ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

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

СОДЕРЖАНИЕ

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

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

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

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 6
June
2014

Published since September 2010

CONTENTS

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

УДК 004.432

И. И. Левин[1], д-р техн. наук, проф., зам. директора по науке,
А. И. Дордопуло[2], канд. техн. наук, ст. науч. сотр.,
И. А. Каляев[1], чл.-корр. РАН, д-р техн. наук, проф., директор,
В. А. Гудков[1], канд. техн. наук, ст. науч. сотр., e-mail: lina@mvs.tsure.ru,
[1]НИИ многопроцессорных вычислительных систем имени академика А. В. Каляева Южного федерального университета, г. Таганрог,
[2]Южный научный центр Российской академии наук, г. Ростов-на-Дону

Высокопроизводительные реконфигурируемые вычислительные системы на основе ПЛИС Virtex-7*

Рассмотрены сравнительные характеристики реконфигурируемых вычислительных систем (РВС) на основе вычислительных модулей 24V7-750 и "Тайгета", содержащих программируемые логические интегральные схемы (ПЛИС) семейства Xilinx Virtex-7. Отличительными характеристиками РВС на основе ПЛИС Xilinx Virtex-7 по сравнению с аналогичными системами на основе ПЛИС Xilinx Virtex-6 являются увеличение производительности в 1,7 раза и улучшение остальных технико-экономических показателей: удельной производительности, энергоэффективности и др. Приведено решение прикладной задачи с помощью разработанного комплекса средств разработки прикладного программного обеспечения для РВС.

Ключевые слова: *реконфигурируемые вычислительные системы, вычислительный модуль, программируемые логические интегральные схемы, прикладное программное обеспечение*

I. I. Levin, A. I. Dordopulo, I. A. Kalyaev, V. A. Gudkov

High-Performance Reconfigurable Computer Systems on the Base of Virtex-7 FPGAs

In the paper we compare parameters of reconfigurable computer systems (RCS) based on computational modules 24V7-750 and "Taygeta", which contain Xilinx Virtex-7 FPGAs. The distinctive features of RCSs based on Xilinx Virtex-7 in comparison with similar systems based on Xilinx Virtex-6 FPGAs are higher performance (in 1,7 times) and better technical and economic parameters such as specific performance, power efficiency, etc. We also consider solution of an application task with the help of developed application development suit for RCS.

Keywords: *reconfigurable computer system, computational module, FPGA, application software*

Введение

Реконфигурируемые вычислительные системы (РВС) в таких областях, как символьная обработка, цифровая обработка сигналов, моделирование лекарственных препаратов и ряде других, по сравнению с многопроцессорными вычислительными системами кластерной архитектуры имеют ряд существенных преимуществ: высокие реальная и удельная производительности при решении задач, высокая энергоэффективность и др. В полной мере преимущества РВС достигаются при использовании в качестве основного вычислительного элемента аппаратного ресурса программируемых логических интегральных схем (ПЛИС) [1], объединенных в единое вычислительное поле высокоскоростными каналами передачи данных.

Методы разработки и создания таких систем успешно развивают в НИИ многопроцессорных вычислительных систем Южного федерального университета (г. Таганрог). Концепция построения РВС [2] позволила создать целый ряд высокопроизводительных

* Исследования выполнены при финансовой поддержке Министерства образования и науки РФ.

систем различных архитектур и конфигураций, выпускаемых серийно и предназначенных для решения вычислительно трудоемких задач различных предметных областей. Опыт успешной эксплуатации в организациях и ведомствах Российской Федерации различных конфигураций ранее созданных РВС на основе ПЛИС Xilinx семейств Virtex-4, Virtex-5 и Virtex-6 использовался для разработки перспективных РВС на основе ПЛИС Xilinx Virtex-7, описание которых представлено в настоящей статье.

РВС на основе ПЛИС Xilinx Virtex-7

РВС на основе ВМ "Плеяда". Реконфигурируемая вычислительная система РВС-7 на основе ПЛИС Virtex-7, разработанная по государственному контракту № 14.527.12.0004 от 03.10.2011, содержит вычислительное поле из 576 микросхем ПЛИС Virtex-7 XC7V585T-FFG1761 объемом 58 млн эквивалентных вентилей каждая, конструктивно объединенных в один вычислительный шкаф высотой 47U[1] с пиковой производительностью 10^{15} операций с фиксированной запятой в секунду.

Основным структурным компонентом РВС-7, предназначенным для установки в стандартную 19" вычислительную стойку, является вычислительный модуль (ВМ) 24V7-750 (ВМ "Плеяда"), в состав которого входят четыре платы вычислительного модуля (ПВМ) 6V7-180, представленные на рис. 1 (см. вторую сторону обложки); управляющий модуль УМ-7; подсистема питания; подсистема охлаждения и другие подсистемы. Внешний вид ВМ 24V7-750 представлен на рис. 2 (см. вторую сторону обложки).

В состав ПВМ 6V7-180 входят:

- вычислительное поле, состоящее из шести ПЛИС XC7V585T-1FFG1761 семейства Virtex-7 производства фирмы Xilinx. Между собой ПЛИС вычислительного поля соединены последовательно, передача данных осуществляется по 144 дифференциальным линиям LVDS-интерфейса на частоте 800 МГц;
- контроллер ПВМ, выполненный на ПЛИС XC6V130T-1FFG1156C производства фирмы Xilinx;
- 12 каналов интерфейса LVDS на частоте 800 МГц по 25 дифференциальных пар каждый (разъемы типа SS4) для связи с другими ВМ;
- узлы основной и резервной загрузки ПЛИС по интерфейсам JTAG-1 и JTAG-2;
- подсистема синхронизации (генераторы ECS-2033-250-BN и распределители тактовых импульсов IDT5T9316NLI);
- распределенная память в составе 12 микросхем динамической памяти (MT47H128M16HR-25E с организацией 128 M*16 и частотой записи/чтения до 400 МГц); к ПЛИС вычислительного поля, а также к ПЛИС контроллера базового модуля подключено по две микросхемы памяти DDR2; объем оперативной памяти на ПВМ — 3 Гбайта;
- два канала интерфейса LVDS по 20 дифференциальных пар для связи с персональным компьютером и внешней аппаратурой;
- подсистема загрузки ПЛИС;
- подсистема питания, в состав которой входят DC-DC-преобразователи напряжения, вырабатывающие следующие напряжения питания: +1 В — питание ядер ПЛИС; +2,5 В — питание узла тактирования; +1,8 В — питание микросхем памяти DDR2, +3,3 В — питание буферных каскадов ПЛИС.

Производительность одной ПВМ 6V7-180 составляет 645,9 Гфлопс при обработке 32-разрядных данных с плавающей запятой на частоте 400 МГц, а производительность ВМ 24V7-750 — 2,58 Тфлопс при обработке 32-разрядных данных с плавающей запятой (для оценки производительности РВС, обобщенной по классам решаемых задач, здесь и далее используется значение, равное произведению числа вычислительных устройств с плавающей запятой, которые можно одновременно разместить в вычислительном ресурсе РВС, и частоты работы этих устройств). Производительность РВС-7 при комплектации от 24 до 36 ВМ 24V7-750 составит 62...93 Тфлопс при обработке 32-разрядных данных с плавающей запятой и 19,4...29,4 Тфлопс при обработке 64-разрядных данных с плавающей запятой. Областью применения РВС-7 и вычислительных комплексов на ее основе, согласно техническому заданию на разработку, является решение задач цифровой обработки сигналов и многоканальная цифровая фильтрация.

РВС на основе ВМ "Тайгета". На основе ПЛИС Virtex-7 также разработан новый вычислительный модуль "Тайгета" в конструктивном исполнении высотой 2U, предназначенный для установки в стандартную 19" вычислительную стойку. Вычислительный модуль "Тайгета", представленный на рис. 3, *а* (см. вторую сторону обложки), содержит четыре ПВМ 8V7-200 (рис. 3, *б*, см. вторую сторону обложки), соединенных быстрыми LVDS-каналами, встроенную управляющую ЭВМ, систему питания, систему управления, систему охлаждения и другие подсистемы. ПВМ 8V7-200, лежащая в основе ВМ "Тайгета", представляет собой 20-слойную печатную плату с двухсторонним монтажом элементов, на которой располагаются 8 ПЛИС типа XC7VX485T-1FFG1761, содержащих 48,5 млн эквивалентных вентилей, 16 микросхем распределенной памяти SDRAM типа DDR2 общим объемом 2 Гбайта, интерфейсы LVDS и Ethernet и другие компоненты.

Производительность одной ПВМ 8V7-200 составляет 667 Гфлопс при обработке 32-разрядных данных с плавающей запятой, а производительность ВМ "Тайгета" — 2,66 Тфлопс при обработке 32-разрядных данных с плавающей запятой. Производительность РВС на основе ВМ "Тайгета" при комплектации от 18 ВМ "Тайгета" составляет 48 Тфлопс при обработке 32-разрядных данных с плавающей запятой и 23 Тфлопс при обработке 64-разрядных данных с плавающей запятой.

[1] U; от англ. Unit, равен 44,45 мм (или 1,75 дюйма).

РВС на основе ВМ "Тайгета" позволяет сократить стоимость поставки вычислительной системы для задач определенного класса (например, для задач символьной обработки), обеспечивая при этом такую же производительность, как и РВС-7 с 24 ВМ 24V7-750. Обладая более высоким по сравнению с РВС-7 значением показателя "производительность/стоимость" (за счет меньшей стоимости кристалла ПЛИС XC7VX485T по сравнению с кристаллами XC7V585T РВС-7) и сбалансированным числом внешних и межмодульных связей, ВМ "Тайгета" являются предпочтительными для построения высокопроизводительных вычислительных комплексов для решения задач символьной обработки данных, поскольку обеспечивают существенное конкурентное преимущество по большинству технико-экономических параметров — удельной производительности, энергоэффективности и др. по сравнению с РВС-7.

Программное обеспечение РВС на основе ПЛИС Xilinx Virtex-7

Программирование РВС, как правило, осуществляется в два этапа: на первом этапе создается вычислительная структура для решения прикладной задачи, а на втором этапе прикладной программист создает параллельную программу, управляющую потоками данных в созданной вычислительной структуре. Большинство существующих коммерческих систем проектирования (Xilinx ISE, Altium Designer и др.) обеспечивают в рамках одного проекта работу только с одним кристаллом ПЛИС. Поэтому при разработке конфигурации для нескольких ПЛИС инженеру-схемотехнику приходится самому распределять элементы вычислительной структуры алгоритма решаемой задачи между различными проектами, которые будут соответствовать определенным кристаллам ПЛИС многокристальной РВС, и учитывать топологию связей между кристаллами ПЛИС. Необходимость учета особенностей внутренней архитектуры, топологии и элементной базы РВС существенно усложняет специалисту-схемотехнику разработку конфигураций вычислительной системы прикладной задачи для многокристальных РВС и практически исключает возможность переноса (портации) готового решения на РВС другой конфигурации или архитектуры. Поэтому сроки разработки прикладных решений для РВС достаточно велики и составляют 4...9 месяцев.

Для программирования РВС в НИИ многопроцессорных вычислительных систем Южного федерального университета используют разработанный комплекс программного обеспечения [1, 2, 4, 6], поддерживающий структурно-процедурные методы организации вычислений и определяющий как структуру вычислительной системы в поле логических ячеек ПЛИС, так и организацию параллельных процессов и потоков данных. Для РВС на основе ПЛИС Virtex 7 преемственность принципов программирования [1, 2, 6] сохранена: программирование осуществляется на языке высокого уровня COLAMO [1, 6], в результате трансляции с которого автоматически формируются конфигурация вычислительной системы в виде файлов конфигурации кристаллов ПЛИС (структурная составляющая параллельной программы) и параллельная программа, управляющая потоками данных и организацией вычислительного процесса в РВС. Отличительными особенностями комплекса программного обеспечения на основе языка программирования COLAMO по сравнению с известными средствами разработки MitrionC [7] и CatapultC [8] являются автоматическое размещение, синхронизация и создание конфигурации для многокристальных РВС, высокий процент заполнения кристалла (60...90 %) и высокие частоты работы (250...350 МГц).

Комплекс программного обеспечения на основе языка программирования COLAMO содержит следующие основные компоненты:

- транслятор языка программирования COLAMO, осуществляющий трансляцию исходного кода на COLAMO в информационный граф параллельной прикладной программы;
- синтезатор масштабируемых схемотехнических решений Fire!Constructor, осуществляющий отображение полученного от транслятора языка программирования COLAMO информационного графа на архитектуру РВС, размещение отображенного решения по кристаллам ПЛИС и автоматическую синхронизацию фрагментов информационного графа в разных кристаллах ПЛИС на уровне логических ячеек ПЛИС;
- библиотеку IP-ядер, соответствующих операторам языка COLAMO (функционально-законченных структурно-реализованных аппаратных устройств), для различных предметных областей и интерфейсов для согласования скорости обработки информации и связи в единую вычислительную структуру.

Для каждого класса задач, решаемых на РВС, можно подобрать определенный набор оптимальных вычислительных структур (макрообъектов), наиболее эффективно решающих задачи данного класса. Поддержка проблемно-ориентированных софт-архитектур, позволяющих создавать и программировать макрообъекты, является отличительной особенностью комплекса программного обеспечения РВС-7.

Макрообъект — это совокупность вычислительных устройств, выполняющих определенную группу команд и соединенных между собой коммутационной системой. Для макрообъекта допустимо изменение числа функциональных вычислительных устройств и их параметров (разрядности операндов, числа информационных каналов, системы команд и т. п.), но недопустимо изменение их назначения. Таким образом, макрообъект с точки зрения прикладного программиста является "заготовкой", которая может доопределяться им при создании конкретного технического решения, а затем тиражироваться в нужном количестве в ПЛИС вычислительных модулей и соединяться с подобными или другими макрообъектами в вычислительные структуры, которые оптимально соответствуют структуре решаемой задачи. На основе макрообъектов возможно создание "софт-архитектуры РВС",

под которой понимается созданная схемотехником вычислительная структура, содержащая макрообъекты, в которой можно без перезагрузки файлов конфигурации ПЛИС вычислительного поля с помощью программной настройки изменять коммутацию между устройствами и создавать необходимые вычислительные структуры для решения прикладных задач пользователя.

Это обеспечивает, при сохранении преемственности принципов программирования и использования языка высокого уровня для программирования РВС, возможность простой адаптации программных компонентов средств разработки для РВС при переходе на новые топологии ПВМ без внесения существенных изменений в код программных компонентов комплекса, а также позволяет сократить время решения прикладных задач.

Программирование софт-архитектур для предметных областей

Для создания софт-архитектур в НИИ многопроцессорных вычислительных систем Южного федерального университета разработан язык SADL (*Soft-Architecture Development Language*) [9]. Описание на этом языке транслируется для синтезатора Fire!Constructor, формирующего виртуальную архитектуру вычислительной системы, на которую отображается информационный граф прикладной задачи. Для создания софт-архитектуры выполняют:

- разработку описания софт-архитектуры на языке SADL;
- трансляцию описания софт-архитектуры в промежуточное представление при помощи синтезатора конфигураций параллельно-конвейерных вычислительных структур Fire!Constructor;
- размещение элементов софт-архитектуры на аппаратной платформе при помощи синтезатора масштабируемых параллельно-конвейерных процедур Steam!Constructor.

Транслятор языка SADL преобразует текст программы в промежуточное представление, используемое синтезатором Fire!Constructor для размещения на аппаратной платформе РВС. Результатом размещения софт-архитектуры на аппаратную платформу являются модифицированный файл промежуточного представления и конфигурационные файлы для ПЛИС, участвующих в размещении софт-архитектуры на аппаратной платформе РВС. После того как софт-архитектура была размещена на аппаратной платформе РВС, она может быть использована для решения различных прикладных задач заданной проблемной области.

Основными этапами разработки прикладной программы для софт-архитектуры РВС являются:

- разработка параллельной программы на языке высокого уровня COLAMO;
- трансляция параллельной программы, преобразование информационного графа в структурный и процедурный компоненты;
- отображение структурного компонента параллельной программы на софт-архитектуру при помощи синтезатора конфигураций параллельно-конвейерных вычислительных структур Steam!Constructor;
- трансляция процедурного компонента параллельной программы на уровень команд устройств софт-архитектуры;
- формирование загрузочного out-файла, содержащего команды элементов софт-архитектуры;
- загрузка конфигурационных файлов ПЛИС, полученных в результате размещения элементов софт-архитектуры на аппаратной платформе реконфигурируемой системы;
- загрузка out-файла;
- загрузка в софт-архитектуру исходных данных решаемой задачи;
- запуск программы на исполнение и выгрузку результатов.

Взаимодействие средств разработки прикладных программ при разработке софт-архитектуры показано на рис. 4.

Рис. 4. Взаимодействие средств разработки прикладных программ

Благодаря разработанным программным средствам разработка и модификация софт-архитектур не требует привлечения высококвалифицированного специалиста-схемотехника для изменения вычислительной структуры. При этом время, затрачиваемое на создание или модификацию софт-архитектуры, значительно сокращается, а получаемые многокристальные архитектурные решения сравнимы по эффективности с решениями, выполненными специалистами-схемотехниками вручную. В соответствии с основными принципами языка COLAMO параллельные прикладные программы могут быть легко модифицированы для адаптации к доступному вычислительному ресурсу. Автоматизация отображения графов на ресурс РВС позволяет разработчикам прикладных задач мыслить не несколькими ПЛИС, а одной виртуальной ПЛИС с большим логическим объемом. Комплекс разработанных программных средств позволяет программисту РВС разрабатывать и выполнять отладку прикладных параллельных программ для РВС без детального знания архитектуры РВС, а также самостоятельно создавать и модифицировать различные софт-архитектуры, ориентируясь на предметную область, которой принадлежит решаемая задача.

Заключение

Конструктивные решения, положенные в основу перспективных вычислительных модулей на ПЛИС Xilinx Virtex-7, позволяют сосредоточить в пределах одной вычислительной стойки высотой 47U мощный вычислительный ресурс и обеспечивают высокие значения таких технико-экономических параметров, как удельная производительность и энергоэффективность, показатель "производительность/стоимость" РВС на уровне лучших мировых показателей для суперЭВМ с кластерной архитектурой. Разработанный и успешно используемый комплекс программного обеспечения на основе языка программирования COLAMO в отличие от известных высокоуровневых средств разработки MitrionC и CatapultC поддерживает автоматическое размещение, синхронизацию и создание конфигурации ПЛИС для многокристальных РВС, обеспечивая высокие частоты работы 250...400 МГц при высокой плотности заполнения кристалла (не менее 60...90 %).

Это позволяет рассматривать РВС на основе ПЛИС Xilinx Virtex-7 как основание для создания высокопроизводительных вычислительных комплексов нового поколения, обеспечивающих высокую эффективность вычислений и близкий к линейному рост производительности при наращивании вычислительного ресурса.

Список литературы

1. **Каляев А. В., Левин И. И.** Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. М.: Янус-К, 2003. 380 с.
2. **Каляев И. А., Левин И. И., Семерников Е. А., Шмойлов В. И.** Реконфигурируемые мультиконвейерные вычислительные структуры. Изд. 2-е, перераб. и доп. / Под общ. ред. И. А. Каляева. Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. 344 с.
3. **Каляев И. А., Левин И. И.** Семейство реконфигурируемых вычислительных системы с высокой реальной производительностью // Тр. Междунар. науч. конф. "Параллельные вычислительные технологии 2009" (ПАВТ'2009). Нижний Новгород: НГУ имени Н. И. Лобачевского [Электронный ресурс]. 2009. С. 186—196.
4. **Дордопуло А. И., Каляев И. А., Левин И. И., Семерников Е. А.** Высокопроизводительные реконфигурируемые вычислительные системы нового поколения // Тр. Междунар. суперкомпьютерной конф. с элементами научной школы для молодежи "Научный сервис в сети Интернет: экзафлопсное будущее". М.: Изд-во МГУ, 2011. С. 42—49.
5. **Каляев И. А., Левин И. И., Семерников Е. А., Дордопуло А. И.** Реконфигурируемые вычислительные системы на основе ПЛИС семейства Virtex-6 // Сборник тр. Междунар. науч. конф. "Параллельные вычислительные технологии 2011" (ПАВТ, 2011). Челябинск: Издательский центр ЮУрГУ [Электронный ресурс], 2011. С. 203—210.
6. **Kalyaev I. A., Levin I. I., Semernikov E. A., Shmoilov V. I.** Reconfigurable multipipeline computing structures. N.-Y.: Nova Science Publishers, 2012.
7. http://www.mitrionics.com
8. http://calypto.com/en/products/catapult/overview/
9. **Семерников Е. А., Коваленко В. Б.** Организация многоуровневого программирования реконфигурируемых вычислительных систем // Вестник компьютерных и информационных технологий. 2011. № 9. С. 3—10.

ИНФОРМАЦИЯ

УДК 004.4'418

Н. И. Вьюкова, ст. науч. сотр., e-mail: niva@niisi.msk.ru,
В. А. Галатенко, д-р физ.-мат. наук, ст. науч. сотр., e-mail: galat@niisi.msk.ru,
С. В. Самборский, ст. науч. сотр., НИИ системных исследований РАН, г. Москва

Генерация кода методом точного совместного решения задач выбора и планирования команд

Представлен метод генерации кода для линейных участков программ путем точного совместного решения задач выбора и планирования инструкций с учетом ограничений по числу регистров. Преимущества предлагаемого метода — максимальный учет параллелизма исполнения при выборе команд, генерация оптимального кода откачки при дефиците регистров, автоматическое использование команд, имеющих несколько результатов. Рассмотрена экспериментальная реализация генератора кода, а также возможности развития данного подхода.

Ключевые слова: *оптимизация кода, выбор команд, планирование команд, целочисленное линейное программирование*

N. I. Vyukova, V. A. Galatenko, S. V. Samborskij

Code Generation by Exact Joint Solution of the Instruction Selection and Scheduling Tasks

The paper presents a code generation method based on the exact joint solution of instruction selection and scheduling tasks. The benefits of the method proposed are the maximal account for parallel execution during instruction selection, generation of the optimal spill code, automated use of instructions with multiple results. An experimental implementation of the code generator and possible directions of evolving the approach are discussed.

Keywords: *code optimization, instruction selection, instruction scheduling, integer linear programming*

Введение

Данная работа продолжает исследования, представленные в работе [1], в области совместного решения задач выбора и планирования команд при генерации кода в компиляторах для процессоров, используемых во встроенных системах. Как правило, в архитектуре таких процессоров отсутствуют аппаратные средства оптимизации, такие как аппаратное планирование с переупорядочением команд, ротация регистрового файла, переименование регистров и др. Необходимость такого рода упрощений архитектуры обусловлена ограничениями по энергопотреблению и размеру процессора.

Предполагается, что необходимый уровень эффективности приложений для встроенных систем должен достигаться за счет высокого качества оптимизации кода, обеспечиваемого компилятором (либо программистом, разрабатывающим на ассемблере критические участки приложений). При генерации кода необходимо в максимальной степени использовать возможности системы команд, в частности, наличие специализированных команд, а также аппаратные средства параллельного исполнения команд.

Процессор рассматриваемого класса, например, может иметь два (или более) сопроцессора, наборы команд которых могут включать похожие операции. Соответственно, при генерации кода необходимо обеспечить оптимальное использование обоих сопроцессоров, учитывая стоимости пересылок значений между регистрами сопроцессоров, а также возможности параллельного исполнения команд.

Традиционные методы генерации кода, в которых выбор команд, их планирование (составление расписания) и распределение регистров осуществляются раздельно, не позволяют решать подобные задачи, требующие одновременного учета таких факторов, как возможность параллельного исполнения команд, латентность, дефицит регистров, стоимость пересылок, зависимости по данным. Для их решения был

предложен метод генерации кода с отложенным выбором команд [2], при котором выбор команд откладывается до стадии планирования.

Идея метода заключается в том, чтобы представить задачу выбора и планирования команд в виде задачи математического программирования. Для этого применяют методы целочисленного линейного программирования (ЦЛП). Подобные подходы требуют высоких затрат вычислительных ресурсов и значительно увеличивают время компиляции, однако они могут являться оправданной альтернативой ассемблерному кодированию, которое нередко применяется при создании приложений для встроенных систем.

Результаты, представленные в работе [1], показывают, что возможности предложенного метода на самом деле шире, чем предполагалось изначально. В частности, он позволяет автоматически генерировать и планировать код откачки и восстановления регистров (либо перевычисления значений) в ситуации дефицита регистров. Возможны также расширения этого метода, реализующие ряд других оптимизаций, например, оптимизации цепочек ассоциативно-коммутативных операций, оптимизации, основанные на тождественных преобразованиях, различные виды спиллинга в зависимости от диапазона значений и т. д.

Для того чтобы упростить эксперименты, связанные с развитием метода генерации кода с отложенным выбором команд, был реализован прототип кодогенератора ISched (*Instruction selection & Scheduling*). Генератор кода ISched позволяет по описанию системы команд и входного линейного участка сгенерировать формулировки серии ЦЛП-задач, реализующих генерацию кода, запустить один из имеющихся солверов ЦЛП-задач и распечатать сгенерированный код в виде расписания, т. е. в виде последовательности команд с указанием номеров тактов, на которых они были запущены.

Дальнейшее содержание статьи построено по следующему плану. Разд. 1 содержит пример описания системы команд гипотетического процессора и входного линейного участка, представляющего вычисление корней квадратного уравнения. В разд. 2 кратко изложена последовательность действий при генерации кода. В разд. 3 обсуждены примеры генерации кода для данных, описанных в разд. 1. В заключение просуммирован опыт, приобретенный авторами в ходе выполнения работы, и рассмотрены перспективы дальнейших исследований.

1. Пример входных данных для генератора кода ISched

Генератор кода ISched использует модификацию известного алгоритма выбора команд BURG [3], принадлежащего к семейству декларативных методов выбора команд, обзор которых можно найти в работе [4]. Вследствие этого входные данные для ISched имеют ту же структуру, что и входные данные для методов BURG. Они включают описание грамматики, задающей систему команд целевого процессора, и набор деревьев, представляющий входной линейный участок. Эти два компонента будут описаны в последующих подразделах.

