На правах рукописи

Павлухин Павел Викторович

Эффективное решение задач газовой динамики на кластерных системах с графическими ускорителями

Специальность 05.13.18 —
«Математическое моделирование, численные методы и комплексы программ»

Автореферат
диссертации на соискание учёной степени
кандидата физико-математических наук

Москва — 2019

Работа выполнена на кафедре вычислительной механики Механико-математического факультутета МГУ имени М.В.Ломоносова.

Научный руководитель: **Меньшов Игорь Станиславович**
д. ф.-м. н., профессор
ведущий научный сотрудник
ИПМ им М.В. Келдыша РАН

Официальные оппоненты: **Дерюгин Юрий Николаевич,**
доктор физико-математических наук,
РФЯЦ «ВНИИЭФ»,
главный научный сотрудник Института Теоретической и Математической Физики

Семенов Илья Витальевич,
кандидат физико-математических наук,
ФГБУ ИАП РАН,
ведущий научный сотрудник Отдела вычислительных методов и турбулентности

Ведущая организация: Федеральный исследовательский центр «Информатика и управление» Российской академии наук (ФИЦ ИУ РАН)

Защита состоится «__» _____ 2019 года в «__» час. «__» мин. на заседании диссертационного совета Д002.024.03 при ИПМ им. М.В. Келдыша РАН по адресу: 125047, Москва, Миусская пл., д.4.

С диссертацией можно ознакомиться в библиотеке Института прикладной математики им. М.В. Келдыша РАН и на сайте *www.keldysh.ru/council/3/.*

Автореферат разослан «__» _____ 2019 года.

Ученый секретарь
диссертационного совета
Д002.024.03,
кандидат физико-математических наук Корнилина М.А.

Общая характеристика работы

Актуальность темы исследования. Решение задач газовой динамики на современных вычислительных системах с новыми архитектурами сопряжено с рядом возникающих при этом трудностей. Первая связана с дискретизацией расчетной области. В случае, если она является геометрически сложной, сеточное разбиение в подавляющем большинстве случаев не структурировано. Построение сетки при этом требует значительных вычислительных ресурсов и нередко «ручного» вмешательства для ее коррекции, что приводит к немалым временным затратам. Неструктурированные сеточные разбиения порождают нерегулярный доступ к памяти. Как результат, производительность программных реализаций методов для сеток данного типа оказывается ограничена не числом выполняемых в единицу времени арифметических операций (compute-bound), а пропускной способностью памяти (memory-bound). В большей степени это критично для массивно-параллельных вычислителей, поскольку эффективность их работы зависит в первую очередь от упорядоченности обращений в память. Поэтому наиболее предпочтительно использовать структурированные сеточные разбиения, для которых характерен регулярный шаблон обращений в память. Но построение таких сеток, согласованных с границей расчетной области, может оказаться в ряде случаев трудновыполнимой или вовсе неразрешимой задачей.

Другая сложность обусловлена взаимосвязанным развитием численных методов и архитектуры процессоров. Для решения задач на системах с ограниченными вычислительными ресурсами использовались сетки низкого разрешения. Чтобы получить на них более точное решение, приходилось строить численные методы высокого порядка со сложной алгоритмической структурой. С другой стороны, вычислительные ядра процессоров также становились все более “тяжелыми”: внеочередное исполнение команд, преднакачка данных, прогнозирование ветвлений, векторные инструкции позволяли эффективно реализовывать сложные методы. Но масштабируемость вычислительных систем на таких “тяжелых” ядрах ограничена, и их дальнейшее усложнение приводит к большему снижению эффективности. Это стало причиной появления систем на новых массивно-параллельных архитектурах с большим числом простых ядер. Вот здесь и проявилась проблема: накопленный за десятилетия “багаж” численных методов оказался плохо подходящим для реализации на новых вычислителях, поскольку высокая производительность в них достигается не за счет эффективного исполнения каждого из небольшого (порядка 10) числа “тяжелых” потоков, а за счет одновременной обработки значительно большего (порядка 1000) числа “легких”. Иными словами, простое устройство ядра массивно-параллельных вычислителей влечет за собой требование вычислительного примитивизма для применяемых численных методов.

Явные методы хорошо подходят для реализации на новых вычислительных архитектурах, но их применение сильно ограничено из-за условия

устойчивости. В задачах со сложной геометрией, где неизбежно в сеточном разбиении будут присутствовать разномасштабные ячейки, глобальный шаг интегрирования по времени будет определяться размерами наименьшей из них, что приведет к неоправданно высокому росту вычислительной сложности. Явно-неявные, неявные методы позволяют обойти это ограничение, но они значительно сложнее с точки зрения указанных выше требований, особенно в части возможного их распараллеливания.

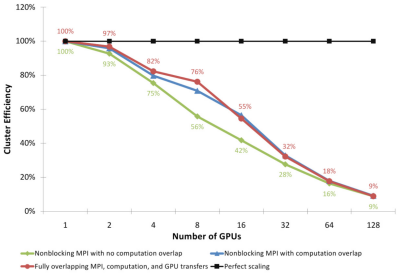
Реализация неявных схем для графических ускорителей (GPU) является непростой задачей, поскольку имеющаяся, как правило, в них зависимость по данным значительно затрудняет написание эффективного решателя, особенно в отсутствие средств глобальной синхронизации на GPU. В частности, в одной из работ метод LU-SGS, использующийся для решения СЛАУ, порожденной неявными схемами, рассматривается как один из наиболее подходящих методов для расчетов на графических ускорителях. Однако, из-за возникающей в нем зависимости по данным, сильно усложняющей распараллеливание, в указанной работе был выбран другой – DP-LUR, в котором отсутствует зависимость по данным, но при этом и выше вычислительная сложность: сравнение этих двух методов показывает, что для нахождения стационарного решения с помощью DP-LUR затрачивается процессорное время, почти двукратно превышающее время решения с помощью LU-SGS, поэтому с точки зрения производительности и эффективности использования вычислительных ресурсов предпочтительнее параллельная реализация последнего метода.

В большинстве работ, параллельный алгоритм для LU-SGS соблюдает работу данного метода лишь частично – отдельно на каждой из подобластей, полученных после декомпозиции расчетной области на параллельные процессы, с упрощенной процедурой обмена граничными ячейками (т.е. с нарушением зависимости по данным в рассматриваемом методе), что в итоге может приводить к весьма значительному снижению скорости сходимости решения. В тех же работах, где LU-SGS соблюдается на *всей* расчетной области, масштабируемость программных реализаций весьма ограничена.

