Всероссийская научно-практическая конференция
Статистика. Моделирование. Оптимизация
Челябинск, 28 ноября – 3 декабря 2011 г.

АРХИТЕКТУРА БИБЛИОТЕКИ JInterval

Жилин С.И.
Алтайский госуниверситет,
Барнаул
sergei@asu.ru

Надёжин Д.Ю.
Оракл Девелопмент СПб
Москва
dmitry.nadezhin@gmail.com

Тепикин Е.Н.
Энтерра Софт
Барнаул
tepikinegor@gmail.com

План

- Предпосылки и мотивация
- Исходные требования
- Основные архитектурные решения
- Примеры использования JInterval
- Что дальше?

Архитектура библиотеки JInterval

ПРЕДПОСЫЛКИ И МОТИВАЦИЯ

Интервальные вычисления на Java

- Интервальные программные средства
  - Много инструментов разработчика: компиляторы, библиотеки
    - **>75%** в списке ***Interval and Related Software*** на сайте ***Interval Computations*** www.cs.utep.edu/interval-comp/intsoft.html
    - **>80%** в списке ***Программное обеспечение и языки программирования*** на сайте ***Интервальный анализ и его приложения*** www.nsc.ru/interval/
  - Мало продуктов для конечного пользователя
- Java – привлекательная технология для разработки прикладных программных продуктов
  - Межплатформная переносимость приложений
  - Высокоуровневый язык (языки)
  - Широкий выбор библиотек
  - Развитые средства для построения распределенных систем

Интервальные вычисления на Java

❑ Пригодна ли Java для научных вычислений?

За:

- Переносимость Java Virtual Machine (JVM)
- Безопасное управление памятью
- Встроенная поддержка сети
- Параллельные и распределенные вычисления (потоки, RMI)
- Строгая модель безопасности
- Стандартные интерфейсы для графики, GUI, БД
- Широкое распространение

Против:

- Невысокая производительность
  - Виртуальная машина
  - Медленные интерпретаторы
  - Накладные расходы на безопасный доступ к памяти
- Ограничения языка
  - Нет примитивного типа «структура»
  - Нет перегрузки операторов
  - Нет истинно многомерных массивов
  - Неполная поддержка IEEE 754
- Относительно небольшое количество научных библиотек на Java
- Традиции научных вычислений: Fortran, Си

Интервальные вычисления на Java

- Интервальные Java-библиотеки
  - IA_math, 1997
    - Timothy J. Hickey (Brandeis University, Boston, USA)
    - interval.sourceforge.net/interval/
  - Java-XSC, 1999
    - Benjamin R.C. Bedregal (Universidade Federal do Rio Grande do Norte, Natal, Brazil)
    - www.dimap.ufrn.br/~java-xsc/
  - Java-XSC, 2004
    - Marcilia A. Campos (Universidade Federal de Pernambuco, Recife, Brazil)
    - www.cin.ufpe.br/~javaxsc

Архитектура библиотеки JInterval

ИСХОДНЫЕ ТРЕБОВАНИЯ

Требования к интервальной библиотеке

1. Понятность и удобство для пользователя
2. Гибкость в выборе используемой в вычислениях интервальной арифметики
3. Гибкость в расширении ее функциональных возможностей
4. Гибкость в выборе базовых типов для концов интервалов и связанных с ними политик округления
5. Переносимость
6. Высокая производительность вычислений

Требования к интервальной библиотеке

1. **Понятность и удобство для пользователя**

   – Сколь замечательным ни был бы программный инструмент, вряд ли он окажется востребован, если не прозрачен и не комфортен для пользователя

Требования к интервальной библиотеке

2. **Гибкость в выборе интервальной арифметики, используемой в вычислениях**

   – Востребованные интервальные арифметики
     - классическая интервальная арифметика
     - полная интервальная арифметика Каухера
     - комплексные интервальные арифметики (прямоугольная, круговая, и т.д.)

   – Необходима возможность выбора арифметики и перехода при вычислениях от одной арифметики к другой (в случае их совместимости)

   – Синтаксические различия при работе с различными арифметиками должны быть минимизированы

Требования к интервальной библиотеке

3. Гибкость в расширении ее функциональных возможностей

- Функциональность библиотеки подразделяется на слои
  - Интервальные арифметические операции
  - Элементарные интервальные функции
  - Интервальные векторно-матричные операции.
  - Высокоуровневые методы интервального анализа
    - решатели алгебраических уравнений и их систем
    - решатели дифференциальных уравнений и их систем
    - процедуры оптимизации

- Архитектура библиотеки должна допускать расширение и пополнение на всех слоях

Требования к интервальной библиотеке

4. **Гибкость в выборе базовых типов для концов интервалов и связанных с ними политик округления**

   – Для доказательных вычислений и гарантированных результатов необходимо точное представление границ интервалов и/или учитывать влияние ошибок округления

   – При выполнении прикидочных расчетов, анализе данных с большой исходной неопределенностью управление режимом округления и борьба за гарантированность результатов лишены смысла

   – Возможность выбора физического представления границ интервалов и режима округления позволит пользователю влиять на эффективность вычислений в смысле их точности и скорости

Требования к интервальной библиотеке

5. Переносимость

- Межплатформная переносимость библиотеки – одно из главных ее достоинств и ключевых отличий от прочих интервальных библиотек
- В значительной мере это требование обеспечивается выбором Java-технологии, построенной по принципу «написано однажды – работает везде»
- Для практического воплощения этого девиза при разработке необходимо придерживаться некоторых ограничений

