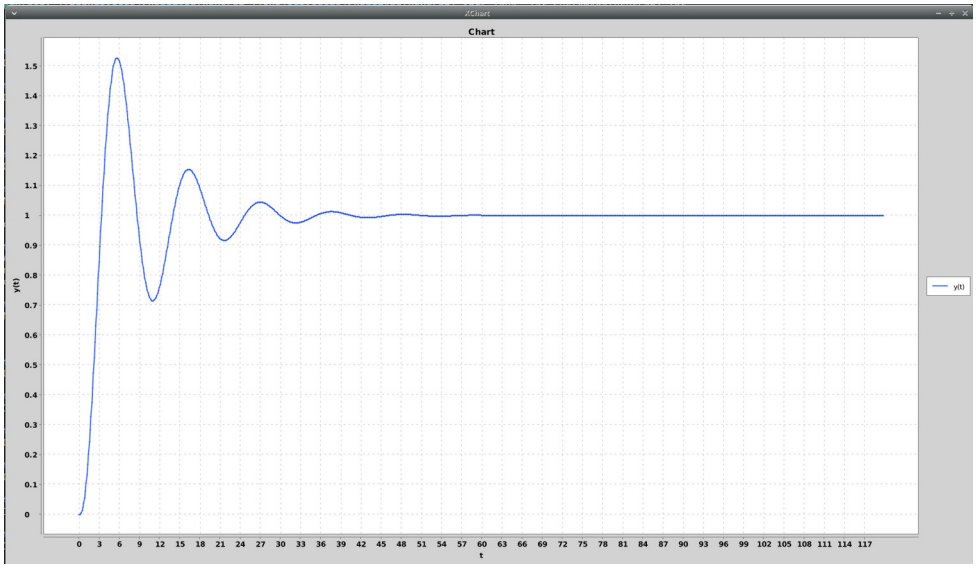
Принципи виконання роботи

Маємо лінійну стаціонарну систему диференціальних рівнянь $\dot{x}(t)=A*x(t)+B*u(t), t\in[0,inf)$. Для початку знаходимо на осі часу точки, утворені поділом заданого проміжку заданим користувачем періодом квантування T0. Вхідний процес u апроксимується кусково-постійним процесом. За допомогою рекурентної формули ($x_{k+1}=F*x_k+G*u_k$, $де F=e^{A*T0}$, $G=(F-I)*A^{(-1)}*B$, $x_0=0$) обчислюємо стани системи в точках k*T0, де k=0,1,…n. Матрицю F отримуємо розкладанням матричної експоненти в ряд Тейлора до заданої користувачем точності q. Матрицю G обчислюємо таким чином, щоб уникнути обернення матриці A. Використовуючи обчислені стани, можна отримати значення вихідного процесу $y_{k+1}=C*x_{k+1}$. Елементи матриці A $a_{32}=a1, a_{33}=a2$ задаються користувачем, мають належати проміжку від одиниці до десяти включно. Матриця B є вектором-стовпчиком розмірності 3 на 1 з ненульовою лише останню компонентою, що дорівнює одиниці. Матриця C = (1, 0, 0). Маємо поставлені три варіанти для спостереження за вихідними сигналами в залежності від вхідних сигналів u :

1. u = 1 для всього проміжку t;
2. u = 1 першу половину ітерацій, u = -1 - другу;
3. u = 1 першу третину ітерацій, u = -1 другу третину ітерацій, u = 1 останню третину ітерацій.