Синтаксис входных данных для ISched обусловлен тем, что эти данные представляют собой программу на языке Python [5]. Каждый элемент описания является присваиванием, правая часть которого является литеральным значением одной из структур данных языка Python: список, словарь и т. д. Символ # является началом комментария, который продолжается до конца строки.

1.1. Описание системы команд

Описание целевого процессора для генератора кода ISched состоит из следующих пунктов:

- ресурсы (вычислительные устройства, регистры) процессора;
- типы команд процессора, различающиеся по набору потребляемых ресурсов;
- нетерминальные символы грамматики, задающие различные типы хранилищ данных (регистры, память);
- терминальные символы грамматики, задающие набор поддерживаемых операций;
- правила, описывающие команды процессора или оптимизации.

В статье приведен пример простой системы команд, достаточной для решения задачи вычисления корней квадратного уравнения.

Ресурсы. Под ресурсами понимают вычислительные устройства и регистры процессора:

```
Res = {"iss":2, "add":2, "mul":1,
       "divsqrt":1, "div":1, "dreg":32,"cc":1,
      }
```

Процессор содержит два устройства выдачи команд на исполнение `iss`, два устройства сложения `add`, устройство умножения `mul`, устройство деления и извлечения квадратного корня `divsqrt`, дополнительное устройство деления `div`, 32 регистра общего назначения `dreg`. Символ `cc` соответствует регистру кода условия (*condition code*).

Типы команд. Перечислим имена типов команд с описанием таблиц резервирования ресурсов:

```
Ctypes = {"c_empty":[],
          "c_simple":[("iss",0)],
          "c_div1":[("iss",0),
                    ("divsqrt",0,1,2,3,4],
          "c_div2":[("iss",0),
                    ("div",0,1,2,3)],
          "c_sqrt":[("iss",0),
                    ("divsqrt",0,1,2,3,4,5,6)],
          "c_mem":[("iss",0)],
          "c_add":[("iss",0), ("add",0)],
          "c_mul":[("iss",0), ("mul",0,1)],
          "c_madd":[("iss",0), ("mul",0,1),
                    ("add",2)],
          "c_single":[("iss",0), ("iss",0)]
}
```

Информация этого пункта определяет ограничения на параллельное исполнение команд. Здесь описаны десять типов команд; все они используют на нулевом такте устройство выдачи команд на исполнение iss:

- простые команды c_simple, занимающие только устройство выдачи на исполнение;
- два типа команд деления, исполняющихся на разных устройствах; c_div1 занимает устройство divsqrt на тактах с 0 по 4, c_div2 занимает устройство div на тактах с 0 по 3;
- команда извлечения квадратного корня c_sqrt, использующая устройство divsqrt на тактах с 0 по 6;
- команды доступа к памяти c_mem;
- команды сложения/вычитания c_add;
- команды умножения c_mul;
- команды умножения с накоплением или вычитанием c_madd;
- команды c_single, занимающие оба устройства выдачи на исполнение.

Нетерминальные символы описывают различные хранилища значений, с которыми могут работать команды процессора. После каждого символа в квадратных скобках задается список соответствующих ему ресурсов:

```
Nonterm = {"D":[("dreg",1)],
           "CC":[("cc",1), "M":[]]}
```

В данном примере описано три типа хранилищ данных: D — регистры общего назначения, занимающие одну единицу ресурса dreg; CC — регистр кода условия, занимает ресурс cc; M — память; память может использоваться без ограничений.

Если регистры процессора 32-битные, но система команд включает команды, работающие с 64-битными значениями, занимающими пару соседних регистров, то хранилища 64-битных значений можно было бы описать при помощи конструкции "D64":[("reg",2)].

Терминальные символы. Терминальные символы определяют набор операций, которые могут использоваться в описании входного линейного участка. Для каждого символа через двоеточие задается "арность" соответствующей операции:

```
Term = {"add":2, "sub":2, "mul":2, "neg":1,
        "div":2, "sqrt":1,
        "if":4, "cond":1, "var":1, "const":1,
        "look":1, "out":2}
```

Символы в этом примере описывают следующие операции: add — сложение; sub — вычитание; mul — умножение; neg — смена знака; div — деление; sqrt — извлечение квадратного корня. Операция if означает условное действие, в зависимости от текущего значения кода условия. Операция cond устанавливает код условия как результат сравнения своего аргумента с нулем. Операция var соответствует считыванию из памяти значения переменной. Операция const описывает загрузку константы из памяти.

Символы look, out служебные: look используется для ссылки на одно из деревьев входного линейного участка, out задает выходные значения линейного участка.

Правила. Рассмотрим структуру описания правил на примере команды сложения:

```
Rule = (
# 1. Сложение
#    Имя правила и тип команды
  ("ADD", "c_add",
#    Описания операндов
{ "D0":(Out,"D",2),
  "C":(Out,"CC",2),
  "D1":(In,"D",0),
  "D2":(In,"D",0)},
#    Шаблоны сопоставления
[(("D0",), ("add","D1","D2")),
 (("C",), ("cond", ("add","D1","D2")))],
#    Формат печати ассемблерной команды
("add ","D0","D1","D2"),
),
```

В описании правила выделяют четыре раздела. В первом содержится имя команды (ADD) и ее тип (c_add). Второй раздел содержит описатели операндов. В данном случае команда имеет два входных операнда (D1, D2) и два выходных (D0, C), имеющих латентность записи два такта. Операнды D0, D1, D2 являются регистрами общего назначения (D), а C — код условия (CC). В третьем разделе для каждого выходного операнда задается шаблон сопоставления — древовидная конструкция. Если в дереве, описывающем входной линейный участок, имеется конструкция такого вида, то соответствующее вычисление может быть реализовано при помощи данной команды. Наконец, последний раздел задает шаблон для печати команды при выдаче результата или в отладочных выдачах.

В таблице приведены команды, которые использованы в рассматриваемом примере, с кратким описанием характеристик. В столбце **CC** показано наличие кода условия. Столбец **L** задает латентности.

В пп. 4—7 второе значение (2) в столбце **L** соответствует латентности записи кода условия. В других случаях латентности записи результата и кода условия совпадают.

В пп. 12, 13 заданы правила оптимизации для умножения на 2, которое заменяется более быстрой операцией сложения. При этом также экономится команда загрузки константы 2 на регистр. Два правила введены потому, что ISched не учитывает коммутативность операций при сопоставлении шаблонов. По этой же причине введены два правила 5, 6.

В пп. 14, 15 описаны команды выгрузки регистра в память и его восстановление. Наличие этих правил обеспечивает автоматическую генерацию команд спиллинга при дефиците регистров.

Характеристики команд

№ пп	Вычисление	Команда	CC	L	Тип	Комментарий
1	D0=D1+D2	**add** D0,D1,D2	+	2	c_add	Сложение
2	D0=D1-D2	**sub** D0,D1,D2	+	2	c_add	Вычитание
3	D0=D0+D1 D1=D0-D1	**addsub** D0,D1	–	2	c_add	Сложение и вычитание
4	D0=D1*D2	**mul** D0,D1,D2	+	3,2	c_mul	Умножение
5	D0=D0+(D1*D2)	**madd** D0,D1,D2	+	3,2	c_madd	Умножение с накоплением
6	D0=(D1*D2)+D0	**madd** D0,D1,D2	+	3,2	c_madd	
7	D0=D0-(D1*D2)	**msub** D0,D1,D2	+	3,2	c_madd	Умножение с вычитанием
8	D0=-D1	**neg** D0,D1	+	2	c_add	Смена знака
9	D0=D1/D2	**div1** D0,D1,D2	–	5	c_div1	Деление на устройстве divsqrt
10	D0=D1/D2	**div2** D0,D1,D2	–	4	c_div2	Деление на устройстве div
11	D0=sqrt(D1)	**sqrt** D0,D1	–	7	c_sqrt	Извлечение квадратного корня
12	D0=2*D1	**add** D0,D1,D1	+	2	c_add	Оптимизация умножения на 2
13	D0=D1*2	**add** D0,D1,D1	+	2	c_add	
14	M=D	**save** D,M	–	1	c_mem	Команды сохранения и восстановления регистра
15	D=M	**restore** D,M	–	2	c_mem	
16	D=L	**load** D,L	–	2	c_mem	Чтение переменной из памяти в регистр
17	D=C	**load** D,C	–	2	c_mem	Чтение константы из памяти в регистр
18	D0=D1	**move** D0,D1	–	2	c_simple	Копирование регистра в другой регистр
19	C=cond(D0)	**test** D0	+	1	c_simple	Выработка кода условия
20	D0=CC	**move_cc** D0	–	2	c_simple	Копирование кода условия в регистр
21	D0=if*U*(C,D2,D3)	**if***U* D0,D2,D3	–	2	c_simple	Условное копирование
22	D0=if*U*(D1,D2,D3)	**if***U* D0,D1,D2,D3	–	2	c_simple	Условное копирование по CC, сохраненному в D1

В пп. 21, 22 описаны команды условного копирования. Выходной регистр D0 получает значение D2 или D3 в зависимости от значения кода условия. Символ U обозначает одну из операций сравнения (GT, GE, LT, LE, EQ, NE ...). В пп. 21 код условия находится на регистре CC, а в пп. 22 — на регистре общего назначения (D1).

1.2. Описание входного линейного участка

Входной линейный участок задается как список деревьев, описывающих вычисления, для которых необходимо сгенерировать код. Деревья задаются в списочной ЛИСП-подобной форме. Узлами деревьев могут быть только терминальные символы грамматики, описывающей систему команд.

Рассмотрим линейный участок, реализующий вычисление корней квадратного уравнения. Вычисление по стандартной формуле $\frac{-b \pm \sqrt{b^2-4ac}}{2a}$ может приводить к потере точности, если $4ac$ мало по абсолютной величине по сравнению с b. В данном примере применяется следующий метод вычисления (предполагается, что $D \geqslant 0$):

$$D = b^2 - 4ac$$

$$y_1 = \sqrt{D} + b$$

$$y_2 = \sqrt{D} - b$$

```
Test = (

  ("sub", ("mul",("var","b"),("var","b")),
                  ("mul",("const",4),
                         ("mul",("var","a"),("var","c")))),    # 0 D

  ("add", ("sqrt",("look",0)), ("var","b")),                   # 1 y1

  ("sub", ("sqrt",("look",0)), ("var","b")),                   # 2 y2

  ("if","GE",("cond",("var","b")),("look",1),("look",2)),      # 3 y

  ("out", ("D","M"), ("div",("neg",("look",3)),
                         ("mul",("const",2),("var", "a")))),   # 4 x1
  ("out", ("D","M"), ("div",("mul",("const",2),("var", "c")),
                         ("neg",("look",3)))),                 # 5 x2
)
```

Рис. 1

$$y = \text{if } (b \geqslant 0)\ y_1 \text{ else } y_2$$

$$x_1 = (-y)/2a$$

$$x_2 = 2c/(-y).$$

Формула для x_2 получается из стандартной формулы домножением числителя и знаменателя на y_1 или y_2 в зависимости от знака b. На рис. 1 представлено описание этого алгоритма на языке генератора кода ISched.

На рис. 1 справа в качестве комментариев указаны номера деревьев и переменные, вычислению которых соответствует каждое дерево. Узел вида `("look",i)` трактуется как подстановка дерева номер `i`. Использование ссылок позволяет сделать запись входных линейных участков более компактной и удобочитаемой. Узлы `out` задают выходные значения линейного участка.

2. Метод точного совместного решения задач выбора и планирования команд

Процесс генерации кода в ISched состоит из двух основных этапов.

1. Первичный выбор команд, т. е. формирование множества команд, которые в принципе могут участвовать в реализации входного вычисления.

2. Формирование и поиск решения ЦЛП-задач планирования и выбора команд для заданного числа тактов (T) процессора, где T пробегает значения 1, 2, ... Процесс завершается, когда для очередного значения T удается найти решение ЦЛП-задачи.

Более подробно эти два этапа описаны далее.

2.1. Первичный выбор команд

На этапе первичного выбора команд генерируется множество команд, которые могут быть использованы для реализации вычислений, заданных во входном линейном участке. Вначале для этого формируется множество уникальных поддеревьев, содержащихся в представлении линейного участка. В результате автоматически сливаются общие подвыражения, присутствующие в представлении линейного участка.

Затем к каждому поддереву применяются все правила, шаблон сопоставления которых сопоставляется с корневой частью поддерева. В отличие от алгоритма BURG, генератор ISched не пытается на этом этапе выбрать оптимальный набор команд, а порождает для каждого поддерева все возможные команды, которые могут быть в дальнейшем использованы при генерации кода. Окончательный выбор команд происходит на этапе планирования и выбора команд.

2.2. Планирование и выбор команд

На этом этапе проводится поиск точного совместного решения задач выбора и планирования команд. Для значений $T = 1, 2, \ldots$ последовательно генерируются ЦЛП-задачи для выбора команд и составления расписаний их выполнения, укладывающиеся в T тактов.

Генератор ISched использует формулировку ЦЛП-задачи выбора и планирования команд, представленную в работе [1], с некоторыми уточнениями, в частности, в ISched реализованы возможности оптимизации общих подвыражений, использование команд, имеющих несколько результатов, а также команд с аргументами, являющимися одновременно входными и выходными.

3. Пример применения генератора ISched

Для примера, представленного в разд. 1, на этапе первичного выбора команд сгенерировано 19 уникальных поддеревьев и 181 команда, из которых после фильтрации осталось 85 команд.

На этапе планирования, при наличии 32 доступных регистров сгенерирован код, выполняющийся за 28 тактов (рис. 2).

```
0:  load D_1,b                                ;;
1:  load D_5,c                                ;;
2:  mul D_2,D_1,D_1     load D_4,a            ;; D_1
3:                                            ;; D_1    D_5
4:  mul D_6,D_4,D_5     add D_18,D_5,D_5      ;; D_1    D_4    D_5
5:  move D_3,4                                ;; D_1    D_2    D_4
6:                                            ;; D_1    D_2    D_4    D_18
7:  msub D_8<<D_2,D_3,D_6 add D_16,D_4,D_4 ;; D_1 D_2    D_3    D_4
                                              ;; D_6    D_18
8:                                            ;; D_1    D_2    D_18
9:                                            ;; D_1    D_2    D_16   D_18
10: sqrt D_9,D_8                              ;; D_1    D_8    D_16   D_18
11:                                           ;; D_1    D_16   D_18
12:                                           ;; D_1    D_16   D_18
13:                                           ;; D_1    D_16   D_18
14:                                           ;; D_1    D_16   D_18
15:                                           ;; D_1    D_16   D_18
16:                                           ;; D_1    D_16   D_18
17: addsub D_10<<D_9,D_11<<D_1                ;; D_1    D_9    D_16   D_18
18: test D_1                                  ;; CC_12  D_1    D_9
                                              ;; D_16   D_18
19: ifGE,D_13,=,D_10,D_11                     ;; CC_12  D_10   D_11
                                              ;; D_16   D_18
20:                                           ;; D_16   D_18
21: neg D_14,D_13                             ;; D_13   D_16   D_18
22:                                           ;; D_16   D_18
23: div1 D_17,D_14,D_16  div2 D_19,D_18,D_14 ;; D_14    D_16 D_18
24:                                           ;;
25:                                           ;;
26:                                           ;;
27:                                           ;; D_19
28:                                           ;; D_17    D_19
```

Рис. 2

На рис. 2 слева даны номера тактов. Номер регистра (`D_1`, `D_2`, `C_12`, ...) соответствует номеру уникального поддерева (см. подразд. 2.1), значение которого вычисляется на этом регистре. Справа, в виде комментариев, показаны имена "живых" виртуальных регистров, значения которых вычислены к данному такту, но еще не использованы всеми командами-потребителями.

Нотация вида `D_8 < < D_2` (например, в такте 7) применяется для команд, которые записывают результат на место входного аргумента. Здесь `D_2` соответствует входному значению, `D_8` – выходному.

В сгенерированном коде использованы команды со сложными шаблонами, такие как `msub`, условная пересылка (`ifGE`). Для вычисления суммы и разности $\sqrt{D} + b$, $\sqrt{D} - b$ использована команда с двумя результатами `addsub`. Для вычисления выражений $2c$, $2a$ было применено правило оптимизации, реализующее умножение на 2 командой сложения. Для операций деления сгенерированы разные команды `div1` и `div2`.

Ограничим теперь число доступных регистров до трех. На рис. 3 представлен код, который был получен для той же программы при этом ограничении.

На рис. 3 можно видеть, что ни на одном такте нет одновременно более трех "живых" `D`-регистров, т. е. ограничение по числу регистров соблюдено. Дефицит регистров в данном примере компенсируется повторной загрузкой регистров `D_1`, `D_5`, в которых хранятся коэффициенты `b`, `c`. Сохранять в памяти эти регистры не требуется, достаточно вновь загрузить их значения из памяти там, где они понадобятся. Таким образом, генератор кода ISched выбрал наиболее экономный способ спиллинга. В общем случае данный метод генерации кода позволяет использовать различные механизмы спиллинга, включая выгрузку значений в память или в регистры менее дефицитных классов, перевычисление значений (*rematerialization*).

В рассмотренном примере сгенерированный код в недостаточной степени использует вычислительные ресурсы процессора. На многих тактах команды не запускаются либо запускается только одна команда, что связано с недостаточным параллелизмом кода в исходном

```
0:  load D_1,b                                  ;;
1:  load D_4,a                                  ;;
2:  mul D_2,D_1,D_1     load D_5,c              ;; D_1
3:                                              ;; D_4
4:  mul D_6,D_4,D_5                             ;; D_4   D_5
5:  move D_3,4                                  ;; D_2   D_4
6:  add D_16,D_4,D_4                            ;; D_2   D_4
7:  msub D_8<<D_2,D_3,D_6                       ;; D_2   D_3   D_6
8:                                              ;; D_2   D_16
9:                                              ;; D_2   D_16
10: sqrt D_9,D_8                                ;; D_8   D_16
11:                                             ;; D_16
12:                                             ;; D_16
13:                                             ;; D_16
14:                                             ;; D_16
15: load D_1,b                                  ;; D_16
16:                                             ;; D_16
17: addsub D_10<<D_9,D_11<<D_1                  ;; D_1   D_9   D_16
18: test D_1                                    ;; CC_12 D_1   D_9   D_16
19: load D_5,c     ifGE,D_13,=,D_10,D_11        ;; CC_12 D_10  D_11  D_16
20:                                             ;; D_16
21: neg D_14,D_13  add D_18,D_5,D_5             ;; D_5   D_13   D_16
22:                                             ;; D_16
23: div1 D_17,D_14,D_16     div2 D_19,D_18,D_14 ;; D_14 D_16  D_18
24:                                             ;;
25:                                             ;;
26:                                             ;;
27:                                             ;; D_19
28:                                             ;; D_17  D_19
```

Рис. 3

линейном участке. Очевидно, что подобный метод генерации кода имеет смысл применять в сочетании с методами, повышающими степень параллелизма кода в программе, например в сочетании с разверткой циклов.

Заключение

Метод совместного точного решения задач выбора и планирования команд, реализованный в прототипе генератора кода ISched, для заданного линейного участка позволяет находить набор команд и расписание, оптимальные по времени выполнения, с учетом ограничений на доступное число регистров.

Хотя данный метод требует больших затрат вычислительных ресурсов, его применение может быть оправдано в компиляторах для процессоров, используемых во встроенных системах. Такие процессоры могут иметь высокий потенциал параллелизма выполнения команд и содержать специализированные команды, предназначенные для эффективной реализации типичных вычислений, но обычно не включают средств аппаратной оптимизации (аппаратное планирование, переименование регистров и т. п.). Эффективное использование возможностей подобных процессоров при компиляции с языков высокого уровня невозможно без применения сложных высокоинтеллектуальных подходов к генерации кода.

По сравнению с результатами, представленными в работе [1], в прототипе генератора кода ISched реализован ряд новых возможностей. К их числу относятся перечисленные далее.

1. Язык описания ресурсов и системы команд процессора. Язык позволяет описывать команды с произвольными древовидными шаблонами сопоставления. Поддерживается описание команд, имеющих более одного результата, команд с аргументами, являющимися входными и выходными, а также описание некоторых правил оптимизации вычислений.

2. Язык описания входного линейного участка, для которого требуется сгенерировать код. Входной участок описывается как последовательность выражений, представленных в списочной ЛИСП-подобной форме. Поддерживаются средства для описания входных и выходных значений участка, а также ссылки из одних выражений на другие в пределах линейного участка.

3. Оптимизация общих подвыражений при генерации кода.

4. Использование команд, имеющих более одного результата, таких как `addsub`, команд, порождающих

код условия, условных пересылок, а также применение правил оптимизации при генерации кода.

Перспективным направлением развития данного подхода представляется использование тождественных преобразований для оптимизации вычислений. Простой пример такого рода оптимизаций — применение свойств коммутативности и ассоциативности операций. Рассмотрим в качестве примера выражение $((a + b) + c)$. Если a находится в памяти, а b, c — в регистрах, то преобразование этого выражения в $(a + (b + c))$ позволит ускорить вычисления за счет совмещения загрузки a и сложения $(b + c)$.

Другой пример — формула разности квадратов. Замена $(a^2 - b^2)$ на $(a + b) \cdot (a - b)$ как правило приведет к ускорению программы, поскольку аддитивные операции обычно выполняются быстрее, чем умножение. Но если в том же участке программы присутствует выражение $(a^2 + b^2)$, то такая замена может оказаться невыгодной. Метод точного совместного решения задач выбора и планирования команд позволяет принять решение о целесообразности той или иной комбинации трансформаций с учетом различных факторов, включая возможности параллельного выполнения команд и наличие общих подвыражений.

Одна из сложностей, связанных с применением тождественных преобразований, заключается в необходимости так или иначе ограничивать экспоненциальный рост числа рассматриваемых альтернатив.

Еще одно важное направление исследований — распространение метода точного совместного решения задач выбора и планирования команд на генерацию кода для циклических участков программ, т. е. разработка методов конвейеризации циклов с подбором команд. Применение выбора команд при конвейеризации циклов может давать существенный выигрыш в эффективности кода (см. пример в работе [2]). Привлекательной стороной этого подхода является также возможность автоматически генерировать и планировать spill-код при дефиците регистров.

В то же время следует отметить, что планирование циклических участков кода с выбором команд представляется существенно более сложной задачей по сравнению с применением данного подхода для линейных участков. Основные трудности связаны с тем, что при конвейеризации цикла необходимо поддерживать версии переменных, относящиеся к нескольким соседним итерациям цикла.

Список литературы

1. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** Совместное решение задач выбора и планирования команд в условиях дефицита регистров // Программная инженерия. 2012. № 2. С. 35—41.

2. **Вьюкова Н. И.** Генерация кода с отложенным выбором инструкций. М.: НИИСИ РАН, 2010. С. 80—96.

3. **Fraser C. W., Henry R. R., Proebsting T. A.** BURG — Fast optimal instruction selection and tree parsing // SIGPLAN Notices 27. 1992. № 4. P. 68—76.

4. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** К проблеме выбора машинных инструкций в генераторах кода // Моделирование и визуализация. Многопроцессорные системы. Инструментальные средства разработки ПО. Сборник статей. М.: НИИСИ РАН, 2009. С. 3—31.

5. **Язык** программирования Python. URL: http://www.python.ru/

6. **Самборский С. В.** Формулировка задачи планирования линейных и циклических участков кода // Программные продукты и системы. 2007. № 3 (79). С. 12—16.

ИНФОРМАЦИЯ

С 24 по 27 сентября 2014 г. в г. Казань состоится **Четырнадцатая национальная конференция по искусственному интеллекту с международным участием (КИИ-2014).** Конференция проводится Российской ассоциацией искусственного интеллекта (РАИИ) совместно с Академией наук Республики Татарстан, Казанским Федеральным (Приволжским) университетом, Институтом проблем управления РАН и Институтом системного анализа РАН. Во время конференции будет проведен очередной съезд РАИИ.

Тематика конференции:

- моделирование рассуждений и неклассические логики;
- машинное обучение в интеллектуальных системах и интеллектуальный анализ данных;
- компьютерная лингвистика;
- когнитивное моделирование;
- планирование и моделирование поведения;
- интеллектуальные роботы;
- искусственный интеллект в социальной сфере и гуманитарных исследованиях;
- интеллектуальные обучающие системы и среды;
- сетевые модели в искусственном интеллекте;
- нечеткие модели и "мягкие" вычисления;
- эволюционное моделирование и генетические алгоритмы;
- моделирование образного мышления и когнитивная графика;
- инженерия знаний, онтологии и управление знаниями;
- инструментальные системы для искусственного интеллекта;
- многоагентные и распределенные интеллектуальные системы;
- интеллектуальные Интернет-технологии, семантический Web;
- прикладные интеллектуальные системы, динамические интеллектуальные системы и системы реального времени;
- интеллектуальные системы поддержки принятия решений и управления;
- интеллектуальные организации и виртуальные сообщества;
- методологические и философские проблемы искусственного интеллекта;
- другие темы, относящиеся к проблемам искусственного интеллекта.

Во время конференции состоится **Выставка интеллектуальных продуктов**, тематика которой включает следующие направления:

- программное обеспечение интеллектуальных систем;
- прикладные интеллектуальные системы;
- интеллектуальные роботы;
- объекты современного научно-технического искусства.

Адрес сайта конференции: http://raai.org/cai-14/

УДК 004.3

М. В. Юрушкин, аспирант, Южный федеральный университет, г. Ростов-на-Дону,
e-mail: m.yurushkin@gmail.com

Автоматизация блочного размещения данных в оперативной памяти компилятором языка Си

Дано описание расширения языка Си, поддерживающего блочное размещение массивов, которое реализовано в системе OPC (оптимизирующая распараллеливающая система) в виде нескольких директив компилятора. Приведены результаты ускорения программ за счет использования реализованных директив.

