ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 8
August
2014

Published since September 2010

CONTENTS

**Information about the jornal is available online at:
http://novtex.ru/pi.html, e-mail:prin@novtex.ru**

УДК 004.4'42

А. В. Пилипенко, аспирант, СПбГУ, e-mail: artur.pilipenko@gmail.com,
О. А. Плисс, канд. физ.-мат. наук, Principal Member of Technical Staff, Oralce, Санта-Клара, США, e-mail: oleg.pliss@gmail.com

Анализ достижимости методов при выборочной инициализации классов в программах на языке Java

Рассмотрена задача анализа достижимости методов при создании загрузочного образа инициализированного состояния виртуальной машины. Подготовка образа включает в себя инициализацию классов программы. Объекты, созданные при инициализации классов, могут влиять на достижимость методов. Такие зависимости не учитываются стандартными алгоритмами анализа. Предложен алгоритм анализа достижимости методов, осуществляющий выборочную инициализацию используемых классов и учитывающий влияние объектов, созданных при инициализации, на достижимость методов.

Ключевые слова: *Java, виртуальные машины, инициализация классов, анализ достижимости методов, косвенные вызовы*

Введение

Каждый раз при старте и инициализации виртуальной машины выполняются одни и те же действия: создаются системные объекты, загружаются классы стандартных библиотек, разрешаются символические ссылки, инициализируются загруженные классы. Этот процесс можно оптимизировать, если сохранить состояние инициализированной виртуальной машины в виде загрузочного образа. Тогда при старте достаточно будет восстановить состояние из сохраненного образа. Подготовка образа осуществляется специальной версией виртуальной машины. При этом помимо стандартной инициализации могут осуществляться различные оптимизации, например, удаление неиспользуемых методов, полей и классов. Такой подход часто используется для оптимизации виртуальных машин, предназначенных для работы на устройствах с ограниченными ресурсами [1—3].

В случае открытой модели, когда виртуальной машиной может быть загружено и исполнено произвольное приложение, загрузочный образ используется для оптимизации системных классов. В случае закрытой модели приложения, исполняемые виртуальной машиной, определены заранее. В этом варианте образ содержит не только системные классы, но и классы приложений.

Инициализация классов при подготовке образа не только уменьшает время инициализации приложения на этапе исполнения, но и во многих случаях позволяет уменьшить размер образа. Инициализация класса в Java подразумевает исполнение статического инициализатора — специального метода с именем `<clinit>`. Этот метод не может быть вызван стандартными средствами, а значит может быть удален у инициализированного класса.

Удаление неиспользуемых методов уменьшает размер образа. Эта оптимизация особенно эффективна при закрытой модели, так как весь код, исполняемый виртуальной машиной, доступен для статического анализа. Классический подход к удалению недостижимого кода состоит в построении транзитивного замыкания методов, достижимых из точек входа программы, и удаления методов, не попавших в замыкание. При таком подходе возникает вопрос: в какой момент инициализировать классы? До построения замыкания нет информации о том, какие классы используются достижимым кодом. Инициализация классов и удаление статических инициализаторов после построения замыкания может сделать некоторые методы недостижимыми.

Объекты, созданные при инициализации классов, могут быть использованы для виртуальных и интерфейсных вызовов. Классические алгоритмы построения транзитивного замыкания не учитывают влияние таких объектов на достижимость методов. В статье предложен алгоритм анализа достижимости методов Java-программ, осуществляющий выборочную инициализацию используемых классов и учитывающий зависимости между методами и объектами, созданными при инициализации классов.

В разд. 1 описаны классические алгоритмы построения транзитивного замыкания достижимых методов. В разд. 2 описана инициализация классов при подготовке образа. Разд. 3 посвящен описанию предложенного алгоритма. В разд. 4 приведены результаты экспериментов для оценки эффективности алгоритма. В разд. 5 представлен обзор существующих работ по данной теме.

1. Транзитивное замыкание графа вызовов

Если метод *f* может быть вызван в результате выполнения метода *g*, метод *f* считается достижимым из метода *g*. Для нахождения используемых методов строится транзитивное замыкание методов, достижимых из точек входа программы. В случае открытой модели точками входа считаются все методы, доступные для приложений. При закрытой модели точками входа являются методы *main*.

Для нахождения методов, достижимых из заданного, анализируются инструкции вызова в коде метода. Инструкции вызова JVM байт-кода можно разделить на две следующие группы[1].

- Инструкции прямого вызова (`invokestatic`, `invokespecial`). Аргумент такой инструкции однозначно определяет вызываемый метод.
- Инструкции косвенного вызова (`invokevirtual`, `invokeinterface`). Метод, вызываемый такой инструкцией, определяется на этапе исполнения. Аргумент инструкции статически описывает вызываемый метод. Для динамического связывания используется первый операнд на стеке — объект-получатель. При выполнении инструкции вызывается реализация метода-аргумента в классе данного объекта.

При статическом анализе информация об объектах времени исполнения отсутствует. Для анализа достижимости используется консервативное приближение множества классов объектов-получателей. Существует несколько подходов к вычислению множества классов объектов-получателей.

Статическим классом вызова будем называть класс, в котором определен метод-аргумент косвенного вызова. В самом простом случае можно считать, что для косвенного вызова может быть использован экземпляр статического класса вызова или любого его подкласса, т. е. множество возможных классов — это множество всех подклассов статического класса вызова. Здесь и далее будем считать, что сам класс принадлежит множеству его подклассов. Описанный подход называется *Class Hierarchy Analysis* (CHA) [4].

В большинстве случаев алгоритм CHA достаточно эффективен, за исключением случая универсальных библиотек. Универсальные библиотеки часто содержат большое число различных реализаций базовых интерфейсов или классов. Обычно приложением используется небольшая часть этих реализаций, но для большей гибкости при работе с объектами используют интерфейсы базовых типов.

Примером может служить библиотека стандартных коллекций Java. Пакет java.util содержит десять классов, реализующих интерфейс List. Для следующего кода достижимыми будут считаться реализации метода add всех десяти классов:

```
List<String> list = new ArrayList<String>();
list.add("one");
```

Точность анализа можно повысить, если учесть, что нестатический метод класса может быть вызван только тогда, когда существуют или могут быть созданы объекты данного класса или его подклассов. Уточненный алгоритм называется *Rapid Type Analysis* (RTA) [5]. В процессе работы алгоритма вычисляется множество косвенных вызовов и множество классов, экземпляры которых могут быть использованы для косвенных вызовов. Обозначим эти множества indirect_invocations и instantiable_classes соответственно. Итеративный алгоритм начинается с добавления в замыкание точек входа приложения. Алгоритм завершается по достижению неподвижной точки. На каждой итерации выполняются следующие действия.

1. Для каждого ранее не обработанного метода из замыкания просматривается его код:
   - методы, вызываемые инструкциями прямого вызова, добавляются в замыкание;
   - косвенные вызовы добавляются в множество indirect_invocations;
   - классы, используемые для создания объектов с помощью инструкции `new`, добавляются в множество instantiable_classes.
2. Для каждого вызова из множества indirect_invocations вычисляется пересечение множества подклассов статического класса вызова с множеством instantiable_classes. Полученное пересечение — это классы, методы которых могут быть вызваны. Соответствующие методы добавляются в замыкание.

При построении замыкания методы `<clinit>` и `finalize` требуют специальной обработки. Метод `<clinit>` выполняется при инициализации класса и не может быть вызван стандартными инструкциями. Метод `finalize` определен в классе java.lang.Object и может быть переопределен в других классах. Финализатором объекта называется реализация метода `finalize` в классе объекта. Финализатор выполняется при уничтожении объекта сборщиком мусора. Кроме того, он может быть вызван явно.

2. Инициализация классов

В языке Java применяется ленивая инициализация классов — класс инициализируется при первом обращении, либо при инициализации одного из его подклассов. Обращением к классу считается:

- выполнение одной из инструкций `new`, `invokestatic`, `getstatic`, `putstatic`;
- рефлексивный доступ к классу, например, с помощью метода `Class.forName`.

В процессе инициализации вызывается статический инициализатор класса — метод `<clinit>` [6]. Так как данный метод недоступен для вызова стандартными

[1] Набор инструкций в Java 7 был расширен инструкцией вызова `invokedynamic` для поддержки динамических языков программирования. Для анализа нового механизма диспетчеризации описываемый алгоритм нуждается в дополнении.

инструкциями, после инициализации класса он становится недостижим.

Ранняя инициализация классов при создании образа ускоряет инициализацию приложения на этапе исполнения. Инициализация классов уменьшает набор достижимых методов, но в то же время может увеличить число объектов в образе за счет объектов, созданных инициализаторами. Для большинства классов вклад созданных объектов в размер образа оказывается меньше, чем сокращение размера за счет уменьшения числа достижимых методов. Таким образом, в большинстве случаев ранняя инициализация классов сокращает размер образа.

Не все классы можно инициализировать при создании образа. Результат работы инициализатора может иметь побочные эффекты, например, инициализатор может осуществлять операции ввода-вывода. Результат работы инициализатора может зависеть от того, на какой виртуальной машине он вызван. Например, инициализатор может использовать метод `System.getProperty()`, результаты работы которого отличаются на этапе подготовки образа и на этапе исполнения. В общем случае ранняя инициализация классов нарушает семантику языка и может изменить видимое поведение программы.

Класс не будет инициализирован при создании образа, если в его инициализаторе присутствует хотя бы одна из следующих операций.

- *Цикл.* Инициализатор с циклами может никогда не завершиться.
- *Вызов методов.* В частности, запрещен вызов конструкторов объектов. Запрет вызова методов позволяет не анализировать код вызываемых методов.
- *Запись в статическое поле класса, за исключением данного класса и его надклассов.* Эта операция может также привести к нежелательной инициализации класса.

Данная эвристика может оказаться слишком консервативной. Запрет на инициализацию класса можно отменить при помощи аннотации @InitAtBuild.

Стоит отметить, что при выполнении инициализатора может быть брошено исключение. При ленивой инициализации классов это исключение может быть обработано кодом приложения. При создании образа такое исключение нельзя обработать, поэтому при возникновении исключительной ситуации создание образа завершается ошибкой. Таким образом, для некоторых корректных с точки зрения языка приложений нельзя создать загрузочный образ.

Наиболее эффективно инициализировать классы в процессе построения замыкания. До построения замыкания невозможно определить, какие классы могут быть инициализированы достижимым кодом. Инициализация лишних классов может создать в памяти достижимые, но не используемые объекты. Инициализация классов после построения замыкания менее эффективна, так как после удаления инициализаторов некоторые ранее достижимые методы могут утратить это свойство.

Каждый потенциально инициализируемый достижимым кодом класс можно либо инициализировать на этапе построения замыкания, либо отложить его инициализацию до этапа исполнения. В первом случае метод `<clinit>` должен быть исполнен, во втором случае `<clinit>` и все достижимые из него методы должны быть добавлены в замыкание.

В результате выполнения инициализаторов могут быть созданы объекты, некоторые из которых останутся доступными приложению. Будем считать, что приложению доступны все объекты, пережившие сборку мусора после инициализации классов.

3. Алгоритм

Дополним алгоритм RTA инициализацией классов. В процессе работы алгоритма будем вычислять следующие множества:

- reachable_methods — множество достижимых методов;
- analyzed_reachable_methods — множество достижимых методов, код которых был проанализирован;
- indirect_invocations — множество косвенных вызовов;
- instantiable_classes — множество классов, экземпляры которых доступны для косвенных вызовов;
- initializable_classes — множество классов, потенциально инициализируемых достижимым кодом;
- processed_initializable_classes — множество потенциально инициализируемых достижимым кодом классов, инициализаторы которых выполнены или добавлены в замыкание.

Итеративный алгоритм начинает работу с добавления точек входа приложения в множество reachable_methods. Алгоритм завершается по достижению неподвижной точки. На каждой итерации выполняются следующие шаги.

Шаг 1. Для каждого метода из разности множеств reachable_methods и analyzed_reachable_methods просматривается его код:

- методы, вызываемые инструкциями прямого вызова, добавляются в множество reachable_methods;
- косвенные вызовы запоминаются в множестве indirect_invocations;
- классы, используемые для создания объектов с помощью инструкции `new`, запоминаются в множестве instantiable_classes;
- классы, на которые ссылаются инструкции `new`, `invokestatic`, `getstatic`, `putstatic`, добавляются в множество initializable_classes;

Проанализированные методы добавляются в множество analyzed_reachable_methods.

Шаг 2. В множество reachable_methods добавляются методы классов из множества instantiable_classes, доступные через косвенные вызовы.

Шаг 3. В множество reachable_methods добавляются финализаторы классов из instantiable_classes.

Шаг 4. Для каждого класса из разности множеств initializable_classes и processed_initializable_classes его ини-

циализатор добавляется в множество reachable_methods или выполняется. Обработанные классы добавляются в множество processed_initializable_classes.

Шаг 5. Выполняется сборка мусора, финализируются удаленные объекты. Классы объектов, переживших сборку, добавляются в множество instantiable_classes.

Заметим, что описанный алгоритм извлекает статические зависимости из байт-кода программы. Однако не все зависимости можно проанализировать таким образом. Зависимости, возникающие при доступе к сущностям языка Java из нативного кода, не отражены в байт-коде. Использование рефлексивного доступа порождает динамические зависимости, которые невозможно проанализировать на этапе построения образа. Для корректной обработки таких ситуаций реализация предложенного алгоритма позволяет вручную описывать дополнительные зависимости методов.

4. Анализ эффективности

Для анализа эффективности предложенного алгоритма был измерен размер образа нескольких конфигураций при различных оптимизациях. Для измерений были использованы следующие конфигурации:

- CLDC — реализация стандарта Connected Limited Device Configuration 8 (JSR360)[2] в открытой модели;
- MEEP — реализация стандарта Java ME Embedded Profile (JSR361) в открытой модели;
- CLDC + Hello World — минимальное приложение для платформы CLDC в закрытой модели;
- CLDC + EEMBC — набор тестов EEMBC GrinderBench в закрытой модели.

В табл. 1 приведена информация о размере класс-файлов используемых конфигураций.

Таблица 1

Размер класс-файлов используемых конфигураций

Конфигурация	Размер класс-файлов, байт
CLDC	531 120
MEEP	2 275 847
CLDC + Hello World	531 461
CLDC + EEMBC	873 504

Для каждой конфигурации был измерен размер образа при использовании следующих оптимизаций:

- none — без инициализации классов и удаления методов;
- init — с инициализацией классов, без удаления методов (для выбора инициализируемых классов использовался критерий, описанный в разд. 2);
- method — без инициализации классов, с удалением недостижимых методов (для анализа достижимости методов использовался алгоритм RTA);
- init + method — с инициализацией классов и удалением недостижимых методов (для анализа до-

Таблица 2

Размер образа при различных оптимизациях, байт

Конфигурация	Вид оптимизации			
	none	init	method	init + method
CLDC	314 739	314 171	306 599	305 759
MEEP	1 172 111	1 168 383	1 105 839	1 099 899
CLDC + Hello World	263 472	262 828	49 412	48 660
CLDC + EEMBC	435 920	435 276	230 792	229 508

Таблица 3

Число классов

Конфигурация	Всего классов	Классы с непустым статическим инициализатором	Классы, потенциально инициализируемые достижимым кодом	Классы, инициализированные предложенным алгоритмом
CLDC	373	46	365	354
MEEP	1275	186	1097	988
CLDC + Hello World	374	46	51	51
CLDC + EEMBC	496	63	208	205

[2] JSR — Java Specification Request, спецификация стандарта Java-платформы. Упомянутые спецификации доступны на сайте Java Community Process URL: http://www.jcp.org.

стижимости и инициализации классов использовался предложенный алгоритм).

Результаты измерений представлены в табл. 2. Использование предложенного алгоритма в случае открытой модели сокращает размер образа на 2,7...6,2 %. В случае закрытой модели эффективность оптимизаций сильно зависит от приложения. Для измеренных приложений уменьшение размера образа составляет 47,4...81,5 %.

В большей степени размер образа уменьшается за счет удаления недостижимых методов, вклад инициализации классов в уменьшение размера незначителен. Это объясняется тем, что лишь небольшая часть инициализированных классов содержит непустой статический инициализатор. Информация о числе инициализированных классов приведена в табл. 3.

5. Обзор смежных работ

Помимо алгоритмов CHA и RTA, описанных в разд. 1, существуют и другие алгоритмы построения замыкания графа вызовов. Для оценки возможных типов объектов в любой точке программы алгоритм RTA использует одно множество instantiable_classes. Более точные алгоритмы строят граф потока данных программы и для каждого узла в графе вычисляют множество достигающих типов. Алгоритмы отличаются точностью моделирования потока данных. Например, алгоритм XTA, описанный в работе [7], моделирует поток данных между методами программы. В языке Java объекты передаются между методами через аргументы, возвращаемые значения и ссылки в куче (ссылки могут содержаться в полях объектов и элементах массивов). Алгоритм не учитывает порядок выполнения программы и состояние локальных переменных.

Примером более точного алгоритма является алгоритм *Variable-Type Analysis* (VTA) [8], который моделирует поток данных между переменными. Алгоритм анализирует операции присваивания между локальными переменными, полями объектов и элементами массивов. Разные экземпляры одной переменной моделируются с помощью одного узла в графе. При анализе не учитывается порядок выполнения программы. В работе [8] также предложена вариация алгоритма под названием *Declared-Type Analysis* (DTA), которая моделирует разные переменные одного типа с помощью одного узла в графе.

В работе [9] рассмотрены алгоритмы, которые учитывают контекст исполнения при моделировании потока данных между переменными. При этом для каждой переменной в графе потока данных может существовать несколько узлов, соответствующих разным контекстам исполнения. К таким алгоритмам относятся *k-Control Flow Analysis* (k-CFA), *Cartesian Product Algorithm* (CPA), *Simple Class Set* (SCS).

Более точные алгоритмы незначительно уменьшают число достижимых методов, но существенно уменьшают число ребер в графе вызовов. Сокращение числа ребер в графе позволяет более точно идентифицировать мономорфные вызовы, которые впоследствии могут быть оптимизированы. Все описанные выше алгоритмы могут быть дополнены инициализацией классов.

Раздельная инициализация часто используется для оптимизации программ для встроенных систем. Например, в языке Virgil [2] инициализация программы осуществляется на этапе компиляции. Язык не поддерживает динамического создания объектов, все используемые программой объекты должны быть созданы во время инициализации. При инициализации исполняются конструкторы компонентов, которые могут содержать произвольный код. Инициализация осуществляется до анализа достижимости методов. Для анализа достижимости методов, полей и объектов используется алгоритм *Reachable Member Analysis* (RMA). Данный алгоритм расширяет алгоритм RTA анализом достижимости полей и объектов. Алгоритм анализирует чтения полей и считает доступными приложению только те объекты, которые доступны через ссылки в используемых полях.

Другой пример применения алгоритма RMA для анализа Java-программ дан в работе [3], где описана система ExoVM, автоматически специализирующая виртуальную машину для заданного приложения. Специализация осуществляется за счет анализа достижимости не только сущностей языка, но и отдельных компонентов виртуальной машины. Определены отношения достижимости между методами, полями, объектами, классами и компонентами виртуальной машины. В итоговую виртуальную машину включаются только достижимые из точек входа приложения конструкции.

В отличие от предложенного алгоритма, ExoVM инициализирует классы до анализа достижимости методов. Это может приводить к инициализации неиспользуемых классов и объектов. Анализ доступности объектов в алгоритме RMA позволяет удалять такие объекты. Более простой критерий доступности объектов в предложенном алгоритме не позволяет находить такие объекты. Однако выборочная инициализация используемых классов существенно сокращает число не используемых приложением объектов. Кроме того, выборочная инициализация классов в предложенном алгоритме позволяет инициализировать только те классы, инициализация которых не может изменить видимое поведение программы.

В работе [10] развита идея раздельной инициализации для встроенных систем. Авторами предлагается перенести создание и инициализацию потоков приложения с этапа исполнения на этап подготовки загрузочного образа. В статье отмечается, что такой подход не только ускоряет запуск приложения, но и сокращает число достижимых методов.

Заключение

Предложен алгоритм анализа достижимости методов Java-программ, осуществляющий выборочную инициализацию используемых классов. Предложенный алгоритм дополняет алгоритм RTA инициализа-

цией классов. Алгоритм инициализирует только те классы, ранняя инициализация которых не может изменить поведение программы. Для выбора таких классов предложена консервативная эвристика.

В отличие от алгоритма RTA предложенный алгоритм учитывает созданные при инициализации классов объекты. Доступными считаются все объекты, пережившие сборку мусора.

Возможным развитием алгоритма является уточнение критерия доступности объектов с помощью подхода, аналогичного алгоритму RMA.

Список литературы

1. **Aslam F., Fennell L., Schindelhauer C. et. al.** Optimized Java Binary and Virtual Machine for Tiny Motes // Distributed Computing in Sensor Systems (DCOSS). 2010. Vol. 6131. P. 15—30.

2. **Titzer B.** Virgil: objects on the head of a pin // ACM SIGPLAN Notices. 2006. Vol. 41, Is. 10. P. 191—208.

3. **Titzer B., Auerbach J., Bacon D., Palsberg J.** The ExoVM System for Automatic VM and Application Reduction // ACM SIGPLAN Notices. 2007. Vol. 42, Is. 6. P. 352—362.

4. **Dean J., Grove D., Chambers C.** Optimization of Object-Oriented Programs Using Static Class Hierarchy Analysis // Proceedings of the 9th European Conference on Object-Oriented Programming. Aarhus, Denmark. 7—11 August, 1995. Berlin: Springer, 1995. P. 77—101.

5. **Bacon D., Sweeney P.** Fast static analysis of C++ virtual function calls // ACM SIGPLAN Notices. 1996. Vol. 31, Is. 10. P. 324—341.

6. **Lindholm T., Yellin F.** The Java Virtual Machine Specification. SUN Microsystems, 1999.

7. **Tip F., Palsberg J.** Scalable Propagation-Based Call Graph Construction Algorithms // ACM SIGPLAN Notices. 2000. Vol. 35, Is. 10. P. 281—293.

8. **Sundaresan V., Hendren L., Razafimahefa C. et. al.** Practical Virtual Method Call Resolution for Java // ACM SIGPLAN Notices. 2000. Vol. 35, Is. 10. P. 264—280.

9. **Grove D., DeFouw G., Dean J., Chambers C.** Call Graph Construction in Object-Oriented Languages // ACM SIGPLAN Notices. 1997. Vol. 32, Is. 10. P. 108—124.

10. **Courbot A., Grimaud G., Vandewalle J.** Romization: Early Deployment and Customization of Java Systems for Constrained Devices // Proceeding of CASSIS'05. Nice, France. 8—11 March, 2005. Berlin: Springer, 2006. P. 57—76.

A. V. Pilipenko, Postgraduate Student, SPbSU, e-mail: artur.pilipenko@gmail.com,
O. A. Pliss, Principal Member of Technical Staff, Oralce, Santa Clara, USA,
e-mail: oleg.pliss@gmail.com

Method Reachability Analysis and Selective Class Initialization for Java Programs

Method reachability analysis during Java program romization is considered. Eager class initialization is one of the romization optimizations. Class initialization in Java involve execution of class initializers. Execution of class initializers may create objects which affect methods reachability. Standard analysis algorithms don't take these objects into account. We propose method reachability analysis algorithm which selectively initializes classes and takes objects created by class initializers into consideration.

This algorithm is based on Rapid Type Analysis (RTA) algorithm. RTA algorithm keeps track of classes which could be instantiated by reachable methods. Instantiable classes set is used to determine which methods could be invoked by virtual and interface calls. The proposed algorithm also keeps track of classes which could be initialized by reachable methods. Not all of these classes could be initialized at romization stage. Eager initialization of some classes may affect application behavior. Simple euristic is used to choose safe to initialize classses.

All the objects which remain reachable after class initialization considered reachable for the application. These objects could be used for virtual and interface calls. Therefore classes of reachable objects are included to the set of instantiable classes.

Keywords: *Java, virtual machine, romization, class initialization, method reachability analysis, indirect invocations*

References

1. **Aslam F., Fennell L., Schindelhauer C. et. al.** Optimized Java Binary and Virtual Machine for Tiny Motes. *Distributed Computing in Sensor Systems (DCOSS)*. 2010. Vol. 6131. P. 15—30.

2. **Titzer B.** Virgil: objects on the head of a pin. *ACM SIGPLAN Notices*. 2006. Vol. 41, Is. 10. P. 191—208.

3. **Titzer B., Auerbach J., Bacon D., Palsberg J.** The ExoVM System for Automatic VM and Application Reduction. *ACM SIGPLAN Notices*. 2007. Vol. 42, Is. 6. P. 352—362.

4. **Dean J., Grove D., Chambers C.** Optimization of Object-Oriented Programs Using Static Class Hierarchy Analysis. *Proceedings of the 9th European Conference on Object-Oriented Programming*. Aarhus, Denmark. 7—11 August, 1995. Berlin: Springer, 1995. P. 77—101.

5. **Bacon D., Sweeney P.** Fast static analysis of C++ virtual function calls. *ACM SIGPLAN Notices*. 1996. Vol. 31, Is. 10. P. 324—341.

6. **Lindholm T., Yellin F.** *The Java Virtual Machine Specification*. SUN Microsystems, 1999.

7. **Tip F., Palsberg J.** Scalable Propagation-Based Call Graph Construction Algorithms. *ACM SIGPLAN Notices*. 2000. Vol. 35, Is. 10. P. 281—293.

8. **Sundaresan V., Hendren L., Razafimahefa C. et. al.** Practical Virtual Method Call Resolution for Java. *ACM SIGPLAN Notices*. 2000. Vol. 35, Is. 10. P. 264—280.

9. **Grove D., DeFouw G., Dean J., Chambers C.** Call Graph Construction in Object-Oriented Languages. *ACM SIGPLAN Notices*. 1997. Vol. 32, Is. 10. P. 108—124.

10. **Courbot A., Grimaud G., Vandewalle J.** Romization: Early Deployment and Customization of Java Systems for Constrained Devices. *Proceeding of CASSIS'05*. Nice, France. 8—11 March, 2005. Berlin: Springer, 2006. P. 57—76.

УДК 621.039.58
В. А. Федотов[1, 2], аспирант, руководитель группы системного обслуживания ИВЦ, e-mail: workstudio@yandex.ru,
О. М. Гулина[1], д-р техн. наук, проф., e-mail: olga@iate.obninsk.ru,
[1]Национальный исследовательский ядерный университет "МИФИ" Обнинский институт атомной энергетики,
[2]Приборный завод "Сигнал", г. Обнинск

Разработка системы поддержки принятия решений по прогнозированию ресурса оборудования АЭС в условиях эрозионно-коррозионного износа

Рассмотрены вопросы создания отечественной автоматизированной системы, способной решать задачи сбора информации о режиме эксплуатации и для прогнозирования остаточного (гарантирующего неразрушение в заданном временном интервале) ресурса оборудования АЭС, работающего в условиях эрозионно-коррозионного износа. Представленная система поддержки принятия решений охватывает заданный жизненный цикл критического оборудования и с помощью структурированного интерфейса предоставляет персоналу актуальную базу знаний о состоянии оборудования, что позволяет оптимально планировать профилактические и ремонтные работы.

Ключевые слова: *автоматизированная система, атомная электростанция, база данных, ГОСТ, интерфейс, жизненный цикл, трубопровод, эррозионно-корозионный износ*

Введение

Обеспечение безопасной и надежной работы АЭС в целом и отдельных элементов ее оборудования в процессе эксплуатации является важной инженерной и научной задачей. Оборудование и трубопроводные системы энергоблоков второго контура, изготовленные из углеродистых сталей, подвержены воздействию процесса эрозионно-коррозионного износа (ЭКИ). В результате такого износа повышается вероятность размывов элементов трубопроводов, особенно тех, где действует турбулентный поток среды, что может привести к недопустимым уменьшениям толщины стенок трубопроводов (утонениям) или даже к их разрывам. Повреждения вследствие ЭКИ могут вызвать неплановый останов энергоблока и тем самым существенно повлиять на качество, стоимость и безопасность работы всей станции. В связи с этим обстоятельством остро стоит вопрос предотвращения такого рода аварий. Для этого необходимы эффективная методика мониторинга технического состояния и модель прогнозирования срока службы элементов оборудования и трубопроводов.

