ИНСТИТУТ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ
РОССИЙСКОЙ АКАДЕМИИ НАУК

ГОРОБЕЦ АНДРЕЙ ВЛАДИМИРОВИЧ

МАСШТАБИРУЕМЫЕ ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ПОВЫШЕННОЙ ТОЧНОСТИ ДЛЯ ПРЯМОГО ЧИСЛЕННОГО МОДЕЛИРОВАНИЯ ЗАДАЧ ГАЗОВОЙ ДИНАМИКИ И АЭРОАКУСТИКИ

Специальность 05.13.18 Математическое моделирование, численные методы и комплексы программ

Диссертация на соискание ученой степени кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук Т.К. Козубская

Москва 2007

Оглавление

Введение

В настоящее время математическое моделирование активно входит в практику инженерных исследований и промышленного конструирования. Одной из наиболее актуальных и в то же время сложных областей применения математического моделирования является газовая динамика и аэроакустика.

Задачи, связанные с газовой динамикой, играют важную роль во многих научных и инженерных приложениях. Некоторые из них широко известны. В авиастроении это, например, моделирование внешнего газодинамического обтекания [1], струй смешения и так далее. Все большее значение приобретают задачи аэроакустики, исследование эффектов генерации и поглощения шума в авиационном двигателестроении. В настоящее время математическое моделирование находит применение в разработке звукопоглощающих конструкций реактивных двигателей [2]. Вычислительная газовая динамика также имеет широкое применение во многих других областях, в том числе и в инженерных целях, как, например, при разработке теплообменников и тепловых накопителей энергии, активных и пассивных систем солнечной энергии. Также следует упомянуть моделирование горения и течений с химическими реакциями, распространение загрязняющих веществ [3], прогнозирование погоды и многое другое. Так же в недавнее время появились такие приложения, как моделирование кровообращения [4] и процессов микробиологии, моделирование экологических проблем, на-

пример, влияния деятельности человека на глобальное потепление.

Как известно, прогресс в дальнейших исследованиях и улучшение используемых конструкций не может быть достигнуто без проведения экспериментов. Современный уровень развития техники и технологии и необходимый уровень оптимизации требуют постановки все более многочисленных и сложных экспериментов. Метод физического эксперимента становится все более сложным и дорогостоящим. В это же время развитие методов численного моделирования и вычислительной техники подготовили базис для формирования нового подхода к экспериментированию в газовой динамике, а именно методы вычислительного эксперимента. В различных отраслях промышленности численный эксперимент позволяет моделировать широкий спектр технологических процессов, а также явлений, связанных с эксплуатацией конечного изделия. В частности, в авиации численный эксперимент позволяет значительно сократить затраты, к примеру, на оптимизацию аэродинамических свойств летательного аппарата, поскольку вычислениями заменяется существенная часть дорогостоящих продувок конструкций в аэродинамических трубах, а также летных испытаний. В качестве примера можно привести применение методов вычислительной газовой динамики компанией Boeing [5], одним из мировых лидеров в производстве магистральных пассажирских воздушных судов. В настоящее время для разработки самолетов Boeing вычислительный эксперимент заменил большинство натурных экспериментов в аэродинамических трубах. В частности, методы вычислительной газовой динамики активно используются при разработке крыла, закрылков, фюзеляжа, салона, сочленения двигателя с крылом, обтекателя сочленения крыла с фюзеляжем, кормовой

части, стоек шасси и так далее.

Большой вклад в расширение возможностей вычислительного эксперимента внесли бурно развивающиеся многопроцессорные вычислительные системы. Существует несколько классов многопроцессорных систем. Наиболее распространенными архитектурами является системы с общей памятью SMP (Symmetric Multiprocessing - симметричное мультипроцессирование) и системы с распределенной памятью или, другими словами, массивно-параллельные системы MPP (Massively Parallel Processing). Первые состоят из нескольких однородных процессоров и массива общей памяти. Все процессоры имеют доступ к любой точке памяти с одинаковой скоростью. Вторые представляют собой множество вычислительных узлов, объединенных компьютерной сетью и имеющих один или несколько процессоров и локальную память, недоступную напрямую другим узлам. Соответственно этим классам существуют технологии параллельного программирования, существенно различающиеся между собой. Для систем с общей памятью это, например, интерфейс прикладного программирования OpenMP. Для систем с распределенной памятью самой распространенной технологией является интерфейс обмена данными MPI (Message Passing Interface). В данной работе рассматривается только технология параллельного программирования MPI, поскольку в настоящее время для задач параллельной вычислительной газовой динамики MPI имеет наиболее широкое применение. Это связано с тем, что системы с распределенной памятью, на которые ориентирована технология MPI, сами по себе имеют более широкое применение в данной области, чем системы с общей памятью. SMP системы имеют ряд ограничений: это существенное ограничение по числу

процессоров, намного более высокая стоимость и низкое соотношение цены и производительности. В то же время MMP системы намного превосходят по производительности системы с общей памятью, число процессоров может достигать десятков тысяч (например BlueGene и Marenostrum производства IBM или Jaguar производства Cray Inc). Так же широкое распространение MPP обусловлено простотой построения малобюджетного варианта такой системы из обычного офисного компьютерного оборудования, что позволяет многим исследовательским группам иметь собственный параллельный компьютер. Такой тип систем принято называть Beowulf кластер (http://www.beowulf.org). Помимо того, технология MPI универсальна - она также может эффективно применяться и на системах с общей памятью.

Быстрый рост производительности многопроцессорных вычислительных систем привел к новому этапу развития вычислительного эксперимента, а также к проблеме перехода на многопроцессорные системы. Этот переход связан с адаптацией существующих алгоритмов и последовательных комплексов программ, рассчитанных на однопроцессорный режим, к параллельным вычислениям, что является достаточно сложной задачей для многопроцессорным систем в целом, а для систем с распределенной памятью в особенности [6, 7]. К примеру, одной из проблем является балансировка загрузки, то есть обеспечение равномерной загрузки процессоров при параллельных вычислениях [8], а также минимизация межпроцессорного обмена данными, что особенно сложно в случае использования неструктурированных сеток и обширных пространственных шаблонов [9]. Метод геометрического параллелизма, наиболее широко применяемый в задачах

параллельной вычислительной газовой динамики и также использующийся в данной работе, предполагает разбиение расчетной области на множество подобластей, соответствующих процессорам. Каждый процессор производит вычисления для получения решения на узлах своей подобласти. В этом случае требуется минимизировать объем обмена данными и в тоже время как можно более равномерно распределить вычисления между процессорами, чтобы максимально сократить время вычислений. Существует множество последовательных комплексов программ, основанных на явных численных методах и реализующих эффективные численные алгоритмы, прошедших верификацию, но устаревших и неприменимых к актуальным современным задачам из-за ограничений производительности одного процессора. При этом, на разработку подобных комплексов программ в свое время требовала больших трудозатрат, и было бы нерационально просто отказываться от их использования. Таким образом, возникает проблема эффективного распараллеливания существующих последовательных кодов, разработанных без учета специфики параллельных вычислений. При этом, под эффективностью распараллеливания понимается не только эффективность вычислений, но и минимизация трудозатрат на разработку параллельной версии. Это сформировало одну из целей данной работы, а именно разработку и применение технологии распараллеливания последовательных кодов, эффективную как с точки зрения производительности, так и с точки зрения трудозатрат. Задача становится особенно сложной применительно к неструктурированным сеткам и обширным неструктурированным пространственным шаблонам повышенного порядка точности (которые могут включать в себя неизвестное заранее число узлов). Поэтому для при-

менения технологии распараллеливания был выбран комплекс программ Noisette, предназначенный для решения задач газовой динамики и аэроакустики, алгоритм которого описан в [10]. Данный комплекс программ как раз обладает этими осложняющими факторами: Noisette использует неструктурированные сетки и алгоритмы повышенного порядка точности со сложными шаблонами [11]. Аэроакустика, основная область применения Noisette, является сравнительно новым направлением в газовой динамике. Одно из типичных современных приложений аэроакустики - это снижение шума авиационных двигателей. Звукопоглощающие конструкции (ЗПК) резонансного типа широко распространены в авиационном строении для подавления шума турбореактивных двигателей. ЗПК представляет собой перфорированную панель, конфигурация и геометрические параметры которой существенным образом влияют на эффективность поглощения шума. Для оптимизации параметров ЗПК удобным инструментом может служить математическое моделирование. Хорошо отлаженная вычислительная среда, обеспечивающая расчеты ЗПК в различных конфигурациях, может рассматриваться как виртуальный экспериментальный стенд, легко адаптируемый к широкому диапазону допустимых геометрических параметров и амплитудно-частотных характеристик входного сигнала, и, соответственно, как эффективное средство в помощь физическому эксперименту при конструировании ЗПК. Детальное численное моделирование, к тому же, способствует глубокому пониманию физических механизмов, определяющих звукопоглощение. Математическое моделирование поглощения шума в ЗПК резонансного типа является типичной задачей нелинейной аэроакустики. Спецификой численного моделирования таких задач является на-

личие как линейных, так и нелинейных явлений, что требует применения схем высокого порядка точности. Сложность геометрии и большая разница в геометрических размерах элементов конструкций (например, соотношение размера канала и горла резонатора) требуют применения неструктурированных сеток. Так же для таких задач характерен большой перепад пространственных и временных масштабов наименьших и наибольших структур течения, что требует подробной пространственной дискретизации и большого числа шагов по времени. Все перечисленные факторы приводят к большим вычислительным затратам. Поэтому использование высокопроизводительных многопроцессорных вычислительных систем особенно актуально для такого типа задач. В данной работе приводятся два типа вычислительных экспериментов по ЗПК, а именно моделирование свойств ячейки ЗПК в импедансной трубе; а также моделирование потерь энергии звукового сигнала при его прохождении в дозвуковом течении в канале, облицованном перфорированными панелями. Задачи носят модельный характер, однако могут служить начальным приближением к прямому численному моделированию ЗПК. Другой проблемой, которая возникает при параллельных вычислениях, является необходимость обеспечения масштабируемости используемых алгоритмов на большое число процессоров. Как известно, эффективность параллельных вычислений начинает резко снижаться, когда число процессоров становится больше некоторого ограничения, свойственного данному алгоритму или размеру задачи. Это происходит в частности из-за того, что время, затрачиваемое на обмен данными, с ростом числа процессоров начинает превосходить время, затрачиваемое непосредственно на вычисления. Поэтому достичь высокой параллельной

эффективности представляется достаточно сложной задачей при большом числе процессоров. Ситуация особенно осложняется в случае моделирования несжимаемых течений: поскольку скорость звука в несжимаемых течениях равна бесконечности и возмущения из любой точки мгновенно влияют на всю расчетную область, требуется передача информации между всеми процессорами, а не только между соседями по декомпозиции, как в случае со сжимаемыми течениями. В то же время, необходимость расчетов с использованием сотен и тысяч процессоров вызвана, к примеру, вычислительной сложностью моделирования турбулентных течений. Большинство сложных и интересных с точки зрения приложений газовой динамики течений являются турбулентными. Для расчета турбулентных течений на основе прямого численного моделирования DNS (Direct Numerical Simulations) уравнений Навье-Стокса требуются особенно большие вычислительные затраты. Это обусловлено необходимостью очень подробной пространственной и временной дискретизации. К примеру, согласно широко известным оценкам, число узлов в трехмерной задаче пропорционально [12], где - число Рейнольдса. Даже при умеренных числах Рейнольдса вычислительная стоимость расчета может оказаться настолько большой, что под силу только самым мощным многопроцессорным системам. Поэтому для инженерных приложений в настоящее используются полуэмпирические модели турбулентности, например, осредненные уравнения Навье-Стокса RANS (Reynolds averaged Navier Stokes) [13] или моделирование крупных вихрей LES [14, 15], (Large Eddy Simulation), а так же метод отсоединенных вихрей DES (Detached-Eddy simulation) [16], сочетающий в себе и RANS и LES.Задачей подобных моделей турбулентности является предсказывание

осредненных физических величин турбулентного течения без нахождения решения для всех пространственных и временных масштабов течения.

В случае RANS оператор осреднения применяется к системе уравнений Навье-Стокса и мгновенные значения выражаются в виде суммы среднего значения и возмущения. Таким образом, получается система осредненных по времени уравнений Навье-Стокса, которая содержит дополнительные члены, решение для которых не может быть найдено без информации о возмущениях. Для замыкания системы уравнений RANS было предложено множество методов, которые имеют набор параметров, определяемых экспериментально. К сожалению, ни один из этих методов не может рассматриваться как точная модель для всех течений [17]. Обычно RANS используется для нахождения установившихся решений, в частности, среднего поля течения. И, как правило, RANS применяется в случаях, когда не требуется высокая точность результата, а необходимо лишь достаточно грубое качественное сравнение. Таким образом, RANS можно рассматривать как стационарные уравнения Навье-Стокса, дополненные нелинейными величинами и уравнениями.

Основная идея метода LES - находить решение только для крупномасштабных структур течения, а мелкие масштабы, которые не могут быть разрешены из-за ограничений вычислительных ресурсов, моделируются. Этот подход основан на пространственной фильтрации уравнений Навье-Стокса. Модель турбулентности используется для масштабов течения которые не вычисляются напрямую. Эта модель может быть намного проще и более универсальной, чем в случае RANS, но к сожалению, в случае LES также не существует единой модели для всех актуальных задач. Перед тем

как использовать модель, она должна быть верифицирована экспериментально для конкретной задачи, что часто бывает проблематично.

В связи со сложностями в получении всей необходимой информации, которая требуется для разработки и подтверждения модели турбулентности, часто модели сравниваются с результатами DNS, а не с результатами эксперимента. Это является одной из основных причин развития методов DNS, несмотря на пессимистичные прогнозы относительно вычислительной стоимости. Набор вычислительных экспериментов DNS позволяет построить базис для калибровки моделей RANS, LES и других, при этом точность DNS в модельных постановках с использованием подробной пространственной дискретизации порядка сотен миллионов узлов превосходит возможности физического эксперимента и измерительных приборов. Кроме того, с учетом бурного роста производительности вычислительных систем, DNS в скором времени может найти более широкое применение для инженерных задач. Но и в настоящее время DNS используется, например, для физических исследований, понимания сути явления турбулентности. [18, 19, 20, 21].

Применение параллельных технологий для моделирования несжимаемого течения более проблематично, по сравнению со сжимаемыми течениями. Это объясняется таким физическим свойством несжимаемой жидкости, как бесконечная скорость распространения возмущений. Уравнение Пуассона, к которому приводит уравнение неразрывности, соответствует этому физическому свойству: оператор Пуассона имеет бесконечную скорость распространения информации в пространстве (то есть на каждом шаге по времени требуется обмен данными между всеми процессорами,

что существенно сказывается на параллельной эффективности особенно при большом числе процессоров). Поэтому эффективное решение уравнения Пуассона на многопроцессорных системах является ключевой проблемой при моделировании несжимаемых течений. Основное внимание в части работы, посвященной DNS несжимаемых течений, уделено разработке масштабируемого метода для решения уравнения Пуассона.

Также следует отметить, что современные параллельные вычислительные системы с распределенной памятью существенно различаются между собой по производительности, числу процессоров, латентности сети и другим параметрам. Поэтому метод, который эффективен на одной многопроцессорной системе, может оказаться практически неприменимым на другой. Системы варьируются от малобюджетных кластеров на основе офисного компьютерного оборудования до суперкомпьютеров с высокопроизводительной сетью и тысячами процессоров. Первые имеют очень высокое соотношение производительности и цены и, благодаря своей низкой стоимости, широко используются. Но вторые имеют гораздо большую вычислительную мощность, столь необходимую для DNS и LES на подробных сетках. Наиболее существенными различиями между параллельными системами с распределенной памятью являются, во-первых, число процессоров и, во-вторых, производительность сети. Алгоритмы, которые работают эффективно на малобюджетном кластере, могут оказаться неэффективными на суперкомпьютере из-за проблем масштабирования на большое число процессоров. И наоборот, эффективные на суперкомпьютерах алгоритмы могут иметь неудовлетворительную эффективность на малобюджетном кластере из-за низкой производительности сети, в частности, зна-

чительно большей латентности. Поэтому требование эффективности алгоритма для моделирования несжимаемых течений на различных типах параллельных систем еще более усложняет задачу. Алгоритм также должен иметь низкую вычислительную стоимость и широкую область применимости. Большинство из существующих алгоритмов для несжимаемых течений не удовлетворяет этой совокупности требований. Например, многосеточные методы [22, 23, 24] - одно из наиболее мощных средств для последовательных вычислений. В них используется иерархический набор сеток, самая грубая из которых имеет сильно сокращенное число узлов. Это позволяет эффективно переносить информацию между удаленными частями расчетной области, что как раз необходимо для уравнения Пуассона. Многосеточный метод выполняет большинство итераций на грубых сетках, для которых вычислительные затраты очень небольшие. Но это в случае однопроцессорного режима. В параллельном режиме будет доминировать латентность сети, которая приведет к значительно большим затратам времени чем сами вычисления. Поэтому метод эффективен только на системах с низкой латентностью сети и сравнительно небольшим числом процессоров [25]. Методы Крыловского типа, такие как метод сопряженных градиентов или обобщенный метод минимальных невязок [26] неплохо поддаются распараллеливанию, хотя требуют на каждой итерации несколько обменов данными. Но, во-первых, их эффективность сильно зависит от используемого предобуславливателя, и, во-вторых, для уравнения Пуассона сложно добиться хорошей сходимости. В настоящее время в этой области ведутся активные исследования [27, 28, 29, 30]. В частности, в [27] предлагается алгоритм для моделирования течения вязкой несжи-

маемой жидкости, в котором для решения уравнения Пуассона используется метод сопряженных градиентов с предобуславливателем MICCG(0). Но данный алгоритм, во-первых, более подходит для стационарных задач и, во-вторых, эффективен только на сравнительно небольшом числе процессоров. Метод быстрого преобразования Фурье (БПФ) [31], примененный сразу по нескольким осям имеет низкую вычислительную стоимость порядка $O(N\log(N))$, но он также имеет существенные ограничения. Применение БПФ требует равномерного шага сетки и исключает возможность постановки препятствий в потоке. Поэтому такой метод применим только для простейших модельных постановок, как, например, течение в канале (в этом случае БПФ применяется по двум осям, по которым используются периодические граничные условия) или каноническое турбулентное течение (БПФ по трем осям, все граничные условия периодические) [19]. Таким образом, сформировалась еще одна цель данной работы, а именно построение гибкого и масштабируемого метода для решения уравнения Пуассона, эффективного как на малобюджетных кластерах, так и на суперкомпьютерах. В качестве исходного базиса были взяты работы [32, 33, 34]. В частности, в [34] предложен прямой алгоритм для дискретного уравнения Пуассона высокого порядка аппроксимации. Он основан на сочетании БПФ (Быстрое Преобразование Фурье) метода и метода дополнений Шура и имеет хорошую производительность на малобюджетных кластерах с относительно небольшим (20-30) числом процессоров и большой латентностью сети. Этому методу необходим только один обмен данными для решения уравнения Пуассона. Следует отметить, что БПФ используется только по одному направлению, что существенно расширяет область применимости.

А именно, возможно сгущение шага сетки по двум осям для разрешения пограничных слоев, а также возможно помещение в течение препятствий. Но метод также имеет специфические ограничения, в частности, связанные с размером требуемой памяти и объемом обмена данными, которые растут достаточно быстро как с числом процессоров, так и с числом узлов сетки. Это существенно ограничивает масштабирование, особенно для схем высокого порядка аппроксимации. Поэтому метод, имея хорошую производительность на небольших малобюджетных кластерах, с ростом числа процессоров быстро теряет эффективность, из-за чего практически не применим на суперкомпьютерах. Целью данной работы является расширение возможностей метода [34], условно обозначаемого далее как метод Фурье-Шура, для применения на суперкомпьютерах, используя сотни и тысячи процессоров. Значительное лучшее масштабирование метода основано на сочетании прямого метода Фурье-Шура с итерационным методом на основе подпространств Крылова, а именно методом сопряженных градиентов. Этот метод будет далее условно обозначен как метод Крылова-Фурье-Шура. Новый хорошо масштабируемый и гибкий метод, описанный в данной работе, может эффективно использоваться как на системах с высокопроизводительной сетью и большим числом процессоров, так и на малобюджетных системах с сетью большой латентности. Продемонстрирована высокая параллельная эффективность метода с использованием до тысячи процессоров суперкомпьютера Марениструм Берселонского Суперкомпьютерного Центра. Описано применение метода для крупномасштабного DNS с использованием сетки с числом узлов более 108. Данный расчет является на момент завершения самым крупным в мире для данного класса задач.

При этом используется спектрально-согласованная разностная схема 4-го порядка, описанная в [35, 36, 37].

Далее приводится краткое содержание работы по главам. **Первая глава** диссертации посвящена проблеме распараллеливания последовательного комплекса программ для расчета задач газовой динамики и аэроакустики, основанного на явных высокоточных алгоритмах с использованием неструктурированных сеток. Технология распараллеливания продемонстрирована на примере комплекса программ Noisette, который обладает основными осложняющими факторами, такими как повышенный порядок аппроксимации и обширный неструктурированный пространственный шаблон разностной схемы. Предлагаются несколько основных идей для существенного увеличения параллельной производительности. Представленная технология распараллеливания позволяет разработчикам последовательного комплекса программ с минимальными трудозатратами получить параллельную версию, обладающую высокой параллельной эффективностью. При этом, от разработчиков не требуется глубоких знаний в области параллельных вычислений.

Во второй главе приводятся основные вычислительные эксперименты по моделированию звукопоглощающих конструкций, выполненные с использованием параллельной версии Noisette, разработанной по технологии, описанной в первой главе. Группа 2D и 3D модельных задач воспроизводит условия физического эксперимента в импедансной трубе и в канале с вмонтированными в стенки резонаторами. Эти задачи посвящены изучению звукопоглощающих свойств резонатора и механизма потери акустической энергии.

Третья глава посвящена эффективному решению уравнения Пуассона при моделировании несжимаемых течений на параллельных системах различных масштабов. В этой главе предложен метод, основанный на сочетании метода Фурье-Шура с итерационным методом крыловского типа. Новый метод Крылова-Фурье-Шура имеет такие важные преимущества как хорошая масштабируемость и гибкость. В данной главе показан способ адаптации метода к различному числу процессоров и к сетям различной латентности. Продемонстрирована высокая параллельная эффективность как на малобюджетных кластерах, так и на суперкомпьютере Марenострум Барселонского суперкомпьютерного центра.

В четвертой главе приводятся описание крупномасштабного прямого численного моделирования, а именно DNS турбулентного течения при естественной конвекции от воздействия выталкивающих сил. Расчет выполнен с использованием численного метода, в основе которого описанный в данной работе метод Крылова-Фурье-Шура. Рассматривается течение несжимаемой жидкости в закрытой каверне с разными температурами на двух противоположных вертикальных стенках.

В заключении приведены основные результаты диссертации.

Цели и задачи диссертационной работы:

1. Разработка эффективной технологии распараллеливания последовательных комплексов программ для решения задач газовой и аэроакустики на основе явных алгоритмов повышенного порядка точности и неструктурированных сеток.

2. Применение технологии распараллеливания для разработки парал-

лельного комплекса программ на основе последовательного кода.

3. Проведение при помощи разработанного параллельного программного комплекса расчетов двумерных и трехмерных задач газовой динамики и аэроакустики.

4. Разработка на основе ранее известного метода Фурье-Шура для решения уравнения Пуассона на малобюджетных параллельных системах с небольшим числом процессоров нового масштабируемого метода повышенного порядка точности, который может эффективно применяться на суперкомпьютерах с использованием до тысячи процессоров.

5. Проведение при помощи нового метода для решения уравнения Пуассона крупномасштабного прямого численного моделирования. Достижение высокой эффективности на числе процессоров не менее 512 и обеспечить возможность использовать сетки с числом узлов не менее 10^8 при условии применения схемы повышенного порядка аппроксимации (не ниже 4-го).

Достоверность результатов

Разработанный параллельный комплекс программ надежно верифицирован путем сравнения на совпадение результатов параллельной и исходной последовательной версий. При этом исходная последовательная версия была ранее подробно верифицирована на серии широко известных тестовых задач. Эффективность параллельных вычислений подтверждается серией тестов на параллельную производительность и эффективность, выполненных на различных многопроцессорных системах. Масштабируемый па-

раллельный метод Крылова-Фурье-Шура для уравнения Пуассона обеспечивает требуемую заданную точность решения, которая автоматически контролируется в расчетах путем явного вычисления невязки. При этом данный метод применяется в составе комплекса программ, который верифицирован ранее на основе широко известного метода MMS (Method of Manufactured Solutions) [38], а также путем сравнения с результатами других авторов. Параллельная эффективность подтверждается серией тестов, выполненных на различных вычислительных системах при варьировании числа процессоров в широком диапазоне до 1024 включительно.

Основные положения диссертации, выносимые на защиту

1. Технология распараллеливания комплекса программ, основанного на явных высокоточных алгоритмах, использующих обширный пространственный шаблон и неструктурированные сетки.

2. Параллельный комплекс программ SuperNoisette 2D/3D для расчетов двухмерных и трехмерных задач газовой динамики и аэроакустики.

3. Результаты серии вычислительных экспериментов по звукопоглощающим конструкциям (ЗПК)

4. Масштабируемый метод Крылова-Фурье-Шура для решения уравнения Пуассона на различных параллельных системах от малобюджетных кластеров до суперкомпьютеров

5. Крупномасштабное прямое численное моделирование турбулентного течения при естественной конвекции в закрытой каверне

Публикации автора по теме диссертации

1. I.V.Abalakin, A.V.Gorobets, T.K.Kozubskaya, A.K.Mironov, Simulation of Acoustic Fields in Resonator-Type Problems Using Unstructured Meshes, AIAA 2006-2519 Paper (2006).

2. А.В.Горобец, Т.К.Козубская, Технология распараллеливания явных высокоточных алгоритмов вычислительной газовой динамики и аэроакустики на неструктурированных сетках. - Математическое моделирование, т.19, № 2, (2007), стр. 68-86.

3. И.В.Абалакин, А.В.Горобец, Т.К.Козубская, Вычислительные эксперименты по звукопоглощающим конструкциям. - Математическое моделирование, т. 19, № 8, (2007), стр. 15-21.

