**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Курсовой проект**

по курсу «Численные методы».

Тема: «Autoganerator».

Студент: Якимович А.И.

Группа: 80-308Б

Вариант: 22 («Autogenarator»)

Оценка:

Москва, 2017

Постановка задачи.

Модель автогенератора с запаздыванием аргументов описывается уравнением:

.

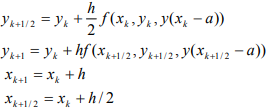
1. Какие решения существуют при и вариации ?
2. Исследуйте динамику уравнения при вариации и
3. ,
4. ,

ВЭ начинайте от значения и постепенно увеличивайте . Найдите интервалы существования стационарных решений, периодических решений и хаоса. Как меняется расстояние между экстремумами при увеличении (соответственно уменьшении )? Каков сценарий перехода к хаосу?

Описание методов.

**Модифицированный метод Эйлера для уравнений с запаздывающим аргументом.**

Задано дифференциально уравнение с запаздывающим аргументом *y’ = f(x, y, y(x – a))* и условие *.* Требуется найти сеточную функцию *y* с шагом *h* на отрезке *[a, b],* где . Итерационный процесс выглядит следующим образом:



Опишем процедуру нахождения решения в точках на *k+1-*ом шаге. Если лежит левее начальной точки , то определяется из начальных условий (в этом случае должно быть задано поведение решения на интервале левее точки, достаточном для определения значения в точке . Если совпадает с одним из узлов правее , тогда принимает значение функции в этом узле. Если величина не совпадает ни с одним узловым значением, то она лежит внутри некоторого отрезка и можно по значениям *y* в трех узлах построить интерполяционный многочлен для определения приближенного значения. Контроль точности осуществляется методом Румбе-Ромберга или путём сравнения с точным результатом.

Общая информация.

Данная работа состоит из 6 модулей, которые позволяют решать различные задачи решения СЛАУ и нахождения собственных векторов и собственных значений. Полученные в ходе расчетов результаты сохраняются в отдельный файл. Что касается технических деталей реализации, все программы написаны на языке Java.

Входные данные и результаты.

Вариант 1.

|  |  |
| --- | --- |
| // {a1, a3}  // {a2, ...}  {1.0, 0.0},  {0.0, 0.5, 1.0, 1.5, 2.0} |  |
| {1.0, 15.0},  {0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.1} | **C:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_2_1_a1=1.0_a2=0.5_a3=15.0.pngC:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_2_2_a1=1.0_a2=0.6_a3=15.0.pngC:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_2_3_a1=1.0_a2=0.7_a3=15.0.pngC:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_2_4_a1=1.0_a2=0.8_a3=15.0.png** |
| {1.5, 10.0},  {0.5, 0.8, 1.0, 1.3, 1.5, 1.7, 2.0, 2.1, 2.3, 2.5, 2.6, 3.0, 3.3, 3.5, 3.6} | C:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_3_1_a1=1.5_a2=0.5_a3=10.0.png |
| //Экстремумы  {1.0, 15.0},  {0.5, 0.9, 1.4} | C:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_4_1_a1=1.0_a2=0.5_a3=15.0.png  Расстояние между экстремумами: 15.009999999999724  C:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_4_2_a1=1.0_a2=0.9_a3=15.0.png  Расстояние между экстремумами: 15.969999999985475C:\Users\prost(polzovatel)\AppData\Local\Microsoft\Windows\INetCache\Content.Word\plot_4_3_a1=1.0_a2=1.4_a3=15.0.png |
| // При увеличении a1 (в 2 раза)  {2.0, 15.0},  {0.5, 0.9, 1.4} | Расстояние между экстремумами: 15.009999999999724  Расстояние между экстремумами: 15.009999999999724  Расстояние между экстремумами: 15.009999999999724 |
| // При уменьшении a3 (в 3 раза)  {1.0, 5.0},  {0.5, 0.9, 1.4} | Расстояние между экстремумами: 5.009999999999938  Расстояние между экстремумами: 5.679999999999879  Расстояние между экстремумами: 5.4600000000008535 |

Выводы.

a1 = 1.5 a3 = 10 Стационарные 0.5 < a2 <1.7 Периодические 1.7 < a2 <3.3 Хаос при a2 >= 3.3

a1= 1, a3 = 15 Стационарные 0.5 < a2 < 0.7 Периодические 0.7 < a2 <1.5 Хаос a2 >= 1.5

При увеличении расстояние между экстремумами почти меняется, при уменьшении в k раз, расстояние уменьшается во столько же раз.

