Инновационная образовательная программа

ЧЕТВЕРТАЯ СИБИРСКАЯ ШКОЛА-СЕМИНАР

ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 9–11 октября 2007 года

Под редакцией доктора физико-математических наук
профессора А. В. Старченко

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ОСНОВАН В 1878 ГОДУ

Министерство образования и науки Российской Федерации
Томский государственный университет

ЧЕТВЕРТАЯ СИБИРСКАЯ ШКОЛА-СЕМИНАР

ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 9—11 октября 2007 г.

Под редакцией доктора физико-математических наук
профессора А. В. Старченко

Томск
2008

УДК 519.6
ББК 22.19
Ч52

Ч52 **Четвертая Сибирская школа-семинар по параллельным и высокопроизводительным вычислениям** (Томск, 9—11 октября 2007 г.) / Под ред. проф. А. В. Старченко. — Томск : Дельтаплан, 2008. — 244 с.

ISBN 978-5-94154-140-9

В сборнике содержатся материалы, представленные во время работы Четвертой Сибирской школы-семинара по параллельным и высокопроизводительным вычислениям, проходившей 9—11 октября 2007 года в Томском государственном университете при поддержке Министерства образования и науки РФ.

В докладах участников школы-семинара рассмотрены актуальные проблемы организации параллельных вычислений на многопроцессорных системах, современное состояние и перспективы развития методов параллельных вычислений.

Для студентов, аспирантов, преподавателей, научных работников, желающих изучить и практически использовать в научной работе высокопроизводительные вычислительные ресурсы.

УДК 519.6
ББК 22.19

ISBN 978-5-94154-140-9

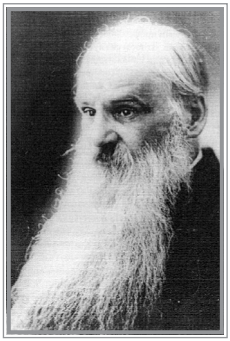
© Томский государственный университет, 2008
© Оформление. ООО «Дельтаплан», 2008

Из истории развития вычислительной математики в Томском университете

В. Н. Берцун
Томский государственный университет, Томск

Математическое образование в Сибири началось с открытия в 1917 г. в Томском университете физико – математического факультета (ФМФ), который был одним из четырех факультетов в структуре университета, перечисленных еще в царском указе от 16 мая 1878 г. [1, 2]. На факультете открыли четыре кафедры (чистой математики, теоретической и практической механики, астрономии и геодезии, технологии и технической химии) и два отделения (физико-математическое и естественнонаучное). На ФМФ было переведено и несколько кафедр с медицинского факультета, а на первый курс было принято 200 студентов. На открывшемся факультете стали преподавать известные ученые: Ф. Э. Молин, Н. Н. Горячев, В. Л. Некрасов, Л. А. Вишневский и др. [3—8].

Федор Эдуардович Молин родился в 1861 г. в г. Риге. После окончания Рижской губернской гимназии, поступил на физико-математический факультет Тартуского (Дерптского) университета и в октябре 1883 г. окончил его со степенью кандидата астрономии. Уже в своей выпускной работе, связанной с определением орбиты кометы 1883 III, Молин продемонстрировал свои незаурядные способности математика-вычислителя. Для продолжения научной деятельности он остался работать в университете, где в 1885 г. успешно защитил магистерскую, а в 1892 г. докторскую диссертации. В должности доцента Ф. Э. Молин проработал в Дерптском (с 1893 г. Юрьевском) университете до 1901 г.



Федор Эдуардович Молин (1861—1941)

В 1901 г. он переехал в Томск и стал заведующим кафедрой математики и ординарным профессором только что открывшегося Томского технологического института (ТТИ). Таким образом, Молин был первым профессором математики в Сибири,

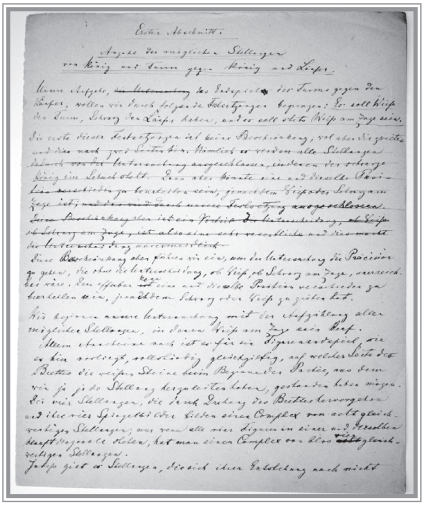


Фото 1. Первая страница рукописи Ф. Э. Молина

принесшим в студенческий Томск красоту и строгость этой науки. Он впервые в Сибири (1912—1913 г.) организовал научный семинар по математике (дифференциальная геометрия, интегральные уравнения, уравнения математической физики и др.) для преподавателей и научных сотрудников Томска, который сыграл важную роль в развитии математического образования в Сибири. С 1918 г. Ф. Э. Молин становится профессором ФМФ ТГУ. Он является одним из основоположников современной алгебры, а часть его работ посвящена решению прикладных задач по теории вероятностей. Следует отметить, что Федор Эдуардович был известным шахматистом, интересовался математическим моделированием шахматной игры на основе комбинаторики и теории графов, переписывался с М. И. Чигориным, избирался председателем Тартуского шахматного общества. В архиве Молина имеется написанная в 1897 г., но не опубликованная до сих пор рукопись по теории эндшпиля — ладьи против слона, одна из страниц которой представлена на фото 1. Рукопись содержит 124 страницы на немецком языке, в которых имеются рисунки 124 позиций.



Николай Никанорович Горячев (1883—1940)

Важную роль в развитии исследований по вычислительной математике в ТГУ сыграл выпускник ФМФ МГУ (1905 г.) Николай Никанорович Горячев. Работая астрономом-вычислителем Пермского университета, летом 1919 г. (во время гражданской войны)

он был эвакуирован в Томск и стал работать на ФМФ ТГУ преподавателем-вычислителем при астрономическом кабинете. С 1920 г. впервые в Томске начал читать на ФМФ курс «Методы приближенных вычислений». Горячев в своих работах по астрономии постоянно использовал вычислительную математику и математическую статистику, усовершенствовал вычислительную схему эволюций орбит тел Солнечной системы. С 1923 г. он стал профессором, заведующим кафедрой астрономии и геодезии.

В 1931 г. вместо факультетов в ТГУ создаются отделения, а на механико-математическом отделении физико-механического факультета (так тогда назывался ФМФ) появилась специальность по прикладной математике. В плане этой специальности содержались такие теоретические курсы: уравнения математической физики, теория конечных разностей, приближенное интегрирование дифференциальных уравнений, прикладная механика, сопротивление материалов, термодинамика, неорганическая химия, вычислительная техника и др. В 1931 г. по такой специальности, например, получили свидетельство об окончании университета [5]: П. П. Куфарев, М. С. Горохов и др.

Следует отметить, что в начале 30-х гг. ТГУ был единственным вузом в стране, где на механико-математическом отделении готовили специалистов по прикладной (вычислительной) математике [4].

Таким образом, в Томске интерес к вычислительной математике и практическому ее использованию появился уже в начале прошлого века.

Новый этап в развитии и применении вычислительных методов в университете связан с именем профессора Льва Александровича Вишневского.

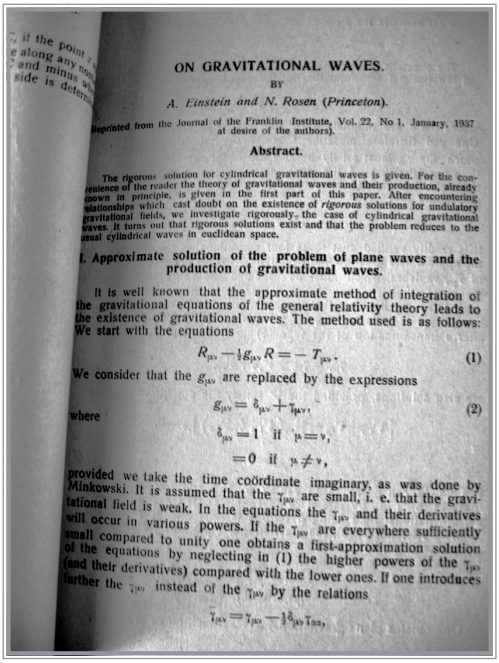
Выпускник МГУ Л. А. Вишневский после закрытия Крымского университета в 1925 г. переехал в Томск. Работая в Томском университете, он значительное внимание стал уделять подготов-



Лев Александрович Вишневский
(1887—1938)

ке специалистов по прикладной (вычислительной) математике. По его инициативе в 1930 г. при университете создается вычислительное бюро, в котором были начаты работы по расчету траекторий полета снарядов и составлению баллистических таблиц. При этом использовались вычислительные схемы метода Адамса, прямое и обратное интерполирование. С 1932 г. по 1937 г. Л. А. Вишневский был первым директором Научно-исследовательского института математики и механики (НИИММ) при Томском университете и руководил отделом прикладной математики в этом институте.

Заметим, что работы математиков и механиков тех лет публиковались в «Известиях НИИММ», главным редактором которых был Ф. Э. Молин. Одна из форм общения ученых того времени выражалась во взаимной пересылке статей для публикации. Так появились изданные в ТГУ статьи А. Эйнштейна и Дж. Фон Неймана. На фото 2 представлена первая страница статьи А. Эйнштейна в Известиях НИИММ.



ON GRAVITATIONAL WAVES.

BY

A. Einstein and N. Rosen (*Princeton*).

Reprinted from the Journal of the Franklin Institute, Vol. 22, No 1, January, 1937 at desire of the authors).

Abstract.

The rigorous solution for cylindrical gravitational waves is given. For the convenience of the reader the theory of gravitational waves and their production, already known in principle, is given in the first part of this paper. After encountering relationships which cast doubt on the existence of *rigorous* solutions for undulatory gravitational fields, we investigate rigorously the case of cylindrical gravitational waves. It turns out that rigorous solutions exist and that the problem reduces to the usual cylindrical waves in euclidean space.

I. Approximate solution of the problem of plane waves and the production of gravitational waves.

It is well known that the approximate method of integration of the gravitational equations of the general relativity theory leads to the existence of gravitational waves. The method used is as follows: We start with the equations

$$R_{\mu\nu} - \tfrac{1}{2} g_{\mu\nu} R = - T_{\mu\nu}. \qquad (1)$$

We consider that the $g_{\mu\nu}$ are replaced by the expressions

$$g_{\mu\nu} = \delta_{\mu\nu} + \gamma_{\mu\nu}, \qquad (2)$$

where

$$\delta_{\mu\nu} = 1 \quad \text{if} \quad \mu = \nu,$$
$$= 0 \quad \text{if} \quad \mu \neq \nu,$$

provided we take the time coördinate imaginary, as was done by Minkowski. It is assumed that the $\gamma_{\mu\nu}$ are small, i. e. that the gravitational field is weak. In the equations the $\gamma_{\mu\nu}$ and their derivatives will occur in various powers. If the $\gamma_{\mu\nu}$ are everywhere sufficiently small compared to unity one obtains a first-approximation solution of the equations by neglecting in (1) the higher powers of the $\gamma_{\mu\nu}$ (and their derivatives) compared with the lower ones. If one introduces further the $\bar{\gamma}_{\mu\nu}$ instead of the $\gamma_{\mu\nu}$ by the relations

$$\bar{\gamma}_{\mu\nu} = \gamma_{\mu\nu} - \tfrac{1}{2} \delta_{\mu\nu} \gamma_{\alpha\alpha},$$

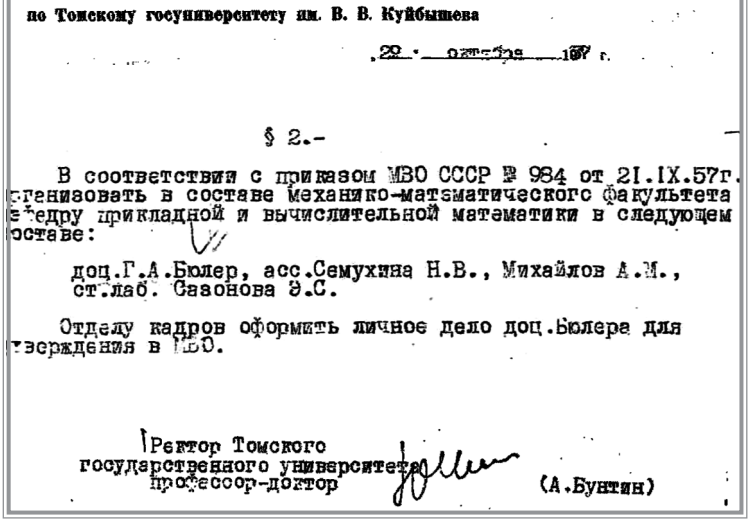
Фото 2. Первая страница статьи А. Эйнштейна в Известиях НИИММа

К числу первых работ, посвященных разностным уравнениям, можно отнести статью А. К. Минятова «Сравнительное исследование дифференциальных и разностных уравнений» и работу М. С. Горохова «Связь графического метода интегрирования с численным», опубликованные в «Известиях НИИММ» [3].

В 1948 г. ФМФ разделился на два факультета: физический (ФФ) и механико-математический (ММФ). Появление на ММФ подготовки математиков, астрономов и механиков было связано с необходимостью обновления промышленного производства, активизацией в СССР научных исследований в области ядерной энергетики, ракетостроения и космонавтики.

Первая группа математиков-вычислителей была сформирована весной 1957 г., когда математикам-третьекурсникам ММФ предложили перейти на новую специальность «Вычислительная математика».

Следующим важным событием в развитии вычислительной математики в Томском университете стало открытие в сентябре 1957 г. по приказу МВО СССР (см. фото 3) первой за Уралом



ПРИКАЗ №209.

по Томскому госуниверситету им. В. В. Куйбышева

29 октября 1957 г.

§ 2.-

В соответствии с приказом МВО СССР № 984 от 21.IX.57г. организовать в составе механико-математического факультета кафедру прикладной и вычислительной математики в следующем составе:

доц.Г.А.Бюлер, асс.Семухина Н.В., Михайлов А.М., ст.лаб. Сазонова Э.С.

Отделу кадров оформить личное дело доц.Бюлера для утверждения в МВО.

Ректор Томского государственного университета профессор-доктор (А.Бунтин)

Фото 3

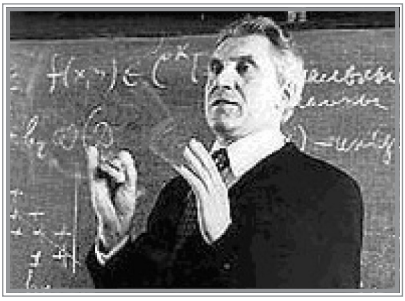
кафедры прикладной и вычислительной математики (КПиВМ) [8, 9].С этого момента на факультете стали готовить специалистов, использующих компьютер как инструмент познания.

Заведующим кафедрой с октября 1957 г. стал доцент Георгий Александрович Бюлер, который был инициатором ее создания. С отличием закончив физико-математический факультет ТГУ в 1936 г., он через пять лет защитил кандидатскую диссертацию. Бюлер был известным математиком благодаря научным публикациям по численному решению задач математической физики (теплопроводности, упругости, магнитодинамики и др.).

С сентября 1957 г. до февраля 1958 г. Георгий Александрович Бюлер, Нинель Викторовна Семухина (Генина), Арнольд Петрович Михайлов, Эльвира Сергеевна Сазонова, были направлены в МГУ на курсы переподготовки по вычислительной математике. В начале1958 г. при КПиВМ в главном корпусе ТГУ был создан кабинет вычислительных машин, где в конце 50-х — начале 60-х гг. студенты ММФ выполняли задания по «Методам вычислений» на арифмометрах «Феликс» и «Рейнметалл», составлявших парк кабинета.

Поскольку на просторах Сибири еще не было электронных вычислительных машин, то в декабре 1958 г. студенты-вычислители ММФ были направлены на практику в Вычислительный центр МГУ, где была установлена машина «Стрела». Руководителем практики был Юрий Яковлевич Попов. Летом 1959 г. состоялся первый в Сибири выпуск 24 математиков — вычислителей 443-й группы. Среди них был и Моисей Аронович Тынкевич. Это имя широко известно среди выпускников томских вузов 60-х гг. Работая на кафедре, он научил не одно поколение вычислителей любить выбранную специальность. Попытка университета купить ЭВМ в 1960 г. успешно завершилась: в университете заработала ЭВМ «Урал-1». Поместили ее в правом крыле коридора второго этажа главного корпуса рядом с гербарием. В те времена кафедра, так же как и факультет, находилась на третьем этаже южного крыла главного корпуса ТГУ.

Новым руководителем КПиВМ с сентября 1961 г. стал доцент Юрий Семенович Завьялов [3], который окончил ММФ в 1953 г., получив диплом с отличием по специальности «Газовая динамика».



Юрий Семенович Завьялов (1931—1998)

Еще студентом он активно занимался научной работой, поэтому после окончания университета поступил в аспирантуру к Е. Д. Томилову и в 1956 г. защитил кандидатскую диссертацию «Об интегрировании некоторых уравнений неизэнтропического движения газа». После защиты диссертации молодой ученый занимался преподавательской деятельностью. Его научные интересы постепенно стали смещаться в область вычислительной математики. Важную роль в развитии вычислительной математики и механики в Томске сыграло создание в 1957 г. Сибирского отделения Академии Наук СССР с Институтом математики и Вычислительным центром в г. Новосибирске. Сотрудниками различных институтов СО АН стали и преподаватели ММФ: Б. Г. Кузнецов, В. Н. Шепеленко, В. А. Топоногов и др. В 1963 г. Ю. С. Завьялов тоже переехал в Новосибирск, где стал работать в Институте Математики СОАН СССР. Именно здесь наиболее полно раскрылись способности Ю. С. Завьялова как ученого и организатора научной деятельности. Под его руководством с середины 60-х гг. были начаты работы по теории и приложениям сплайнов. Ю. С. Завьялов был в числе пионеров в развитии этого направления вычислительной математики в СССР. Он автор более 70 научных публикаций, в том числе четырех монографий [10, 11]. В 1972 г. Ю. С. Завьялов защитил одну из первых в стране докторских диссертаций по теории сплайнов. Среди его учеников 16 кандидатов и 2 доктора наук. Большое внимание Ю. С. Завьялов уделял пропаганде сплайновых методов. В 70—80-е гг. в Новосибирском Академгородке под его председательством регулярно проводились всесоюзные школы-семинары по теории и приложениям сплайнов с участием ведущих специалистов страны, работающих в этой области. Создание научной школы по теории и приложениям сплайнов стало, по существу, главным делом его жизни. Как всякий по-настоящему крупный ученый Ю. С. Завьялов интересовался не только вычислительной ма-

тематикой и механикой, но и глубоко изучал вопросы истории, философии, политики, экономики. Он был избран членом-корреспондентом Петровской Академии наук и искусств.

Работая в Новосибирске, Юрий Семенович не порывал связей с Томском, приезжал на конференции и читал лекции для студентов и преподавателей ММФ, поддерживал исследования по созданию сплайновых методов решения краевых задач. Помогал организовать прием на практику студентов ММФ в институты СОАН.

Его научная и производственная деятельность отмечена орденом «Знак почета». За крупный вклад в создание новых технологий в авиационном производстве Ю. С. Завьялов в составе группы сотрудников Института математики СО АН СССР и работников Новосибирского авиационного завода имени В. П. Чкалова в 1981 г. был удостоен премии Совета Министров СССР.

С декабря 1962 г. заведующей КПиВМ была избрана выпускница ММФ 1952 г. доцент Роза Михайловна Малаховская, которая проработала на кафедре 41 год.

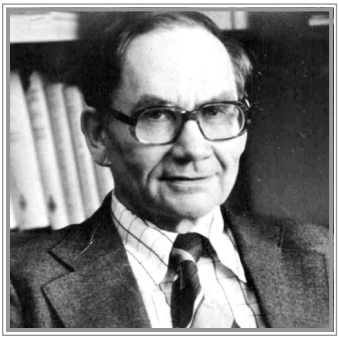


Роза Михайловна Малаховская
(1926—1998)

Читала курсы лекций по методам вычислений, теории конечно-разностных схем, теории обобщенных функций, прямым методам и др. Около 50 работ опубликовала Р. М. Малаховская в математических журналах, трудах конференций различного ранга. Весом ее вклад в совершенствование подготовки по вычислительной математике на ММФ и установление научных контактов между кафедрой и СО АН СССР. Здесь ей существенную помощь оказывали выпускники ММФ ТГУ: академик Николай Николаевич Яненко и профессор Юрий Семенович Завьялов.

Благодаря этому сотрудничеству, студенты-вычислители ММФ проходили преддипломную практику и выполняли дип-

ломные работы у ведущих ученых Академгородка и имели доступ к ЭВМ «БЭСМ-6». Так, например, в 1966/67 уч. г. в Академгородке проходили преддипломную практику и защищали дипломные работы студенты 424 гр.: В. Н. Берцун, В. И. Боброва, Ю. П. Зуйков — ВЦ (отдел Н. Н. Яненко); Л. В. Артюхова — НИИЯФ; Л. А. Новоселова — ИМ СО АН СССР. С 1960 г. ежегодный набор по специальности «Вычислительная математика» составлял 50 человек, студенты этой специальности получали повышенную стипендию, а за первые 10 лет существования кафедра выпустила около 300 математиков-вычислителей.



Николай Николаевич Яненко

Николай Николаевич Яненко родился 22 мая 1921 г. в г. Куйбышеве Новосибирской области [12—16]. В 1939 г. поступил на ФМФ ТГУ. Вот что говорил о нем профессор ФМФ Захар Иванович Клементьев, преподававший в то время математический анализ: «Николай Яненко был идеальный студент. Со своими прекрасными способностями, богатыми знаниями, часто превышающими учебный курс, он никогда не выделялся среди ребят поведением, манерами — совершенно не было в нем шика отличника. Рабочий день студента Коли Яненко начинался в семь часов утра и заканчивался в час ночи».

Во время учебы он познакомился с эвакуированным в Томск преподавателем МГУ профессором-геометром П. К. Рашевским, сыгравшим большую роль в научной судьбе Яненко. В 1942 г. досрочно, за три года, он с отличием закончил университет и ушел на фронт. Был на передовой рядовым, потом переводчиком штаба полка и дивизии. Награжден орденом Красной Звезды, медалью «За отвагу» и др. В 1946 г. поступил в аспирантуру математического факультета МГУ. Под руководством П. К. Рашевского защитил в 1949 г. кандидатскую диссертацию «О некоторых необходимых признаках изгибаемых поверхностей в n-мерном евклидовом пространстве». С августа 1948 г. по июль 1953 г. Н. Н. Яненко — сотрудник географической комплексной экспедиции Геофизического института АН СССР. Затем работает

в Отделе прикладной математики Математического института АН СССР, возглавляемого академиком А. Н. Тихоновым (1949—1955 гг.). После защиты докторской диссертации по многомерной дифференциальной геометрии (1954 г.), работает в Федеральном ядерном Центре (г. Снежинск, Челябинской области, 1955—1963 г.), а затем в Новосибирске: с 1963 г. он — профессор Новосибирского государственного университета и заведующий лабораторией, а затем по 1976 г. — заведующий отделом Вычислительного центра Сибирского отделения АН СССР. В 1966 г. Николай Николаевич избран членом-корреспондентом, а в 1970 г. — действительным членом АН СССР по отделению механики и процессов управления. Он создал новое направление в науке, названное им математической технологией: основные принципы модульного анализа задач, первые определения модуля и пакета прикладных программ, их классификация.

Разностными методами решения краевых задач Н. Н. Яненко начал заниматься с 1949 г., а первую работу на эту тему опубликовал в 1951 г. Одним из первых он понял огромную роль распараллеливания вычислений в разработке путей резкого увеличения производительности ЭВМ, создания ЭВМ со сверхвысокой производительностью. Он показал наличие тесной взаимосвязи между структурой алгоритмов решения задач механики сплошной среды и собственной структурой вычислительной машины. Областью его творчества становятся аналитические исследования систем нелинейных уравнений с частными производными, широкий круг проблем вычислительной математики. С 1976 г. Н. Н. Яненко становится директором Института теоретической и прикладной механики СО АН СССР.

В московский и уральский периоды жизни Н. Н. Яненко связи с Томским университетом были ограничены отсутствием времени и режимом работы. Но он был в Томске в январе 1956 г., когда участвовал в работе комиссии по распределению на работу в свою организацию выпускников ТГУ. После его переезда в г. Новосибирск, уже в 1964 г. Николай Николаевич участвовал в работе III Сибирской конференции по математике и механике с докладом «О слабой аппроксимации систем дифференциальных уравнений». Выступая перед преподавателями ММФ ТГУ, он призывал писать новые учебники по численным методам, представляющие собой органическую связь между технологией

вывода законов сохранения и методами конструирования консервативных разностных схем для решения задач механики сплошной среды.

Благодаря его поддержке сотрудниками ТГУ публиковались монографии, статьи в ДАН СССР и сборнике «Численные методы механики сплошной среды» Н. Н. Яненко способствовал созданию на ММФ кафедры физической механики (1977 г.) и отдела механики реагирующих сред в НИИ ПММ. Неоценимая помощь Н. Н. Яненко в организации длительных практик студентов 4—5 курсов ММФ на ВЦ и в ИТПМ СО АН СССР в 1964—1984 гг. способствовала повышению качества подготовки студентов, часть из которых распределялась на работу и в институты СО АН СССР. Например, в отдел Н. Н. Яненко был распределен на работу выпускник ММФ 1967 г. Ю. П. Зуйков. После защиты диссертации он стал ученым секретарем Президиума СО АН СССР, а позднее начальником управления организации научных исследований.

Научное творчество Н. Н. Яненко многогранно. Более 200 научных работ и 7 монографий. Большей частью они посвящены численным методам решения задач математической физики, а также нелинейным уравнениям в частных производных.

Одним из главных достижений Н. Н. Яненко в вычислительной математике является создание (1957—1963 гг.) «метода дробных шагов», позволяющего свести решение многомерной задачи к последовательному решению соответствующей цепочки одномерных задач. В 1966 г. вышли из печати его лекции, написанные для студентов НГУ: «Метод дробных шагов решения многомерных задач математической физики». Эти лекции были привезены в Томск выпускником кафедры прикладной и вычислительной математики ММФ В. А. Сапожниковым (аспирантом Н. Н. Яненко) и изучались на семинаре кафедры под руководством доцента Р. М. Малаховской, что сыграло важную роль в развитии нового научного направления по численным методам решения многомерных краевых задач в Томске.

Мировую известность получила его монография «Метод дробных шагов решения многомерных задач математической физики» (изд-во Наука, 1967 г.), которая позднее была переведена и издана в Германии, Франции и США.

Особого внимания заслуживает педагогическая работа Н. Н. Яненко. Она также была весьма многогранной. Он не толь-

ко преподавал в Московском, Уральском, Новосибирском, Томском государственных университетах (среди его учеников около 20 докторов и более 50 кандидатов наук), но и блестяще читал лекции в Сорбонне и Кембридже, знал пять иностранных языков. Очень важным в его деятельности было создание и бессменное руководство шестью всесоюзными семинарами. Без всякого преувеличения можно сказать, что благодаря неустанному труду, доброжелательному отношению к научной молодежи и таланту педагога, академиком Н. Н. Яненко в Сибири была создана мощная научная школа по математическому моделированию сложных задач механики сплошных сред.

Николай Николаевич Яненко — Герой Социалистического Труда, трижды лауреат Государственных премий СССР. Он награжден орденом Октябрьской Революции, тремя орденами Трудового Красного Знамени. Член оргкомитета Международной конференции по численным методам в гидродинамике, редколлегий четырех иностранных научных журналов, Рабочей группы Международной федерации по обработке информации (IFIP). С 1977 г. иностранный член Американского института по аэронавтике и астронавтике. В 1975 г. ему вручена именная медаль Коллеж де Франс.

В 60-х гг. на ММФ был выполнен цикл работ по статистической обработке результатов наблюдений с помощью ЭВМ в связи с исследованиями по Тунгусскому метеориту (В. Г. Фаст, Э. Н. Кривякова, О. П. Федорова, А. П. Бояркина, С. А. Разин, Ю. К. Устинов, В. Д. Гольдин и др.) [17; 18].

С 1965 г. в ТГУ появился свой Вычислительный Центр, где была установлена трехадресная машина М-20. Появилось первое за Уралом учебное пособие сотрудника КПиВМ ММФ М. А. Тынкевича «Программирование на М-20», по которому учились программированию в кодах студенты всех вузов города Томска.

Первоначально ВЦ располагался на заводе математических машин, а позднее был переведен во второй учебный корпус ТГУ, где работали отечественные машины БЭСМ-4, М-220 и машины серии ЕС-ЭВМ. В этот период в приложении к диплому выпускника ММФ специальности «Вычислительная математика» содержались следующие специальные дисциплины: конечно — разностные методы, теория алгоритмов, теория приближений, теория

графов, интегральные уравнения, основы вычислительной техники, операционное исчисление, методы Монте-Карло, математическая логика, методы линейной алгебры, Алгол-60, математическое программирование, автоматизация программирования.

В конце 60-х гг. в университете в основном завершился переход от преподавания программирования в кодах к алгоритмическим языкам и операционным системам, что способствовало повышению качества программных продуктов, созданию и освоению новых вычислительных алгоритмов в сравнительно короткие сроки, привлечению талантливой молодежи к исследованиям в области вычислительной математики.

В 1968 г. в ТГУ был создан НИИПММ [9], что сыграло важную роль в развитии математики и механики в университете. Возрождение НИИММа способствовало появлению новой научной тематики и укреплению связей с промышленностью, новых математических моделей и новых оригинальных вычислительных алгоритмов, освоению неявных разностных схем и технологии метода дробных шагов Н. Н. Яненко на ЭВМ М-222 (1973 г.) и БЭСМ-6 (1976 г.). В эти годы в ТГУ изучались языки программирования: Алгол, PL-1, Алгол-ГДР, Ляпас, Фортран и Альфа.

Начиная с работ С. А. Бетехтина (1948—1953), который применил метод характеристик к расчету одномерных газовых потоков в цилиндрических каналах, в НИИПММ под руководством В. Н. Вилюнова появились новые постановки задач газовой динамики, горения и теплообмена, были получены разностные схемы повышенного порядка точности (И. М. Васенин, А. Д Рычков, Е. А. Козлов, Э. Р. Шрагер, В. М. Ушаков, А. А. Глазунов, В. Г. Бутов и др.).

Значительный вклад в развитие, созданного профессором А. М. Гришиным в конце прошлого века научного направления по механике реагирующих сред и экологии [19], внесли работы В. И. Зинченко, А. Н. Субботина, А. С. Якимова, О. П. Федоровой, В. Д. Гольдина, В. Г. Зверева, С. И. Пырха, А. Д. Грузина, Ф. М. Пахомова и др. На основе предложенного А. М. Гришиным в 1970 г. итерационно-интерполяционного метода (ИИМ), являющегося видоизменением метода М. Е. Швеца, В. Н. Берцун получил новые экономичные разностные схемы сплайновой интерполяции для решения двумерных сопряженных нестационарных задач горения неоднородных безгазовых составов, которые

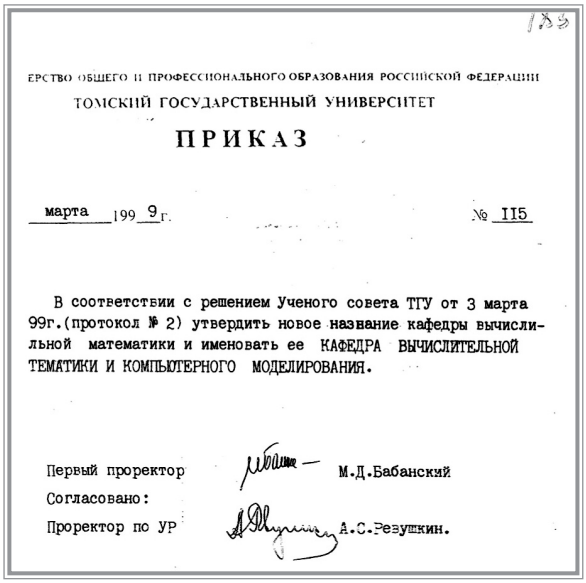
были основаны на методе суммарной аппроксимации А. А. Самарского. В этот же период В. Н. Трушниковым и А. Я. Кузиным разрабатывались новые регуляризирующие численные методы решения обратных задач.

Под руководством профессора Т. В. Бордовицыной, выпускницы ММФ 1963 г., продолжая традиции астрономов ФМФ, в 70—80-е гг. в НИИПММ была начаты активные научные исследования в области построения высокоточных вычислительных алгоритмов для моделирования движения малых тел Солнечной системы [6]. В этом новом научном направлении активно работали: А. М. Черницов, Г. О. Рябова, Л. Е. Быкова и др.

Под руководством профессора В. А. Гридневой были созданы новые постановки задач и разностные схемы для расчета высокоскоростного соударения металлических тел [6]. Активное участие в разработке этой тематики принимали: А. И. Корнеев, Л. М. Артищева, М. Д. Михайлов, Н. Н. Меркулова, Н. Н. Белов, Л. И. Шахтмейстер и др.

В начале 70-х гг. доцентом В. Е. Томиловым на ММФ были начаты исследования по нестационарному сопряженному теплообмену при ламинарном и турбулентном течении в каналах. В разработке численных методов решения таких задач принимали участие: В. И. Кондрашов, С. Г. Иванушкин, Л. В. Ким, А. М. Бубенчиков, А. В. Старченко и др.

Важным событием для развития прикладной (дискретной) математики в ТГУ явилось открытие в 1970 г. факультета прикладной математики (ФПМ), куда были переведены с ММФ две группы студентов и некоторые преподаватели ММФ, а также сокращен набор на ММФ. Следует отметить, что декан ММФ доцент Р. М. Малаховская и ученый совет ММФ поддерживали идею организации на ММФ соответствующего отделения. Заметим, что в начале 60-х гг. слово «прикладной» исчезло из названия КПиВМ. Происходило это постепенно. Сначала для сокращения записи перестали писать в приказах, а затем и вовсе забыли про него. Не будь этой ошибки, изменения в структуре ММФ в 1970 г. могли пойти по другому варианту. Ошибку обнаружил заведующий кафедрой В. Н. Берцун лишь в 1999 г. В этом же году кафедра была официально переименована (см. фото 4) в кафедру вычислительной математики и компьютерного моделирования (КВМиКМ).



ЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПРИКАЗ

марта 199 9 г. № 115

В соответствии с решением Ученого совета ТГУ от 3 марта 99г.(протокол № 2) утвердить новое название кафедры вычислительной математики и именовать ее КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕМАТИКИ И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ.

Первый проректор М.Д.Бабанский

Согласовано:

Проректор по УР А.С.Резушкин.

Фото 4

Появление ПЭВМ на ММФ способствовало изменению планов преподавания базовых курсов «Информатика» и «Методы вычислений», проведение практических занятий по этим курсам было переориентировано на персональную вычислительную технику. Появились новые спецкурсы, такие как «Теория сплайнов», «Вычислительные методы в задачах экологии», «Базы данных».

Обучение параллельным вычислительным технологиям в Томском государственном университете впервые появилось на ММФ в 1987 г., когда доцент В. Н. Берцун начал читать специализированный курс лекций по данной тематике для студентов пятого курса специализации «вычислительная математика». В 1998 г. на КПиВМ стал работать профессор А. В. Старченко, что привело к активизации исследований по численным методам в задачах экологии и разработке экономичных высокопроизводительных вычислительных алгоритмов на ММФ.

Таким образом, в конце XX века были сделаны необходимые научные и технические предпосылки для развития в ТГУ перспективных направлений вычислительной математики и компьютерного моделирования в начале XXI века.

Литература

1. Краткий исторический очерк Томскаго Университета. Томск, 1917. — 544 с.

2. Томский университет. 1880—1980. — Томск : Изд-во Том. гос. ун-та. — 430 с.

3. **Круликовский Н. Н.** Из истории развития математики в Томске. — Томск : Изд-во Том. гос. ун-та, 2006. — 174 с.

4. **Фоминых С. Ф., Некрылов С. А., Берцун Л. Л., Литвинов А. В.** Профессора Томского университета : библиографический словарь. — Томск : Изд-во Том. ун-та, 1998. — Т. 2. — 544с.

5. **Фоминых С. Ф., Некрылов С. А., Берцун Л. Л., Литвинов А. В., Петров К. В., Зленко К. В.** Профессора Томского университета : Библиографический словарь. — Томск: Изд-во Том. гос. ун-та, 2001. — Т. 3. — 532 с.

6. **Фоминых С. Ф., Некрылов С. А., Берцун Л. Л., Литвинов А. В., Петров К. В., Зленко К. В.** Профессора Томского университета : библиографический словарь. — Томск: Изд-во Том. гос. ун-та, 2003. — Т. 4. — Ч. 1. — 534 с.

7. **Боголюбов А. Н.** Математики и механики. — Киев, 1983. — 638 с.

8. **Берцун В. Н., Михайлов М. Д., Тынкевич М. А.** Кафедре вычислительной математики и компьютерного моделирования 45 лет // Вестник ТГУ : бюл. оперативной научной информации. — 2003, апрель. — № 10. — С. 7—16.

9. Развитие математики, механики и кибернетики в Томском университете : сб. ст. — Томск: Изд-во Том. гос. ун-та, 1981. — 124 с.

10. **Завьялов Ю. С., Леус В. А., Скоропоспелов В. А.** Сплайны в инженерной геометрии. — М. : Машиностроение, 1985. — 224 с.

11. **Завьялов Ю. С., Квасов Б. И., Мирошниченко В. Л.** Методы сплайн-функций. — М. : Наука, 1980. — 352 с.

12. **Гришин А. М.** Академик Николай Николаевич Яненко — ученый, педагог, человек. — Томск : Аудит-Информ, 1997. — 61 с.

13. **Яненко Н. Н.** Метод дробных шагов решения многомерных задач математической физики. — Новосибирск : Наука, 1967. — 197 с.

14. **Дмитриев Н. А.** Теоретическая физика. Математика : Избр. тр. — Саров : ФУГУП «РФЯЦ-ВНИИЭФ», 2004. — 511 с.

15. **Шпак Г.** Мир спасет математика! (Штрихи к портрету Н. Н. Яненко (1921—1984 гг.)) // Наука в Сибири. — 2001, июнь. — № 24. — С. 3—4.

16. **Шокин Ю. И.** О системе академика Н. Н. Яненко по подготовке специалистов в области математического моделирования // Вычисл. технологии. — 2001. — Т. 6, спец. вып. — С. 29—36.

17. Проблема Тунгусского метеорита. — Томск : Изд-во Том. гос. ун-та, 1967. — Вып. 2. — 238 с.

18. Метеоритные исследования в Сибири (75 лет Тунгусскому феномену). — Новосибирск : Наука, 1984. — 217 с.

19. **Гришин А. М.** Моделирование и прогноз катастроф. — Томск : Изд-во Том. гос. ун-та, 2003. — 524 с.

Стратегии распараллеливания в задаче о динамике протопланетного диска[1]

В. А. Вшивков, С. Е. Киреев, Е. А. Кукшева,*
А. В. Снытников, Н. В. Снытников
Институт вычислительной математики и математической геофизики СО РАН
*Институт катализа им. Г. К. Борескова СО РАН, Новосибирск

В работе представлены некоторые стратегии распараллеливания, использованные при решении конкретной задачи, а именно, моделировании динамики протопланетного диска. Среди них: упрощение трехмерной модели протопланетного диска до квазитрехмерной, использование фундаментального решения уравнения Пуассона, моделирование в естественной (цилиндрической) системе координат и декомпозиция области моделирования. Показано, что стратегия на основе декомпозиции области позволяет достигнуть наибольшего ускорения.

1. Введение

Проблема происхождения органической материи в Солнечной системе всегда представляла большой интерес. Одна из гипотез происхождения органической материи рассмотрена в [1], где протопланетный диск рассматривается как каталитический химический реактор для синтеза первичных органических соединений. В связи с этим широко исследовались происхождение и эволюция протопланетных дисков [см. обзор: 2]. В простейшем случае считается, что протопланетный диск состоит из газа (в основном, водорода и гелия) и твердых пылевых частиц, из которых в конце концов и образуются планеты. Полное моделирование динамики протопланетного диска должно включать в себя решение системы

[1] Работа выполнена при поддержке программы президиума РАН «Происхождение и эволюция биосферы» № 18-2, программы президиума РАН П-04 «Происхождение и эволюция звезд и галактик», программы СО РАН по супер-ЭВМ, грантов РФФИ 05-01-00665 и 08-01-00615, программы Рособразования «Развитие научного потенциала ВШ» (проект РНП.2.2.1.1.3653 и проект РНП.2.2.1.1.1969).

уравнений газовой динамики, уравнений, моделирующих движение пылевых частиц при наличии сил гравитации, взаимного трения. Необходимо учитывать излучение центрального тела, коагуляцию частиц, химические реакции и другие физико-химические процессы. В настоящей работе рассмотрена только «механическая» часть этой модели, включающая решение уравнений газовой динамики и бесстолкновительного кинетического уравнения Власова-Лиувилля, описывающего движение частиц.

Численное решение системы уравнений газовой динамики ведется на равномерной прямоугольной сетке с помощью явной конечно-разностной схемы [8]. Распараллеливание данной схемы не представляет трудностей, поэтому оно здесь не рассматривается. Наибольшие трудности представляет решение кинетического уравнения Власова-Лиувилля и уравнения Пуассона для гравитационного потенциала. Численное решение уравнения Власова-Лиувилля выполняется методом частиц-в-ячейках (PIC-методом) [4, 5].

Узким местом численных экспериментов является решение трехмерного уравнения Пуассона на каждом шаге по времени. Кроме того, необходимо отслеживать индивидуальные траектории большого числа частиц [7]. Наконец, трехмерные массивы плотности, потенциала и гравитационных сил должны храниться в оперативной памяти. Для изучения нелинейных процессов, таких как формирование сгустков, пространственное разрешение вычислительной сетки (т. е. ее размер) должно быть достаточно велико [6]. Сгустки рассматриваются как вероятные зародыши планет, и их размер намного меньше размера вычислительной области. Поэтому моделирование формирования сгустков будет проходить неверно при недостаточно большом пространственном разрешении (на грубой сетке).

Перечисленные выше трудности были частично устранены путем упрощения трехмерной модели диска до квазитрехмерной [8–10]. В квазитрехмерной модели вещество не имеет вертикальной скорости, но гравитационное поле должно рассматриваться в трехмерной области, поэтому модель была названа квазитрехмерной, а не просто двумерной.

Квазитрехмерная модель считается применимой для протопланетного диска, толщина которого на порядок меньше его радиуса. Еще одним условием применимости квазитрехмерной

модели диска является наличие в центре массивного тела [12]. Таким образом, квазитрехмерная модель подходит для изучения поздних стадий эволюции протопланетного диска. С другой стороны, существуют проблемы, которые не могут быть решены с помощью квазитрехмерной модели. Например, восстановление наблюдаемой диаграммы спектрального излучения (SED) протопланетного диска [3] может быть выполнено только с помощью полностью трехмерного моделирования.

На основе PIC-метода нами разработано несколько численных реализаций математической модели протопланетного диска, которые различаются способом решения уравнения Пуассона [7, 8, 11, 12]. Это было необходимо по следующим причинам:

- проблема не имеет аналитического решения для большого диапазона начальных параметров и сравнение численных экспериментов с помощью различных программ может использоваться для верификации численного решения;
- разработанные параллельные алгоритмы могут работать по-разному при различных начальных параметрах численного эксперимента, таких как число частиц и количество узлов сетки.

В настоящей работе мы рассматриваем различные стратегии параллельной реализации модели протопланетного диска в зависимости от особенностей модели. В главе 2 представлена рассматриваемая модель протопланетного диска, и кратко рассмотрена ее численная реализация. Затем, в главе 3 перечислены цели параллельной реализации модели протопланетного диска (раздел 3.2), и представлен общий метод достижения этих целей (раздел 3.3). В дальнейших главах представлены различные стратегии реализации модели протопланетного диска.

Все численные эксперименты проводились на суперкомпьютерах МВС-1000М на процессорах Alpha 21264, установленных в Сибирском суперкомпьютерном центре (Новосибирск) и Объединенном суперкомпьютерном центре (Москва). Для выполнения межпроцессорных коммуникаций использовалась библиотека MPI.

2. Модель протопланетного диска

2.1. Основные уравнения

Динамика пылевой компоненты протопланетного диска описывается кинетическим уравнением Власова-Лиувилля. Гравитационное поле определяется уравнением Пуассона. Эти уравнения представлены в следующей форме:

$$\begin{cases} \dfrac{\partial f}{\partial t} + \vec{v}\nabla f + \vec{a}\dfrac{\partial f}{\partial \vec{v}} = 0 \\ \Delta\Phi = 4\pi G\rho \end{cases} \qquad (1)$$

где $f(t, r, v)$ — зависимая от времени одночастичная функция распределения по координатам и скоростям, $\vec{a} = -\nabla\Phi$ — ускорение частицы. G — гравитационная постоянная, Φ — гравитационный потенциал. Здесь мы используем бесстолкновительное приближение усредненного самосогласованного поля. Подробное описание модели может быть найдено в [7—10].

Полная модель протопланетного диска также включает газовую динамику, радиацию, химические реакции, коагуляцию частиц пыли и т. д. Но настоящая работа сосредоточена на двух представленных уравнениях, так как параллельная реализация их решения является наиболее трудной.

2.2. Численная реализация модели

Моделирование протопланетного диска включает решение комплексной системы уравнений: кинетического уравнения Власова-Лиувилля и уравнения Пуассона. Уравнение Власова-Лиувилля в большинстве случаев [4] решается методом частиц-в-ячейках [5].

Существует большое число методов решения уравнения Пуассона (они могут быть найдены в [14]). Однако наша модель имеет особые требования и ограничения. Прежде всего, нам необходимо решать трехмерное уравнение Пуассона на очень подробных сетках. Самые быстрые методы, основанные на циклической редукции (например, FACR, DCR) не могут быть использованы на мелких сетках вследствие присущей им численной неустойчивости. Во-вторых, численный метод должен легко распаралле-

ливаться. И, наконец, уравнение Пуассона должно решаться на каждом временном шаге вычислительного эксперимента. Таким образом, итерационные методы оказываются наиболее подходящими, так как они могут использовать знание потенциала с предыдущего шага.

Разработанные нами параллельные программы, прежде всего, различаются решателями уравнения Пуассона. Были использованы следующие методы:

- 3D быстрое преобразование Фурье (в программе с декомпозицией области);
- быстрое преобразование Фурье с последовательной верхней релаксацией (3D программа в цилиндрических координатах и квази-3D программа);
- и метод, основанный на фундаментальном решении уравнения Пуассона.

В последовательном случае метод, основанный на трехмерном БПФ, является самым быстрым, если уравнение решается однократно, БПФ с ПВР быстрее обычного БПФ, если уравнение Пуассона решается в серии временных шагов. Метод, основанный на фундаментальном решении, является самым медленным из трех, но лучше всех распараллеливается.

3. Параллельная реализация модели протопланетного диска

3.1. Необходимость распараллеливания

Вычислительные ресурсы, требуемые для решения рассматриваемой проблемы, могут быть оценены, например, следующим способом. При моделировании эволюции Солнечной системы наиболее интересующая область расположена в пределах орбиты Марса, радиус которой в 40 раз меньше радиуса всей системы (радиуса орбиты Плутона). Если на расстояние радиуса орбиты Марса положить минимально необходимое разрешение в 10 узлов сетки, то общий размер сетки получится 400 узлов по радиальному направлению в цилиндрической системе координат. Таким образом, мы получаем оценку в 400^3 узлов для 3D сетки. Для того, чтобы уровень шумов составлял менее 10 %, необходимо иметь более 100 частиц на ячейку сетки. Отсюда мы получаем 6,4 млрд частиц, что в сумме составляет около 300 ГБ памяти

(3 координаты и 3 компоненты скорости с двойной точностью для каждой частицы).

Современные персональные компьютеры позволяют проводить вычислительные эксперименты с максимальным числом узлов сетки 128^3 и числом частиц не более 10 млн. И даже они требуют от двух до семи дней [12]. Таким образом, параллельная реализация математической модели (1) абсолютно необходима для моделирования протопланетного диска.

3.2. Цели распараллеливания

Ниже представлены цели создания параллельной реализации математической модели протопланетного диска:

- Проводить численные эксперименты с достаточно большим числом частиц и узлов сетки, чтобы обеспечить требуемую точность вычислений. Вычислительные эксперименты такого большого размера обычно невозможны на однопроцессорных рабочих станциях.
- Проводить серии вычислительных экспериментов (возможно, десятки вычислительных экспериментов с различными начальными данными) за обозримое время. С нашей точки зрения время, необходимое для проведения серии вычислительных экспериментов, нацеленных на проверку некоторой гипотезы, не должно превышать одного месяца.

Таким образом, высокие показатели ускорения не так важны для параллельной программы, реализующей математическую модель протопланетного диска. Более того, само понятие ускорения должно быть уточнено. Если оно рассматривается как оценка качества параллельной реализации математической модели, то его необходимо скорректировать. Ускорение обычно рассматривается как отношения времени вычислений на одном процессоре ко времени вычислений на многопроцессорной системе. Но проблема состоит в том, что вычисления на одном процессоре могут не интересовать физика.

Основным требованием к параллельной программе, реализующей модель протопланетного диска, является возможность распределить вычисления равномерно по максимальному числу процессоров. Необходимо достигнуть высокой точности вычис-

лений, используя большой объем оперативной памяти, и высокой скорости вычислений, используя большое число процессоров. Чтобы обеспечить возможность распределить вычисления, необходимо иметь:

- равномерную загрузку процессоров;
- минимальный объем межпроцессорных коммуникаций.

3.3. Стратегии распараллеливания

При выборе метода распараллеливания нужно принять во внимание то, что различные параллельные реализации могут иметь различные времена вычислений в зависимости от параметров численных методов. Поэтому мы рассматриваем различные стратегии распараллеливания.

Кроме того для параллельной реализации математической модели должны быть выбраны специальные численные методы, т. е. такие методы, которые позволяют разделить вычисления на максимальное число независимых частей. Это одна из ключевых идей сборочной технологии синтеза параллельных программ [15]. Это условие удовлетворяется благодаря использованию PIC-метода для решения уравнения Власова-Лиувилля и метода преобразования Фурье для решения уравнения Пуассона.

Дискретное преобразование Фурье сводит трехмерное уравнение Пуассона (в цилиндрической системе координат):

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \Phi}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 \Phi}{\partial \varphi^2}+\frac{\partial^2 \Phi}{\partial z^2}=4\pi G\rho \qquad (2)$$

к набору независимых двумерных уравнений для гармоник потенциала (обозначенных здесь как $H_k(r,z)$), k — номер гармоники):

$$\begin{gathered}\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial H}{\partial r}\right)+\frac{1}{r^2}\sin\frac{\pi k}{N_\varphi}H+\frac{\partial^2 H}{\partial z^2}=4\pi GR,\ k=0,\ ...,\ N_\varphi-1\\ H_k(r,z)=\sum_{k=0}^{N_\varphi-1}\Phi(r,\varphi,z)\cos\frac{\pi k\varphi}{N_\varphi}\ R_k(r,z)=\sum_{k=0}^{N_\varphi-1}\rho(r,\varphi,z)\cos\frac{\pi k\varphi}{N_\varphi}\end{gathered} \qquad (3)$$

Так что вполне естественным является использование этой особенности для распараллеливания решения уравнения Пуассона, распределяя группы гармоник по процессорам.

PIC-метод сводит решение кинетического уравнения Власова-Лиувилля к решению уравнений движения отдельных частиц. Так как вычисление координат и скоростей каждой частицы не зависит от других частиц, PIC-метод является естественным для параллельного решения уравнения Власова-Лиувилля.

Таким образом, возможность распараллеливания должна присутствовать в самой структуре применяемых математических методов, как это видно в формуле (3).

4. Параллельная программа для квази-3D модели

Параллельная реализация 3D модели диска потребует межпроцессорных пересылок трехмерных массивов. Самый легкий способ реализации параллельной модели протопланетного диска — свести трехмерную модель к квазитрехмерной. В квазитрехмерной модели вертикальное движение диска не учитывается, однако гравитационное поле должно рассматриваться как трехмерное.

Уравнения Пуассона решается на сетке в цилиндрической координатной системе, чтобы учесть симметрию диска и избавиться от нефизических структур, возникающих при решении на декартовой сетке. Подробности используемого метода решения уравнения Пуассона даны в [8, 13]. Сначала по радиальному направлению применяется БПФ, а затем каждая гармоника потенциала решается методом последовательной верхней релаксации.

Рассматриваемый метод решения уравнения Пуассона позволяет полностью исключить обмены во время выполнения итераций. Это достигается благодаря тому, что уравнения для гармоник потенциала не зависят друг от друга. После итерационной стадии потенциал должен быть собран со всех процессоров для выполнения дальнейших вычислений. Поэтому возможно разделить вычислительную область на полностью независимые подобласти в соответствии с волновым числом по углу.

Частицы также равномерно распределяются между процессорами независимо от их расположения в пространстве. Так как частица может в процессе моделирования может перелетать в любую точку диска, каждый процессор должен иметь значение потенциала со всей поверхности диска.

На каждом временном шаге обмен данными выполняется дважды. Сначала после достижения сходимости гармоники

потенциала в плоскости диска собираются для выполнения обратного преобразования Фурье. Затем частичные распределения плотности, вычисленные на каждом процессоре, суммируются и рассылаются по всем процессорам.

Такие взаимодействия типа все-со-всеми стали возможными благодаря тому, что модель является квазитрехмерной: передаются двумерные массивы вместо трехмерных в случае полной 3D модели.

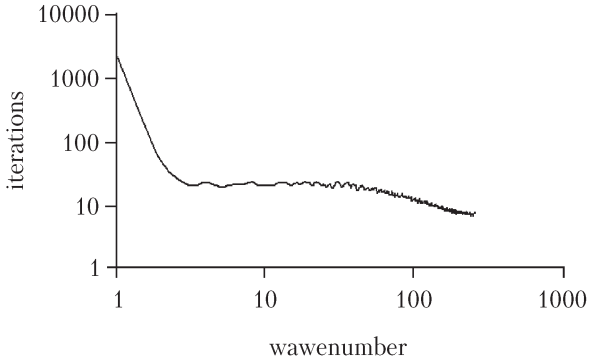


Рис. 1. Число итераций ПВР в зависимости от волнового числа

Двумерные системы для гармоник потенциала требуют различного числа итераций для сходимости, как показано на рис. 1. Здесь число итераций зависит от обусловленности матрицы системы уравнений. Это значит, что процессоры будут иметь различную загрузку при одинаковом числе гармоник. Таким образом, изначально одинаковая загрузка не может быть обеспечена для всех процессоров. Есть два пути решения этой проблемы: использовать более быстрые методы в тех случаях, где скорость сходимости низкая, и использовать динамическую балансировку загрузки. Динамическая балансировка загрузки здесь означает перемещение некоторых гармоник с перегруженных процессоров.

В результате упрощения 3D модели до квазитрехмерной мы получили параллельную программу, которую было довольно просто реализовать, и которая способна выдавать приемлемые физические результаты. Тем не менее, в квази-3D модели загрузка процессоров не равномерная, и ускорение не слишком высо-

кое: в 5 раз на 32 процессорах для сетки размером $400 \times 512 \times 200$ узлов с 20 млн частиц. Время одного временного шага моделирования для задачи указанного размера составляет 25 с на один процессор. Ускорение небольшое из-за того, что наибольшее время отнимало решение системы линейных уравнений с плохо обусловленной матрицей для нулевой гармоники. Даже выделение ее на отдельный процессор не изменяло ситуации. Очевидно, что для получения более высокого ускорения необходимо использовать другой метод.

5. Параллельная программа, основанная на фундаментальном решении уравнения Пуассона

Уравнение Пуассона имеет фундаментальное решение в форме:

$$\Phi(\tau) = \int \frac{\rho(\tau')}{|R|} d\tau' \tag{4}$$

где $|R| = |\tau - \tau'|$, $\Phi(\tau)$ — это потенциал, который должен быть вычислен в точке τ. Пусть задана двумерная вычислительная область в полярных координатах радиуса R_N. Тогда равномерная 2D сетка для потенциала Φ с шагами h_r и h_φ вводится следующим образом:

$$\begin{cases} r_i = h_r \cdot i, \quad i = 0,1,\ldots,N_r, \quad h_r = \dfrac{R_N}{N_r} \\ \varphi_j = h_\varphi \cdot j, \quad j = 0,1,\ldots,N_\varphi, \quad h_\varphi = \dfrac{2\pi}{N_\varphi} \end{cases} \tag{5}$$

В узлах заданной сетки интеграл (4) заменяется на сумму:

$$\Phi_{i',j'} = \sum_i \sum_j \frac{M_{i,j}}{R_{i,j,i',j'}} \tag{6}$$

$\Phi_{i',j'}$ вычисляется путем суммирования потенциалов точечных масс, находящихся в центрах ячеек сетки. Таким образом, сетка на которой берутся точечные массы, оказывается сдвинутой относительно сетки, на которой вычисляется потенциал, а $R_{i,j,i',j'}$ — расстояние от узла (i', j') сетки потенциала до узла (i, j) сетки точечных масс.

Вычисление потенциала по формуле (6) может быть просто распределено между процессорами, например, следующим образом:

$$\Phi_{i',j'} = \sum_{i=N_r/N_{proc}\cdot rank}^{N_r/N_{proc}\cdot(rank+1)} \sum_j \frac{M_{i,j}}{R_{i,j,i',j'}} \qquad (7)$$

Здесь N_{proc} — число процессоров, *rank* — номер процессора (*rank* = 0, ..., N_{proc} — 1). Данный метод дает полностью равномерную загрузку процессоров, и, следовательно, ускорение параллельной программы получается высоким (см. табл. 1).

Таблица 1

Время счета для различного числа процессоров для сетки $N_r = 400$, $N\phi = 400$ *узлов*

Число процессоров	**Время, минуты**
1	75
12	5,55
25	2,63
50	1,36
100	0,75
200	0,49

К сожалению, для небольшого числа процессоров вычисление потенциала по формуле (7) выполняется намного медленнее, чем для метода, представленного в главе 4. Более того, обе программы могут выполнять моделирование динамики протопланетного диска только для ограниченного круга задач.

6. 3D параллельная программа в цилиндрической координатной системе

Как было сказано в разделе 3.1, существуют проблемы, требующие полного 3D моделирования. Поэтому была реализована численная модель, в которой вещество протопланетного диска имеет вертикальную компоненту скорости [12].

В данном случае уравнение Пуассона также решается комбинацией БПФ по угловой координате и ПВР для отдельных гармоник потенциала, как в главе 4. Как было сказано в главе 4, одной

из трудностей является неравномерная загрузка процессоров вследствие различного времени вычислений для различных гармоник. Эта трудность присутствует и в 3D случае, как и в квази-3D, несмотря на то, что модели имеют множество различий, например, в граничных условиях, распределении плотности и т. п.

Поэтому необходимо построить такой алгоритм распределения гармоник, чтобы сделать загрузку процессоров близкой к равномерной. Принимая во внимание, что динамическая балансировка загрузки может иметь негативный эффект вследствие увеличения межпроцессорных коммуникаций, мы решили сделать статическое распределение загрузки. Этот алгоритм загрузки основан на экспериментальных данных о временах вычислений каждой гармоники:

- для вычисления гармоники с волновым числом $m = 0$ выделяется отдельный процессор;
- каждый процессор (кроме указанного выше) получает гармоники с большими и малыми волновыми числами. Каждая гармоника вычисляется только на одном процессоре.

Этот простой алгоритм обеспечивает загрузку процессоров близкую к равномерной. В табл. 2 показано время вычислений для различных этапов решения уравнения Пуассона. Распараллеленным является этап вычисления гармоник, распределенных по процессорам. Но, как видно из табл. 2, самыми затратными по времени этапами являются БПФ для плотности и потенциала, а также сборка вычисленных гармоник (передача данных между процессорами), так как они работают с 3D массивами данных.

Таблица 2

Время работы (в секундах) для различных этапов решения уравнения Пуассона на одном временном шаге на процессор

Число процессоров	**БПФ для плотности**	**Вычисление гармоник**	**Сборка гармоник**	**БПФ для потенциала**	**Общее время**
10	0,3	0,7	0,65	0,4	2,05

Следует заметить, что сложность с долгим вычислением БПФ возникает только в 3D случае, потому что преобразование Фурье должно быть применено к целому 3D массиву плотности и потенциала. Это означает, что дальнейшее улучшение

параллельной программы, реализующей 3D модель протопланетного диска возможна только путем декомпозиции области моделирования.

7. 3D параллельная программа, основанная на декомпозиции области моделирования

Декомпозиция области используется для того, чтобы удовлетворить требованиям параллельной реализации модели протопланетного диска, данным в разделе 3.2. Более того, частицы, принадлежащие одной подобласти, распределяются между процессорами, каждый из которых содержит часть сетки соответствующей подобласти.