4. А.В.Горобец, Масштабируемый алгоритм для моделирования несжимаемых течений на параллельных системах. - Математическое моделирование, т. 19, № ,10, (2007), стр 105-128.

5. A. Gorobets, F. X. Trias, M. Soria and A. Oliva, A scalable Krylov-Schur-Fourier Decomposition for the efficient solution of high-order Poisson equation on parallel systems from small clusters to supercomputers. - Computers&fluids (готовится к публикации)

6. A.V. Gorobets, I.V. Abalakin, T.K. Kozubskaya, Technology of parallelization for 2D and 3D CFD/CAA codes based on high-accuracy explicit methods on unstructured meshes - In Parallel Computational Fluid Dynamics. Elsevier, 2007.

7. F. X. Trias, A.V. Gorobets, M. Soria and A. Oliva, DNS of natural convection flows on MareNostrum supercomputer - In Parallel Computational Fluid Dynamics. Elsevier, 2007.

Автор выражает благодарность своему научному руководителю Татьяне Константиновне Козубской и Манелу Сориа за постоянное внимание, помощь и поддержку; Илье Владимировичу Абалакину и Хавьеру Триасу за плодотворную совместную работу, многочисленные полезные обсуждения и советы; Борису Николаевичу Четверушкину и Ассенси Олива за всестороннюю поддержку данной работы.

1. Технология распараллеливания высокоточных алгоритмов для моделирования вязких сжимаемых течений

1.1. Математическая основа комплекса программ NOISETTE

1.1.1. Математические модели

В комплексе программ NOISETTE реализован класс моделей аэроакустики, построенных на основе уравнений Навье-Стокса (Н-С) [39]. В данное семейство входят три основные модели: первая- линейная, описываемая линеаризованными уравнениями Н-С; вторая - нелинейная для пульсационных компонент течения, описываемая различными формами NLDE (NonLinear Disturbance Equations) уравнений [40]; третья - нелинейная для всего течения, описываемая полными уравнениями Н-С. В NOISETTE предусмотрено также введение членов, реализующих действие молекулярной вязкости и теплопроводности, в рамках моделей на основе уравнений Навье-Стокса и их линейных аналогов. В общем виде все модели семейства Н-С могут быть представлены в форме матричных уравнений вида (1.1-1.3) .

Полная система уравнений Навье-Стокса

$$\frac{\partial \mathbf{Q}}{\partial t}+\frac{\partial \mathbf{F}(\mathbf{Q})}{\partial x}+\frac{\partial \mathbf{G}(\mathbf{Q})}{\partial y}+\frac{\partial \mathbf{H}(\mathbf{Q})}{\partial z}=\frac{1}{Re}\left(\frac{\partial \mathbf{F}_\nu(\mathbf{Q})}{\partial x}+\frac{\partial \mathbf{G}_\nu(\mathbf{Q})}{\partial y}+\frac{\partial \mathbf{H}_\nu(\mathbf{Q})}{\partial z}\right), \tag{1.1}$$

где $\mathbf{Q} = (\rho, u, v, w, E)^T$ - вектор полных консервативных переменных, $\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{F}_\nu, \mathbf{G}_\nu, \mathbf{H}_\nu$ – конвективные и вязкие потоки соответственно.

Система уравнений NLDE

$$\begin{gathered}\frac{\partial \mathbf{Q}'}{\partial t}+\frac{\partial\left(\mathbf{F}(\overline{\mathbf{Q}}+\mathbf{Q}')-\mathbf{F}(\overline{\mathbf{Q}})\right)}{\partial x}+ \\ \frac{\partial\left(\mathbf{G}(\overline{\mathbf{Q}}+\mathbf{Q}')-\mathbf{G}(\overline{\mathbf{Q}})\right)}{\partial y}+\frac{\partial\left(\mathbf{H}(\overline{\mathbf{Q}}+\mathbf{Q}')-\mathbf{H}(\overline{\mathbf{Q}})\right)}{\partial z}= \\ \frac{1}{Re}\left(\frac{\partial \mathbf{F}_\nu(\mathbf{Q}')}{\partial x}+\frac{\partial \mathbf{G}_\nu(\mathbf{Q}')}{\partial y}+\frac{\partial \mathbf{H}_\nu(\mathbf{Q}')}{\partial z}\right)+\overline{\mathbf{S}},\end{gathered} \tag{1.2}$$

где $\overline{\mathbf{S}} = -\left[\frac{\partial \overline{\mathbf{Q}}}{\partial t}+\frac{\partial \mathbf{F}(\overline{\mathbf{Q}})}{\partial x}+\frac{\partial \mathbf{G}(\overline{\mathbf{Q}})}{\partial y}+\frac{\partial \mathbf{H}(\overline{\mathbf{Q}})}{\partial z}-\frac{\partial \mathbf{F}_\nu(\overline{\mathbf{Q}})}{\partial x}-\frac{\partial \mathbf{G}_\nu(\overline{\mathbf{Q}})}{\partial y}-\frac{\partial \mathbf{H}_\nu(\overline{\mathbf{Q}})}{\partial z}\right]$,
а вектор $\mathbf{Q}$ разложен на среднее и пульсационное составляющие $\mathbf{Q} = \overline{\mathbf{Q}}+\mathbf{Q}'$ [41].

Система линеаризованных уравнений Н-С

$$\frac{\partial \mathbf{Q}'}{\partial t}+\frac{\partial \overline{\mathbf{A}_x}\mathbf{Q}'}{\partial x}+\frac{\partial \overline{\mathbf{A}_y}\mathbf{Q}'}{\partial y}+\frac{\partial \overline{\mathbf{A}_z}\mathbf{Q}'}{\partial z}=\frac{\partial \mathbf{F}_\nu(\mathbf{Q}')}{\partial x}+\frac{\partial \mathbf{G}_\nu(\mathbf{Q}')}{\partial y}+\frac{\partial \mathbf{H}_\nu(\mathbf{Q}')}{\partial z}+\overline{\mathbf{S}}, \tag{1.3}$$

где $\overline{\mathbf{A}_x} = \mathbf{A}_x\left(\overline{\mathbf{Q}}\right)$, $\overline{\mathbf{A}_y} = \mathbf{A}_y\left(\overline{\mathbf{Q}}\right)$ и $\overline{\mathbf{A}_z} = \mathbf{A}_z\left(\overline{\mathbf{Q}}\right)$ являются стандартными матрицами Якобиана.

Каждая система уравнений (1.1-1.3) замыкается соответствующим уравнением состояния идеального газа на основе закона $p = \rho\varepsilon(\gamma - 1)$, где γ - показатель адиабаты. Более подробно описание моделей приводится в [41]. Следует отметить, что базовые численные алгоритмы, используемые в комплексе NOISETTE, разрабатывались в виде, применимом для всех моделей данного класса.

1.1.2. Базовая численная схема

В рамках конечно-объемного подхода схема для численного решения какой-либо из систем уравнений (1.1-1.3) может быть представлена в виде полудискретной аппроксимации:

$$\frac{\partial \mathbf{Q}_i}{\partial t} + \frac{1}{|C_i|} \sum_{j \in \Omega_i} \Phi_{ij} = 0 \tag{1.4}$$

где Q_i - основная газодинамическая переменная модели, Φ_{ij} - поток через грань контрольного объема вокруг узла i между сеточными узлами i и j, Ω_i - множество узлов, соседних к узлу i, $|C_i|$ - площадь контрольного объема вокруг узла i. Типы контрольных объемов, используемых в NOISETTE, представлены на рисунке 1.1. При таком представлении уравнение (1.5) является обыкновенным дифференциальным и для его интегрирования в NOISETTE используется явный метод Рунге-Кутты четвертого порядка точности по времени для нелинейных моделей и линейный аналог метода Рунге-Кутты произвольно высокого порядка точности для линейных моделей.

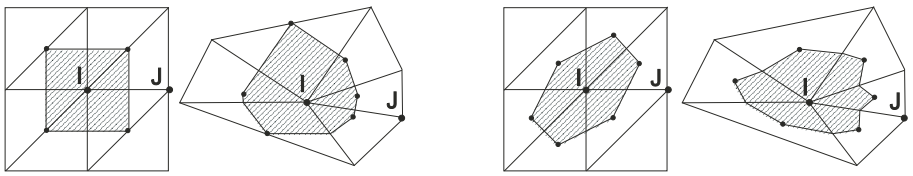

Рис. 1.1. Разные типы используемых контрольных объемов (барицентрические - слева, медианные - справа) для структурированных и неструктурированных треугольных сеток

Точность аппроксимации по пространству определяется точностью определения потока Φ_{ij} через грань контрольной ячейки.

1.1.3. Вычисление потока через грань контрольного объема

Более подробно со схемой высокого порядка точности, реализованной в комплексе NOISETTE, можно ознакомиться в [42], [43],[44]. Здесь следует отметить, что для аппроксимации потока через грань контрольного объема с повышенным порядком точности (до четвертого порядка включительно) используется базовый шаблон, представленный на рисунке 1.2. Базовый шаблон в двухмерном случае состоит из 6 сеточных узлов {i, m, n, j, r, s}, являющихся вершинами "противопоточных"треугольников. В трехмерном случае базовый шаблон состоит из 8 сеточных узлов {i, l, m, n, j, q, r, s}, являющихся вершинами "противопоточных"тетраэдров.

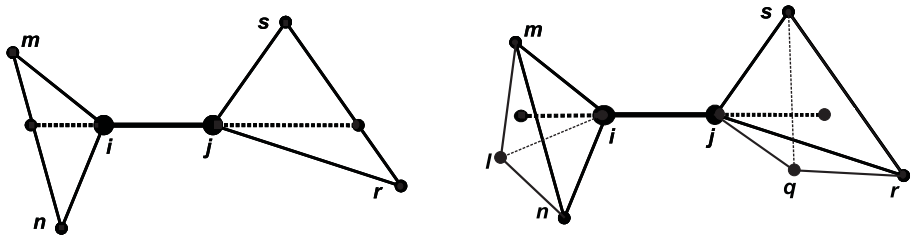

Рис. 1.2. Базовый шаблон пространственной аппроксимации: 2D слева, 3D - справа

Аппроксимация потока с высоким порядком точности (до шестого включительно) реализуется на расширенном шаблоне, представленном для двухмерного случая на рисунке 1.3. Помимо точек базового шаблона он включает в себя все узлы, соседние к узлам {i, m, n, j, s, r} в 2D случае и к узлам {i, l, m, n, j, q, r, s} в 3D случае. Точное количество точек расширенного шаблона заранее неизвестно, причем для разных граней ij оно, вообще говоря, разное. Появление в расширенном шаблоне дополнительных узлов связано с необходимостью вычисления во всех узлах базового

шаблона так называемых узловых градиентов $\nabla\mathbf{F}$,$\nabla\mathbf{G}$ и в 3D случае $\nabla\mathbf{H}$, которые определяются путем суммирования по всем треугольникам (тетраэдрам в 3D случае), содержащим "базовый"узел в качестве вершины. Например, узловой градиент $\nabla\mathbf{F}$ в точке i вычисляется по формуле

$$(\nabla\mathbf{F})_i = \frac{1}{|C_i|}\sum_{j\in\Omega_i}\left(\frac{|C_{ij}|}{6}\sum_{k\in T_j}\mathbf{F}_k\varphi_k\right) \tag{1.5}$$

где $\sum_{k\in T_j}\mathbf{F}_k\varphi_k$ - конечно-объемный градиент на тетраэдре (треугольнике в 2D случае) T_j, а C_{ij} - объем кусочка j-го тетраэдра (площадь кусочка j-го треугольника в 2D случае) из опирающихся на узел i, общего с контрольным объемом узла i.

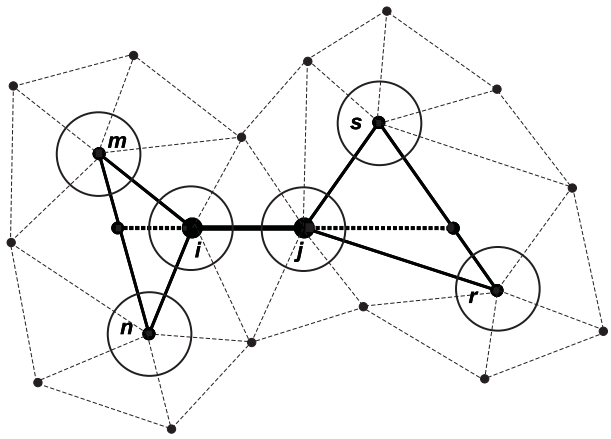

Рис. 1.3. Расширенный шаблон пространственной аппроксимации в двухмерном случае

1.2. Адаптация к параллельным вычислениям

Задача распараллеливания - это адаптация существующего последовательного комплекса программ для применения на многопроцессорных вычислительных системах. Распараллеливание комплексов последовательных программ должно удовлетворять ряду требований. Целесообразность

разработки параллельных программ на основе имеющихся последовательных требует жесткого ограничения на трудозатраты, а именно трудозатраты на распараллеливание должны быть существенно меньше трудозатрат на разработку и тестирование нового комплекса параллельных программ, реализующего те же численные алгоритмы. Естественным требованием является также требование максимизации эффективности параллельных вычислений. При этом, эффективность не должна быть существенно меньше той, которая могла бы быть достигнута при разработке аналогичного комплекса программ, при условии разработки его изначально как параллельного. Отличия параллельной версии от последовательной должны быть минимизированы, чтобы не создавать сложностей в работе с программой пользователям, не являющихся специалистами в области параллельных вычислений. С этой целью предлагается дополнительные параллельные программы-утилиты встраивать в комплекс программ на основе принципа модульности. Работу по распараллеливанию предлагается разделять на следующие основные этапы:

1. подготовка инфраструктуры (т.е. дополнительных программных модулей и внешних программ), которая будет обеспечивать: автоматическую подготовку сетки для параллельных расчетов; построение схемы пересылки данных; параллельный вывод информации и сборку данных;

2. модификация кода: реализация в коде пакета дополнительных структур данных; модификация расчетных циклов; реализация функций обмена данными; внесение других необходимых изменений;

3. Верификация параллельного кода, а именно тестирование параллельной версии на совпадение результатов вычислений с результатами исходной, последовательной, версии.

Далее в работе подробно рассматриваются все этапы распараллеливания на примере разработки параллельной версии комплекса программ NOISETTE.

1.3. Инфраструктура для параллельных вычислений

1.3.1. Подготовка сетки для параллельных расчетов

Для параллельной работы процессоров при использовании метода геометрического параллелизма необходима декомпозиция сетки на подобласти и определение узлов, участвующих в пересылке. Каждому процессору соответствует своя подобласть. Для расчетов по подобласти процессору требуются данные из приграничных узлов соседних подобластей, т.к. шаблон разностной схемы затрагивает эти узлы. Таким образом, подготовка сетки состоит из двух этапов: разбиения сетки и определение узлов, участвующих в пересылке. Разбиение сетки - это определение для каждого узла сетки, к какой подобласти он принадлежит. Разбиение должно быть максимально сбалансированным, чтобы на каждый процессор приходилась одинаковая вычислительная нагрузка. В противном случае будут возникать простои процессоров, что приведет к падению производительности. Для уменьшения времени на межпроцессорный обмен данными, требуется минимизировать число узлов, участвующих в пересылке. Количество таких узлов прямо пропорционально протяженности границ, разделяющих об-

ласть на подобласти. Таким образом, длина границ между подобластями должна быть минимизирована. Задача оптимального разбиения является отдельной достаточно сложной проблемой и в работе не рассматривается. Для разбиения сетки могут использоваться внешние средства декомпозиции, например, широко известное программное обеспечение Metis. В данном случае использовалась программа, выполненная Сергеем Болдыревым [45]. Эта программа выполняет разбиение сетки с вышеуказанной оптимизацией.

1.3.2. Построение и оптимизация схемы пересылки данных

Для расчета по подобласти требуются значения некоторых величин в узлах, принадлежащих соседним подобластям, причем структура пересылки зависит от используемой разностной схемы. Перед началом вычислений нужно определить для каждой подобласти, какие узлы из соседних подобластей понадобятся для расчета. Также надо определить какие узлы подобласти понадобятся соседним подобластям. Для определения узлов, участвующих в пересылке, используется представление неструктурированной сетки в виде графа. Если разностная схема использует обширный шаблон, то потребуются не только узлы, лежащие на границе между подобластями, но и узлы, находящиеся рядом с границей на некотором расстоянии. Узел принадлежит границе между подобластями, если он имеет соединение ребром с узлом из другой подобласти. Под расстоянием до границы понимается длина кратчайшего пути от узла подобласти до узла, принадлежащего границе между подобластями. Длина пути равна количеству ребер, составляющих путь. В первую очередь, по шаблону схемы

определяется расстояние от границы между подобластями, в пределах которого узлы участвуют в обмене данными. Оптимизация схемы пересылки заключается в минимизации этого расстояния. Шаблон схемы высокого порядка состоит из большого числа узлов, но не обязательно все узлы шаблона должны участвовать в обмене данными. Не должны участвовать в пересылке узлы, которые используются в шаблоне только в групповых операциях, таких как суммирование, поиск минимума, максимума, среднего и т.д. Например, в случае схемы, реализованной в NOISETTE, часть узлов в шаблоне используется только для вычисления узловых градиентов. Выберем узел, составляющий один из "противопоточных"треугольников ребра, соединяющего узлы из разных подобластей. Этот узел участвует в обмене данными, то есть значения из этого узла используется для вычислений соседней подобластью. Чтобы вычислить узловой градиент в этом узле, необходимо просуммировать определенные значения из соседних узлов. Поэтому и узлы, соседние с выбранным узлом должны были бы также участвовать в пересылке. Но на самом деле нам не требуется передавать соседям значения из этих узлов, а достаточно передать лишь само значение суммы, которое соответствует выбранному узлу. Таким образом, требуемое расстояние уменьшается на единицу, и из пересылки исключается большое количество узлов и существенно сокращается объем обмена данными. Для примера на Рис. 1.4 показана неструктурированная двухмерная сетка на треугольниках. Белой линией обозначена граница, разделяющая подобласти. Штриховкой обозначена область, в которой находятся узлы, участвующие в пересылке. Стрелками показано направление пересылки. В шаблоне схемы пунктирной линией очерчены узлы, необходимые для

вычисления узловых градиентов в точках шаблона, очерченных толстой черной линией. Узлы, очерченные пунктиром, в пересылке не участвуют. Рассмотрим шаблон разностной схемы, используемой в NOISETTE (Рис. 1.4). Чтобы определить узлы для пересылки, рассмотрим ребро, соединяющее узлы из разных подобластей. Например, в случае NOISETTE, для расчета потока через ребро требуются соседние узлы, составляющие "противопоточные"треугольники с каждой стороны ребра. Таким образом, из соседней подобласти требуются узлы, удаленные от границы на расстояние 1. Для расчета узловых градиентов в этих узлах требуются соседние с ними узлы. То есть требуются узлы, удаленные от границы на расстояние 2. Но т.к. эти узлы используются только в групповой операции - суммировании, то они не включаются в пересылку, а передается только сумма, соответствующая узлам, удаленным от границы на расстоянии до 1. Таким образом, в обмене данными участвуют только узлы, удаленные от границы между подобластями на расстояние до 1 включительно, то есть узлы, принадлежащие границе между подобластями и все соседние с ними узлы. Для определения узлов, участвующих в пересылке, была создана отдельная программа. Преимущество реализации построения пересылки в виде отдельной программы, помимо того что не затрагивается исходный код вычислительной части комплекса программ, состоит в том, что эту программу без каких либо изменений можно использовать в составе других параллельных пакетов. В программе используется представление сетки в виде графа и рекурсивный алгоритм для маркировки граничных узлов, удаленных от границы между подобластями до требуемого расстояния. Эта программа получает на вход файл разбиения - результат работы программы разби-

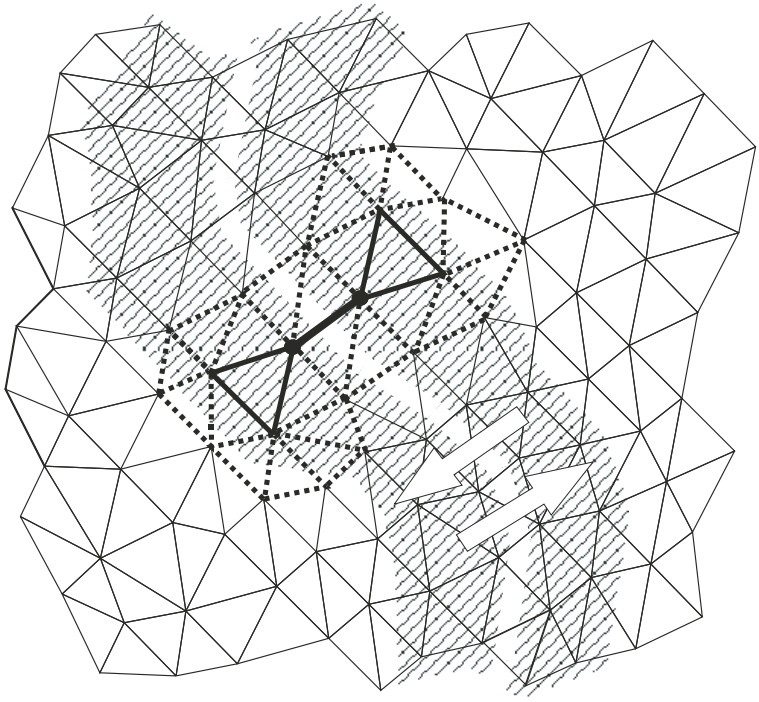

Рис. 1.4. Схема обмена данными в NOISETTE

ения, описанной в предыдущем пункте, а также расстояние до границы между подобластями, в пределах которого узлы участвуют в пересылке. На выходе - файл, содержащий для каждого узла список процессоров, которые используют этот узел. Этот файл далее используется параллельной версией исследовательского пакета Noisette, и будет далее называться коммуникационным файлом.

Файл имеет простейшую структуру, показанную на Рис. 1.5. Для каждого узла сетки последовательно записаны группы чисел: Первое число NP_j - число процессоров, использующих j-й узел, $j = 1, .., N$, N - число узлов в сетке. После этого следуют $NP_j + 1$ чисел. Первое из них - номер подобласти, которой принадлежит узел. Далее идут номера соседних подобластей, которые используют этот узел.

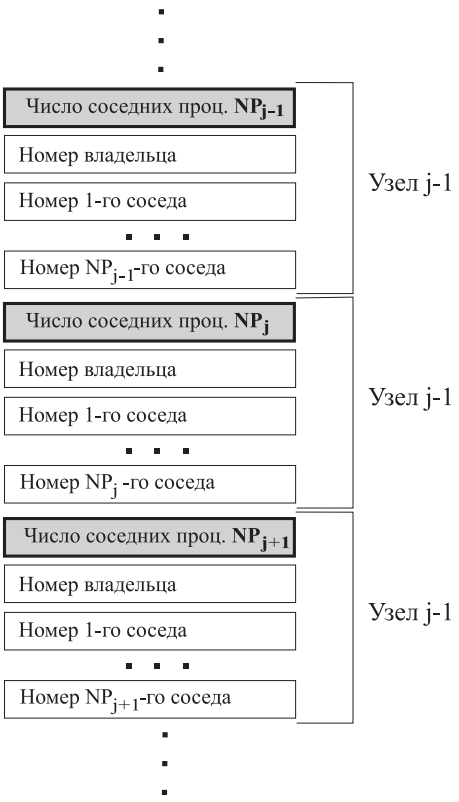

Рис. 1.5. Структура коммуникационного файла

1.3.3. Параллельный вывод результата и сборка данных

В исходной однопроцессорной версии Noisette результаты вычислений выводятся в набор файлов различных форматов: бинарные, текстовые, файлы Tecplot. В параллельной версии каждый процесс выводит в файлы данные только для своей подобласти. Поэтому имена файлов имеют формат: <имя файла в исходной версии>.<номер процессора> Таким образом, вывод файлов происходит корректно как на системах с раздельной файловой системой, так и на системах с общей сетевой файловой системой. Каждому файлу результата в исходной версии соответствует набор файлов в параллельной версии. Число этих файлов равно числу используемых процессоров. Требуется сборка файлов результата, соответствующих подобластям, в один файл с данными по всей области, эквивалентный фай-

лу исходной версии. Для этой цели была создана специальная программа сборки результата, а также были внесены некоторые изменения в функции записи файлов в Noisette.

Чтобы упростить сборку, в файлы результатов добавляется информация о позиции записи в собранном файле. Если в файле для каждого узла хранится набор каких-то значений, то к этому набору добавляется еще одно число - глобальный номер этого узла.

Программа сборки результата принимает на вход файлы, полученные со всех процессоров. Результат работы программы сборки - набор файлов, эквивалентный тому, который был бы получен при расчетах на одном процессоре. Программа работает следующим образом: последовательно обрабатывается каждая группа файлов, соответствующая одному файлу результата. Данные из файлов загружаются в память и записываются в объединенный файл в порядке следования узлов в глобальной нумерации.

Сборка требуется также для восстановления расчета. Например, при расчетах на N процессорах был записан промежуточный результат. Требуется продолжить расчет, но уже на M процессорах. Для этого необходима сборка файлов промежуточного результата. При запуске программы на M процессорах, данные для восстановления счета считываются из собранных файлов, соответствующих всей области. При этом каждый процессор считывает свой фрагмент из общего файла.

1.4. Модификация вычислительной части пакета программ

1.4.1. Наложение вычислений

На этапе обработки сетки внешней программой или дополнительным программным модулем определена принадлежность узлов подобластям. На основе этой информации необходимо определить принадлежность подобластям других элементов сетки. В двухмерном случае эти элементы - ребра и треугольники, в трехмерном - ребра, поверхностные треугольники и тетраэдры. Для повышения эффективности распараллеливания, принадлежность в двухмерном определяется следующим образом: каждый из этих элементов принадлежит подобласти, если хотя бы один его узел принадлежит подобласти.

Таким образом, ребро, треугольник или тетраэдр могут принадлежать одновременно более чем одной подобласти. Несмотря на то, что это приводит к некоторому наложению вычислений, т.е. более одного процессора производят вычисления по одному и тому же элементу, это избавляет от необходимости пересылки большого количества различных величин. Так, полностью отпадает необходимость пересылки данных, соответствующих тетраэдрам, треугольниками и ребрами. В зависимости от обширности шаблона, принадлежность ребер и треугольников может определяться и с большим наложением вычислений. Большинство вычислительных циклов по узлам в параллельной версии NOISETTE также включают узлы из соседних подобластей, которые используются для вычислений по данной подобласти, и это наиболее существенно сокращает объем пересылки.