Ключевые слова: *оптимизирующие компиляторы, блочное размещение данных, кеш-память, TLB-кеш*

M. V. Yurushkin

Block Data Layout Automation in C Language Compiler

In this paper C language extension enabling automatic block data layout support is described. It was implemented in OPS (Optimizing Parallel System) using several compiler directives. Numerical experiments results of program acceleration using implemented directives are presented.

Keywords: *optimizing compilers, block data layout, data cache, TLB cache*

Введение

Процесс оптимизации программы для работы на суперкомпьютере не имеет смысла без оптимизации работы этой программы на одном процессоре. Для многих задач одним из способов оптимизации доступа к памяти в рамках одного процессора является использование блочного размещения. Так, в статье [1] приводится описание алгоритма блочного умножения матриц, использующего двойное блочное размещение данных. По производительности реализованный алгоритм превосходит библиотеку MKL и библиотеку PLASMA. Следует отметить, что библиотека PLASMA уже использует [2] блочное размещение данных в оперативной памяти (далее для краткости изложения — блочное размещение) в качестве основного.

Блочное размещение встречается в различных задачах. Так, в алгоритме генерации полигонов, который используют в методе конечных элементов, замена стандартного распределения на блочное дает ускорение в 1,7 раза [3]. В алгоритме умножения матриц библиотеки GotoBLAS блочное распределение матриц используют для реализации эффективной векторизации (*register blocking*) [4].

В то же время использование блочного размещения данных затруднено в силу необходимости высокой квалификации программиста. Современные языки программирования используют стандартные схемы размещения данных в оперативной памяти, такие как размещение по строкам [5] (язык Си) и размещение по столбцам [5] (язык ФОРТРАН). Непосредственное применение блочного размещения увеличивает объем результирующей программы и порождает сложные индексные выражения во вхождениях блочно размещаемых массивов. По этой причине представляется интересной задача автоматизации поддержки блочного размещения массивов в языках программирования. Такая автоматизация была реализована в системе OPC (оптимизирующая распараллеливающая система) в виде нескольких директив компилятора.

В статье приводятся результаты ускорения программ за счет данных директив на задачах линейной алгебры (LU-разложение матрицы, перемножение матриц), двумерной фильтрации изображений и ряда других.

Блочное размещение данных

Блочное размещение — это способ хранения массива в памяти, при котором массив разбивается на блоки одинакового размера. Блоки матрицы хранятся в памяти последовательно без промежутков. Элементы, находящиеся внутри одного блока, хранятся в памяти стандартным образом, например, по строкам.

В некоторых блочных алгоритмах блочное размещение данных дает существенное увеличение производительности. Рассмотрим блочный алгоритм умножения квадратных матриц $\mathbf{C} = \mathbf{A} \cdot \mathbf{B}$ (рис. 1).

```
void matrix_mult(int N, int d) {
    ...

    double *A, *B, *C;
    A = malloc(sizeof(double)*N*N);
    B = malloc(sizeof(double)*N*N);
    C = malloc(sizeof(double)*N*N);

    //  инициализация матриц A, B
    …

    for(di = 0; di < N; i+=d)
    for(dj = 0; dj < N; j+=d)
    for(dk = 0; dk < N; k+=d)
        for(i = di; i < MIN(N, di+d); i++)
        for(k = dk; k < MIN(N, dk+d); k++)
        for(j = dj; j < MIN(N, dj+d); j++)
           C[i*N+j] += A[i*N+k]*B[k*N+j];
    …
}
```

Рис. 1. Пример реализации блочного алгоритма умножения матриц, использующего стандартное размещение данных

Если размер матриц *N* больше размера виртуальной страницы, то соседние по вертикали элементы матрицы будут находиться в различных виртуальных страницах. Нетрудно подобрать такой размер блока *d*, при котором TLB-кеш уже не будет способен хранить физические адреса всех используемых виртуальных страниц. При выполнении программы при этом будет происходить большое число промахов к TLB-кешу. Ситуация усугубляется еще больше, так как в случае промаха к TLB-кешу процессор вынужден простаивать (в отличие от случая с промахом к кешу данных). Эту ситуацию можно разрешить, если разместить матрицы **A**, **B**, **C** блочно с размером блока, равным *d*. В этом случае при перемножении блоков задействуется минимальное число виртуальных страниц.

Если размер блока *d* не кратен размеру кеш-линейки, то блочное размещение увеличивает также эффективность использования кеша данных. Этот факт обусловлен тем, что данные пересылаются кеш-линейками. Если размер блока не кратен размеру кеш-линейки, то при стандартном размещении данных в кеш данных будут попадать не только элементы перемножаемых блоков, но также элементы и соседних блоков. Кеш данных будет засоряться неиспользуемыми данными, что отрицательно скажется на производительности. Напротив, при блочном размещении в кеш будут подкачиваться только элементы перемножаемых блоков.

Директивы блочного размещения данных в компиляторе языка Си

В системе компиляторов OPC поддержка блочного размещения массивов реализована в виде нескольких директив компиляции. Перед объявлением блочно размещаемого массива указывается директива:

```
#pragma ops array declare(name, array
   dimension size list, block dimension size
   list)
```

Параметры директивы:

- `name` — имя размещаемого массива;
- `array dimension size list` — массив размерностей размещаемого массива по каждому измерению;
- `block dimension size list` — массив размерностей блока по каждому измерению.

В данной директиве указывается информация о размере массива, а также о размере блоков, на которые его целесообразно разбить.

Оператор, в котором осуществляется выделение памяти для массива **A**, следует пометить директивой

```
#pragma ops array allocate(name)
```
(1)

```
void matrix_mult(int N, int d) {
    ...
    #pragma ops array declare(A, N, N, d, d)
    #pragma ops array declare(B, N, N, d, d)
    #pragma ops array declare(C, N, N, d, d)
    double *A, *B, *C;

    #pragma ops array allocate(A)
    A = malloc(sizeof(double)*N*N);
    #pragma ops array allocate(B)
    B = malloc(sizeof(double)*N*N);
    #pragma ops array allocate(C)
    C = malloc(sizeof(double)*N*N);

    //  инициализация матриц A, B
    …
    for(di = 0; di < N; i+=d)
    for(dj = 0; dj < N; j+=d)
    for(dk = 0; dk < N; k+=d)
        for(i = di; i < MIN(N, di+d); i++)
        for(k = dk; k < MIN(N, dk+d); k++)
        for(j = dj; j < MIN(N, dj+d); j++)
           C[i*N+j] += A[i*N+k]*B[k*N+j];

    #pragma ops array release(A)
    free(A);
    #pragma ops array release(B)
    free(B);
    #pragma ops array release(C)
    free(C);
}
```

Рис. 2. Пример реализации блочного алгоритма умножения матриц, использующего директивы блочного размещения данных

Результаты тестирования директив блочного размещения в памяти (размер блока 256 × 256)

Алгоритм	Размер матриц	Время работы алгоритма без директив, с	Время работы алгоритма с директивами, с	Ускорение, %
Блочное умножение квадратных матриц	2048 × 2048	14,93	11,2	25
Блочное возведение матрицы в квадрат	2048 × 2048	81,37	17,36	78,6
Блочное LU-разложение матрицы	2048 × 2048	27,4	14,44	47
Блочное QR-разложение матрицы	2048 × 1024	19,6	17,11	12,7
Двумерная свертка матрицы	1024 × 1024	17,9	10,54	41

Директива (1) заменяет исходный оператор выделения памяти на новый оператор освобождения памяти, в котором выделяется память под блочно размещаемый массив. Блочно размещаемый массив должен иметь размер, кратный размеру блока, так как выполнение данного условия на практике сильно упрощает генерацию индексных выражений. Аналогично, оператор, в котором осуществляется освобождение памяти массива **A**, следует пометить директивой

```
#pragma ops array release (name)        (2)
```

Директива (2) заменяет исходный оператор освобождения памяти на новый оператор освобождения памяти, в котором выделяется память под блочно размещаемый массив. На рис. 2 представлен пример использования таких директив в алгоритме умножения матриц.

Для того чтобы компилятор переразметил данные, к входной программе предъявляется следующее требование: в программе не должно быть операций взятия адреса блочно размещаемого массива; исключения составляют только операции выделения и освобождения памяти массива. Операции выделения и освобождения памяти должны быть аннотированы директивами (1) и (2) соответственно. Из данного ограничения следует, в частности:

- блочно размещаемый массив нельзя передавать в качестве аргумента другим функциям;
- над элементами блочно размещаемого массива запрещается использовать векторные операции, которые поддерживаются в языке Си в виде встроенных функций.

В следующем разделе приводятся результаты численных экспериментов, в которых использованы описанные выше директивы блочного размещения данных.

Численные эксперименты

Для проверки эффективности и корректности реализованных директив написан пакет прикладных блочных программ на языке Си, аннотированных данными директивами. Тексты программ автоматически преобразованы системой ОРС в двух режимах: с включенной опцией блочного размещения данных и без нее. В таблице приведены результаты сравнения производительности полученных программ на компьютере с процессором Intel Core i5-2410M Processor (3M Cache, 2.90 GHz). Для компиляции программ использовался компилятор gcc 4.7.2.

Согласно данным, приведенным в таблице, использование реализованных директив блочного размещения данных может существенно ускорить входную программу. В частности, на программе блочного возведения матрицы в квадрат ускорение достигает 78 %.

Заключение

В данной статье представлены директивы блочного размещения данных, реализованные в системе ОРС. Результаты численных экспериментов подтверждают, что эти директивы могут существенно увеличить производительность входной программы. На основании полученных результатов представляется перспективным дальнейшее развитие оптимизирующих компиляторов, учитывающих метод размещения массивов в памяти.

Также представляется интересным расширение класса входных программ, для которых блочное размещение имеет преимущество по сравнению со стандартными размещениями.

Список литературы

1. **Гервич Л., Штейнберг Б., Юрушкин М.** Программирование экзафлопсных систем // Открытые системы. 2013. № 08. URL: http://www.osp.ru/os/2013/08/13037853/

2. **PLASMA** Users' Guide. September 4th, 2010. URL: http://icl.cs.utk.edu/projectsfiles/plasma/pdf/users_guide.pdf

3. **Park N., Liu W., Prasanna V. K.**, **Raghavendra C.** Efficient Matrix Multiplication Using Cache Conscious Data Layouts // HPCMO User Group Conference 2000. URL: http://halcyon.usc.edu/~pk/prasannawebsite/papers/parkHPCMO00.pdf

4. **Goto Kazushige, van de Geijn R. A.** Anatomy of High-Performance Matrix Multiplication // ACM Transactions on Mathematical Software. 2008. Vol. 34, N 3. P. 1—25.

5. **Chatterjee S., Jain V. V., Lebeck A. R., Mundhra S., and Thottethodi M.** Nonlinear Array Layouts for Hierarchical Memory Systems // Proc. of 13th international conference on Supercomputing. June 1999. New York: ACM, 1999. P. 444—453. URL: http://dl.acm.org/citation.cfm?id = 305231

УДК 519.674

С. А. Карпухин, аспирант, МГУ им. Ломоносова,
e-mail: ks-linp@yandex.ru

О геометрической оптимизации методом растеризации сумм Минковского

Рассмотрена задача поиска наибольшего многогранника заданной формы внутри другого. Предложен численный метод решения, основанный на растеризации сумм Минковского внутреннего и внешнего многогранников. Доказана сходимость метода в случае звездного внутреннего многогранника.

Ключевые слова: *геометрическая оптимизация, размещение многогранников, суммы Минковского, наибольший вписанный многогранник*

S. A. Karpukhin

On Geometric Optimization by Means of Minkowski Sums Rasterisation

A problem of findind the largest polytope of specied shape inside another is considered. A numerical method based on the rasterisation of inner and outer polytopes' Minkowsky sums is proposed. A proof of method's convergence in case of star-shaped polytopes is provided.

Keywords: *geometric optimisation, polytope placement, Minkowski sums, largest inscribed polytope*

В ряде задач геометрической оптимизации возникает задача наилучшего размещения некоторых трехмерных объектов внутри другого трехмерного тела. Решение этой задачи автоматизируется при помощи компьютеров: объекты заменяют на их геометрические модели и решается задача оптимального размещения при заданных преобразованиях внутренних объектов. В качестве моделей чаще всего используют многогранники, а под оптимальным размещением понимается, например, наибольшее число объектов или наименьший незаполненный объем.

Подобного рода задачи возникают, например, в процессе огранки драгоценных камней, где внутренний объект представляет собой многогранник, определяющий форму огранки. Внешним объектом является многогранник — модель исходного камня. Оптимальным при этом считается размещение, при котором полученный в результате ограненный камень имеет наибольший размер при соблюдении заданных геометрических пропорций. Допустимыми преобразованиями естественно полагаются движения и масштабирование. В такой постановке задача сводится к поиску наибольшего многогранника заданной формы (*шаблона*) внутри данного другого многогранника (*контура*).

В современных программных пакетах обозначенная задача решается в два этапа: перебираются различные углы поворота и для каждого набора углов ищется наилучшее решение при заданной ориентации шаблона в пространстве. В настоящей статье рассмотрим второй этап, как более простой и, тем не менее, имеющий практическую ценность как часть существующего общего метода.

Перейдем к формальной постановке задачи. Будем рассматривать точки и многогранники в евклидовом пространстве $\mathbb{R}^d$.

Определение 1. Назовем *шаблоном* некоторый многогранник $P \subset \mathbb{R}^d$, для которого точка $0 \in \mathbb{R}^d$ — строго внутренняя. *Реализацией* с центром C радиуса r шаблона P будем называть многогранник $R_{C,r}(P) = T_{0C} \circ H_0^r(P)$, где T_{0C} — параллельный перенос, переводящий 0 в центр C, а H_0^r — гомотетия шаблона с коэффициентом r.

В представленных определениях имеем следующую постановку задачи.

Задача 1. Пусть даны *контур* — некоторый многогранник и шаблон. Требуется найти реализацию шаблона наибольшего радиуса, не выходящую за границу контура.

В настоящее время методы глобального решения задачи 1 разработаны только для выпуклых шаблонов [1, 2]. Используемые в них геометрические конструкции сложны в реализации и практически не масштабируются на имеющиеся многопроцессорные и графические архитектуры. Это происходит потому, что в исследуемых объектах часто встречаются локальные особенности (включения, сколы), приводящие к рез-

кому усложнению модели в районе особенности. Вследствие этого обстоятельства локализовать и независимо вычислить на большом числе процессоров взаимное расположение множества граней и геометрическую структуру на их основе не удается. При попытках получить более быстрые реализации исследовалось графическое представление геометрической структуры [3]. Подобный метод оказался не применим к поставленной выше оптимизационной задаче в силу высокой чувствительности к погрешностям растеризации: неудачный выбор разрешения приводил к ошибкам в структуре и, как следствие, в решении. Для устранения возникших трудностей был предложен метод локального уточнения структуры, однако вопрос поиска окрестностей, требующих уточнения, остается открытым.

В имеющихся промышленных комплексах данная задача решается при помощи одного из вариантов локального метода градиентного спуска: каким-либо образом перебираются стартовые решения и применяется процесс их улучшения. В отличие от глобальных методов, локальные обладают высокой производительностью и в случае сложных задач не требуют большого числа ресурсов. Однако локальные методы не всегда приводят к оптимальному решению. В силу их геометрической природы остаются те же вопросы, связанные с масштабированием и реализацией на графических процессорах.

В данной статье предлагается новый графический метод глобального решения задачи 1. Как глобальный метод он расширяет класс шаблонов до звездных, покрывающих все практические задачи. В его основе лежат простые графические процедуры на множестве однотипных объектов, благодаря чему ожидается высокая производительность нового метода на графических процессорах [4]. Для сложных задач имеется возможность ограничить используемые ресурсы и применять новый метод как локальный. В этом случае он может оказаться эффективнее на системах с большим числом процессоров, в особенности, графических.

Решение задачи в терминах сумм Минковского

Рассмотрим задачу 1. Заметим, что помещение шаблона внутрь контура эквивалентно наращиванию контура отражением шаблона. При этом точки внутри полученной фигуры, называемой далее ρ-суммой, представляют собой центры вписанных шаблонов. Увеличивая радиус шаблона можно найти предельное значение, при котором внутри построенной ρ-суммы все еще имеются точки. Они будут центрами решений задачи 1, а предельное ρ — радиусом. Исследуем этот процесс формально.

Определение 2. В условиях задачи 1 назовем *ρ-суммой* сумму Минковского [5] в $\mathbb{R}^d$ поверхности контура с внутренней областью множества $H_0^{-\rho}(P)$, где $H_0^{-\rho}$ — гомотетия с коэффициентом — ρ, т. е. $H_0^{-\rho}(P)$ — центрально-отраженный шаблон радиуса ρ. Обозначим через I_ρ *внутреннюю область* ρ-*суммы* — разность данного в задаче контура как многогранника с внутренними точками и ρ-суммы.

Лемма 1. *Если некоторая точка C лежит внутри* ρ-*суммы, то реализация* $R_{C,\rho}(P)$ *выходит за границу контура. Если же* $C \in I_\rho$, *то* $R_{C,\rho}(P)$ *не выходит за границу контура.*

Доказательство. $R_{C,\rho}(P)$ выходит за границу контура тогда и только тогда, когда на границе контура есть точка $C' \in R_{C,\rho}(P) = T_{0C} \circ H^{\rho}_0(P)$, что эквивалентно $C \in T_{0C'} \circ H_0^{-\rho}(P)$. Если C лежит внутри ρ-суммы, то по определению есть точка C' на поверхности контура, такая, что $C \in T_{0C'} \circ H_0^{-\rho}(P)$. Если $C \in I_\rho$, то по определению $C \notin T_{0C'} \circ H_0^{-\rho}(P)$ для всех точек C' на поверхности контура.

Лемма доказана.

Определение 3. Назовем шаблон *звездным*, если его поверхность видна из точки $0 \in \mathbb{R}^d$.

Лемма 2. *Для звездного шаблона если* $\rho_1 > \rho_2$, *то* $I_{\rho_1} \subset I_{\rho_2}$.

Доказательство. Пересечение звездного шаблона P с любым лучом, выходящим из 0, есть некоторый отрезок $0C$, где C лежит на границе P, и все точки P лежат на подобных отрезках. При $r < 1$ $H_0^r(0C) \subset 0C$, значит $H_0^r(P) \subset P$.

В условиях леммы, если точка $A \in I_{\rho_1}$, то по лемме 1 $R_{C,\rho_1}(P) = T_{0C} \circ H_0^{\rho_1}(P)$ не выходит за границу контура. Из $\rho_2/\rho_1 < 1$ и полученного выше соотношения имеем $R_{C,\rho_2}(P) = T_{0C} \circ H_0^{\rho_2/\rho_1} \circ H_0^{\rho_1}(P) \subset R_{C,\rho_1}(P)$. Тогда $R_{C,\rho_2}(P)$ не выходит за границу контура и по лемме 1 $A \in I_{\rho_2}$, значит $I_{\rho_1} \subset I_{\rho_2}$.

Лемма доказана.

Теорема 1. *В условиях задачи 1 пусть внутренняя область контура не пуста, а шаблон звездный. Тогда существует* $r > 0$, *такое, что* $I_\rho \neq \varnothing$ *при* $\rho < r$ *и* $I_\rho = \varnothing$ *при* $\rho > r$. *При этом* $I = \bigcap_{\rho < r} I_\rho$ *не пусто, и* $\rho < r$ *множество решений задачи 1 есть множество реализаций шаблона с центром в I радиуса r.*

Доказательство. Найдем такое ρ_1, что реализация радиуса ρ_1 данного в задаче шаблона P содержит сферу с центром в 0 радиуса D, где D — диаметр контура. Это возможно, так как 0 по определению лежит строго внутри шаблона, значит шаблон содержит некоторую сферу радиуса D_0. Можем положить $\rho_1 = D/D_0$. Для любой точки C контура $C + H_0^{-\rho_1}(P)$ содержит все точки на расстоянии не больше D от C, т. е. все точки контура. По определению ρ_1-суммы, она содержит такие множества для всех C, следовательно ρ_1-сумма содержит контур и $I_{\rho_1} = \varnothing$.

$I_0 = \varnothing$, так как I_0 совпадает с контуром, а внутренняя область контура не пуста по условию.

Пусть $r = \inf_{\rho \geqslant 0}\{\rho : I_\rho = \varnothing\}$. $r > 0$, так как $I_0 \neq \varnothing$, и $r < \infty$, так как $I_{\rho_1} = \varnothing$. Для всех $\rho < r$ $I_\rho = \varnothing$ по

определению r. Если $\rho > r$, то существует такое r_1, что $r \leqslant r_1 < \rho$ и $I_{r_1} = \varnothing$ по определению r. Тогда по лемме 2 $I_\rho = \varnothing$ и первая часть теоремы доказана.

Для всех $\rho > r$ имеем $I_\rho = \varnothing$, значит все точки контура лежат в ρ-сумме и по лемме 1 любая реализация шаблона радиуса ρ выходит за границу контура. Следовательно, радиус решения задачи 1 не превосходит r. Докажем, что $\mathcal{I}$ не пусто. Действительно, I_ρ — компакт для любого ρ, так как ограничена контуром и замкнута как разность замкнутого и открытого множеств. По лемме 2 I_ρ — вложенные компакты. По теореме о вложенных компактах $\mathcal{I} \neq \varnothing$. Докажем, что для любой точки $M \in \mathcal{I}$, $R_{M,r}(P)$ не выходит за границу контура. Предположим противное, тогда есть некоторая точка B на границе контура, лежащая строго внутри $R_{M,r}(P)$, т. е. $R_{M,r}(P)$ содержит шар с центром В некоторого радиуса $\varepsilon > 0$. Гомотетия H_0^k — линейное преобразование, поэтому можно выбрать $k < 1$, такое, что $P \backslash H_0{}^k(P)$ лежит в $\varepsilon/2r$-окрестности границы P. Тогда $R_{M,r}(P) \backslash R_{M,r}(H_0^k(P))$ лежит в ε/2-окрестности границы $R_{M,r}(P)$. Точка B удалена от границы $R_{M,r}(P)$ как минимум на ε, значит B лежит строго внутри $R_{M,kr}(P) = R_{M,r}(H_0^k(P))$. Следовательно, реализация с центром M радиуса $kr < r$, выходит за границу контура и по лемме 1 $M \notin I_{kr}$, а значит $M \notin \mathcal{I}$. Получили противоречие, предположение неверно и $R_{M,r}(P)$ не выходит за границу контура. Следовательно, все реализации с центром в $\mathcal{I}$ радиуса r являются решениями задачи 1. Обратно, любое решение задачи 1 — реализация радиуса r, не пересекающаяся с контуром, значит ее центр $C \in I_r \subset I_\rho \; \forall \rho < r$ по леммам 1, 2 и по определению $\mathcal{I}$: $C \in \mathcal{I}$.

Теорема полностью доказана.

Численное решение методом растеризации

Пользуясь полученной теоремой построим численный метод решения задачи 1: положим $\rho = 0$ и будем его увеличивать с некоторым приращением $\delta = \varepsilon$, пока $I_\rho \neq \varnothing$. Когда процесс остановится, получим решение задачи с точностью ε. Поскольку ε мало́ по сравнению с радиусом решения r, возьмем на первых шагах алгоритма приращение порядка диаметра контура и будем уменьшать его до необходимой точности ε. Основная сложность такого алгоритма заключается в проверке $I_\rho \neq \varnothing$. Заметим, что ρ на каждом шаге определено лишь с точностью до текущего приращения δ, тогда проверку $I_\rho \neq \varnothing$ достаточно выполнить с точностью порядка δ. Это можно сделать с помощью объемного растра [6]: рассмотрим растр в $\mathbb{R}^d$ как набор кубов (вокселей), и изображение ρ-суммы на этом растре. Тогда $I_\rho \neq \varnothing$, если внутри контура есть пустые воксели построенного изображения. Перейдем к формальному описанию метода.

Определение 4. Будем называть кубы в $\mathbb{R}^d$ с гранями, параллельными координатным плоскостям, *простыми*. Под *растром* в $\mathbb{R}^d$ будем понимать набор $(\mathcal{G}_n, \mathcal{P})$, где $\mathcal{G}_n = \{1, ..., n\}^d$ — сетка, а $\mathcal{P}$: $\mathcal{G} \to \mathbb{R}^d$ — вложение растра в $\mathbb{R}^d$. При этом полагаем, что $\mathcal{P}$ переводит $\mathcal{G}_n$ в некоторый простой куб G, называемый далее *контейнером* растра, и образом точки $g \in \mathcal{G}_n$ является *воксел* — простой куб со стороной в n раз меньше стороны G, расположенный в G соответственно своим *координатам g. Шагом* растра назовем длину стороны его вокселя. *Изображением* множества $\mathcal{M} \subset \mathbb{R}^d$ на растре $(\mathcal{G}_n, \mathcal{P})$ будем называть этот растр с отображением $\mathcal{R} : \mathcal{G}_n \to \{0, \chi, 1\}$, для которого

$\mathcal{R}(g) = 1$ тогда и только тогда, когда $\mathcal{P}(g) \subset \mathcal{M}$;

$\mathcal{R}(g) = 0$, тогда и только тогда, когда $\mathcal{P}(g) \cap \mathcal{M} = \varnothing$;

$\mathcal{R}(g) = \chi$ в остальных случаях.

Воксел изображения назовем *установленным*, если для его координат g $\mathcal{R}(g) = 1$, *пустым*, если $\mathcal{R}(g) = 0$, и *граничным*, если $\mathcal{R}(g) = \chi$.

Определение 5. Будем считать, что внутренняя область ρ-суммы I_ρ *не пуста* с точностью δ, если существует простой куб K_δ со стороной δ, такой, что $K_\delta \subset I_\rho$. В противном случае говорим, что ρ-сумма пуста с точностью δ.

Для построения алгоритма понадобится следующая лемма.