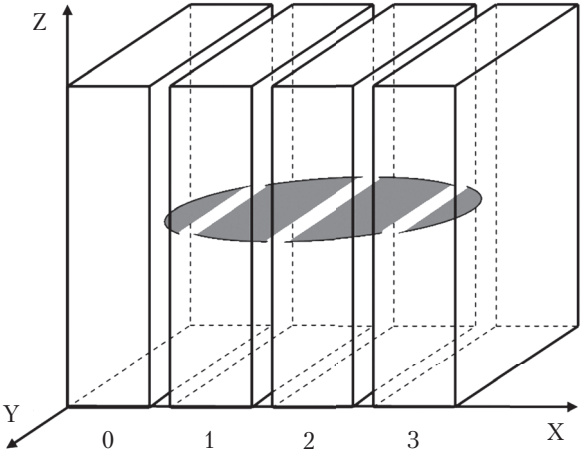


Рис. 2. Декомпозиция 3D области моделирования

Декомпозиция выполняется простейшим образом — путем разделения области на части одинакового размера вдоль одного из координатных направлений, как показано на рис. 2. В этом случае каждый слой имеет только два смежных слоя. В результате обмен частицами со смежными слоями выполняется очень просто.

Исходя их свойств решения задачи моделирования протопланетного диска, большинство частиц расположено около плоскости диска. Диск вращается вокруг оси, параллельной оси Z.

Поэтому разделение области на подобласти осуществляется плоскостями, параллельными плоскости YZ. Таким образом, мы избегаем заведомо неравномерного распределения частиц по процессорам, которое может получиться, если разрезать область параллельно плоскости XY. В текущей реализации вычислительная область разделяется на слои почти одинаковой толщины с частичным перекрытием границ.

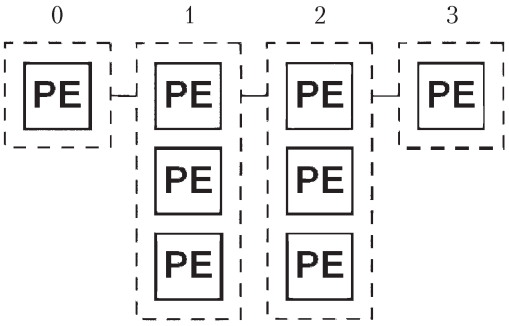


Рис. 3. Распределение групп процессоров по подобластям

Уравнение Пуассона решается на равномерной сетке в декартовой системе координат. Такая простая сетка позволила нам использовать достаточно эффективный метод, основанный на трехмерном быстром преобразовании Фурье. Для дискретизации вторых производных в уравнении Пуассона используются 27-точечный шаблон. БПФ на многопроцессорной вычислительной системе выполняется с использованием свободно распространяемой библиотеки FFTW.

Чтобы обеспечить равномерное распределение частиц между процессорами, каждому слою сопоставляется группа процессоров, как показано на рис. 3. Сеточные значения для всего слоя (потенциала, плотности и т. д.) хранятся во всех процессорах группы. Каждая группа имеет главный процессор. Распределение процессоров по группам выполняется статически на основе распределения частиц по всей области моделирования. В ходе вычислений частицы перелетают из одного слоя в другой, поэтому они должны пересылаться между процессорами. Алгоритм выбора отправителя и получателя сохраняет распределение частиц в группах, близкое к равномерному.

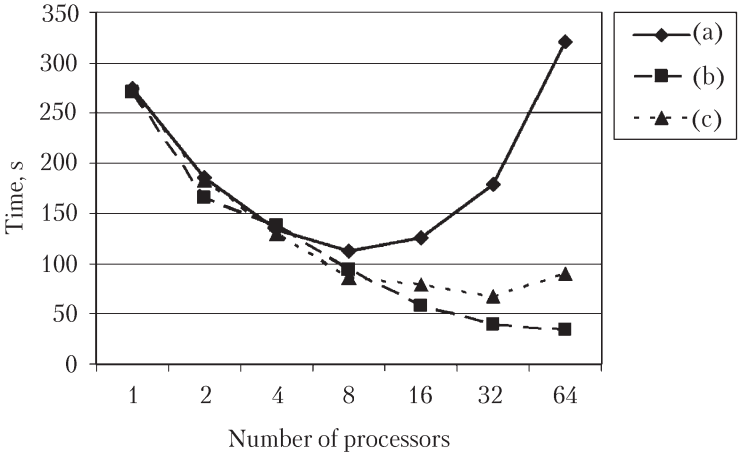


Рис. 4. Время работы программы на основе декомпозиции области:
a — одна подобласть, b — число подобластей равно числу процессоров,
c — три подобласти

Рис. 4 показывает, что когда подобласть обрабатывается группой процессоров (вариант «c»), время работы немного дольше, чем когда каждая подобласть обрабатывается одним процессором (вариант «b»). Но в первом случае есть возможность рассчитывать модель большего размера, что является главной целью параллельной реализации модели протопланетного диска, как было сказано в разделе 3.2.

Двумерные распределения плотности, потенциала и т. д., полученные с помощью квази-3D и 3D программы почти идентичны в том случае, если квази-3D модель корректна. Итак, полностью трехмерная модель необходима, чтобы отслеживать вертикальное движение вещества.

Заключение

Программа для квази-3D модели может рассматриваться как прототип полной параллельной реализации модели протопланетного диска. Программа, основанная на фундаментальном решении, достигает хороших результатов в качестве параллельной реализации модели диска (хорошее ускорение) вследствие сильного упрощения модели. 3D модель в цилиндрической системе

координат ясно демонстрирует все трудности полной 3D реализации модели протопланетного диска. Программа на основе декомпозиции области лучше всего удовлетворяет требованиям модели протопланетного диска: она полностью трехмерная, она теоретически не имеет ограничений на размер сетки и число частиц, она может значительно сократить время вычислений с увеличением числа процессоров.

Литература

1. **Snytnikov V. N., Dudnikova G. I., Gleaves J. T., Nikitin S. A., Parmon V. N., Stoyanovsky V. O., Vshivkov V. A., Yablonsky G. S., Zakharenko V. S.** Space chemical reactor of protoplanetary disk // Space Res. — 2002. — Vol. 30. — № 6. — P. 1461—1467.

2. **Woolfson M. M.** The Origin and Evolution of the Solar System. — Bristol : IoP Publishing, 2000.

3. **D'Alessio P., Calvet N., Hartmann L., Lizano S., Canto J.** Accretion disks around young objects // Astrophysical Journal. — 1999. — Vol. 2 Tests of well-mixed models with ISM dust. — P. 893—909.

4. **Hockney R. W., Eastwood J. W.** Computer Simulation Using Particles. — Bristol : IOP Publishing, 1988.

5. **Grigoryev Yu. N., Vshivkov V. A., Fedoruk M. P.** Numerical «Particle-in-Cell» Methods. Theory and Applications. — VSP. 2002.

6. **Nikitin S. A., Snytnikov V. N., Vshivkov V. A.** Ion-acoustic type instability in protoplanetary disk // Plasma in the Laboratory and in the Universe. AIP Conf. Proc — 2004. — Vol. 703. — P. 280—283.

7. **Snytnikov V. N., Vshivkov V. A., Neupokoev E. V., Nikitin S. A., Parmon V. N., Snytnikov A. V.** Three-Dimensional Numerical Simulation of a Nonstationary Gravitating N-Body System with Gas // Astronomy Letters. — 2004. — Vol. 30. — № 2. — P. 124—138.

8. **Kuksheva E. A., Malyshkin V. E., Nikitin S. A., Snytnikov A. V., Snytnikov V. N., Vshivkov V. A.** Numerical Simulation of Self-Organisation in Gravitationally Unstable Media on Supercomputers. PaCT-2003 proceedings. LNCS 2763. — 2003. — P. 354—368.

9. **Kuksheva E. A., Malyshkin V. E., Nikitin S. A., Snytnikov V. N., Vshivkov V. A.** Supercomputer Simulation of Self-Gravi-

tating Media // Future Generation Computer Systems. — 2005. — Vol. 21. — P. 749—757.

10. **Snytnykov A.** Parallel gravitational solver for protoplanetary disc simulation // Science and Technology, 2003. Proceedings KORUS 2003: The 7th Korea-Russia International Symposium. — 2003. — Vol. 2. — P. 390—395.

11. **Kuksheva E. A., Snytnikov V. N.** Parallel implementation of the fundamental solution for Poisson equation // Vychislitel'nye technologii. — 2005. — Vol. 10. — № 4. — P. 63—71. — [in Russian].

12. **Vshivkov V. A., Snytnikov N. V., Snytnikov V. N.** Simulation of three-dimensional dynamics of matter in gravitational field with the use of multiprocessor computer // Vychislitel'nye technologii. — 2006. — Vol. 11. — № 2. — P. 15—27. — [in Russian].

13. **Snytnikov A. V., Vshivkov V. A.** A Multigrid Parallel Program for Protoplanetary Disc Simulation. PaCT-2005 proceedings. LNCS 3606. — 2005. — P. 457—467.

14. **Demmel J.W.** Applied Numerical Linear Algebra. SIAM. — 1997.

15. **Kraeva M. A., Malyshkin V. E.** Assembly Technology for Parallel Realization of Numerical Models on MIMD-multicomputers // Future Generation Computer Systems. — 2001. — Vol 17. — № 6. — P. 755—765.

Распараллеливание вычислений при построении гиперболических сплайнов[1]

Б. И. Квасов
Институт вычислительных технологий СО РАН
Россия, 630090, г. Новосибирск, пр-т Лаврентьева, 6

Задача построения изогеометрического гиперболического сплайна может быть сформулирована как дифференциальная многоточечная краевая задача (ДМКЗ). Ее дискретизация приводит к необходимости решения линейной системы с пятидиагональной матрицей. Прямое расщепление этой системы на цепочку трехдиагональных систем дает систему, коэффициенты которой вычисляются с помощью гиперболических функций. В работе показано, что применение алгоритма распараллеливания прогонки позволяет избежать этой трудности. Приведены числовые примеры.

Введение

В настоящее время основным аппаратом компьютерной геометрии [1—3] при описании сложных кривых и поверхностей являются методы сплайн-функций [4—6]. Вычерчивание кривых и поверхностей по дискретным данным требует наличия сплайновых методов, которые сохраняли бы геометрические свойства исходных данных такие, как положительность, монотонность, выпуклость и т. д. Стандартные методы сплайн-интерполяции не дают удовлетворительного решения этой задачи. Задача построения по дискретным данным сложных кривых и поверхностей с сохранением их формы называется задачей изогеометрической интерполяции.

Введенные в рассмотрение Швайкертом в 1966 г. [7] гиперболические сплайны с натяжением остаются до сих пор весьма популярным аппаратом решения задачи изогеометрической интерполяции [8—10]. Такие сплайны обладают достаточными для многих приложений свойствами гладкости и в сочетании

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований по проекту 06-01-00030.

с алгоритмами автоматического выбора параметров натяжения [5, 11, 12] хорошо адаптируют поведение сплайна по отношению к данным. К сожалению, задача вычисления значений гиперболических сплайнов из-за ошибок округления (при малых значениях параметров натяжения) и проблем переполнения (при больших значениях параметров натяжения) является очень трудной.

В работах [5, 10] рассматривался сеточный метод табуляции гиперболических сплайнов, основанный на решении пятидиагональной линейной системы, который не требует вычисления значений гиперболических функций. Расщепление этой системы на трехдиагональные системы приводило к необходимости вычисления гиперболических функций. В данной работе показано, что табуляцию гиперболических сплайнов без вычисления значений гиперболических функций можно выполнить путем решения как трех- так и пятидиагональных линейных систем. Приводимые алгоритмы допускают эффективное распараллеливание для многопроцессорных ЭВМ. Особенности и преимущества предлагаемой методики иллюстрируются на примерах решения задачи изогеометрической интерполяции.

1. Постановка задачи

Пусть имеются данные

$$(x_i, f_i),\ i = 0,1,\ ...,\ N+1 \tag{1}$$

где $a = x_0 < x_1 < \ldots < x_{N+1} = b$. Положим

$$f[x_i, x_{i+1}] = (f_{i+1} - f_i)/h_i,\ h_i = x_{i+1} - x_i,\ i = 0,1,\ ...,\ N$$

Данные (1) будем называть монотонно возрастающими, если

$$f[x_i, x_{i+1}] \geq 0,\ i = 0,1,\ ...,\ N$$

и выпуклыми, если

$$f[x_i, x_{i+1}] \geq f[x_{i-1}, x_i],\ i = 1,2,\ ...,\ N$$

Задача изогеометрической интерполяции состоит в построении достаточно гладкой функции S такой, что $S(x_i) = f_i$, $i = 0,1,\ ...,\ N+1$, и S монотонна/выпукла на участках монотонности/выпуклости исходных данных.

Очевидно, что решение задачи изогеометрической интерполяции не единственно. Будем искать решение этой задачи в виде гиперболического сплайна с натяжением.

Определение 1. Интерполяционным гиперболическим сплайном S с множеством параметров натяжения $\{p_i \geq 0,\ i = 0,1,\ ...,\ N\}$ назовем решением дифференциальной многоточечной краевой задачи (сокращенно ДМКЗ):

$$\frac{d^4 S}{dx^4} - \left(\frac{p_i}{h_i}\right)^2 \frac{d^2 S}{dx^2} = 0 \text{ для всех } x \in (x_i, x_{i+1}),\ i = 0,1,\ ...,\ N, \qquad (2)$$

$$S \in C^2[a, b] \qquad (3)$$

с условиями интерполяции

$$S(x_i) = f_i,\ i = 0,1,\ ...,\ N+1 \qquad (4)$$

и краевыми условиями

$$S''(a) = F_0'' \text{ и } S''(b) = F_{N+1}''. \qquad (5)$$

Краевые условия (5) используются здесь для простоты изложения. Могут быть рассмотрены краевые условия и других типов [5] $i = 1,2,...,N, [x_i, x_{i+1}]\ j = -1,0,...,n_i + 1 n \in \mathrm{N}\ \tau_i = h_i / n_i$.

2. Конечно-разностная аппроксимация

Рассмотрим теперь дискретизацию сформулированной ДМКЗ. Для этого на каждом подотрезке введем дополнительную равномерную сетку с шагом, и узлами $x_{ij} = x_i + j\tau_i,\ i = 0,1,...,N$. Будем искать сеточную функцию

$$\{u_{ij},\ j = -1,0,...,n_i + 1,\ i = 0,1,...,N\}$$

удовлетворяющую разностным уравнениям

$$\left[\Lambda_i^2 - \left(\frac{p_i}{h_i}\right)^2 \Lambda_i\right] u_{ij} = 0,\ j = 1,2,...,n_i - 1,\ i = 0,1,...,N, \qquad (6)$$

где

$$\Lambda_i u_{ij} = \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{\tau_i^2}$$

Аппроксимация условий гладкости (3) дает соотношения

$$u_{i-1,n_{i-1}} = u_{i0}$$

$$\frac{u_{i-1,n_{i-1}+1} - u_{i-1,n_{i-1}-1}}{2\tau_{i-1}} = \frac{u_{i,1} - u_{i,-1}}{2\tau_i},\ \Lambda_{i-1}u_{i-1,n_{i-1}} = \Lambda_i u_{i,0} \tag{7}$$

Условия (4) и (5) преобразуются к виду

$$u_{i0} = f_i, i = 0,1,...,N,\ u_{N,n_N} = f_{N+1} \tag{8}$$

и

$$\Lambda_0 u_{0,0} = f_0^{''},\ \Lambda_N u_{N,n_N} = f_{N+1}^{''} \tag{9}$$

Соотношения (7) и краевые условия (9) позволяют исключить «лишние» неизвестные $u_{i,-1}, u_{i,n_i+1}, i = 0,1,...,N$. Дискретное сеточное решение теперь будет определено как

$$\left\{u_{ij},\ j = 1,0,...,n_i, i = 0,1,...,N\right\} \tag{10}$$

Существование и единственность решения полученной линейной системы (6)—(9) будут доказаны в следующем пункте. Обсуждение практических аспектов прямого решения этой пятидиагональной системы проведено в [5]. Нахождение ее решения (10) не требует вычисления значений гиперболических функций.

3. Расщепление системы и продолжение сеточного решения

Для изучения решения системы (6)—(9) введем обозначение

$$m_{ij} = \Lambda_i u_{ij},\ j = 0,1,...,n_i, i = 0,1,...,N \tag{11}$$

Тогда на отрезке $\left[x_i, x_{i+1}\right]$ разностные уравнения (6) принимают вид

$$\begin{gathered} m_{i0} = m_i \\ \frac{m_{i,j-1} - 2m_{i,j} + m_{i,j+1}}{\tau_i^2} - \left(\frac{p_i}{h_i}\right)^2 m_{i,j} = 0,\ j = 1,2,...,n_i - 1 \\ m_{i,n_i} = m_{i+1} \end{gathered} \tag{12}$$

где m_i и m_{i+1} — заданные числа. Система (12) имеет единственное решение, которое может быть записано следующим образом:

$$m_{ij} = M_i\left(x_{ij}\right),\ j = 0,1,\ldots,n_i$$

где

$$M_i(x) = m_i \frac{\mathrm{sh}\left(k_i(1-t)\right)}{\mathrm{sh}\left(k_i\right)} + m_{i+1}\frac{\mathrm{sh}\left(k_i t\right)}{\mathrm{sh}\left(k_i\right)},\ t = \frac{x - x_i}{h_i} \qquad (13)$$

Параметры k_i являются решениями трансцендентных уравнений

$$2n_i sh\frac{k_i}{2n_i} = p_i,\ \ p_i \geq 0$$

т. е.

$$k_i = 2n_i \ln\left(\frac{p_i}{2n_i} + \sqrt{\left(\frac{p_i}{2n_i}\right)^2 + 1}\right) \geq 0,\ i = 0,1,\ldots,N$$

В силу соотношений (11) и с учетом условий интерполяции (8) имеем

$$u_{i0} = f_i$$

$$\frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\tau_i^2} = m_{i,j},\ j = 0,1,\ldots,n_i \qquad (14)$$

$$m_{i,n_i} = f_{i+1}.$$

Для всякой последовательности $m_{i,j}$, $j = 0,1,\ldots,n_i$, система (14) имеет единственное решение, которое может быть представлено в виде

$$u_{ij} = U_i(x_{ij}),\ j = -1,0,\ldots,n_i + 1$$

где

$$U_i(x) = f_i(1-t) + f_{i+1}t + \varphi_i(1-t)h_i^2 m_i + \varphi_i(t)h_i^2 m_{i+1}$$
$$\varphi_i(t) = \frac{\mathrm{sh}\left(k_i t\right) - t\,\mathrm{sh}\left(k_i\right)}{p_i^2\,\mathrm{sh}\left(k_i\right)} \qquad (15)$$

Отметим, что формулы (13) и (15) связаны соотношением

$$M_i(x) = \Lambda_i U_i(x) = \Lambda_i \varphi_i(1-t) h_i^2 m_i + \Lambda_i \varphi_i(t) h_i^2 m_{i+1} \qquad (16)$$

Чтобы решить системы (12) и (14), требуется единственно определить значения m_i, $i = 0,1, ..., N+1$ так, чтобы выполнялись условия стыковки (7) и краевые условия (9). Согласно уравнениям (14) и (15), условия (7) можно переписать в виде

$$U_{i-1}(x_i) = U_i(x_i)$$

$$\frac{U_{i-1}(x_i + \tau_{i-1}) - U_{i-1}(x_i - \tau_{i-1})}{2\tau_{i-1}} = \frac{U_i(x_i + \tau_i) - U_i(x_i - \tau_i)}{2\tau_i} \qquad (17)$$

$$\Lambda_{i-1} U_{i-1}(x_i) = \Lambda_i U_i(x_i)$$

где

$$\Lambda_j U_j(x) = \frac{U_j(x + \tau_j) - 2U_j(x) + U_j(x - \tau_j)}{\tau_j^2}, \; x \in [x_j, x_{j+1}].$$

Согласно уравнениям (11), (12) и (15), первое и третье равенства в (17) тождественно выполняются. В силу формулы (15) и краевых условий (9), второе равенство в соотношениях (17) дает нам следующую линейную систему с трехдиагональной матрицей для неизвестных значений m_i:

$$m_0 = f_0''$$

$$\alpha_{i-1} h_{i-1} m_{i-1} + (\beta_{i-1} h_{i-1} + \beta_i h_i) m_i + \alpha_i h_i m_{i+1} = d_i, \; i = 1,2, ..., N \qquad (18)$$

$$m_{N+1} = f_{N+1}'',$$

где

$$d_i = \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}},$$

$$\alpha_i = -\frac{\varphi_i\left(1/n_i\right) - \varphi_i\left(-1/n_i\right)}{2/n_i} = -\frac{n_i sh\left(k_i/n_i\right) - sh(k_i)}{p_i^2 sh(k_i)}$$

$$\beta_i = \frac{\varphi_i\left(1+\,{}^{1}\!/_{n_i}\right)-\varphi_i\left(1-\,{}^{1}\!/_{n_i}\right)}{{}^{2}\!/_{n_i}} = \frac{n_i ch(k_i)\, sh\left({}^{k_i}\!/_{n_i}\right)-sk(k_i)}{p_i^2 sh(k_i)}$$

Используя разложение гиперболических функций в приведенных выше выражениях в степенные ряды, получаем

$$\beta_i \geq 2\alpha_i > 0, i = 0,1,...,N \text{ для всех } n_i > 1, p_i \geq 0.$$

Следовательно, система (18) имеет диагональное преобладание и в силу этого обладает единственным решением. Теперь мы можем заключить, что система (6)—(9) имеет единственное решение, которое может быть представлено в виде $U_i(x_{ij})$, $j = -1,0,...,n_i+1, i = 0,1,...,N$, тогда как постоянные m_i находятся из системы (18).

$$m_i = 2U[x_i - \tau_{i-1}, x_i, x_i + \tau_i], u_{i,n_i+1} = 2f_{i+1} - u_{i,n_i-1} + \tau_i^2 m_{i+1}$$

$$\frac{u_{i,1}-u_{i,-1}}{2\tau_i} = \frac{u_{i,1}-f_i}{\tau_i} - \frac{\tau_i}{2}m_i,\ u_{i,1},\ u_{i,-1} = 2f_i - u_{i,1} + \tau_i^2 m_i, x \in [x_i, x_{i+1}],$$

$$\frac{\tau_{i-1}+\tau_i}{2}m_i = \frac{u_{i,1}-f_i}{\tau_i} - \frac{f_i - u_{i-1,n_{i-1}-1}}{\tau_{i-1}}, U(x) = U_i(x),$$

Положим

$$U(x) = U_1(x), x \in [x_i, x_{i+1}]\ \ x \in [x_i, x_{i+1}],\ i = 0,1,...,N \qquad (19)$$

В силу приведенного построения будем называть U дискретным гиперболическим сплайном с натяжением, интерполирующем данные (1).

4. Распараллеливание вычислений в расщепленной системе

Коэффициенты системы (18) содержат значения гиперболических функций sh(·) и ch(·). Чтобы исключить их вычисление, воспользуемся алгоритмом распараллеливания прогонки [13].

Из уравнений (14) при $j = 0$ и $j = n_i$ имеем

$$u_{i,-1} = 2f_i - u_{i,1} + \tau_i^2 m_i,\ u_{i,n_i+1} = 2f_{i+1} - u_{i,n_i-1} + \tau_i^2 m_{i+1}$$

или

$$\frac{u_{i,1}-u_{i,-1}}{2\tau_i}=\frac{u_{i,1}-f_i}{\tau_i}-\frac{\tau_i}{2}m_i,\ \frac{u_{i,n_i+1}-u_{i,n_i-1}}{2\tau_i}=\frac{f_{i+1}-u_{i,n_i-1}}{\tau_i}+\frac{\tau_i}{2}m_{i+1}.$$

Поэтому второе из условий (17) можно переписать в виде

$$\frac{\tau_{i-1}+\tau_i}{2}m_i=\frac{u_{i,1}-f_i}{\tau_i}-\frac{f_i-u_{i-1,n_{i-1}-1}}{\tau_{i-1}} \qquad (20)$$

или по соглашению (19)

$$\frac{\tau_{i-1}+\tau_i}{2}m_i=\frac{u_{i,1}-f_i}{\tau_i}-\frac{f_i-u_{i-1,n_{i-1}-1}}{\tau_{i-1}}$$

Для исключения в (20) неизвестных величин и $u_{i-1},n_{i-1}-1$ воспользуемся следующим приемом. Пусть α_{ij} и β_{ij}, $j=0,1,...,n_i$ — решения системы (12) с краевыми условиями $m_{i0}=1$, $m_{i,n_i}=0$ и $m_{i0}=0$, $m_{i,n_i}=1$ соответственно. Тогда очевидно, что решение системы (12) можно записать в виде

$$m_{ij}=m_i\alpha_{ij}+m_{i+1}\beta_{ij},\ j=0,1,...,n_i \qquad (21)$$

Пусть теперь γ_{ij} и δ_{ij} — решения системы (14) с краевыми условиями $u_{i0}=u_{i,ni}=0$ и правыми частями α_{ij} и β_{ij} соответственно, а ε_{ij} – решение однородной системы (14) с краевыми условиями $\varepsilon_{i0}=f_i$ и $\varepsilon_{i,ni}=f_{i+1}$. Так как

$$\varepsilon_{ij}=f_i\left(1-t_{ij}\right)+f_{i+1}t_{ij},\ j=0,1,...,n_i$$

где $t_{ij}=\dfrac{\left(x_{ij}-x_i\right)}{h_i}$, то для решения системы (14) получаем

$$u_{ij}=f_i\left(1-t_{ij}\right)+f_{i+1}t_{ij}+m_i\gamma_{ij}+m_{i+1}\delta_{ij},\ j=0,1,...,n_i \qquad (22)$$

Пользуясь этой формулой и подставляя $u_{i,1}$ и $u_{i-1,n_{i-1}-1}$ в (20), находим с учетом краевых условий (9)

$$m_0=f_0^{''},$$

$$-\frac{\gamma_{i-1,n_{i-1}-1}}{\tau_{i-1}}m_{i-1}+\left(\frac{\tau_{i-1}+\tau_i}{2}-\frac{\delta_{i-1,n_{i-1}-1}}{\tau_{i-1}}-\frac{\gamma_{i,1}}{\tau_i}\right)m_i-\frac{\delta_{i,1}}{\tau_i}m_{i+1}= \qquad (23)$$

$$= \frac{f_{i+1} - f_i}{h_i} - \frac{f_i - f_{i-1}}{h_{i-1}},$$
$$i = 1,2,...,N, m_{N+1} = f''_{N+1}$$

Покажем, что эта система отличается от системы (18) лишь формой записи.

Явный вид величин α_{ij}, β_{ij}, γ_{ij}, δ_{ij} легко может быть получен из формул (13) и (15). Так как согласно формуле (15)

$$u_{ij} = f_i(1-t_{ij}) + f_{i+1}t_{ij} + \varphi_i(1-t_{ij})h_i^2 m_i + \varphi_i(t_{ij})h_i^2 m_{i+1},$$

то для коэффициентов системы (23) получаем

$$-\frac{\delta_{i1}}{\tau_i} = -\frac{1}{\tau_i}\varphi_i(t_{i1})h_i^2 = -n_i h_i \varphi_i\left(\frac{1}{n_i}\right) \qquad (24)$$

$$\frac{\tau_i}{2} - \frac{\gamma_{i1}}{\tau_i} = \frac{\tau_i}{2} - \frac{1}{\tau_i}\varphi_i(1-t_{i1})h_i^2 = h_i\left[\frac{1}{2n_i} - n_i\varphi_i\left(1-\frac{1}{n_i}\right)\right].$$

Функция φ_i удовлетворяет условиям:

$$\varphi_i(0) = \varphi_i(1) = \Lambda_i\varphi_i(0) = 0,\ \Lambda_i\varphi_i(1) = h_i^{-2}$$

откуда, в частности, следуют равенства

$$\varphi_i\left(-\frac{1}{n_i}\right) + \varphi_i\left(\frac{1}{n_i}\right) = 0,\ \varphi_i\left(1-\frac{1}{n_i}\right) + \varphi_i\left(1+\frac{1}{n_i}\right) = \frac{1}{n_i^2}$$

С учетом этих соотношений из равенств (24) имеем

$$-\frac{\delta_{i1}}{\tau_i} = -h_i\frac{\varphi_i\left(1/n_i\right) - \varphi_i\left(-1/n_i\right)}{2/n_i} = \alpha_i h_i,$$

$$\frac{\tau_i}{2} - \frac{\gamma_{i1}}{\tau_i} = h_i\frac{\varphi_i\left(1+1/n_i\right) - \varphi_i\left(1-1/n_i\right)}{2/n_i} = \beta_i h_i$$

Аналогично получаем, что

$$\frac{\gamma_{i-1,n_{i-1}-1}}{\tau_{i-1}} = \alpha_{i-1}h_{i-1}, \frac{\tau_{i-1}}{2} - \frac{\delta_{i-1,n_{i-1}-1}}{\tau_{i-1}} = \beta_{i-1}h_{i-1}$$

Таким образом, система (23) отличается от системы (18) лишь формой записи. Как следствие, эта система имеет диагональное преобладание. В отличие от системы (18) система (23) не требует вычисления значений гиперболических функций и может быть эффективно решена методом обычной трехточечной прогонки.

5. Алгоритмы вычисления сеточного решения

В отличие от стандартного подхода [5], основанного на прямом решении пятидиагональной линейной системы (6)—(9), рассмотренный в предыдущих двух пунктах метод использует расщепление этой системы на трехдиагональные линейные системы, решение которых также не требует вычисления значений гиперболических функций и обеспечивает дешевое и простое получение сеточного решения (10).

Сформулируем основные шаги, рассмотренного в п. 3—4 алгоритма.

Шаг 1. Решить $2(N+1)$ трехдиагональных систем (12) с краевыми условиями $m_{i,0}=1$. $m_{i,n_i}=0$ и $m_{i,0}=0$, $m_{i,n_i}=1$ для нахождения соответственно величин α_{ij} и β_{ij}, $j=0,1,...,n_i$, $i=0,1,...,N$.

Шаг 2. Решить 2(N+1) трехдиагональных систем (14) с однородными краевыми условиями $u_{i,0}=u_{i,n_i}=0$ и правыми частями α_{ij} и β_{ij} для нахождения соответственно величин γ_{ij} и δ_{ij}, $j=0,1,\ ...,\ n_i$, $i=0,1,...,N$.

Шаг 3. Из системы (23) вычислить параметры m_i, $i=0,1,...,N+1$.

Шаг 4. По формуле (22) найти значения u_{ij}, $j=0,1,...,n_i$, $i=0,1,...,N$.

В этом алгоритме решение любой из систем (12) и (14) требует выполнения $8q$ арифметических операций [4], где q — число неизвестных. Это существенно дешевле, чем прямые вычисления по формуле (15).

Шаги 1 и 2 в этом алгоритме можно объединить. Вместо $2(N+1)$ трехдиагональных систем (12) и (14) рассмотрим $N+1$ пятидиагональных систем

$$u_{i,0} = f_i, \Lambda_i u_{i,0} = m_i,$$

$$\Lambda_i^2 u_{i,j} - \left(\frac{p_i}{h_i}\right)^2 \Lambda_i u_{i,j} = 0, \ j = 1,2,..., n_i - 1, i = 0,1,..., N \qquad (25)$$

$$u_{i,n_i} = f_{i+1}, \ \Lambda_i u_{i,n_i} = m_{i+1}$$

Вначале находятся решения этих систем γ_{ij} и δ_{ij} с краевыми условиями $u_{i,0} = u_{i,n_i} = 0$ и соответственно, $\Lambda_i u_{i,0} = 1$, $\Lambda_i u_{i,n_i} = 0$ и $\Lambda_i u_{i,0} = 0$, $\Lambda_i u_{i,n_j} = 1$, т. е. решаются $2(N+1)$ пятидиагональных систем вида (25). Далее для нахождения сеточного решения u_{ij}, $j = 0,1,...,n_i$, $i = 0,1,...,N$ выполняются шаги 3 и 4 основного алгоритма.

Данные алгоритмы могут быть эффективно распараллелены и реализованы на многопроцессорных ЭВМ.

6. Прямое расщепление пятидиагональной системы

В системе (6)—(9) неизвестные $\{u_{0,-1}, i = 0,1,...,N\}$ и $\{u_{i,n_i+1}, i = 0,1,...,N-1\}$ исключаем посредством соотношений (7). Значения сеточного решения в точках задания данных x_i определяем согласно условиям интерполяции. Неизвестные $u_{0,-1}$ и u_{N,n_N+1} исключаем, используя краевые условия (9). В результате система (6)—(9) приводится в виду

$$Au = b, \qquad (26)$$

где A — пятидиагональная матрица.

Число неизвестных в системе (26) иногда может быть очень большим (например, при генерировании сетки в случае многомерной задачи интерполяции). По этой причине даже линейный рост стоимости вычислений для получения решения нашей пятидиагональной системы может оказаться неприемлемым. Однако для вычисления сеточного решения при использовании ЭВМ параллельного действия можно легко распределить вычисления между процессорами. Рассмотрим вкратце эту возможность.

Основная идея состоит в преобразовании матрицы А, вид которой для $N = 2$, $n_i = 18$ показан на рис. 1 слева, к матрице К (см. рис. 1 справа). Полагая $r_i = \sum_{\nu=0}^{i-1} (n_\nu - 1)$, замечаем, что строки

$r_i+1,\ ...,\ r_i-1$ в A соответствуют уравнениям (6) для подотрезка $[x_i,x_i+1]$. Если вычесть из K строки $r_i+1,\ ...,\ r_i+4, i=0,1,\ ...,\ N$, то получим блочную матрицу E, вид которой показан на рис. 2 слева. Соответствующая линейная система имеет совсем мало уравнений, и, решив ее, можно далее параллельно решить $N+1$ линейных систем, полученных из «оставшейся» матрицы F (см. рис. 2 справа) путем извлечения ее независимых блоков.

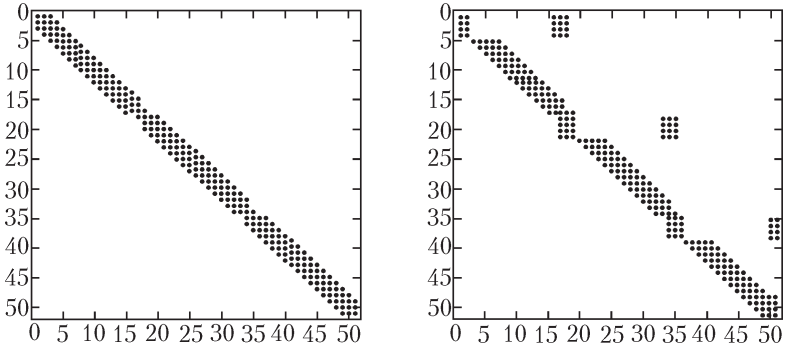


Рис. 1. Слева: вид A для $N=2$, $n_i=18$. Справа: матрица K

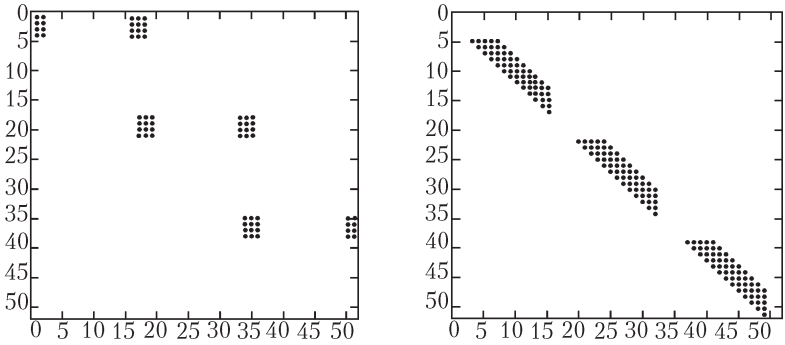


Рис. 2. Слева: блочная матрица E. Справа: блочная матрица F

Таким образом, задача состоит в том как преобразовать матрицу A в матрицу K. В п. 3 и 4 были получены следующие два факта. Исходя из структуры матрицы A, показанной на рис. 1, рассмотрим блок из строк $r_i+1,\ ...,\ r_{i+1}$. Заметим, что элементы столбцов с индексами $r_i+3,\ ...,\ r_{i+1}-2$ суть $1, a_i, b_i, a_i, 1$, являются коэффициентами разностного уравнения (6). Однако, в п. 3 было показано, что всякая функция вида

$$\Upsilon_i(x) = c_1(1-t) + c_2 t + c_3\varphi_i(1-t) + c_4\varphi_i(t) \qquad (27)$$

является решением для (6). Таким образом, если умножить строку с индексом $r_i + j, j = 1,2, \ldots, n_i - 1$ на $\Upsilon_i(x_{i,j}) = \Upsilon_i(x_i + j\tau_i)$ и затем сложить все эти строки, то вклад всех столбцов с $r_i + 3$ по $r_{i+1} - 2$ будет равен нулю. Идея получения матрицы К из матрицы А состоит в следующем: четыре строки с индексами $r_i + 1$, $r_i + 2$, $r_i + 3$, $r_i + 4$ заменяются суммой строк с $r_i + 1$ по r_{i+1}, умноженных на значения в x_{ij} четырех линейно независимых функций вида (27). Остается решить вопрос, как выбрать эти функции. Серия численных экспериментов показала, что наименьшее число обусловленности матрицы К (которое, вообще говоря, больше числа обусловленности для А) достигается при использовании фундаментальных многочленов Лагранжа, строящихся по узлам x_{iv}, ближайшим к точкам x_i, $x_i + h_i/3$, $x_{i+1} - h_i/3$, x_{i+1}.

7. Числовые примеры

В этом пункте на ряде общеупотребительных примеров будут проиллюстрированы изогеометрические свойства дискретных гиперболических сплайнов с натяжением. Заметим прежде, что наш дискретный гиперболический сплайн U_i, задаваемый формулой (15), обладает свойствами сохранения формы данных, присущими стандартному гиперболическому сплайну (см., например, [12]). А именно, функция U_i является выпуклой (вогнутой) на $[x_i, x_{i+1}]$ тогда и только тогда, когда $m_{i+j} \le 0\ (\ge 0)$, $j = 0,1$ и U_i может иметь на $[x_i, x_{i+1}]$ не более одной точки перегиба. Следовательно, чтобы сохранить форму исходных данных, достаточно проанализировать величины $\Lambda_i u_{i,0}$ и $\Lambda_i u_{i,n_i}$ и увеличить при необходимости значения параметров натяжения. Зарекомендовавшие себя стратегии автоматического выбора параметров натяжения для стандартных гиперболических сплайнов [см., напр., 5, 11, 12] могут быть успешно использованы и в нашем дискретном контексте.

В первом примере интерполировались радиохимические данные, приведенные в табл. 1. Данные табл. 1 являются неравноотстоящими. Между точками задания данных использовалась.

Таблица 1

x_i	7.99	8.09	8.19	8.7	9.2
f_i	0	2.76429E-5	4.37498E-2	0.169183	0.469428

x_i	10	12	15	20
f_i	0.943740	0.998636	0.999916	0.999994

Краевые условия задавались в виде $m_0 = \Lambda_0 u_{0,0} = 0$, $m_N + 1 = \Lambda_N u_{N,n_N} = 0$. Эффект изменения значений параметров натяжения показан на рис. 3 и 4. Рис. 3 получен при нулевых значениях параметров натяжения ($p_i = 0$ для интерполяционного дискретного гиперболического сплайна с параметрами натяжения $p_0 = p_1 = 300$, $p_i = 15$, $i = 2, 3, ..., 7$ показан на рис. 4. Очевиден эффект стягивания при увеличении значений параметров натяжения.

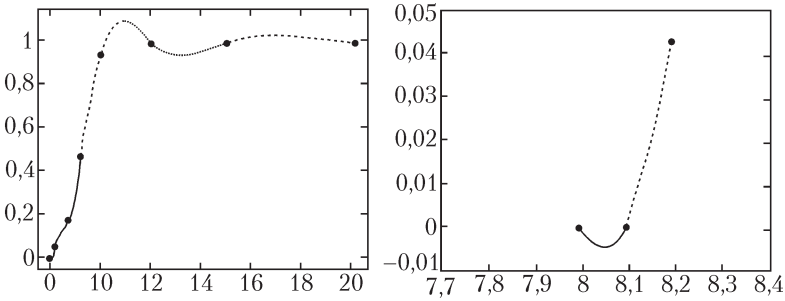


Рис. 3. Радиохимические данные с краевыми условиями $m_0 = m_N + 1 = 0$. Интерполяция дискретным гиперболическим сплайном с $p_i = 0$ для всех i. Справа: увеличение левого нижнего угла

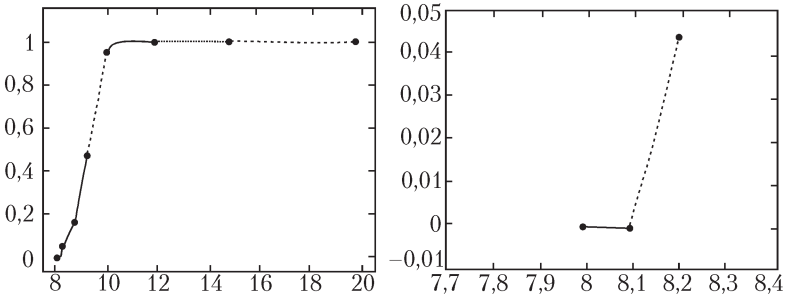


Рис. 4. То же, что и на рис. 3, с $p_0 = p_1 = 300$, $p_i = 15$, $i = 2, 3, ..., 7$

Таблица 2

Данные Акимы [14]

x_i	0	2	3	5	6	8	9	11	12	14	15
f_i	10	10	10	10	10	10	10,5	15	56	60	85

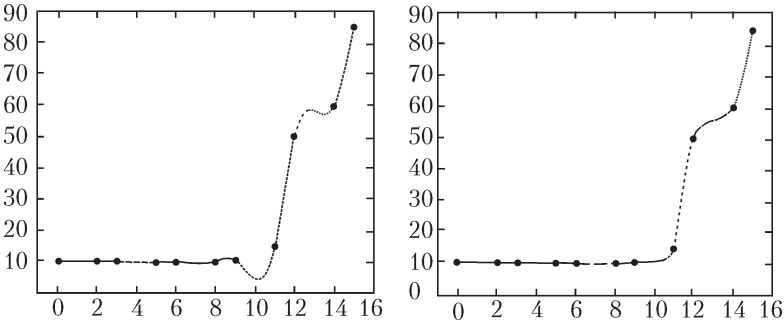


Рис. 5. Данные Акимы с краевыми условиями $m_0 = m_{N+1} = 0$.
Слева: интерполяционный дискретный кубический сплайн ($p_i = 0$).
Справа: дискретный гиперболический сплайн с $p_5 = p_6 = p_8 = 10$

Во втором примере использовались данные Акимы [14], приведенные в табл. 2. Для этих неравноостоящих данных строился интерполяционный дискретный гиперболический сплайн с 20 узлами равномерной сетки на каждом подотрезке между точками задания данных. Краевые условия брались в виде $m_0 = m_{N+1} = 0$. На рис. 5 слева показан график интерполяционного дискретного кубического сплайна, полученный при $p_i = 0$ при всех i. Заметно нарушение присущих исходным данным свойств монотонности и выпуклости. На этом же рисунке справа интерполяционный дискретный кубический сплайн при $p_5 = p_6 = p_8 = 10$ и $p_i = 0$ для остальных значений i адекватно воспроизводит форму исходных данных.

Литература

1. **Голованов Н. Н., Ильютко Д. П., Носовский Г. В., Фоменко А. Т.** — М. : Академия, 2006.

2. **Farin G.** Curves and Surfaces for Computer Aided Geometric Design. — San Diego : Academic Press, 2002.

3. **Hoschek J., Lasser D.** Fundamentals of Computer Aided Geometric Design. — Wellesley, Massachusetts : A K Peters, 1993.

4. **Завьялов Ю. С., Квасов Б. И., Мирошниченко В. Л.** Методы сплайн-функций. — М. : Наука, 1980.

5. **Квасов Б. И.** Методы изогеометрической аппроксимации сплайнами. — М. : Физматлит, 2006.

6. **Späth H.** One Dimentional Spline Interpolation Algorithms. — Massachusetts : A K Peters, 1995.

7. **Schweikert D. G.** An Interpolating Curve Using a Spline in Tension // J. Math. Phys. — 1966. — Vol. 45. — P. 312—317.

8. **Koch P. E., Lyche T.** Interpolation with exponential B-splines in tension // Geometric Modelling : Computing Supplementum 8 / Ed. by G. Farin et al. — Wien : Springer Verlag, 1993. — P. 173—190.

9. **Sadipis N. S., Kaklis P. D.** An algorythm for constructing convexity and monotonicity-preserving splines of tension // Computer Aided Geometric Design. — 1988. — Vol. 5. — P. 127—137.

10. **Kvasov B. I.** Methods of Shape-Preserving Spline Approximation. — Singapore : World Scientific Publ. Co. Pte. Ltd., 2000.

11. **Renka R. J.** Interpolation Splines with Automatic Selection of Tension Factors // SIAM J. Sci. Stat. Comp. — 1987. — Vol. 8. — P. 393—415.

12. **Rentrop P.** An algorythm for computation of exponential splines // Numer. Math. — 1980. — Vol. 35. — P. 393—415.

13. **Яненко Н. Н., Коновалов А. Н., Бугров А. Н., Шустов Г. В.** Об организации параллельных вычислений и «распараллеливании» прогонки // Численные методы механики сплошной среды. — 1978. — Т. 9. — № 7. — С. 139—146.

14. **Akima H.** A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures // J. Assoc. Comput. Mach. — 1970. — Vol. 17. — P. 589—602.

Научно-образовательный портал параллельных вычислений

К. Е. Афанасьев, А. В. Демидов, С. Н. Трофимов
Кемеровский государственный университет, Кемерово

Высокопроизводительные вычисления на ЭВМ с параллельной архитектурой получили широкое распространение в науке и промышленности. Следуя тенденциям рынка, российские вузы все шире внедряют тематику параллельного программирования в свои учебные программы. Для выпуска качественных специалистов в данной области необходимо иметь как информационно-методические материалы для теоретической подготовки, так и многопроцессорные вычислительные системы для проведения практикумов.

Кафедра ЮНЕСКО по новым информационным технологиям КемГУ приступила к чтению спецкурсов по тематике параллельных вычислений с момента своего основания в 2000 г. За это время разработано несколько курсов и накоплен большой опыт по их ведению, а сотрудники кафедры используют высокопроизводительные ресурсы не только в учебных, но и в научно-исследовательских целях. Для систематизации имеющихся материалов и организации удаленного доступа к имеющимся кластерным системам, в КемГУ разрабатывается информационно-вычислительный портал (ИВП).

ИВП является единым пространством для организации учебной и научной деятельности вуза в сфере решения различных задач с использованием вычислительного эксперимента, организации научных конференций, хранения информационных ресурсов и работы с ними.

Создание типового информационно-вычислительного портала ведется с целью обеспечения:

- доступа к пополняемой и обновляемой электронной библиотеке информационных образовательных и научных ресурсов по тематике параллельных вычислений;
- проведения численного эксперимента на высокопроизводительном кластере в режиме удаленного доступа;
- представления информации о предстоящих и проходящих научных конференциях по тематике портала;

- представления информации о жизни университета в целом;
- организации виртуальных обсуждений (форумов) — новых видов коммуникации людей, способных конкурировать с традиционными СМИ;
- доступа к пополняемой и обновляемой коллекции аннотированных ссылок на ресурсы в Интернете.

В основе технологических решений Портала лежат следующие принципы.

Открытость. Для обеспечения перспективы развития Портала предусмотрена интеграция гетерогенных вычислительных компонент и различных приложений.

Модульность. Архитектура Портала является гибкой и допускает простое, без структурных изменений, развитие и наращивание функций и ресурсов системы в соответствии с расширением сфер и задач ее применения. Этот принцип также позволяет проводить отладку, удаление, добавление компонент системы без потери работоспособности.

Масштабируемость. Архитектура Портала позволяет наращивать производительность системы, объемы хранимой и обрабатываемой информации без длительной остановки работы Портала и модификации его программного кода.

Индивидуализация. Портал обеспечивает удобный индивидуальный доступ для всех потенциальных групп пользователей с функциональностью, соответствующей задачам, решаемым каждой группой пользователей.

Функциональное соответствие. Архитектура Портала отвечает текущим и перспективным целям и функциональным задачам создаваемой системы.

Управляемость. Предоставляются гибкие, полнофункциональные механизмы управления Порталом на всех уровнях архитектуры.

Системность. Архитектура Портала такова, что все взаимосвязанные подсистемы строятся по единой методологии и отвечают единым принципам взаимодействия, надежности и управления.

Иерархическая структура. Структура Портала основывается на трехуровневой модели:

- презентационный уровень, обеспечивающий представление информационных объектов и сервисов Портала пользователям;

- прикладной уровень — система серверов приложений, обеспечивающая выполнение преобразований информационных объектов в соответствии с заданными процедурами и правилами;
- уровень хранения данных, реализующий задачи контроля за целостностью и корректностью структуры информационных объектов.

Принципы интеграции данных и систем. Унификация обмена и представления данных на Портале основывается на:

- принятии формата XML в качестве основного стандарта для интеграции и представления данных в Интернете для всех систем общественного сектора: XML для хранения и транспортировки данных, XSLT для преобразования данных, XML-схемы для интеграции данных;
- принятии веб-браузеров в качестве единственной клиентской программы.

В число технических и организационных задач, которые решаются в ходе разработки Портала, входят:

- разработка способов представления информации, механизмов доступа к ней с целью улучшения восприятия и упрощения освоения системы неподготовленными пользователями;
- стандартизация процедур и протоколов обмена информацией между различными информационными системами, взаимодействующими с Порталом;
- исключение дублирования информационных объектов и программных компонентов в различных пользовательских сервисах Портала;
- организация доступа к территориально распределенным вычислительным ресурсам параллельной архитектуры через единый пользовательский интерфейс.

Таким образом, Портал является основной технологической единицей общей инфраструктуры единой информационной научно-образовательной среды [1]. По сути, Портал — это точка с определенным регламентом доступа к научно-образовательной жизни вуза (информационным ресурсам, научным конференциям, вычислительным экспериментам) независимо от места расположения потенциального пользователя, учебного заведения или места подключения пользователя сети Интернет.

Доступ ко всем подсистемам Портала осуществляется через единую систему безопасности, что позволяет пользователю, зарегистрировавшись в системе, получить доступ ко всем его функциям в рамках имеющихся привилегий.

Целевой аудиторией портала являются студенты, преподаватели, научные сотрудники, организаторы и участники конференций, другие пользователи.

В качестве основных разделов Портала выделяются наиболее значимые, с точки зрения большинства пользователей, аспекты деятельности. По мере развития и реализации новых сервисов и информационных объектов Портала предусматривается возможность создания альтернативных (в т. ч. персонализированных) рубрикаторов, ориентированных на более узкие целевые группы пользователей.

Основные подсистемы Портала

Система удаленного доступа и управления распределенными вычислительными ресурсами — единая точка доступа к набору удаленных кластерных установок [2].

Основные задачи системы:

- хранение пользовательских файлов с исходными и бинарными кодами программ, входными и выходными данными;
- хранение метаинформации о пользовательских файлах, необходимой для организации запуска программ на кластерных установках;
- отслеживание состояния доступных вычислительных ресурсов;
- взаимодействие с вычислительными ресурсами для передачи файлов и команд между ресурсом и хранилищем данных.

Система построена по трехзвенной архитектуре «клиент-сервер-сервер», где клиентским приложением является web-браузер, промежуточным сервером выступает соответствующий компонент портала («Менеджер вычислительных ресурсов»), а конечным сервером — удаленный кластер, где производится запуск программ.

Через портал пользователь может размещать в базе данных свои расчетные программы и задавать для них условия запуска.

Менеджер вычислительных ресурсов просматривает базу данных и выбирает из нее новые задания, которые нужно отправить на удаленные кластеры. При обращении кластера за новым заданием, менеджер отдает ему файлы новой программы и забирает файлы результатов ранее отправленных программ. Для управления кластером на нем устанавливается удаленный агент, который осуществляет связь с менеджером и обрабатывает полученные от него команды по запуску приложений и передаче файлов.

Каталог ресурсов — является хранилищем описаний электронных ресурсов и их полнотекстовых экземпляров. Пользователь может размещать новые ресурсы, выполнять редактирование, удаление ресурсов, передавать данные на экспертную оценку. По результатам экспертной оценки ресурс может быть опубликован (размещен) в каталоге ресурсов для просмотра.

Основные сервисы каталога ресурсов:

Поисковые механизмы — для обеспечения поиска информации на Портале реализованы следующие поисковые механизмы:

- быстрый поиск — поиск по названию объекта;
- поиск по рубрикатору;
- многокритериальный поиск по информационным объектам определенного типа с возможностью задания значений искомых метаданных.

Управление просмотром — выполняет действия по настройке параметров просмотра результатов поиска в каталоге ресурсов.

Результатом поиска является список объектов (описаний ресурсов), который содержит информацию об объекте в зависимости от выбранных параметров просмотра. В системе предусмотрены различные варианты просмотра:

- вывод обязательной информации о ресурсе;
- вывод дополнительной информации о ресурсе;
- вывод полной информации о ресурсе;
- вывод определенного набора полей описания ресурса.

Результатом выполнения управления просмотром будут данные для просмотра, сформированные определенным образом.

Предоставление полнотекстовой, ознакомительной или полнотекстовой-постраничной версии электронного ресурса осуществляется в зависимости от прав доступа пользователя электронной библиотеки.

Сервер конференций — централизованная система для обслуживания и информационного сопровождения научных конференций, обладающая следующими функциями:

- создание и редактирование свойств конференции;
- публикация информационных сообщений (итогов конференции и другой информации);
- единая регистрация участников на сервере конференций;
- просмотр текущего состояния заявки;
- электронная рассылка информации участникам;
- подача заявки на текущую конференцию с возможностью указать тип участия и проживания в гостинице;
- публикация тезисов и статей.

Публикация информационных сообщений и других документов осуществляется посредством регистрации через веб-интерфейс самой конференции, а затем размещения необходимой информации.

Архитектура и технологии Портала

Обмен данными проходит по стандартным сетевым протоколам в среде Intranet / Internet. Такая архитектура позволяет работать с системой из любой точки, имеющей выход в Intranet / Internet, а также обеспечивает независимость от платформы и минимальные требования к клиентам.

Клиент для системы выбирался максимально «тонким», поэтому в роли него выступает стандартный веб-браузер. Такой выбор позволяет минимизировать системные требования к клиентскому месту, а также снизить расходы за счет отсутствия потребности в установке дополнительного программного обеспечения, помимо уже имеющегося стандартного, и обучении пользователей работе с ним.

Посредством веб-браузера пользователь связывается с сервером Портала. Данные хранятся на сервере базы данных. Обмен данными между сервером приложений и сервером базы данных осуществляется по локальной сети (Intranet).

Такая архитектура позволяет работать с системой из любой точки, имеющей выход в Internet, а также обеспечивает независимость от платформы и минимальные требования к клиентам.

Подсистемы Портала располагаются на сервере приложений (Apache Tomcat), предназначенного для реализации технологии Java Servlets.

Вся логика приложений расположена на стороне сервера БД и реализована в виде набора пакетов. Для каждой таблицы, а также для функциональных блоков создаются пакеты для осуществления операций с данными. Тем самым обеспечивается возможность манипуляции необходимыми данными в каждой из подсистем путем вызова хранимых процедур.

Для реализации web-интерфейса используются технологии J2EE и XML:

- формирование выходных данных для отображения в браузере: XSLT;
- формирование списков и отчетов в формате PDF: XSL-FO;
- выходные данные XHTML, CSS.

На сервере приложений Apache Tomcat используется библиотека KemSUWeb, разработанная в центре новых информационных технологий (ЦНИТ) КемГУ [3]. Библиотека KemSUWeb располагается на сервере приложений и обеспечивает единую среду для создания приложений, основанных на трехуровневой архитектуре в среде Internet за счет адаптеров, которые удовлетворяют различные потребности разработчика: в операциях с БД, в защите информации, в управлении ходом приложения.

Реализация ИВП ведется в рамках выполнения проекта № РНП.3.2.3.4256 аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы (2006—2008 гг.)».

Литература

1. **Гудов, А. М.** Система электронного документооборота [Текст] / А. М. Гудов, С. Ю. Завозкин // Тр. VI Всерос. науч.-практ. конф. «Системы автоматизации в образовании, науке и производстве». — Новокузнецк, 2007. — С. 278—281.

2. **Демидов, А. В.** Система удаленного доступа и управления вычислительными ресурсами [Текст] / А. В. Демидов, Е. В. Бусло в, М. А. Поспелов // Гидродинамика больших скоростей

и численное моделирование : мат-лы III Международ. летней школы. — Кемерово : ИНТ, 2006. — С. 365—370.

3. **Гудов, А. М.** Об одной технологии создания web-интерфейса к программному обеспечению [Текст] / А. М. Гудов, Е. А. Ростовцев // Тр. III регион. науч.-практ. конф. «Информационные недра Кузбасса». — Кемерово, 2004. — С. 84—88.

Опыт создания вычислительного метакластера на базе кластерных систем Томского научного центра[1]

А. В. Старченко, Г. М. Ибраев
Томский государственный университет, Томск

В настоящее время основное направление развития высокопроизводительных вычислительных ресурсов связано с разработкой технологий, позволяющих объединять возможности вычислительных комплексов, принадлежащих разным организациям, в том числе, территориально удаленным друг от друга. За этим направлением закрепилось название «Grid-технология». Под Grid-технологией понимают инфраструктуру, делающую возможным совместное использование высокопроизводительных компьютеров, сетей, баз данных и исследовательских систем, принадлежащих различным организациям. Grid-приложения часто предназначены для обработки очень больших распределенных объемов данных, которые должны перемещаться между компонентами Grid-инфраструктуры, что требует обеспечения определенных мер защиты этих данных, что не всегда легко достижимо при использовании распространенных сегодня протоколов сети Интернет [1].

Решение проблем, связанных с использованием Grid-технологий, является предметом нескольких исследовательских проектов, выполняющихся в различных организациях. Одним из наиболее известных является проект Globus Allians [2]. В рамках этого проекта ведется разработка сетевых протоколов и приложений, предназначенных для реализации Grid-технологии в сфере обеспечения вычислительных потребностей фундаментальных и прикладных научных исследований. Работы по данному проекту выполняются совместно в целом ряде организаций США и Западной Европы. Крупными корпоративными партнерами являются компании IBM и Microsoft. Основное внимание разработчиками проекта уделяется вопросам управления вычислительными ресурсами, механизмам доступа и управления данными, созданию средств разработки Grid-приложений,

[1] Работа выполнена при поддержке РФФИ, грант № 07-05-01126, и программы «СКИФ-ГРИД».

информационному обеспечению и безопасности. В результате работ в рамках проекта создан набор библиотек и прикладных программ, предназначенных для реализации Grid-технологии, а также поддержки и разработки Grid-приложений.

В качестве более простого по настройке и применению пакета для организации распределенных параллельных вычислений на метакомпьютере, созданном на основе ресурсов географически удаленных вычислительных кластеров, может быть рассмотрено разработанное в Суперкомпьютерном центре г. Штутгарта специализированное программное обеспечение PACX-MPI (в дальнейшем PACX — PArallel Computer eXtension) [3—5]. С технической точки зрения библиотека PACX является прослойкой между MPI-приложением и нижележащей MPI-библиотекой и может эффективно использоваться для решения сильносвязанных (с большим числом межпроцессорных обменов) MPI-программ на ресурсах удаленных кластерных систем. Программы такого рода получаются обычно при численном решении конечно-разностными методами задач механики сплошных сред или физики атмосферного пограничного слоя.

Целью данной работы является построение многопроцессорной вычислительной установки (метакластера) на базе ресурсов вычислительных кластеров ТГУ и ИОА СО РАН и программного обеспечения PACX-MPI, ее апробация на задаче переноса примеси, поступающей из приподнятого над землей точечного источника, в атмосфере.

1. Библиотека для организации параллельных вычислений PACX-MPI

Предположим, у нас имеется вычислительные кластерные системы с распределенной памятью, которые используются для параллельных вычислений на базе библиотеки MPI, но расчеты на них можно производить лишь по отдельности, т. е. вычислительные ресурсы данных кластеров нельзя использовать совместно для запуска одной параллельной MPI-программы. Для преодоления этой ограниченности в Суперкомпьютерном центре г. Штутгарта была создана библиотека PACX-MPI.

Библиотека PACX обладает следующими преимуществами:

- не требуется изменение кода MPI-приложения, достаточно лишь его откомпилировать с библиотекой PACX;

- осуществляется сжатие данных перед отправкой межкластерных сообщений;
- производится буферизация передаваемых данных;
- имеется поддержка аппаратно-программной гетерогенности кластеров;
- для внешней коммуникации реализована поддержка стандартных протоколов TCP, ATM, SSL;
- для внутрикластерных сообщений используются стандартные (оптимизированные) MPI-вызовы нижележащей MPI библиотеки.

Концептуальная схема взаимодействия кластеров PACX проиллюстрирована на рис. 1 [3]. На каждом кластере для коммуникации с другими кластерами используется два процесса. Один — PACX In-Server — фоновый процесс для отправки исходящих сообщений, другой — PACX Out-Server — фоновый процесс для приема входящих сообщений. PACX In-Server каждого кластера устанавливает связь с PACX Out-Server'ами остальных кластеров, которые планируется объединить в метакластер.