Исходный код.

//Vector.java

package libnm.math;  
  
import java.util.Arrays;  
  
public class Vector {  
 public Vector() {  
 this(1);  
 }  
  
 public Vector(int size) {  
 m\_vec = new double[size];  
 }  
  
 public Vector(double[] array) {  
 m\_vec = array.clone();  
 }  
  
 public void resize(int size) {  
 m\_vec = new double[size];  
 }  
  
 public double get(int index) {  
 return m\_vec[index];  
 }  
  
 public void set(int index, double value) {  
 m\_vec[index] = value;  
 }  
  
 public int getSize() {  
 return m\_vec.length;  
 }  
  
 public Vector add(Vector other) {  
 Vector res = new Vector(getSize());  
  
 for (int i = 0; i < getSize(); ++i) {  
 res.set(i, get(i) + other.get(i));  
 }  
  
 return res;  
 }  
  
 public Vector sub(Vector other) {  
 Vector res = new Vector(getSize());  
  
 for (int i = 0; i < getSize(); ++i) {  
 res.set(i, get(i) - other.get(i));  
 }  
  
 return res;  
 }  
  
 public double normC() {  
 double res = 0.0;  
  
 for (int i = 0; i < getSize(); ++i) {  
 double curX = Math.abs(get(i));  
  
 if (curX > res) {  
 res = curX;  
 }  
 }  
  
 return res;  
 }  
  
 public void copy(Vector other) {  
 for (int i = 0; i < getSize(); ++i) {  
 set(i, other.get(i));  
 }  
 }  
  
 public void swap(int index1, int index2) {  
 double tmp = get(index1);  
  
 set(index1, get(index2));  
 set(index2, tmp);  
 }  
  
 @Override  
 public String toString() {  
 return Arrays.toString(m\_vec);  
 }  
  
 private double[] m\_vec;  
}

//PolynomLagrange.java

package main;  
  
import libnm.math.Vector;  
  
public class PolynomLagrange {  
 public PolynomLagrange(Vector vecX, Vector vecY) {  
 Vector vecW = new Vector(vecX.getSize());  
  
 m\_vecX = vecX;  
 m\_vecY = new Vector(vecX.getSize());  
 m\_vecY.copy(vecY);  
  
 for (int i = 0; i < m\_vecX.getSize(); ++i) {  
 double w = 1.0;  
  
 for (int j = 0; j < m\_vecX.getSize(); ++j) {  
 if (i != j) {  
 w \*= m\_vecX.get(i) - m\_vecX.get(j);  
 }  
 }  
  
 vecW.set(i, w);  
 }  
  
 for (int i = 0; i < m\_vecX.getSize(); ++i) {  
 m\_vecY.set(i, m\_vecY.get(i) / vecW.get(i));  
 }  
 }  
  
 public double getValue(double x) {  
 int n = m\_vecX.getSize();  
 double res = 0.0;  
  
 for (int i = 0; i < n; ++i) {  
 double w = 1.0;  
  
 for (int j = 0; j < n; ++j) {  
 if (i != j) {  
 w \*= x - m\_vecX.get(j);  
 }  
 }  
  
 res += m\_vecY.get(i) \* w;  
 }  
  
 return res;  
 }  
  
 private Vector m\_vecX;  
 private Vector m\_vecY;  
}

//Plotter.java

package util;  
  
import java.awt.\*;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import de.erichseifert.gral.data.DataSeries;  
import de.erichseifert.gral.data.DataTable;  
import de.erichseifert.gral.graphics.\*;  
import de.erichseifert.gral.graphics.Label;  
import de.erichseifert.gral.io.plots.DrawableWriter;  
import de.erichseifert.gral.io.plots.DrawableWriterFactory;  
import de.erichseifert.gral.plots.XYPlot;  
import de.erichseifert.gral.plots.axes.AxisRenderer;  
import de.erichseifert.gral.plots.lines.DefaultLineRenderer2D;  
import de.erichseifert.gral.plots.lines.LineRenderer;  
import de.erichseifert.gral.plots.points.DefaultPointRenderer2D;  
import de.erichseifert.gral.plots.points.PointRenderer;  
import libnm.math.Vector;  
  
