DOI: 10.15514/ISPRAS-2023-35(2)-13

Об использовании открытых сторонних библиотек при программной реализации вихревых методов вычислительной гидродинамики

[1,2] *И.К. Марчевский, ORCID: 0000-0003-4899-4828 <iliamarchevsky@mail.ru>*
[1] *Ю.А. Измайлова, ORCID: 0000-0003-1137-3235 <yulia.izmailova@mail.ru>*
[1] *М.А. Ерофеева, ORCID: 0009-0000-1061-6974 <mariya.a.erofeeva@gmail.com>*
[1] *Д.Ю. Кобзарь, ORCID: 0009-0006-0746-6497 <cobzardash@yandex.ru>*

[1] *Московский государственный технический университет имени Н.Э. Баумана, 105005, Россия, г. Москва, ул. 2-я Бауманская, д. 5, к. 1*
[2] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*

Аннотация. Обсуждается наиболее общая структура вычислительного алгоритма, реализующего бессеточные лагранжевы методы вычислительной гидродинамики. Затронуты не только основные, но и «вспомогательные», но оттого не менее важные процедуры, реализациям которых часто практически не уделяется внимания. Последнее может приводить к значительному дисбалансу и снижению эффективности кодов, в которых «основные» вычислительные операции существенно оптимизированы. Обсуждаются авторские коды VM2D и VM3D, развитие которых на первом («поисковом») этапе шло главным образом по пути выбора и реализации необходимых математических моделей, а достижение приемлемой эффективности обеспечивалось «экстенсивным» путем – привлечением значительных вычислительных ресурсов (в частности, видеокарт). Предпринята попытка сделать заключение о целесообразности использования существующих сторонних библиотек для выполнения операций вычислительной геометрии, решения задач на графах и т.п.

Ключевые слова: бессеточные лагранжевы методы вычислительной гидродинамики; вихревые методы; код VM2D; код VM3D; алгоритмы вычислительной геометрии; свободные библиотеки.

Для цитирования: Марчевский И.К., Измайлова Ю.А., Ерофеева М.А., Кобзарь Д.Ю. Об использовании открытых сторонних библиотек при программной реализации вихревых методов вычислительной гидродинамики. Труды ИСП РАН, том 35, вып. 2, 2023 г., стр. 181-200. DOI: 10.15514/ISPRAS–2023–35(2)–13

On open third-party libraries usage in implementation of vortex particle methods of computational fluid dynamics

[1,2] *I.K. Marchevsky, ORCID: 0000-0003-4899-4828 <iliamarchevsky@mail.ru>*
[1] *Yu.A. Izmailova, ORCID: 0000-0003-1137-3235 <yulia.izmailova@mail.ru>*
[1] *M.A. Erofeeva, ORCID: 0009-0000-1061-6974 <mariya.a.erofeeva@gmail.com>*
[1] *D.Yu. Kobzar, ORCID: 0009-0006-0746-6497 <cobzardash@yandex.ru>*

[1] *Bauman Moscow State Technical University, 105005, Moscow, 2-nd Baumanskaya st., 5.*
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*

Abstract. The most general structure of a computational algorithm that implements meshless Lagrangian methods of computational fluid dynamics is discussed. Not only the main ones are touched upon, but also "auxiliary", but therefore no less important procedures, which implementation is often practically ignored. The latter can lead to a significant imbalance and decrease in the efficiency of codes in which the "basic" computational operations are significantly optimized. The author's in-house codes VM2D and VM3D are discussed, the development of which at the first ("exploratory") stage proceeded mainly along the path of choosing and implementing the necessary mathematical models, and the achievement of acceptable efficiency was ensured by an "extensive" way – involving significant computing resources (in particular, graphical accelerators). An attempt was made to make a conclusion about the expediency of using existing third-party libraries to perform computational geometry operations, solve problems on graphs, etc..

Keywords: meshless Lagrangian CFD methods; vortex particle methods; VM2D code; VM3D code; computational geometry algorithms; open source libraries

For citation: Marchevsky I.K., Izmailova Yu.A., Erofeeva M.A., Kobzar D.Yu. On open third-party libraries usage in implementation of vortex particle methods of computational fluid dynamics. Trudy ISP RAN/Proc. ISP RAS, vol. 35, issue 2, 2023. pp. 181-200 (in Russian). DOI: 10.15514/ISPRAS-2023-35(2)-13

1. Введение

Задачи вычислительной гидро- и газовой динамики традиционно и вполне заслуженно относят к наиболее вычислительно сложным, прежде всего по причине нелинейности уравнений и определяющих соотношений, а также необходимости воспроизведения процессов, протекающих на различных пространственных и временных масштабах – от малых и даже подсеточных (если говорить об использовании традиционных численных методов) до больших, соизмеримых со всей расчетной областью.

В то же время в инженерной практике приходится сталкиваться с целым классом задач моделирования, в которых течение среды как таковое не представляет самостоятельного интереса, однако его необходимо учитывать, чтобы правильно определять величины аэрогидродинамических нагрузок, действующих на находящиеся в потоке тела. Приближенные методы, появившиеся еще в «докомпьютерную» эру, такие как расчет по стационарным коэффициентам или использование коэффициентов присоединенных масс (и те, и другие могут быть определены экспериментально или в результате моделирования сравнительно простого режима обтекания – стационарного или потенциального соответственно), вполне пригодны и не потеряли актуальности, но лишь для узкого диапазона условий: (квази)стационарного обтекания неподвижных или медленно движущихся тел в практически постоянном потоке, или, наоборот, вибрирующих в неподвижной среде тел. В существенно же нестационарных условиях обтекания, сопровождаемого интенсивным вихреобразованием, отрывами потока и т.п., указанные методики неприменимы.

Простого и универсального решения у данной проблемы нет; в то же время, если ограничиться рассмотрением сравнительно простых моделей среды – однофазной,

однородной, несжимаемой, нетеплопроводной – то подобные течения возможно моделировать с использованием т.н. вихревых методов вычислительной гидродинамики. Последние, по существу, восходят к классическим работам Н.Е. Жуковского, связавшего многие наблюдаемые гидродинамические эффекты с вихревой природой течения. Не вдаваясь в историю их развития (см. обстоятельные обзоры [1–3]), отметим лишь следующее обстоятельство: после этапа интенсивного развития в 60-х годах XX века – в начале «компьютерной эры» (а мощности ЭВМ того времени позволяли реализовать лишь их, для более привычных на сегодня сеточных методов вычислительных ресурсов было явно недостаточно) – наступил этап некоторого «застоя», во время которого, наоборот, бурное развитие получили сеточные методы. Их очевидными преимуществами являются универсальность, возможность моделирования в широком диапазоне свойств среды в частности и постановок задач в целом, сравнительная простота перехода к решению многодисциплинарных задач и др. Вихревые методы при этом заняли свою, довольно узкую нишу, и продолжили планомерно развиваться, превратившись на сегодняшний день в довольно мощный инструмент инженерного анализа.

2. Структура вычислительного алгоритма в вихревых методах

Особенностью вихревых методов в частности и лагранжевых методов частиц в целом, существенно отличающей их от классических сеточных, является структура «вычислительного конвейера». В алгоритмах сеточных методов на каждом шаге расчета по времени решается некоторая типовая задача: строится разностный аналог оператора задачи (как правило, дифференциального), представляющий собой систему алгебраических уравнений, которая, в свою очередь, решается тем или иным итерационным методом; при этом могут активно использоваться различные схемы интерполяции, производиться решения некоторых вспомогательных задач и т.п. При решении многодисциплинарных задач в большинстве случаев используют схемы расщепления, рассматривая таким образом на каждом «подшаге» задачу только для одной неизвестной величины, однако это не нарушает общей логики – выполнения сходных операций для всех узлов или ячеек сетки и решения разреженных систем уравнений.