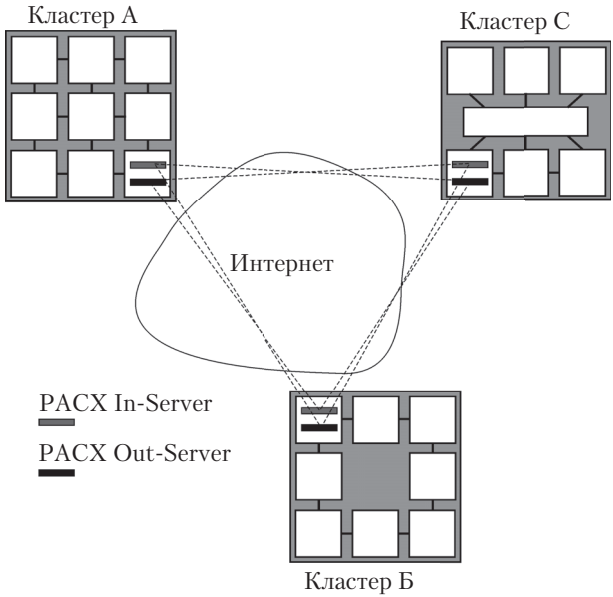


Рис. 1. Концептуальная схема взаимодействия кластеров PACX-MPI

В библиотеке PACX для организации межкластерных сообщений каждый вычислительный узел имеет два номера: локальный номер и глобальный номер. Глобальный номер идентифицирует узел как узел создаваемого метакомпьютера, а локальный номер является внутреннем идентификатором в соответствующем кластере — компоненте метакомпьютера. Для понимания необходимости использования двойной нумерации рассмотрим пример передачи сообщения типа «точка-точка» из глобального узла 2 в глобальный узел 7 на рис. 2.

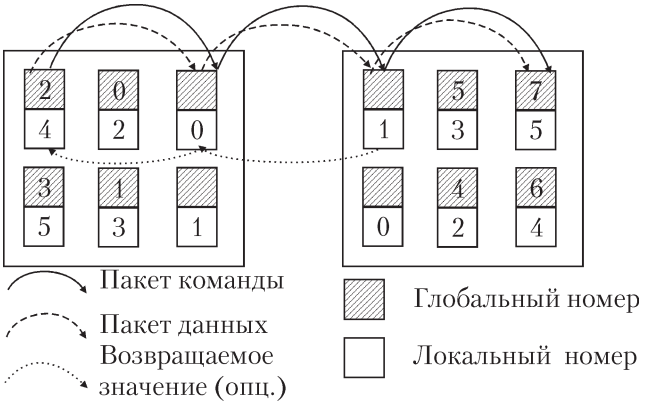


Рис. 2. Пример передачи сообщения типа «точка-точка»

Как видно на рис. 2, каждый кластер для коммуникации использует два *выделенных коммуникационных узла*. Эти коммуникационные узлы прозрачны для пользователя и, соответственно, они не имеют глобальных номеров. При отправке сообщения узел с глобальным номером 2 обнаруживает, что его адресат находится на втором кластере, поэтому он передает сообщение своему коммуникационному узлу (процесс PACX Out-Server). Коммуникационный узел первого кластера, в свою очередь, упаковывает сообщение и передает его коммуникационному узлу второго кластера (процесс PACX In-Server). Коммуникационный узел второго кластера распаковывает сообщение, сопоставляет глобальный номер узла (7) с локальным номером (5), затем передает сообщение с помощью MPI-вызова нижележащей библиотеки MPI узлу 5.

2. Задача о переносе примеси и численный метод ее решения

Предполагается, что в определенный момент времени из приподнятого на высоте h над поверхностью точечного источника начался выброс примеси (рис. 3), и необходимо рассчитать распределение концентрации примеси в различных точках рассматриваемой области через некоторое время.

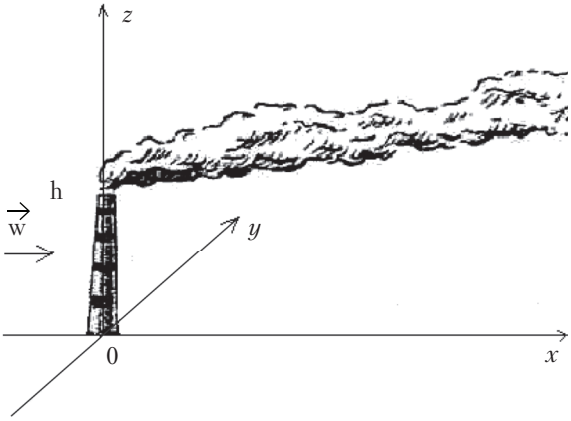


Рис. 3. Система координат

Известными являются локальные метеорологические параметры в приземном слое атмосферы (компоненты вектора скорости ветра и коэффициенты турбулентной диффузии) и начальное распределение концентрации примеси в рассматриваемой области. Интенсивность выброса примеси является переменной величиной и зависит от времени t.

2.1. Математическая постановка задачи

Математически процесс переноса примеси от точечного источника описывается краевой задачей для адвективно-диффузионного уравнения следующего вида [6]:

$$\frac{\partial C}{\partial t}+u\frac{\partial C}{\partial x}+v\frac{\partial C}{\partial y}+w\frac{\partial C}{\partial z}= \\ =\frac{\partial}{\partial x}\left(D\frac{\partial C}{\partial x}\right)+\frac{\partial}{\partial y}\left(D\frac{\partial C}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial C}{\partial z}\right)+S(t,x,y,z) \qquad (1)$$

где u, v, w — проекции вектора скорости ветра на оси OX, OY, OZ соответственно; D, K_z — коэффициенты горизонтальной и вертикальной турбулентной диффузии; $S = f(t)\delta(x)\delta(y)\delta(z-h)$ — источник примеси, который располагается в точке с координатами $(0;0;h)$; $C = C(t, x, y, z)$ — концентрация примеси.

В начальный момент времени известно распределение концентрации примеси $C_0(x,y,z)$ в рассматриваемой области $\left\{-L_x/2 \dots L_x/2; -L_y/2 \dots L_y/2; 0 \dots L_z\right\}$, а на границах выполняются следующие граничные условия:

$$\frac{\partial C}{\partial x} = 0 \text{ при } x = \frac{-L_x}{2}, \ \frac{\partial C}{\partial x} = 0 \text{ при } x = \frac{L_x}{2};$$

$$\frac{\partial C}{\partial y} = 0 \text{ при } y = \frac{-L_y}{2}, \ \frac{\partial C}{\partial y} = 0 \text{ при } y = \frac{L_y}{2}; \qquad (2)$$

$$\frac{\partial C}{\partial z} = 0 \text{ при } z = 0, \ \frac{\partial C}{\partial z} = 0 \text{ при } z = L_z,$$

где L_x, L_y, L_z — размеры области исследования по осям OX, OY, OZ соответственно.

Решение данной краевой задачи состоит в нахождении концентрации примеси в рассматриваемой области $\left\{0 \dots L_x; 0 \dots L_y; 0 \dots L_z\right\}$ через требуемый промежуток времени T. Аналитическое решение данного дифференциального уравнения может быть получено только в простейших случаях [7]. Поэтому для решения таких задач в основном используются численные методы решения [8].

2.2. Метод решения

Для решения поставленной задачи используется метод конечных разностей или метод сеток [8]. Суть данного метода заключается в следующем. Область непрерывного изменения аргументов заменяется конечным множеством дискретных узлов (сеткой). Вводятся сеточные функции, определяемые значениями функций непрерывных аргументов в узлах сетки. Входящие в уравнение частные производные заменяются *разностными соотношениями*. После такой замены решение задачи сводится к решению системы разностных алгебраических уравнений.

Определим конечно-разностную сетку следующим образом:

$$\varpi_h = \left\{\left(t^n, x_i, y_j, z_k\right)\right\},$$

где

$$t^n = n \cdot \tau,\ n = 0 \ldots N_T,\ \tau = T / N_T$$

$$x_i = i \cdot h_x,\ i = 0 \ldots N_x,\ h_x = L_x / N_x$$

$$y_j = j \cdot h_y,\ j = 0 \ldots N_y,\ h_y = L_y / N_y$$

$$z_k = k \cdot h_z,\ k = 0 \ldots N_z,\ h_z = L_z / N_z$$

Таким образом, будем искать приближенное решение задачи (1)—(2) в узлах конечной разностной сетки $C^n_{i,j,k} \approx C\left(t^n, x_i, y_j, z_k\right)$, где $n = 0...N_T$, $i = 0...N_x$, $j = 0...N_y$, $k = 0...N_z$. N_T, N_x, N_y, N_z — количество разбиений по времени и по осям OX, OY, OZ соответственно.

При замене производных по x, y, z, входящих в уравнение (1), во внутренних узлах сетки воспользуемся центрально-разностными аппроксимациями и правыми разностями — для производной по времени. Граничные условия (2) будем аппроксимировать с помощью разностных соотношений первого порядка точности.

В результате получим явную разностную схему следующего вида (для случая постоянных коэффициентов):

$$\begin{cases} \dfrac{C^{n+1}_{i,j,k} - C^n_{i,j,k}}{\tau} + u\dfrac{C^n_{i+1,j,k} - C^n_{i-1,j,k}}{2h_x} + v\dfrac{C^n_{i,j+1,k} - C^n_{i,j-1,k}}{2h_y} + w\dfrac{C^n_{i,j,k+1} - C^n_{i,j,k-1}}{2h_z} = \\ = D\left(\dfrac{C^n_{i+1,j,k} + C^n_{i-1,j,k} - 2C^n_{i,j,k}}{h_x^{\,2}} + \dfrac{C^n_{i,j+1,k} + C^n_{i,j-1,k} - 2C^n_{i,j,k}}{h_y^{\,2}}\right) + K_z\dfrac{C^n_{i,j,k+1} + C^n_{i,j,k-1} - 2C^n_{i,j,k}}{h_z^{\,2}} + \\ + S^n_{i,j,k}, \quad i = 1,...,N_x - 1, j = 1,...,N_y - 1, k = 1,...,N_z - 1, n > 0; \\ C^0_{i,j,k} = C0(x_i, y_j, z_k), i = 0,...,N_x, j = 0,...,N_y, k = 0,...,N_z; \\ C^{n+1}_{0,j,k} = C^{n+1}_{1,j,k}; C^{n+1}_{N_x,j,k} = C^{n+1}_{N_x-1,j,k}; j = 1,...,N_y - 1, k = 1,...,N_z - 1, n = 0,...,N_T - 1; \\ C^{n+1}_{i,0,k} = C^{n+1}_{i,1,k}; C^{n+1}_{i,N_y,k} = C^{n+1}_{i,N_y-1,k}; i = 1,...,N_x - 1, k = 1,...,N_z - 1, n = 0,...,N_T - 1; \\ C^{n+1}_{i,j,0} = C^{n+1}_{i,j,1}; C^{n+1}_{i,j,N_z} = C^{n+1}_{i,j,N_z-1}; i = 1,...,N_x - 1, j = 1,...,N_y - 1, n = 0,...,N_T - 1; \end{cases}$$

Эта разностная схема имеет первый порядок аппроксимации по времени и координатам. Она является условно устойчивой и монотонной при выполнении неравенств

$$\tau < \frac{1}{2D\left(\frac{1}{h_x^{\,2}} + \frac{1}{h_y^{\,2}}\right) + \frac{2K_z}{h_z^{\,2}}},\ h_x < \frac{2D}{u}, h_y < \frac{2D}{v}, h_z < \frac{2K_z}{w}.$$

3. Параллельная реализация численного метода и результаты

Для распараллеливания рассматриваемой задачи существуют различные способы декомпозиции вычислений по активным процессам. Одним из них является декомпозиция по данным (data decomposition) [9]. Суть данного способа декомпозиции заключается в равномерном распределении данных для однородных вычислений по имеющимся процессам.

При решении задачи методом конечных разностей на каждом шаге по времени необходимо приближенно рассчитать значения $C(t, x, y, z)$ в точках сетки ϖ_h, покрывающей рассматриваемую область, т. е. фактически необходимо найти $N_x \cdot N_y \cdot N_z$значений. При декомпозиции необходимо учесть, что при вычислении значения $C_{i,j,k}^{n+1}$ используются значения сеточной функции в соседних по осям Ox, Oy, Oz узлах сетки, полученные на предыдущем шаге времени.

Известно, что время передачи одного числа между вычислительными узлами на современных вычислительных кластерах с распределенной памятью в среднем на два-три порядка превышает время одной арифметической операции. Поэтому при декомпозиции расчетной области необходимо минимизировать количество узлов сеточного шаблона, принадлежащих различным процессам. С учетом этого требования ниже рассмотрены следующие способы декомпозиции расчетной области: одномерная и двухмерная.

3.1. Распределение обрабатываемых данных между вычислительными узлами

В данной работе разбиение при одномерной декомпозиции производится по одному из координатных направлений — оси OY. Если визуально всю расчетную область представить в виде параллелепипеда, то в результате одномерной декомпозиции параллелепипед по одной координатной оси разбивается на p

равных параллелепипедов (рис. 4), где p — число активированных копий параллельной MPI-программы.

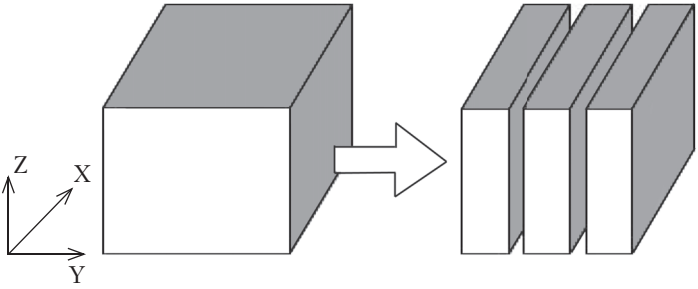


Рис. 4. Одномерная декомпозиция расчетной области

В результате разбиения на соприкасающихся границах полученных параллелепипедов для вычисления значений концентраций по выбранному шаблону одно требуемое значение попадает на соседний параллелепипед. Поэтому на каждом шаге соседним процессам необходимо обмениваться значениями для обеспечения следующего $n+1$-го шага вычисления. На рис. 5 изображен обмен результатами вычислений между процессами при одномерной декомпозиции расчетной области. Показаны узлы сетки, в которых вычисляются значения концентрации на каждом процессе. Более темным цветом выделены граничные значения концентрации, которые передаются на соседний процесс на каждом шаге по времени.

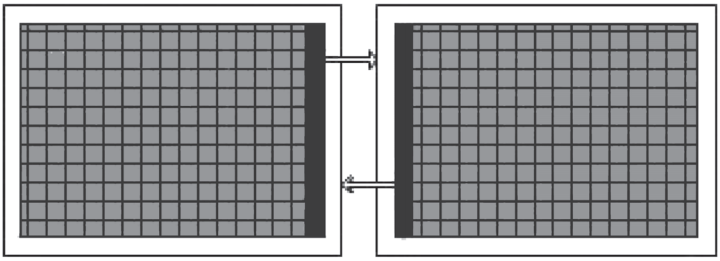


Рис. 5. Обмен результатами вычислений между соседними процессами

Разбиение при двухмерной декомпозиции производится по двум направлениям (например, по осям OY, OZ). В отличие от одномерной декомпозиции разбиение производится сначала

в направлении оси OY, зачем получившиеся параллелепипеды разбиваются на такое же количество областей в другом направлении — оси OZ (рис. 6).

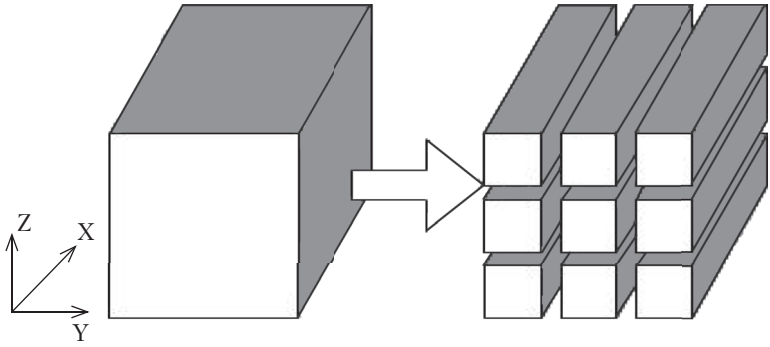


Рис. 6. Двухмерная декомпозиция расчетной области

В результате такого разбиения каждый получившийся параллелепипед граничит с не более чем с четырьмя соседними параллелепипедами. На рис. 7 изображен обмен результатами вычислений при двухмерной декомпозиции расчетной области для разбиения $p = 4$. Ограничение данной декомпозиции заключается в том, что число разбиений должно быть равно квадрату целого числа, т. е. $p = 1, 4, 9, 16, 25, ...$

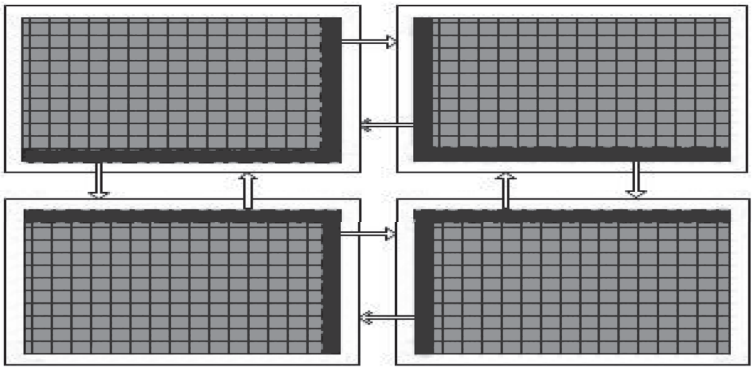


Рис. 7. Обмен результатами вычислений
с соседними процессами для разбиения $p = 4$

При одномерной и двухмерной декомпозиции задачи переноса примеси на эффективность параллельной программы значительное влияние оказывает способ передачи сообщений между процессами. В соответствии с [10] наиболее приемлемым способом при одномерной и двухмерной декомпозиции является передача и прием сообщений типа «точка-точка».

Библиотека MPI предоставляет широкий набор функций для передачи и приема сообщений типа «точка-точка» [10]. В одномерной и двухмерной декомпозиции использованы следующие способы передачи и приема сообщений:

MPI_SendRecv — блокирующая операция приема-передачи объединяет в едином вызове передачу сообщения одному процессу и прием сообщения от того же процесса;

Неблокирующие прием и передача состоят из двух этапов: инициализация передачи (MPI_Isend) и приема (MPI_Irecv); блокирующая проверка завершения обмена (MPI_Wait).

Первый метод передачи и приема сообщений выгоден с точки зрения используемой памяти, но влечет за собой простаивание процессоров. В свою очередь если кластерная сеть высокоскоростная, то это простаивание процессов может оказывать незначительное влияние на общее время вычислений. Второй метод приема и передачи является полной противоположностью, вероятность простаивания процессов очень мала, но свою очередь требуется больше объема оперативной памяти. В итоге выбор использования того или иного способа во многом обусловлен спецификой задачи, которую во многих случаях трудно определить без проведения экспериментальных опытов на многопроцессорной системе.

3.2. Теоретическая оценка ускорения и эффективности

Для оценки производительности параллельной реализации решаемой задачи применяются следующие характеристики: *ускорение* и *эффективность* [11]. Ускорение на p процессорах, полученное при решении задачи с размерностью N (для удобства проведения оценок принимаем, что $N_x = N_y = N_z = N-1$), рассчитывается по следующей формуле:

$$S(N,p) = \frac{T(N,1)}{T(N,p)}$$

т. е. через отношение времени решения задачи на одном процессоре ко времени ее решения на p-процессорной системе.

Для рассматриваемых подходов декомпозиции общее время выполнения параллельной программы состоит из времени, затрачиваемого на арифметические операции, и времени, требуемого на обмен значениями между процессами. Предполагается, что процессоры обладают одинаковой вычислительной производительностью и вычислительная нагрузка сбалансирована. Поэтому общее время на арифметические операции при использовании одномерной декомпозиции $\left(T_a^{1D}\right)$ обратно пропорционально количеству активных процессов. Если процессы осуществляют обмен данными одновременно, то общее время на обмен значениями $\left(T_o^{1D}\right)$ не будет зависеть от количества процессов. В итоге

$$T^{1D} = T_a^{1D} + T_o^{1D} \approx t_a \frac{N^3}{p} + 2N^2 t_o$$

где t_a — время выполнения арифметических операций для вычисления значения C_{ijk}^{n+1}, t_o — время передачи одного значения между процессами, p — количество процессов

Таким образом, теоретическая оценка ускорения для одномерной декомпозиции:

$$S(N,p) = \frac{t_a N^3}{t_a \frac{N^3}{p} + 2N^2 t_o} = \frac{p}{1 + 2\frac{t_o}{t_a}\frac{p}{N}} \qquad (3)$$

Для анализа ускорения параллельного алгоритма в случае одномерной декомпозиции были проведены серии экспериментов для $p = 1 \ldots 20$, $N = 189$ ($189 \times 189 \times 189$) на кластере ИОА СО РАН (рис. 8).

Использование блокирующих вызов при одномерной декомпозиции дает лучшее ускорение, чем неблокирующих. Это объясняется тем, что внутренние механизмы управления посылкой асинхронных сообщений не совсем эффективны, когда размер сообщения длинный (при $N = 189$) ~ 300 Kb.

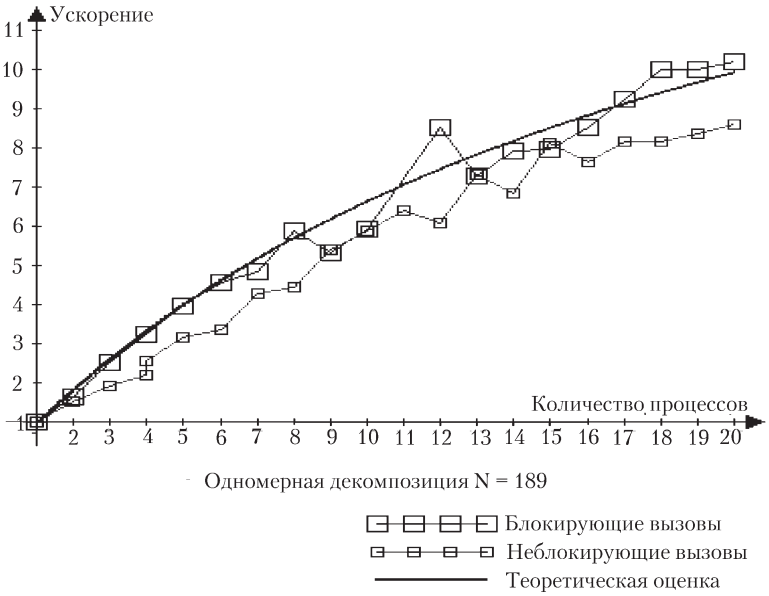


Рис. 8. Ускорение параллельной программы при одномерной декомпозиции

При двухмерной декомпозиции общее время выполнения параллельной программы состоит из времени, затраченного на арифметические операции, и времени на обмен значениями между процессами по двум координатным направлениям (OY, OZ):

$$T^{2D} = T_a^{2D} + T_o^{2D} = t_a \frac{N^3}{p} + 4\frac{N^2}{\sqrt{p}} t_o$$

Тогда теоретическая оценка ускорения для двумерной декомпозиции будет иметь вид:

$$S(N,p) = \frac{t_a N^3}{t_a \dfrac{N^3}{p} + 4t_o \dfrac{N^2}{\sqrt{p}}} = \frac{p}{1 + 4\dfrac{t_o}{t_a}\dfrac{\sqrt{p}}{N}} \qquad (4)$$

Для анализа ускорения параллельного алгоритма в случае двухмерной декомпозиции были проведены серии экспериментов для $p = 1, 4, 9, 16$, $N = 189$ $(189 \times 189 \times 189)$ на кластере ИОА СО РАН (рис. 9).

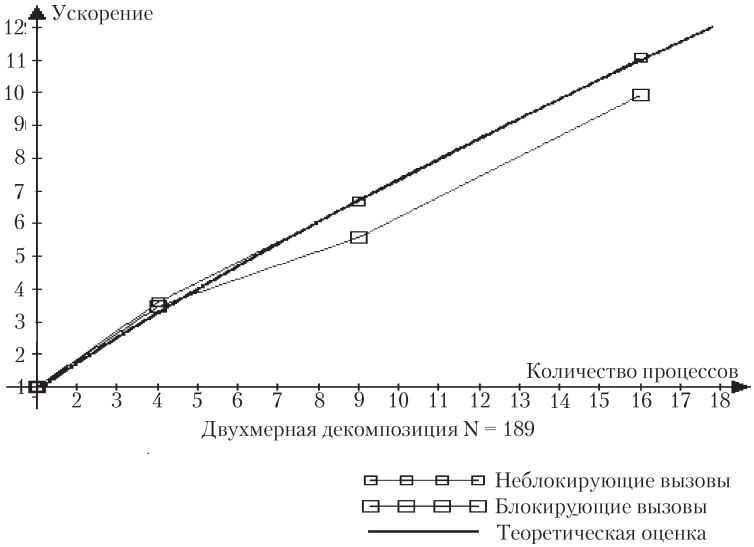


Рис. 9. Ускорение параллельной программы при двухмерной декомпозиции

Как видно, в случае двухмерной декомпозиции по мере уменьшения размера сообщения (при 16 процессах размер сообщения ~75 Kb) неблокирующие вызовы дают лучшее ускорение, чем синхронные вызовы. Таким образом, асинхронные вызовы уместнее использовать, когда межпроцессорные сообщения достаточно небольшого размера.

В результате проведенных экспериментов показано, что двухмерная декомпозиция является более эффективной по сравнению с одномерной. На шестнадцати процессорах двухмерная декомпозиция дает такое же ускорение, которое при одномерной декомпозиции достигается только на двадцати процессорах.

4. Тестирование метакомпьютера на примере задачи переноса примеси

Тестирование сконфигурированной многопроцессорной системы проводилось на примере задачи переноса примеси для одномерной декомпозиции с использованием блокирующего вызова (MPI_SendRecv) при размерности $N = 200$. Запуск MPI-приложения с высокой долей обменных операций осуществлялся на

10 процессорах метакомпьютера, состоящего из вычислительных узлов кластеров ТГУ и ИОА СО РАН (см. табл. 1). В качестве канала связи между кластерами рассматривались общественный Интернет г. Томска и выделенная оптоволоконная линия.

Таблица 1

Технические характеристики кластеров ТГУ и ИОА СО РАН

	Кластер ИОА СО РАН	**Кластер ТГУ**
Число узлов	10	9
Конфигурация узла	2 ЦП Pentium 3 1 GHz, RAM 1 Gb, бездисковый	2 ЦП Pentium 3 650 MHz, RAM 256 Mb, бездисковый
Конфигурация сервера	2 ЦП Pentium 3 1 Ghz, RAM 1 Gb, 3 SCSI HDD 18Gb	2 ЦП Pentium 3 650 MHz, RAM 512Mb, 36 Gb
Коммуникационное оборудование	Gigabit Ethernet на коммутаторе 3Com SuperStack 4900	100 Mbit Ethernet на коммутаторе Cisco Catalyst 2924
Программное обеспечение	Linux Kernel 2.4.28 , MPICH 1.2.4, Intel C++ & Fortran Compiler 7.1	Linux Kernel 2.6.9-22, MPICH 1.2.6, Intel C++ & Fortran Compiler 9.0
Реальная производительность на тесте LINPACK	10.5 GFlops	5.5 GFlops
Пиковая производительность	20 GFlops	11,7 GFlops

Гетерогенность метакомпьютера, построенного на базе разнородных кластеров, можно характеризовать 3 видами сообщений: внутрикластерные сообщения кластера ТГУ, внутрикластерные сообщения кластера ИОА СО РАН, межкластерные сообщения. Эффективность передачи таких сообщений зависит от латентности и пропускной способностью коммуникационной среды, по которой они передаются.

В случае использования канала связи «Интернет» (см. табл. 2) скорость передачи межкластерных и внутрикластерных сообщений имеет существенные различия, что не дает никакого преимущества для использования метакомпьютера для сильносвязанных MPI-приложений с интенсивными блокирующими сообщениями. В этом случае актуальным является использование метакомпьютера для задач пакетной обработки.

Таблица 2

Результаты тестирования метакомпьютера

Суммарная пиковая мощность в GFlops	Количество ЦП (CPU)		Время работы в минутах			
	Кластер ТГУ	Кластер ИОА СО РАН	MPI запуск	PACX-MPI запуск	Совместный PACX-MPI запуск, канал связи Интернет	Совместный PACX-MPI запуск, канал связи Оптоволокно
6,5	10		39,45	40,25		
10		10	18,43	19,53		
7,5	7	3			120,39	34,11
8,95	3	7			79,32	31,10
8,25	5	5			93,58	32,19

В случае использования выделенного канала связи «Оптоволокно» конфигурация (7,3) имеет примерно такую же производительность, что и конфигурация (10,0). Причем, возрастание числа более производительных процессоров кластера ИОА СО РАН в составе вычислительных узлов метакомпьютера все в меньшей степени влияет на увеличение производительности метакомпьютера.

Для оптимального использования метакомпьютера при решении сильносвязанных параллельных задач в его конфигурацию необходимо включать в большом количестве узлы менее производительного кластера, чем узлы более производительного кластера. Кроме того, поскольку на каждом кластере необходимо выделять два коммуникационных узла, то суммарная пиковая производительность полученного метакомпьютера может составить 28,4 GFlops. Это на 8,4 GFlops больше, чем пиковая производительность кластера ИОА СО РАН.

Заключение

Для запуска сильносвязанных параллельных MPI-программ построен метакомпьютер на базе двух вычислительных кластеров Томского научного центра (ТГУ и Института оптики атмосферы СО РАН). В качестве программной инфраструктуры, создающей единый высокопроизводительный вычислительный ресурс, использовалось программное обеспечение PACX-MPI, разработанное в Суперкомпьютерном центре г. Штутгарта. Этот пакет был выбран в силу относительной простоты в установке, настройке и конфигурировании. Кроме того, он позволяет без какой-либо дополнительной доработки использовать созданные для вычислительных кластеров с распределенной памятью MPI-программы.

Для тестирования созданного метакомпьютера была разработана программа решения трехмерного нестационарного адвективно-диффузионного уравнения конечно-разностными методами с использованием явных разностных схем; получена параллельная реализация этого метода с использованием одномерной и двумерной декомпозиции; исследовано влияние способа организации межпроцессорных обменов (блокирующих и неблокирующих) на основе стандарта MPI; выполнены теоретические оценки ускорения разработанной параллельной программы, сделано сравнение с результатами вычислительных экспериментов на кластере ИОА СО РАН.

На основе разработанной параллельной программы для переноса примеси с использованием блокирующих межпроцессорных обменов (MPI_SendRecv) исследовано влияние конфигурации гетерогенного метакомпьютера на его реальную производительность; показана перспективность рассматриваемого способа объединения кластеров с точки зрения увеличения производительности; продемонстрированы повышенные требования к высокоскоростной сети, связывающей компоненты метакомпьютера.

Литература

1. **Эндрюс Г. Р.** Основы многопоточного, параллельного и распределенного программирования. — М. : Вильямс, 2003. — 512 с.

2. [Электронный ресурс]. — Режим доступа: http://www.globus.org , свободный.

3. **Keller R., Muller M.** PACX-MPI Home Page [Электронный ресурс]. — Режим доступа: http://www.hlrs.de/organization/amt/projects/pacx-mpi , свободный.

4. **Gabriel E., Resch M., Beizel T., Keller R.** Distributed Computing in Heterogeneous Computing Environment EuroPVMM-PI'98 Liverpool/UK, 1998 [Электронный ресурс]. — Режим доступа: http://www.hlrs.de/organization/amt/projects/pacx-mpi.

5. **Trams M.** Feasibility of PACX-MPI for use in a Cluster of Clusters Environment [Электронный ресурс]. — Режим доступа: http://www.hlrs.de/organization/amt/projects/pacx-mpi.

6. **Старченко А. В., Беликов Д. А.** Численная модель для оперативного контроля уровня загрязнения городского воздуха // Оптика атмосферы и океана. — 2003. — Т. 16. — № 7. — С. 655—667.

7. **Берлянд М. Е.** Прогноз и регулирование загрязнения атмосферы. — Л. : Гидрометиздат, 1985. — 272 с.

8. **Самарский А. А.** Введение в численные методы. — СПб. : Лань, 2005. — 228 с.

9. **Minty E., Davey R., Simpson A., Henty D.** Decomposing the Potentially Parallel — The Edinburgh Parallel Computing Centre [Электронный ресурс]. — Режим доступа : http://www.epcc.ed.ac.uk.

10. **Немнюгин С. А., Стесик О. Л.** Параллельное программирование для многопроцессорных вычислительных систем. — СПб. : БХВ-Петербург, 2002. — 400 с.

11. **Воеводин В. В.** Распределенная обработка данных // Вторая Сибирская школа-семинар по параллельным вычислениям. — Томск: Изд-во Том. ун-та, 2004. — С. 3—9.

Новые ингибиторы тромбина: молекулярный дизайн с использованием суперкомпьютеров и экспериментальное подтверждение активности

А. Н. Романов[1], В. Б. Сулимов[1], О. А. Кондакова[1], А. В. Сулимов[1], Ф. В. Григорьев[1], Е. И. Синауридзе[2], Ю. В. Кузнецов[3], В. В. Воеводин[1], Ф. И. Атауллаханов[2]

[1]Научно-исследовательский вычислительный центр МГУ, Москва,
[2]Гематологический научный центр РАМН, Москва,
[3]Институт органической химии им. Зелинского РАН, Москва

В настоящее время становится общепризнанным, что современные лекарственные средства должны избирательно воздействовать на определенные белки в организме человека, отвечающие за то или иное заболевание, не затрагивая остальные органы и ткани. Блокировать работу поврежденного белка можно путем заполнения его активного центра малой органической молекулой — ингибитором. Таким образом, начальный этап создания нового лекарства заключается в нахождении подходящего ингибитора для активного центра выбранного белка-мишени.

Значительные достижения в области изучения строения генов в последние десятилетия привели к экспоненциальному росту числа потенциальных терапевтических белков-мишеней, доступных для исследования. В то же время в фармацевтической индустрии значительно возросли временные и финансовые затраты для того, чтобы пройти путь от нахождения соединения-лидера (хорошего активного ингибитора) до официального одобрения лекарства и вывода его на рынок. При этом основные затраты приходятся на начальную стадию разработки лекарства — определение соединения-лидера. Поэтому основные усилия компаний, работающих в данной области, направлены на сокращение времени создания потенциальных кандидатов в лекарство и увеличение их количества. Значительную помощь в решении данных проблем может оказать использование компьютеров для поиска соединений-лидеров. Начиная с конца 90-х гг. прошлого века, отбор потенциальных кандидатов с помощью компьютерных программ или «виртуальный скрининг» является реальной альтернативой традиционным методам поиска,

принятым в фармацевтической промышленности. В ходе процедуры виртуального скрининга химическое соединение (лиганд) позиционируется в активном центре протеина. Позиционирование осуществляется путем поиска глобального минимума на сложной энергетической поверхности протеина. Учитывая большое количество внутренних степеней свободы лиганда, решение задачи глобальной минимизации становится нетривиальной проблемой. Даже применение эвристических стратегий поиска не позволяет сократить количество вычислений ниже 10^8—10^9. Процедуру виртуального скрининга можно ускорить, применяя технологии распределенных вычислений на суперкомпьютерах любой архитектуры.

Существует целый ряд обзорных статей, посвященных подробному описанию методологии компьютерного моделирования лекарственных средств [1—4]. Поэтому в данной статье хотелось бы только привести краткое описание процедуры виртуального скрининга. Данная процедура состоит из следующих этапов:

1. Выбор белка-мишени и исходного набора органических соединений.

2. Подготовка исходных данных для последующей компьютерной обработки.

3. Позиционирование набора соединений в белке-мишени (процедура докинга).

4. Ранжирование соединений по значению оценочной функции (процедура скоринга).

5. Последующая обработка полученных результатов.

6. Отбор ряда соединений для синтеза и экспериментальной проверки.

Такая схема является в настоящее время традиционной и в зависимости от разных исследовательских групп может различаться только используемыми программами и способами подготовки и обработки данных.

Подготовка исходных данных

Начальным этапом процесса виртуального скрининга является стадия выбора и подготовки исходных данных: белков-мишеней и набора органических соединений, среди которых будет проводиться поиск. В качестве исходного набора органических

веществ могут выступать коллекции физически доступных соединений, которые могут быть заказаны и получены как от компаний, занимающихся продажей химических веществ, так и от некоммерческих организаций. Наиболее подробный список доступных баз данных такого рода можно найти в информационной системе MDL (http://www.mdli.com). Примером такой коллекции может служить база данных Национального института рака США NCI diversity [5]. Другой тип — это так называемые виртуальные библиотеки, где соединения конструируются из определенных элементарных блоков. Такие соединения могут впоследствии быть синтезированы, поскольку элементарные блоки, из которых они создаются, являются реагентами или веществами, содержащимися в базах данных химических компаний.

Основной ресурс, в котором содержаться все общедоступные кристаллические структуры белков — это Protein Data Bank [6] (www.rcsb.org/pdb). Данный ресурс содержит информацию о том, кто и когда получил и расшифровал структуру данного белка, когда данные были впервые опубликованы, разрешение с которым определялась кристаллическая структура и параметры эксперимента по рентгеновской дифракции, классификация белка и описание молекулярной структуры и биологических функций. Также содержаться файлы с координатами структуры белка и лиганда, химические компоненты, находящиеся в активном центре, изображение комплекса, а также масса другой информации, полезной при изучении данной мишени. В настоящее время в базе данных PDB содержится более 40 000 кристаллических структур комплексов протеин-лиганд.

Для использования исходного набора соединений в последующей процедуре докинга необходима предварительная подготовка входящих в этот набор структур. Прежде всего, необходимо сгенерировать трехмерные структуры молекул, включая все их возможные таутомерные формы. Для генерации трехмерных структур можно использовать такие программы, как например Corina [7]. Помимо этого молекула должна находиться в зарядовом состоянии, соответствующем степени кислотности среды, в которой проводиться докинг. Это достигается удалением или добавлением протона в определенных группах молекулы.

Белки-мишени также требуют определенной подготовки, которая во многом зависит от используемой программы докинга.

Во многих случаях, необходимо расставить атомы водорода на аминокислотных остатках и тщательно проверить зарядовые состояния остатков, расположенных в активном центре протеина.

Процедура докинга

Одна из основных трудностей при проведении процедуры докинга — это учет внутренних степеней свободы лиганда и протеина. Существующие на сегодняшний день программы докинга можно разделить на три группы в зависимости от того, каким образом в них учитываются внутренние степени свободы лигандов.

К первой группе относятся программы, использующие систематические методы: метод малых приращений, конформационный поиск и использование баз данных. В основе систематических методов лежит попытка перебора всех степеней свободы молекулы, что зачастую приводит к проблеме комбинаторного взрыва, поэтому были разработаны методики постепенного построения (приращения) лиганда внутри активного сайта. Одна из наиболее распространенных методик — это разделение лиганда на жесткую часть, которая позиционируется в активном центре, и гибкие фрагменты, которые постепенно пристраиваются к данному ядру. По данному принципу построены программы DOCK, FlexX, Glide [8—10].

Другой метод — это использование библиотек заранее рассчитанных конформаций. В таком случае проблема сводиться к процедуре докинга заранее оптимизированного и жесткого лиганда в активный центр. Данная методика используется в такой программе как FLOG [11].

Вторая группа — методы случайного поиска или стохастические методы. К ним относятся моделирование методом Монте-Карло, генетический алгоритм и табу поиск. В данных алгоритмах случайным образом меняется конформация лиганда или группы лигандов, а затем оценивается энергия взаимодействия получившейся структуры с активным центром протеина. Наиболее распространенными в данной группе являются метод Монте-Карло и генетический алгоритм. Они используются в программах AutoDock, DOCK, Gold [12—15]. Несколько отличную идеологию имеет метод табу поиска. Основная идея данного

алгоритма заключается в анализе исследованного ранее конформационного пространства группы лигандов. Критерием отбора нужной структуры служит величина среднеквадратичного отклонения вновь определенных значений координат данного соединения от координат этого же вещества, полученных ранее. Одним из примеров реализации данного алгоритма служит программа PRO_LEADS [16].

И наконец, третья группа — это прямые методы моделирования: молекулярная динамика и минимизация энергии. Молекулярная динамика — это наиболее популярный метод моделирования, но зачастую молекула может не преодолеть энергетический барьер за разумное время расчета и лиганд будет позиционирован в локальном минимуме.

Процедура скоринга

Оценка энергии взаимодействия белок-лиганд и ранжирование предсказанных конформаций лигандов (скоринг) — это ключевой аспект виртуального скрининга. Скоринг должен отделять «настоящие» лиганды от «ложно позитивных», т. е. лигандов, которые, несмотря на значение оценочной функции, предсказывающее их активность, не будут проявлять таковой в эксперименте. Также как и для докинга существует три типа оценочных (скоринг) функций.

Первый тип — это физические или основанные на силовых полях функции (force-field based). Такого рода функции определяют сумму двух энергий: энергии взаимодействия протеин-лиганд и внутренней энергии лиганда. Наиболее часто взаимодействие между лигандом и протеином описывается только Ван-дер-Ваальсовым и электростатическим членами, пренебрегая энтропийными эффектами. Кроме того, возникает проблема учета эффектов связанных с участием растворителя в протеин-лиганд взаимодействии. Наиболее известные силовые поля, на которых строятся скоринг-функции это Tripos, Amber и MMFF94 [17—19].

Второй тип — эмпирические скоринг-функции, которые параметризуются таким образом, чтобы адекватно воспроизводить экспериментальные результаты, например, энергии связывания протеин-лиганд или энергии конформаций лигандов. Функцио-

нальная форма таких функций обычно проще, чем у физических, и состоит из суммы некоррелирующих членов. Такой тип скоринг-функции используется, например, в программа LUDI [20].

Третий тип — функции, основанные на базах знаний (knowledge-based). Идеология данного типа функций сходна с предыдущим типом, но в данном случае делается попытка воспроизвести экспериментальные структуры, а не энергии взаимодействия. Подобные функции включены в программы PMF и DrugScore [21, 22].

Обработка результатов и отбор соединений

В настоящее время на этом этапе компьютерные методы играют лишь вспомогательную роль, а основная нагрузка по-прежнему ложится на самих исследователей. Существует ряд программно реализованных физических фильтров, таких как оценка лекарственного подобия, токсичности, растворимости, которые могут быть использованы в ходе последующего отбора соединений. Также можно применять геометрические и статистические критерии для выбора наиболее перспективных лигандов. Однако, чаще всего, исследователю приходиться ориентироваться на свою химическую интуицию или имеющиеся знания, чтобы отобрать для последующего синтеза вещества, которые со значительной долей вероятности проявят свои ингибирующие качества в экспериментах.

Система молекулярного моделирования KeenBASE

Система молекулярного моделирования KeenBASE была разработа для того, чтобы помогать разработчикам новых лекарств находить новые ингибиторы для заданных белков-мишеней с помощью компьютерного моделирования **in silico**. При этом применение этой системы не ограничивается только областью фармацевтики, но относится также к сельскому хозяйству (разработке гербицидов), ветеринарии и экологии. Общая схема работы программного комплекса KeenBASE приведена на рис. 1. Легкость в перенастройках создаваемой системы позволяет быстро переключаться на новые белки-мишени и значительно расширить круг потенциальных пользователей. Схема работы системы в общем виде соответствует общепринятой схеме виртуального

скрининга. Сложность и точность, используемых в системе алгоритмов, приводит к тому, что время расчета одного лиганда существенно возрастает и может составлять от 1 до 100 часов на одном процессоре. Для того, чтобы иметь возможность анализировать базы данных, содержащие тысячи молекул, в наш комплекс была встроена система распределенных вычислений X-Com (разработка лаборатории профессора В. В. Воеводина НИВЦ МГУ). Эта система позволяет одновременно производить расчет группы лигандов на различных кластерных платформах и персональных компьютерах независимо от того, какая операционная система на них установлена.

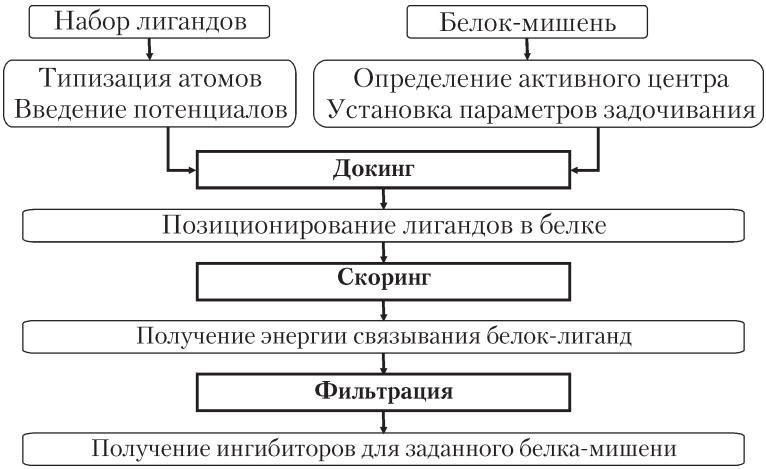


Рис. 1. Общая схема работы системы KeenBASE

Генетический алгоритм

Основным вычислительным модулем программного комплекса KeenBASE является программа докинга SOL [23], основным алгоритм которой — генетический алгоритм. Перед генетическим алгоритмом глобальной оптимизации стоит задача нахождения глобального минимума общей энергии в пространстве размерностью $N+6$, где N — количество внутренних степеней свободы лиганда. Под общей энергией мы понимаем сумму внутренней энергии лиганда и энергии лиганда в потенциальном поле протеина. Цифра 6 учитывает сумму трех вращательных и трех поступательных

степеней свободы при движении лиганда как целого. Понятно, что даже при умеренной сложности молекулы лиганда (N=5–6) проблема поиска глобального минимума функции в пространстве размерностью более 10 является непростой задачей.

Суть генетического алгоритма заключается в моделировании процессов эволюции по Дарвину некоей популяции, составленной из отдельных особей. Под особью мы понимаем лиганд в одной из возможных конформаций и его положение в активном центре. Наследственность каждой особи или генотип, который служит критерием для естественного отбора, представляет собой одномерный массив a_i вещественных чисел (генов) в диапазоне от 0 до 1 — хромосому. Каждая хромосома однозначно определяет положение лиганда в пространстве, при этом величина каждого гена соответствует нормированной на 1 величине соответствующей степени свободы лиганда. Так, например, если лиганд имеет 4 внутренних степени свободы, то его хромосома состоит из 10 генов, из которых 4 описывают внутренние вращения в лиганде, 3 — его перемещение как целого и 3 — вращение лиганда как целого.

Первоначальная популяция особей создается заполнением генов всех хромосом при помощи генератора случайных чисел. После этого происходит подсчет полной энергии для каждой особи и лучшие особи с минимальной энергией отбираются в так называемый mating pool, где они участвуют в создании нового поколения особей. Новое поколение формируется из особей mating pool при помощи трех генетических операторов — скрещивания, прямого переноса и мутации. **Скрещивание** представляет собой создание новой хромосомы из двух родительских хромосом путем случайного объединения их генов. Все остальные особи создаются путем **прямого переноса** из mating pool. Затем на образованную таким образом популяцию действует оператор мутации. **Оператор мутации** с некоторой вероятностью подвергает случайному изменению значение гена в хромосоме. Наряду с описанными операторами осуществляется также **прямой перенос элиты** (особей с наименьшей общей энергией) из предыдущего в последующее поколение. Это операция проводиться во избежание утери уже найденных в процессе оптимизации лучших решений. Описанная процедура смены поколений повторяется заданное пользователем число раз. Для того чтобы

еще более стимулировать исследование разнообразных областей энергетической поверхности и предотвратить преждевременную концентрацию процесса поиска вокруг локального минимума, в алгоритме реализована так называемая концепция **нишинга**. Она предусматривает наблюдаемое в природе ослабление конкуренции между особями, занимающими различные экологические ниши, т. е. сильно друг от друга отличающимися по генотипу. Нишинг провоцирует разнообразие особей в популяциях и предотвращает преждевременное попадание поиска в локальный минимум.

В ходе работы программы процедура вызова генетического алгоритма совершается определенное число раз, получая несколько решений. Анализ взаимного положения этих решений важен для выяснения качества работы программы и корректности полученных результатов. С этой целью производится кластеризация найденных решений соответственно их расположению. Для этого вычисляется матрица среднеквадратичного отклонения (RMSD) решений друг от друга. После получения матрицы RMSD происходит процедура кластеризации — решения, RMSD между которыми не превышают некоторой заданной величины (обычно 1 Е), собираются в один кластер. Кластеризация выявляет качество докинга. Если имеется небольшое количество многонаселенных кластеров, то можно считать, что докинг произведен успешно. Если кластеров много и они содержат небольшое количество решений, то, возможно, глобальный минимум не найден и необходим новый запуск программы с другими параметрами.

Для подтверждения корректности физических и математических закономерностей, заложенных в программу докинга, было проведено специальное исследование, называемое валидацией программы. Результаты валидации приведены на сайте www.keenbase.ru и они показывают, что точность позиционирования лигандов с помощью нашей программы соизмерима с точностью существующих мировых аналогов. Так для нативных лигандов тромбина, взятых из базы данных PDB, точность позиционирования составляет порядка 1.6 Е. Также по результатам валидации был определен размер кубической области, в которую осуществляется позиционирование лиганда. В наших расчетах ребро куба составляет 22 Е или 101 точку сетки и включает в

себя активный центр протеина и окружающие его аминокислотные остатки.

Энергия лиганда в поле протеина дает указание на прочность комплекса протеин-лиганд. Чем ниже эта энергия, тем интереснее данный лиганд. Однако при оценке прочности связывания нужно принимать во внимание и энтропийные факторы, возникающие из-за ограничений степеней свободы лиганда при его связывании с протеином. Поэтому, для окончательной оценки прочности связывания лиганда с протеином в программе докинга используется скоринг-функция, имеющая следующий вид:

$$\Delta G_{\text{bind}} = \alpha E_{VdW} + \beta E_{ES} + \delta G_{GB(\text{protein-ligand})} +$$
$$+\varepsilon G_{\text{proteindesolvation}} + \mu G_{\text{liganddesolvation}} + E_{\text{ligandinner}}$$

где E_{VdW}, E_{ES} — Ван-дер-Ваальсово и электростатическое взаимодействия между протеином и лигандом, которые описываются в рамках модели силового поля MMFF94. $G_{GB(\text{protein-ligand})}$ – потенциал экранирования электростатического взамодействия зарядов протеина с зарядами лиганда, $G_{\text{proteindesolvation}}$, $G_{\text{liganddesolvation}}$ — потенциал десольватации соответственно протеина и лиганда и описываются в рамках приближения обобщенного Борна [24]. Значения скоринг-функции рассчитываются для каждого полученного генетическим алгоритмом решения.

Разработка новых ингибиторов тромбина

Нарушения процесса свертываемости крови — одна из распространенных проблем, с которыми сталкивается современная медицина. Образование нежелательных тромбов угрожает, например, пассажирам, совершающим длительные авиаперелеты (так называемый синдром экономического класса), а также пациентам, проходящим процедуру гемодиализа или возмещения большой кровопотери. Кроме того, в группу риска входят все люди старше 60 лет. Ключевым ферментом в сложной системе свертываемости крови является тромбин. Именно он производит превращение белка фибриногена, в фибрин, который и образует основу тромба. Казалось бы, наиболее очевидный путь подавить нежелательное образование тромба — блокировать работу тромбина — т. е. найти прямой ингибитор этого фермента.

Но, как ни странно, все современные антикоагулянты (лекарственные вещества, предотвращающие или замедляющие свертывание крови), применяемые в нашей медицинской практике реализуют другие, непрямые механизмы воздействия на систему свертываемости. Отсюда — узкий терапевтический диапазон, непредсказуемость воздействия и, как следствие, — ограниченное применение. Более того, в некоторых тяжелых случаях такие антикоагулянты оказываются попросту бессильны предотвратить тромбообразование. В настоящее время в клинической практике для предотвращения избыточного тромбообразования используется **гепарин**, который является всего лишь кофактором природного ингибитора тромбина — антитромбина (ATIII). При недостатке в крови ATIII введение гепарина не подавляет активность тромбина и единственный выход в данном случае — это добавить прямой ингибитор тромбина. Понятен интерес, который проявляют ведущие мировые фармакологические фирмы (Merck, Boehringer-Mannheim и др.) к разработке лекарственного средства — прямого ингибитора тромбина. Существует только один прямой низкомолекулярный синтетический ингибитор тромбина — **аргатробан**, разрешенный к клиническому применению.

Наша лаборатория в НИВЦ МГУ вместе с Гематологическим научным центром РАМН (лаборатория профессора Ф. И. Атауллаханова) и Институтом органической химии РАН на протяжении более полутора лет занимается разработкой прямых ингибиторов тромбина. В рамках разработанной системы KeenBASE и с помощью технологии распределенных вычислений X-Com было проведено несколько больших компьютерных экспериментов для выявления соединений, потенциально являющихся ингибиторами тромбина.

Первый эксперимент проводился в Рождественские каникулы 2006 г. в течение 10 дней. За это время было рассчитано 2123 молекулы из базы данных NCI diversity, где среднее время расчета составило 20 часов на 1 молекулу. В эксперименте участвовали 273 процессора компьютеров НИВЦ МГУ и Южноуральского университета в г. Челябинске. Результаты этих расчетов позволили нам сделать первые предсказания и отобрать соединения, которые должны проявлять активность в экспериментах. Проведенные измерения заказанных соединений из этой бесплатной базы данных показали, что часть соединений действительно

обладала ингибирующей активностью, что подтвердило наши предсказания и позволило создать фокусированные виртуальные библиотеки для дальнейших исследований.

Второй эксперимент, продолжительностью 4 дня, состоялся в рамках первого фестиваля науки, проходившего в МГУ в октябре 2006 года. Тогда в эксперименте были задействованы компьютеры НИВЦ, ВМиК МГУ, университета Дубны, Южноуральского (г. Челябинск) и Башкирского университетов (г. Уфа), 600 процессоров. Было рассчитано 1840 молекул из которых около 37 штук было отобрано для последующего синтеза.

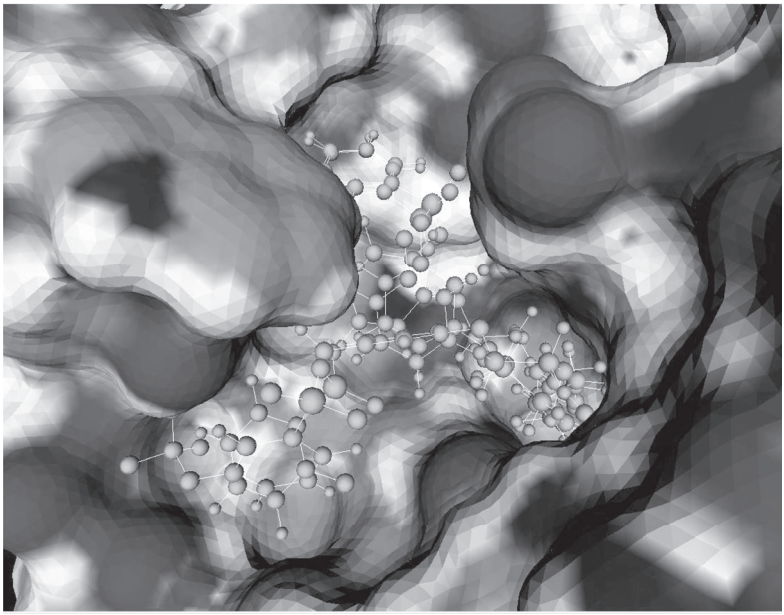


Рис. 2. Расположение ингибиторов в активном центре тромбина

И, наконец, самый последний эксперимент проводился на кластере «Скиф-Cyberia», установленном в Томском госуниверситете в течение 5 дней в конце мая начале июня 2007 г., 1120 процессоров. В связи с тем, что расчеты проводились на фокусированной библиотеке, которая была построена специально для оптимизации соединения-лидера, то требовалась их высокая точность, поэтому параметры расчета были отличными от пре-

дыдущих экспериментов. В результате среднее время, затраченное на 1 лиганд, составило около 72 часов.

Наряду с большими компьютерными экспериментами, текущими расчетами, которые проводились в ходе работы, непрерывно было занято около 20 процессоров на кластере НИВЦ МГУ.

По результатам расчетов несколько сотен соединений имеют значения скоринг-функции, сравнимые со значением для соединений-лидеров, что позволило спрогнозировать их высокую ингибирующую способность. На рис. 2 показано положение нескольких лигандов из данной библиотеки в активном центре тромбина.

Часть из отобранных соединений была синтезирована в ИОХ РАН и проверена экспериментально в Гематологическом Научном центре. Измерения показали, что отобранные нами лиганды не только ингибируют тромбин, но кроме того, их ингибирующая активность существенно превышает активность **аргатробана**, что позволяет предполагать, что созданный нами новый класс низкомолекулярных синтетических ингибиторов тромбина может успешно быть использован для создания средства для коррекции гиперкоагуляции, вызываемой разбавлением плазмы.

Заключение

В заключении хочется отметить, что на разработку нового класса ингибиторов тромбина было затрачено всего 1,5 года, в большой степени благодаря тесному сотрудничеству молекулярных дизайнеров, биохимиков, биофизиков и химиков-синтетиков. В целом за время работы было рассчитано порядка 6000 соединений, около 50 из них было синтезировано. Использование компьютеров позволило не только сократить время начальной стадии разработки нового лекарства, но и материальные затраты по данному проекту. Кроме того, виртуальный скрининг увеличил химическое разнообразие потенциальных кандидатов в лекарства за счет проведения первоначального этапа поиска в базе данных, содержащих существенно различные классы органических соединений. Это позволило выбрать наиболее перспективные ингибиторы для создания фокусированных библиотек, которые впоследствии доказали свою эффективность.

Литература

1. **Lyne P. D.** Drug Discovery Today. — 2002. — **7**. — P. 1047–1055.

2. **Kitchen D. B., Decornez H., Furr J. R., Bajorath J.** Nat Rev. Drug Discov. — 2004 — Vol. 3. — P. 935—949.

3. **Klebe G.** Drug Discovery Today. — 2006. — 11. — P. 580—594.

4. **Sousa S. F., Fernandes P. A., Ramos M. J.** Proteins: Struct., Funct. and Bioinf. — 2006 — Vol. 65. — P. 15—26.

5. NCI (National cancer institute) diversity data base [Электронный ресурс]. — Режим доступа: http://dtp.nci.nih.gov/docs/3d_database/Structural_information/structural_data.html.

6. **Berman H. M., Westbrook J., Feng Z., Gilliland G., Bhat T. N., Weissing H., Shindyalov I.N., Bourne P. E.** Nucleic Acids Res. — 2000. — 28. — P. 235—242.

7. **Sadowski J.** J. Comput. Aided Mol. Design — 1997. — **11**. — P. 53—60.

8. **Ewing T. J. A., Makino S., Skillman A. G., Kuntz I. D.** J. Comput. Aided Mol. Design. — 2001. — 15. — P. 411—528.

9. **Rarey M., Kramer B., Lengauer T., Klebe G.** J Mol. Biol. 1996. — 261. — P. 470—489.

10. **Friesner R. A.** et al. J Med. Chem. — 2004. — 47. — P. 1739—1749.

11. Kearsley S. K., Underwood D. J., Sheridan R. P., Miller M. D. J. Comput. Aided Mol. Design. — 1994. — **8**. — P. 565—582.

12. **Olson A. J., Goodsell D. S.** Proteins 1993. — **17**. — P. 1—10.

13. **Goodsell D. S., Olson A. J.** Proteins: Structure, Function and Genetics. — 1990. — 8. — P. 195—202.

14. **Morris G. M., Goodsell D. S., Halliday R. S., Huey R., Hart W. E., Belew R. K., Olson A. J.** J. Computational Chemistry — 1998. — 19. — P. 1639—1662.

15. GOLD Version 1.2 [Электронный ресурс]. — Режим доступа : http://www.ccdc.cam.ac.uk/product/life_sciences/gold. — 2003.

16. **Westhead D. R., Clark D. E., Murray C. W.** J.Comput. Aided Mol. Design. — 1997. — 11. — P. 209—228.

17. **Kramer B., Rarey M., Lengauer T.** Proteins. — 1999. — **37**. — P. 228—241.

18. **Weiner S. J., Kollman P. A., Nguyen D. T., Case D. A.** J Comp. Chem. — 1986. — 7. — P. 252.

19. **Halgren T. A.** J Comp. Chem. — 1996. — **17.** — P. 490—519, 520—552, 553—586, 587—615, 616—641.

20. **Bohm H.-J.** J.Comput. Aided Mol.Design. — 1992. — **6**. — P. 593—606.

21. **Muegge I.** Perspect. Drug. Disc. Des. — 2000. — **20**. — P. 99—114.

22. **Gohlke H., Hendlich M., Klebe G.** J Mol. Biol. — 2000. — **295**. — P. 337—356.

23. Свидетельство об официальной регистрации программ для ЭВМ № 2006611167 «SOL» от 31 марта 2006 г.

24. **Romanov A.** et.al J. Phys. Chem. A. — 2004. — 108. — 9323.

Исследование некоторых задач молекулярной биологии и химии поверхностно-активных соединений при помощи компьютерного эксперимента, проведенного на Армкластере

А. А. Шагинян, Г. А. Гарабекян, А. Г. Погосян, Л. Г. Арсенян
Международный научно-образовательный центр НАН РА,
Институт прикладных проблем физики НАН РА

В последнее время применение компьютерного эксперимента в научно-исследовательских работах приобретает все большее и большее распространение. Его возникновение, в первую очередь, связано с высоким развитием информационных технологий: созданием высокопроизводительных вычислительных комплексов и новейших инструментов вычисления, а также с резким ростом расходов по применению методов физического эксперимента. Компьютерный эксперимент представляет собой симбиоз физического эксперимента и теории. С одной стороны цифровая модель системы строится по данным физического эксперимента, с другой стороны само исследование проводится при помощи воздействия силовых полей, описываемых формулами, на систему.