Требования к интервальной библиотеке

6. Высокая производительность вычислений

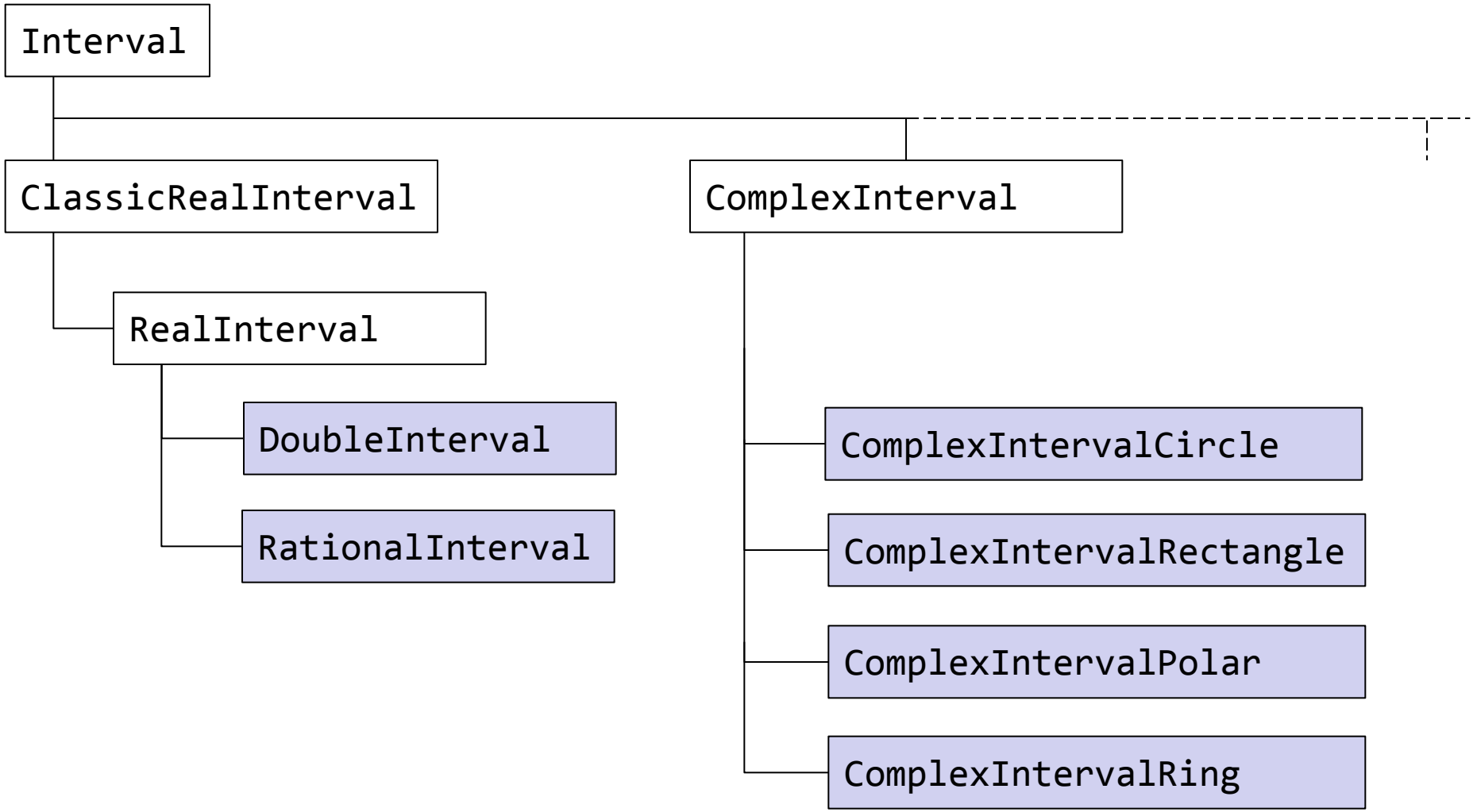
- Чаще необходима при крупномасштабных научных вычислениях, реже – в прикладных программных продуктах

- В эпоху многоядерных и многопроцессорных систем необходимо обеспечить возможности безопасного использования библиотеки при разработке многопоточных приложений

Архитектура библиотеки JInterval

ОСНОВНЫЕ АРХИТЕКТУРНЫЕ РЕШЕНИЯ

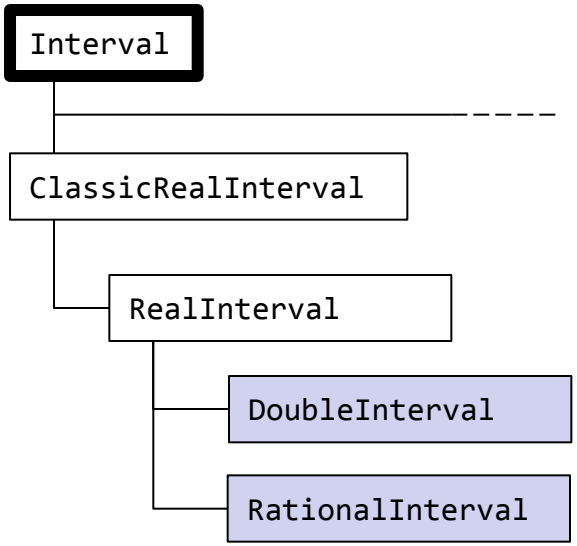
Фрагмент иерархии типов



Унификация имен интервальных операций

- Вне зависимости от типа интервалов
  - вещественный классический
  - вещественный каухеров
  - комплексный прямоугольный
  - комплексный круговой
  - …

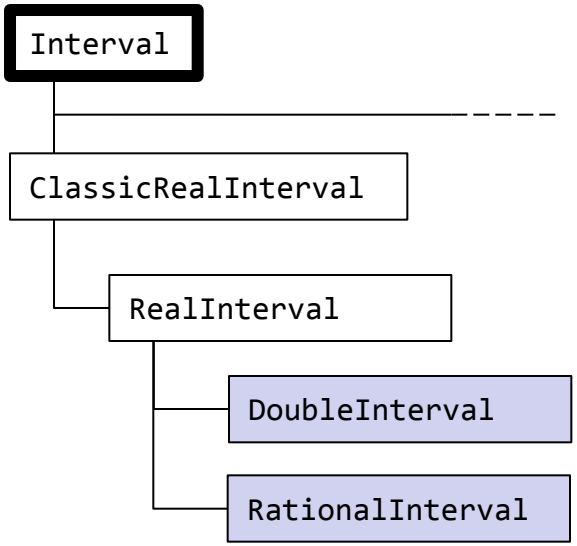
  имена основных операций над интервалами унифицированы и сведены в интерфейсе `Interval`