Актуальность рассматриваемой тематики обусловлена потребностью отрасли добиваться экономически эффективного результата и приемлемого уровня безопасности АЭС, применяя такие сбалансированные и оптимальные решения, при реализации которых требования безопасности вырабатываются с помощью системного процесса, использующего качественные и количественные результаты детерминистического и вероятностного анализов (риск-информативный подход). Для проведения такого расчета необходимо наличие прослеживаемого и регулярно пополняемого исторического машиночитаемого материала об эксплуатационных характеристиках оборудования и трубопроводов АЭС, находящихся в агрессивной среде и подверженных разрушительным воздействиям процесса ЭКИ. Прогнозирование безаварийного периода эксплуатации с помощью эмпирических моделей расчетов или на основе результатов неразрушающего контроля должно обеспечить "разумную" консервативность расчетов [1].

Постановка задачи

Модели прогнозирования остаточного ресурса в условиях ЭКИ. В исследованиях в области неразрушающего контроля под остаточным рабочим ресурсом понимается время, отсчитываемое от начала проведения работ по определению фактического технического состояния трубопровода (диагностики) до того момента, когда считается, что трубопровод исчерпал свой рабочий ресурс. Опыт решения отмеченной выше задачи в разных странах показывает, что существуют два основных направления определения остаточного ресурса трубопроводов: прогнозирование на основе расчета

скорости ЭКИ на базе эмпирических моделей и прогнозирование на основе обработки данных системы контроля (мониторинга) состояния оборудования.

Первое направление заключается в построении динамических многофакторных зависимостей, учитывающих влияние теплогидравлических, конструкционных и эксплуатационных параметров на интенсивность процесса ЭКИ. Второе направление предполагает оценку скорости ЭКИ по результатам замеров толщины стенки трубопровода в течение определенного промежутка времени [2, 3].

Для построения динамических многофакторных зависимостей необходима нормативно-справочная информация о материале, типе и конструкции элемента оборудования, а также справочные данные, полученные на основе лабораторных исследований и опыта эксплуатации. Во втором случае необходимо обеспечить хранение и автоматизированную обработку данных замеров толщин стенок в течение эксплуатации оборудования.

Очевидно, что эффективность расчетов и в первом, и во втором случаях напрямую зависит от наличия тщательно спланированной и регулярно пополняемой базы данных (БД) [4].

Нормативная база. В целях решения задачи контроля и определения остаточного технического ресурса оборудования проводится актуализация нормативной документации. В 2012 г. в отрасли утверждена последняя редакция руководящего документа (РД) эксплуатирующей организации (ЭО) ВНИИ АЭС (РД ЭО 1.1.2.11.0571—2010) "Нормы допускаемых толщин стенок элементов трубопроводов из углеродистых сталей при эрозионно-коррозионном износе" [5]. В данном документе впервые описывается не только методика расчета допустимой толщины стенок элементов трубопроводов, подверженных ЭКИ, но и устанавливается однозначная связь этих процессов со стадиями жизненного цикла всей атомной станции. Кроме этого, РД определяет также требования к размещению информации в информационной БД станции, среди которых основными можно назвать следующее:

— сведения о марке металла, физико-химических и водно-химических свойствах;

— рабочие параметры (среда, давление, температура, скорость среды, концентрация активных веществ);

— данные эксплуатационного контроля толщин стенок трубопровода (толщинометрии);

— сведения обо всех проводимых работах на элементах трубопровода.

Помимо расширения области применения РД, при определении допустимых толщин стенок элементов трубопроводов (утонений в миллиметрах относительно расчетного номинала), необходимо обеспечить автоматизированную оценку требуемых характеристик также для элементов, отличных от содержащихся в РД типоразмеров и условий их эксплуатации. Такого рода инструментарий необходим при оценке остаточного ресурса.

Теория и практика. На каждой АЭС разработаны и применяются в целом очень похожие методики регламентного контроля толщин стенок элементов трубопроводов. Результаты таких замеров фиксируются на каждой АЭС в специальных журналах, преимущественно на бумажных носителях. В результате сложилась система принятия решений, разорванная в нескольких местах участием в ней человека и несвязанными между собой цифровыми интерфейсами, что приводит к росту издержек в работе персонала. А самое главное, сегодня нет машиночитаемой БД, которая необходима для качественного анализа и прогнозирования ресурса оборудования. Иными словами, нет аккумулированного материала по истории эксплуатации с учетом всех замеров, работ по ремонту и изменений параметров рабочей среды. Как следствие, сложно не только делать прогнозы на короткую перспективу для конкретного оборудования, но и говорить об оптимизации процесса контроля остаточного ресурса.

Разработанный за рубежом программный комплекс (система) CHEC (версия 1.0 выпущена EPRI в июле 1987 г., а в 1989 г. — версия 2.0, коммерческая) предназначался для расчета скорости ЭКИ в трубопроводах с однофазной средой [6]. При его разработке были использованы данные лабораторных исследований, которые проводились в Англии, Франции и США с 1970 г. и в Германии с 1973 г., а также данные эксплуатационного контроля на реальных АЭС. Программный комплекс CHECWORKS был разработан в 1995 г. для того чтобы обеспечить возможность работы с вычислительной техникой в операционной среде Windows. Комплекс CHECWORKS интегрирует и включает функции CHEC, CHEC-T, CHEC-NDE и CHECMATE. Основной задачей комплекса CHECWORKS является расчет скорости ЭКИ и определение длительности эксплуатации трубопроводов на основе эмпирической модели, что позволяет повысить безопасность эксплуатации, оптимизировать периоды ремонта, а также уменьшить необходимое число контрольных замеров при планово-предупредительных работах. Высокая стоимость этих систем, а также необходимость их адаптации к условиям российских АЭС с водо-водяным энергетическим реактором (ВВЭР) и реактором большой мощности канальным (РБМК) выдвигают требования о создании отечественных систем, позволяющих решать аналогичные задачи [7].

Существующие и аттестованные отечественные программные средства успешно решают задачи расчета скорости ЭКИ и прогнозирования износа (ЭКИ-02 для однофазной среды на АЭС с ВВЭР-440 и ЭКИ-03 для двухфазной среды на АЭС с ВВЭР-440, аналогичные программному комплексу CHECWORKS). Однако они не аккумулируют результаты таких расчетов, которые остаются результатами разового использования. Эта ситуация не позволяет персоналу АЭС отказаться от ручного ввода исходных данных и фиксации выходных значений с помощью бумажных и/или

электронных журналов произвольного формата [8]. Кроме упомянутых действий каждый этап занесения и обработки информации выполняется разрозненными программными решениями, что в итоге не дает возможности свести всю информацию об оборудовании и проведенных замерах к единой БД. Наличие же такой БД существенно повысило бы точность и производительность обработки результатов при расчете скорости процесса ЭКИ и определении остаточного ресурса оборудования трубопроводов.

Таким образом, с учетом требований РД ЭО 1.1.2.11.0571—2010 и реальных потребностей отрасли очевидна необходимость создания для технического персонала АЭС качественного аналитического программного решения, которое будет учитывать специфику отечественных типов реакторов и проводить сопровождение всего жизненного цикла оборудования с использованием единой и пополняемой БД [7]. По этой причине в качестве стратегического направления разработки выбран путь создания жестко структурированной автоматизированной системы, отвечающей всем необходимым техническим, системным и эргономическим требованиям к ней. Такая система должна в комплексе решать обозначенные выше задачи в режиме "одного окна".

Основные принципы. При создании и последующем сопровождении такой системы за основу выбраны риск-информативный и процессный подходы. Согласно последнему, главным выделяется цельный и неделимый горизонтальный процесс (жизненный цикл указанного оборудования), протекающий сквозным образом через зоны ответственности структур вертикального уровня (департаменты, отделы, лаборатории и т. д.), ответственные за мониторинг и поддержания остаточного ресурса на проектном уровне.

При этом важно не просто объединить в единое целое различные виды эксплуатационной информации и методик расчетов (рис. 1). Ключевым результатом всех работ в системе должна стать реализация обоснованной методики расчета скорости ЭКИ по накопленным данным замеров и последующее прогнозирование остаточного ресурса оборудования, подверженного ЭКИ. На рис. 2 показана последовательность ввода и обработки информации в системе.

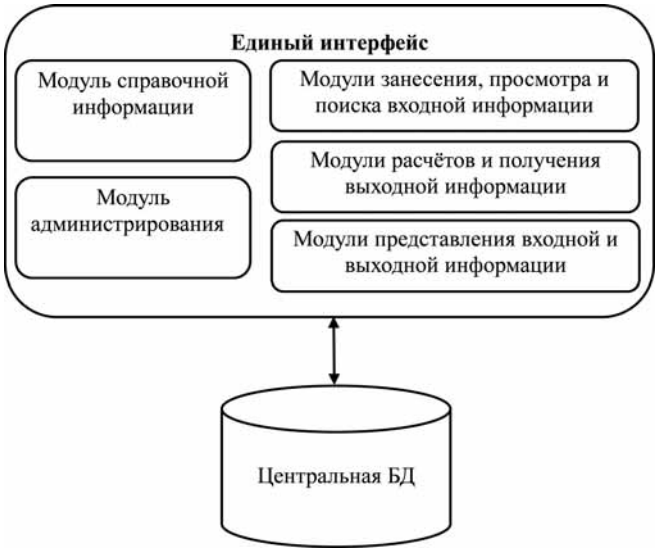

Рис. 1. Структурная схема системы

Рис. 2. Жизненный цикл информации в системе

Основными функциональными требованиями к системе являются: наличие пользовательского интерфейса, осуществляющего ввод и просмотр информации о замерах; предоставление математического аппарата для достоверного расчета скорости ЭКИ и остаточного ресурса оборудования; размещение всего набора обрабатываемой информации в защищенной центральной базе данных. Кроме перечисленных выше требований, как потенциально конкурентоспособный промышленный образец, система должна быть реализована в виде единого, хорошо оформленного и функционально продуманного программного продукта. Этот продукт должен поддерживать богатый набор нормативно-справочной информации, учитывающей максимально возможное число видов объектов и оборудования, а также быть готовым к его государственной сертификации и патентной регистрации, что позволит не только вывести его на IT-рынок, но и обеспечить конкуренцию с зарубежными аналогами.

Реализация

Методика разработки программной системы. Проект технического задания на автоматизированную систему предъявляет требования по организации реляционной структуры таблиц в БД под управлением СУБД типа Oracle или MS SQL Server, а также по разработке многопользовательского оконного интерфейса с возможностью управления информацией на основе прав доступа. Однако движение в этом направлении на этапе исследования могло существенно увеличить сроки проведения работ, так как было бы сложно гарантировать должный уровень достоверности результатов конечного моделирования. Более того, применение СУБД промышленного типа на ранней стадии разработки затруднило бы автономность работы системы при демонстрации предлагаемых решений на локальной рабочей станции (переносном компьютере).

В результате более детальной оценки перечисленных выше рисков была применена методика ускоренной разработки. Такой подход позволил не отказываясь от базовых положений проекта технического задания "вынести" результаты проведенной работы из стадии закрытого исследования на этап реального тестирования полученных промежуточных решений всеми заинтересованными специалистами.

Концептуальная схема БД была развернута с использованием СУБД MS Access, где по правилам реляционной модели были созданы таблицы нормативно-справочных данных (марки сталей, типоразмеры трубопроводов, схемы измерений, сотрудники и т. д.), таблицы мастер-данных об объекте эксплуатации (атомные станции, трубопроводные системы и составляющие их элементы)

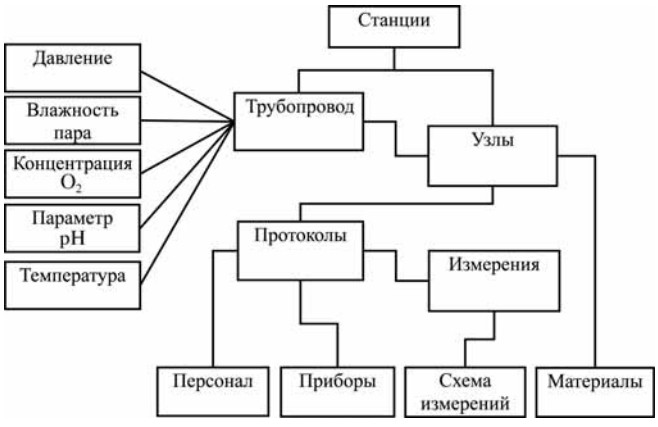

Рис. 3. Обобщенная схема базы данных

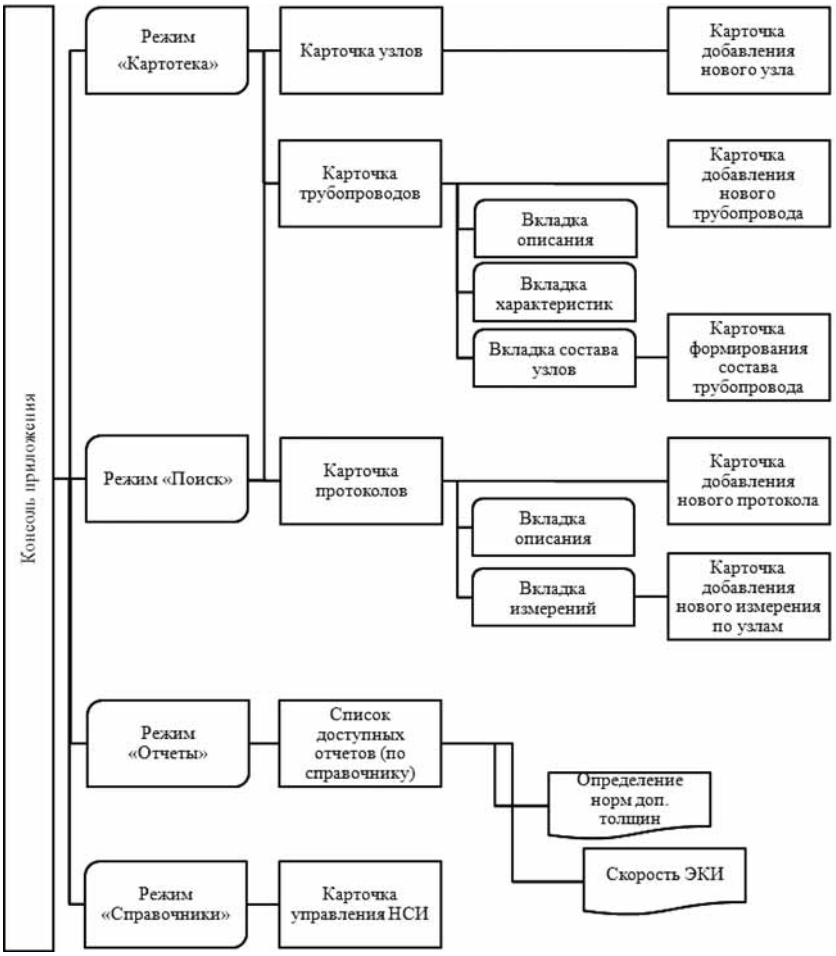

Рис. 4. Структурная схема интерфейса пользователя

и таблицы фиксации результатов контроля толщин стенок элементов трубопроводов (рис. 3).

Тестовая модель пользовательского интерфейса ("толстый" клиент) разработана в среде объектно-ориентированного языка программирования Visual Basic и представляет собой набор форм в составе главного окна — консоли приложения (рис. 4).

Консоль приложения — это основная функциональная оболочка, которая предоставляет возможность просмотра и редактирования информации посредством персональных карточек — "узлов (элементов)", "трубопроводов" и "протоколов замеров". Помимо этого реализован доступ к получению отчетной информации и управлению нормативно-справочной информацией (НСИ) в рамках станции.

При этом в исходном коде интерфейса связь с данными уже сейчас реализована через универсальный механизм, что позволит провести последующий перенос данных в сетевые СУБД (Oracle MS или SQL Server) без перепрограммирования интерфейса. Для успешной инсталляции программы на компьютере пользователя создан автоматизированный пакет в среде конструктора дистрибутивов Install Shield.

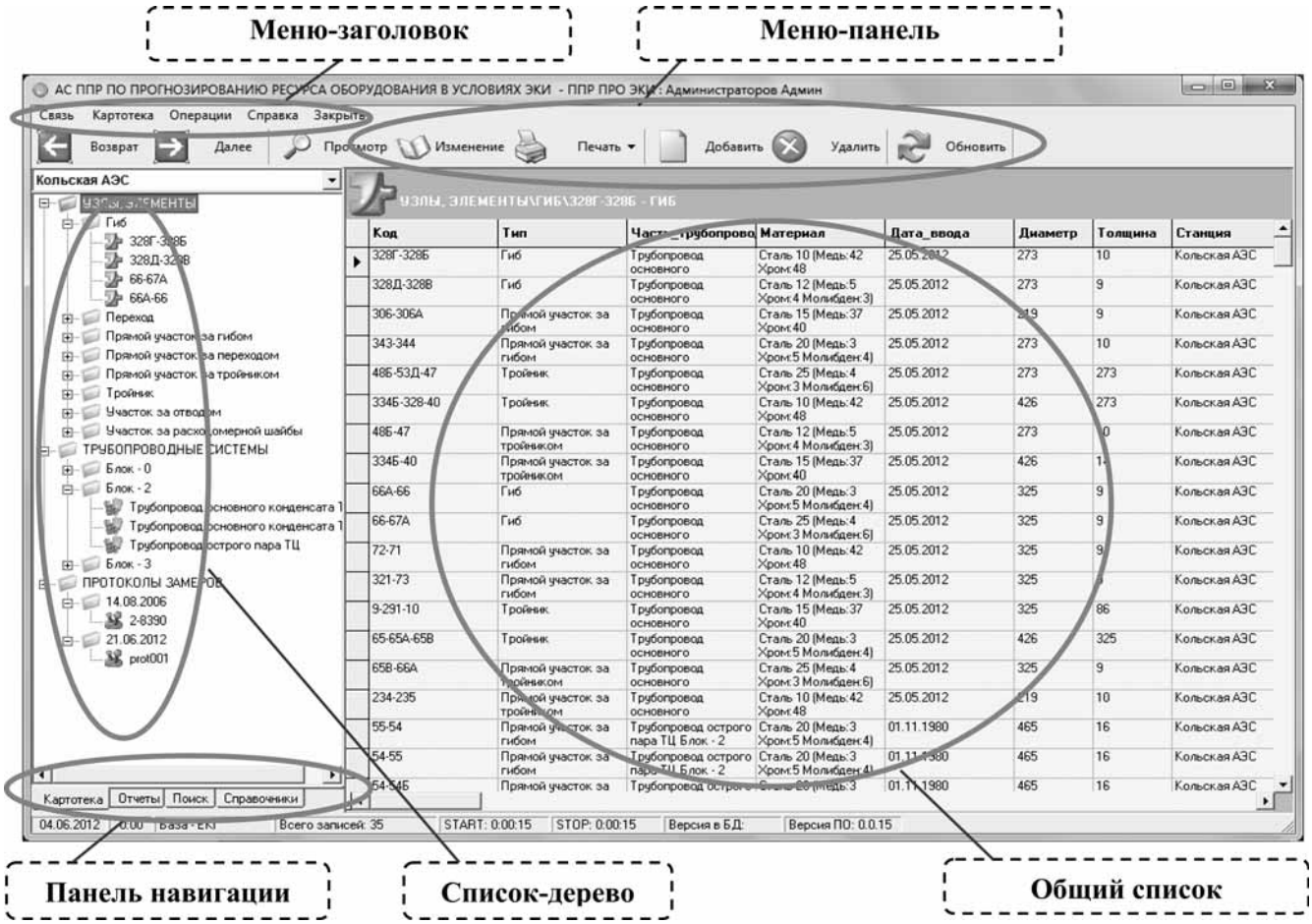

Рис. 5. Главная консоль приложения в режиме "Картотека"

Консоль приложения — это головной модуль приложения. Общий принцип работы всей программы и консоли приложения (рис. 5) построен по иерархическому принципу доступа к информации, где на каждом следующем уровне происходит увеличение детализации данных, что в конце концов приводит к возможности модификации содержимого.

Персональные карточки открывают информацию для детального изучения и/или редактирования в индивидуальной карточке, соответствующей определенной категории информации (рис. 6). Введена классификация и реализованы следующие отдельные карточки:

- для составных узлов (элементов) трубопровода АЭС;
- для трубопроводов АЭС, состоящих из узлов (элементов);
- для протоколов замеров, где заполняются данные эксплуатационного контроля по каждому узлу (элементу).

Сначала в карточку узлов заносится информация об используемых на АЭС узлах (элементах), после чего в карточке трубопроводов из БД выбираются составные узлы для данного трубопровода.

В карточку протоколов вначале вносится информация о конкретном протоколе, затем выбираются доступные в БД узлы (элементы), по которым осуществляется заполнение результатов замеров толщин стенок.

Функция ***формирования электронного протокола*** является необходимой функцией обратной связи системы, предоставляющей возможность формирования и печати электронного документа в виде исходного бумажного варианта протокола замеров.

Функция ***программного определения допустимых толщин трубопроводов*** осуществляет автоматизированный расчет допустимых толщин трубопроводов. При этом в последующих расчетах скорости ЭКИ и остаточного ресурса можно уже не выбирать вручную табличные данные из печатного варианта РД, а проводить оперативный расчет допустимых толщин по реальным теплогидравлическим и режимным параметрам, прочностным свойствам материалов и конструктивным особенностям рассматриваемого узла, уже имеющимся в БД системы.

Методология определения допустимой толщины стенки трубопровода базируется на РД ЭО 1.1.2.11.0571—2010. Для каждого типа элементов реа-

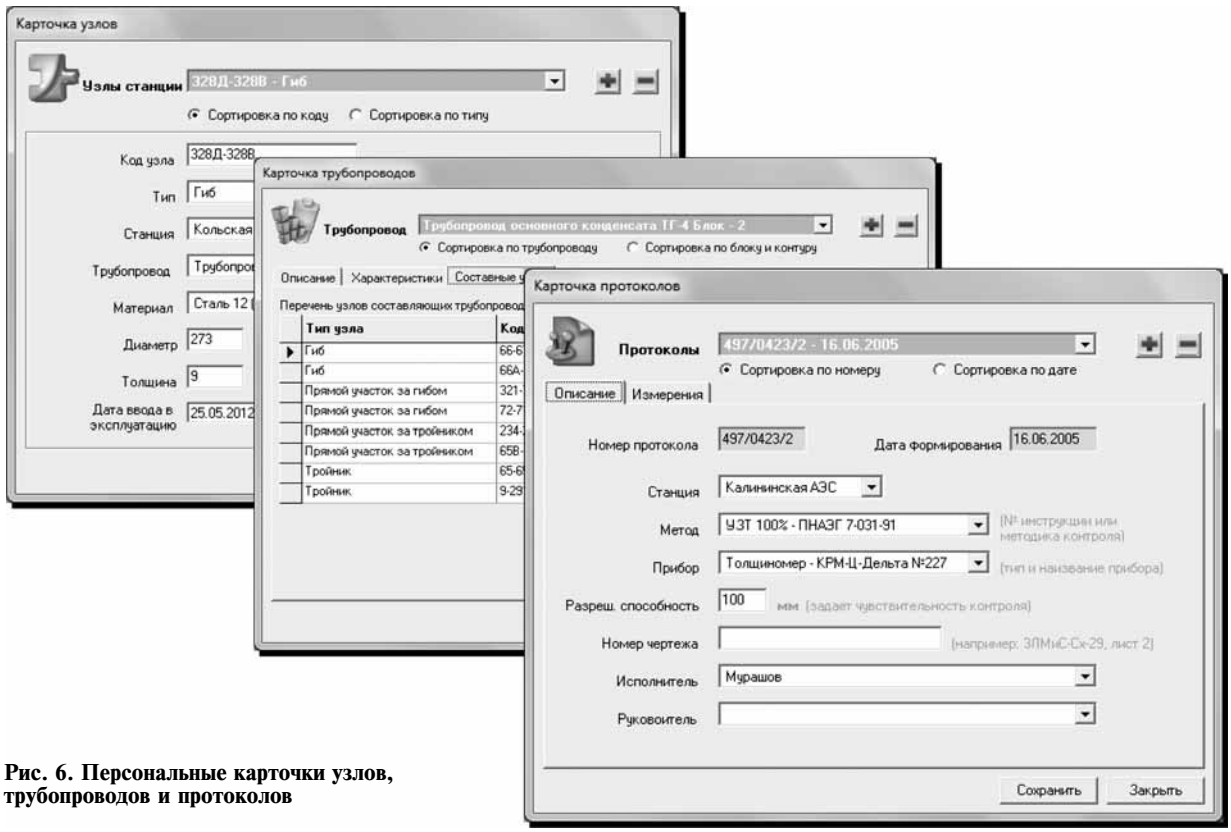

Рис. 6. Персональные карточки узлов, трубопроводов и протоколов

лизуется своя методика расчетов, где допустимая толщина S_{RT} выбирается из условия

$$S_{RT} = \max\{(S^*_{RT} + 0{,}2), (0{,}5S_{\text{ном}} + 0{,}2)\},$$

где $S_{\text{ном}}$ — номинальное значение толщины стенки, мм; S^*_{RT} — расчетная толщина стенки, мм, по условию прочности. Например, для прямолинейного участка трубопровода, нагруженного внутренним давлением, S^*_{RT} определяется как

$$S^*_{RT} = \frac{pD_a}{2\varphi_{\text{T}}[\sigma] + p},$$

где D_a — наружный диаметр трубопровода, мм; φ_T — расчетный коэффициент снижения прочности для прямого участка трубопровода, рекомендуемое значение $\varphi_{\text{T}} = 1$; p — рабочее внутреннее давление, МПа; $[\sigma]$ — допускаемое напряжение, МПа.

Основной функцией представленного расчета является последующее определение остаточного ресурса, чего нельзя сделать без нахождения допустимой толщины стенки трубопровода для каждого элемента.

Функция ***программного определения скорости ЭКИ и остаточного ресурса*** основана на методе первичной обработки, целью которого является определение поврежденных участков трубопроводов [9]. Расчет скорости ЭКИ при этом проводится по данным, введенным из протоколов контроля (рис. 7), а расчет остаточного ресурса учитывает еще и предварительно рассчитанные данные о допустимых толщинах стенок трубопроводов по каждому элементу трубопровода.

Скорость ЭКИ вычисляется по следующей формуле:

$$W_{\text{ЭКИ}} = \frac{\Delta S}{T_{\text{экспл}}} = \frac{S_{\text{ном}} - S_{\min}}{T_{\text{экспл}}},$$

где $S_{\text{ном}}$ — номинальное значение толщины стенки, мм; $S_{\min}$ — минимальное значение толщины стенки, мм; $T_{\text{экспл}}$ — время эксплуатации оборудования, годы.

Рис. 7. Методика расчета скорости ЭКИ и остаточного ресурса

В свою очередь, остаточный ресурс определяется по следующей формуле:

$$T_{\text{ост}} = \frac{\Delta S_{\text{доп}}}{W_{\text{ЭКИ}}} = \frac{S_{\text{ном}} - S_{\min}}{W_{\text{ЭКИ}}},$$

где $S_{\text{доп}}$ — допустимое значение толщины стенки, мм; $\Delta S_{\text{доп}}$ — допустимое отклонение относительно номинального значения, мм; $W_{\text{ЭКИ}}$ — скорость ЭКИ (мм/год).

