ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/Issn.2220-3397 ISSN 2220-3397

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

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/prin/rus 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

Vol. 7
N 4
2016

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.4'416 DOI: 10.17587/prin.147-157

Н. И. Вьюкова, ст. науч. сотр., e-mail: niva@niisi.msk.ru,
В. А. Галатенко, д-р физ.-мат. наук, ст. науч. сотр., зав. сектором, e-mail: galat@niisi.msk.ru,
С. В. Самборский, ст. науч. сотр., e-mail: sambor@niisi.msk.ru,
Федеральное государственное учреждение Федеральный научный центр
Научно-исследовательский институт системных исследований
Российской академии наук, г. Москва

Использование векторных расширений современных процессоров

Статья посвящена анализу подходов к эффективному использованию векторных расширений современных процессоров. Такое использование рассматривается применительно к сфере разработки программ для научных приложений. Проанализированы вопросы ручной и автоматической векторизации программ. Особое внимание уделено свойствам системы команд векторного расширения, способствующим эффективной векторизации кода.

Ключевые слова: *векторное расширение, SIMD (Single Instruction Multiple Data), автоматическая векторизация, SLP (Superword Level Parallelism)*

Введение

На раннем этапе развития процессоров, используемых в настольных компьютерах, рост производительности достигался за счет наращивания тактовой частоты. Однако к началу текущего столетия возможности такого подхода приблизились к пределу, обусловленному современными технологиями. Возникла необходимость в поиске других резервов роста производительности, что привело, в частности, к применению различных механизмов параллельного исполнения в процессорах. К их числу относятся:

- параллельное исполнение команд, включая суперскалярные, VLIW-, EPIC-процессоры;
- параллелизм данных, который поддерживают векторные расширения в универсальных процессорах;
- параллелизм многопоточного выполнения за счет использования многоядерных архитектур.

Если наращивание тактовой частоты процессоров позволяло повышать производительность приложений без каких-либо усилий со стороны разработчиков программного обеспечения, то использование перечисленных видов параллелизма требует: создания соответствующих методов оптимизации в компиляторах; использования новых подходов к разработке программ; разработки языков параллельного программирования, векторных и многопоточных расширений в языках программирования; создания эффективных механизмов поддержки со стороны операционных систем, средств отладки и профилирования.

Поддержка векторных вычислений впервые появилась в так называемых векторных процессорах [1], которые составляли основу суперкомпьютеров, широко применявшихся в сфере научных и инженерных вычислений в период 1970—1990-х гг. По сравнению с современными векторными расширениями, отличительной чертой классических векторных процессоров являлась поддержка векторов большого размера (или даже векторов неограниченного размера). Так, суперкомпьютер CRAY-1 поддерживал векторы из 64 элементов размером 64 бит, суперкомпьютер CDC STAR-100 — векторы до 65 535 элементов, ЭВМ архитектуры FPS T-Series [2] — векторы из 128 или 256 элементов размером 64 или 32 бит соответственно.

Первые векторные (или SIMD — *Single Instruction Multiple Data*) расширения в универсальных процессорах появились в связи с распространением мультимедийных приложений и потребностью в эффективной обработке мультимедийных данных. Этим определялись характерные черты таких расширений. Например, расширение MMX (*MultiMedia eXtension*) [3], появившееся в 1997 г., поддерживало операции над 64-битными упакованными векторами с элементами целочисленных типов размеров 8, 16 и 32 бит. В технологии 3Dnow (AMD) [4] функциональные возможности MMX были дополнены механизмами поддержки векторов с элементами вещественного типа одинарной точности, что позволяло значительно ускорить работу графических приложений. Появившиеся позже расширения AltiVec (IBM, Motorola, Apple) [5], а также серия расширений SSE, SSE2, SSE3, SSE4 [6] (Intel и AMD) поддерживали уже векторы размера 128 бит с элементами целочисленных и вещественных типов. Начиная с SSE2 в этих рас-

ширениях поддерживается тип данных с плавающей запятой двойной точности. Расширения семейства Intel AVX (*Advanced Vector Extension*) [7] поддерживают векторы размером 256 и 512 бит (в анонсированном AVX-512).

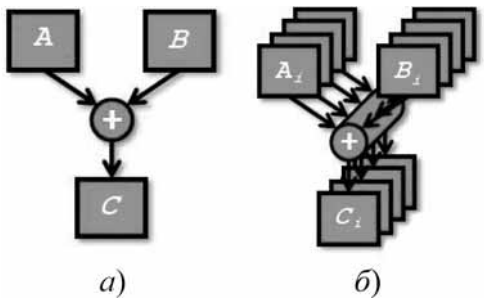
Векторные расширения присутствуют практически во всех современных микропроцессорных архитектурах. Так, расширение ARM NEON [8] поддерживает векторы размеров 64 или 128 бит с элементами целочисленных и вещественных типов. Архитектура MIPS SIMD Architecture (MSA) [9], выпущенная в 2012 г., поддерживает операции над 128-битными векторами. Векторные команды поддерживаются также в процессорах обработки сигналов [10—12].

Сфера применимости векторных расширений неуклонно расширяется. В настоящее время она включает не только приложения мультимедиа, но и различные области научно-технических вычислений, в том числе сегмент высокопроизводительных встраиваемых вычислений для оборонных, аэрокосмических и государственных проектов (MAG HPEC), а также приложения баз данных.

Понятия "векторная инструкция" и "SIMD-инструкция" часто рассматривают как синонимы. Правильнее считать векторные инструкции частным случаем SIMD-инструкций. В строгом смысле векторной инструкцией можно назвать векторный аналог скалярной инструкции, выполняющий одно и то же вычисление над соответствующими данными (см. рисунок).

К векторным инструкциям естественно также отнести инструкции, выполняющие свертку (редукцию) вектора, например вычисление суммы элементов вектора или значения максимального элемента. Обратными, в некотором смысле, к операциям редукции являются операции, заполняющие вектор копиями скалярного значения, последовательными целыми числами и т. п. Все такие инструкции могут применяться как при ручной, так и при автоматической векторизации программы.

Вместе с тем в SIMD-расширении могут присутствовать инструкции, которые нельзя рассматривать в качестве векторных, несмотря на то что данные, которые они обрабатывают, находятся на векторных регистрах. Например, инструкции для распаковки мультимедийных данных или инструкции, ускоряющие реализации криптографических алгоритмов.



Скалярная (*а*) и векторная (*б*) инструкции

Такие команды иногда можно успешно задействовать при ручном программировании на ассемблере. Однако следует отметить, что на настоящее время они бесполезны при применении средств автоматической векторизации.

На начальном этапе основным методом создания программ, использующих векторные расширения процессоров, была ручная векторизация, осуществляемая путем программирования на ассемблере или использования ассемблерных вставок либо встраиваемых функций в программах на языке высокого уровня. Однако в силу трудоемкости этого подхода, а также трудностей с портируемостью этот подход не мог удовлетворить растущих потребностей в создании векторизованных приложений. Возникла необходимость разработки средств автоматической векторизации, арсенал которых в настоящее время представлен векторизующими компиляторами (gcc [13], llvm [14], Open64 [15], Intel C/C++ Compiler [16], xlc [17]), векторизующими препроцессорами (Scout [18], SAC [19], SWARP [20]), осуществляющими векторизацию программ путем их трансформации на уровне входного языка, а также рядом разработок в области динамической и JIT (*Just-in-Time*) векторизации программ [21, 22].

Теория и методы автоматической векторизации программ развивались начиная с 1970-х гг. в связи с разработкой компиляторов для классических векторных процессоров [23]. Изначально именно эти подходы внедрялись в векторизующие компиляторы для процессоров с векторными расширениями [24]. Однако в силу существенных отличий современных векторных расширений от классических векторных процессоров (малый размер векторов, отличия в системах команд, а также в характере типичных программ) существующие методы оказались для них недостаточно эффективными. Соответственно возникла потребность в пересмотре принципов и методов векторизации программ [25]. В условиях постоянно растущего числа различных векторных расширений и их модификаций насущной задачей также является разработка универсальных инструментальных средств векторизации, допускающих настройку на специфику целевой архитектуры.

Способы использования векторных и SIMD-расширений

Векторные и мультимедийные расширения в современных процессорных архитектурах предоставляют потенциальную возможность многократного увеличения производительности программ. Однако практическая польза от них может быть получена только при условии, что SIMD-инструкции действительно использованы, по крайней мере, в "горячих" участках программы.

Задействовать векторные инструкции можно разными способами, существенно отличающимися трудоемкостью программирования, условиями применимости, теоретически и практически достижи-

мой эффективностью, а также портируемостью (т. е. легкостью адаптации программы для иных процессорных архитектур и векторных расширений).

Классификация этих методов в порядке возрастания трудоемкости их использования может выглядеть следующим образом.

- Использование библиотек, предоставляющих эффективные реализации функций, выполнение которых занимает значительную часть времени работы программы.
- Полностью автоматическая векторизация программ средствами векторизующего компилятора.
- Автоматическая векторизация с использованием дополнительной информации, предоставляемой пользователем.
- Директивная векторизация (на основе стандарта OpenMP [26], Intel Cilk Plus [27] или некоторого специфичного языка директив [18]). В этом случае программист явно указывает циклы, которые необходимо векторизовать, возможно, дополнительно указывая некоторые параметры либо гарантируя важные для векторизации свойства обрабатываемых данных (например, выровненность массивов или кратность числа итераций длине векторных регистров).
- Использование специальных языков программирования (Fortran 90 [28], Cilk) или языковых расширений (Intel Cilk Plus). Задача трансляции изначально векторной программы в машинный код, использующий заданное векторное расширение, существенно проще, чем векторизация скалярной программы.
- Ручная реализация векторизованной программы на языке высокого уровня с использованием встроенных векторных типов и встроенных функций, соответствующих векторным инструкциям.
- Реализация важных с точки зрения производительности участков на ассемблере с использованием всех предоставляемых аппаратурой возможностей.

Все виды автоматической, полуавтоматической и директивной векторизации могут быть реализованы либо в составе оптимизирующего компилятора, либо как отдельный инструмент, трансформирующий программу на языке высокого уровня в другую программу, как правило, на том же языке высокого уровня.

Векторизация на уровне исходного кода имеет преимущества по сравнению с векторизацией, реализованной непосредственно в компиляторе. В частности, программист может инспектировать векторизованный код в целях оценки его качества для проведения экспериментов с параметрами векторизации и, возможно, для того, чтобы делать исправления вручную. Тем самым векторизация на уровне исходного кода вплотную примыкает к ручному векторному программированию на языке высокого уровня.

В настоящей статье будут обсуждены преимущества и ограничения некоторых из перечисленных выше подходов: использование библиотек; автоматическая векторизация; ручное векторное программирование на языке высокого уровня и на ассемблере. Также будут рассмотрены требования к системе команд, необходимые для эффективной векторизации программ. Средствам директивной векторизации и векторизации на уровне исходного кода авторы планируют посвятить отдельную публикацию.

Использование библиотек

Очевидно, что наилучший способ задействовать возможности векторных и SIMD расширений — пользоваться библиотеками, в которых все необходимые алгоритмы эффективно реализованы с полным учетом возможностей, предоставляемых аппаратурой.

Многие библиотеки в области научных и инженерных расчетов [28—30] реализованы для различных целевых платформ и поддерживают современные векторные расширения. Производители микропроцессоров также предоставляют библиотеки, оптимизированные для соответствующих архитектур и векторных расширений [31, 32]. Использование библиотек, предоставляющих векторные реализации базовых для разрабатываемого приложения алгоритмов, может обеспечить уровень производительности, близкий к теоретически максимальному. Преимущество этого подхода состоит в том, что он не требует от прикладного программиста очень высокой квалификации и знания используемых векторных расширений. Однако возможности его применения ограничены тем обстоятельством, что далеко не всегда доступны библиотеки, предоставляющие все необходимые функциональные возможности и поддерживающие необходимый набор целевых платформ.

Тем не менее, когда есть принципиальная возможность использования качественно реализованной библиотеки, этот путь, безусловно, оказывается предпочтительным, причем не только по причинам низкой трудоемкости и хорошей (во многих случаях) портируемости. Высокая эффективность библиотечной реализации может быть следствием не только использования векторного расширения, но и реализации хороших алгоритмов, тщательно адаптированных к специфике конкретной аппаратуры. Примером такой специфики может служить иерархия кэш-памяти и способы управления ее содержимым.

В то же время для создания новой библиотеки, доработки существующей или ее настройки на определенную целевую архитектуру и/или векторное расширение приходится, как правило, пользоваться методами ручной векторизации. Это исключительно трудоемкая задача, требующая высокой квалификации, которая предполагает как доскональное знание процессорной архитектуры, так и глубокое понимание особенностей решаемой задачи и используемых алгоритмов [33—35].

Автоматическая векторизация программ

Изначально методы автоматической векторизации программ развивались в компиляторах для классических векторных суперкомпьютеров. Типичный подход [23] включал выявление векторизуемых циклов, выполнение ряда трансформаций и генерацию векторного кода. Возможность векторизации оператора

в цикле (или гнезде циклов) определяется отсутствием циклических зависимостей для него. Цикл может быть векторизован, если он может быть трансформирован в набор операторов над вырезками массивов в формате языка Fortran 90. Например, цикл

```
for (i = 0; i < 4096; i++)
      Z[i] = X[i]+Y[i];
```

может быть векторизован, так как он эквивалентен оператору

```
Z[0:4095] = X[0:4095] + Y[0:4095].
```

Основной задачей, на решении которой были сосредоточены усилия разработчиков, являлся анализ зависимостей, которые могли быть препятствием для векторизации, а также реализация различных вспомогательных трансформаций, позволяющих приводить циклы к полностью векторизуемому виду. Этот метод применялся в компиляторе gcc версий 4.0.x [24], а также в некоторых векторизующих препроцессорах.

Недостатком этого подхода является тот факт, что он неприменим для векторизации развернутых вручную циклов, линейных участков, а также вычислений, использующих не массивы, а указатели. Ларсен предложил метод [25], основанный на парадигме SLP (*Superword Level Parallelism* — параллелизм на уровне суперслов, содержащих упакованные векторы), схожей с парадигмой ILP (*Instruction Level Parallelism* — параллелизм на уровне команд). Алгоритм SLP-векторизации включает следующие несколько шагов.

Шаг 1. Развертка с кратностью, равной длине векторов (только для циклических участков и если цикл не был развернут вручную).

Шаг 2. Анализ адресов памяти на предмет выровненности.

Шаг 3. Выявление групп инструкций чтения/записи, обращающихся к соседним элементам в памяти или элементам, расположенным с фиксированным шагом.

Шаг 4. Дополнение выявленных групп операций чтения/записи связанными с ними вычислительными операциями. Это дополнение реализуется путем обхода графа определений/использований. В результате формируются группы изоморфных выражений (в которых одноименные операции следуют в одинаковом порядке).

Шаг 5. Замена скалярных типов данных и операций на векторные типы и операции.

Рассмотрим шаги применения SLP-векторизации к циклу из примера, приведенного выше.

Шаг 1. После развертки:

```
for (i = 0; i < 4096; i += 4)
   {
     sX0 = ld(&X[i+0]);
     sY0 = ld(&Y[i+0]);
     sZ0 = sX0+sY0;
     st(sZ0, &Z[i+0]);
               ...
     sX3 = ld(&X[i+3]);
     sY3 = ld(&Y[i+3]);
     sZ3 = sX3+sY3;
     st(sZ3, &Z[i+3]);
   }
```

Шаг 2. После выявления групп изоморфных операций и упаковки:

```
for (i = 0; i < 4096; i += 4) {
   (sX0,sX1,sX2,sX3) = ld(&X[i+0:i+3]);
   (sY0,sY1,sY2,sY3) = ld(&Y[i+0:i+3]);
   (sZ0,sZ1,sZ2,sZ3) = (sX0,sX1,sX2,sX3)
                     + (sY0,sY1,sY2,sY3);
   st((sZ0,sZ1,sZ2,sZ3), &Z[i+0:i+3]);
}
```

Шаг 3. После замены скалярных операций векторными:

```
for (i = 0; i < 4096; i += 4)
  {
    vX = vec_ld(&X[i]);
    vY = vec_ld(&Y[i]);
    vZ = vec_add(vX, vY);
    vec_st(vZ, &Z[i]);
  }
```

Этот подход применим к циклам, развернутым вручную, а также к линейным участкам. Например, он позволяет выполнять векторизацию линейных участков, реализующих начальное заполнение небольших массивов исходными значениями.

Векторизация кода по методу SLP используется в версиях компилятора gcc [36] начиная с 4.4.x, а также в компиляторе llvm [14]. В версии llvm 3.7 используется усовершенствованный вариант SLP, получивший название PSLP (Padded SLP) [37], который позволяет векторизовать более широкий класс программных участков, содержащих "почти изоморфные" группы операций. Применение SLP-векторизации для участков, содержащих условные переходы, рассмотрено в работе [38]. В работе [39] предложена модификация SLP, позволяющая векторизовать гнезда циклов, содержащие зависимости во внутреннем цикле.

В работах [25, 40] рассмотрены также упрощенные модификации метода SLP, которые применимы только к циклам и основываются на том факте, что при развертке циклов обычно автоматически получаются группы изоморфных операций. Для векторизации циклов, развернутых вручную, компилятор должен предварительно выполнить свертку цикла.

Метод SLP и его модификации позволяют эффективно векторизовать "хорошие" циклы, в частности, циклы, работающие с выровненными данными, расположенными в последовательных позициях памяти, не имеющие зависимостей по данным, не содержащие условных ветвлений и вызовов функций и т. д. Если же эти (и некоторые другие) ограничения не

выполнены или недоступна информация, позволяющая судить об их выполнении, возникают осложнения. Компилятор может:

— сгенерировать невекторизованную и векторизованную версии цикла (или несколько векторизованных версий), а также динамические проверки свойств данных, по результатам которых выбирается одна из версий;

— сгенерировать недостаточно эффективно векторизованный цикл;

— полностью отказаться от векторизации цикла.

В работе [41] приведены результаты тестирования средств автоматической векторизации в компиляторах gcc, icc, xlc и дан подробный анализ причин, по которым векторизация не выполняется или оказывается неэффективной. Согласно результатам этой работы в реальных программах автоматически векторизуется от 18 до 30 % циклов, а в специальных синтетических наборах тестов — от 45 до 71 %.

Представление о возможностях, эффективности и ограничениях автоматической векторизации можно получить также из публикаций [42, 43], где рассмотрены примеры циклов с подробным анализом эффективности ассемблерного кода, сгенерированного компиляторами gcc и icc для векторных расширений Intel.

Результаты этих и других работ, а также опыт использования векторизующих компиляторов показывают, что добиться эффективной векторизации приложений, полагаясь только на возможности автоматической векторизации, как правило, нельзя. Следует также учесть, что средства автоматической векторизации не позволяют использовать различные специализированные SIMD-команды, которые могут поддерживаться процессором. Поэтому при разработке высокоэффективных приложений в дополнение к полностью автоматической векторизации должны применяться также другие средства и методы. В их числе: директивы информационной поддержки векторизации [44]; директивная векторизация [26, 18], векторные языки и языковые расширения [27]; хорошо векторизуемые алгоритмические решения [45, 35], а также ручная векторизация программ, рассмотрению которой посвящены следующие два раздела статьи.

Ручная реализация векторной программы на языке высокого уровня

Поскольку программирование на языке высокого уровня намного проще, чем на ассемблере, то эффективным представляется подход, основанный на использовании возможностей векторных и мультимедийных расширений в программе на языке высокого уровня.

Векторные инструкции можно вставлять в код программы в виде ассемблерных вставок. Преимущество этого подхода заключается в том, что он ничего не требует от компилятора — нужно лишь, чтобы векторные команды были правильно обработаны ассемблером. Однако, во-первых, если компилятор "ничего не знает" про векторные регистры, то он не в состоянии отследить зависимости между разными ассемблерными вставками и может некорректно их переставить. Во-вторых, для многих компиляторов (например, gcc) само использование ассемблерных вставок затрудняет анализ программы и препятствует ее оптимизации. Остается один вариант: объединять ассемблерные вставки, например, оформляя весь внутренний цикл как одну ассемблерную вставку. На практике сделать это не проще, чем реализовать эту же часть программы в виде ассемблерной функции в отдельном файле. На самом деле написание ассемблерных вставок оказывается даже сложнее, поскольку требуется аккуратно описать интерфейсы вставок, позаботиться о том, чтобы были свободны используемые во вставках регистры и т. д. Поэтому использование в программе на языке высокого уровня ассемблерных вставок, содержащих векторные инструкции, нельзя считать удачным подходом.

Таким образом, эффективное использование векторного (мультимедийного) расширения требует поддержки со стороны компилятора даже в том случае, когда векторизация программы выполняется программистом. Обычно компилятор предоставляет программисту типы данных, соответствующие возможному содержимому векторных регистров, и встроенные функции для их обработки. В большинстве случаев эти функции однозначно соответствуют инструкциям векторного расширения. Чтобы было понятно, о чем идет речь, рассмотрим пример:

```
typedef float V4SF
      __attribute__((vector_size (16)));
void foo (V4SF a[1024], V4SF b[1024]) {
  for (int i = 0; i<1024; i++)
    a[i] = __builtin_ia32_rsqrtps (b[i]);
}
```

Используя расширения языка С, определяем тип `V4SF`, представляющий векторы из четырех компонент типа `float`. Здесь `__builtin_ia32_rsqrtps` — встроенная функция, вычисляющая обратные величины квадратного корня из компонент вектора (соответствует команде SSE `rsqrtps`). Заметим, что в данном случае чтение из памяти векторов и запись в память не требуют вызова встроенной функции, так как компилятор сам правильно генерирует эти команды.

Этот подход предпочтительнее ассемблерных вставок, однако программист может столкнуться с тем, что само использование языка высокого уровня имеет ограничения, препятствующие получению эффективного кода. Анализ сложностей, связанных с ручным программированием на языке высокого уровня, заслуживает внимания. Все, что мешает эффективному использованию векторных команд вручную, неизбежно проявится при использовании инструментальных средств векторизации на уровне исходного языка. Перечислим основные затрудне-

ния, возникающие при ручном векторном программировании на языке высокого уровня.

- В векторные и мультимедийные расширения могут входить инструкции, смысл которых нельзя естественно свести к действию встроенной функции. Примером могут служить инструкции, разрушающие свои аргументы, или инструкции, вычисляющие несколько результатов, такие, например, как `swap` или `addsub`. В традиционных языках программирования (C, FORTRAN) функция может возвращать лишь один результат. Здесь заметим, что во многих современных языках программирования нет ограничений на число результатов. Сложности могут возникать с битами условий: языки программирования высокого уровня не содержат такого понятия, поэтому, если смысл команды сводится к выработке битов условий, ей трудно поставить в соответствие встроенную функцию. Такие же трудности возникают при использовании битов условий, особенно в командах переходов.
- Потери эффективности возможны также в силу неудачного распределения регистров. Для достижения максимальной производительности программист иногда вынужден выбирать алгоритм, использующий все доступные векторные регистры. Например, в программах умножения матриц в некоторых случаях только это позволяет сбалансировать скорости арифметических вычислений и чтения из памяти. Программист может ожидать, что определенный набор переменных будет постоянно находиться на регистрах, однако нет гарантий, что компилятор обеспечит ожидаемое распределение регистров. Использование спиллинга (сохранения регистров в памяти и их восстановление) или лишних пересылок между регистрами может снизить производительность в 2—3 раза. Важно, что все это может происходить даже тогда, когда регистров на самом деле хватает и спиллинг не требуется.
- Еще одной причиной неэффективности кода могут оказаться сгенерированные компилятором лишние пересылки. По-видимому, все три отмеченных затруднения (неточное описание инструкций встроенными функциями, неудачное распределение регистров и лишние пересылки) связаны между собой. Неудачное распределение регистров само по себе влечет дополнительные копирования регистров, а если действие команды не сводится к вычислению значения функции, то предоставляемая компилятором встроенная функция изначально может содержать подразумеваемое копирование регистров. В частности, использование инструкций, разрушающих свои аргументы, провоцирует компилятор сделать копии этих аргументов, тем самым используется лишний регистр и проводится лишняя пересылка.

В принципе, эти копирования, как и использование лишних регистров, могут быть в дальнейшем удалены генератором кода. Но эффективность этой оптимизации зависит от деталей реализации компилятора, в частности от взаимодействия механизмов распределения регистров, подбора команд и их планирования.

- Критически значимые потери производительности могут происходить в силу того, что обрабатываемые данные не попадают вовремя в кэш-память. Для тех случаев, когда структура используемого алгоритма не обеспечивает эффективного использования кэш-памяти, предусмотрен механизм предвыборки (`prefetch`). Команда предвыборки позволяет указать, что потребуется чтение или запись некоторых данных. При программировании на ассемблере можно разместить команду `prefetch` в правильном месте (за определенное число команд до чтения или записи либо в цикле выполнять `prefetch` для данных, требуемых на следующей итерации). Однако в семантике языков высокого уровня предвыборка не отражена. Хотя многие компиляторы позволяют вставлять `prefetch` при помощи встроенных функций, толку от этого немного. Во-первых, плохо учитывается связь между предвыборкой и последующим обращением в память, в результате не гарантируется, что `prefetch` попадет в код в правильное место. Во-вторых, пытаясь это частично исправить, компилятор трактует предвыборку как действие с неясным смыслом и может отменить агрессивные оптимизации, связанные с этим участком программы.

Конечно, предвыборка важна в любых программах, от которых требуется высокая производительность. Следует отметить, что для SIMD-вычислений это особенно важно, поскольку высокая скорость арифметических вычислений требует соответствующей скорости доступа к памяти.

Рассмотрим языковые средства, которые могли бы способствовать разрешению некоторых из перечисленных затруднений. Для борьбы со спиллингом полезен атрибут `strict _ register`, гарантирующий, что данный объект всегда будет храниться на регистре. Если компилятор не может без спиллинга разместить все `strict _ register`-данные на регистрах, то он обязан сообщить об ошибке.

Некоторые расширения языка высокого уровня не являются необходимыми, так как их эффект может быть достигнут вручную. Тем не менее эти расширения облегчили бы программирование. Например, маленькие массивы размещаемые на регистрах (`strict _ register` для массивов). При этом предполагается полная развертка циклов, обрабатывающих такие массивы. Аналогично можно было бы ввести атрибут `strict _ register` для структур, подразумевая при этом, что все поля такой структуры по отдельности хранятся на регистрах (структура рассматривается как коллекция переменных, а не область памяти).

Дальнейшее совершенствование методов задания SIMD-инструкций встроенными функциями должно помочь с использованием сложных инструкций (например, с несколькими результатами).

К сожалению, трудно предложить способ управления предвыборкой данных в языках высокого уровня. Либо понятие кэш-памяти должно быть как-то включено в семантику языка, либо необходим способ описывать заранее последовательность

операций с памятью (например, в духе коллекций и итераторов).

Следует отметить, что необходимость явно использовать команды предвыборки возникает не всегда. Во многих случаях структура используемого алгоритма такова, что механизмы работы кэш-памяти автоматически обеспечивают приемлемо малую частоту промахов в кэш. Некоторые современные процессоры способны автоматически выполнять предвыборку для ожидаемых операций доступа к памяти. Если программная предвыборка все-таки необходима, то следует рассмотреть возможность реализации критически важных участков программы на ассемблере.

Причина всех описанных сложностей заключается в том, что ключевые понятия всех, даже самых новых императивных языков высокого уровня сформировались в самом начале развития вычислительной техники (конец 1950-х — начало 1960-х гг.). Авторы этих концепций формулировали их, отталкиваясь от конструкции вычислительных машин своего времени. Тем самым, эти языки хороши для компьютеров 50-летней давности и не очень удобны для эффективного использования в современной параллельной и гетерогенной вычислительной среде.

Программирование на ассемблере

Очевидно, что, реализуя критичный по производительности участок программы на ассемблере, квалифицированный программист способен добиться максимально возможной эффективности, использовав все возможности векторных и мультимедийных расширений. Это позволяет рассматривать производительность качественной ассемблерной программы как оценку сверху для производительности решающих эту же задачу программ, векторизованных тем или иным способом, и, соответственно, оценивать другие методы векторизации по тому, какую долю производительности векторизованной вручную ассемблерной программы удалось получить.

Для подобной оценки не всегда необходима реальная ассемблерная программа. Если, во-первых, известна процессорная архитектура, включая время доступа к памяти, пропускную способность шин, размеры и структуру кэш-памятей и т. д., и во-вторых, хорошо изучена задача, тогда может быть известно "узкое место", которое и ограничивает производительность любой программы, решающей данную задачу. В таком случае можно полагать, что производительность оптимальной ассемблерной программы приближается на входных данных большого размера к этому ограничению (пропускной способности "узкого места").

Есть и вторая причина, которая оправдывает изучение вопросов, связанных с ассемблерной реализацией векторных программ. Все затруднения ("узкие места", несбалансированность, отсутствие полезных команд), которые обнаруживаются при программировании на ассемблере, остаются и при использовании подходов более высокого уровня. Конечно, используя векторизующий компилятор, программист может ничего не знать, например, о неэффективной реализации невыровненной записи вектора в память, но возникающие трудности от этого не исчезают. При каждом повышении уровня, на котором реализуется программа, возникают свои дополнительные трудности, которые препятствуют эффективной векторизации. При этом все "узкие места" низкоуровневого программирования остаются. Поэтому имеет смысл перечислить свойства векторного расширения, способствующие его эффективному использованию.

Требования к системе команд векторного расширения

Для успешного использования векторного расширения следует предусматривать в нем максимально полный набор векторных аналогов стандартных бинарных и унарных операций. Это особенно важно, если невозможен свободный доступ к компонентам векторных регистров как к скалярным регистрам. В случае бинарных операций также полезны смешанные инструкции с одним векторным и одним скалярным операндом.