- Интерфейс `Interval` – предельно общая абстракция интервальных алгебр

Интерфейс Interval

```
public interface Interval<I extends Interval<I>> {

    public static enum RoundingMode {

        EXACT, OUTWARDS, INWARDS, ANY
    }

    public I add(I a);

    public I subtract(I a);

    public I negate();

    public I multiply(I a);

    public I divide(I a);

    public I intersect(I a);

    public I union(I a);

    public boolean subset(I a);

    public boolean isEmpty();
}
```

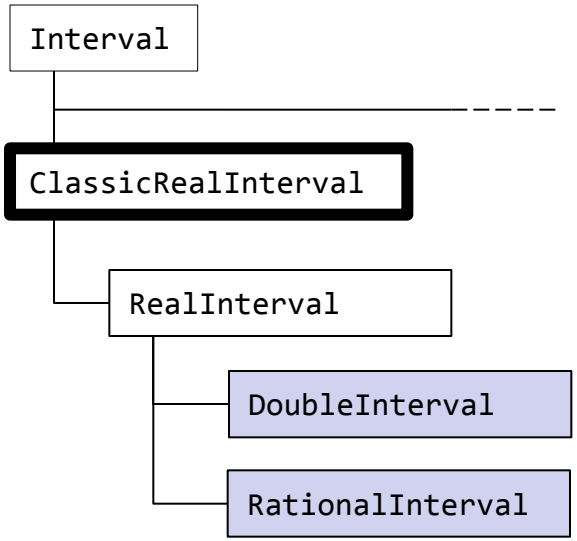


Унификация синтаксиса и семантики операций над классическими интервалами

- Вне зависимости от
  - типа границ интервала
  - поддерживаемых режимов округления

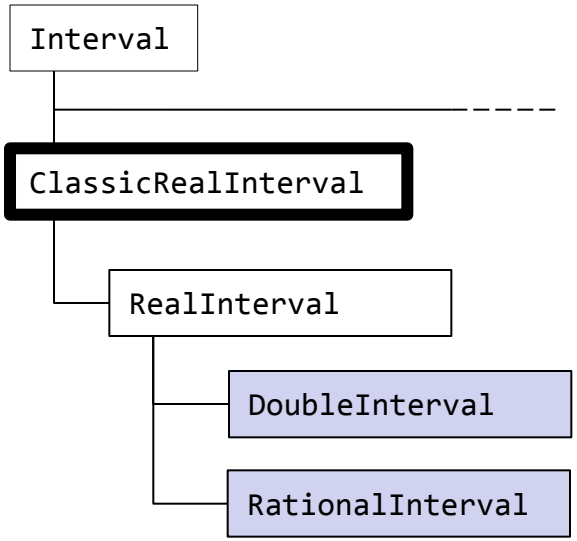
  вещественные классические интервалы реализуют интерфейс `ClassicRealInterval`

- В `ClassicRealInterval` включены
  - операции отношения
  - элементарные функции
  - функции, возвращающие границы интервала как
    - объекты типа `Rational` (точное значение)
    - значения типа `double`



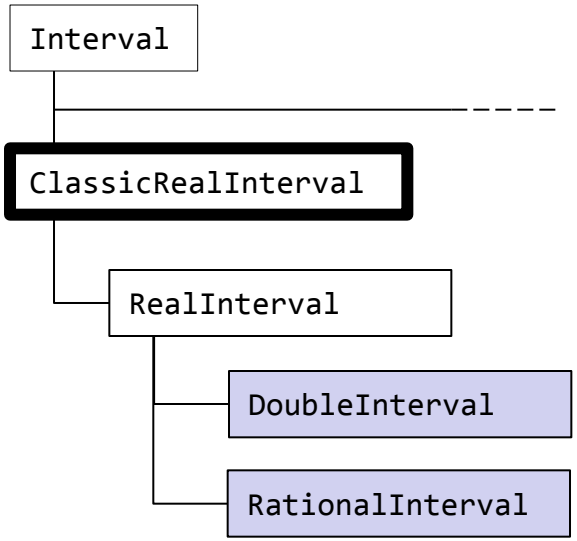
Интерфейс ClassicRealInterval

```
public interface ClassicRealInterval<I extends ClassicRealInterval<I>>
                                     extends Interval<I> {

    public Rational exactWid();

    public double doubleWid();

    public Rational exactMid();

    public double doubleMid();

    public ClassicRealIntervalFactory<I> getFactory_();


    public Rational exactInf();

    public Rational exactSup();

    public double doubleInf();

    public double doubleSup();

    ...
```



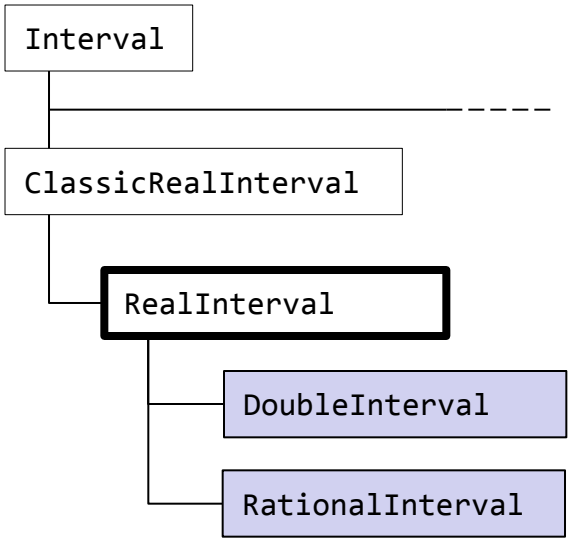
Интерфейс ClassicInterval

```
public interface ClassicRealInterval<I extends ClassicRealInterval<I>>
                                     extends Interval<I> {
    ...
    public I pow(I a);

    public I pow(long n);

    public I exp();

    public I sin();

    public I cos();

    public I tan();

    public I asin();

    public I acos();

    public I atan();

    public I log();
    ...
}
```