Исследования ученых последних лет показали, что методы и возможности информационных технологий можно эффективно использовать и для исследования молекулярной структуры и свойств сложных, многокомпонентных биологических и химических систем в динамике [1—3]. Возникают совершенно новые перспективные области применения вычислительных методов, которые позволяют рассматривать более комплексные и близкие к реальным системам модели. Уже созданы и развиваются специализированные алгоритмы и инструменты расчета, наиболее распространенным из которых является метод «Молекулярной динамики» [4].

В настоящей работе методом компьютерного эксперимента исследована структура мембраны эритроцита человека и ламеллярной жидкокристаллической фазы некоторых поверхностно-активных веществ (ПАВ).

Компьютерные эксперименты были проведены на высокопроизводительном Армкластере НАН РА. На данный момент Армкластер является самой мощной высокопроизводительной системой на Кавказе, с пиковой производительностью 783,36 Гигафлоп и 2 Гб оперативной памяти на каждом узле. Узлы объединены в сеть при помощи высокоскоростной сети Myrinet (2 Гбит/с). Кластер работает под ОС Linux RedHat 9.0 с поддержкой SMP. Для визуализации молекулярной динамики использован пакет VMD: Visual Molecular Dynamics.

Ниже представлены параметры симулирования.

Использованные ноды / процессоры:	25/50
Тип процессора:	Intel Xeon 3.06GHz
Тип связи:	Myrinet
Программное обеспечение:	Gromacs, NAMD 2.5, MDesigner
Силовые поля:	CHARMM all27
Температура и давление в эксперименте: (Langevin and Nose-Hoover Langevin piston methods)	320K и 1 атм
Модель воды:	TIP3
Биологическое время симулирования:	60—100 нсек.
Химическое время симулирования:	20—230 нсек.

В качестве основного метода моделирования был использован метод «Молекулярной динамики» (МД), который позволяет моделировать детальную микроскопическую картину внутренних движений молекул (или макромолекул) и успешно используется в теоретических исследованиях структуры и динамики биологических макромолекул, жидкостей, газов и других сложных молекулярных систем. Методом МД рассчитываются классические (ньютоновские) траектории движения атомов молекулы в силовом поле эмпирического атом-атомного потенциала, т. е. моделируется детальная микроскопическая картина внутреннего теплового движения макромолекулы в субнаносекундных интервалах времен. Основу метода составляет численное решение классических уравнений Ньютона для системы взаимодействующих частиц:

$$m_i \frac{d^2 r_i(t)}{dt^2} = F_i(r), \; i = 1,2,...,n \qquad (1)$$

где r_i — радиус-вектор i-го атома, m_i — его масса, F_i суммарная сила, действующая на i-й атом со стороны остальных частиц:

$$F_i(r) = -\frac{\partial U(r)}{\partial r_i}. \quad (2)$$

Здесь $r = \{r_1, r_2, ...r_n\}$, а $U(r)$ — потенциальная энергия, зависящая от взаимного расположения всех атомов; n — число атомов.

Задав координаты и скорости всех атомов в начальный момент времени, численно решают уравнения движения, вычисляя на каждом шагу все силы, новые координаты и скорости частиц. Температура определяется как средняя кинетическая энергия, приходящаяся на одну степень свободы системы:

$$T(t) = \frac{1}{3Nk_B}\sum_{i=1}^{n} m_i \nu_i^2 \quad (3)$$

$$\nu_i = \frac{dr_i}{dt} \quad (4)$$

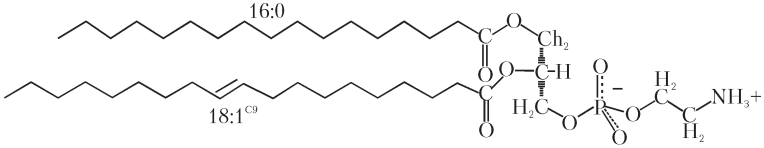
Здесь N — полное число степеней свободы молекулы, k_B — постоянная Больцмана. В случае изолированной системы $N = 3n - 6$, поскольку сохраняется ее полный импульс и момент импульса. Кроме того, в этом случае сохраняется полная энергия системы, а температура получается усреднением ее мгновенных значений $T(t)$ по некоторому интервалу времени.

Для моделирования методом МД нами использованы следующие файлы:

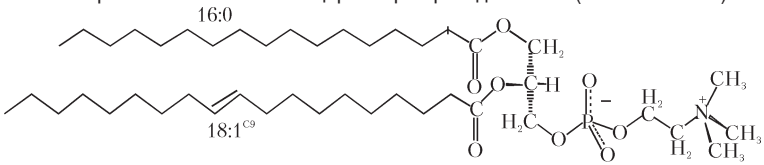
Файл координат атомов системы и файл топологий молекул модели, в котором содержатся параметры всех ковалентных связей, валентных и торсионных углов, и т. д., атомов системы. За основу были взяты типовые топологии фосфолипидных молекул с полярными головками фосфатидилэтаноламина и фосфатидилхолина, содержащиеся в стандартном файле топологии top_all27_prot_lipid. Эти данные имеются в последней версии силовых полей CHARMM27 [5], а для манипуляций с атомами использовалась программа VMD. Был написан скрипт на макроязыке TCL, который в автоматическом режиме создает стартовый фосфолипидный бислой или мицеллу ПАВ, случайным образом вращая молекулы фосфолипидов или ПАВ вокруг своих

осей, помещая каждую молекулу в прямоугольные ячейки, после чего эти ячейки случайным образом распределяются в массиве размером 64 × 2 в случае мембраны и 256 × 2 в случае мицелл. Для симуляции мембран эритроцита был проведен сравнительный анализ [6] известных пакетов Gromacs и NAMD, на основании которого было выбрано программное обеспечение для наших исследований. Для проведения компьютерного эксперимента на биологических мембранах был разработан также специальный пакет программ Mdesigner [7]. При моделировании использованы пять типов фосфолипидов, имеющихся в мембране эритроцита человека [8].

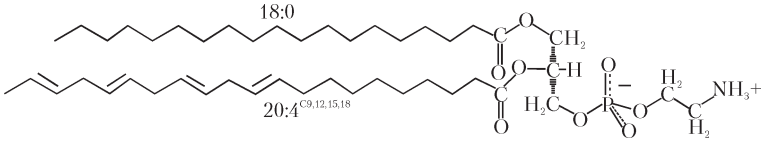
1-палмитоил-2-олеоил-СН-глицеро-3-фосфатидилэтаноламин (ПОФЭ) — 10 %



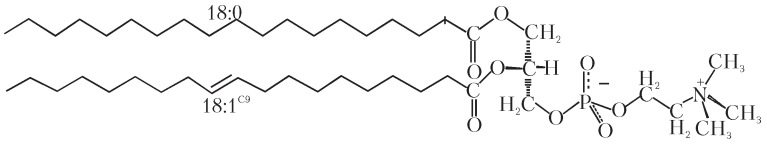
1-стеароил-2-олеоил-СН-глицеро-3-фосфатидилхолин (СОФХ — 12 %)

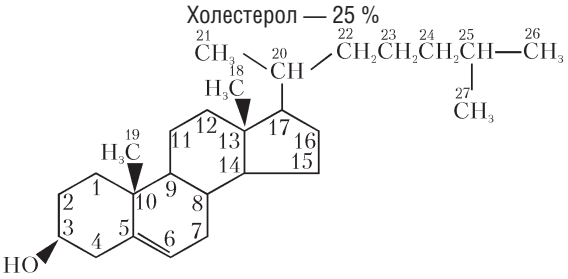


1-стеароил-2-арахидиноил-СН-глицеро-3-фосфатидилэтаноламин (САФЭ) — 12 %



1-палмитоил-2-олеоил-СН-глицеро-3-фосфатидилхолин (ПОФХ) — 13 %



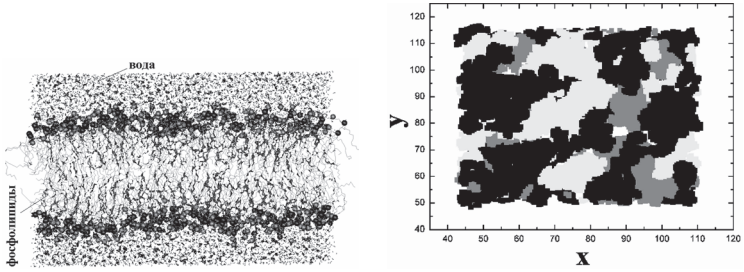


В фосфолипидную мембрану ввели также важный функциональный белок характерный для эритроцита: Гликофорин А.

В качестве ПАВ использованы Додецилсульфат натрия- $C_{12}H_{25}SO_4Na$ и Пентадецилсульфонат натрия — $C_{15}H_{31}SO_3Na$, промышленной марки Е-30 и К-30.

Результаты компьютерного эксперимента

Мембрана эритроцита. На рис. 1 представлена модель равновесного фосфолимодного бислоя мембраны эритроцита человека в присутствии воды, полученная компьютерным экспериментом при помощи минимизации свободной энергии системы. Анализ данных, представленных на рис. 1б, показывает, что в мембране разные типы фосфолипидов не смешиваются.



а б

Рис. 1. Равновесная модель фосфолипидного бислоя (а), при соотношении фосфолипидов: 12 % ПОФХ / 13 % СОФХ / 12 % САФЭ / 10 % ПОФЭ/25 %ХОЛ. Распределение фосфолипидов в мембране (б). Проекция на плоскости x—y

Таким образом, из данных компьютерного эксперимента можно делать предположение, что в мембране эритроцита должны существовать домены отдельных фосфолипидов.

Если это предположение верно, то оно должно сказываться на процессе диффузии фосфолипидных молекул в мембране. С этой целью была исследована латеральная диффузия молекул всех типов фосфолипидов в модельной мембране эритроцита.

Для определения коэффициента латеральной диффузии (проекция диффузии молекул фосфолипидов в плоскости $x-y$) воспользовались формулой, описывающей зависимость среднеквадратичного отклонения центра тяжести отдельных типов фосфолипидов (MSD) от времени:

$$MSD = \sum_{\text{all}} < \left| \vec{r}(t) - \vec{r}(0) \right|^2 >,$$

где $\vec{r}(t)$ и $\vec{r}(0)$ радиус-векторы центра тяжести молекул в начале процесса диффузии и в момент t.

При этом зависимости MSD для отдельных фосфолипидов от времени симуляции t описываются формулой:

$$D_{lat} = \frac{MSD}{4 \cdot t}.$$

На рис. 2 представлены зависимости MSD от времени симуляции для отдельных фосфолипидов при латеральной диффузии их молекул в мембране, полученные компьютерным экспериментом.

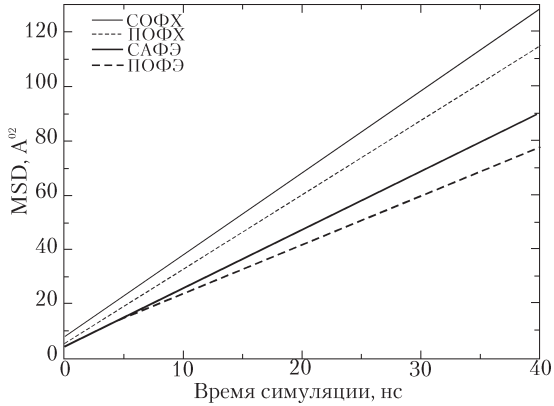


Рис. 2. Зависимость MSD от времени симуляции для фосфолипидов: СОФХ, ПОФХ, САФЭ и ПАФЭ в мембране эритроцита

Значения коэффициентов латеральной диффузии отдельных фосфолипидов в мембране эритроцита представлены в табл. 1.

Таблица 1

Коэффициенты латральной диффузии для разных фосфолипидов в мембране

Фосфолипиды	**СОФХ**	**ПОФХ**	**САФЭ**	**ПОФЭ**
Кооффициент диффузии ($см^2/с$)	$4{,}2\times10^{-8}$	$3{,}7\times10^{-8}$	$2{,}9\times10^{-8}$	$2{,}6\times10^{-8}$

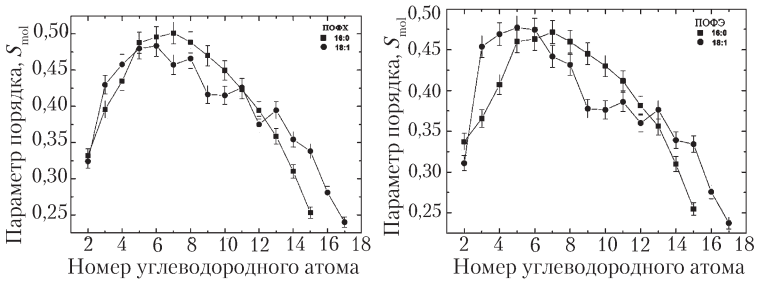
В настоящее время, методы физического эксперимента позволяют измерять коэффициенты латеральной диффузии только для искусственных бимолекулярных слоев, состоящих из одного типа фосфолипида [9, 10]. Сравнивая значения коэффициентов латеральной диффузии для бислоев отдельных фосфолипидов и мембраны эритроцита, содержащей все эти фосфолипиды, обнаруживаем их совпадение. Такое совпадение коэффициентов латеральной диффузии может быть только в том случае, если отдельные фосфолипиды в мембране имеют свои собственные ареалы существования, как это представлено на рис. 1б.

В динамике при симулировании до 100 нс видно, что все компоненты фосфолипидной мембраны находятся в состоянии непрерывного теплового движения. При этом, интенсивным конформационным изменениям подвергаются углеводородные цепочки молекул фосфолипидов внутри мембраны. Для изучения усредненной картины конформационных изменений углеводородных цепочек молекул отдельных фосфолипидов в мембране методом компьютерного эксперимента было исследовано изменение параметра ориентационного порядка углеводородных цепочек молекул:

$$S_{\text{mol}} = \left| \frac{3}{4} < \cos^2 \theta_i > -\frac{1}{4} \right|,$$

где θ_i — угол между осью молекулы и нормали к поверхности мембраны. Скобки обозначают ансамбль и усреднение во времени. Из приведенной формулы видно, что если углеводородная цепочка или ее фрагмент расположены параллельно к нормали поверхности бислоя, то $S_{\text{mol}} = 0{,}5$, а если перпендикулярно, то $S_{\text{mol}} = 0{,}25$.

На рис. 3, в качестве примера, показаны изменения степени ориентации углеводородных цепочек молекул ПОФХ и ПОФЭ по мере углубления в гидрофобный объем бислоя.



а б

Рис. 3. Зависимость параметра ориентационного порядка углеводородных цепочек молекул фосфатидилхолина — ПОФХ (а) и фосфатидилэтаноламина — ПОФЭ (б) от глубины погружения в мембрану

Как видно из рисунков, участки углеводородных цепочек молекул фосфолипидов, расположенных ближе к поверхности бислоя (к глицерольной группе) ориентируются приблизительно перпендикулярно к нормали, обусловленное вероятно тем, что глицерольная группа расположена перпендикулярно к нормали поверхности бислоя. По мере удаления от глицерольной группы в глубь бислоя углеводородные цепочки становятся конформационно более свободными и начинают ориентироваться преимущественно параллельно к нормали поверхности бислоя. Дальнейшее удаление от полярной группы молекул фосфолипида из-за увеличения степени свободы движения углеводородных цепочек их ориентационный порядок начинает уменьшаться, стремясь к полной дезориентации.

Ориентационный порядок экспериментально был исследован для фосфолипидных бислоев, состоящих из одного типа фосфолипида [11].

Как видно из рис. 4, данные, полученные экспериментально методом ЯМР спектроскопии и компьютерным экспериментом для дипальмитоилфосфатидилхолина, хорошо согласуются.

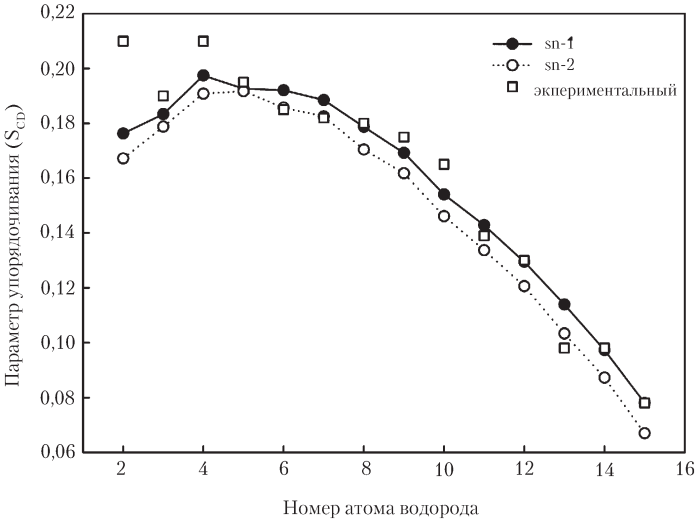


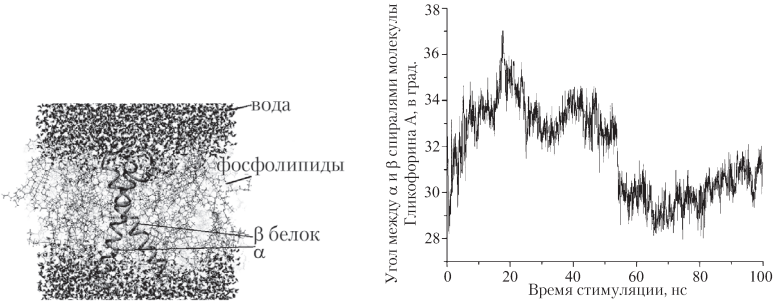
Рис. 4. Ориентационные параметры порядка углеводородных цепочек молекул дипальмитоилфосфатидилхолина, полученные физическим экспериментом и моделированием

Таким образом, в тех случаях, когда можно поставить физический эксперимент, результаты физического и компьютерного экспериментов достаточно хорошо совпадают.

Известно, что одним из наиболее изученных из мембранных белков эритроцита является — Гликофорин А представляющий собой гликопротеид, пронизывающий фосфолипидные слои мембраны и выступающий наружу. Гликофорин образует устойчивый димер не только в природных системах, но и в искусственных липидных средах, таких как мицеллы додецилфосфатидилхолина. Для этого белка найдена пространственная структура димера методом ЯМР-спектроскопии [12].

Модель фрагмента мембраны эритроцита нами была построена при помощи внедрения мембранного белка эритроцита Гликофорина А в равновесный гидратированный фосфолипидный бислой. На рис. 5а представлена равновесная модель фрагмента мембраны эритроцита, содержащая Гикофорин А.

Рассмотрим динамику поведения α и β субединиц Гликофорин А в мембране. На рис. 5б представлена зависимость угла наклона между α и β субединицами белка внутри мембраны.



а б

Рис.5. Равновесная модель фрагмента мембраны эритроцита (а). Угол между α, β спиралями молекулы Гликофорина А (б)

Имеются работы, где методами ЯМР-спектроскопии и динамического моделирования изучено влияние структуры молекул фосфолипида, на конформацию Гликофорина А для исскуственной мембраны, состоящей из одного типа фосфолипида [13]. Исследована зависимость конформации Гликофорина от длины и степени насыщенности углеводородных цепей фосфолипидов окружающих Гликофорин. С помощью МД и ЯМР установлено, что с увеличением количества CH_2-групп и двойных связей углеводородных цепей фосфолипидов увеличивается угол между спиралями димера гликофорина. Показано, что при увеличении числа CH_2-групп в углеводородных цепочках молекул фосфолипида с 14 до 18, угол между спиралями в среднем достигает величины 11 градусов. В нашем случае, где вместо гомогенной фосфолипигной мембраны исследуется гетерогенная мембрана с молекулами фосфолипидов имеющих углеводородные цепочки с числом CH_2-групп от 16 до 20, в среднем угол имеет значение порядка 30 градусов. По всей вероятности увеличение угла связано с доминированием гидрофобности в окружении белка.

Большой интерес представляет взаимодействие холестерина с Гликофорином А и влиянием белка на динамику поведения молекул воды в мембране.

На рис. 6 представлен белок Гликофорин А в окружении молекул холестерина, в мембране эритроцита. Как видно из рисунка, в мембране эритроцита молекулы холестерина взаимодействуют с молекулой Гликофорина А и окружают ее.

Важной проблемой является проникновение молекул воды в мембрану. Компьютерный эксперимент показывает отсутствие молекул воды в гидрофобной части мембраны в отсутствии белка и проникновение в мембрану в присутствии Гликофорина А.

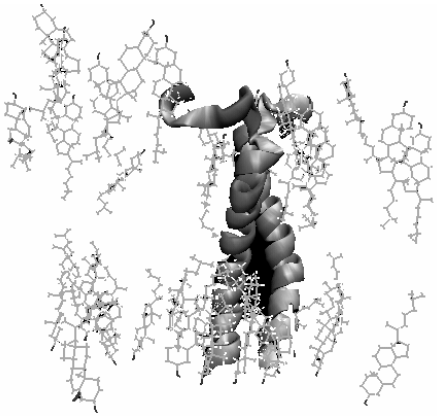
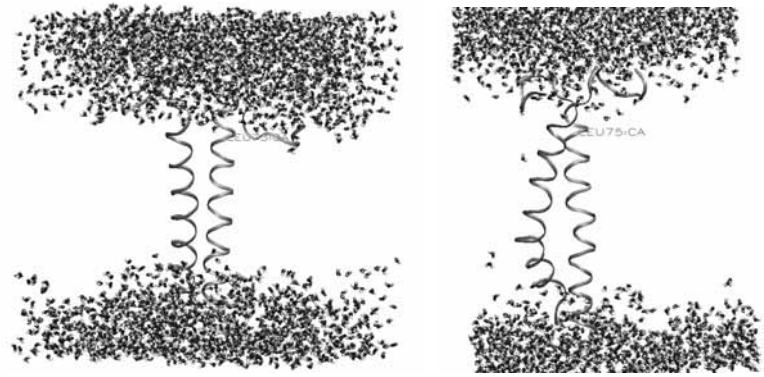


Рис. 6. Белок в окружении холестерина.
Молекулы остальных фосфолипидов и воды на рисунке не показаны

На рис. 7 показано стартовое состояние системы вода-Гликофорин А в начале компьютерного эксперимента и после 100 нс симулирования. Молекулы фосфолипидов на рисунке не показаны.



а б

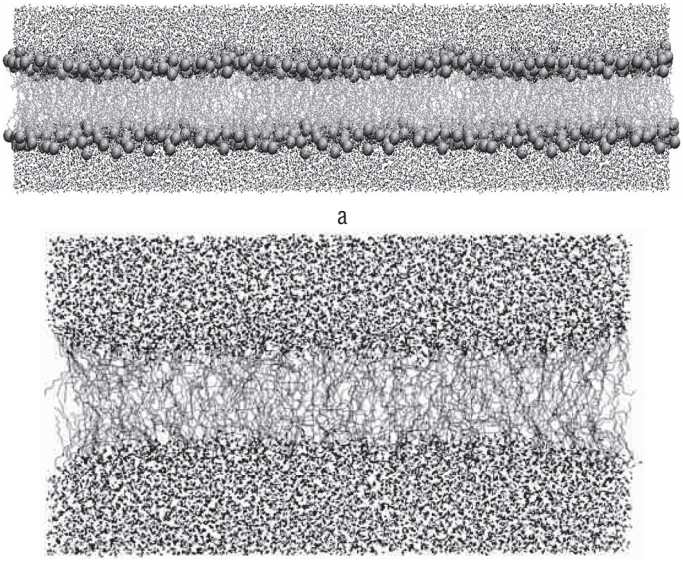
Рис. 7. Стартовое состояние мембраны эритроцита (а), и после 100 нс симуляции. Молекулы фосфолипида на рисунке не показаны

Как видно из рисунка, в непосредственной близости молекулы белка имеет место проникновение молекул воды в мембрану, что вероятно обусловлено асимметричной структурой белка.

Плоские мицеллы ПАВ в воде. Водно-мицеллярные растворы ПАВ имеют большое применение в химии и биологи, в частности: в эмульсионной полимеризации, в нефтехимии, в мицеллярном катализе, в фармакологии, в экологии и санитарии и т. д, выступая в качестве солюбилизирующего агента, способствующего растворению водонерастворимых жидкостей в водно-мицеллярной системе.

Многие структурные свойства водно-мицеллярных растворов ПАВ можно исследовать различными физическими методами в высококонцентрированных водных растворах ПАВ в жидкокристаллическом состоянии.

В настоящей работе нами впервые предпринята попытка, при помощи компьютерного эксперимента, исследовать плоские мицеллы Додецилсульфат натрия-$C_{12}H_{25}SO_4Na$ (ДДС) и Пентадецилсульфонат натрия-$C_{15}H_{31}SO_3Na$(ПДС) в ламеллярной, жидкокристаллической фазе.



а

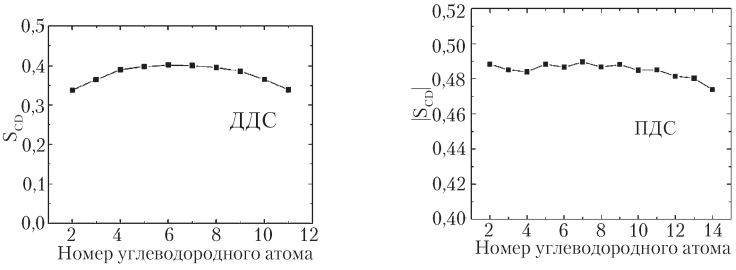
б

Рис. 8. Равновесная модель плоской мицеллы ДДС (а) и ПДС (б) в воде, в ламеллярной жидкокристаллической фазе

Исследования проводились по той же методике, что и мембраны эритроцита. Время симуляции до 230 нс.

На рис. 8 представлена равновесная модель плоских мицелл ДДС и ПДС в водном окружении.

Одним из главных параметров, определяющих структуру мицеллы ПАВ, который можно измерять при помощи физического эксперимента, является коэффициент ориентационного порядка углеводородных цепочек молекул ПАВ.



а б

Рис. 9. Зависимость параметра ориентационного порядка углеводородных цепочек молекул ДДС (а) и ПДС (б) от глубины погружения в мицеллу

На рис. 9. представлены зависимости параметра ориентационного порядка углеводородных цепочек молекул ДДС и ПДС от глубины погружения в мицеллу.

Данные, полученные компьютерным экспериментом, хорошо совпадают с данными физического эксперимента [14, 15].

Таким образом, показана возможность применения компьютерного эксперимента для исследования сложных биологических и химических систем.

Литература

1. **Tieleman D. P., Berendsen H. J. C.** Molecular dynamics simulations of a fully hydrated dipalmitoylphosphatidylcholine bilayer with different macroscopic boundary conditions and parameters. // J. Chem. Phys. — 1996. — 105. — P. 4871—4880.

2. **Tu K., Tobias D. J., Klein M. L.** Biophys. J. — 1995. — 69. — P. 2558—2562.

3. **Chiu S. W., Clark M., Balaji V., Subramaniam S., Scott L. H., Jakobsson E.** Biophys. J. — 1995. — 69. — P. 1230—1245.

4. **Lindahl E., Hess B., D. van der Spoel.** J. Mol. Mod. — 2001. — 7. — P. 306—317.

5. **Brooks B. R., Bruccoleri R. E., Olafson B. D., States D. J., Swaminathan S., Karplus M.,** CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations // J. Comput. Chem. — 1983. — 4. — P. 187—217.

6. **Pogosyan A. H., Yegiazaryan G. A., Gharabekyan H. H., Shahinyan A. A.** Commun. Comput. Phys. — 2006. — Vol. 1. — № 4 — P. 736—743.

7. **Shahinyan A. A., Pogosyan A. H., Yegiazaryan G. A., Gharabekyan H. H.** Electronic J. of Natutal Sci. — 2004. — 1 (2).

8. Op den Kamp JAF, Roelofsen B & van Deenen LLM (1985) Structural and dynamic aspects of phosphatidylcholine in the human erythrocyte membrane. rends Biochem Sci 8: 320—323.

9. **Blume A.** Dynamic properties. In Phospholipid Handbook / Ed. by G. Cevc. — New York : Marcel Dekker, 1993. — P. 455—552.

10. **O'Leary T. J.** Proc Natl Acad Sci USA. January. — 1987. — 84(2). — P. 429—433.

11. **Tanford C.** 1973 in The Hydrophobic Effect: Formation of Micelles and Biological Membranes, 1st Ed. — N. Y. : John Wiley & Sons. — P. 1—200.

12. **MacKenzie K., Prestegard J., Engelman D.** A transmembrane helix dimmer: Structure and Implications // Science. — 4 april. — 1997. — Vol. 276. — P. 131—133.

13. **Petrache H., Grossfield A., MacKenzie K., Engelman D., Woolf Th.** Modulation of Glycophorin A Transmembrane Helix Interactions by Lipid Bilayers : Molecular Dynamics Calculations. // J. Molec. Biol. — 2000. — 302. P. 727—746.

14. **Kekicheff P., Grabielle-Madelmont C., Ollivon M.** Journal of Colloid. and Interface Science. — 1989. — 131. — P. 112.

15. **Шагинян А. А.** Роль структурной организации ионных мицелл в механизме формирования макромолекул в эмульсии. — Ереван : Изд. АН Арм ССР, 1985.

Моделирование распространения крупномасштабных волн в морях и океанах с параллельной реализацией

Л .П. Каменщиков, Е. Д. Карепова, В. В. Шайдуров
Институт вычислительного моделирования СО РАН,
Красноярск

Для практики важно уметь точно и оперативно прогнозировать поведение крупномасштабных гравитационных поверхностных волн, возникающих в морях и океанах (цунами, приливные течения и т. п.). Данная задача в пределах приемлемой точности хорошо описывается уравнениями мелкой воды.

Пусть (r, λ, θ) — стандартная сферическая система координат с началом в центре земного шара. Далее будем использовать вместо угла θ географическую широту $\varphi = \pi - \theta$, так что $0 \le \varphi \le \pi$. Через λ обозначим географическую долготу $0 \le \lambda \le 2\pi$. Полагаем всюду $r = R_E$, где R_E — радиус Земли, который считается постоянным. Пусть Ω — заданная область на сфере с границей $\Gamma = \Gamma_1 \cup \Gamma_2$, где Γ_1 — часть границы, проходящая вдоль берега, а $\Gamma_2 = \Gamma$, Γ — часть границы, проходящая по морю. Обозначим через m_1 и m_2 — характеристические функции соответствующих участков границы. Для простоты считается, что точки $\varphi = 0$ и $\varphi = \pi$ (полюса) не входят в Ω. Относительно неизвестных функций $u = u(t, \lambda, \varphi)$, $v = v(t, \lambda, \varphi)$ и $\xi = \xi(t, \lambda, \varphi)$ запишем в $\Omega \times (0, T)$ уравнения баланса импульсов и уравнение неразрывности [1]:

$$\begin{aligned} \frac{\partial u}{\partial t} &= lv + mg\frac{\partial \xi}{\partial \lambda} - R_f u + f_1, \\ \frac{\partial v}{\partial t} &= -lu + ng\frac{\partial \xi}{\partial \varphi} - R_f v + f_2, \\ \frac{\partial \xi}{\partial t} &= m\left(\frac{\partial}{\partial \lambda}(Hu) + \frac{\partial}{\partial \varphi}\left(\frac{n}{m}Hv\right)\right) + f_3, \end{aligned} \tag{1}$$

где u, v — компоненты вектора скорости U по осям λ и φ соответственно; ξ — отклонение свободной поверхности от невозмущенного уровня; $H(\lambda, \varphi) > 0$ — глубина водоема в точке (λ, φ);

функция $R_f = r_+ |U| / H$ учитывает силу трения о дно, r_+ — коэффициент трения; $l = -2\omega\cos\varphi$ — параметр Кориолиса; $m = 1/(R_E \sin\varphi)$; $m = 1/R_E$; g — ускорение силы тяжести; $f_1 = f_1(t,\lambda,\varphi)$, $f_2 = f_2(t,\lambda,\varphi)$ и $f_3 = f_3(t,\lambda,\varphi)$ — заданные функции внешних воздействий.

Граничные условия рассмотрим в следующем виде:

$$HU_n + \beta m_2 \sqrt{gH}\xi = m_2 \sqrt{gH} d \text{ на } \Gamma \times (0,T), \tag{2}$$

где $U_n = U \cdot n$, $n = \left(n_1 \frac{n}{m} n_2\right)$ — вектор внешней нормали к границе; $\beta \in [0,1]$ — заданный параметр, $d = d(t,\lambda,\varphi)$ — граничная функция, которая либо задана на границе Γ_2 (прямая задача), либо подлежит определению одновременно с решением (тогда требуется задать дополнительные данные). Для корректной постановки задачи рассмотрим также начальные условия:

$$u(0,\lambda,\varphi) = u_0(\lambda,\varphi),\ v(0,\lambda,\varphi) = v_0(\lambda,\varphi),\ \xi(0,\lambda,\varphi) = \xi_0(\lambda,\varphi)$$

Для дискретизации по времени разобьем временной отрезок $[0,T]$на K интервалов: $0 = t_0 < t_1 < \ldots < t_K = T$ с шагом $\tau = T/K$, аппроксимируем производные по времени левыми разностями. В результате система (1)—(2) на каждом временном интервале (t_k, t_{k+1})имеет вид:

$$\left(\frac{1}{\tau} + R_f\right)u - lv - mg\frac{\partial \xi}{\partial \lambda} = f_1 + \frac{1}{\tau}u^k \text{ в } \Omega,$$

$$\left(\frac{l}{\tau} + R_f\right)v - lu - ng\frac{\partial \xi}{\partial \varphi} = f_2 + \frac{l}{\tau}v^k, \text{ в } \Omega, \tag{3}$$

$$\frac{1}{\tau}\xi - m\left(\frac{\partial}{\partial \lambda}(Hu) + \frac{\partial}{\partial \varphi}\left(\frac{n}{m}Hv\right)\right) = f_3 + \frac{1}{\tau}\xi^k \text{ в } \Omega$$

$$HU_n + \beta m_2 \sqrt{gH}\xi = m_2 \sqrt{gH} d \text{ на } \Gamma, k = 0,1,\ldots,K-1 \tag{4}$$

где $f(t_k,\lambda,\varphi) = f^k$, $f(t_{k+1},\lambda,\varphi) = f^{k+1} = f$, донное трение берется с предыдущего слоя по времени $Rf = r_+ |U_k| / H$. Индекс $k+1$ в разностных выражениях далее будем опускать там, где это не вызывает двойного толкования.

Систему (3) также можно записать в операторном виде:

$$L\Phi = F \tag{5}$$

где L — дифференциальный оператор системы (3), $F = \left(f_1 + \frac{1}{\tau} u^k, f_2 + \frac{1}{\tau} v^k, f_3 + \frac{1}{\tau} \xi^k \right)^T$.

Для решения системы (3)—(4) применим метод Галеркина. Для этого разобьем область Ω на некоторое число N_{el} треугольных элементов, т. е. $\Omega = \bigcup_{k=1}^{N_{el}} \Delta_k$. Пусть N_{nd} — число узлов полученной сетки. Рассмотрим множество базисных функций $\left\{ \Psi_j(\lambda, \varphi) \right\}_{j=1}^{N_{nd}}$, равных единице в j-ом узле, равных нулю во всех остальных узлах и линейных на каждом треугольном элементе. Для каждого фиксированного $t = t^{n+1}$ ищем решение в виде:

$$\Phi\left(\lambda, \varphi, t^{n+1}\right) = \sum_{j=1}^{N_{nd}} \alpha_j^{\Phi}\left(t^{n+1}\right) \Psi_j(\lambda, \varphi), \ \Phi = (u, v, \xi) \tag{6}$$

Неизвестные коэффициенты $\alpha_j^{\Phi}\left(t^{n+1}\right)$ определяем из требования ортогональности каждой из трех невязок $R_{\Phi}\left(\lambda, \varphi, t^{n+1}\right)$ всем базисным функциям:

$$\iint_{\Omega} R_{\Phi}\left(\lambda, \varphi, t^{n+1}\right) \Psi_j(\lambda, \varphi) d\Omega = 0, \ j = 1, ..., N_{nd} \tag{7}$$

В результате получаем систему линейных уравнений

$$A\Phi = B$$

относительно $3N_{nd}$ неизвестных $\left(\alpha_j^u, \alpha_j^v, \alpha_j^{\xi} \right)$. Для ее решение используем метод Гаусса-Зейделя:

$$\Phi_j^{(v+j)} = \frac{1}{a_{jj}} \left(b_j - \sum_{\substack{i \in Nb(j) \\ i<j}} a_{ij} \Phi_j^{(v+1)} - \sum_{\substack{i \in Nb(j) \\ i>j}} a_{ij} \Phi_j^{(v)} \right), \ j = 1, ..., 3N_{nd} \tag{8}$$

где $Nd(j)$ — множество узлов, состоящее из узла j и всех соседних к нему, т. е. соединенных с j-м узлом стороной кого-либо треугольника, v — номер итерации.

Для построения параллельного приложения используется параллелизм по данным, т. е. все множество узлов разбивается на непересекающиеся подмножества индексов j, и организуется па-

раллельное выполнение итерации в (8) для каждого подмножества. При вычислении сумм вида $\sum_{i \in Nb(j)}$, входящих в (8), некоторые из требующихся узлов могут оказаться на другом процессоре. Для доступа к ним на каждом процессоре организуются вспомогательные массивы, которые и играют роль соседних узлов, физически располагающихся в момент нахождения упомянутых сумм на другом процессоре. Сообщения между процессорами для обновления вспомогательных массивов проводятся перед очередной итерацией метода Гаусса-Зейделя.

Программы написаны на языке Fortran-DVM, разработанного в ИПМ им. М. В. Келдыша РАН. Расчеты проводились на кластере MVS-15000BM, построенном на основе 1148 процессоров PowerPC970 2.2 ГГц, объединенных коммуникационной сетью Myrinet, с пиковой производительностью 10102.4 Gflops, а на тесте Linpack, показавшей производительность 6645.53 Gflops.

В качестве примера параллельных расчетов рассмотрим решение исходной системы на сетке 201×201 с выполнением 1000 шагов по времени. Здесь Ω есть квадрат на земной сфере со стороной 2000 км. В табл. 1 приведены характеристики расчетов при разном числе процессоров, где использованы следующие обозначения: P — число процессоров, t_P — время счета на P процессорах; S — коэффициент ускорения; E — коэффициент эффективности.

Таблица 1

P	1	2	3	4	6	8	10	12	16
t_P, мин	26,71	14,45	9,57	7,51	5,30	4,21	3,57	3,18	2,66
$S = t_1 / t_P$	1,00	1,85	2,79	3,55	5,04	6,35	7,48	8,41	10,04
$E_1 = S / P$	1,00	0,92	0,93	0,89	0,84	0,79	0,75	0,70	0,63

Рис. 1 и 2 демонстрируют результаты решения прямой задачи о распространении локального возмущения в акватории Охотского моря[1].

[1] Сетки для Охотского моря и Мирового океана подготовлены С. Ф. Пятаевым и И. В. Киреевым

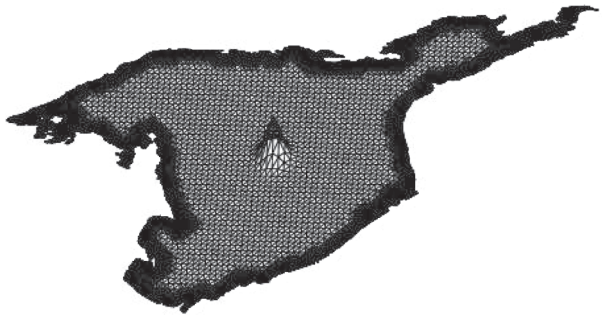


Рис. 1. Вид поверхности в момент $t = 0$

Поскольку в общем случае функция d неизвестна, то для замыкания задачи (3)—(4) следует задать дополнительное условие. Пусть Γ_0 — часть «жидкой» границы $(\Gamma_0 \subset \Gamma_2)$ с характеристической функцией m_0. Пусть на Γ_0 заданы данные наблюдений:

$$m_0 \xi = m_0 \xi_{obs}, \tag{9}$$

где ξ_{obs} — заданная функция. Тогда задача на усвоение данных может быть сформулирована следующим образом [1].

Пусть ξ_{obs} задана на Γ_0, функция d неизвестна на Γ_2 и равна нулю на Γ_1. Найти u, v, ξ, d, удовлетворяющие системе (3), граничному условию (4) и условию замыкания (9).

Эта задача имеет классическое решение в редких случаях, поэтому согласно [2] рассмотрим следующее семейство задач оптимального управления.

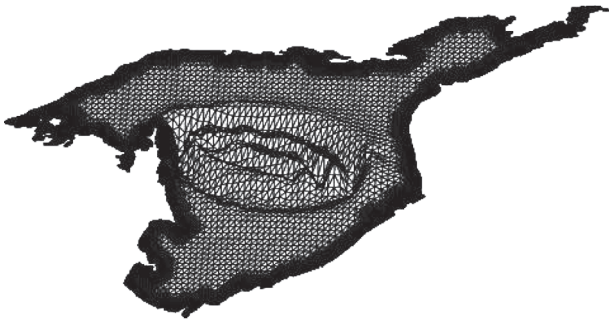


Рис. 2. Момент $t = 67$ мин.

Дано α_0. Пусть ξobs задана на Γ_0. Найти $\Phi_\alpha = (u_\alpha, v_\alpha, \xi_\alpha)$, d_α,

удовлетворяющие системе

$$L\Phi_\alpha = F \text{ в } \Omega$$

граничному условию

$$HU_{\alpha^n} + \beta m_2 \sqrt{gH}\xi_\alpha = m_2\sqrt{gH}d_\alpha \text{ на } \Gamma,$$

и минимизирующие функционал стоимости

$$J_\alpha\left(d_\alpha, \xi_\alpha\left(d_\alpha\right)\right) = \frac{1}{2} gR_E\left(\alpha\int_\Gamma m_2\sqrt{gH}d_\alpha^2 ds + \int_\Gamma m_0\sqrt{gH}\left(\xi_\alpha - \xi_{obs}\right)^2 ds\right)$$

Согласно [2] каждое решение d_α этой задачи удовлетворяет следующей системе вариационных уравнений:

$$L\Phi_\alpha = F \text{ в } \Omega,\ HU_{\alpha^n} + \beta m_2\sqrt{gH}\xi_\alpha = m_2\sqrt{gH}d_\alpha \text{ на } \Gamma, \quad (10)$$

$$L^+\hat{\Phi}_\alpha = 0 \text{ в } \Omega,\ H\hat{U}_{\alpha n} + \beta m_2\sqrt{gH}\hat{\xi}_\alpha = m_0\sqrt{gH}\left(\xi_\alpha - \xi_{obs}\right) \text{ на } \Gamma, \quad (11)$$

$$\alpha m_2 d_\alpha + m_2\hat{\hat{\xi}}_\alpha = 0 \text{ на } \Gamma \quad (12)$$

где L^+ — сопряженный к L оператор.

Для отыскания решения u^{k+1}, v^{k+1}, ξ^{k+1}, d^{k+1}, на каждом k-ом временном шаге в [1] предложено использовать итерационный алгоритм.

1. Зададимся некоторым $d_\alpha^{(0)}$ на Γ_2. Здесь и далее по алгоритму в верхнем индексе в скобках указывается номер итерации. Положим $u_\alpha^k = u^k, v_\alpha^k = v^k, \xi_\alpha^k = \xi^k$.

2. Пока

$$\left\|\gamma_l\left(\alpha d_\alpha^{(l)} + \hat{\xi}_\alpha^{(l)}\right)\right\| \geq \varepsilon \quad (13)$$

выполняется следующий шаг итерации. Здесь под нормой понимается дискретный аналог равномерной нормы, γ_p α — параметры метода, ε — заданная точность.

2.1. Решая прямую задачу (10), находим $u_\alpha^{(l)}, v_\alpha^{(l)}, \xi_\alpha^{(l)}$ (индекс $k+1$ опущен).

2.2. Используя $\xi_\alpha^{(l)}$, решаем обратную задачу (11) и находим $\hat{u}_\alpha^{(l)}, \hat{v}_\alpha^{(l)}, \hat{\xi}_\alpha^{(l)}$.

2.3. Используем (12) для итерационного уточнения $d_\alpha^{(l)}$:

$$d_\alpha^{(l+1)} = d_\alpha^{(l)} - \gamma_l \left(\alpha d\, d_\alpha^{(l)} + \hat{\xi}_\alpha^{(l)} \right). \qquad (14)$$

2.4. Переходим на пункт с $d_\alpha^{(l)} = d_\alpha^{(l+1)}$, $l = l + 1$.

После M шагов итерационного процесса будут определены $u^{k+1} \approx u_\alpha^{(M)}, v^{k+1} \approx v_\alpha^{(M)}, \xi^{k+1} \approx \xi_\alpha^{(M)}$.

Были проведены тестовые расчеты для акватории Мирового океана по следующей схеме.

(1) Задаем граничную функцию в виде $d = 2\sin\lambda$, где $0 \le \lambda \le 2\pi$.

(2) При таком d решаем прямую задачу до некоторого момента t_1 и запоминаем значение $\xi = \xi\big|_{\Gamma_2}$ на момент t_1.

(3) Приняв полученные значения $\xi = \xi\big|_{\Gamma_2}$ за данные наблюдений $\xi_{obs} = \xi\big|_{\Gamma_2}$, восстанавливаем функцию d согласно приведенному выше алгоритму (начиная с $d = 0$).

На рис. 3 изображены заданная d и восстановленная граничная функция d_{approx} после 300 итераций. Показано, что по мере роста числа итераций максимальный (по всей границе Γ_2) модуль разницы между восстановленной функцией d и заданной стремится к нулю.

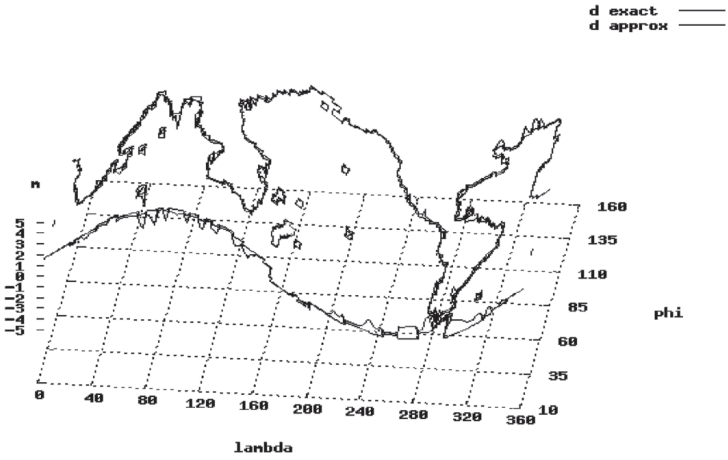


Рис. 3. Вид заданной и восстановленной функции d на границах Мирового океана

В табл. 2 показаны затраты времени на одном временном шаге при разном числе процессоров. Потребовалось 647 итераций для восстановления d с точностью 10^{-3} м.

Таблица 2

P	1	2	4
t_P, мин	902,98	479,06	275,07
$S = t_1 / t_P$	1,00	1,88	3,28
$E_1 = S / P$	1,00	0,94	0,82

Литература

1. **Agoshkov V. I.** Inverse problems of the mathematical theory of tides: boundary-function problem // Russ. J. Numer. Anal. Math. Modelling. — 2005. — Vol. 20. — № 1. — P. 1–18.

2. **Агошков В. И.** Методы оптимального управления и сопряженных уравнений в задачах математической физики. — М. : ИВМ РАН, 2003. — 256 с.

3. **Kamenshchikov L. P., Karepova E. D., Shaidurov V. V.** Simulation of surface waves in basins by the finite element method // Russian J. On Numerical Analysis and Mathematical Modelling — 2006. — Vol. 21. — № 4. — P. 305–320.

Опыт применения параллельных вычислений к проблематике шахтной вентиляции

И. М. Васенин, Б. Л. Петушкеев
Томский государственный университет, Томск

Одной из важных задач безопасности жизнедеятельности является мониторинг и прогноз газодинамической обстановки в горных выработках. С целью очистки воздушной среды и создания условий для нормальной работы людей применяется вентилирование. К тому же образующиеся в выработках угольных шахт участки обрушенного пространства являются потенциально опасным очагом выделения и скопления метана, образующего с воздухом взрывоопасную смесь.

Задача проектирования вентиляции в этих условиях для поддержания состава атмосферы, регламентированного нормативными документами, становится еще более актуальной и насущной. Задачи газовой динамики подобного типа являются, особенно в многомерном случае с большим количеством ячеек сетки, типичным примером требовательной к ресурсам компьютера области численного моделирования. Сокращения время расчета можно достичь за счет применения суперкомпьютеров и, соответственно, параллельного программирования.

В данной работе рассматривается трехмерная постановка задачи о переносе метана в обрушенном пространстве. Уравнения сохранения массы (1), импульса (2) и энергии (3) имеют вид:

$$\frac{\partial}{\partial t}\int_{\Omega}\varepsilon\rho d\Omega+\oint_{\Sigma}\varepsilon\rho\left(\vec{q}\cdot\vec{n}\right)d\Sigma=0; \qquad (1)$$

$$\frac{\partial}{\partial t}\int_{\Omega}\varepsilon\rho\vec{q}d\Omega+\oint_{\Sigma}\varepsilon\left[p\vec{n}+\left(\vec{q}\cdot\vec{n}\right)\vec{q}\rho\right]d\Sigma=\int_{\Omega}\vec{F}_c d\Omega+\int_{\Omega}pgrad\varepsilon d\Omega+\int_{\Omega}\varepsilon\rho\vec{g}d\Omega \qquad (2)$$

$$\frac{\partial}{\partial t}\int_{\Omega}\varepsilon\rho\left(e+\frac{|\vec{q}|^2}{2}\right)d\Omega+\oint_{\Sigma}\varepsilon\rho\left(\vec{q}\cdot\vec{n}\right)\left[e+\frac{p}{\rho}+\frac{|\vec{q}|^2}{2}\right]d\Sigma-\int_{\Omega}\varepsilon\rho\left(\vec{q}\cdot\vec{n}\right)d\Omega=0, \qquad (3)$$

где $\vec{q}$ — вектор скорости, ε — пористость среды, $\vec{g}$ — ускорение силы тяжести, $\vec{F}_c$ — вектор объемных сил сопротивления, взятого в форме Эргуна [1]. Слагаемое $\int_{\Omega} pgrad\varepsilon d\Omega$ в правой части уравнения движения появляется в результате учета импульса дав-

ления со стороны пористой среды, слагаемое $\int_{\Omega} \varepsilon \rho \vec{g} d\Omega$ учитывает силу тяжести. В уравнении энергии член $\int_{\Omega} er(\vec{g} \cdot \vec{q}) d\Omega$ отвечает за влияние силы тяжести на энергию объема двухфазной среды.

Уравнение переноса запишем в форме уравнения неразрывности для метана:

$$\frac{\partial}{\partial t} \int_{\Omega} \varepsilon c \rho d\Omega + \oint_{\Sigma} \varepsilon c \rho (\vec{q} \cdot \vec{n}) d\Sigma = 0, \tag{4}$$

где c — концентрация метана.

Для замыкания системы уравнений нужно добавить уравнение состояния вида:

$$e = \frac{1}{k-1} \frac{p}{\rho}, \; k = \frac{C_{pм} c + C_{pв}(1-c)}{C_{vм} c + C_{vв}(1-c)}, \tag{5}$$

где индекс «*в*» относится к воздуху, а «*м*» означает метан; C_p и C_v — удельные теплоемкости при постоянном давлении и объеме соответственно.

В работе используются реальные геометрические размеры выработанного пространства, происходит учет различного рода скважин и штреков вентилирования, а также особенностей метановыделения, что закономерно увеличивает достоверность прогноза. Физическая область расчетной задачи показана на Рис.1.

Геометрические размеры следующие:

- длина — 1600 м (координата x),
- ширина — 400 м (координата z),
- высота — 12м.

На рис. 1 введены обозначения: 1 — конвейерный штрек, 2 — вентиляционный штрек, 3 и 4 — газоотсасывающие скважины, 5 — сбойка на газодренажный штрек.

Задание пористости среды определялось свойством прогиба свода кровли по направлениям обоих координат и, таким образом, представлялась функцией $f(x,y)$.

Для решения поставленной задачи используется явный численный метод Годунова первого порядка точности [2], показавшего свою надежность в нестационарных задачах газовой динамики. Расчет проводится до момента равенства общего прихода и расхода газов (воздух, метан) с точностью до нескольких процентов. Шаг по времени для простоты считался постоянным.

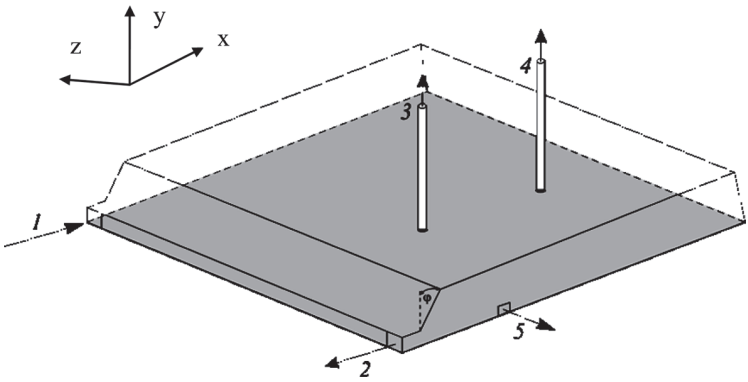


Рис. 1. Граничные условия задавались следующим образом: 1 — через канал 1 происходит вдув воздуха с постоянной скоростью $U = 0{,}67$ м/с, вторым условием является значение энтропии для воздуха при стандартном давлении $P0 = 101325$ Па. 2 — канал 2 — выпускной, поэтому достаточно определения давления на выходе, принятое равным $P0$. 3 — элементы 3, 4 являются вытяжками: на всех задавалось значение энтропии S при давлении $P0$ для метана. Скорость отсоса вентиля 4 — U1 = 0,3 м/с, вентиля 3 также $U1$. 4 — приток метана происходит с «потолка» выработки таким образом, что максимум имеет место при x = 50 м, экспоненциально убывая по длине. При этом суммарный объем метановыделения составляет V = 52 м^3/с. Данные получены на основании проведенных экспериментов горными инженерами. 5 — Сбойка 5 была заглушена и U = 0 м/с. 6 — На всех других границах действуют условия непротекания: U = 0.

Задание начальных данных предполагает, что объем покоится: $\vec{q} = 0$ и его снизу наполовину высоты пространства заполняет метан: c = 1, остальное — воздух (в целях уменьшения времени установления). Распределение давления отражает факт изменения давления с высотой, а распределение плотности находится по уже заданным величинам с помощью уравнения состояния.

Также в работе предложен метод адаптации вычислительной схемы Годунова на компьютер с массивно-параллельной архитектурой, показавший свою эффективность. Алгоритм осуществлен на основе библиотеки передачи данных MPI и языка C++. Расчеты проводились на кластере СКИФ Cyberia Томского государственного университета с привлечением до нескольких сотен процессоров. На рис. 2 приведены результаты исследования

реализованной программы на масштабируемость в двух случаях используемой топологии процессов.

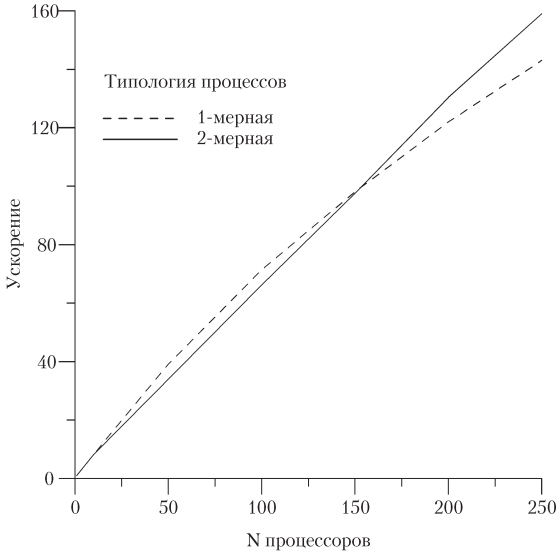


Рис. 2

На рис. 3 показана концентрация метана в горизонтальных сечениях на различных высотах в момент времени 500 ч. Характерно присутствие на уровне 1 м четко выраженной вблизи скважины 3 тянущейся вплоть до стенки струйки газа с малым содержанием метана, причем у второй скважины такой же мощности подобного не наблюдается. Такое движение за время порядка нескольких десятков часов становится ярко выраженным и, по-видимому, вызвано наличием внизу течения, сносящего область пониженного давления на некоторое расстояние от оси действия верхнего насоса. Чуть выше (средний график на рис. 3) это воздействие уже не так остро выражено, но все же заметно смещение в сторону от входного штрека. Изолинии концентрации метана на нижнем графике рис. 3 имеют, как и ожидалось, продольный по координате x характер, отчетливо видна работа обоих насосов. К тому же в удаленной от лавы области, где действие насосов практически не сказывается, явно выделилась обширная зона повышенного содержания метана.

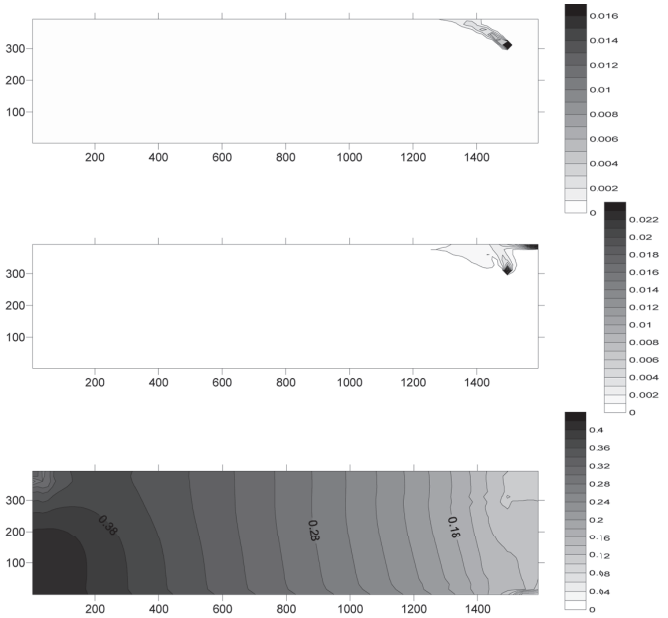


Рис. 3. Концентрация метана на высоте 1 м, 5 м, 11 м (сверху вниз) на момент времени 500 ч (установившееся распределение)

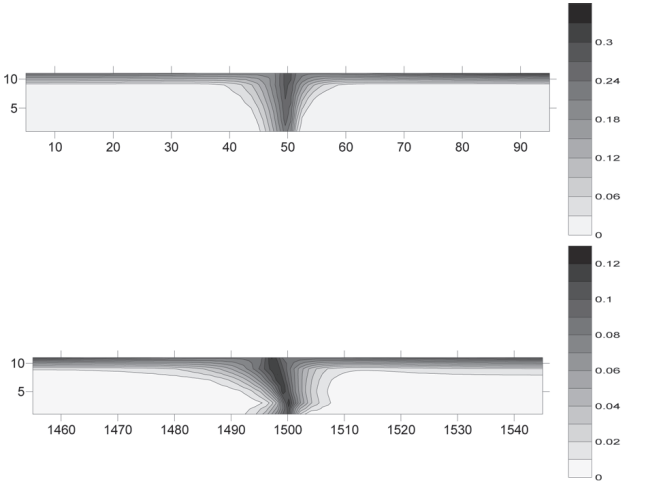


Рис. 4. Концентрация метана в сечении скважин (момент времени 500 ч)

В сечениях по оси z на рис. 4 приведены профили концентрации метана в местах расположения обоих насосов на момент окончания расчета. Можно отметить закономерный факт скопления метана в верхней части выработанного пространства, что естественно в силу меньшей плотности по сравнению с воздухом, и достижение нижнего основания зоной отсоса скважин. Последнее явление экспериментально не подтверждается, поскольку в этом случае имеет место эффект уплотнения тяжелыми и плотными обрушенными породами нижнего настила, тем самым не позволяя проникать посторонним примесям. Отметим также важный факт непревышения допустимой концентрации метана, равной 2 %, на выходном сечении 2.

Представленная модель может быть использована для оперативного прогноза качества воздуха в выработках, поскольку ее применение обеспечивает высокую скорость получения результатов при достаточно хорошей точности предсказания. На данный момент при поддержке Минестерства Энергетики РФ ведется работа с целью создания программного комплекса, направленного на получение рекомендаций к созданию эффективных или доработке текущих систем вентиляции с учетом горно-геологических условий на шахтах России.

Литература

1. **Справочник по теплообменникам** : В 2 т. — М. : Энергоатомиздат, 1987. — Т. 1. — 561 с.

2. **Годунов С. К., Забродин А. В., Иванов М. Я.** Численное решение многомерных задач газовой динамики. — М. : Наука, 1976. — 333 с.

Разработка параллельного алгоритма LU-SGS для решения многомерных задач вычислительной газодинамики

И. В. Семенов, И. Ф. Ахмедьянов
Институт автоматизации проектирования РАН, Москва

Введение

Современные многомерные задачи вычислительной газовой динамики требуют значительного количества вычислительных ресурсов. Решение таких задач за приемлемое время невозможно без использования высокопроизводительных параллельных компьютеров. Для целого ряда задач, например стационарных, применение неявных численных схем даёт возможность существенно сократить время расчета. Но переход к неявной схеме вычислений ведет к существенному усложнению построения алгоритма распараллеливания. Прежде всего, это связано с тем, что в результате линеаризации возникает система линейных уравнений большого порядка, параллельное решение которой сопряжено с рядом трудностей.