Лемма 3. *Если I_ρ не пуста с точностью δ, то на любом изображении ρ-суммы с шагом δ/2, покрывающем I_ρ, есть пустой воксел внутри контура. Если на некотором изображении ρ-суммы с шагом δ/2 нет пустых вокселей внутри контура, то I_ρ пуста с точностью δ.*

Доказательство. Пусть I_ρ не пуста, т. е. содержит простой куб K_δ. Для любого изображения ρ-суммы с шагом δ/2 рассмотрим координаты границ вокселей вдоль каждой координатной оси $0i$. Найдем из них наименьшие p, лежащие в проекции $[k_1, k_2]$ куба K на оси $0i$. Если $p > k_1 + \delta/2$, то $k_1 < p - \delta/2 < p$ — координата границы вокселя вдоль $0i$ и имеем противоречие с минимальностью p. Иначе, $p \leqslant k_1 + \delta/2$ и $p + \delta/2 \leqslant k_2$, так как $k_2 = k_1 + \delta$. Получаем некоторую координату g_i, такую, что проекции всех вокселей с координатой g_i на $0i$ лежат внутри соответствующей проекции K_δ. Рассмотрим воксел с координатами $g = (g_1, ..., g_n)$. Все его проекции лежат в соответствующих проекциях K_δ, значит этот воксел лежит в K_δ и он пустой, так как $K_\delta \subset I_\rho$. Второе утверждение леммы есть отрицание первого.

Лемма доказана.

В данных определениях алгоритм численного метода решения задачи 1 с заданной точностью ε выглядит следующим образом.

Шаг 1. Зададим начальные радиус $\rho = 0$, приращение $\delta = D(G)$, где $D(G)$ — диаметр куба G, содержащего контур, и текущий контейнер — куб $K = G$.

Шаг 2. Будем увеличивать ρ на δ, пока I_ρ не пуста с точностью δ.

Шаг 2.1. Построим [7] изображение $(\rho + \delta)$-суммы, на растре $(\mathcal{G}_n, \mathcal{P})$ с контейнером $\mathcal{P}(\mathcal{G}_n) = K$ и шагом δ/2. Все воксели вне контура считаем установленными.

Шаг 2.2. Если построенное изображение содержит пустой воксел, то увеличим ρ на δ и перейдем к предыдущему шагу. По лемме 3 условие этого шага эквивалентно пустоте $I_{\rho + \delta}$ с точностью δ.

Шаг 2.3. В противном случае перейдем к следующему шагу.

Шаг. 3. Примем в качестве нового контейнера K куб, содержащий все пустые и граничные воксели по-

лученного на шаге 2.1 изображения. Затем уменьшим δ в 2 раза и перейдем к шагу 2, если $\delta \geqslant \varepsilon$.

Шаг 4. Положим приближенным решением задачи 1 с точностью ε реализацию шаблона радиуса ρ с центром в любом пустом вокселе полученного на шаге 2.1 изображения.

Докажем основную теорему настоящей статьи о сходимости построенного метода к решению задачи 1 на некотором классе шаблонов.

Определение 6. Назовем шаблон *невырожденным*, если он звездный и все прямые, лежащие в гиперплоскостях граней шаблона, не проходят через 0.

Лемма 4. *Если шаблон P невырожден, то существует такое $\delta > 0$, что при любой гомотетии H_0^k с $1 - \delta < k < 1$ все точки границы P становятся строго внешними.*

Доказательство. Если есть точка границы P, остающаяся на границе при гомотетии с любым $1 - \delta < k < 1$, то выбрав δ достаточно малым, получим, что P остается при гомотетии в пределах одной грани. Получаем отрезок, лежащий в гиперплоскости грани шаблона, сохраняющийся при гомотетии, значит прямая, содержащая этот отрезок, сохраняется при гомотетии и проходит через ее центр. Получили противоречие с невырожденностью шаблона, значит для некоторого δ при любой гомотетии с $1 - \delta < k < 1$ граничные точки становятся либо строго внешними, либо строго внутренними. Второй случай невозможен, так как при гомотетии с коэффициентом $k < 1$ образ звездного шаблона содержится внутри прообраза (см. доказательство леммы 2).

Лемма доказана.

Теорема 2. *Для невырожденного шаблона результат работы предложенного алгоритма сходится к решению задачи 1 при $\varepsilon \to 0$.*

Доказательство. На шаге 2 ρ увеличивается с постоянным приращением δ и $\rho < r$ из теоремы 1, так как всегда остается пустой воксел внутри I_ρ, поэтому число итераций шага 2 конечно на каждую итерацию шага 3. Число итераций шага 3, очевидно, равно $[\log_2 D(G)/\varepsilon]$ и также конечно. Следовательно, алгоритм всегда сходится.

Изображение ρ-суммы (шаг 2.2) всегда содержит пустой воксел внутри I_ρ и в конце работы алгоритма получается реализация радиуса ρ с центром внутри I_ρ. По лемме 1 эта реализация не выходит за границы контура и для доказательства утверждения достаточно показать, что $\rho \to r$ при $\varepsilon \to 0$. Поскольку от ε в алгоритме зависит только условие остановки, при $\varepsilon_1 < \varepsilon_2$ алгоритм с точностью ε_2 содержит все действия алгоритма с точностью ε_1 и, возможно, дополнительные после них. Следовательно, для доказательства утверждения достаточно рассмотреть последовательность ρ в алгоритме без условия остановки и доказать, что $\rho \to r$.

Последовательность ρ монотонно возрастает и ограничена r (теорема 1), значит она имеет предел $\bar{\rho}$. Пусть $\bar{\rho} < r$, тогда $\exists \rho' : \bar{\rho} < \rho' < r$ и $I_{\rho'}$ не пуста с некоторой точностью ε', так как по лемме 4 можно взять $\rho' = r - \delta$, где $\delta < |r - \bar{\rho}|$ так, что все точки I_r, включая границу r-суммы, станут строго внешними к ρ'-сумме, а значит, строго внутренними для $I_{\rho'}$. Строго внутренняя точка вместе с собой содержит куб с некоторой стороной ε'.

Возьмем некоторое $\bar{\varepsilon} < \min(\varepsilon', |\rho' - \bar{\rho}|)$ и наибольшее δ из алгоритма с условием $\delta < \bar{\varepsilon}$. Тогда на некотором шаге получим $\rho > \bar{\rho} - \delta$. Можно увеличить ρ на δ, так как $\rho + \delta < \rho'$ (так выбрано δ), контейнер (см. шаг 3) всегда содержит I_ρ, а значит и $I'_\rho \subset I_{\rho+\delta} \subset I_\rho$. Область I'_ρ не пуста с точностью ε', значит на шаге 2.2 изображение будет содержать пустой воксел, ρ увеличится на δ и получим $\rho > \bar{\rho}$, что противоречит определению $\bar{\rho}$.

Следовательно, исходное предположение $\bar{\rho} < r$ неверно и $\bar{\rho} \geqslant r$. Из теоремы 1 следует, что $\bar{\rho} \leqslant r$, а значит $\bar{\rho} = r$.

Теорема доказана.

Результаты расчетов и выводы

Предложенный в статье метод был реализован на практике. Растеризация проводилась путем вычисления ρ-суммы для каждой грани контура аналитически с последующей растеризацией полученных треугольников. Время работы полученной реализации на контуре с 8192 гранями и шаблоне с 1980 гранями составляет 14 с в один поток на процессоре CPU Intel Core i5 2.4GHz. Это подтверждает применимость нового метода к задаче 1, а результаты [4, 7] позволяют рассчитывать на высокую производительность в будущем.

С теоретической точки зрения пока не найдены оценки размера растра на шаге 2.1 алгоритма, однако в ходе практических испытаний установлено, что для реальных контуров и шаблонов достаточно растров со стороной в 10...18 вокселей. В случае сложных задач можно жестко ограничить размер растра в алгоритме и на шаге 3 выбирать в качестве нового контейнера некоторую окрестность пустого вокселя, наиболее удаленного от ρ-суммы. Здесь в качестве удаленности точки от ρ-суммы понимается наибольший радиус шаблона с центром в этой точке, вписанный в I_ρ. Такой метод был проверен для выпуклых шаблонов и он сходится к глобальному решению задачи 1, если ограничение на размер растра не достигается, и к локальному, если достигается.

Таким образом, предложенный в статье метод является универсальной альтернативой имеющимся методам и может быть более эффективно реализован с использованием современных графических процессоров.

Список литературы

1. **Agarwal P. K., Amenta N., Aronov B., Sharir M.** Largest Placements and Motion Planning of a Convex Polygon // In Proc. 2nd Annual Workshop Algorithmic foundations of robotics. August 1996. Toulouse, France. 1997. P. 143—145.

2. **Koltun V., Sharir M.** Polyhedral Voronoi Diagrams of Polyhedra in Three Dimensions // In Proc. of the Eighteenth Annual Symposium on Computational Geometry. June 5—7 2002. Barcelona. Spain. ACM, 2002. P. 227—236.

3. **Hoff K. E., Culver T., Keyser J., Lin M., Manocha D.** Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware // In Proc. of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99). August 8—13 1999. Los Angeles, USA. N.-Y.: ACM Press, 1999. P. 277—286.

4. **Schwarz M., Seidel H.-P.** Fast Parallel Surface and Solid Voxelization on GPUs. // ACM Transactions on Graphics. 2010. Vol. 29, Is. 6. Article N 179.

5. **Половинкин Е. С., Балашов М. В.** Элементы выпуклого и сильно выпуклого анализа. М.: ФИЗМАТЛИТ, 2004.

6. **Kaufman A. E.** Voxels as a Computational Representation of Geometry. // In The Computational Representation of Geometry. SIGGRAPH Course Notes, 1994.

7. **Varadhan G., Manocha D.** Accurate minkowski sum approximation of polyhedral models // In Proc. of 12th Pacific Conference on Computer Graphics and Applications. October 6—8 2004. Seoul, Korea. IEEE Computer Society, 2004. P. 392—401.

УДК 004.5

Я. А. Туровский, канд. мед. наук, зав. лаб., Воронежский государственный университет,
e-mail: yaroslav_turovsk@mail.ru

Создание фильтров для анализа ЭЭГ-состояний на основе генетических алгоритмов

Рассмотрен генетический алгоритм создания специализированных фильтров для анализа одномерных биомедицинских сигналов. Предложено несколько способов реализации для каждого этапа работы этого алгоритма. Продемонстрирована возможность сравнения как средних модулей значений разности коэффициентов свертки с такими фильтрами двух наборов ЭЭГ-сигналов, соответствующих двум исследуемым состояниям, так и модулей максимальной разности значений коэффициентов свертки. Предложены подходы к усреднению фильтров в ходе их "скрещивания" во временной и частотной областях. Обсуждены преимущества и недостатки каждого из этих подходов. Для проведения "мутации" используют аддитивные синий, фликкер и/или белый шумы; а для "кроссинговера" — обмен фильтров временными последовательностями. Продемонстрировано успешное применение данного алгоритма определения двух состояний ЭЭГ, полученных в ходе экспериментов с системами человеко-машинного взаимодействия. Показано, что уже к 20-му поколению генерации фильтров различие коэффициентов свертки сигналов с селекционируемыми фильтрами существенно возрастает по сравнению с первыми поколениями. Предложенный подход позволяет значительно расширить возможности исследований при детекции различных функциональных состояний человека и животных.

Ключевые слова: *электроэнцефалограмма, фильтрация сигналов, генетические алгоритмы*

Ya. A. Turovsky

The Creating, on the Genetic Algorithm Base, Filters for EEG-conditions Analysis

The given article deals with the genetic algorithm of creating specialized filters for undimensional biomedical signals analysis. We offer several ways of action for each stage of the given algorithm's performance. We have shown the possibility of comparing both average modules of values of the difference between the coefficients of the convolution with such filters of two sets of EEG signals that correspond to the two explored states, and modules with a maximum difference of values of convolution coefficients. We offer two approaches of averaging the filters over the course of their "crossing": in the time and frequency domains. The advantages and disadvantages of each of these approaches are discussed. To carry out the "mutation", we apply noises: additive blue, flicker and/or white noise, and for "crossover" — the exchange of time sequences for the filters. There is demonstrated a successful implementation of the given algorithm for defining two states of EEG, received during the experiments with systems of human-machine interaction. We demonstrated that already by the 20th generation of filters the difference of coefficients of signals convolution with bred filters significantly increases compared to first generations. The offered approach lets drastically increase the research opportunities white detecting different functional states in humans and animals.

Keywords: *genetic algorithm, signal filtration, EEG*

Введение

Одним из важных элементов обработки одномерных биомедицинских сигналов является выделение определенных пространственно-временных паттернов, отражающих активность тех или иных органов или систем человека и животных. Имеющиеся пути решения этой задачи наиболее часто сводятся к оценке существующих различий в амплитудно-частотных феноменах в пространственной и/или во временной областях. Сравнение при этом, как правило, осуществляется на уровне групп наблюдений, без учета индивидуальных особенностей исследуемых объектов, см.

например [1, 2]. В качестве примера можно привести анализ электрокардиограммы (ЭКГ), по сути сводящийся к оценке значений амплитуд и ширин зубцов и сегментов или длительности интервалов на полученной кривой [3]. Похожий подход наблюдается при анализе эпилептической активности головного мозга [4], вспышек α-ритма при засыпании и т. п. [5]. Однако подобный подход содержит ряд ограничений. К их числу относится спектральное оценивание одномерных сигналов в заданных диапазонах частот, не учитывающее их динамических вариаций [6], что приводит к искажению получаемого значения мощности исследуемых сигналов. В то же время анализ информации во временном диапазоне далеко не всегда учитывает спектральные особенности исследуемых паттернов. Очевидным решением данных трудностей было бы использование адаптивных фильтров в различных вариантах, приспособленных для изучения тех или иных сигналов и их особенностей [7]. Нельзя не отметить, что конструирование и последующее применение адаптивных фильтров, меняющих свои свойства непосредственно в процессе регистрации и обработки сигнала, содержат потенциальные риски. Они связаны с тем обстоятельством, что динамические изменения параметров фильтров во время регистрации сигнала могут приводить к одной и той же интерпретации принципиально разных клинико-физиологических свойств, проявляющихся в характеристиках сигнала. Это, безусловно, сказывается на содержательной части проводимой интерпретации. Использование метода синтеза фильтров для анализа сигнала по принципу "соответствия образцу" с использованием ЭЭГ-сигналов в качестве образца подразумевает корректное выявление самих образцов, что далеко не всегда возможно [1, 2, 4—6]. Действительно, сигнал электроэнцефалограммы (ЭЭГ) представляет собой суперпозицию активности значительного числа генераторов электрического поля. Немалая часть из таких генераторов функционально не связана с теми нейрональными пулами, активность которых представляет интерес в рамках проводимого исследования. Следовательно, использовать фрагмент ЭЭГ в качестве "образца" без предварительной обработки — это значит получить мгновенный "слепок" электрической активности мозга, содержащий не только "полезный", относящийся к данному эксперименту или наблюдению сигнал. Такой слепок включает ряд шумов подчас весьма значительной амплитуды, содержащих не интересующую исследователя информацию, которую желательно устранить. Очевидно, что в норме и для большинства патологий, находящих отражение на ЭЭГ в виде широкого спектра амплитудно-частотных феноменов, данный подход не применим. Причина в отсутствии возможности или больших трудностях учета значительного числа дополнительных параметров электрогенеза мозга [4, 8]. Следовательно, метод создания анализирующих фильтров "по образцу" применим только тогда, когда исследуемый процесс в сигнале имеет ярко выраженную доминирующую амплитуду и слабо пересекается в частотном пространстве с иными процессами. Примером может служить гиперсинхронизированная ЭЭГ, возникающая при эпилептической активности, отражающей существование патологического доминирующего пула нейронов, активность которых регулирует электрогенез мозга [4]. Таким образом, представляется перспективной разработка методов конструирования оптимальных фильтров для анализа ЭЭГ, адаптированных к низкоамплитудным особенностям сигнала, для различных видов деятельности человека и животных.

Целью работы, описанной в статье, является создание метода построения фильтров на основе генетических алгоритмов, адаптированных к выявлению определенных паттернов медико-биологических сигналов и его программная реализация.

Генетические алгоритмы в создании фильтров для анализа ЭЭГ

Выбор именно генетических алгоритмов не случаен. Их использование для оптимизации поиска фильтра, позволяющего определять те или иные особенности сигнала (включая низкоамплитудные), имеет следующие преимущества по сравнению с конструированием фильтра "по образцу".

Итерационность. Алгоритм генерирует большое число поколений фильтров (в нашем случае — до 1000), что позволяет получить в итоге значительное число фильтров, обладающих необходимыми свойствами. Среди них, безусловно, можно более эффективным образом выбрать те, которые обладают нужными свойствами, чем это можно было бы сделать из относительно меньшего числа фильтров, которые получаются путем их создании "по образцу".

Скрещивание. Полученные фильтры обладают свойствами нескольких "предшественников", что позволяет комбинировать амплитудно-частотные особенности выявляемого паттерна из разных фильтров.

Конкурентность. Каждый из имеющихся фильтров данного поколения должен в ходе процесса свертки показать наиболее высокий результат для "своего" блока паттернов ЭЭГ и минимальный для иных паттернов с тем, чтобы он был допущенным "к размножению". Конкурентность позволяет за счет итерационности процесса отбора получать наиболее оптимальные фильтры из значительно большего числа вариантов, чем при "классическом" выборе анализирующей функции, например, для вейвлет-преобразования.

Детализируем заявленные выше преимущества, описав алгоритм создания фильтров.

Пусть имеется начальное поколение функций фильтров $Q_m^n(t)$, где верхним индексом n обозначен номер поколения, а нижним индексом m — номер фильтра внутри поколения. При этом сам фильтр имеет нулевое среднее и единичную норму.

Выберем один из каналов ЭЭГ и свернем фильтром $Q_m^n(t)$ зарегистрированный с него сигнал $E_x^k(t)$,

где нижний индекс x обозначает отведение (канал) в ЭЭГ-монтаже [4, 8], а k — номер одного из исследуемых состояний.

Проведем свертку сигнала $E_x^k(t)$ с фильтром $Q_m^n(t)$:

$$S_m^n(\tau, x, k) = \int Q_m^n(t) E_x^k(t - \tau) dt.$$

Здесь $S_m^n(\tau, x, k)$— функция, образованная коэффициентами свертки ЭЭГ-сигнала m-м фильтром n-го поколения.

Очевидно, что если имеется k состояний, которые нужно различить по l реализациям ЭЭГ для каждого состояния, и i фильтров, то общее число функций $S_m^n(\tau, x, k)$ будет равно $l \cdot i$ для каждого состояния.

Необходимо оценить, какие именно фильтры позволят наилучшим образом различить исследуемые состояния.

Рассмотрим для простоты только два состояния ($k = 2$) органа или системы органов. В этом случае можно последовательно и попарно сравнивать результаты свертки с каждым из фильтров каждой ЭЭГ-реализации и находить модули разности коэффициентов $S_m^n(\tau, x, k)$ для каждой пары сигналов из блоков "состояние 1" — "состояние 2" для каждого значения τ. В случае если $k > 2$, то попарно сравнивается одно состояние со всеми остальными. Таким образом, осуществляется селекция фильтров, направленная на выделение именно конкретного паттерна ЭЭГ.

Для анализа амплитуды $S_m^n(\tau, x, k)$ можно использовать два подхода. Первый подразумевает расчет среднего для всех модулей разности коэффициентов каждой сравниваемой пары ЭЭГ-сигналов, второй — выявление только максимальной точки модуля разности функций свертки, отражающей наиболее значимые различия между парами сигналов. Очевидно, что первый из них лучше подходит для проведения клинико-физиологических исследований квазистационарных состояний, таких как медленный сон, расслабленное состояние при закрытых глазах и т. д. Второй же подход, вероятно, наиболее удобен в тех случаях, когда имеется большое число стимулов разной природы, на которые мозг и, следовательно, электрогенез, отраженный в ЭЭГ, реагируют по-разному. Важно отметить, что параметр времени, соответствующий максимальному различию в значении свертки для двух и более состояний с одним и тем же фильтром, не участвует в дальнейшей селекции. Тем не менее его учет в виде оценки среднего, медианы, квантилей и доверительного интервала дает возможность при анализе записей ЭЭГ определять отрезки времени, содержащие для различных состояний точки максимальных различий функций $S_m^n(\tau, x, k)$. В ряде случаев такой подход позволяет усовершенствовать процесс синхронизации анализа ЭЭГ по активности пользователя.

Получив для каждого фильтра l сравнений для изучаемых состояний, можно рассчитать интегральную оценку эффективности этого фильтра. Как уже отмечалось выше, для разового сравнения — это максимальный (или средний) модуль разности коэффициентов $S_m^n(\tau, x, k)$ для каждой пары "состояние 1" — "состояние 2" и для каждой исследуемой реализации ЭЭГ-сигнала.

Просуммировав результаты, полученные по всем реализациям исследуемого процесса, нетрудно рассчитать показатель Φ^n, отражающий различие значений $S_m^n(\tau, x, k)$ для каждой пары состояний:

$$\Phi_{k_1, k_2}^n = \sum_{m=1}^{m_{\max}} \left| \overline{S_m^n(\tau, x, k_1) - S_m^n(\tau, x, k_2)} \right|, \qquad (1)$$

где k_1 и k_2 — номера различных состояний; $m_{\max}$ — максимальное число фильтров в поколении; черта сверху означает усреднение модулей разности коэффициентов каждой сравниваемой пары ЭЭГ-сигналов.

Возможен и другой способ расчета показателя Φ^n, основанный на учете только модуля максимальной разницы между функциями свертки, который относится к разным исследуемым состояниям:

$$\Phi_{k_1, k_2}^n = \sum_{m=1}^{m_{\max}} \left| S_m^n(\tau, x, k_1) - S_m^n(\tau, x, k_2) \right|_{\max}, \qquad (2)$$

где индекс max у знака модуля означает выбор из разности модулей только максимального значения для каждой сравниваемой пары функций свертки $S_m^n(\tau, x, k)$ из разных состояний.

После ранжирования значений Φ^n, полученных согласно (1) и (2), установим, какие именно фильтры позволяют выявить наибольшие различия между исследуемыми состояниями, т. е. такие, у которых значения Φ^n максимальны. Очевидно, что к дальнейшему "скрещиванию" должны быть допущены только те фильтры, которые дают наибольшие различия между состояниями.

Процесс "скрещивания" также реализуется в двух вариантах: в одном из них "скрещивание" происходит во временной области, в другом — в частотной области.

Итак, имея два разных фильтра $Q_{m1}^n(t)$ и $Q_{m2}^n(t)$, для поколения $n + 1$ получим новый фильтр:

$$Q_m^{n+1}(t) = (Q_{m1}^n(t) + Q_{m2}^n(t))/2.$$

Селекция в частотной области заключается в усреднении Фурье-образов фильтров $\hat{Q}_m^n(g)$:

$$Q_m^{n+1}(g) = 1/N \sum_{k=0}^{N-1} \mathbf{e}^{-i2\pi kg/N} ((\hat{Q}_{m1}^n(g) + \hat{Q}_{m2}^n(g))_k/2),$$

где g — порядковый номер отсчета в дискретном фильтре; N — общее число отсчетов в фильтре.

Очевидно, что реализация селекции путем преобразования Фурье более ресурсоемка по сравнению с суммированием фильтров во временной области. Однако

ее преимуществом является тот факт, что один и тот же Фурье-образ могут иметь несколько фильтров, имеющих во временном пространстве разную структуру. Следовательно, в том случае, когда важен именно сам спектр фильтра, а не последовательность его дискретных значений $Q_m^n(t)$, логичнее использовать решение, полученное путем использования преобразования Фурье. Важно отметить, что "скрещивание" фильтров происходит случайным образом среди тех из них, которые показали наилучшие результаты при осуществлении свертки со всей обучающей выборкой. Таким образом, получив и проранжировав значения Φ^n, оставляем только определенную долю (в наших экспериментах это 50 %) от популяции фильтров, которая затем порождает новые фильтры путем "скрещивания". При этом число новых фильтров составляет также 50 % от общей заданной численности популяции. Нетрудно заметить, что $(n + 1)$-е поколение на 50 % состоит из фильтров предыдущего поколения, демонстрирующих наилучшие показатели Φ^n, и на 50 % — из их потомков. Данный подход позволяет избежать ситуации, когда фильтр, дающий высокие показатели Φ^n в процессе "скрещивания", теряет свои свойства, а его потомки элиминируются из множества анализирующих функций. В предлагаемом варианте "удачный" фильтр будет сохраняться во всем ряду поколений.

Важно отметить, что сама по себе селекция возможна только в том случае, когда имеется различие значений фильтров внутри каждого поколения. Очевидно, если фильтры внутри одного поколения различаются незначительно, то "скрещивание" приведет к появлению в новом поколении похожих фильтров. В ходе применения "скрещивания" фильтров к сигналам с известными свойствами, такими как амплитудно-модулированные гармоники, было получено снижение дисперсии, начиная с 20-го поколения, а к 25—35-м поколениям все фильтры уже представляли собой практически идентичные последовательности.

Для устранения отмеченного выше недостатка было введено два механизма увеличения разнообразия фильтров, а именно "мутации" и "кроссинговер". "Мутация" представляет собой искусственно зашумленный фильтр, при этом использовался либо белый шум, либо цветные шумы — фликкер и/или синий. Вторым механизмом являлся сдвиг по фазе при усреднении фильтров, в ходе которого порождается новое поколение. Механизм "кроссинговера" заключается в том, что два случайным образом выбранных фильтра обмениваются фрагментами (рис. 1). При этом возможно появление "ступеньки" — резкого изменения значений функции $Q_m^n(t)$, порождающей высокочастотные колебания. Это связано с тем обстоятельством, что в ходе обмена фрагментами временных последовательностей фильтров (они отмечены на рис. 1 стрелками) последняя точка фрагмента одного фильтра, имеющая временную координату t, может существенно отличаться от соседней точки в момент времени $t + 1$, принадлежащей уже другому фильтру и оказавшейся с ней в дочернем фильтре в результате "кроссинговера".

Сигналом для начала "мутаций" или "кроссинговера" служит снижение коэффициента вариации ниже заданного процента от общего числа отсчетов во всем множестве фильтров поколения.