Унификация управления разрядностью границ и округлением

- Класс `RealInterval` содержит методы, позволяющие динамически настраивать точность операций
- Объекты этого класса могут иметь различную разрядность границ



Класс `RealInterval`: решения

- Интервал – изменяемый (mutable) или неизменяемый (immutable) объект?
  - Неизменяемый (обладает более понятным поведением)

- Пользователь может получить границы типа `Rational`, но если границы представимы без потерь типом `double`, то может использоваться оптимизированный подкласс класса `RealInterval`

- Режим округления – наружу, внутрь, куда угодно – задается глобальным (внутри потока) контекстом

Класс RealInterval

```
public abstract class RealInterval implements ClassicRealInterval<RealInterval>{

    ...

    public static void enterFastContext() {
        enterContext(Interval.RoundingMode.ANY, Rational.valueOf(0.01));
    }

    public static void enterExactContext() {
        enterContext(Interval.RoundingMode.EXACT, Rational.ZERO);
    }

    public static void enterContext(Interval.RoundingMode roundingMode, Rational relativeAccuracy) {
        if (relativeAccuracy.signum() < 0) {
            throw new IllegalArgumentException("relative accuracy must be positive");
        }
        switch (roundingMode) {
            case ANY:
                if (relativeAccuracy.compareTo(DoubleContextNearestRounding.RELATIVE_ACCURACY) >= 0) {
                    enterContext(new DoubleContextNearestRounding());
                } else if (relativeAccuracy.signum() > 0) {
                    enterContext(new BinaryContext(roundingMode, relativeAccuracy));
                } else {
                    enterContext(new RationalContext());
                }
                break;
            case OUTWARDS:

    ...
```

Простой пример 1

```
package jintervaltutorial;

import com.kenai.jinterval.rational_bounds.RealInterval;

public class JIntervalTutorial {

    public static void main(String[] args) {
        System.out.println("test "+RealInterval.currentRoundingMode() +
                           " "+RealInterval.currentRelativeAccuracy());
        RealInterval x = RealInterval.valueOf(1.0, 2.0);
        RealInterval y = RealInterval.valueOf(3.0, 4.0);
        RealInterval s = x.add(y);
        RealInterval q = x.divide(y);
        System.out.println("x="+x);
        System.out.println("y="+y);
        System.out.println("x+y="+s);
        System.out.println("x/y="+q);
    }
}
```

```
test OUTWARDS +0x3p-53
x=[1.0,2.0]
y=[3.0,4.0]
x+y=[3.9999999999999996,6.000000000000001]
x/y=[0.24999999999999997,0.6666666666666667]
```

Простой пример 2

```
package jintervaltutorial;

import com.kenai.jinterval.rational_bounds.RealInterval;

public class JIntervalTutorial {

    public static void main(String[] args) {
        // Перейти к быстрым вычислениям
        RealInterval.enterFastContext();
        System.out.println("test "+RealInterval.currentRoundingMode() +
                           " "+RealInterval.currentRelativeAccuracy());
        RealInterval x = RealInterval.valueOf(1.0, 2.0);
        RealInterval y = RealInterval.valueOf(3.0, 4.0);
        RealInterval s = x.add(y);
        RealInterval q = x.divide(y);
        System.out.println("x="+x);
        System.out.println("y="+y);
        System.out.println("x+y="+s);
        System.out.println("x/y="+q);
    }
}
```

```
test ANY +0x1p-53
x=[1.0,2.0]
y=[3.0,4.0]
x+y=[4.0,6.0]
x/y=[0.25,0.6666666666666666]
```

Простой пример 3

```
package jintervaltutorial;

import com.kenai.jinterval.rational_bounds.RealInterval;

public class JIntervalTutorial {

    public static void main(String[] args) {
        RealInterval.enterExactContext();
        System.out.println("test "+RealInterval.currentRoundingMode() +
                           " "+RealInterval.currentRelativeAccuracy());
        RealInterval x = RealInterval.valueOf(1.0, 2.0);
        RealInterval y = RealInterval.valueOf(3.0, 4.0);
        RealInterval s = x.add(y);
        RealInterval q = x.divide(y);
        System.out.println("x="+x);
        System.out.println("y="+y);
        System.out.println("x+y=["+s.exactInf()+","+s.exactSup()+"]="+s);
        System.out.println("x/y=["+q.exactInf()+","+q.exactSup()+"]="+q);
    }
}
```

```
test EXACT +0.0
x=[1.0,2.0]
y=[3.0,4.0]
x+y=[+0x1p2,+0x3p1]=[4.0,6.0]
x/y=[+0x1p-2,+0x1/0x3*2^1]=[0.25,0.6666666666666667]
```

Простой пример 4

```
package jintervaltutorial;

import com.kenai.jinterval.rational_bounds.RealInterval;
import com.kenai.jinterval.number.Rational;
import com.kenai.jinterval.Interval.RoundingMode;

public class JIntervalTutorial {

    public static void main(String[] args) {
        RealInterval.enterContext(RoundingMode.OUTWARDS, Rational.pow2(-100+1));
        System.out.println("test "+RealInterval.currentRoundingMode() +
                           " "+RealInterval.currentRelativeAccuracy());
        RealInterval x = RealInterval.valueOf(1.0, 2.0);
        RealInterval y = RealInterval.valueOf(3.0, 4.0);
        RealInterval s = x.add(y);
        RealInterval q = x.divide(y);
        System.out.println("x="+x);
        System.out.println("y="+y);
        System.out.println("x+y=["+s.exactInf()+","+s.exactSup()+"]="+s);
        System.out.println("x/y=["+q.exactInf()+","+q.exactSup()+"]="+q);
    }
}
```

```
test OUTWARDS +0x1p-99
x=[1.0,2.0]
y=[3.0,4.0]
x+y=[+0x1p2,+0x3p1]=[4.0,6.0]
x/y=[+0x1p-2,+0xaaaaaaaaaaaaaaaaaaaaaaabp-100]=[0.25,0.6666666666666667]
```