В вихревых методах на каждом шаге расчета приходится решать достаточно большое количество совершенно разнотипных задач, конкретный перечень которых зависит от рассматриваемой модификации. Если попытаться представить в наиболее общем виде алгоритм, выполняемый на каждом временном шаге, то можно выделить следующие наиболее трудозатратные блоки операций:

1) моделирование процесса генерации завихренности на обтекаемых поверхностях;
2) вычисление скоростей движения вихревых частиц;
3) восстановление поля скоростей и поля давления; расчет гидродинамических нагрузок, действующих на обтекаемые тела;
4) моделирование эволюции завихренности в области течения и движения обтекаемых поверхностей.

Примерная схема вычислительного алгоритма представлена на рис. 1.

При этом следует иметь в виду, что каждый из отмеченных пунктов включает в себя целый перечень подзадач, часто совершенно различных по своим «характеристикам»; например, к первому блоку относятся такие процедуры, как

- построение системы линейных алгебраических уравнений – дискретного аналога граничного (вообще говоря, сингулярного или гиперсингулярного) интегрального уравнения, где нетривиальным является не только шаг формирования матрицы, аппроксимирующей оператор, но и расчет правой части;
- решение возникающей линейной системы с полностью заполненной матрицей.

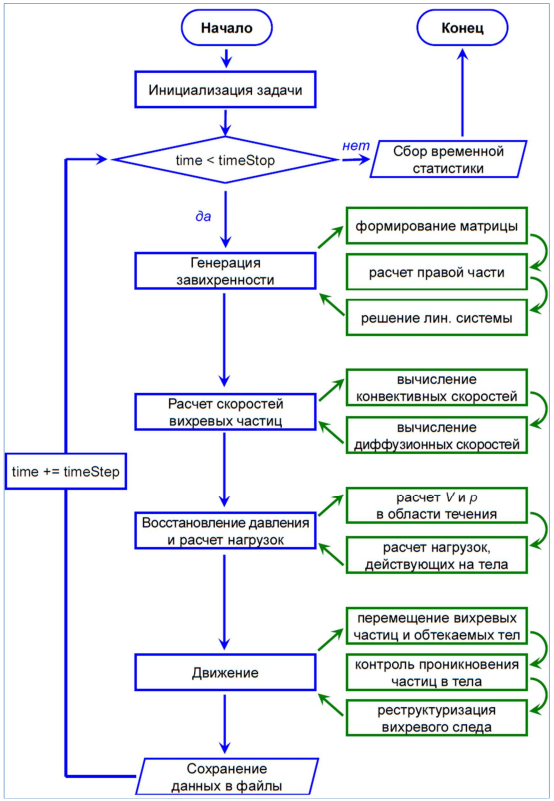

Рис. 1. Общая схема алгоритма вихревых методов
Fig. 1. The general scheme of the algorithm of vortex methods

Второй блок включает в себя операции вычисления скоростей движения вихревых частиц, как конвективных, т.е. фактически скоростей среды в точках расположения вихревых частиц, так и диффузионных, выражающих влияние вязкости среды. Задача расчета конвективных скоростей вихревых частиц аналогична задаче N тел. Отметим, что известно несколько подходов к моделированию вязких эффектов в вихревых методах; в качестве альтернативы упомянутому методу диффузионных скоростей могут быть использованы метод случайных блужданий (Random walk), метод обмена интенсивностями (Particle strength exchange), метод деформирования ядра вихря (Core spreading) и некоторые другие; данные методы далее рассматривать не будем; общее представление о них можно получить из обзора [3].

Для расчета гидродинамических нагрузок, выполняемого в рамках третьего блока, могут применяться различные алгоритмы; наиболее удобным представляется использование обобщений интегралов Бернулли и Коши – Лагранжа [4] и большого количества их модификаций, адаптированных для решения задач различных типов [5–7]. Возможно как восстановление поля давления с последующим его интегрированием по поверхности тела, так и определение «интегральных» величин нагрузок – главного вектора сил и главного момента. Соответствующие вычислительные алгоритмы либо сравнительно просты, либо аналогичны тем, что используются при решении задачи типа N тел. Операция восстановления поля скоростей среды в заданных точках по существу идентична расчету конвективных скоростей вихревых частиц.

Четвертый блок операций предполагает реализацию таких алгоритмов как

- расчет перемещений вихревых частиц и обтекаемых поверхностей, включая «сопряжение» представления завихренности в виде вихревого слоя на обтекаемой поверхности и ее же представления в виде вихревых структур в области течения;
- контроль проникновения завихренности внутрь обтекаемых поверхностей;
- реструктуризация вихревого следа.

В абсолютном большинстве публикаций по вихревым методам авторы уделяют внимание лишь отдельным вопросам, чаще всего – реализации алгоритмов моделирования движения вихревых частиц, а также схемам дискретизации граничного интегрального уравнения. Не подвергая сомнению тот факт, что данные вопросы являются наиболее сложными и, как следствие, интересными для исследования, отметим, что в области решения указанных проблем к настоящему времени накоплен значительный опыт и достигнут существенный прогресс: в литературе известны приближенные быстрые методы, имеющие квазилинейную вычислительную сложность, позволяющие эффективно рассчитывать взаимное влияние вихревых частиц, а также разработаны весьма точные, робастные, и при этом обладающие невысокой вычислительной сложностью схемы для численного решения ГИУ (расшифровки этой и остальных используемых в статье аббревиатур, исключая названия организаций и названия конкретных кодов, если последние не имеют непосредственной расшифровки, для удобства сведены в таблицу в конце статьи). Тем не менее, перечисленные алгоритмы во многих случаях пока находятся на разных стадиях методических исследований и «отработки», и далеко не всегда могут быть включены в программные реализации.

3. Программные реализации вихревых методов

Несмотря на то, что вихревые методы имеют по крайней мере 60-летнюю историю развития, как в нашей стране, так и за рубежом, их современные модификации практически не реализованы в программных комплексах, которые были бы доступными широкому кругу исследователей. Известны реализации так называемых панельных методов, которые можно рассматривать лишь как самые простые версии вихревых методов, однако они не раскрывают истинный потенциал современных математических моделей и алгоритмов. Подобное положение дел, безусловно, негативно сказывается на популярности вихревых методов среди исследователей и расчетчиков.

Программные реализации вихревых методов стали появляться лишь в последнее время, начиная примерно с 2017-2018 года, некоторые из них поддерживаются и развиваются, при этом практически все они представлены с открытым исходным кодом, что вписывается в актуальную тенденцию отказа от использования коммерческого программного обеспечения для моделирования в механике сплошной среды и перехода к свободно распространяемому ПО с открытыми исходными кодами.

Одной из немногих доступных реализаций вихревых методов применительно к моделированию плоских течений является опубликованный в 2018 году программный комплекс vvflow, доступный по ссылке https://packagecloud.io/vvflow/stable, в котором реализован метод вязких вихревых доменов, разработанный проф. Г.Я. Дынниковой [5, 8–10]. Возможности программного комплекса позволяют, в том числе решать сопряженные задачи в гидроупругой постановке для систем недеформируемых профилей с упругими связями между ними. Из технологий параллельных вычислений в vvflow использована лишь OpenMP, что позволяет проводить расчеты только на системах с общей памятью.

В 2018 году появились пакеты с открытым кодом Omega2D и Omega3D, (доступные соответственно из репозиториев http://github.com/Applied-Scientific-Research/Omega2D и http://github.com/Applied-Scientific-Research/Omega3D), реализующие методы вихревых частиц для моделирования плоских и пространственных течений и описанные в [11]. Данные коды позиционируются разработчиками как платформы для разработки и реализации

различных модификаций вихревых методов; существенное внимание уделено вопросам визуализации. На данный момент в указанных пакетах нет возможности моделирования обтекания подвижных профилей/тел, не вполне ясен вопрос о точности вычисления действующих на обтекаемые поверхности гидродинамических нагрузок (по умолчанию их вычисление вообще не производится). Кроме того, при расчете взаимного влияния вихревых частиц используется только «прямая» реализация закона Био – Савара (т. е. алгоритм квадратичной вычислительной сложности), ускорение вычислений может быть достигнуто лишь за счет подключения внешних библиотек, позволяющих выполнять часть вычислительной работы на графических ускорителях.