public class Plotter {  
 public Plotter(double width, double height) {  
 m\_width = width;  
 m\_height = height;  
  
 m\_initPlot();  
 }  
  
 public void addData(Vector vecX, Vector vecY, Color color, String legend) {  
 DataTable data = new DataTable(Double.class, Double.class);  
 PointRenderer pointRenderer = new DefaultPointRenderer2D();  
 LineRenderer lineRenderer = new DefaultLineRenderer2D();  
 DataSeries dataSeries = new DataSeries(legend, data);  
  
 for (int i = 0; i < vecX.getSize(); ++i) {  
 data.add(vecX.get(i), vecY.get(i));  
 }  
  
 pointRenderer.setColor(new Color(0, 0, 0, 0));  
 lineRenderer.setColor(color);  
  
 m\_plot.add(dataSeries);  
 m\_plot.setPointRenderers(dataSeries, pointRenderer);  
 m\_plot.setLineRenderers(dataSeries, lineRenderer);  
 }  
  
 public void clearData() {  
 m\_initPlot();  
 }  
  
 public void savePng(String fileName) {  
 DrawableWriter writer = DrawableWriterFactory.getInstance().get("image/png");  
  
 try {  
 writer.write(m\_plot, new FileOutputStream(fileName), m\_width, m\_height);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void m\_initPlot() {  
 final double MARGIN = 64.0;  
  
 m\_plot = new XYPlot();  
 m\_plot.setInsets(new Insets2D.Double(MARGIN, MARGIN, MARGIN, MARGIN));  
 m\_plot.setLegendVisible(true);  
  
 AxisRenderer axisX = m\_plot.getAxisRenderer(XYPlot.AXIS\_X);  
 AxisRenderer axisY = m\_plot.getAxisRenderer(XYPlot.AXIS\_Y);  
  
 axisX.setLabel(new Label("X"));  
 axisY.setLabel(new Label("Y"));  
 }  
  
 private double m\_width;  
 private double m\_height;  
 private XYPlot m\_plot;  
}

//Logger.java

package libnm.util;  
  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class Logger {  
 public Logger(String fileName) {  
 try {  
 m\_br = new BufferedWriter(new FileWriter(fileName));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void close() {  
 try {  
 m\_br.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void write(String text) {  
 try {  
 m\_br.write(text);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void writeln() {  
 try {  
 m\_br.write("\n");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void writeln(String text) {  
 try {  
 m\_br.write(text + "\n");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private BufferedWriter m\_br;  
}

//Euler.java

package main;  
  
import libnm.math.Vector;  
import libnm.util.Logger;  
  
public class Euler {  
 public Euler(double a1, double a2, double a3, double y0, double a, double b, double h) {  
 m\_a1 = a1;  
 m\_a2 = a2;  
 m\_a3 = a3;  
 m\_y0 = y0;  
 m\_a = a;  
 m\_b = b;  
 m\_h = h;  
 m\_logger = null;  
 }  
  
 public void setLogger(Logger logger) {  
 m\_logger = logger;  
 }  
  
 public void solve(Vector vecX, Vector vecY) {  
 int n = (int)((m\_b - m\_a) / m\_h) + 1;  
 int maxInd = 0;  
 int minInd = 0;  
 double maxValue = m\_y0;  
 double minValue = m\_y0;  
  
 vecX.set(0, m\_a);  
 vecY.set(0, m\_y0);  
  
 for (int i = 1; i < n; ++i) {  
 double x = vecX.get(i - 1);  
 double y = vecY.get(i - 1);  
 double xkh = x + 0.5 \* m\_h;  
 double ykh = y + 0.5 \* m\_h \* f(vecX, vecY, y, x - m\_a3);  
 double yk = y + m\_h \* f(vecX, vecY, ykh, xkh - m\_a3);  
  
 vecX.set(i, x + m\_h);  
 vecY.set(i, yk);  
  
 if (vecY.get(i) > maxValue) {  
 maxValue = vecY.get(i);  
 maxInd = i;  
 }  
  
 if (vecY.get(i) < minValue) {  
 minValue = vecY.get(i);  
 minInd = i;  
 }  
 }  
  