Согласно рассматриваемой методике поврежденным считается такой участок трубопровода, у которого толщина стенки меньше номинала; критическим элементом — такой элемент, у которого толщина стенки меньше допускаемой или ожидается, что толщина меньше допускаемой будет достигнута до даты следующего контроля.

При эксплуатации трубопроводов коррозионные повреждения не должны превышать нормируемые значения. Поэтому приоритет в выборе участков для следующего контроля определяется на основе учета скорости ЭКИ. Таким образом, выявляются наиболее опасные участки, которым следует уделить повышенное внимание при контроле остаточного ресурса трубопроводов АЭС [10]. По этой причине для того чтобы используемые в составе автоматизированной системы процедуры были легитимны (сертифицированы), необходим документ, регламентирующий расчет скорости ЭКИ по данным контроля. Аналогичные требования должны быть предъявлены к верификации программных средств, используемых для прогнозирования скорости ЭКИ элементов трубопроводов по параметрам, отражающим состояние их эксплуатации.

В настоящее время на стадии подготовки находится этап регистрации авторских прав коллектива разработчиков на исходный компьютерный код программ для ЭВМ (программный комплекс и БД). Далее, по ожидаемой утверждения в отрасли методике будет проходить верификация программной реализации математического алгоритма расчета, а затем сертификация всего программного комплекса как коммерческого продукта.

Заключение

Основными результатами исследований, представленных в настоящей статье, являются выявление и обобщение максимального числа объектов, факторов, характеристик и методик, участвующих в формировании единого информационного поля, описывающего жизненный цикл заданного типа оборудования трубопроводов АЭС. Такая система поддержки принятия решений, отвечающая всем необходимым требованиям, необходима ответственному персоналу АЭС для ведения комплексного учета собранных данных мониторинга по толщинометрии трубопроводов и при разработке рабочих программ по дальнейшему контролю. В результате выбрано направление на создание комплексной автоматизированной системы, основанной на едином понятийном и алгоритмическом аппарате.

Получена работоспособная модель пользовательского интерфейса для внесения и редактирования данных контроля элементов трубопроводов, центральная база данных с массивом предзаполненных нормативно-справочных данных, а также реализованы математические модели для расчета допустимых толщин трубопроводов, прогнозирования скорости процесса ЭКИ и определения остаточного ресурса его элементов.

Список литературы

1. **Пономаренко Г. Л., Рыжов С. Б., Быков М. А., Ермаков Д. Н.** Новый концептуальный подход к определению минимально достаточной эффективности аварийной защиты ВВЭР // Научно-технический сборник ВАНТ, серия "Обеспечение безопасности АЭС". Реакторные установки с ВВЭР. 2006. Вып. 13. С. 27—41.

2. **Королев А. В., Кравченко В. П.** Сравнительный анализ двух подходов к прогнозированию эрозионно-коррозионного износа трубопроводов АЭС // Труды Одесского гос. политехн. ун-та. 2001. Вып. 3 (12). С. 55—56.

3. **Kastner W., Hofmann P., Nopper H.** Erosionskorrosion in Kraftwerksanlagen — Entscheidungshilfe fuer Massnahmen zur Schadensvermeidung // VGB Kraftwerkstechnik. 1990. B. 70, heft 11. P. 939—948.

4. **Буртаев Ю. Ф., Острейковский В. А.** Статистический анализ надежности объектов по ограниченной информации. М.: Энергоатомиздат, 1995.

5. **РД ЭО** 1.1.2.11.0571—2010. Нормы допустимых толщин элементов трубопроводов из углеродистых сталей при эрозионно-коррозионном износе атомных станций. М.: ОАО "Концерн Росэнергоатом", введен в действие с 29 октября 2012 г.

6. **Chexal V. K., Dietrich E. B., Horowitz J. S. et al.** CHEC (Chexal-Horowitz-Erosion-Corrosion) Computer Program User's Manual. NSAC-112. Electrical Power Research Institute, Palo Alto, CA, Final Report. February 1988.

7. **Гулина О. М., Бараненко В. И.** О прогнозировании длительности эксплуатации элементов трубопроводных систем АЭС, подверженных эрозионно-коррозионному износу // Сб. ВНИИАЭС "Основные работы, выполненные в 2011 г." М.: ВНИИАЭС, 2012. С. 81—93.

8. **Гулина О. М., Бараненко В. И., Просвирнов А. А., и др.** Разработка программных средств и нормативной документации по эрозионно-коррозионному износу на АЭС // Теплоэнергетика. 2012. № 5. С. 34—39.

9. **Бараненко В. И., Янченко Ю. А., Гулина О. М., Тарасова О. С.** Эксплуатационный контроль трубопроводов, подверженных эрозионно-коррозионному износу // Теплоэнергетика. 2009. № 5. С. 20—27.

10. **Бараненко В. И., Янченко Ю. А., Гулина О. М., Докукин Д. А.** О расчете скорости эрозионно-коррозионного износа и остаточного ресурса трубопроводов АЭС // Известия вузов. Ядерная энергетика. 2010. № 2. С. 55—63.

V. A. Fedotov[1, 2], Postgraduate Student, System Maintenance Team Leader of the IT department, e-mail: workstudio@yandex.ru,
O. M. Gulina[2], Professor, e-mail: olga@iate.obninsk.ru
[1]National Research Nuclear University MEPhI (Moscow Engineering Physics Institute) Institute of Nuclear Power Engineering, Obninsk
[2]JSC "Instrumental plant "Signal", Obninsk

Development of Decision Support System on the Prediction of the Life of NPP Equipment under Erosion-Corrosion Conditions

Ensuring of safe and reliable operation of nuclear power plants equipment must be based on making decisions on risk management throughout its life cycle.

The pipeline system of power units, working in conditions of flow-accelerated corrosion (FAC), significantly affects the quality, cost and safety of the nuclear power station in whole.

On this point reliable and long-term practice of observations, as well as application of sophisticated mathematical methods of forecasting, based on physical or empirical models of the investigated process is important.

In this regard there is an urgent need for creation of Russian analytical software with a common structured database of performance characteristics conducting reliable calculation of FAC process intensity as well as determining the residual life of equipment elements.

Such computer system should not concede to analogues in the possibilities of ergonomics and process orientation, but should take into account the specifics of Russian nuclear industry and experience of research in the field of designing analytical models for the prediction of the equipment life.

The article is devoted to the description of scientific and practical researches on creation of decision support system on the management of the life of NPP equipment.

The test model of the application solves the tasks of regular pipes wall thickness control data collection and storage mathematical prediction of the FAC speed and determination of residual life of nuclear power plant pipelines elements.

The system is intended primarily for the personnel of the nuclear power station, so it is developed with use of modern algorithmic and software tools to ensure a safe and comfortable work with a common database in the mode of "one window".

Keywords: *automated system, nuclear power plant, data base, standard, interface, lifetime, piping, flow-accelerated corrosion*

References

1. **Ponomarenko G. L., Ryzhov S. B., Bykov M. A., Yermakov D. N.** Novy konceptualny podhod k opredeleniyu minimalno dostatochnoy effektivnosti avariynoy zashchity VVER. *Nauchno-tehnichesky sbornik VANT, seriya "Obespechenie bezopasnosti AES". Reaktornye ustanovki s VVER*. 2006. N. 13. P. 27—41.

2. **Korolev A. V., Kravchenko V. P.** Sravnitelny analiz dvuh podhodov k prognozirovaniyu erozionno-korrozionnogo iznosa truboprovodov AES. *Trudy Odesskogo gosudarstvennogo politehnicheskogo universiteta*. 2001. N. 3 (12). P. 55—56.

3. **Kastner W., Hofmann P., Nopper H.** Erosionskorrosion in Kraftwerksanlagen — Entscheidungshilfe fuer Massnahmen zur Schadensvermeidung. *VGB Kraftwerkstechnik*. 1990. B. 70, heft 11. P. 939—948.

4. **Burtaev Y. F., Ostreikovsky V. A.** *Statistichesky analiz nadezhnosti ob'ektov po ogranichennoy informatsii*. M.: Energoatomizdat, 1995.

5. **RD EO 1.1.2.11.0571—2010**. Normy dopustimyh tolshchin elementov truboprovodov iz uglerodistyh staley pri erozionno-korrozionnom iznose. M.: OAO "Kontsern Rosenergoatom", vveden v deistvie s 29 oktyabrya 2012.

6. **Chexal V. K., Dietrich E. B., Horowitz J. S., et al.** CHEC (Chexal-Horowitz-Erosion-Corrosion) Computer Program User's Manual, NSAC-112, Electrical Power Research Institute, Palo Alto, CA, Final Report, February 1988.

7. **Gulina O. M., Baranenko V. I.** O prognozirovanii dlitelnosti ekspluatatsii elementov truboprovodnyh sistem AES, podverzhennyh erozionno-korrozionnomu iznosu. *Sb. VNIIAES "Osnovnye raboty, vypolnennye v 2011"* M.: VNIIAES, 2012. P. 81—93.

8. **Gulina O. M., Baranenko V. I., Prosvirnov A. A. etc.** Razrabotka programmnyh sredstv i normativnoj dokumentacii po jerozionno-korrozionnomu iznosu na AES. *Teplojenergetika*. 2012. N. 5. P. 34—39.

9. **Baranenko V. I., Yanchenko Y. A., Gulina O. M., Tarasova O. S.** Ekspluatatsionny kontrol truboprovodov, podverzhennyh erozionno-korrozionnomu iznos. *Teploenergetika*. 2009. N. 5. P. 20—27.

10. **Baranenko V. I., Yanchenko Y. A., Gulina O. M., Dokukin D. A.** O raschete skorosti erozionno-korrozionnogo iznosa i ostatochnogo resursa truboprovodov AES. *Izvestiya vuzov. Yadernaya energetika*. 2010. N. 2. P. 55—63.

УДК 519.612

К. Ф. Иванова, канд. техн. наук, докторант, СПбГУ, e-mail: klara.i2010@yandex.ru

Угловые решения интервальной системы линейных алгебраических уравнений высокого порядка на основе "знаковой" методики

Целью работы, описанной в данной статье, является оценка погрешности решения интервальной системы линейных алгебраических уравнений (ИСЛАУ) с интервальными коэффициентами матриц и компонентами правых частей в узком интервале их изменения на основе "знаковой" методики. Конструируется специализированный алгоритм, позволяющий найти оценки угловых векторов решения формализованных точечных систем, аналогичные результатам "внешнего" оценивания множества решений ИСЛАУ в интервальной алгебре. Теоретически обоснована методика алгебраического подхода к решению систем разного порядка и подтверждено численными примерами, в какой степени количественная неопределенность исходной информации оказывает качественное влияние на решение.

Ключевые слова: *"знаковая" методика, угловые векторы, "внешнее" оценивание, точечные системы, множество решений*

Введение

К решению систем линейных алгебраических уравнений (СЛАУ) сводятся многие физические, технические и экономические задачи. В практических задачах, описываемых СЛАУ, элементы матриц и векторы правых частей известны, как правило, неточны, и решение обратной задачи по определению вектора неизвестных в ряде случаев получается со значительными погрешностями. Эти причины приводят к неизбежности интервального задания коэффициентов матриц и компонент правых частей уравнений и к решению интервальных систем линейных алгебраических уравнений (ИСЛАУ), позволяющих находить решение задачи, ограниченное определенными интервалами для компонент вектора неизвестных. Однако несмотря на всю общность предлагаемых подходов, существуют значительные трудности при их реализации. В литературе, посвященной решению интервальных систем, приведено большое число методик, проводимых по оценкам решения интервальных систем [1—7]. Прежде всего в интервальном анализе основным допущением является требование к неособенности интервальной матрицы и проводится анализ частных видов интервальных матриц, как наиболее доступных для решения. В модифицированном методе Ньютона и его разновидностях требуются удачное задание начального приближения и другие конкретные требования к каждой частной задаче. Но самое большое неудобство при использовании интервальных подходов к моделям задач, выраженных интервальными системами — их операции со всем интервальным массивом матрицы, что является, как отмечают специалисты, NP-трудной задачей (труднорешаемой) [2, 3]. Такие задачи требуют не только четкого, не перегруженного алгоритма, но и большого числа вычислений со значительным временем счета, как и вычисление внешних покоординатных оценок для множества решений с любой заданной абсолютной или относительной точностью. И эти трудности экспоненциально возрастают с ростом размера системы.

Кроме того, модели практических задач зачастую не содержат априорной информации относительно вырожденности или плохой обусловленности матриц системы, определение которой, в свою очередь, также является NP-трудной задачей [4—6]. И этот факт может выявиться только в процессе решения задачи. Поэтому непреложным остается только требование к конструированию алгоритмов для получения непрерывности — приближенного решения задачи при возмущении исходных данных системы.

Разработанная новая методика расчета ИСЛАУ позволяет получить оценку угловых компонент вектора неизвестных значительно менее сложным путем, чем при применении общеизвестных методик, а именно:

- предложен алгебраический подход к решению интервальной системы любого порядка и любого вида с заполненными матрицами (частные случаи незаполненных и разреженных матриц могут быть любыми), приводящий к системе точечных уравнений, число которых ограничено порядком системы;
- получены результаты по оценке чувствительности ИСЛАУ с матрицами любого порядка;
- проведен анализ влияния неполного числа интервальных коэффициентов матрицы на погрешность решения системы;

- проведено доказательство простого способа определения неособенности интервальной системы;
- неособенность матриц и плохо обусловленные возмущенные матрицы системы анализируются при работе программы на основании разработанного алгоритма и автоматически фиксируются в процессе решения задачи.

При этом к допущениям, которые приняты при использовании "знаковой" методики, относятся следующие:

1) матрицы интервальных систем должны быть квадратными;

2) системы должны быть с достаточно узкими матрицами, относительные погрешности элементов которых не превосходят проценты и доли процентов, так как в этом случае предложенная методика позволяет получить наиболее точные результаты, что вполне естественно для большинства практических задач;

3) элементы матриц и компоненты правых частей не должны иметь противоположные по знаку границы интервала их изменения, так как в этом случае нарушается допущение 1.

Основные положения "знаковой" методики для граничных матриц ИСЛАУ

Неточность измерений как следствие неопределенности элементов матрицы и компонент вектора правой части, истинные значения которых попадают в интервалы их изменения, приводит к интервальной системе

$$\mathbf{Ax} = \mathbf{b}, \tag{1}$$

где матрица и правая часть представляются в виде интервалов

$$\mathbf{A} = [\underline{A}, \overline{A}]; \ \mathbf{b} = [\underline{b}, \overline{b}] \tag{2}$$

с интервальными элементами ненулевой ширины с левыми и правыми границами и интервальными компонентами правой части:

$$\boldsymbol{a}_{ij} = [\underline{a}_{ij}, \overline{a}_{ij}], \ \boldsymbol{b}_i = [\underline{b}_i, \overline{b}_i], \ \ \forall i, j = \overline{1, n}, \tag{3}$$

где n — порядок системы, нижнее и верхнее подчеркивания означают границы матрицы $\mathbf{A} = (\boldsymbol{a}_{ij}) \in \mathbf{IR}^{n\times n}$, вектора правой части $\mathbf{b} = (\boldsymbol{b}_i) \in \mathbf{IR}^n$ и их компонент; $\mathbf{x}$ — искомый вектор с координатами $\boldsymbol{x}_j = [\underline{\boldsymbol{x}}_j, \ \overline{\boldsymbol{x}}_j]$, $\mathbf{x} = \{\boldsymbol{x}_j\}$, $\mathbf{x} \in \mathbf{IR}^n$.

Постановка известной интервальной задачи [4] заключается в определении покоординатной оценки объединенного множества решений

$$\Sigma_{\exists\exists} = \{x \in \mathbf{R}^n | (\exists A \in \mathbf{A})(\exists b \in \mathbf{b})(Ax = b)\}, \tag{4}$$

образованного всеми решениями точечных систем $Ax = b$ с $A \in \mathbf{A}$ и $b \in \mathbf{b}$.

Двойной квантор $\exists\exists$ обозначает существование матриц $\mathbf{A}$ и векторов b, которые определяют существование всех решений, подчиненных $\Sigma_{\exists\exists}$, в том числе и угловых, т. е. оценок $\min\{x_k | x \in \Sigma_{\exists\exists}\}$ снизу и $\max\{x_k | x \in \Sigma_{\exists\exists}\}$ сверху для $k = \overline{1, n}$, так, что оценивающими подмножествами являются интервальные векторы в $\mathbf{R}^n$ со сторонами, параллельными координатным осям.

Используемый алгебраический подход выполняет эту процедуру поиска двусторонних угловых отклонений решения аналогично оптимальному внешнему оцениванию объединенного множества решений. Знаковый подход к решению системы (1) помогает проанализировать влияние интервалов неопределенности, выраженных двухсторонними границами интервалов для матриц, правых частей, их элементов и компонент (2), (3). Интервальная система (1) является совокупностью всех точечных $n\times n$-систем $Ax = b$ с точечными матрицами $A \in \mathbf{A}$, $A = (a_{ij}) \in \mathbf{R}^{n\times n}$ и правыми частями $b \in \mathbf{b}$, $b = \{b_i\} \in \mathbf{R}^n$, $i, j = \overline{1, n}$. Решение системы (1) существует, если все входящие в нее точечные системы являются неособенными.

Основные положения знакового подхода к решению ИСЛАУ

Идея знакового подхода к решению СЛАУ Ю. П. Петрова [8] состоит в утверждении существенного влияния знаков относительных погрешностей элементов матрицы и компонент правых частей точечной системы $Ax = b$ на появление значительных погрешностей вектора решения x.

Основными положениями "знаковой" методики, на основании которых конструируется алгоритм оценки погрешности вектора неизвестных x, являются следующие.

Тезис 1. Детерминант матрицы СЛАУ с возмущенными элементами и компонентами правых частей записывается в виде

$$\tilde{D} = \sum_{j=1}^{n} a_{ij}(1 + \varepsilon_{ij})A_{ij}, \ i = \overline{1, n}, \ A_{ij} = (-1)^{i+j} M_{ij}, \tag{5}$$

где A_{ij} — алгебраические дополнения элемента a_{ij}; M_{ij} — миноры элементов a_{ij} матрицы A_{ij}, ε_{ij} — относительная погрешность элемента a_{ij}.

Тезис 2. Приращение определителя в линейном приближении представляется как

$$\Delta = \sum_{j=1}^{n} a_{ij}A_{ij}\varepsilon_0, \ i = \overline{1, n}, \tag{6}$$

при допущении, что приращение относительной погрешности не превосходит самой погрешности $\Delta\varepsilon_{ij} \leqslant |\varepsilon_{ij}|$, а абсолютное значение ε_{ij} не превосходит значения ε_0 (выбранного как максимальное из всех ε_{ij}), $|\varepsilon_{ij}| \leqslant \varepsilon_0$.

Тезис 3. При замещении одного из столбцов матрицы системы (5) столбцом правой части линейное приращение вспомогательной матрицы δ_i записывается аналогично тезису 2:

$$\Delta = \sum_{j=1}^{n} b_i A_{ij} \delta_0, \; i = \overline{1, n}, \tag{7}$$

где A_{ij} — алгебраические дополнения элемента b_i; δ_0 выбирается максимальным из всех δ_i, так что $|\delta_i| \leqslant \delta_0$.

Тезис 4. Выявление наибольших значений приращений детерминантов (6), (7) происходит благодаря выбору определенного знака "+" или "−" погрешностей ε_0 и δ_0 в соответствии со знаками произведений a_{ij} и A_{ij}:

$$\begin{cases} (+)\varepsilon \vee (+)\delta \text{ for } \varepsilon_0 \;\&\; \delta_0 \text{ if } \operatorname{sign}(\varepsilon_0) = \\ = \operatorname{sign}(a_{ij}A_{ij}) \;\&\; \operatorname{sign}(\delta_0) = \operatorname{sign}(b_i A_{ij}); \\ (-)\varepsilon \vee (-)\delta \text{ for } \varepsilon_0 \;\&\; \delta_0 \text{ if } \operatorname{sign}(\varepsilon_0) = \\ = (-1)\operatorname{sign}(a_{ij}A_{ij}) \;\&\; \operatorname{sign}(\delta_0) = \\ = (-1)\operatorname{sign}(b_i A_{ij}), \end{cases}$$

$$i, j = \overline{1, n}. \tag{8}$$

Значения $+\varepsilon_0$ и $+\delta_0$ относительных погрешностей вызывают максимальные линейные приращения выражений (6) и (7) в положительном направлении; а $-\varepsilon_0$ и $-\delta_0$ — в отрицательном направлении. При реализации первого условия (8) конструируются элементы правых границ $A^+ \in R^{n\times n}$ и $b^+ \in R^n$, при выполнении второго условия формируются элементы левых границ $A^- \in R^{n\times n}$ и $b^- \in R^n$, соответствующие интервальным границам матрицы **A** и вектора **b**:

$$\mathbf{A} = [A^-, A^+], \; \mathbf{b} = [b^-, b^+] \tag{9}$$

с элементами

$$a_{ij}^- = a_{ij}^{\text{cp}} - \varepsilon_0 a_{ij}^{\text{cp}}, \; b_i^- = b_i^{\text{cp}} - \delta_0 b_i^{\text{cp}},$$
$$a_{ij}^+ = a_{ij}^{\text{cp}} - \varepsilon_0 a_{ij}^{\text{cp}}, \; b_i^+ = b_i^{\text{cp}} - \delta_0 b_i^{\text{cp}}, \tag{10}$$

где индексом "ср" отмечены средние значения интервалов, соответствующие средней точечной системе $A^{\text{cp}}x = b^{\text{cp}}$.

При этом новые (9) и прежние (2) интервальные границы матрицы **A** и вектора **b** в общем случае не совпадают: $A^- \neq \underline{A}$, $A^+ \neq \overline{A}$ и $b^- \neq \underline{b}$, $b^+ \neq \overline{b}$.

На основании условий (8) выполняется формальный переход от интервальных коэффициентов (3) к новым (10), которые, в соответствии с тезисами 1—4, определяют наибольшее значение приращений детерминантов (6) и (7) в положительном или отрицательном направлениях. Все характеристики, включая средние значения матрицы mid(A), сохраняют идентичные представления для систем с матрицами и правыми частями (2) и (9), которые выражаются через их интервальные границы.

Решение системы (1) существует и единственно, если матрица **A** неособенная (невырожденная). Как показано в работе [4], интервальная матрица $\mathbf{A} \in \mathrm{IR}^{n\times n}$ неособенна, если неособенны все точечные $n\times n$ матрицы $A \in \mathbf{A}$. Только в случае неособенности интервальной матрицы речь может идти о существовании обратных граничных матриц $A_b^{(\pm)} \in R^{n\times n}$. По определению, для неособенной матрицы $\mathbf{A} \in \mathbf{IR}^{n\times n}$ обратной интервальной матрицей $\mathbf{A}^{-1} = \{A^{-1} | A \in \mathbf{A}\}$ является множество всех обратных точечных матриц, содержащихся в $\mathbf{A}^{-1}$. Интервальная матрица $\mathbf{A} \in \mathbf{IR}^{n\times n}$ особенна, если существует особенная точечная матрица. В дополнение к этому определению существует много критериев неособенности интервальной матрицы, но как отмечают специалисты, исследование этого вопроса является NP-трудной задачей.

По критерию Баумана [7], интервальная матрица **A** неособенна тогда и только тогда, когда произведение определителей всех ее крайних матриц положительно, т. е. $(\det A')(\det A'') > 0$ для любых $A', A'' \in \mathrm{vert}A$. Покажем, что при использовании знакового подхода, достаточно рассмотреть знак произведения определителей только двух матриц.

Теорема 1. Для существования неособенности матриц **A** достаточно выполнение неравенства $\det(A^-)\times \times \det(A^+) > 0$.

Доказательство. Допустим, что значение определителя средней матрицы $\det(A^{\text{cp}}) > 0$. Тогда исходя из условий (7), (9), A^+ и A^- представляют собой граничные матрицы с максимальными и минимальными определителями $\det(A^-)$ и $\det(A^+)$, в основу которых заложено формирование наименьших и наибольших приращений по сравнению со средней невозмущенной матрицей: $\det(A^-) < \det(A^{\text{cp}}) < \det(A^+)$. В этом случае возможны два варианта знаков определителей граничных матриц: $\det(A^-) > 0$, $\det(A^+) > 0$ и $\det(A^-) < 0$, $\det(A^+) > 0$. Смена знака меньшего из детерминантов $\det(A^-) > 0$ в случае плохой обусловленности матрицы системы может привести к его переходу через ноль и отрицательному значению. Если $\det(A^{\text{cp}}) < 0$, то смена знака $\det(A^+)$ с отрицательного на положительное значение ($\det(A^+) > 0$) также приводит к особенности матрицы. Отсюда следует, что условием неособенности интервальной матрицы A является выполнение условия $\det(A^-)\det(A^+) > 0$.

Проведенная оценка аналогична оценке неособенности интервальной матрицы по критерию Баумана, но позволяет для узких матриц рассмотреть определители не всех крайних $2^{n\times n}$ матриц, а только двух матриц A^+ и A^-.

Применение "знаковой" методики для оценки угловых решений ИСЛАУ

Покажем операцию построения интервального бруса, состоящего из угловых компонент вектора решения, имеющего место при оценивании множества решений, на примере использования "знаковой" методики.

Проведем расчет компонент вектора $x = [x^-, x^+]$ по формулам Крамера. Основными вариантами получения угловых решений системы с интервальными коэффициентами (9), (10) являются следующие подстановки формализованных по условиям (8) левых b^- и правых b^+ границ вектора **b** в матрицы A^- или A^+ главных определителей условий:

$$x_i1 = \frac{\Delta_+^{b_j^-}}{\Delta_+}, \; x_i2 = \frac{\Delta_-^{b_j^+}}{\Delta_-}, \; x_i3 = \frac{\Delta_+^{b_j^+}}{\Delta_+^{a_j^-}}, \; x_i4 = \frac{\Delta_-^{b_j^-}}{\Delta_-^{a_j^+}}, \qquad (11)$$

где Δ_+ — определитель знаменателя, сформированный в сторону увеличения на основании выбора границ элементов интервальной матрицы **A**; Δ_- — определитель, полученный в сторону уменьшения на основании выбора границ элементов интервальной матрицы A; $\Delta_+^{b_j^-}$ — замещение *j*-го столбца матрицы определителя, сформированного в сторону возрастания, вектором, полученным в сторону убывания определителя; $\Delta_-^{b_j^+}$ — замещение *j*-го столбца матрицы определителя, сформированного в сторону убывания, вектором, полученным в сторону возрастания определителя; $\Delta_-^{a_j^+}$ — замещение в определителе, сформированном в сторону уменьшения, *j*-го столбца, полученного в сторону увеличения определителя; $\Delta_+^{a_j^-}$ — замещение в определителе, сформированном в сторону увеличения, *j*-го столбца, полученного в сторону уменьшения определителя.

Формирование двухсторонних границ вектора $\mathbf{x} = [x^-, x^+]$ проводится по принципу отбора наибольших и наименьших компонент x_j^- и x_j^+, $j = \overline{1, n}$ при каждой подстановке векторов правых частей в матрицы соответствующих определителей формул (11) так, что $x^- = (x_1^-, x_2^-, ..., x_n^-)$ и $x^+ = (x_1^+, x_2^+, ..., x_n^+)$, $x \in R^{2n}$.

Такие же результаты можно получить через обратные матрицы с подстановкой *j*-х векторов b_j^- и b_j^+ в уравнения точечных систем $x = A^{-1}b$.

В общем виде систему точечных уравнений в матричном виде можно записать как

$$\begin{cases} A^-x = b_j^+; \\ A^+x = b_j^-, \end{cases} \; j = \overline{1, n}, \qquad (12)$$

где $b^- = (b_1^-, b_2^-, ..., b_n^-)$, $b^+ = (b_1^+, b_2^+, ..., b_n^+)$ получены на основании условий (8).