В 2019 году появился код FLOWVPM, являющийся частью более крупного проекта FLOWUnsteady (http://github.com/byuflowlab/flowunsteady), свободно доступный с исходным кодом, написанным на языке Julia. Пакет позволяет моделировать пространственные течения и обтекание тел, в том числе подвижных, сложной геометрической формы; описание реализованных в нем моделей и методов приведено в [12]. Технологии распараллеливания вычислений в доступной версии кода в явном виде не используются (однако средства языка Julia позволяют это делать неявно); для расчета скоростей вихревых частиц использован быстрый метод мультиполей. В качестве вихревых частиц используются вихревые сгустки (vortex blobs), таким образом, поле завихренности в общем случае является несоленоидальным.

В диссертациях [13, 14] описаны программные пакеты, реализующие метод вязких вихревых доменов для моделирования плоских течений, однако недоступные для широкого круга потенциальных пользователей. Известно также о разработанном проф. G. Morgenthal (ФРГ) и активно используемом в его научной школе пакете VXflow (https://www.igorkav.com/category/vxflow/), созданном для решения задач промышленной аэродинамики зданий и сооружений в двумерной и квази-трехмерной постановке; в данном пакете реализован метод случайных блужданий [15] и оригинальная модификация быстрого метода расчета скоростей вихревых частиц [16]; основные модели и алгоритмы, положенные в основу кода, описаны в [17]. Алгоритмы распараллеливания вычислений на кластерных системах и возможности выполнения расчетов на графических картах в VXflow не предусмотрены.

По публикациям известно также о существовании кодов, реализующих вихревые методы (метод дискретных вихрей в различных модификациях, метод замкнутых вихревых рамок), в ВВИА им. Н.Е. Жуковского, МГУ им. М.В. Ломоносова, НИИ Парашютостроения, ЦАГИ им. Н.Е. Жуковского, Санкт-Петербургском государственном морском техническом университете, Институте машиноведения им. А.А. Благонравова РАН, Институте математики им. С.Л. Соболева СО РАН и некоторых других российских организациях; из зарубежных разработок отметим оригинальные методы моделирования двумерных вязких течений, описанные и реализованные в коде, разработанном итальянскими авторами [18]; давно и успешно развиваются методы моделирования пространственных течений в Японии в научной школе проф. K. Kamemoto [19].

Значительный вклад как в развитие вихревых методов, особенно применительно к моделированию течений в пространственной постановке, так и в решение проблем их эффективной программной реализации внесли G.-H. Cottet (Франция), P.D. Koumoutsakos (Греция, Швейцария), G. Winkelmans (Бельгия) и др. Тем не менее, о разработанных ими кодах известно лишь по публикациям, для свободного доступа они недоступны. Известно также о создании программ авторами J.S. Uhlman, J.S. Marshal (США) и другими.

Коллективом авторов [20, 21] разработан пакет VM2D, свободно доступный по ссылке https://github.com/vortexmethods/VM2D с исходным кодом на языке C++. Он предназначен для решения широкого круга плоских задач, в том числе в сопряженной гидроупругой постановке. В отличие от остальных кодов в нем реализованы усовершенствованные схемы повышенной точности для моделирования генерации завихренности [22], имеется удобный

интерфейс для решения большого числа однотипных задач на кластерных системах. Код распараллелен с применением технологий OpenMP, MPI и CUDA, включая возможность их совместного использования. Однако многие из реализованных в нем алгоритмов являются «тривиальными», т.е. реализуют наиболее очевидные, но, к сожалению, далеко не самые эффективные способы выполнения соответствующих операций.

Аналогичным образом устроен и разрабатываемый авторами код VM3D [23], реализующий оригинальную модификацию вихревого метода для моделирования пространственных течений. Отметим, что для решения пространственных задач в VM3D реализован оригинальный алгоритм построения дискретного аналога граничного интегрального уравнения (также записываемого не так, как в большинстве известных реализаций), в основе которого – схема Галеркина. Для вычисления возникающих при этом двойных сингулярных интегралов по треугольным панелям можно вместо авторского алгоритма [24] использовать несколько менее эффективный, но весьма универсальный метод Тейлора – Даффи, достаточно подробно описанный в [25]; свободная и открытая реализация данного алгоритма имеется в составе кода scuff-em, предназначенного для решения задач электростатики и электродинамики и доступного из репозитория https://github.com/HomerReid/scuff-em под лицензией GPL-3 (модуль TaylorDuffy; пример вычисления интегралов имеется в документации к коду).

Далее рассмотрим пути оптимизации обсуждавшихся выше вычислительных алгоритмов, в основе которых лежит использование сторонних библиотек. При этом будем говорить лишь о высокоуровневых, или прикладных библиотеках, не «спускаясь» на уровень, к примеру, библиотек типа OpenMP или Intel TBB, которые, без сомнений, могут быть крайне эффективны при выполнении отдельных частных операций сравнительно «низкого» уровня (параллельное исполнение витков циклов, параллельная сортировка, операции редукции и конкурентного доступа к данным и т.п.). Не будем также обсуждать использование «сервисных» библиотек, обеспечивающих, к примеру, инициализацию задач и сохранение результатов расчетов. Укажем лишь, что для чтения исходных данных, параметров расчетов и разнообразных настроек, обычно хранимых в текстовых файлах, удобно использовать какой-либо универсальный парсер, например, toml11, доступный из репозитория https://github.com/ToruNiina/toml11 (лицензия MIT). При сохранении результатов расчетов, в том числе с целью визуализации целесообразно пользоваться возможностями открытой библиотеки VTK (https://www.vtk.org, лицензия BSD) и универсального постпроцессора Paraview (https://www.paraview.org, лицензия BSD)

4. Операции линейной алгебры

Несмотря на относительную простоту выполняемых в вихревых методах операций линейной алгебры (к примеру, решение линейной системы с заполненной хорошо обусловленной матрицей или обращение такой матрицы), сравнительно большая размерность задачи, достигающая, особенно при моделировании пространственного обтекания тел, десятков тысяч, требует эффективного выполнения соответствующих процедур.

4.1 Выполнение расчетов на CPU. Библиотека Eigen

При проведении вычислений на центральном процессоре представляется эффективным использование библиотеки Eigen. Библиотека развивается с 2006 года сообществом TuxFamily и Inria, последние версии доступны на сайте https://eigen.tuxfamily.org и распространяются под свободной лицензией MPL 2.0. Особенность библиотеки – отсутствие необходимости ее отдельной компиляции: исходный код представляет собой набор заголовочных файлов, непосредственно включаемых директивами #include в необходимые модули пользовательского ПО.

К ее преимуществам следует отнести простоту интеграции в код, отсутствие зависимостей с другими библиотеками, большое количество реализованных алгоритмов, близкую к естественной математической нотацию записи выражений. Важное значение имеет наличие качественно подготовленной обстоятельной документации. Имеется возможность выполнения некоторых операций в параллельном OpenMP-режиме на ЭВМ с общей памятью: к таковым относятся умножение матриц, LU-разложение с частичным выбором ведущего элемента (PartialPivLU), итерационное решение линейных систем методами сопряженных и бисопряженных градиентов (ConjugateGradients, BiCGStab), реализация метода наименьших квадратов с использованием метода сопряженных градиентов (LeastSquaresConjugateGradient). При наличии установленных на компьютере пользователя реализаций BLAS/LAPACK библиотека Eigen может выступать в качестве интерфейса к ним; то же относится к Intel MKL.

Высокая производительность Eigen достигается благодаря логике построения библиотеки на основе т.н. «ленивых» шаблонов: это означает (не погружаясь в подробности), что на выходе получается абстрактное синтаксическое дерево правой части исходного выражения в виде шаблонных объектов, известных на стадии компиляции. Это дерево превращается в реальный исполняемый код только при исполнении оператора присваивания. Обратной стороной такого подхода является «плохопредсказуемое» поведение при написании кода, в котором используются операции неявного приведения типов или автоматического вывода типа. К примеру, исполнение кода

```
Eigen::MatrixXd A({ { 1.0, 2.0 }, { 3.0, 4.0 } });
Eigen::MatrixXd B({ { 1.0, 0.0 }, { 0.0, 1.0 } });
auto C = A * B;
Eigen::MatrixXd R1 = C;
B(0,0) = B(1,1) = 2.0;
Eigen::MatrixXd R2 = C;
```

приведет к тому, что матрицы R1 и R2 будут отличаться в 2 раза, поскольку в данном случае переменная C имеет тип не матрицы, а некоторого «ленивого» шаблонного выражения, т.е. при ее определении никакого перемножения матриц в реальности не происходит. Решить эту проблему можно либо явным указанием типа переменной C (Eigen::MatrixXd), либо модификацией строки:

```
auto C = (A * B).eval();
```

смысл которой понятен из сокращенного названия операции «evaluate».