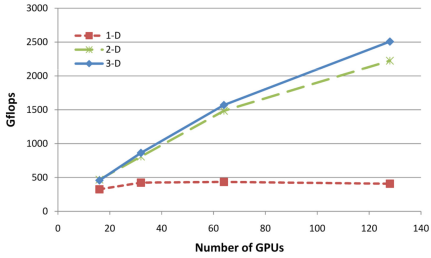
Помимо собственно численного метода, важную роль, как уже было сказано, играет тип сеточного разбиения – GPU значительно эффективнее работают с регулярными структурами данных, в которых представляются структурированные сеточные разбиения, чем с нерегулярными, характерными для неструктурированных сеток. В свою очередь, применение структурированных сеток, адаптированных (конформных) к геометрии расчетной области, накладывает ограничения на сложность геометрии расчетной области. Поэтому возникает задача разработки и адаптации численных методов для решения задач газовой динамики со сложной геометрией на структурированных сетках.

Наконец, весьма остро стоит затронутый выше вопрос масштабируемости программных реализаций решателей на вычислительных системах. В частности, на Рис. 1 приведен график производительности реализации метода Якоби с использованием декартовых сеток – масштабируемость составила 18% на

64 GPU (1 млн ячеек/GPU) и 67.5% на 128 GPU (2 млн ячеек/GPU) соответственно. Реализация на GPU неявного метода ADI на декартовых сетках имеет эффективность в 75% уже на 4 GPU (2 млн ячеек/GPU).



а)



б)

Рис. 1 — Масштабируемость решателей на вычислительных системах с множеством GPU

В части решателей масштабируемость изначально ограничена самим параллельным алгоритмом, лежащим в основе программной реализации, т.е. она лимитирована той частью вычислений, которую в соответствии с алгоритмом необходимо выполнять исключительно последовательно. Однако даже если алгоритм не содержит таких последовательных вычислений, эффективность его программной реализации тем не менее может значительно падать с увеличением вычислительных ресурсов для нее. Связано это, как правило, с тем, что отношение времени обмена данными между параллельными процессами ко времени счета начинает расти при выполнении задачи на всё бóльшем числе вычислительных ядер; иными словами, время межпроцессных коммуникаций начинает вносить всё бóльший вклад в общее время работы солвера. И здесь ключевой для повышения эффективности и масштабируемости является возможность *совмещения* по времени вычислений и обмена данных между процессами, т.е. пересылки сообщений между ними на фоне “полезного” счета. Данную возможность должен, во-первых, предоставлять параллельный алгоритм (т.е. обеспечивать корректность при совмещении счета и обменов данными), во-вторых, она должна быть соответствующим образом реализована в программном коде. Применительно к кластерным системам, оснащенным графическими ускорителями в качестве сопроцессоров, последнее требование означает использование нетривиальных схем координации доступа к совместно используемым CUDA и MPI областям памяти через вызовы неблокирующих функций, предоставляемыми библиотеками указанных интерфейсов. Тем не менее, как будет показано в данной работе, даже при использовании сложных схем взаимодействия между CUDA и MPI совмещение счета и обмена данными между процессами может не работать, и для решения этой проблемы необходимо проводить достаточно глубокий анализ производительности программной реализации с привлечением знаний о деталях реализации библиотек MPI.

Таким образом, проблема адаптации методов, разработки эффективных параллельных алгоритмов для них и соответствующих программных реализаций для решения широкого класса задач газовой динамики на кластерных вычислительных системах с графическими ускорителями носит актуальный характер.

Целью диссертационной работы является построение вычислительных алгоритмов для получения максимальной эффективности при использовании современных и перспективных систем гибридной архитектуры для решения прикладных задач газовой динамики. Для достижения поставленной цели рассматриваются следующие задачи:

1. Разработка высокоэффективного масштабируемого параллельного алгоритма гибридной явно-неявной схемы и метода свободной границы на вычислительных системах с графическими ускорителями, который в точности реализует работу своего последовательного прототипа на всей расчетной области с доказательством его корректности.
2. Программная реализация разработанного алгоритма для высокопроизводительных вычислительных систем с графическими ускорителями с использованием технологии CUDA и стандарта MPI.
3. Проведение расчетов газодинамических течений с помощью разработанного программного комплекса и сравнение с результатами, полученными альтернативными методами, и экспериментальными данными.
4. Исследование эффективности и масштабируемости разработанной программной реализации при расчете больших задач с использованием нескольких сотен графических ускорителей.

Научная новизна. Представленные в работе результаты являются новыми. В частности, новыми являются разработанный параллельный алгоритм для методов свободной границы и LU-SGS и реализованный на их основе программный комплекс для решения нестационарных задач газовой динамики на вычислительных системах с большим числом графических ускорителей.

Теоретическая ценность и практическая значимость диссертационной работы состоят в разработанном масштабируемом параллельном алгоритме для методов свободной границы и LU-SGS, а также в разработанном программном комплексе на его основе с использованием технологий CUDA и MPI, который позволяет решать задачи газовой динамики, эффективно задействуя несколько сотен графических ускорителей.

На защиту выносятся следующие основные результаты и положения:

1. Предложено обобщение математической и численной модели свободной границы под специфику архитектуры графических ускорителей.
2. На основе модели свободной границы, гибридной явно-неявной схемы и итерационного метода LU-SGS разработан параллельный алгоритм для решения задач газовой динамики на вычислительных системах с графическими ускорителями.

3. Реализован программный комплекс на основе разработанного алгоритма с использованием технологий CUDA и MPI. Показана эффективность счета до 75% при использовании 768 GPU суперкомпьютера «Ломоносов».
4. С помощью разработанного программного комплекса проведены расчеты ряда газодинамических течений, включая моделирование обтекания вокруг профиля NACA0012 и DLR F6 на декартовой структурированной сетке, которые подтвердили высокую эффективность его работы.

Достоверность и обоснованность полученных результатов обеспечены строгостью используемого математического аппарата и подтверждаются сравнением результатов вычислительных экспериментов с известными в литературе экспериментальными и расчетными данными, а также данными, полученными с помощью других методов.