Серьезным ограничением для использования векторного расширения оказывается неудобная реализация перемещений векторов и их компонент между векторными регистрами, скалярными регистрами и памятью. Должны быть предусмотрены возможности собирать вектор из скалярных значений, разбирать обратно, а также копировать отдельные компоненты векторов. Важно иметь возможность заполнять регистр копиями скалярного значения. Для копирования векторов в память и обратно полезны инструкции, позволяющие отображать компоненты вектора в память со смещением (*strided access*), команды доступа к памяти по вектору адресов (*gather/scatter*), а также эффективно выполняющиеся команды чтения/записи по невыровненным адресам. Желателен также достаточный набор инструкций, выполняющих переупаковки вектора. Переупаковка нужна, в частности, для решения вопросов, связанных с невыровненностью: для вычленения невыровненного векторного значения из двух соседних выровненных.

В векторном расширении должна в том или ином виде поддерживаться if-конверсия, т. е. замена ветвлений (условных передач управления) средствами условного вычисления. Это может быть поддержка векторных предикатов, позволяющих условно выполнять любые операции над компонентами вектора или хотя бы только операции условного копирования векторов. Под условным копированием в данном случае понимается параллельное выполнение присваиваний:

```
C[i] = (if A[i] then B[i] else C[i]).
```

В дополнение к предикатным вычислениям или условному копированию полезны также предикаты

vec _ any, vec _ all, позволяющие в некоторых случаях более эффективно реализовать векторизацию участков программы, содержащих ветвления [38]. Рассмотрим пример векторизации цикла, иллюстрирующий применение предиката vec _ any.

Исходный цикл:

```
for (i = 0; i < 1024; i++)
  {
   if (A[i] > 0)
   {
      C[i] = B[i];
      if (B[i] < 0)
        D[i] = E[i];
   }
  }
```

После if-конверсии, векторизации и вставки векторных ветвлений:

```
for (i = 0; i < 1024; i += 4) {
   if (vec _ any _ gt(A[i:i+3],(0,0,0,0)))
   {
     vPA = A[i:i + 3] > (0,0,0,0);
     C[i:i+3] = vec _ sel(C[i:i+3],B[i:i+3],vPA);
     vT = B[i:i+3] < (0,0,0,0);
     vPB = vec _ sel((0,0,0,0), vT, vPA);
     if (vec _ any _ ne(vPB,(0,0,0,0)))
       D[i:i+3] = vec _ sel(D[i:i+3],E[i:i+3],vPB);
   }
}
```

При любых SIMD-вычислениях должны быть запрещены прерывания, вызванные некорректными аргументами или результатами (деление на ноль, квадратный корень из отрицательного числа, переполнения). Вместо этого векторное расширение должно корректно обрабатывать "бесконечности" и "нечисла" (NaN). Возможность выполнять некорректные операции необходима для того, чтобы можно было заменять ветвления операциями условного копирования векторов. Конечно, если в командном слове есть место, то можно предусмотреть два варианта SIMD-инструкций: вызывающих и не вызывающих прерывания, например, можно кодировать способность вызывать прерывания определенным битом в командном слове.

Еще одно более частное замечание, касающееся так же, как и предыдущее, не только векторных, но и любых SIMD-расширений. Во многих алгоритмах результаты операций умножения используются в последующих операциях сложения или вычитания (иногда и сложения, и вычитания). Отсюда следует, что имеет смысл у всех мультипликативных SIMD-инструкций, на которых предполагается добиться максимальной производительности, предусматривать применение аддитивной операции к каждому результату умножения. В частности, в векторном расширении, кроме векторного аналога операции a=b*c, должны предусматриваться векторная операция a+=b*c и, желательно, a-=b*c.

При проектировании SIMD-расширения следует учитывать, что обладающие спецификой сложные инструкции (например, матричные, а не векторные в буквальном смысле) могут успешно применяться при ручном векторном программировании на ассемблере или на языке высокого уровня, но существующие средства автоматической и директивной векторизации не способны их использовать.

Заключение

Исходя из представленных в настоящей статье соображений, можно сделать вывод, что в настоящее время не существует универсального подхода к использованию SIMD-расширений, который был бы применим ко всем целевым архитектурам и всем задачам, требующим высокой производительности. Во всяком случае, не следует ожидать, что использование векторизующего компилятора и процессора с векторным расширением автоматически позволит в несколько раз ускорить работу программы. Современные компиляторы способны векторизовать в среднем не более 30 % циклов в реальных программах, причем не всегда наиболее эффективным способом.

На практике оправданно сочетание разных методов. В первую очередь следует максимально полно использовать эффективно реализованные библиотечные функции. Например, в задачах линейной алгебры одно только использование SIMD-команд в принципе не позволяет добиться высокой производительности, если не согласована скорость загрузки данных и выполнения арифметических вычислений, а также не учтены структура и размеры кэш-памяти всех уровней [34].

Далее возможно применение векторизующего компилятора или автоматических и директивных векторизаторов в сочетании с современными средствами профилирования [46]. При этом может потребоваться пересмотр некоторых алгоритмических и программных решений для эффективного применения средств векторизации [41, 44, 45].

Если все перечисленные выше подходы не позволяют добиться требуемой производительности (или же подобные средства просто отсутствуют), то после профилирования программы можно вручную использовать SIMD-инструкции, программируя на ассемблере или на языке высокого уровня.

В заключение отметим, что возможности как автоматической векторизации, так и ручного векторного программирования существенно зависят от полноты набора команд, представленных в векторном расширении. Как следствие, важное значение имеют вопросы проектирования таких расширений и эффективные методы "программной компенсации" отсутствия тех или иных аппаратных средств.

Список литературы

1. **Векторные** вычислительные системы. Новосибирский Государственный Университет. URL: http://www.sbras.ru/win/elbib/data/show_page.dhtml?77+843 (дата обращения 27.11.2015).

2. **Gustafson J. L., Hawkinson S., Scott K.** The Architecture of a Homogeneous Vector Supercomputer. URL: http://johngustafson.net/pubs/pub1986.5/Arch.ofaVectorSupercomputer.pdf (дата обращения 27.11.2015).

3. **Mittal M., Peleg A., Weiser U.** MMX™ Technology Architecture Overview. URL: http://www.smtnet.com/library/files/upload/mmx-technology.pdf (дата обращения 27.11.2015).

4. **3DNow!™** Technology Manual. AMD. URL: http://support.amd.com/TechDocs/21928.pdf (дата обращения 27.11.2015).

5. **AltiVec** Technology Programming Interface Manual. URL: http://www.freescale.com/files/32bit/doc/ref_manual/ALTIVECPIM.pdf (дата обращения 27.11.2015).

6. **Intel®** SSE4 Programming Reference. URL: https://software.intel.com/sites/default/files/m/8/b/8/D9156103.pdf (дата обращения 27.11.2015).

7. **Lomont C.** Introduction to Intel® Advanced Vector Extensions. URL: https://software.intel.com/sites/default/files/m/d/4/1/d/8/Intro_to_Intel_AVX.pdf (дата обращения 27.11.2015).

8. **ARM.** The architecture for Digital World®. NEON. URL: http://www.arm.com/products/processors/technologies/neon.php (дата обращения 27.11.2015).

9. **MIPS** SIMD Programming White Paper. URL: https://imagination-technologies-cloudfront-assets.s3.amazonaws.com/documentation/MD00927-SIMD-programming-whitepaper.pdf (дата обращения 27.11.2015).

10. **TMS320C6000** CPU and Instruction Set Reference. (Rev.F). Texas Instruments, Inc., 2000.

11. **SC140** DSP Core Reference Manual. Semiconductor, Inc. 2004.

12. **General-Purpose** TigerSHARC™ Processor. URL: http://www.analog.com/media/en/news-marketing-collateral/product-highlight/4667563674187GPTigerSHARC(B)Final.pdf (дата обращения 27.11.2015).

13. **GCC**, the GNU Compiler Collection. URL: http://gcc.gnu.org (дата обращения 27.11.2015).

14. **The LLVM** Compiler Infrastructure. URL: http://llvm.org (дата обращения 27.11.2015).

15. **x86** Open64 Compiler Suite. URL: http://developer.amd.com/tools-and-sdks/cpu-development/x86-open64-compiler-suite/ (дата обращения 27.11.2015).

16. **Intel®** C++ and Fortran Compilers. URL: https://software.intel.com/en-us/intel-compilers (дата обращения 27.11.2015).

17. **XL** C/C++ for Linux. URL: http://www.ibm.com/developerworks/downloads/r/xlcpluslinux/ (дата обращения 27.11.2015).

18. **Scout** — A Source-to-Source Transformator for SIMD Optimizations. URL: http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/scout/index_html_en?set_language=en (дата обращения 27.11.2015).

19. **Guelton S., Irigoin F., Keryell R.** SAC: An Efficient Retargetable Source-to-Source Compiler for Multimedia Instruction Sets. URL: http://www.cri.ensmp.fr/classement/doc/A-429.pdf (дата обращения 27.11.2015).

20. **Pokam G., Bihan S., Simmonet J., Bodin F.** SWARP — A Retargetable Preprocessor for Multimedia Instructions // Concurrency and Computation: Practice and Experience. 2004. N. 16 (2—3). P. 303—318.

21. **Pajuelo A., Gonzalez A., Valero M.** Speculative Dynamic Vectorization //Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA'02). 2002. P. 271—280.

22. **Nuzman D., Dyshel S., Rohou E.** et al. Vapor SIMD: Auto-Vectorize Once, Run Everywhere // Proceedings of the CGO 2011, The 9th International Symposium on Code Generation and Optimization. April 2—6, 2011. Chamonix, France. 2011. P. 151—160.

23. **Kennedy K., Allen J. R.** Optimizing compilers for modern architectures: a dependence-based approach. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. 2001.

24. **Naishlos D.** Autovectorization in GCC // The 2004 GCC Developers' Summit. 2004. P. 105—118.

25. **Larsen S., Amarasinghe S.** Exploiting Superword Parallelism with Multimedia Instruction Sets // Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation (PLDI 2000). Vancouver, British Columbia, Canada. 2000. P. 145—156.

26. **OpenMP** Application Program Interface. Version 4.0. July 2013. URL: http://openmp.org/mp-documents/OpenMP4.0.0.pdf (дата обращения 27.11.2015).

27. **Intel** Cilk™ Plus. URL: https://www.cilkplus.org/ (дата обращения 27.11.2015).

28. **FFTW.** URL: http://www.fftw.org/index.html (дата обращения 27.11.2015).

29. **GotoBLAS2.** Texas Advanced Computing Center. URL: https://www.tacc.utexas.edu/research-development/tacc-software/gotoblas2 (дата обращения 27.11.2015).

30. **SLEEF** — SIMD Library for Evaluating Elementary Functions. URL: http://shibatch.sourceforge.net (дата обращения 27.11.2015).

31. **Intel®** Math Kernel Library (Intel® MKL). URL: https://software.intel.com/en-us/intel-mkl (дата обращения 27.11.2015).

32. **ACML —** AMD Core Math Library. URL: http://developer.amd.com/tools-and-sdks/archive/amd-core-math-library-acml (дата обращения 27.11.2015).

33. **Goto K., Van de Geijn R. A.** Anatomy of high-performance matrix multiplication // ACM Trans. Math. Softw. 2008. Vol. 34, N. 3. Article 12.

34. **Frigo M., Johnson S. G.** The Design and Implementation of FFTW3 // Proceedings of IEEE. 2005. Vol. 93, N. 2. P. 216—231.

35. **Shibata N.** Efficient evaluation methods of elementary functions suitable for SIMD computation// Computer Science — Research and Development. 2010. Vol. 25, Issue 1. P. 25—32.

36. **Rosen I., Nuzman D., Zaks A.** Loop-Aware SLP in GCC, GCC summit, July 2007. URL: https://gcc.gnu.org/wiki/HomePage?action=AttachFile&do=view&target=GCC2007-Proceedings.pdf (дата обращения 27.11.2015).

37. **Porpodasy V., Magniz A., Jones T. M.** PSLP: Padded SLP Automatic Vectorization // IEEE/ACM International Symposium on Code Generation and Optimization. 2015. P. 190—201.

38. **Shin J., Hall M., Chame J.** Superword-level parallelism in the presence of control flow // Proceedings of the International Symposium on Code Generation and Optimization. 2005. P. 165—175.

39. **Tenllado C., Pinuel L. P., Prieto M.** et al. Improving Superword Level Parallelism Support in Modern Compilers // Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. 2005. P. 303—308.

40. **Волконский В., Дроздов А., Ровинский Е.** Метод использования мелкоформатных векторных операций в оптимизирующем компиляторе // Информационные технологии и вычислительные системы. 2004. № 3. С. 63—77.

41. **Maleki S., Gao Y., Garzaran M. J.** et al. An Evaluation of Vectorizing Compilers. URL: http://polaris.cs.uiuc.edu/~garzaran/doc/pact11.pdf (дата обращения 27.11.2015).

42. **Auto-vectorization** with gcc 4.7. Lockless, Inc. URL: http://locklessinc.com/articles/vectorize/ (дата обращения 27.11.2015).

43. **GCC 4.8** vs ICC 14.0 Update 1 auto-vectorization analysis. URL: https://software.intel.com/sites/default/files/managed/98/b4/gcc_icc_auto-vectorization_comparison.pdf (дата обращения 27.11.2015).

44. **A Guide** to Vectorization with Intel® C++ Compilers. URL: https://software.intel.com/sites/default/files/8c/a9/CompilerAutovectorizationGuide.pdf (дата обращения 27.11.2015).

45. **Satish N., Kim C., Chhugani J.** et al. Can Traditional Programming Bridge the Ninja Performance Gap for Parallel Computing Applications? URL: http://web.eecs.umich.edu/~msmelyan/papers/isca-2012-paper.pdf (дата обращения 27.11.2015).

46. **Intel®** Advisor. URL: URL: https://software.intel.com/en-us/intel-advisor-xe (дата обращения 27.11.2015).

Exploiting Vector Extensions of Modern Processors

N. I. Vyukova, e-mail: niva@niisi.ras.ru, **V. A. Galatenko,** e-mail: galat@niisi.ras.ru,
S. V. Samborskij, e-mail: sambor@niisi.ras.ru, Federal State Institution "Scientific Research Institute of System Analysis of the Russian Academy of Science", Moscow, 117218, Russian Federation

Corresponding author:

Galatenko Vladimir A., Head of Sector, Federal State Institution "Scientific Research Institute of System Analysis of the Russian Academy of Science", Moscow, 117218, Russian Federation
e-mail: galat@niisi.ras.ru

Received on December 11, 2015
Accepted on December 18, 2015

Modern architectures increasingly rely on SIMD vectorization to improve performance for multimedia applications, various kinds of scientific applications, database processing. Compiler-based automatic vectorization seems to be a simple way to exploit vector processing in application code. However experience and recent research show that modern vectorizing compilers are capable of vectorizing only a small part of loops in real-life applications. Therefore programmers have to resort to various kinds of other means and tools for vectorization of program code.

The paper provides a brief overview of common ways for exploiting vector instructions in application code. Some of these approaches are then discussed in more detail, namely the methods of automatic vectorization in modern compilers, use of libraries, manual writing of vector code both in high-level programming languages and in assembly. Special attention is paid to the properties of vector instruction set which help or impede efficient use of vector computations in manual and automatic vectorization.

Keywords: *vector extensions, SIMD (Single Instruction Multiple Data), automatic vectorization, SLP (Superword Level Parallelism)*

For citation:

Vyukova N. I., Galatenko V. A., Samborskij S. V. Exploiting Vector Extensions of Modern Processors, *Programmnaya Ingeneria*, 2016, vol. 7, no. 4, pp. 147—157.

DOI: 10.17587/prin.7.147-157

References

1. **Vektornye** vychislitel'nye sistemy (Vector Computing Systems), Novosibirskij Gosudarstvennyj Universitet, available at: http://www.sbras.ru/win/elbib/data/show_page.dhtml?77+843 (in Russian).

2. **Gustafson J. L., Hawkinson S., Scott K.** The Architecture of a Homogeneous Vector Supercomputer, available at: http://johngustafson.net/pubs/pub1986.5/Arch.ofaVectorSupercomputer.pdf

3. **Mittal M., Peleg A., Weiser U.** MMX™ Technology Architecture Overview, available at: http://www.smtnet.com/library/files/upload/mmx-technology.pdf

4. **3DNow!™** Technology Manual. AMD, available at: http://support.amd.com/TechDocs/21928.pdf

5. **AltiVec** Technology Programming Interface Manual, available at: http://www.freescale.com/files/32bit/doc/ref_manual/ALTIVECPIM.pdf

6. **Intel®** SSE4 Programming Reference, available at: https://software.intel.com/sites/default/files/m/8/b/8/D9156103.pdf

7. **Lomont C.** Introduction to Intel® Advanced Vector Extensions, available at: https://software.intel.com/sites/default/files/m/d/4/1/d/8/Intro_to_Intel_AVX.pdf

8. **ARM.** The architecture for Digital World®. NEON, available at: http://www.arm.com/products/processors/technologies/neon.php

9. **MIPS** SIMD Programming White Paper, available at: https://imagination-technologies-cloudfront-assets.s3.amazonaws.com/documentation/MD00927-SIMD-programming-whitepaper.pdf

10. **TMS320C6000** CPU and Instruction Set Reference. (Rev.F). Texas Instruments, Inc., 2000.

11. **SC140** DSP Core Reference Manual. Semiconductor, Inc. 2004.

12. **General-Purpose** TigerSHARC™ Processor, available at: http://www.analog.com/media/en/news-marketing-collateral/product-highlight/4667563674187GPTigerSHARC(B)Final.pdf

13. **GCC,** the GNU Compiler Collection, available at: http://gcc.gnu.org

14. **The LLVM** Compiler Infrastructure, available at: http://llvm.org

15. **x86** Open64 Compiler Suite, available at: http://developer.amd.com/tools-and-sdks/cpu-development/x86-open64-compiler-suite/

16. **Intel®** C++ and Fortran Compilers, available at: https://software.intel.com/en-us/intel-compilers

17. **XL** C/C++ for Linux, available at: http://www.ibm.com/developerworks/downloads/r/xlcpluslinux/

18. **Scout** — A Source-to-Source Transformator for SIMD Optimizations, available at: http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/scout/index_html_en?set_language=en

19. **Guelton S., Irigoin F., Keryell R.** SAC: An Efficient Retargetable Source-to-Source Compiler for Multimedia Instruction Sets, available at: http://www.cri.ensmp.fr/classement/doc/A-429.pdf.

20. **Pokam G., Bihan S., Simmonet J., Bodin F.** SWARP — A Retargetable Preprocessor for Multimedia Instructions, *Concurrency and Computation: Practice and Experience,* 2004, no. 16 (2—3), pp. 303—318.

21. **Pajuelo A., Gonzalez A., Valero M.** Speculative Dynamic Vectorization, *Proceedings of the 29th Annual International Symposium on Computer Architecture (ISCA'02)*, 2002, pp. 271—280.

22. **Nuzman D, Dyshel S., Rohou E., Rosen I., Williams K., Yuste D., Cohen A., Zaks A.** Vapor SIMD: Auto-Vectorize Once, Run Everywhere, *Proceedings of the CGO 2011, The 9th International Symposium on Code Generation and Optimization*, April 2—6, 2011. Chamonix, France, 2011, pp. 151—160.

23. **Kennedy K., Allen J. R.** *Optimizing compilers for modern architectures: a dependence-based approach*, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. 2001.

24. **Naishlos D.** Autovectorization in GCC, *The 2004 GCC Developers' Summit*, 2004, pp. 105—118.

25. **Larsen S., Amarasinghe S.** Exploiting Superword Parallelism with Multimedia Instruction Sets, *Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation (PLDI 2000),* Vancouver, British Columbia, Canada. 2000, pp. 145—156.

26. **OpenMP** Application Program Interface. Version 4.0. July 2013, available at: http://openmp.org/mp-documents/OpenMP4.0.0.pdf

27. **Intel** Cilk^{TM} Plus, available at: https://www.cilkplus.org/

28. **FFTW**, available at: http://www.fftw.org/index.html

29. **GotoBLAS2.** Texas Advanced Computing Center, available at: https://www.tacc.utexas.edu/research-development/tacc-software/gotoblas2

30. **SLEEF** — SIMD Library for Evaluating Elementary Functions, available at: http://shibatch.sourceforge.net

31. **Intel®** Math Kernel Library (Intel® MKL), available at: https://software.intel.com/en-us/intel-mkl

32. **ACML** — AMD Core Math Library, available at: http://developer.amd.com/tools-and-sdks/archive/amd-core-math-library-acml

33. **Goto K., Van de Geijn R. A.** Anatomy of high-performance matrix multiplication, *ACM Trans. Math. Softw.*, 2008, vol. 34, no. 3, Article 12.

34. **Frigo M., Johnson S. G.** The Design and Implementation of FFTW3, *Proceedings of IEEE*, 2005, vol. 93, no. 2, pp. 216—231.

35. **Shibata N.** Efficient evaluation methods of elementary functions suitable for SIMD computation, *Computer Science — Research and Development*, 2010, vol. 25, issue 1, pp. 25—32.

36. **Rosen I., Nuzman D., Zaks A.** Loop-Aware SLP in GCC, *GCC summit*, July 2007, available at: https://gcc.gnu.org/wiki/HomePage?action=AttachFile&do=view&target=GCC2007-Proceedings.pdf

37. **Porpodasy V., Magniz A., Jones T. M.** PSLP: Padded SLP Automatic Vectorization, *IEEE/ACM International Symposium on Code Generation and Optimization*, 2015, pp. 190—201.

38. **Shin J., Hall M., Chame J.** Superword-level parallelism in the presence of control flow, *Proceedings of the International Symposium on Code Generation and Optimization,* 2005, pp. 165—175.

39. **Tenllado C., Pinuel L. P., Prieto M., Tirado F., Catthoor F.** Improving Superword Level Parallelism Support in Modern Compilers, *Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis,* 2005, pp. 303—308.

40. **Volkonskij V., Drozdov A., Rovinskij E.** Metod ispol'zovanija melkoformatnyh vektornyh operacij v optimizirujushhem kompiljatore (A Method of Using Short Vector Operations in Optimizing Compiler), *Informacionnye Tehnologii i Vychislitl'nye Sistemy*, 2004, no. 3, pp. 63—77 (in Russian).

41. **Maleki S., Gao Y., Garzaran M. J. Wong T., Padua D. A**. An Evaluation of Vectorizing Compilers, avnilable at: URL: http://polaris.cs.uiuc.edu/~garzaran/doc/pact11.pdf

42. **Auto-vectorization** with gcc 4.7. Lockless, Inc., available at: http://locklessinc.com/articles/vectorize/

43. **GCC 4.8** vs ICC 14.0 Update 1 auto-vectorization analysis, available at: https://software.intel.com/sites/default/files/managed/98/b4/gcc_icc_auto-vectorization_comparison.pdf

44. **A Guide** to Vectorization with Intel® C++ Compilers, available at: https://software.intel.com/sites/default/files/8c/a9/CompilerAutovectorizationGuide.pdf

45. **Satish N., Kim C., Chhugani J., Saito H., Krishnaiyer R., Smelyanskiy M., Girkar M., Dubey P.** Can Traditional Programming Bridge the Ninja Performance Gap for Parallel Computing Applications? available at: http://web.eecs.umich.edu/~msmelyan/papers/isca-2012-paper.pdf

46. **Intel®** Advisor, available at: https://software.intel.com/en-us/intel-advisor-xe

ИНФОРМАЦИЯ

УДК 004.75 + 004.41 DOI: 10.17587/prin.7.158-166

М. Кайко-Маттсон, Associate Professor, e-mail: mekm2@kth.se,
Школа информационных и телекоммуникационных технологий,
Королевский технологический институт KTH, Стокгольм, Швеция

Преподавание стандарта ESSENCE: минимизация усилий, максимизация результатов на выходе

(перевод – канд. физ.-мат. наук А. В. Галатенко, e-mail: agalat@msu.ru)

Описан четырехлетний опыт преподавания ESSENCE одним из создателей этого стандарта. Стандарт ESSENCE преподавался в рамках программы бакалавриата Королевского технологического института KTH на протяжении четырех лет в курсе, посвященном выполнению программного проекта. Представлена история преподавания ESSENCE, описан процесс развития обучающих материалов на протяжении четырех лет, приведены мнения студентов об ESSENCE, а также проанализировано отношение студентов к этому стандарту.

Ключевые слова: *ESSENCE, ядро, альфа-атрибут, стандарт группы OMG, обучение в области программной инженерии*

Введение

Как в индустрии производства программного обеспечения, так и в академическом сообществе распространено мнение, что выпускники профильных вузов, несмотря на полученное университетское образование, слабо подготовлены к своей дальнейшей карьере в качестве программных инженеров. В индустрии в течение многих лет жалуются на то, что даже аспиранты не владеют в достаточной степени ни знаниями, ни навыками в области программной инженерии [4, 6, 12]. В то же время специализированные факультеты вузов в течение многих лет пытаются бороться со сложившейся практикой, когда большинство тем из области программной инженерии спрессованы в один или два курса бакалавриата [11, 12, 18].

Имеется много причин слабой подготовки студентов к их дальнейшей профессиональной деятельности в области программной инженерии. Первая причина состоит в том, что в рамках обучения много времени уделяется смежным темам, с которыми программной инженерии приходятся явно или неявно конкурировать. К подобным "конкурентам" относятся теоретическая информатика, высшая математика, искусственный интеллект, операционные системы, базы данных, сети и ряд других тем [3]. Вторая причина заключается в том, что программная инженерия представляет собой обширную и постоянно расширяющуюся предметную область. Перманентно возникают новые подходы к проблемному моделированию и анализу, проектированию программного обеспечения, управлению процессами жизненного цикла и др. [7]. Третья причина состоит в следующем. Помимо качественного образования, в становлении полноценного зрелого инженера в области создания и сопровождения программ важную роль играет еще и глубокое понимание всех хитросплетений программной инженерии, а также наличие значительного практического опыта, в том числе и негативного [8, 20].

Даже если бы в рамках образовательных программ было бы выделено больше времени на обучение в области инженерии программ, то все равно бы остались некоторые трудности. Существует ряд сложных и специализированных тем, включающих в себя сильно взаимосвязанные технические и нетехнические части [11], которые сложны как для изложения преподавателем, так и для понимания студентом. Также в процессе обучения невозможно адекватно смоделировать производственные условия, в рамках которых накапливается реальный опыт в области программной инженерии. Подобному опыту нельзя научить, он может быть только самостоятельно наработан на практике. Существует также огромный разрыв [18] между двумя видами деятельности — промышленной разработкой программного обеспечения и написанием учебных программ. Например, студенты никогда не будут поставлены перед необходимостью решать задачи с нечеткими формулировками и неясными требованиями, также они не будут иметь возможности работать с унаследованным программным кодом и с сопровождением программного обеспечения. Наконец, студенты просто никогда не смогут изучить целиком всю область программной инженерии

в рамках учебной образовательной программы [18] подготовки бакалавров.

Сократить разрыв между учебной и реальной разработкой непросто, наверное, даже просто невозможно. Необходимо идти на компромиссы. В настоящей публикации описывается такой компромисс, основанный на недавно разработанном стандарте программной инженерии ESSENCE [5]. На настоящий момент стандарт ESSENCE преподается уже четыре года в рамках специального курса, входящего в программу бакалавриата в области информационных и телекоммуникационных технологий в Королевском технологическом институте KTH. В работе описан четырехлетний опыт преподавания этого курса, представлено выбранное авторами курса решение вопросов, обусловленных нехваткой обучающего материала. Кроме того, дан анализ отношения студентов к курсу, приведены их мнения о курсе.

1. Курс программной инженерии в Королевском технологическом институте KTH

Автор настоящей статьи читала специальный курс по программной инженерии для студентов-второкурсников, обучающихся по специальности "Информационные и коммуникационные технологии" (*Information and Communication Technology*, CINTE) [2]. В настоящее время курс называется "Современная разработка программного обеспечения", однако до 2014 г. он назывался "Проект в области информационных технологий. Часть 1" (IT Project. Part 1) [14]. Как видно на рис. 1, этот курс тесно связан с курсом под названием "Проект в области информационных технологий. Часть 2" (IT Project. Part 2). Оба курса составляют образовательный комплекс, освещающий область программной инженерии как с теоретической, так и с практической точек зрения.

Курсы тесно переплетены и преподаются в три этапа. Как показано на рис. 1, на первом этапе (Phase 1) студенты приобретают теоретические познания, готовясь таким образом к участию в проектах по разработке программного обеспечения. Длительность первого этапа составляет всего две недели. На втором этапе (Phase 2) студенты применяют полученные теоретические знания на практике, разрабатывая программные и/или интегрированные программно-аппаратные системы. Длительность второго этапа составляет четыре недели. Занятия в рамках второго этапа ведут другие преподаватели. Наконец, на третьем этапе (Phase 3) студенты оценивают разработки, сделанные на втором этапе, с точки зрения теории программной инженерии. Длительность третьего этапа составляет три недели.

Не нужно быть специалистом в области программной инженерии, чтобы понять, что длительность первого этапа очень мала. За две недели студентам необходимо ознакомиться с наиболее существенными компонентами программной инженерии и подготовиться к выполнению собственных проектов. Казалось бы, задача невыполнима в рамках традиционных подходов к обучению. На настоящее время не существует учебников, написанных на достаточно высоком, вводном уровне, предназначенных для изучения этой темы за указанный срок. Учитывая, сколь широка область программной инженерии, единственным адекватным подходом к преподаванию представляется извлечение информации с самой поверхности. Однако одного лишь такого поверхностного изложения недостаточно, требуется выбрать нужную "поверхность" и извлекать информацию в нужном направлении.