Как и по формулам Крамера, искомый вектор $\mathbf{x} = [x^-, x^+]$ формируется двумя наборами решений $2n$ $x^- = (x_1^-, x_2^-, ..., x_n^-)$ и $x^+ = (x_1^+, x_2^+, ..., x_n^+)$ уравнений системы (12). Матрицы и правые части с верхними символами A^- и b^+ определяют правые границы, а A^+ и b^- — левые границы компонент x^+ и x^-, аналогичные расчетам компонент по формулам (11) и тождественные граничным значениям минимальных и максимальных векторов при оценивании множества решений (4).

Если в процессе варьирования элементов матриц A^+ и A^- через ноль переходит один из определителей и меняет знак, то возникает особенность главных матриц системы (Теорема 1) и аварийное возрастание относительной погрешности вектора x, т. е. решения не существует. Если для вспомогательных детерминантов $\det(A_b^+)$ и $\det(A_b^-)$ возникают вариации, при которых $\det(A_b^-)\det(A_b^+) < 0$, это приводит к смене знака одной или более компонент векторов x^+ или x^-, что противоречит правильному решению задачи.

Примеры решения и оценка погрешности СЛАУ разного порядка

Расчеты показывают, что "знаковая" методика позволяет выявить погрешности решения, которые на несколько порядков превышают погрешность исходных данных, даже при сохранении неособенности матриц системы, что является нарушением третьего условия корректности по Ж. Адамару [9], когда незначительные погрешности исходных данных приводят к большим и даже аварийным погрешностям решения. При увеличении ε и δ для плохо обусловленных матриц особенность возникает раньше или позже в зависимости от порядка системы и числа элементов, подверженных знаковой зависимости их относительных погрешностей. С увеличением порядка системы порог минимума для ε, приводящий к смене знака определителей, сильно снижается.

На примере системы второго порядка получим наглядное представление о построении интервального бруса для вектора x, т. е. интервальной оболочки, с применением "знаковой" методики. Заданы исходные

средняя матрица A, вектор b и относительные погрешности:

$$A = \begin{pmatrix} 0{,}2762 & 0{,}8814 \\ 0{,}6962 & 0{,}0247 \end{pmatrix};\ b = (0{,}0242;\ 0{,}5500);$$

$$\varepsilon = \begin{pmatrix} 0{,}0100 & 0{,}0100 \\ 0{,}0100 & 0{,}0100 \end{pmatrix};\ \delta = (0{,}0100;\ 0{,}0100).$$

Согласно знаковому подходу получены следующие граничные векторы решения:

$$x^{cp} = \begin{pmatrix} 0{,}7979 \\ -0{,}2226 \end{pmatrix};\ x^{-} = \begin{pmatrix} 0{,}7817 \\ -0{,}2329 \end{pmatrix};\ x^{+} = \begin{pmatrix} 0{,}8144 \\ -0{,}2167 \end{pmatrix}.$$

На рис. 1 (см. вторую сторону обложки) и всех последующих рисунках прямые и ломаные линии соединяют показания значений относительных отклонений компонент вектора x, полученных при переменных по знаку погрешностях ε и δ в отрицательном (отмечены синим цветом) и в положительном (отмечены красным цветом) направлениях. Зеленым цветом отмечены относительные отклонения тех же компонент при постоянных по знаку погрешностях ε и δ. На рис. 1 (см. вторую сторону обложки) видно, что относительная погрешность первой компоненты обоих векторов при переменных по знаку ε и δ составляет ≈2%, а второй компоненты ≈4,5 % (отмечены на прямых otn1 и otn2), превосходя исходные погрешности более чем на порядок. Относительные погрешности при постоянных по знаку ε и δ отмечены на прямых otn3 и otn4. Абсолютные отклонениями при переменных по знаку ε и δ (abs1, abs2) превосходят абсолютные отклонения (abs3, abs4) при постоянных по знаку ε и δ практически в 2 раза.

Рассмотрим на примере системы десятого порядка характер изменения относительной погрешности граничных значений вектора x от значения входных относительных погрешностей ε и δ и от числа элементов, имеющих знаковые односторонние погрешности. Средняя матрица A^{cp} — случайная, вектор x задается средним вектором x^{cp} = [1111111111]; вычисленный средний вектор

b^{cp} = [5,79 3,31 4,36 4,70 3,44 4,10 4,04 5,81 3,56 4,86],
ε = 1 %, δ = 1 %,

$$A^{cp} = \begin{bmatrix} 0{,}5144 & 0{,}5341 & 0{,}8507 & 0{,}6126 & 0{,}7386 & 0{,}7690 & 0{,}5523 & 0{,}1465 & 0{,}1239 & 0{,}9479 \\ -0{,}8843 & 0{,}0900 & 0{,}5606 & 0{,}9900 & 0{,}5860 & 0{,}5814 & 0{,}6299 & 0{,}1891 & 0{,}4904 & 0{,}0821 \\ 0{,}5880 & 0{,}1117 & 0{,}9296 & 0{,}5277 & 0{,}2467 & 0{,}9283 & 0{,}0320 & 0{,}0427 & 0{,}8530 & 0{,}1057 \\ 0{,}1548 & -0{,}1363 & 0{,}6967 & 0{,}4795 & 0{,}6664 & 0{,}5801 & 0{,}6147 & 0{,}6352 & 0{,}8739 & 0{,}1420 \\ 0{,}1999 & 0{,}6787 & 0{,}5828 & 0{,}8013 & 0{,}0835 & 0{,}0170 & 0{,}3624 & 0{,}2819 & 0{,}2703 & 0{,}1665 \\ 0{,}4070 & 0{,}4952 & 0{,}8154 & 0{,}2278 & 0{,}6260 & 0{,}1209 & 0{,}0495 & 0{,}5386 & 0{,}2085 & 0{,}6210 \\ 0{,}7487 & 0{,}1897 & 0{,}8790 & -0{,}4981 & 0{,}6609 & 0{,}8627 & 0{,}4896 & 0{,}6952 & -0{,}5650 & 0{,}5737 \\ 0{,}8256 & 0{,}4950 & 0{,}9889 & 0{,}9009 & 0{,}7298 & 0{,}4843 & 0{,}1925 & 0{,}4991 & 0{,}6403 & 0{,}0521 \\ 0{,}7900 & 0{,}1476 & 0{,}0005 & 0{,}5747 & 0{,}8908 & -0{,}8449 & 0{,}1231 & 0{,}5358 & 0{,}4170 & 0{,}9312 \\ 0{,}3185 & 0{,}0550 & 0{,}8654 & 0{,}8452 & 0{,}9823 & 0{,}2094 & 0{,}2055 & 0{,}4452 & 0{,}2060 & 0{,}7287 \end{bmatrix}.$$

Для заданных ε и δ определитель левой границы матрицы $\det(A^{-})$ меняет знак по сравнению со знаком определителя матрицы $\det(A^{+})$: $\det(A^{cp}) = 0{,}0204$; $\det(A^{+}) = 0{,}0601$; $\det(A^{-}) = -0{,}0193$, и значение относительной погрешности решения становится аварийно большим. В то же время по норме граничные матрицы отличаются очень незначительно: $\mathrm{norm}(A^{cp}) = 5{,}7764$; $\mathrm{norm}(A^{+}) = 5{,}7798$; $\mathrm{norm}(A^{+}) = 5{,}7736$. С уменьшением ε и δ определители граничных матриц начинают различаться по знаку при увеличении числа строк, получивших погрешности по "знаковой" методике. Так, при ε = 0,2 %, δ = 0,2 % получим два варианта решения (рис. 2 см. вторую сторону обложки): первый (рис. 2, см. вторую сторону обложки) — знакопеременные погрешности получили элементы только одной строки матрицы и всех компонент правой части, второй (рис. 2, *б*, см. вторую сторону обложки) — знакопеременные погрешности получили все элементы матрицы и правой части.

Во втором варианте расчета появляется смена знака определителей вспомогательных матриц в отличие от первого варианта, когда смены знака определителей нет. Однако относительные погрешности превышают исходные в сотни раз в обоих вариантах.

Для случайных матриц порядка $n = 40$ (рис. 3, см. вторую сторону обложки) по сравнению с матрицами порядка $n = 10$ при прежних ε = 0,2% и δ = 0,2% вероятность появления чрезвычайно больших относительных погрешностей решения значительно возрастает. Диапазон различия погрешностей увеличивается при знаковых относительных погрешностях одной строки матрицы и всех строк, так как в этом случае элементы одной строки составляют только 1/40 часть от всех элементов матрицы.

В случае всех знакопеременных погрешностей матрицы погрешность решения становится авостной.

Таким образом, решение ИСЛАУ со знаковыми погрешностями элементов матрицы и вектора правой части очень чувствительно к увеличению порядка системы и зависит от числа возмущенных элементов, но может сильно разниться от системы к системе одного порядка, что легко проверяется на случайных матрицах. В то же время постоянные по знаку относительные погрешности системы инициируют погрешность, практически не превышающую суммарную погрешность заданных ε и δ.

Заключение

Методика оценки погрешности решения СЛАУ по "знаковой" методике модифицирована и применена к оценке погрешности решений ИСЛАУ. С этой целью выявлены односторонние границы интервалов элементов и компонент вектора правой части матриц, позволяющих интерпретировать интервальную систему двумя точечными матричными уравнениями, но с подстановкой $2n$ правых частей. В результате алгоритмизации решения точечных систем получена двухсторонняя оценка отклонений решения по заданным относительным погрешностям коэффициентов интервальной системы.

Теоретически обоснованы условия и доказана теорема существования неособенности у систем с узкими интервальными матрицами, получены оценки максимальных отклонений вектора решения x, идентичные внешнему интервальному оцениванию множества решений задачи интервальной алгебры.

Впервые построены примеры и проведены численные эксперименты по оценке двухсторонних относительных погрешностей вектора x для систем с заполненными матрицами высокого порядка и со знакопеременными относительными погрешностями различного числа интервальных элементов матрицы системы. Установлено, в какой степени количественная неопределенность исходной информации оказывает качественное влияние на решение.

Список литературы

1. **Hansen R. E.** On linear algebraic equations with interval coefficients // Topics in Interval Analysis / Ed. E. Hansen. Oxford Univ. Press. 1969. P. 35—46.

2. **Алефельд Г., Херцбергер Ю.** Введение в интервальные вычисления. М.: Мир, 1987.

3. **Ning S., Kearfott R. B.** A comparison of some methods for solving linear intervalequations // SIAM J. Numer. Anal. 1997. N. 34. P. 3—17.

4. **Шарый С. П.** Конечномерный интервальный анализ. Новосибирск: XYZ, 2007. 700 с.

5. **Rohn J.** Solvability of systems of linear interval equations // SIAM Journal on Matrix Analysis and Applications. 2003. N 25. P. 237—245.

6. **Добронец Б. С.** Интервальная математика. Красноярск: Изд-во Сиб. гос. ун-та, 2007. 276 с.

7. **Baumann M.** A regularity criterion for interval matrices. Collection of Scientific Papers, Honoring Prof. Dr. K. Nickel on Occasion of his 60th Birthday. Part I / J. Garloff eds. University of Freiburg im Breis., 1984.

8. **Петров Ю. П.** Как обеспечить надежность решения систем уравнений. СПб.: БХВ, 2009. 172 с.

9. **Тихонов А. Н., Арсенин В. Я.** Методы решения некорректных задач. М.: Наука, 1979. 284 с.

K. F. Ivanova, Saint-Petersburg State University, e-mail: klara.i2010@yandex.ru

Angular Solutions ISLAE of a High Order on the Basis of a Sign Technique

The work purpose is the estimation of an error of the solution of interval system of the linear algebraic equations (ISLAE) with interval factors of matrixes and components of the right parts in a narrow interval of their change on a basis of a "sign technique". The specialized algorithm is designed, allowing to find estimations of angular vectors of the solution of the formalized point systems, similar to results of "external" estimation of set of solutions ISLAE in interval algebra. The technique of the algebraic approach to the solution of systems of a different order is theoretically proved and is confirmed by numerical examples, in what degree quantitative uncertainty of the initial information makes qualitative impact on the solution.

Keywords: *"sign" technique, angular a vector, "external" estimation, point systems, set of solutions*

References

1. **Hansen R. E.** On linear algebraic equations with interval coefficients. *Topics in Interval Analysis* / Ed. E. Hansen. Oxford Univ. Press. 1969. P. 35—46.

2. **Alefel'd G., Hercberger Ju.** *Vvedenie v interval'nye vychislenija*. M.: Mir, 1987.

3. **Ning S., Kearfott R. B.** A comparison of some methods for solving linear intervalequations. *SIAM J. Numer. Anal.* 1997. N. 34. P. 3—17.

4. **Sharyj S. P.** *Konechnomernyj interval'nyj analiz.* Novosibirsk: XYZ, 2007. 700 p.

5. **Rohn J.** Solvability of systems of linear interval equations. *SIAM Journal on Matrix Analysis and Applications.* 2003. N. 25. P. 237—245.

6. **Dobronec B. S.** Interval'naja matematika. Krasnojarsk: Izd-vo Sib. gos. un-ta, 2007. 276 p.

7. **Bauman M.** *A regularity criterion for interval matrices.* Collection of Scientific Papers, Honoring Prof. Dr. K. Nickel on Occasion of his 60th Birthday. Part I / J. Garloff eds. University of Freiburg im Breis, 1984.

8. **Petrov Ju. P.** *Kak obespechit' nadezhnost' reshenija sistem uravnenij.* SPb.: BHV, 2009. 172 p.

9. **Tihonov A. N., Arsenin V. Ja.** *Metody reshenija nekorrektnyh zadach*. M.: Nauka, 1979. 284 p.

УДК 510.22/519.61

Я. А. Воронцов, аспират, e-mail: voroncov_ya@sc.vsu.ru, **М. Г. Матвеев**, д-р техн. наук, проф., зав. каф., e-mail: mgmatveev@yandex.ru, Воронежский государственный университет

Алгебраические операции с нечеткими *LR*-числами с использованием преобразования *L*

Предложена формализация нечетких чисел, основанная на преобразовании L α-интервалов и направленная на повышение адекватности результатов решения задач с нечеткими параметрами. Исследованы и доказаны свойства преобразования L в целях демонстрации его пригодности для вычислений в нечетких задачах, решаемых как совокупность четких на заданном множестве α-интервалов. Введены алгебра для модифицированных нечетких чисел, доказаны свойства основных алгебраических операций, нейтрального и обратного элементов. Применение алгебры для нечетких LL/RR-чисел проиллюстрировано примерами задач, простота которых, однако, не ограничивает общности предлагаемой методики вычислений. Данная методика актуальна ввиду необходимости решения задач, которые могут возникнуть в процессе принятия управленческих решений на предприятиях в условиях нечеткой неопределенности.

Ключевые слова: *нечеткие параметры, α-уровень, L-преобразование, алгебра нечетких чисел, точечные оценки, модифицированные нечеткие числа*

Введение

Нечеткие параметры обычно применяют в задачах принятия решений и управления с неопределенностью тогда, когда использование случайных величин затруднено по тем или иным причинам. В нечетких вычислениях приходится выбирать вид нечетких чисел и соответствующие операции над этими числами. В частности, наиболее популярны *LR*-числа, применяемые в большинстве программных пакетов для обработки нечеткой информации.

Используемые на практике подходы к реализации нечетких вычислений, как правило, обладают теми или иными существенными недостатками, описанными в работах [1—3]:

- необходимы значительные вычислительные ресурсы для выполнения операций (принцип обобщения Заде);
- функция принадлежности результата определяется на максимально широком носителе, что при обеспечении математической строгости завышает степень неопределенности (принцип обобщения Заде, α-уровневый принцип обобщения и интервальные алгебры, арифметики нечетких *LR*-чисел);
- область определения функции принадлежности ограничена, например, условием неотрицательности переменных, поскольку не вводятся понятие обратного числа и операция деления для тех чисел, носитель которых делится нулем на две части (арифметики нечетких *LR*-чисел);
- вычислительные операции над нечеткими числами могут приводить к нарушению естественных отношений, например, операция вычитания с равными нечеткими операндами не приводит к нулю, не выполняется четкая тождественность уравнения с нечеткими параметрами после подстановки нечеткого решения и т. п. (арифметики нечетких *LR*-чисел);
- нелинейные операции умножения и деления искажают форму *LR*-числа (арифметики нечетких треугольных чисел, α-уровневый принцип обобщения).

Основной причиной указанных недостатков является то, что структура множества нечетких чисел (т. е. множества нечетких подмножеств числовой оси) представляет собой векторную решетку [4], которая неадекватна вычислительным задачам выбора и управления. Возможное разрешение проблемной ситуации состоит в подборе подходящего пространства над адекватной алгебраической структурой множества нечетких чисел. Такой подход был использован, например, в работе [3], где введены операции, формально удовлетворяющие аксиомам поля нечетких *LR*-чисел. Однако введенные автором работы [3] обратные по сложению и умножению элементы не отвечают определению *LR*-чисел, так как содержат отрицательные коэффициенты нечеткости, что ставит под сомнение существование поля *LR*-чисел.

Исследование, результаты которого приведены в данной статье, направлено на формирование алгебраической структуры с обратными элементами, отвечающей аксиомам поля для множества треугольных нечетких *LL/RR*-чисел, т. е. таких нечетких чисел, у которых правый (левый) коэффициент нечеткости равен нулю. Для перехода от обычных треугольных чисел к *LL/RR*-числам предлагается использовать линейное преобразование *L*, доказательства свойств которого также изложены в рамках данного исследования.

1. *L*-преобразование нечеткого числа и построение α-уровневых решений

Введем несколько определений, используемых в статье.

Определение 1. Нечеткое число *LR*-типа — разновидность нечетких чисел, функция принадлежности которых задается с помощью невозрастающих на множестве действительных чисел функций действительного переменного $L(x)$ и $R(x)$, для которых справедливы соотношения

- $L(-x) = L(x)$; $R(-x) = R(x)$;
- $L(0) = R(0)$.

Определение 2. Нечеткое треугольное число $\tilde{A}$ — частный случай числа *LR*-типа, задаваемый в виде тройки действительных чисел $\langle x^L, m, x^R\rangle$, где m — мода, а x^L, x^R — координаты точек пересечения функции принадлежности с осью *Ox*. Функция принадлежности числа $\tilde{A}$ определяется как

$$\mu_{\tilde{A}}(x) = \begin{cases} \dfrac{x - x^L}{m - x^L};\ x \in [x^L, m]; \\ \dfrac{x - x^R}{m - x^R};\ x \in (m, x^R]; \\ 0, \text{ в остальных случаях.} \end{cases}$$

Определение 3. Эквивалентным представлением числа $\tilde{A}$ является тройка $\langle a, m, b\rangle$, где m — мода числа, а $a, b \geqslant 0$ — числа, называемые левым и правым коэффициентами нечеткости, причем $a = m - x^L$, $b = x^R - m$.

Определение 4. Нечетким числом *LL*/*RR*-типа будем называть треугольное нечеткое число, у которого отсутствует правая/левая ветвь функции принадлежности. Другими словами, у нечеткого числа *LL*/*RR*-типа один из коэффициентов нечеткости равен нулю.

При использовании α-уровневого представления число $\tilde{A}$ определяется как объединение четких α-интервалов, границы которых являются функциями параметра $\alpha \in [0; 1]$:

$$\begin{cases} x^L(\alpha) = m - a + a\alpha \\ x^R(\alpha) = m + b - b\alpha \end{cases}. \quad (1)$$

Данное представление сохраняет неопределенность в интервальной форме. Для перехода к полной определенности можно выбирать точку внутри α-интервала $\bar{x}(\alpha) \in X_\alpha$. Такой выбор четкого значения позволяет в задаче $\tilde{Y} = f(\tilde{X})$ с нечеткими параметрами решать на каждом α-срезе четкую задачу и получать решения $y(\alpha)$, совокупность которых рассматривается как дискретное нечеткое решение с функцией принадлежности $\tilde{Y} = \{y(\alpha) | \mu_{\tilde{Y}}(y) = \alpha\}$.

Очевидно, что на итоговый результат будет влиять выбор значения $\bar{x}(\alpha) \in X_\alpha$. Выбор данного значения предлагается осуществлять с помощью преобразования *L*:

$$\bar{x}(\alpha) = L(X_\alpha) = \lambda x^L(\alpha) + (1 - \lambda) x^R(\alpha). \quad (2)$$

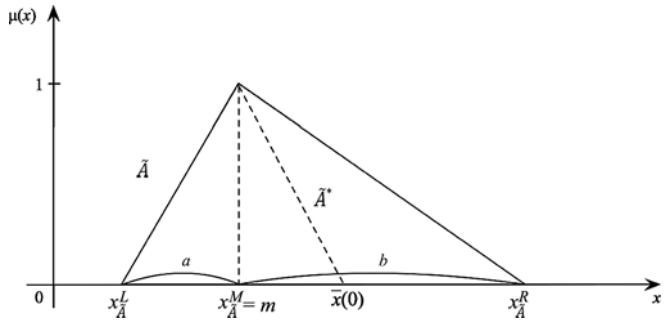

Рис. 1. Исходное нечеткое число $\tilde{A}$ и его модификация $\tilde{A}^*$

Параметр $\lambda \in [0; 1]$ в формуле (2) выбирается для каждого нечеткого числа индивидуально. Влияние λ на свойства и результаты преобразования *L*, а также наиболее важные его значения исследуются в следующих разделах данной статьи.

В таком случае α-уровневые решения будут иметь вид

$$y(\alpha) = f(\bar{x}(\alpha)) = f(L(X_\alpha)). \quad (3)$$

Определение 5. Совокупность решений (3) для различных значений α будем называть модифицированным решением задачи $\tilde{Y} = f(\tilde{X})$.

Определение 6. По аналогии, число $\tilde{A}^*$, функция принадлежности которого $\mu_{\tilde{A}^*}(x) = (\bar{x}(\alpha))^{-1}$, будем называть модифицированным нечетким числом. Очевидно, что число $\tilde{A}^*$ является числом *LL*/*RR*-типа, поскольку у него есть только одна ветвь функции принадлежности. В дальнейшем для обозначения модифицированных чисел будем использовать обозначение $\bar{x}(\alpha)$, которое указывает на механизм построения модифицированных нечетких чисел как совокупности точек на каждом из выбранных α-интервалов.

Пример нечеткого и модифицированного нечеткого чисел приведен на рис. 1.

Очевидно, что введенное преобразование сокращает информативность исходной нечеткой величины. Насколько существенно это сокращение с точки зрения принятия решений, должно показать исследование свойств преобразования *L*.

2. Свойства преобразования *L*

Для изучения свойств предлагаемого преобразования введем характеристические показатели нечеткого числа, которые определяют его информативность с точки зрения принятия решений [5]:

- длина носителя $d_{\tilde{A}}$;
- мода $m_{\tilde{A}}$;
- степень асимметрии *AS*.

Определение 7. Степенью асимметрии *AS* будем называть характеристику нечеткого числа, определяемую как разность площадей прямоугольных треугольников, на которые исходное нечеткое число делится модой (рис. 2):

$$AS = \frac{b - a}{2} \in \left[-\frac{a}{2}; \frac{b}{2}\right].$$

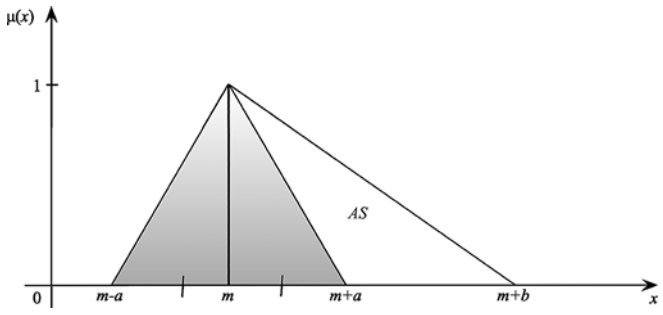

Рис. 2. Геометрический смысл степени асимметрии *AS*

Очевидно, что запись нечеткого числа в виде тройки $\langle m_{\tilde{A}}, d_{\tilde{A}}, AS_{\tilde{A}}\rangle$ эквивалентна описанным ранее формам записи. Ниже перечислены свойства преобразования L, описанные в работах [5, 6], с их доказательствами.

Свойство 1. $\forall\lambda(\alpha)$: $m_{\tilde{A}} = m_{\tilde{A}*}$, т. е. применение L сохраняет моду нечеткого числа. Кроме того, $\forall\lambda = \text{const}$; $\lambda \in [0; 1]$ модифицированное нечеткое число относится к типу LL/RR-чисел.

Доказательство. Сначала докажем, что модифицированное число сохраняет моду исходного нечеткого числа. Перепишем (1) с учетом следующего равенства:

$$\begin{bmatrix} x^L(\alpha) = x_A^L + \alpha\left(m_A - x_A^L\right) \\ x^R(\alpha) = x_A^R + \alpha\left(m_A - x_A^R\right) \end{bmatrix}.$$

Учитывая данные соотношения и подставляя $\alpha = 1$ в (3), получаем:

$$x_{\tilde{A}*}^M = \bar{x}(1) = \lambda x^L(1) + (1-\lambda)x^R(1) = \\ = \lambda m_A + (1-\lambda)m_A = m_A,$$

т. е. моды модифицированного числа и исходного числа совпадают при любых значениях параметров отображения (2).

Доказательство второго утверждения основывается на линейном характере зависимости $\bar{x}(\alpha)$; принадлежность к классу LL/RR-чисел обеспечивается механизмом построения модифицированного нечеткого числа — функция принадлежности $\mu_{\tilde{A}*}(x)$ будет иметь только правую или левую ветку.

Свойство 2. $\exists\lambda \in [0; 1]$: $\text{sign}(AS_{\tilde{A}}) = \text{sign}(AS_{\tilde{A}*})$, т. е. при некоторых значениях λ преобразование L сохраняет знак степени асимметрии. Кроме того, $\exists\lambda \in [0; 1]$: $AS_{\tilde{A}} = AS_{\tilde{A}*}$, т. е. преобразование L при определенных значениях λ сохраняет значение степени асимметрии.

Доказательство. Докажем вначале первое утверждение. Степень асимметрии исходного числа

$$AS_{\tilde{A}} = \frac{b-a}{2}.$$

Абсолютная величина степени асимметрии модифицированного числа равна

$$|AS_{\tilde{A}*}| = \frac{|m - \bar{x}(0)|}{2}. \qquad (4)$$

Поскольку $\bar{x}(0) = \lambda x^L(0) + (1-\lambda)x^R(0) = m + b - \lambda(a+b)$, выражение (4) преобразовывается к виду

$$|AS_{\tilde{A}*}| = \frac{|b - \lambda(a+b)|}{2}.$$

В простейшем случае, когда $a = b$, степень асимметрии исходного числа равна нулю. В этом случае равенство степеней асимметрии достигается при $\lambda = \frac{1}{2}$.

Если $a > b$, то $AS_{\tilde{A}} < 0$, и для выполнения первой части свойства необходимо и достаточно, чтобы $\bar{x}(0) < m$. Отсюда $\bar{x}(0) - m < 0$, т. е. $b - \lambda(a+b) < 0$. В результате

$$\lambda \in \left(\frac{b}{a+b}; 1\right].$$

Случай $a < b$ рассматривается аналогично и приводит к значениям

$$\lambda \in \left[0; \frac{b}{a+b}\right).$$

Докажем теперь второе утверждение. Пусть, для определенности, у исходного числа $a > b$, тогда $AS_{\tilde{A}} < 0$, и модифицированное число является LL-числом. Отсюда следует, что

$$AS_{\tilde{A}*} = \frac{\bar{x}(0) - m}{2} = \frac{b - \lambda(a+b)}{2} < 0.$$

Решая уравнение $AS_{\tilde{A}} = AS_{\tilde{A}*}$ относительно λ, получаем

$$\lambda = \frac{a}{a+b}. \qquad (5)$$

Случай $a < b$ рассматривается аналогично и также приводит к значению λ, полученному в (5).