Лістинг програми

1. b = 1, a1 = 1, a2 = 3, q = 10, T0 = 0.02, варіант 1



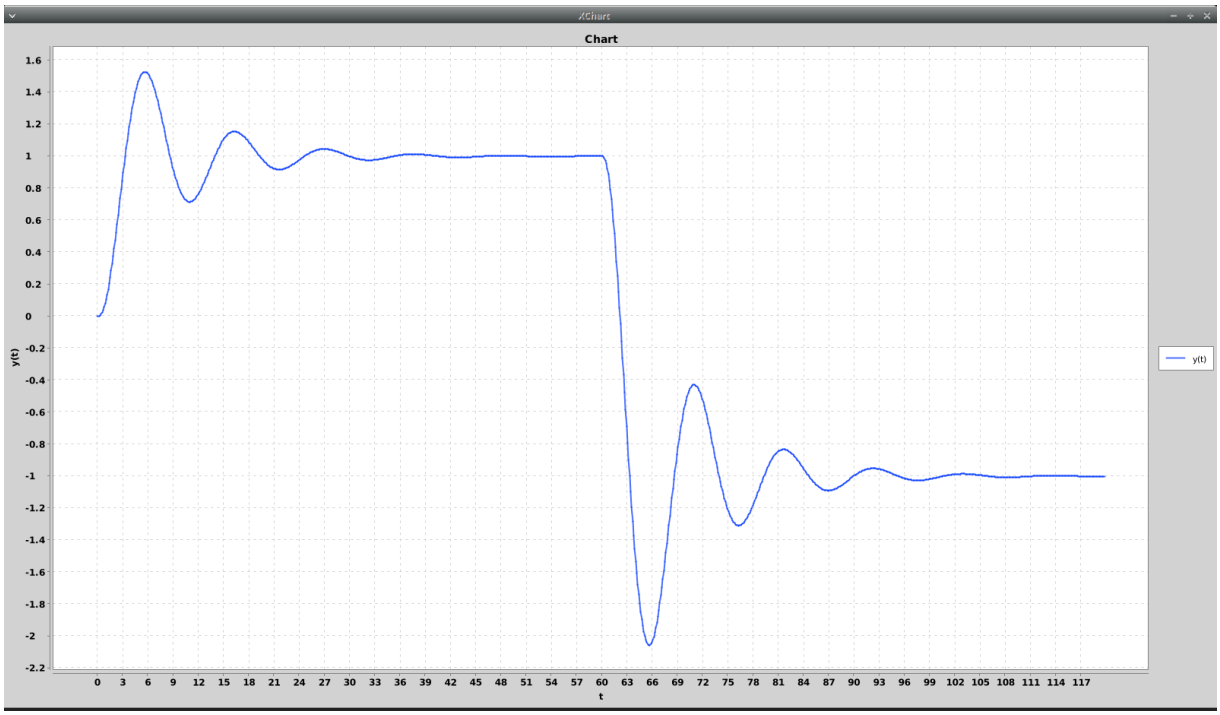
Компоненти вектора X:

/home/bartimaeues/Documents/SystemAnalysisLab_1/Variant_1.txt - Mousepad

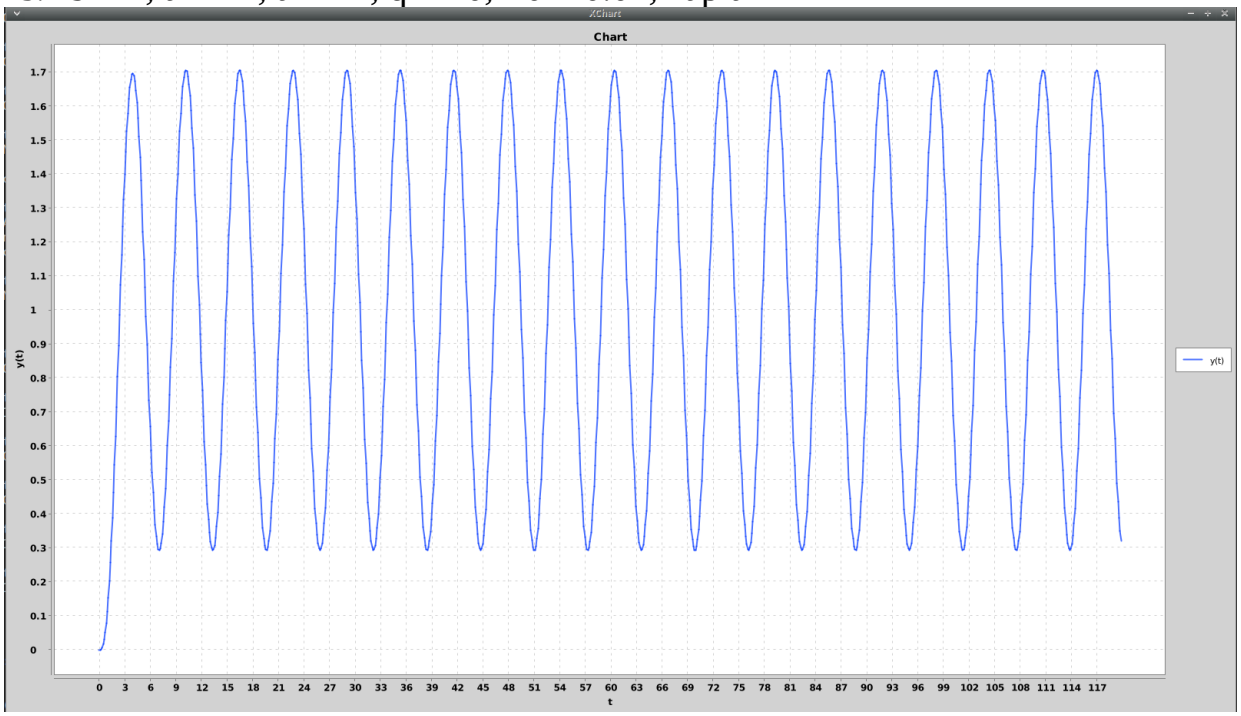
File Edit Search View Document Help

k	x_1	x_2	x_3
100	0.40685260770220694	0.4064339167297216	0.11900004940730632
200	1.240473069358401	0.331122918789906	-0.16323373830986623
300	1.5194346445082343	-0.05533612336365703	-0.17480300547178934
400	1.1629990012088856	-0.24224942325797255	-0.0029706474190478
500	0.771272221551095	-0.11183306144485615	0.10839504772426306
600	0.7587197510816096	0.08498100330878357	0.06752127394733974
700	0.9980584150337514	0.12196661001824635	-0.027437517146384052
800	1.1509249500593017	0.020296869746225008	-0.0591821074305228
900	1.092630462144591	-0.06459399319963036	-0.01854681712979144
1000	0.9609592041972848	-0.05191580266382786	0.026075124241033792
1100	0.9179618565183907	0.009277064671057789	0.02748393658222458
1200	0.9749252054614452	0.03834582677546035	2.0791013135527947E-4
1300	1.03652974171668	0.017375670862178998	-0.01719970238294686
1400	1.0379308702185246	-0.013649419539392964	-0.010547928635098996
1500	0.9999445180761369	-0.019220126738289753	0.004454264884646571
1600	0.976055125162435	-0.003036049187195425	0.009347007827111339
1700	0.9855329508411897	0.010277432363334418	0.002853013792389069
1800	1.0063327753144142	0.008138343351864731	-0.004164362709437073
1900	1.012955290470608	-0.0015498756703865174	-0.004320627130942074
2000	1.0038541459527137	-0.006069006963509065	8.422332328519855E-6
2100	0.9941673368018765	-0.0026986187606501434	0.0027287846296054873
2200	0.9940378753150013	0.0021914491662891828	0.00164735751228854
2300	1.0000659436194819	0.00302841840347733	-7.224917243827 47E-4
2400	1.0037984069438923	4.5266825114068955E-4	-0.0014760504750725434
2500	1.002258904932904	-0.0016349322285509795	-4.385061644643015E-4
2600	0.9989736070487782	-0.0012755417133476213	6.64910882275889E-4
2700	0.9979543802612352	2.5812450858005755E-4	6.791282915702829E-4
2800	0.9994081152521762	9.604208669428733E-4	-7.845031395323221E-6
2900	1.000931069593049	4.1895199526218493E-4	-4.3286752675066736E-4
3000	1.0009370136660452	-3.517055353895132E-4	-2.572173652355923E-4
3100	0.9999805572004181	-4.771111763802652E-4	1.170940510921599E-4