Рассмотрим пример. Пусть существуют 100 фильтров в одном поколении, в каждом из которых по 1000 отсчетов. Если в 20 % отсчетов, т. е. в 200 отсчетах (например, в 1-м для всех фильтров, 2-м для всех фильтров, 14-м для всех фильтров и т. д. пока не наберется 200 отсчетов), коэффициент вариации будет меньше 15 %, то запускается механизм создания искусственного разнообразия в виде "мутаций" и "кроссинговера".

Таким образом, формируется цикл, в ходе которого для каждого нового поколения фильтров рассчитывается коэффициент вариации (рис. 2).

Очевидно, что генетический алгоритм поиска наилучшего фильтра является, в целом, достаточно ресурсоемким. Для ускорения процесса расчета нового

Рис. 1. Схема "кроссинговера" фильтров. Числа в рамках — порядковые номера отсчетов в каждом из фильтров

Рис. 2. Схема работы генетического алгоритма селекции фильтров для медико-биологических исследований. Выносками отмечены события, при которых возможен выбор из нескольких вариантов реализации

поколения фильтров разработаны и реализованы алгоритмы для использования видеокарт по технологии CUDA. При этом обеспечивается возрастание скорости расчетов с увеличением длины анализируемых файлов.

Анализ ЭЭГ фильтрами, созданными с участием генетических алгоритмов

Рассмотрим реализацию генетического алгоритма поиска оптимального фильтра применительно к ЭЭГ-сигналам.

Регистрация ЭЭГ для построения фильтров осуществлялась в отведении Pz с частотой дискретизации 5 кГц. Рассматривалось обучение фильтров применительно к двум состояниям. В первом из них испытуемый следил за экраном, где в случайном порядке менялись цвета изображения, задаваемые по системе RGB. Во втором случае оценивался диапазон α-ритма, рассчитанный на основе непрерывного вейвлет-преобразования с использованием вейвлета Morlet [6]. Полученные значения с трех пар отведений ЭЭГ F3-F4, P3-P4, C3-C4 кодировали красный, зеленый и синий цвета соответственно. Полученные безартефактные ЭЭГ-записи разделялись на двухсекундные последовательности, которые и обрабатывались алгоритмом отбора фильтров.

На рис. 3 показано изменение логарифма среднего модуля разности коэффициентов свертки (*а*) и логарифма модуля максимальной разности коэффициентов свертки (*б*) в зависимости от поколения фильтров для исследуемых состояний.

На рис. 3 видно, что с 1-го по 20-е поколение разность сверток увеличивается, а в дальнейшем — ста-

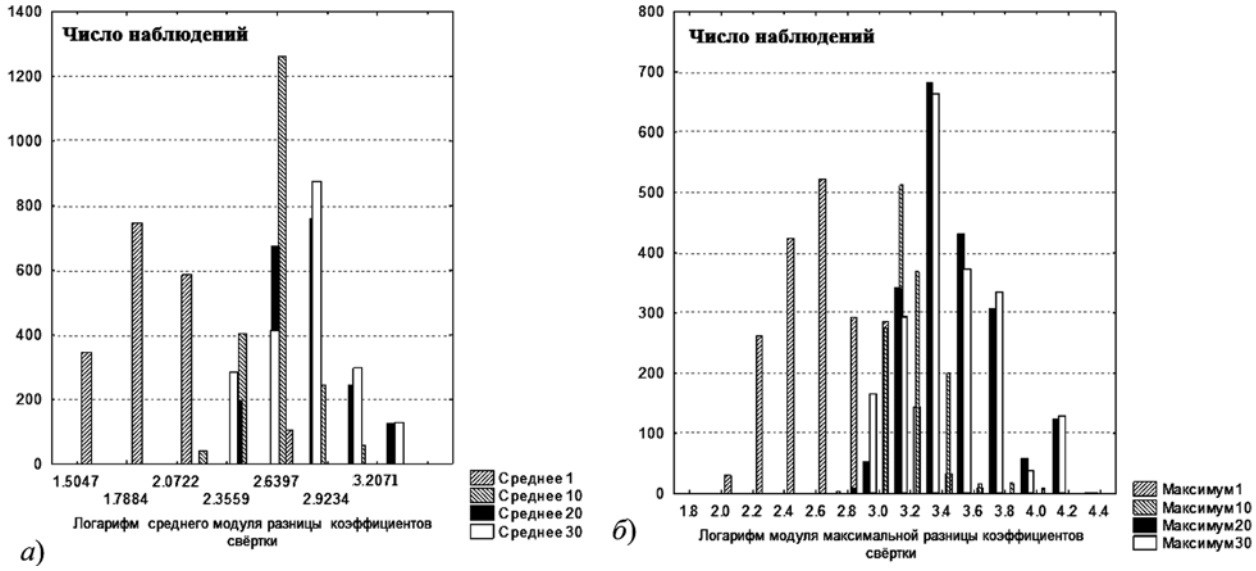

Рис. 3. Распределение значений модулей разности коэффициентов $S_m^n(\tau, x, k)$ в зависимости от поколения, к которому принадлежит фильтр. Представлены 1-е, 10-е, 20-е, 30-е поколения фильтров

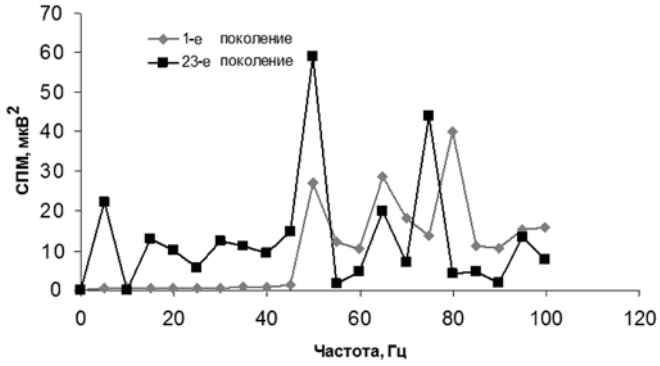

Рис. 4. Усредненные спектры (значения спектральной мощности) 1-го и 23-го поколений фильтров

билизируется и даже несколько уменьшается. Таким образом, можно констатировать, что поколение фильтров, дающих наибольшие различия между двумя состояниями, приходится примерно на 20-е поколение (фактически, на 23-е поколение, не показанное на рис. 3).

На рис. 4 представлены усредненные спектры для 1-го и 23-го поколений фильтров. Нетрудно заметить, что в диапазонах 4...9 Гц и 10...48 Гц поколения фильтров существенно отличаются друг от друга ($p < 0{,}01$, критерий Манна-Уитни с поправкой на эффект множественных сравнений). При этом для поколений, следующих за 23-м, несмотря на включенный механизм мутаций, коэффициент вариации также существенно снижается по сравнению с более ранними поколениями. В целом можно отметить, что возрастает энергия всех компонентов ЭЭГ за исключением нижнего диапазона α-ритма. По этому диапазону поколения фильтров практически не различаются, что наиболее вероятно связано с отсутствием различий в самих исследуемых состояниях. Таким образом, можно утверждать, что через 23 поколения удается получить популяцию фильтров, адаптированных для разделения двух экспериментальных состояний на ЭЭГ.

Заключение

В статье предложен генетический алгоритм создания специализированных фильтров для анализа одномерных медико-биологических сигналов. Продемонстрированы несколько способов обработки фильтров для каждого этапа работы этого алгоритма. Отмечена возможность сравнения как средних модулей значений разности коэффициентов свертки двух наборов ЭЭГ-сигналов, соответствующих двум же исследуемым состояниям, так и модулей максимальной разности значений коэффициентов свертки. Предложены варианты усреднения фильтров в ходе их "скрещивания" во временной и частотной областях. Представлены преимущества и недостатки каждого из этих подходов. Отмечено, что существенной трудностью при создании фильтров для анализа одномерных медико-биологических сигналов является значительное снижение дисперсии мгновенных амплитуд фильтров и, как следствие, прекращение их селекции еще до достижения оптимума в плане наилучшего разделения состояний. Таким образом, нахождение и использование не глобального, а лишь локального результата в виде фильтра, разделяющего два состояния, может существенно снизить качество полученных фильтров. Рассмотрены и применены два традиционно используемых механизма противодействия снижению эффективности генетических алгоритмов: "мутация" и "кроссинговер". Для проведения "мутации" предложено использовать аддитивный синий, фликкер и/или белый шумы, а в качестве "кроссинговера" — обмен фильтров временными последовательностями. Продемонстрировано применение данного алгоритма для разделения двух состояний ЭЭГ, полученных в ходе экспериментов с системами человеко-машинного взаимодействия. Показано, что уже к 20-му поколению фильтров различие коэффициентов свертки сигналов с селекционируемыми фильтрами значительно возрастает по сравнению с первыми поколениями. Таким образом, предложенный подход позволяет существенно расширить возможности как экспериментальных, так и клинических исследований при детекции различных функциональных состояний человека и животных.

Автор выражает глубокую признательность С. Д. Кургалину, В. А. Белобродскому, А. А. Вахтину, С. В. Борзунову за неоценимую помощь в проведении исследования.

Список литературы

1. **Алфимова М. В., Мельникова Т. С., Лапин И. А.** Использование когерентного анализа ЭЭГ и его реактивности на психофизиологические тесты при первом эпизоде у больных шизофренией // Журнал неврологии и психиатрии им. С. С. Корсакова. 2010. Т. 110, № 3. С. 97—102.

2. **Свидерская Н. Е.** Особенности пространственной организации ЭЭГ и психофизиологических характеристик человека при дивергентном и конвергентном типах мышления // Физиология человека. 2011. Т. 37, № 1. С. 36—44.

3. **Мурашко В. В., Струтынский А. В.** Электрокардиография. М.: МЕДпресс-информ, 2001. 324 с.

4. **Зенков Л. Р.** Клиническая электроэнцефалография (с элементами эпилептологии). М.: МЕДпресс-информ, 2011. 356 с.

5. **Дорохов В. Б.** Альфа-веретена и К-комплекс-фазические активационные паттерны при спонтанном восстановлении нарушений психомоторной деятельности на разных стадиях дремоты // Журнал высшей нервной деятельности 2003. Т. 53 (4). С. 503—512.

6. **Способ** исследования электроэнцефалограммы человека и животных. Я. А. Туровский, С. А. Запрягаев, С. Д. Кургалин. Патент РФ RU 2332160.

7. **Адаптивные** фильтры: пер. с англ. / Под ред. К. Ф. Н. Коуэна и П. М. Гранта. М.: Мир, 1988. 392 с.

8. **Гнездицкий В. В.** Обратная задача ЭЭГ и клиническая электроэнцефалография. М.: МЕДпресс-информ, 2004. 626 с.

УДК 534.1

О. А. Иванова, канд. физ.-мат. наук, асс., e-mail: ivanovaolgaal@mail.ru,
И. К. Марчевский, канд. физ.-мат. наук, доц.,
Московский государственный технический университет имени Н. Э. Баумана

Моделирование нестационарных аэроупругих колебаний провода ЛЭП с использованием возможностей многопроцессорных вычислительных комплексов

Рассмотрена задача о моделировании высокоамплитудных колебаний провода воздушной линии электропередачи — пляски. Предполагается, что провод покрыт несимметричной наледью. Аэродинамические нагрузки, действующие на сечения провода, вычисляются в ходе прямого моделирования их нестационарного обтекания бессеточным методом вязких вихревых доменов. Для проведения эффективного численного анализа рассмотренной задачи разработан параллельный программный комплекс.

Ключевые слова: *математическое моделирование, воздушная линия электропередачи, пляска, аэродинамические нагрузки, метод вязких вихревых доменов, параллельный алгоритм*

O. A. Ivanova, I. K. Marchevsky

Modeling of the Unsteady Aeroelastic Oscillations of the Transmission Line Conductor Using Capabilities of Multiprocessor Computer Systems

The problem of mathematical modeling of overhead transmission line conductor galloping motion is considered. The conductor is assumed to be covered with asymmetrical icing. The aerodynamic loads acting on the conductor cross-sections are calculated through direct numerical simulation of the flow around them using meshfree viscous vortex domains method. In order to carry out effective numerical analysis of the problem the parallel computer program was developed.

Keywords: *mathematical modeling, overhead transmission line, galloping, aerodynamic loads, viscous vortex domains method, parallel algorithm*

Введение

Устойчивый поперечный ветер, воздействующий на провода воздушных линий электропередачи (ЛЭП), может приводить к пляске (галопированию) — высокоамплитудным низкочастотным колебаниям преимущественно в вертикальной плоскости [1]. Пляске наиболее подвержены провода, покрытые несимметричной наледью. Высокие динамические нагрузки, действующие на провода, опоры и арматуру ЛЭП при пляске, могут всего за несколько часов привести к повреждениям линии и большим материальным затратам на проведение ремонтных работ. Поэтому разработка программного комплекса, позволяющего в ходе вычислительного эксперимента анализировать характеристики колебаний линии электропередачи, является актуальной задачей.

Ключевой задачей при проведении численного анализа колебаний провода ЛЭП является определение действующих на него распределенных аэродинамических нагрузок. Принято считать, что воздушный поток направлен перпендикулярно плоскости начального провисания провода и обтекание его поперечных сечений является плоскопараллельным [1, 2]. Во всех известных работах при вычислении аэродинамических нагрузок, действующих на сечения провода, принимается гипотеза об их квазистационарности, т. е. предполагается, что в каждый момент времени обте-

кание каждого сечения является установившимся. В пользу целесообразности применения гипотезы о квазистационарном характере нагрузок говорит тот факт, что период колебаний провода по низшей собственной форме в десятки-сотни раз превышает период схода вихрей с его сечений [3]. Сравнение результатов расчета колебаний упругозакрепленного жесткого цилиндра по квазистационарным нагрузкам с результатами эксперимента, приведенное в работе [2], показало их хорошее согласие. Однако повторить с приемлемой точностью результаты эксперимента [4] по исследованию колебаний модели расщепленной фазы путем расчета по квазистационарным нагрузкам не удалось. Это обстоятельство свидетельствует об ограниченности области применимости квазистационарного подхода.

Альтернативой квазистационарному подходу к определению аэродинамических нагрузок, действующих на сечения провода, является прямое моделирование их нестационарного обтекания [5]. Для этого целесообразно применять бессеточные вихревые (лагранжевы) методы, позволяющие моделировать течение вокруг движущегося тела и определять действующие на него аэродинамические нагрузки с приемлемой точностью и разумными затратами вычислительных ресурсов. В данной работе описана модифицированная расчетная схема [6] метода вязких вихревых доменов [7].

Для сокращения времени проведения расчетов предусмотрено распараллеливание вычислений на двух уровнях. Во-первых, на каждом шаге расчета по времени задачи моделирования обтекания сечений провода являются независимыми. Во-вторых, могут быть распараллелены наиболее трудоемкие операции каждой из задач расчета обтекания сечений.

1. Постановка задачи и математическая модель

Введем декартову прямоугольную систему координат как показано на рис. 1. Ускорение свободного падения направлено противоположно оси Ox_3; точки крепления концов провода в положении равновесия имеют координаты $(-X_{10}, 0, -X_{30})$ и $(X_{10}, 0, X_{30})$, т. е. середина отрезка, соединяющего концы провода, совпадает с началом координат. Тогда в положении равновесия под действием силы тяжести провод целиком лежит в плоскости Ox_1x_3. Воздушный поток направлен вдоль оси Ox_2 и имеет постоянную скорость $\mathbf{V}_\infty$.

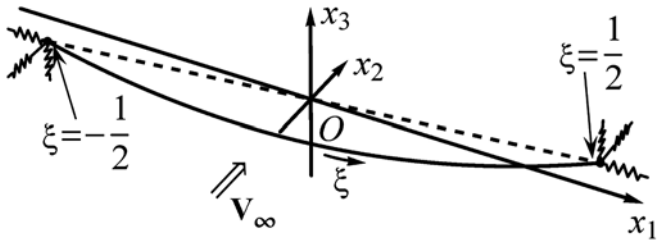

Рис. 1. Расчетная схема:
ξ — безразмерная дуговая координата на проводе

1.1. Математическая модель движения провода

В основу математической модели провода положим уравнения движения растяжимой нити — абсолютно гибкого стержня, сопротивляющегося только растяжению [8]. Их необходимо дополнить уравнением для изменения угла поворота сечений θ, а также учесть слагаемые, возникающие вследствие несовпадения центра масс сечения G с центром голого провода C. Будем считать деформацию провода малой, а материал провода — линейно упругим, подчиняющимся закону Гука при растяжении и аналогичному линейному соотношению для кручения. Предположим при этом, что наличие обледенения на проводе не влияет на его сопротивление растяжению и кручению. Примем также, что аэродинамическая нагрузка в направлении оси Ox_1 (вдоль провода) отсутствует. Тогда движение провода будет описываться следующей нелинейной системой дифференциальных уравнений в частных производных:

$$\left(\frac{Q}{1+Q/EF}x_1'\right)' + C_1 - \ddot{x}_1 = 0;$$

$$\left(\frac{Q}{1+Q/EF}x_2'\right)' + C_2 + \left(1+\frac{Q}{EF}\right)q_2^a - \ddot{x}_2 - h\left(\sin(\theta_s+\theta)\ddot{\theta} + \cos(\theta_s+\theta)\dot{\theta}^2\right) = 0;$$

$$-\left(\frac{Q}{1+Q/EF}x_3'\right)' + C_3 + \left(1+\frac{Q}{EF}\right)q_3^a - 1 - \ddot{x}_3 - h\left(\cos(\theta_s+\theta)\ddot{\theta} - \sin(\theta_s+\theta)\dot{\theta}^2\right) = 0;$$

$$(x_1')^2 + (x_2')^2 + (x_3')^2 = (1+Q/EF)^2;$$

$$GJ\theta'' + C_\theta + M^a - \ddot{\theta} - h\beta(\sin(\theta_s+\theta)\ddot{x}_2 + \cos(\theta_s+\theta)\ddot{x}_3 + \cos(\theta_s+\theta)) = 0.$$

Здесь и далее при моделировании колебаний провода все величины являются безразмерными: величины размерности длины, массы и силы отнесены к длине $\tilde{L}$ нерастянутого провода, его массе и весу соответственно; время отнесено к $\sqrt{\tilde{L}/\tilde{g}}$ ($\tilde{g}$ — ускорение свободного падения); знак тильда указывает на то, что соответствующая величина является размерной. Параметр $\xi \in [-1/2; 1/2]$ — дуговая координата (натуральный параметр) на проводе; τ — безразмерное время; производные по ξ и τ обозначены штрихом и точкой соответственно; $x_j(\xi, \tau)$ — декартовы координаты оси провода — линии, проходящей через центры его сечений без обледенения; $\theta(\xi, \tau)$ — угол поворота сечений вокруг оси; $Q(\xi, \tau)$ — тяжение провода; EF и GJ —

жесткости на растяжение и кручение соответственно (считается, что наличие обледенения не влияет на их значения); $q_i^a(\xi, \tau)$, $i = 2, 3$ и $M^a(\xi, \tau)$ — аэродинамические силы и момент в расчете на единицу длины провода; $C_j(\xi, \tau)$, $j = 1, 2, 3$, $C_\theta(\xi, \tau)$ — функции, определяющие демпфирование; $\theta_s(\xi, \tau) = \theta_e(\xi, \tau) + \theta_G$, $\theta_e(\xi, \tau)$ — угол поворота сечений провода в ненапряженном состоянии; β — параметр, характеризующий инерционные свойства провода. Сечения провода считаются перпендикулярными оси, а положение оси в конкретном сечении обозначается точкой C. Положение центра масс G сечения провода с обледенением задается с помощью расстояния $h = CG$ и угла θ_G, отсчитываемого по часовой стрелке от хорды сечения в положении, соответствующем нулевому углу атаки.

В качестве начального условия (при $\tau = \tau_0$) задается равновесное положение провода в отсутствие ветра $x_{10}(\xi)$, $x_{20}(\xi) \equiv 0$, $x_{30}(\xi)$, $\theta_0(\xi)$:

$$x_j(\xi, \tau_0) = x_{j0}(\xi), \left.\frac{\partial x_j(\xi, \tau)}{\partial \tau}\right|_{\tau_0} = 0 \ (j = 1, 2, 3),$$

$$\theta(\xi, \tau_0) = \theta_0(\xi), \left.\frac{\partial \theta(\xi, \tau)}{\partial \tau}\right|_{\tau_0} = 0.$$

Тяжение провода $Q(\xi, \tau_0) = Q_0(\xi)$ в равновесном положении определяется координатами $x_{10}(\xi)$, $x_{20}(\xi)$, $x_{30}(\xi)$. Граничные условия имеют вид

$$x_i\left(\pm\frac{1}{2}, \tau\right) - x_{i0}\left(\pm\frac{1}{2}\right) =$$

$$= \mp S_i^{\pm}\left(\frac{Qx_i'}{1 + Q/EF} - \frac{Q_0 x_{i0}'}{1 + Q_0/EF}\right)\Bigg|_{\xi = \pm 1/2},$$

$$\theta\left(\pm\frac{1}{2}, \tau\right) = 0,$$

т. е. концы провода закреплены с помощью линейных безынерционных пружин с податливостями $S_j^{\pm}$, $j = 1, 2, 3$, где, как правило, $S_3^{\pm} = 0$.

1.2. Математическая модель обтекания сечений провода потоком

Рассматривается плоская задача о расчете обтекания подвижного жесткого телесного профиля произвольной формы вязкой несжимаемой средой. Все характеристики течения зависят от времени и декартовых координат x_2, x_3 и не зависят от x_1.

В данной задаче величины размерности длины отнесены к хорде профиля $\tilde{d}$; величины размерности скорости — к скорости набегающего потока $\tilde{V}_\infty$; время отнесено к $\tilde{d}/\tilde{V}_\infty$, давление — к $\tilde{\rho}\,\tilde{V}_\infty^2$ ($\tilde{\rho}$ = const — плотность среды).

Движение среды описывается уравнениями неразрывности и Навье—Стокса:

$$\nabla \cdot \mathbf{V} = 0, \ \frac{\partial \mathbf{V}}{\partial \tau} = \boldsymbol{\Omega} \times \mathbf{V} = -\nabla\left(p + \frac{\mathbf{V}^2}{2}\right) + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{V},$$

где Re — число Рейнольдса; $\mathbf{V}$ — скорость среды; $\boldsymbol{\Omega} = \nabla \cdot \mathbf{V}$ — завихренность; p — давление; оператор ∇ означает дифференцирование по безразмерным пространственным координатам x_2 и x_3.

На границе профиля ∂K задано условие прилипания

$$\mathbf{V}(\mathbf{r}) = \mathbf{V}_K(\mathbf{r}), \ \mathbf{r} \in \partial K, \qquad (1)$$

на бесконечном удалении от профиля задано условие затухания возмущений

$$\mathbf{V}(\mathbf{r}) \to \mathbf{V}_\infty = \mathrm{const}, \ p(\mathbf{r}) \to p_\infty = \mathrm{const}, \ |\mathbf{r}| \to \infty,$$

Здесь $\mathbf{r} = x_2\mathbf{e}_2 + x_3\mathbf{e}_3$ — радиус-вектор точки пространства, векторы $\mathbf{e}_2$ и $\mathbf{e}_3$ — орты координатных осей x_2 и x_3 соответственно.

Для описания движения жесткого профиля на нем выбирается точка C с радиус-вектором $\mathbf{r}_C(\tau) = x_{C2}(\tau)\mathbf{e}_2 + x_{C3}(\tau)\mathbf{e}_3$, тогда положение профиля характеризуется вектором $\mathbf{r}_C$ и углом поворота $\theta(\tau)$ профиля вокруг точки C, отсчитываемым по часовой стрелке. Скорость произвольной точки $\mathbf{r}$ профиля равна

$$\mathbf{V}_K(\mathbf{r}, \tau) = \mathbf{V}_C(\tau) - \boldsymbol{\omega} \times (\mathbf{r} - \mathbf{r}_C),$$

где $\mathbf{V}_C(\tau) = \dot{\mathbf{r}}_C = d\mathbf{r}_C/d\tau$ — скорость выделенной точки C; $\boldsymbol{\omega} = \omega\mathbf{e}_1$; $\omega = \dot{\theta}$ — угловая скорость профиля; $\mathbf{e}_1$ — орт координатной оси x_1. Профиль может двигаться по заданному закону либо под действием аэродинамических нагрузок и наложенных на него связей.

Суть вихревых методов состоит в описании движения среды и профилей через завихренность — "реальную" ($\boldsymbol{\Omega}$) в области среды и "сопряженную" ($\boldsymbol{\Omega}_{in}$) в области профиля. В однородной несжимаемой среде завихренность $\boldsymbol{\Omega}$ может генерироваться только на границах области течения, т. е. на границе профиля. Уничтожение завихренности происходит вследствие взаимной диффузии завихренности противоположного знака и имеет место внутри вязкой среды [9]. Закон эволюции $\boldsymbol{\Omega}$ является следствием уравнений Навье—Стокса: вихревая линия движется со скоростью, складывающейся из скорости среды $\mathbf{V}$ (конвективной скорости) и диффузионной скорости $\mathbf{V}_d$, пропорциональной вязкости среды [7]. Распределение сопряженной завихренности $\boldsymbol{\Omega}_{in}$ в области, занятой профилем, может быть задано множеством способов [10]; в данной работе $\boldsymbol{\Omega}_{in} = \nabla \times \mathbf{V}_K = -2\boldsymbol{\omega} = -2\omega\mathbf{e}_1$.