Если значение какой-то величины в узле вычисляется в цикле по уз-

лам, то гораздо эффективнее расширить цикл по узлам, включив в него узлы из соседних подобластей, участвующие в обмене данными. То есть процессор, вместо того, чтобы запрашивать данные по этим узлам у соседнего процессора, сам производит вычисления и находит необходимые значения в узлах из соседних подобластей. Таким образом, наложение вычислений происходит как по ребрам, треугольникам, и тетраэдрам в трехмерном случае, так и по узлам, многократно сокращая объем пересылки.

Рассмотрим подробнее наложение вычислений по узлам. Пусть N_T - число узлов, участвующих в пересылке, которое совпадает с числом узлов, участвующих в дополнительных вычислениях. N - общее число узлов в сетке. Объем дополнительных вычислений при условии $N_T << N$ незначителен по сравнению с объемом вычислений по всей подобласти, а по затрате времени может быть значительно выгоднее пересылки данных, от которой он избавляет. Кроме существенного сокращения объемов пересылки, такой подход значительно упрощает схему обмена данными и всю процедуру распараллеливания. Также минимизируются отличия между параллельной и исходной версиями.

Пусть τ_c - время вычислений, затрачиваемое на один узел, τ_e- время, затрачиваемое на передачу одного вещественного числа (без учета времени на инициализацию операции обмена данными), τ_i - время, затрачиваемое на инициализацию операции обмена данными, N_T - число узлов, участвующих в пересылке, k - число массивов по узлам, участвующих в пересылке, n_e - число операций обмена данными Время, затрачиваемое на пересылку, в случае отсутствия наложение вычислений, будет следующим:

$$T = \tau_e k N_T + \tau_i n_e$$

Если использовать наложение вычислений по N_T узлам, то время, затрачиваемое на наложение вычислений и пересылку, будет равно:

$$\overline{T} = \tau_c N_T + \tau_e \overline{k} N_T + \tau_i \overline{n}_e$$

Условие эффективности наложения вычислений: $\overline{T} < T$ Если пренебречь затратами времени на инициализацию обмена, получим условие эффективности наложения вычислений в следующем виде:

$$\tau_c N_T + \tau_e \overline{k} N_T < \tau_e k N_T$$

которое эквивалентно

$$\tau_c < \tau_e (k - \overline{k})$$

Где $(k - \overline{k})$ - количество массивов, которые исключаются из пересылки. Из этого условия видно, что если скорость обмена данными очень высокая, как, например, на многопроцессорных системах с общей памятью, то дополнительные вычисления могут оказаться не выгоднее пересылки. Поэтому, если комплекс программ предполагается использовать только на системах с общей памятью, единственным весомым аргументом в пользу наложения вычислений остается упрощение распараллеливания. Дополнительные вычисления не будут выгодны в тех редких случаях, когда их объем достаточно большой, а объем пересылки, от которой они избавляют, незначителен. Высокая эффективность наложения вычислений в случае с NOISETTE была подтверждена тестированием на различных параллельных системах. Результаты тестирования приведены далее в работе.

1.4.2. Дополнительные структуры данных для параллельных вычислений

В последовательном коде Noisette используются следующие типы массивов:

1. Массивы по узлам.

   Число элементов этого массива равно числу узлов. Номер элемента массива равен номеру узла в сетке.

2. Массивы по ребрам.

   Число элементов массива равно числу ребер. Номер элемента массива равен номеру ребра в сетке.

3. Массивы по треугольникам (В трехмерном случае - поверхностным треугольникам).

   Число элементов массива равно числу треугольников. Номер элемента массива равен номеру треугольника в сетке.

4. Массивы по тетраэдрам (в трехмерном случае).

   Число элементов массива равно числу тетраэдров. Номер элемента массива равен номеру тетраэдра в сетке.

5. Массивы по определенному набору узлов.

   Число элементов массива равно числу узлов в наборе. Пример такого набора - граничные узлы определенного типа, массивы контрольных точек или другие дополнительные массивы. Соответствие между номером узла сетки и номером элемента массива устанавливается соответствующим этому массиву индексным массивом.

6. Индексные массивы.

   Эти массивы используются в паре с массивами по наборам узлов. Номер элемента массива соответствует номеру узла в соответствующем наборе. Значение элемента массива равно номеру этого узла в сетке.

В отличие от последовательной версии, в структуре данных присутствуют дополнительные массивы - массивы перенумерации, которые составляют основу функционирования параллельной версии. Для расчета на многопроцессорной системе, исходная сетка разбивается на подобласти, и каждый процессор производит вычисления для узлов своей подобласти. Для расчета по своей подобласти каждому процессору требуются значения некоторых величин в узлах, принадлежащих соседним подобластям. Назовем расширенной подобластью множество узлов, необходимых процессору для расчета по узлам своей подобласти. В расширенную подобласть кроме узлов подобласти будут входить также приграничные узлы из соседних подобластей, необходимые для расчета по подобласти. Таким образом, определены три набора узлов:

1. Все узлы сетки
2. Узлы подобласти
3. Узлы расширенной подобласти

Эти наборы узлов удовлетворяют условию:

Любой узел из подобласти принадлежит расширенной подобласти.

Любой узел из расширенной подобласти принадлежит всей области.

Соответственно вводятся три типа нумерации узлов:

1. Глобальная нумерация

   Номер узла - порядковый номер узла в исходной сетке. Эта нумерация используется для сборки результата.

2. Локальная нумерация

   Номер узла - порядковый номер узла в подобласти. Эта нумерация используется только для операций суммирования по узлам.

3. Локальная расширенная нумерация

   Номер узла - порядковый номер узла в расширенной подобласти. Это основная нумерация, которая используется в большинстве вычислительных циклов.

Аналогичная нумерация вводится также других элементов сетки - ребер, треугольников и тетраэдров:

1. Локальная нумерация

   Элемент принадлежит подобласти, если хотя бы один его узел принадлежит подобласти. Номер элемента - порядковый номер в подобласти.

2. Расширенная локальная нумерация

   Элемент принадлежит расширенной подобласти, если все его узлы принадлежат расширенной подобласти. Номер элемента - порядковый номер в расширенной подобласти.

Для ребер, треугольников и тетраэдров глобальная нумерация не используется. Для описанных выше нумераций узлов вводятся соответствующие массивы перенумерации:

- LocalGlobal : Локальная нумерация → Глобальная нумерация
- GlobalLocal : Глобальная нумерация → Локальная нумерация
- LocalLocalExt : Локальная нумерация → Локальная расширенная нумерация
- LocalExtLocal : Локальная расширенная нумерация → Локальная нумерация
- LocalExtGlobal :Локальная расширенная нумерация →Глобальная нумерация
- GlobalLocalExt : Глобальная нумерация → Локальная расширенная нумерация

Номер элемента массива перенумерации соответствует номеру узла в первой нумерации. Значение этого элемента массива перенумерации соответствует номеру узла во второй нумерации. Если узла не существует во второй нумерации, значение элемента массива должно быть меньше ноля. Например, при перенумерации из глобальной нумерации в локальную, все элементы массива перенумерации, соответствующие узлам, не принадлежащим подобласти, будут иметь значение меньше ноля. На Рис. 1.6 изображена схема действия массивов перенумерации.

Для нумераций ребер также используется массив переиндексации LocalEdges : Локальная расширенная нумерация → Локальная нумерация. Перенумерация ребер в обратную сторону в данном случае не используется. Не используется также перенумерация для треугольников и тетраэдров,

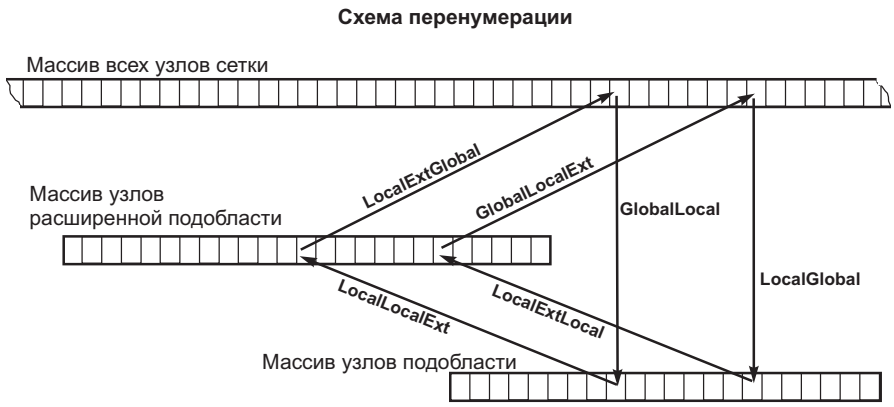

Рис. 1.6. Схема перенумерации массивов по узлам

поскольку в данном случае используется только одна нумерация - локальная расширенная.

Основные массивы с данными в исходной версии имели следующие размеры:

- Массивы по узлам: 1..N, где N - число узлов в сетке.

- Массивы по ребрам: 1..NS, где NS - число ребер в сетке. "

- Массивы по треугольникам в 2D и по тетраэдрам в 3D случае: 1..NT, где NT - число треугольников/тетраэдров в сетке.

- Массивы по поверхностным треугольникам (только в 3D случае): 1..NTr, где NTr - число поверхностных треугольников в сетке.

В параллельной версии размеры основных массивов сужаются до **расширенной подобласти** (для минимизации отличий используются те же обозначения):

N - число узлов в расширенной подобласти

NS - число ребер в расширенной подобласти.

NT - число треугольников/тетраэдров в расширенной подобласти.

NTr - число поверхностных треугольников в расширенной подобласти.

Остальным типам нумерации соответствуют следующие диапазоны:

Локальная нумерация узлов: 1,..,Nloc где Nloc - число узлов в подобласти;

Глобальная по узлам: 1,..,Nglob, где Nglob - число узлов во всей сетке;

Локальная нумерация ребер: 1,..,NSloc, где NSloc - число ребер в локальной нумерации.

При такой структуре, **подавляющее большинство вычислительных циклов вообще не требует внесения изменений**. Таким образом, минимизируются отличия между параллельной и исходной версиями. Циклы по всей области автоматически становятся циклами по расширенной подобласти. Требуют модификации только некоторые циклы, связанные с глобальными операциями по всей области, такими как вычисление суммы, среднего и т.д. Такие циклы должны быть сужены до подобласти. В NOISETTE такое сужение выполнено с помощью добавления одной строки к заголовку цикла:

Исходный заголовок цикла.

DO IS = 1 , N

Цикл по подобласти в параллельной версии:

DO ISLOC = 1 , Nloc

IS=LocalLocalExt(ISLOC)

1.4.3. Организация обмена данными в параллельной версии

Обмен данными построен на основе индексных массивов пересылки данных. Эти массивы содержат номера узлов, участвующих в пересылке, и формируются на основании данных из коммуникационного файла comm.dat.

Основные массивы для организации межпроцессорного обмена данными:

- NBProc(N_{NB}+1,3) Содержит информацию о соседних процессорах и количестве узлов для пересылки:

  NBProc(1,1) = N_{NB}- число соседних процессоров.

  NBProc(1,2), NBProc(1,3) - не используются.

  Далее для остальных i=2,. . . ,$N_{NB} + 1$:

  NBProc(i+1,1) - номер i-го соседа.

  NBProc(i+1,2) - число узлов, которые нужно переслать i-му соседу

  NBProc(i+1,3) - число узлов, которые нужно получить от i-го соседа

- Sendnodes(N_{send}) В этом массиве содержатся номера узлов, данные из которых надо передать. N_{send} - число таких узлов. Для каждого соседнего процессора последовательно записаны номера узлов, данные из которых нужно передать этому процессору. При передаче данных

из массива NBProc определяется количество узлов для передачи, и номера узлов последовательно считываются из массива Sendnodes.

- Recvnodes(N_{recv}) В этом массиве содержатся номера узлов, данные для которых надо получить. N_{recv} - число таких узлов. Для каждого соседнего процессора последовательно записаны номера узлов, данные из которых ему нужно получить. При приеме данных из массива NBProc определяется количество узлов для приема, и номера узлов последовательно считываются из массива Recvnodes.

В параллельной версии Noisette используется технология параллельного программирования MPI (Message Passing Interface), предназначенная для вычислительных систем с распределенной памятью. Используются как операции типа точка-точка, так и групповые операции обмена данными.

Групповой обмен данными используется только для глобальных операций по всей области, таких как суммирование значений в узлах, поиск минимума и максимума. Для этих целей используется MPI функция MPI_AllReduce, которая обеспечивает поиск минимума, максимума и суммы.

Обмен данными типа точка-точка (т.е. обмен данными происходит в группах из 2-х процессоров - отправителя и получателя) используется для пересылки значений в узлах. Для этого используются функции MPI буферизованной неблокирующей передачи данных:

- MPI_Isend - инициализирует посылку данных
- MPI_Irecv - инициализирует прием данных
- MPI_Waitall - ожидает завершения операций передачи данных

Обмен данными реализован в виде отдельной функции Update. Функция выполняет двусторонний обмен данными со всеми соседями для обновления значений в гало, то есть в узлах расширенной подобласти, принадлежащих другим подобластям. Таким образом, в коде, в месте, где необходимо получить значения в узлах из соседних подобластей, достаточно добавить вызов этой функции. Главная особенность данной функции в том, что даже если требуется передача нескольких величин, то есть передача данных из нескольких массивов, пересылка происходит за одну операцию обмена данными. Для этого все значения, которые необходимо передать, предварительно помещаются в специальный буфер. Таким образом, минимизируется количество операций обмена данными, а следовательно минимизируются затраты времени на инициализацию обмена данными, то есть на латентность сети, которая существенно сказывается на производительности в случае малобюджетных кластеров.

Функция Update устроена следующим образом:

1. Формирование посылки Данные из массивов, содержащего значения в узлах, размещаются в буфере для отправки посылки. Sbuf(I,NB)-буфер для отправки посылки. Первый индекс I служит для доступа к пересылаем значениям, второй индекс NB - номер соседа, для которого предназначена посылка. В цикле по всем соседям происходит выборка номеров узлов для отправки и размещение данных из этих узлов в буфере. Для этого используется массив Sendnodes.

2. Инициализация отправки данных всем соседям В цикле по всем со-

седям вызывается функция MPI_Isend

3. Инициализация приема данных от всех соседей В цикле по всем соседям вызывается функция MPI_Irecv

4. Ожидание завершения операций обмена данными. Вызывается функция MPI_Waitall, которая ожидает завершения всех функций MPI_Isend и MPI_Irecv

5. Распаковка посылки Данные из буфера приема посылки размещаются в массиве, содержащем значения в узлах. Rbuf(I,NB)- буфер для приема посылки. Первый индекс I служит для доступа к получаемым значениям, второй индекс NB - номер соседа, от которого получена посылка.

В цикле по всем соседям происходит выборка номеров узлов, для которых получены данные, и размещение данных из буфера в этих узлах. Для этого используется массив Recvnodes.

1.5. Эффективность параллельных вычислений

1.5.1. Верификация параллельного кода

Верификация параллельной версии Noisette, т.е. подтверждение отсутствия ошибок при распараллеливании, проводилось посредством сравнения результатов вычислений исходной и параллельной версии на полное совпадение.

В качестве тестовой задачи был взят известный аероакустический тест, предложенном К.Тамом на 2-м семинаре по тестовым проблемам в

аэроакустике [46] (категория 3, задача 1), проводившемся в 1994 году. Тест описывает эволюцию начального возмущения плотности и давления, заданную в виде гауссовского распределения, на однородном среднем поле течения. Тестовый расчет выполнялся следующим образом: сначала проводилось сравнение контрольных значений на каждом шаге по времени, затем сравнивались результаты вычислений. В качестве контрольных значений брались минимум, максимум и сумма по всем узлам основных физических величин, а также значение шага по времени. Были выполнены тесты с использованием различных сеток, различного набора параметров модели и разностной схемы, а также различного числа процессоров. Совпадение было получено во всех случаях, и, таким образом, отсутствие ошибок при распараллеливании было подтверждено.

1.5.2. Описание оборудования

Измерение производительности производилось на двух параллельных системах разных типов. Первая система состоит из узлов с 32-битными процессорами и построена на основе стандартной сети Ethernet 1GBit, которая используется во многих малобюджетных параллельных системах. Вторая система состоит из узлов с 64-битными процессорам и построена на основе высокопроизводительной сети Myrinet, типичной для суперкомпьютеров. Эти различия оказывают существенное влияние на параллельную эффективность, поэтому, для сравнения, тестирование было проведено на обеих системах. Особенно важно было сравнение параллельной эффективности на системах со стандартной и высокопроизводительной сетью, чтобы определить влияние обмена данными на падении эффективности при росте

числа процессоров.

- Кластер ИММ2, установленный в Институте Мат. Моделирования

  Эта параллельная система состоит из 25 двухпроцессорных узлов, имеющих следующую конфигурацию:

  2 32-битных процессора Intel Xeon, 3.0 ГГц, кеш-память 512 Кб;

  2 Гб оперативной памяти.

  Узлы соединены сетью Ethernet 1Gbit.

  Кластер управляется операционной системой Linux.

  Для работы параллельных процессов используется библиотека MPICH.

- Кластер RSC4, установленный в Институте Прикладной Математики им. Келдыша

  Эта параллельная система состоит из 48 двухпроцессорных узлов, имеющих следующую конфигурацию:

  2 64-битных процессора AMD Opteron, 2.4 GHz, кеш-память 1024 Кб;

  2 Гб оперативной памяти.

  Сеть: Для работы параллельных процессов используется сеть Myrinet. Кроме того, для работы распределенной файловой системы используется сеть Ethernet 1Gbit.

  Кластер управляется операционной системой Linux.

  Для работы параллельных процессов используется библиотека MPICH.

1.5.3. Измерение производительности

Для получения информации о производительности программного комплекса на параллельной системе, измерялось время, затрачиваемое на вычисление фиксированного числа шагов по времени, с использованием различного числа процессоров. Физический смысл и подробности постановки тестовой задачи, такие как начальное возмущение и граничные условия, в данном случае значения не имеют, т.к. измерялось лишь время вычислений фиксированного числа шагов по времени. Для тестов использовалась неструктурированная треугольная сетка, применяемая в расчетах звукопоглощающих конструкций. Сетка имеет характерный для такой задачи размер 84 тысячи узлов. Параметры разностной схемы были выбраны таким образом, чтобы обеспечивался наибольший порядок точности и, таким образом, объем пересылки был максимальным.

- **Ускорение**

  Ускорение S_P - это отношение времени вычислений на одном процессоре ко времени вычислений на P процессорах. Ускорение показывает, во сколько раз быстрее проходят вычисления в параллельном режиме по сравнению с однопроцессорным режимом. В идеальном случае ускорение на P процессорах равно P. В реальности ускорение обычно ниже из-за затрат времени на обмен данными и другие операции. Результаты теста показаны на рисунке 1.7.

- **Эффективность**

  Эффективность параллельных вычислений E - это отношение полученного ускорения к числу процессоров, умноженное на 100%, а имен-

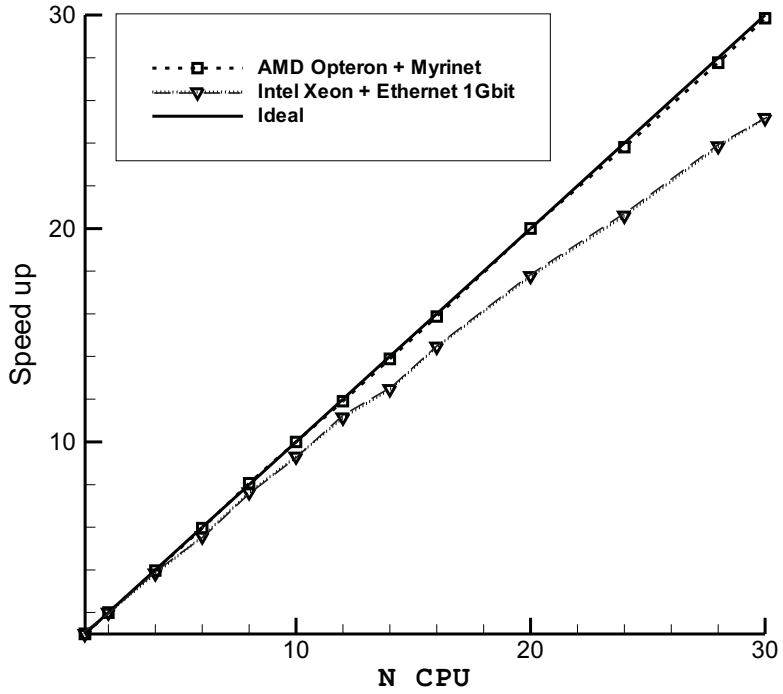

Рис. 1.7. Ускорение вычислений на параллельных системах

но

$$E = \frac{S_P}{P} 100\%$$

где S_P - ускорение, полученное на P процессорах. Эта величина показывает, насколько эффективно используется параллельная вычислительная система. Результаты теста приведены на рисунке 1.8.

1.5.4. Анализ результатов

Одна из основных целей сравнения производительности кода на сети со стандартной и высокой производительностью - оценить влияние обмена данными на падение производительности. Результаты тестов приведены в таблице 1.1. Поскольку сеть Myrinet имеет существенно меньшую латентность, чем сеть Ethernet, скорость передачи коротких сообщений между

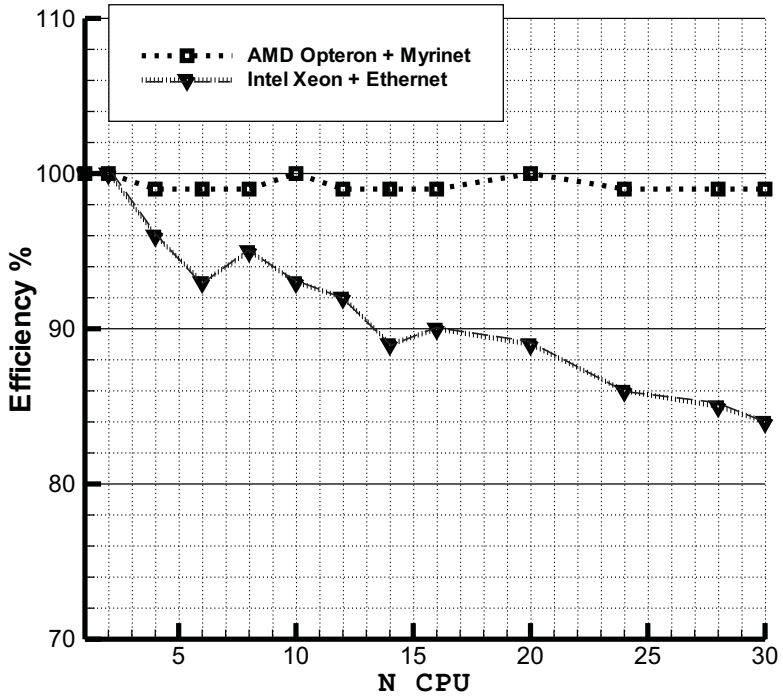

Рис. 1.8. Эффективность вычислений на параллельных системах

процессами на порядки выше с использованием Myrinet. Поэтому в данном случае время, затрачиваемое на обмен данными на кластере с Myrinet пренебрежимо мало по сравнению со временем на обмен данными на кластере с Ethernet. Этим объясняется такая высокая параллельная эффективность на системе с высокопроизводительной сетью на 30 процессорах - 99%. На системе со стандартной сетью эффективность распараллеливания составила 84%. Таким образом, было показано, что производительность при росте числа процессоров падает исключительно из-за обмена данными, а наложения вычислений вообще не сказывается на производительности. **Это подтверждает эффективность использованного при распараллеливании подхода - заменять по возможности пересылку данных наложением вычислений**.

Таблица 1.1. Результаты измерения производительности

	Ускорение, раз		Эффективность, %	
Число процессоров	ИММ2	РСЦ4	ИММ2	РСЦ4
1	1	1	100	100
2	2	2	100	100
4	3.84	3.96	96	99
6	5.57	5.95	93	99
8	7.6	7.93	95	99
10	9.32	10.0	93	100
12	11.1	11.91	92	99
14	12.48	13.89	89	99
16	14.47	15.78	90	99
20	17.78	20	89	100
24	20.61	23.81	86	99
28	23.86	27.78	85	99
30	25.19	29.85	84	99

1.6. Структура многоплатформенной параллельной версии пакета SuperNoisette

SuperNoisette - совмещенная версия для расчетов 2D и 3D задач на многопроцессорных вычислительных комплексах с операционными системами Windows, Unix, Linux.

1.6.1. Состав пакета

- Программы для подготовки сетки
    1. Программа декомпозиции сетки [45]
    2. Программа построения обмена данными
    3. Конвертер для преобразования формата файлов сетки
- Вычислительная часть
    1. Программа SuperNoisette 2D-3D

- Обработка результатов и визуализация

    1. Программа для объединения файлов восстановления счета с каждого процессора в один файл.
    2. Программа для объединения файлов визуализации в формате Tecplot с каждого процессора в один файл.