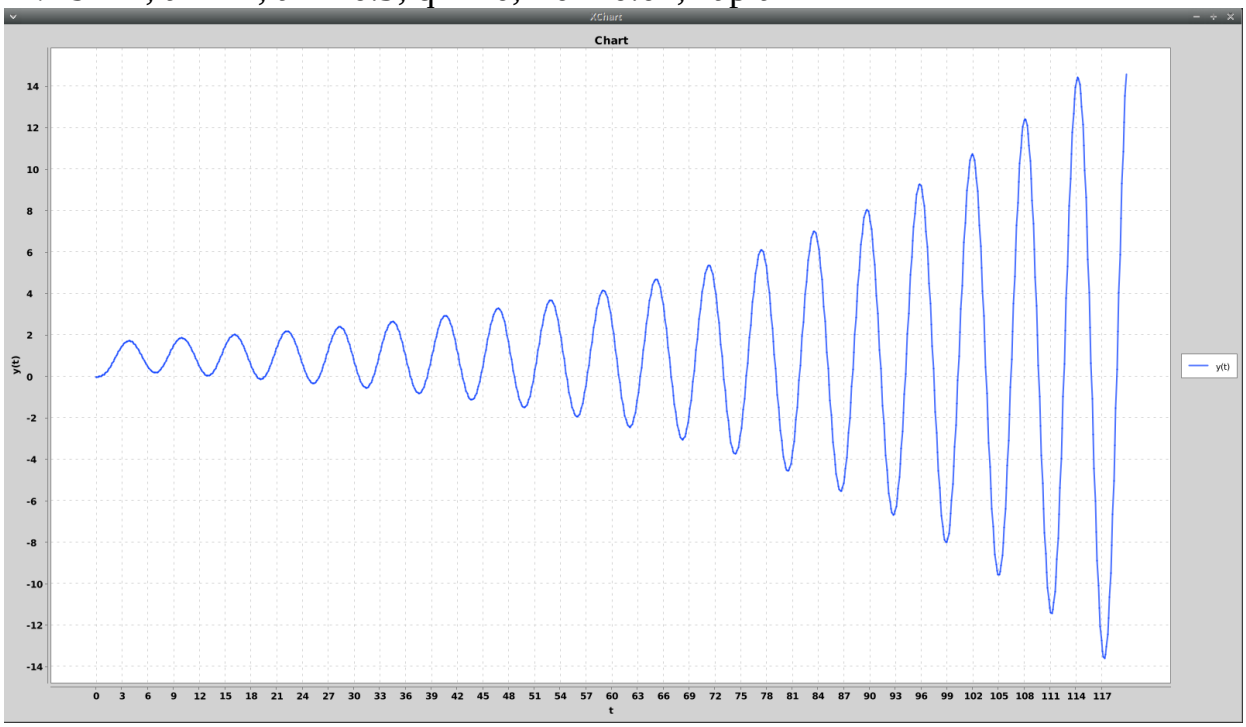
2. b = 1, a1 = 1, a2 = 3, q = 10, T0 = 0.02, варіант 2



3. b = 1, a1 = 1, a2 = 1, q = 10, T0 = 0.02, варіант 1



4. b = 1, a1 = 1, a2 = 0.9, q = 10, T0 = 0.02, варіант 1



Висновок

Було отримано наочні результати, які дозволяють спостерігати за поведінкою системи за заданими константними керуваннями, що мають можливість бути зміненими впродовж обчислення рекурентного алгоритму. Графіки набору і-тих елементів кожного вектору мають особливість в стуханні хвиль з пройденим часом. Бачимо, що процес збігається до значення b (при u = 1 і до -b при u = -1), якщо a2 > 1, процес проходить без зміни амплітуди при a2 = 1 і амплітуда збільшується при a2 < 1.