Представленная работа посвящена разработке параллельного алгоритма Lower Upper — Symmetric Gauss-Seidel для сильносвязанных кластеров с распределенной памятью. Последовательный вариант алгоритма описан в работе [1]. В работе [2] в качестве параллельного алгоритма LU-SGS предлагается гибридный алгоритм, который соответствует последовательному алгоритму только в определенных подобластях исходной расчетной области. В данной работе при распараллеливании осуществляется строгое соблюдение последовательного алгоритма LU-SGS.

Последовательный алгоритм

Будем называть ячейкой сетки выпуклый многогранник и значения искомых функций в области, ограниченной этим многогранником. Естественным образом с каждой ячейкой сопоставляется список ее непосредственных геометрических соседей. Солвер — функция, в качестве параметров принимающая некоторую текущую ячейку с ее геометрическими соседями,

и модифицирующая значения искомых функций в текущей ячейке и соседних. Обход — процесс последовательного применения солвера к ячейкам сетки. Под обходом так же будем понимать некоторую последовательность ячеек сетки.

Таким образом, если рассмотреть алгоритм LU-SGS как объект для распараллеливания, то его последовательный вариант выглядит как два прохода (прямой и обратный) по ячейкам расчетной области. Оба прохода осуществляются по одной последовательности ячеек, но в разных направлениях.

В общем случае значения искомых функций в ячейке на следующем временном слое зависят от всех ячеек расчетной области на текущем временном слое, что затрудняет построение параллельного алгоритма.

Параллельный алгоритм

Построим параллельный алгоритм LU-SGS для сильносвязанных кластеров с распределенной памятью.

Так как обходить солвером расчетную область можно в любом порядке (не обязательно, чтобы следующая ячейка в обходе была геометрическим соседом предыдущей), то построение параллельного алгоритма LU-SGS сводится к выбору последовательного обхода области, который можно корректно декомпозировать на параллельные обходы. То есть обходы, выполняющиеся одновременно на разных процессорах кластера в распределенной памяти. Результат параллельного выполнения таких обходов идентичен результату последовательного.

В данной работе сначала строятся параллельные обходы подобластей, после чего показывается возможность сведения их в один обход. Структура параллельных обходов эффективно отображается на распределенную память.

Разбиение области (поддомены, внутренние/внешние ячейки)

Все множество ячеек области разбивается на N подмножеств, где N — количество участвующих в расчете процессоров кластера. Каждое такое подмножество назовем поддоменом.

Внутри каждого из построенных поддоменов ячейки сетки разбиваются на два класса. Внутренние ячейки — ячейки, не име-

ющие общих граней с ячейками других поддоменов, и внешние ячейки — ячейки, имеющие хотя бы одну общую грань с ячейкой из другого поддомена. Ячейка, не имеющая общую грань с ячейкой другого поддомена, но имеющая грань-границу расчетной области считается внутренней.

Изолированность обходов

Два обхода называются изолированными, если множества их ячеек являются подмножествами разных поддоменов расчетной области, и результат их параллельного расчета в распределенной памяти идентичен результату последовательного расчета в единой памяти.

Пусть K_1 и K_2 — два множества ячеек. $N(K) = K \cup N(K)$, где $N(K)$ — множество, состоящее из всех геометрических соседей ячеек из K. Пусть $N(K_1) \cap N(K_2) = \varnothing$.

Так как во время расчета ячеек из K_1 изменению могут подвергнуться только параметры в ячейках из $N(K_1)$, аналогично при расчете K_2 измениться могут только параметры в ячейках из $N(K_2)$, и $N(K_1) \cap N(K_2) = \varnothing$, то два обхода K_1 и K_2 являются изолированными.

M обходов называются изолированными, если они являются попарно изолированными.

Построение разбиения множества всех поддоменов

Необходимо построить разбиение множества всех поддоменов Q на Q_i такое, что внешние ячейки поддоменов из множества Q_i образуют параллельные изолированные обходы (это означает, что ячейки поддоменов из Q_i можно считать параллельно).

Назовем два поддомена i и j зависимыми, если существуют различные ячейки a_1, a_2 и a_3 такие, что:

a_1 – геометрический сосед a_2, a_2 — геометрический сосед a_3;

a_1 принадлежит поддомену i, a_3 принадлежит поддомену j.

Из определения следует, что любые два параллельных обхода внутри независимых поддоменов являются изолированными.

Построим функцию $G(M)$. Пусть M, D — некоторые множества поддоменов области, $R(M,D)$ – функция, возвращающая поддомен с наименьшим порядковым номером из множества M,

такой, что он независим от любого поддомена из множества D (порядок нумерации поддоменов может быть любой).

Пусть M_0 некоторое непустое множество поддоменов, построим множества M_i следующим образом:

$$M_1 = \{R(M_0, \varnothing)\}, M_2 = M_1 \cup \left\{R\left(M_0 \Big/ M_1, M_1\right)\right\}, ...,$$

$$..., M_q = M_{q-1} \cup \left\{R\left(M_0 \Big/ M_{q-1}, M_{q-1}\right)\right\}$$

Построение множеств останавливается, когда функция R не может найти поддомен, удовлетворяющий ее условиям.

Тем самым построена функция $G(M_0) = M_q$, которая выбирает из множества M_0 некоторое достаточно большое подмножество попарно независимых поддоменов M_q. Из непустого множества функция G выбирает хотя бы один узел.

Алгоритм построения Q_i — следующий:

$$Q_1 = G(Q);$$

$$Q_2 = G\left(Q \Big/ Q_1\right);$$

...

$$Q_n = G\left(Q \Big/ (Q_1 \cup ... \cup Q_{n-1})\right);$$

$$Q_{n+1} = G\left(Q \Big/ (Q_1 \cup ... \cup Q_n)\right) = G(\varnothing) = \varnothing$$

Построение множеств останавливается, когда $Q_{n+1} = \varnothing$, что эквивалентно $Q_1 \cup ... \cup Q_n = Q$. Такой момент всегда наступает, так как множество поддоменов Q конечно, а каждая итерация добавляет в множество $Q_1 \cup ... \cup Q_i$ хотя бы один поддомен. Таким образом построено искомое разбиение, удовлетворяющие следующим условиям:

$$\cup Q_i = Q$$

$$Q_i \cap Q_j = \varnothing, i \neq j$$

Внешние ячейки поддоменов из множества Q_i образуют параллельные изолированные обходы. Порядок обхода внешних ячеек внутри одного поддомена не важен.

Идея поэтапного вычисления

Базовой идеей описываемого алгоритма является двухэтапный расчет, когда на первом этапе производится расчет внутренних ячеек области, на втором внешних. Таким образом, для расчета первого этапа не требуются каких-либо пересылок ячеек между узлами кластера. На первом этапе расчета организуется N параллельных обходов расчетной области, с тем условием что эти обходы идут только по внутренним ячейкам поддоменов (порядок внутри каждого поддомена не важен). По определению эти N обходов являются изолированными.

Здесь можно предложить простейший вариант параллельного алгоритма, когда для расчета второго этапа вся расчетная область собирается на одном процессоре. На этом процессоре последовательно рассчитываются внешние ячейки всех поддоменов, после чего измененные ячейки рассылаются на соответствующие узлы. Однако в этом случае, если внешних ячеек всего 5 % от общего количества ячеек, то время работы параллельной реализации составит не менее 5 % от времени работы последовательной. Что соответствует максимальному ускорению не более чем в 20 раз (закон Амдала).

Расчет второго этапа

Хотя на первом этапе производится параллельный расчет внутренних ячеек, последовательный расчет второго этапа сильно снижает эффективность параллельного алгоритма. Поэтому расчет второго этапа производится итерациями $P_1, ..., P_n$. Во время итерации P_i параллельно рассчитываются внешние ячейки поддоменов из множества Q_i. Это означает, что процессоры соответствующие поддоменам из множества Q_i на i-ой итерации обходят солвером свои внешние ячейки. Процессоры соответствующие поддоменам из $Q \setminus Q_i$ простаивают.

Сведение построенных обходов в один

Так как итерация P_i — это расчет некоторого количества изолированных обходов, то ее можно подменить последовательным обходом внешних ячеек поддоменов из множества Q_i. Расчет первого этапа алгоритма также эквивалентен некоторому после-

довательному обходу множества всех внутренних ячеек. Объединение $n+1$ последовательных обходов в один тривиально. Таким образом, найдена параллельная декомпозиция некоторого последовательного обхода области.

Обратный обход

Обратный обход строится следующим образом. Рассчитываются итерации $P_1, ..., P_n$, после чего рассчитывается первый этап алгоритма с обходом ячеек в обратном порядке. Внутри каждого параллельного обхода ячейки обрабатываются в обратном порядке. Нетрудно заметить, что сведение этих параллельных обходов в один даст корректный обратный обход области. Следовательно, предложенный алгоритм позволяет параллельно считать обратный обход.

Пересылки между процессорами кластера

Как уже сказано выше, пересылки необходимы только во время расчета второго этапа алгоритма. Так как итерации P_i выполняются последовательно, то достаточно привести стратегию пересылок ячеек между процессорами кластера внутри одной итерации. Для итерации P_i назовем процессоры, соответствующие множеству Q_i, ведущими, а соответствующие множеству $Q \backslash Q_i$, — ведомыми.

С точки зрения обменов расчет итерации P_i разбивается на 3 стадии:

- ведущие процессоры получают от ведомых необходимые ячейки;
- ведущие процессоры обходят свои внешние ячейки солвером, ведомые процессоры простаивают;
- ведущие процессоры передают обратно измененные ячейки.

Необходимые ячейки — это внешние ячейки ведомых процессоров, являющиеся геометрическими соседями ячеек, которые планируется рассчитать во время итерации P_i.

Из построения множеств Q_i следует, что в пределах итерации P_i ни одна ячейка ведомых процессоров не может быть передана более чем одному ведущему процессору.

Идеальное теоретическое ускорение

Рассмотрим какое максимальное ускорение возможно при использовании данного алгоритма на идеальном кластере, где время передачи данных равно нулю.

Пусть расчетная область равномерно разбита на N подобластей и в каждой подобласти доля внешних ячеек к внутренним равна α.

Тогда для расчета первого этапа алгоритма требуется

$$\frac{(1-\alpha)t}{N}$$

t — время расчета всех ячеек в последовательном алгоритме, $\alpha t/N$ — время расчета внешних ячеек в одном поддомене. Пусть n — это количество итераций второго этапа алгоритма. Так как внутри итерации поддомены из множества Q_i рассчитываются параллельно, то время расчета n итераций составляет

$$\frac{\alpha t}{N} n$$

что равно времени расчета второго этапа. Стоит отметить, что это время не зависит от количества процессоров, участвующих в расчете итерации.

Таким образом теоретическое ускорение можно оценить как

$$\frac{N}{1+\alpha(n-1)}$$

Оценим значение параметра α для трехмерной области при $1 \le N \le M/8$. Пусть область равномерно разбита на кубические поддомены и количество ячеек в области равно M. В каждом поддомене одинаковое количество ячеек. Тогда длину ребра поддомена можно оценить как $\sqrt[3]{M/N}$, в поддоменах без граничных ячеек внешних ячеек будет $6\left(\sqrt[3]{M/N}-1\right)^2$, количество внешних ячеек области $6\left(\sqrt[3]{M}-1\right)^2+2$ и

$$\alpha = \frac{6N\left(\sqrt[3]{M/N}-2\right)^2 + 2N - 6\left(\sqrt[3]{M}-2\right)^2}{M}$$

Предположив, что допустимой считается утилизация ресурсов кластера не ниже 50 % и $n=7$ — характерное значение для трехмерных задач, получаем нижнюю оценку для количества ячеек на один процессор кластера в 45 000 ячеек.

Нетрудно заметить, что в асимптотике, когда $\alpha \to 1$, идеальное ускорение стремится к N/n.

Заключение

Реализация предложенного алгоритма хорошо себя зарекомендовала в проводимых расчетах [3]. Возможно улучшение алгоритма, если строить множества Q_i не на основе внешних ячеек, а на основе более мелких групп ячеек, что позволит повысить параллелизм.

Работа выполнена при поддержке РФФИ грант № 05-08-50115-а и гранта Президента РФ № МК-5068.2007.1

Литература

1. **Menshov, Nakamura Y.** Hybrid Explicit-Implicit, Unconditionally Stable Scheme for Unsteady Compressible Flows // AIAA Journal. — 2004. — Vol. 42. — № 3. — P. 551—559.

2. **Sharov D., Luo H., Baum J. D., Loehner R.** Implementation of unstructured grid GMRES-SGS method on shared-memory, cache-based parallel computers // AIAA-2000-927 — Aerospace Sciences Meeting and Exhibit, 38th, Reno, NV, Jan. — 2000. — P. 10—13.

3. **Семенов И. В., Ахмедьянов И. Ф., Уткин П. С.** Разработка вычислительного комплекса для решения двух и трехмерных задач газодинамики реагирующих течений на многопроцессорных ЭВМ // Высокопроизводительные параллельные вычисления на кластерных системах : Мат-лы 6 Международ. науч.-практ. семинара. 12–17 декабря 2006 г. — СПб. : Изд-во Санкт-Петербургского госуниверситета, 2007. — Т. 2. — С. 138—145.

Численный метод решения уравнений Навье-Стокса на многопроцессорных вычислительных системах[1]

А. О. Есаулов, А. В. Старченко
Томский государственный университет, Томск

Во всех современных методах решения уравнений гидродинамики более 50 % вычислительных затрат алгоритмов занимает интегрирование уравнения Пуассона для определения давления (в переменных «скорость-давление») или функции тока (в переменных «функция тока-завихренность»). В связи с этим в данной работе построен новый эффективный метод решения уравнений Навье-Стокса, в котором на конечно-разностном уровне проведено исключение сеточной функции давления.

Рассматривается система уравнений Навье-Стокса, описывающая изотермическое движение вязкой несжимаемой жидкости [1, 2] в двумерной области:

- уравнение неразрывности:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0; \tag{1}$$

- уравнения движения:

$$\begin{aligned} u\frac{\partial u}{\partial x} + \nu\frac{\partial u}{\partial y} &= \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{1}{\rho}\frac{\partial p}{\partial x}; \\ u\frac{\partial v}{\partial x} + \nu\frac{\partial v}{\partial y} &= \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{1}{\rho}\frac{\partial p}{\partial y} \end{aligned} \tag{2}$$

- уравнения состояния:

$$\nu = \text{const}, \ \rho = \text{const} \tag{3}$$

где u, v — горизонтальная и вертикальная проекции вектора скорости; ν — коэффициент кинематической вязкости; ρ — плотность; p — давление.

[1] Работа выполнена при финансовой поддержке РФФИ, грант № 07-05-01126.

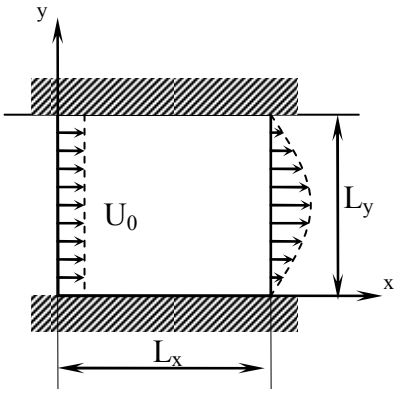


Рис. 1. Канал

Предлагаемый алгоритм решения двумерных уравнений гидродинамики строится на примере ламинарного внутреннего течения вязкой несжимаемой жидкости в прямоугольной области. Требуется выполнение условий прилипания частиц жидкости на твердой стенке и непротекания на ограничивающих течение твердых поверхностях. Имеет место простейший случай изотермического движения вязкой несжимаемой жидкости с постоянными значениями плотности и коэффициента вязкости. Граничные условия в данном случае задаются следующим образом:

$$\begin{aligned} &x = 0 \ : \ \mathrm{u} = U_o; \mathrm{v} = 0; \ \ x = \mathrm{L_x} \ : \ \frac{\partial \mathrm{u}}{\partial \mathrm{x}} = 0; \mathrm{v} = 0; \\ &y = 0 \ : \ \mathrm{u} = \mathrm{v} = 0; \ \ y = \mathrm{L_y} \ : \mathrm{u} = 0; \mathrm{v} = 0; \end{aligned} \qquad (4)$$

Получение конечно-разностного аналога исходной системы дифференциальных уравнений (1)—(4) производится методом конечного объема на шахматной сетке. В центре каждого конечного объема рассчитывается давление, а на его гранях — компоненты скорости [3]. Путем интегрирования по конечному объему получаются конечно-разностные аналоги уравнений (1)—(2).

При аппроксимации конвективных членов используется противопотоковая разность первого порядка, диффузионных членов — центрально-разностная схема. При интегрировании по смещенным конечным объемам системы (2) получаются сеточ-

ные уравнения, содержащие (u, v)-переменные, а также разности значений функции давления в соседних конечных объемах [3]:

$$ae_{i+1j}\left(u_{i+1j}-u_{i+2j}\right)+aw_{i+1j}\left(u_{i+1j}-u_{ij}\right)+an_{i+1j}\left(u_{i+1j}-u_{i+1j+1}\right)+$$
$$+as_{i+1j}\left(u_{i+1j}-u_{i+1j-1}\right)=hy_j\cdot\left(p_{ij}-p_{i+1j}\right); \quad (5a)$$
$$i=1,Nx-2;\quad j=1,Ny-1;$$

$$be_{ij+1}\left(v_{ij+1}-v_{i+1j+1}\right)+bw_{ij+1}\left(v_{ij+1}-v_{i-1j+1}\right)+bn_{ij+1}\left(v_{ij+1}-v_{ij+2}\right)+$$
$$+bs_{ij+1}\left(v_{ij+1}-v_{ij}\right)=hx_i\cdot\left(p_{ij}-p_{ij+1}\right); \quad (5b)$$
$$i=1,Nx-1;\quad j=1,Ny-2;$$

$$ae_{i+1j+1}\left(u_{i+1j+1}-u_{i+2j+1}\right)+aw_{i+1j+1}\left(u_{i+1j+1}-u_{ij+1}\right)+$$
$$+an_{i+1j+1}\left(u_{i+1j+1}-u_{i+1j+2}\right)+$$
$$+as_{i+1j+1}\left(u_{i+1j+1}-u_{i+1j}\right)=hy_{j+1}\cdot\left(p_{ij+1}-p_{i+1j+1}\right); \quad (5c)$$
$$i=1,Nx-2;\qquad j=0,\ Ny-2;$$

$$be_{i+1j+1}\left(v_{i+1j+1}-v_{i+2j+1}\right)+bw_{i+1j+1}\left(v_{i+1j+1}-v_{ij+1}\right)+$$
$$+bn_{i+1j+1}\left(v_{i+1j+1}-v_{i+1j+2}\right)+$$
$$+bs_{i+1j+1}\left(v_{i+1j+1}-v_{i+1j}\right)=hx_{i+1}\cdot\left(p_{i+1j}-p_{i+1j+1}\right); \quad (5d)$$
$$i=0,Nx-2;\qquad \mathrm{j}=1,Ny-2.$$

Давление исключается при комбинации (5a) – (5b) – (5c) + (5d) дискретных аналогов уравнений движения для четырех соседних контуров. С помощью уравнения неразрывности из получившегося соотношения также можно исключить вертикальную компоненту скорости v. В итоге получается система сеточных уравнений:

$$A_{i+1j}u_{i+1j}+B_{i+1j}u_{i+2j}+C_{i+1j}u_{ij}+D_{i+1j}u_{i+1j+1}+E_{i+1j}u_{i+1j-1}+$$
$$F_{i+1j}u_{i+2j+1}+G_{i+1j}u_{ij+1}+H_{i+1j}u_{i+1j+2}+\sum_{m=-1}^{3}\sum_{k=1}^{j}I^m_{i+1j}u_{i+m,k}=0; \quad (6)$$
$$i=1,Nx-2,\ j=1,Ny-2.$$

В качестве алгоритма решения исходной задачи предлагается следующая последовательность действий:

1. Задание начальных распределений и граничных условий.

2. Расчет коэффициентов A_{i+1j}, B_{i+1j}, C_{i+1j}, D_{i+1j}, E_{i+1j}, F_{i+1j}, G_{i+1j}, H_{i+1j}.

3. Расчет горизонтальной компоненты скорости u из СЛАУ (6). (7)

4. Расчет вертикальной компоненты v.

5. Проверка невязки и ошибки; если они велики, то возврат к п. 2.

Пункт 4 алгоритма представляет собой расчет компоненты скорости v по столбцам методом прогонки из следующих сеточных уравнений:

$$v_{ij+1} - 2v_{ij} + v_{ij-1} = \frac{\left(hy_j\left(u_{i+1j} - u_{ij}\right) - hy_{j-1}\left(u_{i+1j-1} - u_{ij-1}\right)\right)}{hx_i}; \quad (8)$$

$$i = 1, Nx;\ \ j = 2, Ny - 1;$$

Полученная система (6) удовлетворяет одновременно уравнениям движения и неразрывности и связывает значения сеточной функции скорости во всей расчетной области. Однако данная система уравнений является плохообусловленной и при больших размерностях разностной задачи в общем случае требует разработки специальных методов решения.

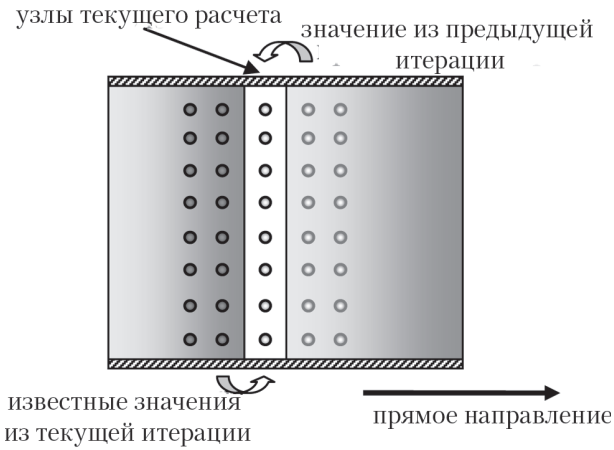


Рис. 2. Схема маршевого метода

В данной работе для отрывного течения с преобладающим направлением потока построен итерационный «маршевый» алгоритм расчета, позволяющий в несколько раз быстрее получить результат по сравнению с известным алгоритмом SIMPLE Патанкара для решения уравнений Навье-Стокса [3]. Суть этого алгоритма заключается в последовательном расчете по столбцам (рис. 2) значений сеточной функции скорости u во всей расчетной области (значения переменных из соседних столбцов берутся из текущей или предыдущей итерации). Таким образом, пункт 3 алгоритма (7) для нахождения поля скорости u по формуле (6) заменяется на схему, дающую приближение к точному решению этих сеточных уравнений. Подобная замена замедляет сходимость алгоритма в целом, но упрощает его программную реализацию и требования к вычислительным ресурсам.

Проверка правильности работы предложенного метода и адекватности построенной модели проводилось путем сравнения результатов расчетов с приближенным аналитическим решением в случае развивающегося течения в начальном участке канала, а также на примере сопоставления численного и физического экспериментов [5] течения жидкости в канале с внезапным расширением (рис. 3). Во всех случаях наблюдается хорошее согласование результатов, что говорит о корректности проводимых расчетов.

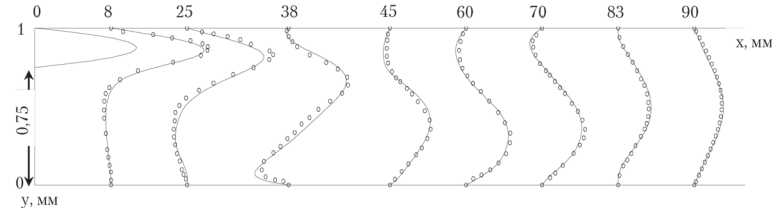


Рис. 3. Профили продольной скорости для канала с расширением. Re = 142, U_0 = 43,9 см/с, ○ — экспериментальные данные, ∿ — результаты расчетов

В табл. 1 приведено время расчетов последовательных алгоритмов на различных сетках, проведенных на одном вычислительном узле кластера ТГУ СКИФ Cyberia.

Предложенный «маршевый» алгоритм эффективно распараллеливается на многопроцессорных системах с распределенной памятью с использованием конвейерной схемы организации

параллельных вычислений. Учитывая специфику применяемого алгоритма, сложность вычислительного шаблона, а также тот факт, что для нахождения значений в одном столбце горизонтальной скорости применяется прямой метод решения СЛАУ, эффективной может быть только одномерная декомпозиция расчетной области поперек направления доминирующего течения. На рис. 4 схематично изображена одномерная декомпозиция, а также значения сеточной функции *u*, которыми должны обмениваться соседние параллельные процессы.

Таблица 1

Сравнение времени счета алгоритмов на различных сетках (в секундах)

	16×16	**32×32**	**48×48**	**64×64**	**80×80**
Simple	0,25	2,14	7,42	29,91	109,59
Маршевый метод	0,00648	0,22	1,6	5,99	16,63

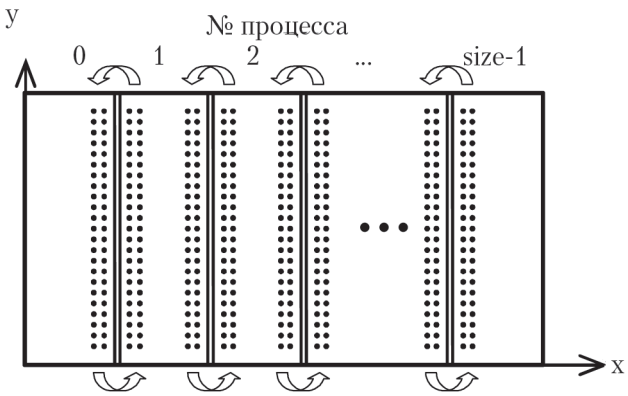


Рис. 4. Схема одномерной декомпозиции расчетной области

В ходе одного шага работы параллельного алгоритма каждый процесс проводит «маршевый» просчет горизонтальной компоненты скорости, обменивается граничными значениями с соседними процессами, затем вычисляет вертикальную компоненту скорости и пересчитывает коэффициенты системы (6). Если в качестве начального приближения взять нулевое поле скоростей, т. е. отсутствие движения во всем канале, то расчеты в каждой по-

добласти будут давать также нулевое поле скоростей в случае нулевых граничных условий на левой и правой границе. Так как на выходе канала заданы мягкие граничные условия, то на распространение возмущения скорости влияет только граничное условия на входе канала. Таким образом, за первый шаг итерационное приближение получает только один процесс, за второй шаг — два, за третий шаг — три и т. д. Шаги работы параллельной программы до момента, когда возмущение достигнет последней расчетной подобласти соответствует загрузке конвейера. Следующим этапом является параллельный расчет приближенного решения каждым процессом в своей подобласти при равномерном распределении вычислений между всеми вычислительными процессами.

Подобная «конвейерная» организация параллельных вычислений практически повторяет все операции исходного последовательного алгоритма и требует близкого числа глобальных итераций. Замедление ускорения расчетов связано лишь с затратами на обмены данными между процессами. Таким образом, итерационный в силу нелинейности исходной задачи характер нахождения решения позволил создать эффективный параллельный алгоритм.

Тестирование параллельной программы проводилось на кластере ТГУ СКИФ Cyberia на примере задачи о течении жидкости в канале с внезапным расширением и показало значительное ускорение расчетов. На рис. 5. приведен график ускорения работы параллельного алгоритма на различных сетках. Из графика можно сделать вывод о том, что распараллеливание эффективно даже при небольших размерах сетки. Ускорение при $N = 100$ получилось даже несколько больше, чем при $N = 200$, что объясняется, возможно, более эффективным использованием КЭШ-памяти вследствие небольшого объема вычислительной информации на каждом вычислительном процессе.

В целом заметна закономерность, заключающаяся в том, что с увеличением размерности сетки ускорение будет все ближе приближаться к идеальному (штриховая линия на рис. 5). Действительно, с увеличением загрузки каждого процесса доля межпроцессорных обменов в общем объеме временных затрат уменьшается, в то же время параллельные программы практически не требуют дополнительных итераций метода в сравнении с последовательной версией.

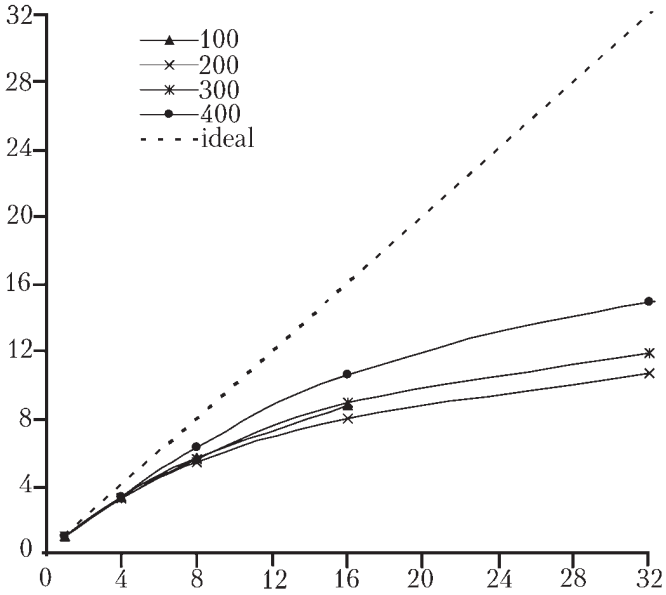


Рис. 5. Ускорение работы параллельной программы в зависимости от числа процессов

Таким образом, применение конвейерной организации параллельных вычислений на многопроцессорных вычислительных системах с распределенной памятью эффективно для разностной задачи любой размерности и дает значительное ускорение в решении задач гидродинамики с доминирующим направлением течения.

Литература

1. **Андерсон Д., Таннехилл Дж., Плетчер Р.** Вычислительная гидромеханика и теплообмен: В 2 т. / Пер. с англ. — М. : Мир, 1990. — Т. 2. — 392 с.

2. **Лойцянский Л. Г.** Механика жидкости и газа: Учеб. для вузов. — изд. 6-е, перераб. и доп. — М. : Наука. Гл. ред. физ.-мат. лит., 1987. — 840 с.

3. **Патанкар С.** Численные методы решения задач теплообмена и динамики жидкости / Пер. с англ. — М. : Энергоатомиздат, 1984. — 152 с.

4. **Ghia U., Ghia K. N., Shin C. T.** High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method // J. Comp. Phys. — 1982. — Vol. 48. — P. 387—411.

5. **Tropea C. D., Gackstatter R.** The Flow Over Two-Dimensional Surface-Mounted Obstacles at Low Reynolds Numbers // Journal of Fluids Engineering. — 1985. — № 4. — P. 489—494.

Численное решение некоторых обратных задач переноса примеси на многопроцессорных вычислительных системах[1]

Е. А. Панасенко, А. В. Старченко
Томский государственный университет, Томск

В настоящее время проблема охраны окружающей среды и ее восстановления становится одной из важных задач науки, развитие которой стимулируется всевозрастающими темпами технического прогресса во всех странах мира [1]. Ухудшение качества атмосферного воздуха, накопление в нем газообразных компонентов вследствие выбросов в атмосферу отходов промышленных предприятий и выхлопных газов автотранспорта приводит к ухудшению здоровья населения, а в глобальном масштабе — к изменению климата на планете. Таким образом, проблема взаимодействия человека с окружающей средой в настоящее время представляет новую и активно развивающуюся область применений методов математического моделирования.

Одним из способов регулирования уровня загрязнения атмосферного воздуха является контроль интенсивности выбросов вредных веществ. Однако даже разветвленная сеть постов наземных наблюдений не всегда в полной мере способна предоставить необходимую для природоохранных служб информацию. Большую помощь здесь могут оказать применение методов математического моделирования и, особенно, технологии численного решения обратных задач по определению характеристик источников загрязнения атмосферного воздуха по данным измерений концентрации вредных веществ [1, 2, 4]. Существенное ускорение получения результатов расчетов и повышение качества численного прогноза в настоящее время обеспечивают современные высокопроизводительные многопроцессорные вычислительные системы.

Целью данной работы является построение параллельных алгоритмов решения обратных задач охраны окружающей среды методом Г. И. Марчука [1], опирающимся на использование сопряженных уравнений переноса и диффузии субстанции.

[1] Работа выполнена при поддержке РФФИ, грант № 07-05-01126.

Физическая постановка задачи

Требуется по известным метеорологическим параметрам атмосферы и результатам измерений концентрации примеси, проводимых в N точках в течение некоторого периода времени T, определить мощность, координаты и время срабатывания мгновенного точечного источника примеси. Задача такого рода чрезвычайно актуальна для обработки данных автоматизированной системы контроля экологической или радиационной обстановки вблизи источников повышенной опасности [3]. На рис. 1 проиллюстрирована данная ситуация.



Рис. 1. Размещение постов наблюдения (•) в 30-километровой зоне Сибирского химического комбината [3]

Математическая постановка обратной задачи с мгновенным источником

Для постановки обратной задачи используется метод Марчука, который основан на решении уравнения, сопряженного с полуэмпирическим уравнением турбулентной диффузии, и двойственным представлением функционала от концентрации примеси [1].

Для сокращения выкладок рассмотрим двумерное нестационарное уравнение «адвекции-диффузии», представляющее перенос примеси в заданном потоке:

$$\frac{\partial C}{\partial t}+U\frac{\partial C}{\partial x}+V\frac{\partial C}{\partial y}=\frac{\partial}{\partial x}\left[\Gamma\frac{\partial C}{\partial x}\right]+\frac{\partial}{\partial y}\left[\Gamma\frac{\partial C}{\partial y}\right]+Q, \quad (1)$$
$$\left(0<t<T,0<x<L_x,0<y<L_y\right)$$

где: C — концентрация примеси; U, V — компоненты вектора скорости потока; Γ — коэффициент турбулентной диффузии; $Q=Q_0\delta(t-t_0)\delta(x-x_0)\delta(y-y_0)$ — член, моделирующий выброс из мгновенного точечного источника; Q_0 — мощность источника; t_0, x_0, y_0 — время и координаты выброса соответственно.

Начальные и граничные условия будут иметь вид:

$$t=0: C\left(t,x,y\right)=0;$$
$$x=0: \quad \frac{\partial C}{\partial x}=0; \quad y=0: \quad \frac{\partial C}{\partial y}=0; \quad (2)$$
$$x=L_x: \quad \frac{\partial C}{\partial x}=0; \quad y=L_y: \quad \frac{\partial C}{\partial y}=0.$$

Сопряженная к (1) и (2) задача получается следующим образом. Уравнение (1) умножается на некоторую функцию $C^*=C^*\left(t,x,y\right)$ и интегрируется по времени и пространству:

$$\int_0^T\int_0^{L_x}\int_0^{L_y}C^*\frac{\partial C}{\partial t}dydxdt+\int_0^T\int_0^{L_x}\int_0^{L_y}C^*U\frac{\partial C}{\partial x}dydxdt+$$
$$+\int_0^T\int_0^{L_x}\int_0^{L_y}C^*V\frac{\partial C}{\partial y}dydxdt=\int_0^T\int_0^{L_x}\int_0^{L_y}C^*\frac{\partial}{\partial x}(\Gamma\frac{\partial C}{\partial x})dydxdt+$$
$$+\int_0^T\int_0^{L_x}\int_0^{L_y}C^*\frac{\partial}{\partial y}(\Gamma\frac{\partial C}{\partial y})dydxdt+\int_0^T\int_0^{L_x}\int_0^{L_y}C^*Qdydxdt$$

Используя интегрирование по частям, начальные и граничные условия (2) приходим к выражению:

$$\int_0^T\int_0^{L_x}\int_0^{L_y} C\left(-\frac{\partial C^*}{\partial t}-\frac{\partial UC^*}{\partial x}-\frac{\partial VC^*}{\partial y}-\frac{\partial}{\partial x}\left(\Gamma\frac{\partial C^*}{\partial x}\right)-\frac{\partial}{\partial y}\left(\Gamma\frac{\partial C^*}{\partial y}\right)\right)dydxdt -$$

$$-\int_0^{L_x}\int_0^{L_y} C\cdot C^*\Big|_0^T dydx-\int_0^T\int_0^{L_y} U\cdot C^*\cdot C\Big|_0^{L_x} dydt-\int_0^T\int_0^{L_y} V\cdot C^*\cdot C\Big|_0^{L_y} dxdt+$$

$$+\int_0^T\int_0^{L_y}\Gamma C^*\frac{\partial C}{\partial x}\Big|_0^{L_x} dydt-\int_0^T\int_0^{L_y}\Gamma C\frac{\partial C^*}{\partial x}\Big|_0^{L_x} dydt+$$

$$+\int_0^T\int_0^{L_x}\Gamma C^*\frac{\partial C}{\partial y}\Big|_0^{L_x} dxdt-\int_0^T\int_0^{L_x}\Gamma C\frac{\partial C^*}{\partial y}\Big|_0^{L_x} dxdt+\int_0^T\int_0^{L_x}\int_0^{L_y} C^*Qdydxdt.$$

Предположим, что функции $C_i^*\cdot(i=1,...,N)$ удовлетворяют уравнениям:

$$-\frac{\partial C^*}{\partial t}-\frac{\partial UC^*}{\partial x}-\frac{\partial VC^*}{\partial y}-\frac{\partial}{\partial x}\Gamma\frac{\partial C^*}{\partial x}-\frac{\partial}{\partial y}\Gamma\frac{\partial C^*}{\partial y}=P_i, \qquad (3)$$

$$i=1,...,N,$$

$$T>t>0, 0<x<L_x, 0<y<L_y$$

с соответствующими начальными и граничными условиями:

$$t=T:\ C^*(t,x,y)=0;$$

$$x=0:\ UC^*+\Gamma\frac{\partial C^*}{\partial x}=0\ \ y=0:\ \ VC^*+\Gamma\frac{\partial C^*}{\partial y}=0; \qquad (4)$$

$$x=L_x:\ UC^*+\Gamma\frac{\partial C^*}{\partial x}=0\ \ y=L_y:\ \ VC^*+\Gamma\frac{\partial C^*}{\partial y}=0.$$

где $P_i=\delta(x-x_i)\delta(y-y_i)\delta(t-t_i), i=1,...,N; N$ — количество измерений концентрации C в точках с координатами (x_i,y_i) в моменты t_i.

С учетом уравнений вида (3) и выбранных начальных и граничных условий прямой и сопряженной задачи можно записать следующее двойственное представление функционала от концентрации [1, 3]:

$$J_i=\int_0^T dt\int_0^{L_x}\int_0^{L_y} P_iCdxdy=\int_0^T dt\int_0^{L_x}\int_0^{L_y} C_i^*Qdxdy;$$

или с учетом вида функций P_i и Q

$$J_i = C_i = Q_0 C_i^*\left(t_0, x_0, y_0\right),\ i = 1, ..., N, \tag{5}$$

где $C_i^*\left(t,x,y\right)$ — решение сопряженной задачи с P_i правой частью; $C_i = C\left(t_i, x_i, y_i\right)$ — измеренные значения концентрации.

Соотношения (5) можно рассматривать как уравнения относительно неизвестных Q_0, t_0, x_0, y_0, причем решение i-ой сопряженной задачи (3)—(4) $C_i^*\left(t,x,y\right)$ зачастую может быть найдено только численно. Поэтому для нахождения параметров источника Q_0, t_0, x_0, y_0 был построен следующий функционал

$$\begin{gathered}\phi\left(R,x,y,t\right) = \sum_{i=1}^{N}\left(RC_i^*\left(t,x,y\right) - C_i\right)^2, \\ R > 0, 0 < x < L_x, 0 < y < L_y, 0 < t < T,\end{gathered} \tag{6}$$

глобальный минимум которого достигается при $R = Q_0$, $x = x_0$, $y = y_0$, $t = t_0$. Проверку глобального минимума $\phi\left(R,x,y,t\right)$ необходимо проводить на каждом шаге по времени по всей расчетной области при одновременном численном решении N сопряженных задач (3), (4) с различными правыми частями $P_i = \delta\left(t - t_i\right)\delta\left(x - x_i\right)\delta\left(y - y_i\right), i = 1, ..., N.$

Для сокращения числа независимых переменных, по которым ведется поиск минимума, пользуясь необходимым условием экстремума функции нескольких переменных, из (6) можно выразить Q_0:

$$Q_0 = \frac{\sum_{j=1}^{N}\left(C_j^*\left(t_0, x_0, y_0\right) \cdot C_i\right)}{\sum_{j=1}^{N}\left(C_j^*\left(t_0, x_0, y_0\right)\right)^2} \tag{7}$$

Тогда после подстановки (7) в (6) потребуется искать глобальный минимум следующего функционала:

$$\phi_1\left(x,y,t\right) = \sum_{i=1}^{N}\left(\frac{\sum_{j=1}^{N}\left(C_j^*\left(t,x,y\right) \cdot C_j\right)}{\sum_{j=1}^{N}\left(C_j^*\left(t,x,y\right)\right)^2} C_i^*\left(t,x,y\right) - C_i\right)^2 \tag{8}$$

Нахождение глобального в $\left(0,T\right) \times \left[0,L_x\right] \times \left[0,L_y\right]$ минимума $\phi_1\left(x,y,t\right)$ позволит установить координаты и момент срабаты-

вания мгновенного источника, а по формуле (7) определить его интенсивность.

Численное решение сопряженной задачи

Для численного решения задачи (3)—(4) использовались метод конечного объема и явные разностные схемы. Для аппроксимации адвективных членов применялась схема MLU Ван Лира, которая незначительно уступает по точности при решении задач подобного класса схеме четвертого порядка Ботта [5, 6].

Тестирование правильности решения обратной задачи (определения координат, времени срабатывания и интенсивности источника) проводилось с использованием результатов численного решения прямой задачи для получения необходимых входных данных по концентрации примеси.

Параллельная реализация численного решения обратной задачи

Следует обратить внимание на то, что при решении обратной задачи на каждом шаге по времени решается не одна, а N независимых сопряженных задач переноса типа (3)—(4), следовательно, они могут решаться параллельно.

Такие условия проведения численного моделирования заставляют привлекать высокопроизводительную вычислительную технику, в частности, вычислительный кластер ТГУ СКИФ Cyberia, на котором установлена библиотека передачи сообщений MPI (Message Passing Interface). Эта библиотека функций обеспечивает возможность обмена информацией между процессами параллельной программы, запущенными на многопроцессорной вычислительной системе.

Распараллеливание метода численного решения обратной задачи производилось с использованием принципа «master-slave» (рис. 2). При такой реализации управляющий процесс (master) передает каждому slave-процессу значения (U, V, Γ, x_i, y_i, t_i), необходимые для расчетов, а подчиненные процессы (slave), получив исходные данные, в свою очередь, ведут расчеты независимо друг от друга, а полученные решения своей сопряженной задачи возвращают управляющему процессу, который прово-

дит вычисления $\phi_1(x,y,t)$ и ищет глобальный минимум этого функционала в $(0,T)\times[0,L_x]\times[0,L_y]$.

Заметим, что при таком способе организации параллельных вычислений при запуске параллельной программы на $N+1$ вычислительных узлах большую долю вычислительной работы обычно выполняют N подчиненных slave-узлов. Один управляющий master-узел координирует работу остальных (подготавливает и рассылает slave-узлам данные для расчета, собирает данные и осуществляет их дополнительную обработку). На каждом узле в данный момент исполняется единственная задача (процесс), в случае недоступности требуемого количества узлов задание ставится в очередь до момента освобождения нужного количества узлов.

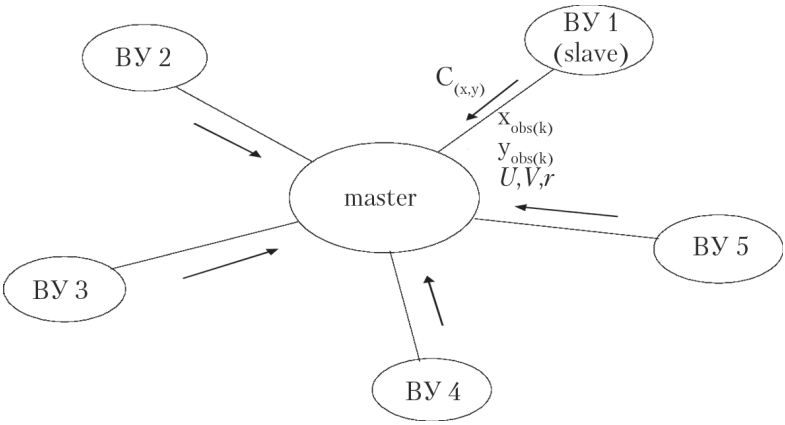


Рис. 2. Схема параллельной реализации

В нашем случае управляющий процесс передает каждому процессу значения параметров постов наблюдения, скорость, размеры области, коэффициент диффузии. Каждый из процессов рассчитывает решение сопряженной задачи и передает его мастеру, который, в свою очередь, обрабатывает полученные значения, вычисляет функционал и в точке его минимума с начала решения сопряженных задач получает параметры источника. Затем время уменьшается на величину шага, и мастер передает процессам новые значения метеорологических параметров, и т. д. расчет повторяется до окончания счета.

Также в данной работе был рассмотрен способ организации параллельных вычислений, при котором управляющий процесс дополнительно численно интегрировал одну сопряженную задачу (3)—(4).

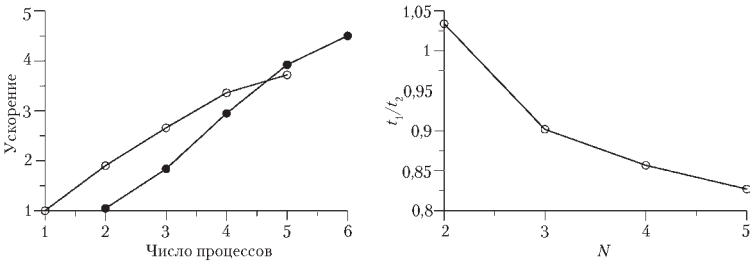


Рис. 3. Ускорение для двух способов параллельной реализации и отношение времен работы параллельных программ

На рис. 3 представлены характеристики ускорения рассмотренных выше параллельных алгоритмов численного решения обратной задачи, полученного при запуске параллельных программ на вычислительном кластере ТГУ СКИФ Cyberia.

Ускорение рассчитывается как отношение времени выполнения программы, решающей обратную задачу на одном процессоре, к времени численного решения обратной задачи на p процессорах. Из рис. 3 видно, что при выполнении параллельной программы на 5 процессорах вычислительного кластера СКИФ Cyberia при указанных выше параметрах и при условии, что master-узел только подсчитывает суммы для $\phi_1(x,y,t)$ и ищет его минимум, удается получить ускорение вычислений почти в 4 раза (на рис. 3 — это кривая с закрытыми значками). В случае же, когда работают все процессы (и master и slave), но master-процесс, рассчитывая значения функционала $\phi_1(x,y,t)$ и осуществляя поиск его минимума, еще решает одну сопряженную задачу, как и подчиненные процессы, получено ускорение в 3,5 раза (кривая с открытыми значками). Разница в ускорении связана с тем, что в последнем случае неравномерность загрузки процессоров более значительна, поскольку при увеличении числа используемых процессоров возрастает количество простаивающих slave-процессоров.

На рис. 3 также представлено изменение отношения времен работы параллельных программ в зависимости от количества со-

пряженных задач. Из рисунка также следует, что первый способ организации параллельных вычислений имеет решительное преимущество.

Заключение

В представленной работе сформулирована постановка обратной задачи через аппарат сопряженных функций. Разработаны методы численного решения обратных задач, в которых по известным значениям измеренной концентрации примеси определяется месторасположение, время и мощность мгновенного выброса примеси. Описанные в работе подходы можно применять и для реальных обратных задач, например для определения расположения и интенсивности источников загрязнения атмосферного воздуха в г. Томске.

При параллельной реализации обратной задачи переноса примеси применена функциональная декомпозиция, в соответствии с которой решаемые сопряженные задачи распределяются по имеющимся вычислительным узлам, а управляющий узел осуществляет расчет значений функционала, нахождение минимума которого дает решение обратной задачи. На основе проведенных вычислительных экспериментов установлено, что параллельная реализация алгоритма ускоряет процесс вычислений почти в 5 раз на 6 процессорах. Кроме того, было показано, что нецелесообразно нагружать управляющий узел решением сопряженной задачи, поскольку в этом случае простаивают подчиненные вычислительные узлы.

Литература

1. **Марчук Г. И.** Математическое моделирование в проблеме окружающей среды. — М. : Наука, 1982. — 315 с.

2. **Агошков В. И.** Методы оптимального управления и сопряженных уравнений в задачах математической физики. — М. : ИВМ РАН, 2003. — 256 с.

3. [Электронный ресурс]. — Режим доступа: http://gis.green.tsu.ru/Website/askro/viewer.htm, свободный.

4. **Алоян А. Е., Пененко В. В., Козодеров В. В.** Математическое моделирование в проблеме окружающей среды // Современные проблемы вычислительной математики и математичес-

кого моделирования. — М. :Наука, 2005. — Т. 2 Математическое моделирование. — С. 277—349.

5. **Панасенко Е. А.** Численное исследование переноса примеси в атмосфере // Третья всероссийская конференция молодых ученых. Фундаментальные проблемы новых технологий в 3-м тысячелетии. — 2006. — С. 582—586.

6. **Панасенко Е. А., Старченко А. В.** Численное решение некоторых обратных задач переноса примеси в атмосфере // Материалы XV Международ. конф. по механике и современным прикладным программным системам (ВМСППС 2007). — М. : Вузовская книга, 2007. — С. 404—405.

Решение задачи схлопывания каверны на поверхности жидкости в трехмерной постановке на кластере «СКИФ Cyberia»

В. В. Малышенко, Д. Р. Авзалов
Кемеровский государственный университет, Кемерово

Впервые задача схлопывания каверны на свободной поверхности под действием силы тяжести была поставлена М. А. Лаврентьевым. В двумерной плоской постановке данная задача в идеальной жидкости решалась многими авторами (В. К. Кедринский, А. Г. Терентьев и др.). Далее в работах К. Е. Афанасьева, Г. Г. Короткова рассматривалась эта же задача, но уже в осесимметричной постановке [2]. В настоящей работе рассматривается задача об эволюции выемки на свободной поверхности в трехмерной постановке.

Для решения поставленной задачи используется метод частиц, разработанный А. М. Франком [1]. Метод частиц основан на свободно-лагранжевом подходе, что позволяет легко отслеживать границы раздела, причем произвольное изменение связности течения и границ не доставляет никаких дополнительных алгоритмических сложностей.

В связи с тем, что решение задач гидродинамики со свободными границами в трехмерной постановке требует больших вычислительных высокопроизводительных ресурсов, для численного моделирования применяется параллельный метод частиц.

Данная работа направлена на исследование вычислительных свойств метода частиц, его распараллеливание и оценку качества параллельного алгоритма.

Постановка задачи

Рассматривается течение идеальной несжимаемой невязкой жидкости в области V, ограниченной свободной поверхностью C_3 с полусферической выемкой радиуса R, боковыми твердыми стенками C_1 и непротикаемым дном C_2, расположенным на глубине H.

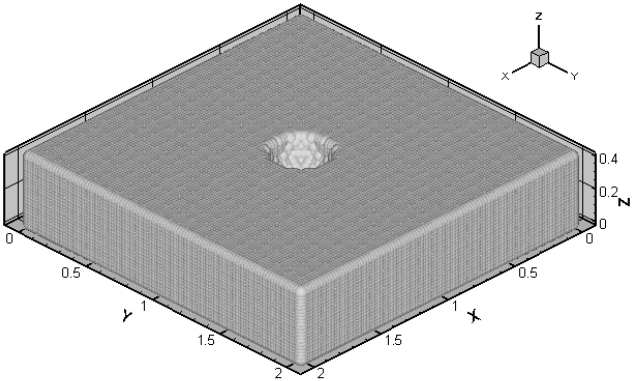


Рис. 1. Начальная геометрия эволюции полусферической выемки над ровным дном

В области V решается уравнение Эйлера:

$$\frac{d\vec{n}}{dt} = \frac{\vec{F}}{r}, r \in V. \tag{1}$$

Условие несжимаемости жидкости и условие непротекания на твердых стенках:

$$div\vec{u} = 0, r \in V, \tag{2}$$

$$u_n = 0, r \in C_2 \tag{3}$$

где n – внешняя нормаль к границе жидкости.

Начальное поле скоростей $\vec{u}_0 = (u_0, v_0) \in (0,0)$.

Распараллеливание алгоритма метода частиц.

Основной задачей распараллеливания алгоритма метода частиц является:

- определение количества вычислительных операций;
- определение, какие вычислительные операции могут выполняться независимо друг от друга;
- распределение операций по узлам многопроцессорного ресурса;
- определение необходимых коммуникаций между узлами многопроцессорного ресурса и вычисление объемов передаваемых данных.

Метод частиц подробно описан в [1]. Алгоритм метода частиц, распараллеливание которого выполняется, описан в [6].

Определим количество операций, выполняемое на каждом шаге по времени.

Задание начальных условий

Пусть степень B-сплайна равна 2, количество частиц в области $N = 200\ 000$, и задана сетка из $M = m_x \times m_y \times m_z = 30 \times 30 \times 30$ базисных функций.

Предиктор

В начале шага по времени, вычисляется предиктор. На вычисление предиктора для одной частицы в трехмерном пространстве затрачиваем 9 арифметических операций, тогда на весь блок необходимо затратить $9N$ операций.

Вычисление дополнительного блока

Квадратная матрица $Mv(i,k)$ размера $N \times 18$ содержит значения $v_i\left(r_k^{n+1/2}\right)$. Вычисление одного значения базисной функции требуют $t = 232$ операций, из них для вычисления B-сплайна второго порядка необходимо 59 операций, для производной необходимо 57 операций. Тогда, для формирования матрицы $Mv(i,k)$ необходимо $N \times 18 \times 232 = N \times 4176$ операций.

Вычисление матрицы Грамма

Для вычисления вклада одной частицы в значение одного элемента матрицы необходимо вычислить значение базисной функции в частице и выполнить скалярное произведение, что в трехмерном пространстве даст $2x + 7$ арифметические операции. Далее требуется просуммировать найденные значения по k. Таким образом, для вычисления одного a_{ij} необходимо $(2x+7)N$ операций. Всего перебирается M^2 вариантов. Поэтому для вычисления матрицы $\{a_{ij}\}$ затрачивается $(2x+7)NxM^2$ операции.

Определение правой части СЛАУ

Для вычисления одного элемента правой части в первом алгоритме требуется $(t+7+9)N$ операций. Тогда для всей правой части $(t+7+9)NM$.

Решение СЛАУ и определение новых скоростей частиц

Решение СЛАУ методом Гаусса с выбором ведущего элемента требует для обоих алгоритмов M^3 операций.

Вычисление скорости на промежуточном шаге по времени

При определении скорости S вычисляются значения M базисных функций, для этого нужно затратить $t\ xM$ операций, затем, умножая вектора на скаляры в трехмерном случае, получаем еще $5M$ операций. Тогда для вычисления скоростей S для всех частиц требуется $NxM(t+5)$ арифметических операций.

Определение новых скоростей и координат частиц

Вычисляются скорости частиц на $n+1$ шаге. Число операций требуемых равно $9N$. На определение новых координат частиц затрачивается еще $9N$ операций.

Общая вычислительная сложность алгоритма

В результате определено количество выполняемых операций за один шаг по времени:

$$T = 9N + (2x+7)NxM^2 + (t+7+9)NM + M^3 +$$
$$+NxM(t+5) + 2x9N = 1{,}86941E+13$$

Декомпозиция вычислительных операций

Проведем декомпозицию каждого блока алгоритма метода частиц. Операции во всех блоках, кроме «решения СЛАУ», имеют идеальный параллелизм, поэтому вычисления в каждом блоке могут быть выполнены независимо друг от друга на SIZE процессорах.

Основной задачей декомпозиции является распределение нагрузки по процессорам таким образом, чтобы минимизировать количество пересылок и увеличить число параллельно выполняемых операций.

Блок «Координаты и скорости на целом шаге» хранит на каждом процессора часть массива координат $ChCoor\left(N_p\right)$ и часть массива скоростей $ChSkor\left(N_p\right)$. Пересылок не требуется.

Блок «вычисление предиктора» вычисляет на дробном шаге $n+1/2$ значение массива координат $ChCN12\left(N_p\right)$. Пересылок не требуется.

Блок «формирование матрицы СЛАУ». В данном блоке формируется матрица A, учитывая, предложенные выше механизмы удаления ненужных строк не являются самыми эффективными. Например, при решении данной задачи размер матрицы A изменяется от 3700 от 4200 в зависимости от характера течения. На каждом процессоре сформирована матрица A, состоящая из частичных сумм, собираются все части матрицы на одном процессоре, а потом раздаются остальным процессорам. Количество пересылок равно M, следовательно, время, затраченное на пересылки $t_S M + t_w M^2$.

Блок «формирование правой части» может выполняться одновременно с предыдущим блоком, так как полностью от него независим. При учете того, что количество процессоров во время расчета остается числом неизменным, то более эффективной будет реализация параллельного метода частиц, при которой данный блок выполняется после блока «формирование матрицы СЛАУ». Аналогично предыдущему блоку, вектор правой части f суммируется на одном процессоре и раздается остальным. Количество пересылок равно 1, следовательно, время, затраченное на пересылки $t_S + t_w M$.

Блок «решение СЛАУ». Для решения СЛАУ применяется параллельный метод Гаусса с выбором ведущего элемента. Количество пересылок равно M, время, затраченное на пересылки $t_S M + t_w \frac{M^2}{2}$.

Блок «нахождение скорости на дробном шаге». Каждый процессор после выполнения предыдущего блока хранит весь вектор коэффициентов разложения скорости на дробном шаге по базису

пространства, поэтому каждый процессор вычисляет свою часть массива скоростей $ChUckr\left(N_p\right)$. Пересылок не требуется.

Блок «нахождение новых координат и скорости» каждой процессор вычисляет часть массива координат $ChCoor\left(N_p\right)$ и часть массива скоростей $ChSkor\left(N_p\right)$. Пересылок не требуется.

Далее осуществляется переход на следующую итерацию.

Определение теоретического ускорения распараллеленной программы

Для подсчета теоретического ускорения $S_p = T_1 / T_p$ нужно определить время выполнения одного шага по времени (задачи) на одном процессоре и на p процессорах.

Проведем теоретическую оценку ускорения и эффективности параллельного алгоритма метода частиц. Сделаем предположение о соотношении времени выполнения одной арифметической операции ко времени выполнения пересылки одной переменной длиной 8 байт: $c = t_a / t_w$. Если для объединения узлов в кластере используется сети типа Infiniband или Quadrics, характеризуемые высокой скоростью передачи данных 800 МБ/с и низкой латентностью менее 8 мкс, то величина $c = 100$. Если в качестве сетевой инфраструктуры используется сеть типа GigabitEthernet, характеризуемая с скоростью передачи данных 300 МБ/с и латентностью более 100 мкс, то величина $c = 1000$.

В табл. 1 и 2 представлены теоретические ускорения и эффективность параллельного алгоритма для разных типов сетевой инфраструктуры кластеров при условии одинаковой производительности процессоров.

Таблица 1

Ускорение и эффективность (теоретические)

Количество процессоров	Ускорение $c = 100$	Ускорение $c = 1000$	Эффективность $c = 100$	Эффективность $c = 1000$
2	1,88871	1,28855	94,44 %	64,42 %
4	3,57521	1,90008	89,38 %	47,50 %
8	6,45894	2,49124	80,74 %	31,14 %
16	10,82435	2,95016	67,65 %	18,43 %
32	16,3493	3,24947	51,09 %	10,15 %
64	21,9517	3,42311	34,30 %	5,34 %

Полученные теоретические оценки позволяют охарактеризовать параллельный алгоритм метода. Алгоритм является сильно связанным, поэтому изменение характеристик сети сильно сказывается на эффективности параллельного алгоритма. В результате чего можно сделать вывод о том, что данных подход к распараллеливанию сильно ограничивает круг многопроцессорных вычислительных кластеров, на которых можно производить расчеты задач решаемых данным методом.

Результаты распараллеливания алгоритма метода частиц

Для проведения численных расчетов использовался кластер «СКИФ Cyberia» Томского государственного университета. В расчетах использовалось от 1 до 64 процессоров.

Таблица 2

Ускорение и эффективность (экспериментальные)

Количество процессоров	Время, сек	Ускорение	Эффективность
1	30,5625	—	—
2	12,8092	2,3860	119,29%
4	7,4191	4,1194	102,98%
8	3,1271	9,7734	122,16%
16	1,4908	20,5010	128,13%
32	1,0712	28,5305	89,15%
64	0,9273	32,9574	51,49%

В табл. 2 показана зависимость времени выполнения одного итерационного шага задачи от количества процессоров, на которых выполнялся расчет. Анализируя полученные ускорение и эффективность параллельной задачи можно сделать вывод о хорошем качестве распараллеливания алгоритма метода частиц для решения задач гидродинамики со свободными границами.

Анализируя ускорение, можно сделать вывод о том, что для решения поставленной задачи с заданным набором параметров не целесообразно использование более 16 процессоров, так как время, затрачиваемое на пересылки данных, становиться преобладающим над временем, затрачиваемым на выполнение вычислительных операций.

Высокая эффективность параллельного алгоритма является следствием того, что с увеличением количества процессоров, задействованных в расчете, уменьшается объем оперативной памяти, используемой на каждом процессоре, это приводит к эффективному использованию кэш-память процессора, уменьшению кэш-промахов, и как результат, уменьшение времени выполнения арифметических операций.