Scala-интерфейс к JInterval

- Синтаксис языка Java не склонен к лаконичности и выразительности при записи математических выражений
- Программы для JVM можно писать на других языках, получая при этом прозрачный доступ к Java-библиотекам
- Для JInterval начата разработка программного интерфейса из языка Scala

Java

```
r = x.add(y.multiply(z));
```

Scala

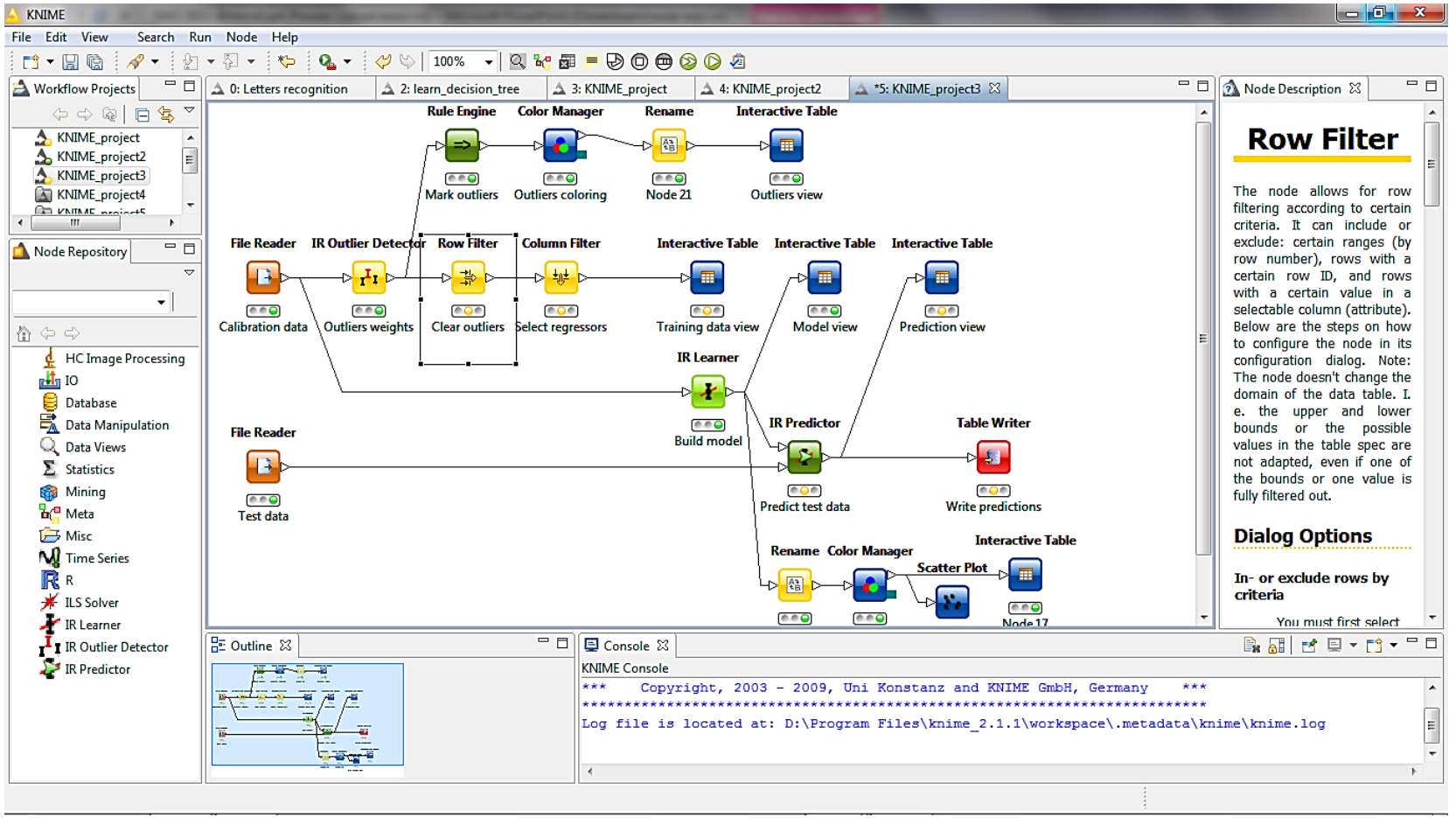
```
r = x + y*z
```

Архитектура библиотеки JInterval

ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ JInterval

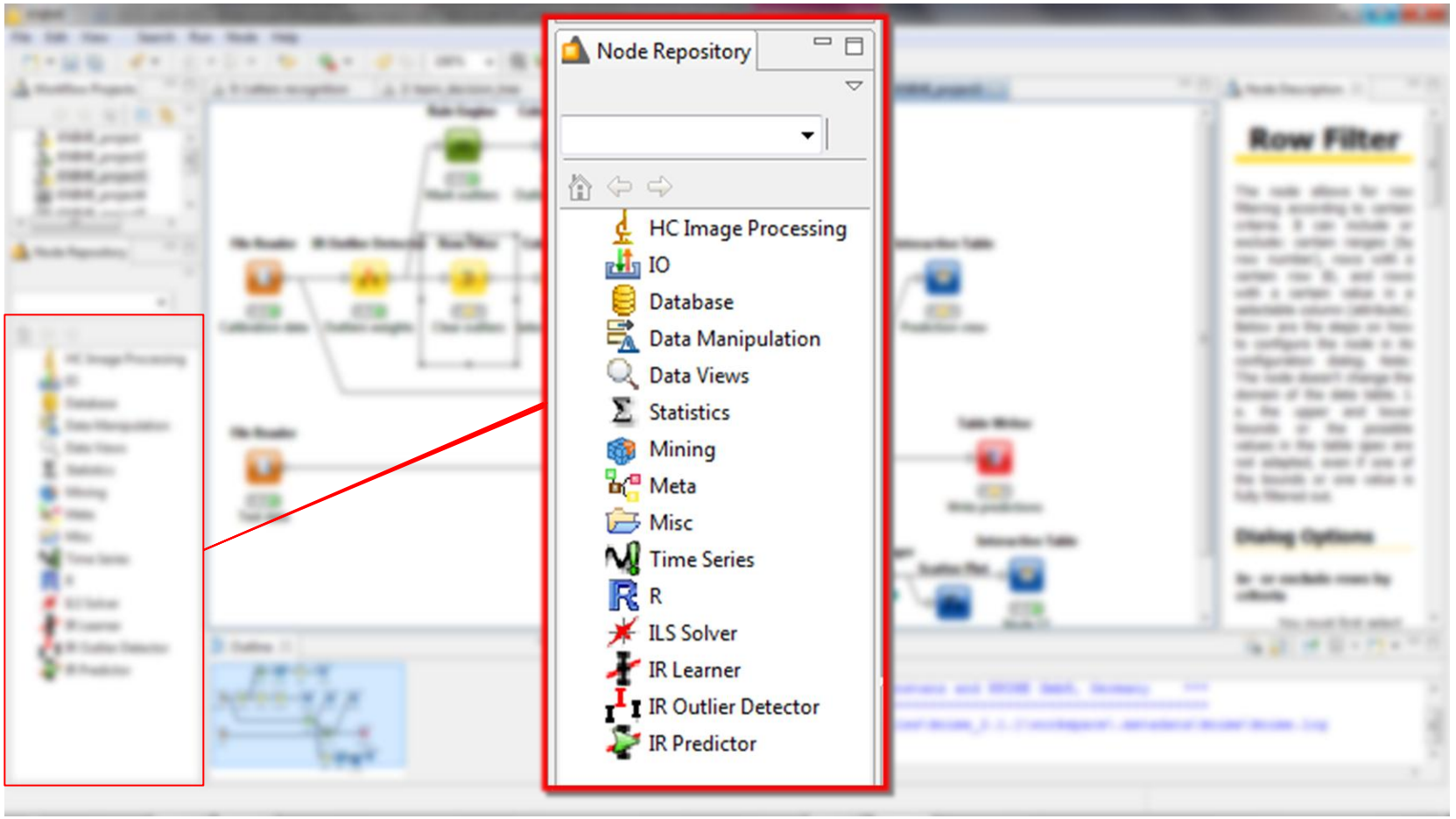
KNIME

- KNIME (Konstanz Information Miner) — модульная платформа для анализа данных с открытыми исходными кодами



KNIME

- KNIME (Konstanz Information Miner) — модульная платформа для анализа данных с открытыми исходными кодами



Узлы KNIME для интервального анализа

- Набор узлов для построения и анализа интервальной регрессии

IR Learner