Додаток (Код програми)

```
package main;

import command.RunCommand;
import framework.application.Application;
import framework.state.ApplicationState;
import state.LaboratoryState;

public class Main {

   private static final String PATH_TO_PROPERTIES = "/laboratory-
framework.properties";

   public static void main(String[] args) {
      ApplicationState state = new LaboratoryState();
      Application application = new
Application.ApplicationBuilder(PATH_TO_PROPERTIES, state)
            .addCommand("run", new RunCommand())
            .build();
      application.start();
   }

}
```

```
package state;

import framework.state.AbstractApplicationState;
import framework.state.StateHelper;
import framework.utils.ConsoleUtils;
import framework.variable.holder.VariableHolder;
```

```
import framework.variable.holder.VariableHolderAware;
import lombok.Getter;
import org.apache.commons.math3.linear.Array2DRowRealMatrix;
import org.apache.commons.math3.linear.RealMatrix;

@Getter
public class LaboratoryState extends AbstractApplicationState implements
VariableHolderAware {

    //T must be T >= 0
    private double T = 0.02;

    //q must be >= 2 and <= 10
    private int q = 10;

    private double a1 = 1;

    private double a2 = 3;

    private final int k = 60;

    private double b = 1;

    private VariableHolder variableHolder;

    private final RealMatrix C;

    public LaboratoryState() {
        this.C = new Array2DRowRealMatrix(new double[][]{{1, 0, 0}});
    }

    @Override
    protected void initVariableNameToSettersMap() {
        this.variableNameToSetter.put("T", (name, value) -> StateHelper
                .defaultSet(name, "T", value, Double.class, (val) -> (Double) val,
this::setT));
        this.variableNameToSetter.put("b", (name, value) -> StateHelper
                .defaultSet(name, "b", value, Double.class, (val) -> (Double) val,
this::setB));
        this.variableNameToSetter.put("q", (name, value) -> StateHelper
                .defaultSet(name, "q", value, Integer.class, (val) -> (Integer) val,
this::setQ));
        this.variableNameToSetter.put("a1", (name, value) -> StateHelper
                .defaultSet(name, "a1", value, Double.class, (val) -> (Double) val,
this::setA1));
        this.variableNameToSetter.put("a2", (name, value) -> StateHelper
```

```
            .defaultSet(name, "a2", value, Double.class, (val) -> (Double) val,
this::setA2));
    }

    @Override
    protected void initVariableNameToGettersMap() {
        this.variableNameToGetter.put("B", () -> new Array2DRowRealMatrix(new
double[][]{{0}, {0}, {b}}));
        this.variableNameToGetter.put("C", this::getC);
        this.variableNameToGetter.put("T", this::getT);
        this.variableNameToGetter.put("q", this::getQ);
        this.variableNameToGetter.put("b", this::getB);
        this.variableNameToGetter.put("k", this::getK);
        this.variableNameToGetter.put("a1", this::getA1);
        this.variableNameToGetter.put("a2", this::getA2);
    }

    public void setT(double T) {
        if (T <= 0) {

ConsoleUtils.println(variableHolder.getVariable("T").getConstraintViolationMessage(
));
            return;
        }
        this.T = T;
    }

    public void setB(double b) {
        this.b = b;
    }

    public void setQ(int q) {
        if (q < 2 || q > 10) {

ConsoleUtils.println(variableHolder.getVariable("q").getConstraintViolationMessage(
));
        } else {
            this.q = q;
        }
    }

    public void setA1(double a1) {
        this.a1 = a1;
    }

    public void setA2(double a2) {
        this.a2 = a2;
    }
```

```
    @Override
    public void setVariableHolder(VariableHolder variableHolder) {
        this.variableHolder = variableHolder;
    }

}
```

```
package dto;

import lombok.Data;
import org.apache.commons.math3.linear.RealMatrix;

@Data
public class MatricesDto {

    private final RealMatrix F;

    private final RealMatrix G;

}
```

```
package dto;

import lombok.Data;
import java.util.List;

@Data
public class ChartDto {

    private final List<? extends Number> xData;

    private final List<? extends Number> yData;

}
```

```
package dao;

import org.apache.commons.math3.linear.RealVector;

public interface VectorXDao {

    void write(int iterationStep, RealVector x);
```

```
}
```

```
package dao;

import framework.exception.LaboratoryFrameworkException;
import org.apache.commons.io.FileUtils;
import org.apache.commons.math3.linear.RealVector;

import java.io.File;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;

public class FileSystemVectorXDao implements VectorXDao {

    private static final Path directoryPath;

    private final Path path;

    static {
        String userHome = System.getProperty("user.home");
        String directoryPathString = userHome + File.separator +
                "Documents" + File.separator +
                "SystemAnalysisLab_1";