Связь между плоским соленоидальным векторным полем $\mathbf{V}$, таким что $\mathbf{V} \to \mathbf{V}_\infty$ при $|\mathbf{r}| \to \infty$, и его ротором $\mathbf{\Omega}$ задается законом Био—Савара:

$$\mathbf{V}(\mathbf{r}) = \iint\limits_{R^2} (\mathbf{\Omega}(\xi) \times \mathbf{K}(\mathbf{r} - \xi)) dS_\xi + \mathbf{V}_\infty;$$
$$\mathbf{K}(\mathbf{r} - \xi) = \frac{1}{2\pi} \frac{(\mathbf{r} - \xi)}{|\mathbf{r} - \xi|^2}. \quad (2)$$

Под полями $\mathbf{V}$ и $\mathbf{\Omega}$ здесь понимаются поля $\mathbf{V} \cup \mathbf{V}_K$ и $\mathbf{\Omega} \cup \mathbf{\Omega}_{in}$. Для наглядности разобьем интеграл в (2) на две части:

$$\mathbf{V}(\mathbf{r}) = \mathbf{V}_\Omega(\mathbf{r}) + \mathbf{V}_f(\mathbf{r}) + \mathbf{V}_\infty, \quad (3)$$

где $\mathbf{V}_\Omega(\mathbf{r}) = \iint\limits_{R^2/K} (\mathbf{\Omega} \times \mathbf{K}) dS_\xi$, $\mathbf{V}_f(\mathbf{r}) = \iint\limits_K (\mathbf{\Omega}_{in} \times \mathbf{K}) dS_\xi$.

Действующая на профиль гидродинамическая сила и момент гидродинамических сил относительно фиксированной точки C вычисляются на основе интегральных представлений, полученных в работе [7]. В соответствии с методом вязких вихревых доменов [7] влияние вязкости среды моделируется перемещением завихренности по траекториям поля $(\mathbf{V} + \mathbf{V}_d)$, где $\mathbf{V}_d$ — диффузионная скорость:

$$\mathbf{V}_d = -\frac{1}{\mathrm{Re}} \frac{\nabla \Omega}{\Omega}, \; \Omega = \mathbf{\Omega} \mathbf{e}_1.$$

2. Алгоритм решения задачи

Рассмотрим численные методы, используемые для решения задачи о расчете движения провода и серии задач о моделировании обтекания его сечений потоком несжимаемой среды, а также опишем алгоритм решения общей сопряженной задачи аэроупругости.

2.1. Расчет движения провода

Для моделирования движения провода используется метод Бубнова—Галеркина; неизвестные функции x_1, x_2, x_3, Q и θ, зависящие от ξ и τ, представляются в виде линейных комбинаций функций, зависящих от ξ, с коэффициентами, зависящими от τ:

$$x_j(\xi, \tau) = x_{j0}(\xi, \tau) + \sum_{k=1}^{S_j} a_k^{(j)}(\tau) \varphi_k^{(j)}(\xi), \; j = 1, 2, 3;$$
$$Q(\xi, \tau) = Q_0(\xi) + \sum_{k=1}^{S_Q} a_k^{(Q)}(\tau) \varphi_k^{(Q)}(\xi);$$
$$\theta(\xi, \tau) = \theta_0(\xi) + \sum_{k=1}^{S_\theta} a_k^{(\theta)}(\tau) \varphi_k^{(\theta)}(\xi),$$

где φ_k^*, $k = 1, \ldots, S_*$ (символ * заменяет индексы 1, 2, 3, Q или θ) — ортонормированные системы базисных функций. В качестве базисных функций для координат провода x_j целесообразно выбрать первые $(S_j - 2)$ собственные формы малых свободных колебаний провода, определяемые, например, методом сагиттарной функции [11], и две дополнительные функции для корректного учета нелинейных граничных условий; аналогично для тяжения Q. Для угла поворота θ в качестве базисных выбирают функции $\varphi_k^{(\theta)}(\xi) = \sqrt{2} \sin \pi k(\xi + 1/2)$, близкие к собственным формам малых крутильных колебаний.

Функции $C_j(\xi, \tau)$; $j = 1, 2, 3$; $C_\theta(\xi, \tau)$, определяющие демпфирование, задаются в виде

$$C_j(\xi, \tau) = -2\varsigma_j \sum_{k=1}^{S_j - 2} \omega_k^{(j)} \frac{da_k^{(j)}(\tau)}{d\tau} \varphi_k^{(j)}(\xi),$$
$$C_\theta(\xi, \tau) = -2\varsigma_\theta \sum_{k=1}^{S_\theta - 2} \omega_k^{(\theta)} \frac{da_k^{(\theta)}(\tau)}{d\tau} \varphi_k^{(\theta)}(\xi),$$

где ω_k — собственные частоты малых колебаний; ς_j, ς_θ — коэффициенты демпфирования.

Подставляя разложение решения по базисным функциям в уравнения движения провода и вычитая из них уравнения равновесия, получаем невязку, требование ортогональности которой базисным функциям приводит к системе $(S_1 + S_2 + S_3 + S_\theta - 6)$ обыкновенных дифференциальных уравнений; из граничных условий получаем $(S_Q + 6)$ алгебраических уравнений. Получившаяся система дифференциально-алгебраических уравнений решается с помощью полунеявной схемы, имеющей при $h = 0$ второй порядок точности.

2.2. Расчет обтекания сечений провода

В расчетах непрерывное поле завихренности заменяется набором из $N_{\text{ВЭ}}$ вихревых элементов — вихревых нитей, характеризуемых положениями $\mathbf{r}_i$ и циркуляциями Γ_i:

$$\hat{\mathbf{\Omega}}(\mathbf{r}) = \sum_{i=1}^{N_{\text{ВЭ}}} \Gamma_i \delta(\mathbf{r} - \mathbf{r}_i), \quad (4)$$

где $\delta(\mathbf{r} - \mathbf{r}_i)$ — двумерная дельта-функция. Граница обтекаемого профиля аппроксимируется ломаной, состоящей из M отрезков-панелей.

Вычисление скоростей среды и вихревых элементов. Подставляя (4) в (3), получим

$$\mathbf{V}_\Omega(\mathbf{r}) = \sum_{i=1}^{N_{\text{ВЭ}}} \mathbf{V}_i(\mathbf{r}) =$$
$$= \sum_{i=1}^{N_{\text{ВЭ}}} \Gamma_i (\mathbf{e}_1 \times \mathbf{K}(\mathbf{r} - \mathbf{r}_i)) = \sum_{i=1}^{N_{\text{ВЭ}}} \Gamma_i \mathbf{Q}(\mathbf{r} - \mathbf{r}_i).$$

При приближении к вихревому элементу скорость, индуцируемая им, неограниченно возрастает, поэтому вводится малый радиус вихревых элементов ε и

влияние i-го вихревого элемента $\mathbf{V}_i$ вычисляется по формуле

$$\mathbf{V}_i(\mathbf{r}) = \Gamma_i \hat{\mathbf{Q}}(\mathbf{r} - \mathbf{r}_i),\ \hat{\mathbf{Q}}(\mathbf{r} - \mathbf{r}_i) = \\ = \frac{1}{2\pi}\mathbf{e}_1 \times \frac{\mathbf{r} - \mathbf{r}_i}{\max\{\varepsilon^2, |\mathbf{r} - \mathbf{r}_i|^2\}}.$$

Составляющая $\mathbf{V}_f$ скорости среды в точке $\mathbf{r}$, вызванная вращением профиля, вычисляется с помощью преобразования интеграла по площади в (3) в контурный интеграл [7]:

$$\mathbf{V}_f(\mathbf{r}) = \iint_K (\boldsymbol{\Omega}_{in} \times \mathbf{K}) dS_\xi \approx -\frac{\omega}{\pi} \sum_{k=1}^{M} \boldsymbol{\tau}_k \ln|\mathbf{r} - \hat{\mathbf{r}}_k| \Delta l_k,$$

где $\hat{\mathbf{r}}_k$ и $\boldsymbol{\tau}_k$ — радиус-вектор середины k-й панели и орт касательной к ней соответственно; Δl_k — длина k-й панели.

Для вычисления диффузионной скорости $\mathbf{V}_d$ используют формулы, основанные на интегральном представлении завихренности $\boldsymbol{\Omega}$ [7]; диффузионная скорость i-го вихревого элемента определяется влиянием всех остальных элементов, а также границы профиля.

Генерация завихренности. Можно показать, что если на границе профиля отсутствует присоединенная завихренность, то условие прилипания (1) эквивалентно условию [12]

$$(\mathbf{V}_-(\mathbf{r}) - \mathbf{V}_K(\mathbf{r})) \cdot \boldsymbol{\tau}(\mathbf{r}) = 0,\ \mathbf{r} \in \partial K, \qquad (5)$$

где $\mathbf{V}_-$ — предельное значение скорости среды со стороны профиля. За малый временной шаг $\Delta\tau$ вблизи контура образуется тонкий слой завихренности. Пренебрегая толщиной этого слоя, будем приближенно считать его сосредоточенным на границе профиля ∂K. Тогда $\mathbf{V}_-(\mathbf{r}) = \mathbf{V}(\mathbf{r}) - \gamma(\mathbf{r})\boldsymbol{\tau}(\mathbf{r})/2$, где $\gamma(\mathbf{r})$ — интенсивность вихревого слоя.

Условие (5) приводит к интегральному уравнению Фредгольма второго рода (в случае гладкого профиля K) относительно искомой интенсивности вихревого слоя на профиле:

$$\boldsymbol{\tau}(\mathbf{r}) \cdot \oint_{\partial k} \gamma(\xi)\mathbf{Q}(\mathbf{r} - \xi) dl_\xi - \frac{\gamma(\mathbf{r})}{2} = \\ = \boldsymbol{\tau}(\mathbf{r}) \cdot (\mathbf{V}_K(\mathbf{r}) - \mathbf{V}_\infty - \mathbf{V}_\Omega(\mathbf{r}) - \mathbf{V}_f(\mathbf{r})),\ \mathbf{r} \in \partial K. \qquad (6)$$

Ядро уравнения (6) ограничено $\kappa^*/(4\pi)$, где κ^* — наибольшая кривизна профиля; его решение неединственно, оно выделяется при помощи дополнительного условия

$$\oint_{\partial k} \gamma(\xi) dl_\xi = 2S\Delta\omega - \Delta G, \qquad (7)$$

где $\Delta\omega$ — приращение угловой скорости профиля за шаг по времени $\Delta\tau$, S — площадь профиля; смысл параметра ΔG будет пояснен ниже. Интегральное уравнение (6) и условие (7) аппроксимируются системой линейных алгебраических уравнений; при этом обеспечивается выполнение уравнения (6) в среднем по панелям, а вихревой слой кусочно-постоянной интенсивности считается распределенным вдоль панелей. После нахождения интенсивности вихревого слоя на каждой панели он "стягивается" в отдельные вихревые элементы, которые затем пополняют вихревой след. Расчетные формулы для вычисления коэффициентов матрицы системы линейных уравнений и правых частей приведены в работе [13].

Эволюция поля завихренности моделируется перемещением вихревых элементов в соответствии с системой обыкновенных дифференциальных уравнений

$$\frac{d\mathbf{r}_i}{d\tau} = \mathbf{V}(\mathbf{r}_i) + \mathbf{V}_d(\mathbf{r}_i),\ i = 1, \ldots, N_{\text{ВЭ}}. \qquad (8)$$

Вследствие дискретности численной схемы некоторые вихревые элементы могут оказываться внутри профиля. Чтобы этого избежать, проводится проверка и все такие вихри удаляют. При этом запоминают их циркуляции, которые затем учитывают при коррекции аэродинамических нагрузок [7]; сумма циркуляций удаленных вихрей ΔG входит в уравнение (6) на следующем шаге расчета.

Также на каждом шаге проводится объединение ("коллапс") вихревых элементов в один, если расстояние между ними меньше некоторой (малой) длины ε_{col}. Это позволяет моделировать аннигиляцию вихрей разного знака и приводит к уменьшению числа вихрей в следе и, следовательно, сокращению времени расчета. Кроме того, из вихревого следа исключают вихри, удалившиеся от профиля на (большое) расстояние — за границу моделирования вихревого следа L_{far} (в расчетах, как правило, принимается $L_{far}/b = 5...20$, где b — хорда профиля), поскольку влияние таких вихревых элементов на течение вблизи профиля и действующие на него аэродинамические нагрузки обычно несущественно.

Реализация метода вихревых элементов для сопряженной задачи обтекания профиля предполагает необходимость выполнения на каждом шаге расчета по времени следующих операций.

Шаг 1. Определение циркуляций вихревых элементов, рожденных на границе профиля, путем решения соответствующей системы линейных алгебраических уравнений и вычисление аэродинамических нагрузок, действующих на профиль.

Шаг 2. Определение нового положения профиля.

Шаг 3. Вычисление новых положений вихревых элементов путем интегрирования уравнений их движения (8) явным методом Эйлера.

Шаг 4. Проверка попадания вихревых элементов внутрь профиля, объединение ("коллапс") близкорасположенных вихрей; удаление "дальнего" следа.

Отметим, что вычисление правых частей системы (8) — наиболее трудоемкая операция алгоритма, тре-

бующая выполнения $O(N_{\text{ВЭ}}^2)$ операций. Для сокращения времени исполнения данной операции могут быть использованы параллельные вычислительные технологии. В рамках описываемой работы применяется технология MPI [14, 15].

Моделирование обтекания профиля осуществляется параллельно m процессорами, из которых один является головным. Этот процессор проводит расчет обтекания профиля в соответствии с вышеуказанным алгоритмом, управляя при выполнении шага 3 распараллеливанием вычислений по следующей схеме:

1) головной процессор пересылает остальным (подчиненным) процессорам массив вихревых элементов и равномерно распределяет между ними вихревые элементы, для которых необходимо определить конвективную $\mathbf{V}$ и диффузионную $\mathbf{V}_d$ скорости;

2) все процессоры вычисляют скорости вихревых элементов;

3) скорости вихревых элементов пересылаются на головной процессор, который затем вычисляет их новые положения.

Распараллеливание операции реструктуризации вихревого следа (шаг 4 алгоритма) проводится аналогично. Возможны и иные подходы к ускорению вычислений, например, использование быстрого метода расчета вихревого влияния [16, 17]. Однако при решении данной задачи их применение не приводит к существенному повышению скорости вычислений в силу сравнительно небольшого числа вихревых элементов в расчете. Обычно число вихревых элементов в каждой задаче обтекания сечения провода составляет несколько тысяч, тогда как быстрый метод становится эффективным при $N_{\text{ВЭ}} \geqslant 10\,000$.

2.3. Общий алгоритм

Алгоритм расчета движения провода с вычислением аэродинамических нагрузок, действующих на его сечения, методом вихревых элементов, реализован в виде программного комплекса PROVOD, схема работы которого изложена ниже.

Блок 1. Главный головной процессор (ГГП) осуществляет загрузку всех необходимых исходных данных для расчета из текстовых файлов.

Пересылка 1. ГГП пересылает всем локальным головным процессорам (ЛГП) необходимые им параметры расчета, загруженные из текстовых файлов.

Блок 2. Подготовка расчетных схем на профилях: с учетом начального положения сечений определяются координаты контрольных точек и точек рождения вихрей, а также касательных и нормальных векторов к панелям.

Блок 3. Каждым ЛГП определяются циркуляции вихревых элементов (ВЭ), образующихся на "своем" профиле на данном шаге расчета по времени. Этим обеспечивается выполнение граничных условий на профилях, поэтому на следующем этапе расчета могут быть корректно вычислены аэродинамические нагрузки, действующие на профили.

Рис. 2. Иллюстрация метода определения нестационарных аэродинамических нагрузок, действующих на провод

Блок 4. Вычисляются аэродинамические нагрузки на каждое сечение (рис. 2).

Блок 5. Вычисляются скорости ВЭ. Этот блок выполняется каждой подгруппой в параллельном режиме. Затем проводится интегрирование уравнений движения ВЭ методом Эйлера на одном шаге расчета по времени.

Пересылка 2. Все ЛГП пересылают значения аэродинамических нагрузок на ГГП.

Блок 6. ГГП приводит аэродинамические нагрузки к безразмерному виду, как это требуется для решения уравнений движения провода, и интерполирует их. Далее осуществляется решение системы уравнений движения провода методом Ньютона.

Пересылка 3. ГГП пересылает всем ЛГП значения переменных $a_k^{(*)}$ и их производных в следующий момент времени (символ * заменяет индекс 1, 2, 3, Q или θ).

Блок 7. ЛГП проводят вычисления положений сечений на новом шаге расчета по времени и перестроение расчетных схем на профилях.

Блок 8. ЛГП проводят реструктуризацию вихревых следов за сечениями: удаляют ВЭ, попавшие внутрь профиля, проводят "коллапс" ВЭ и удаление дальнего следа.

Затем происходит переход на новый шаг расчета по времени и возврат к **Блоку 3**.

Программный комплекс также позволяет определять зависимости стационарных аэродинамических характеристик профиля сечения провода от угла атаки ($C_{xa}(\alpha)$, $C_{ya}(\alpha)$, $C_m(\alpha)$), моделировать движение одного упругозакрепленного профиля и проводить квазистационарные расчеты.

Поскольку проведение нестационарного расчета с применением метода вихревых элементов требует значительных вычислительных затрат, а для развития пляски провода ЛЭП, первоначально находившегося в положении равновесия, и достижения им периодической траектории иногда необходимо длительное время (порядка десятков-сотен секунд), предлагается проводить расчет в несколько стадий.

1. Построение зависимостей стационарных аэродинамических характеристик профиля сечения провода от угла атаки. Данный этап является трудоемким

с вычислительной точки зрения. Целесообразно проводить его на многопроцессорной ЭВМ.

2. Проведение расчета по квазистационарным нагрузкам до достижения проводом периодической траектории движения.

3. Проведение расчета, в котором провод движется по заданному закону — участку полученной на предыдущем этапе периодической траектории, с генерацией вихревых элементов и моделированием эволюции вихревых следов за сечениями. Данный этап необходим для формирования за сечениями провода развитого вихревого следа.

4. Проведение расчета по нестационарным аэродинамическим нагрузкам до заданного конечного момента времени.

3. Результаты расчетов

В данном разделе приведены результаты расчетов, проведенных с помощью программного комплекса PROVOD.

3.1. Верификация программного комплекса PROVOD

В работе [18] описаны результаты наблюдения пляски центрального пролета экспериментальной трехпролетной ЛЭП, на котором имелась насадка *U*-образной формы. В работе [2] приведены полученные в эксперименте зависимости стационарных аэродинамических коэффициентов профиля сечения провода от угла атаки и результаты моделирования пляски. Размерным параметрам задачи соответствуют безразмерные параметры $EF = 6909$, $GJ = 742{,}5$, $h = 6{,}5 \cdot 10^{-6}$, $\beta = 6{,}3 \cdot 10^{8}$, горизонтальная компонента тяжения $Q_{hor} = 7{,}44$; угол поворота сечений в ненапряженном состоянии $\theta_e = 141^\circ$.

Расчет, проведенный по квазистационарным нагрузкам с помощью разработанного программного комплекса, показал, что размах колебаний составляет около 2,3 м. На рис. 3 приведены вертикальное и угловое отклонения центральной точки пролета от положения равновесия в зависимости от времени. Видно, что результаты, полученные в данной работе, хорошо согласуются с расчетом, приведенным в работе [2], и удовлетворительно согласуются с экспериментом, описанным в работе [18].

3.2. Вычисление стационарных аэродинамических коэффициентов

Поскольку пляске чаще всего подвержены провода ЛЭП с несимметричным обледенением, был проведен расчет стационарных аэродинамических коэффициентов профиля обледенелого провода, рассмотренного в работах [19, 20] (рис. 4, *а*). Безразмерные параметры расчета выбирались следующим образом: радиус вихревого элемента $\varepsilon = 0{,}008$, радиус коллапса $\varepsilon_{col} = 0{,}002$, скорость потока $\mathbf{V}_\infty = 1$, шаг расчета $\Delta\tau = 0{,}004$, число Рейнольдса $\mathrm{Re} = 1000$. Профиль моделировался с помощью 245 отрезков-панелей. На рис. 4, *б* приведены зависимости стационарных аэродинамических коэффициентов рассмотренного профиля, полученные путем осреднения их мгновенных значений по 7000 шагам расчета. Расчет выполнялся для углов атаки в диапазоне от -40 до 50° с шагом в 4°.

Данный профиль и полученные для него зависимости стационарных аэродинамических коэффициентов от угла атаки будут использованы далее при моделировании колебаний упругозакрепленного профиля и провода ЛЭП целиком.

3.3. Расчет движения упругозакрепленного профиля

При моделировании пляски упругозакрепленного профиля обледенелого провода ЛЭП, рассмотренного выше, безразмерные параметры расчета его динамики выбирались следующими: жесткости пружин $k_2 = 0{,}0116$, $k_3 = 0{,}0169$, $k_\theta = 2{,}06$, коэффициенты демпфирования $\varsigma_2 = \varsigma_3 = 0{,}001$, $\varsigma_\theta = 0{,}01$. Положение центра масс профиля характеризовалось расстоянием $h = CG = 0{,}0054$ и углом $\theta_G = 27^\circ$.

Сначала проводился квазистационарный расчет, в результате которого была получена периодическая траектория движения профиля с амплитудой вертикальных колебаний около 0,9 м. Далее как продолжение квазистационарного расчета был проведен нестационарный расчет с теми же параметрами, что и в предыдущем примере. Расчет проводился на Учебно-экспериментальном вычислительном кластере кафедры "Прикладная математика" МГТУ им. Н. Э. Баумана [14].

На рис. 4, *в* показаны траектория нестационарного движения профиля и периодическая траектория, полученная в ходе проведения квазистационарного рас-

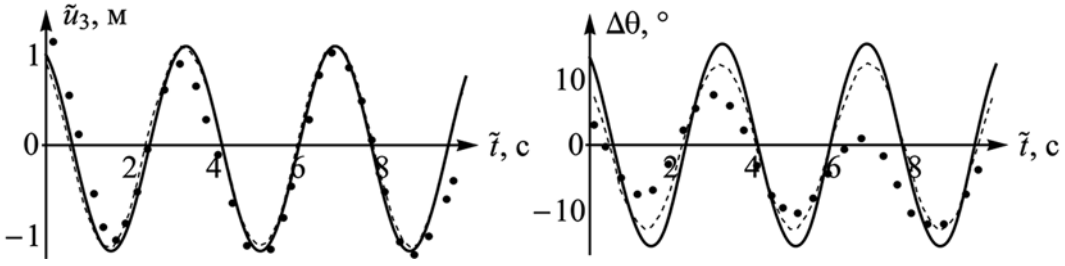

Рис. 3. Зависимости вертикального $\tilde{u}_3$ и углового Δθ перемещений центра пролета от времени: точки — эксперимент [18], штриховая линия — расчет [2], сплошная линия — данная работа

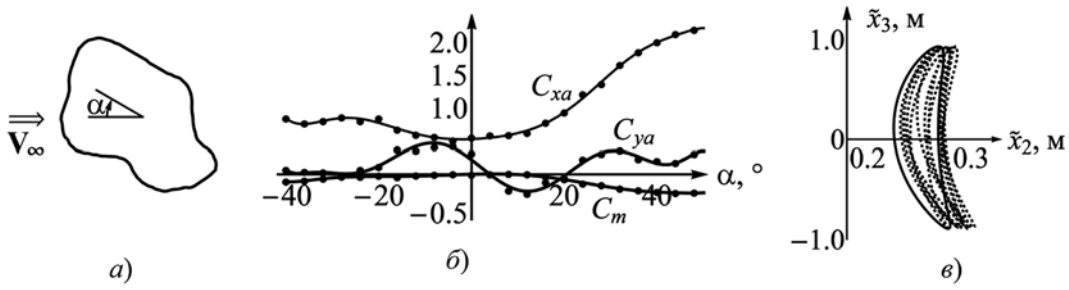

Рис. 4. Зависимости C_{xa}, C_{ya}, C_m от угла атаки α профиля обледенелого провода (*а*), аппроксимированные гладкими функциями (*б*); *в* — траектория движения профиля: результат квазистационарного (сплошная линия) и нестационарного (пунктир) расчета

чета. Видно, что траектории движения профиля при использовании двух рассмотренных подходов к вычислению действующих на него аэродинамических нагрузок практически не различаются.

В таблице приведены значения ускорения расчета — отношения времени счета t_1 на одном процессоре к времени параллельного расчета с использованием m процессоров t_m при различных значениях числа задействованных процессоров m.

Ускорение вычислений

Число процессоров m	1	2	4	8
Ускорение t_1/t_m	1,00	1,78	3,07	4,78

3.4. Моделирование движения провода

Ниже приведены результаты моделирования нестационарного движения провода ЛЭП с поперечным сечением, рассмотренным в подразд. 3.2, по алгоритму, изложенному в подразд. 2.3.

Безразмерная жесткость на растяжение провода равна $EF = 6906$; безразмерное тяжение $Q_{hor} = 7{,}44$; угол поворота сечений в ненапряженном состоянии выбирался равным $\theta_e = -46°$, тогда в результате действия силы тяжести угол поворота центрального сечения в положении равновесия был близок к 7°. Податливости пружин на концах провода составляли $S_1^{\pm} = 0{,}00096$, $S_2^{\pm} = 0{,}0085$. В базис включалось по одной собственной форме малых колебаний по каждому направлению, поскольку расчеты по стационарным нагрузкам показали [11], что увеличение числа базисных функций не приводит к изменению характеристик движения провода.

Сначала был проведен расчет по стационарным аэродинамическим нагрузкам до момента времени $\tilde{t}^* = 180$ с, затем в продолжение этого расчета был проведен расчет с вычислением нестационарных аэродинамических нагрузок методом вихревых элементов. Число сечений на проводе было выбрано равным $N = 16$, обтекание каждого из них моделировалось в параллельном режиме с помощью $m = 8$ процессоров. Таким образом, нестационарный расчет потребовал использования 128 вычислительных ядер кластера МВС-100К (Межведомственный суперкомпьютерный центр РАН) и занял около 5 сут. Ускорение по сравнению с проведением этого расчета в последовательном режиме составило примерно 65 раз.

На рис. 5 приведены зависимости вертикальной координаты $\tilde{x}_3$ и угла поворота θ центральной точки пролета от времени. Видно, что амплитуда вертикальных колебаний провода, полученная в результате нестационарного расчета, снижается на 12 % по сравнению с квазистационарным расчетом и составляет в итоге 0,65 м. Амплитуда угловых колебаний остается практически неизменной и составляет около 9°. При этом можно отметить, что на угловые колебания, согласованные с вертикальным движением провода, накладываются его собственные крутильные колебания с высокой частотой.