Апробация работы. Основные результаты работы докладывались на:

1. Ломоносовских чтениях, МГУ им. М.В. Ломоносова (Москва, 2011)
2. Международной научной конференции “Параллельные вычислительные технологии - 2012”. (Новосибирск, 2012)
3. Международной научной конференции “Параллельные вычислительные технологии - 2013”. (Челябинск, 2013)
4. International Conference MATHEMATICAL MODELING AND COMPUTATIONAL PHYSICS, 2013 (MMCP 2013), (Дубна, 2013)
5. Международной суперкомпьютерной конференции “Научный сервис в сети Интернет: многообразие суперкомпьютерных миров” (Абрау-Дюрсо, 2014)
6. Пятой всероссийской конференции “Вычислительный эксперимент в аэроакустике” (Светлогорск, 2014)
7. 13th International Conference on Parallel Computing Technologies (PACT-2015), (Петрозаводск, 2015)
8. 14th International Conference on Parallel Computing Technologies (PACT-2017), (Нижний Новгород, 2017)
9. 7th International Conference on Mathematical Modeling in Physical Sciences (Москва, 2018)

Данная работа участвовала в следующих конкурсах с результатом:

1. Победитель первого этапа конкурса «Эффективное использование GPU-ускорителей при решении больших задач» (ОАО Т-Платформы, Москва, 2011).
2. 1-ое место в конкурсе научных статей Всероссийской конференции молодых ученых "Параллельные и распределенные вычисления"(Новосибирск, 2012).
3. 1-ое место в конкурсе «GPU: серьезные ускорители для больших задач» (ССОЕ МГУ, Nvidia, 2013)

Исследования, описанные в §5.5 пятой главы диссертации, были поддержаны грантом Министерства образования и науки РФ (договор № 14.G39.31.0001 от 13 февраля 2017 г.).

Публикации. Основные результаты по теме диссертации изложены в 6 печатных изданиях [1–6], 2 из которых изданы в журналах, рекомендованных ВАК [3; 4], 3 – в периодических научных журналах, индексируемых Web of Science и Scopus [1; 2; 5].

Личный вклад соискателя. Все исследования, изложенные в диссертационной работе, проведены лично соискателем в процессе научной деятельности. В публикациях в соавторстве с научным руководителем Меньшовым И.С. соискателю принадлежат описания параллельных алгоритмов и моделей, а также результаты вычислительных экспериментов; заимствованный материал обозначен в работе ссылками.

Структура и объем диссертации. Диссертация состоит из введения, шести глав, заключения и списка литературы. Работа представлена на 109 страницах, содержит 46 иллюстраций и 3 таблицы. Список литературы содержит 95 наименований.

Содержание работы

Во **введении** обосновывается актуальность исследований, проводимых в рамках данной диссертационной работы, приводится обзор научной литературы по изучаемой проблеме, формулируется цель, ставятся задачи работы, излагается научная новизна и практическая значимость представляемой работы.

Также приводится обоснование направления исследования, в рамках которого выполнялась данная работа, т.е. обоснование создания масштабируемых параллельных алгоритмов и их реализации на вычислительных системах с большим числом графических ускорителей.

Первая глава посвящена обзору этапов развития вычислительных систем, архитектуры графических ускорителей, а также вопросам реализации на них различных численных методов и схем.

Во **второй главе** приведено описание модели и метода свободной границы, явно-неявной схемы и итеративного метода LU-SGS, используемого для решения результирующей СЛАУ.

В правую часть определяющей системы уравнений Эйлера вводится специальная добавка F_ω, называемая *компенсационным потоком*:

$$\frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{f}_k}{\partial x_k} = -\vec{F_\omega}, \tag{1}$$

$$\vec{F}_w = \begin{pmatrix} \rho u_k n_k \\ \rho u_k u_m n_k + (p - p_w) n_m \\ \rho u_k n_k H \end{pmatrix} \delta(\vec{x}, \Gamma), \tag{2}$$

где $\delta(\vec{x},\Gamma)$ обозначает обобщенную функцию Дирака поверхности Γ – границы области, занятой твердым телом Ω, $\vec{n} = (n_k)$ – вектор единичной внешней к Ω нормали на поверхности Γ, p_ω – мгновенная реакция жесткой стенки на воздействие со стороны потока жидкости. Система модифицированных уравнений (1) решается во всем пространстве $\mathfrak{R}^3$, а компенсационный поток $\vec{F}_w$ подбирается таким образом, чтобы сужение решения уравнения (1) в $\mathfrak{R}^3$ на область $\mathfrak{R}^3\backslash\Omega$ в точности совпадало с решением задачи в стандартной постановке с граничными условиями непротекания.

Для вычисления компенсационного потока в каждой ячейке используется линейное восполнение, которое задается объемной долей, занимаемой жидкостью в пересекаемой ячейке, ω_f, и вектором внешней нормали $\vec{n_f}$, $|\vec{n}_f| = s_f$, где s_f – площадь плоского элемента, аппроксимирующего пересечения счетной ячейки с поверхностью Γ.

Применяя после дискретизации пространства декартовой сеткой метод конечного объема к уравнениям (1) и используя метод разделения по физическим процессам, на первом этапе выполняется интегрирование однородной системы на множестве жидких и пересекаемых ячеек по схеме типа предиктор-корректор:

$$\begin{aligned} \widetilde{\vec{q}}_i &= \vec{q}_i^{\,n} - \frac{\Delta t}{2}\left(\frac{\Delta F_k(\vec{z}^{\,n})}{h_k}\right)_i, \\ \vec{q}_i^{\,n+1} &= \vec{q}_i^{\,n} - \Delta t\left(\frac{\Delta F_k(\widetilde{\vec{z}})}{h_k}\right)_i; \end{aligned} \tag{3}$$

с потоком $\vec{F}_{k,i+1/2} = \vec{F}_k\big(\widetilde{\vec{z}}_i^{\,+}, \widetilde{\vec{z}}_{i+1}^{\,-}\big)$, определяемым по интерполированным на грани ячейки значениям по MUSCL схеме третьего порядка аппроксимации. Метод С.К. Годунова применяется для аппроксимации функции численного потока.