Строит модель интервальной регрессии $\boldsymbol{Y} = F(X, A)$

IR Predictor

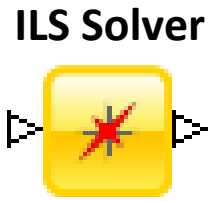
Строит интервальный прогноз $\boldsymbol{Y}^*$ для X^*, используя модель $\boldsymbol{Y} = F(X, A)$

IR Outlier Detector

Выявляет выбросы, требующие уширения интервала ошибки для достижения совместности с прочими наблюдениями

Узлы KNIME для интервального анализа

- Узел-решатель интервальных систем линейных уравнений



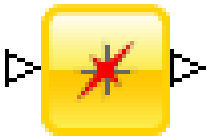
1) Строит внешние и внутренние оценки объединенного и допускового множеств решений ИСЛАУ $\boldsymbol{Ax=b}$

  - Для внешней оценки объединенного множества решений (OMP) использованы
    - Интервальный метод Гаусса
    - Интервальный метод Гаусса-Зейделя
  - Внутренняя оценка множеств решений
    - Алгоритм NonNeg для OMP ИСЛАУ с неотрицательными матрицами
    - Формальный метод для OMP с использованием субдифференциального метода Ньютона
    - Метод Шайдурова для ДМР ИСЛАУ