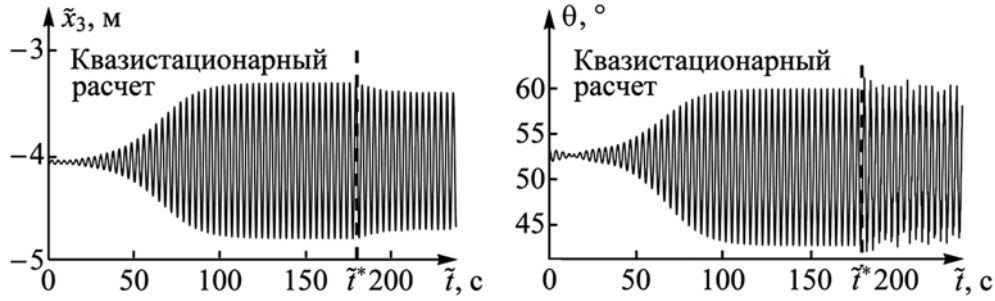

Рис. 5. Зависимость вертикальной координаты $\tilde{x}_3$ (*а*) и угла поворота θ (*б*) центрального сечения провода от времени

Заключение

Предложен алгоритм моделирования аэроупругих колебаний провода под действием нестационарных аэродинамических нагрузок. На его основе разработан параллельный программный комплекс PROVOD, в котором расчет нестационарного обтекания сечений провода и вычисление аэродинамических нагрузок проводятся методом вихревых элементов. Разработанные модели, алгоритмы и программы могут быть использованы при исследовании движения провода в существенно нестационарных условиях.

Работа выполнена при поддержке гранта Президента РФ МК-3705.2014.8.

Авторы благодарят Межведомственный суперкомпьютерный центр РАН за предоставленную им возможность использовать кластер МВС-100К.

Список литературы

1. **EPRI.** Transmission line reference book: wind-induced conductor motion. Palo Alto (California): Electrical Power Research Institute, 1979. 255 p.

2. **Keutgen R.** Galloping phenomena: a finite element approach: Ph. D. Thesis. Liege (Belgium), 1998. 202 p.

3. **Blevins R. D.** Flow-induced vibration. N.-Y.: Van Nostrand reinhold, 1990. 451 p.

4. **Waris M. B., Ishihara T., Sarwar M. W.** Galloping response prediction of ice-accreted transmission lines // The 4th International Conference on Advances in Wind and Structures: book of proceedings. Jeju, 2008. P. 876—885.

5. **Иванова О. А.** Математическое моделирование аэроупругих колебаний провода линии электропередачи: дис. ... канд. физ.-мат. наук. М., 2013. 142 с.

6. **Moreva V. S., Marchevsky I. K.** Vortex element method for 2D flow simulation with tangent velocity components on airfoil surface // ECCOMAS 2012 — European Congress on Computational Methods in Applied Sciences and Engineering, e-Book Full Papers. 2012. P. 5952—5965.

7. **Андронов П. Р., Гувернюк С. В., Дынникова Г. Я.** Вихревые методы расчета нестационарных гидродинамических нагрузок. М.: Изд-во МГУ, 2006. 184 с.

8. **Светлицкий В. А**. Механика абсолютно гибких стержней / под ред. А. Ю. Ишлинского. М.: Изд-во МАИ, 2001. 432 с.

9. **Morton B. R.** The generation and decay of vorticity // Geophys. Astrophys. Fluid Dynamics. 1984. V. 28. P. 277—308.

10. **Сэффмен Ф. Дж.** Динамика вихрей. М.: Научный мир, 2000. 375 с.

11. **Иванова О. А.** О выборе базиса для моделирования движения провода ЛЭП методом Галеркина // Наука и образование. 2013. № 9. С. 243—250.

12. **Accuracy** considerations for implementing velocity boundary conditions in vorticity formulations / S. N. Kempka et al. SANDIA Rep. SAND96-0583 UC-700, 1996. 50 p.

13. **Морева В. С.** Вычисление вихревого влияния в модифицированной численной схеме метода вихревых элементов // Вестник МГТУ Н. Э. Баумана. Сер.: Естественные науки. 2012. Спец. вып. № 2 "Математическое моделирование в технике". С. 137—144.

14. **Лукин В. В., Марчевский И. К., Морева В. С., Попов А. Ю., Шаповалов К. Л., Щеглов Г. А.** Учебно-экспериментальный вычислительный кластер. Ч. 2. Примеры решения задач // Вестник МГТУ Н. Э. Баумана. Сер.: Естественные науки. 2012. № 4. С. 82—102.

15. **Марчевский И. К., Щеглов Г. А.** Применение параллельных алгоритмов при решении задач гидродинамики методом вихревых элементов // Вычислительные технологии и программирование. 2010. Т. 11. С. 105—110.

16. **Кузьмина К. С., Марчевский И. К.** Оценка трудоемкости быстрого метода расчета вихревого влияния в методе вихревых элементов // Наука и образование. 2013. № 10. С. 399—414.

17. **Марчевский И. К., Щеглов Г. А.** Моделирование динамики вихревых структур высокопроизводительным методом вихревых элементов // Известия высших учебных заведений. Машиностроение. 2013. № 9. С. 26—36.

18. **Chan J. K.** Modelling of single conductor galloping. Canadian Electrical Association Report 321-T-672A. Montreal, 1992. 180 p.

19. **Лукьянова В. Н.** Разработка методов расчета абсолютно гибких стержней (проводов) при обледенении и нестационарных колебаниях: дис. ... канд. техн. наук. М., 1987. 240 с.

20. **Марчевский И. К.** Математическое моделирование обтекания профиля и исследование его устойчивости в потоке по Ляпунову: дис. ... канд. физ.-мат. наук. М., 2008. 119 с.

ИНФОРМАЦИЯ

В № 1, 2014 г. в статье "Разрешение неоднозначности авторства публикаций при автоматической обработке библиографических данных" в фамилии одного из автоов была допущена опечатка. Следует читать **А. Э. Гаспарянц**. Приносим свои извинения.

УДК 929:004

Р. И. Подловченко, д-р физ.-мат. наук, проф., вед. науч. сотр.,
НИВЦ МГУ имени М. В. Ломоносова, e-mail: podlovchenko.rimma@gmail.com

Истоки российского программирования глазами очевидца*

Отражены годы становления программирования в нашей стране через призму ощущений непосредственного очевидца и участника событий тех лет.

Ключевые слова: *вычислительная математика, программирование, теория схем программ, кибернетика*

R. I. Podlovchenko

The Origins of Russian Software Engineering in the Eyes of the Witness

This paper is a story of the formation of software engineering in Russia in the light of contemporary and participant of the bygone events.

Keywords: *computational mathematics, software engineering, theory of program schemata, cybernetics*

Можно сказать, что автор, родившаяся в далеком 1931 г., принадлежит к уже "вымирающему" племени людей, чья профессиональная деятельность в программировании начиналась в годы его становления в нашей стране. Эти годы, по мнению автора, не нашли должного отражения в литературе. В связи с этим интересно проследить вехи становления программирования через призму ощущений непосредственного очевидца и участника событий тех лет. Чтобы лучше понять личное восприятие этих событий, начну с себя, с полученного мною воспитания в семье.

Итак, коротко о моих родителях.

Мой отец был выходцем из безграмотной крестьянской семьи, переселившейся, по призыву царского правительства, из бесплодных земель Белоруссии в благодатные земли Башкирии. Возможности, предоставленные советской властью, позволили ему окончить педагогический институт в г. Уфа и посвятить себя педагогической работе. Он занимал должность директора в Талдомской средней школе и в Расторгуевской школе № 6, одновременно проводя уроки по истории и конституции СССР.

Отец моей матери, будучи известным портным в своем городе Белебей, Башкирия, владел начальной грамотностью, а его жена так и не научилась читать и писать. Отец считал блажью давать образование выше начального своим дочерям, и моей маме пришлось пробивать дорогу к образованию вопреки желаниям родителя. Помогла бытовавшая в те годы установка советской власти на всеобщее образование. В результате мама тоже окончила педагогический институт в г. Уфа, но по естественным наукам, и стала преподавателем химии и биологии в средней школе. Понятно, что в семье родителей культом служила идея обязательности высшего образования для детей.

Война застала нашу семью, в которой было четверо детей, в поселке Расторгуево, который находится в 23 км от Москвы. Первого июля 1941 г. отец в свои 40 лет записался в народное ополчение и стал проходить обучение военному делу. Начавшиеся бомбежки Москвы побудили его отправить семью в эвакуацию. Он как и многие считал, что война окончится через пару месяцев и семья воссоединится. В начале августа 1941 г. (во время массового бегства из Москвы) отцу с трудом удалось втиснуть нас в поезд, отправлявшийся на восток.

* Статья публикуется в редакции автора.

Четверо детей в возрасте от двух до десяти лет (я была старшей) плюс минимальный набор носильных вещей составили наше богатство, как оказалось, на многие годы.

В эвакуации мы жили в доме маминых родителей в г. Белебей. Мама сразу устроилась на работу учительницей, я пошла в четвертый класс, старший из моих братьев начал учиться в первом классе, два других брата коротали дни дома. Так и жили, во всяком случае, не голодая, почти год.

В феврале 1942 г. отец погиб в боях под Ржевом, позднее приобретших известность своей кровопролитностью, он уже был в регулярных действующих войсках, куда влили народное ополчение. Получив похоронку, мама тут же начала хлопотать о разрешении вернуться на прежнее место жительства, т. е. в Расторгуево. Основной причиной этого решения была боязнь, что в таком захолустье, как Белебей, трудно рассчитывать на то, что удастся дать детям достойное высшее образование.

Разрешение на возвращение было получено, и к осени 1942 г. мы вернулись в нашу старую квартиру. Она была полностью разворована, отопление, естественно, не работало, что оказалось весьма существенным при отсутствии у нас необходимой одежды. Это послужило основой нашей постоянной борьбы с холодом. К тому же на протяжении всех военных лет мы страшно голодали, поскольку к маминой зарплате учителя и весьма скромной пенсии за папу добавить было нечего, у нас ничего не было для обмена на продукты, что практиковалось в те годы в семьях, подобных нашей. Летом я работала в соседнем совхозе, небольшая, но все-таки подмога семье. Одного из моих братьев мы потеряли в этих лишениях.

Школа, в которой работала мама, а мы учились, была хорошей по составу учителей. Согласно общему настрою в семье я училась с должным прилежанием и всегда была отличницей. Моим классным руководителем была Нина Михайловна Часовникова, преподававшая русский язык и литературу. Она в свое время окончила гимназию и сумела превратить в праздники уроки литературы, организовала драматический кружок в школе, непременным членом которого была и я. Она направила меня в Московский дом детского творчества осваивать азы декламаторского искусства, и я их постигала под руководством двух дам — Розы Соломоновны и Иды Исааковны. И хотя обучение шло на материале, не слишком для меня интересном, определенные навыки грамотного и выразительного чтения стихов и отрывков прозы я получила. Итак, все складывалось в пользу выбора филологии в качестве будущей профессии.

Однако внезапно мой настрой изменился. В старших классах математику стал преподавать Геннадий Михайлович Сухарев, демобилизованный из армии, а в прошлом преподаватель вуза. Он заметил у меня способности к математике и стал их развивать посредством индивидуальных заданий. Благодаря этому выбор будущей профессии пал на математику.

Нельзя не упомянуть преподавателя немецкого языка Марию Федоровну Лесковец, урожденную немку, чудом не подвергнутую высылке в начале войны. Строгость Марии Федоровны на уроках заставляла буквально трепетать учеников. Ко мне она прониклась симпатией как к усердной ученице и стала заниматься со мной немецким языком у себя дома. Эти уроки сочетались с подкармливанием меня перед ними, так как по своей доброте она не могла не учитывать моего всегдашнего полуголодного состояния. Под ее руководством я освоила немецкий язык как разговорный (жаль, что потом у меня не было практики в его использовании), на выпускном экзамене я прочитала известное стихотворение Гейне "Lorelei", тем самым мне посчастливилось продемонстрировать ее педагогический дар.

В школе я впервые получила практику преподавания: проводила уроки русского языка в параллельном классе, где учились ребята из соседних деревень.

Закончив в 1948 г. школу с золотой медалью, я отправилась поступать на механико-математический факультет Московского государственного университета и была принята после предписываемого собеседования с одним из членов приемной комиссии.

Вживание в новую для меня атмосферу не прошло безболезненно: обнаружилось, что я любила лишь подступы к настоящей математике — таким огромным был разрыв между школьной математикой и той, что преподносилась в лекциях на нашем факультете в Московском государственном университете. "Акклиматизировалась" я благодаря великолепному составу преподавателей. С неизменной благодарностью вспоминаю тех, кто читал нам лекции. Выдающимся преподавателем был Александр Яковлевич Хинчин, который читал курс по математическому анализу: он демонстрировал искусство речи, которому, как говорили, обучался по примеру Демосфена, практикуясь с камушком во рту. А Александр Геннадиевич Курош так вел лекции по высшей алгебре, что мы просто видели, не обращаясь к доске, векторы и матрицы, о которых он говорил. Сергей Владимирович Бахвалов поражал тем, что после с вдохновением прочитанной

лекции по аналитической геометрии даже карманы его пиджака были усыпаны мелом. Практические занятия по математическому анализу вела Зоя Михайловна Кишкина, она строго проверяла, как мы прорешали заданные на дом примеры, число которых исчислялось десятками, и сопровождала проверку нетривиальными комментариями; мы горячо ее любили. Неизгладимое впечатление оставили и другие менторы.

К концу второго курса обучения нужно было выбрать кафедру, на которой следовало специализироваться дальше. Это был 1950 г., а на факультете в 1949 г. была учреждена новая кафедра — кафедра вычислительной математики. Ее открытие было данью времени: сначала за рубежом, а затем и у нас для решения различных задач теории и практики стали применять быстродействующие вычислительные машины, а это сделало актуальным развитие методов вычислительной математики.

Под влиянием хотя скупых, но завораживающих сведений о новых вычислительных машинах, создаваемых за рубежом, я выбрала именно эту кафедру. Первый набор на кафедру состоял всего из тринадцати человек, привожу их список: Гера Артамонов, Кирилл Багриновский, Сергей Ломов, Сусанна Каменомостская, Кариженский (не помню его имени), Спартак Разумовский, Нонна Тархнишвили, Анна Фалетова и я, кроме того, проучившиеся на ней всего один год Валентин Волков, Саша Комаров, Виктор Ломакин и Михаил Шабунин. Последних перевели на другие, так называемые "закнопочные" кафедры, вход на территорию которых был ограничен.

Заведующим кафедрой вычислительной математики был назначен профессор Борис Михайлович Щиголев, специалист в области небесной механики, который занимался и вычислительной математикой. Он-то и стал читать нам курс по вычислительным методам. В преподавательский состав кафедры входил, кроме него, Николай Петрович Жидков, остальные были совместителями. И это неудивительно, поскольку план обучения студентов кафедры исходил из намерения готовить специалистов, которые параллельно с методами решения задач на грядущей вычислительной технике владели бы на инженерном уровне тонкостями ее устройства. Поэтому в план были включены годовые курсы электротехники и радиотехники, семестровые курсы импульсной техники, теории механизмов и машин, начертательной геометрии, сведения о счетных механизмах и приборах, и разумеется, черчение. Вместе с тем практические занятия по методам вычислений проходили на допотопных машинах "Феликс", не было даже машин "Мерседес".

Полученные нами знания практически не потребовались в будущем, но ведь все эти предметы вытеснили целый ряд дисциплин, необходимых для полноценного математического образования, таких как теория функций действительного переменного, функциональный анализ, спецглавы уравнений математической физики. Уместно заметить, что в то время теория алгоритмов и математическая логика еще не преподавались на механико-математическом факультете. Определенная обделенность математическими знаниями в будущем остро ощущалась выпускниками нашей кафедры.

Малочисленность нашей группы слегка расхолаживала лекторов. Например, профессор Делоне чуть ли не половину своих лекций посвятил основной теореме планиметрии, а К. А. Семендяев весь лекционный час, стоя к нам спиной, рисовал схемы коммутационных досок без необходимых содержательных пояснений. И слушатели имели время для развлечений, пытаясь определить, где же находится у Щиголева вторая нога, если всю лекцию он стоит на одной.

На третьем курсе за отличную учебу и активное участие в комсомольской жизни факультета мне назначили сталинскую стипендию, и в статусе сталинского стипендиата я продержалась до конца обучения. Сталинская стипендия, 780 руб. в месяц, из коих, правда, в принудительном порядке 100 руб. уходили на обязательную подписку на какие-то облигации, существенно улучшила материальное положение нашей семьи. Должна заметить, что участие в комсомольской жизни, необходимое для получения этой стипендии, отнюдь не было для меня тягостным, я верила в полезность этой работы.

Программа обучения на кафедре вычислительной математики не придавала ей популярности, желающих поступить на нее было так мало, что руководство факультета в 1951 г. решило зачислять студентов на кафедру в принудительном порядке. Это произошло, когда мы перешли на четвертый курс, а наша группа сократилась до девяти человек. Среди зачисленных на нашу кафедру были Андрей Ершов, Саша Любимский, Сева Штаркман, Оля Кулагина, Игорь Задыхайло, чьи имена оказались потом на слуху у работающих в области программирования, но были и те, кто приложил усилия по переходу на другую кафедру.

Принципиальным образом ситуация на кафедре поменялась с нашим переходом на пятый курс, т. е. в 1952 г. Заведовать кафедрой стал академик Сергей Львович Соболев, и с его именем связано становление кафедры [1]. В первую очередь был вы-

работан новый учебный план и для его выполнения к преподаванию привлечены известные специалисты. В числе первых были Алексей Андреевич Ляпунов, Лазарь Аронович Люстерник, а со временем в жизни кафедры приняли участие Мстислав Всеволодович Келдыш, Михаил Романович Шура-Бура, Иван Семенович Березин и многие другие, список которых есть в работе [1], в разделе "Страницы истории факультета". Кафедрой были заложены основы целого ряда научных направлений.

В 1954 г. при кафедре был создан отдел вычислительных машин, руководство которым было поручено И. С. Березину. Отдел был оснащен лишь счетно-перфорационными, настольными клавишными и аналоговыми машинами, и основной его задачей было сопровождение учебного процесса. А в 1955 г. по инициативе С. Л. Соболева и при поддержке ректора МГУ академика И. Г. Петровского на базе этого отдела был создан Вычислительный центр, заведовать которым стал И. С. Березин. Началось бурное оснащение его современной вычислительной техникой [2].

Этот краткий экскурс в историю завершу заключением: усилиями С. Л. Соболева кафедра вычислительной математики приобрела чрезвычайно весомый авторитет.

Вернусь к 1952 г. Тогда особенно ярко в жизни кафедры проявилось преподавание на ней Алексея Андреевича Ляпунова. Именно в 1952/53 учебном году Алексей Андреевич прочел курс из восьми лекций под названием "Принципы программирования", явившийся первым в отечестве курсом по программированию. В процессе преподавания у лектора сформировалась система понятий, положенная в основу его операторного метода. Фактически этим курсом лекций было провозглашено, что программирование — это самостоятельная область знания. Он был опубликован только годы спустя [3], хотя сразу стал руководством для всех, кто занимался программированием.

Обращение к данному предмету не было случайным: в отечестве появились быстродействующие электронные вычислительные машины собственного производства. Огромным толчком к пониманию основных задач программирования стали для Алексея Андреевича поездка в Феофанию под Киевом и личное знакомство с первой отечественной ЭВМ МЭСМ, сконструированной Сергеем Алексеевичем Лебедевым. Следом за созданием МЭСМ Лебедев, уже в Институте точной механики и вычислительной техники АН СССР, находящемся в Москве, приступил к разработке БЭСМ. Тут уже Алексей Андреевич был вынужден вникнуть в то, как функционирует эта машина, не дожидаясь окончания работ над ней, для чего он однажды провел полную ночную смену вместе с программистами, работавшими в машинном зале. Об этом эпизоде расскажу подробнее дальше.

Размышляя над процессом программирования на этих машинах, Алексей Андреевич в качестве первоочередной поставил задачу его автоматизации путем использования языка описания программ, предложенного им в курсе "Принципы программирования" и названного операторным. В сущности, он был первым алгоритмическим языком программирования, определенным не формально, а на содержательном уровне. Составление программы по алгоритму, записанному на операторном языке, должно быть реализовано так называемой программирующей программой — прообразом будущих трансляторов с алгоритмических языков программирования. Задачу построения программирующей программы Алексей Андреевич разбил на подзадачи, на входе каждой из которых были блоки исходного алгоритма, и поручил решение таких подзадач студентам четвертого курса. В частности, Андрею Ершову была поручена работа с арифметическим блоком. Так начался профессиональный путь будущих известных создателей трансляторов, Ершова и Любимского, именно с использования операторного языка.

Оценив перспективы, открытые быстродействующими счетными машинами, Алексей Андреевич поставил задачу автоматического перевода с одного языка человеческого общения на другой. Перевод с французского языка на русский составил задачу, предложенную Оле Кулагиной, а возникшие при этом трудности положили начало новому направлению в науке — математической лингвистике. О. С. Кулагина стала впоследствии одним из самых известных специалистов в этой области.

Однако все это прошло мимо нашей группы, поскольку общение с кафедрой свелось к минимуму. Согласно учебному плану, в первом семестре пятого года обучения мы должны были пройти производственную практику, а второй посвятить написанию дипломных работ.

На практику нас направили в Институт точной механики и вычислительной техники (ИТМиВТ), где мы должны были получить навыки в программировании на БЭСМ — большой электронной счетной машине, которая там разрабатывалась. Директором института тогда был академик Михаил Алексеевич Лаврентьев, а конструктором БЭСМ — Сергей Алексеевич Лебедев. Сама машина создавалась на наших глазах. Так, придя в машинный зал, занимавший почти весь первый этаж здания, мы увидели там штабеля ртутных трубок, используе-

мых ранее как память машины и уступивших эту роль электронным лампам. И далее на наших глазах постоянно шло усовершенствование техники, обеспечивающей основное назначение машины.

Мы пришли в отдел, занимающийся программированием. Моим куратором стал Владимир Михайлович Курочкин, защитивший кандидатскую диссертацию по алгебре и переключившийся на новое направление в науке. Кроме него в отделе была единственная сотрудница-программист Александра Ивановна Срагович, окончившая пару лет тому назад механико-математический факультет МГУ, и очень скоро ставшая для нас просто Шурочкой. Задания мне давал Курочкин, и эта процедура вначале была необычной: не помня имени практиканта, он предварительно забегал за поворот коридора, где заглядывал в записную книжицу, а выходя обращался непременно по имени-отчеству. Так меня называли впервые. А Шурочка помогала в выполнении заданий, обучая меня мастерству составления программы для решения конкретной задачи. Задачи поступали из различных организаций, поток их был неиссякаем. Работа на машине проходила посменно, в дневные и ночные часы, поскольку сама она функционировала круглосуточно. Мне тяжело давались ночные смены, и при первой возможности я отказалась от них.

Отдел программирования постепенно разрастался, его руководителем стал Иван Сергеевич Мухин, который первым в отечестве защитил кандидатскую диссертацию в области программирования.

Составление программы, помимо предварительной работы над выбором алгоритма решения рассматриваемой задачи, выливалось в написание ее команд последовательностями из 32 нулей и единиц, что было непросто. Дальше программа переносилась на перфокарты, и требовалась проверка, конечно, вручную, правильности пробивки перфокарт. Все это составляло весьма трудоемкий процесс.

Работа на машине проходила в компании обслуживающих ее инженеров. Компания была дружной, так как почти одновременно с нашим приходом на практику в ИТМиВТ пришли молодые, только что окончившие институты инженеры, и мы прекрасно понимали друг друга. Среди них были Володя Мельников и Сева Бурцев — будущие академики, а тогда просто милые веселые ребята. Замечательный вклад в общее дело вносил Андрей Соколов, который чувствовал, в каком месте машинного зала нужно подпрыгнуть, чтобы восстановить контакт, нарушивший работу машины.

Вне общения с машиной мы, практиканты, сидели в отведенной отделу комнате, что находилась на третьем этаже в башенном торце здания института. Дружно ходили обедать в столовую Дома профсоюзов, а остатки бесплатного хлеба раскладывали на батареи в комнате, где они ожидали нашего прихода следующим утром и составляли наш завтрак. В положенные перерывы играли в домино, сочиняли стишки на злобу дня (жаль, что ничего из них не сохранилось), забавлялись прибаутками типа "сколько же в институте сотрудников с птичьими фамилиями — Курочкин, Дроздов ... Мухин", в общем, радовались каждому дню. Обычная наша компания — это Кирилл, Гера, Спартак, Аня и я. Нонна Тархнишвили, формально числившаяся практикантом, возложила на меня выполнение своих заданий и почти не появлялась в институте. А Сергей Ломов и Сусанна проходили практику где-то в других местах. Что же касается Кариженского, он, бывший фронтовик и явный выпивоха, был под особым попечительством факультета, ему прощалось многое, в частности, игнорирование производственной практики. Со временем в нашей комнате появился еще один "соплеменник" — Коля Трифонов, аспирант С. Л. Соболева, работающий над решением на БЭСМ задач структурного анализа кристаллов. Он был старше нас и не принимал участия в наших развлечениях, занимаясь своим делом. Отмечу, что защищенная им в 1954 г. диссертация была второй в отечестве, посвященной программированию. Сама защита проходила закрыто, так как считалась секретной.

Наше обучение программированию "со слуха" дополнилось откровениями, почерпнутыми из уникальной в то время и ставшей знаменитой впоследствии монографии [4]: "Решение математических задач на автоматических цифровых машинах. Программирование для быстродействующих электронных счетных машин", авторами которой были Л. А. Люстерник, А. А. Абрамов, В. И. Шестаков, М. Р. Шура-Бура. Монография была опубликована издательством Академии наук в 1952 г. и стала первым в отечестве пособием по программированию. Она имела гриф "секретно", а мы все "ходили" под этим грифом и с упоением изучали ее в институте, так как выносить секретные материалы запрещалось. Кстати, и отчеты по выполненным нами работам тоже снабжались грифом "секретно".