Таким образом, при $\lambda = \frac{a}{a+b}$ значение степени асимметрии числа сохраняется.

Свойство 3. $\forall\lambda(\alpha)$: $A_\alpha^* \subset A_\alpha$; $d_{\tilde{A}} \geqslant d_{\tilde{A}*}$, т. е. преобразование L уменьшает длину носителя нечеткого числа и оставляет α-интервалы модифицированного числа в рамках исходных α-интервалов. Другими словами, модифицированное число всегда содержится внутри исходного числа.

Доказательство. Вначале покажем, что $A_\alpha^* \subset A_\alpha$. Очевидно, что $A_\alpha^* = [\bar{x}(\alpha); m_{\tilde{A}}]$ для LL-числа ($[m_{\tilde{A}}; \bar{x}(\alpha)]$ для RR-числа). В силу определения нечеткого числа

$$\forall\lambda \in [0; 1]\ x_{\tilde{A}}^L(\alpha) \leqslant m_{\tilde{A}} \leqslant x_{\tilde{A}}^R(\alpha). \qquad (6)$$

Кроме того, из выражения (2) следует, что

$$\forall\alpha \in [0;\ 1]\ x_{\tilde{A}}^{L}(\alpha) \leqslant \bar{x}(\alpha) \leqslant x_{\tilde{A}}^{R}(\alpha). \qquad (7)$$

Исходя из (6) и (7), $[x_{\tilde{A}}^{L}(\alpha);\ x_{\tilde{A}}^{R}(\alpha)] \supset [x_{\tilde{A}}^{M};\ \bar{x}(\alpha)]$ для LL-числа ($[x_{\tilde{A}}^{L}(\alpha);\ x_{\tilde{A}}^{R}(\alpha)] \supset [\bar{x}(\alpha);\ x_{\tilde{A}}^{M}]$ для RR-числа).

Для доказательства первого утверждения достаточно показать, что все возможные значения $\lambda \in [0;\ 1]$ являются решениями неравенства $d_{\tilde{A}^*} \leqslant d_{\tilde{A}}$. Очевидно, что

$$d_{\tilde{A}^*} = |\bar{x}(0) - m_{\tilde{A}}| = |b - \lambda(a + b)|;$$

$$d_{\tilde{A}} = b + a.$$

Имеем неравенство $|b - \lambda(a + b)| \leqslant a + b$, решением которого является отрезок $\left[-\frac{a}{a+b};\ 1 + \frac{b}{a+b}\right] \supset [0;\ 1]$.

Из доказанных выше свойств следует, что применение преобразования L к нечетким исходным данным в основном сохраняет их информативность при целенаправленном выборе параметра преобразования. Уменьшение длины носителя в задачах управления при определенных оговорках можно рассматривать как положительное явление.

Из приведенных выше свойств вытекает несколько важных следствий.

Следствие 1. Если в нечетком числе $\tilde{A}$ $a = b$ и $\lambda = 0{,}5$, то модифицированное число является нечетким синглтоном (т. е. нечетким представлением обычного действительного четкого числа m).

Действительно,

$$\bar{x}(\alpha) = \frac{1}{2}(m - a + a\alpha) + \frac{1}{2}(m + b - b\alpha) =$$

$$= \frac{1}{2}(m - a + a\alpha + m + b - b\alpha) = m.$$

Это следствие накладывает ограничения на возможность использования в задачах управления симметричных нечетких чисел, поскольку все нечеткие вычисления с использованием L-преобразования будут сведены к операциям над модами нечетких чисел.

Следствие 2. Применение преобразования L с параметром $\lambda = \frac{a}{a+b}$ к числу LL/RR-типа не изменяет его.

В самом деле, в случае $b = 0$ (LL-число) $x_{\tilde{A}}^{R}(\alpha) = m$, $\lambda = \frac{a}{a+0} = 1$, и поэтому $\bar{x}(\alpha) = \frac{a}{a}(m - a + a\alpha) + \left(1 - \frac{a}{a}\right)m = m - a + a\alpha = x_{\tilde{A}}^{L}(\alpha)$. Для RR-числа доказательство аналогичное.

Стоит отдельно выделить несколько интересных значений λ:

- при $\lambda = \frac{a}{a+b} = \frac{a}{d_{\tilde{A}}}$ степень асимметрии числа сохраняется, т. е. $AS_{\tilde{A}^*} = AS_{\tilde{A}}$, причем значение $AS_{\tilde{A}}$ численно равно площади модифицированного числа;
- при $\lambda = \frac{b}{a+b} = \frac{b}{d_{\tilde{A}}}$ применение преобразования L равносильно выбору моды на каждом α-уровне и полностью уничтожает информацию, заложенную экспертом в нечеткое число, т. е. справедливо $\bar{x}(\alpha) = m$; $\alpha \in [0;\ 1]$;
- при $\lambda = \frac{b}{d_{\tilde{A}}} - \frac{b-a}{3(b+a)} = \frac{2b+a}{3(b+a)} = \frac{2b+a}{3d_{\tilde{A}}}$ значение $\bar{x}(\alpha)$ является проекцией центра тяжести треугольника, построенного на отрезке $[x^{L}(\alpha),\ x^{R}(\alpha)]$ как на основании, на ось Ox.

3. Определение операций сложения и умножения для *LL/RR*-чисел

Поставим задачу построить алгебру $P = \langle K;\ +,\ ^*\rangle$ на множестве $K = \{\bar{x}(\alpha)\}$; $\alpha \in [0;\ 1]$ нечетких LL/RR-чисел по аналогии с тем, как это сделано в работе [7]. Для удобства в дальнейшем будем использовать следующую форму записи для $\bar{x}(\alpha)$:

$$\bar{x}(\alpha) = c + k\alpha, \qquad (8)$$

где

$$c = m + b - \lambda(a + b);\ \ k = \lambda(a + b) - b.$$

Введем на множестве K бинарную операцию сложения следующим образом:

$$\bar{x}_1(\alpha) + \bar{x}_2(\alpha) = r_1(\alpha) = c_1 + c_2 + (k_1 + k_2)\alpha;\ r_1(\alpha) \in K. \qquad (9)$$

Операция (9) коммутативна

$$\bar{x}_1(\alpha) + \bar{x}_2(\alpha) = c_1 + c_2 + (k_1 + k_2)\alpha =$$
$$= c_2 + c_1 + (k_2 + k_1)\alpha = \bar{x}_2(\alpha) + \bar{x}_1(\alpha)$$

и ассоциативна

$$\bar{x}_1(\alpha) + (\bar{x}_2(\alpha) + \bar{x}_3(\alpha)) =$$
$$= c_1 + k_1\alpha + (c_2 + c_3 + (k_2 + k_3)\alpha) =$$
$$= (c_1 + c_2 + (k_1 + k_2)\alpha) + c_3 + k_3\alpha =$$
$$(\bar{x}_1(\alpha) + \bar{x}_2(\alpha)) + \bar{x}_3(\alpha).$$

Определим единственный нулевой элемент

$$\bar{0} = (0 + 0\alpha) \in K$$

такой, что

$$\forall \bar{x}(\alpha) \in K\colon\ \bar{x}(\alpha) + \bar{0} = c + k\alpha + 0 + 0\alpha = \bar{x}(\alpha).$$

Введем на множестве K бинарную операцию умножения. Эту операцию можно было бы определить как обычное перемножение компонент нечеткого числа

$$\bar{x}_1(\alpha)\bar{x}_2(\alpha) = r_2'(\alpha) = c_1c_2 + c_1k_2\alpha + c_2k_1\alpha + k_1k_2\alpha^2. \qquad (10)$$

Однако такое определение влечет искажение треугольного вида нечеткого числа за счет появления в (10) слагаемого с α^2, а следовательно, результат $r_2'(\alpha) \notin K$. Устранить такую ситуацию можно с помощью линейной интерполяции зависимости $r_2'(\alpha)$ по значениям в двух точках — при $\alpha = 0$ $r_2'(\alpha) = a_1a_2$, при $\alpha = 1$ $r_2'(\alpha) = (a_1 + a_2)(b_1 + b_2)$. Уравнение прямой, проведенной

через эти две точки, позволяет определить операцию умножения следующим образом:

$$\bar{x}_1(\alpha)\bar{x}_2(\alpha) = r_2(\alpha) = \\ = c_1c_2 + (c_1k_2 + c_2k_1 + k_1k_2)\alpha;\ r_2(\alpha) \in K. \qquad (11)$$

Нетрудно убедиться, что операция (11) коммутативна и ассоциативна. Доказательства этих свойств следуют из определения операции умножения и аналогичны доказательствам свойств операций сложения.

Умножение нечеткого числа на скаляр β является частным случаем (11), так как скаляр представляется в виде нечеткого синглтона

$$\bar{\beta} = \beta + 0\alpha.$$

Определим единичное число

$$\bar{1} = 1 + 0\alpha$$

такое, что

$$\forall x(\alpha) \in K\colon \bar{1}\bar{x}(\alpha) = (1 + 0\alpha)(c + k\alpha) = \bar{x}(\alpha).$$

Убедимся, что операция умножения дистрибутивна относительно операции сложения, т. е. справедливо равенство

$$\bar{x}_1(\alpha)\bar{x}_2(\alpha) + \bar{x}_1(\alpha)\bar{x}_3(\alpha) = \bar{x}_1(\alpha)(\bar{x}_2(\alpha) + \bar{x}_3(\alpha)). \qquad (12)$$

Выполним действия в левой и правой частях равенства (12) и сравним полученные результаты:

$$(c_1 + k_1\alpha)(c_2 + k_2\alpha) + (c_1 + k_1\alpha)(c_3 + k_3\alpha) = \\ = c_1c_2 + (c_1k_2 + c_2k_1 + k_1k_2)\alpha + c_1c_3 + \\ + (c_1k_3 + c_3k_1 + k_1k_3)\alpha.$$

$$(c_1 + k_1\alpha)(c_2 + k_2\alpha + c_3 + k_3\alpha) = \\ = (c_1 + k_1\alpha)(c_2 + c_3 + (k_2 + k_3)\alpha) = \\ = c_1c_2 + c_1c_3 + (c_1k_2 + c_2k_1 + k_1k_2)\alpha + \\ + (c_1k_3 + c_3k_1 + k_1k_3)\alpha.$$

Сравнение слагаемых в первом и втором результатах подтверждает равенство (12), т. е. дистрибутивность операции умножения относительно операции сложения.

4. Определение операций вычитания и деления *LL*/*RR*-чисел

Интервальная алгебра, лежащая в основе определения арифметических операций с помощью α-уровневого принципа обобщения, определяет операции вычитания и деления как самостоятельные операции [8]. При этом, в общем случае, интервальные числа не обладают свойством дистрибутивности умножения относительно сложения; отсутствуют понятия противоположного элемента (по сложению) и обратного элемента (по умножению). Это приводит к искажению важных математических соотношений. Так, вычитание из интервального числа равного ему в общем случае не приводит к нуль-интервалу, как и деление интервального числа на равное ему не дает единицы интервала [2]. Следствием этого является невозможность предметной интерпретации математических моделей, используемых при решении задач выбора и управления. Например, решение балансового уравнения, полученное в ходе нечетких вычислительных операций, становится неадекватным реально существующему балансу. В связи с этим определение противоположного и обратного элементов и их использование для реализации операций вычитания и деления может оказаться целесообразным.

Для каждого элемента $\bar{x}(\alpha) = c + k\alpha \in K$ определим единственный противоположный элемент

$$-\bar{x}(\alpha) = -c - k\alpha$$

такой, что

$$\bar{x}(\alpha) + (-\bar{x}(\alpha)) = c + k\alpha - c - k\alpha = \bar{0}.$$

Несколько сложнее определение обратного элемента $\bar{x}^{-1}(\alpha) \in K$ такого, что

$$\bar{x}(\alpha)\bar{x}^{-1}(\alpha) = \bar{1}. \qquad (13)$$

Поскольку предполагается, что $\bar{x}^{-1}(\alpha) \in K$, то $\bar{x}^{-1}(\alpha)$ должен иметь структуру $\bar{x}^{-1}(\alpha) = c' + k'\alpha$. Очевидно, что первое слагаемое обратного элемента должно иметь вид $c' = c^{-1} = \frac{1}{c}$; $c \neq 0$. Тогда

$$\bar{x}(\alpha)\bar{x}^{-1}(\alpha) = (c + k\alpha)\left(\frac{1}{c} + k'\alpha\right) = \\ = 1 + \left(ck' + \frac{k}{c} + kk'\right)\alpha.$$

Для выполнения (13) необходимо и достаточно, чтобы

$$ck' + \frac{k}{c} + kk' = 0.$$

Отсюда

$$k' = \frac{-k}{c(k + c)}.$$

Таким образом, обратный элемент вводится в виде

$$\bar{x}^{-1}(\alpha) = \frac{1}{c} - \frac{k}{c(k + c)}\alpha, \quad c \neq 0. \qquad (14)$$

Очевидно, что для существования обратного элемента необходимо, чтобы дополнительно выполнялось условие $c + k \neq 0$, т. е. исходное число должно иметь ненулевую моду.

Полученные обратные элементы позволяют ввести операции вычитания и деления *LL*/*RR*-чисел как операции сложения и умножения с соответствующими противоположными и обратными числами.

5. Примеры вычислений с использованием преобразования *L*

Проиллюстрируем предлагаемую методику на нескольких примерах, условия которых взяты из работы [9]. Во всех примерах используется представление нечеткого числа в виде тройки $\langle x^L, m, x^R\rangle$.

1. Решение линейного уравнения вида $\tilde{A}x = \tilde{B}$.

Пусть $\tilde{A} = \langle 2; 3; 5\rangle$, $\tilde{B} = \langle 0; 4; 5\rangle$. После применения преобразования *L* решение уравнения будет иметь вид

$$x(\alpha) = \frac{x_{\tilde{B}}(\alpha)}{x_{\tilde{A}}(\alpha)}. \qquad (15)$$

Поскольку операция деления для преобразованных нечетких чисел вводится как умножение на обратное число, перепишем (15) в виде

$$x(\alpha) = \bar{x}_{\tilde{B}}(\alpha)\bar{x}_{\tilde{A}}^{-1}(\alpha). \quad (16)$$

Для числа $\tilde{A}$

$$\begin{cases} x_{\tilde{A}}^{L}(\alpha) = \alpha + 2 \\ x_{\tilde{A}}^{R}(\alpha) = 5 - 2\alpha \end{cases},$$

и поэтому

$$\bar{x}_{\tilde{A}}(\alpha) = \lambda_{\tilde{A}}(2 + \alpha) + (1 - \lambda_{\tilde{A}})(5 - 2\alpha). \quad (17)$$

Аналогично, для $\tilde{B}$ имеем

$$\begin{cases} x_{\tilde{B}}^{L}(\alpha) = 4\alpha \\ x_{\tilde{B}}^{R}(\alpha) = 5 - \alpha \end{cases},$$

откуда

$$\bar{x}_{\tilde{B}}(\alpha) = \lambda_{\tilde{B}}4\alpha + (1 - \lambda_{\tilde{B}})(5 - \alpha). \quad (18)$$

Выбирая значения $\lambda_{\tilde{A}}$ и $\lambda_{\tilde{B}}$ в соответствии с формулой (5), получаем

$$\lambda_{\tilde{A}} = \frac{1}{3};\ \lambda_{\tilde{B}} = \frac{4}{5}. \quad (19)$$

Подставляя значения (19) в (17), (18) и преобразовывая выражения к виду (8), в результате имеем

$$\begin{cases} \bar{x}_{\tilde{A}}(\alpha) = 4 - \alpha \\ \bar{x}_{\tilde{B}}(\alpha) = 3\alpha + 1 \end{cases}. \quad (20)$$

Обратный элемент $\bar{x}_{\tilde{A}}^{-1}(\alpha)$, согласно (14), равен

$$\bar{x}_{\tilde{A}}^{-1}(\alpha) = \frac{1}{4} + \frac{1}{12}\alpha. \quad (21)$$

Пользуясь значениями из (20), (21) и подставляя их в (16) с учетом формулы (11), окончательно получаем

$$x(\alpha) = (1 + 3\alpha)\left(\frac{1}{4} + \frac{1}{12}\alpha\right) = \frac{1}{4} + \frac{13}{12}\alpha. \quad (22)$$

Носителем модифицированного решения является отрезок $\left[\frac{1}{4}; \frac{4}{3}\right]$. Функция принадлежности модифицированного решения определяется как обратная к (22) — $\mu_{x(\alpha)}(x) = \frac{12}{13}x - \frac{3}{13}$. Нетрудно проверить, что моды модифицированного и полного (полученного с использованием принципа обобщения Заде) решений совпадают.

Подстановка решения задачи в исходное уравнение $\tilde{A}x = \tilde{B}$ с учетом формулы (11) приводит к верному равенству:

$$(4 - \alpha)\left(\frac{1}{4} + \frac{13}{12}\alpha\right) = 3\alpha + 1;$$

$$\frac{52 - 13 - 3}{12}\alpha + 1 = 3\alpha + 1.$$

2. Решение задачи линейного программирования с нечеткими коэффициентами.

Пусть дана следующая задача

$$\begin{cases} \tilde{A}_1x_1 + \tilde{A}_2x_2 \to \max \\ x_1 + x_2 = q \\ x_1, x_2 \geqslant 0, \end{cases} \quad (23)$$

в которой $q = 10$, $\tilde{A}_1 = \langle 0; 4; 5\rangle$, $\tilde{A}_2 = \langle 2; 3; 5\rangle$. Применяя преобразование L к параметрам $\tilde{A}_1$ и $\tilde{A}_2$, получаем:

$$x_{\tilde{A}_1}(\alpha) = 3\alpha + 1;\ \lambda_{\tilde{A}_1} = \frac{4}{5}; \quad (24)$$

$$x_{\tilde{A}_2}(\alpha) = 4 - \alpha;\ \lambda_{\tilde{A}_2} = \frac{1}{3}. \quad (25)$$

Задача (23) с использованием результатов (24) и (25) преобразования L выглядит следующим образом

$$\begin{cases} (3\alpha + 1)x_1 + (4 - \alpha)x_2 \to \max \\ x_1 + x_2 = 10 \\ x_1, x_2 \geqslant 0. \end{cases} \quad (26)$$

(26) решается отдельно на каждом α-уровне. Например, для $\alpha = 0{,}5$ задача принимает вид

$$\begin{cases} \frac{5}{2}x_1 + \frac{7}{2}x_2 \to \max \\ x_1 + x_2 = 10 \\ x_1, x_2 \geqslant 0. \end{cases} \quad (27)$$

Решение задачи (27) — вектор [0; 10]. Для $\alpha = 1$ задача (26) сводится к системе

$$\begin{cases} 4x_1 + 3x_2 \to \max \\ x_1 + x_2 = 10 \\ x_1, x_2 \geqslant 0, \end{cases} \quad (28)$$

решение которой — вектор [10; 0].

Объединяя решения подзадач (27) и (28), получим модифицированное решение исходной задачи (23) {[0; 10] | 0,5; [10; 0] | 1}.

Заключение

Результаты исследования показали, что предложенное преобразование L для треугольных нечетких чисел позволяет перейти в поле P относительно сложения и умножения и преодолеть основные недостатки алгебр LR-чисел. Для L доказаны важные свойства, а также показано, что преобразование в основном сохраняет информацию, заложенную в нечеткое число экспертом. В процессе доказательства свойств преобразования также показано, что существующие методы выбора четкого значения на альфа-интервале — среднее арифметическое концов, выбор моды числа, метод центра тяжести — являются частными случаями преобразования L для разных значений коэффициента λ. На множестве K результатов L-преобразования определяются коммутативные и ассоциативные операции сложения и умножения, причем умножение дистрибутивно относительно сложения, а также

единственные нулевой и единичный элементы. С помощью понятий противоположного и обратного элементов вводятся операции вычитания и деления нечетких чисел. Таким образом, множество *LL*/*RR*-чисел можно рассматривать как линейное пространство над полем *P*, что открывает дальнейшие перспективы для применения преобразования *L* и развития вычислительных операций с элементами множества *K*.

Преобразование *L*, вкупе с вводимой алгеброй для модифицированных чисел *LL*/*RR*-типа, обеспечивает выполнение естественных математических отношений и сохраняет форму нечеткого числа. Как показано в примерах, вводимая алгебра для модифицированных чисел может обеспечить решение задач в случаях, когда традиционные подходы не работают вследствие ограничений на знаки характеристик нечеткого числа. При этом простота рассматриваемых примеров не ограничивает универсальности предлагаемой методики, что очень важно в задачах оптимального управления и планирования на предприятиях.

Список литературы

1. **Евдокимов А. В.** Метод нечеткой линеаризации для численного решения алгебраических и дифференциальных уравнений // Электронный журнал "Исследовано в России". 2003, № 6. URL: http://zhurnal.ape.relarn.ru/articles/2003/168.pdf (дата обращения: 27.05.2014).

2. **Соколов А. М.** Методы и алгоритмы нечеткого моделирования механических систем // Информационные технологии. 2007. № 3. С. 13—20.

3. **Усков А. А., Сургучева И. В., Горбунов А. М.** Анализ систем обработки информации и управления с помощью групповых нечетких чисел / // Программные продукты и системы. 2009. № 3. С. 19—21.

4. **Кофман А.** Введение в теорию нечетких множеств: Пер. с франц. М.: Радио и связь, 1982. 432 с.

5. **Воронцов Я. А., Матвеев М. Г.** Исследование свойств линейного отображения в задачах с нечеткими параметрами // Информатика: проблемы, методология, технологии: материалы XIII Международной научно-методической конференции. Воронеж: издательско-полиграфический центр Воронеж. гос. ун-та, 2013. Т. 1. С. 298—304.

6. **Воронцов Я. А., Матвеев М. Г.** Влияние преобразования *L* на результаты арифметических операций с нечеткими *LR*-числами // Современные технологии в задачах управления, автоматики и обработки информации: сборник трудов XXII Международного научно-технического семинара (Алушта, 18—24 сентября 2013 г.). М.: Изд-во МГУПИ, 2013. С. 14—15.

7. **Яхъяева Г.** Основы теории нечетких множеств. Лекция 7: Нечеткие числа и операции над ними // Национальный открытый университет "Интуит". URL: http://www.intuit.ru/studies/courses/87/87/lecture/20511 (дата обращения 27.05.2014)

8. **Аверкин А. Н., Батыршин И. З., Блишун А. Ф. и др.** Нечеткие множества в моделях управления и искусственного интеллекта / под ред. Д. А. Поспелова. М.: Наука, 1986. 311 с.

9. **Лебедев Г. Н., Матвеев М. Г., Семенов М. Е., Канищева О. И.** Методы решения задач управления предприятием в условиях расплывчатой неопределенности // Вестник ВГУ. Серия "Системный анализ и информационные технологии". 2012. № 1. С. 102—106.

Y. A. Vorontsov, Postgraduate Student, e-mail: voroncov_ya@sc.vsu.ru,
M. G. Matveev, Professor, Head of Chair, e-mail: mgmatveev@yandex.ru, Voronezh State University

L-transform Based Algebraic Operations on fuzzy LR-numbers

This paper proposes fuzzy calculations technique which is based on L-transform of α-cuts of fuzzy numbers. It covers the properties of the transform in order to demonstrate applicability of L-transform for fuzzy problems which are solved as a union of crisp problems on each predefined α-level. The paper also introduces an algebra for modified fuzzy numbers and proves the properties of basic algebraic operations, neutral and inverse elements of modified fuzzy numbers set. Usage of these operations is illustrated through some examples. The results of the research show that the proposed L-transform for triangular fuzzy numbers allows to overcome drawbacks of existing algebras of LR-numbers. The transform itself is proven to be a general case of existing methods of selection of point valuation on α-cuts. Since the set K of L-transform results is viewed as a linear space, it allows the further development of L-transform-based calculations for fuzzy numbers in perspective.

Keywords: *fuzzy parameters, α-cut, L-transform, algebra of fuzzy numbers, point valuations, modified fuzzy numbers*

References

1. **Evdokimov A. V.** Metod nechetkoj linearizacii dlja chislennogo reshenija algebraicheskih i differencial'nyh uravnenij. *Jelektronnyj zhurnal "Issledovano v Rossii".* 2003. N. 6. URL: http://zhurnal.ape.relarn.ru/articles/2003/168.pdf (Reference date: 27.05.2014).

2. **Sokolov A. M.** Metody i algoritmy nechjotkogo modelirovanija mehanicheskih sistem. *Informacionnye tehnologii.* 2007. N. 3. P. 13—20.

3. **Uskov A. A. Surgucheva I. V., Gorbunov A. M.** Analiz sistem obrabotki informacii i upravlenija s pomoshh'ju gruppovyh nechetkih chisel. *Programmnye produkty i sistemy*. 2009. N. 3. P. 19—21.

4. **Kofman A.** *Vvedenie v teoriju nechetkih mnozhestv:* Per. s franc. M.: Radio i svjaz', 1982. 432 p.

5. **Vorontsov Y. A.**, **Matveev M. G.** Issledovanie svojstv linejnogo otobrazhenija v zadachah s nechetkimi parametrami. *Informatika: problemy, metodologija, tehnologii: materialy XIII Mezhdunarodnoĭ nauchno-metodicheskoĭ konferencii*. Voronezh: izdatel'sko-poligraficheskij centr Voronezh. gos. un-ta, 2013. V. 1. P. 298—304.

6. **Vorontsov Y. A.**, **Matveev M. G.** Vlijanie preobrazovanija L na rezul'taty arifmeticheskih operacij s nechjotkimi LR-chislami. *Sovremennye tehnologii v zadachah upravlenija, avtomatiki i obrabotki informacii: Sbornik trudov XXII Mezhdunarodnogo nauchno-tehnicheskogo seminara (*Alushta, 18—24 sentjabrja 2013). M.: Izd-vo MGUPI, 2013. P. 14—15.

7. **Jahjaeva G.** Osnovy teorii nechjotkih mnozhestv. Lekcija 7: Nechjotkie chisla i operacii nad nimi. *Nacional'nyj otkrytyj universitet "Intuit"*. URL: http://www.intuit.ru/studies/courses/87/87/lecture/20511 (Reference date 27.05.2014)

8. **Averkin A. N. Batyrshin I. Z., Blishun A. F. i dr.** *Nechetkie mnozhestva v modeljah upravlenija i iskusstvennogo intellekta* / pod red. D. A. Pospelova. M.: Nauka, 1986. 311 p.

9. **Lebedev G. N., Matveev M. G., Semenov M. E., Kanishheva O. I.** Metody reshenija zadach upravlenija predprijatiem v uslovijah rasplyvchatoj neopredeljonnosti. *Vestnik VGU, serija "Sistemnyj analiz i informacionnye tehnologii"*. 2012. N. 1. P. 102—106.

УДК 004.912

К. В. Лунев, студент, программист, МГУ им. М. В. Ломоносова, e-mail: kirilllunev@gmail.com

К вычислению смысловой близости предложений

Представлены результаты начального этапа исследований на направлении поиска моделей, алгоритмов и программных средств для определения семантической близости двух предложений. Результаты исследований могут использоваться в поисковых системах для выдачи более релевантного контента, а также для решения задач кластеризации, обобщения, индексирования текстов и многих других. В предлагаемом автором подходе предполагается, что предложения можно разбить на три части, каждая из которых является описанием некоторого факта, а именно — что произошло, где и когда. Алгоритм разделения предложения на такие части на данном этапе исследования не рассматривается. Предложены метрики, на основе которых определяется смысловая близость частей предложений. Данные о близости частей предложений в свою очередь используют для нахождения семантической схожести целых предложений. Для этого применяют: семантическую сеть WordNet; поисковую систему Яндекс; сервис геокодирования Google Geocoding; собственные алгоритмы.

