**Министерство образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(ВлГУ)**

Кафедра «Физика и прикладная математика»

**Лабораторная работа №2**

**по дисциплине «Неклассическая логика»**

**тема «Нечеткая арифметика»**

Выполнил:

ст. гр. ИТс-112

Власов И. Д.

Принял:

Абрахин С.И.

Владимир, 2014

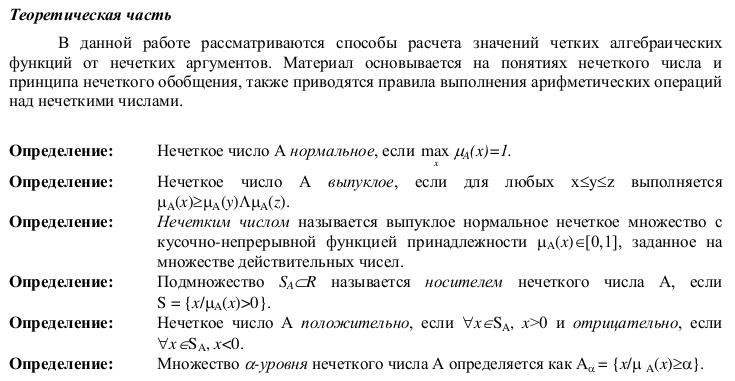
Цель работы:

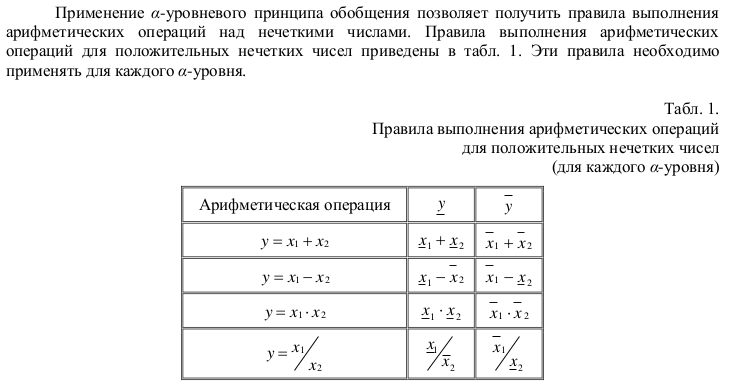
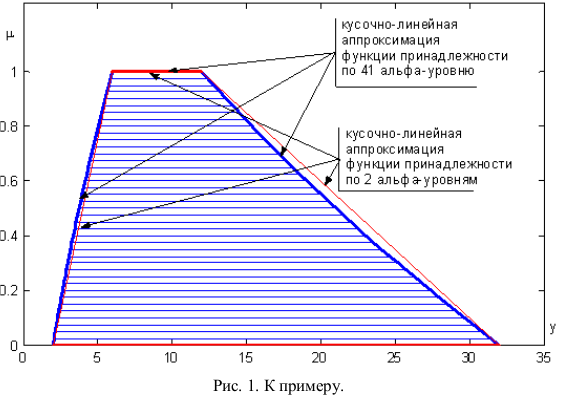
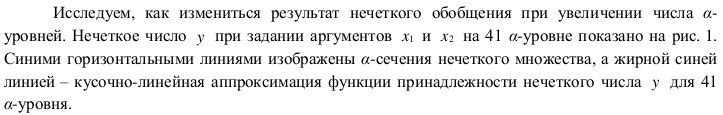
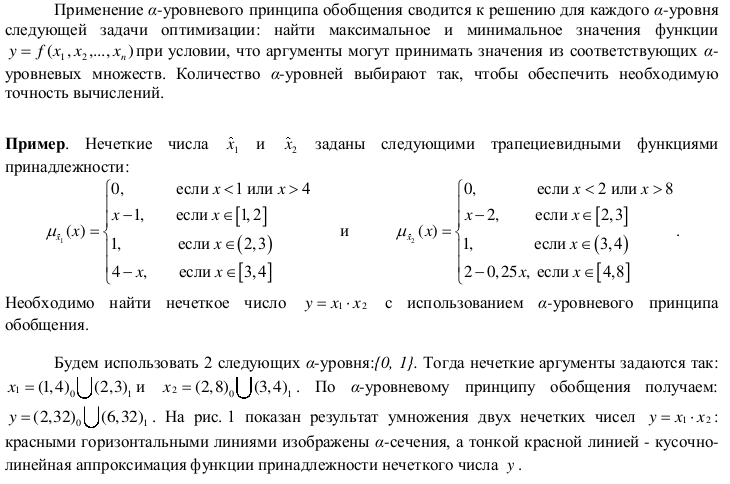
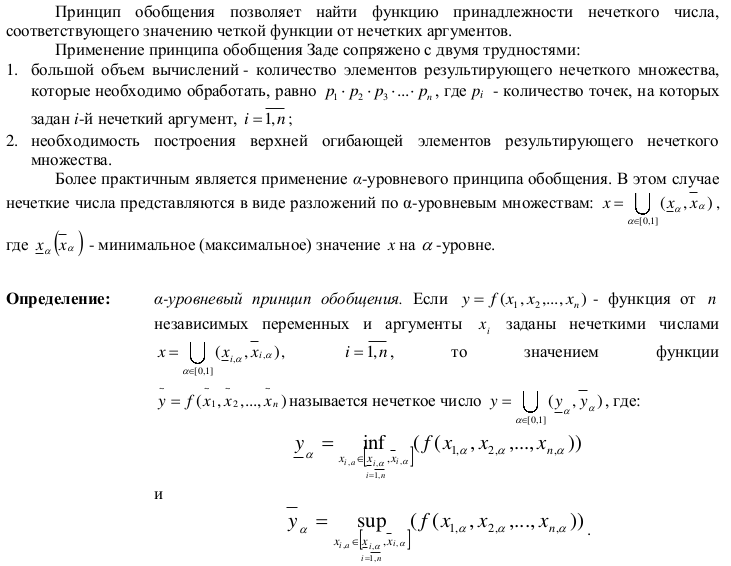
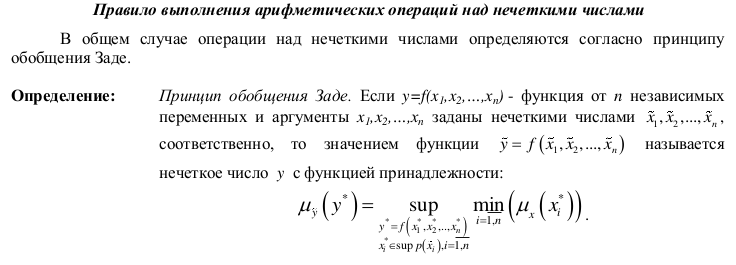
Изучить правила выполнения основных арифметических операций (сложение, вычитание, умножение, деление) с нечеткими числами, с применением α-уровневого принципа обобщения Заде.

Задача:

Перегрузить операции сложения, вычитания, умножения, деления, в соответствии с α-уровневым принципом обобщения Заде, для класса представления нечетких чисел (разработанного в лабораторной работе No2), функция принадлежности которых строится на основе прямых методов для одного эксперта, и хранится в объекте как динамическая структура, каждый элемент которой включает в себя тройку чисел { i , Ai , Ai }, где ai – значение i-уровня разложения, Ai – нижняя граница обычного множества разложения, Ai – верхняя граница обычного множества разложения.

Краткая теоретическая часть:





Выполнение:

package classes;

import java.awt.Graphics;

import java.util.ArrayList;

import java.util.List;

public abstract class FuzzySet {

private static int countExpansionLeavels = 100;

private List<Expansion> expansion;

private List<Double> parameters;

public FuzzySet(List<Expansion> expansion) {

this.expansion = expansion;

}

public FuzzySet(double ... parameters) {

this.parameters = new ArrayList<Double>();

for(double d : parameters)

this.parameters.add(d);

this.expansion = new ArrayList<Expansion>();

double alphaLevel = 1.0 / countExpansionLeavels;

for(int i = 0; i < countExpansionLeavels - 1; i++) {

expansion.add(new Expansion(alphaLevel,

calculatingLowerLimit(alphaLevel),

calculatingUpperLimit(alphaLevel)));

alphaLevel += 1.0 / countExpansionLeavels;