Базовую явную схему (3) можно записать в операторном виде:

$$\vec{q}_i^{\,n+1} = \vec{q}_i^{\,n} + \Delta t L_2\big(\Delta t, \vec{q}^{\,n}\big), \tag{4}$$

где $L_2(\cdot)$ обозначает дискретный двухшаговой оператор перехода (3). Далее определяется также вектор решения на промежуточном слое $\vec{q}^{\,\omega} = \omega\vec{q}^{\,n} + (1-\omega)\vec{q}^{\,n+1}$, ω_i – скалярный параметр в каждой ячейке, $0 \le \omega_i \le 1$. Тогда гибридная явно-неявная схема записывается в виде

$$\vec{q}_i^{\,n+1} = \vec{q}_i^{\,n} + \Delta t L_2\big(\omega_i\Delta t, \vec{q}^{\,\omega}\big). \tag{5}$$

Выбор ω_i делается так, чтобы обеспечивалось максимальное участие явной компоненты при условии невозрастания max нормы (НВМН) для случая линейных уравнений.

На втором этапе полученное решение корректируется компенсационным потоком правой части (1), и итоговая схема выглядит следующим образом:

$$\vec{q}_i^{\,n+1} = \vec{q}_i^{\,n} - \Delta t\left(\frac{\Delta F_k(\widetilde{\vec{z}})}{h_k}\right)_i - \frac{\Delta t s_f}{\omega_f V_i}\vec{F}_w\big(\vec{q}_i^{\,n+1}\big). \tag{6}$$

Итерационный метод Ньютона для (6) приводит к линейной системе относительно итерационных вариаций $\delta^s(\cdot) = (\cdot)^{n+1,s+1} - (\cdot)^{n+1,s}$, где s – итерационный индекс:

$$\begin{aligned}\left(I + \frac{\Delta t s_f}{\omega_f V_i}A_w^s\right)\delta^s\vec{q}_i = -\Delta^s\vec{q}_i - \Delta t\left(\frac{\Delta^s\vec{F}_k}{h_k}\right)_i - \\ -\frac{\Delta t s_f}{\omega_f V_i}\vec{F}_w\big(\vec{q}_i^{\,n+1,s}\big) - \Delta t\frac{\delta^s\vec{F}_{k,i+1/2} - \delta^s\vec{F}_{ki-1/2}}{h_k}.\end{aligned} \tag{7}$$

В этом уравнении $\Delta^s = (\cdot)^{n+1,s} - (\cdot)^n$ обозначает итерационное приращение на временном шаге, $A_w = \partial\vec{F}_w/\partial\vec{q}$. Используя приближенную линеаризацию численного потока $\delta^s\vec{F}_{k,i+1/2}$, итоговую систему можно записать в расщепленном операторном виде с верхне- (U) и нижнетреугольным (L) матричными операторами:

$$D\delta\vec{q} + L(\delta\vec{q}) + U(\delta\vec{q}) = \vec{R}, \tag{8}$$

где

$$D_i^s = I + \frac{\Delta t s_f}{\omega_f V_i}A_w^s + \frac{\Delta t(1-\omega_i)}{2h_k}\left[\big(|u_k| + c\big)_{i-1/2}^{\omega,s} + \big(|u_k| + c\big)_{i+1/2}^{\omega,s}\right],$$

$$\vec{R}_i^s = -\Delta^s\vec{q}_i - \Delta t\left(\frac{\Delta^s\vec{F}_k}{h_k}\right)_i - \frac{\Delta t s_f}{\omega_f V_i}\vec{F}_w\big(\vec{q}_i^{\,n+1,s}\big),$$

$$U(\delta\vec{q}) = \frac{\Delta t(1-\omega_{i+1})}{2h_k}\big(\delta^s\vec{f}_{k,i+1}^{\,\omega} - r_{k,i+1/2}^{\omega,s}\delta^s\vec{q}_{i+1}\big),$$

$$L(\delta\vec{q}) = \frac{\Delta t(1-\omega_{i-1})}{2h_k}\big(\delta^s\vec{f}_{k,i-1}^{\,\omega} - r_{k,i-1/2}^{\omega,s}\delta^s\vec{q}_{i-1}\big)$$

Данная система решается безматричным итерационным методом LU-SGS с приближенной факторизацией матрицы системы $A = L + D + U \approx (D+L)D^{-1}(D+U)$, который сводится к решению следующих подсистем с обходом по ячейкам в прямом и обратном направлениях (по индексу ячейки) соответственно:

$$\begin{aligned}\delta\vec{q}^{\,*} &= D^{-1}[\vec{R} - L(\delta\vec{q}^{\,*})],\\ \delta\vec{q}^{\,s} &= \delta\vec{q}^{\,*} - D^{-1}U(\delta\vec{q}^{\,s}).\end{aligned} \tag{9}$$

Третья глава посвящена задаче построения параллельного алгоритма для метода LU-SGS на кластерных системах, оборудованных множеством графических ускорителей. Рассматриваются особенности метода, приводящие к зависимостям по данным между соседними ячейками, что значительно затрудняет создание для него масштабируемого параллельного алгоритма. Выполнен обзор реализаций этого метода для систем с общей и распределенной памятью. В большинстве случаев для систем с распределенной памятью LU-SGS соблюдается лишь локально, на отдельных подобластях, каждая из которых назначается отдельному процессу. Для обмена данными между соседствующими граничными ячейками разных подобластей используются упрощенные процедуры без соблюдения зависимости по данным, требуемой в LU-SGS. В результате это может приводить к уменьшению скорости сходимости решения, которая имеет тенденцию к замедлению при увеличении числа процессов (подобластей) в параллельной задаче. На некоторых задачах для достижения заданной точности "упрощенным" методом требуется до 2-х раз больше итераций по сравнению с "точным" LU-SGS.

Еще одной альтернативой избавления от зависимости по данным при распараллеливании является модифицирование самого метода LU-SGS. Такой подход реализован в DP-LUR. В нем работа с неявными компонентами, определяемыми операторами $L(\delta\vec{q})$ и $U(\delta\vec{q})$ (8) заменяется на релаксационную процедуру, фактически реализующую для решения (8) метод Якоби. Данный метод позволяет полностью избавиться от зависимости по данным и провести простое распараллеливание как для явных схем. Однако платой за такое упрощение является бо́льшая вычислительная сложность, т.е. увеличенное время счета, ведь теперь внутри каждой ньютоновской итерации необходимо выполнять $2-8$ релаксационных итераций, в результате суммарное время счета для падения невязки до заданного значения оказывается почти вдвое больше при использовании DP-LUR по сравнению с LU-SGS.