Результаты распараллеливания программы позволяют сделать следующие выводы:

Параллельный вариант программы решения задач гидродинамики со свободными границами позволяет решать задачи большей размерности за счет использования больших объемов оперативной памяти; решение задачи на кластере «СКИФ Cyberia» позволяет существенно сократить время численных расчетов; для поставленной задачи эффективным является использование 16 процессоров.

Литература

1. **Франк А. М.** Дискретные модели несжимаемой жидкости. — М. : Физматлит, 2001.

2. **Afanas'ev K. E., Korotkov G. G.** Эволюция полукруговой выемки на свободной поверхности в плоском и осесимметричном случаях // The International Summer Scientific School «High Speed Hydrodynamics», June 2002, Cheboksary.

3. **Афанасьев К. Е., Стуколов С. В.** КМГЭ для решения плоских задач гидродинамики и его реализация на параллельных компьютерах : Учеб. пособие. — Кемерово : КемГУ, 2001. — 208 с.

4. **Овсянников Л. В.** Общие уравнения и примеры // Задачи о неустановившемся движении жидкости со свободной границей. – Новосибирск : Наука, 1967. – С. 5—75.

5. **Завьялов Ю. С., Квасов Б. И., Мирошниченко В. Л.** Методы сплайн-функций. – М. : Наука, 1980. – 352 с.

6. **Малышенко В. В.** Параллельная реализация метода частиц для решения задач гидродинамики со свободными границами // Третья Сибирская школа-семинар по параллельным вычислениям / Под ред. проф. А. В. Старченко. – Томск : Изд-во Том. ун-та, 2006. – С. 50—60.

Фрагментированный алгоритм LU-разложения и его параллельная реализация

К. Г. Чаюк
Новосибирский государственный технический университет,
Новосибирск

Фрагментированный (сборочный) подход к параллельному программированию был предложен в [1], рассматривался в [2] и применялся для распараллеливания метода частиц-в-ячейках (PIC) в [3]. В [4] описана runtime-система для исполнения фрагментированной программы.

В данной работе необходимо было разработать и исследовать фрагментированный алгоритм LU-разложения и программно реализовать его, изучить качество исполнения фрагментированной программы, сравнить его с качеством исполнения последовательной и параллельной программ LU-разложения. Результаты работы предполагается использовать для создания стандартной параллельной подпрограммы LU-разложения.

Рассмотрим LU-разложение. Пусть дана система линейных алгебраических уравнений:

$$Ax = F,$$
$$A = LU,$$
$$LUx = F,$$
$$n^2.$$

Рассмотрим решение этой системы методом, основанном на LU-разложении.

Предположим, что нам удалось разложить матрицу

$$A = LU.$$

Подставляя в СЛАУ, получаем:

$$LUx = F$$

Обозначим $Ux = y$.

Тогда получим $Ly = F$.

Рассмотрим алгоритм получения LU-разложения. Матрицы L и U будем искать в следующем виде:

$$L = \begin{pmatrix} l_{11} & 0 & 0 & \dots \\ l_{21} & l_{22} & 0 & \\ l_{31} & l_{32} & l_{33} & \\ \dots & & & \end{pmatrix} \qquad U = \begin{pmatrix} 1 & u_{12} & u_{14} & \dots \\ 0 & 1 & u_{23} & \\ 0 & & 1 & \\ \dots & & & \end{pmatrix}$$

Учитывая равенство $A = LU$. и умножая последовательно строки матрицы L на столбцы матрицы U, получаем систему, состоящую из $n2$ уравнений с $n2$ неизвестными (в результате чего LU-разложение единственно) $l_{ij}, i = \overline{1,n}, j = \overline{1,i}$ и $u_{ij}, i = \overline{1,n}, j = \overline{i+1,n}$ (n – размерность СЛАУ):

$$a_{11} = l_{11}; a_{12} = l_{11}u_{12}; a_{13} = l_{11}u_{13}; \dots$$
$$a_{21} = l_{21}; a_{22} = l_{21}u_{12}; a_{12} = l_{11}u_{12}; a_{23} = l_{21}u_{13} + l_{22}u_{23}; \dots$$
$$a_{31} = l_{31}; a_{23} = l_{31}u_{12} + l_{32}; a_{33} = l_{31}u_{13} + l_{32}u_{23} + l_{33}; \dots$$
$$\dots$$

Решая систему можно получить общие формулы для нахождения элементов матриц L и U:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \; u_{ij} = \frac{1}{l_{ii}}\left[a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \right] \qquad (1)$$

По этим формулам производились вычисления элементов матриц L и U в фрагментах.

Идея фрагментированного алгоритма заключается в обработке в нужном порядке и параллельно (где это возможно) одним и тем же последовательным программным кодом различных частей матрицы (всех подматриц). Рассмотрим фрагментированный алгоритм.

Квадратами на рис. 1 обозначены фрагменты исходной матрицы, обрабатываемые как отдельный процесс. Каждый процесс производит LU-разложение в своем фрагменте. Цифрами и стрелками показан порядок обработки фрагментов.

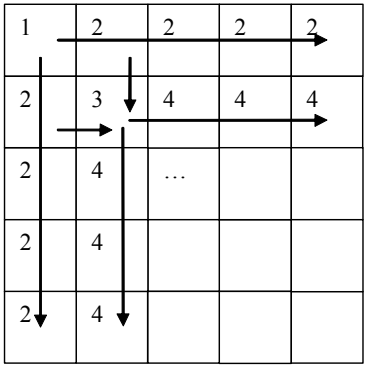


Рис. 1

Управление вычислениями основано на зависимости фрагментов по данным. Если посмотреть на формулы (1), по которым рассчитывались элементы матрицы, то видно, что для вычисления одного элемента необходима сумма $\sum_{k=1}^{j-1} l_{ik} u_{kj}$, т. е. нужно знать результаты разложения предыдущих фрагментов до диагонали. Поэтому сразу может вычисляться только фрагмент с номером один. После того, как он сделал LU-разложение, могут начать вычисления все фрагменты с номером два, и т. д. И, исходя из этой зависимости, выбирается последовательность запуска фрагментов на исполнение. Для исполнения каждого фрагмента необходимо чтобы все соседи слева и сверху (до диагональных включительно) уже завершили вычисления.

Алгоритм предназначен для исполнения на многопроцессорных машинах с общей памятью. В этом случае отсутствует необходимость пересылки и фрагментации данных.

Все расчеты проводились на smp4x64.sscc.ru. Это четырехпроцессорный сервер с общей памятью HP Integrity rx4640. Он включает 4 процессора Intel Itanium2 (1,5 ГГц) 4 Мбайт кэш, 64 Гбайт оперативной памяти при пропускной способности шины 12,8 Гб/с, 2 жестких диска 146 Гбайт.

Результаты численных экспериментов

Размер матрицы	2500x2500		10000x10000			22500x22500		
Время счета последовательной программы (с)	0.210367		3.535771			17.840454		
Размер фрагмента	1250×1250	625×625	5000×5000	2500×2500	2000×2000	11250×11250	7500×7500	5625×5625
Количество фрагментов	4	16	4	16	25	4	9	16
Время счета фрагментированной программы (с)	0.002578	0.013047	0.037955	0.355869	0.591370	0.171227	0.503275	1.070491

По таблице видно, что на четырех процессорах мы получаем ускорение в 100 раз. При увеличении количества фрагментов ускорение падает, это связано с увеличением затрат на управление,

но и в этом случае, получено ускорение в 10 раз, и с дальнейшим ростом количества фрагментов существенно ниже оно не опускается.

В случае распределенной памяти (мультикомпьютеров), появляется необходимость в пересылке вектора накопленных сумм, что замедляет работу программы. Но затраты на пересылку сопоставимы с тем, что в случае общей памяти, каждый фрагмент сам считает необходимые суммы. При этом появляется более жесткое управление запуском фрагментов. Время счета текущей реализации программы в случае распределенной памяти оказалась в среднем в полтора раза хуже, чем в случае общей.

Подобный выигрыш в системах с общей памятью получается в задачах, использующих при решении явные разностные схемы. Проводя анализ возможных задач, можно предположить, что задачи, в которых допустимо разбиение расчетной области на подобласти или задачи, в которых параллелизм есть на уровне алгоритма (слабые зависимости между действиями или ветвями алгоритма), должны хорошо фрагментироваться. Например, явные разностные алгоритмы, нейросетевые модели и т. д.

В фрагментированной версии размер фрагмента задается пользователем до начала вычислений и, исходя из этого, получается общее количество фрагментов, которых может быть много на узле. Наличие многих фрагментов на узле позволяет скрыть обмены на фоне вычислений.

Динамические свойства фрагментированной программы основаны на том, что на каждом узле исполняется много фрагментов, и они могут единообразным управлением перемещаться на другие узлы, тем самым, обеспечивая равномерную загрузку вычислительных ресурсов.

Таким образом, фрагментированный подход к параллельному программированию численных алгоритмов позволяет, используя последовательные программы технологично конструировать параллельные.

Фрагментированное программирование позволяет автоматически обеспечивать различные динамические качества параллельной прикладной программы, что очень важно при решении задач большого размера на мультикомпьютерах, в том числе и на кластерах и т. д. При этом оно позволяет получить выигрыш во времени исполнения. Главным преимуществом фрагменти-

рованного подхода является высокая способность программы настраиваться на все доступные ресурсы вычислителя (конфигурации сети и связей между узлами мультикомпьютера, количество каждого ресурса и т. д.).

В дальнейшем планируется развивать реализацию с распределенной памятью, так как она лучше использует явные преимущества фрагментированного подхода, а также реализовать другие численные алгоритмы и на их базе накопить библиотеку фрагментированных программ.

Литература

1. **Малышкин В. Э.** Синтез параллельных программ и систем на основе вычислительных моделей. – Новосибирск : Наука, 1988. — 128 с.

2. **Малышкин В. Э.** Введение в параллельное программирование мультикомпьютеров. — Новосибирск, 2003.

3. **Kraeva M. A., Malyshkin V. E.** Assembly Technology for Parallel Realization of Numerical Models on MIMD-Multicomputers // International Journal on Future Generation Computer Systems. — 2001. — Vol. 17. — № 6. — P. 755—765.

4. **Kalgin K. V., Malyshkin V. E., Nechaev S. P., Tschukin G.** Runtime System for Parallel Execution of Fragmented Subroutines // Springer Verlag: LNCS, — Vol. 4671. — P. 546-554.

Реализация алгоритмов цифровой фильтрации и масштабирования видеоизображений на специализированном векторном процессоре G80

Г. Н. Мельников
Томский государственный университет, Томск

Пришло время новой эры расчетов с помощью графических процессоров. Не секрет, что попытки переложить часть вычислений на GPU (Graphic Processor Unit) предпринимались и ранее, однако не имели значимых успехов. 16 февраля 2007 г. корпорация NVIDIA, которая является одним из мировых лидеров по производству графических процессоров, объявила о выпуске набора компонентов для разработчиков под названием CUDA (Compute Unified Device Architecture), а также компилятор языка C для расчета на собственных GPU. Это событие должно дать новый толчок для развития рынка аппаратных вычислений с использованием графических процессоров. По своей сложности GPU уже давно обогнали процессоры, хотя бы за счет использования большего количества транзисторов, а также своей изначальной ориентации.

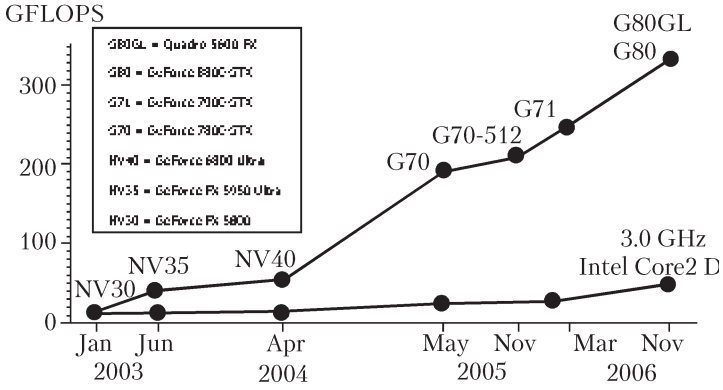


Рис. 1. Вычисление операций с плавающей точкой для CPU и GPU

Так что в современной системе видеокарта могла бы спокойно выступать в качестве помощника для решения тяжелых вычислительных задач. Технология CUDA призвана распараллелить вычислительные задачи между центральный процессором

(CPU) и GPU через стандартный язык программирования C. Пока что поддержка новой технологии присутствует в серии видеокарт GeForce 8800, однако она также будет реализована и в следующих решениях. Разработчики могут загрузить пока что тестовую версию CUDA с официального сайта NVIDIA [1].

Один из графических процессоров, поддерживающих CUDA-технологию, — это процессор G80 (он присутствует в видеокартах GeForce 8800 GTX (GTS)). Данный процессор предлагает огромные вычислительные ресурсы для обработки графической и неграфической информации. Его стоит рассматривать как дополнительное вычислительное устройство, с помощью которого можно разгрузить центральный процессор. Однако, не любую задачу выгодно реализовывать на GPU. GPU предназначен для решения тех задач, которые легко распараллеливаются. Для задач, которые выполняют одинаковые арифметические операции, но над разными данными.

Модель программирования

Программирование на процессоре G80 не стоит рассматривать так же, как и на векторных процессорах x86. Для эффективной работы, в отличие от x86, каждый из потоков, выполняющихся на G80, должен выполнять одни и те же операции, но над разными данными. Отсюда вытекает класс алгоритмов, хорошо ложащихся на данный процессор. Самым простым примером такого алгоритма является, например, независимая обработка каждого элемента массива. Кроме того, G80 специализирован под вычисления с плавающей точкой (тип float (32 бита) в языке C). Сложение, умножение, сложение плюс умножение в данном типе происходят всего за два такта.

Те части приложения, которые вычисляются много раз, но над различными данными, могут быть легко преобразованы в функцию, которая выполняется на устройстве посредством большого числа потоков (Threads). Такая функция в терминах CUDA называется кернелом (Kernel).

Потоки, которые выполняются в кернеле, организованы в таблицу блоков (Grid). Каждый блок содержит некоторое число потоков. Все потоки блока имеют доступ к одной кеш-памяти процессора, на котором они выполняются и могут синхронизироваться. Потоки разных блоков синхронизироваться не могут.

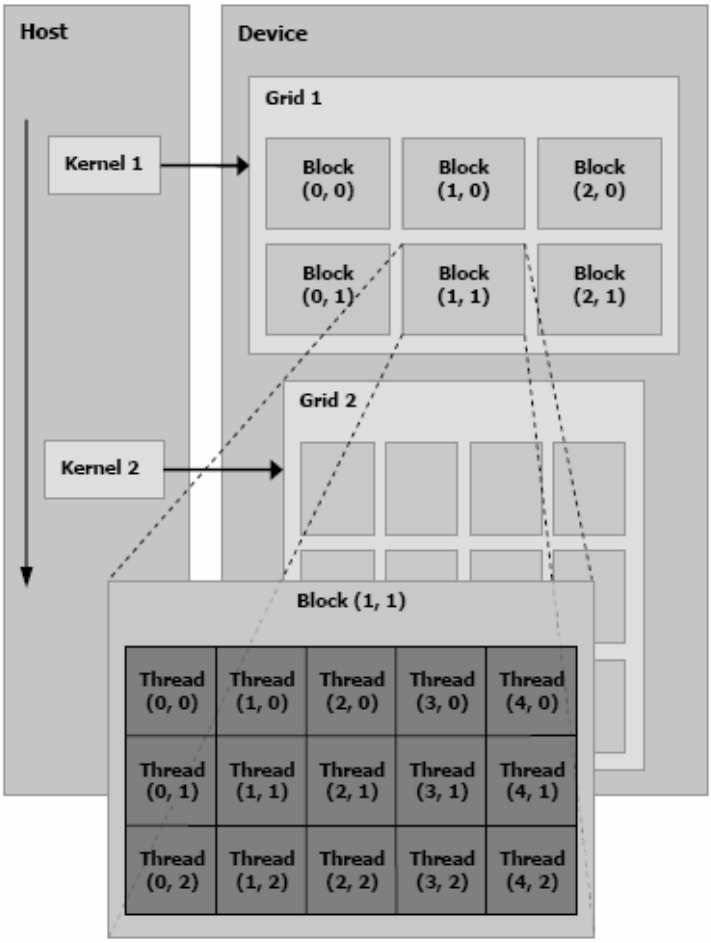


Рис 2. Организация системы потоков (threads) на G80

Блоки и потоки могут индексироваться одномерными, двумерными и трехмерными массивами (один из способов показан на рис. 1). Индексирование является чисто формальным, и размерность массива индексации стоит выбирать исходя из решаемой задачи. Каждый Kernel может предоставить номер блока и потока, который обрабатывается в текущий момент. Исходя из этого, можно однозначно определить часть данных, которую следует обработать.

Количество блоков и входящих в блок потоков должны быть выбраны так, чтобы максимизировать вычислительную мощь процессора. Это означает, что количество блоков должно быть не меньше, чем мультипроцессоров в устройстве. Кроме того, мультипроцессор будет простаивать (в моменты синхронизации и при чтении из памяти) при размещении в нем только одного блока. Поэтому количество блоков должно быть не менее чем в два раза больше, чем мультипроцессоров в устройстве. Кроме того, количество кэш-памяти, приходящейся на блок, должно быть не более половины от общего количества памяти, приходящейся на мультипроцессор. При полной загрузке кэш-памяти одним блоком не будет места в памяти для обработки иного.

Кроме того, нежелательно, чтобы потоки хаотично обращались к кэш-памяти процессора: их хаотичное обращение снизит вычислительную мощь процессора. Доступ к памяти должен быть выровненным [1].

Таким образом, общая схема программирования на G80 заключается в следующем:

1. Копирование данных в память устройства;
2. Копирование каждым потоком своих данных в кэш-память;
3. Обработка данных в кэше;
4. Копирование полученных данных в память устройства;

Модель памяти

G80 состоит из 16 (12) мультипроцессоров по 675 (600) МГц каждый. Каждый из процессоров имеет память следующего типа:

1. 32 битные регистры (8192 шт.)
2. Shared memory — кэш-память (16 Kb).
3. Константный кеш (только для чтения) (8 Kb).
4. Текстурный кеш (только для чтения) (8 Kb).
5. Глобальная память (DRAM), доступна для всех мультипроцессоров. Она не кэшируема.

Все потоки, которые выполняются на устройстве, имеют доступ только к DRAM памяти устройства.

Ниже проиллюстрирована модель доступа к памяти.

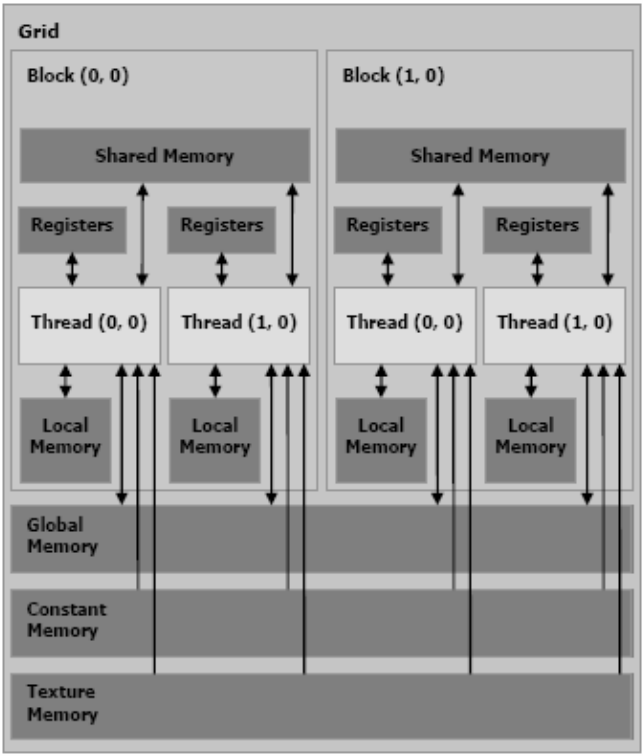


Рис. 3. Модель доступа к памяти

Модель вычисления

Таблица блоков вычисляется на устройстве посредством вычисления одного или нескольких блоков на каждом мультипроцессоре. Каждый блок разбивается на группы потоков, называемые варпами (warp). Каждый варп состоит из некоторого числа потоков (для G80 в варпе 32 потока). Планировщик потока периодически переходит от одного варпа к другому для максимизации использования вычислительных инструкций процессора. Таким образом, структура программы должна быть так разбита на потоки, чтобы каждые 32 потока выполняли одни и те же операции, но над разными данными (так как они выполняются одновременно). Следующие 32 потока могут производить иные вычисления.

Каждый мультипроцессор может обрабатывать несколько блоков параллельно. Это зависит от количества регистров и кэш-памяти, приходящейся на блок.

Заключение

Вычислительная мощь процессора G80 поражает: алгоритмы, которые можно эффективно на нем реализовать, могут выполняться в десятки раз быстрее, чем на центральном процессоре. К таким алгоритмам можно отнести алгоритмы, которые выполняют одни и те же операции, но над различными данными. Причем желательно чтобы количество памяти, необходимой для текущего вычисления, в одном потоке, было не велико (так как у процессора маленький кэш).

Одни из задач, которые можно эффективно решать с использованием CUDA технологии, это задачи, основанные на фильтрах и вейвлетах (например, задача подавления шума и масштабирования видеоизображений). Вейвлет-преобразование, требующее вычислений с плавающей точкой на GPU, будет выполняться в десятки раз быстрее, чем на центральном процессоре. Самая простая реализация данных алгоритмов с использованием технологии CUDA, выглядит следующим образом: мы создаем количество потоков, равных количеству пикселей в изображении, причем каждый поток обрабатывает соответствующий ему пиксель. Реализованный фильтр подавления шума (по времени и пространству), выполняется на устройстве в 50 раз быстрее, чем на центральном процессоре с полной оптимизацией кода. Однако, алгоритмы, быстро выполняющиеся на центральном процессоре, не стоит реализовывать на устройстве вследствие низкой пропускной способности памяти (с компьютера на устройство и обратно). Необходимо реализовывать такие алгоритмы, вычислительная мощь которых сможет компенсировать потери, связанные с пропусканной способностью памяти. Пиковая пропускная способность согласно техническим данным составляет 2 Gb/s, а реальная порядка 1—1.5 Gb/s.

Литература

1. [Электронный ресурс]. — Режим доступа : http://developer.nvidia.com/object/cuda.html, свободный.

Оптимизация выполнения параллельных MPI-программ на мультикластерных вычислительных системах

М. Г. Курносов
Сибирский государственный университет
телекоммуникаций и информатики, Новосибирск

Введение

Одной из основных тенденций современной индустрии высокопроизводительной обработки информации является построение масштабных пространственно-распределенных вычислительных и GRID-систем. В архитектурном плане распределенная вычислительная система (ВС) представляется как множество элементарных машин (ЭМ), взаимодействие между которыми осуществляется через коммуникационную среду [1].

Коммуникационные среды современных распределенных вычислительных систем (ВС) имеют иерархическую организацию, каждый уровень которых характеризуется своими значениями показателей производительности. Например, в GRID-системах и пространственно-распределенных мультикластерных ВС нулевой уровень коммуникационной среды — сеть связи между вычислительными кластерами, первый уровень — сеть связи внутри кластеров, второй уровень — среда доступа процессоров вычислительного узла к общей памяти.

В настоящее время основным подходом к организации параллельных вычислений на таких системах является использование библиотек стандарта MPI. Параллельная программа, разработанная в модели передачи сообщений, может быть представлена информационным графом, вершинам которого соответствуют параллельные ветви программы, а ребрам — коммуникации между ними.

Многие параллельные программы характеризуются неоднородным информационным графом. Неоднородность выражается в объемах данных, передаваемых между ветвями, размерах используемых сообщений, а также в количестве смежных ветвей у каждого процесса.

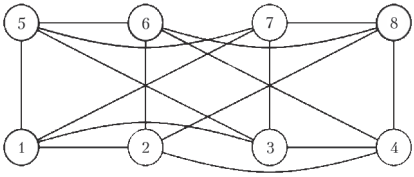


Рис. 1. Информационный граф теста HPL (NP=8, PMAP=1, BCAST=5)

На рис. 1 приведен информационный граф теста HPL (High-Performance Linpack), реализующего LU-разложение для решения системы линейных алгебраических уравнений. Время выполнения указанного теста на 4 узловом кластере, укомплектованном двухпроцессорными узлами на базе двухъядерных процессоров Intel Xeon 5150 и сетью связи стандарта Gigabit Ethernet, зависит от распределения ветвей по процессорным ядрам — распределения рангов MPI-процессов по процессорным ядрам. Например, время выполнения теста на указанном кластере с распределением ветвей (1, 2, 3, 4) (5, 6, 7, 8) ( ) ( ) составляет 118 с (44 GFLOPS, скобками обозначены узлы, а числа — номера ветвей). Очевидно, что целесообразнее распределять ветви так, чтобы интенсивно взаимодействующие ветви передавали сообщения через быстрые каналы связи, например, через общую память узла. Так время выполнения теста HPL с распределением ветвей (1, 3, 5, 7) (2, 4, 6, 8) ( ) ( ) составляет 100 с (53 GFLOPS).

Поэтому при выполнении параллельных MPI-программ на (мульти)кластерных ВС целесообразно учитывать производительности каналов связи коммуникационных сред и структуры информационных графов задач.

Задача оптимального назначения ветвей параллельной программы на процессорные ядра распределенной ВС с целью минимизации времени выполнения приложения ставится как задача дискретной оптимизации.

Постановка задачи

Пусть распределенная ВС состоит из H подсистем, объединенных каналами связи. Введем обозначения: $S=\{1, 2, ..., H\}$ — множество подсистем образующих распределенную ВС; n_i — количество процессорных ядер, входящих в состав подсистемы $i \in S$; N — количество процессорных ядер, входящих в распределенную

ВС; ω_i — производительность процессорного ядра i-ой подсистемы ($[\omega_i]$ = FLOPS, i ∈ S).

Коммуникационная среда i-ой подсистемы допускает представление в виде дерева высоты h_i. Для каждого элемента на уровне $k \in \{1, 2, ..., h_i - 1\}$ задано количество n_{ik} его дочерних элементов. Дочерними элементами нулевого уровня являются вычислительные подсистемы. Далее считаем $n_{i0} = H$.

Каждый уровень $k \in \{1, 2, ..., h_i\}$ коммуникационной среды i-ой подсистемы характеризуется своими значениями показателей производительности: $l_i^{(k)}(a,b,m)$ — зависимость латентности канала связи между элементами a и b на уровне k от размера m передаваемого сообщения ($a,b \in \{1,2,...,n_{ik}\}$, $\left[l_i^{(k)}(a,b,m)\right]$ = с). Аналогично $b_i^{(k)}(a,b,m)$ — пропускная способность канала связи ($\left[b_i^{(k)}(a,b,m)\right]$ = бит/с), а $l^{(0)}(a,b,m)$ и $b^{(0)}(a,b,m)$ определяют производительность каналов связи нулевого уровня коммуникационной среды.

Обозначим через $C = \{c_1, c_2, ..., c_N\}$ множество процессорных ядер, входящих в распределенную ВС. За g_r обозначим номер подсистемы, которой принадлежит процессорное ядро с номером $r \in \{1,2,...,N\}$. Каждый элемент $c_r = \left(e_0^r, e_1^r, ..., e_{h_{g_r}-1}^r\right)$, $c_r \in C$ определяет координату размещения процессорного ядра в распределенной ВС (номер кластера, узла, процессора и т. д., см. рис. 2).

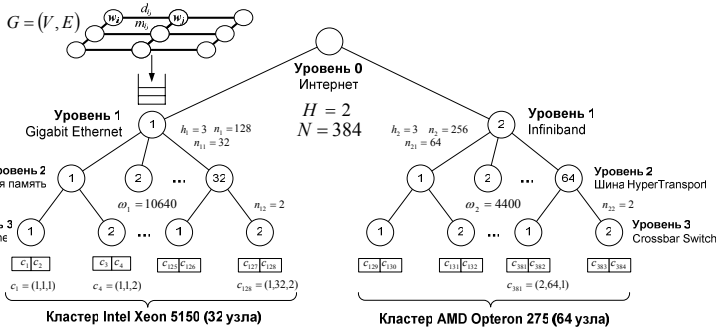


Рис. 2. Пример иерархической организации коммуникационной среды

Пусть процедура назначения ветвей параллельной программы на процессорные ядра осуществляется с подсистемы $s \in S$, а поступившая на выполнение параллельная программа представлена

ее информационным графом $G=(V,E)$. Где $V=\{1,2,...,M\}$ — множество параллельных ветвей программы; $E\subset V\times V$ — множество информационно-логических связей между ее параллельными ветвями. Обозначим: w_i — количество элементарных операций (арифметических и логических) выполняемых ветвью $i\in V$; d_{ij} — объем данных передаваемый по ребру $(i,j)\in E$ за время выполнения параллельной программы ($\lceil d_{ij} \rceil$ = байт); m_{ij} — средний размер сообщения передаваемого по ребру $(i,j)\in E$ за одну операцию приема/передачи ($\lceil m_{ij} \rceil$ = байт); z — размер файла и данных параллельной программы ($[z]$ = байт).

Требуется найти инъективную функцию $f: V\to C$, ставящую в соответствие ветвям параллельной программы процессорные ядра распределенной ВС. Искомая функция допускает представление в виде вектора:

$$X=\{x_{ij}: i\in V, j\in C\},\ x_{ij}=\begin{cases}1, & \text{если } f(i)=j;\\ 0, & \text{иначе.}\end{cases}$$

Качество назначения ветвей параллельной программы на процессорные ядра будем оценивать ожидаемым временем ее обслуживания, которое складывается из времени доставки программы на процессорные ядра и времени выполнения ветвей на них.

Время взаимодействия $t^{(k)}(i,j,p,q)$ параллельных ветвей i и j будем оценивать через латентность и пропускную способность канала связи между процессорными ядрами p и q, на которые они назначены:

$$t^{(k)}(i,j,p,q)=\begin{cases}l^{(0)}\left(e_0^p,e_0^q,m_{ij}\right)\cdot\dfrac{d_{ij}}{m_{ij}}+\dfrac{d_{ij}}{b^{(0)}\left(e_0^p,e_0^q,m_{ij}\right)}, & \text{если } k=0,\\ l_{g_p}^{(k)}\left(e_k^p,e_k^q,m_{ij}\right)\cdot\dfrac{d_{ij}}{m_{ij}}+\dfrac{d_{ij}}{b_{g_p}^{(k)}\left(e_k^p,e_k^q,m_{ij}\right)}, & \text{иначе,}\end{cases}$$

где k — номер уровня, являющийся ближайшим общим предком для ядер p и q.

Учитывая инъективность функции f, получаем задачу (1)—(4) оптимального назначения ветвей параллельной программы на процессорные ядра распределенной ВС с целью минимизации времени ее выполнения.

$$F(X)=\max_{i\in V}\left\{\sum_{j=1}^{N}x_{ij}\cdot t^{(0)}\left(s,g_j,z\right)\right\}+$$
$$+\max_{i\in V}\left\{\sum_{j=1}^{N}x_{ij}\cdot\frac{\varpi_i}{\omega_{g_j}}+\sum_{j=1}^{M}\sum_{p=1}^{N}\sum_{q=1}^{N}x_{ip}\cdot x_{jq}\cdot t^{(k(p,q))}\left(i,j,p,q\right)\right\}\to\min_{(x_{ij})} \quad (1)$$

при ограничениях:

$$\sum_{j=1}^{N}x_{ij}=1,\ \ i=1,2,...,M \quad (2)$$

$$\sum_{i=1}^{M}x_{ij}\le 1,\ \ j=1,2,...,N \quad (3)$$

$$x_{ij}\in\{0,1\},\ \ i\in V, j\in C \quad (4)$$

Ограничения (2), (4) гарантируют назначение каждой ветви параллельной программы на единственное процессорное ядро, ограничения (3) обеспечивают назначение на ядро не более одной ветви.

Задача (1)—(4) представляет собой задачу дискретной оптимизации, мощность множества D допустимых решений которой оценивается величиной $|D|=A_N^M=N!/(N-M)!$. Для большемасштабных распределенных ВС отыскание точного решения связано со значительной вычислительной сложностью. Учитывая данное обстоятельство, целесообразно применение приближенных методов решения задачи.

Методы решения

На основе метаэвристики имитации отжига (Simulated Annealing) [2, 3] разработан стохастический алгоритм PTA_SA (Parallel Task Allocation Simulated Annealing), позволяющий отыскивать приближенные решения задачи за приемлемое время. Общая схема алгоритма выглядит следующим образом. На первом шаге алгоритма строится начальное допустимое решение $x^{(0)}\in D$. На k-ом шаге имеется текущее решение $x^{(k)}\in D$ и наилучшее на данный момент $x^*\in D$. Шаг начинается с выбора случайным образом соседнего решения $y\in N\left(x^{(k)}\right)$ из окрестности

текущего. Если значение целевой функции от соседнего решения меньше значения целевой функции от текущего, то соседнее решение принимается за текущее. Кроме того, даже если соседнее решение по целевой функции превосходит текущее, то с вероятностью

$$P\left(x^{(k)}, y, k\right) = \begin{cases} 1, & \text{если } F(y) \le F(x); \\ \exp\left(\left(F(x) - F(y)\right)/c_k\right), & \text{иначе.} \end{cases}$$

соседнее решение принимается за текущее. Данный шаг позволяет алгоритму имитации отжига не «застревать» в локальных оптимумах и исследовать больше количество допустимых решений.

Результаты экспериментов

Разработанные алгоритмы вложены в программный пакет mpitaskmap (см. рис. 3), в который входя средства анализа протоколов выполнения MPI-программ и построения их информационных графов, средства оценки производительности каналов связи, а также модуль распределения ветвей параллельных программ по процессорным ядрам.

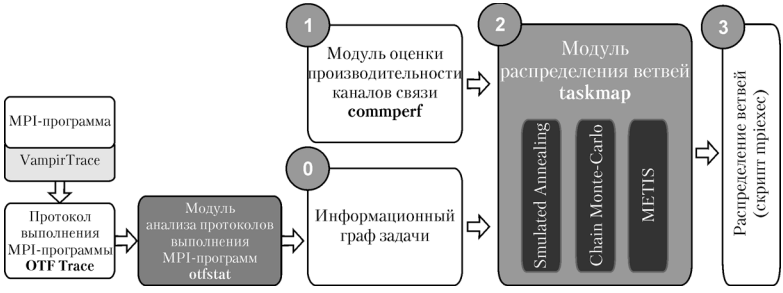


Рис. 3. Функциональная структура программного пакета mpitaskmap

На вычислительных кластерах Центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики (ЦПВТ СибГУТИ) и Лаборатории вычислительных систем института физики полупроводников СО РАН (ИФП СО РАН) проведено исследование разработанных алгоритмов и программного пакета. Использовались вычислительные кластеры следующих конфигураций:

- кластер ЦПВТ СибГУТИ: 4 узла: 2x Intel Xeon 5150, сети Gigabit/Fast Ethernet;
- кластер Лаборатории вычислительных систем ИФП СО РАН: 5 узлов: 2x AMD Opteron 248—252, сети Gigabit/Fast Ethernet.

В качестве тестовых задач использовались MPI-программы из пакетов NPB (NAS Parallel Benchmarks), HPL и SPEC MPI2007.

На рис. 4 и 5 приведены структуры информационных графов тестов NPB CG и HPL, а в табл. 1, результаты запуска тестов на различных сетях связи с распределением ветвей по ядрам различными алгоритмами.

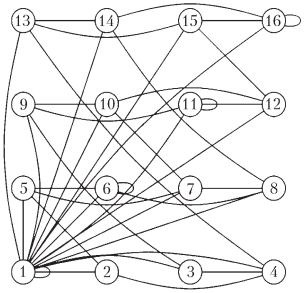


Рис. 4. Структура теста NPB CG

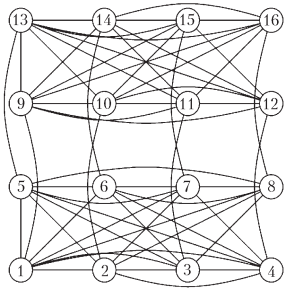


Рис. 5. Структура теста HPL

Таблица 1

Результаты экспериментов

Сеть	RR (mpiexec), с.	PTA_SA, с.	%	Xsubopt
Тест NPB CG, кластер: 5 узлов 2 x AMD Opteron 248				
Fast Ethernet	402.73	140.80	65	(2, 4) (6, 8) ( ) (5, 7) (1, 3)
Gigabit Ethernet	107.72	56.06	47	(2, 4) (6, 8) ( ) (5, 7) (1, 3)
Тест NPB CG, кластер: 4 узла 2 x Intel Xeon 5150				
Fast Ethernet	726.02	400.36	44	(1,2,3,4) (13,14,15,16) (5,6,7,8) (9,10,11,12)
Gigabit Ethernet	97.56	42.05	57	(1,2,3,4) (9,10,11,12) (5,6,7,8) (13,14,15,16)
Тест HPL, кластер 5 узлов 2 x AMD Opteron 248				
Fast Ethernet	1108.69	911.81	17	(5,6,13,14) (3,4,11,12) (7,8,15,16) (1,2,9,10)
Gigabit Ethernet	263.15	231.72	12	(5,6,13,14) (3,4,11,12) (7,8,15,16) (1,2,9,10)

Заключение

Результаты исследования разработанных алгоритмов и программного пакета показывают, что даже на сосредоточенных SMP-кластерах учет производительности каналов связи коммуникационной сред и структур информационных графов задач позволяет сократить время выполнения MPI-программ.

Литература

1. **Хорошевский В. Г.** Архитектура вычислительных систем. — М. : МГТУ им. Н. Э. Баумана, 2005.

2. **Береснев В. Л.** Дискретные задачи размещения и полиномы от булевых переменных. — Новосибирск : ИМ СО РАН, 2005.

3. **Кочетов Ю. А.** Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения. — М. : МГУ, 2001. — С. 87—117.

Методы отладки функционально-потоковых параллельных программ

Ю. В. Удалова
Сибирский федеральный университет, Красноярск

Разработка программного обеспечения немыслима без отладки и тестирования. В настоящее время эти этапы разработки программ хорошо проработаны для последовательного программирования. Существуют разнообразные отладчики, обеспечивающие пошаговый анализ кода и отображающие состояние переменных. Они также обеспечивают использование разнообразных приемов, позволяющих пропускать уже отлаженные фрагменты программ и акцентировать внимание только на требуемых функциях или процедурах.

Перенос методов отладки последовательных программ на параллельные вычисления сопровождается определенными трудностями, обуславливаемыми спецификой организации параллельных процессов. В частности, необходимы дополнительные функции, позволяющие анализировать конфликтные и тупиковые ситуации. Но в настоящее время методы отладки параллельных программ развиваются. Достигнуты определенные успехи в разработке соответствующих средств для систем, использующих потоки или процессы, расширяющие технику последовательного выполнения программ.

Существуют разнообразные способы построения параллельных программ, ряд которых существенно отличается от широко используемых в настоящее время. Их исследование может привести к перспективным технологиям, повышающим эффективность процесса разработки и использования параллельных программ. К таким способам построения следует отнести и функционально-потоковое параллельное программирование, базирующееся на модели вычислений, обеспечивающей описание ресурсно-независимого параллелизма. На основе этой модели разработан язык программирования Пифагор, позволяющий писать архитектурно-независимые параллельные программы.

Методы отладки функционально-потоковых параллельных программ требуют использования специфических приемов, что

обуславливается спецификой модели вычислений, в которой отсутствует понятие переменной. Выполняемые функции связаны напрямую, а существующие типы данных требуют нетривиальных вычислений и методов отображения результатов.

Целью работы является исследование и систематизация методов отладки функционально-потоковых параллельных программ. На данный момент разработаны три режима отладки, использующие информационный граф функционально-потоковой программы, также реализованы различные способы отображения процесса отладки.

Режимы проведения отладки

Отладка функционально-потоковой параллельной программы может проходить в одном из трех режимов: пошаговой отладке, отладке слоев и отладке ветвей.

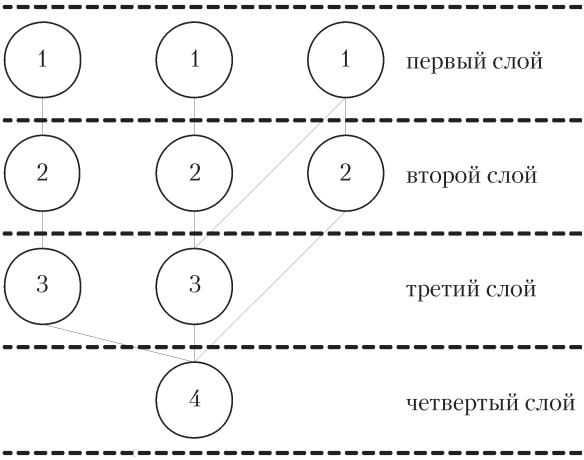


Рис. 1. Пример выделения слоев в программном графе

Режим пошаговой отладки потоковой параллельной программы, подобен режиму отладки последовательной. Возможность использования такого режима обусловлена моделью вычислений функционально-потоковой параллельной программы, определяющей программу как информационный граф, в котором вершины являются операторами, а дуги определяют связи между операторами. Параллельное выполнение основано на том, что од-

новременно выполняться могут только те операторы, между которыми нет информационных связей. Чтобы оператор мог быть вычислен, необходимо выполнение всех тех операторов-вершин, которые образуют путь к данному оператору. Отсюда следует, что непосредственное вычисление программы может пойти по другому пути, чем пошаговая отладка. Но при этом, такая ситуация не приведет к возникновению конфликтов, таких, как появление невычисленных вершин, бесконечное ожидание вычисления предыдущего результата, возникновение данных, отличных от значений, полученных при выполнении программы. Например, если выполняется множество независимых операторов, то не важно в каком порядке они были вычислены, процесс отладки все равно показывает верный результат для каждого оператора, и позволяет выявить ошибку. Если последовательность выполнения операторов идет в глубину графа, каждая отдельная ветвь может быть выполнена отладчиком только в определенном порядке, и различные ветви, не обладающие общим путем, не могут содержать взаимосвязанные операторы. Благодаря этому, переход пошагового отладчика от выполнения одной ветви к другой не приводит к противоречиям, на какой бы глубине они не произошли.

Режим отладки слоев (ярусов) предполагает, что на первом шаге отладки выполняются все те программные операторы, которые не требуют никаких входных данных, кроме входного параметра функции — первый слой. После того, как эти операторы были вычислены, в программном графе соответственные вершины помечаются как выполненные, и на следующем шаге для вычисления выбираются все те операторы-вершины, которые имеют на своем входе только вычисленные данные. Таким образом, формируется следующий слой.

Отладка по слоям предоставляет пользователю возможность наглядно увидеть количество слоев программы и количество операторов в каждом слое, т. е. фактически оценить уровень параллелизма написанной программы. Кроме того, такая схема выполнения программы при отладке наиболее точно показывает работу модели вычислений функционально-параллельных программ.

В режиме отладки слоев за один шаг отладки выполняются несколько операторов, и здесь возникает вопрос о поведении

системы при получении ошибки на одном или нескольких операторах. Такая ситуация не является критической вследствие того, что операторы одного слоя независимы между собой. Но группа операторов в следующем слое получает ошибку как входной параметр. В функционально-потоковой программе, если некоторый программный оператор получает на входе ошибку, то он все равно производит над ней необходимые действия. Чаще всего, в этом случае формируются выходные данные, также содержащие ошибку, и такие выходные данные обрабатываются последующими операторами. В процессе отладки можно рассмотреть все последствия распространения возникшей ошибки.

Режим отладки слоев обладает следующей спецификой, если среди совокупности операторов существуют вызовы функций, то такие вызовы выполнятся сразу в полном объеме, и лишь по завершении вычисления всего слоя, будет произведена отладка вызванной функции. Такая схема работы, с одной стороны, не разрушает целостности восприятия слоев, с другой стороны, делает затруднительным, но не невозможным поиск ошибки не завершения рекурсивного вызова функции.

Режим отладки ветвей предполагает, что для выполнения на каждом шаге отладки выбираются операторы независимой ветви графа. Такие операторы всегда выполняются в строгой последовательности, определенной графом программы. Этот режим отладки позволяет четко выделять в параллельной программе линию операторов, которые могут быть выполнены только последовательно один за другим, но при этом не нуждаются в получении каких либо других входных значений из других вершин на данном этапе отладки. При этом цепочка операторов — вершин, выполняемая на отдельном шаге отладки, является логически связанной, все операторы, стоящие в цепи служат одной и той же цели вычисления. Подобная схема отладки не слишком хорошо согласуется с моделью вычислений потоковых параллельных программ, однако здесь не возникает несоответствия между данными, полученными на шагах отладки, и данными, вычисленными при выполнении программы. Режим отладки ветвей выполняет односвязную последовательность команд за один шаг, что в определенных случаях может сократить время проверки программы.

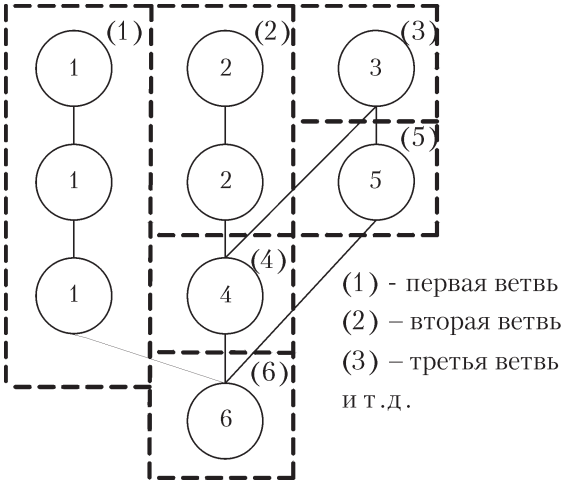


Рис. 2. Пример выделения ветвей в программном графе

Описанные режимы отладки реализованы в среде разработки программ на функционально-потоковом параллельном языке ПИФАГОР. Каждый из режимов обеспечивает использование графического и текстового представления программы

Графическое представление программы при отладке состоит в следующем. Перед началом вычисления любой функции строится ее информационный граф, при выполнении оператора отмечается соответствующая ему вершина графа. Каждая вершина содержит доступную пользователю информацию об операторе и результате его вычисления.

Текстовое представление отладки оперирует списком функций или сверткой функций. Список функций показывает историю шагов отладки, сохраняя текст выполняемой функции на каждом шаге. Свертка функции отображает этот же процесс более компактно, запоминая только предыдущее и вычисленное состояние функции.

Перед отладкой программы, вызывается диалог выбора режима отладки.

Здесь пользователь определяет режим отладки, способ отображения процесса отладки и ряд других опций. «Функция для запуска» — это функция, которая будет запущена для отладки, она выбирается из списка, который формируется автоматически. «Ар-

гумент» — входной параметр для запускаемой функции, он задается пользователем. Функции, не имеющие входного параметра, помечаются в списке функций восклицательными знаками.

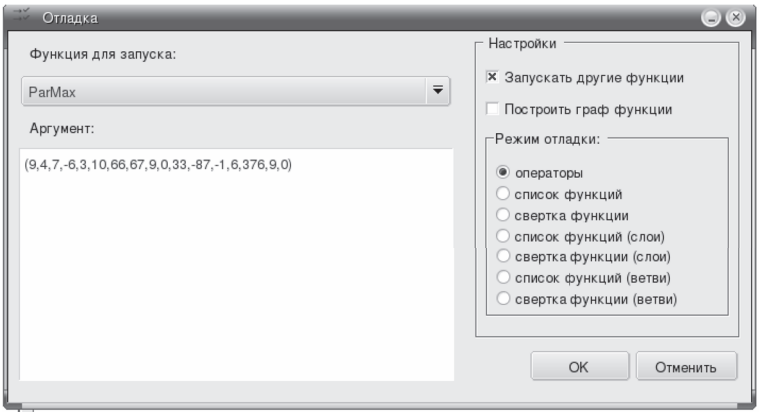


Рис. 3. Диалог выбора режима отладки

Если флаг «Запускать другие функции» отключен, то другие функции не будут запускаться для отладки, это дает возможность сосредоточиться на выполнении выбранной функции, не переключая внимание на просмотр ненужных вычислений. Если флаг включен, отладка проходит полный путь, начиная с запуска, выбранной пользователем функции. Флаг «Построить граф функции» позволяет по желанию пользователя построить граф функции, тогда во время отладки программы будет доступен как граф функции, так и текстовое отображение процесса отладки.

Режимы «операторы», «список функции», «свертка функции» — это различные способы отображения режима пошаговой отладки, «список функции (слои)», «свертка функции (слои)» — различные способы отображения режима отладки слоев. И соответственно «список функции (ветви)», «свертка функции (ветви)» — это различные способы отображения режима отладки ветвей.

Таким образом, режим пошаговой отладки имеет один дополнительный способ текстового отображения — «операторы», по сравнению с режимами отладки ветвей и отладки слоев. Данный способ отображения подобен большинству стандартных способов отображения процесса отладки для последовательных про-

грамм. Существует текстовое поле, содержащее всю программу, по которому перемещается цветовой указатель, показывающий, какой оператор выполняется на данном шаге. Также на главном окне располагается дополнительное текстовое поле, отображающее список всех вычисленных операторов, в виде: 1) исходная строка оператора, 2) строка оператора после подстановки в нее всех значений, вычисленных на предыдущих шагах, 3) вычисленное значение оператора. Программный оператор, вычисляемый на текущем шаге, ставится в конец списка.

В своей пользовательской части такой способ отображения отладки отличается от аналогичных способов отображения отладки последовательных программ путем выполнения операторов. Если для последовательной программы операторы выполняются строго в том порядке, в каком они записаны в тексте программы, то для параллельной программы такая ситуация, скорее является исключением, здесь последовательность вычисления операторов, определяется не расположением в тексте, а с помощью предварительного построения схемы программного графа. Таким образом, оператор, стоящий в n-ой строке, может быть вычислен, ранее оператора, стоящего в $(n-m)$-ой строке.

Подобная ситуация, не противоречит модели вычислений потоковых параллельных программ, но является не лучшей для восприятия процесса отладки пользователем. В связи с этим, были разработаны такие способы текстового представления отладки как создание списка функций и свертка функций, применимые ко всем трем описанным режимам отладки.

Создание списка функций предполагает, что на первом шаге отладки текстовое поле дополняется телом функции, запущенной для отладки, в котором все операторы, вычисляемые на данном шаге, выделяются зеленым цветом. После получения вычисленных значений в поле текста снова дублируется тело функции, но уже с подставленными в него полученными на этом шаге отладки значениями, которые также выделяются цветом. При переходе к следующему шагу отладки поле текста снова дополняется функцией, в которую подставляются вычисленные значения, причем функция содержит и все предыдущие полученные значения. Таким образом, текстовое поле содержит список функций, где каждая последующая функция содержит те операторы, которые не были выполнены на предыдущих шагах,

и также содержит все данные, которые уже были вычислены, выделенные цветом.

Свертка функции отличается от списка функций, только формой отображения информации, получаемой при выполнении шагов отладки. Если была выбрана свертка функции, то на экране отображается исходное тело функции, которое не меняется при отладке, а используется для выделения тех операторов, которые выполняются на данном этапе отладки. Кроме этого, отображается и текущее состояние функции, в котором записаны все предыдущие вычисления, и также добавлены данные, полученные на текущем шаге, последние выделяются цветом. Таким образом, начиная с некоторого шага отладки, исследуемая функция начинает «свертываться». На момент возврата результата вычислений функции, тело функции содержит только записи вида: значение >> идентификатор.

Оба рассмотренных способа текстового отображения процесса отладки имеют как достоинства, так и недостатки. Свертка функции отображает процесс отладки более компактно и наглядно, чем список функций, но при этом не позволяет просмотреть историю всех изменений и переходов.

Графическое отображение отладки выполнено посредством визуализации программного графа отлаживаемой функции. Графическое отображение используется совместно с текстовым отображением, оно может быть включено по желанию пользователя. В программном графе еще не вычисленные вершины имеют серый цвет, вычисленные — зеленый. По нажатию на любую вершину графа появляется информация об исходной строке соответствующего этой вершине оператора, и вычисленном значении оператора, если оно существует. Каждая вершина отображает тип соответственного ей оператора, посредством специального символа. Граф функции всегда располагает операторы-вершины одного слоя на одном уровне, не зависимо от того, какой режим отладки был выбран.

Каждый из рассмотренных режимов отладки и способов отображения процесса отладки реализован практически и может быть использован для отладки функционально-потоковых параллельных программ.

Эффективность каждого отдельного режима зависит от свойств исполняемой программы. Так, простая функция может

содержать только одну ветвь, и это значит, что при выборе последнего режима все вычисления будут выполнены за один шаг, тогда как отладка по слоям, в этом же случае, будет полностью аналогична пооператорной отладке. Плюсы и минусы описанных режимов находятся в зависимости и от восприятия процесса отладки пользователем. Но, в общем случае, пооператорная отладка позволяет сосредоточиться на каждом программном операторе, отладка ветвей выполняет прямые последовательности команд за один шаг, что может сократить время проверки программы, отладка слоев дает представление об уровне параллелизма рассматриваемой функции.

Различные текстовые способы отображения отладки предоставляют такие возможности, как сохранение истории вычислений и вызовов функций, выделение текущих результатов отладки. Графическое представление отладки позволяет увидеть историю вычисления отдельной функции и эффективно проследить все последствия возникновения ошибки, наглядно выделяет ветви и слои, показывает уровень параллелизма функции.

Заключение

Исследование методов отладки и отображения информации при отладке функционально-потоковых параллельных программ позволило выделить ряд основных направлений развития инструментальных средств, повышающих эффективность процесса разработки. Предлагаемые методы реализованы в рамках экспериментальной среды. Полученная реализация повышает эффективность отладки и позволяет более наглядно представлять параллельные вычисления, обеспечивая их различное отображение. Дальнейшие исследования, проводимые на основе разработанной среды, позволят выделить наиболее эффективные методы отладки функционально-потоковых параллельных программ.

О целесообразности изменения количества уровней в многоуровневой памяти

Д. В. Биматов
Томский государственный университет, Томск

Рассмотрим многоуровневую подсистему памяти с количеством уровней в иерархии равным $U > 1$ (рис. 1). На каждом уровне $u = \overline{1...U-1}$, кроме последнего, находится кэш; последний уровень $u = U$ занимает оперативное запоминающее устройство (ОЗУ).

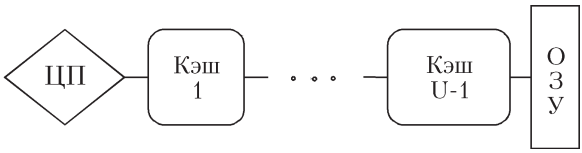


Рис. 1. Схема многоуровневой памяти

Среднее время доступа к адресуемым объектам многоуровневой памяти с U уровнями блокирующего типа определяется соотношением [1]:

$$T_U\left(K_1,\ R_1,\ K_2,\ R_2,\ ...,\ K_U,\ R_U\right) = \sum_{u=1}^{U} K_u \prod_{i=1}^{u-1} R_i \qquad (1)$$

где R_i — вероятность промаха в кэш уровня i, а K_i — время выбора блока из кэша уровня i, выраженное в тактах. В рамках рассматриваемой модели очевидно, что $R_U = 0$.

Предположим, что в подсистему иерархической памяти добавляется еще один уровень памяти перед уровнем с номером a. Пусть время доступа к добавляемому уровню памяти — K_*, а вероятность промаха — R_*. Естественно, что параметры добавляемого уровня памяти должны соответствовать логике построения многоуровневой памяти: кэш более высокого уровня характеризуется большим временем доступа и меньшей вероятностью промаха:

$$K_{a-1} \le K_* \le K_a$$
$$R_{a-1} \ge R_* \ge R_a$$

Целесообразность добавления уровня памяти в существующую систему перед уровнем с номером a выражается неравенством [2]:

$$\Delta T^{(a)} = T_U\left(K_1, R_1, ..., K_U, R_U\right) -$$
$$-T_{U+1}^{(a)}\left(K_1, R_1, ..., K_{a-1}, R_{a-1}, K_*, R_*, K_a, R_a, ..., K_U, R_U\right) \geq 0$$

Здесь $T_{U+1}^{(a)}\left(K_1, R_1, ..., K_{a-1}, R_{a-1}, K_*, R_*, K_a, R_a, ..., K_U, R_U\right)$ — среднее время доступа к подсистеме памяти с дополнительно введенным на a-ом уровне кэшем. Подставляя в последнее неравенство соотношение (1), после приведения подобных получим:

$$\frac{K_*}{1-R_*} \leq \sum_{u=a}^{U} K_u \prod_{i=a}^{u-1} R_i \qquad (2)$$

Неравенство (2) означает, что добавление еще одного уровня памяти в иерархию перед уровнем с номером а целесообразно, если отношение скорости обращения к добавляемому уровню памяти к вероятности попадания в добавляемый уровень памяти меньше среднего времени доступа к уровням памяти более высокого уровня. Для удобства выражение (2) можно переписать в виде

$$K_* \leq \Pi_* \cdot \left(\sum_{u=a}^{U} K_u \prod_{i=a}^{u-1} R_i \right)$$

или в виде

$$\Pi_* \geq \frac{K_*}{\sum_{u=a}^{U} K_u \prod_{i=a}^{u-1} R_i}$$

где $\Pi_* = 1 - R_*$ — вероятность попадания в добавляемый кэш.

Рассмотрим двухуровневую подсистему памяти «Кэш-ОЗУ». Целесообразность добавления кэша первого уровня будет выражаться неравенством:

$$\Delta T^{(1)} = T_2\left(K_1, R_1, \mathrm{K_{ОЗУ}}\right) - T_3^{(1)}\left(K_*, R_*, K_1, R_1, K_{ОЗУ}\right) \geq 0$$

Отсюда получаем:

$$\frac{K_*}{1-R_*} \leq K_1 + R_1 K_{ОЗУ}$$

Целесообразность добавления кэша второго уровня к исходной подсистеме памяти задается условием:

$$\Delta \mathrm{T}^{(2)} = T_2\left(\mathrm{K}_1, R_1, K_{ОЗУ}\right) - T_3^{(2)}\left(K_1, R_1, K_*, R_*, K_{ОЗУ}\right) \geq 0$$

которое приводит к неравенствам:

$$\frac{K_*}{1-R_*} \le K_{ОЗУ}$$

В общем случае вероятность промаха в кэш R (или вероятность попадания П = 1 – R) не известна и определяется как параметрами подсистемы памяти (объем кэша — V_i, объем ОЗУ — $V = V_{ОЗУ}$, коэффициент ассоциативности — A_i, где i — уровень кэша), так и выполняемой на вычислительной системе смеси задач. В данной работе используется модель с идеальной стратегией вытеснения блоков [3], т. е. предполагается, что заранее известно распределение вероятностей востребованности блоков основной памяти вычислителем:

$$p(i),\ \ i = \overline{0, \mathrm{V}-1},\ \ \sum_{i=0}^{V-1} p(i) = 1. \qquad (3)$$

Полагаем, что вероятности востребованности блоков основной памяти (3) определены на интервале $m = \overline{0, \alpha A_1 - 1}$ и убывают с ростом m по усеченному геометрическому закону (заданному на конечной области определения):

$$p(g + G_1 m) = p(g) \cdot q^m,\ g = \overline{0,\ G_1 - 1}$$

где G_1 — количество групп в кэше уровня 1.

Здесь $\frac{1}{A_1} \le \alpha \le \frac{M_1}{A_1}$ — доля ОЗУ, занятая востребованными вычислителем приложениями, выраженная в количестве объемов кэша первого уровня, а $0 \le q \le 1$ — параметр усеченного геометрического распределения.

Для усеченного геометрического распределения получены аналитические зависимости для вычисления вероятности промаха в кэш уровня 1

$$R_1 = 1 - \frac{1 + q - 2q^{A_1} + q^{\alpha A_1}(1-q)}{(1+q)\cdot\left(1 - q^{\alpha A_1}\right)},\ \ 0 \le q < 1,$$

и в кэш произвольного уровня u:

$$R_u = 1 - \frac{1 + q^{e_u} - 2q^{e_u A_u} + q^{\alpha A_1}\left(1 - q^{e_u}\right)}{\left(1 + q^{e_u}\right)\cdot\left(1 - q^{\alpha A_1}\right)},\ \ 0 \le q < 1,$$

где $e_u = G_u / G_1$ — отношение количества групп в кэше уровня u и кэше уровня 1.

Для равномерного распределения ($q = 1$) выражения для вычисления вероятности промаха в кэш многоуровневой памяти выглядят следующим образом:

$$R_1 = 1 - \frac{1}{\alpha},\ R_u = 1 - \frac{V_u}{\alpha V_1},\ u = \overline{2, U-1}.$$

Рис. 2 иллюстрирует в пространстве быстродействия и вероятности промаха в кэш добавляемого уровня области целесообразности добавления кэша к двухуровневой подсистеме памяти с параметрами: $R_1 = 0{,}5$, $K_1 = 8$, $K_{\text{ОЗУ}} = 64$. Область $0 \le R_* \le R_1$ соответствует добавлению уровня между центральным процессором и существующим кэшем (добавление кэша первого уровня), а область $R_1 \le R_* \le 1$ — добавлению кэша между существующим кэшем и ОЗУ (добавление кэша второго уровня).