Аналогичное, на первый взгляд некорректное поведение, может наблюдаться при использовании тернарного оператора, а также при обработке выражений, в которых один и тот же объект встречается и слева, и справа от знака присваивания.

Таким образом, особенности внутреннего устройства библиотеки линейной алгебры Eigen, получить первоначальное представление о которых можно из документации (https://eigen.tuxfamily.org/dox/UserManual_UnderstandingEigen.html), необходимо учитывать при написании кода, чтобы обеспечить наиболее полное и эффективное использование ее возможностей.

4.2 Выполнение расчетов на GPU. Библиотеки CUDA

Возможности использования библиотеки Eigen при выполнении расчетов на графических картах, используя технологию Nvidia CUDA, ограничены работой с матрицами и векторами фиксированной размерности – 1…4-мерными. В этой связи использование Eigen для исполнения основной массы необходимых операций оказывается невозможным. Заменой ей может служить библиотека cuBLAS или cuSPARSE (для работы с заполненными и разреженными матрицами соответственно); обе входят в состав Nvidia Toolkit. Удобство их использования заключается в отсутствии необходимости писать код, компилируемый для

устройства компилятором `nvcc`, т.е. все используемые функции являются host-функциями. Также указанные библиотеки снабжены подробной документацией.

Недостаток использования cuBLAS/cuSPARSE с точки зрения пользователя, не владеющего «низкоуровневой» технологией работы с GPU, – ненаглядная и не интуитивно-понятная нотация при написании кода. В отличие от Eigen, библиотеки для работы с CUDA не предоставляют пользователю собственную «экосистему» типов и структур данных, а используются лишь для выполнения вычислительно трудоемких операций. При этом следует иметь в виду, что выполнение действий на устройстве (видеокарте) требует накладных расходов – копирования на устройство исходных данных, а затем копирования результата в обратном направлении. Говоря об операциях BLAS уровня 3 (решение линейных систем, умножение и обращение заполненных матриц), необходимо помнить, что их сложность пропорциональна кубу размерности матриц, а объем обрабатываемых данных – ее квадрату, поэтому выполнение таких операций на устройстве оправдано лишь для достаточно «больших» задач. К примеру, для матриц, имеющих размер порядка нескольких десятков тысяч, выигрыш от решения линейной системы на устройстве, в зависимости от соотношения производительностей конкретного процессора и видеокарты, может достигать нескольких десятков раз.

Исполнение на устройстве операций BLAS уровня 2 едва ли может быть оправданным, исключая случай, когда матрица уже загружена в память устройства.

4.3 Решение задачи типа *N* тел

«Прямой» алгоритм решения задачи *N* тел реализуется исключительно просто (без разницы, для CPU или GPU) и практически идеально масштабируется. Главный его недостаток – квадратичная вычислительная сложность, что при количестве вихревых частиц порядка сотен тысяч или миллионов приводит к неприемлемым затратам времени.

Эффективное решение данной проблемы – в использовании приближенных быстрых алгоритмов, все многообразие которых можно разделить на два класса: методы, в основе которых построение иерархической пространственной структуры дерева (treecodes), и методы, предполагающие выполнение быстрого преобразования Фурье (FFT). К первым относятся метод Барнса – Хата, быстрый метод мультиполей (FMM) и многочисленные их вариации; именно эти методы, судя по публикациям, завоевали наибольшую популярность.

Большой интерес представляет достаточно обстоятельное методическое исследование, проведенное для данной задачи анонимным автором [26], полученные там результаты весьма наглядны и отражают основные тенденции: в частности, главным выводом является то, что наибольшую производительность обеспечивает CUDA-реализация алгоритма метода Барнса – Хата, при этом следует уделять самое пристальное внимание программной реализации с учетом специфики выполнения вычислений и организации памяти на GPU. Там же [26] приведена ссылка на репозиторий https://github.com/drons/nbody (свободное ПО, лицензия не указана), где представлены реализации всех описанных алгоритмов. Представленный код требует библиотек Qt, Boost, OpenCL, OpenGL, CUDA Toolkit.

К сожалению, даже тот алгоритм, что признан автором оптимальным, реализован в указанной библиотеке не самым эффективным образом (по крайней мере, если требуется обеспечивать достаточно высокую точность расчетов); это становится ясно из сопоставления указанной реализации с кодом ECL-BH автора M.Burcher, доступным на его персональной странице http://www.cs.txstate.edu/~burtscher/research/ECL-BH/ (лицензия BSD) и детально описанным в [27]: при идентичных исходных данных и одинаковой точности время вычислений на одной и той же видеокарте отличается примерно на порядок (в 10 раз). Следует отметить, что указанный код ECL-BH не лишен недостатков: на производительных видеокартах типа Nvidia Titan V, Tesla V100, Tesla A100, имеющих порядка сотни мультипроцессоров, наблюдается «гонка данных», которая, впрочем, может быть устранена после внимательного

изучения и анализа структуры кода путем добавления в необходимых местах квалификатора `volatile`. Код ECL-BH включен в проект Lonestar (доступен из репозитория https://github.com/chenxuhao/lonestaromp, лицензия MIT), а также используется в ряде “in-house”-разработок, в которых приходится решать гравитационную задачу *N* тел. При этом тот факт, что основная характеристика каждого тела – его масса – это скалярная положительная величина, существенно используется в данном коде, и для его обобщения даже на задачи взаимодействия точечных зарядов (положительного и отрицательного знаков), и тем более на задачи взаимодействия вихревых частиц, требуется нетривиальная модификация, приводящая к некоторому снижению производительности. Сравнительную простоту модификации кода обеспечивает его весьма ясная, хотя и несколько громоздкая структура, и отсутствие зависимостей от других библиотек, не считая стандартных.

Другим представителем семейства treecodes является семейство быстрых методов мультиполей. По методам мультиполей в различных модификациях имеется огромное количество публикаций, среди которых отметим монографию [28] и серию работ [29, 30] международного коллектива, в состав которого входят L. Barba (США), R. Yokota (Япония), Н. Гумеров (Россия, США), посвященных большей частью именно проблемам использования быстрого метода мультиполей совместно с методом вихревых частиц. Как следует из указанных, а также других их публикаций, данными авторами разработаны соответствующие коды, допускающие в том числе проведение расчетов с использованием графических ускорителей. Именно эти коды в свободном доступе отсутствуют, однако на в репозитории https://github.com/barbagroup/gemsfmm присутствует код gemsfmm тех же авторов, в котором реализован быстрый метод мультиполей для решения плоской и пространственной задачи *N* тел, причем как для CPU, так и для GPU.

В основе метода Барнса – Хата, быстрого метода мультиполей и их многочисленных вариаций – иерархическое разбиение пространства и построение дерева. Для FMM требуется quad/oct-дерево, для алгоритма Барнса – Хата подходит также и *k*-d дерево. Наименее тривиальной является эффективная реализация процедуры построения дерева на GPU. Одна из реализаций имеется в вышеупомянутом коде ECL-BH, однако более эффективным, по крайней мере для реализаций на GPU, оказывается алгоритм, основанный на линейном упорядочивании расположенных на плоскости или в пространстве частиц при помощи фрактальной кривой Мортона [31]. Его реализация в виде кода lbvh, опирающегося на библиотеку thrust, доступна из репозитория https://github.com/ToruNiina/lbvh (лицензия MIT). При этом отметим, что для сортировки частиц по их кодам Мортона можно использовать более эффективную функцию поразрядной сортировки DeviceRadixSort::SortPairs из библиотеки CUB, входящей с определенного момента в состав CUDA Toolkit (https://nvlabs.github.io/cub/structcub_1_1_device_radix_sort.html). Эффективный алгоритм вычисления префиксных сумм описан в [32]; соответствующий код доступен в репозитории https://github.com/TVycas/CUDA-Parallel-Prefix-Sum.