В тех немногих работах, где все-таки приводятся параллельные алгоритмы, *в точности* соблюдающие метод LU-SGS на *всей* расчетной области (а не на отдельных подобластях), масштабируемость реализаций оказывается весьма ограниченной. Так для *wavefront* алгоритма на декартовой сетке $256 \times 256 \times 128$ масштабируемость составила всего лишь 73.6 % на 4 ядрах CPU (21 млн ячеек на ядро). В другом параллельном алгоритме для неструктурированных сеток ячейки делятся на несколько подмножеств, состоящих из граничных и внутренних ячеек подобластей. Расчет всех внутренних ячеек подобластей выполняется параллельно в разных процессах, однако оставшиеся граничные ячейки обсчитываются по сути в квази-последовательном режиме, что приводит в итоге к ограниченной масштабируемости рассматриваемого параллельного алгоритма – на 50 процессорах эффективность (отношение достигнутого ускорения к линейному ускорению времени счета) составила 64 – 72 % в задаче с 600 тыс. ячеек.

Исходя из вышесказанного, можно выделить два ключевых свойства, которыми должны обладать масштабируемый параллельный алгоритм и соответствующая программная реализация:

– сведение к минимуму, а в лучшем случае – полное исключение последовательного исполнения части вычислений и/или простоев одного или нескольких параллельно исполняющихся процессов. Это требование рассматривается при условии реализации алгоритма на гипотетической вычислительной системе с бесконечно быстрой сетью, т.е. с нулевыми накладными расходами на обмен данными. Иными словами, алгоритм должен обладать близким к линейному масштабированием при условии мгновенной передачи сообщений между вычислительными ядрами;
– организация межпроцессного обмена данными на фоне “полезных” (содержательных с т.з. решаемой задачи) вычислений. Это требование уже рассматривается с точки зрения разработки реализаций для реальных вычислительных систем, обладающих сетями с ограниченной пропускной способностью и ненулевыми задержками при передаче данных. Параллельный алгоритм будет обладать бóльшей масштабируемостью, если он позволяет минимизировать возможные простои вычислителей, связанные с пересылками сообщений между процессами, за счет совмещения по времени межпроцессных коммуникаций и вычислений.

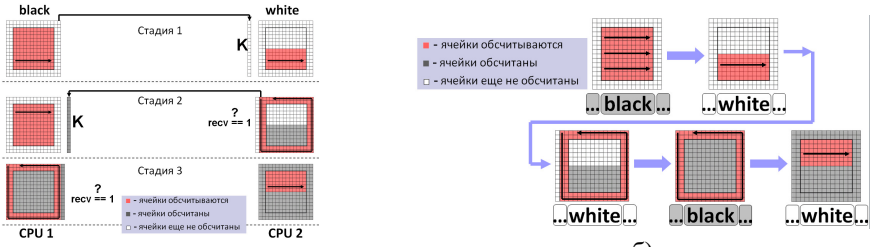
Рассмотренные выше параллельные алгоритмы для LU-SGS не удовлетворяют в той или иной степени этим двум свойствам, вследствие чего их программные реализации обладают весьма ограниченной масштабируемостью. Представленный же ниже параллельный алгоритм для LU-SGS (*в точности* реализующий вычисления своего последовательного прототипа на *всей* расчетной области), напротив, удовлетворяет указанным свойствам, что позволило в итоге соответствующей программной реализации с использованием CUDA и MPI достичь масштабируемости, близкой к линейной, на вычислительных системах с несколькими сотнями графических ускорителей.

Далее приводится (для простоты изложения в двумерном случае) разработанный в рамках данной работы параллельный алгоритм сначала для случая систем с распределенной памятью и последовательным исполнением внутри каждого параллельного процесса. После декомпозиции области с декартовой сеткой каждому процессу назначается подобласть, топологически эквивалентная четырехугольнику, и все они располагаются «стык в стык», т.е. подобласти также являются элементами структурированной (макро-)сетки и делятся на два типа – «черные» и «белые» (black и white) – так, что они формируют «шахматную» структуру (все геометрические «соседи» каждой подобласти имеют отличный от нее тип). Для каждого типа подобласти задается свой порядок операций. Для решения первой подсистемы (9):

В “черных” подобластях:

1. Запускаются пересылки граничных ячеек K соседним (“белым”) подобластям (Рис. 2 а); в “белых” подобластях эти ячейки считаются еще не

обсчитанными. На фоне этих пересылок выполняется обход всех ячеек, кроме тех, которые образуют “кольцо” из двойного ряда граничных ячеек (всегда будут отсылаться немодифицированные ячейки K).



а) б)

Рис. 2 — Последовательность операций в “черных” и “белых” блоках в параллельном алгоритме (а) и эквивалентный последовательный обход всей области (б)

2. Выполняется ожидание получения ghost ячеек от “белых” соседей (которые уже обсчитаны).

3. Выполняется обход по оставшимся ячейкам из двойного граничного кольца.

В “белых” подобластях:

1. Выполняется обход по первой половине внутренних ячеек.

2. Выполняется ожидание получения ghost ячеек K от “черных” соседей (которые уже обсчитаны)

3. Выполняется обход по граничным ячейкам.

4. Запускаются пересылки граничных ячеек K соседним (“черным”) подобластям; в “черных” подобластях эти ячейки считаются уже обсчитанными. На фоне этих пересылок выполняется обход по оставшейся второй половине внутренних ячеек.

Обратный обход для решения второй подсистемы (9) строится в соответствии с прямым: в black он начинается с “двойного периметра” приграничных ячеек в обратном порядке и заканчивается на внутренней оставшейся их части (так же в обратном порядке); в white сначала выполняется обход второй половины внутренних ячеек, затем граничных и, наконец, оставшейся первой части (в обратном порядке). Все необходимые пересылки ячеек выполняются с последующей проверкой их завершения.

Для доказательства корректности предлагаемого параллельного алгоритма строится последовательный глобальный обход ячеек по всей области (Рис. 2 б): сначала выполняется обход всех ячеек без “двойного периметра” во всех блоках black в произвольном порядке, затем в каждом white первой половины внутренних ячеек и граничных ячеек (последовательность обхода самих блоков white произвольная), после обсчитываются все “двойные периметры”

приграничных ячеек в black в произвольном порядке блоков, затем оставшиеся вторые половины внутренних ячеек в white (также произвольно).