}

}

public static int getCountExpansionLeavels() {

return countExpansionLeavels;

}

public static void setCountExpansionLeavels(int countExpansionLeavels) {

FuzzySet.countExpansionLeavels = countExpansionLeavels;

}

public List<Double> getParameters() {

return parameters;

}

public void setParameters(List<Double> parameters) {

this.parameters = parameters;

}

protected abstract double calculatingLowerLimit(double alphaLevel);

protected abstract double calculatingUpperLimit(double alphaLevel);

public void draw(Graphics g) {

double maxValue = Double.MIN\_VALUE;

double minVlaue = Double.MAX\_VALUE;

int scale = 250;

int scaleWeight = 450;

for(int i = 0; i < expansion.size(); i++) {

double lowerLimit = expansion.get(i).getLowerLimit();

if (lowerLimit != -1) {

if (lowerLimit > maxValue) maxValue = lowerLimit;

if (lowerLimit < minVlaue) minVlaue = lowerLimit;

}

double upperLimit = expansion.get(i).getUpperLimit();

if (upperLimit != -1) {

if (upperLimit > maxValue) maxValue = upperLimit;

if (upperLimit < minVlaue) minVlaue = upperLimit;

}

}

double chartWidth = maxValue - minVlaue;

int [] xArrayLower = new int [expansion.size()];

int [] xArrayUpper = new int [expansion.size()];

int [] yArray = new int [expansion.size()];

double tmpLowerLimit = 0;

double tmpUpperLimit = 0;

boolean drawLowerLimit = false;

boolean drawUpperLimit = false;

for (int i = 0; i < expansion.size(); i++) {

if (expansion.get(i).getLowerLimit() != -1) {

tmpLowerLimit = expansion.get(i).getLowerLimit();

drawLowerLimit = true;

}

if (expansion.get(i).getUpperLimit() != -1) {

tmpUpperLimit = expansion.get(i).getUpperLimit();

drawUpperLimit = true;

}

xArrayLower[i] = (int)((tmpLowerLimit - minVlaue) / chartWidth \* scaleWeight) + 100;

xArrayUpper[i] = (int)((tmpUpperLimit - minVlaue) / chartWidth \* scaleWeight) + 100;

yArray[i] = (int)((1 - expansion.get(i).getAlphaLevel()) \* scale) + 100;

}

if (drawLowerLimit)

g.drawPolyline(xArrayLower, yArray, expansion.size());

if (drawUpperLimit)

g.drawPolyline(xArrayUpper, yArray, expansion.size());

}

public String toString() {

return this.getClass().getName() +

" (" + parameters.get(0) +

", " + parameters.get(1) +

", " + parameters.get(2) +

", " + parameters.get(3) + ")";

}

public FuzzySet add(FuzzySet fs) {

List<Expansion> expansions = new ArrayList<Expansion>();

for(int i = 0; i < this.expansion.size() - 1; i++) {

Expansion exp = new Expansion(this.expansion.get(i).getAlphaLevel(),

this.expansion.get(i).getLowerLimit() + fs.expansion.get(i).getLowerLimit(),

this.expansion.get(i).getUpperLimit() + fs.expansion.get(i).getUpperLimit());

expansions.add(exp);

}

return new OtherFuzzySet(expansions);

}

public FuzzySet sub(FuzzySet fs) {

List<Expansion> expansions = new ArrayList<Expansion>();

for(int i = 0; i < this.expansion.size() - 1; i++) {

Expansion exp = new Expansion(this.expansion.get(i).getAlphaLevel(),

this.expansion.get(i).getLowerLimit() - fs.expansion.get(i).getUpperLimit(),

this.expansion.get(i).getUpperLimit() - fs.expansion.get(i).getLowerLimit());

expansions.add(exp);

}

return new OtherFuzzySet(expansions);