2. История преподавания стандарта ESSENCE

В этом разделе представлена история преподавания стандарта ESSENCE по годам, начиная с 2012 г. и заканчивая 2015 г.

2012 год. В марте 2012 г. автор совместно с Иваром Якосбсоном (*Ivar Jacobson*) приняли решение о том, что стандарт ESSENCE должен быть включен в обучение программной инженерии. Это был своего

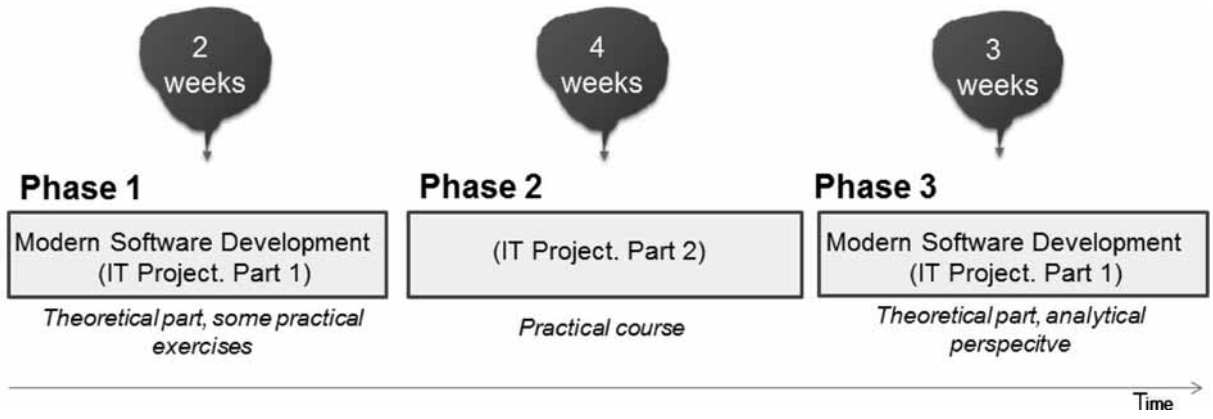


Рис. 1. Комплекс курсов по выполнению программного проекта в Королевском технологическом институте KTH

рода договор о намерениях. В то время никто в мире не преподавал стандарт ESSENCE, были только доклады, посвященные ESSENCE, на различных международных конференциях [9]. Не было и обучающих материалов. Тем не менее мы решили включить ESSENCE в первый же курс по программной инженерии. Таким курсом оказался "Проект в области информационных технологий. Часть 1" (*IT Project, Part 1*) в Королевском технологическом институте KTH.

Решение о преподавании стандарта ESSENCE казалось тогда слишком смелым и авантюрным, ведь на разработку обучающих материалов оставалось всего несколько недель, а нам не хотелось стать саботажниками, разрушившими отличный курс, который развивался в течение многих лет. Более того, нам предстояли две поездки на конференции во время подготовки и чтения курса. Если трезво взглянуть на ситуацию, казалось, что мы напрашиваемся на неприятности. Однако мы были полны энтузиазма, чтобы обращать на это внимание.

Мы решили давать на лекциях подробные презентации, посвященные ядру ESSENCE[1]. Лекции включали в себя общий взгляд на ядро и его структуру, а также обсуждение всех альфа-атрибутов (*Abstract-Level Progress Health Attribute*, ALPHA)[2] и состояний этих атрибутов. Этот материал должен был равномерно распределиться по лекциям так, что общая схема курса не должна была потребовать слишком большого числа изменений. Однако в силу цейтнота разработка курса свелась к копированию альфа-атрибутов на слайды. Примеры слайдов приведены на рис. 2, см. третью сторону обложки. Как видно на рис. 2, мы включили в презентации все альфа-атрибуты и все элементы контрольных списков для каждого состояния альфа-атрибутов. Проницательному читателю достаточно бросить быстрый взгляд на наши слайды, чтобы понять, что с педагогической точки зрения презентация далека от идеала и не очень удобна для преподавания.

После того как лекции по ядру ESSENCE были дочитаны, стало понятно, что, несмотря на все наши усилия по подробному изложению альфа-атрибутов ядра и их состояний, у студентов возникли трудности с усвоением материала за столь короткое время. Кроме того, в нашей презентации не хватало конкретных примеров и не было времени ни на их разработку, ни на их рассмотрение в рамках курса. В лучшем случае иногда удавалось формулировать критерии успешности с тем, чтобы студентам не пришлось проходить через трудоемкий процесс идентификации точных требований. В рамках такого подхода им не приходилось и управлять изменениями на лету в рамках разработки соответствующих примеров. Когда была закончена презентация по ядру, первый этап курса закончился и студенты перешли ко второму этапу.

Тем не менее битва за ознакомление студентов с ядром ESSENCE была еще не закончена. Опыт, полученный за время чтения лекций, показал, что непосредственное изложение материалов группы SEMAT (*Software Engineering Method and Theory* — Методология и теория программной инженерии) не слишком привлекало слушателей к стандарту ESSENCE. Не слишком радовало аудиторию и то обстоятельство, что они оказались первыми студентами в мире, изучающими ESSENCE. Наоборот, некоторые студенты были куда более заинтересованы в получении положительной оценки за курс, чем в получении знаний об инновациях в области программной инженерии.

В рамках продолжения начатого курса было решено разработать методичку. Студенты уже перешли ко второму этапу, и время стало нашим худшим врагом. Нам очень хотелось, чтобы студенты использовали ESSENCE для оценки своих проектов. Основной задачей было заставить студентов понять, что такое ESSENCE, в процессе интенсивной разработки собственных программно-инженерных проектов. Единственным решением, пришедшим в голову, было написание небольшого рассказа о программной разработке, который студенты смогли бы отождествить со своей разработкой и отобразить объекты из рассказа на альфа-атрибуты ядра ESSENCE.

Этот рассказ назывался "Уголок Греты". Героем рассказа был Адам, второкурсник, изучающий программную инженерию в Королевском технологическом институте KTH. Адам занимался разработкой системы заказа столиков в ресторане "Уголок Греты". Текст рассказа и отображение на альфа-атрибуты ядра ESSENCE доступны по требованию. Как видно на рис. 3, отображение задано в виде таблицы, в которой для каждого элемента контрольного списка альфа-атрибута приводится событие и/или состояние из рассказа "Уголок Греты". В третьем столбце указано, выполнен ли соответствующий элемент контрольного списка, в четвертом — причина, по которой элемент выполнен или не выполнен.

Рассказ "Уголок Греты" был написан и передан студентам в три приема. Было непросто написать простую историю, которая покрыла бы большинство альфа-атрибутов со всеми элементами контрольных списков. По мере продолжения рассказа всегда приходилось начинать анализ сначала, чтобы убедиться, что состояния всех альфа-атрибутов покрыты, причем согласованным образом. Так или иначе, рассказ

---

[1] Согласно определению, представленному в статье Якобсон И., Сейдевитц Э. Новая программная инженерия // Программная инженерия. 2015. № 5. С. 3—9, "Ядро можно рассматривать как минимальный набор элементов, которые являются универсальными для всех специалистов в сфере разработки программного обеспечения. Ядро включает минимальный набор сущностей, которые универсальны для всех способов организации деятельности по разработке программного продукта". *Прим. переводчика.*

[2] В соответствии с определением в статье Якобсон И., Сейдевитц Э. Новая программная инженерия // Программная инженерия. 2015. № 5. С. 3—9. "Термин альфа является изначально акронимом для атрибута прогресса и здоровья на абстрактном уровне. ...он, как отмечено в ядре, используется для определения степени прогресса и состояния здоровья в разработке продукта". *Прим. переводчика.*

был дописан и роздан студентам задолго до окончания второго этапа.

Однако просто раздать рассказ студентам было недостаточно. Пришло время убедиться, что студенты поняли отображение рассказа на альфа-атрибуты ядра ESSENCE. Это оказалось нетривиальной задачей. Большой вопрос состоял в том, каким образом ввести в курс этот рассказ так, чтобы не мешать собственно выполнению проекта на втором этапе.

Студенты работали, разбившись на команды по восемь или девять участников. Чтобы минимизировать нагрузку, связанную с изучением ESSENCE, было решено сделать каждого участника команды ответственным за один альфа-атрибут. Другими словами, студенту требовалось изучить только один альфа-атрибут, попытаться понять соответствие в рассказе "Уголок Греты" только для одного атрибута, а затем отобразить на этот атрибут состояние собственного проекта. Это отображение нужно было вычислять в письменном виде, в форме, аналогичной представленной на рис. 3, и непрерывно обновлять в процессе выполнения второго этапа. Мы проверяли значения отображения у всех студентов и персонально обсуждали со студентами результаты проверки.

Результаты студенческих работ с вычислением альфа-атрибутов оказались приятным сюрпризом. Оказалось, что никаких существенных трудностей с пониманием того, как это сделать, у студентов нет. Были разве что мелкие недопонимания некоторых вопросов программной инженерии, вызванные недостатком глубины познаний в этой области.

2013—2014 гг. В эти годы мы были лучше оснащены обучающими материалами и накопили некоторый опыт. У нас уже был рассказ "Уголок Греты", отображение рассказа на альфа-атрибуты и слайды, пусть и не лучшие, но готовые. Кроме того, в 2012 г. мы получили урок, который использовали для улучшения презентации по ESSENCE.

На первом этапе использовались имевшиеся учебные материалы без изменений. Внести существенные улучшения вновь не позволил цейтнот. Единственные изменения были направлены на улучшение понимания материала и на сокращение числа слайдов, посвященных альфа-атрибутам ядра ESSENCE. В 2013—2014 гг. мы концентрировали внимание на одном-двух начальных состояниях каждого альфа-атрибута и на их отображении на рассказ "Уголок Греты". Специально не были включены в презентацию все альфа-атрибуты и все их состояния. У студентов теперь был рассказ "Уголок Греты", так что они могли изучить материал самостоятельно.

Первый этап прошел очень гладко. Студенты очень хорошо подготовились к отслеживанию состояния собственных проектов с точки зрения альфа-

Status	Checklist	Yes/No	Explanation
Conceived	The initial set of stakeholders agrees that a system is to be produced.	Yes	All the restaurant staff and the owner have agreed that a computerized restaurant reservation system should be produced.
	The stakeholders that will use the new system are identified.	Yes	Yes, the restaurant staff will use the system.
	The stakeholders that will fund the initial work on the new system are identified.	Yes	Yes, it is Sven – the restaurant owner who will fund the initial work on the system.
	There is a clear opportunity for the new system to address.	Yes	Everybody agreed that there was a clear opportunity to be gained from developing a new system. Less booking mistakes would be made and the customers would be much happier.
Bounded	The stakeholders involved in developing the new system are identified.	Yes	It was only Adam who would develop the system. So, yes they have been identified.
	The stakeholders agree on the purpose of the new system.	Yes	Adam knows the purpose of the system. It is to manage restaurant bookings.
	It is clear what success is for	Yes	The success has not been explicitly stated. However,

Рис. 3. Отображение рассказа "Уголок Греты" на элементы контрольных списков альфа-атрибутов

атрибутов ядра. На втором этапе студенты без всяких трудностей пользовались контрольными списками элементов ESSENCE для отслеживания состояния и продвижения проекта.

2015 год. В этом году мы были подготовлены еще лучше. Было заготовлено два сценария и два комплекта раздаточных материалов, разработанных совместно с другими членами инициативы SEMAT в 2013—2014 гг. [13, 15]. Появилась также полная спецификация всех альфа-податрибутов (*subalphas*).

Первый сценарий заключался в инициализации проекта, связанного с масштабированием и модернизацией устаревшей системы [13]. В данном случае альфа-атрибуты ядра ESSENCE использовались для выяснения статуса проекта. В рамках второго сценария требовалось отслеживать состояние и продвижение проекта по разработке системы управления университетскими курсами. В этом случае альфа-атрибуты использовались для идентификации и исправления различных "болевых точек" проекта.

Ко второму сценарию прилагалось два комплекта раздаточных материалов с упражнениями в рамках изучения ядра стандарта ESSENCE [16, 19]. В каждом комплекте рассматривался только один альфа-атрибут — Требования и Команда (*Requirements and Team*). Цель использования этих материалов состояла в том, чтобы помочь студенту-новичку сфокусироваться на одном альфа-атрибуте и, таким образом, помочь ему лучше осознать практическое использование этого атрибута.

В 2015 г. практически полностью были переделаны слайды. Благодаря наличию сценариев стало возможным оптимизировать презентации. Мы сфокусировались на жизненном цикле каждого альфа-атрибута. Студенты были ознакомлены со сценариями. Наши студенты оказались первыми студентами в мире, которым преподают SEMAT-сценарии [13, 15].

Ознакомление студентов со сценариями проводилось в два приема. Сначала студенты прочитали первый сценарий, связанный с началом проекта. Была проведена проверка степени понимания студентами задач, а также оценена степень создания трудностей в их решении, что было задокументировано [13]. Существовала двойная цель: помочь студентам понять, что из себя представляет оценка статуса проекта, и проверить, есть ли вопросы с пониманием. К нашему удивлению, вопросы с пониманием возникли у очень небольшого числа студентов. Некоторые студенты оказались настолько умными, что нашли несоответствия в документе.

Затем студентам предлагалось проделать те же действия, но уже для второго сценария, ориентированного на отслеживание продвижения и состояния проекта, и выполнить упражнения из двух комплектов раздаточных материалов [15, 16, 19]. Для каждого раздаточного материала требовалось определить состояние соответствующего альфа-атрибута по представленному описанию ситуации. И здесь мы вновь были приятно удивлены. Большинство студентов смогли с легкостью найти состояние каждого альфа-атрибута из раздаточных материалов.

В 2015 г. студенты готовились не только к отслеживанию продвижения и состояния своих проектов, но и к планированию проектов, идентификации проблемных вопросов и рисков при реализации проекта, а также к поиску решений и необходимых действий. Две фотографии студентов, работающих с альфа-атрибутами, представлены на рис. 4 (см. третью сторону обложки). В зависимости от цели карточки с альфа-атрибутами либо развешивались на стене, либо раскладывались на столе.

3. Отношение студентов к стандарту ESSENCE и высказанные мнения

В данном разделе описано отношение студентов к стандарту ESSENCE и приведены высказанные ими мнения. Первый подраздел посвящен отношению к использованию материалов группы SEMAT применительно к отдельным проектам. Во втором подразделе приведены точки зрения с позиции пользы использования ESSENCE.

Отношение студентов к стандарту ESSENCE

В рамках второго этапа реализации курса большинство студентов использовали методологию Scrum. Выбор делался самими студентами совместно с персоналом, преподававшим на втором этапе. Все студенты использовали альфа-атрибуты стандарта ESSENCE для отслеживания и оценивания состояния отдельных проектов. Во всех случаях каждый студент отвечал за отслеживание одного или двух альфа-атрибутов, а команда в целом отвечала за набор всех альфа-атрибутов.

Отношение студентов менялось с течением времени. Большинство студентов 2012 г. (60 %) не высказало однозначно положительного отношения к ESSENCE. Многие студенты посчитали ESSENCE обузой, ненужной задачей, которая только отнимает время и мешает выполнению проекта. Даже несмотря на то, что мы неоднократно повторяли, что ESSENCE — это не метод, некоторые студенты, казалось, не понимали или не хотели понять это. Были, однако, и студенты (40 %), которые восприняли ESSENCE с энтузиазмом и считали, что использование ESSENCE оказало огромную помощь в эффективном выполнении проектов.

С нашей точки зрения, столь широкий спектр студенческих мнений, высказанных в 2012 г., был вызван попыткой протеста против того, что студенты оказались первыми, кто был вынужден использовать нечто новое, на использованное больше никем в мире. В то время ESSENCE даже не имел статус стандарта. Кроме того, то, что учебные материалы разрабатывались в процессе чтения курса, не способствовало созданию положительного образа ESSENCE. Отдельные консервативно настроенные студенты говорили, что чувствуют себя подопытными кроликами, которых

заставляют делать вещи, о существовании которых другие студенты в мире даже не догадываются. Конечно же, в 2012 г. такие высказывания влияли и на мнения других студентов.

В 2013—2015 гг. отношение студентов радикально изменилось. Этому немало поспособствовало то обстоятельство, что ESSENCE стал международным стандартом. Большинство (80 %) студентов в 2013—2015 гг. уже относилось к ESSENCE положительно и с энтузиазмом.

Приблизительно у 10 % студентов, большинство которых посещало курс в 2013 г., отношение к ESSENCE было скептическое. Несмотря на наличие рассказа "Уголок Греты", некоторые из них утверждали, что контрольные списки кажутся им чем-то чужеродным. Им хотелось, чтобы в курсе было больше примеров с подробным объяснением. Студенты также жаловались, что отслеживание состояния выполнения проектов с помощью ESSENCE отнимает слишком много времени.

Оставшиеся 10 % относились к ESSENCE нейтрально. Они утверждали: так как раньше им не приходилось участвовать в проектах по разработке программного обеспечения, у них нет уверенности, что использование ESSENCE привносит какие-то положительные моменты. У них просто не было эталона, на основе которого можно было бы сформировать более осознанное мнение. Однако они соглашались, что ESSENCE предоставляет необходимые знания о том, на что нужно обращать внимание при сопровождении проекта.

Мнения студентов об ESSENCE

Большинство студентов 2013—2015 гг. придерживалось точки зрения, что ESSENCE — хороший стандарт, предоставляющий отличное описание проектов. Этот стандарт помог им структурировать работу над проектом и позволил получить информацию, в каком направлении работы над проектом они могли бы поучаствовать и/или улучшить собственные навыки. Интересно, что студенты указывали на то, что альфа-компоненты очень хорошо дополняют друг друга. Многие контрольные элементы при этом совпадали. Однако каждый контрольный элемент изучался отдельно с позиции рассматриваемого альфа-атрибута.

Замечание студентов о повторяющихся контрольных элементах позволило понять, что альфа-атрибуты сильно взаимосвязаны с позиции обеспечения полноты покрытия области программной инженерии. Несмотря на это, альфа-атрибуты можно использовать и в слабо связном режиме, рассматривая их по отдельности. По мнению студентов, все альфа-атрибуты были очень хорошо структурированы и описаны и ни один атрибут при этом не казался бесполезным с точки зрения контроля за ходом выполнения проекта в целом. Однако некоторые состояния атрибутов казались избыточными в рамках выполняемых проектов, так как были недостижимыми. В качестве примера такого состояния можно привести состояние "Снятый с эксплуатации" атрибута "Программная система".

Студенты высказывали мнение, что стандарт ESSENCE просто понять и что многие элементы контрольных списков интуитивно понятны даже для людей, не обладающих опытом работы по участию в выполнении проектов. Некоторые студенты говорили, что если даже они, неопытные студенты, смогли успешно использовать ядро ESSENCE, то человек с опытом работы точно смог бы сделать это. Тем не менее студенты утверждали, что некоторые элементы контрольных списков трудно понять или трудно однозначно трактовать. Для некоторых студентов такое понимание потребовало существенных усилий и дальнейшего осмысления.

Многие студенты признавали, что альфа-атрибуты составляют своего рода дорожную карту области программной инженерии. Альфа-атрибуты помогли им увидеть, какие составляющие проекта являются существенными. Таким образом, студенты выучили и поняли, на что нужно обращать внимание при работе над проектом. Альфа-атрибуты позволили не только определять состояние выполнения студенческих проектов, но и найти текущие начальные точки и определить следующие действия в рамках проектов. В результате работа над проектами оказалась более эффективной.

Небольшое число студентов считали, что полностью использовать весь потенциал стандарта ESSENCE в рамках курса невозможно, так как студенческие проекты слишком малы. Этот стандарт куда полезнее в больших проектах, в которых возникают трудности с отслеживанием разнообразных аспектов состояния их выполнения. В маленьких проектах следование стандарту ESSENCE и отслеживание состояний всех альфа-атрибутов представлялось избыточным. Особенно обременительным это казалось в проектах со сжатыми сроками их выполнения. Однако такой вывод противоречил мнению большинства студентов, считавших, что временные затраты на изучение ESSENCE с лихвой окупились. Большинство студентов соглашалось с тем, что на понимание ESSENCE пришлось затратить некоторые усилия, однако в результате, по окончании проекта, общее количество потраченных на его выполнение ресурсов уменьшилось. Время, потраченное на ESSENCE, стало своего рода задатком, существенно улучшившим итоговое качество.

Стандарт ESSENCE высоко оценили студенты как с опытом работы над реальными программными проектами, так и без такого опыта. Стандарт позволил им выделять жизненно важные аспекты проектов. В результате студенты, имевшие опыт работы над реальными проектами, смогли проанализировать процессы выполнения своих прошлых проектов и выделить имевшиеся там проблемные вопросы. Такие студенты высказывали мнение, что если бы в рамках программно-инженерных проектов

использовались контрольные списки альфа-атрибутов, вероятность успешного выполнения проектов, несомненно, повысилась бы. Неопытные студенты ценили ESSENCE за наличие элементов контрольных списков, позволявших управлять своими проектами. Такие студенты считали, что наличие элементов контрольных списков, выполнение которых нужно проверять, лучше, чем ситуация, когда ничего нет и нужно придумывать, что делать дальше. В любом случае, большинство студентов смогло найти многочисленные достоинства в стандарте ESSENCE. Перечислим некоторые из них.

- ESSENCE — это стандарт, написанный из соображений минимизации усилий, поддерживающий непрерывную оценку состояния выполнения проекта. Он превосходит существующие методы благодаря полноте покрытия аспектов, характеризующих такое состояние, которые существенны для всех программных проектов. Таким образом, его следует использовать во всех программных проектах для обеспечения качества разработки.
- ESSENCE — это единая основа, которая пригодна и для больших, и для маленьких проектов, и для опытных, и для неопытных разработчиков. Он позволяет почувствовать уверенность, что различные шаги проекта хорошо сбалансированы, а все существенные аспекты учтены.
- Время, потраченное на изучение ESSENCE, окупится в будущем. Положительные знания и навыки, полученные в результате такого изучения, разнообразны. Стандарт ESSENCE позволяет структурировать работу над проектом, сделать ее более систематической. Он позволяет выделить важные детали, которые легко пропустить, помогает обнаружить проблемные вопросы на ранних этапах выполнения проекта, перечисляет варианты следующих далее действий и, таким образом, к завершению проекта экономит существенный объем ресурсов.
- Пройдя по всему списку контрольных элементов альфа-атрибутов, студенты получают уверенность в том, что не пропущена ни одна существенная деталь в работе с проектом. Стандарт ESSENCE можно рассматривать как страховой полис, гарантирующий, что проект полностью определен, а все существенные аспекты рассмотрены. Чтобы обеспечить успешное выполнение проекта, студенты должны просто проследить, что все элементы контрольных списков просмотрены. Таким образом, ESSENCE можно рассматривать как путь проекта к успеху в его выполнении. Студенты объясняют это тем, что стандарт ESSENCE был разработан не одним человеком, а стал результатом международного сотрудничества специалистов как из академической области, так и из промышленной сферы.
- Стандарт ESSENCE облегчает взаимодействие. Взаимодействие требуется во всех проектах, в то же время для правильной организации такого взаимодействия необходимо понимание того, что именно является его предметом. Как отмечали многие студенты, ESSENCE помог определить точки взаимодействия и даже помог это взаимодействие структурировать.
- Стандарт ESSENCE облегчает отслеживание состояния проекта. Он также способствует распределению работы. Даже если бы все альфа-атрибуты были тесно взаимосвязаны, их отслеживание все равно может быть легко распределено между участниками проекта. Таким образом, общий объем усилий на управление всеми альфа-атрибутами уменьшается. Это обстоятельство также помогает в распределении работы между участниками выполнения проекта.
- Стандарт ESSENCE делает проект более зримым, таким образом повышая понимание сложности работы над проектом. Он стимулирует студентов погружаться в работу над своими проектами, благодаря чему повышается производительность труда и удовлетворение от проделанной работы.
- Стандарт ESSENCE может применяться и к другим типам проектов, не из сферы программной инженерии. По мнению некоторых студентов, ряд компонентов ядра может быть исключен и адаптирован к другому типу проекта. Таким образом, спектр применимости ESSENCE может быть существенно расширен.
- С помощью стандарта ESSENCE может быть повышено качество существующих проектов за счет увеличения познаний участников проекта и за счет выделения аспектов, в улучшении которых проект нуждается.
- Стандарт ESSENCE особенно важен для проектов, в которых нет разработанной методологии. Он помогает участникам группы выполнения проекта избежать многих проблемных вопросов и ловушек, о которых другим способом было бы сложно узнать.

Как видно из представленного выше списка, студенты смогли найти большое число достоинств в стандарте ESSENCE. Некоторые студенты выражали опасение, что ESSENCE может ошибочно восприниматься как просто еще один метод, авторы которого пытаются превзойти другие методы. Важно понимать, что ESSENCE — это не конкурент, а дополнение существующих методов, предназначенное для повышения контролируемости состояния проекта.

Заключение

Опыт, полученный в 2012—2015 гг., позволяет сделать вывод, что стандарт ESSENCE может успешно применяться в среде образовательных проектов. Этот стандарт помогает убедиться, что все важнейшие подобласти сферы программной инженерии рассмотрены в рамках курса. Даже если не удается раскрыть эти подобласти достаточно глубоко, все они так или иначе упоминаются в курсе, и студенты узнают об их существовании. Стандарт ESSENCE также помог переделать части лекционного материала, сделать

лекции более структурированными. Кроме того, были покрыты все существенные подобласти программной инженерии, что способствовало лучшей подготовке студентов к нуждам реальных проектов.

В рамках сравнения ESSENCE с другими стандартами, такими как [1, 17], было обнаружено, что все еще остаются пробелы, не закрытые ни стандартами [1, 17], ни другими обучающими материалами. С помощью ESSENCE стало возможным уменьшить эти пробелы и сделать преподавание программной инженерии более полным. Таким образом, можно утверждать, что ESSENCE как инструментарий в образовании обладает огромным потенциалом. Созданный из соображений минимизации усилий, он тем не менее выделяет существенные аспекты, на которые должны обращать внимание участники выполнения проектов.

Потребовалось некоторое время, чтобы студенты приняли ESSENCE. Будем верить, что это произошло не только потому, что стандарт ESSENCE был принят консорциумом OMG (*Object Management Group* — Группа управления объектами), но и благодаря усилиям по развитию образовательных материалов и аккуратной переработке курса. Написание рассказа "Уголок Греты" сильно помогало студентам понять альфа-атрибуты в течение первых трех лет чтения курса. Создание группой SEMAT специальных сценариев оказалось отличным дополнением, позволяющим лучше понять альфа-атрибуты и их разнообразное использование в проектах.

Усилия, потраченные на введение стандарта ESSENCE в образовательный процесс, однозначно окупились. Так как ESSENCE открывает знания программной инженерии с совершенно нового угла зрения на состояние в этой области, возникает искушение сделать вывод, что для введения ESSENCE в образовательный процесс потребуется приложить существенные усилия по переработке курсов и созданию учебного материала. Однако в силу интуитивной ясности стандарта ESSENCE усилия по его внедрению в учебный процесс существенно минимизируются. Как видно из данной публикации, весь объем программной инженерии может быть преподан даже для новичков. Конечно, с точки зрения программной инженерии это всего лишь проход по "поверхности", однако есть уверенность, что, несмотря на малую глубину, это было погружение в правильном направлении. С минимальными усилиями был получен максимальный выход. Студентам была преподнесена вся область программной инженерии всего за две недели. Без стандарта ESSENCE было бы невозможно настолько широко осветить область программной инженерии за столь короткое время.

Список литературы

1. **ACM** Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. A Volume of the Computing Curricula Series, SE2004. URL: http://sites.computer.org/ccse/

2. **CINTE** — Degree Programme in Information and Communication Technology (CINTE) at KTH Royal Institute of Technology. URL: http://www.kth.se/student/kurser/program/CINTE/HT13/kurslista?l=en

3. **Comber T., Lo B., Watson R.** Achieving balance in software engineering curricula // Proceeding of the international Conference on Software Engineering Education and Practice, SEEP'96. IEEE Computer Society Press, 1996. P. 271—278.

4. **Crnkovic I., Land R., Sjogren A.** Is software engineering training enough for software engineers? // Proceeding of 16th Conference on Software Engineering Education and Training, CSEE&T 2003. 20—22 March 2003, IEEE, 2003. P. 140—147. DOI: 10.1109/CSEE.2003.1191371

5. **Essence** — Kernel and Language for Software Engineering Methods, URL: http://www.omg.org/spec/Essence/1.0/

6. **Finkelstein A.** Student problems in software engineering education // IEE Colloquium on Teaching of Software Engineering-Progress Reports. 1991. P. 1/1—1/3.

7. **Ghezzi C., Mandrioli D.** The challenges of software engineering education // Proceeding of International Conference on Software Engineering, ICSE. 2005. P. 637—638.

8. **Inverardi M, Jazayeri M.** Software Engineering Education in the Modern Age. Springer, 2006.

9. **Jacobson I., Huang S., Kajko-Mattsson S.** et al. Semat — Three Year Vision // Programming and Computer Software. 2012. Vol. 38. P. 1—12. DOI: 10.1134/S0361768812010021

10. **Jacobson I.** Article on Wikipedia. URL: https://en.wikipedia.org/wiki/Ivar_Jacobson

11. **Jalote P.** Teaching an Introductory Software Engineering Course in a Computer Science Program // Software Engineering Education and Training, 2009. CSEET '09. 22nd Conference on. IEEE, 2009. P. 7.

12. **Kajko-Mattsson M.** Towards a mobile software engineering education // 2010 International Conference on Information Society (i-Society). 2010. P. 529—535.