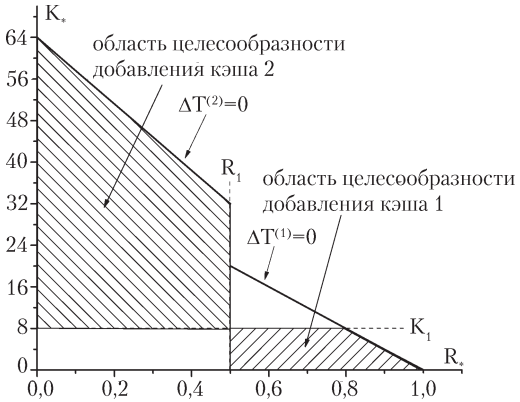


Рис. 2. Области целесообразности введения кэша первого и второго уровней

Рис. 3—5 показывают эффект от добавления кэша второго уровня к двухуровневой системе памяти с параметрами: $V_1 = 128$, $V_{\text{ОЗУ}} = 512$ ($\alpha = 4$), $K_1 = 8$, $K_{\text{ОЗУ}} = 64$ при равномерном распределении востребованности блоков ОЗУ вычислителем. На рис. 3 изображена зависимость среднего времени доступа от времени обращения к добавляемому кэшу K_*. Горизонтальная линия соответствует среднему времени доступа к исходной двухуровневой памяти. Видно, что чем больше объем добавля-

емого кэша, тем шире диапазон значений K_*, при которых трехуровневая память имеет меньшее среднее время доступа. Рис. 4 иллюстрирует влияние объема добавляемого кэша на среднее время доступа к модифицированной подсистеме памяти. Горизонтальная линия соответствует среднему времени доступа к исходной подсистеме памяти. Видно, что чем меньше время обращения к добавляемому кэшу, тем шире диапазон объемов добавляемого кэша V_*, при которых имеет смысл использовать трехуровневую память вместо двухуровневой. Наконец, рис. 5 представляет область целесообразности добавления кэша с объемом V_* и временем доступа K_*.

Рис. 6, 7 показывают эффект от добавления кэша второго уровня к двухуровневой подсистеме памяти при усеченном геометрическом распределении востребованности блоков ОЗУ вычислителем. Исходная подсистема памяти характеризуется параметрами: $V_1 = 16$, $V_{ОЗУ} = 64$ ($\alpha = 4$); $A_1 = 2$, $K_1 = 8$, $K_{ОЗУ} = 64$. Добавляемый кэш имеет объем V_2=32 и ассоциативность $A_2 = 2$. На рис. 6 приведен график среднего времени доступа исходной (T_2) и модифицированной ($T_3^{(2)}$) подсистем памяти от параметра усеченного геометрического распределения q при различных временах доступа к добавляемому кэшу K_2. Рис. 7 показывает выигрыш от добавления кэша при заданном параметре q и различных временах обращения к добавляемому кэшу K_2.

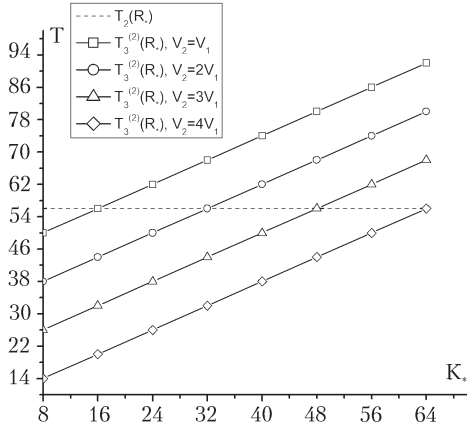


Рис. 3. Влияние времени обращения K_* к добавляемому кэшу второго уровня на среднее время доступа T при равномерном распределении

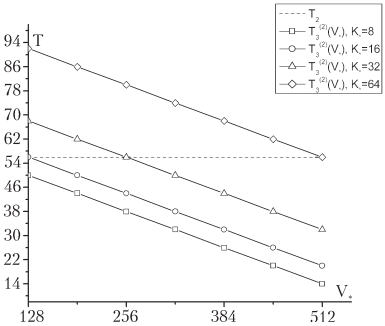


Рис. 4. Влияние объема V_* добавляемого кэша второго уровня на среднее время доступа T при равномерном распределении

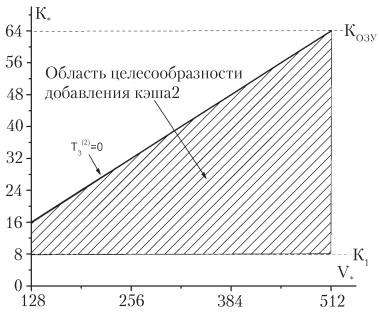


Рис. 5. Область целесообразности добавления кэша второго уровня с объемом V_* и временем доступа K_* при равномерном распределении

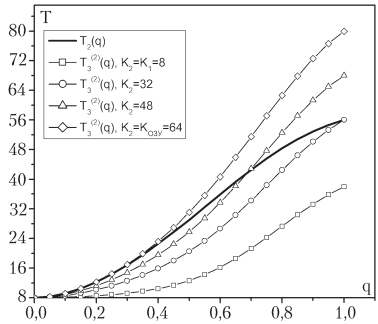


Рис. 6. Зависимость среднего времени доступа к исходной и измененной подсистеме памяти от параметра распределения q при различных временах доступа к добавляемому кэшу второго уровня K_2

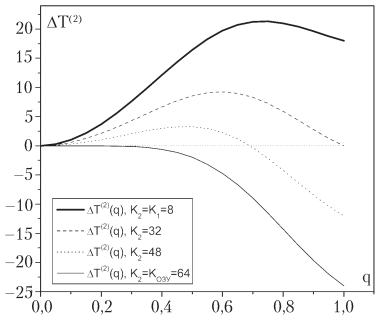


Рис. 7. Выигрыш от добавления кэша второго уровня с различными временами обращения K_2

Литература

1. **Сущенко М. С., Сущенко С. П.** Анализ эффективности многоуровневой памяти вычислительных систем // Обозрение прикл. и промышл. матем. — 2001. — Т. 8. — В. 1. — С. 336—337.

2. **Сущенко М. С., Сущенко С. П.** Условия целесообразности увеличения числа уровней иерархической памяти // Мат-лы Всерос. науч.-практ. конф. «Наука и практика: диалоги нового века». — Томск : Твердыня, 2003. — Ч. 3 Информационные технологии и математическое моделирование. — С. 194—196.

3. **Сущенко М. С., Сущенко С. П.** Анализ производительности множественного ассоциативного кэша // Вестник Томского гос. ун-та. — 2002. — № 275. — С. 218—223.

Полинейный рекуррентный метод решения СЛАУ с пятидиагональной матрицей

А. А. Фомин, Л. Н. Фомина
Кемеровский государственный университет, Кемерово

Современные методы решения краевых задач, связанных с исследованием явлений гидродинамики и тепломассопереноса, сводятся, как правило, к разностной аппроксимации многомерных дифференциальных уравнений, что, в свою очередь, приводит к построению системы линейных алгебраических уравнений (СЛАУ), матрица которой имеет большую размерность и разреженно-упорядоченную структуру [1]. Данная работа посвящена изложению итерационного экономичного устойчивого к ошибкам округления метода, отличительной особенностью которого является одномоментный учет граничных условий при построении очередного приближения численного решения во всей области постановки задачи.

Рассматривается двумерная прямоугольная область (рис. 1) $\Omega = \{(x,y) \in \Omega, 0 \le x \le xL, 0 \le y \le yL\}$ с границей

$$\Gamma = \left\{ \begin{array}{l} (x,y) \in \partial\Omega : x = 0, 0 \le y \le yL; \\ x = xL, 0 \le y \le yL, y = 0, 0 \le x \le xL; y = yL, 0 \le x \le xL \end{array} \right\}.$$

Внутри области стационарное поведение искомой функции Ф описывается дифференциальным уравнением:

$$\frac{\partial}{\partial x}\left(v_x \frac{\partial \Phi}{\partial x} \right) + \frac{\partial}{\partial y}\left(v_y \frac{\partial \Phi}{\partial y} \right) = S \qquad (1)$$

а на границе Г — граничными условиями 1-го рода $\Phi = \Phi_\Gamma$. Здесь ν_x, ν_y — коэффициенты диффузии, S — источниковый член порождения Ф.

Аппроксимация (1) выполняется методом контрольного объема, хотя принципиального значения конкретный способ дискретизации (1) не имеет, лишь бы были выполнены четыре основных правила построения разностных схем, обеспечивающих, в частности, диагональное преобладание матрицы СЛАУ [2].

Разностное уравнение имеет вид:

$$a_{P_{ij}}\Phi_{ij} = a_{E_{ij}}\Phi_{i+1j} + a_{W_{ij}}\Phi_{i-1j} + a_{N_{ij}}\Phi_{ij+1} + a_{S_{ij}}\Phi_{ij-1} + b_{ij} \qquad (2)$$

$1 \le i \le n$, $1 \le j \le m$, причем $a_P = a_E + a_W + a_N + a_S$. Здесь i, j — индексы, а n, m — количество узлов сеточного разбиения расчетной области по координатам x, y соответственно. Разностный шаблон и соответствующее обозначение узлов представлены на рис. 2.

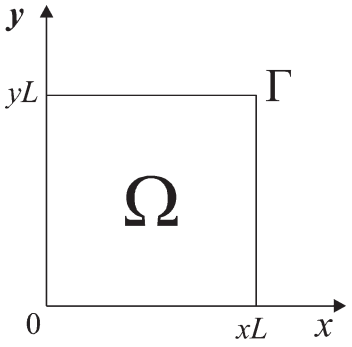


Рис. 1

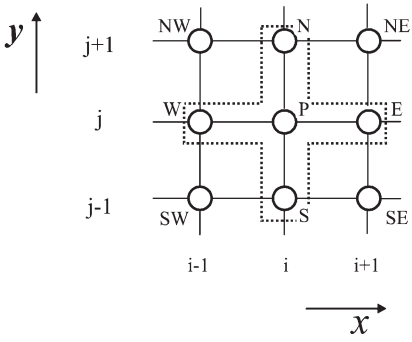


Рис. 2

Расчетная процедура на каждой итерации состоит из двух этапов: сначала находится промежуточное решение путем проведения вычислений по линиям $x = x_i$ (в начале i меняется от 1 до n — прямой ход, затем i меняется от n до 1 — обратный ход), при этом на каждой линии пробегаются узлы сетки от $j = 1$ до $j = m$ и обратно; затем находится окончательное решение итерации путем повторения алгоритма, но уже по линиям $y = y_j$.

Последовательность вычислений вдоль координаты x при построении вектора промежуточного решения $\left\{\Phi_{ij}^{05}\right\}$ состоит из следующих шагов (λ — рабочая переменная).

I. Прямой ход — возрастание индекса i от 1 до n.

1. $i = 1$:

$j = 1$: $\quad \overline{a}_{P_{11}} = \alpha_{P_{11}};\ \overline{a}_{E_{11}} = \alpha_{E_{11}};\ \overline{a}_{N_{11}} = \alpha_{N_{11}};\ \overline{b}_{11} = b_{11}$

$j = \overline{2, m-1}$: $\quad \overline{a}_{P_{1j}} = \alpha_{P_{1j}};\ \overline{a}_{E_{1j}} = \alpha_{E_{1j}};\ \overline{a}_{S_{1j}} = \alpha_{S_{1j}};\ \overline{a}_{N_{1j}} = \alpha_{N_{1j}};\ \overline{b}_{1j} = b_{1j}$

$j = m$: $\quad \overline{a}_{P_{1m}} = \alpha_{P_{1m}};\ \overline{a}_{E_{1m}} = \alpha_{E_{1m}};\ \overline{a}_{S_{1m}} = \alpha_{S_{1m}};\ \overline{b}_{1m} = b_{1m}$

2. $i = \overline{1.n-1}$:

2a. Вычисление α_{nb}, β

$j=2$: $\lambda = \overline{a}_{S_{i2}} \Big/ \alpha_{P_{i1}}$; $\alpha_{P_{i2}} = \overline{a}_{P_{i2}} - \lambda \overline{a}_{N_{i1}}$; $\alpha_{N_{i2}} = \overline{a}_{N_{i2}}$; $\alpha_{E_{i2}} = \overline{a}_{E_{i2}}$

$\overline{a}_{SE_{i2}} = \lambda \overline{a}_{E_{i1}}$; $\alpha_{NE_{i2}} = 0$; $\beta_{i2} = \lambda \overline{b}_{i1} + \overline{b}_{i2}$.

$j = \overline{3, m-1}$: $\lambda = \overline{a}_{S_{ij}} \Big/ \alpha_{P_{ij-1}}$;

$\alpha_{P_{ij}} = \overline{a}_{P_{ij}} - \lambda \alpha_{N_{ij-1}}$; $\alpha_{N_{ij}} = \overline{a}_{N_{ij}}$;

$\alpha_{E_{ij}} = \overline{a}_{E_{ij}} + \lambda \left(\alpha_{NE_{ij-1}} - 3\theta \overline{a}_{SE_{ij-1}} \right)$;

$\alpha_{NE_{ij}} = \theta \lambda \alpha_{SE_{ij-1}}$; $\alpha_{SE_{ij}} = \lambda \left(\alpha_{E_{ij-1}} + 3\theta \overline{a}_{SE_{ij-1}} \right)$;

$\beta_{ij} = b_{ij} + \lambda \left\{ \beta_{ij-1} + \alpha_{SE_{ij-1}} \left[\Phi^{k-1}_{i+1 j-2} - \theta \left(3 \left(\Phi^{k-1}_{i+1 j-1} - \Phi^{k-1}_{i+1 j} \right) + \Phi^{k-1}_{i+1 j+1} \right) \right] \right\}$;

$j = m$: $\lambda = \overline{a}_{S_{im}} \Big/ \alpha_{P_{im-1}}$; $\alpha_{P_{im}} = \overline{a}_{P_{im}} - \lambda \overline{a}_{N_{im-1}}$;

$\alpha_{E_{im}} = \overline{a}_{E_{im}} + \lambda \left(\alpha_{NE_{im-1}} - \theta \overline{a}_{SE_{im-1}} \right)$; $\alpha_{SE_{im}} = \lambda \left(\alpha_{E_{im-1}} + 2\theta \overline{a}_{SE_{im-1}} \right)$;

$\beta_{im} = b_{im} + \lambda \left\{ \beta_{im-1} + \alpha_{SE_{im-1}} \left[\Phi^{k-1}_{i+1 m-2} - \theta \left(2\Phi^{k-1}_{i+1 m-1} - \Phi^{k-1}_{i+1 m} \right) \right] \right\}$.

2b. Вычисление γ_{nb}, δ

$j = m-1$: $\lambda = \overline{a}_{N_{im-1}} \Big/ \alpha_{P_{im}}$; $\gamma_{P_{im-1}} = \overline{a}_{P_{im-1}} - \lambda \overline{a}_{S_{im}}$;

$\gamma_{E_{im-1}} = \overline{a}_{E_{im-1}}$; $\gamma_{S_{im-1}} = \overline{a}_{S_{im-1}}$;

$\gamma_{NE_{im-1}} = \lambda \overline{a}_{E_{im}}$; $\gamma_{SE_{im-1}} = 0$; $\delta_{im-1} = \overline{b}_{im-1} + \lambda \overline{b}_{im}$.

$j = \overline{m-2, 2}$ $\lambda = \overline{a}_{N_{ij}} \Big/ \gamma_{P_{ij+1}}$; $\gamma_{P_{ij}} = \overline{a}_{P_{ij}} - \lambda \gamma_{S_{ij+1}}$; $\gamma_{S_{ij}} = \overline{a}_{S_{ij}}$;

$\gamma_{E_{ij}} = \overline{a}_{E_{ij}} + \lambda \left(\gamma_{SE_{ij+1}} - 3\theta \overline{a}_{NE_{ij+1}} \right)$; $\gamma_{SE_{ij}} = \theta \lambda \gamma_{NE_{ij+1}}$;

$\gamma_{NE_{ij}} = \lambda \left(\gamma_{E_{ij+1}} + 3\theta \gamma_{NE_{ij+1}} \right)$;

$\delta_{ij} = \overline{b}_{ij} + \lambda \left\{ \delta_{ij+1} + \gamma_{NE_{ij+1}} \left[\Phi^{k-1}_{i+1 j+2} - \theta \left(3 \left(\Phi^{k-1}_{i+1 j+1} - \Phi^{k-1}_{i+1 j} \right) + \Phi^{k-1}_{i+1 j-1} \right) \right] \right\}$.

$j=1$: $\lambda = \overline{a}_{N_{i1}} \Big/ \gamma_{P_{i2}}$;

$\gamma_{P_{i1}} = \overline{a}_{P_{i1}} - \lambda\gamma_{S_{i2}}; \gamma_{E_{i1}} = \overline{a}_{E_{i1}} + \lambda\left(\gamma_{SE_{i2}} - \theta\overline{a}_{NE_{i2}}\right);$

$\gamma_{NE_{i1}} = \lambda\left(\gamma_{E_{i2}} + 2\theta\gamma_{NE_{i2}}\right);$

$\delta_{i1} = \overline{b}_{i1} + \lambda\left\{\delta_{i2} + \gamma_{NE_{i2}}\left[\Phi_{i+13}^{k-1} - \theta\left(\Phi_{i+12}^{k-1} - \Phi_{i+11}^{k-1}\right)\right]\right\}.$

2c. Вычисление p_{nb}, q

$j=1$: $p_{P_{i1}} = \gamma_{P_{i1}};\ p_{E_{i1}} = \gamma_{E_{i1}};\ p_{NE_{1j}} = \gamma_{NE_{i1}};\ q_{i1} = \delta_{i1};$

$j=\overline{2,m-1}$ $p_{P_{ij}} = \alpha_{P_{ij}} + \gamma_{P_{ij}} - \overline{a}_{P_{ij}};\ p_{E_{ij}} = \alpha_{E_{ij}} + \gamma_{E_{ij}} - \overline{a}_{E_{ij}};$

$p_{SE_{ij}} = \alpha_{SE_{ij}} + \gamma_{SE_{ij}};\ p_{NE_{ij}} = \alpha_{NE_{ij}} + \gamma_{NE_{ij}};\ q_{ij} = \beta_{ij} + \delta_{ij} - \overline{b}_{ij};$

$j=m$: $p_{P_{im}} = \alpha_{P_{im}};\ p_{E_{im-1}} = \alpha_{E_{im}};\ p_{SE_{1m}} = \alpha_{SE_{im}};\ q_{im} = \beta_{im}.$

2d. Вычисление $\overline{b}, \overline{a}_{nb}$

$j=\overline{1,m}$: $\overline{b}_{i+1j} = b_{i+1j} + a_{W_{i+1j}} q_{ij} / p_{P_{ij}};$

$\overline{a}_{E_{i+1j}} = a_{E_{i+1j}}$ при условии $i < n-1$

$j=1$: $\overline{a}_{P_{i+11}} = a_{P_{i+11}} - a_{W_{i+11}} p_{E_{i1}} / p_{P_{i1}};$

$\overline{a}_{N_{i+11}} = a_{N_{i+11}} + a_{W_{i+11}} p_{NE_{i1}} / p_{P_{i1}};$

$j=\overline{2,m-1}$ $\overline{a}_{P_{i+1j}} = a_{P_{i+1j}} - a_{W_{i+1j}} p_{E_{ij}} / p_{P_{ij}};$

$\overline{a}_{S_{i+1j}} = a_{S_{i+1j}} + a_{W_{i+1j}} p_{SE_{ij}} / p_{P_{ij}};$

$\overline{a}_{N_{i+1j}} = a_{N_{i+1j}} + a_{W_{i+1j}} p_{NE_{ij}} / p_{P_{ij}};$

$j=m$: $\overline{a}_{P_{i+1m}} = a_{P_{i+1m}} - a_{W_{i+1m}} p_{E_{im}} / p_{P_{im}};$

$\overline{a}_{S_{i+1m}} = a_{S_{i+1m}} + a_{W_{i+1m}} p_{SE_{im}} / p_{P_{im}}.$

II. Обратный ход — убывание индекса i от n до 1.

1. Вычисление промежуточного решения при $i=n$.

1a. Прямой ход по j.

$j=1$: $\varphi_{n1}=\dfrac{\overline{a}_{N_{n1}}}{\overline{a}_{P_{n1}}};\ \kappa_{n1}=\dfrac{\overline{b}_{n1}}{\overline{a}_{P_{n1}}};$

$j=\overline{2,m-1}$ $\varphi_{nj}=\dfrac{\overline{a}_{N_{nj}}}{\left(\overline{a}_{P_{nj}}-\varphi_{nj-1}\overline{a}_{S_{nj}}\right)};$

$$\kappa_{nj}=\frac{\left(\overline{b}_{nj}+\kappa_{nj-1}\overline{a}_{S_{nj}}\right)}{\left(\overline{a}_{P_{nj}}-\varphi_{nj-1}\overline{a}_{S_{nj}}\right)}.$$

1b. Обратный ход по j.

$j=m$: $\Phi^{05}_{nm}=\dfrac{\left(\overline{b}_{nm}+\kappa_{nm-1}\overline{a}_{S_{nm}}\right)}{\left(\overline{a}_{P_{nm}}-\varphi_{nm-1}\overline{a}_{S_{nm}}\right)};$

$j=\overline{m-1,1}$ $\Phi^{05}_{nj}=\varphi_{nj}\Phi^{05}_{nj+1}+\kappa_{nj}.$

2. Вычисление промежуточного решения при $i=\overline{n-1,1}$:

2a. Прямой ход по j.

$j=1$: $\varphi_{i1}=\dfrac{\overline{a}_{N_{i1}}}{\overline{a}_{P_{i1}}};\ \kappa_{i1}=\dfrac{\left(\overline{b}_{i1}+\overline{a}_{E_{i1}}\Phi^{05}_{i+11}\right)}{\overline{a}_{P_{i1}}};$

$j=\overline{2,m-1}$ $\varphi_{ij}=\dfrac{\overline{a}_{N_{ij}}}{\left(\overline{a}_{P_{ij}}-\varphi_{ij-1}\overline{a}_{S_{ij}}\right)};$

$$\kappa_{ij}=\frac{\left(\overline{b}_{ij}+\kappa_{ij-1}\overline{a}_{S_{ij}}+\overline{a}_{E_{ij}}\Phi^{05}_{i+1j}\right)}{\left(\overline{a}_{P_{ij}}-\varphi_{ij-1}\overline{a}_{S_{ij}}\right)}.$$

2b. Обратный ход по j.

$j=m$: $\Phi^{05}_{im}=\dfrac{\left(\overline{b}_{im}+\kappa_{im-1}\overline{a}_{S_{im}}+\overline{a}_{E_{im}}\Phi^{05}_{i+1m}\right)}{\left(\overline{a}_{P_{im}}-\varphi_{im-1}\overline{a}_{S_{im}}\right)};$

$j=\overline{m-1,1}$ $\Phi^{05}_{ij}=\varphi_{ij}\Phi^{05}_{ij+1}+\kappa_{ij}.$

Окончательное значение вектора решения на k-ой итерации $\left\{\Phi_{ij}^{k}\right\}$ получается путем повтора приведенной выше процедуры вычислений вдоль линий $y=y_j$, в которой в качестве решения с предыдущей итерации принимается вектор $\left\{\Phi_{ij}^{05}\right\}$.

Сходимость решения определяется по значению отношения норм векторов невязок $\dfrac{\left\|R^{k}\right\|}{\left\|R^{0}\right\|}$, где:

$$\left\|R^{k}\right\|=\left\{\frac{1}{(xL,yL)}\sum_{ij}h_x h_y\left[\begin{array}{l}a_{P_{ij}}\Phi_{ij}^{k}-\\-\left(a_{E_{ij}}\Phi_{i+1j}^{k}+a_{W_{ij}}\Phi_{i-1j}^{k}+a_{N_{ij}}\Phi_{ij+1}^{k}+a_{S_{ij}}\Phi_{ij-1}^{k}+b_{ij}\right)\end{array}\right]^{2}\right\}^{\frac{1}{2}} \quad (3)$$

Здесь h_x, h_y — сеточные шаги по координатам x и y соответственно. Решение считается найденным, если $\dfrac{\left\|R^{k}\right\|}{\left\|R^{0}\right\|}<\varepsilon$, где ε — заданная точность решения.

Для апробации метода выбрана следующая краевая задача: в области $\Omega=\left\{(x,y)\in\Omega, 0\le x\le 1, 0\le y\le 1\right\}$ выполняется уравнение (1), при этом коэффициенты ν_x и ν_y определяются как

$$\begin{aligned} v_x &= 1+C_2\left[(x-0{,}5)^2+(y-0{,}5)^2\right],\\ v_y &= 1+2\left[0{,}5-(x-0{,}5)^2-(y-0{,}5)^2\right]. \end{aligned} \quad (4)$$

Решение задачи определено в виде пробной функции

$$u(x,y)=256\left[xy(1-x)(1-y)\right]^2, \quad (5)$$

откуда легко получить выражения для Φ_Γ при задании граничных условий первого рода. Подстановка (4) и (5) в (1) позволяет получить аналитическое выражение для источника S.

Разностная аппроксимация задачи проведена на равномерной разностной сетке $\Omega_h=\left\{\begin{array}{l}(x_i,y_j):x_i=(i-1)h_x, y_j=(j-1)h_y;\\ 1\le i\le n, 1\le i\le m; h_x=\dfrac{1}{(n-1)}, h_y=\dfrac{1}{(m-1)}\end{array}\right\}$.

Начальное приближение решения выбрано постоянное по всей области $\Phi_{ij}^{0}=1$.

Чтобы показать насколько эффективен предложенный метод, проводились тестовые расчеты одной и той же задачи, с двойной

точностью представления чисел при предельно высоком требовании к точности решения для следующих методов: верхней релаксации [1], полинейного [3], модифицированного полинейного [4] с постоянным итерационным параметром, компенсационным параметром [5] и полинейного рекуррентного. Расчеты проводились для каждого метода при оптимальных значениях итерационного параметра, которые подбиралось для каждого метода и каждой сетки экспериментально.

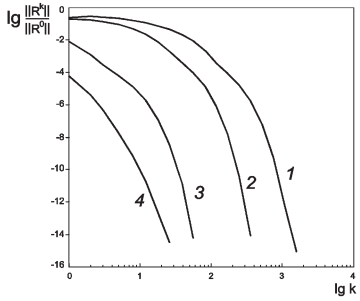


Рис. 3

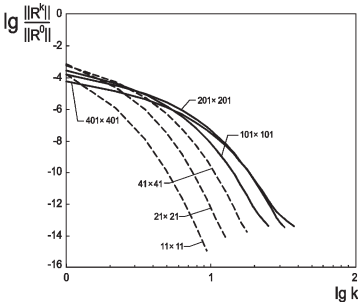


Рис. 4

На рис. 3 приведен график отношения норм невязок в зависимости от количества итераций для методов: 1 — верхней релаксации, 2 — полинейного, 3 — модифицированного полинейного с постоянным итерационным параметром (МР), 4 — полинейного рекуррентного (R2), на сетке 101×101. Из поведения кривых видна высокая эффективность полинейного рекуррентного метода, который достигает заданной точности решения за наименьшее количество итераций, снижая при этом за первую итерацию норму невязки на 4×5 порядков.

Зависимости отношений норм невязок от номера итерации при различных сеточных разбиениях расчетной области приведены на рис. 4. Здесь следует отметить одну любопытную деталь: полинейный рекуррентный метод при увеличении количества узлов сеточного разбиения области вместо привычного увеличения количества итераций, при требовании к точности ε практического порядка $10^{-4} \times 10^{-6}$, дает обратный результат — их уменьшение. В рассматриваемом примере на сетках 101×101 и выше в этом случае метод становится фактически прямым. Объясняется это, по-видимому, тем, что на более мелких сеточных шагах

экстраполяционная формула дает более точный прогноз поведения поправки решения, что, соответственно, усиливает неявный характер разрешающего оператора и, следовательно, уменьшает количество необходимых для сходимости итераций.

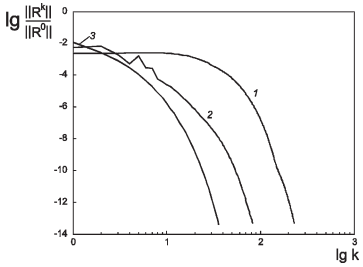


Рис. 5

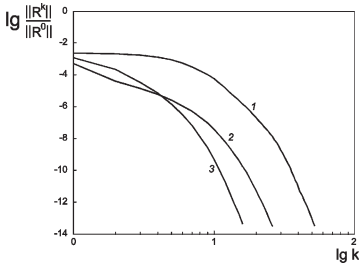


Рис. 6

1 — без компенсации, $\theta = \theta_0 = 1{,}0$; 2 — с компенсации, $\theta_0 = 1{,}0$;
3 — компенсации, $\theta = \theta_{opt} = 0{,}997$

В работе [5], в целях повышения эффективности используемого модифицированного полинейного метода (МР) решения СЛАУ [4], предлагается технология подбора значения итерационного параметра индивидуально для каждого уравнения на каждом итерационном шаге. На рис. 5 (метод МР) и 6 (метод R1 — полинейный рекуррентный с линейной экстраполяцией «внешаблонного» узла) представлена динамика отношения норм невязок в зависимости от номера итерации на сетке 101×101. На этих графиках хорошо видно, что во всем диапазоне требования по точности технология компенсации итерационного параметра не исчерпывает всего потенциала ускорения расчетов, а в случае оптимизации θ иным способом — практически не влияет на скорость сходимости. К тому же здесь выявляется еще одна деталь применения технологии компенсации: она может приводить к понижению устойчивости вычислительного процесса на начальных итерациях, либо вообще не позволяет достигнуть наперед заданной точности. Подобные результаты наблюдаются при использовании метода R1 (рис. 6) за исключением того, что в данных расчетах не выявлено негативное влияние применения технологии компенсации на устойчивость вычислений. Однако применение технологии компенсации в случае метода R2 (полинейный рекуррентный с квадратичной экстраполяцией «внешаблонного» узла), не сказывается на повышении скорости сходимости.

Литература

1. **Самарский А. А., Гулин А. В.** Численные методы. — М. : Наука, 1989.

2. **Фомин А. А., Фомина Л. Н.** Полинейный рекуррентный метод решения разностных эллиптических уравнений. — Кемерово : КемГУ. — 2007. — 78 с. — Деп. в ВИНИТИ 06.04.2007, № 385-В2007.

3. **Патанкар С.** Численные методы решения задач тепломассобмена и динамики жидкости. — М. : Энергоатомиздат, — 1984.

4. **Зверев В. Г.** Модифицированный полинейный метод решения разностных эллиптических уравнений // ЖВМ и МФ. — 1998. — Т. 38. — № 9. — С. 1553—1562.

5. **Zverev V. G.** About the iteration method for solving difference equations // Lecture notes in computer science.– Berlin Heidelberg : Springer-Verlag. — 2005. — Vol. 3401.– P. 621—628.

Метод регуляризации решения обратных задач теплопроводности

О. А. Есаулова, Н. Н. Меркулова
Томский государственный университет, Томск

Среди математических задач выделяется класс задач, решения которых неустойчивы к малым изменениям начальных данных. Они характеризуются тем, что сколь угодно малые изменения исходных данных могут приводить к произвольно большим изменениям решений. Задачи подобного типа принадлежат к классу некорректно поставленных задач [1].

Целый класс некорректных задач составляют так называемые обратные задачи, возникающие в тех случаях, когда необходимые данные для постановки прямой корректной задачи отсутствуют, но зато есть некоторая дополнительная информация о решении, позволяющая формулировать обратную задачу. К числу таких задач принадлежат обратные задачи теплопроводности (ОЗТ) [2].

При рассмотрении обратных задач теплообмена естественно выделить ретроспективную обратную задачу теплопроводности, граничные обратные задачи и коэффициентные обратные задачи.

Целью данной работы является применение метода регуляризации к решению некорректных ОЗТ, а также разработка алгоритма решения таких задач .

Рассмотрим пример, представляющий собой случай некорректной граничной обратной задачи теплопроводности. Пусть процесс теплопроводности в пластине описывается уравнением:

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2},\ x \in (0,1),\ t \in (0,1) \qquad (1)$$

Известно начальное распределение температуры

$$T(x,0) = \varphi(x),\ x \in [0,1]$$

В точке $x = d\ (0 < d \le 1)$ задано изменение температуры во времени

$$T(d,t) = \psi(t),\ t \in [0,1]$$

На правой границе $x = 1$ известно условие второго рода

$$-\lambda\frac{\partial T(1,t)}{\partial x}=\Gamma_1(t),\ t\in[0,1]$$

где $\lambda=\text{const}>0$.

Требуется определить плотность теплового потока на левой границе $\Gamma_2(t)=-\lambda\dfrac{\partial T(0,t)}{\partial x}$.

Пусть $T(x,t)=e^{-\sqrt{n}}e^{-n^2t}Sin(nx)$. Тогда начальные и граничные условия примут вид:

$$T(x,0)=e^{-\sqrt{n}}Sin(nx),\ x\in[0,1];$$

$$T(d,t)=e^{-\sqrt{n}}e^{-n^2t}Sin(nd),\ t\in[0,1],\ 0<d\le 1;$$

$$-\lambda\frac{\partial T(1,t)}{\partial x}=-\lambda ne^{-\sqrt{n}}e^{-n^2t}Cos(n),\ t\in[0,1]$$

Следовательно, $\varphi(x)=e^{-\sqrt{n}}Sin(nx), x\in[0,1]$,

$\psi(t)=e^{-\sqrt{n}}e^{-n^2t}Sin(nd),\ t\in[0,1],\ \Gamma_1(t)=-\lambda ne^{-\sqrt{n}}e^{-n^2t}Cos(n),\ t\in[0,1]$, $0<d\le 1$ и требуется найти численным методом значение потока на левой границе $\Gamma_2(t)$. Очевидно, что точное решение определяется как $\Gamma_2(t)=\lambda ne^{-\sqrt{n}}e^{-n^2t}Cos(n)$.

Имеем следующую расчетную область (рис. 1).

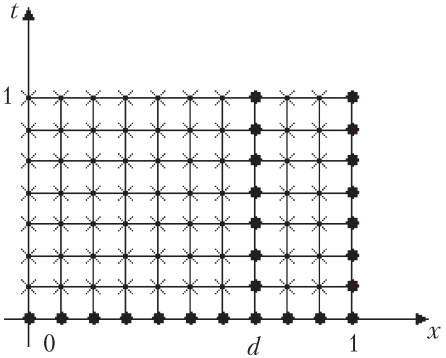


Рис. 1. Расчетная сеточная область

Предположим, что есть алгоритм решения соответствующей прямой задачи и найдем его «обращение» с целью установления искомой связи между входными данными и граничными условиями [3].

Решение прямой задачи для уравнения (1), согласно принципам суперпозиции и Дюамеля [4], можно представить в виде:

$$T(x,t)=\int_0^t \Gamma_2(\xi)\frac{\partial\theta(x,t-\xi)}{\partial t}d\xi+\int_0^t \Gamma_1(\xi)\frac{\partial\theta(1-x,t-\xi)}{\partial t}d\xi, \qquad (2)$$

где $\theta(x,t)$ — температурная реакция тела на единичный тепловой поток на одной из границ ($x=0$) при условии теплоизолированности другой ($x=1$).

Величина $\theta(x,t)$ может быть рассчитана по формуле [3]:

$$\theta(x,t)=\frac{1}{\lambda}\left[t+\frac{3(1-x)^2-1}{6}+\frac{2}{\pi^2}\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k^2}\exp\left(-k^2\pi^2t\right)\cos k\pi(1-x)\right].$$

Таким образом, обратная задача в данном случае сводится к следующей интегральной задаче:

$$Au\equiv\int_0^t u(\xi)K(t-\xi)d\xi=f(t),\ t\in[0,1], \qquad (3)$$

где $u(t)=\Gamma_2(t)$, $K(t-\xi)=\dfrac{\partial\theta(d,t-\xi)}{\partial t}$,

$$f(t)=\psi(t)-\int_0^t \Gamma_1(\xi)\frac{\partial\theta(1-d,t-\xi)}{\partial t}d\xi.$$

Найдя производную от функции $\theta(x,t)$ по времени, получим представление ядра и правой части из уравнения (3):

$$K(t-\xi)=\frac{\partial\theta(d,t-\xi)}{\partial t}=\frac{1}{\lambda}\left[1-2\sum_{k=1}^{\infty}(-1)^{k+1}e^{-k^2\pi^2(t-\xi)}\cos k\pi(1-d)\right], \qquad (4)$$

$$f(t)=\psi(t)-\frac{1}{\lambda}\int_0^t \Gamma_1(\xi)\left[1-2\sum_{k=1}^{\infty}(-1)^{k+1}e^{-k^2\pi^2(t-\xi)}Cosk\pi d\right]d\xi.$$

Для одномерного уравнения теплопроводности с постоянными коэффициентами граничные обратные задачи имеют общее интегральное представление (2). Разделим полный временной отрезок на конечное число интервалов с переменными шагами с целью замены уравнения (3) приближенным дискретным аналогом: $\Delta t_k=t_k-t_{k-1}$, $k=\overline{1,m}$ [3]. В этом случае уравнение (3) перепишется в виде [4]:

$$\sum_{i=1}^{k}\int_{t_{i-1}}^{t_i} u(\xi)K(t_k-\xi)d\xi = f(t_k),\ k=\overline{1,m}. \qquad (5)$$

Полагая искомую функцию $u(t)$ равной постоянному значению $\hat{u}_k$ на каждом интервале, введем ее ступенчатую аппроксимацию:

$$\sum_{i=1}^{k}\hat{u}_k\Delta\theta_{ik} = f_k,\ k=\overline{1,m}, \qquad (6)$$

где $f_k = f(t_k)$, $\Delta\theta_{ik} = \int_{t_{i-1}}^{t_i} K(t_k-\xi)d\xi$.

После аппроксимации получим систему линейных алгебраических уравнений $A_\Delta u = f$, $u = \{\hat{u}_k\}_1^m$, $f = \{f_k\}_1^m$ с нижней треугольной матрицей.

Определитель матрицы A_Δ равен произведению ее диагональных элементов: $\det A_\Delta = \Delta\theta_{11}\Delta\theta_{22}\ldots\Delta\theta_{mm}$. Из физического смысла величины $\theta(d,t)$ следует, что все $\Delta\theta_{ik} > 0$. Отсюда $\det A_\Delta \neq 0$. Следовательно, матрица A_Δ теоретически является невырожденной и существует обратная матрица A_Δ^{-1}. Легко получить следующую рекуррентную формулу, позволяющую шаг за шагом найти все компоненты m-мерного вектора u [3]:

$$\hat{u}_k = \frac{1}{\Delta\theta_0}\left(f_k - \sum_{i=1}^{k-1}\hat{u}_i\Delta\theta_{k-i}\right),\ k=\overline{1,m} \qquad (7)$$

Следовательно, вместо $\frac{m(m+1)}{2}$ коэффициентов матрицы необходимо рассчитывать и хранить в памяти компьютера всего m коэффициентов.

Решение уравнения (3) было найдено с использованием формул (7) при разных значениях m и, как показало сравнение, вело себя неустойчиво. Для решения данной некорректной задачи использовалась регуляризация Тихонова [1].

Процесс регуляризации решения состоит из трех основных этапов:

- найти регуляризирующий оператор $R(f,\alpha)$;
- задать параметр регуляризации α (по дополнительной информации о задаче);
- найти регуляризованное решение $u_\alpha = R(f_\alpha,\alpha)$.

Таким образом, вместо операторного уравнения $Au = f$ будем решать регуляризованную систему линейных алгебраических уравнений [1]:

$$\alpha u_k^{\alpha} + \sum_{j=1}^{m} \overline{a}_{kj} u_j^{\alpha} = b_k, k = 1,2,...,m. \tag{8}$$

Коэффициенты системы и правая часть вычисляются по формулам:

$$\overline{a}_{kj} = \sum_{i=1}^{m} a_{ik} a_{ij}, \; b_k = \sum_{i=1}^{m} a_{ik} f_i.$$

В теории методов приближенного решения некорректных задач вопросу выбора параметра регуляризации уделяется значительное внимание. Среди ряда алгоритмов, позволяющих подобрать параметр регуляризации так, чтобы минимизировать норму невязки, наиболее известным является априорный выбор параметра регуляризации α.

Априорный выбор параметра регуляризации α

Невязка зависит некоторым образом от выбора α [2]. Обозначим $\varphi(\alpha) = \|Au_{\alpha} - f_{\delta}\|$. Тогда нахождение параметра регуляризации состоит в соответствии с принципом невязки в решении уравнения

$$\varphi(\alpha) = \delta. \tag{9}$$

Для приближенного решения уравнения (9) используются различные вычислительные процедуры. Например, задается последовательность

$$\alpha_k = \alpha_0 q^k \tag{10}$$

и вычисления проводятся с $k = 0$ до некоторого $k = K$, при котором равенство (9) выполняется с приемлемой точностью.

Таким образом, алгоритм метода регуляризации в применении к тестовому примеру состоит в следующем:

1) вычисляем параметр регуляризации, пользуясь формулой (10), в которой положим $\alpha_0 = 1$ и $q = 0{,}99$;

2) составляем матрицу регуляризованной системы линейных алгебраических уравнений (8) и столбец свободных членов с найденным α_k.

3) решаем полученную регуляризованную систему линейных алгебраических уравнений $A_R u^{\alpha} = b$ методом Гаусса с выбором главного элемента и получаем u^{α_k};

4) находим невязку $\varphi(\alpha) = \|Au_{\alpha} - f_{\delta}\|$, где A — первоначальная матрица. Если $\varphi(\alpha) = \delta$, то процесс окончен и u_{α_k} – найденное решение с заданной точностью. Если же $\varphi(\alpha) \neq \delta$, то подбираем следующее α_k.

Апостериорный выбор параметра регуляризации α

На практике, как правило, значения функции $u(t)$ снимаются с показаний приборов и, следовательно, известны с погрешностью. Будем считать, что вместо значений функции $f(t_i)$ известны значения $f_{\delta}(t_i)$, определяемые формулой:

$$f_{\delta}(t_i) = f(t_i)\left(1 + \gamma_i \sqrt{\frac{3(b-a)}{b^3 - a^3}}\sigma\right)$$

Здесь γ_i — случайные числа из промежутка [0;1] с равномерным законом распределения, σ — коэффициент запаса и выбирается как $\sigma = 0{,}002$. Таким образом, имеем систему линейных уравнений, полученную по описанному выше алгоритму метода регуляризации:

$$A_R u^{\alpha} = b. \tag{11}$$

Параметр регуляризации α должен подбираться так, чтобы принимаемое за решение задачи (11) регуляризованное решение $y\alpha$ было, по возможности, наилучшим в смысле удачного согласования левой и правой частей уравнения $\tilde{A}y = f_{\delta}$, что определяется малостью невязки. Таким образом, основной проблемой метода регуляризации является вопрос подбора параметра регуляризации α [5].

Так, согласно [5], зависимость величин α и δ, представляемая в виде $\alpha = \delta^p$, является регуляризирующим алгоритмом при условии, что $\alpha(\delta) \to 0$ при $\delta \to 0$. Поскольку на практике неизвестно, какой случай выбора показатель степени p имеет место, то можно в качестве значения $\alpha(\delta)$ выбирать любую зависимость, сохраняющую условие согласования нормы невязки и погрешности правой части:

$$\|Au_{\alpha} - f\| \leq \|f - f_{\delta}\|$$

Выбор параметра регуляризации α методом секущих

Для приближенного решения уравнения (9) можно использовать и более быстро сходящиеся итерационные методы. Установлено, что функция $\tilde{\varphi}(\beta) = \varphi\left(1/\beta\right)$ является убывающей и выпуклой функцией. Поэтому для решения уравнения $\tilde{\varphi}(\beta) = \delta$ можно применить итерационный метод Ньютона, когда

$$\beta_{k+1} = \beta_k - \frac{\tilde{\varphi}(\beta_k)}{\tilde{\varphi}'(\beta_k)}$$

Этот метод будет сходиться при любом начальном приближении $\beta_0 > 0$. Для того, чтобы не вычислять производную функции $\tilde{\varphi}(\beta)$, можно использовать итерационный метод секущих, в котором:

$$\beta_{k+1} = \beta_k - \frac{\beta_k - \beta_{k-1}}{\tilde{\varphi}(\beta_k) - \tilde{\varphi}(\beta_{k-1})}\tilde{\varphi}(\beta_k)$$

Применение подобных итерационных процедур позволяет сократить общие вычислительные затраты на определение параметра регуляризации α.

Оценивая реализованные схемы, можно отметить в отдельности особенности каждой. Вычисления регуляризованного решения с использованием априорного выбора параметра регуляризации дало результаты достаточно близкие по своему характеру к точному решению, но в то же время реализация этого метода требует большого объема используемой оперативной памяти, что не позволяет проводить численный эксперимент для числа разбиений $m > 20$, и уже начиная с $m > 10$ расчеты требуют много времени и метод становится неэффективным. В этом смысле использование алгоритма апостериорного выбора параметра регуляризации позволяет проводить расчеты при большом числе разбиений $m (m > 20)$ и дает более приемлемые результаты. Однако можно утверждать, что ошибка вычислений уменьшается еще больше в случае применения итерационного метода секущих за счет более оптимального выбора параметра регуляризации. Более того, по сравнению с алгоритмом априорного выбора параметра регуляризации α, который сходится медленно (более 300 итераций), метод секущих имеет высокую скорость сходимости (2 итерации), что обусловлено уменьшением числа выпол-

няемых арифметических операций. На рис. 2 показан характер поведения регуляризованного решения, полученного методом секущих.

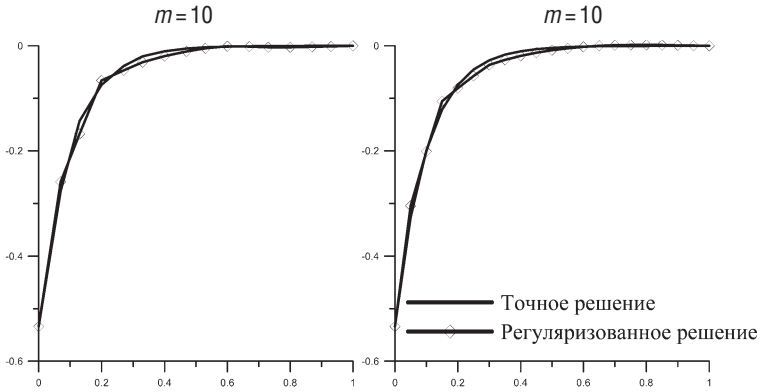


Рис. 2. Характер поведения точного и регуляризованного решений, полученных методом секущих

Литература

1. **Тихонов А. Н., Арсенин В. Я.** Методы решения некорректных задач. — М. : Наука. Гл. ред. физ.-мат. лит., 1986. — 288 с.

2. **Самарский А. А., Вабищевич П. Н.** Вычислительная теплопередача. — М. : Едиториал УРСС, 2003. — 784 с.

3. **Алифанов О. М., Вабищевич П. Н., Михайлов В. В. и др.** Основы идентификации и проектирования тепловых процессов и систем. — М. : Логос, 2001. — 400 с.

4. **Тихонов А. Н., Самарский А. А.** Уравнения математической физики. — М. : Наука. Гл. ред. физ.-мат. лит., 1972. — 735 с.

5. **Верлань А. Ф., Сизиков В. С.** Интегральные уравнения: методы, алгоритмы, программы. — Киев : Наукова Думка, 1986.

К вопросу о выборе турбулентной схемы замыкания для микромасштабных моделей городского атмосферного пограничного слоя[1]

[1]Р. Б. Нутерман, [1]А. В. Старченко, [2]А. А. Бакланов
[1]Томский государственный университет, Томск, Россия
[2]Датский метеорологический институт, Копенгаген, Дания

Введение

В настоящее время значительное внимание уделяется разработке математических моделей турбулентности для анализа и прогноза поведения метеополей вблизи больших мегаполисов. Значительное влияние на формирование приземного ветра оказывает структура подстилающей поверхности. Особенно это проявляется для участков многоэтажной городской застройки, где в уличных каньонах создаются замкнутые циркуляции приземного воздуха, способствующие накоплению повышенных концентраций вредных веществ. Важную роль в процессах рассеяния примеси играет турбулентная диффузия, поэтому для численных расчетов необходимо выбирать модели турбулентности, корректно описывающие турбулентную структуру рециркуляционного потока.

Целью данной работы является апробация трех моделей турбулентности на основе экспериментальных данных для течения за обращенным назад уступом.

Математическая постановка задачи и метод решения

В данном исследовании рассматривается стационарное турбулентное изотермическое движение воздуха. Математическая постановка задачи включает уравнения Рейнольдса [1]:

$$\frac{\partial \overline{u}_j}{\partial x_j} = 0,$$
$$\frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \overline{p}}{\partial x_i} + \nu \frac{\partial}{\partial x_j}\left(\frac{\partial \overline{u}_i}{\partial x_j}\right) - \frac{\partial}{\partial x_j}\left(\overline{u_i' u_j'}\right), \; i, j = 1,2,3 \qquad (1)$$

[1] Работа выполнена при финансовой поддержке граната INTAS № 06-1000016-5928 и гранта РФФИ № 07-05-01126.

Здесь $\overline{u}_j$ — осредненные проекции вектора скорости на оси координат Ox_j; $\overline{p}$ — давление, ρ — плотность, ν — кинематическая вязкость воздуха, а $\overline{u_i' u_j'}$ — компоненты тензора напряжений Рейнольдса.

Замыкание системы уравнений Рейнольдса строится на основе трех моделей: k—ε модели [1], модели нелинейной вихревой вязкости [2] и дифференциальной модели переноса напряжений Рейнольдса [3].

Линейная k—ε модель турбулентности

В этом случае замыкающие соотношения Буссинеска для системы осредненных уравнений (1) записываются в виде:

$$-\overline{u_i' u_j'} = \nu_T \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) - \frac{2}{3} \delta_{ij} k, \tag{2}$$

где δ_{ij} — символ Кронекера; ν_T — турбулентная вязкость, а k — кинетическая энергия турбулентности. В линейной k—ε модели Лаундера и Сполдинга [1] турбулентная вязкость задается как

$$\nu_T = C_\mu \frac{k^2}{\varepsilon} \tag{3}$$

где $C_\mu = 0{,}09$ — это безразмерная константа. Кроме того, модель включает два транспортных уравнения для кинетической энергии k и для ее диссипации ε:

$$\overline{u}_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_T}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] + G - \varepsilon, \tag{4}$$

$$\overline{u}_j \frac{\partial \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_T}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] + \left(c_{\varepsilon 1} G - c_{\varepsilon 2} \varepsilon \right) \frac{\varepsilon}{k}. \tag{5}$$

Здесь генерация энергии турбулентности

$G = \nu_T \left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} \right) \frac{\partial \overline{u}_i}{\partial x_j}$, а константы $\sigma_k = 1{,}0$, $\sigma_\varepsilon = 1{,}3$, $c_{\varepsilon 1} = 1{,}44$, $c_{\varepsilon 2} = 1{,}92$.

Нелинейная k-ε модель турбулентности (Non-linear eddy viscosity model, NLEVM)

В соотношениях (2) представлена линейная зависимость тензора анизотропии турбулентности $a_{ij} = \dfrac{\overline{u_i' u_j'}}{k} - \dfrac{2}{3}\delta_{ij}$ от тензора скорости деформации $S_{ij} \equiv \left(\partial \overline{u}_i / \partial x_j + \partial \overline{u}_j / \partial x_i\right)$. Однако более общее соотношение для тензора a_{ij} имеет вид [2]:

$$a_{ij} = A_{ij}\left(S_{ij}, \Omega_{ij}\right)$$

где A_{ij} — тензор второго порядка, который представляет собой взаимосвязь между тензорами скорости деформации S_{ij} и завихренности $\Omega_{ij} \equiv \left(\partial \overline{u}_i / \partial x_j - \partial \overline{u}_j / \partial x_i\right)$.

Исходя из результатов исследований [2], выражение для тензора анизотропии турбулентности может быть представлено в виде:

$$\begin{aligned} a_{ij} = \frac{\overline{u_i' u_j'}}{k} - \frac{2}{3}\delta_{ij} = -\frac{\nu_T}{k} S_{ij} + c_1 \frac{\nu_T}{\varepsilon}\left(S_{ik}S_{kj} - \frac{1}{3}S_{kl}S_{kl}\delta_{ij}\right) + \\ + c_2 \frac{\nu_T}{\varepsilon}\left(\Omega_{ik}S_{kj} + \Omega_{jk}S_{ki}\right) + c_3 \frac{\nu_T}{\varepsilon}\left(\Omega_{ik}\Omega_{jk} - \frac{1}{3}\Omega_{kl}\Omega_{kl}\delta_{ij}\right) + \\ + c_4 \frac{\nu_T k}{\varepsilon^2}\left(S_{ki}\Omega_{lj} + S_{kj}\Omega_{li}\right)S_{kl} + \\ + c_5 \frac{\nu_T k}{\varepsilon^2}\left(\Omega_{il}\Omega_{lm}S_{mj} + S_{il}\Omega_{lm}\Omega_{mj} - \frac{2}{3}S_{lm}\Omega_{mn}\Omega_{nl}\delta_{ij}\right) + \\ + c_6 \frac{\nu_T k}{\varepsilon^2} S_{ij}S_{kl}S_{kl} + c_7 \frac{\nu_T k}{\varepsilon^2} S_{ij}\Omega_{kl}\Omega_{kl} \end{aligned} \qquad (6)$$

где $c_1 = 0{,}1, c_2 = 0{,}1, c_3 = 0{,}26, c_4 = -10C_\mu^2, c_5 = 0, c_6 = 5C_\mu^2, c_7 = 5C_\mu^2$. Квадратичные комбинации S и Ω необходимы для описания анизотропии нормальных напряжений в простом сдвиговом течении, а кубические члены вводятся для учета корректного воздействия кривизны линий тока и закрутки течения. В работе [2] также определяется функциональная зависимость для C_μ:

$$C_\mu = \frac{0{,}3}{1 + 0{,}35\left\{\max\left(\tilde{S}, \tilde{\Omega}\right)\right\}^{1{,}5}}\left[1 - \exp\left\{-0{,}36\exp\left\{0{,}75\max\left(\tilde{S}, \tilde{\Omega}\right)\right\}\right\}\right] \qquad (7)$$

где безразмерные инварианты деформации и завихренности определяются следующим образом:

$$\tilde{S} \equiv \frac{k}{\varepsilon}\sqrt{\frac{S_{ij}S_{ij}}{2}},\ \tilde{\Omega} \equiv \frac{k}{\varepsilon}\sqrt{\frac{\Omega_{ij}\Omega_{ij}}{2}}$$

Максимумы $\tilde{S}$ и $\tilde{\Omega}$ используются для увеличения чувствительности модели к влиянию кривизны линий тока.

Дифференциальная модель переноса напряжений Рейнольдса (Reynolds Stress Model, RSM)

Можно получить замыкание более высокого порядка для системы (1) если использовать транспортные уравнения для каждой независимой компоненты тензора напряжений Рейнольдса. Эти дифференциальные уравнения для напряжений Рейнольдса включают все физические процессы, которые влияют на корреляции второго порядка. Они способны моделировать нестационарные и нелокальные эффекты, что невозможно реализовать с помощью линейных и нелинейных двухпараметрических моделей, которые опираются на локальные градиенты скорости для расчета напряжений Рейнольдса.

Уравнения для напряжений Рейнольдса без учета перераспределение нормальных напряжений у стенки имеет вид [3, 4]:

$$\begin{aligned}\overline{u}_k \frac{\partial \overline{u_i'u_j'}}{\partial x_k} = P_{ij} + \frac{\partial}{\partial x_k}\left(\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial \overline{u_i'u_j'}}{\partial x_k}\right) - \\ -C_1\frac{\varepsilon}{k}\left[\overline{u_i'u_j'} - \frac{2}{3}\delta_{ij}k\right] - C_2\left[P_{ij} - \frac{2}{3}\delta_{ij}P\right] - \varepsilon\delta_{ij},\end{aligned} \tag{8}$$

где $\delta_k = 0{,}82$, $C_1 = 1{,}8$, $C_2 = 0{,}6$, а $P_{ij} = -\overline{u_i'u_k'}\frac{\partial \overline{u}_j}{\partial x_k} - \overline{u_j'u_k'}\frac{\partial \overline{u}_i}{\partial x_k}$, $P = \frac{1}{2}P_{kk}$, $i,j = 1,2,3$.

Граничные условия

На входной границе, где поток поступает в область исследования, граничные условия выбираются в соответствии с данными наблюдений или экспериментов, а на выходе и верхней границах используются нулевые градиентные соотношения.

Взаимодействие турбулентного потока с препятствиями и твердыми поверхностями учитывается в соответствии с методом пристенных функций Лаундера-Сполдинга [1], согласно которому касательная к поверхности компонента скорости представляется вблизи поверхности как

$$U_\tau = \frac{\tau_w}{\rho C_\mu^{1/4} \kappa k^{1/2}} \ln\left[\frac{E C_\mu^{1/4} k^{1/2} n}{\nu}\right]$$

где $\kappa = 0{,}42$, $E = 9{,}0$, τ_w — трение на поверхности, n — расстояние от обтекаемой поверхности.

Кинетическая энергия турбулентности k вблизи поверхности (в пристеночной ячейке разностной сетки с индексом P) определяется из транспортного уравнения для k (4), при этом для генерации G и диссипации ε энергии турбулентности вблизи стенки используются следующие зависимости:

$$\overline{G} = \frac{\tau_w^2}{\rho \kappa C_\mu^{1/4} k_P^{1/2} y_n} \ln\left(\frac{y_n}{y_v}\right), \quad \overline{\varepsilon} = \frac{1}{y_n}\left[\frac{2k_P^{3/2}}{k_P^{1/2} y_v / \nu} + \frac{k_P^{3/2}}{c_l} \ln\left(\frac{y_n}{y_v}\right)\right], \quad y_v = \frac{20\nu}{k_P^{1/2}}$$

Здесь координата y_ν соответствует области вязкого подслоя пограничного слоя, а y_n области развитой турбулентности, при этом для расчетов с помощью RSM модели компоненты тензора a_{ij} выбираются постоянными в пристеночной области.

Методика решения

Задача решается численно на основе метода конечного объема [5]. Дискретизация дифференциальных уравнений осуществляется на неравномерной структурированной сетке. Для расчета интегралов используются кусочно-линейные профили, которые описывают изменение функции от одного узла разностной сетки к другому. Конвективные члены транспортных уравнений аппроксимируются с использованием монотонизированной линейной противопотоковой схемы Ван Лира [6].

Такая дискретизация дифференциальных уравнений приводит к системе сеточных уравнений, решение которых построено

на основе метода неполной факторизации Булеева [7]. Для согласования полей скорости и давления используется процедура SIMPLE Патанкара-Сполдинга [5]. Расчет параметров в областях со сложной геометрией осуществляется на основе метода фиктивных областей.

Результаты расчетов

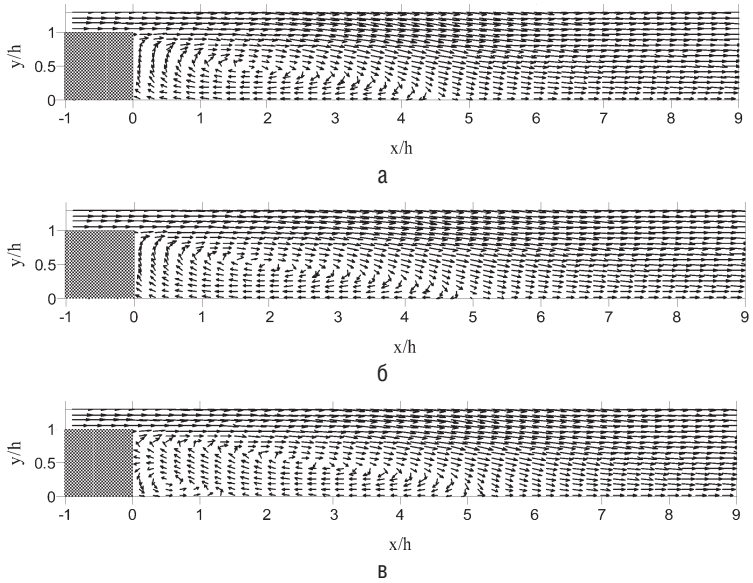


Рис. 1. Векторное поле турбулентного движения за уступом:
а) линейная k—ε модель, б) NLEVM модель, в) RSM модель

Проведено сравнение экспериментальных данных [8] с расчетами при турбулентном движении за уступом. Высота уступа $h = 0{,}076$ м, а скорость жидкости во входном сечении равна 10 м/с. Режим течения соответствует числу Рейнольдса $Re = 50\ 000$. Вычисления проводились в области размером 7 м по оси Ox и 0,5 м по оси Oy на сетке размером 231×177.

При этом расчеты осуществлялись для трех различных моделей турбулентности. Ниже представлены векторное поле скорости турбулентного движения за уступом (рис. 1 а, б, в), график изменения коэффициента давления (рис. 2), безразмерная

скорость и кинетическая энергия турбулентности вниз по потоку за уступом (рис. 3 а, б).