Ключевые слова: *обработка естественного языка, смысловая близость предложений, алгоритмы, поисковые системы, геокодирование*

Постановка задачи

Вопросы вычисления смысловой близости между предложениями являются очень важными в задачах, связанных с анализом текстовых данных в электронном виде. Примерами таких задач могут служить автоматическое определение и коррекция неправильного употребления слов в тексте; кластеризация; обобщение и индексирование текстов. Определение степени семантической близости предложений может заметно улучшить работу поисковых систем, которые повседневно используют миллионы людей. Эти знания могут быть полезны для переформулирования запросов в целях получения более релевантного ответа и сбора более качественной статистики от пользователей, а также для решения других практически значимых задач.

В настоящее время задача автоматической оценки смысловой близости между предложениями остается нерешенной, как и более общая задача понимания естественного языка, а ее сложность эквивалентна сложности создания систем искусственного интеллекта. Вычисление семантической близости — относительно простая задача для человека, однако очень сложная для решения на ЭВМ. Существует два основных подхода к решению отмеченной задачи. Первый из них основан на базах знаний [3, 6], второй — на использовании корпусов больших размеров [2, 8]. В работах [5, 9] рассмотрено применение гибридных методов. Заметим, что оба этих метода имеют свои слабые стороны. Слабой стороной первого метода является то обстоятельство, что значение слов может меняться при переходе от одной области знаний к другой. Второй метод проигрывает за счет того, что со статистической точки зрения язык содержит большое число относительно редких слов. По этой причине, чтобы редкие слова имели правильную статистику, нужен корпус огромных размеров. В противном случае некоторое редкое слово получит слишком большой вес в силу того, что несколько раз попадет в один конкретный текст из корпуса. В работе, результаты которой представлены далее, в определенной степени используются тот и другой методы, а также следующая интуитивно понятная идея: если каждое из предложений F_1, F_2 разбить на три отдельные части, а именно — что произошло, где и когда, сравнить соответствующие части, а затем на основании результатов этих сравнений вычислить близость целых предложений, то можно получить более точную оценку, чем ту, которая получится в результате сравнения предложений сразу целиком. Таким образом, итоговая функция, которая показывает насколько факты схожи, зависит от функций близости между соответствующими частями:

$$S(F_1, F_2) = f(f_t(what_1, what_2), f_r(where_1, where_2), f_n(when_1, when_2)),$$

где f_t, f_r, f_n — функции сравнения частей "что", "где" и "когда" соответственно, а f — общая функция близости двух предложений.

Преимущество предлагаемого подхода достигается за счет того обстоятельства, что можно использовать и улучшать существующие методы с учетом специфики той или иной части. Например, в части "что" имеется подлежащее и сказуемое (кто и что сделал), в частях "где" и "когда" — слова, определяющие соответственно некоторые географические и временные сведения о представленном в тексте событии.

В настоящей статье предложены функции f_t, f_r, f_n сравнения частей предложения и результирующая функция f, отображающая степень схожести фактов. При этом оговаривается, что разбиение предложения на части на рассматриваемом, начальном этапе исследований считается выполненным и предложения, поступающие для сравнения, проходят через некоторый "черный ящик", который делит его на три части. Для сравнения частей "что" использована семантическая сеть WordNet, описанная в работе [4], с сопутствующей библиотекой для языка C++. Использованы также результаты работ [1, 3, 6]. Словарь в данной семантической базе состоит из четырех сетей для основных частей речи: существительных, глаголов, прилагательных, наречий. Базовой словарной единицей в WordNet является не отдельное слово, а так называемый синонимический ряд (синсеты), объединяющий слова со схожим значением, которые, по сути своей, являются узлами сети. Синсеты в WordNet связаны такими семантическими отношениями, как гипероним, гипоним, мероним, антоним и др. Кроме онтологии используется определитель частей речи (*Part-of-speech tagger*) LAPOS, ставящий рядом с каждым словом его часть речи. Важной составляющей программной реализации алгоритма являются механизмы поисковой системы Яндекс, позволяющие путем использования сервиса Яндекс.XML совершать автоматические поисковые запросы к Яндексу и получать ответы в формате XML. Идеи алгоритмов, основанных на поисковых системах, заимствованы из работы [2].

Для того чтобы сравнить части "где" используется сервис геокодирования Google Geocoding API. Геокодирование — процесс преобразования адресов в географические координаты. Посылая запрос, программа получает некоторые сведения об объекте, основные из которых — географические координаты, тип объекта, полное название. Из этого ответа выбираются необходимые данные и на основании предложенного метода вычисляется семантическая близость.

Алгоритм сравнения частей "когда" использует методы перевода текстовой строки в некоторые метаслова, определяющие временные объекты. Эти методы основаны на работе с регулярными выражениями. При этом текстовая строка может иметь самые разные формы, например: "два месяца назад", "12.05.1992", "в прошлый четверг", "в рождество", "в начале лета" и др.

Вычисление смысловой близости частей "что" в описании двух фактов

Далее рассмотрены основные подходы и алгоритмы, которые используются для предлагаемой автором оценки степени близости части "что" в описании двух фактов. Отметим, что данный алгоритм работает только для предложений на английском языке, потому что для решения задачи используется семантическая сеть WordNet, которая недостаточно развита для других языков.

Функция для оценки смысловой близости слов. Для того чтобы научиться сравнивать по смыслу предложения, которые могут состоять из любых слов, нужно сначала научиться сравнивать слова по смыслу. Для решения этой задачи выбрана семантическая сеть WordNet и алгоритм, предложенный в работе [7]. Синсеты, в базе WordNet составляют дерево, корень которого слово-сущность (*entity*). От него идут узлы — синсеты являющиеся гипонимами для этого слова — физическая сущность (*physical entity*), абстракция (*abstract entity*), вещь (*thing*). В свою очередь эти слова конкретизируются следующими синсетами и т. д. Функция, показывающая близость слов, зависит от длины пути между этими словами в базе. Очевидно, чем больше расстояние по такому дереву нужно пройти от одного слова к другому, тем более "разные" эти слова, т. е. меньше их смысловая близость. Вместе с тем авторы работы [7] отметили, что расстояние между словами *boy* и *animal* меньше, чем расстояние между словами *boy* и *teacher*, хотя интуитивно понятно, что так быть не должно. Чтобы избавиться от этого недостатка, функция близости s слов w_1, w_2, описанная в работе, зависит также от глубины слов h в иерархии базы. Причина в том, что слова на большей глубине более конкретны и более схожи, а слова на меньшей глубине более абстрактны и сильнее различаются. Функция приобретает вид

$$s(w_1, w_2) = f_{length}(l) f_{depth}(h) = \frac{\mathbf{e}^{-\alpha l}\left(\mathbf{e}^{\beta h} - \mathbf{e}^{-\beta h}\right)}{\mathbf{e}^{\beta h} + \mathbf{e}^{-\beta h}},$$

где f_{length}, f_{depth} — функции длины пути между словами и их глубины в иерархии базы соответственно, оптимальные для WordNet параметры α и β: $\alpha = 0{,}2$ и $\beta = 0{,}45$.

Описание используемых алгоритмов. Чтобы избавиться от части отмеченных в описании постановки задачи проблемных вопросов, был реализован следующий алгоритм. Так как части предложений отвечают на вопрос "что произошло?", то в них обязательно должен быть глагол, описывающий действие, а также существительное, показывающее, кто это действие совершил. Для определения частей речи в предложении была использована программа LAPOS. Эта программа показала точность определения частей речи на тестовых данных 97 %. Также она имеет возможность обучаться на размеченных текстах.

После определения частей речи осуществляется отдельное сравнение существительных и глаголов. Из предложения выбирается первая последовательность подряд идущих существительных, а затем из этой последовательности слов выбирается последнее, которое и будет принято подлежащим для данного предложения. Описанные действия проводятся для того, чтобы, например, из словосочетания *football team* взять именно слово *team*, а не слово *football*, которое тоже является существительным. Первый глагол после подлежащего будет искомым глаголом (сказуемым).

Существительные сначала приводятся к начальной форме. Далее происходит поиск этих существительных по базе WordNet. Если в базе слово не найдено, то скорее всего это некоторое название объекта. Для определения, какого именно объекта, используется поисковая система Яндекс. На систему отправляется запрос вида "*X is a*", где *X* — слово, для которого ищется определяющее слово (другими словами, для слова ищется гипероним). Ответ содержит сниппеты, в которые входит предложение запроса и некоторый контекст. Из каждого сниппета выбирается существительное, идущее после "*is a*" (если быть более точным, то выбирается последнее в цепочке существительных по причинам, описанным выше). Таким образом формируются самые часто употребляемые определяющие слова. Для запроса "*CSKA is a*" получены результаты, представленные в табл. 1.

Далее с использованием WordNet происходит поиск гиперонимов, начиная с самого частого и до тех пор, пока гипероним не найдется в базе. Таким образом решаются вопросы, связанные со словами, которые отсутствуют в семантической сети. В табл. 2 приведены результаты работы программы поиска гиперонимов для некоторых слов, отсутствующих в базе WordNet.

Таблица 1

Гиперонимы для слова CSKA

Гипероним	Частота
team	305
power	198
club	195
share	189
problem	81
pleasure	80
teammate	65
crazy	54
cska	34

Таблица 2

Примеры гиперонимов к словам

Слово	Гипероним
megafon	operator
pepsi	drink
ipad	device
iphone	phone
sarov	town
nba	league
ronaldo	player
ruby	language
adata	company

В целом результаты работы программы определения гиперонимов могут квалифицироваться как неплохие. Однако необходимо отметить, что иногда ответ оказывается неправильным. Причина может быть связана с тем обстоятельством, что алгоритм никак не различает разные значения слова, для которого ищется гипероним. Например, по запросу *Jordan* найден гипероним *country*, хотя возможно, из контекста предложения явно видно, что речь идет об известном баскетболисте с фамилией *Jordan*. Следует также отметить, что алгоритм ищет самый популярный результат поиска, например, по запросу *Lunev* было получено *musician*. Хотя, вполне возможно, что имелся в виду совсем другой человек по фамилии *Lunev*, профессия которого никак не связана с музыкой.

После того как существительное найдено в базе, возникает другой проблемный вопрос. Он заключается в том, что очень часто слово упоминается в WordNet в нескольких смыслах и нужно выбрать только один верный (верное значение смысла этого слова). Идея решения этой проблемы многозначности (известной как *word sensedisambiguation*, или WSD) состоит в следующем. Слово многозначно, если оно входит в несколько разных синсетов. В синсет каждого найденного слова, как правило, входят еще несколько слов, неотделимых по смыслу от данного. Будем подменять данное слово на неотделимые слова из некоторого синсета, содержащего слово, и отправлять запрос в поисковую машину. Чем больше результатов получено, тем чаще слово употребляется в данном смысле.

Итак, пусть $S(a)$ — предложение, содержащее слово a на некотором фиксированном месте, a_j^i — j-е слово в синсете i-го значения слова a. Пусть зафиксирован некоторый смысл слова a^p. Для каждого j (т. е. для каждого слова из синсета a^p) отправляется запрос в Яндекс вида $S(a_j^p)$ — т. е. в данном предложении слово a подменяется на слово a_j^p. Пробелы в запросе заменяются на знаки &, чтобы поиск комбинации проходил по предложению, а не по всему документу. Из ответа выбирается число найденных страниц и число, показывающее сколько раз проиндексировано слово a_j^p (*wordstat*). Первое число показывает, как часто предложение встречается с данным значением слова, второе — насколько часто встречается само слово, а их частное отражает частоту встречаемости данного предложения со словом a в смысле a_j^p. Суммируя частные по всем j, получаем частоту встречаемости данного предложения a в смысле p. Такие вычисления проводятся для каждого смысла p и выбирается смысл с наибольшим значением такой суммы.

Отмечается, что если в синсете присутствует только одно слово, то берется строка-определение этого

слова (у каждого слова из WordNet присутствует такая строка) и из него выбирается подлежащее.

Рассмотрим некоторые результаты работы написанной автором программы, реализующей представленный выше алгоритм. В качестве примера взяты два слова — *lap* и *plane*. В базе WordNet для первого указаны шесть смыслов. Для каждого из смыслов было составлено подходящее предложение и с помощью алгоритма решалась задача WSD для слова в контексте данного предложения. Рассмотрим следующие далее различные смыслы слова *lap*:

1) *the upper side of the thighs of a seated person*;
2) *an area of control or responsibility*;
3) *the part of a piece of clothing that covers the thighs*;
4) *a flap that lies over another part*;
5) *movement once around a course*;
6) *touching with the tongue.*

Результаты работы программной реализации алгоритма представлены в табл. 3.

Таким образом, алгоритм был запущен 6 раз и в пяти случаях был дан верный ответ. Для слова указывался именно тот смысл из базы WordNet, который действительно подразумевался в данном предложении.

Аналогичный тест был проведен для слова *plane*. Далее перечислены описания разных смыслов для этого слова:

1) *an aircraft that has a fixed wing and is powered by propellers or jets*;
2) *(mathematics) an unbounded two-dimensional shape*;
3) *a level of existence or development*;
4) *a power tool for smoothing or shaping wood*;
5) *a carpenter's hand tool with an adjustable blade for smoothing or shaping wood*.

Заметим, что результат (табл. 4) оказался противоположным. Верно определен только смысл слова в первом предложении.

Анализ результатов тестирования позволяет предположить, что сказывается содержание контекста. Причина в том, что слова в предложении должны быть из той же предметной области, что и смысл данного слова, и при этом лучше, чтобы как можно меньше употреблялись слова общего назначения. После того как смыслы существительных установлены, их сравнивают при помощи алгоритма сравнения слов.

Таблица 3

Решение задачи WSD для слова lap

Предложение	Смысл реальный/ вычисленный
he picked up girl and plopped her down in his lap	1/5
the job fell right in my lap	2/2
his lap was covered with food stains	3/3
the lap of the shingles should be at least ten inches	4/4
he drove an extra lap	5/5
the dog's laps were warm and wet	6/6

Таблица 4

Решение задачи WSD для слова plane

Предложение	Смысл реальный/ вычисленный
the flight was delayed due to trouble with the plane	1/1
any line joining two points on a plane lies wholly on that plane	2/3
he lived on a worldly plane	3/4
plane is a power tool for smoothing or shaping wood	4/1
the cabinetmaker used a plane for the finish work	5/1

Глаголы также сначала приводятся к начальной форме. После этого практически любой глагол можно найти в базе глаголов WordNet и не возникает вопросов, связанных с именами собственными, которые появляются с существительными. Далее используется описанный ранее алгоритм решения многозначности. Особенность глаголов в том, что они не так сильно связаны в базе WordNet: не существует единого дерева как у существительных, а присутствует только "лес" из деревьев небольшой высоты, поэтому алгоритм сравнения слов неприменим. По этой причине сначала осуществляется попытка преобразовать глагол в существительное, от которого он произошел, средствами функций библиотеки WordNet. Если преобразование закончилось успешно для глаголов из двух предложений, то они сравниваются алгоритмом сравнения слов. Если этого не происходит, то проверяется, лежат ли глаголы в одном дереве. Если они лежат в одном дереве, то выполняется алгоритм сравнения слов, если нет, то смысловая близость между соответствующими глаголами полагается равной нулю.

Таким образом, для пары предложений получены оценки смысловой близости для глаголов и существительных. Эти оценки перемножаются, тем самым получается оценка близости для предложений в целом.

Тестирование части "что". Результаты тестовых испытаний программной реализации алгоритма на примерах нескольких пар предложений, каждая из которых имеет свои особенности, представлены в табл. 5. Эти особенности существенно влияют на точность оценки смысловой близости предложений по части "что".

Пример 1 показывает работу алгоритма в случае, когда глаголы синонимичны и близость предложений фактически считается по связи существительных. Пример 2 демонстрирует — если взять слишком разные по смыслу существительные, то мера близости резко уменьшится. Пример 3 показывает работу алгоритма поиска гиперонима в сети Интернет: слово *CSKA* не присутствует в базе WordNet и оно было заменено на слово *team*. В итоге получены два одинаковых существительных и два синонимичных глагола — результат единица. Пример 4 демонстрирует работу алгоритма, выбирающего смысл слова. По контексту было вычислено, что глаголы имеют разные смыслы:

Таблица 5

Результаты тестовых испытаний

№ примера	Пара предложений	Мера близости
1	*Boy learns mathematics.* *Student discovers mathematics*	0,447
2	*Boy learns mathematics.* *Airplane discovers mathematics*	0,007
3	*CSKA won a cup.* *Football team got a cup*	1,000
4	*Approach has been tried with good results.* *Judge tried killer for murder*	0,000
5	*Camp Fire Girls organized.* *Camp Fire Girls organization announced by Mrs Gulick*	0,008
6	*Central African Republic suspended from African Union.* *The African Union has suspended the Central African Republic*	0,425
7	*Syrian rebels bombard central Damascus, army artillery hits back.* *Syrian rebels fired dozens of mortar bombs into central Damascus on Monday*	0,040
8	*Russia, China must be part of Syria chemical arms inquiry.* *China, Russia must be part of Syria chemical arms inquiry*	1,000
9	*Two killed in north Lebanon in third day of Syria-linked clashes.* *Two people were killed in the northern Lebanese city*	0,215

в первом случае это слово в значении "испытывать", во втором — "судить". Эти глаголы разные, а следовательно, мера равна нулю. В примере 5 показана неправильная работа алгоритма: в первом предложении выделено существительное *Girls*, а во втором — *organization*, их мера близости очень мала. Этот пример также показывает неточность определения подлежащего. В качестве подлежащего всегда берется одно единственное слово, хотя явно видно, что это может быть словосочетание (такое как *Camp Fire Girls*). Предложения примера 6 идентичны, однако в одном из них применяется страдательный залог, а в другом — действительный. По этой причине алгоритм работает неправильно, так как в качестве подлежащих получены разные слова — *Republic* и *Union*. Если бы такая структура предложений применялась к существительным, более далеким друг от друга по смыслу, то результат сравнения был бы еще ниже, чего, конечно, быть не должно. Пример 7 показывает работу алгоритма, когда глаголы можно привести к существительным и сравнить как существительные. В примере 8 подлежащие определены разные — *China* и *Russia*, но тем не менее, на результате работы это не сказалось. Пример 9 — ошибка в обработке первого предложения: в качестве подлежащего выбрано слово *killed*, в базе WordNet такого существительного нет, а поиск гиперонима не привел к хорошему результату — было выбрано слово *bit*.

Вычисление смысловой близости частей "где" в описании двух фактов

Далее в кратком виде представлен предлагаемый алгоритм выделения смысловой близости частей "где" в описании двух фактов, а также результаты тестовых испытаний этого алгоритма.

Описание алгоритма. Для сравнения предложений, в которых описывается место действия, было использовано геокодирование — процесс перевода названий географических объектов в их координаты. Такое преобразование предоставляет сервис GoogleMaps. С использованием механизма XML-запросов можно получить ответ, в котором будут указаны координаты, а также другие свойства и характеристики объекта: тип, размеры, полное название и др. Основное, что потребуется алгоритму для вычисления смысловой близости, это размеры, а если быть более точным, *viewport* — окно, через которое можно просматривать объект. Фактически, *viewport* — это наилучшая прямоугольная область на карте, в которую попадает искомый объект. Наилучшая в том плане, что кроме объекта в этот прямоугольник попадает наименьшее пространство, не принадлежащее этому объекту. Объект получается вписанным в эту область. Идея алгоритма состоит в том, что *viewport* приравнивается к самому объекту, т. е. является его прямоугольной аппроксимацией. Таким образом, географический объект из каждого предложения заменяется на прямоугольник на карте. Каждый прямоугольник двухмерный, задается широтой и долготой. Далее задача сводится к сравнению положения геометрических тел (прямоугольников) на плоскости.

Пример определения прямоугольников представлен на рис. 1, см. третью сторону обложки (большой красный прямоугольник — Нижегородская область, маленький синий — Нижний Новгород). Очевидно, что близость между описаниями тем меньше, чем меньше площадь пересечения областей. Если они не пересекаются, то считается, что факты разные и мера близости равна нулю исходя из утверждения, что два события не могли произойти в разных местах. Если они все-таки пересекаются по большей площади, то мера близости должна быть больше. Вместе с тем оценка близости должна также зависеть от того, насколько велика объединенная площадь областей. Например, пусть указано, что первое событие произошло в доме A, второе событие — в городе B, а третье — в стране C, а также известно, что дом A находится в городе B, а город B — в стране C. Тогда мера близости первого и второго предложений должна быть больше, чем мера близости первого и третьего предложений. Этот факт можно записать следующим образом:

$$P(A|B) > P(A|C).$$

Это выражение означает, что если событие произошло в стране C, то вероятность того, что оно про-

изошло при этом в доме *A* очень мала (страна большая, разных домов много). Если же событие произошло в городе *B*, то вероятность "попасть" в дом *A* становится больше, а вместе с этим растет мера близости соответствующих фактов. Близость предложений, описывающих географическое положение событий, задается формулой

$$f_r(where_1, where_2) = \frac{\log_{10}(\alpha SC + 1)}{\log_{10}(\alpha SU + 1)},$$

где *SC* — площадь пересечения, *SU* — площадь объединения, $\alpha = 10^5$. Параметр α необходим для того, чтобы мера не была слишком маленькой, если площадь пересечения намного меньше площади объединения. Такое часто случается, когда размеры разных географических объектов сильно разнятся.

Тестирование части "где". Основываясь лишь на одной идеи *viewport*, получен простой алгоритм, достаточно хорошо сравнивающий места описания фактов. Результаты работы программной реализации этого алгоритма даны в табл. 6.

Из представленных результатов тестирования следует, что если объекты не пересекаются, то уровень смысловой близости равен нулю. В противном случае, если один объект является частью другого, то уровень смысловой близости отличен от нуля.

Вычисление смысловой близости частей "когда" в описании двух фактов

В этом разделе в конспективной форме описан предлагаемый алгоритм выделения смысловой близости частей "когда" в предложениях, описывающих некоторые факты.

Описание алгоритма. Сложность реализации вычисления степени смысловой близости в данных частях состоит в том, что необходимо уметь работать с датами совершенно произвольных типов. Это могут быть различные даты, указывающие некоторый абсолютный временной интервал, например, "в 2000 году"; "в мае"; "в рождественские каникулы"; "во времена Ельцина" и т. д. Могут также использоваться относительные временные интервалы: "несколько недель назад"; "в прошлом столетии"; "10 лет назад"; "недавно" и др.

Для сравнения близости возникает необходимость вводить некоторые вероятностные функции, показывающие, с какой вероятностью то или иное событие могло произойти в конкретный момент времени, в зависимости от того, как сформулировано предложение. Причем эта функция должна зависеть от типа предложения. Например, фраза "in april, 1999" показывает, что событие произошло в определенное время, а именно — в апреле 1999 г. При этом не сказано, в какой из апрельских дней произошло событие. По этой причине достаточной вероятностной функцией для предложений такого типа будет функция кусочно-постоянная. Она равна некоторому числу α в интервале времени, о котором говорится в предложении, и равна нулю вне этого интервала. Пусть есть другое предложение: "*two years ago*". Слово *ago* показывает, что дата, когда событие произошло, отсчитывается от времени создания сообщения. При этом обычно такие сообщения не обладают точностью. Этот факт означает, что вероятнее всего в предложении говорится не о том, что событие произошло ровно два года назад, а констатируется факт, что оно произошло примерно в это время. Логично, что наибольшей вероятностью должна, все-таки, обладать точка на временном интервале, удаленная ровно на два года от даты написания. Если отходить от этой даты, то вероятность должна уменьшаться.

В основу алгоритма обработки части "где" входит преобразование предложения в структуру данных DateStruct. Основные поля структуры представлены на рис. 2.

Изначально элементы этой структуры считаются неопределенными. Для заполнения структуры предложение сравнивается с регулярными выражениями, описанными в конфигурационном файле formats. Этот файл состоит из предложений (шаблонов), которые включают в себя некоторые слова языка (или языковые конструкции), а также объекты, называемые ключами. Ключом именуется такое метаслово, которое обозначает

Таблица 6

Сравнение предложений, содержащих части «где»

Параметры сравнения	Мера близости					
	г. Саров (Нижегородская обл.)	г. Нижний Новгород	Нижегородская обл.	Ул. Бекетова (г. Нижний Новгород)	Приокский р-н (г. Нижний Новгород)	г. Москва
г. Саров (Нижегородская обл.)	1,000	0,000	0,439	0,000	0,000	0,000
г. Нижний Новгород	0,000	1,000	0,624	0,409	0,828	0,000
Нижегородская обл.	0,439	0,624	1,000	0,255	0,530	0,000
Ул. Бекетова (г. Нижний Новгород)	0,000	0,409	0,255	1,000	0,481	0,000
Приокский р-н (г. Нижний Новгород)	0,000	0,828	0,530	0,481	1,000	0,000
г. Москва	0,000	0,000	0,000	0,000	0,000	1,000

Рис. 2. Структура DateStruct

объект времени (название месяца, день недели, слово "месяц", слово "неделя"), вербальное дополнение, которое обозначает относительный характер времени (две недели *назад*, *конец* года, в *прошлом* месяце, *вчера*); дополнение с использованием числа (или слово, обозначающее некоторую количественную меру), относящееся к времени (*100* лет назад, *12* мая, *несколько* лет назад, *15.01.1992*). Шаблон поддерживает также синтаксис регулярных выражений. Пример строки из конфигурационного файла formats: %END% of %s% %Y%.

Здесь %END% — ключ, обозначающий конец некоторого временного периода; of — слово языка; %s% — ключ, обозначающий сезон (весна, лето, осень, зима); %Y% обозначает год, записанный в формате YYYY. Определение ключей находится в другом конфигурационном файле vocabulary. Под определением ключа понимается набор слов языка, соответствующий данному ключу. Например, для ключа %s% (сезон) определение выглядит следующим образом:

%s%
winter зима зимой
spring весна весной
summer лето летом
autumn осень осенью

Некоторые ключи не нуждаются в определении, например, %INT% %d% — в основном это ключи, обозначающие некоторую количественную характеристику. Для обработки предложения нужно найти соответствующий шаблон в конфигурационном файле formats. Для этого ключи в данном шаблоне заменяются на некоторое выражение, состоящее из скобок, логического оператора "|" и слов определения ключа. В таком представлении шаблон принимает вид регулярного выражения. В этом случае с помощью функций библиотеки boost::regexp можно узнать, соответствует ли данное предложение регулярному выражению. Если такое предложение не соответствует регулярному выражению, то берется следующая строка конфигурационного файла formats и разбор проводится заново для этой строки. Если предложение соответствует регулярному выражению, то из него извлекаются необходимые слова, которые входят в определения ключей. В дополнение к изложенному отмечается, какие ключи были задействованы.

Зная, какие слова из каких ключей присутствуют в предложении, можно заполнять поля структуры DateStruct. В программной реализации алгоритма для каждого ключа присутствует обрабатывающая его функция. Обработка ключей — это заполнение структуры в зависимости от названия ключа и найденного слова, соответствующего этому ключу. Следует отметить, что полей в структуре обычно больше, чем ключей в шаблоне, поэтому многие поля остаются неинициализированными. В этом случае они заполняются наиболее вероятными значениями. Например, если в предложении указано число и месяц, то год определяется как ближайший к году создания сообщения. Рассмотрим обработку ключей на примере шаблона %INT% %DATE_UNIT% %AGO% и предложения "3 *years before*". Сначала в шаблонах отыскиваются все ключи:

%INT% — ключ целого числа;

%DATE _ UNIT % — ключ единицы времени;

%AGO% — ключ, показывающий что событие произошло в прошлом.