1.6.2. Структура каталогов пакета и основные файлы

Корневой каталог

main.exe - Исполнимый файл программы Noisette 2D-3D в случае ОС Windows

main.px - Исполнимый файл программы Noisette 2D-3D в случае ОС Unix/Linux

makefile - Файл для сборки исполнимого файла Noisette 2D-3D

- **fort.1** - файл параметров расчета

_mpirun.bat - скрипт для запуска расчета в случае ОС Unix/Linux

main.pg - файл параметров для многопроцессорного запуска в случае ОС Windows

DOCS - каталог с документацией

INPUT - каталог с входными файлами

CONTROL_POINTS - каталог контрольных точек

contr_points.dat - файл контрольных точек

DATA - каталог с различным набором данных в зависимости от типа расчета

EXSOL - каталог с данными точного решения

MESH - каталог с файлами сетки

Comm.dat.## - файл схемы обмена данными

Mesh.msh - заголовок сетки

Coordinate.msh - координаты узлов сетки

Tetrahedron.msh - Список тетраэдров сетки (только в 3Д случае)

Triangle.msh - Список треугольников сетки в 2Д случае и список поверхностных треугольников в 3Д случае

Nodetype.msh - Типы узлов (только в 2Д случае)

Label.msh - Типы граничных условий

MESHDEC - Каталог с программами для подготовки сетки (используется только под ОС Windows)

BIN - Исполнимые файлы

CODE - Исходный код

SCRIPTS - Скрипты автоматизации обработки данных

1cpu.bat - Генерация фиктивной схемы обмена данными для 1 процессора

check_tetrahedrons.bat - Проверка и исправление нумерации тетраэдров в 3Д и треугольников в 2Д и 3Д случае

convert_fort16_2d.bat - преобразование формата файла сетки

decompose.bat - разбиение сетки и построение схемы обмена данными

decreport.txt - отчет о результатах разбиения

NOISETTE - исходный код программы Noisette 2D-3D

OUTPUT - каталог с выходными файлами

Parlog - каталог с файлами вывода текущей информации

RESULS - каталог с файлами результата

DATA - каталог с различным набором данных в зависимости от типа расчета

EXSOL - каталог с файлами сравнения с точным решением

HISTORY - каталог с файлами значений в контрольных точках

RECOVERY - каталог с файлами восстановления счета

VISUA - каталог с файлами визуализации в формате Tecplot

UTIL - каталог со вспомогательными программами

COMBINE - каталог с программой для объединения файлов восстановления счета

combine.exe - исполнимый файл программы

combine.ini - файл параметров

TECJOIN - каталог с программой для объединения файлов Tecplot

tecjoin.exe - исполнимый файл программы

2. Численное моделирование задач резонаторного типа

2.1. Моделирование звукопоглощающих конструкций

Одно из основных направлений применения параллельного комплекса программ NOISETTE 2D/3D является моделирование звукопоглощающих конструкий (ЗПК). ЗПК резонансного типа широко распространены в авиационном строении для снижения шума турбореактивных двигателей. ЗПК представляет собой одно- или двухслойную перфорированную панель, конфигурация и геометрические параметры которой существенным образом влияют на эффективность поглощения шума. Для оптимизации параметров ЗПК удобным инструментом может служить математическое моделирование. Хорошо отлаженная вычислительная среда, обеспечивающая расчеты ЗПК в различных конфигурациях, может рассматриваться как виртуальный экспериментальный стенд, легко адаптируемый к широкому диапазону допустимых геометрических параметров и амплитудно-частотных характеристик входного сигнала, и, соответственно, как эффективное средство в помощь физическому эксперименту при конструировании ЗПК. Детальное численное моделирование, к тому же, способствует глубокому пониманию физических механизмов, определяющих звукопоглощение. Математическое моделирование поглощения шума в ЗПК резонансного типа является типичной задачей нелинейной аэроакустики. Спецификой численного моделирования таких задач является то факт, что

в случае высокой мощности входящего акустического сигнала, расчетная область включает в себя зоны как линейных, так и нелинейных взаимодействий. В "линейном"регионе определяющим физическим процессом является распространение акустических волн, моделирование которого требует высокой точности для обеспечения их передачи во времени и пространстве без искажений. В "нелинейных"зонах методы высокой точности должны быть адаптированы к возможным высоким градиентам решения, включая слабые разрывы. Поэтому численные алгоритмы нелинейной аэроакустики должны сочетать в себе лучшие свойства методов линейной акустики и нелинейной газовой динамики, а также адаптироваться в процессе расчета к решению задачи в зависимости от его свойств. Класс подобных алгоритмов на декартовых сетках применительно к задачам о моделировании ЗПК развивается в работах К.Тама [47, 48] на основе метода DRP (Dispersion Relation Preserving) [49], сохраняющего дисперсионные соотношения. Трудность в конструировании схем такого класса усугубляется при использовании неструктурированных сеток. Ситуацию несколько смягчает только предположение о существенной гладкости рассматриваемых в аэроакустике решений, допускающее лишь слабые разрывы. Для численного моделирования рассматриваемых в работе задач используется метод повышенной точности для расчета задач нелинейной аэроакустики на неструктурированных сетках [50, 51, 52]. Он основан на многопараметрической схеме с определением переменных в узлах и достигает своей максимальной теоретической точности, до 6го порядка включительно, в "декартовых"подобластях расчетной сетки. Под "декартовым"подмножеством сетки понимается участки, полученные делением ячеек декартовой сетки

на два прямоугольных треугольника. В данной работе описываются два основных типа вычислительных экспериментов по ЗПК, а именно моделирование свойств ячейки ЗПК в импедансной трубе; а также моделирование потерь энергии звукового сигнала при его прохождении в дозвуковом течении в канале, облицованном перфорированными панелями. Задачи носят модельный характер, однако могут служить начальным приближением к прямому численному моделированию ЗПК.

2.2. Вычислительный эксперимент в импедансной трубе

Первая группа проведенных вычислительных экспериментов (ВЭ) имитирует условия лабораторного физического исследования ячеек ЗПК в импедансных трубах. Экспериментальная установка представляет собой длинную трубу, в торец которой отделен перфорированным экраном. Получившийся таким образом резонатор представляет собой модель ячейки ЗПК. С противоположного окончания трубы нагнетается монохромная акустическая волна заданной мощности. Наибольший интерес для исследования представляет течение воздуха, генерируемое внутри отверстия (горла резонатора) и в его окрестности. Эти области наиболее сложны для проникновения измерительных приборов, а потому постановка вычислительного эксперимента на основе полного газодинамического описания, включая нелинейные эффекты, а также вязкость и теплопроводность, может оказаться чрезвычайно полезной. Во всех рассматриваемых постановках используется резонатор с глубиной, равной четверти длины набегающей акустической волны. При этом длина расчетной области немного превышала длину входящей волны и, соответственно, более, чем в четыре раза

превышала глубину резонаторной ячейки. Расчеты проводились в двух постановках - плоской двухмерной и **полностью трехмерной**. В трехмерном случае труба и горло резонатора имеют форму цилиндра (трехмерная расчетная область получена поворотом 2D области вокруг продольной оси симметрии).

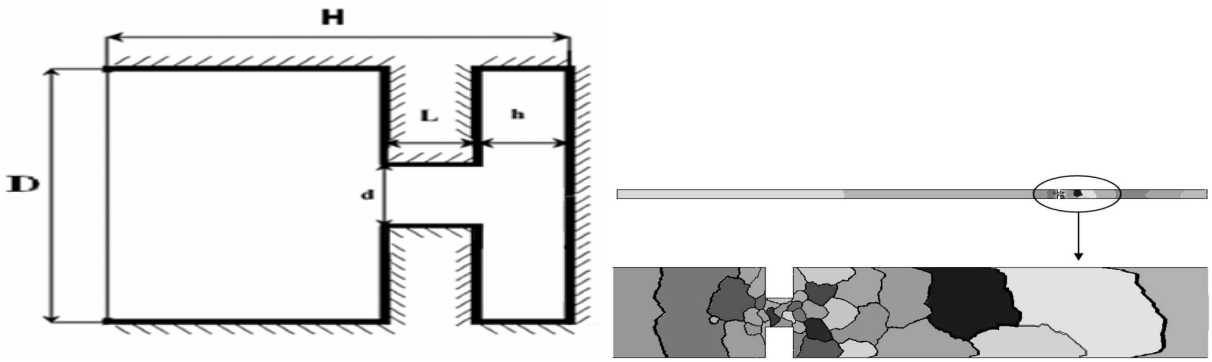

Рис. 2.1. Схема расчетной области (слева) и пример разбиения двухмерной расчетной области (справа)

Схематически расчетная область задачи изображена на Рис. 2.1 (слева). Используемая в расчётах конфигурация имеет следующие фиксированные размеры: диаметр трубы D=2.35 см, диаметр отверстия d=0.75 см, толщина перфорированного экрана L=0.6 см. Монохромная акустическая волна мощностью 147.1 дБ и частотой 273 Гц входит в трубу слева. На Рис. 2.1 (справа) показано также разбиение области на 40 доменов, соответствующим 40 процессорам, для параллельных вычислений по принципу геометрического параллелизма на многопроцессорном кластере. Т.к. при разбиении соблюдалась балансировка загрузки, определяемая в данном случае примерным равенством количества расчетных узлов сетки на каждый процессор, схема на Рис. 2.1 даёт представление об участках разрежения и сгущения неструктурированной сетки. Фрагменты расчётной

сетки приведены на Рис. 2.2. Сетка сгущена в районе горла резонатора для более аккуратного разрешения сложных газодинамических процессов, а также вблизи твердых стенок для адекватного моделирования вязких эффектов и теплопроводности.

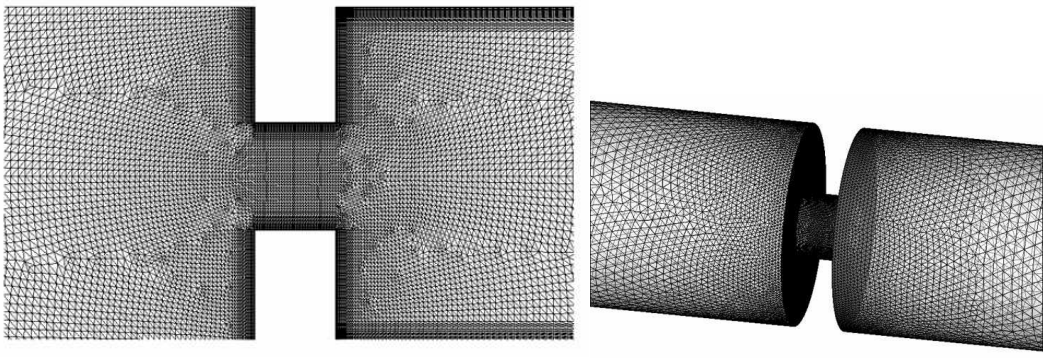

Рис. 2.2. Фрагменты неструктурированной расчётной сетки. (ВЭ в импедансной трубе)

Рис. 2.3 на примере полей плотности показывает качественную разницу в результатах 2D расчетов с использованием трёх математических моделей, а именно линеаризованных уравнений Эйлера (левый верхний рисунок), полных уравнений Эйлера (правые верхний и нижний рисунки), а также уравнений Навье-Стокса (левый нижний рисунок) для сжимаемого газа. Видно, что линейная постановка не моделирует вихревую динамику в окрестности горла резонатора. Использование системы уравнений Эйлера с условиями отражения на стенке приводит, похоже, к чересчур интенсивному вихреобразованию, что объясняется отсутствием диссипации вихрей в этой постановке. Наиболее реалистичную картину течения даёт использование уравнений Навье-Стокса. При заданных условиях задачи входящая акустическая волна теряет свою мощность за счет преобразования акустической энергии в энергию вихрей и ее диссипация за счёт вязкости и

теплопроводности. Формирование парных вихрей в окрестности горла резонатора в двухмерной постановке показано на Рис. 2.4 (слева). Справа на Рис. 2.4 показана картина течения в трехмерном случае.

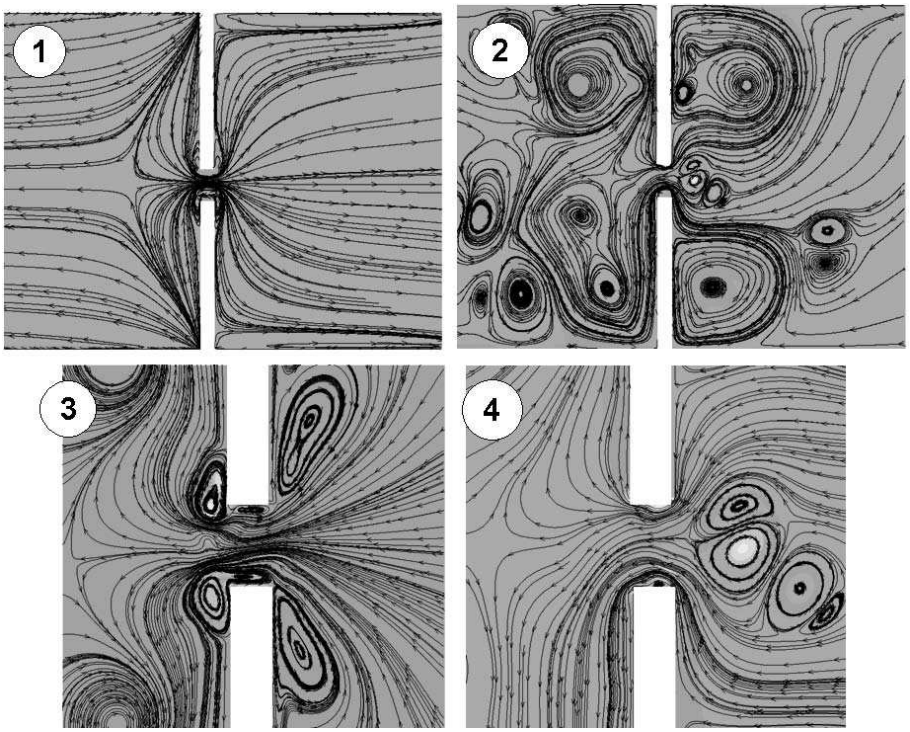

Рис. 2.3. Разница в полях плотности при использовании различных математических моделей (двухмерная постановка): линеаризованные уравнения Эйлера (1); полные уравнения Эйлера (2,4); уравнения Навье-Стокса (3)

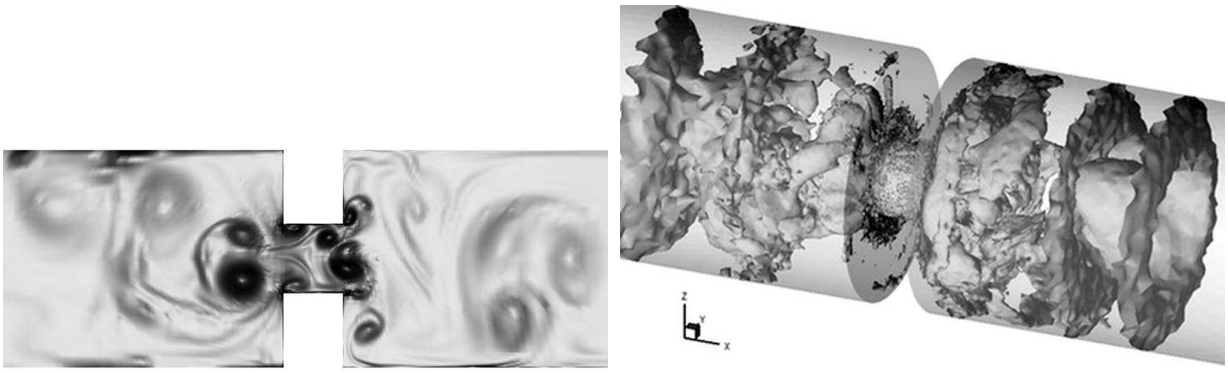

Рис. 2.4. Вихревое течение в окрестности горла резонатора: 2D - слева, 3D - справа

3D расчет также показал, что течение является полностью трехмерным, без осевой симметрии (что, впрочем, вполне естественно для турбулентного течения). На Рис. 2.5 изображена картина течения в сечении за отверстием резонатора. Визуализация плотности и компоненты вектора скорости по оси X четко показывает нерегулярность течения и отсутствие симметрии.

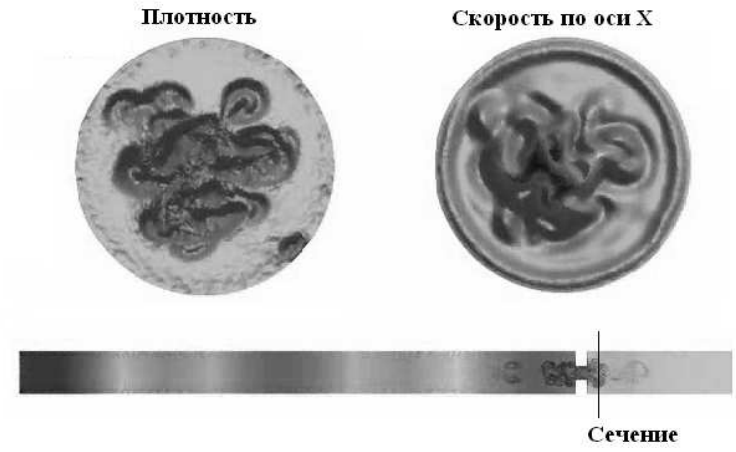

Рис. 2.5. Картина течения в трехмерном случае в сечении, перпендикулярном оси X, за отверстием внутри резонатора (теневая визуализация возмущений плотности и скорости по оси X).

2.3. Вычислительный эксперимент в канале со встроенными в стенку резонаторами

Формулировка второй рассматриваемой задачи наиболее близка к условиям модельного физического эксперимента по ЗПК резонансного типа в канале. Физическая постановка заключается в исследовании динамики акустического сигнала проходящего вместе с дозвуковым течением в канале, стенки которого покрыты ЗПК панелями. Расчеты проводились только

для **двухмерной** постановки.

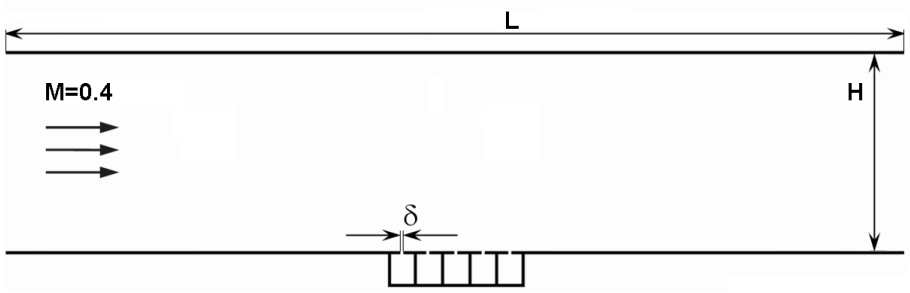

Рис. 2.6. Схема эксперимента в канале.

Схема физического эксперимента приведена на Рис. 2.6.

Описываемая в работе модель вычислительного эксперимента рассматривает двухмерное дозвуковое течение вязкого сжимаемого газа с соответствующим акустическим сигналом в канале, в стенки которого встроен один или несколько резонаторов. При численном моделировании данной задачи задаются следующие входные условия: дозвуковой поток при числе Маха 0.4 (что соответствует условиям захода самолетов на посадку) сопровождается входящей монохромной акустической волной с частотой 3431 Гц и мощностью 150 дБ. Первые расчеты проводились для случая одного резонатора, встроенного в стенку канала. Численное моделирование задач такого типа, также как и задач резонаторного типа вообще, представляет серьезные трудности, связанные с существенной разницей в пространственных масштабах задачи, в частности, между диаметрами канала (30 см) и входного отверстия резонатора (2 мм).

Фрагмент используемой при расчетах неструктурированной треугольной сетки показан на Рис. 2.7. Данная задача решалась при помощи уравнений Эйлера и Навье-Стокса. При использовании уравнений Эйлера

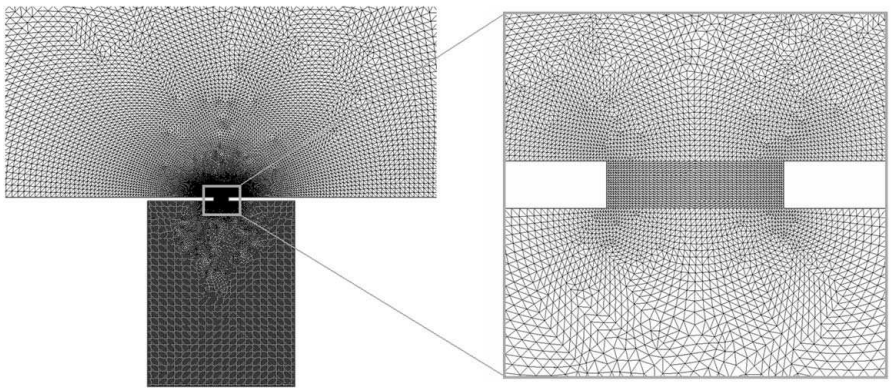

Рис. 2.7. Фрагменты неструктурированной расчётной сетки (ВЭ в канале).

без учета пограничных слоев и последующем анализе результатов можно заметить "свист", вызванный взаимодействием развиваемого слоя смешения между кромками входного отверстия (см. Рис. 2.8 слева), а также многочисленных отражений внутри горла резонатора. В центре на Рис. 2.8 показан турбулентный слой смешения в горле резонатора, а на правом рисунке представлено поле кинетической энергии турбулентности, посчитанное путем соответствующего осреднения по времени в процессе прямого расчета. Как это и ожидалось, максимум турбулентной энергии располагается вдоль центральной линии слоя смешения.

Таблица 2.1. Сравнение звукопоглощения

Число резонаторов	Акустическая мощность, дБ	Поглощение, дБ
0	148.8	0
5	147.1	1.7
11	144.3	4.5

Рисунок 2.9 демонстрирует искажение входящего длинноволнового акустического сигнала из-за влияния высокочастотных цилиндрических волн ("свиста"), генерируемых в потоке в результате взаимодействия с отверстием в рамках описания на основе уравнений Эйлера.

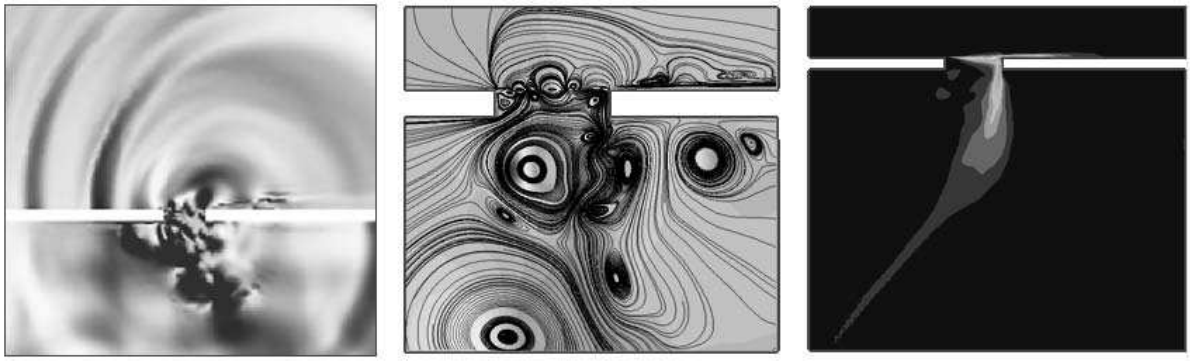

Рис. 2.8. Формирование сдвигового слоя между кромками входного отверстия резонатора (слева и по центру). Распределение кинетической энергии турбулентности (справа).

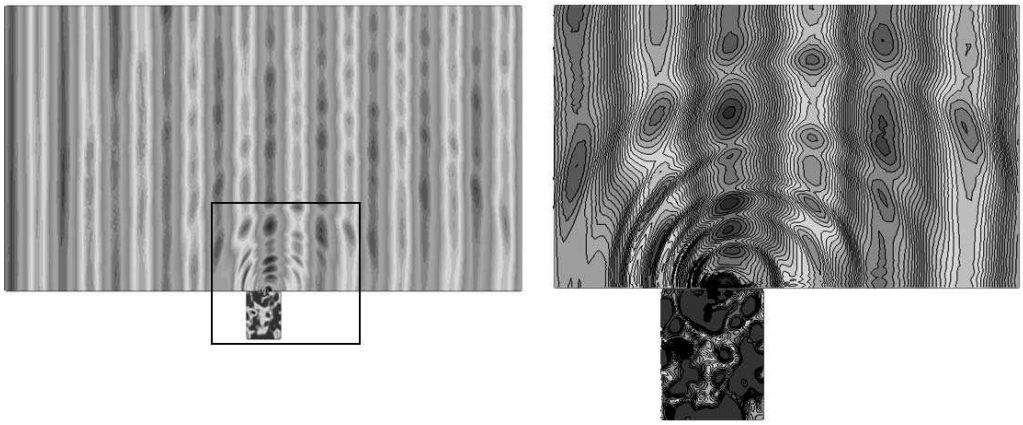

Рис. 2.9. Искажение входящей акустической волны из-за высокочастотных цилиндрических волн, излучаемых входным отверстием резонатора.

При использовании уравнений Навье-Стокса и специальных входных граничных условий, задающих течение с толстым погранслоем (погранслой по стандартному адиабатическому профилю с толщиной 5 диаметров отверстия резонатора), наличие медленного пристеночного течения существенным образом меняет газодинамическую картину в горле резонатора. В данной постановке отсутствует какой бы то ни было "свист".

Случай системы из пяти резонаторов, встроенных в стенки канала

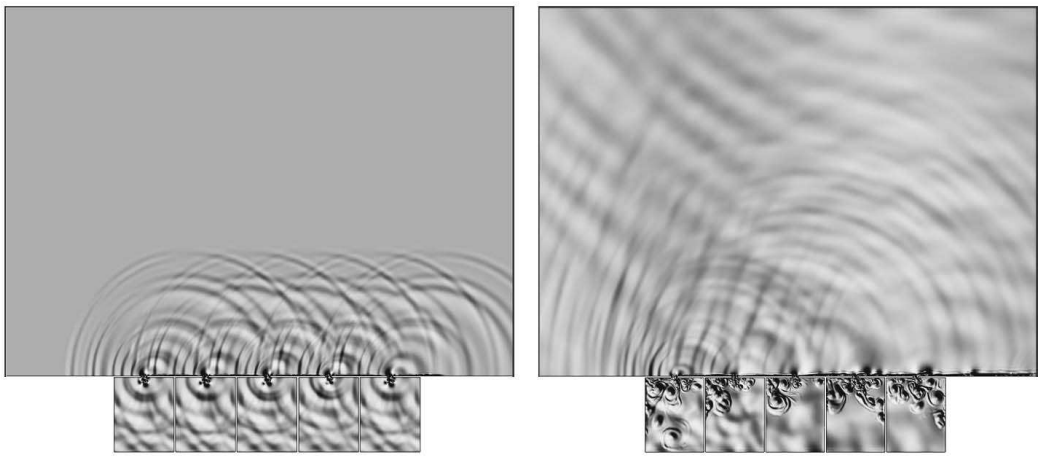

Рис. 2.10. Начальная и развитая стадии в случае пяти резонаторов, встроенных в стенку канала. Поле плотности.