Алгоритмы приближенного решения задачи *N* тел, основанные на быстром преобразовании Фурье, распространены значительно меньше; применительно к вихревым методам наиболее известна работа [16], положенная в основу упоминавшегося выше кода vxflow. Наиболее трудоемкая операция там – это собственно двумерное быстрое дискретное преобразование Фурье, выполняемое для двух перемножаемых матриц, и обратное преобразование, выполняемое для матрицы-результата. Для его оптимальной реализации на CPU представляется целесообразным использование обсуждавшейся выше библиотеки Eigen или широко известной библиотеки FFTW (лицензия GPL-2.0). Для GPU практически безальтернативным вариантом является использование библиотеки cuFFT (составная часть CUDA Toolkit).

5. Формирование вихревых петель

При решении двумерных задач, в которых рассчитывается плоское обтекание профилей или систем профилей, после решения граничного интегрального уравнения определяется величина интенсивности вихревого слоя на обтекаемых контурах, затем завихренность, содержащаяся в этом слое, «стягивается» в точечные вихри, которые «пополняют» вихревой след. Данная процедура с вычислительной точки зрения является крайне простой.

В трехмерных задачах при расчете пространственного обтекания тел реализовать аналогичный алгоритм нельзя, поскольку требуется обеспечить соленоидальность представления завихренности в области течения; для этого достаточно представлять завихренность в виде замкнутых структур (петель). Чтобы определить положения вновь генерируемых на поверхности тела петель следует решить обратную задачу восстановления потенциала двойного слоя на поверхности по известному значению его градиента (интенсивность вихревого слоя есть развернутый на $\pi/2$ вокруг внешней нормали поверхностный градиент потенциала двойного слоя). С учетом того, что задача решается на триангулированной поверхности, число ячеек которой практически точно в два раза больше числа вершин (как следует из формулы Эйлера, $F = 2V - 4$), и можно считать, что в центре каждой ячейки известно значение поверхностного градиента функции, задача определения ее узловых значений сводится к поиску псевдорешения системы с разреженной матрицей, что, в свою очередь, эквивалентно процедуре метода наименьших квадратов, рис. 2.

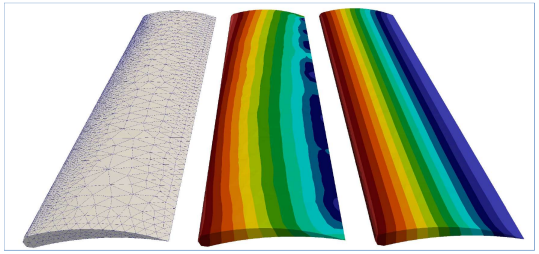

Рис. 2. Поверхностная сетка на модели крыла (слева) и плотность потенциала двойного слоя, восстановленная путем решения гиперсингулярного ГИУ (в центре) и при помощи процедуры метода наименьших квадратов после определения интенсивности вихревого слоя (справа)
Fig. 2. The surface mesh on the wing model (left) and the double layer potential density reconstructed by solving the hypersingular integral equation (center) and using the least squares procedure that follows the reconstruction of the vortex sheet intensity (right)

Для реализации метода наименьших квадратов могут быть использованы уже обсуждавшиеся библиотеки Eigen (для CPU) и cuSPARSE (для GPU).

После восстановления потенциала по его линиям уровня задаются положения вихревых петель. Каждая петля задается узлами, при этом движение петли определяется движением ее узлов. При этом после выполнения шага расчета отдельные узлы петли сближаются, другие удаляются друг от друга. Неравномерность дискретизации петель ведет к их неестественному «выгибанию» и закладывает источник вычислительной неустойчивости. Чтобы этого избежать, на каждом шаге расчета нужно производить редискретизацию вихревых петель, т.е. заново расставлять узлы вдоль петли. При этом до этой процедуры целесообразно исключать из рассмотрения или объединять сблизившиеся узлы (рис. 3).

Рис. 3. Схема исключения и объединения сблизившихся узлов вихревой петли
Fig. 3. Scheme of exclusion and merging of closely placed nodes of the vortex loop

Собственно редискретизацию нужно производить так, чтобы насколько это возможно сохранять длину петли. Это можно обеспечить, если по текущему положению узлов восстановить кривую в виде сплайна, а затем разбить полученный сплайн на дуги равной длины. Хотя построение и работа со сплайнами – довольно простая операция, для этих целей удобно использовать существующие библиотеки, например, библиотеку spline, доступную из репозитория https://github.com/ttk592/spline и опубликованную под лицензией GPL-2.0. Отметим, что в указанной библиотеке нет возможности построения циклического сплайна (с периодическими граничными условиями, рис. 4, что позволяет избежать появления угловой точки), однако такую возможность можно сравнительно легко добавить. Библиотека исключительно легковесна (один заголовочный файл), зависимости отсутствуют.

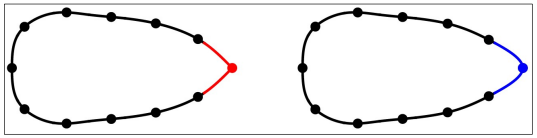

Рис. 4. Форма вихревой петли, восстановленная классическим (слева) и циклическим(справа) сплайном
Fig. 4. Vortex loop smooth shape reconstruction with classical (left) and periodic (right) splines

6. Алгоритмы вычислительной геометрии

В трехмерном случае возникает необходимость решения ряда задач вычислительной геометрии, чаще всего простых и понятных по постановке, но нетривиальных в смысле использования эффективных алгоритмов. Одна из них возникает, когда фрагменты вышеупомянутых вихревых петель располагаются вблизи обтекаемой поверхности и после их перемещения на шаге расчета вследствие неизбежных погрешностей моделирования оказываются внутри тела. В этом случае представляется наиболее естественным отыскать кратчайший путь на поверхности между точками пересечения петли с ней и «проложить» вихревую петлю по этому пути. Здесь же с очевидностью возникает и вторая задача: установить факт пересечения вихревых петель с триангулированной поверхностью. Отметим, что сходная задача актуальна и в двумерном случае: требуется идентифицировать попадания точечных вихревых частиц внутрь контура обтекаемого профиля.

Эти и многие другие задачи могут быть эффективно решены с использованием библиотеки алгоритмов вычислительной геометрии CGAL, написанной на C++ и обеспечивающей удобный доступ к эффективным реализациям множества геометрических и смежных алгоритмов. Библиотека CGAL свободно доступна вместе с исходными кодами по лицензии LGPL-v3; она развивается с 1996 года как крупная совместная разработка Университета г. Утрехт, ETH (г. Цюрих), Свободного университета Берлина, Университета Мартина Лютера (г. Галле), Университета Иоганна Кеплера (г. Линц), Тель-Авивского университета, а также Института Макса Планка и INRIA. Следует отметить, что библиотека весьма громоздка, основана на Boost, Qt и Eigen; требует отдельной компиляции. Фактически она состоит из множества тематически объединенных пакетов, каждый из которых отвечает за структуры данных и базовые алгебраические, геометрические, логические операции или алгоритмы, включающие построение, перестроение и оптимизацию поверхностных и объемных сеток, реконструкцию поверхностей, а также решение отдельных задач.