        directoryPath = Paths.get(directoryPathString);
        try {
            FileUtils.forceMkdir(directoryPath.toFile());
        } catch (IOException e) {
            throw new ExceptionInInitializerError(e);
        }
    }

    public FileSystemVectorXDao(String fileName) {
        this.path = directoryPath.resolve(fileName);
        try {
            FileUtils.touch(path.toFile());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        clearFile(path);
    }

    @Override
    public void write(int iterationStep, RealVector x) {
        try {
```

```
            String s = (Files.size(path) == 0) ? getHeader(x.getDimension()) :
getRow(iterationStep, x);
            Files.write(path, s.getBytes(StandardCharsets.UTF_8),
StandardOpenOption.APPEND);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    private String getRow(int iterationStep, RealVector x){
        StringBuilder sb = new StringBuilder(100);
        sb.append(iterationStep).append("\t");
        for (int i = 0; i < x.getDimension(); i++){
            sb.append(x.getEntry(i)).append("\t");
        }
        sb.delete(sb.length() - 1, sb.length());
        sb.append(System.lineSeparator());
        return sb.toString();
    }

    private String getHeader(int length) {
        StringBuilder sb = new StringBuilder("k\t\t");
        for (int i = 0; i < length; i++){
            sb.append("x_").append(i + 1).append("\t\t\t");
        }
        sb.delete(sb.length() - 3, sb.length());
        sb.append(System.lineSeparator());
        return sb.toString();
    }

    private void clearFile(Path path) {
        try {
            Files.write(path, "".getBytes(StandardCharsets.UTF_8));
        } catch (IOException e) {
            throw new LaboratoryFrameworkException(e);
        }
    }

}

package command;

import chart.ChartHelper;
import dao.FileSystemVectorXDao;
import dao.VectorXDao;
import dto.MatricesDto;
import framework.command.RunnableCommand;
import framework.state.ApplicationState;
```

```
import framework.state.ApplicationStateAware;
import framework.utils.ConsoleUtils;
import framework.utils.MatrixUtils;
import framework.utils.ValidationUtils;
import org.apache.commons.math3.linear.Array2DRowRealMatrix;
import org.apache.commons.math3.linear.ArrayRealVector;
import org.apache.commons.math3.linear.RealMatrix;
import org.apache.commons.math3.linear.RealVector;
import org.apache.commons.math3.util.CombinatoricsUtils;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.function.Function;

public class RunCommand implements RunnableCommand, ApplicationStateAware
{

    private static final String VARIANT_ONE_TXT = "Variant_1.txt";

    private static final String VARIANT_TWO_TXT = "Variant_2.txt";

    private static final String VARIANT_THREE_TXT = "Variant_3.txt";

    private ApplicationState state;

    @Override
    public void execute(String[] strings) {
        ValidationUtils.requireNonNull(state);
        MatricesDto matricesDto = computeMatrices();
        List<RealVector> sequenceY = getSequenceOfY(matricesDto, strings[0]);
        if (!sequenceY.isEmpty()) {
            double T = (double) state.getVariable("T");
            ChartHelper.getInstance().showNextChart(sequenceY, T);
        }
    }

    @Override
    public void setApplicationState(ApplicationState applicationState) {
        ValidationUtils.requireNonNull(applicationState);
        this.state = applicationState;
    }

    private List<RealVector> getSequenceOfY(MatricesDto dto, String variant) {
        RealVector one = new ArrayRealVector(new double[]{1.0});
        RealVector minusOne = new ArrayRealVector(new double[]{-1.0});
        int period = getIterationCount(1);
```

```
        Function<Integer, RealVector> alternateMapper = (i) -> {
            if ((i / period) % 2 == 0) {
                return one;
            }
            return minusOne;
        };
        switch (variant) {
            case "1":
                return computeYsAndWriteXs(dto, (i) -> one, getIterationCount(2),
                        new FileSystemVectorXDao(VARIANT_ONE_TXT));
            case "2":
                return computeYsAndWriteXs(dto, alternateMapper, getIterationCount(2),
                        new FileSystemVectorXDao(VARIANT_TWO_TXT));
            case "3":
                return computeYsAndWriteXs(dto, alternateMapper, getIterationCount(3),
                        new FileSystemVectorXDao(VARIANT_THREE_TXT));
        }
        ConsoleUtils.println(String.format("Unknown variant: %s", variant));
        return new ArrayList<>();
    }

    private List<RealVector> computeYsAndWriteXs(MatricesDto dto,
Function<Integer, RealVector> iterationStepToUTransformer,
                                         int iterationCount, VectorXDao dao) {
        List<RealVector> out = new ArrayList<>(iterationCount);
        RealMatrix C = (RealMatrix) state.getVariable("C");