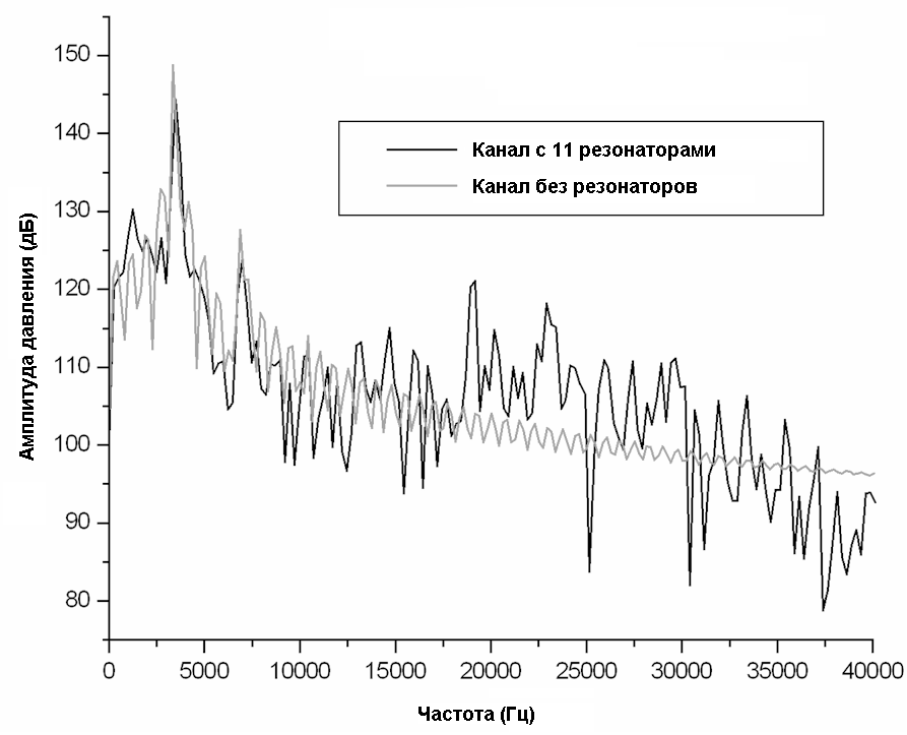

Рис. 2.11. Сравнение спектр выходного акустического сигнала.

был рассмотрен в рамках модели на основе уравнений Эйлера. Распределение плотности с видимыми акустическими полями в разные моменты времени представлены на рисунке 2.10.

Для оценки эффекта поглощения звука было проведено сравнение спектра и выходной акустической энергии между тремя постановками: канал без резонаторов, канал с 5-ю резонаторам, канал с 11 резонаторами. Результаты представлены в таблице 2.1. Сравнение спектра показано на Рис. 2.11.

3. Масштабируемый параллельный алгоритм для численного моделирования несжимаемых течений

3.1. Математическая модель и дискретизация

Рассматривается несжимаемая жидкость с постоянными физическими свойствами. Приближение Буссинеска используется для зависимости плотности от температуры. Тепловым излучением пренебрегается. В этом случае система уравнений Навье–Стокса в безразмерном виде будет иметь вид

$$\nabla \cdot \mathbf{u} = 0\,, \tag{3.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} + Pr\nabla^2 \mathbf{u} - \nabla p + \mathbf{f}\,, \tag{3.2}$$

$$\frac{\partial T}{\partial t} = -\mathbf{u} \cdot \nabla T + \nabla^2 T\,, \tag{3.3}$$

где $\mathbf{f} = [0, 0, RaPrT]^T$ — вектор массовых сил.

В рассматриваемом случае расчетная область представляет собой параллелепипед. На сторонах, перпендикулярных оси X, используются периодические граничные условия. Это позволяет изучить трехмерные эффекты, которые вызваны именно внутренней нестабильностью потока, а не граничными условиями. В этом случае можно использовать сетку с постоянным шагом по оси X, поскольку не требуется сгущение сетки для разрешения пограничного слоя, что позволяет применить БПФ метод для уравнения Пуассона. На других сторонах могут использоваться различные

граничные условия.

3.1.1. Метод интегрирования по времени

Для упрощения записи, уравнение момента (3.2) может быть записано в виде

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{R}\left(\mathbf{u}\right) - \nabla p\,,$$

где $\mathbf{R}(\mathbf{u})$ — члены правой части уравнения, за исключением градиента давления.

Для дискретизации по времени используется схема центральной разности для $\partial\mathbf{u}/\partial t$, полностью явная схема Адамса–Башфора второго порядка для $\mathbf{R}$ и обратная схема Эйлера первого порядка для ∇p и для уравнения неразрывности.

Таким образом, считая шаг по времени постоянным, получим полудискретную систему уравнений Навье–Стокса

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\Delta t} = \frac{3}{2}\mathbf{R}^{n} - \frac{1}{2}\mathbf{R}^{n-1} - \nabla p^{n+1}, \tag{3.4}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u}^{n+1} = 0\,. \tag{3.5}$$

Для обеспечения связей между скоростью и давлением используется классический проекционный метод с дробным шагом [53, 54]. В проекционном методе решения нестационарных уравнений Навье–Стокса получаются в два этапа: сначала находится поле скоростей на новом временном слое без учета требования соленоидальности (3.5), затем восстанавливается правильное соленоидальное поле скоростей $\mathbf{u}^{n+1}$ ($\boldsymbol{\nabla} \cdot \mathbf{u}^{n+1} = 0$). Эта проекция получена из теоремы Гельмгольца–Ходжа о разложении вектора [55], следуя которой, скорость $\mathbf{u}^{n+1}$ может быть единственным образом разложена

на соленоидальную составляющую $\mathbf{u}^p$ и на безвихревую составляющую, выраженную как градиент от скалярного поля, $\nabla\tilde{p}$:

$$\mathbf{u}^p = \mathbf{u}^{n+1} + \nabla\tilde{p}\,. \tag{3.6}$$

Представив "псевдодавление" в виде $\tilde{p} = \Delta t\, p^{n+1}$, получим

$$\mathbf{u}^p = \mathbf{u}^n + \Delta t\left[\frac{3}{2}\mathbf{R}^n - \frac{1}{2}\mathbf{R}^{n-1}\right]. \tag{3.7}$$

Применяя оператор дивергенции к уравнению (3.6), получим уравнение Пуассона для $\tilde{p}$

$$\boldsymbol{\nabla}\cdot\mathbf{u}^p = \boldsymbol{\nabla}\cdot\mathbf{u}^{n+1} + \boldsymbol{\nabla}\cdot(\nabla\tilde{p}) \longrightarrow \Delta\tilde{p} = \boldsymbol{\nabla}\cdot\mathbf{u}^p\,. \tag{3.8}$$

Вопрос о том, какие граничные условия использовать для уравнения (3.8) в направлениях без периодических условий является спорным. Основные идеи вкратце изложены в [56]. Использование нормальной компоненты уравнения момента обычно считается наиболее подходящим граничным условием, см., например, [57]. Однако на дискретном уровне на смещенных сетках с граничными условиями для скоростей как в случае рассматриваемых в работе задач (условия прилипания - все компоненты скорости равны нолю), условие несжимаемости удовлетворяется естественным образом, и никаких особых граничных условий для давления не требуется, как показано в [58].

Когда решение уравнения (3.8) получено, $\mathbf{u}^{n+1}$ находится коррекцией:

$$\mathbf{u}^{n+1} = \mathbf{u}^p - \nabla\tilde{p}\,. \tag{3.9}$$

Таким образом, на каждом шаге по времени выполняется следующий алгоритм:

1. Нахождение $\mathbf{R}^n = \mathbf{R}(\mathbf{u}^n)$.

2. Нахождение $\mathbf{u}^p$ из уравнения (3.7).

3. Нахождение $\boldsymbol{\nabla} \cdot \mathbf{u}^p$ и решение дискретного уравнения Пуассона (3.8).

4. Получение нового поля скоростей $\mathbf{u}^{n+1}$ из уравнения (3.9).

5. Нахождение нового поля температуры T^{n+1} из уравнения энергии (3.3).

Хорошо известно, что требования устойчивости явной схемы ведут к ограничениям на шаг по времени. Неявная схема обладает лучшей устойчивостью и позволяет использовать больший шаг по времени. Но, во-первых, для неявной схемы требуются существенно большие вычислительные затраты, чем для явной схемы. Во-вторых, при прямом численном моделировании турбулентных течений ограничение на шаг по времени также накладывается необходимостью полностью разрешать все временные масштабы в уравнениях Навье-Стокса [59, 21, 37]. Поэтому был использован только явный метод с целью уменьшения вычислительной стоимости и сложности алгоритма.

Явные вычисления включают уравнения, решаемые на шагах 1,2,4 алгоритма интеграции по времени, а также решение уравнение энергии (3.3). Для параллельного подхода эти уравнения не представляют проблем, поскольку явный метод требует для получения решения лишь один обмен данными типа точка-точка с соседними процессорами. Подобные явные методы хорошо масштабируются. Основную проблему для параллельных вычислений представляет уравнение Пуассона, которое связывает всю рас-

четную область.

3.1.2. Пространственная дискретизация

Уравнения (3.1)-(3.3) заменяются дискретными по пространству на смещенной сетке с помощью спектрально согласованных схем второго или четвертого порядков, описанных в [35, 36, 37]. Используя те же обозначения, сохраняющая симметрию дискретизация уравнений Навье–Стокса представляется в следующем виде:

$$\begin{aligned} \mathsf{M}\boldsymbol{u}_h &= \mathbf{0}, \\ \Omega\frac{d\boldsymbol{u}_h}{dt} &= -\mathsf{C}\left(\boldsymbol{u}_h\right)\boldsymbol{u}_h + \mathsf{D}\boldsymbol{u}_h + \boldsymbol{f}_h - \mathsf{M}^t\boldsymbol{p}_h\,. \end{aligned} \tag{3.10}$$

где $\boldsymbol{u}_h$ – дискретный вектор скорости, Ω – положительно определенная диагональная матрица, представляющая размеры контрольных объемов, $\mathsf{C}\left(\boldsymbol{u}_h\right)$ – кососимметричная матрица конвективных коэффициентов, дискретный оператор диффузии D — симметричная отрицательно определенная матрица, и M — дискретный оператор дивергенции. Такая дискретизация сохраняет свойства симметрии, лежащие в основе непрерывных дифференциальных операторов.

Эти глобальные свойства дискретных операторов в случае условия несжимаемости гарантируют стабильность и точное удовлетворение глобального баланса кинетической энергии даже для грубых сеток. Следовательно, кинетическая энергия не демпфируется систематически дискретным конвективным членом, и для того, чтобы гарантировать устойчивость метода, не требуется явное демпфирование. Это особенно важно, поскольку искусственная диссипация была бы помехой очень тонкому балансу между

конвективным переносом и физической диссипацией, особенно в наименьших масштабах движения. Дискретное уравнение переноса энергии также получено с использованием спектрально-согласованных схем.

3.1.3. Решение дискретного уравнения Пуассона

Дискретный оператор Лапласа в уравнении Пуассона (3.8) может быть рассмотрен как произведение дискретного оператора дивергенции M на дискретный оператор градиента, который представляет собой транспонированный дискретный оператор дивергенции, умноженный на диагональное масштабирование $\mathsf{G} = -\Omega^{-1}\mathsf{M}^t$. Таким образом, оператор Лапласа аппроксимируется матричным произведением $\mathsf{L} = -\mathsf{M}\Omega^{-1}\mathsf{M}^t$. Следовательно, дискретное уравнение Пуассона, которое необходимо решать на каждом шаге по времени, имеет форму

$$\mathsf{L}\boldsymbol{p}_h = \mathsf{M}\boldsymbol{u}_h^p \,. \tag{3.11}$$

Для решения уравнения Пуассона используется метод разложения Фурье в периодическом направлении. Это позволяет разделить исходное 3D уравнение на набор независимых 2D задач. Детали реализации данного метода описаны в [32, 33, 34].

3.2. Метод быстрого преобразования Фурье (БПФ)

Методы, основанные на дискретном преобразовании Фурье, хорошо известны и высокоэффективны для решения систем линейных уравнений рассматриваемого типа [60, 61]. Далее приводится описание метода для дискретизации 2-го порядка аппроксимации. Полностью для 4-го порядка

аппроксимации метод описан в [34]. Дискретизация уравнения Пуассона в 3D области может быть представлена в виде системы линейных уравнений:

$$\mathbf{A}^{3D}\mathbf{x}^{3D} = \mathbf{b}^{3D}\,, \tag{3.12}$$

где $\mathbf{A}^{3D} \in \mathbb{R}^{N\times N}$ — сингулярная симметричная матрица, $\mathbf{x}^{3D} \in \mathbb{R}^N$, $\mathbf{b}^{3D} \in \mathbb{R}^N$ и $N = N_x \times N_y \times N_z$ - число узлов в 3D дискретизации. Векторы $\mathbf{x}^{3D}$ и $\mathbf{b}^{3D}$ разбиты на N_yN_z подвекторов, каждый из которых состоит их N_x компонент:

$$\mathbf{x}^{3D} = \left[x_{1,1}, x_{2,1}, \cdots x_{j,k}, \cdots, x_{N_y,N_z}\right]^t\,, \tag{3.13}$$

где $x_{j,k} = [x_{1,j,k}, x_{2,j,k}, \cdots, x_{N_x,j,k}]^t$. С таким разбиением уравнение (3.12) может быть записано в блочной форме:

$$\begin{bmatrix}
A^p_{1,1} & A^n_{1,1} & \cdots & A^t_{1,1} & & & & & \\
A^s_{2,1} & A^p_{2,1} & A^n_{2,1} & \cdots & A^t_{2,1} & & & & \\
 & & \ddots & & & & & & \\
 & A^b_{j,k} & \cdots & A^s_{j,k} & A^p_{j,k} & A^n_{j,k} & \cdots & A^t_{j,k} & \\
 & & & & & \ddots & & & \\
 & & & & & A^b_{N_y,N_z} & \cdots & A^s_{N_y,N_z} & A^p_{N_y,N_z}
\end{bmatrix}
\begin{bmatrix} x_{1,1} \\ x_{2,1} \\ \vdots \\ x_{j,k} \\ \vdots \\ x_{N_y,N_z} \end{bmatrix}
=
\begin{bmatrix} b_{1,1} \\ b_{2,1} \\ \vdots \\ b_{j,k} \\ \vdots \\ b_{N_y,N_z} \end{bmatrix}, \tag{3.14}$$

где $A^n_{j,k}$, $A^s_{j,k}$, $A^t_{j,k}$ и $A^b_{j,k}$ — диагональные матрицы размера $N_x \times N_x$:

$$A^n_{j,k} = a^n_{j,k} I\,. \tag{3.15}$$

А матрицы $A^p_{j,k}$ — циклические [62] трехдиагональные матрицы вида

$$A^p_{j,k} = \begin{bmatrix}
\beta & \alpha & & & \alpha \\
\alpha & \beta & \alpha & & \\
 & & \ddots & & \\
\alpha & & & \alpha & \beta
\end{bmatrix}, \tag{3.16}$$

где $\alpha = a^w_{j,k} = a^e_{j,k}$ (поскольку шаг сетки постоянный по оси X), и $\beta = a^p_{j,k}$. Уравнение (3.14) может быть записано в виде

$$A^b_{j,k}\, x_{j,k-1} + A^s_{j,k}\, x_{j-1,k} + A^p_{j,k}\, x_{j,k} + A^n_{j,k}\, x_{j+1,k} + A^t_{j,k}\, x_{j,k+1} = b_{j,k}\,, \quad (3.17)$$

где индексы j и k имеют диапазон $1 \cdots N_y$ и $1 \cdots N_z$ соответственно (естественно, члены, соответствующие несуществующим блокам вектора вне домена, должны быть удалены).

Все циклические матрицы размера $N_x \times N_x$ (и, в частности, все $A^p_{j,k}$)) имеют одинаковый базис собственных векторов. Следуя обозначениям, принятым в [61], матрица Q — это матрица, у которой столбцы являются собственными векторами $A^p_{j,k}$. Произведение $x = Q\overline{x}$ — это обратное преобразование Фурье, которое может быть получена следующим образом:

$$x_i = \frac{1}{2}\overline{x}_1 + \sum_{\nu=1}^{\frac{N_x}{2}-1} \left(\overline{x}_{2\nu} \cos\left(\nu i \frac{2\pi}{N_x}\right) + \overline{x}_{2\nu+1} \sin\left(\nu i \frac{2\pi}{N_x}\right)\right) + \frac{1}{2}\overline{x}_{N_x}\,(-1)^i \quad i = 1, \ldots, N_x \quad (3.18)$$

и произведение $\overline{x} = Q^{-1}x$:

$$\begin{aligned}
\overline{x}_1 &= \frac{2}{N_x}\sum_{i=1}^{N_x} x_i\,, \\
\overline{x}_{2\nu} &= \frac{2}{N_x}\sum_{i=1}^{N_x} x_i \cos\left(\nu i \frac{2\pi}{N_x}\right) \qquad \nu = 1, \ldots, \frac{N_x}{2} - 1\,, \\
\overline{x}_{2\nu+1} &= \frac{2}{N_x}\sum_{i=1}^{N_x} x_i \sin\left(\nu i \frac{2\pi}{N_x}\right) \qquad \nu = 1, \ldots, \frac{N_x}{2} - 1\,, \\
\overline{x}_{N_x} &= \sum_{i=1}^{N_x} x_i\,(-1)^i\,. \qquad (3.19)
\end{aligned}$$

Поскольку все матрицы $A^p_{j,k}$ имеют одинаковые собственные векторы, то:

$$Q^{-1} A^p_{j,k} Q = \lambda_{j,k}\,, \quad (3.20)$$

где $\lambda_{j,k}$ — диагональная матрица, элементы которой зависят от α и β (и, следовательно, от j,k):

$$\begin{aligned} \lambda_1 &= \beta + 2\alpha\,, \\ \lambda_{2\nu} &= \lambda_{2\nu+1} = -4\alpha \sin^2\left(\frac{\nu\pi}{N_x}\right) + \beta + 2\alpha \qquad\qquad \nu = 1 \cdots \frac{N_x}{2} - 1\,, \\ \lambda_{N_x} &= \beta - 2\alpha\,. \end{aligned} \tag{3.21}$$

Предыдущие соотношения позволяют разложить систему уравнений с семидиагональной матрицей (3.12) в набор из N_x независимых систем уравнений с пятидиагональными матрицами. Для этого уравнение (3.17) домножается справа на Q^{-1} и подвекторы $x_{j,k}$ выражены как $Q\overline{x}_{j,k}$. После этих операций матрицы $A^p_{j,k}$ становятся диагональными, в то время как матрицы $A^{nb}_{j,k}$, каждая из которых есть единичная матрица, умноженная на скаляр, не подвергаются изменениям (то есть $Q^{-1}a^n_{j,k}IQ = a^n_{j,k}I$). В результате получается выражение:

$$A^b_{j,k}\,\overline{x}_{j,k-1} + A^s_{j,k}\,\overline{x}_{j-1,k} + \lambda_{j,k}\,\overline{x}_{j,k} + A^n_{j,k}\,\overline{x}_{j+1,k} + A^t_{j,k}\,\overline{x}_{j,k+1} = \overline{b}_{j,k}\,. \tag{3.22}$$

Это выражение, как и (3.17), является блочной пятидиагональной системой уравнений с N_yN_z неизвестными, каждое из которых имеет N_x компонент. Но в уравнении (3.22), поскольку элементы вне главной диагонали матриц $A^p_{j,k}$ были исключены, неизвестные $x_{i,j,k}$ с неизвестными только в той же плоскости i. Следовательно, выбирая для решения i-ю компоненту из всех N_yN_z блочных уравнений, получаем пятидиагональную систему линейных уравнений, которая может быть записана в виде

$$a^b_{j,k}\,\overline{x}_{i,j,k-1} + a^s_{j,k}\,\overline{x}_{i,j-1,k} + \overline{a}^p_{i,j,k}\,\overline{x}_{i,j,k} + a^n_{j,k}\,\overline{x}_{i,j+1,k} + a^t_{j,k}\,\overline{x}_{i,j,k+1} = \overline{b}_{i,j,k}\,, \tag{3.23}$$

где диагональный член — это i-й элемент диагонали матрицы $\lambda_{j,k}$,

$$\overline{a}^{p}_{i,j,k} = \lambda_{i,j,k} \,. \tag{3.24}$$

Уравнение (3.23) может быть записано более компактно:

$$\overline{A}_i \overline{x}_i = \overline{b}_i \qquad\qquad i = 1 \cdots N_x \,, \tag{3.25}$$

где $\overline{A}_i$ — пятидиагональная матрица, соответствующая преобразованному уравнению для i-й плоскости.

Таким образом, получен набор независимых систем уравнений, и трехмерная расчетная область разделена на независимые плоскости.

Преобразованная с помощью БПФ система (3.12) может быть записана в виде набора независимых систем:

$$\mathbf{A}_i^{2D} \mathbf{x}_i^{2D} = \mathbf{b}_i^{2D}, i = 1, .., N_x \,, \tag{3.26}$$

где $\mathbf{A}_i^{2D} \in \mathbb{R}^{N_{yz} \times N_{yz}}$, $\mathbf{x}_i^{2D} \in \mathbb{R}^{N_{yz}}$, $\mathbf{b}_i^{2D} \in \mathbb{R}^{N_{yz}}$, $N_{yz} = N_y \times N_z$ и N_x, N_y,N_z — число узлов по осям X, Y и Z соответственно. Таким образом, набор из $N_x \times N_y \times N_z$ узлов трехмерной сетки разделен на N_x независимых поднабора размером $N_y \times N_z$, которые будут называться плоскостями.

Когда решение системы (3.26) получено, обратное БПФ должно быть применено для получения решения исходной системы (3.12). Это приводит к трехшаговому алгоритму:

1. БПФ преобразует (3.12) к (3.26).

2. Решение (3.26).

3. Обратное БПФ преобразует решение (3.26) к решению (3.12).

Преобразования с использованием БПФ имеет низкую вычислительную стоимость и требует порядка $O(N \log_2(N_x))$ операций. Главная проблема для дальнейшего рассмотрения – это решение системы (3.26), которое требует намного больше вычислительных затрат.

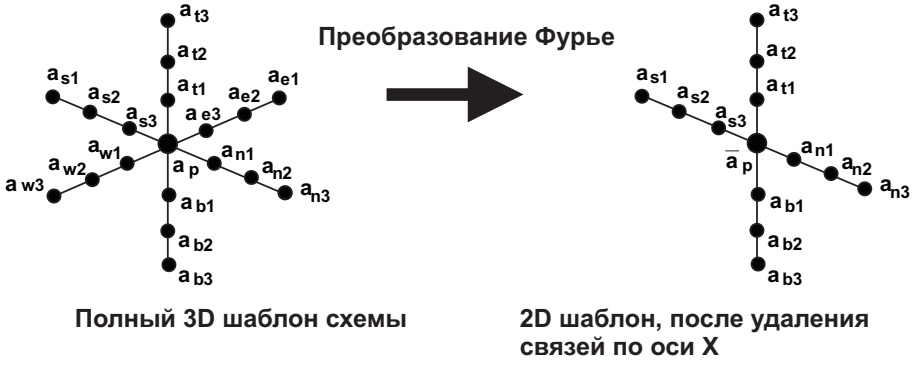

Рис. 3.1. Трансформация пространственного шаблона схемы с помощью преобразования Фурье.

На рисунке 3.1 в качестве примера показан пространственный шаблон схемы 4-го порядка. Каждый элемент шаблона соответствует диагонали в матрице $\mathbf{A}^{3D}$. Элементы a_{w_i} и a_{e_i} обеспечивают связи между узлами по оси X. БПФ изменяет главную диагональ матрицы $\mathbf{A}^{3D}$ и уничтожает эти элементы. После БПФ матрица $\mathbf{A}^{3D}$ может быть представлена в блочной форме, где каждый блок независим и может рассматриваться как одельная система уравнений. Детали этого метода описаны в [34, 63].

3.3. Метод дополнения Шура

В результате применения БПФ трехмерная задача распадается на N_x двумерных задач, то есть параллелепипед превращается в набор плоскостей. Поскольку плоскости независимы, достаточно рассмотреть нахождение решения только для одной из них. Матрица $\mathbf{A}$ (верхний и нижний

индексы могут быть отброшены) — это матрица системы линейных уравнений, соответствующей данной плоскости.

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,, \tag{3.27}$$

где $\mathbf{A} \in \mathbb{R}^{N_{yz} \times N_{yz}}$ и $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{N_{yz}}$.

Плоскость разделена на P подобластей, и метод Шура, как прямой параллельный метод, обеспечивает нахождение решения для всей плоскости. Каждый узел может быть либо внутренним, либо интерфейсным. Внутренний узел – это узел, который не связан напрямую ни с одним внутренним узлом из другой подобласти (под прямой связью одного узла с другим понимается следующее: если узел сделать центром шаблона разностной схемы, то все остальные узлы шаблона связаны напрямую с этим узлом). Если узел не внутренний, то он интерфейсный (см. Рис. 3.2).