Далее из конфигурационного файла vocabulary берутся определения найденных ключей. Определения имеют ключи %DATE_UNIT% и %AGO%. Эти ключи имеют следующий вид:

%DATE_UNIT%
day days день дней
week weeks недель неделя
month months месяц месяцев
year years лет год
%AGO%
ago before назад ранее до

Ключ %INT% не нуждается в определении в конфигурационном файле. Далее шаблон заменяется на следующее регулярное выражение:

.*(d{1, })(day | days | день | дней | week | weeks | недель | неделя | month | months | месяц | месяцев | year | years | год | года | лет)(ago | before | назад | ранее | до этого).*

После того как все ключи обработаны и поля структуры DateStruct заполнены, включается алгоритм обработки структуры. В этот момент структура переводится в точную дату (наиболее вероятное время совершения события), а также временной интервал (возможное время совершения события). Эти два параметра и являются основными объектами, которые необходимо создать из данного предложения. Исходя из состава ключей, присутствующих в шаблоне, выбирается тип предложения, а именно — CENTR_WEIGHT или CONST_WEIGHT. Выражение CONST_WEIGHT соответствует типу предложения, в котором вероятность совершения события не меняется в зависимости от дня (например, "на прошлой

Рис. 3. Пример функции типа CONST_WEIGHT

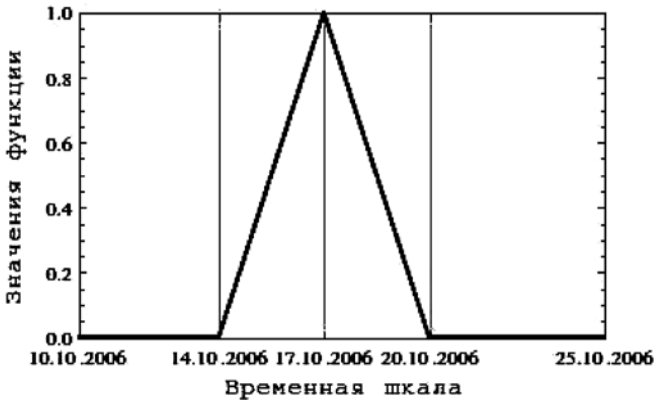

Рис. 4. Пример функции типа CENTR_WEIGHT

неделе" — любой день недели, предшествующей дате создания сообщения, равновероятен). Фактически, точно предсказанная дата совершения события в этом случае становится ненужной. Тип CENTR_WEIGHT, наоборот, указывает на то, что точная дата имеет наибольшую вероятность совершения события. Вместе с тем чем дальше день отходит от точной даты, тем меньше вероятность того, что событие произошло в этот день. Со знанием точной даты и возможного интервала двух предложений представляется возможность определить степень близости этих предложений. Важно заметить, что дата создания сообщения считается известной и все относительные даты отсчитываются именно от нее.

Таким образом, текстовые предложения переводятся в конкретные отрезки на временной шкале. Остается лишь сравнить такие отрезки для разных предложений.

Алгоритм сравнения для разных предложений. Пусть имеются два предложения для сравнения. После их обработки, на выходе алгоритма вместо каждого предложения образуется точная дата и возможный временной интервал. Для сравнения точная дата, возможный интервал и тип преобразуются в функцию распределения следующим образом. Если предложение имеет тип CONST_WEIGHT, то значение функции в каждой точке возможного интервала равно 0,5 (рис. 3).

Максимум функции не обязательно должен быть в центре возможного интервала. Например, в предложении "в конце лета" пик функции будет сдвинут вправо по временной оси. Если предложение имеет тип CENTR_WEIGHT, то значение функции в точной дате по времени равно 1. Двигаясь к концам интервала, значение функции линейно убывает до нуля (рис. 4).

Теперь для сравнения необходимо вычислить площадь пересечения и объединения подграфиков функций распределения. Общая формула:

$$f_n(when_1,\ whene_2),\ =\ \frac{S_c}{S_u},$$

где S_c — площадь пересечения, а S_u — площадь объединения.

Максимальные значения функций распределения разного типа (1 и 0,5) определены таким образом, чтобы подграфики функций разных типов имели одинаковую площадь, если у них совпадают возможные интервалы.

Тестирование части "когда". Некоторые результаты тестовых испытаний программной реализации алгоритма представлены в табл. 7.

В табл. 7 дата, от которой идет отсчет — 01.05.2013. Результаты тестовых испытаний показывают, если возможные интервалы не пересекаются, то значение, характеризующее степень смысловой близости, равно нулю. Чем больше пересечение интервалов, тем больше значение смысловой близости. Если интервалы совпали, то степень близости максимальна.

Таблица 7

Сравнение предложений, содержащих части «когда»

Параметры сравнения	Мера близости					
	В прошлом месяце	Одну неделю назад	30.04.2013	Весной	В конце апреля	В День Космонавтики
В прошлом месяце	1,000	0,025	0,004	0,233	0,615	0,032
Одну неделю назад	0,025	1,000	0,000	0,011	0,032	0,000
30.04.2013	0,004	0,000	1,000	0,011	0,033	0,000
Весной	0,233	0,011	0,011	1,000	0,326	0,011
В конце апреля	0,615	0,032	0,033	0,326	1,000	0,033
В День Космонавтики	0,032	0,000	0,000	0,011	0,033	1,000

Общая формула для вычисления степени смысловой близости двух предложений

Сведения о том, как сравнить по смыслу каждую из описанных ранее частей предложений ("что", "где", "когда"), позволяют сравнить и предложения целиком. Для вычисления меры, определяющей степень смысловой близости предложений, применяется следующая формула:

$$S(F_1, F_2) = -m^2/2 + 3m/2,$$

где $m = f_t(what_1, what_2)f_r(where_1, where_2)f_n(when_1, when_2)$; f_t, f_r, f_n — функции сравнения частей предложения.

Таким образом, степень близости предложений равна нулю, если хотя бы одна из составляющих равна нулю, что интуитивно понятно, так как два события не могут произойти в разных местах, в разное время или с разными объектами. Функция равна единице тогда и только тогда, когда все составляющие равны единице. Выбор формулы для $S(F_1, F_2)$ обусловлен тем эвристическим предположением, что если предложения хотя бы немного похожи (ни одна из трех функций близости не равна нулю), то увеличение значения одной из составляющих влечет за собой резкое увеличение значения общей функции. Если же предложения практически одинаковы, то увеличение общей близости проходит медленнее: трудно быть уверенным на 100 % в том, что предложения действительно описывают один и тот же факт. Исходя из этого, если зафиксировать два значения функций близости из трех, то будет получена выпуклая вверх парабола.

Выводы

С учетом представленных в статье результатов начального этапа исследований на направлении вычисления смысловой близости двух предложений, можно констатировать следующее. Получены методы, позволяющие сравнить по смыслу описания двух фактов, что дает надежду на решение очень актуальной и нерешенной до настоящего времени задачи. При этом считается, что предложения разбиты на три составные части в некотором "черном ящике" и далее с помощью описанных выше алгоритмов идет вычисление оценки их близости. Исследование на начальном этапе связано с большим числом трудностей. К одной из них относится тот факт, что пока нет программной реализации "черного ящика", способного автоматически разделить предложение на части. Существуют только некоторые идеи и методы по его созданию. Этот факт означает, что нет возможности автоматически собрать базу реальных предложений из сети Интернет и в больших масштабах проверить работу предложенных алгоритмов. Приходится отыскивать предложения, а затем вручную делить их на части. В ходе исследования возникали трудности в работе со сторонними сервисами Яндекс, Google Maps. Причина в том, что каждый из них имеет довольно низкое ограничение на число запросов и, зачастую, их не хватало на полноценную работу по созданию и отладке алгоритмов.

В ходе работы были изучены многие методы для получения оценки семантической близости между словами и предложениями. Сюда входит и семантическая сеть WordNet с рядом алгоритмов определения степени близости, предложенных авторами ряда научных статей. Рассмотрены статистические алгоритмы получения результатов, такие как TF-IDF. Проведена также работа в области исследования и разработки мер, показывающих близость предложений, вид которых имеет определенную специфику.

Преимущества предложенного алгоритма сравнения частей "что" — широкий охват (обрабатываются практически любые предложения данного вида) и возможность задействовать в этом процессе каждое слово из подлежащих сравнению предложений. Однако этот алгоритм обладает и недостатками, связанными с тем, что большое число запросов отправляется стороннему сервису через сеть Интернет. Чтобы сравнить n предложений, в которых существительные находятся в базе WordNet, необходимо отправить следующее число запросов:

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{S_N^i} N_j^i + \sum_{j=1}^{S_V^i} V_j^i\right),$$

где S_N^i, S_V^i — число значений существительного и глагола в предложении i соответственно; N_j^i, V_j^i — число слов в синсете значения j предложения i для существительного и глагола соответственно. Для каждого существительного, которое не входит в WordNet, необходимо дополнительно сделать H запросов для получения гиперонима. Параметр H при этом является регулируемым. На настоящее время затруднен перенос алгоритма на русский и другие языки в силу того, что база WordNet для этих языков неразвита. Однако следует отметить, если базы для других языков будут созданы, то перенести алгоритм будет довольно просто. Решение задачи многозначности очень сильно зависит от того, какие слова входят в предложение. Если слова резко отличаются по смыслу от общеупотребительных, то "правильный" смысл предложения выявляется значительно чаще.

Алгоритм сравнения частей "где" прост и интуитивно понятен, способен определять широкий спектр географических названий. Среди них есть такие сложные, как школы, деревни, озера, реки, горы, улицы, дороги, памятники и др. Возможен также перенос алгоритма на другие языки. Хорошо определяются географические гиперонимы. Недостатками алгоритма являются скорость работы (необходимо отправить запрос для каждого из предложений); неточность определения; погрешность, вызванная аппроксимацией объектов прямоугольниками.

Алгоритм сравнения частей "когда" корректно обрабатывает точные даты ("*12.05.1992*", "*1 Dec*"), относительные даты ("*3 weeks ago*", "*on last weekends*"), также некоторые абстрактные даты ("*recently*", "*some days ago*"). Обрабатывается список праздников и особых дат, заданный в файле SpecialDates. Существует возможность расширения множества обрабатываемых предложений: пользователь может добавлять инфор-

мацию в конфигурационные файлы, разработчик — добавлять обработку новых метаслов (при этом не сильно изменяя код). Недостаток алгоритма в том, что тяжелые регулярные выражения в сочетании с длинными предложениями и большими конфигурационными файлами могут сильно замедлить процесс выполнения обработки. Разделение конфигурационных файлов по языкам частично решает эту задачу.

Достоинством представленного подхода является тот факт, что он может работать практически с любыми корректными данными, хотя, и это следует отметить, не слишком удачно на некоторых из них. В дальнейшем, улучшая и модернизируя программу, можно будет получить результаты более высокого качества и уже применять методы на практике. В будущем планируется акцентировать внимание на обработку части "что" предложений, и внедрять более сложные семантические и статистические алгоритмы.

Автор выражает благодарность своим научным руководителям д-ру физ.-мат. наук, проф. В. А. Васенину и канд. физ.-мат. наук, вед. науч. сотр. С. А. Афонину за постановку задачи, внимание к работе и за помощь в подготовке статьи.

Список литературы

1. **Agirre E., Alfonseca E., Hall K. et. al.** A study on similarity and relatedness using distributional and wordnet-based approaches // In Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL'09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009. P. 19—27.

2. **Bollegala D., Matsuo Y., Ishizuka M.** Measuring semantic similarity between words using web search engines // Proc. of the 16th international conference on World Wide Web WWW 07. New York: ACM, 2007. P. 757—766.

3. **Budanitsky A., Hirst G.** Evaluating WordNet-based Measures of Lexical Semantic Relatedness // Computational Linguist. 2006. N. 32. P. 13—47.

4. **Fellbaum C.** (editor). WordNet: an electronic lexical database. 2001. MIT Press, 1998.

5. **Hatzivassiloglou V., Klavans J. L., Eskin E.** Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning // In Proc. of the 1999 joint SIGDAT conference on empirical methods in natural language processing and very large corpora. Main conference. 1999. P. 203—212.

6. **Leacock C., Chodorow M.** Combining local context and wordnet similarity for word sense identification / eds. by C. Fellbaum. Cambridge, Massachusetts: MIT Press, 1998. P. 265—283.

7. **Li Y., McLean D.** An approach for measuring semantic similarity between two words using multiple information sources // IEEE Transactions on Knowledge and Data Engineering. 2003. N. 15 (4). P. 871—882.

8. **Li Y., McLean D. A., Bandar Z. A., OShea J., Crockett K.** Sentence similarity based on semantic nets and corpus statistics// IEEE Transactions on Knowledge and Data Engineering. 2006. N. 18 (8). P. 1138—1150.

9. **Metzler D., Dumais S., Meek C.** Similarity measures for short segments of text. // In Proc. of the 29th European conference on IR research (ECIR'07) / Eds. by G. Amati, C. Carpineto, G. Romano. Berlin, Heidelberg: Springer-Verlag, 2007. P. 16—27.

K. V. Lunev, Student, Programmer, Lomonosov Moscow State University, e-mail: kirilllunev@gmail.com

Measuring Semantic Similarity between Two Sentences

In this paper the results of initial researches on retrievals of models, algorithms and software tools of the determination of the semantic similarity of two sentences are presented. These results can be used by search engines for outputting more relevant content, and also in solving the problems of clustering, generalization, text indexing and many others. As it is supposed by the author in the paper, sentences can be divided into three parts, each of these parts is the definition of a fact, to be exact: what happened, where and when. The algorithm of the division into these parts is not considered in this stage of the research. The author proposes the metrics, on the basis of which the semantic similarity of the parts of sentences is determined. The data of this semantic similarity are used for the searching the semantic similarity the whole sentences. For this purpose the semantic net WordNet, the search engine Yandex, the Geocoding API and proper algorithms are used.

Keywords: *natural language processing, sentence semantic similarity, algorithms, search engine, geocoding*

References

1. **Agirre E., Alfonseca E., Hall K. et al.** A study on similarity and relatedness using distributional and wordnet-based approaches. *In Proc. of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics.* NAACL'09. Stroudsburg, PA, USA, Association for Computational Linguistics, 2009. P. 19—27.

2. **Bollegala D., Matsuo Y., Ishizuka M.** Measuring semantic similarity between words using web search engines. *Proc. of the 16th international conference on World Wide Web WWW 07.* New York: ACM, 2007. P. 757—766.

3. **Budanitsky A., Hirst G.** Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguist.* 2006. N. 32. P. 13—47.

4. **Fellbaum C.** (editor). *WordNet: an electronic lexical database.* 2001. MIT Press, 1998.

5. **Hatzivassiloglou V., Klavans J. L., Eskin E.** Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. *In Proc. of the 1999 joint SIGDAT conference on empirical methods in natural language processing and very large corpora. Main conference.* 1999. P. 203—212.

6. **Leacock C., Chodorow M.** *Combining local context and wordnet similarity for word sense identification* / eds. by C. Fellbaum. Cambridge, Massachusetts: MIT Press, 1998. P. 265—283.

7. **Li Y., McLean D.** An approach for measuring semantic similarity between two words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering.* 2003. N. 15 (4). P. 871—882.

8. **Li Y., McLean D. A., Bandar Z. A. OShea J., Crockett K.** Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering.* 2006. N. 18 (8). P. 1138—1150.

9. **Metzler D., Dumais S., Meek C.** Similarity measures for short segments of text. *In Proc. of the 29th European conference on IR research (ECIR'07)* / Eds. by G. Amati, C. Carpineto, G. Romano. Berlin, Heidelberg: Springer-Verlag, 2007. P. 16—27.

УДК 629.73.02; 629.73.05/.06; 535.643

И. О. Жаринов, д-р техн. наук, доц., зав. каф., Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (НИУ ИТМО), руководитель учебно-научного центра, Санкт-Петербургское ОКБ "Электроавтоматика" им. П. А. Ефимова, e-mail: igor_rabota@pisem.net,
О. О. Жаринов, канд. техн. наук., доц., Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Исследование распределения оценки разрешающей способности преобразования Грассмана в системах кодирования цвета, применяемых в авионике

Рассмотрена задача исследования распределения оценки разрешающей способности преобразования Грассмана в целях обоснования выбора метрологической характеристики (разрешающая способность) колориметра, используемого в экспериментах со средствами отображения информации, в которых применяют системы кодирования цвета в форматах RGB и XY. Предложено аналитическое выражение для оценки разрешающей способности преобразования Грассмана, справедливое для различных стандартов задания компонентов основных цветов (красный, зеленый, синий) и систем оценки баланса белого цвета, введенных Международной комиссией по освещению. Показано, что оценка формируется путем вычисления глобального минимума для горизонтального, вертикального и диагонального приращений координат цветности на XY-плоскости, возникающих при изменении значения кода RGB на Δ. Приведены экспериментально полученные гистограммы распределения оценок при $\Delta = 1$, $\Delta = 5$. Табулированы минимальные и максимальные значения приращений. Показано, что уровень разрешающей способности современных колориметров недостаточен для различения всех значений приращений координат цветности. Приведены семейства зависимостей, связывающих относительную долю различимых измерительным прибором координат цветности на XY-плоскости и значение разрешающей способности колориметра $0{,}0001 \leqslant h \leqslant 0{,}01$ для группы дискретных шагов $1 \leqslant \Delta \leqslant 10$ для горизонтального, вертикального и диагонального приращений.

Ключевые слова: *преобразование Грассмана, координаты цветности, оценка разрешающей способности, гистограмма распределения, рабочие характеристики различения*

Введение

При разработке современных бортовых средств отображения информации класса МФЦИ (многофункциональные цветные индикаторы) актуальной является задача выбора кодов компонентов основных цветов (красный, зеленый, синий), используемых в программном обеспечении МФЦИ для задания цветовой палитры индицируемой информации с повышенными визуальными характеристиками восприятия для летного состава. Такая задача имеет важнейшее практическое значение [1—3], так как условия эксплуатации МФЦИ предполагают визуальный контроль пилотажно-навигационной информации и геоинформационных данных (карты местности) в условиях наличия повышенных уровней внешней солнечной засветки, существенно усложняющей восприятие информации.

Средством отображения информации в МФЦИ является жидкокристаллический (ЖК) экран, цвет свечения каждого пикселя которого определяется программно-управляемым кодом *RGB* (*R — Red*, *G — Green*, *B — Blue*), задающим угол поворота жидких кристаллов, моделирующих белый спектр свечения заднего или бокового источников подсвета.

Опытным путем замечено, что каждый цвет или оттенок цвета, в котором элемент изображения индицируется на экране МФЦИ, обладает своим значением контраста изображения — отношения яркости цвета изображения к яркости цвета, на фоне которого индицируется изображение. При этом даже незначительные программные изменения кода *RGB*, в пределах которых наименование цвета (оттенка) визуально не изменяется, приводят к существенному изменению значения контраста индицируемого изображения.

Решение задачи поиска значений кодов *RGB*, для которых контраст индицируемой информации максимален [4, 5], осуществляется с использованием яркомера

для измерения яркости индицируемого изображения и колориметра для измерения координат цветности изображения. Измерение координат цветности необходимо, так как согласно действующей в авиационной промышленности нормативно-технической документации (НТД) регламентируются наименования цветов, разрешенных к использованию в бортовых системах индикации, и при сертификации МФЦИ требуется заключение специального вида с оценкой соответствия наименования используемых цветов требованиям НТД.

Практический опыт применения колориметров и анализ их технических характеристик показывают, что разрешающая способность измерительных приборов такого типа сегодня составляет до 0,002 ед. [6, 7]. При этом измерение большинство колориметров осуществляет в системе кодирования цвета *XY*. Таким образом, разработчики программного обеспечения МФЦИ оперируют с кодами *RGB*, а измерительное оборудование осуществляет измерение в системе кодирования цвета *XY*.

Преобразование Грассмана вводит взаимооднозначное соответствие числовых значений (координат цветности), представляющих один и тот же цвет (оттенок цвета) в различных системах кодирования — *RGB* и *XY*. Теоретически каждому дискретному изменению кода *RGB*, в том числе и единичному, соответствует дискретное приращение (x, y)-координат. Минимальное (по модулю) значение приращения (x, y)-координат определяет разрешающую способность преобразования Грассмана. Однако на практике зафиксировать такое приращение не всегда представляется возможным в силу ограничений, обусловленных разрешающей способностью измерительного оборудования. Разрешающая способность колориметра на уровне 0,002 ед. приводит к тому, что одни и те же значения (x, y)-координат соответствуют не одному уникальному коду, а целой группе кодов *RGB*.

В связи с этим актуальной является задача поиска характеристики различения координат цветности элементов изображения, представляющей собой зависимость числа различимых измерительным прибором цветов и оттенков от разрешающей способности колориметра и шага изменения кода *RGB*.

1. Системы кодирования цвета. Преобразование Грассмана

Система *RGB* (рис. 1, *а*, см. третью сторону обложки) предполагает способ задания каждого цвета или оттенка цвета в виде двоичного позиционного кода, в котором присутствуют группы разрядов, соответствующие компонентам основных цветов (красный, зеленый, синий). Значения кодов основных цветов могут находиться в произвольных пропорциях. На практике коды *RGB* принято представлять в десятичной системе счисления. Система *XY* предполагает способ задания каждого цвета или оттенка цвета в виде пары вещественных (x, y)-координат на *XY*-плоскости.

Переход из одной системы кодирования цвета в другую осуществляется через цветовой треугольник Максвелла (рис. 1, *б*, см. третью сторону обложки) по правилам прямого и обратного преобразования Грассмана [8, 9]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad (1)$$

где X, Y, Z — компоненты цвета в системе *XYZ* цветового треугольника Максвелла; X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b — компоненты цвета, определенные Международной комиссией по освещению (МКО) и используемые в качестве эталона для точного стандарта определения цвета; R, G, B — десятичный код цвета компонентов основных цветов в системе *RGB*. Компоненты X_r, Y_r, Z_r определяют правило преобразования кода *RGB* для эталонного значения красного цвета, компоненты X_g, Y_g, Z_g и X_b, Y_b, Z_b, — для зеленого и синего цветов соответственно.

Переход от значений сторон *XYZ* треугольника Максвелла к (x, y)-координатам цветности на *XY*-плоскости (см. рис. 1, *в*, см. третью сторону обложки) осуществляется по следующим формулам:

$$x = \frac{X}{X+Y+Z}; \quad y = \frac{Y}{X+Y+Z}. \quad (2)$$

Оценка разрешающей способности преобразования Грассмана имеет следующий вид:

$$\begin{cases} \Delta x = \dfrac{\det\begin{pmatrix} \det\begin{pmatrix} X_b & Y_b+Z_b \\ X_g & Y_g+Z_g \end{pmatrix} & R_i & R_{i+1} \\ \det\begin{pmatrix} X_r & Y_r+Z_r \\ X_b & Y_b+Z_b \end{pmatrix} & G_i & G_{i+1} \\ \det\begin{pmatrix} X_g & Y_g+Z_g \\ X_r & Y_r+Z_r \end{pmatrix} & B_i & B_{i+1} \end{pmatrix}}{\left(a_r R_i + a_g G_i + a_b B_i\right)\left(a_r R_{i+1} + a_g G_{i+1} + a_b B_{i+1}\right)}; \\ \Delta y = \dfrac{\det\begin{pmatrix} \det\begin{pmatrix} Y_b & X_b+Z_b \\ Y_g & X_g+Z_g \end{pmatrix} & R_i & R_{i+1} \\ \det\begin{pmatrix} Y_r & X_r+Z_r \\ Y_b & X_b+Z_b \end{pmatrix} & G_i & G_{i+1} \\ \det\begin{pmatrix} Y_g & X_g+Z_g \\ Y_r & X_r+Z_r \end{pmatrix} & B_i & B_{i+1} \end{pmatrix}}{\left(a_r R_i + a_g G_i + a_b B_i\right)\left(a_r R_{i+1} + a_g G_{i+1} + a_b B_{i+1}\right)}, \end{cases} \quad (3)$$

где Δx — разрешающая способность преобразования Грассмана по x-координате; Δy — разрешающая способность преобразования Грассмана по y-координате; R_i, G_i, B_i, R_{i+1}, G_{i+1}, B_{i+1} — десятичные коды для начальной i-й и конечной $(i+1)$-й точки в системе

RGB, вызвавшие приращения Δx, Δy в системе *XY*; $a_r = X_r + Y_r + Z_r$, $a_g = X_g + Y_g + Z_g$, $a_b = X_b + Y_b + Z_b$.

Эта оценка получена теоретически путем вычисления разниц:

$$\Delta x = x_{i+1}\big|_{R_{i+1}G_{i+1}B_{i+1}} - x_i\big|_{R_iG_iB_i},$$

$$\Delta y = y_{i+1}\big|_{R_{i+1}G_{i+1}B_{i+1}} - y_i\big|_{R_iG_iB_i}$$

для $R_{i+1} = R_i + \Delta R$, $G_{i+1} = G_i + \Delta G$, $B_{i+1} = B_i + \Delta B$ с использованием выражений (1), (2).

2. Автоматизированное рабочее место в экспериментах по оценке координат цветности

Оценка координат цветности элементов изображения, индицируемых на ЖК-экране МФЦИ, проводится с использованием инструментальной ЭВМ (ИЭВМ) и системы автоматизированного проектирования (САПР), входящих в состав автоматизированного рабочего места (АРМ).

САПР АРМ МФЦИ составляют (рис. 2) следующие компоненты.

1. Программное обеспечение (ПО) САПР, включающее компоненты данных, компоненты функционального программного обеспечения (ФПО) и тестового программного обеспечения (ТПО), компоненты прикладного ПО для загрузки данных и программ в МФЦИ. Компонентами ПО являются документы с текстами программ и кодами данных. Компонентами данных являются:

- палитра (таблица десятичных кодов *RGB* цветов, используемых разработчиками ПО при создании индикационных кадров авионики, включающих отображение пилотажно-навигационных параметров и геоинформационных данных);
- библиотека шрифтов и символов согласованной конфигурации, представленных в памяти графического контроллера [10] МФЦИ в векторном и растровом видах для различных по размеру (в пикселях) знакомест (библиотека включает изображения букв и символов согласованной конфигурации на различных языках — русский, латиница и др., цифры в римском и арабском алфавите и т. д.);
- библиотека внутренних углов (приращений, рассчитанных по тригонометрическим функциям для построения на экране МФЦИ графических примитивов типа дуга и окружность с использованием аппроксимации вписанным многоугольником).

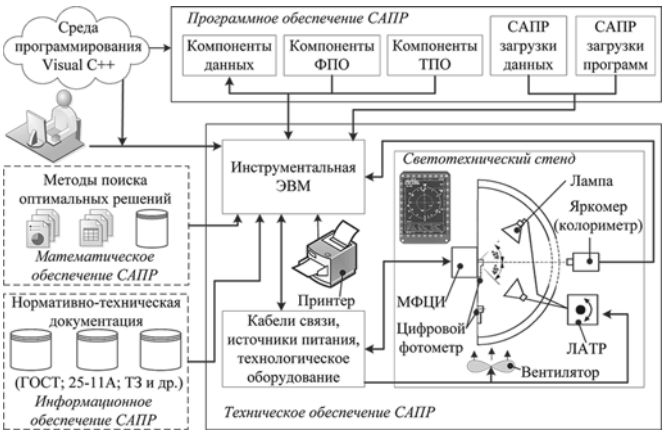

Рис. 2. Компоненты САПР автоматизированного рабочего места МФЦИ

Программное обеспечение САПР позволяет оператору решать автоматизированным способом следующие задачи:

- создавать и редактировать на ИЭВМ файлы загрузочных компонентов данных, используемых при отображении индикационных кадров МФЦИ;
- заносить из ИЭВМ в память графического контроллера МФЦИ по технологическому интерфейсу загрузочные компоненты данных;
- заносить из ИЭВМ в память вычислительного модуля МФЦИ компоненты ФПО и ТПО для оценки работоспособности изделия во всех режимах работы.