        RealVector previousX = new ArrayRealVector(dto.getG().getRowDimension());
        for (int i = 0; i < iterationCount; i++) {
            RealVector y = C.operate(previousX);
            out.add(y);
            if (i % 100 == 0) {
                dao.write(i, previousX);
            }
            RealVector u = iterationStepToUTransformer.apply(i);
            previousX = computeVectorX(dto.getG(), dto.getF(), u, previousX);
        }
        return out;
    }

    private int getIterationCount(int coefficient) {
        double k = (double) ((int) state.getVariable("k"));
        double T = (double) state.getVariable("T");
        int out = coefficient * (Math.max((int) (k / T), 1));
        if (out < 0) {
            throw new IllegalStateException("Iteration count is negative");
        }
        return out;
    }
```

```
    private RealVector computeVectorX(RealMatrix G, RealMatrix F, RealVector u,
RealVector previousX) {
        return F.operate(previousX).add(G.operate(u));
    }

    private MatricesDto computeMatrices() {
        double a1 = (double) state.getVariable("a1");
        double a2 = (double) state.getVariable("a2");
        double[] coefficients = {1, a1, a2};
        RealMatrix A = MatrixUtils.getFrobeniusMatrix(coefficients);
        int q = (int) state.getVariable("q");
        Map<Integer, RealMatrix> powerToMatrixInThatPower =
MatrixUtils.getPowerToMatrixInThatPower(A, q);
        double T = (double) state.getVariable("T");
        RealMatrix F = computeMatrixF(A, T, q, powerToMatrixInThatPower);
        RealMatrix B = (RealMatrix) state.getVariable("B");
        RealMatrix G = computeMatrixG(A, B, T, q, powerToMatrixInThatPower);
        return new MatricesDto(F, G);
    }

    private RealMatrix computeMatrixF(RealMatrix A, double T, int q,
                                      Map<Integer, RealMatrix> powerToMatrixInThatPower) {
        RealMatrix F = new Array2DRowRealMatrix(A.getRowDimension(),
A.getColumnDimension());
        for (int i = 0; i <= q; i++) {
            RealMatrix matrixToAdd = powerToMatrixInThatPower.get(i)
                    .scalarMultiply(Math.pow(T, i)).scalarMultiply(1.0 /
CombinatoricsUtils.factorial(i));
            F = F.add(matrixToAdd);
        }
        return F;
    }

    private RealMatrix computeMatrixG(RealMatrix A, RealMatrix B, double T, int q,
                                      Map<Integer, RealMatrix> powerToMatrixInThatPower) {
        RealMatrix G = new Array2DRowRealMatrix(A.getRowDimension(),
A.getColumnDimension());
        for (int i = 0; i <= q - 1; i++) {
            RealMatrix matrixToAdd = powerToMatrixInThatPower.get(i)
                    .scalarMultiply(Math.pow(T, i)).scalarMultiply(1.0 /
CombinatoricsUtils.factorial(i + 1));
            G = G.add(matrixToAdd);
        }
        G = G.scalarMultiply(T);
        G = G.multiply(B);
        return G;
    }
```

```
}
```

```
package chart;

import dto.ChartDto;
import org.apache.commons.math3.linear.RealVector;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
import org.knowm.xchart.XYSeries;
import org.knowm.xchart.style.markers.SeriesMarkers;

import javax.swing.*;
import java.util.ArrayList;
import java.util.List;

public class ChartHelper {

    private static final ChartHelper INSTANCE = new ChartHelper();

    private JFrame previousChart;

    private ChartHelper() {
    }

    public static ChartHelper getInstance() {
        return INSTANCE;
    }

    public void showNextChart(List<RealVector> sequenceY, double T) {
        if (previousChart != null) {
            previousChart.dispose();
        }
        ChartDto dto = getChartDto(sequenceY, T);
        XYChart chart = getChart(dto.getXData(), dto.getYData());
        this.previousChart = new SwingWrapper<>(chart).displayChart();

this.previousChart.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOS
E);
    }

    private XYChart getChart(List<? extends Number> xData, List<? extends
Number> yData) {
        XYChart chart = new XYChart(1600, 900);
        chart.setTitle("Chart");
        chart.setXAxisTitle("t");
        chart.setYAxisTitle("y(t)");
```

```
        XYSeries series = chart.addSeries("y(t)", xData, yData);
        series.setMarker(SeriesMarkers.NONE);
        return chart;
    }

    private ChartDto getChartDto(List<RealVector> sequenceY, double T) {
        int factor = 1000;
        int step = Math.max(sequenceY.size() / factor, 1);
        List<Double> yData = new ArrayList<>(factor);
        List<Double> xData = new ArrayList<>(factor);
        for (int i = 0; i < sequenceY.size(); i += step) {
            RealVector vector = sequenceY.get(i);
            if (vector.getDimension() == 1) {
                yData.add(vector.getEntry(0));
                xData.add(i * T);
            }
        }
        return new ChartDto(xData, yData);
    }
}
```