13. **Kajko-Mattsson M., Palank B., Myburgh B.** et al. A Scenario on Kick-Starting a Project. URL: http://semat.org/kick-starting-a-project-scenario

14. **Modern** Software Development 6.0 credits, Undergraduate course at KTH Royal Institute of Technology. URL: https://www.kth.se/student/kurser/kurs/IV1303?l=en

15. **Peraire C., Kajko-Mattsson M., Myburgh B.** et al. A Scenario on Solvning Pain Points. URL: http://semat.org/solving-pain-points-scenario.

16. **Peraire C., Kajko-Mattsson M., Myburgh B.** et al. Solving Pain Points with Team Alpha. URL: http://semat.org/scenario-supporting-handouts.

17. **SWEBOK.** Guide to the Software Engineering Body of Knowledge (SWEBOK). URL: http://www.swebok.org

18. **Vallino J.** What should students learn in their first (and often only) software engineering course? // Software Engineering Education and Training, CSEET '13, 26th Conference on. 19—21 May 2013. IEEE, 2013. P. 335—337. DOI: 10.1109/CSEET.2013.6595273

19. **Vierira Nelson M. A., Kajko-Mattsson M., Myburgh B.** et al. Solvning Pain Points with Team Alpha. URL: http://semat.org/scenario-supporting-handouts

20. **Yanchun Sun, Xuanzhe Liu.** Educating students by "real-world" software engineering practice — Designing software engineering practice course // 7th International Conference on Computer Science & Education (ICCSE). 14—17 Jule 2012. IEEE, 2012. P. 1356—1359. DOI: 10.1109/ICCSE.2012.6295315

Minimal Effort and Maximal Output of Teaching Essence
(translated by A. V. Galatenko)

Kajko-Mattsson M., e-mail: mekm2@kth.se, School of Information and Communication Technology, KTH Royal Institute of Technology, SE-100 44 STOCKHOLM, Sweden

Corresponding author:

Kajko-Mattsson Mira, Associate Professor, School of Information and Communication Technology, KTH Royal Institute of Technology, SE-100 44 STOCKHOLM, Sweden
e-mail: mekm2@kth.se

Received on January 20, 2016
Accepted on January 27, 2016

KTH Royal Institute of Technology Essence has been teaching ESSENCE on a bachelor degree level for four years. It started in 2012. At that time, ESSENCE was still in its cradle. The Alphas and its checklists were in a final polishing state and there were neither any teaching materials nor any other pedagogical tools for utilizing ESSENCE. Today, Essence has reached its full maturity and is equipped with educational material that is widely used worldwide. This paper gives an account of the four-year education of ESSENCE as experienced by one of the ESSENCE co-creators. The education took place on one project course on a bachelor degree level at KTH Royal Institute of Technology. The paper describes the history of teaching ESSENCE. It presents the development and evolution of educational material throughout the four years, and gives an account of students' attitudes towards and opinions about ESSENCE.

Keywords: *ESSENCE, Kernel, Alpha, OMG standard, software engineering education*

For citation:

Kajko-Mattsson M. Minimal Effort and Maximal Output of Teaching Essence, *Programmnaya Ingeneria*, 2016, vol. 7, no. 4, pp. 158—166.

DOI: 10.17587/prin.7.158-166

References

1. **ACM** Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A Volume of the Computing Curricula Series, SE2004, available at: http://sites.computer.org/ccse/

2. **CINTE** — Degree Programme in Information and Communication Technology (CINTE) at KTH Royal Institute of Technology, available at: http://www.kth.se/student/kurser/program/CINTE/HT13/kurslista?l=en

3. **Comber T., Lo B., Watson R.** Achieving balance in software engineering curricula, *Proceeding of the international Conference on Software Engineering Education and Practice, SEEP'96*, IEEE Computer Society Press, 1996, pp. 271—278.

4. **Crnkovic I., Land R., Sjogren A.** Is software engineering training enough for software engineers? *Proceeding of 16th Conference on Software Engineering Education and Training,* CSEE&T 2003, 20—22 March 2003, IEEE, 2003, pp. 140—147, doi: 10.1109/CSEE.2003.1191371

5. **Essence** — Kernel and Language for Software Engineering Methods, available at: http://www.omg.org/spec/Essence/1.0/

6. **Finkelstein A.** Student problems in software engineering education, *IEE Colloquium on Teaching of Software Engineering-Progress Reports*, 1991, pp. 1/1—1/3.

7. **Ghezzi C., Mandrioli D.** The challenges of software engineering education, *Proceeding of International Conference on Software Engineering*, (ICSE), 2005, pp. 637—638.

8. **Inverardi M, Jazayeri M.** *Software Engineering Education in the Modern Age*, Springer, 2006.

9. **Jacobson I., Huang S., Kajko-Mattsson S., McMahon M., Seymour P. E.**, Semat — Three Year Vision, *Programming and Computer Software*, 2012, vol. 38, pp. 1—12, doi: 10.1134/S0361768812010021

10. **Jacobson I.** Article on Wikipedia, available at: https://en.wikipedia.org/wiki/Ivar_Jacobson

11. **Jalote P.** Teaching an Introductory Software Engineering Course in a Computer Science Program, *Software Engineering Education and Training,* 2009, CSEET '09, 22nd Conference on, IEEE, 2009, pp. 7.

12. **Kajko-Mattsson M.** Towards a mobile software engineering education, *2010 International Conference on Information Society (i-Society)*, 2010, pp. 529—535.

13. **Kajko-Mattsson M., Palank B., Myburgh B., McMahon P. E., Peraire C., Menezez W.** *A Scenario on Kick-Starting a Project*, available at: http://semat.org/kick-starting-a-project-scenario

14. **Modern** Software Development 6.0 credits, Undergraduate course at KTH Royal Institute of Technology, available at: https://www.kth.se/student/kurser/kurs/1V1303?l=en

15. **Peraire C., Kajko-Mattsson M., Myburgh B., McMahon P. E., Menezez W., Palank B.** *A Scenario on Solvning Pain Points*, available at: http://semat.org/solving-pain-points-scenario

16. **Peraire C., Kajko-Mattsson M., Myburgh B., Vierira Nelson M. A., McMahon P. E.** *Solving Pain Points with Team Alpha*, available at: http://semat.org/scenario-supporting-handouts

17. **SWEBOK.** Guide to the Software Engineering Body of Knowledge (SWEBOK), available at: http://www.swebok.org

18. **Vallino J.** What should students learn in their first (and often only) software engineering course*? Software Engineering Education and Training*, CSEET '13, 26th Conference on, 19—21 May 2013, IEEE, 2013, pp. 335—337, doi: 10.1109/CSEET.2013.6595273

19. **Vierira Nelson M. A., Kajko-Mattsson M., Myburgh B., Peraire C., McMahon P. E.** *Solvning Pain Points with Team Alpha*, available at: http://semat.org/scenario-supporting-handouts

20. **Yanchun Sun, Xuanzhe Liu.** Educating students by "real-world" software engineering practice — Designing software engineering practice course, *7th International Conference on Computer Science & Education (ICCSE),* 14—17 Jule 2012, IEEE, 2012, pp. 1356—1359, doi: 10.1109/ICCSE.2012.6295315

УДК 004.045, 004.657 DOI: 10.17587/prin.7.167-172

О. В. Решетникова, канд. техн. наук, доц., e-mail: ov13r@yandex.ru,
Дальневосточный государственный университет путей сообщения, г. Хабаровск

Реализация NoClone-протокола обращения клиента к базе данных MS SQL Server

Предложен вариант реализации протокола для организации доступа программного клиента к базе данных, позволяющий минимизировать вероятность появления дубликатов записей. Взаимодействие проводится в формате односторонне инициируемого диалога путем обмена информационными пакетами фиксированной структуры. Процесс трансляции информационных пакетов включает обязательный этап их фильтрации на соответствие предписанным структурам. Передаваемый по протоколу пакет представляется при этом XML-объектом, а для физической его передачи используется транспортный механизм компонента Service Broker MS SQL Server. Реализация NoClone-протокола является одним из ключевых шагов к построению среды имитационного моделирования процессов взаимодействия программных объектов с базами данных.

Ключевые слова: *отсоединенная модель доступа к данным, дубликаты, протокол обращения к данным, XML, получение/отправка сообщений, Transact SQL, семантический анализ строки*

Введение

Современный программный продукт есть сложный технический объект, который проектируют, а затем строят из компонентов с одновременным применением различных технологий программирования и, соответственно, языков. Принятие для реализации той или иной технологии программирования определяется, в первую очередь, средой исполнения будущей программы. Организуется же среда исполнения в вычислительной системе так называемой программной платформой.

В настоящее время одной из наиболее востребованных среди разработчиков программных продуктов можно назвать платформу .NET Framework. Главная тому причина — платформа .NET Framework встроена в структуру операционной системы (ОС) MS Windows, а следовательно, любое управляемое такой ОС устройство в принципе способно запускать и исполнять построенную на данной платформе программу.

Для осуществления обращения программного продукта — клиента (далее — клиента) к реляционной базе данных платформа .NET Framework предлагает управляемую библиотеку ADO.NET. Интерфейс ADO.NET реализует модель доступа к данным [1], которая именуется отсоединенной и предполагает исполнение следующей последовательности действий в каждом сеансе обращения: открытие сетевого подключения к базе данных; получение требуемого набора данных; закрытие сетевого подключения. Полученный по завершении сеанса обращения к базе данных результат называется отсоединенным набором данных, и дальнейшие манипуляции клиента осуществляются именно над ним. Тем самым достигается максимально высокая скорость обработки данных клиентом, а также более эффективное использование сетевых каналов. После завершения работы с данными клиент через операцию обновления возвращает накопленные изменения в базу данных.

Однако отсоединенная модель доступа характеризуется значительной вероятностью рассогласованности отсоединенных наборов данных в том случае, когда проводится одновременное обращения к базе данных нескольких экземпляров клиентов (под интервалом времени здесь понимается время существования клиента от старта до завершения). В такой ситуации после получения каждым клиентом первичного отсоединенного набора данных модификация последнего никак не может быть распространена на другие экземпляры. Неизбежным следствием этого факта является появление дубликатов данных в таблицах, ведущих идентификацию записей по первичному суррогатному ключу.

В работе [2] предложены модели нескольких типов протоколов взаимодействия клиента с базой данных посредством отсоединенных наборов (NoClone), следование которым позволяет избежать появления подобных записей-дубликатов. Основная концепция NoClone-протокола заключается в следующем: перед выполнением запроса клиента, предполагающего обновления данных в базе, необходимо предварительно

проверить, не были ли уже сделаны подобные изменения. Набор же проверочных операций определяет тип протокола и зависит от цели обращения клиента (получить, добавить или обновить данные), а также от числа записей в затрагиваемых таблицах. Очевидно, что для выполнения проверки необходимо, чтобы клиент при обращении предоставил все необходимые сведения.

Целью работы, результаты которой представлены далее, являлась физическая реализация NoClone-протокола как комплекта структурированных пакетов, передаваемых между участниками описанного выше информационного взаимодействия, а также механизмов манипуляции ими.

Структура пакетов NoClone-протокола

Для организации информационного обмена в соответствии с концепцией NoClone решено использовать формат классического диалога. Посредниками такого взаимодействия выступают пакеты двух категорий — запрос инициатора и ответ исполнителя. Принято также решение использовать общую структуру информационного пакета протокольного взаимодействия, в соответствии с которой и запрос, и ответ образуют три базовых компонента: статусную строку; заголовок (служебная часть); тело (содержательная часть). Наборы полей каждого из базовых компонентов для запроса и ответа должны при этом максимально удовлетворять входным параметрам конкретного получателя для успешного исполнения возложенных на него функций.

В работе [2] исследование эффективности NoClone-протоколов проводилось на примере обращения к базе данных MS SQL Server. Для определенности решено следовать данному контексту, а именно — получателем пакета-запроса выступает MS SQL Server, точнее — обслуживаемая им хранимая процедура. Через вызов хранимой процедуры от имени клиентского приложения выполняются необходимые манипуляции с данными. Получателем пакета-ответа является клиентское приложение, для которого имеет значение только отсоединенный набор данных, получаемый после запроса к серверу через поставщика ADO.NET.

В итоге получается структура пакетов протокола, необходимая и достаточная для информационного обмена по концепции NoClone (рис. 1).

Статусная строка сообщения-запроса служит указателем способа воздействия на целевой массив данных. Это значит, что в соответствии с NoClone-моделью взаимодействия поле **`Code`** определяет тип обращения (в зависимости от цели клиента [2]), а следовательно, и набор предваряющих запрос оценочных операций. Статусная строка пакета-ответа сигнализирует о результате обработки запроса. Таким образом, в ответе поле **`Code`** подразумевает одно из двух значений — успех (1) или отказ (0).

Служебная часть пакета служит, во-первых, для его однозначной идентификации в рамках установленного сеанса подключения (диалога) — поле **`Id`**. Самым эффективным решением задачи идентификации будет порядковая нумерация генерируемых клиентом запросов. Во-вторых, сведения служебной части пакета формируют структуру его содержательной части, а именно: в поле **`BodyItems`** перечисляются поля тела пакета-запроса; в поле **`BodyType`** определяется тип данных в теле ответа. Признак типа тела ответного пакета введен в связи с потенциальной неоднозначностью возможного результата обработки запроса. Это может быть и скалярное значение, и упорядоченный набор данных, оформленный в виде таблицы. Следует отметить, что в случае отказа обработки поле **`BodyType`** и само тело в ответном пакете отсутствуют.

Относительно содержательной части пакетов можно отметить следующее. Очевидно, что поля тела пакета-запроса используются хранимой процедурой для формулирования непосредственного запроса к базе данных. Основными инструкциями SQL, подлежащими к исполнению посредством NoClone-протокола, являются SELECT, INSERT и UPDATE. В соответствии с базовым синтаксисом означенных инструкций, для однозначного их написания необ-

ЗАПРОС		ОТВЕТ	
Имя поля	Тип данных	Имя поля	Тип данных
Статус		Статус	
Code	String	Code	Bit
Заголовок		Заголовок	
Id	Integer	Id	Integer
BodyItems	String	BodyType	String
Тело		Тело	
TableName	String	Body	String Table
Fields	String		
Mask	String		
RowVersion	Binary[8]		

Рис. 1. Составы полей пакетов протокола NoClone

ходимы, как минимум, следующие параметры: имя таблицы **TableName**, набор интересующих полей **Fields** и конкретизирующие условия **Mask**. Кроме того, тело запроса может включать поле, содержащее информацию о состоянии базы данных на момент одного из предыдущих обращений клиента, относительно которого предполагается оценить изменения. Это поле **RowVersion** типа битового массива.

Для пакета-ответа в тело, как уже отмечено ранее, упаковываются результаты успешного выполнения запроса — либо единичное значение типа **Single**, либо табличное представление набора данных типа **Table**.

Базовые технологии

Современные гетерогенные информационные системы практически без исключения поддерживают механизм интеграции на уровне стандартизации форматов данных. Стандартными с этой точки зрения являются форматы описания бизнес-данных, служащие основой интеграции корпоративных приложений. В частности, к ним относится язык XML, который предоставляет открытый универсальный механизм описания произвольных структур данных без необходимости их строгой формализации. MS SQL Server также обеспечивает возможность работы с XML-данными на различных уровнях, вплоть до создания и сопровождения иерархических (не реляционных) баз данных.

Поскольку по определению не существует никаких строгих правил представления данных в XML-формате, крайне желательно присутствие некоторого механизма, который позволил бы оценить (по меньшей мере на одной из сторон взаимодействия), соответствует ли представленная структура той, что ожидалась, и, как следствие, может ли информация быть воспринята и обработана. Реализуется такая возможность через процедуру валидации (проверки соответствия) XML-кода установленной схеме, описанной на языке определения схем XSD. MS SQL Server тоже имеет механизм валидации XML-данных, но носит он иное наименование — типизация XML. В данном случае сгенерированный и присоединенный к базе данных набор XSD-схем рассматривается набором особых типов данных. Такие данные можно либо присваивать столбцам таблиц, либо представлять объектами иерархических баз данных. При поступлении же запроса, содержащего XML-код, проводится его проверка на соответствие присоединенной XSD-схеме как типа данных, а затем проводится его передача обработчику.

В состав MS SQL Srever входит очень мощный инструментарий — компонент Service Broker. В работе [2] он был применен в качестве средства управления очередями клиентских запросов и для оценки состояния этих очередей. Главное же его предназначение заключается в организации информационного взаимодействия независимых программных объектов. Компонент Service Broker также ориентирован на работу с типизированными XML-данными, однако в особом контексте. Логика его работы изначально подразумевает, что передаваемые через сервисы компонента сообщения будут закодированы в формате XML, поскольку реализация взаимодействия разнородных сущностей и является одной из его ключевых задач. По этой причине при формировании участников реализации конкретного сервиса [3], а именно — при определении типов передаваемых сообщений, для каждого из них возможно назначение XSD-схемы, на соответствие которой должна проводиться проверка его (сообщения) структуры. На дальнейшую обработку (помещение в очередь) при этом допускаются только валидные сообщения.

Вместе с тем, классы ADO.NET также тесно интегрируются с классами XML.NET [4]. Более того, ключевой объект реализации отсоединенного обращения к данным DataSet, наряду с возможностью сохранения своего содержимого (или получения его) в XML-коде, физически хранит свою структуру в виде XSD-схемы. Таким образом, можно однозначно заключить, что реализация NoClone-протокола взаимодействия через набор подключаемых к прикладному проекту XSD-схем и представляющих их экземпляров XML-объектов обоснованна и достаточно эффективна.

Разработчики программных систем, в особенности бизнес-приложений, должны использовать как минимум два языка программирования: язык высокого уровня (типа C#) и язык взаимодействия с базой данных (типа TransactSQL). Кроме необходимой глубины познаний в обоих языках, всегда имеет место несоответствие — отсутствие возможности интерпретации команд одного языка интерпретатором другого. Как следствие — нет никакой гарантии отсутствия синтаксических ошибок. Здесь снова следует обратить внимание на ADO.NET, в состав которого входит и технология LINQ (интегрированный мультиязыковой запрос). Технология LINQ позволяет формировать в программном коде высокоуровневого языка запросы (свободно понимаемые его компилятором) на обращение к различным перечисляемым источникам данных, а именно к объектам в памяти, к наборам DataSet, к базе SQL, к XML-документам.

В представляемой разработке сделан упор на использование LINQ to XML — встраиваемого интерфейса, позволяющего работать с XML-данными из высокоуровневых языков. При этом XML-данные размещаются в памяти и представляются в виде дерева XML, работа с которым в дальнейшем осуществляется через привычные прикладному программисту объекты и методы. Первичное дерево XML может как формироваться программистом вручную, так и загружаться из внешнего ресурса (файла или входного строкового параметра).

Реализация NoClone-протокола

В соответствии с изложенными выше предпосылками, процесс диалога по NoClone-протоколу в реализации на комбинации языков C#, библиотек ADO.NET и LINQ.NET, а также Transact SQL заключается в следующем (рис. 2).

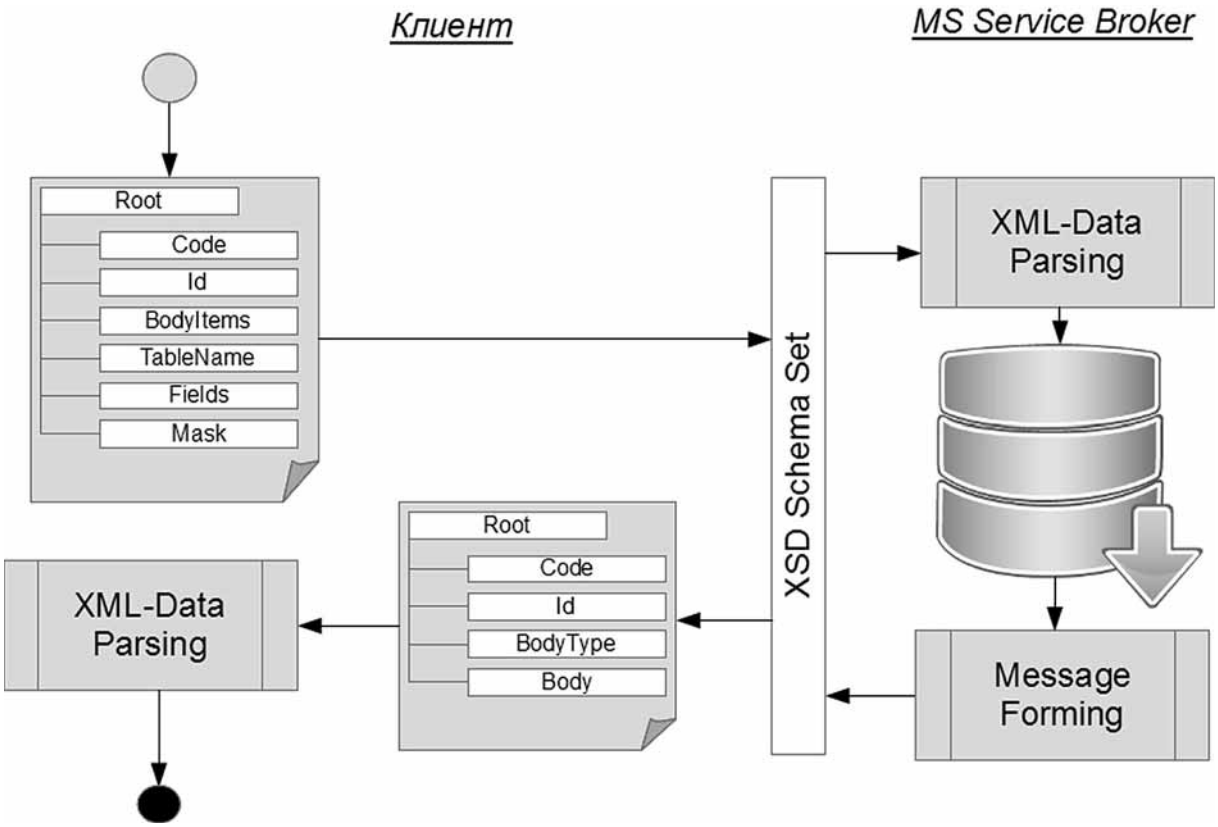


Рис. 2. Обращение к базе данных по протоколу NoClone

До начала обмена пакетами в памяти клиентского приложения должны присутствовать предварительно сформированные первичные XML-деревья, соответствующие описанным на рис. 1 структурам запроса и ответа: **RequsestTree** и **AnswerTree**. Предлагается выполнять данную операцию через процедуру парсинга внешнего ресурса, содержащего шаблонную структуру пустого дерева, а именно — вызовом следующего метода:

```
XDocument.Parse(/*Путь.к.шаблону*/)
```

Непосредственно сеанс обращения к базе данных начинается с наполнения дерева **RequestTree**, а именно — с присвоения необходимого значения каждому его элементу. Затем через вызов инструкций Transact SQL выполняется инициализация диалога и далее осуществляется передача сообщения компоненту MS Service Broker:

```
SEND ON CONVERSATION @InitDlgHandle
MESSAGE TYPE @Type (@Body)
```

При компиляции кода C# проводится замена параметров данной строки действительными представлениями объектов: на место **@Type** дублируется информация элемента **Code** дерева **RequestTree**, а полное представление дерева в формате строки замещает параметр **@Body**. Таким образом, сообщение компонента MS Service Broker выступает транспортной сущностью, инкапсулирующей в себе пакет запроса.

На стороне MS SQL Server каждое поступающее сообщение подвергается проверке на соответствие тела назначенной схеме XSD, и при успешном результате оно помещается в очередь входящих сообщений. Обработка каждого сообщения из очереди входящих имеет целью формулировку последовательности инструкций и запросов к базе данных в зависимости от типа обращения, а также их запуск на исполнение. Для этого проводятся семантический анализ тела сообщения, выделение из XML-кода необходимых сведений и компоновка из них текста запроса, например, посредством кода Transact SQL, представленного на рис. 3.

В последней строке кода на рис. 3 осуществляется передача сформированной строки запроса обработчику запросов SQL Server. Результат его исполнения возвращается в формате кода XML переменной **@r** соответствующего типа, на основе которой генерируется структура ответа. Для этого требуется модификация переменной **@r** путем добавления в ее код полей статуса (**Code**) и заголовка (**Id**, **BodyType**), например, через вызов следующих инструкций: **@r. modify('insert ...')** и **@r. modify('replace value ...')**.

Ответное сообщение отправляется обратно клиенту через отмеченную выше инструкцию **SEND**

```
--ранее в переменную @x помещен XML-код запроса
declare @query varchar(max);
declare @n int;

set @query = 'Select ';
set @items = @x.value('(/Root/BodyItems)[1]','char(40)');

if (PATINDEX('%Fields%',@items) > 0) SET @query = @query +
    @x.value('(/Root/Fields)[1]','char(20)') + ' '
 ELSE SET @query = @query + ' * ';

if (PATINDEX('%TableName%',@items) > 0) SET @query = @query + 'FROM
   ' + @x.value('(/Root/TableName)[1]','char(20)') + ' ';

if (PATINDEX('%Mask%',@items) > 0) SET @query = @query + 'WHERE ' +
    @x.value('(/Root/Mask)[1]','char(20)') + ' ';

declare xml @r;
 @r = SELECT (execute (@query)) FOR XML AUTO, TYPE;
```

Рис. 3. Фрагмент кода Transact SQL

в рамках текущего диалога. Очевидно, что ответное сообщение компонента MS Service Broker тоже является транспортной сущностью: в параметре **TYPE** дублируется значение поля **Code** ответного пакета, а его полное представление (прописанное кодом XML) упаковывается в тело сообщения. Так же как и получение, отправка ответного сообщения предваряется проверкой на соответствие структуры тела схеме XSD.

После получения клиентским приложением ответного сообщения через инструкцию **RECEIVE** Transact SQL, последнему остается проверить значения полей **Code** (действительно ли получены данные) и **Id** (порядок обмена пакетами не нарушен), модифицировать первичное дерево **AnswerTree** и загрузить полученные данные в **DataSet**. Последнее действие осуществляется вызовом следующего метода:

dataSet.ReadXml(AnswerTree, XmlReadMode.IgnoreSchema).

С этого момента объект класса DataSet готов к предоставлению необходимых данных остальным компонентам клиентского приложения.

Заключение

Проведенные ранее исследования производительности клиентских приложений при обращении их к базам данных в условиях высокой взаимной конкуренции показали, что минимизация вероятности появления при этом дубликатов возможна при использовании одновременно нескольких моделей (NoClone-моделей) формулирования и обработки поступающих запросов. В процессе эксперимента был получен ряд интересных эффектов поведения очередей запросов при варьировании их мощностей и удельного соотношения разнородных элементов. В связи с этим обстоятельством появилась необходимость дальнейших исследований в данной области, для чего необходимо совершенствование экспериментального стенда.

Реализованный и представленный в настоящей статье NoClone-протокол взаимодействия клиентского приложения с базой данных по отсоединенной модели доступа является одним из основных шагов к построению программной среды для разностороннего исследования NoClone-моделей обращения. Данный протокол предоставляет возможность максимально гибко формулировать конкретные запросы в соответствии с необходимой моделью. В то же время, он позволяет получать стандартные, ожидаемые результирующие наборы. Следующим этапом к построению среды имитационного моделирования процессов взаимодействия программных объектов с базами данных логично предположить создание модуля разностороннего анализа и статистической обработки получаемых в ходе эксперимента результатов.

Список литературы

1. **Макарцев Е. В.** Сравнительный анализ быстродействия интерфейсов доступа к данным ADO, dbExpress и ADO.NET на примере Microsoft SQL Server [Электронный ресурс] // Электронный научный журнал APRIORI. Серия: Естественные и технические науки. 2013. № 1. URL: http://apriori-journal.ru/journal-estesvennie-nauki/id/87

2. **Родионов А. Н., Решетникова О. В.** Noclone модель и протоколы взаимодействия с семантическими объектами баз данных в многопользовательских приложениях // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2015. Т. 13, № 2. С. 64—75.

3. **Aschenbrenner K.** Pro SQL Server 2008 Service Broker. NewYork: Apress, 2008. 587 p.

4. **Фокс Д.** ADO и XML [Электронный ресурс] // SQL Magazine OnLine. 2001. № 2. URL: http://www.osp.ru/data/www2/win2000/sql/2001/02/650.htm

Implementation of NoClone Customer Access Protocol to MS SQL Server Database

O. V. Reshetnikova, ov13r@yandex.ru, Far Eastern State Transport University, Khabarovsk, 680021, Russian Federation

Corresponding author:

Reshetnikova Olga V., Associate Professor, Far Eastern State Transport University, Khabarovsk, 680021, Russian Federation

e-mail: ov13r@yandex.ru

Received on December 28, 2015
Accepted on January 13, 2016

Earlier studies on the organization of NoClone models of database access have identified the need for more detailed study in this field, requiring improvement of approaches to application implementation, including test bench. A variant of implementation of the protocol to organize client access to a database for specified NoClone models is proposed, it allows to minimize the possibility of duplicated records. This protocol allows to flexibly formulate concrete requests in accordance with desired address model and at the same time to get standard expected result sets. The interaction is carried out in the format of one-sided dialogue initiated by exchange between data packages of a fixed structure. Translation process of data packages includes a compulsory filtration stage against prescribed structures. A package transmitted via protocol is represented as XML-object and transport mechanism of Service Broker MS SQL Server component is used for its physical transmission. NoClone protocol implementation is one of the key steps to build simulation environment of software interaction objects with databases.

Keywords: *disconnected data access, duplicate records, protocol for data access, XML, send/receive message, Transact SQL, parsing strings*

For citation:

Reshetnikova O. V. Implementation of NoClone Customer Access Protocol to MS SQL Server Database, *Programmnaya Ingeneria*, 2016, vol. 7, no. 4, pp. 167-172.