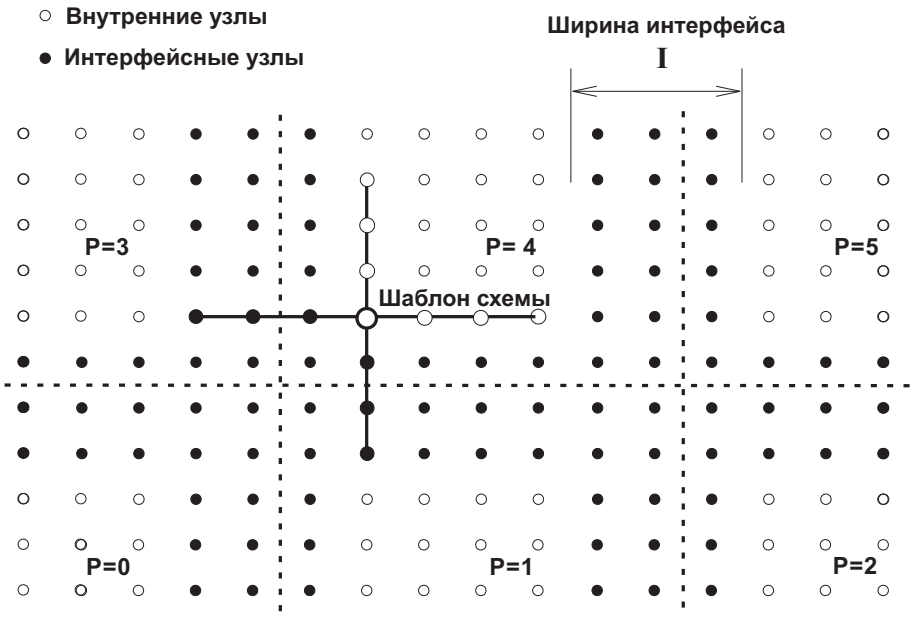

Рис. 3.2. Пример разбиения неизвестных на поднаборы

Таким образом, набор узлов плоскости разделен на $P + 1$ поднабора: первые P поднаборов соответствуют внутренним уздам P подобластей, и

последний поднабор содержит все интерфейсные узлы. Все узлы плоскости перенумерованы в соответствии с этим разделением: если два узла с номерами i_1 и i_2 принадлежат поднаборам P_1 и P_2 соответственно, и $P_1 < P_2$, то $i_1 < i_2$. С таким разделением и нумерацией система (3.27) может быть представлена в блочном виде:

$$\begin{bmatrix} \mathbf{A}_{1,1} & 0 & \cdots & 0 & \mathbf{A}_{1,s} \\ 0 & \mathbf{A}_{2,2} & \cdots & 0 & \mathbf{A}_{2,s} \\ \vdots & & & \vdots & \\ 0 & 0 & \cdots & \mathbf{A}_{P,P} & \mathbf{A}_{P,s} \\ \mathbf{A}_{s,0} & \mathbf{A}_{s,1} & \cdots & \mathbf{A}_{s,P} & \mathbf{A}_{s,s} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_P \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_P \\ \mathbf{b}_s \end{bmatrix}. \tag{3.28}$$

Сначала интерфейсные узлы изолируются с помощью блочного метода исключений Гаусса. Система (3.28) преобразовывается к виду (3.29).

$$\begin{bmatrix} \mathbf{A}_{1,1} & 0 & \cdots & 0 & \mathbf{A}_{1,s} \\ 0 & \mathbf{A}_{2,2} & \cdots & 0 & \mathbf{A}_{2,s} \\ \vdots & & & \vdots & \\ 0 & 0 & \cdots & \mathbf{A}_{P,P} & \mathbf{A}_{P,s} \\ 0 & 0 & \cdots & 0 & \tilde{\mathbf{A}}_{s,s} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_P \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_P \\ \tilde{\mathbf{b}}_s \end{bmatrix}. \tag{3.29}$$

Последнее блочное уравнение, включающее только неизвестные из $\mathbf{x}_s \in \mathbb{R}^{N_s}$ — интерфейсное уравнение

$$\tilde{\mathbf{A}}_{s,s}\mathbf{x}_s = \tilde{\mathbf{b}}_s \tag{3.30}$$

с преобразованной правой частью

$$\tilde{\mathbf{b}}_s = \mathbf{b}_s - \sum_{p=1}^{P} \mathbf{A}_{s,p}\mathbf{A}_{p,p}^{-1}\mathbf{b}_p \tag{3.31}$$

и матрица дополнения Шура

$$\tilde{\mathbf{A}}_{s,s} = \mathbf{A}_{s,s} - \sum_{p=1}^{P} \mathbf{A}_{s,p}\mathbf{A}_{p,p}^{-1}\mathbf{A}_{p,s}\,. \tag{3.32}$$

Таким образом, интерфейсное уравнение (3.30) может быть решено до того, как будут решены уравнения для внутренних узлов. Как только значения x_s становятся известны, каждый из x_p может быть получен независимо своим владельцем p, то есть процессором, которому принадлежит подобласть p, с помощью решения исходного уравнения.

$$\mathbf{A}_{p,p}\mathbf{x}_p = \mathbf{b}_p - \mathbf{A}_{p,s}x_s\,. \tag{3.33}$$

Наконец, следует отметить, что $\tilde{\mathbf{A}}_{s,s} \in \mathbb{R}^{N_s \times N_s}$ является плотной матрицей. Как матрица $\tilde{\mathbf{A}}_{s,s}$, так и вектор $\tilde{\mathbf{b}}_s$ могут быть получены без прямых вычислений $\mathbf{A}_{p,p}^{-1}$, как описано в [32].

Решение интерфейсного уравнения (3.30) является ключевым моментом, определяющим эффективность метода дополнений Шура. В случае несжимаемого течения матрица $\tilde{\mathbf{A}}_{s,s}$ постоянная на всех шагах по времени. Благодаря этому можно один раз найти обратную матрицу $\tilde{\mathbf{A}}_{s,s}^{-1}$ на этапе подготовки вычислений и в дальнейшем на всех шагах по времени явно получать решение интерфейсного уравнения:

$$\mathbf{x}_s = \tilde{\mathbf{A}}_{s,s}^{-1}\tilde{\mathbf{b}}_s\,. \tag{3.34}$$

Параллельная реализация блочного LU разложения для плотных матриц [34] используется для нахождения обратной матрицы. Также используется распределенное хранение данных для обратной матрицы $\tilde{\mathbf{A}}_{s,s}^{-1}$. В итоге параллельная реализация метода Шура имеет следующий алгоритм решения системы $\mathbf{A}\mathbf{x} = \mathbf{b}$:

1) решение $\mathbf{A}_{p,p}\mathbf{t} = \mathbf{b}_p$;

2) нахождение локального вклада в правую часть интерфейсного уравнения $\tilde{\mathbf{b}}_s^p = = \mathbf{A}_{s,p}t$;

3) глобальное суммирование $\mathbf{t} = \sum_{p=1}^{P} \tilde{\mathbf{b}}_s^p$;

4) нахождение правой части интерфейсного уравнения, $\tilde{\mathbf{b}}_s = \mathbf{b}_s - \mathbf{t}$;

5) нахождение решения для необходимых интерфейсных узлов $\mathbf{x}_s = \tilde{\mathbf{A}}_{s,s}^{-1}\tilde{\mathbf{b}}_s$;

6) локальное нахождение решения для внутренних узлов $\mathbf{A}_{p,p}\mathbf{x}_p = \mathbf{b}_p - \mathbf{A}_{p,s}\mathbf{x}_s$.

Вектор $\mathbf{t}$ используется для обозначения временной области хранения данных. Глобальный обмен данными требуется на шаге 3.

3.4. Ограничения применимости метода Шура

Эффективная реализация этого метода для малобюджетных кластеров была предложена в [34]. Алгоритм требует только одного обмена данными для нахождения решения, позволяя получить хорошую параллельную эффективность на системах с высокой латентностью сети и со сравнительно небольшим числом процессоров. Но с ростом числа процессоров

и размеров сетки возникают различные проблемы, затрудняющие масштабируемость.

3.4.1. Ограничения по объему памяти

Оперативная память, требуемая каждому процессору для одной плоскости, может быть приближенно выражена функцией от числа процессоров P, числа неизвестных в плоскости $N_{yz} = N_y N_z$ и размера интерфейса I. Поскольку шаблон схемы симметричный, то размер шаблона $M = 4I + 1$, где $4I$ соответствует правой левой, верхней, нижней частям шаблона, и еще один узел — центральный. Для упрощения рассматривается следующий случай: $N_y = N_z = \sqrt{N_{yz}}$, $P_y = P_z = \sqrt{P}$ и P делитель N_{yz}. Таким образом, подобласти, соответствующие каждому процессору, квадратные $\sqrt{N_{yz}/P} \times \sqrt{N_{yz}/P}$. Пренебрегая тем, что $I\left(\sqrt{P}-1\right)^2$ узлов, лежащих на пересечении интерфейсов, посчитаны дважды, число интерфейсных узлов будет $N_s^{2d} = 2I\sqrt{N_{yz}}(\sqrt{P}-1)$. Максимальное число интерфейсных узлов, приходящееся на каждый процессор равно $2I\sqrt{N_{yz}/P}$ и локальное LU разложение для ленточной матрицы требует хранение $3(4I+1)(N_{yz}/P)^{3/2}$ чисел. Таким образом, суммарное количество чисел в памяти каждого процессора приблизительно следующее:

$$N_t^{2d}(I, N_{yz}, P) \approx 3(4I+1)\left(\frac{N_{yz}}{P}\right)^{\frac{3}{2}} + 4I^2 N_{yz}\frac{P^{1/2}-1}{P^{1/2}}\,. \tag{3.35}$$

Предполагая, что значения N и P достаточно велики, а I — константа, [3.35] может быть записано в виде

$$N_t^{2d}(N_{yz}, P) = O\left(\left(\frac{N_{yz}}{P}\right)^{\frac{3}{2}}\right) + O\left(N_{yz}\right)\,. \tag{3.36}$$

Выражение (3.36) показывает объем памяти, требуемый только для одной плоскости на каждом процессоре. Для получения полного объема памяти, это значение необходимо умножить на N_x.

Первый член (3.36) при заданном N ограничивает число процессоров снизу. Если число процессоров слишком мало, данные не поместятся в памяти. Но если число процессоров достаточно велико, второй член становится главным ограничением.

Второй член (3.36) ограничивает масштабируемость: объем памяти, требуемый на каждом процессоре, не зависит от числа процессоров. Таким образом, использование параллельного компьютера с любым числом процессоров будет строго ограничено объемом памяти одного процессора. Из-за этого метод практически неприменим при достаточно большом числе процессоров.

Еще одна трудность заключается в том, что I возрастает с ростом порядка схемы o[1] и требуемый объем памяти растет квадратично с ростом I. Это особенно сильно ограничивает применимость метода Фурье–Шура в случае схемы высокого порядка.

3.4.2. Ограничения из-за обмена данными

Ограничение памяти — не единственная проблема метода. Даже предполагая, что эта проблема каким-то образом решена, все равно метод будет плохо масштабируемым. Методу Шура требуется один, но большой обмен данными, а именно глобальная операция суммирования значений для всех интерфейсных узлов, необходимая на шаге 3 алгоритма. Как было пока-

[1] В частности, для рассматриваемого семейства схем $I = o - 1$, где o — порядок схемы.

зано выше, число интерфейсных узлов в каждой плоскости $O(\sqrt{N_{yz}P})$. Поскольку число пересылок в одной глобальной операции суммирования оценивается с ростом числа процессоров P как $O(\log_2(P))$, общий объем обмена данными можно оценить как

$$CS = O(N_x\sqrt{N_{yz}P}\log_2(P))\,. \tag{3.37}$$

Оценка (3.37) показывает, что объем обмена данными для каждого процессора растет не только с ростом числа узлов, но и, что гораздо хуже, с ростом числа процессоров. Когда число процессоров небольшое, эта операция не потребляет много времени. Но с ростом числа процессоров, время, затрачиваемое на обмен данными, растет достаточно быстро. Эта проблема существенно сказывается на производительности, и метод Фурье-Шура не может эффективно использоваться на большом числе процессоров, даже если пренебречь ограничением памяти.

3.4.3. Вычислительные ограничения

Еще одна трудность — это рост вычислительной стоимости. Здесь сразу два источника проблем. Первый — это матрично-векторное произведение $x_s = \tilde{\mathbf{A}}_{s,s}^{-1}\tilde{\mathbf{b}}_s$ на шаге 5 алгоритма, которое требует $O(N)$ операций на каждом процессоре, независимо от числа процессоров. Это эквивалентно операции с линейным ростом (от числа узлов) вычислительной стоимости, которая не может быть распараллелена. Наличие такой не распараллеливаемой операции существенно сказывается на эффективности при большом числе процессоров.

Но еще более серьезная проблема — это этап подготовки вычислений, на котором требуется найти обратную матрицу $\tilde{\mathbf{A}}_{s,s}^{-1}$. Вычислительная

стоимость обращения $\tilde{\mathbf{A}}_{s,s}$ растет очень быстро с ростом числа интерфейсных узлов. Это делает метод неприменимым к большим сеткам, которые естественно требуют большого числа процессоров, поскольку подготовительный этап может занять сравнимое или даже большее время, чем сам расчет. Все эти ограничения заставляют использовать другой подход для параллельных систем со сравнительно большим числом процессоров.

3.5. Использование итерационного метода

Если число процессоров или размер сетки слишком большие для метода Шура, то он не может быть использован эффективно для нахождения решения целиком по всей плоскости. Но в этом случае он может быть использован только для некоторых плоскостей, наиболее проблематичных для итерационного метода. Решение на всех остальных плоскостях находится с помощью итерационного метода. Метод Шура также может использоваться как предобуславливатель в составе итерационного метода. Далее в этой главе предлагается **новый метод Крылова-Фурье-Шура**, использующий также итерационный метод на основе подпространств Крылова.

3.5.1. Алгоритм

Метод сопряженных градиентов [64] был выбран как наиболее подходящий для такого типа симметричных положительно определенных матриц (одна из матриц изначально сингулярна, но эта проблема легко устранима, что будет показано далее) . Поскольку все плоскости независимые, будет рассмотрено решение только для одной плоскости. Система, которую

требуется решить — $\mathbf{Ax} = \mathbf{b}$. В составе итерационного метода используется дополнительный прямой метод или, другими словами, предобуславливатель (простейший и в то же время достаточно эффективный). Плоскость разделена на части, которые далее будут называться **блоками**. Локальное решение для каждого блока находится с помощью прямого метода, то есть не учитываются связи между узлами из разных блоков. В свою очередь, итерационный метод обеспечивает связи между всеми узлами.

Таким образом, матрица $\mathbf{A}$ разделена на прямую и итерационную части:

$$\mathbf{A} = \mathbf{A}_D + \mathbf{A}_I \,. \tag{3.38}$$

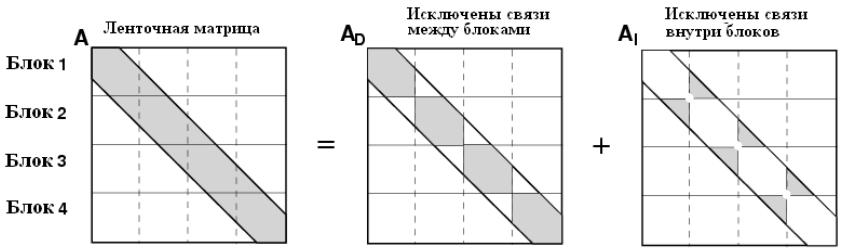

Рис. 3.3. Разделение на прямую и итерационную части.

Каждому узлу на плоскости соответствует строка в матрице $\mathbf{A}$, а также каждый элемент строки соответствует определенному узлу. Если в i-й строке матрицы $\mathbf{A}$ j-й элемент не равен нулю, это означает, что узел с номером i связан напрямую с j-м узлом. Исключить связь между узлами i и j означает приравнять нулю j-й элемент i-й строки и i-й элемент j-й строки. Матрица $\mathbf{A}_D$ получена из матрицы $\mathbf{A}$ исключением связей между узлами из разных блоков. Матрица $\mathbf{A}_I$ получена из $\mathbf{A}$ исключением связей между узлами, принадлежащими одному блоку, а так же исключением диагональных элементов. На Рис. 3.3 показана структура матриц $\mathbf{A}_D$ и $\mathbf{A}_I$

(для случая простейшего разбиения плоскости на блоки по одной оси).

Итерационный метод на каждой итерации содержит обращение к предобуславливателю или, другими словами, дополнительному прямому методу $\mathbf{d}(\mathbf{A}_D, \mathbf{b})$. Этот предобуславливатель находит решение для системы $\mathbf{A}_D\mathbf{x} = \mathbf{b}$.

Метод сопряженных градиентов имеет следующий алгоритм.

Первая итерация:

$\mathbf{r}^0 = b - \mathbf{A}\mathbf{x}^0$.

$\mathbf{z}^0 = D(\mathbf{A}_D, \mathbf{r}^0)$.

$\mathbf{p}^1 = \mathbf{z}^0$.

i-я итерация:

1) $\mathbf{z}^{i-1} = \mathbf{d}(\mathbf{A}_D, \mathbf{r}^{i-1})$.

2)* $\rho_{i-1} = \mathbf{r}^{i-1^T} \cdot \mathbf{z}^{i-1}$.

3) $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$.

4) $\mathbf{p}^i = \mathbf{z}^{i-1} + \beta_{i-1}\mathbf{p}^{i-1}$.

5) Обновить значения в гало[2] вектора $\mathbf{p}^i$.

6) $\mathbf{q}^i = \mathbf{A}\mathbf{p}^i$.

7)* $\alpha_i = \rho_{i-1}/(\mathbf{p}^{i^T} \cdot \mathbf{q}^i)$.

8) $\mathbf{x}^i = \mathbf{x}^{i-1} + \alpha_i\mathbf{p}^i$.

9) $\mathbf{r}^i = \mathbf{r}^{i-1} - \alpha_i\mathbf{q}^i$.

10)* Нахождение нормы невязки .

* Коллективный обмен данными для глобальных операций:

[2]Гало - узлы из соседних подобластей, связанные с узлами подобласти данного процессора

– Суммирование одного значения: шаги 2), 7).

– Нахождение максимума для одного значения: шаг 10).

3.5.2. Свойства матриц $\mathbf{A}_i^{2D}$, определяющие сходимость итерационного метода

Набор систем (3.26) имеет одно важное свойство, влияющее на применение итерационного метода. Сходимость существенно различается для разных систем из набора, поскольку системы имеют разные числа обусловленности матриц. В частности, одна матрица особенно проблематична для итерационного метода. В основе такого различия лежат специфические свойства БПФ. Для удобства, системы уравнений и соответствующие им плоскости упорядочены по убыванию числа обусловленности матриц $\mathbf{A}_i^{2D}$. Таким образом, итерационному методу требуется больше итераций для нахождения решения на плоскости с меньшим номером. При этом первая плоскость представляет особенные трудности. Изначально матрица дискретного уравнения Пуассона сингулярна, но после БПФ сингулярность сохраняет только матрица одной плоскости. Сингулярность этой матрицы устраняется изменением одного элемента на главной диагонали (например – увеличить значение в 2 раза), который соответствует какому-то внутреннему узлу домена. Эта модификация фиксирует значение в этом узле равным нулю, делая решение единственным. При этом, решение удовлетворяет исходной системе.

Характерное распределение числа итераций, необходимого для нахождения решения на плоски, было продемонстрировано на примере DNS 3D турбулентного конвективного течения в каверне с разными темпера-

турами на стенках, с соотношением длин сторон 1 к 4 и $Ra = 10^9$ (подробное описание постановки задачи приводится в 4-й главе). Начальное приближение для всех систем – нулевой вектор. Критерий для невязки решения каждой системы $||\mathbf{r}_i||_\infty < \varepsilon \left|\left|\mathbf{r}_i^0\right|\right|_\infty, i = 1, .., N_x$ (*норма* $||\cdot||_\infty$ *для* $\boldsymbol{x} \in \mathbb{R}^n$ *определена как* $||\boldsymbol{x}||_\infty = \max_{i=1,..,n} |x_i|$), где $\mathbf{r}_i^0$ – начальная невязка, $\varepsilon = 0.01$. В данном тесте каждая плоскость разбита на 32 блока, и для каждого блока находится локальное решение LU методом. Число итераций, необходимое для получения решения на каждой плоскости, осреднено по 1000 шагам по времени.

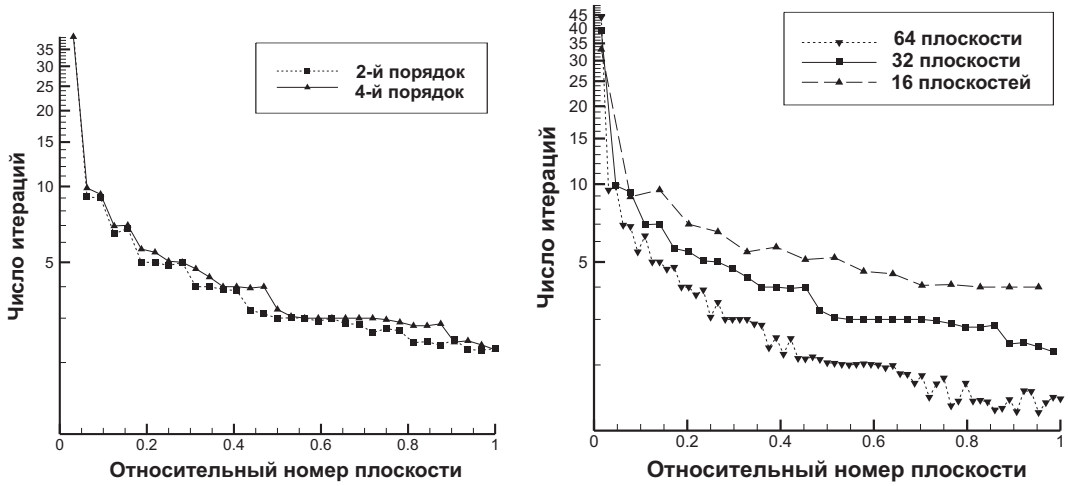

Рис. 3.4. Число итераций, необходимое для нахождения решения на различных плоскостях: сравнение для схем 2-го и 4-го порядка аппроксимации (слева), и для различного числа плоскостей (справа)

На Рис. 3.4 слева показано сравнение числа итераций для схем 2-го и 4-го порядка. Результаты показывают, во-первых, что порядок схемы практически не влияет на сходимость, а во-вторых, что имеется большое различие в числе итераций, требуемом для нахождения решения на разных плоскостях.

Также следует отметить, что с ростом числа плоскостей, улучшаются свойства сходимости — для нахождения решения на всех плоскостях кроме первой требуется меньшее число итераций. Сравнительные тесты для сеток 16×78×156, 32×78×156, 64×78×156 показывает эту особенность. Эти 3 теста имеют одни и те же описанные ранее параметры, за исключением числа узлов по оси X, которое равно 16, 32 и 64 соответственно. На Рис. 3.4 справа ось X — это относительный номер плоскости $i_{rel} = i/N_x$, где i — номер плоскости и N_x — число плоскостей. Результаты показывают, что чем больше число плоскостей, тем меньше среднее по всем плоскостям число итераций. Для 16 плоскостей среднее число итераций в данном примере 7.24, для 32 — 5.20 и для 64 — 3.46. Таким образом, показана важная особенность: число узлов возрастает, но, при этом, число итераций уменьшается. Это свойство вносит вклад в хорошую масштабируемость метода. Например, при пропорциональном увеличении сетки в тех же условиях с 16×39×78 до 32×78×156 среднее число итераций падает с 5.75 до 5.2. **Вторая сетка имеет в 8 раз большее число узлов, при этом требуемое число итераций даже меньше, чем для первой сетки.**

3.5.3. Выбор критерия для невязки

Как только задействован итерационный метод, необходимо определить критерий завершения итерационного процесса. Выбор критерия особенно важен в точном по времени моделировании. В этом случае, пока не выполнены все шаги по времени, требуемые для получения результата, не известно, был ли критерий достаточно жестким для достижения необходимой точности. Поэтому должен быть выбран некоторый универсальный

безразмерный критерий, не зависящий от сетки, который после проверки на какой-то небольшой тестовой задаче гарантировал бы требуемую точность и для других вычислений. Проверка заключается в сравнении результатов, полученных с использованием итерационного метода, с результатами прямого метода.

Анализируя результаты выполненных DNS тестов с различными сетками было найдено подходящее решение — фиксировать величину уменьшения начальной нормы дивергенции поля скоростей $||\mathsf{M}\boldsymbol{u}_h^p||_\infty$ [3]. Эта норма есть норма правой части дискретного уравнения Пуассона (3.11). Задача итерационного метода — обеспечить такую точность решения, чтобы норма дивергенции поля скоростей после коррекции $\left|\left|\mathsf{M}\boldsymbol{u}_h^{n+1}\right|\right|_\infty$ была меньше начальной нормы как минимум в заданное число раз:

$$\frac{\left|\left|\mathsf{M}\boldsymbol{u}_h^{n+1}\right|\right|_\infty}{||\mathsf{M}\boldsymbol{u}_h^p||_\infty} \leq \varepsilon\,. \tag{3.39}$$

В [34] было показано, что норма невязки решения дискретного уравнения Пуассона равна норме дивергенции поля скоростей, получаемого после коррекции: $||\boldsymbol{r}_h||_\infty = = \left|\left|\mathsf{M}\boldsymbol{u}_h^{n+1}\right|\right|_\infty$. Таким образом, окончательно критерий для невязки решения дискретного уравнения Пуассона имеет следующий вид:

$$||\boldsymbol{r}_h||_\infty \leq \varepsilon||\mathsf{M}\boldsymbol{u}_h^p||_\infty\,. \tag{3.40}$$

Фиксирование ε таким способом позволяет получать решение требуемой точности для различных сеток. [4]

[3] Норма $||\cdot||_\infty$ для $\boldsymbol{x} \in \mathbb{R}^n$ определена как $||\boldsymbol{x}||_\infty = \max_{i=1,..,n}|x_i|$

[4] Норма дивергенции вычисляется после обратного БПФ, когда набор плоскостей трансформирован в трехмерную область. Поэтому для итерационного процесса используется другой критерий - это обеспечение нормы невязки меньше величины $\bar{\varepsilon}$, одинаковой для всех плоскостей. Эта величина устанавливается автоматически в процессе вычислений: если после коррекции поля скоростей норма дивергенции превышает ε, значение $\bar{\varepsilon}$ уменьшается в 2 раза и решение для уравнения Пуассона находится заново. Таким образом, по критерию для нормы дивергенции на первом шаге по времени определяется критерий для итерационного процесса, который также может корректироваться в процессе счета.

Для верификации критерия и определения величины ε была выполнена серия 2D и 3D расчетов на различных сетках. Требовалось, во-первых, определить, какое значение ε позволяет получать решение требуемой точности, и во-вторых, то, что это значение одинаково во всех случаях. Тестовые расчеты представляют собой серии DNS на небольших сетках для задачи турбулентного конвекционного течения в каверне с разными температурами на стенках, с соотношением длин сторон 1 к 4. Каждая серия включает два расчета прямым методом и 6 расчетов с использованием итерационного метода с величинами ε, равными 0.7, 0.3, 0.1, 0.03, 0.01, 0.001. Параметры тестов представлены в табл.3.1.