Как я уже отмечала, к нам поступали задачи из разных организаций. Они регистрировались, а потом распределялись по программистам. Получивший задачу поддерживал связь с ее заказчиком. По завершении начиналась работа с другой задачей, и так постоянно.

Конструируемая машина была событием в стране, поэтому ее в действии хотели увидеть многие. Допущенным к посещению машинного зала демонстрировали, как машина считает, для чего заранее подбиралась программа, обязательно с выводом на печать результатов. Во время одного из моих дежурств на машине появились в зале Никита Сергеевич Хрущев (не помню его статуса в то время) и сопровождающий его президент Академии наук Несмеянов. Для них машина работала без сбоя, выводя на печать результаты счета и радуя посетителей ритмичной работой печатающего устройства. Приходилось давать пояснения к тому, что происходит, и отвечать на возникающие вопросы. Запомнился вопрос Хрущева, узнавшего о том, что мы имеем дело с числами, которые записываются наборами из нулей и единиц: "Как же вы понимаете, что выдает вам машина, если она работает не с обычными числами, а с какими-то иными?" Вопрос удивил своей нетривиальностью.

Из всех посетителей глубокое впечатление произвел Алексей Андреевич Ляпунов, имя которого мы узнали, познакомившись по его приходу в одну из наших ночных смен. Явился высокий сутулый человек с какой-то кошелкой в руках, из которой торчал термос. Мы, находящиеся в машинном зале, сразу настроились на веселый лад. Однако задаваемые им вопросы, реакция на наши ответы, а главное манера общения, свидетельствующая о том, что мы имеем дело с редким представителем научной интеллигенции, быстро сбили наш настрой. Установилась дружеская атмосфера, мы узнали, что у Алексея Андреевича сахарный диабет, поэтому требуется регулярное подкрепление едой и питьем, без чего его просто не выпустили бы из дома на целую ночь. Из кошелки были извлечены бутерброды, которыми он щедро поделился с нами. Вдумчивое изучение того, как работает машина, вникание в трудности при составлении программ для нее — все это стало для Алексея Андреевича необходимым подспорьем при дальнейшем развитии начатой им работы по выявлению основных проблем нового направления — программирования. До сих пор я горжусь тем, что в ту ночную смену консультировала его, а он был моим "учеником". Это сыграло решающую роль в моем профессиональном становлении и не только в нем.

В процессе нашей практики приходилось подстраиваться к возможностям, предоставляемым усовершенствованием самой машины, что шло непрерывно. Так, немаловажным новшеством явилась возможность изменять команды программы при ее выполнении, это сильно расширило класс решаемых задач.

К концу семестра закончилась практика, благодаря которой в институте, где она проходила, были взяты на заметку появившиеся новые программисты. Институт пригласил некоторых из нас, в том числе и меня, на работу, совмещая ее с учебой в университете. Я получила должность конструктора на полставки и мне завели трудовую книжку. По законам того времени совмещение работы с учебой разрешалось на срок, не превосходящий двух месяцев, что, конечно, соблюдалось: по истечении этого срока меня увольняли и через пару дней снова принимали на ту же должность.

Работа в институте подсказала тему моего дипломного проекта. Им стало решение на БЭСМ одной из принятых институтом задач. Я должна была проявить полученные знания вычислительных методов, с толком применить их при построении алгоритма решения задачи, составить программу по построенному мной алгоритму и, убедившись в ее правильности, дать численное решение задачи. Не даю формулировки самой задачи, которая не нашла продолжения в моей деятельности, не зафиксирована публикацией, хотя ее решение получило на защите дипломных работ отличную оценку.

Весной 1953 г. на кафедре прошла процедура по рекомендации выпускников в аспирантуру. Это всегда сопровождается выделением для каждого рекомендуемого его будущего научного руководителя. Необходимым условием получения рекомендации, конечно, были отличная учеба и активное участие в жизни факультета. По обеим статьям моя кандидатура проходила, но вот вопрос о научном руководителе требовал специального решения. Дело в том, что моим куратором на кафедре был Николай Павлович Жидков, кандидат физико-математических наук, а по традициям университета руководство аспирантом возлагалось на доктора. И тут, к моей великой радости, на руководство дал согласие Алексей Андреевич Ляпунов. Возможно, он приметил меня при посещении института, когда я выполняла роль гида в машинном зале.

Сдав благополучно осенью 1953 г. вступительные экзамены в аспирантуру, я поступила в распоряжение Алексея Андреевича. Хорошо помню, как он буквально схватился за голову, узнав о том, какие именно курсы я прослушала за три последних года. Естественно, он решил наверстать упущенное мной из настоящего математического образования. Сделать это можно было пользуясь тем, что программой обучения в аспирантуре предписывались три годовых курса с экзаменом по каждому, три семестровых курса с зачетом по каждому, причем предметы этих курсов выбирались руководителем аспиранта и утверждались кафедрой. Экзамен при-

нимала комиссия из двух преподавателей, одним из которых должен быть руководитель аспиранта, а зачет — только руководитель. Понятно, что добавочно обязательными были годовые курсы по философии и какому-нибудь иностранному языку. И вот Алексеем Андреевичем были выбраны для меня экзамены по теории множеств, по теории функций действительного переменного, по специальным главам уравнений математической физики, а зачеты — по теории вероятности, по алгебре и по функциональному анализу. Кое-что еще из предлагаемого Алексеем Андреевичем кафедра отклонила, сочтя достаточным объем утвержденного.

Большим событием осени этого года было открытие на Ленинских горах нового здания университета, куда с Моховой переместился механико-математический факультет. Обилие учебных аудиторий, отдельные помещения для кафедр и библиотек, широкие коридоры и прочие прелести описаны многими поклонниками университета, я же отмечу наличие общежитий, позволившее разместить в них большое число студентов и аспирантов, что исключалось общежитием на ул. Стромынка, куда селили только иногородних. Поэтому при поступлении в университет, засвидетельствовав свое проживание в пригороде Москвы, я и не претендовала на общежитие. Все пять лет ездила в университет на электричке, домой возвращалась на ночь, так как после аудиторных занятий перемещалась в библиотеку, где и проводила часы до вечера. И вот я получила место в общежитии на Ленинских горах, в блоке из двух комнат с собственным туалетом и душевой и даже с телефоном. Через год общежитие стали уплотнять, меня переместили в другой блок, уже без телефона, а моя отдельная комната была восьмиметровой, против двенадцатиметровой прежде. Там я провела последние два года аспирантуры, не переставая радоваться тому, какая благодать выпала на мою долю.

Первый год моей аспирантуры был заполнен подготовкой к очередному экзамену, что строго контролировалось Алексеем Андреевичем, участием в семинаре по программированию, которым он руководил, а главное, приобщением к совершенно новой для меня атмосфере существования. Убедившись, что я не отлыниваю от предписанного им обучения, Алексей Андреевич ввел меня в свою среду. Сначала я была частым гостем в его квартире на Хавско-Шаболовском проезде, а со временем почти членом его семейства. У себя дома Алексей Андреевич проводил многочисленные беседы с соратниками по самым различным направлениям науки, и мною были приобретены знакомства с рядом интереснейших людей. К ним принадлежали Сергей Всеволодович Яблонский, Николай Андреевич Криницкий, Николай Пантелеймонович Бусленко, а позднее Леонид Витальевич Канторович. Беседы в традициях этого гостеприимного дома непременно завершались чаепитием. Приобретенные знакомства оказались прочными.

Круг тем, интересовавших Алексея Андреевича, был воистину необозримым, и на мою долю выпало несомненное счастье находиться вблизи этого яркого человека. Главное направление его деятельности в то время описано мною в статье "Размышления о феномене Алексея Андреевича Ляпунова", в настоящей статье я хочу это повторить.

В 1953 г. академик Мстислав Всеволодович Келдыш организовал в составе Математического института имени В. А. Стеклова Отделение прикладной математики (ныне Институт прикладной математики имени М. В. Келдыша) и предложил Алексею Андреевичу возглавить в нем работы по программированию. Это определило обращение Алексея Андреевича к кибернетике как направлению, призванному выявить законы возникновения, передачи, хранения и переработки информации как в живой природе, так и в различных сферах человеческой деятельности. Этот интерес появился после его знакомства с книгой Винера, давшей название самому направлению — кибернетика.

Работы в области кибернетики начались борьбой Алексея Андреевича за ее существование. Дело в том, что в те годы малоизвестная в нашей стране кибернетика носила ярлык "буржуазной науки", и для развития кибернетики надо было его снять. Алексей Андреевич проводил большую разъяснительную работу, он убеждал людей разного научного и служебного ранга в ошибочности официального суждения о кибернетике, вел многочисленные лекции и беседы об ее истинном содержании. Наконец, совместно с Сергеем Львовичем Соболевым и Анатолием Ивановичем Китовым он опубликовал в "Вопросах философии" обстоятельную статью о том, что составляет предмет кибернетики и сколь важно ее развитие для науки и государства [5]. Алексей Андреевич организовал кибернетический семинар в МГУ, добился издания "Кибернетических сборников", в которых были даны переводы наиболее значительных работ в теоретической кибернетике (они вышли под редакцией А. А. Ляпунова и О. Б. Лупанова). Благодаря настойчивым усилиям Алексея Андреевича вышел в свет перевод книги Винера, началось издание "Проблем кибернетики", сборников, где публиковались отечественные работы в этой области (под его редакцией вышло 29 томов). При Президиуме АН СССР был создан Совет по кибернетике под руководством

академика Акселя Ивановича Берга, и Алексей Андреевич Ляпунов стал его заместителем.

На IV Всесоюзном математическом съезде (1966 г.) Алексей Андреевич подвел итоги борьбы за кибернетику: "За короткий срок отношение к кибернетике прошло следующие фазы: 1) категорическое отрицание; 2) констатация существования; 3) признание полезности, но отсутствие задач для математиков; 4) признание некоторой математической проблематики; 5) полное признание математической проблематики кибернетики". Подробному изложению проблематики кибернетики посвящена статья, написанная Алексеем Андреевичем и Сергеем Всеволодовичем Яблонским.

Приятно сознавать, что я была непосредственным свидетелем этого процесса, более того, принимала участие во многих обсуждениях.

Алексей Андреевич Ляпунов вполне оправданно считается отцом отечественной кибернетики. Он внес существенный научный вклад в различные области кибернетики. Так, параллельно с закладыванием основ теоретического программирования, он организовал первые в нашей стране работы по машинному переводу и математической лингвистике. Глубоким и постоянным был интерес Алексея Андреевича к биологии. Уже в тридцатых годах он столкнулся с тяжелым положением в генетике и встал на ее защиту; тогда по инициативе А. Н. Колмогорова он вместе с Ю. Я. Керкисом проводил статистическое исследование экспериментов по расщеплению признаков при наследовании. В 1950-х гг. Алексей Андреевич возобновил борьбу за восстановление отечественной биологии. Совместно с Сергеем Львовичем Соболевым он подготовил письмо в ЦК КПСС о положении в генетике, письмо было подписано пятнадцатью крупнейшими математиками страны. Оно влилось в поток других писем, и в 1956—1957 гг. в стране были созданы первые исследовательские коллективы ученых-генетиков. С основанием "Проблем кибернетики" в них стали публиковаться работы по генетике и теории эволюции. Собственные активные исследования Алексея Андреевича в биологии относятся к последнему десятилетию его жизни.

Здоровье Алексея Андреевича часто заставляло его оставаться дома, отказываться от дел и встреч за его пределами. Иногда мне удавалось прийти на помощь. Несколько раз я заменяла Алексея Андреевича на лекциях по программированию, которые он проводил для всех желающих постичь эту науку. Это происходило в университете, слушатели собирались не только из московских организаций, но и приезжали из других городов, большая аудитория амфитеатра была полна. По молодости я отважилась замещать Алексея Андреевича, естественно, он меня готовил к предстоящей лекции. А как-то раз он отрядил меня на встречу со школьниками в близлежащей школе, чтобы я провела беседу по программированию. Я не отказалась, хотя это было большой смелостью: не так уж много я знала об этом предмете сама.

Я стала вторым аспирантом Алексея Андреевича, посвятившим себя программированию, первым был Юрий Иванович Янов, которому было поручено использование приемов математической логики в исследованиях схем программ, записанных на операторном языке. В 1954 г. выпускниками кафедры вычислительной математики стали студенты, с которыми он работал с прошлого года. Алексей Андреевич взял в аспиранты Андрея Ершова и Олю Кулагину. Саша Любимский, Сева Штаркман и, если не ошибаюсь, Игорь Задыхайло пошли в Отделение прикладной математики и стали работать с Михаилом Романовичем Шура-Бурой. Там проходили семинары, на которых дебатировались проблемы настоящего и будущего программирования, обсуждались достоинства и недоработки операторного метода, предложенного Ляпуновым. Эти семинары под его руководством вносили свою лепту в формирование взгляда на программирование как на науку.

Тем же целям служили и семинары по программированию, которые проводил Алексей Андреевич на механико-математическом факультете. Семинар проходил при стечении народа, и доклады на нем бурно обсуждались. Запомнилось заседание, на котором выступал Тони Хоар — стажер из Великобритании, ставший со временем одним из ведущих специалистов в области программирования. Возможно, именно участие в семинаре внесло свою лепту в его выбор направления исследований. Приходилось и мне выступать на семинаре. В частности, в полемику по одному из моих докладов вступил и Тони Хоар.

Часто дискуссия, возникшая на семинаре, продолжалась по пути Алексея Андреевича домой на Шаболовку, сопровождаемого не остывшими от споров участниками семинара. Компания вваливалась в дом, и дискуссия мирно завершалась за чайным столом.

К моей работе в институте Алексей Андреевич относился с одобрением, и она вносила дополнительную струю в мою жизнь. Директором института стал Сергей Алексеевич Лебедев, БЭСМ была принята государственной комиссией, а ее конструктору было присвоено звание академика. Это событие бурно праздновалось в институте, конечно

же, программисты были не последними в этих празднованиях.

По-прежнему поступали в отдел И. С. Мухина, в котором я работала, заявки на решение задач, задачи сменяли друг друга, не давая программисту возможности углубиться в осмысление очередной. И хотя Иван Сергеевич ослабил для меня эту текучку, я стала явно ею тяготиться.

В 1954 г. научные сотрудники Физического института Академии наук (ФИАН) предложили мне прочитать им курс лекций по программированию. Алексей Андреевич посоветовал принять это предложение. Конечно, я была вооружена операторным методом, который помогал в описании алгоритмов задач, подлежащих программированию. И тем не менее, мое преподавание программирования шло на примитивном уровне, приемы программирования излагались при описании программ для решения простых задач, например, вычислить значение функций $\sin(x)$, $\cos(x)$, $\log(x)$, найти общий наибольший делитель чисел. Программы составлялись на воображаемой трехадресной машине, а сами адреса были буквенными. Этот прием был предложен Алексеем Андреевичем и сразу вошел в практику.

Моими слушателями в основном были научные сотрудники Лаборатории оптики, которой руководил академик Григорий Самуилович Ландсберг. Они восприняли прочитанный мной курс с явным одобрением, следствием чего явилось приглашение на постоянную работу в их лабораторию. Алексей Андреевич знал о том, что работа в отделе И. С. Мухина перестала меня удовлетворять, и к концу года с его согласия я перевелась в соседний ФИАН. Став младшим научным сотрудником лаборатории оптики (естественно, по совместительству с учебой в университете), я не потеряла связи со своими бывшими коллегами в ИТМиВТ, по-прежнему имела доступ к машине, но теперь занималась одной задачей — расчетом колебательных спектров молекул простейших углеводородов. Моим непосредственным руководителем стал Михаил Михайлович Сущинский, доктор физико-математических наук. В решении этой задачи было немало творческих моментов, начиная с автоматизации построения матриц, собственные значения которых представляют собой искомые частоты колебаний, и кончая выбором метода вычисления этих собственных значений, позволявшего оперировать матрицами высоких порядков. Отбраковывались методы, не обеспечивающие допустимой точности вычисленных значений. Эта работа была утверждена кафедрой в качестве диссертационной, естественно, после представления ее Алексеем Андреевичем.

Уместно отметить следующее. В своей работе с учениками Алексей Андреевич следовал традициям старшего поколения ученых, в значительной мере утраченным в наши дни. Свою задачу он видел, прежде всего, в пробуждении научной активности ученика и в возможно большем расширении его кругозора. Предлагая то или иное направление исследований, Алексей Андреевич на содержательном уровне описывал стоящие в нем задачи, выделяя "стержневую" и рассматривая их во взаимосвязи друг с другом. Да и само направление преподносилось в связи с другими. Предоставлялась свобода в выборе той задачи, которая станет "своей", и отпускалось на это солидное время. А пока шло обрастание знаниями. Когда выбор был позади, Алексей Андреевич не подменял собой ученика даже в вопросе формальной постановки задачи, не говоря уже о поиске методов ее решения. Он играл роль среды, в которой проходила эта работа. Подсказки давались деликатно, не связывая инициативы работающего. Так проходило "выращивание" будущего ученого-исследователя. Понятно, что этот процесс не укладывался в три аспирантских года. Да и выбранное направление бывало таким, что требовалась разработка понятийного аппарата, на что уходило немало времени. Научное руководство Алексея Андреевича часто перерастало в длительную творческую дружбу.

Этот метод "выращивания" давал замечательные плоды, свидетельством чему служат такие ученики Ляпунова, как Сергей Всеволодович Яблонский и Юрий Иванович Журавлев. К слову сказать, испытав на себе этот метод, я ввела его потом в свою практику.

Возвращаюсь к повествованию о том, как шли мои аспирантские дела на факультете. Подготовка к запланированным экзаменам требовала многочасовой работы в библиотеке, ведь необходимых учебников у меня не было. Проникновение в тайны теории множеств дало мне понимание того, почему эта область математики постоянно присутствовала в научной работе Алексея Андреевича. Он не покидал ее, как бы ни отвлекали его другие направления, до последних дней жизни. Экзамен по теории множеств Алексей Андреевич принимал вместе с профессором Ниной Карловной Бари, хотел было снизить оценку до четверки, но Нина Карловна опротестовала такое намерение.

Сдача следующего экзамена, по теории функций действительного переменного, подзадержалась в силу моей занятости в ИТМиВТ, а потом у физиков. Нас с Алексеем Андреевичем по этому поводу вызвал на соответствующее внушение академик Андрей Николаевич Колмогоров, который

тогда заведовал Отделом аспирантуры МГУ. Он отметил недостойное поведение бывшего сталинского стипендиата, но в конечном итоге согласился с тем, что мои аспирантские дела не внушают опасений. Сам экзамен я сдавала двум профессорам — Алексею Андреевичу и Лазарю Ароновичу Люстернику, и в процессе сдачи была просто очарована Лазарем Ароновичем. Получив очередную пятерку, приступила к подготовке к следующему экзамену. В специальные главы уравнений математической физики были включены последние результаты, полученные докторами наук Ладыженской и Олейник, двумя Ольгами. Именно Ольгу Олейник Алексей Андреевич пригласил принимать у меня этот экзамен. При подготовке к нему, как мне представляется, мне удалось постичь методы исследований, выполненных этими научными дамами. Мелькала даже мысль заняться задачами из этой области, но это оказалось нереально при моей общей занятости.

Сдача зачетов не была нормирована сроками, да и проходила она почти всегда в домашней обстановке у Алексея Андреевича. Большим подспорьем при подготовке служили спецсеминары. На одном из них реферировалась книга Розы Петер "Рекурсивные функции", докладчиком был Володя Успенский, тоже аспирант нашего факультета. Этот спецсеминар был первым шагом на пути включения в учебные планы факультета курса по теории алгоритмов. На другом спецсеминаре обсуждались проблемы, относящиеся к математической логике, им руководила профессор С. А. Яновская. Этот курс тоже вскоре вошел в учебный план.

А с признанием кибернетики при поддержке С. Л. Соболева Алексей Андреевич в 1954/55 учебном году организовал при кафедре научный семинар по кибернетике. Он был ориентирован сначала на студентов и аспирантов кафедры, но быстро перерос в общемосковский, а затем — в общесоюзный.

Основную задачу семинара Алексей Андреевич видел в координации исследований в многочисленных направлениях науки и техники, оперирующих понятиями информации и управления. Семинар проработал десять лет и внес решающий вклад в становление кибернетических исследований в нашей стране.

На заседания семинара стекались люди из самых разных мест. В его работе активно участвовали Игорь Андреевич Полетаев, автор нашумевшей тогда книги "Сигнал", и коллеги Алексея Андреевича по Артакадемии, где он служил до того как Соболев пригласил его в МГУ. Выступали с докладами Н. П. Бусленко и Н. А. Криницкий. Сохранилась летопись докладов, сделанных и обсужденных на заседаниях (см. статью М. Г. Гаазе-Раппопорта в работе [6]).

Программой обучения в аспирантуре предусматривалась обязательная педагогическая практика. Алексей Андреевич придавал большое значение приобретению педагогического опыта. О его поддержке моих первых шагов в этой области я уже писала. На кафедре мне поручили ведение практических занятий по программированию. Они проходили успешно, на третьем году аспирантуры мне доверили чтение собственного курса по программированию, руководство курсовыми и даже дипломной работой; моей первой дипломницей была Таня Гаврилова, научные и просто дружеские связи с которой продолжались многие годы.

Продолжалась и моя работа по комсомольской линии. Одним из самых удачных мероприятий, организованных мною совместно с В. А. Успенским, был вечер для аспирантов и преподавателей факультета. Он получился ярким и запоминающимся благодаря активному участию Лазаря Ароновича Люстерника, который занимал собравшихся импровизированными шарадами. Например, сначала выходил якобы лектор с докладом, в котором непрерывно повторялось слово "век", затем появлялся подвязанный салфеткой человек, произносящий "уа-уа-уа", а потом Лазарь Аронович выводил к зрителям профессора Илью Несторовича Векуа — отгадку шарады. Восторг был полным.

Годы моей аспирантуры проходили. В заключение кафедра организовала предзащиту моей диссертационной работы, посвященной решению задачи о колебательных спектрах простейших углеводородов, и дала ей положительную оценку. Прошло и распределение на работу, в те времена вуз должен был удовлетворять заявки от работодателей. По заявке от ФИАН меня направили в этот институт.

Во второй половине 1956 г. я вышла замуж за Теодора Михайловича Тер-Микаэляна. Наше знакомство состоялось в ИТМиВТ, где я выполняла функции программиста. Теодор Михайлович, защитивший к тому времени кандидатскую диссертацию под руководством Мстислава Всеволодовича Келдыша, работал в Институте математики в Ереване и был откомандирован в Москву, во Всесоюзный НИИ электромеханики, который возглавлял академик Армянской ССР Андраник Гевондович Иосифьян. Цель командировки — освоить программирование, Теодору Михайловичу было поручено решение на БЭСМ задач по плану этого НИИ. Будучи уже состоявшимся программистом, я консультировала его. Наше эпизодическое знакомство закрепилось, когда он стал регулярно посещать за-

седания кибернетического семинара, которым руководил Алексей Андреевич, и закончилось женитьбой. Назревал мой переезд из Москвы в Ереван, и Алексей Андреевич заблаговременно подготовил почву для моей работы там. Как это происходило — отдельный рассказ, который мыслится как продолжение этого повествования.

Осенью 1956 г. меня перевели в штат ФИАН на должность младшего научного сотрудника. В лаборатории оптики я была уже "старожилом", летом этого года участвовала в организованной институтом конференции в г. Львов, была соавтором Михаила Михайловича Сущинского в публикациях лаборатории, вместе с ним ездила домой к Григорию Самуиловичу Ландсбергу с рабочими отчетами. Цирроз печени держал нашего руководителя в постели. Очень скоро мы простились с ним, похороны Ландсберга на Новодевичьем кладбище были очень торжественными.

Параллельно с выполнением моих институтских обязанностей не только шлифовался текст диссертации, но начались первые попытки формализации процесса программирования, что вылилось, в будущем, в профессиональную работу в области теоретического программирования. Составление программ для решения конкретной задачи, как профессиональная деятельность, ушло в прошлое с моим отъездом из Москвы, а это произошло в мае 1957 г.

Рассказ о том, как сложился мой путь в программировании, и как протекала моя работа в роли программиста, подошел к концу. Я намеренно не упоминала своих друзей, сопровождавших меня на этом отрезке моей жизни, полагая, что это — тема отдельного повествования. Не рассказать же об Алексее Андреевиче, плотно вошедшим в мою жизнь, было бы непростительно. Он был и остается примером того, как нужно строить свою жизнь.

Список литературы

1. **Факультет** вычислительной математики и кибернетики. История и современность. Биографический справочник // автор-составитель Е. А. Григорьев. М.: Изд-во МГУ, 2010. С. 10—11.
2. **Иван Семенович Березин.** Биография, воспоминания, документы // автор-составитель Е. А. Григорьев. М.: Изд-во МГУ, 2011. 167 с.
3. **Ляпунов А. А.** О логических схемах программ // Проблемы кибернетики. 1958. Вып. 1. С. 46—74.
4. **Люстерник Л. А., Абрамов А. А., Шестаков В. И., Шура-Бура М. Р.** Решение математических задач на автоматических цифровых машинах. Программирование для быстродействующих электронных счетных машин. М.: Изд-во АН СССР, 1952.
5. **Соболев С. Л., Ляпунов А. А., Китов А. И.** Основные черты кибернетики // Вопросы философии. 1955. № 4. С. 136—148.
6. **Очерки** истории информатики в России // редакторы-составители Д. А. Поспелов, Я. И. Фет. Новосибирск: Научно-издат. центр ОИ ГТМ СО АН, 1998. 662 с.

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

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

Сдано в набор 07.04.2014 г. Подписано в печать 21.05.2014 г. Формат 60×88 1/8. Заказ PI614
Цена свободная.

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