 if (m\_logger != null) {  
 m\_logger.writeln("Минимум функции в точке x = " + vecX.get(minInd) + ", y = " + minValue);  
 m\_logger.writeln("Максимум функции в точке x = " + vecX.get(maxInd) + ", y = " + maxValue);  
 m\_logger.writeln("Расстояние между экстремумами: " + Math.*abs*(vecX.get(maxInd) - vecX.get(minInd)));  
 }  
 }  
  
 private double f(Vector vecX, Vector vecY, double y, double offset) {  
 int n = vecX.getSize();  
 int ind = -1;  
 double yFinal;  
  
 for (int i = 0; i < n; ++i) {  
 if (vecX.get(i) > offset) {  
 break;  
 }  
  
 ind = i;  
 }  
  
 if (offset < m\_a) {  
 yFinal = m\_y0;  
 } else if (vecX.get(ind) == offset) {  
 yFinal = vecY.get(ind);  
 } else if (ind > 0 && ind < n - 1){  
 Vector tmpVecX = new Vector(new double[] {vecX.get(ind - 1), vecX.get(ind), vecX.get(ind + 1)});  
 Vector tmpVecY = new Vector(new double[] {vecY.get(ind - 1), vecY.get(ind), vecY.get(ind + 1)});  
 PolynomLagrange poly = new PolynomLagrange(tmpVecX, tmpVecY);  
  
 yFinal = poly.getValue(offset);  
 } else {  
 yFinal = m\_y0;  
 }  
  
 return m\_a2 - m\_a1 \* y - Math.*pow*(yFinal, 2.0);  
 }  
  
 private double m\_a1;  
 private double m\_a2;  
 private double m\_a3;  
 private double m\_y0;  
 private double m\_a;  
 private double m\_b;  
 private double m\_h;  
 private Logger m\_logger;  
}

//Main.java

package main;  
  
import java.awt.\*;  
  
import libnm.math.Vector;  
import libnm.util.Logger;  
import util.Plotter;  
  
public class Main {  
 public static void main(String[] args) {  
 double[][] paramsA = {  
 // {a1, a3}  
 // {a2, ...}  
 // Задание 1  
 {1.0, 0.0},  
 {0.0, 0.5, 1.0, 1.5, 2.0},  
 // Задание 2 (а)  
 {1.0, 15.0},  
 {0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.1},  
 // Задание 2 (б)  
 {1.5, 10.0},  
 {0.5, 0.8, 1.0, 1.3, 1.5, 1.7, 2.0, 2.1, 2.3, 2.5, 2.6, 3.0, 3.3, 3.5, 3.6},  
 // Экстремумы  
 {1.0, 15.0},  
 {0.5, 0.9, 1.4},  
 // При увеличении a1 (в 2 раза)  
 {2.0, 15.0},  
 {0.5, 0.9, 1.4},  
 // При уменьшении a3 (в 3 раза)  
 {1.0, 5.0},  
 {0.5, 0.9, 1.4}  
 };  
 double y0 = 0.3;  
 double a = 0.0;  
 double b = 300.0;  
 double h = 0.01;  
 int n = (int)((b - a) / h) + 1;  
 Vector vecX = new Vector(n);  
 Vector vecY = new Vector(n);  
  
 for (int i = 0; i < paramsA.length; i += 2) {  
 double a1 = paramsA[i][0];  
 double a3 = paramsA[i][1];  
  
 for (int j = 0; j < paramsA[i + 1].length; ++j) {  
 double a2 = paramsA[i + 1][j];  
 Logger logger = new Logger("src/data/logs/log\_" + (i / 2 + 1) + "\_" + (j + 1) + ".txt");  
 Plotter plotter = new Plotter(1024.0, 512.0);  
 Euler method = new Euler(a1, a2, a3, y0, a, b, h);  
  
 method.setLogger(logger);  
 method.solve(vecX, vecY);  
  
 plotter.addData(vecX, vecY, Color.*RED*, "Y");  
 plotter.savePng("src/data/plot\_" + (i / 2 + 1) + "\_" + (j + 1) + "\_a1=" + a1 + "\_a2=" + a2 + "\_a3=" + a3 + ".png");  
  
 logger.close();  
 }  
 }  
 }  
}