Таблица 3.1. Параметры тестов

Тест	Параметры сетки	Порядок схемы	Ra	Pr
CC	2D 39x78	2-й	10^9	0.71
BB	2D 78x156	2-й	10^9	0.71
C	3D 16x39x78	4-й	10^9	0.71
B	3D 32x78x156	4-й	10^9	0.71

Сравниваются средние поля течений, полученные прямым методом и итерационным с различными ε. Поскольку среднее поле турбулентного течения – это результат осреднения случайного процесса, то требуется бесконечный период осреднения для получения одинаковых результатов для двух одинаковых случайных процессов. Следовательно, различие в результатах может быть по двум причинам: во-первых, из-за недостаточной точности решения уравнения Пуассона и, во-вторых, из-за конечности периода осреднения. Для определения допустимого уровня различий в результатах, надо найти вклад второй причины. Для этого выполняются два DNS с разным начальным случайным распределением температуры, с использовани-

ем прямого метода. Берут достаточно длинный период осреднения (50-100 периодов низких частот). Различия между результатами этих двух DNS задают допустимый диапазон различий между результатами, полученными с использованием прямого и итерационного метода. Тесты показали, что уменьшение итерационным решателем нормы начальной дивергенции в 10 раз является достаточным. Но это значение $\varepsilon = 0.1$ верно только для данного явного алгоритма моделирования несжимаемых течений и может отличаться для других алгоритмов.

3.5.4. Начальное приближение для итерационного метода

Начальное приближение $\mathbf{x}_i^0$ находится с помощью линейной экстраполяции, с использованием решения на двух предыдущих шагах по времени: $\mathbf{x}_i^0 = 2\mathbf{x}_{i-1} - \mathbf{x}_{i-2}$. Этот способ позволяет существенно уменьшить норму начальной невязки. Такой выбор начального приближения имеет важную особенность – число итераций почти не растет при уменьшении ε. Рост числа итераций начинается только после того, как ε становится меньше некоторого значения. В данном случае это значение менее 10^{-4}, при том, что значение $\varepsilon = 0.1$ достаточно для достижения требуемой точности результатов. При таком выборе начального приближения, решение уравнения Пуассона может быть получено на несколько порядков точнее, при той же вычислительной стоимости, поскольку нет разницы, уменьшать начальную дивергенцию в 10 или в 10000 раз (**при одинаковой вычислительно стоимости целесообразнее использовать более высокую точность**). Такой эффект имеет простое объяснение. Чем точнее решается уравнение Пуассона, тем **больше** для этого требуется итераций. Но чем

точнее решение на данном шаге по времени, тем точнее решение предсказывается экстраполяцией на следующем шаге, следовательно, тем **меньше** потребуется итераций. Эти два свойства компенсируют друг друга, поэтому число итераций изменяется незначительно. Конечно, даже при точном решении уравнения Пуассона, экстраполяция на следующий шаг по времени имеет погрешность. Эта погрешность и определяет то значение ε, после которого начинается рост числа итераций. Рис. 3.5 показывает изменение числа итераций с изменением точности решения уравнения Пуассона.

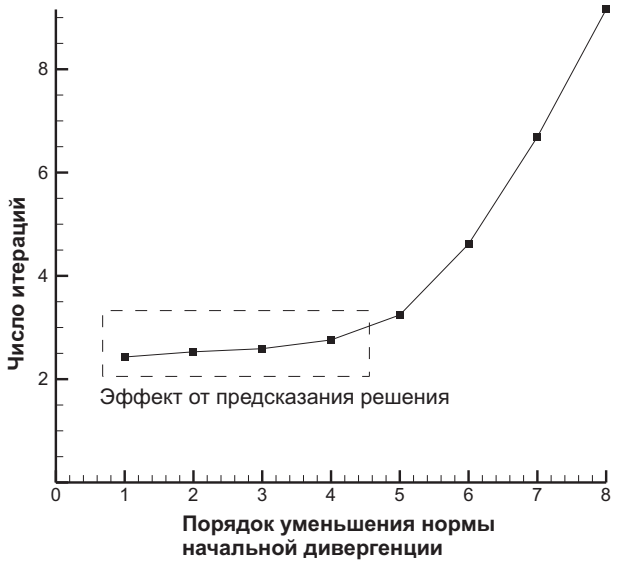

Рис. 3.5. Число итераций для различного порядка ε (тест B в таблице 3.1). На оси X величина $x = -\lg(\varepsilon)$

3.6. Масштабируемая конфигурация метода Крылова-Фурье-Шура

В соответствии с нумерацией плоскостей, в равных условиях для нахождения решения на плоскости с меньшим номером требуется больше итераций, а для первой плоскости требуется особенно много итераций. Следуя

этому ключевому свойству, различные методы могут быть применены для получения решения на различных плоскостях. Каждая плоскость может иметь свой размер блоков, для достижения максимальной производительности.

Таким образом, для каждой плоскости задается число блоков $S_i = N_b/N_{yz}, i = 1, .., N_x$, где N_b — число узлов в блоке и N_{yz} — число узлов в плоскости. S_i может изменяться от 1 до N_{yz}. $S_i = 1$ соответствует решению прямым методом за одну итерацию, $S_i = N_{yz}$ означает решение с помощью итерационного метода без дополнительного прямого метода (используя простейший предобуславливатель Якоби).

Число блоков определяет число итераций: чем больше число блоков, тем больше требуется число итераций. Матрица предобуславливателя $\mathbf{A}_D$ из разложения (3.38) получается из матрицы системы $\mathbf{A}$ исключением элементов, соответствующих прямым связям между узлами из разных блоков. При росте числа блоков, растет число этих связей, и, следовательно, матрица $\mathbf{A}_D$ становится все более грубым приближением матрицы $\mathbf{A}$. Поэтому число итераций увеличивается. Но, чем меньше размер блока, тем меньше элементов содержится в матрице $\mathbf{A}_D$, и, следовательно, тем меньше вычислительная стоимость для нахождения локального решения для блоков прямым методом. Вычислительная стоимость и потребление памяти для нахождения локального решения для каждого блока растут в случае LU метода для такой ленточной матрицы нелинейно как $O(N_b^{1.5})$, и, уменьшая размер блока, можно повысить производительность, несмотря на рост числа итераций. Наибольший выигрыш в производительности можно получить таким способом для последних плоскостей, поскольку число итераций

растет незначительно с ростом числа блоков.

Число блоков может быть выбрано независимо для каждой плоскости. Критерий выбора, естественно, минимизация затрат времени для выполнения всех требуемых итераций. Для этого также следует учитывать обмены данными, которые необходимы на каждой итерации. Этот способ позволяет получить хорошую параллельную эффективность как на малобюджетных кластерах, так и на суперкомпьютерах, поскольку предоставляется возможность управлять балансом между объемом вычислений и объемом обмена данными (суммарный объем обмена данными напрямую зависит от числа итераций).

3.6.1. Выбор прямого метода в зависимости от размера блока

Когда метод Крылова–Фурье–Шура применяется на параллельной системе с P процессорами, каждая плоскость соответственно разбита на P подобластей. Возможны два варианта:

1. **Размер блока больше чем размер подобласти.** В этом случае число блоков должно удовлетворять условию $1 \leq S_i < P$, и число $C_i = P/S_i$ имеет целочисленное значение — число подобластей в одном блоке. Чтобы найти решение для каждого блока, должен быть использован метод Шура, который применяется к блоку, как если бы он был целой плоскостью, состоящей из N_b узлов и разбитой на C_i подобластей. Следует отметить, что метод Шура имеет вычислительную стоимость как минимум двух LU. Поэтому использование блоков большего размера чем подобласти эффективно только в случае, если оно приводит к сокращению числа итераций более чем в 2 раза по

сравнению с блоками размером с подобласть. Конечно же, наиболее эффективно метод Шура может быть применен ко всей плоскости целиком, поскольку сокращает число итераций до 1.

2. **Размер блока меньше или равен размеру подобласти.**

   В этом случае число блоков должно удовлетворять $P \leq S_i N_{yz}$. И S_i/P имеет целочисленное значение — число блоков в одной подобласти. LU метод может быть использован в этом случае для прямого нахождения решения для каждого блока. Метод Холецкого мог бы быть использован вместо LU, так как матрицы $\mathbf{A}_{D_i}$ симметричные. Этот метод требует в 2 раза меньше памяти, так как одна и та же область памяти может быть использована для хранения обеих матриц H_i и H_i^T разложения $\mathbf{A}_{D_i} = H_i H_i^T$ вместо L и U матриц. Но из-за специфических свойств доступа к компьютерной памяти, метод Холецкого, при примерно той же вычислительной стоимости, работает значительно медленнее.

В качестве примера далее приводятся несколько типовых конфигураций решателя. Конфигурации пронумерованы по возрастанию масштаба задачи: больший номер соответствует большему размеру сетки и числу процессоров.

1. Решение на каждой плоскости находится с помощью метода Шура. Эта конфигурация применима на системах с небольшим числом процессоров, например от 1 до 25. Естественно, размер сетки должен быть таким, чтобы методу Шура хватило памяти.

2. Решение на нескольких первых плоскостях находится с помощью метода Шура. На всех остальных плоскостях решение находится итерационным методом, размер блока меньше или равен размеру подобласти. Эта конфигурация может быть использована на более крупных системах, с числом процессоров, например, до 64. На таком числе процессоров итерационный метод может быть более эффективен для большинства плоскостей, чем метод Шура. Эта конфигурация также может быть использована на системах с меньшим числом процессоров, как в первом случае, если методу Шура не достаточно памяти для нахождения решения на всех плоскостях.

3. Решение только на первой плоскости находится с помощью метода Шура. На всех остальных плоскостях решение находится итерационным методом, размер блока меньше или равен размеру подобласти. Эта конфигурация может быть использована на пределе возможностей метода Шура, на системах с числом процессоров до 2-х - 3-х сотен. В этом случае применение метода Шура даже для одной плоскости ограничено в основном этапом подготовки вычислений. Поскольку первая плоскость особенно проблематична для итерационного метода, то использование метода Шура для нахождения решения только на этой плоскости при таком числе процессоров позволяет существенно повысить производительность. При достаточно большом размере сетки это приводит к сокращению до 2-х (в частности, для сетки с 10^8 узлов на 200 процессорах) и более раз времени, затрачиваемого на нахождение решения на всех плоскостях.

3.6.2. Производительность на параллельных системах

В качестве теста используется уже упомянутый расчет 3D турбулентного течения (подробно задача описана в 4 главе). Критерий завершения итерационного процесса — уменьшение начальной дивергенции в 10000 раз. Две существенно различные параллельные системы использовались для расчетов:

- Суперкомпьютер Marenostrum Барселонского Суперкомпьютерного Центра

  Это IBM BladeCenter JS20 кластер с 4800 процессорами PowerPC 970FX 2.2 GHz (конфигурация указана на момент вычислений) . Двухпроцессорные узлы с 4 Gb RAM памяти связаны высокопроизводительной сетью Myrinet. Дополнительная сеть Ehernet 1Gbit используется для распределенной файловой системы.

- Кластер JFF Технологического центра исследования явлений переноса тепла и массы при Политехническом университете Каталонии. Это типичный малобюджетный кластер на основе офисного компьютерного оборудования. 40 однопроцессорных узлов с процессорами AMD Athlon 2.6GHz и 1Gb RAM памяти связаны сетью Ethernet 100Mbit.

Основное отличие между этими системами, помимо числа процессоров – это производительность сети. Сеть Myrinet имеет намного меньшую латентность и во много раз большую пропускную способность, чем 100Mbit Ethernet.

3.6.3. Выбор конфигурации метода

Размер сетки в данном тесте $64 \times 240 \times 460$ (7×10^6 узлов). Тест выполнен на 32 процессорах, расчетная область разбита на 32 подобласти 4×8 по осям Y и Z соответственно. Тест показывает, как метод Крылова–Фурье–Шура может быть адаптирован к различным параллельными системам для достижения максимальной производительности.

Были опробованы 4 базисные конфигурации, имеющие одинаковые параметры для всех плоскостей. Конфигурация 1 — размер блока равен размеру плоскости, то есть решение методом Шура. Конфигурации 2, 3, 4 используют итерационный метод с числом блоков в каждой подобласти 1, 4, 16 соответственно. Для каждой базисной конфигурации измерялось время, затрачиваемое на получение решения для всех плоскостей. Число итераций для каждой плоскости осреднялось по 200 шагам по времени. Поскольку в каждой конфигурации вычислительная стоимость итерации одинакова для всех плоскостей, время, затрачиваемое на получение решения на каждой плоскости, считается пропорциональным числу итераций: $t_i = T(I_i/I_{sum}), i = 1, .., N_x$, где I_i – число итераций для i-й плоскости и $I_{sum} = \sum_{i=1}^{N_x} I_i$ суммарное число итераций для всех плоскостей. Для конфигурации 1, которая использует только прямой метод, число итераций равно единице. Также следует отметить, что конфигурация 1 не может полностью поместиться в память из-за ограничений метода Шура. Эта конфигурация была смоделирована использованием одного и того же набора данных для всех плоскостей. Таким образом, вычислительная стоимость в точности такая же, как если бы методу Шура хватило памяти, хотя получаемое реше-

ние, конечно же, не имеет смысла. Сравнение времени, затрачиваемого на каждую плоскость, позволяет составить на основе базисных конфигураций одну оптимизированную конфигурацию, беря для каждой плоскости лучшую из этих 4-х. На Рис. 3.6 показано сравнение для обеих параллельных систем. Применение итерационного метода для получения решения на первой плоскости вообще не рассматривается, поскольку потребуется слишком большое число итераций.

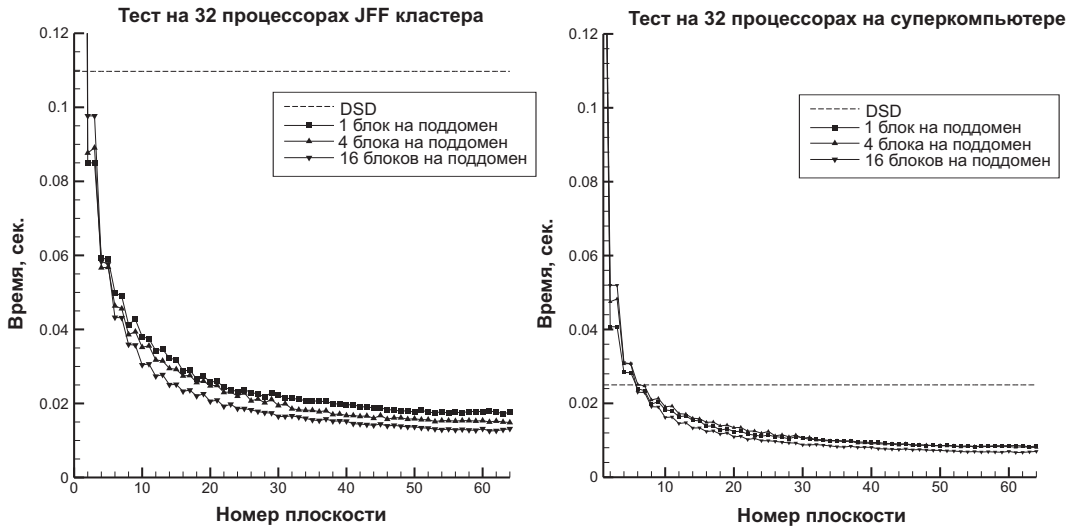

Рис. 3.6. Сравнение конфигураций решателя для различных параллельных систем: кластер JFF слева и суперкомпьютер Marenostrum справа.

Время, затрачиваемое на получения решения на каждой плоскости, показано для каждой базисной конфигурации на Рис.3.6. Это позволяет построить оптимизированную конфигурацию, выбирая для каждой плоскости лучшую из рассматриваемых конфигураций. Используя эти результаты измерений, можно получить следующие оптимизированные конфигурации:

1. Для кластера JFF: Решение на первой плоскости находится целиком методом Шура, на плоскости 2 и 3 – итерационным методом с бло-

ком размером с подобласть, на остальных плоскостях итерационным методом с 16 блоками на подобласть.

2. Для Marenostrum: Решение на первых 5-ти плоскостях находится целиком методом Шура, на остальных плоскостях – итерационным методом с 16 блоками на подобласть.

Следует отметить, что метод Шура в этом случае работает в 4 раза медленнее на малобюджетном кластере, чем на суперкомпьютере, в то время как общая производительность оптимизированной конфигурации отличается только в 2 раза. Это связано с существенным преимуществом Marenostrum в пропускной способности сети, которая особенно важна для операции глобального суммирования большого объема данных в методе Шура. Поэтому в конфигурации, оптимизированной для суперкомпьютера, на большем числе плоскостей решение находится с помощью метода Шура.

3.6.4. Тест на ускорение

Тест на ускорение был выполнен на суперкомпьютере Marenostrum Барселонского Суперкомпьютерного Центра. Размер сетки в данном тесте $32\times170\times320$ (1.7×10^6 узлов). Тест показывает адаптацию метода Крылова-Фурье-Шура к различному числу процессоров для достижения максимальной производительности. Использовались две простейшие конфигурации: первая — решение на всех плоскостях целиком находится методом Шура, вторая — метод Шура применяется только для первой плоскости, решение на остальных плоскостях находится итерационным методом с блоком размером с подобласть.

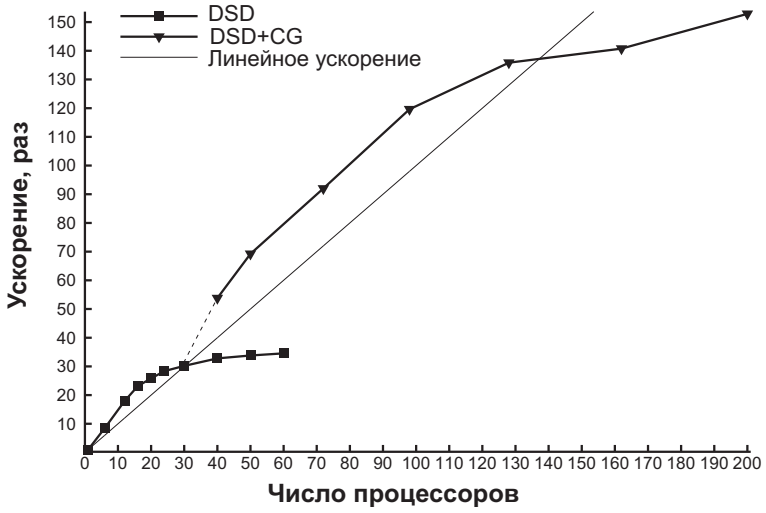

Рис. 3.7. Ускорение решателя на параллельной системе

Измерения начинаются с 1 процессора, где метод Шура вырождается в LU. (LU для всех плоскостей не помещается в памяти одного процессора. Измерялось время для одного LU, затем время умножалось на число плоскостей)

Метод Шура теряет эффективность достаточно быстро и уже при числе процессоров более 30 заменяется на вторую конфигурацию, которая достигает ускорения 136 на 128 процессорах, после чего также начинает терять эффективность. Всего же достигается ускорение как минимум 153. Результаты показаны на Рис.3.7.

3.6.5. Замечания по параллельной оптимизации

Хотя решение для каждой плоскости может быть найдено по отдельности, это не означает, что обмен данными должен быть отдельным для каждой плоскости. В этом случае была бы большая потеря времени на латентность сети и параллельная эффективность существенно пострадала бы из-за увеличения в N_x раз количества пересылок. Поэтому каждый

обмен данными включает данные сразу для всех плоскостей, решение по которым еще не получено. Поскольку для нахождения решения на разных плоскостях требуется разное число итераций, решения получаются не одновременно. Как только решение на какой-то плоскости получено, она естественно исключается из обмена данными. Обмен данными требуется для операций обновления гало, для скалярного произведения, для вычисления нормы невязки и для суммирования интерфейсных узлов в методе Шура. Во всех этих случаях данные со всех оставшихся плоскостей группируются в одно сообщение, исключая, таким образом, потерю времени на многократную инициализацию обмена данными.

Особое внимание уделяется скалярному произведению. Оно требует пересылки небольшого объема данных, следовательно, потери времени в этом случае происходят только из-за латентности сети. Эта коллективная операция суммирования становится особенно затратной на большом числе процессоров (число пересылок в одной операции растет как $\log_2(P)$) и на сетях с большой латентностью. Самое главное, что следует отметить, плоскость, для которой требуется наибольшее число итераций, отвечает за все потери времени на скалярное произведение. От каждой плоскости в пересылку попадает всего лишь одно число. Поэтому объем данных настолько мал, что нет разницы, передается ли одно число, или набор чисел для всех плоскостей (данные помещаются целиком в один сетевой пакет). Для улучшения параллельной эффективности по возможности минимизируется максимальное по всем плоскостям число итераций. Уменьшение числа итераций для первой плоскости особенно важно.

3.7. Распараллеливание по оси X

В конфигурации для большого числа процессоров метод Шура применяется для нахождения решения только на первой плоскости. Но даже в этом случае Метода Шура является основным препятствием для увеличения числа процессоров из-за быстро возрастающего размера обратной интерфейсной матрицы и объема обмена данными. При этом нахождение решения для первой плоскости итерационным методом намного менее эффективно.

Метод Крылова-Фурье-Шура с разбиением на подобласти по двум осям может применяться достаточно эффективно при числе процессоров до нескольких сотен, но это близко к пределу возможностей метода Шура. Для применения метода на тысячах процессоров требуется разбиение области на подобласти по трем осям. Расчетная область разбивается на P подобластей разделением области на P_x частей по оси X, и на P_y и P_z частей по осям Y и Z соответственно: $P = P_x \times P_y \times P_z$. Таким образом, в методе Шура будут задействованы не все процессоры, а только группы по $P_y \times P_z$ процессоров. Если, например, число $P_y \times P_z$ - это предел для применимости метода Шура, то благодаря разбиению по оси X, максимальное число процессоров, на котором метод Крылова-Фурье-Шура может быть применен эффективно, увеличивается в P_x раз. Этот способ позволяет довести максимальное число процессоров до нескольких тысяч.

3.7.1. Группировка плоскостей

Позиция подобласти в расчетной области определяется тремя числами: p_x - позиция по оси X, p_y - по оси Y и p_z - по оси Z. Эти номера удовлетворяют условию: $1 \leq p_x \leq P_x$,$1 \leq p_y \leq P_y$,$1 \leq p_z \leq P_z$

Каждой из P подобластей соответствует один процессор в группе из P процессоров. Процессорная группа разбивается на подгруппы двумя способами:

1. 1D подгруппы

   Каждая подгруппа включает в себя процессоры с одинаковыми номерами p_y и p_z. Число процессоров в подгруппе P_x, а число подгрупп - $Py \times P_z$

2. 2D подгруппы Каждая подгруппа включает в себя процессоры с одинаковым номером p_x. Число процессоров в подгруппе $P_y \times P_z$ число подгрупп - P_x.

Каждый процессор принадлежит к одной 1D подгруппе и к одной 2D подгруппе. После применение БПФ, расчетная область преобразована в набор независимых плоскостей, и 2D подгруппы используются для нахождения решения на плоскостях. 1D подгруппы используются для обеспечения связей по оси X для БПФ.

Каждая из 2D подгрупп имеет свой поднабор из $\frac{N_x}{P_x}$ плоскостей. Плоскости упорядочены по убыванию числа обусловленности соответствующих плоскостям систем уравнений. Поэтому, если плоскости распределены между 2D подгруппами по прядку, то возникает очень большой дисбаланс в

загрузке процессоров: первые 2D подгруппы получают плоскости с меньшими номерами, для которых требуется намного больше итераций, чем для остальных. Поэтому, для того чтобы избежать многократного падения производительности из-за дисбаланса загрузки процессоров, необходимо распределять плоскости другим способом.

Набор из N_x плоскостей разделяется на P_x поднаборов по $\frac{N_x}{P_x}$ плоскостей следующим образом: плоскость с номером k поднабора с номером p_x - это плоскость с номером $j_k^{p_x} = p_x + (k-1)P_x$ в исходном наборе. Каждая из плоскостей с номерами $i = 1, .., P_x$ становится первой плоскостью в соответствующем поднаборе, каждая из плоскостей с номерами $i = P_x + 1, ..., 2 * P_x$ становится второй плоскостью и т.д.

Таким образом, нагрузка на процессоры распределяется более равномерно. Если решение на первых плоскостях из каждого поднабора находится прямым методом, то отличие между средними по поднаборам числами итераций незначительно, и баланс загрузки процессоров не нарушается.

Каждый поднабор плоскостей по отдельности имеет такие же свойства как и весь набор, а именно характерное распределение числа итераций сохраняется и для получения решения на плоскости с меньшим номером требуется больше итераций.

3.7.2. Упрощенная реализация

Простое и при этом достаточно эффективное разбиение по оси X реализовано способом, который не требует даже распараллеливания БПФ (эффективное распараллеливание БПФ для такого короткого набора данных является достаточно сложной и едва ли осуществимой задачей). Пробле-

ма решается простой репликацией данных для уравнения Пуассона между всеми 2D подгруппами. То есть в каждой 1D подгруппе путем группового обмена данными каждый процессор получает со всех остальных процессоров подгруппы принадлежащие им части вектора начального приближения $\mathbf{x_0}^{3D}$ и вектора правой части $\mathbf{b}^{3D}$ системы (3.12). Таким образом, для решения уравнения Пуассона каждый процессор имеет такие же данные, как если бы расчетная подобласть не разбивалась по оси X.

Модифицированный алгоритм:

1) Репликация вектора начального приближения $\mathbf{x_0}^{3D}$ и вектора правой части $\mathbf{b}^{3D}$ внутри 1D подгрупп

2) БПФ преобразует (3.12) к (3.26)

3) Решение (3.26) находится 2D подгруппами процессоров, количество которых равно P_x.

4) Репликация решения внутри 1D подгрупп

5) Обратное БПФ преобразует решение (3.26) к решению (3.12)

Групповой обмен данными требуется внутри 1D подгрупп для репликации данных на этапах 1 и 4 алгоритма. После репликации данных, БПФ применяется на этапах 2 и 5 без распараллеливания к векторам из N_x элементов. То есть каждый процессор в 1D подгруппе производит одни и те же вычисления сразу для всех узлов по оси X для выполнения БПФ. Такое дублирование вычислений, с одной сторон, негативно сказывается на параллельной эффективности. Но, с другой стороны, это негативное влияние незначительно, поскольку вычислительная стоимость БПФ пренебрежимо

мала по сравнению с вычислительными затратами на получение решения на плоскостях. Более того, эффективное распараллеливание БПФ на параллельной системе с распределенной памятью является достаточно сложной задачей, особенно для векторов такого малого размера (от 32 до 512 элементов). В данном программном комплексе для выполнения БПФ используется библиотека FFTW3 [65, 66], которая является одной из самых высокопроизводительных и широко распространенных в мире.