Дальнейшая адаптация параллельного алгоритма на системы с GPU выполняется с помощью задачи о раскраске графа: нужно “раскрасить” ячейки сетки минимальным количеством цветов так, чтобы любые геометрически соседние (по граням) всегда были разных цветов. В таком случае можно запускать на GPU параллельный счет сначала по ячейкам 1-го цвета, затем 2-го и т.д. – результат работы параллельного алгоритма будет всегда корректным, поскольку эквивалентный последовательный обход по ячейкам, дающий в точности такое же решение, выглядит следующим образом: сначала последовательно обходятся все ячейки 1-го цвета (в произвольном порядке), затем аналогично ячейки 2-го цвета и т.д.

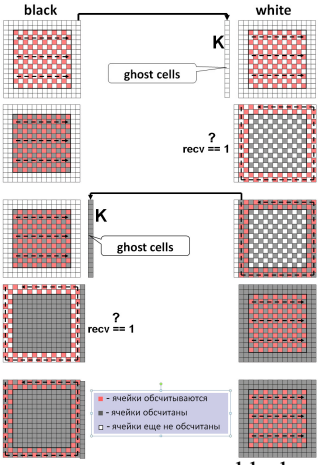


Рис. 3 — Последовательность счета ячеек в black и white блоках при расчете на множестве GPU

Решение задачи о раскраске графа для декартовых (структурированных) сеток по сути являет собой свойство автомодельности предлагаемого параллельного алгоритма – разделение ячеек внутри подобласти (внутренней или граничной части блока) на два множества происходит аналогично разбиению блоков исходной расчетной области: две геометрически соседних ячейки всегда принадлежат разным множествам. Таким образом, два этих множества вновь образуют “шахматное” разбиение ячеек. В итоге, счет каждой подобласти блока – внутренней или граничной его части – выполняется в два последовательных этапа: сначала на GPU запускается счет ячеек первого цвета (при этом все соседние ячейки будут второго цвета, которые в данной подобласти будут всегда еще не обсчитанными), затем, после его завершения, запускается счет ячеек второго цвета (все соседние ячейки будут уже первого цвета, которые в данной подобласти будут всегда уже обсчитанными). Обращение к ячейкам из другой

подобласти всегда будет выполняться как к обсчитанным или необсчитанным – порядок в этом случае определяется порядком обсчета подобластей в блоке (black или white). Итоговый параллельный алгоритм для LU-SGS на системах с множеством графических ускорителей представлен на Рис. 3. Ключевой его особенностью является возможность полного совмещения по времени вычислений в GPU и обмена данными (в граничных ячейках) между ними, что в итоге позволяет достичь высокой масштабируемости на кластерных систмах, однако это требует соответсвующей поддержки в программной реализации алгоритма.

В **четвертой главе** описываются детали программной реализации параллельного алгоритма с использованием технологий CUDA и MPI. Ключевым моментом в ней стало использование полностью асинхронной схемы многоступенчатой передачи данных между графическими ускорителями (GPU VRAM → Pci-e → CPU RAM → Infiniband → CPU RAM → Pci-e → GPU VRAM), позволяющей совместить по времени выполнение *ядер* (*kernels* – функций запускаемых с CPU на GPU) с операциями копирования данных, Рис. 4.

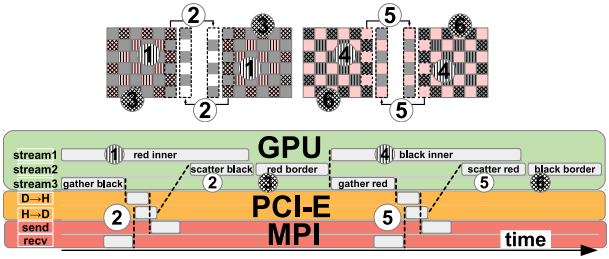


Рис. 4 — Порядок счета и обмена ячеек между блоками (вверху); схема счета в GPU и обмена данными между ними с совмещением по времени (внизу).

Особо стоит отметить, что данная схема реализована для, строго говоря, *неявной* схемы 6 с *точным* соблюдением итеративного метода LU-SGS на *всей* расчетной области с весьма сложной структурой зависимостей по данным. Напротив, для реализаций более простых методов на вычислительных системах с графическими ускорителями, в частности, метода Якоби, часто используются более примитивные (и существенно более простые в реализации и отладке) схемы организации вычислений и обмена данными без их совмещения по времени, Рис. 5, которые в итоге значительно ограничивают масштабируемость решателя на большое число GPU.

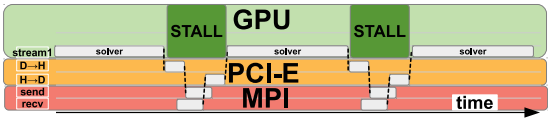
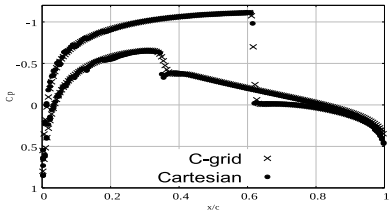
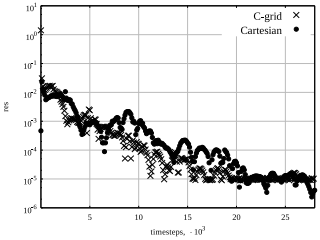


Рис. 5 — Упрощенная схема организации вычислений на GPU без совмещения по времени с обменом данными между GPU, приводящая к простоям в них.

В **пятой главе** приводятся результаты вычислительных экспериментов. В расчетах использовалось до 162 графических ускорителей. В частности, приведены результаты для профиля NACA0012, сравнение решений на связной сетке и на декартовой несвязной с методом свободной границы (решения оказались весьма близкими, разница коэффициентов Cl,Cd составила менее 1 %), Рис. 6, выполнено сравнение с решениями из других работ. Затронуты вопросы сходимости в зависимости от выбора различных «шахматных» обходов области.



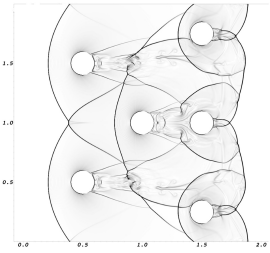
а)



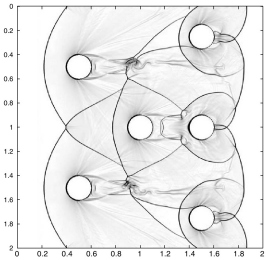
б)

Рис. 6 — Распределение Cp (а) и скорость сходимости (б) на согласованной (C-grid) и несвязной (Cartesian) сетках, NACA0012, $M = 0.8$, $\alpha = 1.25°$, число Куранта 10

Был выполнен расчет нестационарной задачи взаимодействия ударной волны с системой цилиндров и сравнение расчетов с одним из методов штрафных функций. Решения оказались очень близки, но при этом метод свободной границы не дает нефизичных возмущений вблизи поверхностей цилиндров Рис. 7. Это связано с тем, что в отличие от метода свободной границы в методе штрафных функций нет подсеточного разрешения геометрии; геометрия там представлена более грубо, с точностью до ячейки сеточного разбиения.