laboratory-framework.properties:

```
#**********************************
#*          Application           *
#**********************************
application.name=System Analysis Lab #1
application.author=Arzamastsev Vladyslav, SATR-3
application.description=Numerical methods for studying transient processes in linear stationary systems.\n\
  Linear stationary system:\n\
  x'[t] = Ax(t) + Bu(t), t \u2208 [0, \u221E]\n\
  y(t) = Cx(t), x(t) = x0 if t = 0\n\ \n\
  Here:\n\
  \t* x(t) \u2208 R[n] - vector of system's state in a moment in time 't', t \u2208 R[1]\n\
  \t* u(t) \u2208 R[m] - input process\n\
  \t* y(t) \u2208 R[l] - output process\n\
  \t* A \u2208 R[n * n], B \u2208 R[n * m], C \u2208 R[l * n] -matrices of system's parameters (m <= n, l <= n)\n\ \n\
  Details:\n\
  There are 3 variants. To run any of them, use 'run x', x \u2208 \{1, 2, 3\}. After running command, you'll\n\
  get a chart of process and series of x-s at '*your-home*/Documents/SystemAnalysisLab_1/' directory\n\
  C = (1, 0, 0)\n\
  B = (0, 0, 1)\u1D40\n\
  .   0   1   0\n\
  A = 0   0   1\n\
  .  -1  a1  a2\n\
```

```
  u - varies, depending on variant:\n\
  \t* If '1', u = 1;\n\
  \t* If '2', u = 1 the first half of iteration and then u = -1;\n\
  \t* If '3', u = 1 the first 1/3 of iteration, then u = -1 for the second 1/3 of it-n, then u
= 1 for the last 1/3 of it-n


#**********************************
#*          Variables           *
#**********************************
# Supported types:
# BIG_DECIMAL, BIG_INTEGER, BYTE, SHORT, INTEGER, LONG, BOOLEAN,
# CHARACTER, FLOAT, DOUBLE, STRING, VECTOR, MATRIX;
variable.T.name=T
variable.T.type=DOUBLE
variable.T.description=Quantization period. T >= 0
variable.T.constraint-violation-message=T must be >= 0


variable.q.name=q
variable.q.type=INTEGER
variable.q.description=Precision - the maximum degree of expansion of a function in
a Taylor series. q \u2208 [2, 10]
variable.q.constraint-violation-message=q must be >= 2 and <= 10


variable.k.name=k
variable.k.type=INTEGER
variable.k.cannot-be-set-from-input=true
variable.k.description=parameter that is used to compute iteration count (iter-count
= variant-coefficient * (k / T))
variable.k.constraint-violation-message=k >= 0


variable.a1.name=a1
variable.a1.type=DOUBLE
variable.a1.description='a1' parameter in the A matrix
variable.a1.constraint-violation-message=


variable.a2.name=a2
variable.a2.type=DOUBLE
variable.a2.description='a2' parameter in the A matrix
variable.a2.constraint-violation-message=


variable.b.name=b
variable.b.type=DOUBLE
variable.b.description='b' parameter in the B matrix
variable.b.constraint-violation-message=


variable.C.name=C
variable.C.type=MATRIX
variable.C.cannot-be-set-from-input=true
```

```
variable.C.description=The 'C' matrix from the linear stationary system
variable.C.constraint-violation-message=Invalid matrix
variable.C.matrix-row-count=1
variable.C.matrix-column-count=3

#*********************************
#*         Commands              *
#*********************************
command.run.name=run
command.run.arity=1
command.run.description=Get results of laboratory work with specified variant (can
be 1, 2 or 3)
command.run.constraint-violation-message=Make sure you have specified all
required arguments
```

pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>SystemAnalysysLab1</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.vladyslav.arzamastsev.framework.simple-laboratory-
framework</groupId>
            <artifactId>SimpleLaboratoryFramework</artifactId>
            <version>1.0.0</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.20</version>
            <scope>provided</scope>
```

```
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
                <configuration>
                    <annotationProcessorPaths>

                        <path>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                            <version>1.18.20</version>
                        </path>

                    </annotationProcessorPaths>
                    <source>11</source>
                    <target>11</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```