В контексте обсуждаемых вопросов представляют интерес алгоритмы поиска кратчайшего пути на триангулированной поверхности. Зачастую сложность данной задачи недооценивают, ошибочно полагая, что для ее решения могут быть применены многие «стандартные» хорошо известные алгоритмы на графах: алгоритмы Дейкстры, Флойда – Уоршелла, Беллмана – Форда и др. Это верно лишь отчасти: только в случае, если в качестве решения допустим путь на поверхности, проходящий по ее ребрам. Такой путь обычно оказывается весьма «угловатым», что может стать причиной развития вычислительной

неустойчивости в вихревых методах, поэтому требуется его некоторым образом сгладить. Ясно, что никакой путь, лежащий на триангулированной поверхности (если только она не является плоскостью) не может быть гладким в математическом смысле, поэтому в данном можно предложить следующее нестрогое, но интуитивно понятное определение «достаточно гладкого» пути: если сделать развертку для тех граней, по которым проходит путь, то углы между соответствующей ему ломаной не должны превышать (по крайней мере, по порядку величины) двугранных углы между этими же гранями на исходной поверхности. Наиболее гладким в указанном смысле будет кратчайший путь, поиск которого может быть осуществлен с использованием модулей The Heat Method и Triangulated Surface Mesh Shortest Paths библиотеки CGAL. Удобство и простота использования библиотеки в некоторой мере нивелируются не слишком высокой производительностью: во всех алгоритмах такого типа наиболее эффективно могут быть решены задачи в постановке «найти путь от фиксированной точки на поверхности до множества других точек», причем увеличение, даже значительное, числа «финальных» точек мало сказывается на росте вычислительной сложности. В данном же случае требуется решать задачу для множества различных пар точек «начало – конец маршрута» (рис. 5).

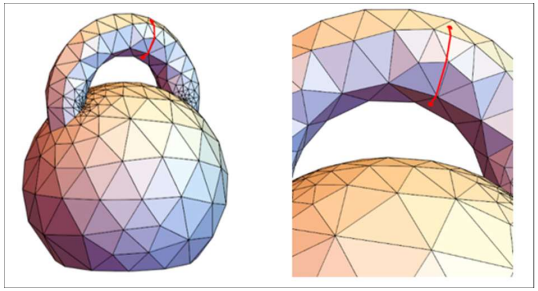

Рис. 5. Пример поиска кратчайшего пути между двумя точками на поверхности
Fig. 5. Example of the shortest path from one fixed point to another on the surface

Необходимо отметить, что на практике можно ограничиться поиском пути, достаточно гладкого в указанном выше смысле, но не обязательно самого короткого из возможных. Алгоритм такого типа [33], особенно если он обеспечивает хорошую экономию времени вычислений, представлялся бы весьма полезным, при этом найденный путь может быть как весьма близким к кратчайшему, так и значительно отличающимся от него (рис. 6).

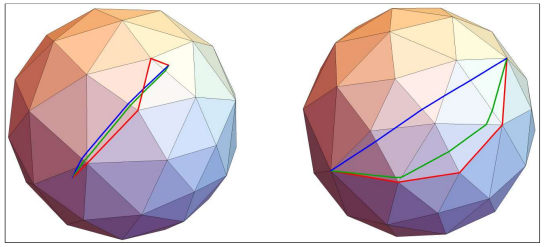

Рис. 6. Кратчайший путь между точками на поверхности (синяя линия), путь найденный по алгоритму Дейкстры (красная линия) и результат «локальной оптимизации» пути [33] (зеленая линия) для двух пар случайным образом выбранных точек на триангулированной сфере
Fig. 6. The shortest path between points on the surface (blue line), the path found by Dijkstra's algorithm (red line) and the result of "local optimization" of the path (green line) for two pairs of randomly selected points on a triangulated sphere

Если говорить о задаче идентификации точек, проникших внутрь триангулированной поверхности или многоугольного контура, которую обычно можно переформулировать как

задачу поиска точек пересечения системы отрезков с множеством треугольников или второй системой отрезков, то она должна быть решена максимально точно, ошибки в ее решении, как показывает опыт, ведут к развитию неустойчивости в алгоритмах вихревых методов. Для решения этой задачи можно использовать такие модули библиотеки CGAL как 2D Intersection of Curves и 3D Fast Intersection and Distance Computation, а также Intersecting Sequences of dD Iso-oriented Boxes. В основе реализованных там алгоритмов – построение иерархического AABB-дерева и его обход. Временные затраты на решение задачи могут быть снижены за счет оптимальной организации процесса, правильного выбора того множества геометрических объектов, по которым это дерево строится: треугольные панели или пересекающие их отрезки, и др. Пример вихревого следа, моделируемого системой вихревых петель за крылом конечного размаха, представленным триангулированной поверхностью, показан на рис. 7. В процессе моделирования на каждом шаге расчета наблюдаются десятки случаев проникновения фрагментов петель внутрь поверхности, которые должны быть идентифицированы, и произведено перестроение петель по кратчайшим путям на поверхности.

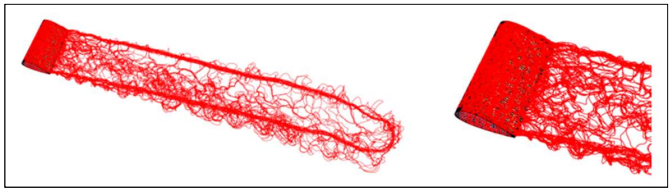

Рис. 7. Формирование вихревого следа за крылом конечного размаха
Fig. 7. Vortex wake formation after the wing of finite span

Отметим, что для плоских задач данный алгоритм поиска пересечений отрезков, вообще говоря, уступает по эффективности т.н. алгоритму Балабана [34], позволяющему отыскать точки пересечения множества отрезков на плоскости, который является неулучшаемым в смысле вычислительной сложности. Реализацию этого алгоритма на C++, выполненную автором, можно найти в репозитории https://github.com/ivvaan/balaban-segments-intersections. Тем не менее, особенности постановки – наличие двух систем отрезков и необходимость искать пересечения отрезков первой группы с отрезками второй группы – делают задачу несколько отличающейся от той, для которой алгоритм Балабана является оптимальным.

Отметим, что имеющиеся в CGAL алгоритмы построения AABB и *k*-d деревьев и их обхода представляет интерес и в контексте выполнения иных процедур вихревых методов: в частности, редискретизации вихревого следа, когда требуется отыскивать близко расположенные вихревые частицы с целью их возможного объединения (в 2D задачах) или выполнения перезамыкания вихревых колец (в 3D задачах). Для этих же целей, а также при вычислении диффузионной скорости вихревых частиц может быть полезен алгоритм поиска ближайших соседей. Перечисленные алгоритмы объединены в CGAL в группу Spatial Searching and Sorting Algorithms.

7. Моделирование динамики обтекаемых тел

При решении сопряженных задач гидроупругости, когда обтекаемые поверхности являются подвижными и/или деформируемыми, вычислительная сложность процедуры расчета движения или деформации тел, как правило, существенно ниже таковой применительно к решению гидродинамической подзадачи. Тем не менее, сама механическая система может быть нетривиальной, и для моделирования ее динамики под действием гидродинамических нагрузок, в зависимости от конкретной постановки задачи, можно с успехом использовать ряд известных библиотек.

К примеру, если обтекаемые тела или профили являются жесткими (недеформируемыми) и лишь перемещаются при наложенных на них связях, может быть эффективна свободная библиотека Project Chrono, разрабатываемая совместными усилиями Университета Висконсина (Мадисон, США) и Пармского университета с 2008 года. Исходный код доступен в репозитории https://github.com/projectchrono/chrono под лицензией BSD. Библиотека «тяжеловесна», требует отдельной компиляции со многими зависимостями (определяются конкретным перечнем компилируемых модулей), поддерживает расчеты на CPU и GPU. Интерфейс библиотеки сравнительно несложный, изучение большого количества прилагающихся примеров позволяет быстро освоить базовые принципы работы с ней.

Если же под действием гидродинамических (или иных) нагрузок обтекаемое тело деформируется, то наиболее перспективным видится использование библиотек конечноэлементного анализа, из числа которых можно выделить Deal.ii – ведущуюся с 2000 года совместную разработку университетов Колорадо, Клемсона, Техаса (все – США), Гейдельбергского университета (ФРГ), а также национальной лаборатории Oak Ridge. Исходный код проекта размещен в репозитории https://github.com/dealii/dealii и доступен под лицензией LGPL-v2. Библиотека также весьма «тяжеловесная», имеет много зависимостей и требует отдельной компиляции. Из ее особенностей – большой набор учебных примеров, снабженных чрезвычайно подробными объяснениями и комментариями. Библиотека поддерживает распараллеливание вычислений посредством MPI и выполнение расчетов на GPU.