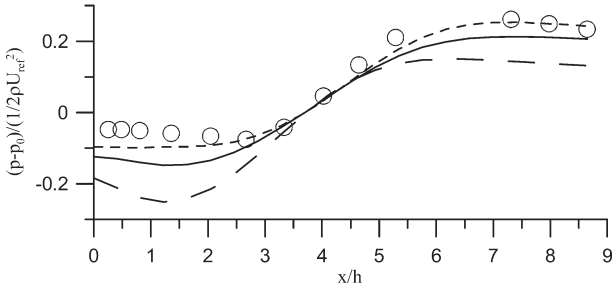


Рис. 2. Распределение коэффициента давления за обращенным назад уступом: — — — линейная k—ε модель, — NLEVM, --- RSM, o — эксперимент [8]

Сравнение трех моделей показывает превосходство RSM модели над всеми остальными и, в особенности, над линейной k—ε. Видно, что коэффициент давления внутри рециркуляционной области предсказывается k—ε моделями хуже, чем RSM (рис. 2). По-видимому, вследствие этого вблизи стенки модуль рассчитанной по k—ε модели скорости при $x/h \leq 2$ имеет большее по сравнению с экспериментом значение (рис. 3а).

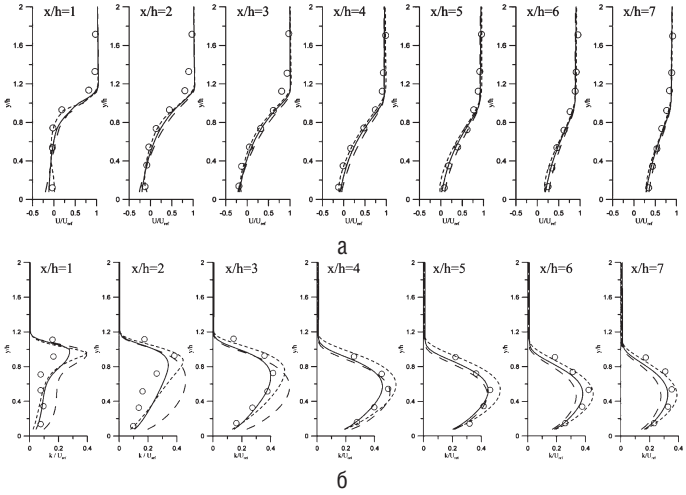


Рис. 3. Профили скорости (а) и кинетической энергии турбулентности (б) за уступом; — – — линейная k—ε модель, — NLEVM модель, --- RSM модель, o — эксперимент [8]

Кроме того, при расчетах по стандартной [1] и нелинейной [2] k—ε модели не наблюдается образования вторичного вихря в нижнем углу за уступом (рис. 1). Тем не менее, нелинейная модель вихревой вязкости значительно лучше стандартной k—ε модели предсказывает осредненные величины потока (рис. 2, 3).

Относительно моделирования турбулентных параметров можно сказать, что использование стандартной k—ε модели [1] завышает уровень кинетической энергии в застойной области течения (рис. 3б) $x/h \leq 3$, где наблюдается вторичная рециркуляция, и недооценивает его вниз по потоку за точкой присоединения $x/h \geq 6$. Этот недостаток в значительно меньшей степени проявляется у нелинейной модели вихревой вязкости.

Если сравнивать временные затраты численного расчета представленного выше течения, то RSM модель по этой характеристике уступает остальным двум моделям на 30—40 %. Кроме того, использование RSM модели может сопровождаться локальной численной неустойчивостью, что, в свою очередь, является недостатком модели. Таким образом, нелинейная k—ε модель [2] может рассматриваться в качестве модели, обеспечивающей более адекватное воспроизведение турбулентного рециркуляционного течения, чем стандартная k—ε модель [1]. В тоже время, нелинейная k—ε модель [2] может использоваться при расчете сложных трехмерных течений в элементах городской застройки, поскольку она требует меньших затрат ЭВМ по сравнению с RSM подходом при незначительном ухудшении качества расчетов.

Заключение

Представлены результаты сравнения трех моделей турбулентности на основе экспериментальных данных для течения за обращенным назад уступом. Несмотря на кажущуюся простоту данного расчетного случая, в нем присутствуют сложные эффекты, такие как отрыв потока и его присоединение, рециркуляционные и застойные области, которые порой не способны учесть обычные двухпараметрические модели.

На основе расчетов можно сделать вывод о том, что использование транспортных уравнений для напряжений Рейнольдса является более трудоемким с точки зрения расчетного времени, но позволяет хорошо предсказывать тонкие эффекты, которые

могут сказываться на процессах переноса. Нелинейная k—ε модель может рассматриваться в качестве вынужденной альтернативы RSM модели для расчета трехмерных течений в уличных каньонах.

Литература

1. **Launder B. E., Spalding D. B.** The numerical computation of turbulent flows // Computational Methods in Applied Mechanics and Engineering. — 1974. — Vol. 3. — № 2. — P. 269—289.

2. **Craft T. J., Launder B. E., Suga K.** Development and application of a cubic eddy-viscosity model of turbulence // Int. J. Heat and Fluid Flow. — 1996. — Vol. 17. — P. 108—115.

3. **Gibson M. M., Launder B. E.** Ground Effects on Pressure Fluctuations in the Atmospheric Boundary Layer // J. Fluid Mech. — 1978. — Vol. 86. — P. 491—511.

4. **Launder B. E.** Second-Moment Closure and Its Use in Modeling Turbulent Industrial Flows // International Journal for Numerical Methods in Fluids. — 1989. — Vol. 9. — P. 963—985.

5. **Патанкар С.** Численные методы решения задач теплообмена и динамики жидкости. — М. : Энергоатомиздат, 1984. — 152 с.

6. **Van Leer B.** Towards the Ultimate Concervative Difference Scheme. II. Monotonicity and Conservation Combined in a Second Order Scheme // Journal of Computational Physics. — 1974. — Vol. 14. — P. 361—370.

7. **Ильин В. П.** Методы неполной факторизации для решения алгебраических систем. — М. : Физматлит, 1995. — 288 с.

8. **Мосс В. Д., Бэкер С., Брэдбери Л. Дж. С.** Измерения средней скорости и рейнольдсовых напряжений в некоторых областях рециркулирующих течений // Турбулентные сдвиговые течения 1 // Пер. А. В. Колесникова и др. [Сб. ст.] Пер. с англ. под ред. А. С. Гиневского. — М. : Машиностроение, 1982. — 432 с.

Рекуррентные сплайн-фильтры

Б. М. Шумилов, Э. А. Эшаров[1]
Томский государственный университет, Томск

Исследуется задача построения рекуррентного сплайн-аппроксимационного фильтра, точного на многочленах. Представлена и обоснована вычислительная схема рекуррентной сплайн-аппроксимации степени 3 произвольной глубины p по зашумленным данным на равномерной сетке.

Введение

Рекуррентная аппроксимация сплайнами встречается в задачах мониторинга, например в палатах интенсивной терапии, при обработке данных траекторных наблюдений, в других задачах слежения и управления. При этом весьма желательно получать результаты обработки с некоторым опережением, т. е. в режиме прогнозирование

Известно [1, с. 35], что интерполяционные сплайны дефекта 1 определяются по заданным значениям интерполируемой функции на всем отрезке наблюдения. Это усложняет обработку данных в темпе их поступления, например, требуется введение дополнительных узлов [2]. С другой стороны, использование локального сглаживания [1, с. 244] приводит к увеличению времени запаздывания в зависимости от степени сплайна и порядка аппроксимации. Поэтому ниже на основе теории сплайн-схем, точных на многочленах [3], строятся формулы рекуррентной аппроксимации сплайнами, позволяющие за счет учета предыстории обработки уменьшить время запаздывания или снизить погрешность приближения.

Общая схема. Пусть на отрезке вещественной оси [a, b] введена сетка Δ: $a = x_0 < x_1 < \ldots < x_N = b$. Добавим узлы $x_{-n} < \ldots < x_{-1} = a$, $b = x_{N+1} < \ldots < x_N$ и образуем множество нормализованных B-сплайнов ([1, с. 23]) $\left\{B_n^i(x), i = -n, \ldots, N-1\right\}$, равных нулю вне интервалов (x_i, x_{i+1}).

[1] Работа поддержана грантами РГНФ 06-02-64202 а/Т и 07-02-94773 и/м.

Определение [4]. Рекуррентным аппроксимационным сплайном $S_n(x)$ назовем функцию вида $S_{n,v}(x)=\sum_{\alpha=0}^{n} d_i B_\alpha^i(x), a\le x\le b.$ с коэффициентами d_i, удовлетворяющими рекуррентному соотношению $d_i=\lambda_i(f)-\sum_{j=1}^{p}\alpha_j^i\cdot d_{i-j}$, $k=p-n,\dots N-1$ и начальным условиям $d_i=\lambda_i(f), i=-n,\dots,p-n-1.$ Здесь $\{d_i=\lambda_i(f), i=-n,\dots,N-1\}$ — множество линейных непрерывных функционалов, определенных по значениям функции $f(x)$ на [a, b] .

2. Метод рекуррентной сплайн-аппроксимации степени 3 глубины 1

Математическая постановка задачи: Пусть измеряемый процесс $f(t)$ регистрируется в равноотстоящие моменты времени t с шагом измерения $\Delta t/m$, причем $f(t)$ допускает адекватное представление в виде сплайна третьей степени и между соседними узлами имеется m-1 измерение. Тогда измеренные значения $y_{i,k}$ на отрезках $[t_k, t_{k+1}]$, $k=0,1,\dots$, имеют вид [1]:

$$y_{i,k}=\frac{1}{6}\left(1-\frac{i}{m}\right)^3 y_{k-1}+\left[\frac{2}{3}-\left(\frac{i}{m}\right)^2+\frac{1}{2}\left(\frac{i}{m}\right)^3\right]y_k+$$
$$+\frac{1}{2}\left[\frac{1}{3}+\frac{i}{m}+\left(\frac{i}{m}\right)^2-\left(\frac{i}{m}\right)^3\right]y_{k+1}+\frac{1}{6}\left(\frac{i}{m}\right)^3 y_{k+2}+\xi_{i,k}, \qquad (2.1)$$

где y_k — параметры сплайна, $\xi_{ik}\in\square(0,\sigma^2)$ — ошибки измерения. Для нахождения оценок $\hat{y}_k$ параметров кубического сплайна будем использовать рекуррентный алгоритм глубины 1 [5]:

$$\hat{y}_k=\ \hat{y}_{k-1}+\sum_{i=0}^{m} a_i\ y_{i,k+1}\quad k=-\ \dots \qquad (2.2)$$

Теорема 2.1. Пусть параметры алгоритма (2.2) удовлетворяют условиям:

$$\sum_{i=0}^{m} a_i=1-\lambda,\sum_{i=0}^{m} a_i\frac{i}{m}=2\lambda-1,\sum_{i=0}^{m} a_i\left(\frac{i}{m}\right)^2=\frac{2-11\lambda}{3},\sum_{i=0}^{m} a_i\left(\frac{i}{m}\right)^3=6\lambda \qquad (2.3)$$

$$|\lambda|<1. \qquad (2.4)$$

Тогда оценки $\hat{y}_k$ вычисляются устойчивым образом и имеют дисперсию:

$$D\{\hat{y}_k\} = \frac{\sigma^2}{1-\lambda^2} \sum_{i=1}^{m} a_i^2. \qquad (2.5)$$

Сплайн, построенный в условиях теоремы 2.1, обеспечивает точность на многочленах третьей степени.

Доказательство. Проверим условия точности на многочленах третьей степени (см. лемма [5]). В обозначениях леммы имеем $\lambda_k(y) = \sum_{i=0}^{m} a_i\, y_{i,k+1}$, глубина фильтра $p = 1$. Тогда для того, чтобы сплайн совпадал тождественно с $f(t)$ для любых $f(t)$ из пространства многочленов степени не выше 3, необходимо и достаточно выполнения равенств

$$\begin{cases} \sum_{i=0}^{m} a_i + \lambda = 1, \\ \sum_{i=0}^{m} a_i\, t_{i,k+1} + \lambda t_{k-1} = t_k, \\ \sum_{i=0}^{m} a_i\, t_{i,k+1}^2 + \lambda\left(t_k^2 - 2t_k m + \frac{2}{3}m^2 \right) = t_k^2 - \frac{m^2}{3}. \\ \sum_{i=0}^{m} a_i\, t_{i,k+1}^3 + \lambda\left(t_k^3 - 3t_k^2 m + 2t_k m^2 \right) = t_k^3 - t_k m^2. \end{cases}$$

Здесь $t_{i,k} = t_k + \frac{i}{m}\Delta t,\ \ i = \overline{0,m}.$ Выполнение первого соотношения (2.3) очевидно, из первого равенства получим $\sum_{i=0}^{m} a_i = 1 - \lambda$. Далее, из второго равенства, после некоторых преобразований получим: $\sum_{i=0}^{m} a_i \frac{i}{m} = 2\lambda - 1$. Перейдем к третьему равенству системы и, подставляя $(t_k + m + i)^2$ вместо $t_{i,k+1}^2$ и в силу доказанных первых двух равенств, получим: $2\sum_{i=0}^{m} a_i mi + \sum_{i=0}^{m} a_i m^2 + \sum_{i=0}^{m} a_i i^2 + \frac{2}{3}\lambda m^2 + \frac{m^2}{3} = 0.$

Поделив обе части полученного равенства на m^2, приходим к третьему из доказываемых условий (2.3) теоремы 2.1. Аналогично, рассматривая четвертое равенство системы, получаем последнее $\sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^3 = 6\lambda$ из соотношений (2.3). Условия (2.3) теоремы 2.1 доказаны.

Обозначим $\tilde{y}_k = \sum_{i=0}^{m} a_i y_{i,k+1}$. Тогда (2.2) можно переписать в виде

$$\hat{y}_k = \lambda \hat{y}_{k-1} + \tilde{y}_k, \tag{2.6}$$

Уравнение (2.6) — линейное неоднородное разностное уравнение первого порядка, характеристическое уравнение которого имеет вид $z - \lambda z = 0$ корень уравнения равен λ, и выполнение условия (2.4) т. е. $|\lambda| < 1$, как известно [6], обеспечивает устойчивость решения уравнения (2.6) к влиянию начального условия $\hat{y}_{k-2}$.

Умножая левую и правую части выражения (2.1) на a_i и суммируя по $i = 0, ..., m$ и путем элементарных преобразований и с учетом условий (2.3), данное выражение принимает вид:

$$\tilde{y}_k = y_k - \lambda y_{k-1} + \sum_{i=0}^{m} a_i \xi_{i,k}, \ k = 1,2,\ldots. \tag{2.7}$$

Используя (2.7), выражение (2.6) может быть представлено в виде $\hat{y}_k - y_k = \lambda\left(\hat{y}_{k-1} - y_{k-1}\right) + \sum_{i=0}^{m} a_i \xi_{i,k}$.

Тогда

$$M\left\{\hat{y}_k - y_k\right\} = \lambda M\left\{\hat{y}_{k-1} - y_{k-1}\right\} + \sum_{i=0}^{m} a_i M\left\{\xi_{i,k}\right\}. \tag{2.8}$$

В выражении (2.8) $M\left\{\xi_{ik}\right\} = 0$ согласно постановке задачи. Таким образом, выражение (2.8) принимает окончательный вид

$$M\left\{\hat{y}_k - y_k\right\} = \lambda M\left\{\hat{y}_{k-1} - y_{k-1}\right\} \tag{2.9}$$

с тем же характеристическими уравнением, что и (2.6). Решение (2.9) асимптотически равно нулю при $|\lambda| < 1$, т. е. несмещенность решения $\hat{y}_k$ следует из его устойчивости. Для несмещенных оценок дисперсия

$$D\left\{\hat{y}_k - y_k\right\} = M\left\{\left(\hat{y}_k - y_k\right)^2\right\} \tag{2.10}$$

Тогда с учетом выражения (2.10) получаем:

$$M\left\{\left(\hat{y}_k - y_k\right)^2\right\} = \lambda^2 M\left\{\left(\hat{y}_{k-1} - y_{k-1}\right)^2\right\} + \sum_{i=0}^{m} a_i^2 \sigma^2$$

откуда при $k \to \infty$ следует $D\left\{\hat{y}_k\right\} = \dfrac{\sigma^2}{1-\lambda} \sum_{i=1}^{m} a_i^2$. Тогда при $k \to \infty$ получим (2.5). Теорема 2.1 доказана.

Таким образом, получен устойчивый рекуррентный вычислительный процесс оценки параметров $\hat{y}_k$ экспериментальной зависимости, представленной в виде сплайна третьей степени. Так как остаточная дисперсия оценок зависит от коэффициентов λ, a_i и дисперсии σ^2, то естественно определить такие параметры, которые минимизировали бы величину:

$$V(\lambda)=\frac{1}{1-\lambda^2}\sum_{i=0}^{m}a_i^2 \tag{2.11}$$

удовлетворяя при этом условиям точности на многочленах (2.3) и устойчивости (2.4). Решение поставленной задачи дает следующая.

Теорема 2.2. Оптимальные параметры алгоритма (2.2), минимизирующие (2.11) при условиях (2.3) и (2.4), имеют вид:

$$\lambda=\frac{\left(19568-\sqrt{357171295}\right)}{5073},\ a_i=\alpha+\beta\frac{i}{m}+\delta\left(\frac{i}{m}\right)^2+\mu\left(\frac{i}{m}\right)^3,$$

где

$$\alpha=\frac{-8\left(244505-13\sqrt{357171295}\right)}{267m},$$

$$\beta=\frac{60\left(2358068-125\sqrt{357171295}\right)}{1691m}, \tag{2.12}$$

$$\delta=\frac{-20\left(17193047-910\sqrt{357171295}\right)}{1691m},$$

$$\mu=\frac{140\left(1607477-85\sqrt{357171295}\right)}{1691m}.$$

Доказательство. С использованием множителей Лагранжа β_1, β_2, β_3 и β_4 задача на условный экстремум сводится к задаче на безусловный экстремум

$$I=\frac{1}{1-\lambda^2}\sum_{i=0}^{m}a_i^2+\beta_1\left(1-\lambda-\sum_{i=0}^{m}\frac{i}{m}a_i\right)+\beta_2\left(2\lambda-1-\sum_{i=0}^{m}\frac{i}{m}a_i\right)+$$

$$+\beta_3\left(\frac{2-11\lambda}{3}-\sum_{i=0}^{m}a_i\left(\frac{i}{m}-\frac{1}{2}\right)^2\right)+\beta_4\left(6\lambda-\sum_{i=0}^{m}a_i\left(\frac{i}{m}\right)^3\right)\underset{\lambda,a_i}{\to}\min.$$

Вычисляя частные производные по варьируемым параметрам и приравнивая их к нулю, получим систему уравнений, обеспечивающих выполнение необходимых условий экстремума.

$$\frac{\partial I}{\partial a_i} = \frac{2a_i}{1-\lambda^2} - \beta_1 - \beta_2 \frac{i}{m} - \beta_3 \left(\frac{i}{m}\right)^2 - \beta_4 \left(\frac{i}{m}\right)^3, \qquad (2.13)$$

$$\frac{\partial I}{\partial \lambda} = \frac{2\lambda}{(1-\lambda^2)^2} \sum_{i=0}^{m} a_i^2 - \beta_1 + \beta_2 - \frac{11}{3}\beta_3 + 6\beta_4. \qquad (2.14)$$

Введем обозначения

$$\alpha = \frac{\beta_1\left(1-\lambda^2\right)}{2}, \beta = -\frac{\beta_2\left(1-\lambda^2\right)}{2}, \delta = \frac{\beta_3\left(1-\lambda^2\right)}{2}, \mu = \frac{\beta_4\left(1-\lambda^2\right)}{2}.$$

Тогда из (2.13), учитывая введенные обозначения, получаем

$$a_i = \alpha + \beta \frac{i}{m} + \delta \left(\frac{i}{m}\right)^2 + \mu \left(\frac{i}{m}\right)^3, \qquad (2.15)$$

из (2.14):

$$\frac{\lambda}{1-\lambda^2} \sum_{i=0}^{m} a_i^2 - \alpha + 2\beta - \frac{11}{3}\delta + 6\mu = 0, \qquad (2.16)$$

где α, β, δ и μ должны быть выбраны так, чтобы выполнялись (2.3) и (2.4).

Умножив (2.15) на a_i и просуммировав по i, получим

$$\sum_{i=0}^{m} a_i^2 - \alpha \sum_{i=0}^{m} a_i - \beta \sum_{i=0}^{m} a_i \frac{i}{m} - \delta \sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^2 - \mu \sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^3 = 0. \qquad (2.17)$$

Далее, с учетом (2.3)

$$\sum_{i=0}^{m} a_i^2 = \alpha(1-\lambda) + \beta(2\lambda - 1) + \delta\left(\frac{2-11\lambda}{3}\right) + 6\mu\lambda. \qquad (2.18)$$

Поэтому (2.16) и (2.18) дают нам

$$\mu = \frac{1}{18}\left[3\alpha(1-\lambda) - 3\beta(2-\lambda) - \delta(2\lambda - 11)\right] \qquad (2.19)$$

Так как данные выражения трудны для анализа, найдем асимптотическое решение при $m \to \infty$. Подставляя (2.19) в (2.15), и с учетом (2.3), при $m \to \infty$ получаем систему:

$$\begin{cases} \alpha + \frac{\beta}{2} + \frac{\delta}{3} + \frac{\mu}{4} = \frac{1-\lambda}{m}, \\ \frac{\alpha}{2} + \frac{\beta}{3} + \frac{\delta}{4} + \frac{\mu}{5} = \frac{2\lambda - 1}{m}, \\ \frac{\alpha}{3} + \frac{\beta}{4} + \frac{\delta}{5} + \frac{\mu}{6} = \frac{2 - 11\lambda}{3m}, \\ \frac{\alpha}{4} + \frac{\beta}{5} + \frac{\delta}{6} + \frac{\mu}{7} = \frac{6\lambda}{m}. \end{cases}$$

Решая систему, находим, что

$$\begin{aligned} \alpha_{1,2} &= \frac{-8\left(244505 \pm 13\sqrt{357171295}\right)}{267m}, \\ \beta_{1,2} &= \frac{60\left(2358068 \pm 125\sqrt{357171295}\right)}{1691m}, \\ \delta_{1,2} &= \frac{-20\left(17193047 \pm 910\sqrt{357171295}\right)}{1691m}, \\ \mu_{1,2} &= \frac{140\left(1607477 \pm 85\sqrt{357171295}\right)}{1691m}, \\ \lambda_{1,2} &= \frac{19568 \pm \sqrt{357171295}}{5073}. \end{aligned} \qquad (2.20)$$

Так как нас интересуют из (2.20), только те λ, которые по модулю меньше 1, т. е.:

$$\lambda = \frac{\left(19568 - \sqrt{357171295}\right)}{5073}, \qquad (2.21)$$

то имеем:

$$\alpha = \frac{-8\left(244505 - 13\sqrt{357171295}\right)}{267m},$$
$$\beta = \frac{60\left(2358068 - 125\sqrt{357171295}\right)}{1691m},$$
$$\delta = \frac{-20\left(17193047 - 910\sqrt{357171295}\right)}{1691m}, \qquad (2.22)$$
$$\mu = \frac{140\left(1607477 - 85\sqrt{357171295}\right)}{1691m}.$$

Подставляя (2.22) в (2.15) имеем:

$$a_i = \frac{-8\left(244505 - 13\sqrt{357171295}\right)}{267m} +$$
$$+ \frac{60\left(2358068 - 125\sqrt{357171295}\right)i}{1691m^2} +$$
$$+ \frac{20\left(-17193047 + 910\sqrt{357171295}\right)i^2}{1691m^3} + \qquad (2.23)$$
$$+ \frac{140\left(1607477 - 85\sqrt{357171295}\right)i^3}{1691m^4}.$$

Собирая (2.21) и (2.22) или (2.23) получаем утверждение теоремы 2.2. Дисперсия, соответствующая оптимальным параметрам (2.12), имеет значение:

$$D\{\hat{y}_k\} = \frac{\text{-8.021}\cdot 10^{-4}\sigma^2}{m^4}\left(9291190m^3 - 494m^3\sqrt{357171295} - \right.$$
$$-106113060im^2 + 5625im^2\sqrt{357171295} - 25789570i^2m -$$
$$\left.-13650i^2m\sqrt{357171295} - 168785085i^3 + 8925i^3\sqrt{357171295}\right)$$

Теорема 2.2. доказана.

3. Рекуррентная сплайн-аппроксимация степени 3 глубины 2

В отличие от предыдущего раздела, для нахождения оценок $\hat{y}_k$ параметров кубического сплайна будем использовать рекуррентный алгоритм глубины 2 [7]. Такой алгоритм можно определить согласно соотношению:

$$\hat{y}_k = \lambda \hat{y}_{k-1} + \gamma \hat{y}_{k-2} + \sum_{i=0}^{m} a_i \, y_{i,k+1}, \;\; k = 1,2,\ldots, \tag{3.1}$$

Теорема 3.1. Пусть параметры алгоритма (3.1) удовлетворяют условиям:

$$\begin{gathered} \sum_{i=0}^{m} a_i = 1-\lambda-\gamma, \sum_{i=0}^{m} a_i \frac{i}{m} = 2\lambda + 3\gamma - 1, \\ \sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^2 = \frac{2-11\lambda-26\gamma}{3}, \sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^3 = 6\lambda + 24\gamma \end{gathered} \tag{3.2}$$

$$|\lambda| + |\gamma| < 1. \tag{3.3}$$

Тогда оценки $\hat{y}_k$ вычисляются устойчивым образом, являются асимптотически несмещенными и имеют дисперсию при $k \to \infty$

$$D\{\hat{y}_k\} = \frac{\sigma^2}{1-\lambda^2-\gamma^2} \sum_{i=0}^{m} a_i^2. \tag{3.4}$$

Сплайн, построенный в условиях теоремы 3.1, обеспечивает точность на многочленах третьей степени.

Доказательство теоремы 3.1 и вычисление оптимальных параметров алгоритма (3.2) для рекуррентной сплайн-аппроксимации степени 3 глубины 2 см в [8].

4. Рекуррентная сплайн-аппроксимация степени 3 произвольной глубины p

В отличие от ранее исследованных случаев рекуррентной сплайн-аппроксимации степени 3 глубины 1 и 2, для нахождения оценок параметров кубического сплайна будем использовать рекуррентный алгоритм произвольной глубины p:

$$\hat{y}_k = \sum_{q=1}^{p} \lambda_q \hat{y}_{k-q} + \sum_{i=0}^{m} a_i y_{i,k+1}, \quad k = p-1, p, \ldots . \tag{4.1}$$

Теорема 4.1. Пусть параметры алгоритма (4.1) удовлетворяют условиям:

$$\begin{gathered}
\sum_{i=0}^{m} a_i = 1 - \sum_{q=1}^{p} \lambda_q, \quad \sum_{i=0}^{m} a_i \frac{i}{m} = \sum_{q=1}^{p} \lambda_q \left(1+q\right) - 1, \\
\sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^2 = \frac{2}{3} - \sum_{q=1}^{p} \lambda_q \left(\frac{2+6q+3q^2}{3} \right), \\
\sum_{i=0}^{m} a_i \left(\frac{i}{m}\right)^3 = \sum_{q=1}^{p} \lambda_q \left(2q + 3q^2 + 3q^3\right)
\end{gathered} \tag{4.2}$$

При

$$\sum_{q=1}^{p} \left|\lambda_q\right| < 1. \tag{4.3}$$

Тогда оценки $\hat{y}_k$ вычисляются устойчивым образом и имеют конечную дисперсию

$$D\left\{\hat{y}_k\right\} = \frac{\sigma^2}{1 - \sum_{q=1}^{p} \lambda_q^2} \sum_{i=1}^{m} a_i^2 \tag{4.4}$$

Сплайн, построенный в условиях теоремы 4.1, обеспечивает точность на многочленах третьей степени.

Доказательство теоремы 4.1 для рекуррентной сплайн-аппроксимации степени 3 глубины p потребовало применения другой техники, основанной на теореме о диагональном преобладании, и будет опубликовано дополнительно.

5. Численные эксперименты

Приведем конкретные примеры для рекуррентной сплайн-аппроксимации степени 3 глубины $p = 1$.

При $m < 2$ решения не существует, так как количество параметров меньше количества условий теоремы 2.1. При $m = 2$ точки

измерения удобно располагать в узлах сплайна и по одной точке посередине между узлами сплайна. Тогда значения $t_i = 0, \frac{1}{2}, 1$. В результате имеем 4 неизвестных значения λ, α_0, α_1, α_2, для определения которых требуется решить систему линейных алгебраических уравнений вида:

$$\begin{cases} a_0 + a_1 + a_2 + \lambda = 1, \\ \frac{1}{2}a_1 + a_2 - 2\lambda = -1, \\ \frac{1}{4}a_1 + a_2 + \frac{11}{3}\lambda = \frac{2}{3}, \\ \frac{1}{8}a_1 + a_2 - 6\lambda = 0. \end{cases}$$

Детерминант матрицы системы равен $-25/8$, поэтому существует единственное решение:

$$a_0 = \frac{271}{75}, a_1 = \frac{-296}{75}, a_2 = \frac{91}{75}, \lambda = \frac{3}{25}.$$

Поскольку значение $\lambda = 3/25$ по модулю <1, полученное решение полностью удовлетворяет условиям теоремы 2.1.

Следующий вариант — когда точки измерения располагаются в узлах сплайна и еще по 2 точки измерения располагаются равномерно между узлами сплайна. Если для расчета коэффициентов сплайна берутся точки, соответствующие $t_i = 1/3, 2/3, 1$ то для определения 4-х неизвестных значений λ, α_1, α_2, α_3 требуется решить систему линейных алгебраических уравнений

$$\begin{cases} a_0 + a_1 + a_2 = 1 - \lambda, \\ \frac{1}{3}a_1 + \frac{2}{3}a_2 = 2\lambda - 1, \\ \frac{1}{9}a_1 + \frac{4}{9}a_2 = \frac{2 - 11\lambda}{3}, \\ \frac{1}{27}a_1 + \frac{8}{27}a_2 = 6\lambda. \end{cases}$$

Детерминант матрицы системы равен 0,016, поэтому единственное решение существует. Однако, поскольку при этом ока-

зывается, что $\lambda = \frac{25}{144} = 0{,}174$, полученное решение полностью удовлетворяет условиям теоремы 2.1. Тем не менее, существует вполне приемлемый вариант, соответствующий точкам $t_i = 0, \frac{1}{3}, \frac{2}{3}$, в котором для определения неизвестных значений λ, α_0, α_1, α_2 требуется решить систему линейных алгебраических уравнений

$$\begin{cases} a_0 + a_1 + a_2 = 1 - \lambda, \\ \frac{1}{3}a_1 + \frac{2}{3}a_2 = 2\lambda - 1, \\ \frac{1}{9}a_1 + \frac{4}{9}a_2 = \frac{2 - 11\lambda}{3}, \\ \frac{1}{27}a_1 + \frac{8}{27}a_2 = 6\lambda. \end{cases}$$

Детерминант матрицы системы равен $\frac{-182}{243}$, поэтому единственное решение существует $a_0 = \frac{1123}{182}, a_1 = \frac{-732}{91}, a_2 = \frac{39}{14}, \lambda = \frac{8}{91}$ и оно также полностью удовлетворяет условиям теоремы 2.1.

Пусть теперь $m = 3$. В результате имеем 5 неизвестных значений λ, α_0, α_1, α_2, α_3, для определения которых требуется решить недоопределенную систему 4-х линейных алгебраических уравнений:

$$\begin{cases} a_0 + a_1 + a_2 + a_3 = 1 - \lambda, \\ \frac{1}{3}a_1 + \frac{2}{3}a_2 + a_3 = 2\lambda - 1, \\ \frac{1}{9}a_1 + \frac{4}{9}a_2 + a_3 = \frac{2 - 11\lambda}{3}, \\ \frac{1}{27}a_1 + \frac{8}{27}a_2 + a_3 = 6\lambda. \end{cases}$$

Детерминант матрицы системы равен $4/243$, поэтому существует общее решение, зависящее от параметра λ,

$$a_0 = -72\lambda + \frac{25}{2},\ a_1 = \frac{363}{2}\lambda - 24,\ a_2 = -156\lambda + \frac{33}{2}, a_3 = \frac{91}{2}\lambda - 4.$$

Графики полученных значений коэффициентов в зависимости от λ при $|\lambda| < 1$ представлены на рис. 1.

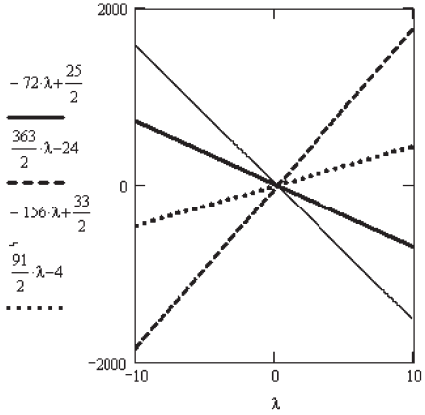


Рис. 1. Графики полученных значений коэффициентов α_0, α_1, α_2, α_3 в зависимости от λ при $|\lambda|<1$

Отметим, что к множеству полученных решений относятся и оба полученных выше решения для случая $m=3$. В силу неединственности естественно определить такой параметр, который минимизировал бы остаточную дисперсию оценок (2.13) $V(\lambda)=\frac{1}{1-\lambda^2}\sum_{i=0}^{m}a_i^2$ удовлетворяя при этом условиям точности на многочленах (2.3) и устойчивости (2.4). График функции $V(\lambda)$ представлен на рис. 2. Единственный минимум функции $V(\lambda)$ достигается при значении $\lambda=0{,}124$, $|\lambda|<1$. Соответствующие значения коэффициентов равны $a_0=3{,}577$, $a_1=-1{,}507$, $a_2=-2{,}833$, $a_3=1{,}639$.

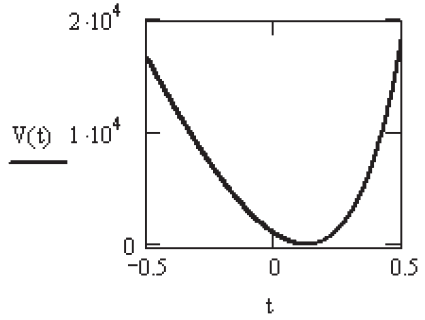


Рис. 2. Графики зависимости $V(\lambda)$ при $|\lambda|<1$

С использованием полученных выше устойчивых рекуррентных формул сплайн-прогнозирования была проделана серия численных экспериментов по аппроксимации мономов степени, меньшей или равной степени сплайна 3.

1. Было установлено, что полученные формулы обеспечивают нулевую погрешность приближения функций $f(t)=t^l$, $l=0,1,2,3$. При этом, полученные графики практически не отличаются.

2. Были проведены численные эксперименты с зашумленными функциями с нормально распределенной помехой с нулевым математическим ожиданием и с дисперсией σ^2, $\sigma=1$, 5, 10, 15. На рис. 3 приведен результат для рекуррентного сплайн-прогноза степени 3 глубины 1 для функции $f(t)=t^3$, $\sigma=10$. Оказалось, что для $\sigma\le 9$ свойства точно воспроизводить многочлены достаточно высокой степени выполняются, для $\sigma\ge 10$ фильтр не может погасить большую погрешность.

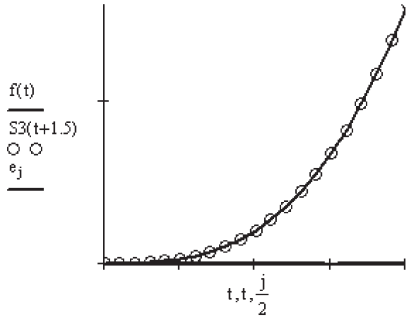


Рис. 3. Графики функций: $f(t)=t^3$, $S3(t)$ — сплайн-прогноза 3-й степени, глубины 1, e_j — помеха

3. В последней части эксперимента рассматривалось приближение многочлена степени 4, т. е. большей, чем степень сплайн-прогноза. Результирующая погрешность $S_3\left(x^{3+l},t\right)-t^{3+l}$ носит название моносплайна степени $3+1$ и в большинстве случаев имеет характерный график. Как известно, моносплайны представляют собой главный член погрешности для приближения достаточно гладких функций. Для $l=1,2$ результаты приведены на рис. 4.

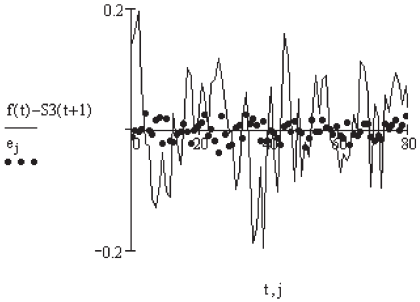


Рис. 4. Погрешность $S3(x^{3+1},t) - t^{3+1}$ для сплайна степени 3 глубины 1

4. Для случая зашумленных данных результат аналогичен предыдущему. В частности, для монома 4-й степени результат приведен на рис. 5.

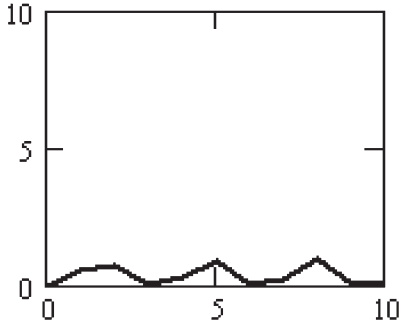


Рис. 5. Результирующая погрешность $S3(x^4+\xi,t) - t^4$ для сплайна степени 3 глубины 1

Таким образом, повышение степени фильтра благоприятно сказывается на погрешности полученных прогнозных значений при условии гладкости аппроксимируемой зависимости [9].

При $m>3$ число неизвестных параметров алгоритма (3) значительно больше, чем количество условий теоремы 2.1. Так как остаточная дисперсия оценок в условиях теоремы 2.1 при $k>>3$ зависит от параметров самого алгоритма λ, α_i и дисперсии σ^2, то естественно определить такие параметры, которые минимизировали бы величину $V = \frac{1}{1-\lambda^2}\sum_{i=0}^{m} a_i^2$, удовлетворяя при этом условиям точности на многочленах (2.3) и устойчивости (2.4). Решение

поставленной задачи в каждом конкретном случае выполняется аналогично теореме 2.2.

Литература

1. **Завьялов Ю. С., Квасов Б. И., Мирошниченко В. Л.** Методы сплайн-функций. — М. : Наука, 1980. — 352 с.

2. **Шумилов Б. М.** Рекуррентная интерполяция кубическими сплайнами с дополнительными узлами // Журн. вычисл. матем. и матем. физ. — 1990. — Т. 29. — № 2. — С. 179—185 с.

3. **Шумилов Б. М.** Сплайн-аппроксимационные схемы, точные на многочленах // Журн. вычисл. матем. и матем. физ. — 1992. — Т. 32. — № 8. — С. 1187—1196.

4. **Шумилов Б. М.** Рекуррентная аппроксимация сплайнами // Известия высших учебных заведений. — 1996. — № 1. — С. 85—87.

5. **Шумилов Б. М., Эшаров Э. А.** Метод рекуррентной сплайн-аппроксимации степени 3 глубины 1 // VI Всеросс. конф. мол. ученых по матем. моделир. и информ. технологиям (с участием иностр. ученых) : Программа и тез. докл. — Кемерово, 2005.

6. **Гельфонд А. О.** Исчисление конечных разностей. — 3-е изд., испр. — М. : Наука, 1967. — 376 с.

7. **Шумилов Б. М., Эшаров Э. А.** Рекуррентная аппроксимация глубины 2 сплайнами третьей степени// VII Всерос. конф. молодых ученых по математ. моделированию и информ. технологиям (с участиям иностр. ученых). — Красноярск, 2006. — С. 32—33.

8. **Эшаров Э. А.** Оценка погрешности и примеры рекуррентного сплайна третьей степени глубины 2 // Мат-лы XLV Международ. науч. студенческой конф. «Студент и научно-технический прогресс»: Математика / Новосиб. гос. ун-т. — Новосибирск, 2007.

9. **Шумилов Б. М., Карлова И. В., Ярушкина Н. А.** Рекуррентные и прогнозирующие сплайн-фильтры 1-го и 2-го порядков // Тр. Международ. конф. по вычислит. математике МКВМ-2004. — Новосибирск : Изд-во ИВМ и МГСО РАН, 2004. — Ч. 1. — С. 152—157.

Параллельная реализация алгоритма решения задачи о транспортировке пучка заряженных частиц

А. А. Трунов, А. В. Старченко,
И. Ю. Турчановский, В. А. Шкляев**
Томский государственный университет, Томск
*Институт сильноточной электроники СО РАН, Томск

Рассматривается осесимметричная задача о транспортировке пучка заряженных частиц в вакуумном зазоре, ограниченном двумя параллельно расположенными проводящими круглыми электродами (рис. 1). Зазор имеет открытую боковую поверхность. Все заряженные частицы обладают одинаковой начальной энергией и удельным зарядом. Плотность тока инжекции распределена равномерно по области инжекции. Область инжекции расположена на одном из торцевых электродов, ее площадь много меньше площади электрода, с которого осуществляется инжекция.

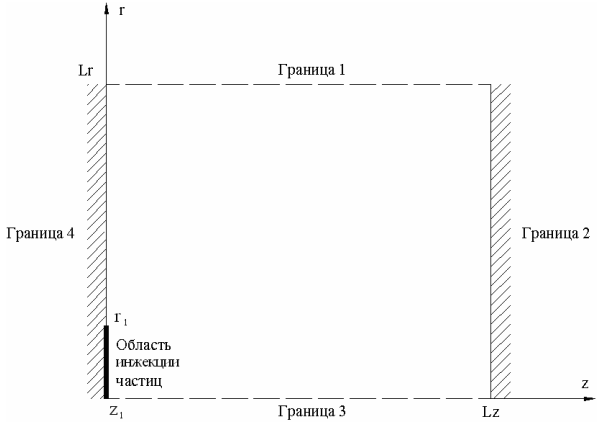


Рис.1. Вакуумный зазор

В такой постановке задача имеет известное стационарное аналитическое решение, что позволяет провести тестирование кода.

Основные уравнения модели

Для решения данной задачи применяется метод крупных частиц (PIC) [1—4]. В модели решаются уравнения движения

частиц под действием собственных электрических и магнитных полей. Собственные поля, создаваемые частицами, находятся из решения уравнений Максвелла (1) по известным координатам и скоростям частиц.

$$\begin{aligned} &\frac{1}{c}\frac{\partial \overline{B}^i}{\partial t} = -rot\overline{E}^i; \\ &\frac{1}{c}\frac{\partial \overline{E}^i}{\partial t} = rot\overline{B}^i - \frac{4\pi}{c}\overline{J}, \end{aligned} \tag{1}$$

где:
$\overline{B}^i = \left(B_r^i, B_\varphi^i, B_z^i\right)$ — вектор собственного магнитного поля;
$\overline{E}^i = \left(E_r^i, E_\varphi^i, E_z^i\right)$ — вектор собственного электрического поля;
$\overline{J} = \left(J_r, J_\varphi, J_z\right)$ — вектор плотности тока;
c — скорость света.

Распишем компоненты векторов электрического и магнитного поля с учетом осевой симметрии задачи $\left(\frac{\partial}{\partial \varphi} = 0\right)$:

$$\begin{aligned} &\frac{1}{c}\frac{\partial B_r^i}{\partial t} = \frac{\partial E_\varphi^i}{\partial z}; \\ &\frac{1}{c}\frac{\partial B_z^i}{\partial t} = -\frac{1}{r}\frac{\partial}{\partial r}\left(rE_\varphi^i\right); \\ &\frac{1}{c}\frac{\partial B_\varphi^i}{\partial t} = -\frac{\partial E_r^i}{\partial z} + \frac{\partial E_z^i}{\partial r}; \\ &\frac{1}{c}\frac{\partial E_r^i}{\partial t} = -\frac{\partial B_\varphi^i}{\partial z} - \frac{4\pi}{c}J_r; \\ &\frac{1}{c}\frac{\partial E_z^i}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(rB_\varphi^i\right) - \frac{4\pi}{c}J_z; \\ &\frac{1}{c}\frac{\partial E_\varphi^i}{\partial t} = \frac{\partial B_r^i}{\partial z} - \frac{\partial B_z^i}{\partial r} - \frac{4\pi}{c}J_\varphi. \end{aligned} \tag{2}$$

Помимо уравнений (1) система уравнений Максвелла содержит также уравнения:

$$\begin{aligned} &div\,\overline{B}^i = 0; \\ &div\,\overline{E}^i = 4\pi\rho, \end{aligned} \tag{3}$$

где ρ — плотность заряда.

Распишем уравнения (3) для компонент векторов электрического и магнитного поля с учетом осевой симметрии задачи $\left(\frac{\partial}{\partial \varphi}=0\right)$:

$$\begin{aligned}&\frac{\partial B_z^i}{\partial x}+\frac{1}{r}\frac{\partial}{\partial r}\left(rB_r^i\right)=0;\\&\frac{\partial E_z^i}{\partial x}+\frac{1}{r}\frac{\partial}{\partial r}\left(rE_r^i\right)=4\pi\rho.\end{aligned}\qquad(4)$$

Соотношения (4) должны выполняться на любом шаге по времени. Известно [2], что если уравнения (4) выполняются на начальном шаге, то они будут справедливы и на последующих шагах по времени, при отсутствии вычислительных погрешностей. Поэтому при приближенном вычислении полей по соотношениям (2) требуется их корректировка.

Начальные условия для векторов электромагнитного поля:

$$E_z^i=E_r^i=E_\varphi^i=B_z^i=B_r^i=B_\varphi^i=0.$$

Граничные условия для векторов электромагнитного поля:
$z=0$: $E_r^i=E_\varphi^i=B_z^i=0$ ($0\le r\le Lr$ — граница идеального проводника), граница 4 (рис. 1);
$z=Lz$: $E_r^i=E_\varphi^i=B_z^i=0$ ($0\le r\le Lr$ — граница идеального проводника), граница 2 (рис. 1);
$r=0$: $E_\varphi^i=B_r^i=0$ ($0\le z\le Lz$ — ось симметрии), граница 3 (рис. 1);
$r=Lr$: $E_z^i=E_r^i=E_\varphi^i=B_z^i=B_r^i=B_\varphi^i=0$ ($0\le z\le Lz$ — открытая граница, достаточно удаленная от зарядов), граница 1 (рис. 1).

Движение частиц описывается уравнениями Ньютона-Лоренца:

$$\frac{d\overline{u}_p}{dt}=\frac{q}{m}\left(\overline{E}+\frac{1}{c}\left[\frac{\overline{u}_p}{\gamma}\times\overline{B}\right]\right),\qquad(5)$$

$$\frac{d\overline{x}_p}{dt}=\frac{\overline{u}_p}{\gamma},\qquad(6)$$

где $\overline{u}_p$ — релятивистская скорость частицы, $\overline{u}_p=\overline{v}_p\cdot\gamma$; $\overline{v}_p=\left(v_{zp},v_{rp},v_{\varphi p}\right)$ — скорость частицы; q — заряд частицы; m — масса покоя частицы; $\overline{E}$ – электрическое поле, действующее на

частицу; γ – релятивистский коэффициент, $\gamma = \sqrt{1 + \frac{\left|\overline{u}_p\right|^2}{c^2}}$; $\overline{B}$ — магнитное поле, действующее на частицу, $\overline{x}_p = \left(r_p, z_p\right)$– радиус-вектор частицы.

Для решения уравнений (5), (6) имеются начальные условия:

$$t = 0 : v_{zp} \approx 1{,}9 \cdot 10^9 \text{ см/с},\ v_{rp} = 0,\ v_{\varphi p} = 0;$$
$$0 \le r_p \le r_1,\ \ 0 \le z_p \le z_1.$$

Частицы распределяются по псевдослучайному равномерному закону на небольшом расстоянии от инжектирующего электрода вблизи оси (см. рис. 1). В случае если частица достигает любой границы расчетной области, кроме оси симметрии, исследование ее движения прекращается. При переходе через ось симметрии частица «отражается», т. е. знак радиальной компоненты скорости v_{rp} изменяется на противоположный.

Вычислительные технологии

Для решения уравнений Максвелла используется явная разностная схема, обеспечивающая точное выполнение граничных условий для векторов электромагнитного поля на идеальном проводнике (на рис. 2 приведен фрагмент сетки). Смещенные в пространстве и времени равномерные сетки обеспечивают второй порядок аппроксимации [2]. Данная разностная схема успешно используется в электромагнитном коде KARAT [5].

Для интегрирования уравнений движения частиц используется алгоритм Бориса для осесимметричной задачи, основывающийся на алгоритме интегрирования с перешагиванием (leap-frog), который имеет второй порядок аппроксимации [2].

Вычислительный цикл программы состоит из попеременного решения двух задач: нахождения собственных полей (эйлеров этап) и интегрирования уравнений движения частиц под действием собственных полей (лагранжев этап).

Поскольку поля определяются на эйлеровой сетке, а координаты и скорости частиц на лагранжевой, то необходима операция раздачи плотности заряда и плотности тока в узлы эйлеровой

сетки для решения уравнений Максвелла и обратная процедура — для интегрирования уравнений движения. В качестве такой процедуры применена PIC-интерполяция [4].

Полученный тайминг последовательной программы показал, что доля временных затрат центрального процессора, приходящихся на вычисление электромагнитного поля по уравнениям (1)—(4), составляет менее 1 % при использовании пространственных сеток 100×100. Поскольку эйлеров этап занимает время много меньшее лагранжева, проведено распараллеливание только лагранжева этапа.

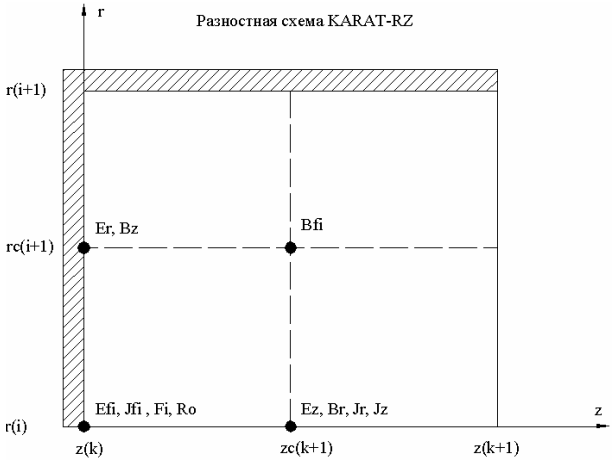


Рис. 2. Сдвинутые сетки

При параллельной реализации алгоритма использовался язык программирования FORTRAN с библиотекой MPI. Уравнения движения (5)—(6) распределенных по процессам частиц интегрируются параллельно, затем процедурой MPI_ALLREDUCE осуществляется сборка рассчитанных сеточных плотностей тока J_r, J_φ, J_z на всех процессах. На эйлеровом этапе все процессы интегрируют уравнения Максвелла во всей расчетной области.

Балансировка загрузки

Поскольку число частиц в расчетной области — величина переменная, то возможна ситуация, когда различные процессы обрабатывают существенно различное число частиц, что неудов-

летворительно сказывается на производительности параллельной программы в целом. Поэтому необходима процедура выравнивания числа частиц каждого процесса, обычно называемая динамической балансировкой загрузки процессов [7].

Балансировка загрузки процессов осуществляется при помощи алгоритма регулирования инжекции частиц: большее число частиц, инжектируемых на данном шаге по времени, получает процесс, имеющий в расчетной области минимальное число частиц. Результаты расчетов показывают, что дисбаланс загрузки составляет доли процента, при этом временные затраты процессорного времени на балансировку минимальны. Также стоит отметить небольшую трудоемкость программирования и отладки данного алгоритма.

Результаты

На рис. 3 приведен график зависимости ускорения и эффективности работы параллельной программы от числа процессов при количестве обрабатываемых частиц порядка 10^6. Пунктиром показано максимальное теоретическое ускорение, вычисленное в соответствии с законом Амдаля [6]. Эффективность работы параллельной программы рассчитывалась как отношение реального ускорения к теоретическому. Полученные результаты говорят о хорошей эффективности распараллеливания алгоритма.

Полученное на кластере ТГУ СКИФ Cyberia решение (см. рис. 4, 5) удовлетворительно согласуется с известным аналитическим решением [8, 9], а также с решением подобной задачи PIC-кодом KARAT [5]. Число частиц в расчетах составляло приблизительно $2 \cdot 10^6$.

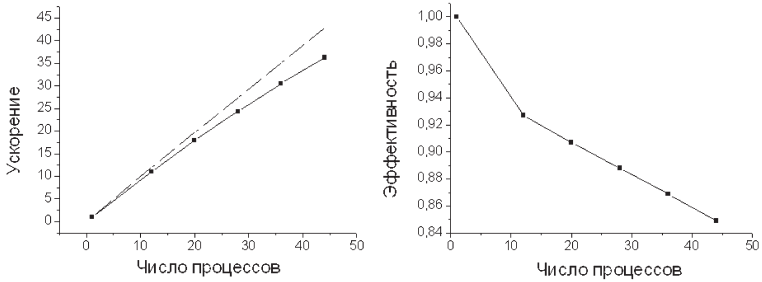


Рис. 3. Зависимости ускорения и эффективности работы параллельной программы от числа процессов

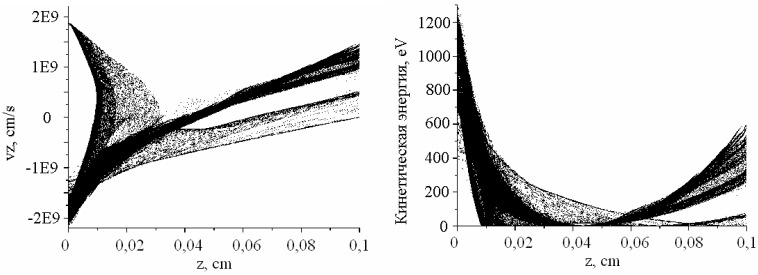
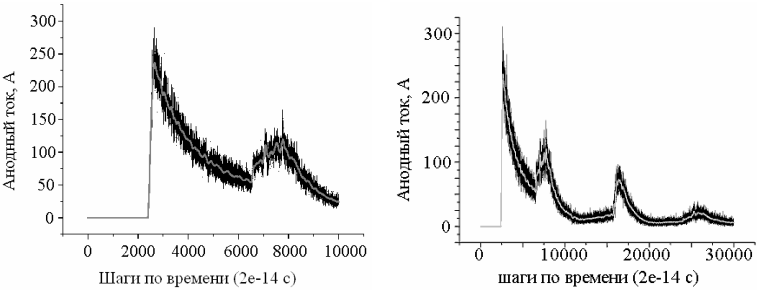


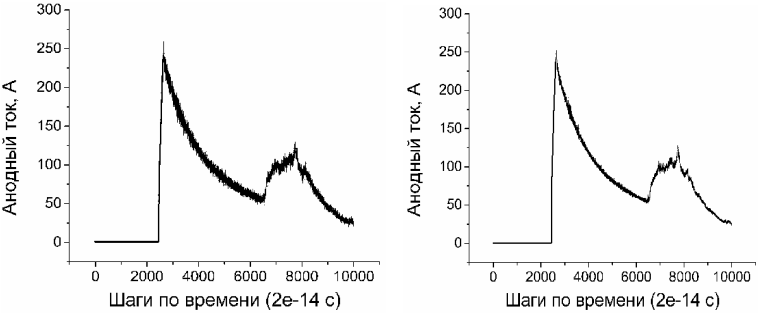
Рис. 4. Зависимости z-компоненты скорости
и кинетической энергии частиц от координаты z



а б

Рис. 5. Зависимости анодного тока от времени (число частиц $\approx 2 \cdot 10^6$)

Также были проведены расчеты с количеством обрабатываемых частиц порядка 10^7 и 10^8 (рис. 6). При этом получено значительное уменьшение шума в анодном токе (см. рис. 5, 6).



а) число частиц $\approx 2 \cdot 10^7$ б) число частиц $\approx 2 \cdot 10^8$

Рис. 6. Зависимости анодного тока от времени

Необходимо отметить, что вследствие отсутствия корректировки полей, о которой упоминалось выше, наблюдается накопление погрешностей решения при счете на значительное число шагов по времени. Это видно из характера поведения анодного тока (рис. 4 а, б) — форма колебаний тока не вполне согласуется с [8, 9]. При дальнейшей доработке кода планируется использовать корректировку полей [2] или расчет плотности тока, при котором выполняется закон непрерывности [10].

Литература

1. **Григорьев Ю. Н., Вшивков В. А., Федорук М. П.** Численное моделирование методами частиц-в-ячейках. — Новосибирск : Издательство СО РАН, 2004. — 360 с.

2. **Бэдсел Ч., Ленгдон А.** Физика плазмы и численное моделирование. — М. : Энергоатомиздат, 1989. — 452 с.

3. **Березин Ю. А., Вшивков В. А.** Метод частиц в динамике разреженной плазмы. — Новосибирск : Наука, 1980. — 96 с.

4. **Хокни Р., Иствуд Дж.** Численное моделирование методом частиц. — М. : Мир, 1987. — 640 с.

5. **Tarakanov V. P.** Users's manual for code KARAT (ver. 7.03, May 97). — Berkley Research Associates, Inc., 1997. — 109 p.

6. **Старченко А. В., Есаулов А. О.** Параллельные вычисления на многопроцессорных вычислительных системах. — Томск : Изд-во Том. ун-та, 2002. — 56 с.

7. **Малышкин В. Э.** Параллельная реализация алгоритмов численного моделирования, Сибирская школа-семинар по параллельным вычислениям / Под ред. проф. А. В. Старченко. — Томск : Изд-во Том. ун-та, 2002. — 129 с.

8. **Миллер Р.** Введение в физику сильноточных пучков заряженных частиц. — М. : Мир, 1984. — 432 с.

9. **Bridges W.B., Birdsall C. K.** Space-charge instabilities in electron diodes // J. Appl. Phys. — 1963. — № 34. — P. 2946.

10. **Скачков М. В.** О проблеме шумов и сохранения заряда в методе крупных частиц // Матем. моделирование. — 2000. — Т. 12. — № 9. — С. 96—108.

Содержание

Берцун В. Н. Из истории развития вычислительной математики в Томском университете 3

Вшивков В. А., Киреев С. Е., Кукшева Е. А., Снытников А. В., Снытников Н. В. Стратегии распараллеливания в задаче о динамике протопланетного диска 20

Квасов Б. И. Распараллеливание вычислений при построении гиперболических сплайнов 37

Афанасьев К. Е., Демидов А. В., Трофимов С. Н. Научно-образовательный портал параллельных вычислений 53

Старченко А. В., Ибраев Г. М. Опыт создания вычислительного метакластера на базе кластерных систем Томского научного центра ... 61

Романов А. Н., Сулимов В. Б., Кондакова О. А., Сулимов А. В., Григорьев Ф. В., Синауридзе Е. И., Кузнецов Ю. В., Воеводин В. В., Атауллаханов Ф. И. Новые ингибиторы тромбина: молекулярный дизайн с использованием суперкомпьютеров и экспериментальное подтверждение активности 79

Шагинян А. А., Гарабекян Г. А., Погосян А. Г., Арсенян Л. Г. Исследование некоторых задач молекулярной биологии и химии поверхностно-активных соединений при помощи компьютерного эксперимента, проведенного на Армкластере 94

Каменщиков Л .П., Карепова Е. Д., Шайдуров В. В. Моделирование распространения крупномасштабных волн в морях и океанах с параллельной реализацией 108

Васенин И. М., Петушкеев Б. Л. Опыт применения параллельных вычислений к проблематике шахтной вентиляции 116

Семенов И. В., Ахмедьянов И. Ф. Разработка параллельного алгоритма LU-SGS для решения многомерных задач вычислительной газодинамики 122

Есаулов Ан. О., Старченко А. В. Численный метод решения уравнений Навье-Стокса на многопроцессорных вычислительных системах 130

Панасенко Е. А., Старченко А. В. Численное решение некоторых обратных задач переноса примеси на многопроцессорных вычислительных системах 139

Малышенко В. В., Авзалов Д. Р. Решение задачи схлопывания каверны на поверхности жидкости в трехмерной постановке на кластере «СКИФ Cyberia» 149

Чаюк К. Г. Фрагментированный алгоритм LU-разложения и его параллельная реализация 157

Мельников Г. Н. Реализация алгоритмов цифровой фильтрации и масштабирования видеоизображений на специализированном векторном процессоре G80 162

Курносов М. Г. Оптимизация выполнения параллельных MPI-программ на мультикластерных вычислительных системах 168

Удалова Ю. В. Методы отладки функционально-потоковых параллельных программ 176

Биматов Д. В. О целесообразности изменения количества уровней в многоуровневой памяти 185

Фомин А. А., Фомина Л. Н. Полинейный рекуррентный метод решения СЛАУ с пятидиагональной матрицей 192

Есаулова О. А., Меркулова Н. Н. Метод регуляризации решения обратных задач теплопроводности 201

Нутерман Р. Б., Старченко А. В., Бакланов А. А. К вопросу о выборе турбулентной схемы замыкания для микромасштабных моделей городского атмосферного пограничного слоя 209

Шумилов Б. М., Эшаров Э. А. Рекуррентные сплайн-фильтры 218

Трунов А. А., Старченко А. В., Турчановский И. Ю., Шкляев В. А. Параллельная реализация алгоритма решения задачи о транспортировке пучка заряженных частиц 234

Научное издание

ЧЕТВЕРТАЯ СИБИРСКАЯ ШКОЛА-СЕМИНАР

ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 9—11 октября 2007 года

Редактор А. В. Старченко

Технический редактор Е. Н. Кабан

Подписано в печать 26.03.2008. Формат 60×84/16.
Бумага офсетная. Гарнитура Petersburg.
Печать трафатерная. Усл. печ. л. 14,18. Уч.-изд. л. 8,00.
Тираж 500 экз. Заказ 240.

Отпечатано ООО «Дельтаплан»
634041, г. Томск, ул. Тверская, 81

435-400  435-600