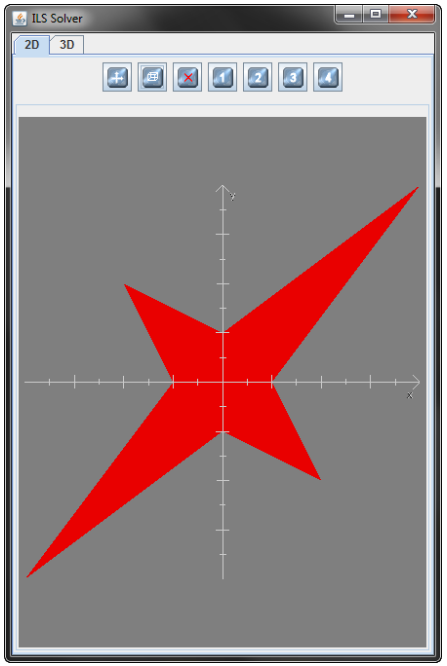
Узлы KNIME для интервального анализа

- Узел-решатель интервальных систем линейных уравнений

ILS Solver

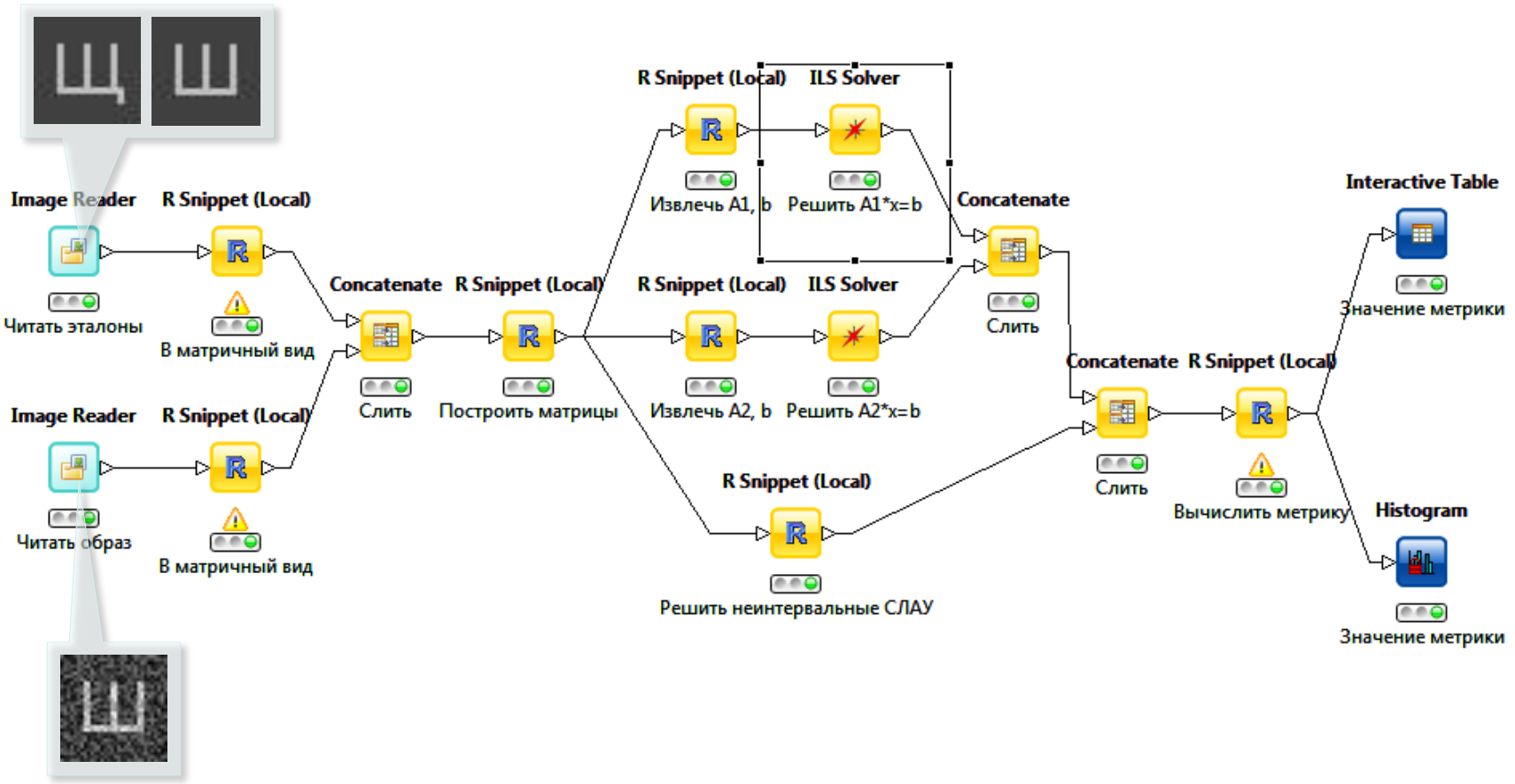


2) Визуализирует* дву- и трехмерные объединенные множества решений ИСЛАУ



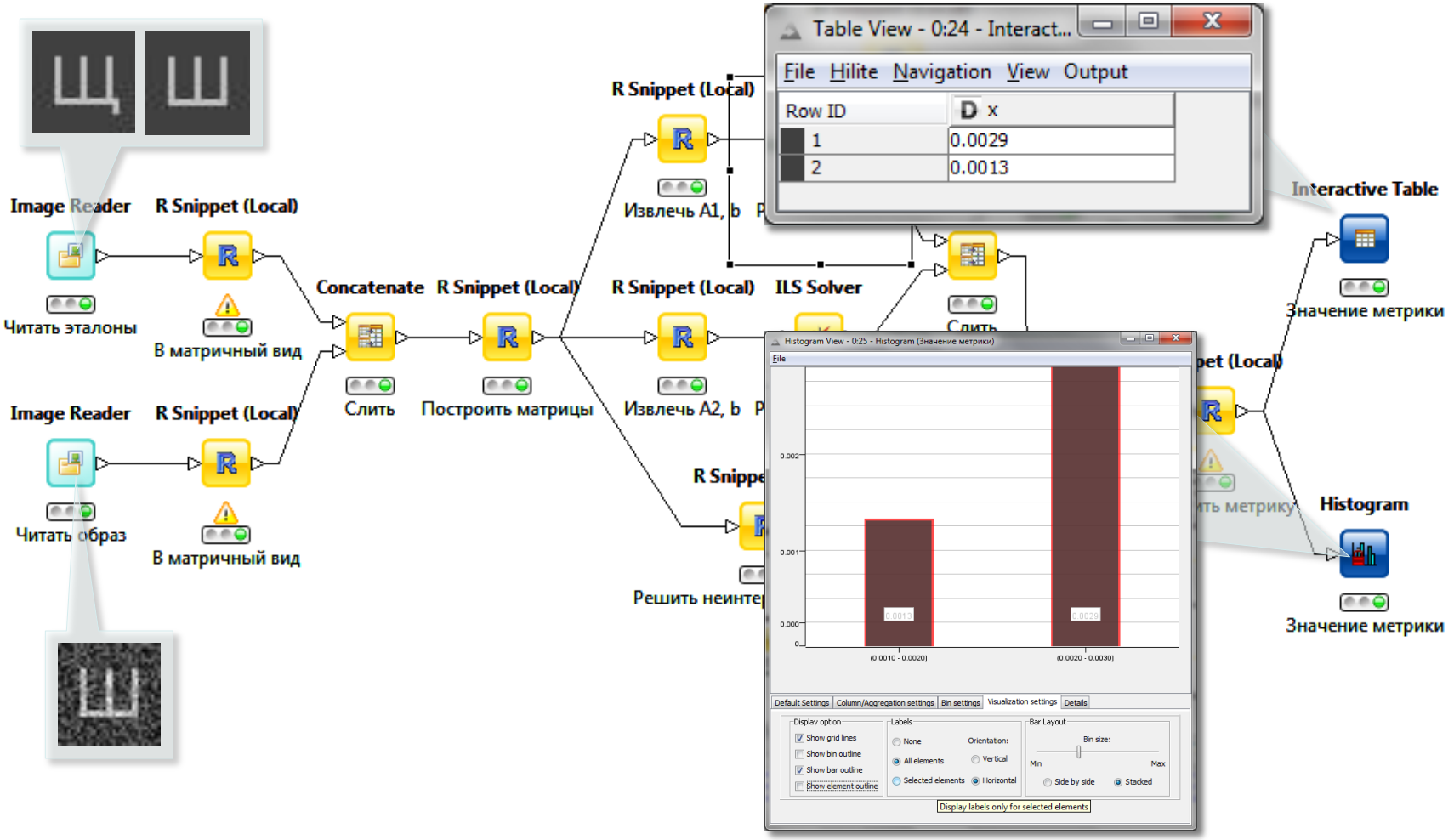
*Kraemer W. *Computing and visualizing solutions sets of interval linear systems*, Serdica J. Computing 1(4) 2007, 455-468.