8. Заключение

Несмотря на существенные трудности при решении задач моделирования гидро- и аэродинамических процессов, протекающих в нестационарных условиях, использование вихревых методов вычислительной гидродинамики позволяет добиться удовлетворительных результатов, отвечающих требованиям инженерной практики, рассматривая при этом достаточно простые модели среды и затрачивая сравнительно скромные вычислительные ресурсы.

В работе применительно к реализациям алгоритмов вихревых методов дана оценка целесообразности и перспектив использования сторонних библиотек линейной алгебры, вычислительной геометрии, параллельных вычислений и т.п. Их использование помогает упростить и оптимизировать алгоритм, однако в ряде случаев может отрицательно влиять на производительность. Иными словами, к использованию каждой из «стандартных» (библиотечных) функций следует подходить достаточно ответственно, тщательно анализируя те постановки задач, для которых соответствующие алгоритмы и библиотеки оптимизированы, и сопоставляя их с контекстом, возникающим в вихревых методах.

Исходя из вышеизложенного, становится ясным одно из важных направлений развития упомянутых в работе и будущих реализаций вихревых методов. Обоснованное использование возможностей открытых библиотек и их сочетаний может позволить не только оптимизировать имеющиеся реализации, в том числе за счет использования возможностей современных ЭВМ различных архитектур, но и расширить их возможности.

Табл. 1. Расшифровки использованных в тексте аббревиатур
Table 1. Descriptions of abbreviations used in the text

Аббреви-атура	*Расшифровка*	*Перевод на русский язык с пояснением*
ГИУ	Граничное Интегральное Уравнение	–
ПО	Программное Обеспечение	–

ЭВМ	Электронная Вычислительная Машина	–
AABB-дерево	(Axis Aligned Bounding Box)-дерево	Пространственное дерево, ячейки которого – параллелепипеды с гранями, параллельными координатным плоскостям
BiCGStab	BiConjugate Gradient Stabilized method	Метод бисопряженных градиентов со стабилизацией (итерационный метод решения линейных систем)
BLAS	Basic Linear Algebra Subroutines	Базовые подпрограммы линейной алгебры (стандарт де-факто для интерфейса библиотек, выполняющих основные операции линейной алгебры)
BSD	Berkeley Software Distribution	Программная лицензия университета Беркли (для распространения ПО с открыты кодом)
CGAL	Computational Geometry Algorithms Library	Библиотека алгоритмов вычислительной геометрии
CPU	Central Processing Unit	Центральный процессор
CUDA	Compute Unified Device Architecture	Архитектура унифицированного вычислительного устройства (программно-аппаратная архитектура параллельных вычислений с их выполнением на графических процессорах фирмы Nvidia)
Deal.ii	Differential equations analysis library (2-nd generation)	Библиотека конечно-элементного анализа (библиотека для численного решения дифференциальных уравнений в частных производных методом конечных элементов)
FFT	Fast Fourier Transform	Быстрое преобразование Фурье
FFTW	Fastest Fourier Transform in the West	Самое быстрое преобразование Фурье на Западе (одна из наиболее известных и эффективных бесплатных программных реализаций быстрого преобразования Фурье)
FMM	Fast Multipole Method	Быстрый метод мультиполей
GMRES	Generalized Minimal RESidual method	Обобщенный метод минимальных невязок (итерационный метод решения линейных систем)
GPL	(GNU) General Public License	Универсальная общественная лицензия GNU (лицензия на свободное программное обеспечение, созданная в рамках проекта GNU)
GPU	Graphics Processing Unit	Графический процессор (графический ускоритель, видеокарта)
Intel MKL	Intel Math Kernel Library	Библиотека математического ядра Intel (библиотека оптимизированных математических алгоритмов, сопровождающая компиляторы Intel)
Intel TBB	Intel Threading Building Blocks	Кроссплатформенная библиотека шаблонов C++, разработанная Intel для параллельного программирования
LAPACK	Linear Algebra PACKage	Библиотека методов решения основных задач линейной алгебры (с открытым исходным кодом)
LGPL	(GNU) Lesser General Public License	Ослабленная универсальная общественная лицензия GNU (лицензия на свободное программное обеспечение, созданная в рамках проекта GNU)
OpenCL	Open Computing Language	Фреймворк для параллельного программирования с выполнением вычислений на центральных и графических процессорах, а также FPGA

OpenGL	Open Graphics Library	Фреймворк для разработки приложений, использующих двумерную и трёхмерную компьютерную графику
OpenMP	Open Multi-Processing	Стандарт для распараллеливания программ на системах с общей памятью
MIT (license)	Massachusetts Institute of Technology (license)	Лицензия Массачусетского технологического университета (разрешительная лицензия на открытое и свободное программное обеспечение)
MPI	Message Passing Interface	Интерфейс передачи сообщений (стандарт для распараллеливания программ на системах с распределенной памятью)
MPL	Mozilla Public License	Общественная лицензия Mozilla Foundation (лицензия на открытое и свободное программное обеспечение)
VTK	Visualization Toolkit	Инструментарий визуализации (открытая кроссплатформенная библиотека для геометрического моделирования, обработки изображений и визуализации)

Список литературы / References

[1] Leonard A. Vortex methods for flow simulation. Journal of Computational Physics, vol. 37, issue 3, 1980, pp. 289-335.

[2] Sarpkaya T. Computational methods with vortices – The 1988 Freeman Scholar Lecture. Journal of Fluids Engineering, vol. 111, issue 1, 1989, pp. 5–52.

[3] Mimeau C., Mortazavi I. A review of vortex methods and their applications: from creation to recent advances. Fluids, vol. 6, issue 2, 2021, article no. 68, 49 p.

[4] Dynnikova G.Ya. An analog of the Bernoulli and Cauchy – Lagrange integrals for a time-dependent vortex flow of an ideal incompressible fluid. Fluid Dynamics, vol. 35, issue 1, 2000, pp. 24-32.

[5] Андронов П.Р., Гувернюк С.В., Дынникова Г.Я. Вихревые методы расчета нестационарных гидродинамических нагрузок. М., Изд-во МГУ, 2006 г.. 184 стр. / Andronov P.R., Guvernyuk S.V., Dynnikova G.Y. Vortex methods for non-stationary hydrodynamic loads estimation. Moscow, Moscow State University, 2006, 184 p. (in Russian).

[6] Dynnikova G.Ya, Andronov P.R. Expressions of force and moment exerted on a body in a viscous flow via the flux of vorticity generated on its surface. European Journal of Mechanics - B/Fluids, vol. 72, 2018, pp. 293–300.

[7] Dynnikova G.Ya. General expression of aerodynamic force under different boundary conditions (slip, partial slip, no-slip). Physics of Fluids, vol. 33, issue 6, 2021, article no. 063104.

[8] Дынникова Г.Я. Лагранжев подход к решению нестационарных уравнений Навье – Стокса. Доклады Академии наук, том 399, вып. 1, 2004 г., стр. 42–46. / Dynnikova G.Ya. The Lagrangian approach to solving the time-dependent Navier-Stokes equations. Doklady Physics, vol. 49, issue 11, 2004, pp. 648–652.

[9] Гувернюк С.В., Дынникова Г.Я. Моделирование обтекания колеблющегося профиля методом вязких вихревых доменов. Известия РАН. Механика жидкости и газа, вып. 1, 2007, стр. 3–14 / Guvernyuk S.V., Dynnikova G.Ya. Modeling the flow past an oscillating airfoil by the method of viscous vortex domains. Fluid Dynamics, vol. 42, issue 1, 2007, pp. 1-11.

[10] Дынникова Г.Я. Вихревые методы исследования нестационарных течений вязкой несжимаемой жидкости. Диссертация на соискание ученой степени доктора физико-математических наук. М., 2011 г., 269 стр. / Dynnikova G.Ya. Vortex methods for studying unsteady flows of a viscous incompressible fluid. Thesis for the degree of Doctor of Physical and Mathematical Sciences. Moscow, 2011. 269 p. (in Russian).

[11] Stock M.J., Gharakhani A. Open-source accelerated vortex particle methods for unsteady flow simulation. In Proc. of the ASME 2020 Fluids Engineering Division Summer Meeting, 2020, paper no: FEDSM2020-20486, 10 p.