}

public FuzzySet mul(FuzzySet fs) {

List<Expansion> expansions = new ArrayList<Expansion>();

for(int i = 0; i < this.expansion.size() - 1; i++) {

Expansion exp = new Expansion(this.expansion.get(i).getAlphaLevel(),

this.expansion.get(i).getLowerLimit() \* fs.expansion.get(i).getLowerLimit(),

this.expansion.get(i).getUpperLimit() \* fs.expansion.get(i).getUpperLimit());

expansions.add(exp);

}

return new OtherFuzzySet(expansions);

}

public FuzzySet div(FuzzySet fs) {

List<Expansion> expansions = new ArrayList<Expansion>();

for(int i = 0; i < this.expansion.size() - 1; i++) {

Expansion exp = new Expansion(this.expansion.get(i).getAlphaLevel(),

this.expansion.get(i).getLowerLimit() / fs.expansion.get(i).getUpperLimit(),

this.expansion.get(i).getUpperLimit() / fs.expansion.get(i).getLowerLimit());

expansions.add(exp);

}

return new OtherFuzzySet(expansions);

}

}

package classes;

public class FirstFunction extends FuzzySet {

public FirstFunction(double ... paraparameters) {

super(paraparameters);

}

double calcFirstFunction(double alphaLevel, double a, double b) {

if (alphaLevel < 0.5)

return a + Math.sqrt(alphaLevel \* Math.pow(b - a, 2) / 2);

else if (alphaLevel == 0.5)

return (a + b) / 2;

else

return b - Math.sqrt((1 - alphaLevel) \* Math.pow(b - a, 2) / 2);

}

@Override

protected double calculatingLowerLimit(double alphaLevel) {

double a = getParameters().get(0);

double b = getParameters().get(1);

return calcFirstFunction(alphaLevel, a, b);

}

@Override

protected double calculatingUpperLimit(double alphaLevel) {

return -1;

}

}

package classes;

import java.util.List;

public class OtherFuzzySet extends FuzzySet{

public OtherFuzzySet(List<Expansion> expansion) {

super(expansion);

}

@Override

protected double calculatingLowerLimit(double alphaLevel) {

return 0;

}

@Override

protected double calculatingUpperLimit(double alphaLevel) {

return 0;

}

}

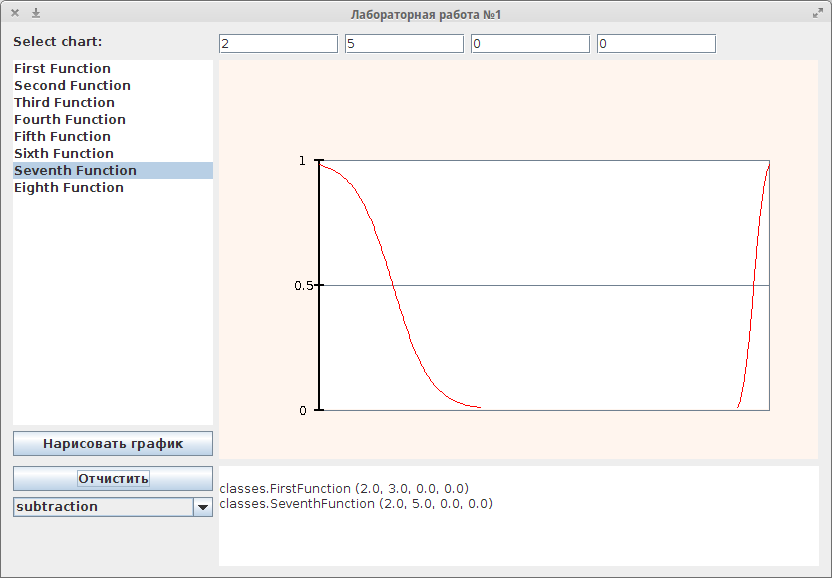


Рисунок 1 – Выполнение программы

Вывод: Изучил правила выполнения основных арифметических операций (сложение, вычитание, умножение, деление) с нечеткими числами, с применением α-уровневого принципа обобщения Заде.