DOI: 10.17587/prin.7.167-172

References

1. **Makartsev E. V.** Sravnitel'nyi analiz bystrodeistviya interfeisov dostupa k dannym ADO, dbExpress i ADO.NET na primere Microsoft SQL Server (A comparative analysis of speed of ADO, dbExpress and ADO.NET data access interfaces on the example of Microsoft SQL Server) [Electronic resource]. *Elektronnyi nauchnyi zhurnal APRIORI. Seriya: Estestvennye i tekhnicheskie nauki,* 2013, no. 1, available at: http://apriori-journal.ru/journal-estesvennie-nauki/id/87 (in Russian).

2. **Rodionov A. N. Reshetnikova O. V.** Noclone model' i protokoly vzaimodeistviya s semanticheskimi ob"ektami baz dannykh v mnogopol'zovatel'skikh prilozheniyakh (Noclone model and protocols of interaction with database semantic objects in multiuser applications), *Vestnik Novosibirskogo gosudarstvennogo universiteta. Seriya: Informatsionnye tekhnologii*, 2015, vol. 13, no. 2, pp. 64—75 (in Russian).

3. **Aschenbrenner K.** *Pro SQL Server 2008 Service Broker*, NewYork, Apress, 2008. 587 p.

4. **Fox D.** ADO i XML (ADO and XML) [Electronic resource], *SQL Magazine OnLine*, 2001, no. 2, available at: http://www.osp.ru/data/www2/win2000/sql/2001/02/650.htm (in Russian).

УДК 629.73.02; 629.73.05/.06; 535.643 DOI: 10.17587/prin.7.173-180

И. О. Жаринов, д-р техн. наук, доц., зав. каф., e-mail: igor_rabota@pisem.net,
Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (Университет ИТМО), АО "ОКБ "Электроавтоматика",
О. О. Жаринов, канд. техн. наук, доц., e-mail: zharinov73@hotbox.ru,
Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Математическое обеспечение для решения практической задачи калибровки мониторов на жидких кристаллах и светодиодах

Экран современных средств отображения информации выполнен на базе жидких кристаллов или светодиодов. Свойство цветовоспроизведения экрана описывается математической моделью. Коэффициенты модели цветовоспроизведения экрана априори неизвестны. Оценка коэффициентов модели может быть выполнена на основе паспортных данных экрана или в результате обработки данных, полученных при колориметрических измерениях. Вывод математических выражений для оценки коэффициентов модели цветовоспроизведения экрана основан на законах смешения цветов Грассмана. Исходными данными для вывода являются: координаты цветности вершин треугольника цветового охвата и точки белого цвета экрана, представленные в цветовом пространстве UV; координаты цветности четырех произвольно заданных цветов, отображаемых на экране, полученные в цветовом пространстве UV в результате колориметрических измерений. Предложены новые: математические выражения для оценки коэффициентов модели цветовоспроизведения экрана двумя способами; выражения для оценки смещения (u',v')-координат цветности изображения на различных образцах экранов; математические выражения для оценки погрешности косвенного измерения компонентов кодов основных цветов (красный, зеленый, синий).

Ключевые слова: *модель цветовоспроизведения, оценка коэффициентов, координаты цветности, цветовое пространство*

Введение

В процессе решения практических задач по улучшению свойств цветовоспроизведения (калибровки) средств отображения информации (мониторов) и при выполнении колориметрических расчетов исследователи сталкиваются [1—6] с необходимостью использования математического обеспечения для описания свойств цветовоспроизведения жидкокристаллического или светодиодного экрана.

Модель цветовоспроизведения экрана, основанного на трехкомпонентной RGB (R — Red, G — Green, B — Blue) схеме формирования цвета, определяется коэффициентами X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b, объединенными в матрицу профиля экрана. Матрица профиля экрана связывает код RGB выводимого на экран цвета и координаты цвета XYZ индицируемого изображения следующим образом:

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

В работах [7, 8] показано, что неизвестные значения коэффициентов X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b профиля экрана могут быть определены тремя различными способами, основанными на использовании:

- (x, y)-координат цветности вершин треугольника цветового охвата экрана и точки белого цвета экрана, заданных в технической документации на экран в цветовом пространстве XY;
- результатов колориметрических измерений (x, y)-координат цветности для четырех произвольно заданных кодами RGB цветов;
- результатов колориметрических измерений координат цвета XYZ для четырех произвольно заданных кодами RGB цветов.

Вместе с этим измерители координат цвета и координат цветности (колориметры, спектрорадиометры), применяемые при калибровке мониторов, осуществляют измерения в числе прочего и в цветовом простран-

стве *UV*, введенном Международной комиссией по освещению в 1976 г. Координаты цветности (u', v') связаны с координатами цвета *XYZ* выражениями

$$u' = \frac{4X}{X + 15Y + 3Z},\ v' = \frac{9Y}{X + 15Y + 3Z},\ X = \frac{9Yu'}{4v'},\ Z = \frac{3Y(4 - u')}{4v'} - 5Y, \tag{2}$$

и с (x, y)-координатами цветности выражениями

$$u' = \frac{2x}{6y - x + 1{,}5},\ v' = \frac{4{,}5y}{6y - x + 1{,}5},\ x = \frac{4{,}5u'}{3u' - 8v' + 6},\ y = \frac{2v'}{3u' - 8v' + 6}. \tag{3}$$

В связи с этим актуальными являются задачи определения математических выражений, необходимых для оценки:

- коэффициентов X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b профиля экрана на основе результатов колориметрических измерений (u', v')-координат цветности или паспортных данных экрана (координаты вершин треугольника цветового охвата, точки белого цвета), заданных в цветовом пространстве *UV*;
- смещения (u', v')-координат цветности изображения на различных экранах одного производителя с близкими по значениям коэффициентами X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b профиля;
- погрешности косвенного измерения компонентов кода *RGB* индицируемого изображения на основе прямых колориметрических измерений координат цветности в цветовом пространстве *UV*.

Решение задачи оценки коэффициентов профиля экрана на основе паспортных данных

В случае если в технической документации на экран координаты цветности (u'_R, v'_R), (u'_G, v'_G), (u'_B, v'_B) вершин треугольника цветового охвата и точки белого цвета (u'_W, v'_W) заданы в цветовом пространстве *UV*, оценка коэффициентов $\widehat{X}_r$, $\widehat{Y}_r$, $\widehat{Z}_r$, $\widehat{X}_g$, $\widehat{Y}_g$, $\widehat{Z}_g$, $\widehat{X}_b$, $\widehat{Y}_b$, $\widehat{Z}_b$ профиля экрана осуществляется в соответствии с выражениями системы

$$\begin{cases}
\widehat{X}_r = \dfrac{9u'_R}{4v'_R}\widehat{Y}_r \\
\widehat{Y}_r = 1 - \widehat{Y}_g - \widehat{Y}_b \\
\widehat{Z}_r = \dfrac{3\widehat{Y}_r(4 - u'_R)}{4v'_R} - 5\widehat{Y}_r \\
\widehat{X}_g = \dfrac{9u'_G}{4v'_G}\widehat{Y}_G \\
\widehat{Y}_g = \dfrac{\det\begin{pmatrix} u'_W & u'_R \\ v'_W & v'_R \end{pmatrix} - \widehat{Y}_b \det\begin{pmatrix} u'_B & u'_R \\ v'_B & v'_R \end{pmatrix}}{\det\begin{pmatrix} u'_G & u'_R \\ v'_G & v'_R \end{pmatrix}} \\
\widehat{Z}_g = \dfrac{3\widehat{Y}_g(4 - u'_G)}{4v'_G} - 5\widehat{Y}_g \\
\widehat{X}_b = \dfrac{9u'_B}{4v'_B}\widehat{Y}_b \\
\widehat{Y}_b = \dfrac{v'_B \det\begin{pmatrix} \det\begin{pmatrix} u'_G & u'_R \\ v'_G & v'_R \end{pmatrix} & \det\begin{pmatrix} 4 - u'_G & v'_G \\ 4 - u'_R & v'_R \end{pmatrix} \\ \det\begin{pmatrix} u'_W & u'_R \\ v'_W & v'_R \end{pmatrix} & \det\begin{pmatrix} 4 - u'_W & v'_W \\ 4 - u'_R & v'_R \end{pmatrix} \end{pmatrix}}{v'_W \det\begin{pmatrix} \det\begin{pmatrix} u'_G & u'_R \\ v'_G & v'_R \end{pmatrix} & \det\begin{pmatrix} 4 - u'_G & v'_G \\ 4 - u'_R & v'_R \end{pmatrix} \\ \det\begin{pmatrix} u'_B & u'_R \\ v'_B & v'_R \end{pmatrix} & \det\begin{pmatrix} 4 - u'_B & v'_B \\ 4 - u'_R & v'_R \end{pmatrix} \end{pmatrix}} \\
\widehat{Z}_b = \dfrac{3\widehat{Y}_b(4 - u'_B)}{4v'_B} - 5\widehat{Y}_b.
\end{cases} \tag{4}$$

Расчет оценок коэффициентов системы (4) начинается с вычисления оценки $\widehat{Y}_b$. Баланс белого цвета в системе (4) учитывается уравнением $\widehat{Y}_r + \widehat{Y}_g + \widehat{Y}_b = 1$, в соответствии с которым относительная яркость изображения в белом цвете при коде RGB = (255, 255, 255) принимается за 100 % [9].

Система (4) получена путем решения системы уравнений

$$\begin{cases} u'_R\left(\widehat{X}_r + \widehat{Y}_r + \widehat{Z}_r\right) = 4\widehat{X}_r = 4\dfrac{9\widehat{Y}_r u'_R}{v'_R} \\ v'_R\left(\widehat{X}_r + \widehat{Y}_r + \widehat{Z}_r\right) = 9\widehat{Y}_r = \dfrac{\widehat{X}_r v'_R}{u'_R} \\ u'_G\left(\widehat{X}_g + \widehat{Y}_g + \widehat{Z}_g\right) = 4\widehat{X}_g = 4\dfrac{9\widehat{Y}_g u'_G}{v'_G} \\ v'_G\left(\widehat{X}_g + \widehat{Y}_g + \widehat{Z}_g\right) = 9\widehat{Y}_g = \dfrac{\widehat{X}_g v'_G}{u'_G} \\ u'_B\left(\widehat{X}_b + \widehat{Y}_b + \widehat{Z}_b\right) = 4\widehat{X}_b = 4\dfrac{9\widehat{Y}_b u'_B}{v'_B} \\ v'_B\left(\widehat{X}_b + \widehat{Y}_b + \widehat{Z}_b\right) = 9\widehat{Y}_b = \dfrac{\widehat{X}_b v'_B}{u'_B} \\ u'_W\left(\widehat{X}_r + \widehat{X}_g + \widehat{X}_b + 15\left(\widehat{Y}_r + \widehat{Y}_g + \widehat{Y}_b\right) + 3\left(\widehat{Z}_r + \widehat{Z}_g + \widehat{Z}_b\right)\right) = 4\left(\widehat{X}_r + \widehat{X}_g + \widehat{X}_b\right) \\ v'_W\left(\widehat{X}_r + \widehat{X}_g + \widehat{X}_b + 15\left(\widehat{Y}_r + \widehat{Y}_g + \widehat{Y}_b\right) + 3\left(\widehat{Z}_r + \widehat{Z}_g + \widehat{Z}_b\right)\right) = 9\left(\widehat{Y}_r + \widehat{Y}_g + \widehat{Y}_b\right) \\ \widehat{Y}_r + \widehat{Y}_g + \widehat{Y}_b = 1 \end{cases}, \quad (5)$$

основанной на выражениях (1)—(3).

Существенное упрощение для вывода (4) достигается при выделении из системы (5) подсистемы уравнений

$$\begin{cases} \dfrac{9\widehat{Y}_r u'_R}{4v'_R} + \dfrac{9\widehat{Y}_g u'_G}{4v'_G} + \dfrac{9\widehat{Y}_b u'_B}{4v'_B} = \dfrac{9u'_W}{4v'_W} \\ \widehat{Y}_r + \widehat{Y}_g + \widehat{Y}_b = 1 \\ \dfrac{3\widehat{Y}_r\left(4 - u'_R\right)}{4v'_R} - 5\widehat{Y}_r + \dfrac{3\widehat{Y}_g\left(4 - u'_G\right)}{4v'_G} - 5\widehat{Y}_g + \dfrac{3\widehat{Y}_b\left(4 - u'_B\right)}{4v'_B} - 5\widehat{Y}_b = \dfrac{3\left(4 - u'_W\right)}{4v'_W} - 5 \end{cases}$$

и ее решении относительно оценок коэффициентов $\widehat{Y}_r$, $\widehat{Y}_g$, $\widehat{Y}_b$.

Решение задачи оценки коэффициентов профиля экрана на основе результатов колориметрических измерений

Для колориметров, осуществляющих измерения в пространстве координат цветности UV, выражения для оценки коэффициентов профиля экрана на основе результатов четырех последовательно выполненных измерений (u'_1, v'_1), (u'_2, v'_2), (u'_3, v'_3), (u'_4, v'_4) для четырех цветов, произвольно заданных кодами $R_1G_1B_1$, $R_2G_2B_2$, $R_3G_3B_3$, $R_4G_4B_4$, имеют вид

$$\begin{bmatrix} \widehat{X}_r \\ \widehat{X}_g \\ \widehat{X}_b \\ \widehat{Y}_g \\ \widehat{Y}_b \\ \widehat{Z}_r \\ \widehat{Z}_g \\ \widehat{Z}_b \end{bmatrix} = \begin{bmatrix} d'_{11} & d'_{12} & d'_{13} & d'_{14} & d'_{15} & d'_{16} & d'_{17} & d'_{18} \\ d'_{21} & d'_{22} & d'_{23} & d'_{24} & d'_{25} & d'_{26} & d'_{27} & d'_{28} \\ d'_{31} & d'_{32} & d'_{33} & d'_{34} & d'_{35} & d'_{36} & d'_{37} & d'_{38} \\ d'_{41} & d'_{42} & d'_{43} & d'_{44} & d'_{45} & d'_{46} & d'_{47} & d'_{48} \\ d'_{51} & d'_{52} & d'_{53} & d'_{54} & d'_{55} & d'_{56} & d'_{57} & d'_{58} \\ d'_{61} & d'_{62} & d'_{63} & d'_{64} & d'_{65} & d'_{66} & d'_{67} & d'_{68} \\ d'_{71} & d'_{72} & d'_{73} & d'_{74} & d'_{75} & d'_{76} & d'_{77} & d'_{78} \\ d'_{81} & d'_{82} & d'_{83} & d'_{84} & d'_{85} & d'_{86} & d'_{87} & d'_{88} \end{bmatrix}^{-1} \begin{bmatrix} 15u'_1R_1 \\ R_1\left(9 - 15v'_1\right) \\ 15u'_2R_2 \\ R_2\left(9 - 15v'_2\right) \\ 15u'_3R_3 \\ R_3\left(9 - 15v'_3\right) \\ 15u'_4R_4 \\ R_4\left(9 - 15v'_4\right) \end{bmatrix}; \quad \widehat{Y}_r = 1 - \widehat{Y}_g - \widehat{Y}_b, \quad (6)$$

где

$$d'_{11} = R_1(4-u'_1),\ d'_{12} = G_1(4-u'_1),\ d'_{13} = B_1(4-u'_1),\ d'_{14} = 15u'_1(R_1-G_1),$$
$$d'_{15} = 15u'_1(R_1-B_1),\ d'_{16} = -3u'_1R_1,\ d'_{17} = -3u'_1G_1,\ d'_{18} = -3u'_1B_1,$$
$$d'_{21} = R_1v'_1,\ d'_{22} = G_1v'_1,\ d'_{23} = B_1v'_1,\ d'_{24} = (R_1-G_1)(9-15v'_1),$$
$$d'_{25} = (R_1-B_1)(9-15v'_1),\ d'_{26} = 3R_1v'_1,\ d'_{27} = 3G_1v'_1,\ d'_{28} = 3B_1v'_1,$$
$$d'_{31} = R_2(4-u'_2),\ d'_{32} = G_2(4-u'_2),\ d'_{33} = B_2(4-u'_2),\ d'_{34} = 15u'_2(R_2-G_2),$$
$$d'_{35} = 15u'_2(R_2-B_2),\ d'_{36} = -3u'_2R_2,\ d'_{37} = -3u'_2G_2,\ d'_{38} = -3u'_2B_2,$$
$$d'_{41} = R_2v'_2,\ d'_{42} = G_2v'_2,\ d'_{43} = B_2v'_2,\ d'_{44} = (R_2-G_2)(9-15v'_2),$$
$$d'_{45} = (R_2-B_2)(9-15v'_2),\ d'_{46} = 3R_2v'_2,\ d'_{47} = 3G_2v'_2,\ d'_{48} = 3B_2v'_2,$$
$$d'_{51} = R_3(4-u'_3),\ d'_{52} = G_3(4-u'_3),\ d'_{53} = B_3(4-u'_3),\ d'_{54} = 15u'_3(R_3-G_3),$$
$$d'_{55} = 15u'_3(R_3-B_3),\ d'_{56} = -3u'_3R_3,\ d'_{57} = -3u'_3G_3,\ d'_{58} = -3u'_3B_3,$$
$$d'_{61} = R_3v'_3,\ d'_{62} = G_3v'_3,\ d'_{63} = B_3v'_3,\ d'_{64} = (R_3-G_3)(9-15v'_3),$$
$$d'_{65} = (R_3-B_3)(9-15v'_3),\ d'_{66} = 3R_3v'_3,\ d'_{67} = 3G_3v'_3,\ d'_{68} = 3B_3v'_3,$$
$$d'_{71} = R_4(4-u'_4),\ d'_{72} = G_4(4-u'_4),\ d'_{73} = B_4(4-u'_4),\ d'_{74} = 15u'_4(R_4-G_4),$$
$$d'_{75} = 15u'_4(R_4-B_4),\ d'_{76} = -3u'_4R_4,\ d'_{77} = -3u'_4G_4,\ d'_{78} = -3u'_4B_4,$$
$$d'_{81} = R_4v'_4,\ d'_{82} = G_4v'_4,\ d'_{83} = B_4v'_4,\ d'_{84} = (R_4-G_4)(9-15v'_4),$$
$$d'_{85} = (R_4-B_4)(9-15v'_4),\ d'_{86} = 3R_4v'_4,\ d'_{87} = 3G_4v'_4,\ d'_{88} = 3B_4v'_4.$$

Система (6) получена путем решения системы уравнений

$$\begin{cases}
4\left(\widehat{X}_rR_1+\widehat{X}_gG_1+\widehat{X}_bB_1\right) = u'_1\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_1+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_1+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_1\right)\\
9\left(\widehat{Y}_rR_1+\widehat{Y}_gG_1+\widehat{Y}_bB_1\right) = v'_1\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_1+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_1+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_1\right)\\
4\left(\widehat{X}_rR_2+\widehat{X}_gG_2+\widehat{X}_bB_2\right) = u'_2\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_2+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_2+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_2\right)\\
9\left(\widehat{Y}_rR_2+\widehat{Y}_gG_2+\widehat{Y}_bB_2\right) = v'_2\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_2+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_2+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_2\right)\\
4\left(\widehat{X}_rR_3+\widehat{X}_gG_3+\widehat{X}_bB_3\right) = u'_3\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_3+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_3+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_3\right),\\
9\left(\widehat{Y}_rR_3+\widehat{Y}_gG_3+\widehat{Y}_bB_3\right) = v'_3\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_3+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_3+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_3\right)\\
4\left(\widehat{X}_rR_4+\widehat{X}_gG_4+\widehat{X}_bB_4\right) = u'_4\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_4+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_4+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_4\right)\\
9\left(\widehat{Y}_rR_4+\widehat{Y}_gG_4+\widehat{Y}_bB_4\right) = v'_4\left(\left(\widehat{X}_r+15\widehat{Y}_r+3\widehat{Z}_r\right)R_4+\left(\widehat{X}_g+15\widehat{Y}_g+3\widehat{Z}_g\right)G_4+\left(\widehat{X}_b+15\widehat{Y}_b+3\widehat{Z}_b\right)B_4\right)\\
\widehat{Y}_r = 1-\widehat{Y}_g-\widehat{Y}_b
\end{cases}$$

основанной на выражениях (1), (2).

Оценка смещения (*u′*, *v′*)-координат цветности изображения с близкими значениями коэффициентов профиля

Смещение координат цветности индицируемого изображения образуется вследствие технологического разброса характеристик цветовоспроизведения экранов [10], присущего серийному производству экранов. Смещение координат цветности визуально проявляется в форме цветовых отличий индицируемого на разных образцах однотипных экранов изображения, программно заданного идентичными кодами *RGB*. Инструментально смещение координат цветности может быть оценено с помощью средств измерительной техники (колориметров, спектрорадиометров).

В этом случае с учетом (1), (2) для любой пары однотипных экранов в цветовом пространстве *UV* справедливо:

$$\begin{bmatrix} X_{r_2} & X_{g_2} & X_{b_2} \\ Y_{r_2} & Y_{g_2} & Y_{b_2} \\ Z_{r_2} & Z_{g_2} & Z_{b_2} \end{bmatrix}^{-1} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} X_{r_1} & X_{g_1} & X_{b_1} \\ Y_{r_1} & Y_{g_1} & Y_{b_1} \\ Z_{r_1} & Z_{g_1} & Z_{b_1} \end{bmatrix}^{-1} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} \Rightarrow \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} X_{r_2} & X_{g_2} & X_{b_2} \\ Y_{r_2} & Y_{g_2} & Y_{b_2} \\ Z_{r_2} & Z_{g_2} & Z_{b_2} \end{bmatrix} \begin{bmatrix} X_{r_1} & X_{g_1} & X_{b_1} \\ Y_{r_1} & Y_{g_1} & Y_{b_1} \\ Z_{r_1} & Z_{g_1} & Z_{b_1} \end{bmatrix}^{-1} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} \breve{X}_r & \breve{X}_g & \breve{X}_b \\ \breve{Y}_r & \breve{Y}_g & \breve{Y}_b \\ \breve{Z}_r & \breve{Z}_g & \breve{Z}_b \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} =$$

$$= \begin{bmatrix} \dfrac{9u'_2 Y_{uv_2}}{4v'_2} \\ Y_{uv_2} \\ \dfrac{3Y_{uv_2}(4-u'_2)}{4v'_2} - 5Y_{uv_2} \end{bmatrix} = \begin{bmatrix} \breve{X}_r & \breve{X}_g & \breve{X}_b \\ \breve{Y}_r & \breve{Y}_g & \breve{Y}_b \\ \breve{Z}_r & \breve{Z}_g & \breve{Z}_b \end{bmatrix} \begin{bmatrix} \dfrac{9u'_1 Y_{uv_1}}{4v'_1} \\ Y_{uv_1} \\ \dfrac{3Y_{uv_1}(4-u'_1)}{4v'_1} - 5Y_{uv_1} \end{bmatrix} \Rightarrow \quad (7)$$

$$\Rightarrow \begin{cases} \dfrac{9u'_2 Y_{uv_2}}{4v'_2} = \breve{X}_r \dfrac{9u'_1 Y_{uv_1}}{4v'_1} + \breve{X}_g Y_{uv_1} + \breve{X}_b \left(\dfrac{3Y_{uv_1}(4-u'_1)}{4v'_1} - 5Y_{uv_1} \right) \\ Y_{uv_2} = \breve{Y}_r \dfrac{9u'_1 Y_{uv_1}}{4v'_1} + \breve{Y}_g Y_{uv_1} + \breve{Y}_b \left(\dfrac{3Y_{uv_1}(4-u'_1)}{4v'_1} - 5Y_{uv_1} \right) \\ \dfrac{3Y_{uv_2}(4-u'_2)}{4v'_2} - 5Y_{uv_2} = \breve{Z}_r \dfrac{9u'_1 Y_{uv_1}}{4v'_1} + \breve{Z}_g Y_{uv_1} + \breve{Z}_b \left(\dfrac{3Y_{uv_1}(4-u'_1)}{4v'_1} - 5Y_{uv_1} \right), \end{cases}$$

где X_1, Y_1, Z_1 — координаты цвета изображения на первом экране; X_2, Y_2, Z_2 — координаты цвета изображения на втором экране; u'_1, v'_1, Y_{uv_1} — координаты цветности и компонент относительной яркости изображения, индицируемого на первом экране; u'_2, v'_2, Y_{uv_2} — координаты цветности и компонент относительной яркости изображения, индицируемого на втором экране; X_{r_i}, X_{g_i}, X_{b_i}, Y_{r_i}, Y_{g_i}, Y_{b_i}, Z_{r_i}, Z_{g_i}, Z_{b_i} (i = 1, 2) — коэффициенты матриц профиля двух экранов; *RGB* — десятичный код цвета индицируемого изображения.

Решение системы уравнений (7) имеет следующий вид:

$$u'_2 = \left(\frac{4}{9} \frac{\breve{X}_r \dfrac{9u'_1}{4v'_1} + \breve{X}_g + \breve{X}_b \left(\dfrac{3(4-u'_1)}{4v'_1} - 5 \right)}{\breve{Y}_r \dfrac{9u'_1}{4v'_1} + \breve{Y}_g + \breve{Y}_b \left(\dfrac{3(4-u'_1)}{4v'_1} - 5 \right)} \right) \left(\frac{1}{9} \frac{\breve{X}_r \dfrac{9u'_1}{4v'_1} + \breve{X}_g + \breve{X}_b \left(\dfrac{3(4-u'_1)}{4v'_1} - 5 \right)}{\breve{Y}_r \dfrac{9u'_1}{4v'_1} + \breve{Y}_g + \breve{Y}_b \left(\dfrac{3(4-u'_1)}{4v'_1} - 5 \right)} + \right.$$

$$\left. + \frac{1}{3} \left(\breve{Z}_r \frac{9u'_1}{4v'_1} + \breve{Z}_g + \breve{Z}_b \left(\frac{3(4-u'_1)}{4v'_1} - 5 \right) \right) + \frac{5}{3} \left(\breve{Y}_r \frac{9u'_1}{4v'_1} + \breve{Y}_g + \breve{Y}_b \left(\frac{3(4-u'_1)}{4v'_1} - 5Y_{uv_1} \right) \right) \right)^{-1},$$

$$v'_2 = \left(\frac{1}{9} \frac{\breve{X}_r \dfrac{9u'_1}{4v'_1} + \breve{X}_g + \breve{X}_b \left(\dfrac{3(4-u'_1)}{4v'_1} - 5 \right)}{\breve{Y}_r \dfrac{9u'_1}{4v'_1} + \breve{Y}_g + \breve{Y}_b \left(\dfrac{3(4-u'_1)}{4v'_1} - 5 \right)} + \right.$$

$$\left. + \frac{1}{3} \left(\breve{Z}_r \frac{9u'_1}{4v'_1} + \breve{Z}_g + \breve{Z}_b \left(\frac{3(4-u'_1)}{4v'_1} - 5 \right) \right) + \frac{5}{3} \left(\breve{Y}_r \frac{9u'_1}{4v'_1} + \breve{Y}_g + \breve{Y}_b \left(\frac{3(4-u'_1)}{4v'_1} - 5Y_{uv_1} \right) \right) \right)^{-1}.$$

В идеальном случае, когда профили двух различных экранов совпадают, а именно $\breve{X}_r = \breve{Y}_g = \breve{Z}_b = 1$, $\breve{X}_g = \breve{X}_b = \breve{Y}_r = \breve{Y}_b = \breve{Z}_r = \breve{Z}_g = 0$ и, следовательно, $u'_2 = u'_1$, $v'_2 = v'_1$, то смещение $\Delta u' = u'_2 - u'_1$, $\Delta v' = v'_2 - v'_1$ координат цветности изображения равно нулю. Для приведения отличающихся профилей матрицы к единому значению в целях компенсации технологического разброса визуальных характеристик экранов необходимо использовать процедуру коррекции [11], учитывающую вносимую погрешность прямого измерения колориметра [12].

Оценка погрешности косвенного измерения кода *RGB* на основе прямого измерения (*u′*, *v′*)-координат цветности

В результате колориметрических измерений экрана исследователь получает данные о значениях (u', v')-координат цветности индицируемого изображения. Средство измерения (колориметр) как измерительный прибор обладает относительной и абсолютной погрешностями прямого измерения.