[12] Alvarez E.J., Ning A. High-fidelity modeling of multirotor aerodynamic interactions for aircraft design. AIAA Journal, vol. 58, issue 10, 2020, pp. 4385-4400..

[13] Гирча А.И. Реализация вихревых методов и алгоритмов моделирования процессов нестационарной гидродинамики на основе эффективного комплекса программ. Диссертация на соискание ученой

степени кандидата физико-математических наук. М., 2009 г., 172 с. / Gircha A.I. Implementation of vortex methods and algorithms for simulation the processes of unsteady hydrodynamics based on an effective software package. Thesis for the degree of candidate of physical and mathematical sciences, Moscow, 2009. 172 p. (in Russian).

[14] Григоренко Д.А. Комплекс программ для реализации семейства вихревых методов и его применение. Диссертация на соискание ученой степени кандидата физико-математических наук. М., 2008 г., 149 с. / Grigorenko D.A. A software package for the implementation of a family of vortex methods and its application. Thesis for the degree of candidate of physical and mathematical sciences, Moscow, 2008. 149 p. (in Russian).

[15] Chorin A.J. Numerical study of slightly viscous flow. Journal of Fluid Mechanics, vol. 57, issue 4, 1973, pp. 785-796..

[16] Morgenthal G., Walther J.H. An immersed interface method for the Vortex-In-Cell algorithm. Computers and Structures, vol. 85, issues 11-14, 2007, pp. 712-726.

[17] Morgenthal G. Aerodynamic analysis of structures using high resolution vortex particle methods. Ph.D. thesis. University of Cambridge, 2002. 209 p.

[18] Giannopoulou O., Colagrossi A. et al. Chorin's approaches revisited: Vortex particle method vs finite volume method. Engineering Analysis with Boundary Elements, vol. 106, 2019, pp. 371-388.

[19] Kamemoto K., Tsutahara M., eds. Vortex methods: selected papers of the First International Conference on Vortex Methods. Wspc, 2000. 220 p.

[20] Marchevsky I., Sokol K., Ryatina E., Izmailova Y. The VM2D Open Source Code for Two-Dimensional Incompressible Flow Simulation by Using Fully Lagrangian Vortex Particle Methods. Axioms, 12(3):248, 2023. DOI: 10.3390/axioms12030248.

[21] Kuzmina K., Marchevsky I. et al. On the scope of Lagrangian vortex methods for two-dimensional flow simulations and the POD technique application for data storing and analyzing. Entropy, vol. 23, issue 1, article no. 118, 2021, 38 p.

[22] Марчевский И.К., Сокол К.С., Измайлова Ю.А. Т-схемы для математического моделирования генерации завихренности на гладких профилях в вихревых методах. Вестник МГТУ им. Н.Э. Баумана. Естественные науки, вып. 6, 2022 г., стр. 33–59 / Marchevskii I.K., Sokol K.S., Izmailova Yu.A. T-schemes for mathematical modelling of vorticity generation on smooths airfoils in vortex particle methods. Herald of the Bauman Moscow State Technical University, Series Natural Sciences, issue. 6, 2022, pp. 33–59 (in Russian).

[23] Dergachev S.A., Marchevsky I.K., Shcheglov G.A. Flow simulation around 3D bodies by using Lagrangian vortex loops method with boundary condition satisfaction with respect to tangential velocity components // Aerospace Science and Technology. 2019. Vol. 94. Art. 105374. DOI: 10.1016/j.ast.2019.105374.

[24] Марчевский И.К., Серафимова С.Р. Аналитическое и полуаналитическое вычисление интегралов от логарифмического и ньютоновского потенциала и их градиентов по прямолинейным отрезкам и треугольным панелям. Вычислительные методы и программирование, том. 23, вып. 2. 2022 г., 137-152. / Marchevsky I.K., Serafimova S.R. Analytic and semi-analytic integration of logarithmic and Newtonian potentials and their gradients over line segments and rectilinear panels. Numerical Methods and Programming, vol. 23, issue 2. 2022, pp. 137-152 (in Russian).

[25] Reid M.T.H., White J.K., Johnson S.G. Generalized Taylor–Duffy Method for Efficient Evaluation of Galerkin Integrals in Boundary-Element Method Computations. IEEE Transactions on Antennas and Propagation, vol. 63, issue 1, 2015, pp. 195-209.

[26] Задача N тел или как взорвать галактику не выходя из кухни / N-body problem or how to blow up a galaxy without leaving the kitchen. Available at: https://habr.com/ru/articles/437014/, accessed 9.05.2023 (in Russian).

[27] Burtscher M., Pingali K. Chapter 6 – An efficient CUDA implementation of the tree-based Barnes Hut n-body algorithm. In GPU Computing Gems Emerald Edition; Applications of GPU Computing Series. Morgan Kaufmann Publishers, 2011. pp. 75-92.

[28] Gumerov N.A., Duraiswami R. Fast Multipole Methods for the Helmholtz Equation in Three Dimensions. Elsevier Science, 2005. 426 p.

[29] Hu Q., Gumerov N.A. et al. Scalable fast multipole accelerated vortex methods. In Proc. of the International Parallel and Distributed Processing Symposium, 2014, pp. 966-975.

[30] Yokota R., Barba L.A. Comparing the treecode with FMM on GPUs for vortex particle simulations of a leapfrogging vortex ring. Computers and Fluids, vol. 45, issue 1, 2011, pp. 155-161.

[31] Karras T. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees. In Proc. of the Fourth ACM SIGGRAPH Eurographics conference on High-Performance Graphics, 2012, pp. 33–37.

[32] Harris M., Sengupta S., Owens J.D. Parallel Prefix Sum (Scan) with CUDA. Chapter 39. In Gpu gems 3, ed. by Hubert Nguyen. Addison-Wesley Professional, 2007. Available at: https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda, accessed 9.05.2023.

[33] Gumirova A.I., Marchevsky I.K., Safronov Yu. The algorithm of the path length optimization on the polyhedron surface. In Proc. of the 49th International Summer School-Conference on Advanced Problems in Mechanics, 2021, Springer, 2023. In press.

[34] Балабан И.Ю. Алгоритмы поиска пересечений множества отрезков. Препринты ИПМ им. М.В.Келдыша, 1994 г., no. 45. 25 стр. / Balaban I.Yu. Algorithms for finding intersections of a set of segments. KIAM preprints, 1994, no. 45. 25 p. (in Russian).

Информация об авторах / Information about authors

Илья Константинович МАРЧЕВСКИЙ – доктор физико-математических наук, доцент, профессор кафедры. «Прикладная математика» МГТУ им. Н.Э. Баумана, старший научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов: вычислительная гидродинамика, вихревые методы, теория устойчивости, численные методы, высокопроизводительные вычисления, элементарная математика.

Ilia Konstantinovich MARCHEVSKY – Doctor of Science (Phys.-Math.), Associate professor, Professor of Applied Mathematics department, Bauman Moscow State Technical University; Senior Researcher in Ivannikov Institute for System Programming of the RAS. Research interests: computational fluid dynamics, vortex particle methods, theory of stability, numerical methods, high performance computing, elementary mathematics.

Юлия Андреевна ИЗМАЙЛОВА – студентка магистратуры кафедры «Прикладная математика». Сфера научных интересов: вихревые методы, высокопроизводительные вычисления.

Yulia Andreevna IZMAILOVA – master's student of Applied Mathematics department. Research interests: vortex particle methods, high performance computing.

Мария Александровна ЕРОФЕЕВА – студентка кафедры «Прикладная математика». Сфера научных интересов: вихревые методы, теория устойчивости, интегральные уравнения.

Maria Aleksandrovna EROFEEVA – student of Applied Mathematics department. Research interests: vortex particle methods, theory of stability, and integral equations.

Дарья Юрьевна КОБЗАРЬ – студентка кафедры «Прикладная математика». Сфера научных интересов: вихревые методы, дифференциальная геометрия, теоретическая механика.

Daria Yurievna KOBZAR – student of the Applied Mathematics department. Research interests: vortex particle methods, differential geometry, theoretical mechanics.