Сценарий распознавания изображений



Пролубников А.В., Силицкий С.А. О решении задачи распознавания числовых матриц по оценкам множеств решений интервальных линейных систем уравнений // Выч. математика. Труды XIV Байкальской международной школы-семинара «Методы оптимизации и их приложения», Иркутск – Байкал, 2-8 июля 2008 г. Том 3. – Иркутск: ИСЭМ СО РАН, 2008. – С. 152-157.

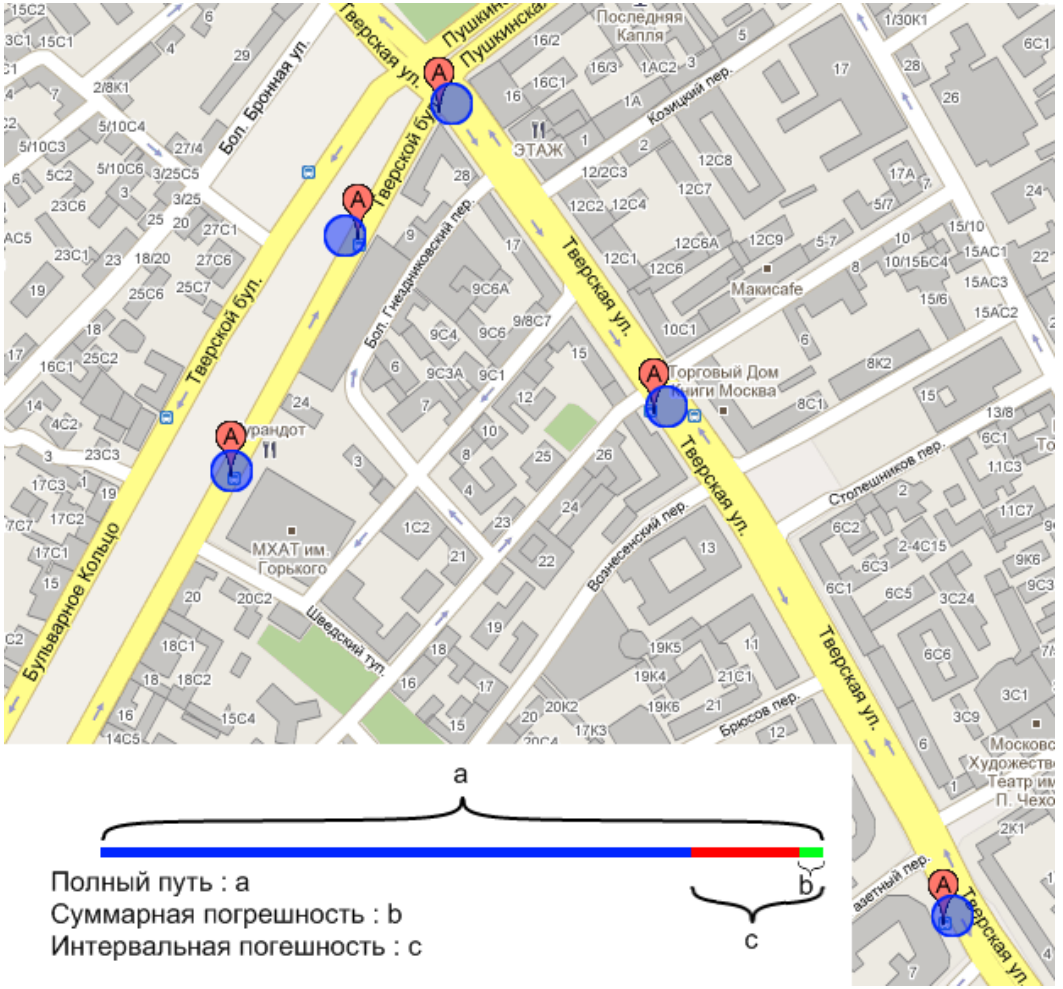
Сценарий распознавания изображений



Пролубников А.В., Силицкий С.А. О решении задачи распознавания числовых матриц по оценкам множеств решений интервальных линейных систем уравнений // Выч. математика. Труды XIV Байкальской международной школы-семинара «Методы оптимизации и их приложения», Иркутск – Байкал, 2-8 июля 2008 г. Том 3. – Иркутск: ИСЭМ СО РАН, 2008. – С. 152-157.

Приложения для мобильных устройств

- Проект «Афиша»: геометрические вычисления с учетом погрешностей (в виде круговых комплексных интервалов)



ЧТО ДАЛЬШЕ?

Направления развития

- Совершенствование JInterval
  - Выработка стабильной версии
  - Разработка документации
- Пополнение функциональности JInterval
  - Обеспечение опциональной возможности подключения через JNI машинно-зависимых реализаций арифметики повышенной точности и интервальных алгоритмов линейной алгебры
  - Разработка интерфейсов к библиотеке из более лаконичных и выразительных, чем Java, языков
  - Пополнение библиотеки реализациями высокоуровневых методов интервального анализа
- Разработка программных продуктов на основе JInterval
  - Облачная система распределенного анализа данных
  - ???

Где взять JInterval?

kenai.com/projects/jinterval

- Исходные коды (SVN)
- JavaDoc
- Wiki
- Руководство для начинающих
- Форум разработчиков
- Списки рассылки


РЕСТОРАН
Журавлина
Украинская кухня
Спасибо за внимание!