Инструментальная погрешность прямого измерения (u', v')-координат цветности влияет на точность результатов косвенного измерения компонентов кода *RGB*. Выражения для оценки абсолютных $\Delta R^{u'v'Y_{uv}}$, $\Delta RG^{u'v'Y_{uv}}$, $\Delta B^{u'v'Y_{uv}}$ и относительных $\varepsilon_R^{u'v'Y_{uv}}$, $\varepsilon_G^{u'v'Y_{uv}}$, $\varepsilon_B^{u'v'Y_{uv}}$ инструментальных погрешностей косвенного измерения компонентов кода *RGB* индицируемого на экране изображения имеют следующий вид:

$$\varepsilon_R^{u'v'Y_{uv}} = \frac{1}{\tilde{X}_r \frac{9\bar{Y}_{uv}\bar{u}'}{4\bar{v}''} + \tilde{X}_g\bar{Y}_{uv} + \tilde{X}_b\left(\frac{3\bar{Y}_{uv}(4-\bar{u}')}{4\bar{v}'} - 5\bar{Y}_{uv}\right)}\left(\left(\left(\tilde{X}_r\frac{9\bar{Y}_{uv}}{4\bar{v}'} - \tilde{X}_b\frac{3\bar{Y}_{uv}}{4\bar{v}'}\right)\Delta u'\right)^2 + \right.$$
$$\left. + \left(\left(\tilde{X}_r\frac{9\bar{Y}_{uv}\bar{u}'}{-4\bar{v}'^2} + \tilde{X}_b\frac{3\bar{Y}_{uv}(4-\bar{u}')}{-4\bar{v}'^2}\right)\Delta v'\right)^2 + \left(\left(\tilde{X}_r\frac{9\bar{u}'}{4\bar{v}'} + \tilde{X}_g + \tilde{X}_b\left(\frac{3(4-\bar{u}')}{4\bar{v}'} - 5\right)\right)\Delta Y_{uv}\right)^2\right)^{1/2}\cdot 100\ \%, \quad (8)$$

$$\varepsilon_G^{u'v'Y_{uv}} = \frac{1}{\tilde{Y}_r \frac{9\bar{Y}_{uv}\bar{u}'}{4\bar{v}'} + \tilde{Y}_g\bar{Y}_{uv} + \tilde{Y}_b\left(\frac{3\bar{Y}_{uv}(4-\bar{u}')}{4\bar{v}'} - 5\bar{Y}_{uv}\right)}\left(\left(\left(\tilde{Y}_r\frac{9\bar{Y}_{uv}}{4\bar{v}'} - \tilde{Y}_b\frac{3\bar{Y}_{uv}}{4\bar{v}'}\right)\Delta u'\right)^2 + \right.$$
$$\left. + \left(\left(\tilde{Y}_r\frac{9\bar{Y}_{uv}\bar{u}'}{-4\bar{v}'^2} + \tilde{Y}_b\frac{3\bar{Y}_{uv}(4-\bar{u}')}{-4\bar{v}'^2}\right)\Delta v'\right)^2 + \left(\left(\tilde{Y}_r\frac{9\bar{u}'}{4\bar{v}'} + \tilde{Y}_g + \tilde{Y}_b\left(\frac{3(4-\bar{u}')}{4\bar{v}'} - 5\right)\right)\Delta Y_{uv}\right)^2\right)^{1/2}\cdot 100\%; \quad (9)$$

$$\varepsilon_B^{u'v'Y_{uv}} = \frac{1}{\tilde{Z}_r \frac{9\bar{Y}_{uv}\bar{u}'}{4\bar{v}'} + \tilde{Z}_g\bar{Y}_{uv} + \tilde{Z}_b\left(\frac{3\bar{Y}_{uv}(4-\bar{u}')}{4\bar{v}'} - 5\bar{Y}_{uv'}\right)}\left(\left(\left(\tilde{Z}_r\frac{9\bar{Y}_{uv}}{4\bar{v}'} - \tilde{Z}_b\frac{3\bar{Y}_{uv}}{4\bar{v}'}\right)\Delta u'\right)^2 + \right.$$
$$\left. + \left(\left(\tilde{Z}_r\frac{9\bar{Y}_{uv}\bar{u}'}{-4\bar{v}'^2} + \tilde{Z}_b\frac{3\bar{Y}_{uv}(4-\bar{u}')}{-4\bar{v}'^2}\right)\Delta v'\right)^2 + \left(\left(\tilde{Z}_r\frac{9\bar{u}'}{4\bar{v}'} + \tilde{Z}_g + \tilde{Z}_b\left(\frac{3(4-\bar{u}')}{4\bar{v}'} - 5\right)\right)\Delta Y_{uv}\right)^2\right)^{1/2}\cdot 100\%; \quad (10)$$

$$\begin{cases} \Delta R^{u'v'Y_{uv}} = \dfrac{\varepsilon_R^{u'v'Y_{uv}}\left(\tilde{X}_r\frac{9\bar{Y}_{uv}\bar{u}'}{4\bar{v}'} + \tilde{X}_g\bar{Y}_{uv} + \tilde{X}_b\left(\frac{3\bar{Y}_{uv}(4-\bar{u}')}{4\bar{v}'} - 5\bar{Y}_{uv}\right)\right)}{100\%} \\ \Delta G^{u'v'Y_{uv}} = \dfrac{\varepsilon_G^{u'v'Y_{uv}}\left(\tilde{Y}_r\frac{9\bar{Y}_{uv}\bar{u}'}{4\bar{v}'} + \tilde{Y}_g\bar{Y}_{uv} + \tilde{Y}_b\left(\frac{3\bar{Y}_{uv}(4-\bar{u}')}{4\bar{v}'} - 5\bar{Y}_{uv}\right)\right)}{100\%} \\ \Delta B^{u'v'Y_{uv}} = \dfrac{\varepsilon_B^{u'v'Y_{uv}}\left(\tilde{Z}_r\frac{9\bar{Y}_{uv}\bar{u}'}{4\bar{v}'} + \tilde{Z}_g\bar{Y}_{uv} + \tilde{Z}_b\left(\frac{3\bar{Y}_{uv}(4-\bar{u}')}{4\bar{v}'} - 5\bar{Y}_{uv}\right)\right)}{100\%} \end{cases}, \quad (11)$$

где $\Delta u'$, $\Delta v'$, ΔY_{uv} — абсолютные погрешности (паспортные данные измерителя) прямых измерений координат цветности (u', v') и компоненты относительной яркости Y_{uv} изображения; $\bar{u}'$, $\bar{v}'$, $\bar{Y}_{uv}$ — средние значения измеренных параметров.

Выражения (8)—(11) получены в соответствии с известными положениями теории метрологии. Согласно этим положениям оценка относительной ε_ψ и абсолютной $\Delta\psi$ инструментальной погрешностей косвенного измерения величины ψ, вычисляемой по формуле $\psi = f(\xi_1, \xi_2, ..., \xi_\zeta)$, где $f(\cdot)$ — функция произвольного вида, с использованием результатов прямых измерений величин ξ_1, ξ_2, ..., ξ_ζ осуществляется следующим образом:

$$\varepsilon_\psi = \sqrt{\sum_{i=1}^{\zeta}\left(\frac{\partial \ln f\left(\xi_1, \xi_2, \ ..., \ \xi_\zeta\right)}{\partial\xi_i}\bigg|_{\xi_i=\bar{\xi}_i}\Delta\xi_i\right)^2}\cdot 100\ \%, \quad \Delta\psi = \frac{\varepsilon_\psi\bar{\psi}}{100\ \%}.$$

Здесь ζ — число аргументов функции $f(\cdot)$; $\partial\ln f(\xi_1, \xi_2, ..., \xi_\zeta)/\partial\xi_i$ — частные производные натурального логарифма от функции $f(\cdot)$ по аргументам ξ_i; $\Delta\xi_i$ — абсолютная погрешность прямого измерения величины ξ_i; $\bar{\psi}$, $\bar{\xi}_i$ — средние значения величин, измеряемых косвенным и прямым способами соответственно.

Логарифмическое дифференцирование в выражениях (8)—(10) для вычисления частных производных $\partial \ln R/\partial u'$, $\partial \ln R/\partial v'$, $\partial \ln R/\partial Y_{uv}$, $\partial \ln G/\partial u'$, $\partial \ln B/\partial v'$, $\partial \ln G/\partial Y_{uv}$, $\partial \ln B/\partial u'$, $\partial \ln B/\partial v'$, $\partial \ln B/\partial Y_{uv}$ выполнено по правилу

$$\frac{d \ln f(t)}{dt} = \frac{1}{f(t)} \frac{df(t)}{dt}.$$

Значения кодов *RGB*, полученных в результате косвенного измерения при заданном наборе отображаемых на экране тестовых цветов, являются исходными данными для методики коррекции технологического разброса [11] серийно выпускаемых экранов.

Перспективные колориметрические измерители с характеристиками $\Delta u' < 0{,}0001$ ед.; $\Delta v' < 0{,}0001$ ед.; $\Delta Y_{uv} < 0{,}01$ ед. позволяют получать результаты косвенного измерения компонентов кодов *RGB* с точностью до единицы младшего разряда кода основного (красный, зеленый, синий) цвета в 8-битной (по каждому цвету) модели *RGB*.

Заключение

В ходе исследования, результаты которого представлены в статье, получены в аналитическом виде математические выражения (4), (6), необходимые для расчета оценок коэффициентов $\widehat{X}_r$, $\widehat{Y}_r$, $\widehat{Z}_r$, $\widehat{X}_g$, $\widehat{Y}_g$, $\widehat{Z}_g$, $\widehat{X}_b$, $\widehat{Y}_b$, $\widehat{Z}_b$ профиля экрана, цветовоспроизведение которого основано на трехкомпонентной (*Red*, *Green*, *Blue*) схеме формирования наблюдаемого цвета.

При использовании в качестве тестовых следующих четырех цветов — красного, зеленого, синего, белого, при задании на экране соответствующих им кодов *RGB* = (255,0,0); (0,255,0); (0,0,255); (255,255,255) система (6) приводится к системе уравнений (4).

Система уравнений (4) позволяет получать оценки коэффициентов профиля экрана для типовых значений (u', v')-координат цветности, предоставляемых разработчиками экрана в технической документации. При этом не учитывается технологический разброс, присущий серийному производству экранов.

Система (6) позволяет получать оценки коэффициентов профиля каждого конкретного "обмеряемого" образца экрана, однако результаты измерений (u', v')-координат цветности получаются с погрешностью. В связи с этим для инструментального метода измерений необходимо использовать колориметры и спектрорадиометры с минимальным значением абсолютной погрешности измерений.

Получаемые в результате расчетов по формулам (4), (6) оценки коэффициентов составляют матрицу профиля экрана, которая согласно выражению (1) должна иметь обратную матрицу. Таким образом, по окончании вычисления коэффициентов матрицы профиля должна быть в обязательном порядке проведена оценка наличия у полученной матрицы профиля обратной матрицы. В случае если полученные с инструментальной погрешностью измерения (u', v')-координаты цветности позволяют получить матрицу профиля с нулевым значением определителя, то такие результаты измерений не могут быть приняты за основу в расчетах и эксперименты должны быть повторены для другой комбинации цветов, заданных на экране кодами *RGB*.

Система (7) позволяет количественно оценить смещение значений координат цветности одного и того же изображения, воспроизводимого на различных однотипных образцах экранов. Смещение может быть сведено к минимуму при использовании специализированной методики коррекции, которая является основой для всех методов калибровки экранов.

Выражения (8)—(11) позволяют количественно оценить погрешность косвенного измерения компонентов кодов *RGB* на основе прямых измерений (u', v')-координат цветности с учетом значений абсолютной и относительной погрешностей измерений. Выражения (8)—(11) имеют практическое значение для определения метрологических характеристик измерительного прибора в целях получения измеренных значений кодов *RGB*, погрешность определения которых не превышала бы единицы младшего разряда восьмибитной модели, т.е. значения, с точностью до которого цвет задается программно в вычислительной технике.

Список литературы

1. **Evanicky D., Granger Ed., Ingulsrud J., Meng A. T.** US Patent 2009/0051711 A1: Compact flat panel color calibration system, Feb. 26, 2009.
2. **Huang W., Li J.-M., Yang L.-M.** et al. Local dimming algorithm and color gamut calibration for RGB LED backlight LCD display // Optics & Laser technology. 2011. Vol. 43. P. 214—217.
3. **Mang O.-Y., Huang T.-W., Hsieh Y.-F., Kuob Y.-T.** Research of the chromaticity coordinates and color spectrum calibration using tristimulus sensors and eigenspectrum method // Proc. of SPIE "Optical inspection and metrology for non-optics industries". 2009. Vol. 7432. P. 743214.
4. **Menesatti P., Angelini C., Pallottino F.** et al. RGB color calibration for quantitative image analysis: the "3D thin-plate alpine" warping approach // Sensors. 2012. N. 12. P. 7063—7079.
5. **Rolkosky D. J., Dagnelie G., Kramer K.** et al. Calibration tools for PC-based vision assessment // Proc. of IEEE Eng. Med. Biol. Soc. 2009. P. 781—784.
6. **Wu Y. H., Hsieh Ch.-H., Kuo Ch.-H.** Color temperature calibration method and related devices. Patent US 7,893,946 B2, Feb. 22, 2011.
7. **Жаринов И. О., Жаринов О. О.** Оценка параметров математической модели цветопередачи жидкокристаллической панели // Информационно-управляющие системы. 2015. № 2. С. 49—56.
8. **Жаринов И. О., Жаринов О. О.** Исследование математической модели цветопередачи жидкокристаллических панелей // Программная инженерия. 2015. № 5. С. 32—42.
9. **Zargaryants G. S., Mikhailov O. M.** Integral remote colorimeter bases on the RGB colorimetric system// Light & Engineering. 2008. Vol. 16, N. 3. P. 69—77.
10. **Жаринов И. О., Жаринов О. О.** Теоретическая оценка функции плотности вероятности распределения координат цвета в системах бортовой индикации // Программная инженерия. 2015. № 3. С. 35—43.
11. **Жаринов И. О., Жаринов О. О.** Метод программной компенсации технологического разброса координат цветности жидкокристаллических панелей // Научно-технический вестник информационных технологий, механики и оптики. 2015. Т. 15, № 3. С. 387—397.
12. **Жаринов И. О., Жаринов О. О.** Оценка инструментальной погрешности косвенного измерения координат цвета в цветовой модели данных, применяемой в авионике // Программная инженерия. 2014. № 12. С. 39—46.

Software for Solving the Practical Problem of Calibrating Liquid Crystal Displays and Light-Emitting Diodes

I. O. Zharinov, igor_rabota@pisem.net, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University), Saint Petersburg, 197101, Russian Federation, Design Bureau "Electroavtomatika", Saint Petersburg, 198095, Russian Federation,
O. O. Zharinov, zharinov73@hotbox.ru, Saint-Petersburg State University of Aerospace Instrumentation, Saint Petersburg, 190000, Russian Federation

Corresponding author:

Zharinov Igor O., Chief of Department, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University), 197101, Saint Petersburg, Russian Federation
e-mail: igor_rabota@pisem.net

Received on January 15, 2016
Accepted on February 1, 2016

The screen of modern displays is made on the basis of either liquid crystals or light emitting diodes. There is a mathematical model, representing color reproduction characteristics. Parameters of color reproduction for any given display aren't known a-priori. The evaluation of those parameters might be performed either by skilful analysis of product's specification or with the use of data processing, obtained through colorimetric measurements. The purpose of the research is to obtain mathematical equations for evaluation of parameters of the color reproduction model of display with the use of colorimetric measurements in the UV color space. To develop mathematical equations for evaluation of parameters of the color reproduction model of display one need to use Grassman's laws. Initial data for the proposal method are: 1) values of chromaticity coordinates in the vertices of display's color gamut triangle and preset white point coordinates as well, specified in the UV color space; 2) set of chromaticity coordinates values for any four arbitrary chosen colors, displayed on the screen, obtained with the use of colorimetric measurements in UV color space. New mathematical expressions for evaluation of parameters of the color reproduction model of display, which may be obtained by two different methods; analytical expressions for evaluation of both absolute and relative components of hardware-caused inaccuracy of indirect measurements for two considered methods; evaluation of chromaticity coordinates shifts for visually percepted image in terms are proposed. Those expressions presume calculations based on values of chromaticity coordinates of four colors. First method is based on values of UV-chromaticity coordinates for Red, Green, Blue and White colors, taken from display's specification. Four different colors for the second proposed method may be chosen arbitrary.

Keywords: *color reproduction, parameters evaluation, chromaticity coordinates, color space*

For citation:

Zharinov I. O., Zharinov O. O. Software for Solving The Practical Problem of Calibrating Liquid Crystal Displays and Light-Emitting Diodes, *Programmnaya Ingeneria,* 2016, vol. 7, no. 4, pp. 173—180.

DOI: 10.17587/prin.7.173-180

References

1. **Evanicky D., Granger Ed., Ingulsrud J., Meng A. T.** Compact Flat Panel Color Calibration System, US Patent 2009/0051711 A1, Feb. 26, 2009.

2. **Huang W., Li J.-M., Yang L.-M., Jin Zh.-L., Zhong Zh.-G., Liu Y.** Local Dimming Algorithm and Color Gamut Calibration for RGB LED Backlight LCD Display, *Optics & Laser technology*, 2011, vol. 43, pp. 214—217.

3. **Mang O.-Y., Huang T.-W., Hsieh Y.-F., Kuob Y.-T.** Research of The Chromaticity Coordinates and Color Spectrum Calibration Using Tristimulus Sensors and Eigenspectrum Method, *Proc. of SPIE "Optical inspection and metrology for non-optics industries"*, 2009, vol. 7432, pp. 743214.

4. **Menesatti P., Angelini C., Pallottino F., Antonucci F., Aguzzi Y., Costa C.** RGB Color Calibration for Quantitative Image Analysis: The "3D Thin-Plate Alpine" Warping Approach, *Sensors*, 2012, no. 12, pp. 7063—7079.

5. **Rolkosky D. J., Dagnelie G., Kramer K., Havey G., Seifert G. J.** Calibration Tools for PC-based Vision Assessment, *Proc. of IEEE Eng. Med. Biol. Soc.*, 2009, pp. 781—784.

6. **Wu Y. H., Hsieh Ch.-H., Kuo Ch.-H.** Color Temperature Calibration Method and Related Devices, *Patent US 7,893,946 B2*, Feb. 22, 2011.

7. **Zharinov I. O., Zharinov O. O.** Ocenka parametrov matematicheskoj modeli cvetoperedachi zhidkokristallicheskoj paneli (Evaluation of parameters of the mathematical model of color response performances of LCD-panel), *Informacionno-Upravljajushhie Sistemy*, 2015, no. 2, pp. 49—56 (in Russian).

8. **Zharinov I. O., Zharinov O. O.** Issledovanie matematicheskoj modeli cvetoperedachi zhidkokristallicheskih panelej (The research of the mathematical model of color representation of LCD-panels, *Programmnaya Ingeneria*, 2015, no. 5, pp. 32—42 (in Russian).

9. **Zargaryants G. S., Mikhailov O. M.** Integral Remote Colorimeter Bases on The RGB Colorimetric System, *Light & Engineering*, 2008, vol. 16, no. 3, pp. 69—77.

10. **Zharinov I. O., Zharinov O. O.** Teoreticheskaja ocenka funkcii plotnosti verojatnosti raspredelenija koordinat cveta v sistemah bortovoj indikacii (Theoretical evaluation of the probability density function of color coordinates in on-board indication equipment), *Programmnaya Ingeneria*, 2015, no. 3, pp. 35—43 (in Russian).

11. **Zharinov I. O., Zharinov O. O.** Metod programmnoj kompensacii tehnologicheskogo razbrosa koordinat cvetnosti zhidkokristallicheskih panelej (The method of software-based compensation of technological variation in chromaticity coordinate values of LCD panels), *Nauchno-Tehnicheskii Vestnik Informatsionnykh Tekhnologii, Mekhaniki i Optiki*, 2015, vol. 15, no. 3, pp. 387—397 (in Russian).

12. **Zharinov I. O., Zharinov O. O.** Ocenka instrumental'noj pogreshnosti kosvennogo izmerenija koordinat cveta v cvetovoj modeli dannyh, primenjaemoj v avionike (The evaluation of hardware-caused inaccuracy of indirect measurements of chromaticity coordinates in color model data used in avionics), *Programmnaya Ingeneria*, 2014, no. 12, pp. 39—46 (in Russian).

УДК 519.61 DOI: 10.17587/prin.7.181-190

К. Ф. Иванова, канд. техн. наук, докторант, e-mail: Klara.I2010@yandex.ru,
Санкт-Петербургский государственный университет

Унификация точечных алгебраических методик внешней оценки решений интервальных систем

Предложен единообразный подход внешнего оценивания множества решений интервальных систем линейных алгебраических уравнений с неопределенными параметрами как в узком, так и в широком интервальном диапазоне изменения элементов для систем с M- и H-матрицами и их модификациями. Предлагаемый способ оценки основан на выборе точечных элементов интервальной матрицы, определяющих максимальные покомпонентные отклонения, из скалярных произведений граничных элементов исходной матрицы на их алгебраические дополнения с дальнейшей формализацией точечных алгебраических уравнений. Программная реализация алгоритмов позволяет определить наименьший интервальный вектор внешней оценки множества решений интервальных систем линейных алгебраических уравнений для систем с широким диапазоном неопределенных параметров аналогично процедуре определения чувствительности решения систем с узким диапазоном интервальности.

Ключевые слова: *интервальная система, неопределенность параметров, методика оценок, множество решений, внешняя оценка, точечные уравнения, чувствительность решения*

Введение

Задача внешнего оценивания объединенного множества решений интервальной системы линейных алгебраических уравнений (ИСЛАУ) занимает, безусловно, одно из центральных мест в исследовании интервальных систем и задач, на них основанных. Существуют четыре различных подхода к вычислению оптимальных (точных) оценок объединенного множества решений для интервальных линейных систем. Первым из таких подходов можно считать подход, описанный в работе У. Оеттли [1]. Он обнаружил, что пересечения объединенного множества решений с ортантами пространства R^n являются выпуклыми полиэдромами, что приводит к определению точных значений $\min\{x_v | x \in \Xi_{uni}\}$ из решения задач линейного программирования в каждом из ортантов. Подход Х. Янссона [2] основан на пассивной процедуре переборки решений системы, трудоемкость которой экспоненциально зависит от размерности системы n, приводящей к ограниченности использования методики. Следующие два вычислительных подхода к оптимальному решению "внешней задачи" для интервальных линейных систем — это методы дробления решений и методы дробления параметров (называемые также PSS-методами и PPS-методами), предложенные С. П. Шарым [3]. Для $n \times n$-системы сложность выполнения методов дробления решений в худшем случае пропорциональна 2^n, а верхняя оценка сложности выполнения методов дробления параметров равна даже 2^{n^2+n} в случае, когда для последующих шагов дробления параметров используются результаты предыдущих шагов.

Четвертый, наиболее популярный в зарубежной литературе подход к оптимальному решению "внешней задачи", принадлежит И. Рону [4]. Автор показывает, что исходя из уравнения Оеттли — Прагера $|\text{mid}\,\mathbf{A}\mathbf{x} - \text{mid}\,\mathbf{b}| = \text{rad}\,\mathbf{A}|\mathbf{x}| + \text{rad}\,\mathbf{b}$ для квадратной невырожденной матрицы $\mathbf{A}$ граничные значения с правыми частями $\mathbf{b}$

$$\left|\min\left\{x_v \mid x \in \Xi_{uni}(\mathbf{A},\mathbf{b})\right\},\ \max\left\{x_v \middle| x \in \Xi_{uni}(\mathbf{A},\mathbf{b})\right\}\right|, \quad v = \overline{1,n} \tag{1}$$

достигаются на множестве не более чем на 2^n решениях уравнений. Алгоритм Рона не является адаптивным, а его трудоемкость в худшем случае пропорциональна 4^n. Таким образом, все предложенные алгоритмы в общем случае имеют экспоненциальную сложность и принадлежат к так называемым NP-трудным задачам.

Постановку задачи по внешнему оцениванию множества решений ИСЛАУ можно причислить к задачам о параметрической чувствительности системы, когда решение задачи, вызванное интервальной неопределенностью входных данных, также принадлежит интервалу [5].

Что касается определения чувствительности решения СЛАУ с узкими границами элементов матриц, то известный подход к оценке погрешности решения с использованием числа обусловленности дает сильно завышенную оценку изменения нормы решения при

вариациях коэффициентов матрицы [6]. Оценка чувствительности решения с помощью так называемой "знаковой" методики ограничена в своем приложении, так как матрицы с погрешностью элементов, превышающих десятые и даже сотые доли процентов отклонений от своих средних значений, в зависимости от порядка системы, могут не соответствовать линейному приращению детерминанта [7]. В силу несоответствия выбора приращения детерминанта линейному закону требуется рассматривать приращения более высокого порядка, что может привнести дополнительные сложности в расчеты.

Предлагаемый в настоящей статье новый универсальный подход к внешней оценке множества решений ИСЛАУ основан на замене интервальной системы точечными алгебраическими уравнениями, формируемыми не в процессе общей или ограниченной процедуры переборки точечных матриц, а в результате получения экстремальных значений компонент двух угловых матриц. Они формируются из компонент скалярного произведения граничных элементов интервальной матрицы на их алгебраические дополнения при использовании полной интервальной арифметики Каухера.

Из решения полученных точечных уравнений определяются компоненты вектора решения x, имеющие оптимальные отклонения в положительном и отрицательном направлениях. Эти компоненты соответствуют минимальному интервальному вектору, понимаемому как вектор внешнего оценивания объединенного множества решений.

Гарантия оптимальности внешней оценки подтверждается тем, что при замене любого элемента в формализованных матрицах на противоположный (имеется в виду на противоположную границу интервального элемента) наименьший и наибольший детерминанты системы изменяются в неблагоприятную сторону, что приводит к нарушению оптимальности решения. Достоверность полученных по предлагаемой новой методике решений примеров подтверждается их совпадением с приводимыми в литературных источниках.

Новый подход может быть применен к системам не только с узкоинтервальными матрицами, но и с матрицами, интервальный диапазон изменения элементов которых может достигать 100 %.

Предлагаемая методика оценивается не экспоненциальной, а полиномиальной сложностью вычислительного алгоритма.

Постановка задачи по внешней оценке множества решений ИСЛАУ

В статье введены следующие обозначения:

$\mathbf{A} = \left[\underline{A}, \overline{A}\right]$ — интервальная матрица с левой и правой границами;

$\mathbf{b} = \left[\underline{b}, \overline{b}\right]$ — интервальный вектор правой части с левой и правой границами;

$\boldsymbol{a_{ij}} = \left[\underline{a}_{ij}, \overline{a}_{ij}\right]$ — интервальные элементы матрицы с левой и правой границами;

$\boldsymbol{b_i} = \left[\underline{b}_i, \overline{b}_i\right]$ — интервальные компоненты вектора правой части с левой и правой границами;

ε_{ij} — относительное приращение погрешности элемента a_{ij};

δ_i — относительное приращение погрешности компоненты правой части b_i;

Ξ_{uni} — объединенное множество решений интервальной системы линейных алгебраических уравнений;

E — декартово произведение векторов;

$\square\Xi_{uni}$ — оболочка множества решений;

$\mathbf{Ad} = \left[\underline{\mathrm{Ad}}^u, \overline{\mathrm{Ad}}^u\right]$ — интервальное алгебраическое дополнение к интервальному элементу матрицы с границами.

Система интервальных уравнений записывается в следующем виде:

$$\mathbf{A}x = \mathbf{b},\ \mathbf{A} \in \mathrm{IR}^{n\times n},\ \mathbf{b} \in \mathrm{IR}^n, \tag{2}$$

$$\mathbf{A} = \left[\underline{A}, \overline{A}\right] = \left\{A \in \mathrm{R}^{n\times n};\ \underline{A} \leqslant A \leqslant \overline{A}\right\},\ \underline{A}, \overline{A} \in \mathrm{R}^{n\times n};$$

$$\mathbf{b} = \left[\underline{b}, \overline{b}\right] = \left\{b \in \mathrm{R}^n;\ \underline{b} \leqslant b \leqslant \overline{b}\right\},\ \underline{b}, \overline{b} \in \mathrm{R}^n,$$

где элементы матрицы и компоненты вектора правой части $\boldsymbol{a_{ij}} = \left[\underline{a}_{ij}, \overline{a}_{ij}\right]$, $\boldsymbol{b_i} = [\underline{b}_i, \overline{b}_i]$ являются интервалами.

В интервальной алгебре внешнее оценивание состоит из покоординатных оценок множества решений ИСЛАУ $\Xi_{uni} = \left\{x \in \mathrm{R}^n \middle| (\exists A \in \mathbf{A},\ \exists b \in \mathbf{b}) \quad Ax = b\right\}$, образованного всеми решениями точечных систем $Ax = b$ с $A \in \mathbf{A}$, $b \in \mathbf{b}$, $A \in \mathrm{R}^{n\times n}$, $b \in \mathrm{R}^n$, соответствующих проекциям внешней оболочки множества решений системы на координатные оси $E \supseteq \square\Xi_{uni}$, где E — прямые произведения вещественных интервалов, что и соответствует наименьшему интервальному вектору оценки объединенного множества решений Ξ_{uni} (1).

Если интервальная матрица **A** неособенная, то объединенное множество решений $\Xi_{uni}(\mathbf{A}x, \mathbf{b})$ связно и компактно, т. е. ограничено.

В отличие от интервальных решений системы (2), принадлежащих интервальному пространству, полученные алгебраические векторы решений точечных уравнений $Ax = b$ принадлежат линейному Евклидову пространству с удвоенной по сравнению с интервальным пространством размерностью.

Теоретические предпосылки создания методики внешней оценки решений ИСЛАУ

Утверждение 1. Как и в случае вещественных квадратных матриц, детерминант неособенной интервальной матрицы выражается суммой поэлементных произведений интервальных элементов $\left[\underline{a}_{ij}, \overline{a}_{ij}\right]$ матрицы **A** на их алгебраические дополнения $\left[\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right]$ [6].

Доказательство. Рассмотрим интервальную матрицу второго порядка

$$\mathbf{A} = \begin{pmatrix} [\underline{a}_{11}, \overline{a}_{11}] & [\underline{a}_{12}, \overline{a}_{12}] \\ [\underline{a}_{21}, \overline{a}_{21}] & [\underline{a}_{22}, \overline{a}_{22}] \end{pmatrix}.$$

Запишем определитель этой матрицы, выраженный суммой произведений интервальных элементов интервалов их алгебраических дополнений:

$$\left|\mathbf{A}^u\right| = \left[\left|\underline{\mathrm{A}}^u\right|, \left|\overline{\mathrm{A}}^u\right|\right] = [\underline{a}_{11},\ \overline{a}_{11}][\underline{a}_{22},\ \overline{a}_{22}] + (-1)[\underline{a}_{21},\ \overline{a}_{21}][\underline{a}_{12},\ \overline{a}_{12}] =$$

$$= [\underline{a}_{11},\ \overline{a}_{11}]\left[\underline{\mathrm{Ad}}^u_{11},\ \overline{\mathrm{Ad}}^u_{11}\right] + [\underline{a}_{21},\ \overline{a}_{21}]\left[\underline{\mathrm{Ad}}^u_{21},\ \overline{\mathrm{Ad}}^u_{21}\right] = \sum_{i=1}^{2}\left[\underline{a}_{ij},\ \overline{a}_{ij}\right]\left[\underline{\ }\underline{\mathrm{Ad}}^u_{ij},\ \overline{\mathrm{Ad}}^u_{ij}\right],\ \text{для}\ \vee\, j = \overline{1,n}.$$

Для матрицы второго порядка знаки элементов и их алгебраических дополнений на побочных диагоналях противоположны.