а)

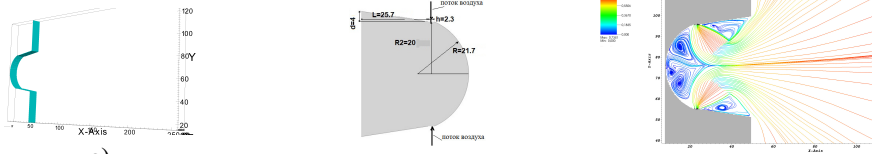


б)

Рис. 7 — Численная визуализация течения около системы цилиндров, $M = 3$, $t = 0.5$; а) настоящий метод свободной границы, б) метод штрафных функций.

Проведено моделирование макета двухщелевого тягового устройства в трехмерной пространственной постановке, геометрия которого представляет собой цилиндрическую поверхность (Рис. 8 а). Во внутреннюю его область

через щели ширины $h = 2.3$ мм в начальный момент времени подается постоянный звуковой поток воздуха с давлением и температурой торможения соответственно 21.8 атм и $308.5°K$ (Рис. 8 б). Предполагалось, что изначально в устройстве и в пространстве вне его (во всей расчетной области) находится неподвижный воздух при нормальных условиях. На Рис.8 в изображены линии тока, которые хорошо показывают вихревые структуры, образующиеся вблизи тяговой стенки, а течение при этом подобно течению в сопловом устройстве с центральным телом.



а) б) в)

Рис. 8 — а) Геометрия макета двухщелевого тягового устройства; б) Схема устройства с размерами в мм; в) Линии тока при $t = 3750$ с.

Проведено моделирование трехмерного обтекания модели пассажирского самолета DLR F6,Рис. 9

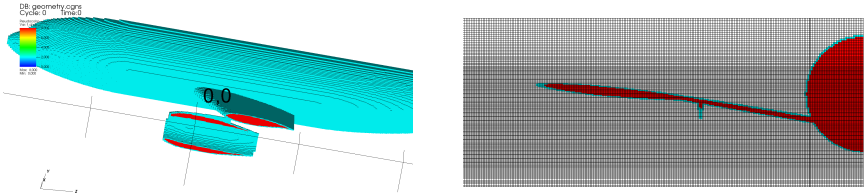


Рис. 9 — DLR F6; декартова сетка с пересекающимися (■) и внутренними ячейками (■)

График зависимости коэффициента подъемной силы Cl от угла атаки представлен на Рис. 10. Отличие от экспериментальных данных и других численных решений объясняется как недостаточностью модели Эйлера, так и низким разрешением сетки, в крупных ячейках которой происходит достаточно грубое линейное восполнение модели DLR F6. Последнее предположение подтверждается расчетом на более подробной сетке ($408 \times 520 \times 1256$): полученное в нем значение $C_l = 0.56$ лучше согласуется с другими решениями по сравнению с $C_l = 0.44$ на грубой сетке ($204 \times 260 \times 628$)

Шестая глава посвящена детальному исследованию эффективности и масштабируемости разработанного программного комплекса.

В зависимости от конкретных моделей вычислителей (центрального процессора и графического ускорителя) ускорение на одном GPU по сравнению с

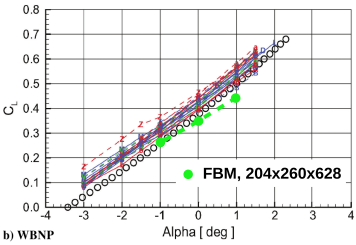


Fig. 10 Composite lift curve results for case 2: $M_\infty = 0.75$.

Рис. 10 — Модель DLR F6, $M = 0.75$; зависимость коэффициента подъемной силы C_l от угла атаки, сравнение с экспериментальными данными (обозначены как “$\circ$”) и решениями, полученными другими программами

.

одним ядром CPU варьировалось от 13 до 30 раз. Тесты показали, для эффективной работы солвера с графическими ускорилями необходимо использовать сетки с разрешением не менее 60 тыс. ячеек / GPU.

Были обнаружены некоторые особенности работы библиотек CUDA и MPI, которые приводили к блокированию обмена данными между графическими ускорителями на фоне вычислений в них, Рис. 11,12.

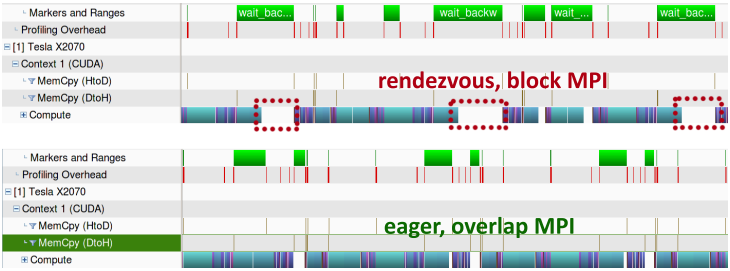


Рис. 11 — Трасса для разработанного программного комплекса с протоколами *rendezvous* (вверху) и *eager* (внизу) передачи сообщений в MPI

В результате проведенного исследования удалось исключить указанные простои в GPU посредством принудительного включения протокола *eager* в MPI (обмен данными «без рукопожатий» вместо используемого по умолчанию *rendevous* – обмен данными «с рукопожатием») и запретом уменьшения *локальной* памяти для тредов в GPU с помощью вызова одной из функций CUDA API. Далее было исследовано влияние данных настроек при проведении крупномасштабных расчетов, Рис. 13. Как видно, масштабируемость комплекса на 180 GPU составила 81 % (относительно 15 GPU) с настройками MPI по умолчанию, а при принудительном использовании *eager* протокола – поднялась до 92 % с соответствующим 27 %-м ростом производительности на данном числе GPU.