2. Математическое обеспечение САПР, включающее совокупность математических методов и критериев качества, математических моделей и алгоритмов автоматизации проектирования, необходимых для выполнения процедуры оптимизации процесса выбора оператором цветовой палитры МФЦИ. Математическое обеспечение САПР позволяет оператору:

- осуществлять выбор методов и критериев, приемлемых для выполнения процедуры оптимизации выбора цветовой палитры;
- проводить математический анализ данных, полученных в результате измерения яркости и оценки яркостного контраста изображения, в целях поиска компонентов цветов и оттенков, обладающих повышенными характеристиками восприятия для человека.

3. Информационное обеспечение САПР, включающее совокупность сведений, необходимых для выполнения процедур автоматизированного выбора. Основой информационного обеспечения САПР являются автоматизированные банки данных, которые состоят из различных баз данных САПР и систем управления базами данных. В информационное обеспечение САПР автоматизированного рабочего места МФЦИ входят:

- нормативно-техническая документация (в частности, руководство 25-11А по сертификации авиационного оборудования), государственные и отраслевые стандарты на средства отображения информации, стандарты на оформление документации, техническое задание (ТЗ) на разработку МФЦИ;
- существующие (полученные ранее другими разработчиками) типовые проектные решения МФЦИ с указанием значимых для процедуры выбора параметров изделий;
- существующие системы кодирования изображений (математические зависимости, связывающие коды, координаты цветности и длины волн цветов и оттенков).

Информационное обеспечение САПР позволяет оператору осуществлять выбор:

- актуального для авионики подмножества цветов и оттенков, используемых при отображении пилотажно-навигационных параметров и геоинформационных данных, из полного множества цветов и оттенков (2^{18} для 6-битного ЖК-экрана или 2^{24} для 8-битного ЖК-экрана), потенциально воспроизводимых на МФЦИ;
- системы кодирования изображения, в которой будет выполняться процедура выбора желаемых значений координат цветности;
- системы кодирования изображения, в которой будет выполняться проектирование физического устройства — графического контроллера МФЦИ;
- системы кодирования изображения, в которой будет выполняться программирование физического устройства.

4. Техническое обеспечение САПР, включающее совокупность взаимосвязанных и взаимодействующих технических средств, предназначенных для выполнения процедуры оптимизации выбора цветовой палитры МФЦИ. Техническое обеспечение САПР МФЦИ позволяет оператору решать автоматизированным способом следующие задачи:

- подготовка компонентов данных, компонентов ФПО и ТПО (техническое обеспечение САПР МФЦИ, предназначенное для решения этой задачи, составляет ИЭВМ; ИЭВМ с установленным ПО позволяет оператору осуществлять программным способом процедуры автоматизации подготовки и редактирования компонентов данных и компонентов программ перед их вводом в МФЦИ; также ИЭВМ предоставляет оператору возможность кодирования загружаемых компонентов, сохранения данных и программ на жестком диске ИЭВМ и редактирования созданных ранее данных);
- передача данных (группу технического обеспечения САПР, предназначенного для решения этой задачи, составляют ИЭВМ, кабели связи и источники питания; группа предоставляет оператору возможности ввода данных в МФЦИ, визуального и аппаратурного контроля целостности занесения данных по значениям контрольных сумм);
- программная обработка данных (группу технического обеспечения САПР, предназначенного для решения этой задачи, составляют ИЭВМ; светотехнический стенд, включающий цифровой фотометр, яркомер (или колориметр), линейный автотрансформатор (ЛАТР), лампы внешней освещенности и вентилятор обдува ЖК экрана; технологическое оборудование и кабели связи; источники питания; группа предоставляет оператору возможность измерения яркости свечения экрана МФЦИ в любом цвете (оттенке цвета), включая фоновый (черный, серый), передачи результатов измерений яркости ЖК экрана МФЦИ в ИЭВМ по технологическому интерфейсу, расчета яркостного контраста элементов изображения на экране МФЦИ в условиях воздействия внешней освещенности уровня 0...75 кЛк);
- отображение и документирование данных (группу технического обеспечения САПР, предназначенного для решения этой задачи, составляют ИЭВМ и принтер; группа предоставляет оператору возможность оперативного представления и документирования полученных результатов (проектных решений) по выбору компонентов цветовой палитры МФЦИ, возможность подготовки научно-технического отчета о полученных результатах исследования; также группа позволяет оператору изменять производительность своей работы на АРМ путем использования ИЭВМ разного типа и использования мультипрограммных и диалоговых режимов работы прикладного ПО, установленного на ИЭВМ);
- поддержка архивирования готовых проектных решений (техническое обеспечение САПР, предназначенное для решения этой задачи, составляет ИЭВМ; ИЭВМ позволяет оператору сохранять полученные в процессе исследования компоненты цветовой палитры МФЦИ, сравнивать результаты проектных решений, полученных для различных моделей МФЦИ, выполненных на базе ЖК экранов различных фирм-производителей в условиях воздействия различных уровней внешней освещенности).

3. Исследование распределения оценки разрешающей способности преобразования Грассмана

Согласно принятой в системе *RGB* 8-битной модели кодирования компонентов основных цветов общее число комбинаций кодов *RGB* составляет $2^8 \cdot 2^8 \cdot 2^8 = 2^{24}$. Опытным путем замечено, что не каждое приращение ΔR, ΔG, ΔB вызывает соответствующие приращения Δx, Δy. В ряде случаев при изменении кода R_i, G_i, B_i до кода R_{i+1}, G_{i+1}, B_{i+1} приращения Δx и/или Δy равны нулю. Такими случаями являются:

- одновременное изменение кода *RGB* с равными значениями R_i, G_i, B_i по всем трем компонентам на одно и то же значение $\Delta R = \Delta G = \Delta B$, т. е. приращение кода *RGB* возникает в одном и том же сером цвете;
- увеличение кода одного компонента, например, *R*, на произвольное значение ΔR при нулевых значениях кода *RGB* двух других компонентов, т. е. приращение кода возникает только в красном цвете и т. д.

Таким образом, изменение приращений не возникает в случаях, при которых наименование цвета с изменением кода *RGB* остается постоянным (не вводится какого-либо оттенка цвета путем смешения цветов в пропорциях) и определитель в числителе уравнений (3) оказывается равным нулю.

Во всех других случаях каждому дискретному приращению ΔR, ΔG, ΔB в системе *RGB* соответствуют приращения Δx, Δy на *XY*-плоскости. Гистограммы распределения значений $|\Delta x|$, $|\Delta y|$ и значения диагонального перехода $\sqrt{\Delta x^2 + \Delta y^2}$ приведены на рис. 3, см. четвертую сторону обложки. Гистограммы получены при размере шага разбиения подинтервалов по оси абсцисс 0,0001 ед. По оси ординат отложены относительные частоты попадания значений соответствующих приращений в подинтервалы.

Как следует из анализа рис. 3, огибающая распределения значений $|\Delta x|$, $|\Delta y|$ при единичных (минимальных)

приращениях Δ имеет вид убывающей экспоненты. С ростом Δ наблюдается смещение среднего значения распределения $|\Delta x|$, $|\Delta y|$ в сторону больших значений по абсолютной величине с одновременным уменьшением общего числа ненулевых переходов на *XY*-плоскости.

Уменьшение общего числа ненулевых переходов эквивалентно снижению числа цветов и оттенков, сосредоточенных в видимой зоне *XY*-плоскости, и более "грубому" представлению цветового пространства МКО. Известны работы, в частности [11], в которых для грубой аппроксимации цветового пространства и покрытия групп (*x*, *y*)-координат цветности, объединенных одним наименованием цвета (оттенка), на *XY*-плоскости вводятся граничные функции специального типа — эллипсы Мак-Адама.

Таким образом, с одной стороны, с ростом Δ увеличивается среднее значение модуля дискрета Δx, Δy, что упрощает различение оттенков цветов между собой, с другой стороны, снижается общее число различимых цветов и оттенков в видимой зоне *XY*-плоскости.

Анализ рис. 3 также показывает, что большинство приращений Δx, Δy и приращения диагонального типа при малых значениях Δ сосредоточены в области значений, не превышающих разрешающей способности колориметра — 0,002 ед. Иными словами, данные переходы не могут быть различены инструментальными средствами измерения с такой метрологической характеристикой. Для разработчиков программного обеспечения МФЦИ это означает, что точность задания кода *RGB* с шагом Δ = 1 не является информативной, так как она не может быть выявлена и подтверждена инструментальными средствами измерения.

Минимальные и максимальные значения приращений координат на *XY*-плоскости, рассчитанные по уравнениям (3), а также начальные коды *RGB*, в точках которых достигаются экстремумы $|\Delta x|$, $|\Delta y|$, $\sqrt{\Delta x^2 + \Delta y^2}$ при поочередном изменении компонентов десятичного кода *RGB* с шагом Δ, приведены в таблице. Для определенности шаг Δ выбран из множества {1, 3, 5, 10, 15}. Минимум дискретного перехода получен на уровне 10^{-10}. Начало итераций осуществлялось с десятичного кода *RGB* (1, 0, 0). Комбинация кода (0, 0, 0), соответствующая черному цвету изображения, в расчетах системы (3) не используется.

Семейство зависимостей числа различимых измерительным прибором координат цветности на *XY*-плоскости от разрешающей способности прибора, заданной в диапазоне [0,0001...0,01] ед., и значения шага $\Delta = \Delta R = \Delta G = \Delta B$, выполняемого раздельно по каждому компоненту основного цвета в диапазоне значений [1...10], приведено на рис. 4.

По оси абсцисс отложена *h* — разрешающая способность колориметра, абсолютные значения которой соответствуют текущему уровню технологий изготовле-

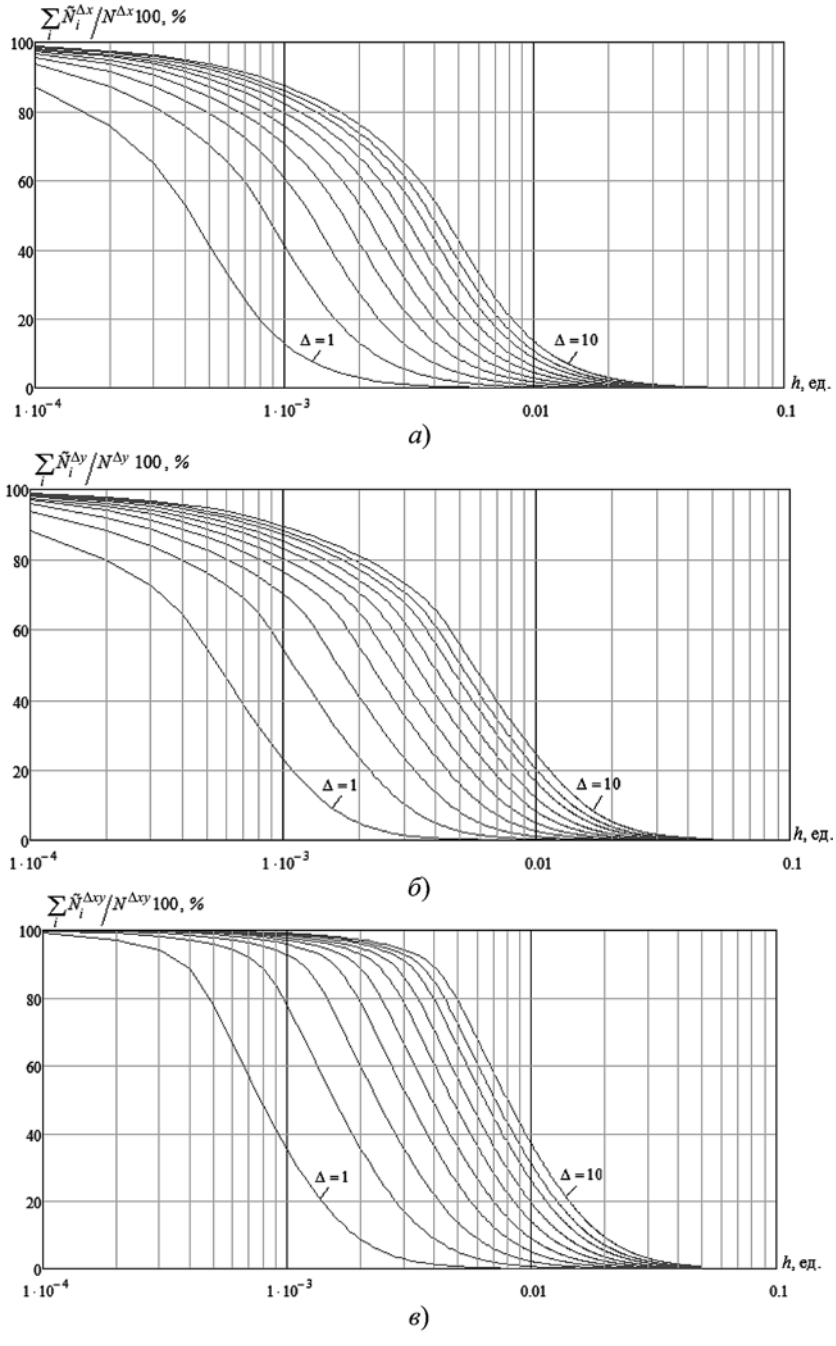

Рис. 4. Рабочие характеристики различения координат цветности на *XY*-плоскости для:

а — $|\Delta x|$; *б* — $|\Delta y|$; *в* — $\sqrt{\Delta x^2 + \Delta y^2}$

Расчетные значения экстремумов приращений координат цветности на *XY*-плоскости

Δ, ед.		*x*-координата		*y*-координата		Диагональный переход	
		$\min\lvert\Delta x\rvert$, ед.	Код *RGB*	$\min\lvert\Delta y\rvert$, ед.	Код *RGB*	$\min\sqrt{\Delta x^2+\Delta y^2}$, ед.	Код *RGB*
Минимальные значения							
1	*R*	$8{,}449\cdot10^{-10}$	7	$7{,}23\cdot10^{-10}$	43 + Δ	$1{,}614\cdot10^{-6}$	253 + Δ
	G		221 + Δ		254 + Δ		254 + Δ
	B		254 + Δ		47		0
3	*R*	$2{,}556\cdot10^{-9}$	7	$1{,}013\cdot10^{-8}$	241	$4{,}965\cdot10^{-6}$	250 + Δ
	G		219 + Δ		237 + Δ		249 + Δ
	B		252 + Δ		252 + Δ		0
5	*R*	$1{,}282\cdot10^{-7}$	246 + Δ	$1{,}698\cdot10^{-8}$	241	$8{,}477\cdot10^{-6}$	246 + Δ
	G		245		235 + Δ		245 + Δ
	B		160 + Δ		250 + Δ		0
10	*R*	$9{,}056\cdot10^{-7}$	61	$3{,}377\cdot10^{-7}$	161	$1{,}731\cdot10^{-5}$	241 + Δ
	G		240 + Δ		230 + Δ		240 + Δ
	B		120		240 + Δ		0
15	*R*	$1{,}342\cdot10^{-6}$	61	$6{,}733\cdot10^{-6}$	226 + Δ	$2{,}884\cdot10^{-5}$	226 + Δ
	G		240 + Δ		180		225 + Δ
	B		120		150		0
Максимальные значения							
1	*R*	0,124	2	0,136	1 + Δ	0,153	2
	G		1 + Δ		2		1
	B		0 + Δ		0 + Δ		0 + Δ
3	*R*	0,137	1 + Δ	0,217	1	0,234	1
	G		0		3		3
	B		3		0 + Δ		0 + Δ
5	*R*	0,155	1 + Δ	0,234	1	0,258	1 + Δ
	G		0		5		0 + Δ
	B		5		0 + Δ		5
10	*R*	0,171	1 + Δ	0,248	1	0,28	1 + Δ
	G		0		10		0 + Δ
	B		10		0 + Δ		10
15	*R*	0,176	1 + Δ	0,253	1	0,288	1 + Δ
	G		0		15		0 + Δ
	B		15		0 + Δ		15

ния аппаратуры такого типа и технологиям создания колориметрических измерителей ближайшего будущего.

По оси ординат на рис. 4 отложены следующие значения:

$$\sum_i \tilde{N}_i^{\Delta x}/N^{\Delta x};\ \sum_i \tilde{N}_i^{\Delta y}/N^{\Delta y};\ \sum_i \tilde{N}_i^{\Delta xy}/N^{\Delta xy},$$

где $\tilde{N}_i^{\Delta x} = \left\{ \begin{matrix} 1,\ h \leqslant |\Delta x_i| < 1 \\ 0,\ 0 < |\Delta x_i| < h \end{matrix} \right\}$; $\tilde{N}_i^{\Delta y} = \left\{ \begin{matrix} 1,\ h \leqslant |\Delta y_i| < 1 \\ 0,\ 0 < |\Delta y_i| < h \end{matrix} \right\}$;

$\tilde{N}_i^{\Delta xy} = \left\{ \begin{matrix} 1,\ h \leqslant \sqrt{\Delta x_i^2 + \Delta y_i^2} < 1 \\ 0,\ 0 < \sqrt{\Delta x_i^2 + \Delta y_i^2} < h \end{matrix} \right\}$, каждая из которых представляет собой долю теоретически различаемых колориметром координат цветности $\sum_i \tilde{N}_i^{\Delta x}$, $\sum_i \tilde{N}_i^{\Delta y}$, $\sum_i \tilde{N}_i^{\Delta xy}$ цветов и оттенков, имеющихся в видимой зоне *XY*-плоскости, по отношению к общему числу $N^{\Delta x}$, $N^{\Delta y}$, $N^{\Delta xy}$ существующих в видимой зоне *XY*-плоскости ненулевых горизонтальных, вертикальных и диагональных переходов соответственно при заданном $\Delta = \Delta R = \Delta G = \Delta B$.

Семейство зависимостей приведено для $\Delta \in [1...10]$, левая кривая на рис. 4 соответствует $\Delta = 1$, правая — $\Delta = 10$, шаг приращения $\Delta = 1$.

Заключение

Полученные в процессе исследования числовые характеристики и графические зависимости позволяют сделать следующие выводы. При планировании экспериментов с привлечением колориметров выбор измерительного прибора необходимо проводить на основе рабочих характеристик различения координат цветности, приведенных на рис. 4.

Анализ рис. 4 показывает, что при разрешающей способности колориметра на уровне 0,002 ед. по *x*-координате неразличимыми оказываются 97 % существующих переходов $|\Delta x|$ при изменении $\Delta = 1$; аналогично (см. рис. 4, *б*), по *y*-координате — 95 %; для диагонального перехода (см. рис. 4, *в*) — 92 %.

Измеритель с той же метрологической характеристикой при большем приращении Δ, например, при $\Delta = 5$, обнаружит 62 % переходов $|\Delta x|$, 75 % переходов $|\Delta y|$, 97 % диагональных переходов $\sqrt{\Delta x^2 + \Delta y^2}$. Однако абсолютное число ненулевых приращений на *XY*-плоскости будет уже примерно на два порядка меньше: $1{,}089 \cdot 10^8$ при $\Delta = 1$ и $8{,}925 \cdot 10^5$ при $\Delta = 5$.

Результаты расчетов, приведенные в таблице, показывают, что для различения минимального дискрета изменения (*x*, *y*)-координат разрешающая способность колориметра, осуществляющего измерения на *XY*-плоскости, должна быть на уровне не 0,002 ед., как сегодня, а порядка 10^{-10}.

Уравнения системы (3) приведены в аналитическом виде (зависимость приращений от коэффициентов матрицы МКО) и могут быть использованы для получения рабочих характеристик различения координат цветности как для стандарта МКО 1931 г., так и для стандарта МКО 1964 г. с различными системами оценки баланса белого цвета: D-75, D-65, D-55, D-50 и др., в пределах каждой из которых компоненты основных цветов X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b базового преобразования (1), (2) являются различными. Представленные числовые примеры и графики зависимостей определены для следующих значений: $X_r = 0{,}478$; $X_g = 0{,}299$; $X_b = 0{,}175$; $Y_r = 0{,}263$, $Y_g = 0{,}650$; $Y_b = 0{,}081$; $Z_r = 0{,}020$; $Z_g = 0{,}160$; $Z_b = 0{,}908$.

Список литературы

1. **Парамонов П. П., Костишин М. О., Жаринов И. О., Нечаев В. А., Сударчиков С. А.** Принцип формирования и отображения массива геоинформационных данных на экран средств бортовой индикации // Научно-технический вестник информационных технологий, механики и оптики. 2013. № 6. С. 136—142.

2. **Костишин М. О., Жаринов И. О., Жаринов О. О.** Оценка светотехнических характеристик бортовых средств индикации геоинформационных данных для основных отображаемых цветов и их оттенков // Сб. докл. XX Междунар. науч.-практ. конф. студентов, аспирантов и молодых ученых "Современные техника и технологии". Томск: Изд-во Том. политехн. ун-та, 2014. Т. 1. С. 111—112.

3. **Костишин М. О., Жаринов И. О., Книга Е. В.** Исследование эргономических свойств бортовых средств отображения пилотажно-навигационных параметров и геоинформационных данных в авионике // Сб. докл. XX Междунар. науч.-практ. конф. студентов, аспирантов и молодых ученых "Современные техника и технологии". Томск: Изд-во Том. политехн. ун-та, 2014. Т. 1. С. 109—110.

4. **Barber S. et al.** US Patent 7,417,641 B1: Aeronautical chart display apparatus and method. 26.08.2008.

5. **Schanda J.** Colorimetry Understanding the CIE System. Wiley-Interscience John Wiley & Sons, INC., 2007. 499 p.

6. **Стороженко А. И.** Оценка погрешностей визуальных и фотоэлектрических методов измерения координат цветности: Автореф. дис. канд. техн. наук. СПб., 2007. 21 с.

7. **Михайлов О. М., Заргарьянц Г. С.** Интегральный дистанционный колориметр на основе трехцветной колориметрической системы КЗФ // Светотехника. 2008. № 3. С. 19—25.

8. **Бондаренко М. Ф., Шабанов-Кушнаренко С. Ю., Шабанов-Кушнаренко Ю. П.** Разработка системы кодирования цвета // Бионика интеллекта. 2009. № 2. С. 13—23.

9. **Хорунжий М. Д.** Метод количественной оценки цветовых различий при восприятии цифровых изображений // Научно-технический вестник информационных технологий, механики и оптики. 2008. № 1. С. 136—144.

10. **Жаринов И. О., Жаринов О. О.** Бортовые средства отображения информации на плоских жидкокристаллических панелях: учеб. пособие. Информационно-управляющие системы. СПб.: ГУАП, 2005. 144 с.

11. **Агостон Ж.** Теория цвета и ее применение в искусстве и дизайне: пер. с англ. М.: Мир, 1982. 184 с.

I. O. Zharinov, Assistant Professor, Chef of Department, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (University ITMO); Chef of learning-scientists center, SPb Scientific Design Bureau "Electroavtomatica" n.a. P. A. Efimova, e-mail: igor_rabota@pisem.net,
O. O. Zharinov, Assistant Professor, Saint-Petersburg state university of aerospace equipment

Research of Properties of an Assessment of the Resolution of Grassmann's Transformation in Chromaticity Coding Systems, Applied in Avionic Equipment

Research of properties of an assessment of the resolution of Grassmann's transformation in order to motivate a choice of metrological characteristics (resolution ability) of colorimeters, used in experiments with color displays, where coding systems of RGB and XY formats are implemented, is considered. An analytical expression, which refers to the assessment value of the resolution of Grassmann's transformation, suitable for various standards of combining of basic colors (red, green, blue) and for various systems of white balance evaluation in according to recommendations of International Commission on Illumination, is proposed. The assessment of the resolution value is obtaining through calculation of minimal value among augmentations, appearing in every coordinate on XY-chromaticity plane, when RGB code components incremented by any presumed value of Δ. Histograms of assessment values distribution, obtained by numerical calculation procedure for $\Delta = 1$ and $\Delta = 5$, are shown. Tables with minimal and maximal values of assessment are presented. There was established, that resolution ability of modern colorimeters can't allow proper differentiation of major part of chromaticity coordinates assessments. Set of characteristics, establishing relations between value of fraction (ranging from 0 to 100 %) of chromaticity coordinates augmentations on XY-plane, which may be correctly measured by colorimeter, out of total quantity of non-zero augmentations, with chromaticity coordinates resolution value of given colorimeter $0{,}0001 \leqslant h \leqslant 0{,}01$, for various incrementation steps $1 \leqslant \Delta \leqslant 10$, separately for horizontal, vertical and diagonal augmentation values.

Keywords: *Grassmann's transformation, chromaticity coordinates, assessment of the resolution ability, histogram, operating characteristics of chromaticity coordinates resolution*

References

1. **Paramonov P. P., Kostishin M. O., Zharinov I. O., Nechaev V. A., Sudarchikov S. A.** Princip formirovanija i otobrazhenija massiva geoinformacionnyh dannyh na jekran sredstv bortovoj indikacii. *Nauchno-tehnicheskij vestnik informacionnyh tehnologij, mehaniki i optiki*. 2013. N. 6. P. 136—142.

2. **Kostishin M. O., Zharinov I. O., Zharinov O. O.** Ocenka svetotehnicheskih harakteristik bortovyh sredstv indikacii geoinformacionnyh dannyh dlja osnovnyh otobrazhaemyh cvetov i ih ottenkov. *Sbornik dokladov XX Mezhdunarodnoj nauchno-prakticheskoj konferencii studentov, aspirantov i molodyh uchenyh "Sovremennye tehnika i tehnologii"*. Tomsk: Izd-vo Tomsk. politehn. un-ta. 2014. V. 1. P. 111—112.

3. **Kostishin M. O., Zharinov I. O., Kniga E. V.** Issledovanie jergonomicheskih svojstv bortovyh sredstv otobrazhenija pilotazhno-navigacionnyh parametrov i geoinformacionnyh dannyh v avionike. *Sbornik dokladov XX Mezhdunarodnoj nauchno-prakticheskoj konferencii studentov, aspirantov i molodyh uchenyh "Sovremennye tehnika i tehnologii"*. Tomsk: Izd-vo Tomsk. politehn. un-ta. 2014. V. 1. P. 109—110.

4. **Barber S. et al.** US Patent 7,417,641 B1. Aeronautical chart display apparatus and method, 26.08.2008.

5. **Schanda J.** *Colorimetry Understanding the CIE System*. Wiley-Interscience John Wiley & Sons, INC., 2007. 499 p.

6. **Storozhenko A. I.** *Ocenka pogreshnostej vizual'nyh i fotojelektricheskih metodov izmerenija koordinat cvetnosti*: Avtoref. diss. kand. tehn. nauk. SPb., 2007. 21 p.

7. **Mihajlov O. M., Zargar'janc G. S.** Integral'nyj distancionnyj kolorimetr na osnove trehcvetnoj kolorimetricheskoj sistemy KZF. *Svetotehnika*. 2008. N. 3. P. 19—25.

8. **Bondarenko M. F., Shabanov-Kushnarenko S. Ju., Shabanov-Kushnarenko Ju. P.** Razrabotka sistemy kodirovanija cveta. *Bionika intellekta: nauchno-tehnicheskij zhurnal*. 2009. N. 2. P. 13—23.

9. **Horunzhij M. D.** Metod kolichestvennoj ocenki cvetovyh razlichij pri vosprijatii cifrovyh izobrazhenij. *Nauchno-tehnicheskij vestniki informacionnyh tehnologij, mehaniki i optiki*. 2008. N. 1. P. 136—144.

10. **Zharinov I. O., Zharinov O. O.** *Bortovye sredstva otobrazhenija informacii na ploskih zhidkokristallicheskih paneljah*: Ucheb. posobie. Informacionno-upravljajushhie sistemy. SPb.: GUAP, 2005. 144 p.

11. **Agoston Zh.** *Teorija cveta i ee primenenie v iskusstve i dizajne*: per. s angl. M.: Mir, 1982. 184 p.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 17.06.2014 г. Подписано в печать 23.07.2014 г. Формат 60×88 1/8. Заказ PI814
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.