Для матрицы третьего порядка

$$\mathbf{A} = \begin{pmatrix} \left[\underline{a}_{11},\ \overline{a}_{11}\right] & \left[\underline{a}_{12},\ \overline{a}_{12}\right] & \left[\underline{a}_{13},\ \overline{a}_{13}\right] \\ \left[\underline{a}_{21},\ \overline{a}_{21}\right] & \left[\underline{a}_{22},\ \overline{a}_{22}\right] & \left[\underline{a}_{23},\ \overline{a}_{23}\right] \\ \left[\underline{a}_{31},\ \overline{a}_{31}\right] & \left[\underline{a}_{32},\ \overline{a}_{32}\right] & \left[\underline{a}_{33},\ \overline{a}_{33}\right] \end{pmatrix}$$

разложение определителя по элементам первого ($j = 1$) столбца можно записать так:

$$\mathrm{Ad}_{11} = [\underline{a}_{22}, \overline{a}_{22}][\underline{a}_{33}, \overline{a}_{33}] - [\underline{a}_{32}, \overline{a}_{32}][\underline{a}_{23}, \overline{a}_{23}] \to \left[\underline{\mathrm{Ad}}^u_{11}, \overline{\mathrm{Ad}}^u_{11}\right];$$

$$\mathrm{Ad}_{21} = [\underline{\mathrm{Ad}}_{21}, \overline{\mathrm{Ad}}_{21}] = \left((-1)^{1+2}[\underline{a}_{12}, \overline{a}_{12}][\underline{a}_{33}, \overline{a}_{33}] - [\underline{a}_{32}, \overline{a}_{32}][\underline{a}_{13}, \overline{a}_{13}]\right) \to \left[\underline{\mathrm{Ad}}^u_{21}, \overline{\mathrm{Ad}}^u_{21}\right];$$

$$\mathrm{Ad}_{31} = [\underline{a}_{12}, \overline{a}_{12}][\underline{a}_{23}, \overline{a}_{23}] - [\underline{a}_{22}, \overline{a}_{22}][\underline{a}_{13}, \overline{a}_{13}] \to \left[\underline{\mathrm{Ad}}^u_{31}, \overline{\mathrm{Ad}}^u_{31}\right];$$

$$|\mathbf{A}| = \left[\left|\underline{\mathrm{A}}\right|, \left|\overline{\mathrm{A}}\right|\right] = \sum_{i=1}^{3} \boldsymbol{a}_{ij} \cdot \mathbf{Ad}_{ij} = [\underline{a}_{11}, \overline{a}_{11}]\left[\underline{\mathrm{Ad}}^u_{11}, \overline{\mathrm{Ad}}^u_{11}\right] + [\underline{a}_{21}, \overline{a}_{21}]\left[\underline{\mathrm{Ad}}^u_{21}, \overline{\mathrm{Ad}}^u_{21}\right] + [\underline{a}_{31}, \overline{a}_{31}]\left[\underline{\mathrm{Ad}}^u_{31}, \overline{\mathrm{Ad}}^u_{31}\right],\ \text{для}\ \vee\, j = \overline{1,n}.$$

Тогда определитель для матрицы n-го порядка будет иметь следующий вид:

$$|\mathbf{A}| = [|\underline{\mathrm{A}}|, |\overline{\mathrm{A}}|] = \sum_{i=1}^{n} \boldsymbol{a}_{ij} \cdot \mathbf{Ad}_{ij} = \sum_{i=1}^{n}\left[\underline{a}_{ij}, \overline{a}_{ij}\right]\left[\underline{\ }\underline{\mathrm{Ad}}^u_{ij},\ \overline{\mathrm{Ad}}^u_{ij}\right],\ \text{для}\ \vee\, j = \overline{1,n}. \quad (2)$$

Такой подход является прямым способом вычисления интервального детерминанта. Диапазон изменения алгебраических дополнений интервальной матрицы является интервальной величиной $p = \left[\underline{\ }\underline{\mathrm{Ad}}^u_{ij}, \overline{\mathrm{Ad}}^u_{ij}\right]$ с левой границей $\underline{\ }\underline{\mathrm{Ad}}^u_{ij}$ и правой границей $\overline{\mathrm{Ad}}^u_{ij}$. Его вычисление требует, в общем случае, длительной и трудоемкой работы. Значения точечных алгебраических дополнений, соответствующих левой и правой границам элемента a_{ij}, составляют интервал $q = \left[\underline{\ }\underline{\mathrm{Ad}}_{ij}, \mathrm{Ad}_{ij}\right]$ такой, что $q \subseteq p$, т. е. интервал q является включением в p.

Утверждение 2. Неособенная квадратная интервальная матрица $\mathbf{A} \in \mathrm{IR}^{n\times n}$ с узкими интервалами для элементов, не превышающими долей процентов, может быть представлена двумя точечными угловыми матрицами A^-, $\mathrm{A}^+ \in \mathrm{R}^{n\times n}$, детерминант каждой из которых определяет левую и правую границы интервального детерминанта $|\mathbf{A}| = \left(\left|\mathrm{A}^-\right|, \left|\mathrm{A}^+\right|\right)$.

Доказательство. Из утверждения 1 следует, что интервальный детерминант матрицы представляется двумя точечными границами $|\mathbf{A}| = \left(\left|\underline{\mathrm{A}}^u\right|, \left|\overline{\mathrm{A}}^u\right|\right)$. Вместе с тем для узкоинтервальных матриц показано, что две точечные матрицы A^- и A^+, полученные по "знаковой" методике [7], соответствуют наименьшему и наибольшему значениям детерминанта det(A) (который в общем случае может принимать 2^{n^2} значения):

$$\min\left(\det(\mathrm{A})\right) = \det\left(\mathrm{A}^-\right) = \left|\mathrm{A}^-\right|,$$

$$\max\left(\det(\mathrm{A})\right) = \det\left(\mathrm{A}^+\right) = \left|\mathrm{A}^+\right|, |\mathrm{A}| = \left|\mathrm{A}^-\right| + \left|\mathrm{A}^+\right|.$$

Применение нового универсального подхода, развиваемого в дальнейшем и применяемого для более широких интервалов интервальных элементов, позволяет получить аналогичные угловые точечные матрицы.

Следовательно, для интервальных матриц всегда можно построить две такие угловые матрицы A^- и A^+, элементам которых будут соответствовать границы интервального детерминанта:

$$\left(\left|\underline{\mathrm{A}}^u\right|, \left|\overline{\mathrm{A}}^u\right|\right) = \left(\left|\mathrm{A}^-\right|, \left|\mathrm{A}^+\right|\right). \quad (3)$$

Утверждение 3. Новый подход к конструкции угловых точечных матриц $\{\mathrm{A}^-, \mathrm{A}^+\} \in \mathrm{R}^{n\times n}$ и векторов $\{\mathrm{b}^-, \mathrm{b}^+\} \in \mathrm{R}^n$ осуществляется на основе фиксации элементов a^-_{ij} и a^+_{ij},, отождествляемых с границами

$\underline{a}_{ij}$ или $\overline{a}_{ij}$, и b^+ и b^-, отождествляемых с границами $\underline{b}_i$ или $\overline{b}_i$ путем их выбора из наибольших и наименьших значений парных компонент произведений

$$\left(\underline{a}_{ij}, \overline{a}_{ij}\right)\left(\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right) \text{ и } \left(\underline{b}_i, \overline{b}_i\right)\left(\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right). \quad (4)$$

Доказательство. Распишем скалярные произведения (4):

$$\begin{aligned}
&[[\underline{a}_{11}, \overline{a}_{11}]\left[\underline{\mathrm{Ad}}_{11}, \overline{\mathrm{Ad}}_{11}\right], \ldots [\underline{a}_{1n}, \overline{a}_{1n}]\left[\underline{\mathrm{Ad}}_{1n}, \overline{\mathrm{Ad}}_{1n}\right],\\
&[\underline{a}_{21}, \overline{a}_{21}]\left[\underline{\mathrm{Ad}}_{21}, \underline{\mathrm{Ad}}_{22}\right] \ldots [\underline{a}_{2n}, \overline{a}_{22}]\left[\underline{\mathrm{Ad}}_{2n}, \overline{\mathrm{Ad}}_{2n}\right], \ldots \quad (5)\\
&\ldots [\underline{a}_{n1}, \overline{a}_{n1}]\left[\underline{\mathrm{Ad}}_{n1}, \underline{\mathrm{Ad}}_{n2}\right], \ldots [\underline{a}_{nn}, \overline{a}_{nn}]\left[\underline{\mathrm{Ad}}_{nn}, \overline{\mathrm{Ad}}_{nn}\right],\\
&\left[\underline{b}_1, \overline{b}_1\right]\left[\underline{\mathrm{Ad}}_{11}, \overline{\mathrm{Ad}}_{11}\right], \ldots \left[\underline{b}_1, \overline{b}_1\right]\left[\underline{\mathrm{Ad}}_{1n}, \overline{\mathrm{Ad}}_{1n}\right],\\
&\left[\underline{b}_2, \overline{b}_2\right]\left[\underline{\mathrm{Ad}}_{21}, \underline{\mathrm{Ad}}_{22}\right] \ldots \left[\underline{b}_2, \overline{b}_2\right]\left[\underline{\mathrm{Ad}}_{2n}, \overline{\mathrm{Ad}}_{2n}\right], \ldots \quad (6)\\
&\ldots \left[\underline{b}_n, \overline{b}_n\right]\left[\underline{\mathrm{Ad}}_{n1}, \underline{\mathrm{Ad}}_{n2}\right], \ldots \left[\underline{b}_n, \overline{b}_n\right]\left[\underline{\mathrm{Ad}}_{nn}, \overline{\mathrm{Ad}}_{nn}\right].
\end{aligned}$$

Границы интервала $\left[\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right]$ представляют собой точечные алгебраические дополнения к точечным значениям границ $\underline{a}_{ij}$ и $\overline{a}_{ij}$, в отличие от выражений алгебраических дополнений в формуле (2), получаемых по правилам интервальной алгебры.

Запишем выражения для выявления минимальных и максимальных значений a_{ij}^- и a_{ij}^+, а также b_i^+ и b_i^-:

$$\begin{aligned}
[a][\mathrm{Ad}] = \Big[&\min\left(\underline{a}_{ij}\underline{\mathrm{Ad}}_{ij}, \underline{a}_{ij}\overline{\mathrm{Ad}}_{ij}, \overline{a}_{ij}\underline{\mathrm{Ad}}_{ij}, \overline{a}_{ij}\overline{\mathrm{Ad}}_{ij}\right),\\
&\max\left(\underline{a}_{ij}\underline{\mathrm{Ad}}_{ij}, \underline{a}_{ij}\overline{\mathrm{Ad}}_{ij}, \overline{a}_{ij}\underline{\mathrm{Ad}}_{ij}, \overline{a}_{ij}\overline{\mathrm{Ad}}_{ij}\right)\Big],
\end{aligned} \quad (7)$$

$$\begin{aligned}
[b][\mathrm{Ad}] = \Big[&\min\left(\underline{b}_i\underline{\mathrm{Ad}}_{ij}, \underline{b}_i\overline{\mathrm{Ad}}_{ij}, \overline{b}_i\underline{\mathrm{Ad}}_{ij}, \overline{b}_i\overline{\mathrm{Ad}}_{ij}\right),\\
&\max\left(\underline{b}_i\underline{\mathrm{Ad}}_{ij}, \underline{b}_i\overline{\mathrm{Ad}}_{ij}, \overline{b}_i\underline{\mathrm{Ad}}_{ij}, \overline{b}_i\overline{\mathrm{Ad}}_{ij}\right)\Big].
\end{aligned} \quad (8)$$

Среди интервалов $\left(\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right)$ могут быть правильные и неправильные, когда левая граница больше правой — $\underline{\mathrm{Ad}}_{ij} > \overline{\mathrm{Ad}}_{ij}$. Поэтому покомпонентное определение минимальных и максимальных границ интервалов по формулам (7), (8) можно интерпретировать с позиции полной интервальной арифметики Каухера. Таким образом, из каждой пары сомножителей, входящих в выражения (7) и (8), можно выбрать наименьшее и наибольшее значения и уже из этих значений выделить те границы элементов $\underline{a}_{ij}$ или $\overline{a}_{ij}$ и $\underline{b}_i$ или $\overline{b}_i$, которым будут соответствовать a_{ij}^- или a_{ij}^+ и b_i^+ или b_i^-.

Составим суммы произведений

$$Z1 = \sum_{i=1}^{n}\left[\underline{a}_{ij}\ \underline{\mathrm{Ad}}_{ij}, \overline{a}_{ij}\overline{\mathrm{Ad}}_{ij}\right],\ Z2 = \sum_{i=1}^{n}\left[a_{ij}^-\mathrm{Ad}_{ij}^-, a_{ij}^+\mathrm{Ad}_{ij}^+\right],$$
$$\text{для } \vee\ j = \overline{1,n}.$$

Как следует из выражения (3), $Z1$ — сумма произведений элементов граничных матриц и их точечных алгебраических дополнений; $Z2$ — выражение для определителя матрицы **A** через его границы. Тогда $Z1 \subseteq Z2$. Условие строго равенства выполняется, когда $\underline{a}_{ij} = a_{ij}^-,\ \overline{a}_{ij} = a_{ij}^+,\ \underline{\mathrm{Ad}}_{ij} = \mathrm{Ad}_{ij}^-,\ \overline{\mathrm{Ad}}_{ij} = \mathrm{Ad}_{ij}^+,\ i, j = \overline{1,n}$.

Тот случай, когда одна и та же граница элемента участвует в двух парных произведениях (7), (8), доставляющих им минимум и максимум, означает, что этот элемент будет присутствовать в обеих новых матрицах A^- и A^+.

Утверждение 4. Основанием для использования нового подхода к конструированию точечных матриц A^- и A^+, определяющих границы интервального детерминанта, является совпадение по знаку границ интервалов Ad_{ij}^- и Ad_{ij}^+, что выполняется для узкоинтервальных матриц и всегда имеет место для систем с одинаковыми знаками обеих границ элементов матрицы.

Доказательство. Одним из допущений "знаковой" методики является то, что алгебраические дополнения к граничным элементам $\left(\mathrm{Ad}_{ij}^-, \mathrm{Ad}_{ij}^+\right)$ должны иметь одинаковые знаки для границ: $\mathrm{Ad}_{ij}^- > 0$ и $\mathrm{Ad}_{ij}^+ > 0$, $\mathrm{Ad}_{ij}^- \geqslant 0$ и $\mathrm{Ad}_{ij}^+ > 0$, $\mathrm{Ad}_{ij}^- = 0$ и $\mathrm{Ad}_{ij}^+ = 0$ либо $\mathrm{Ad}_{ij}^- < 0$ и $\mathrm{Ad}_{ij}^+ < 0$, $\mathrm{Ad}_{ij}^- < 0$ и $\mathrm{Ad}_{ij}^+ \leqslant 0$. Тогда знаки средних значений соответствуют знакам одной из границ интервального элемента $\mathrm{Ad}_{ij} = \left(\mathrm{Ad}_{ij}^-, \mathrm{Ad}_{ij}^+\right)$, а именно $\mathrm{sign}\left(\mathrm{mid}\left(\mathrm{Ad}_{ij}\right)\right) = \mathrm{sign}\left(\mathrm{Ad}_{ij}^-\right) = \mathrm{sign}\left(\mathrm{Ad}_{ij}^+\right)$. Так, для *M*- и *H*-матриц и их модификаций эти условия всегда соблюдаются.

Для широких интервалов, превышающих 100 % относительных отклонений, с разнознаковыми границами элементов матриц, это равенство может не соблюдаться, и к выбору элементов a_{ij}^- и a_{ij}^+ и b_i^+ и b_i^- из условий (7) и (8) по правилам полной интервальной арифметики Каухера должны быть присоединены дополнительные условия. Варианты таких решений будут описаны в последующих работах.

Схематически выбор и построение угловых точечных систем можно представить следующей последовательностью действий:

$$\begin{aligned}
&\begin{cases}\boldsymbol{a}_{ij} = \left[\underline{a}_{ij}, \overline{a}_{ij}\right]\\ \mathbf{Ad}_{ij} = \left[\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right]\end{cases} \to \min, \max\begin{cases}\underline{a}_{ij}\underline{\mathrm{Ad}}_{ij}, \underline{a}_{ij}\overline{\mathrm{Ad}}_{ij}\\ \overline{a}_{ij}\underline{\mathrm{Ad}}_{ij}, \overline{a}_{ij}\overline{\mathrm{Ad}}_{ij}\end{cases} \to\\
&\to \left(a_{ij}^-, a_{ij}^+\right) \to \left(\mathrm{A}^-, \mathrm{A}^+\right) \to \left[\left|\mathrm{A}^-\right|, \left|\mathrm{A}^+\right|\right] = |\mathbf{A}|,
\end{aligned} \quad (9)$$

$$\begin{aligned}
&\begin{cases}\boldsymbol{b}_{ij} = \left[\underline{b}_i, \overline{b}_i\right]\\ \mathbf{Ad}_{ij} = \left[\underline{\mathrm{Ad}}_{ij}, \overline{\mathrm{Ad}}_{ij}\right]\end{cases} \to \min, \max\begin{cases}\underline{b}_i\underline{\mathrm{Ad}}_{ij}, \underline{b}_i\overline{\mathrm{Ad}}_{ij}\\ \overline{b}_i\underline{\mathrm{Ad}}_{ij}, \overline{b}_i\overline{\mathrm{Ad}}_{ij}\end{cases} \to\\
&\to \left(b_i, b_i^+\right) \to \left(\mathrm{A}_{b_i}^-, \mathrm{A}_{b_i}^+\right) \to \left[\left|\mathrm{A}_{b_i}^-\right|, \left|\mathrm{A}_{b_i}^+\right|\right] = |\mathbf{A}_{\boldsymbol{b}}|, i, j = \overline{1,n}.
\end{aligned} \quad (10)$$

Одновременно формируются матрицы $\underline{\mathrm{Ab}} = \underline{\mathrm{A}}^{\underline{b}}$ и $\overline{\mathrm{Ab}} = \overline{\mathrm{A}}^{\overline{b}}$, полученные при подстановке *j*-х столбцов пра-

вых частей в числители исходных граничных матриц $\underline{\mathrm{Ab}} = \underline{\mathrm{A}}^{\underline{b}}$ и $\overline{\mathrm{Ab}} = \overline{\mathrm{A}}^{\overline{b}}$ и матрицы $\mathrm{A}^{-}_{a^{+}}$ и $\mathrm{A}^{+}_{a^{-}}$, полученные при подстановке j-х столбцов элементов a^{+}_{j} и a^{-}_{j} в знаменатели этих матриц. Точечные элементы, формирующие угловые матрицы, находятся из элементов первых двух матриц $\underline{\mathrm{Ab}} = \underline{\mathrm{A}}^{\underline{b}}$ и $\overline{\mathrm{Ab}} = \overline{\mathrm{A}}^{\overline{b}}$, а также из алгебраических дополнений соответствующих элементов $\underline{\mathrm{Adb}} = \mathrm{Ad}\left(\underline{\mathrm{A}}^{\underline{b}}\right)$, $\overline{\mathrm{Adb}} = \mathrm{Ad}\left(\overline{\mathrm{A}}^{\overline{b}}\right)$. Вторые две матрицы $\mathrm{A}^{-}_{a^{+}}$ и $\mathrm{A}^{+}_{a^{-}}$ необходимы для формирования точечных уравнений:

$$\begin{cases} \underline{\mathrm{Ab}} = \underline{\mathrm{A}}^{\underline{b}} \\ \overline{\mathrm{Ab}} = \overline{\mathrm{A}}^{\overline{b}} \\ \underline{\mathrm{Adb}} = \mathrm{Ad}\left(\underline{\mathrm{A}}^{\underline{b}}\right) \\ \overline{\mathrm{Adb}} = \mathrm{Ad}\left(\overline{\mathrm{A}}^{\overline{b}}\right) \end{cases} \rightarrow \min, \max \begin{cases} \underline{ab}_{ij}\,\underline{\mathrm{Adb}}_{ij}, \underline{ab}_{ij}\,\overline{\mathrm{Adb}}_{ij} \\ \overline{ab}_{iji}\,\underline{\mathrm{Adb}}_{ij}, \overline{ab}_{iji}\,\overline{\mathrm{Adb}}_{ij} \end{cases} \rightarrow$$

$$\rightarrow \left(ab^{-}_{ij}, ab^{+}_{ij}\right) \rightarrow \begin{cases} \left(\mathrm{Ab}^{(-)}, \mathrm{Ab}^{(+)}\right) \\ \left(\mathrm{A}^{(-)}_{a^{+}}, \mathrm{A}^{(+)}_{a^{-}}\right) \end{cases}, \quad i, j = \overline{1, n}. \tag{11}$$

Система из двух парных формализованных матриц (11), входящих в угловые уравнения, позволяет получить еще одно множество решений ИСЛАУ, претендующее на максимальные покомпонентные отклонения.

Для узких интервалов, когда знаки точечных алгебраических дополнений граничных элементов совпадают, выражения (6) и (7) можно записать в упрощенном варианте, а именно — алгебраические дополнения представить их средними значениями, а граничные элементы матрицы **A** и компоненты правой части **b** через их средние значения и относительные приращения $\varepsilon = \max|\varepsilon_{ij}|$ и $\delta = \max|\delta_i|$. Тогда границы возмущенных элементов и компонент будут иметь следующий вид: $\underline{a}_{ij} = a_{ij} - \varepsilon$; $\overline{a}_{ij} = a_{ij} + \varepsilon$; $\underline{b}_i = b_i - \delta$; $\overline{b}_i = b_i + \delta$, а уравнения (9), (10) запишутся как

$$\begin{cases} \left(a_{ij} - \varepsilon, a_{ij} + \varepsilon\right) \mathrm{mid}\left(\mathrm{Ad}_{ij}\right) \\ \left(b_i - \delta, b_i + \delta\right) \mathrm{mid}\left(\mathrm{Ad}_{ij}\right). \end{cases} \tag{12}$$

Записав систему (12) в соответствии с выражением (8), получим:

$$\left[\boldsymbol{a}_{ij}\right]\left[\mathbf{Ad}_{ij}\right] =$$
$$= \min\left[\left(a_{ij} - \varepsilon\right)\mathrm{mid}\left(\mathrm{Ad}_{ij}\right), \left(a_{ij} + \varepsilon\right)\mathrm{mid}\left(\mathrm{Ad}_{ij}\right)\right],$$
$$\max\left[\left(a_{ij} - \varepsilon\right)\mathrm{mid}\left(\mathrm{Ad}_{ij}\right), \left(a_{ij} + \varepsilon\right)\mathrm{mid}(\mathrm{Ad}_{ij})\right].$$

Отсюда автоматически фиксируются минимальное и максимальное значения $a_{ij} - \varepsilon$ или $a_{ij} + \varepsilon$, которым соответствуют a^{-}_{ij} и a^{+}_{ij}.

Следствие 1. Утверждения 1—4 состоятельны как в применении к любым невырожденным матрицам с узкими интервалами элементов, так и к *M*- и *H*-матрицам и их модификациям. В этом отношении оба типа матриц (с узкими и широкими интервалами) обладают свойствами регулярных матриц, когда остаются непрерывными не только исходные, но и обратные матрицы. В частности, *M*-матрицы обладают положительной определенностью обратной матрицы, так что $\mathbf{A}^{-1} = \left[\overline{\mathrm{A}}^{-1}, \underline{\mathrm{A}}^{-1}\right] \geqslant 0$ является наименьшей интервальной матрицей, которая содержит множество $\{\mathrm{A}^{-1}: \mathrm{A} \in \mathbf{A}\}$.

Следствие 2. Для граничных точечных интервалов элементов матриц интервалы их алгебраических дополнений являются неправильными (для внедиагональных элементов).

Следствие 3. Существенное различие при использовании идентичных подходов к формализации угловых матриц с узкими интервальными элементами по сравнению с матрицами с широкими интервалами состоит в том, что система в первом случае может оставаться невырожденной при значительном увеличении порядка матрицы и при отсутствии даже слабого диагонального преобладания элементов. Что касается второго случая, то рассчитывать на соблюдение неособенности системы при росте порядка системы можно только для незаполненных и диагональных матриц. Такое может происходить при численной аппроксимации дифференциальных уравнений, в том числе уравнений в частных производных.

Следствие 4. Общий принцип, который позволяет объединить предлагаемый подход внешней оценки множества решений (с узкими и широкими интервалами неопределенности), заключается в том, что относительные погрешности элементов матриц в обоих случаях не превышают 100 % и выбор элементов граничных матриц из выражений (7) и (8) полностью соответствует правилам полной интервальной арифметики Каухера.

Формирование точечных алгебраических уравнений

В соответствии с предлагаемой методикой конструируются угловые точечные матрицы A^{+} и A^{-}и векторы b^{+} и b^{-}: $\mathrm{A}^{-} \in \mathbf{A}$, $\mathrm{A}^{+} \in \mathbf{A}$, $\mathrm{b}^{-} \in \mathbf{b}$, $\mathrm{b}^{+} \in \mathbf{b}$, на основании которых формируются системы уравнений, позволяющие в процессе их решения получить два искомых вектора x^{+} и x^{-} с компонентами $\mathrm{x}_k = \left[\mathrm{x}^{-}_k, \mathrm{x}^{+}_k\right]$, $k = \overline{1, n}$:

$$\begin{cases} \mathrm{A}^{+}x^{-} = \mathrm{b}^{-}, \\ \mathrm{A}^{-}x^{+} = \mathrm{b}^{+}; \end{cases} \tag{13, \textit{а}}$$

$$\begin{cases} \mathrm{A}^{+}_{a^{-}}x^{+} = \mathrm{b}^{+}, \\ \mathrm{A}^{-}_{a^{+}}x^{-} = \mathrm{b}^{-}. \end{cases} \tag{13, \textit{б}}$$

Из решения системы 13, *а,* и 13, *б,* могут быть вычислены значения векторов x^{+} и x^{-}, имеющие максимальные отклонения в положительном и отрицательном направлениях, наибольшие из которых выбираются путем сравнения отдельных компонент для x^{+} и x^{-}, формирующие прямоугольный параллелотоп $\left(x^{\min}_i, x^{\max}_i\right) \times \left(x^{\min}_j, x^{\max}_j\right)$.

Для точечных уравнений (13) этот процесс легко реализуется через обратные матрицы, когда обратная интервальная матрица также представляется двумя границами $\mathbf{A}^{-1} = [(\mathrm{A}^{-})^{-1}, (\mathrm{A}^{+})^{-1}]$. Непрерывность обратной матрицы при этом может и не сохраняться. На выборе максимальных и минимальных отклонений x^{+} и x^{-}, полученных из систем (13), создаются компоненты наименьшего интервального вектора, превосходящего все возможные решения системы (2).

Обоснование. Предложенные оценки могут быть проверены по формулам Крамера. При подстановке b^{+} и b^{-} в наибольшие и наименьшие детерминанты $|\mathrm{A}^{-}|$ и $|\mathrm{A}^{+}|$ вычисляются наибольшие и наименьшие значения компонент вектора x^{+} и x^{-}:

$$x_i^{+} = \frac{\left|\mathrm{A}_{b_i^{+}}^{-}\right|}{\left|\mathrm{A}^{-}\right|} \text{ и } x_i^{-} = \frac{\left|\mathrm{A}_{b_i^{-}}^{-}\right|}{\left|\mathrm{A}^{-}\right|} \text{ или в другом варианте}$$
$$x_i^{+} = \frac{\left|\mathrm{A}_{a^{-}b_i^{+}}^{+}\right|}{\mathrm{A}_{a^{-}}^{+}} \text{ и } x_i^{-} = \frac{\left|\mathrm{A}_{a^{+}b_i^{-}}^{-}\right|}{\mathrm{A}_{a^{+}}^{-}}. \quad (14)$$

Из формул (14) видно, что, как правило, наименьшие значения компонент вектора $\mathbf{x} = \{\mathrm{x}^{-}, \mathrm{x}^{+}\}$ получаются при подстановках минимальной компоненты правой части b^{-} в j-й столбец максимальной матрицы, а наибольшие — при подстановках максимальной компоненты правой части b^{+} в j-й столбец минимальной матрицы.

Если в сконструированной матрице, определяющей минимальный детерминант, один из элементов a_{ij}^{-} заменить на противоположный ему элемент a_{ij}^{+}, то это приведет к возрастанию детерминанта $|\mathrm{A}^{-}|$. При замене a_{ij}^{+} на a_{ij}^{-} в матрице A^{+} определитель $|\mathrm{A}^{+}|$, наоборот, уменьшится, что противоречит построению новых граничных матриц в соответствии с выражением (2).

Для систем уравнений (13) компоненты x^{+} и x^{-}, соответствующие максимальным и минимальным значениям вектора решения x, являются откликом на входную неопределенность, выраженную интервальным представлением матриц и правых частей системы. Вектор решения $x = \left(\left[x_1^{-}, x_1^{+}\right], \ldots \left[x_n^{-}, x_n^{+}\right]\right)^{\mathrm{T}}$, включающий двойной набор компонент, можно интерпретировать как интервальный вектор, т. е. прямое произведение одномерных векторов, или брус, гарантированно содержащий множество решений рассматриваемой ИСЛАУ. Этот брус является наименьшим интервальным вектором внешней оценки.