Особо стоит отметить тот факт, что такой значительный рост был достигнут без модификации исходного кода решателя в разработанном программном

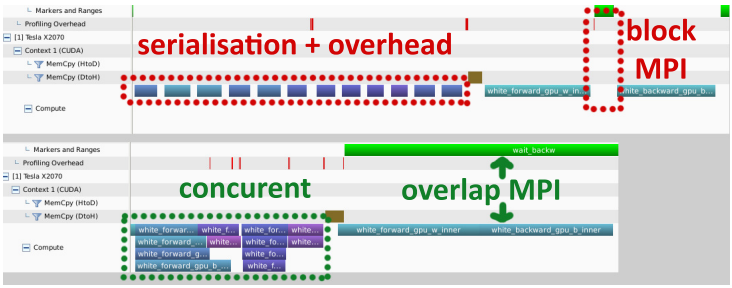


Рис. 12 — Трасса для разработанного программного комплекса с настройками по умолчанию (вверху) и запретом уменьшения размера локальной памяти для треда в GPU (внизу)

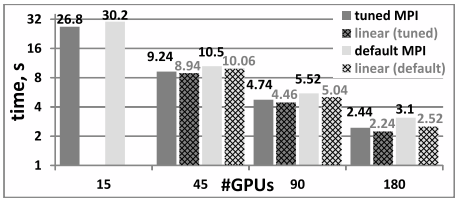


Рис. 13 — Время счета (сек), DLR F6, $C = 8$, 94 млн ячеек, 10 шагов по времени с настройками MPI по умолчанию (default MPI) и оптимизированными настройками (tuned MPI)

комплексе, а лишь путем настройки параметров библиотеки MPI непосредственно перед запуском приложения, однако это потребовало нетривиального исследования особенностей взаимодействия библиотек CUDA и MPI и учета деталей реализации в последней из них.

Для исследования эффективности работы реализованного программного комплекса были проведены замеры времени работы на суперкомпьютере “Ломоносов” с использованием до 768 GPU. Как видно, Рис. 14, масштабируемость решателя на сетке с 150 млн ячеек оказалась достаточно близка к линейной и составила 75 % на 768 GPU (≈200 тыс ячеек/GPU) (относительно 32 GPU – минимальное их число, в памяти которых помещалась расчетная сетка).

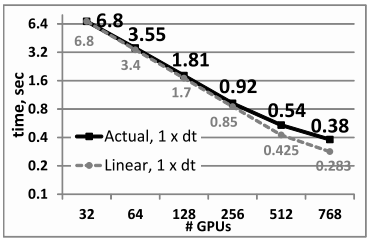


Рис. 14 — Время счета (сек), 150 млн ячеек, 1 шаг по времени, СК “Ломоносов”

В **Заключении** (седьмой главе) приведены основные результаты работы, а также рассматриваются перспективы дальнейших работ. В частности, затрагиваются проблемы, связанные с использованием адаптивных сеток на основе восьмеричных деревьев, приводится параллельный алгоритм для LU-SGS на сетках данного типа, в основе которого лежит специальный алгоритм раскраски ячеек. Рассматривается также разрабатываемая в настоящее время библиотека для динамического перестроения сеток (на основе восьмеричных деревьев), которая в отличие от *P4est* и других подобных программ позволяет это делать *полностью* на GPU, без накладных расходов по перемещению данных между GPU и CPU. Предварительные тесты показали, что можно работать с адаптивными сетками *полностью* на GPU с производительностью, сравнимой с *P4est* на CPU, исключив при этом значительный по времени этап обмена данными между CPU и GPU. Данный результат является очередным шагом к созданию высокомасштабируемого программного комплекса для решения задач газовой динамики с подвижной геометрией на динамически адаптируемых сетках, предназначенного для работы на системах с большим числом вычислителей с массивно-параллельными архитектурами.

Основные результаты работы

В процессе работы над диссертацией получены следующие результаты:

1. Предложено обобщение математической и численной модели свободной границы под специфику архитектуры графических ускорителей.
2. На основе модели свободной границы, гибридной явно-неявной схемы и итерационного метода LU-SGS разработан параллельный алгоритм для решения задач газовой динамики на вычислительных системах с графическими ускорителями.
3. Реализован программный комплекс на основе разработанного алгоритма с использованием технологий CUDA и MPI. Показана эффективность счета до 75% при использовании 768 GPU суперкомпьютера «Ломоносов».
4. С помощью разработанного программного комплекса проведены расчеты ряда газодинамических течений, включая моделирование обтекания вокруг профиля NACA0012 и DLR F6 на декартовой структурированной сетке, которые подтвердили высокую эффективность его работы.

Публикации автора по теме диссертации

[1] *Меньшов И. С., Павлухин П. В.* Эффективный параллельный метод сквозного счета задач аэродинамики на несвязных декартовых сетках // Журнал вычислительной математики и математической физики. — 2016. Т. 56. С. 1677–1691.

[2] *Pavlukhin P., Menshov I.* On Implementation High-Scalable CFD Solvers for Hybrid Clusters with Massively-Parallel Architectures // ParallelComputing Technologies / ed. by V. Malyshkin. – Cham : Springer International Publishing, 2015. – P. 436–444.

[3] *Павлухин П. В., Меньшов И.С.* Эффективная параллельная реализация метода LU-SGS для задач газовой динамики // Научный вестник МГТУ ГА. – 2011. – № 165. – С. 46–55.

[4] *Павлухин П. В.* Реализация параллельного метода lu-sgs для задач газовой динамики на кластерных системах с графическими ускорителями // Вестник Нижегородского университета им. Н.И. Лобачевского. – 2013. – № 1. – С. 213–218.

[5] *Pavlukhin P., Menshov I.* Highly Scalable Implementation of an Implicit Matrix-free Solver for Gas Dynamics on GPU-accelerated Clusters // J. Supercomput. – Hingham, MA, USA, 2017. – Vol. 73, № 2. – P. 631–638.

[6] *Меньшов И. С., Павлухин П. В.* Численное решение задач газовой динамики на декартовых сетках с применением гибридных вычислительных систем // Препринты ИПМ им.М.В.Келдыша. – 2014. – № 92. – URL: http://library.keldysh.ru/preprint.asp?id=2014-92 (дата обращения: 01.07.2019).

Павлухин Павел Викторович

Эффективное решение задач газовой динамики на кластерных системах с графическими ускорителями

Автореф. дис. на соискание ученой степени канд. физ.-мат. наук

Подписано в печать 12.08.2019. Заказ А-9
Формат 60×90/16. Усл. печ. л. 1,0. Тираж 70 экз.
ИПМ им. М.В. Келдыша РАН. 125047, Москва, Миусская пл., 4