Угловые точечные матрицы A^{+} и A^{-} и векторы b^{+} и b^{-} отвечают теореме Бека—Никеля [8], так как в этом случае конструируются две угловые точечные матрицы $\mathrm{A}^{-} \in \mathbf{A}$, $\mathrm{A}^{+} \in \mathbf{A}$ и $\mathrm{b}^{+} \in \mathbf{b}$ и $\mathrm{b}^{-} \in \mathbf{b}$, формирующие крайние точечные уравнения (13, *а*) и (13, *б*). В результате их решения получаются экстремальные значения $\mathbf{x}^{-}$и $\mathbf{x}^{+}$, соответствующие $\min\{x_v \in \Xi_{uni}(\mathrm{A}, \mathrm{b})\}$ и $\max\{\mathrm{x}_v \in \Xi_{uni}(\mathrm{A}, \mathrm{b})\}$ объединенного множества решений $\Xi_{uni}(\mathbf{A}, \mathbf{b})$.

Обязательным условием получения решения интервальной системы по внешней оценке является соблюдение неособенности интервальной матрицы. Это утверждение, соответствующее выполнению критерия Баумана [8], состоит в том, что интервальная матрица неособенна тогда и только тогда, когда определители всех ее крайних матриц A′ и A″ имеют одинаковые знаки, т. е. $\det(\mathrm{A}') \cdot \det(\mathrm{A}'') > 0$, что соблюдается для граничных матриц A^{+} и A^{-}.

Приведем примеры, подтверждающие предлагаемый подход решений ИСЛАУ.

Пример 1. Дано:

$$\mathrm{A} = \begin{pmatrix} [3,7,\ 4,3] & [-1,5,\ -0,5] & [0,\ 0] \\ [-1,5,\ -0,5] & [3,7,\ 4,3] & [-1,5,\ -0,5] \\ [0,\ 0] & [-1,5,\ -0,5] & [3,7,\ 4,3] \end{pmatrix},$$
$$\mathrm{b} = \left([-14,14]; [-9,9]; [-3,3]\right),$$
$$\mathrm{x1} = \begin{pmatrix} [-6,38,\ 6,38] \\ [-6,40,\ 6,40] \\ [-3,40,\ 3,40] \end{pmatrix},\ \mathrm{x2} = \begin{pmatrix} [-6,3767,\ 6,3767] \\ [-6,3982,\ 6,3982] \\ [-3,4047,\ 3,4047] \end{pmatrix}.$$

Детерминанты матриц равны $|\mathrm{A}| = 56$, $|\mathrm{A}^{-}| = 34$, $|\mathrm{A}^{+}| = 77{,}36$, откуда видны наименьшее и наибольшее значения определителей в качестве границ интервального детерминанта.

Решение x1 получено методом Гаусса и неоднократно цитировалось в литературных источниках [9]. Решение x2 получено по алгоритму предлагаемой методики. Левые и правые точечные матрицы исходной системы и их точечные алгебраические дополнения записываются следующим образом:

$$\mathrm{A1} = \begin{pmatrix} 3,7000 & 1,5000 & 0 \\ -1,5000 & 3,7000 & 1,5000 \\ 0 & -1,5000 & 3,7000 \end{pmatrix},$$
$$\mathrm{A2} = \begin{pmatrix} 4,3000 & -0,5000 & 0 \\ -0,5000 & 4,3000 & -0,5000 \\ 0 & -0,5000 & 4,3000 \end{pmatrix},$$
$$\mathrm{Ad1} = \begin{pmatrix} 11,4400 & 5,5500 & 2,2500 \\ 5,5500 & 13,6900 & 5,5500 \\ 2,2500 & 5,5500 & 11,4400 \end{pmatrix},$$
$$\mathrm{Ad2} = \begin{pmatrix} 18,2400 & 2,1500 & 0,2500 \\ 2,1500 & 18,4900 & 2,1500 \\ 0,2500 & 2,1500 & 18,2400 \end{pmatrix}.$$

Для скалярных произведений элементов обеих систем $\boldsymbol{Z}_{11} = \boldsymbol{a}_{11}\mathbf{Ad}_{11}$ и $\boldsymbol{Z}_{21} = \boldsymbol{a}_{21}\mathbf{Ad}_{21}$ получим

$$\boldsymbol{Z}_{11} = (3,7,\ 4,3)(11,44,\ 18,24) = 3,7 \cdot 11,44,$$
$$3,7 \cdot 18,24,\ 4,3 \cdot 11,44,$$
$$4,3 \cdot 18,24 \rightarrow \min\left(\boldsymbol{Z}_{11}\right) = 3,7 \cdot 11,44 \rightarrow a_{11}^{-} = 3,7;$$
$$\max\left(\boldsymbol{Z}_{11}\right) = 4,3 \cdot 18,24 \rightarrow a_{11}^{+} = 4,3;$$

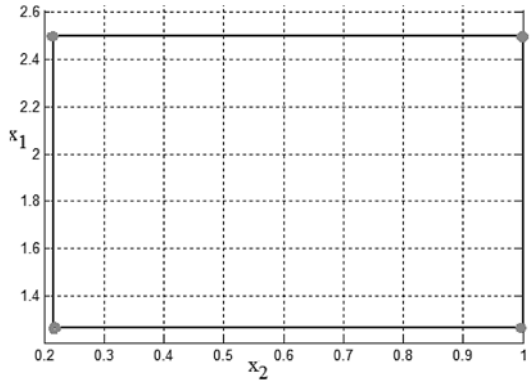


Рис. 1. Брус из оптимальных компонент решения

$$\boldsymbol{Z}_{21} = (-1,5,-0,5)(5,55,\ 2,15) = (-1,5)\cdot 5,55,$$
$$(-1,5)\cdot 2,15,\ (-0,5)\cdot 5,55,\ (-0,5)\cdot 2,15 \rightarrow$$
$$\rightarrow \min(\boldsymbol{Z}_{21}) = (-1,5)\cdot 5,55 \rightarrow a_{21}^{-} = -1,5;$$
$$\max(\boldsymbol{Z}_{21}) = (-0,5)\cdot 2,15 \rightarrow a_{21}^{+} = -0,5.$$

Остальные элементы a_{ij}^{-} и a_{ij}^{+} матриц A1 и A2 вычисляются по этим же правилам и участвуют в формировании матриц $A^{-}\{a_{ij}^{-}\}$ и $A^{+}\{a_{ij}^{+}\}$. Решение x2 получено по новой методике из систем (13).

Пример 2. Рассмотрим вариант с разными знаками одного из элементов.

$$\mathbf{A} = \begin{pmatrix} [5,6] & [-1,-2] \\ [-2,1] & [4,6] \end{pmatrix},\ \mathbf{b} = [0,\ 8].$$

В работе [10] решение получено прямым построением множества решений по частям в каждом отдельном ортанте: $\mathrm{x1} = \begin{pmatrix} [0,2162,\ 1] \\ [1,25\ 2,5] \end{pmatrix}$. По предложенной методике $\mathrm{x2} = \begin{pmatrix} [0,216216,\ 1] \\ [1,26316,\ 2,5] \end{pmatrix}$.

На рис. 1 изображен брус, составленный из экстремальных компонент решения x2.

Пример 3. Рассмотрим систему с *H*-матрицей из пакета программ IntlinIncR3 И. А. Шарой.

Решение из пакета совпадает с решением по предложенной методике:

$$\mathbf{A} = \begin{pmatrix} 3,5 & [0,2] & [0,2] \\ [0,2] & 3,5 & [0,2] \\ [0,2] & [0,2] & 3,5 \end{pmatrix},\ \mathrm{b} = \left([-1\ 1];\ [1\ 1];\ [-1\ 1]\right);$$

$$\mathrm{x1} = \mathrm{x2} = \begin{pmatrix} [-0,84,0,84] \\ [-0,84,0,84] \\ [-0,84,0,84] \end{pmatrix};$$

$|\mathrm{A}| = 34,375$, $|\mathrm{A}^{-}| = 16,875$, $|\mathrm{A}^{+}| = 42,875$ — вычисленные детерминанты угловых матриц. Применение матрицы предобуславливания не изменило решения, в то время как изменились новые значения матриц, правых частей и соответствующие им детерминанты: $|\mathrm{A}| = 1$, $|\mathrm{A}^{-}| = 0,6072$, $|\mathrm{A}^{+}| = 3,3814$.

Применение матрицы предобуславливания дает широкий спектр решений, но иногда, как в данном случае, эти решения совпадают с полученными и без предобуславливания.

Пример 4. Матрица **A** имеет симметричные относительно нуля элементы. Применена процедура предобуславливания системы обратной диагональной матрицей:

$$\mathbf{A} = \begin{pmatrix} [4,6] & [-1,1] & [-1,1] & [-1,1] \\ [-1,1] & [-6,-4] & [-1,1] & [-1,1] \\ [-1,1] & [-1,1] & [9,11] & [-1,1] \\ [-1,1] & [-1,1] & [-1,1] & [-11,-9] \end{pmatrix},\ \mathbf{b} = \begin{pmatrix} [-2,4] \\ [1,8] \\ [-4,10] \\ [2,12] \end{pmatrix}.$$

$$\mathrm{x1} = \begin{pmatrix} [-2,50, & 3,10] \\ [-3,90, & 1,50] \\ [-1,43, & 2,15] \\ [-2,35 & 0,60] \end{pmatrix},\ \mathrm{x2} = \begin{pmatrix} [-2,50, & 3,10] \\ [-3,90, & 1,20] \\ [-1,40, & 2,15] \\ [-2,35, & 0,60] \end{pmatrix},$$

$$\mathrm{x3} = \begin{pmatrix} [-2,2, & 3,10] \\ [-3,4, & 0,5] \\ [-1,3, & 1,44] \\ [-2,1, & 0,15] \end{pmatrix}.$$

Детерминанты средней, минимальной и максимальной матриц:

$|\mathrm{A}| = 2500$, $|\mathrm{A}^{-}| = 1418$, $|\mathrm{A}^{+}| = 4510$.

Решения x1, x2 известны из литературы [11] и получены методом исключения Гаусса и предобуславливанием [11], решение x3 получено по предложенной методике и близко к x1 и x2, но несколько уже.

Пример 5. Рассмотрим систему десятого порядка с относительными погрешностями элементов случайной матрицы и правой части $\varepsilon = 0,0005$, $\delta = 0,0005$ по отношению к mid(**A**) и mid(**b**).

Вариант 1. Решение исходной системы без применения предобуславливания (рис. 2).

Вариант 2. Решение исходной системы с применением предобуславливания (рис. 3).

На рис. 2 и 3 представлены средние решения и максимальные отклонения каждой компоненты вектора. С применением матрицы предобуславливания ширина интервального решения уже, чем без применения. Получено, что относительная погрешность решения на несколько порядков превосходит интервальную входную погрешность элементов. Отсюда следует прямая связь между интервальным объединенным решением и параметрической чувствительностью решения интервальной задачи.

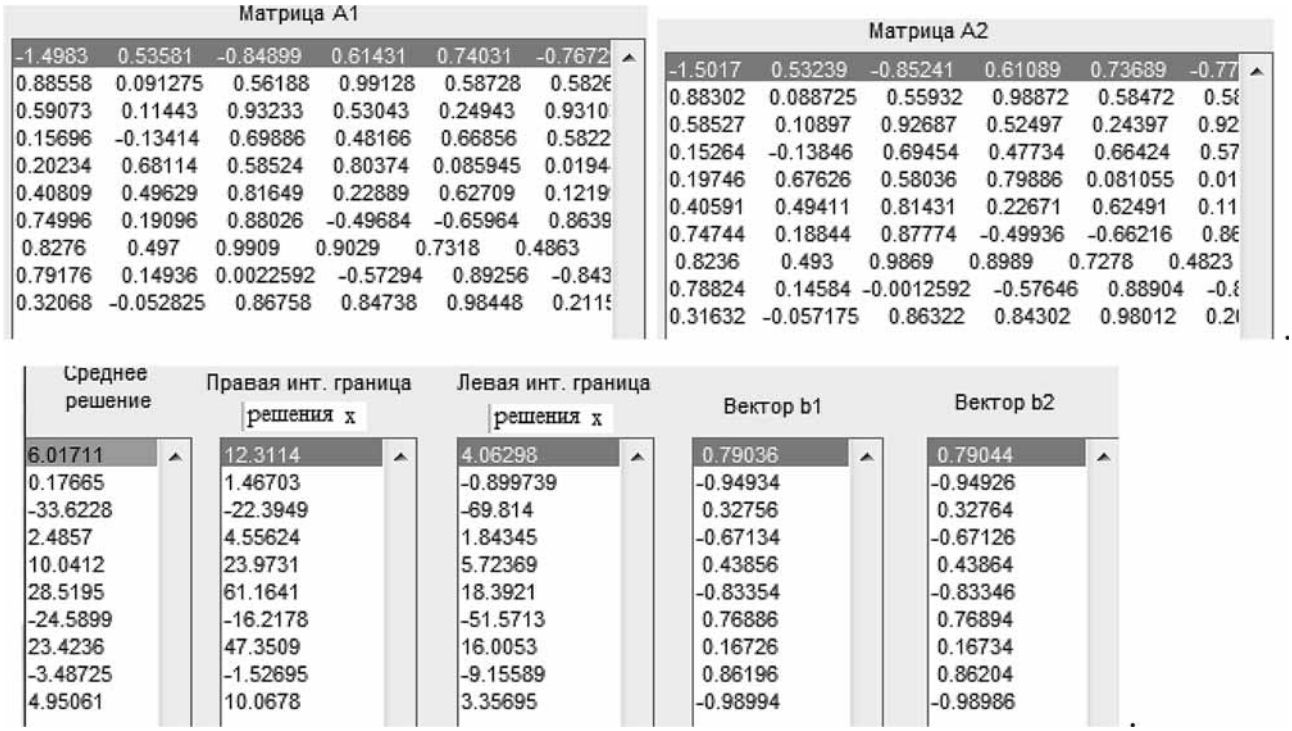


Рис. 2. Листинг программы решения примера 5 без приема предобуславливания:
$\det(A) = -0{,}0662$, $\det(A^{-}) = -0{,}1029$, $\det(A^{+}) = -0{,}0308$ — детерминанты угловых матриц

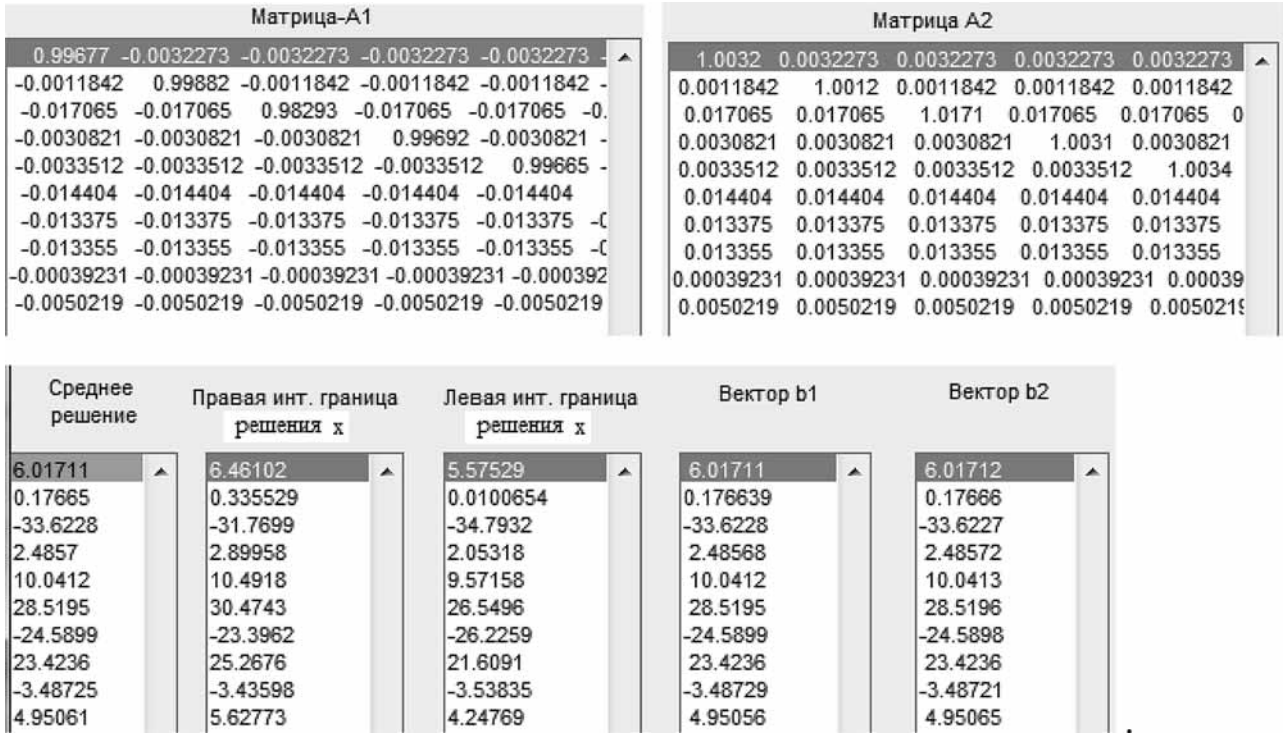


Рис. 3. Листинг программы решения примера 5 с предобуславливающей матрицей:
$\det(A) = 1$, $\det(A^{-}) = 0{,}9256$, $\det(A^{+}) = 1{,}0792$ — детерминанты угловых матриц с применением предобуславливания диагональной средней матрицей

Заключение

Предложена унифицированная методика оценивания внешнего вектора множества решений интервальных систем, имеющих узкий и широкий диапазоны интервального изменения элементов матриц, которая сводится к поиску координат бруса, соответствующего проекциям внешней оболочки множества решений системы на координатные оси.

Эффективность предложенной методики внешней оценки множества решений ИСЛАУ определяется полиномиальной сложностью.

Проведен анализ возможности реализации единого подхода к внешней оценке множества решений ИСЛАУ для матриц с узкими интервалами и для матриц с широкими интервалами для границ элементов, не превышающих 100 % относительной погрешности.

По алгоритмам, формализующим проведенные исследования, написана программа, интегрированная в среду MATLAB, с интерфейсом для ввода и вывода информации [12].

Результаты исследований получены при проведении научно-исследовательских работ по теме НИР "Структурное и фундаментально ориентированное моделирование политехнических наноструктур" по государственному контракту № 9.0.112.2009.

Список литературы

1. **Oettli W.** On the solution set of a linear system with inaccurate coefficients // SIAM J. Numer. Analysis. 1965. Vol. 2, N. 1. P. 115—118.

2. **Jansson C.** Calculation of exact bounds for the solution sets of linear interval systems // Linear Algebra Appl. 1997. Vol. 251. P. 321—340.

3. **Шарый С. П.** Оптимальное внешнее оценивание множества решений интервальных систем уравнений. Часть 2 // Вычислительные технологии. 2003. Т. 8, № 1. С. 84—109.

4. **Rohn J.** Systems of linear interval equations // Linear Algebra Appl. 1989. Vol. 126. P. 39—78.

5. **Shary S. P.** A new class of algorithms for optimal solution of interval linear systems // Interval Computations. 1992. N. 2 (4). P. 18—29.

6. **Цей Р., Шумафов М. М.** Число обусловленности матрицы как показатель устойчивости при решении прикладных задач // Труды ФОРА. 2011. № 16. С. 61—67.

7. **Петров Ю. П.** Как получать надежные решения систем уравнений СПб.: БХВ-СПб, 2009. 172 с.

8. **Шарый С. П.** Конечномерный интервальный анализ. Новосибирск: Изд-во ИВТ СОРАН, 2010. 601 с.

9. **Nirmala T**. Inverse Interval Matrix: A New Approach // Applied Mathematical Sciences. 2011. Vol. 5, N. 13. P. 607—624.

10. **Шарый С. П.** О сравнении теорем Апостолатоса—Кулиша и Майера—Варнке в интервальном анализе // Сибирский журнал вычислительной математики. 2009. Т. 12, № 3. С. 351—359.

11. **Ning S., Kearfott. R. B.** A comparison of some methods for solving linear interval equations // SIAM J. on Num. Anal. 1997. Vol. 34, N. 4. P. 1289—1305.

12. **Иванова К. Ф.** Программа внешней оценки интервальных систем линейных алгебраических уравнений в широком диапазоне погрешностей входных воздействий (BRUS). Свидетельство РФ о государственной регистрации программы для ЭВМ № 2015 618187. Дата регистрации 3 августа 2015 г.

Unification of Point Algebraic Techniques of an Exterior Estimation of Set Solutions Interval Systems

K. F. Ivanova, e-mail:Klara.I2010@yandex.ru, Saint-Petersburg State University, 195252**,** Saint Petersburg, Russian Federation

Corresponding author:

Ivanova Klara F., Doctorant, Saint-Petersburg State University, 195252**,** Saint Petersburg, Russian Federation
e-mail: Klara.I2010@yandex.ru

Received on December 05, 2015
Accepted on February 01, 2016

In the work a unified technique of exterior estimating the solution sets of interval systems of the linear algebraic equations (ISLAE) with uncertain parameters in both narrow and wide interval ranges of modification of elements for systems with M-and H-matrices and to their modifications is offered. The offered technique of estimation is based on choice of point elements of an interval matrix and of a component of right-hand side vector entering into component-wise products of elements of an interval matrix on their algebraic deviations. This choice is carried out based on extremalizating the received scalar product by identifying the point value of an element of the new matrix with one of the corresponding boundary values of an element of an interval matrix and a point right-hand side new vector with boundary of a corresponding component of an interval right-hand side vector. As a result of the performed operation for choosing minimal and universal elements and the component is formed by two point matrices and two point vectors from which it is possible to generate two systems consisting of four point linear algebraic equations. The formalized point matrices ensure the minimal and maximal values of the determinants, similar to two boundaries of an interval determinant of an initial matrix. The solution of the obtained point set of equations there are double-dimension vectors x that have the minimal and maximal deviations as their components.

Keywords: *interval system, uncertainty of parameters, extremalization, a scalar product, the formalized matrixes, a technique of estimate, set solutions, an exterior estimation, the point equations*

Acknowledgements: *This work was supported by research works on theme NIR "The structural fundamental focused modelling of polytechnical nanostructures" under the state contract № 9.0.112.2009.*

For citation:

Ivanova K. F. Unification of Point Algebraic Techniques of an Exterior Estimation of Set Solutions Interval Systems, *Programmnaya Ingeneria*, 2016, vol. 7, no. 4, pp. 181—190.

DOI: 10.17587/prin.7.181-190

References

1. **Oettli W.** On the solution set of a linear system with inaccurate coefficients, *SIAM J. Numer. Analysis,* 1965, vol. 2, no. 1, pp. 115—118.

2. **Jansson C.** Calculation of exact bounds for the solution sets of linear interval systems, *LinearAlgebra Appl.,* 1997, vol. 251, pp. 321—340.

3. **Shary S. P.** Optimal'noe vneshnee otsenivanie mnozhestva reshenii interval'nykh sistem uravnenii. Chast' 2 (Optimum exterior estimation of set solutions of interval sets of equations. Part 2), *Vychislitel'nye Tekhnologii,* 2003, vol. 8, no. 1, pp. 84—109. (in Russian)

4. **Rohn J.** Systems of linear interval equations, *Linear Algebra Appl*. 1989, vol. 126. pp. 39—78.

5. **Shary S. P.** A New class of algorithms for optimal solution of interval linear systems, *Interval Computations*, 1992, no. 2 (4), pp. 18—29.

6. **Tsej R., Shumafov M. M.** Chislo obuslovlennosti matritsy kak pokazatel' ustoichivosti pri reshenii prikladnykh zadach (A matrix Condition number as a stability indicator at a solution of applied problems), *Trudy FORA*, 2011, no. 16, pp. 61—67 (in Russian).

7. **Petrov Yu. P.** *Kak poluchat' nadezhnye resheniya sistem uravnenii* (How to receive reliable solutions of set equations), Saint-Petersburg, BHV-SPb, 2009, 172 p. (in Russian).

8. **Shary S. P.** *Konechnomernyi interval'nyi analiz* (The finite-dimensional interval analysis), Novosibirsk, Izd-vo IVT SORAN, 2010. 601 p. (in Russian).

9. **Nirmala T**. Inverse Interval Matrix: A New Approach, *Applied Mathematical Sciences*, 2011, vol. 5, no. 13, pp. 607—624.

10. **Shary S. P.** O sravnenii teorem Apostolatosa-Kulisha i Maiera—Varnke v interval'nom analize (About comparison of theorems of Apostolatosa-Kulisha and Mayer-Varnke in the interval analysis), *Sibirskii Zhurnal Vychislitel'noi Matematiki,* 2009, vol. 12, no. 3, pp. 351—359 (in Russian).

11. **Ning S., Kearfott R. B.** A comparison of some methods for solving linear interval equations, *SIAM J. on Num. Anal.,* 1997, vol. 34, no. 4. pp. 1289—1305.

12. **Ivanova K. F.** The program of an exterior estimation of interval systems of the linear algebraic equations in a wide range of errors of entering actions (BRUS). The testimony of the Russian Federation on the state registration of the computer program № 2015 618187. Date of registration August, 3rd 2015.

ИНФОРМАЦИЯ

IS&IT‘16

Официальный сайт конгресса
http://icai.tti.sfedu.ru/

Международный конгресс по интеллектуальным системам и информационным технологиям

2—9 сентября 2016 г.

Россия, Черноморское побережье, Геленджик-Дивноморское

Основные даты:

Прием заявок	до 01.05.16 г.
Прием текстов докладов	до 01.06.16 г.
Финальная версия	**до 15.07.16 г.**
Регистрация участников	с 02.09.16 г.
Начало заседаний	с 03.09.16 г.

Организаторы конгресса:

- Министерство образования и науки РФ
- Российская Академия наук
- Российская Академия естественных наук
- Южный федеральный университет
- НКБ «Миус» ЮФУ
- Администрация г. Таганрога;
- Российская ассоциация искусственного интеллекта (РАИИ);
- Университет Артуа (Франция)
- Университет Мюнстер (Германия)
- Университет Монс (Бельгия)

Тематика конгресса:

- Биоинформатика
- Интеллектуальные САПР, CASE-, CALS-технологии
- Искусственный интеллект и мягкие вычисления
- Представление и извлечение знаний
- Многоагентные системы и принятие решений
- Перспективные информационные технологии
- Проблемы образования
- Синергетика и моделирование сложных систем
- Эволюционное моделирование и генетические алгоритмы
- Экспертные системы
- Информационная безопасность
- SAP-технологии
- Инструментальные, математические и информационные средства экономики

Мероприятия, проводимые в рамках конгресса:

- Международная научно-техническая конференция **"Интеллектуальные системы и информационные технологии IS&IT'16"**
- Международная научно-техническая конференция **"Интеллектуальные САПР CAD-2016"**
- Молодежная научно-техническая конференция **"Информационные системы и технологии-2016"**
- Международный форум корпорации **EMC**
- Круглые столы по проблемам интеллектуальных систем и информационных технологий
- Ярмарка-выставка программных продуктов

Рабочие языки конгресса: русский и английский

Труды конгресса будут изданы к началу заседаний и включены в РИНЦ

Адрес оргкомитета:
IS&IT’16, проф. В. В. Курейчику, ЮФУ. 44, пер. Некрасовский, г. Таганрог, 347928, Российская Федерация.
E-mail: *vkur@tsure.ru*, *nev@tsure.ru*, *kur@tsure.ru*, *ivr@tsure.ru*
Телефоны: 8634-37-16-51, 8634-39-32-60, 8634-38-34-51 **Факс:** 8634-37-16-5

Институт прикладной математики им. М. В. Келдыша РАН проводит с 19 по 24 сентября 2016 г. XVIII Всероссийскую научную конференцию "Научный сервис в сети Интернет". Основная цель данной конференции — предоставить возможность для обсуждения, апробации и обмена мнениями о наиболее значимых результатах, полученных ведущими российскими учеными за последнее время в данной области деятельности. Конференция посвящена основным направлениям и тенденциям использования интернет-технологий в современных научных исследованиях.

Основные тематические направления конференции:

- Научные исследования и интернет, интернет-представительство научных организаций и проектов
- Интернет-проекты в области параллельных вычислений и распределенной обработки данных, вычислительные сервисы
- Удаленный доступ к научным исследовательским комплексам
- Математическое моделирование и интернет
- Технологии и системы распределенного хранения данных
- Модели и методы построения поисковых систем и систем навигации в интернете
- Технологии и опыт построения информационных систем и баз данных, документации и результатов эксперимента на основе интернет-технологий
- Цифровые библиотеки и библиографические базы, семантический веб, наукометрия в интернете
- Онлайновая научная публикация, открытая наука, живая публикация, онлайновое рецензирование, мультимедийные иллюстрации
- Популярный научный интернет, онлайновые энциклопедии, история науки в интернете
- Интернет-активность ученого, персональная страница, профили ученого в библиографических базах, аттестация в интернете

Конференция проводится в пансионате "Моряк", расположенном в 20 км от Новороссийска

Ключевые даты конференции:

1 июня — завершение регистрации и представления докладов
1 июля — уведомление о включении доклада в программу конференции
15 июля — представление окончательного варианта статьи
30 августа — оплата оргвзноса и проживания
15 сентября — завершение регистрации участников конференции

19 сентября — день заезда
20—23 сентября — рабочие дни конференции
24 сентября — день отъезда
Сайт конференции: http://agora.guru.ru/abrau2016

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор *З. В. Наумова*

Сдано в набор 05.02.2016 г. Подписано в печать 21.03.2016 г. Формат 60×88 1/8. Заказ PI416
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**