На шаге 3 каждая из 2D подгрупп имеет весь набор из N_x плоскостей, но находить решение каждая подгруппа будет только для своего поднабора из $\frac{N_x}{P_x}$ плоскостей. Найденное решение реплицируется внутри 1D подгрупп на шаге 4. Затем применяется обратное БПФ и каждый процессор получает решение для всего набора плоскостей, из которого он оставляет только необходимые ему $\frac{N_x}{P_x}$ плоскостей.

Также следует отметить, что групповой обмен данными на шагах 1 и 4 имеет линейный рост числа пересылок от числа процессоров P_x в 1D подгруппе. Поэтому P_x должно быть минимизировано. Минимизация P_x также уменьшает негативное влияние от дублирования вычислений БПФ. Другими словами, размер 2D подгрупп $P_y \times P_z$ должен быть как можно больше, естественно, в пределах возможностей метода Шура.

3.7.3. Тестирование параллельной эффективности

В качестве теста используется уже упомянутый расчет 3D турбулентного течения (подробно задача описана в 4 главе). Критерий завершения итерационного процесса — уменьшение начальной дивергенции в 10000 раз. Тест на ускорение был выполнен на суперкомпьютере Marenostrum

Барселонского Суперкомпьютерного Центра. Размер сетки в данном тесте 111×10^6 узлов). Размер 2D подгруппы равен 128 процессоров. Число P_x варьировалось от 2 до 8. Таким образом, расчет выполнялся на 256, 512 и 1024 процессорах. Полученное ускорение показано на рисунке 3.8.

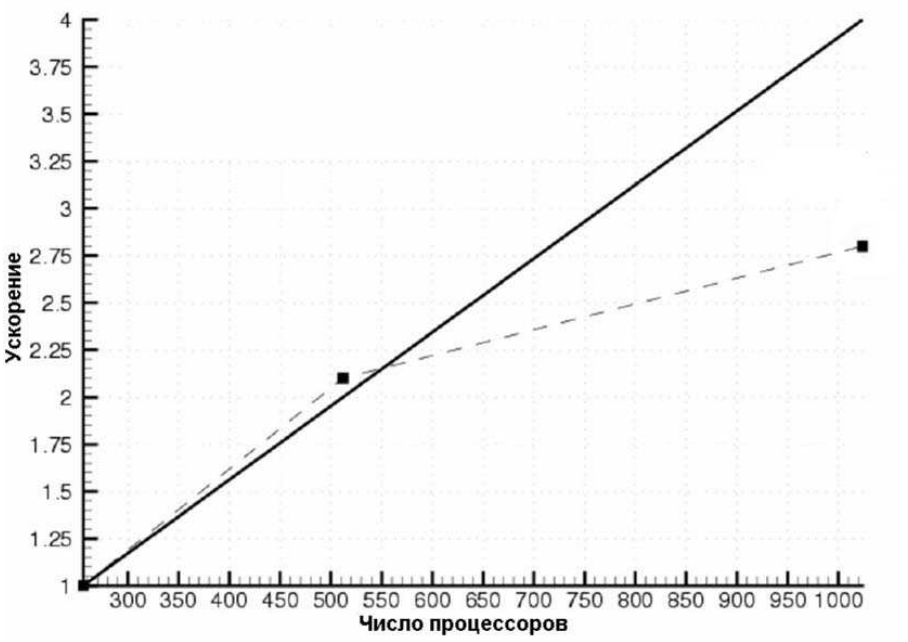

Рис. 3.8. Ускорение до 1024 процессоров на суперкомпьютере Мареностру м

Эффективность при переходе с 256 на 512 процессоров превысила 100% (за счет нелинейного уменьшения вычислительной стоимости), однако на 1024 процессорах эффективность составила около 70%, что соответствует ускорению в 2.8 раза при переходе с 256 процессоров. Ускорение на 1024 процессорах оказалось ниже ожидаемого. Возможно причина в специфике группового обмена данными на этой параллельной системе для такого числа процессоров.

4. Крупномасштабное прямое численное моделирование турбулентного конвекционного течения

4.1. Постановка задачи

Прямое численное моделирование 3D турбулентного течения при естественной конвекции в каверне с разными температурами на стенках является продолжением серии расчетов, описанных в [21, 67]. В данном случае, число Ra увеличено в 10 и размер сетки в 17.4 раз, по сравнению с предыдущим расчетом. Иллюстрация постановки задачи представлена на рис.4.1.

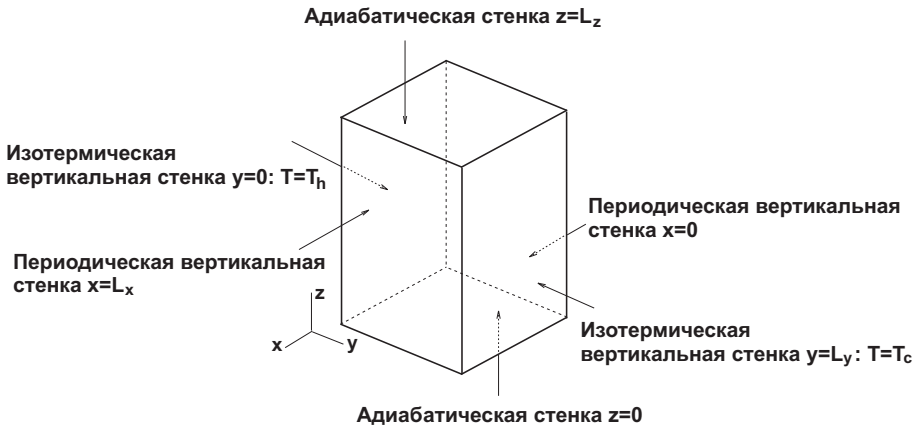

Рис. 4.1. Каверна с разными температурами на стенках: 3D турбулентное течение, вызванное естественной конвекцией

Расчетная область представляет собой каверну с соотношением сторон 0.25:1:4 по осям X, Y и Z соответственно. Сетка состоит из $\mathbf{1.12 \times 10^8}$ узлов 128×682×1278 по осям X,Y и Z соответственно. Шаг сетки нерав-

номерный по оси Y с концентрацией около границ. Граничные условия следующие:

- На вертикальных стенках, ортогональных оси X — периодические условия.
- На вертикальных стенках, ортогональных оси Y — изотермические условия с температурами $T_c = 0$ на стенке $y = 0$ и $T_h = 1$ на стенке $y = 1$.
- На горизонтальных стенках адиабатические условия.
- Для скоростей (кроме границ с периодическими условиями) используется условие прилипания - все компоненты вектора скорости равны 0.

Числа Рэлея и Прандтля равны соответственно $\mathbf{Ra} = \mathbf{10^{11}}$, Pr= 0.71.

Для дискретизации по пространству используется спектрально согласованная схема четвертого порядка аппроксимации. Критерий для итерационного процесса выбран таким образом, чтобы обеспечивать на каждом шаге по времени уменьшение начальной нормы дивергенции поля скоростей как минимум в 10000 раз. DNS выполнено на суперкомпьютере Marenostrum Барселонского Суперкомпьютерного Центра с использованием до 512 процессоров.

4.2. Численные результаты

В качестве примера на рис. 4.2 показаны мгновенные поля двух компонент скорости (по осям Y и Z) и температуры в сечении по центру каверны. Четко видны зоны турбулентности и точки образования турбулентного течения.

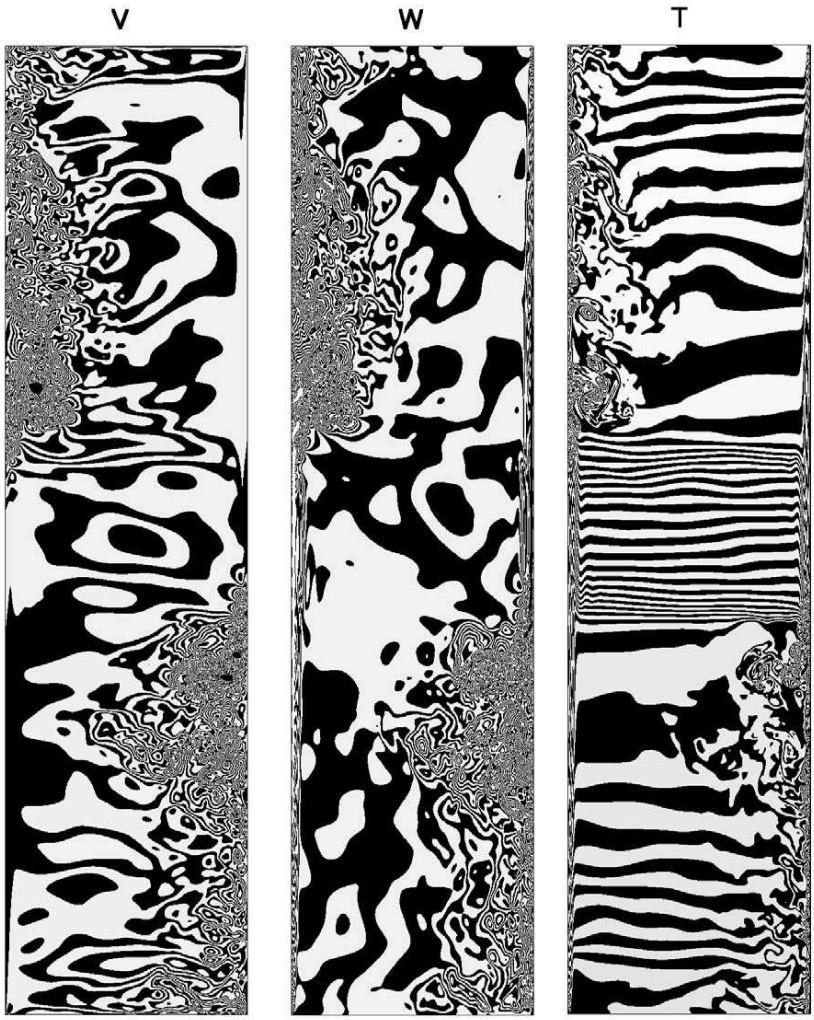

Рис. 4.2. Мгновенные поля: скорость по оси Y и Z, температура в сечении X=0.5

На рис. 4.3 изображены мгновенные поля температуры (изоповерхности) для различных моментов времени. Показана эволюция картины те-

чения до достижения статистически однородного состояния. На рис. 4.4 показана статистика первого порядка - среднее поле температуры (слева) и число Нусельта для левой стенки (справа). Скачок функции около 2.5 на графике справа (Рис. 4.4) соответствует точке образования турбулентности. Статистика 2-го порядка для различных величин показана на Рис. 4.5.

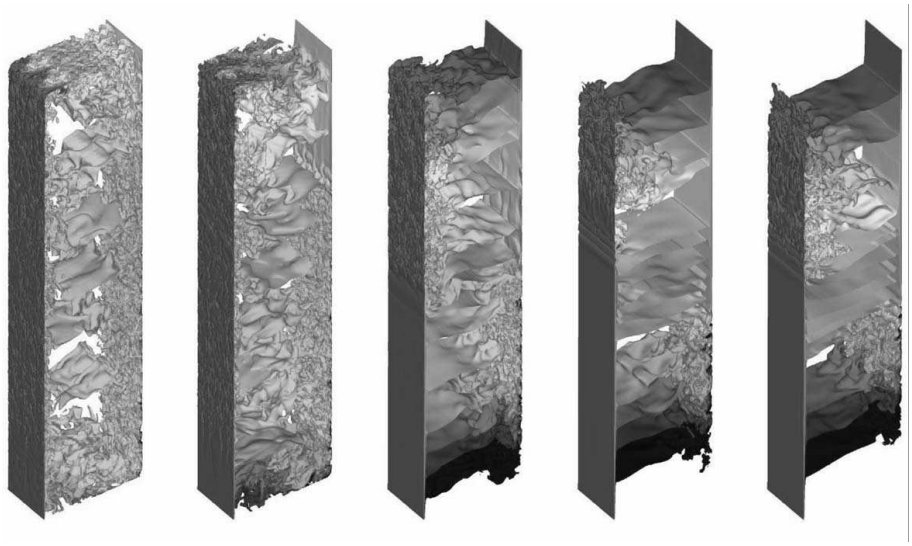

Рис. 4.3. Мгновенные изоповерхности температуры, эволюция картины течения со временем

Результаты этого крупномасштабного расчета в совокупности с результатами предыдущих вычислений послужат базисом для калибровки LES моделей турбулентности.

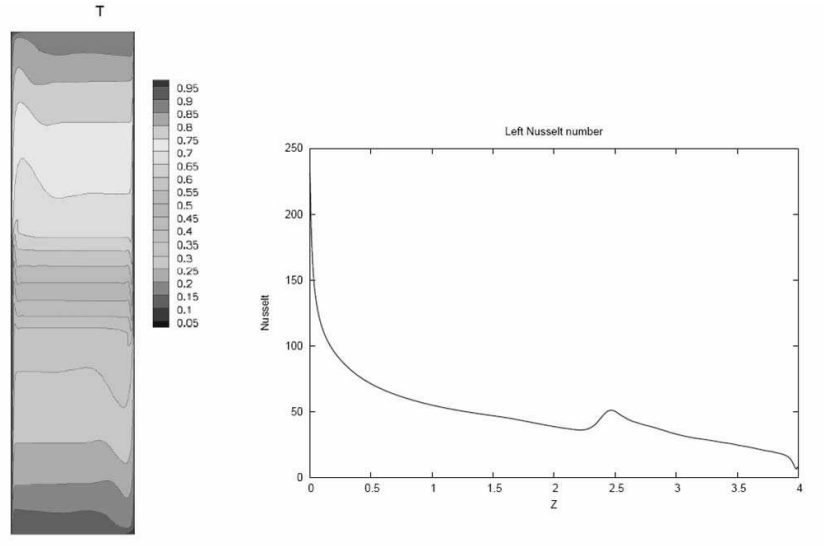

Рис. 4.4. Статистика 1-го порядка

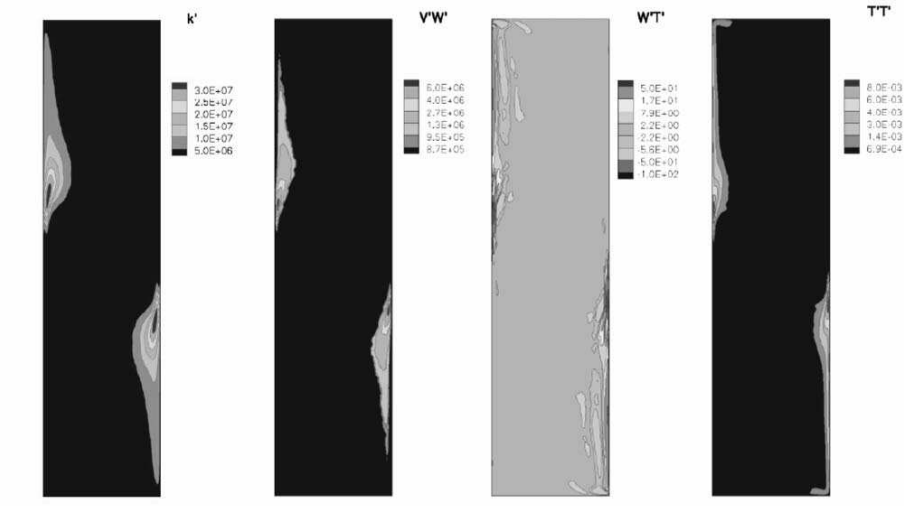

Рис. 4.5. Статистика 2-го порядка

Заключение

Ниже сформулированы основные результаты работы.

1. Разработан эффективный метод распараллеливания явного алгоритма повышенной точности, использующего расширенный неструктурированный шаблон. Данный метод позволяет разработчикам последовательного комплекса программ, которые не являются специалистами в области параллельных вычислений, выполнить распараллеливание с минимальными трудозатратами, достигнув при этом высокой параллельной эффективности.

2. Создан комплекс параллельных программ SuperNoisette 2D/3D с единым алгоритмическим ядром, реализующим расчеты задач газовой динамики и аэроакустики с повышенной точностью, как на треугольных, так и тетраэдральных сетках. Данный комплекс программ был получен на основе последовательного кода с использованием разработанной технологии распараллеливания.

3. На основе ранее известного метода Фурье-Шура для решения уравнения Пуассона, который эффективен на небольших кластерах с сетью высокой латентности, разработан метод Крылова-Фурье-Шура. Новый метод может эффективно применяться на суперкомпьютерах и позволяет использовать сетки размером порядка 10^8 узлов и разностные схемы повышенной точности на числе процессоров порядка

тысячи.

4. При активном участии автора проведены расчеты ряда актуальных задач газовой динамики и аэроакустики. С помощью комплекса программ SuperNoisette 2D/3D выполнены вычислительные эксперименты по моделированию звукопоглощающих конструкций авиадвигателей. С использованием метода Крылова-Фурье-Шура выполнено крупномасштабное прямое численное моделирование турбулентного течения при естественной конвекции в закрытой каверне. Продемонстрирована высокая эффективность разработанных методов.

Литература

1. Абалакин И.В., Суков С.А. Моделирование внешнего обтекания тел на многопроцессорных системах с использованием тетраэдрических сеток. *В сб. 'Фундаментальные физико-математические проблемы и моделирование технико-технологических систем', вып. 7, под ред. Л.А. Уваровой., Изд-во 'Janus-K', 2004, с. 52-57.*

2. И.В. Абалакин, А.В. Горобец, Т.К. Козубская. Вычислительные эксперименты по звукопоглощающим конструкциям. *Математическое моделирование*, т.19, №8:15–21, 2007.

3. Корнилина М.А., Якобовский М.В. Динамическая балансировка загрузки процессоров при моделировании задач горения. *Высокопроизводительные вычисления и их приложения: Тр. Всеросс. науч. конф. — М.: Изд-во МГУ, С.34-39*, 2000.

4. И.В. Ашметков, А.Я. Буничева, С.И. Мухин, Т.В. Соколова, Н.В. Соснин, А.П. Фаворский. Математическое моделирование гемодинамики в мозге и большом круге кровообращения. *Компьютер и мозг. Новые технологии. М.: Наука, сс.39–99*, 2005.

5. Forrester T. Johnson, Edward N. Tinoco, N. Jong Yu. Thirty years of development and application of CFD at Boeing commercial airplanes. *Computers & Fluids*, 34:1115–1151, 2005.

6. Воеводин В.В. *Параллелизм в алгоритмах и программах. // Вычислительные процессы и системы. Выпуск 10. / Под ред. Г.И. Марчука.* . М.: Физматлит., 1993.

7. Т. Акселрод, М. Беккерман и др. . *Программирование на параллельных вычислительных системах: Пер. с англ. / Под ред. Р. Бэбба II.* М.: Мир, 1991.

8. Четверушкин Б.Н. Проблемы эффективного использования многопроцессорных вычислительных систем. *Информационные технологии и вычислительные системы*, №2:22–34, 2000.

9. C. Walshaw, M. Cross, M.G. Everett. *Mesh Partitioning and Load Balancing for Distributed Memory Parallel Systems. In B.H.V. Topping, editor, Advances in Computational Mechanics for Parallel & Distributed Processing, pp.97-104.* Saxe-Coburg Publications, Edinburgh, 1997.

10. Abalakin I.V., Dervieux A., Kozubskaya T.K. High Accuracy Finite Volume Method for Solving Nonlinear Aeroacoustics Problems on Unstructured Meshes. *Chinese Journal of Aeroanautics*, Vol. 19, No 2, 2006.

11. И.В. Абалакин, Т.К. Козубская. Многопараметрическое семейство схем повышенной точности для линейного уравнения переноса. *Математическое моделирование*, т.19, №7:56–66, 2007.

12. Фриш У. *Турбулентность. Наследие А.Н. Колмогорова.* М.: ФАЗИС, 1998.

13. David C. Wilcox. *Turbulence Modeling for CFD.* DCW Industries, Inc. La Cacada, California, 1993.

14. Javier Jimm̆nez and Robert D. Moser. Large-Eddy Simulations: Where Are We and What Can We Expect? *AIAA Journal*, 38(4):605–612, April 2000.

15. J. H. Ferziger. Direct and large eddy simulation of turbulence. In *Numerical Methods in Fluid mechanics*. American Mathematical Society, 1998.

16. P.R. Spalart, S. Deck, M.L. Shur, K.D. Squires, M.Kh. Strelets, A.K. Travin. A new version of Detached-Eddy Simulation, resistant to ambiguous grid densities. *Theoretical and Computational Fluid Dynamics*, 20,3:181–195, 2006.

17. Perez-Segarra, C.D. and Oliva, A. and Costa, M. and Escanes, F. Numerical experiments in turbulent natural and mixed convection in internal flows. *International Journal for Numerical Methods for Heat and Fluid Flow*, Vol. 5, No. 1:13–33, 1995.

18. Y. Morinishi, S. Tamano, and K. Nakayashi. Direct numerical simulation of compressible turbulent channel flow between adiabatic and isothermal walls. *Journal of Fluid Mechanics*, 502:273–308, 2004.

19. Yukio Kaneda and Mitsuo Yokokawa. DNS of Canonical Turbulence with up to 4096^3 Grid Points. In *Parallel Computational Fluid Dynamics*, pages 23–32. Elsevier, May 2004.

20. Juan C. del Álamo, Javier Jiménez, Paulo Zandonade, and Robert D. Moser. Scaling of the energy spectra of turbulent channels. *Journal of Fluid Mechanics*, 500:135–144, 2004.

21. M. Soria, F. X. Trias, C. D. Pérez-Segarra, and A. Oliva. Direct numerical simulation of a three-dimensional natural-convection flow in a differentially

heated cavity of aspect ratio 4. *Numerical Heat Transfer, part A*, 45:649–673, April 2004.

22. Р.П. Федоренко. *Введение в вычислительную физику.* М. МФТИ, 1994.

23. William L. Briggs. *A Multigrid Tutorial.* SIAM, Society for Industrial and Applied Mathematics, 1987.

24. A. Brandt and B. Diskin. Multigrid solvers on decomposed domains, in Domain Decomposition Methods in Science and Engineering (A.Quarteroni, J.Periaux, Yu.A.Kuznetsov and O.Wildlund, eds.). *Contemp. Math.*, 157:135–155, 1994, American Math. Soc.

25. Y.F. Hu, D.R. Emerson, and R.J. Blake. The Communication Performance of the Cray T3D and its Effect on Iterative Solvers. *Parallel Computing*, 22:22–32, 1993.

26. H. A. Van Der Vorst. *Parallel Linear Systems Solvers: Sparse Iterative Methods, in P. Wesseling (ed.), High Performance Computing in Fluid Dynamics, pp. 173–200.* Kluwer, Dordrecht, The Netherlands, 1996.

27. Елизарова Т.Г. Милюкова О.Ю. Численное моделирование течения вязкой несжимаемой жидкости в кубической каверне. *Журнал вычислительной математики и математической физики*, 43:N3, с.453–466, 2003.

28. Широков И.А. Решение уравнения Пуассона на многопроцессорной системе в задачах моделирования течений несжимаемой жидкости. *Дифф. уравнения*, Т. 39:N7, с.993–1000, 2003.

29. A. Matrone E. Bucchigniani and F. Stella. Parallel Polynomial Preconditioners for the Analysis of Chaotic Flows in Rayleigh–Benard Convection, in D. Keyes et al. (eds.). *Parallel Computational Fluid Dynamics: Practice and Theory, pp. , Elsevier Publishing*, page 139–145, 2000.

30. J. Frank C. Vuik and F. J. Vermolen. Parallel Deflated Krylov Methods for Incompressible Flow. *Parallel Computational Fluid Dynamics: Practice and Theory, Elsevier Publishing*, page 381–388, 2002.

31. E. Oran Brigham. *The Fast Fourier Transform and Its Applications*. Prentice-Hall, Inc., 1988.

32. M. Soria, C. D. Pérez-Segarra, and A. Oliva. A Direct Parallel Algorithm for the Efficient Solution of the Pressure-Correction Equation of Incompressible Flow Problems Using Loosely Coupled Computers. *Numerical Heat Transfer, Part B*, 41:117–138, 2002.

33. M. Soria, C. D. Pérez-Segarra, and A.Oliva. A Direct Schur-Fourier Decomposition for the Solution of the Three-Dimensional Poisson Equation of Incompressible Flow Problems Using Loosely Parallel Computers. *Numerical Heat Transfer, Part B*, 43:467–488, 2003.

34. F. X. Trias, M. Soria, C. D. Pérez-Segarra, and A. Oliva. A Direct Schur-Fourier Decomposition for the Efficient Solution of High-Order Poisson Equations on Loosely Coupled Parallel Computers. *Numerical Linear Algebra with Applications*, page (published online), 2005.

35. R.W.C.P. Verstappen and A.E.P. Veldman. Direct Numerical simulation

of turbulence at lower costs. *Journal of Engineering Mathematics*, 32:143–159, 1997.

36. R. W. C. P. Verstappen and A. E. P. Veldman. Spectro-consistent discretization of Navier-Stokes: a challenge to RANS and LES. *Journal of Engineering Mathematics*, 34:163–179, 1998.

37. R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-Preserving Discretization of Turbulent Flow. *Journal of Computational Physics*, 187:343–368, May 2003.

38. Patrick J. Roache. Code Verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, Volume 124, Issue 1:4–10, March 2004.

39. I. Abalakin, A. Dervieux, and T. Kozubskaya. *Computational Study of Mathematical Models for Noise DNS*. AIAA-2002-2585 paper.

40. Long L. Bangalore A. Morris, P. A Parallel Three-Dimensional Computational Aeroacoustics Method Using Nonlinear Disturbance Equations. *Journal of Computational Physics*, 133:56–74, 1997.

41. Ilya V. Abalakin, Alain Dervieux, Tatyana K. Kozubskaya. On the accuracy of direct noise calculations based on the Euler model. *International journal of aeroacoustics*, volume 3, number 2:157–180, April 2004.

42. Ilya Abalakin, Alain Dervieux, and Tatiana Kozubskaya. High Accuracy Finite Volume Method for Solving Nonlinear Aeroacoustics Problems. In *In Proc. of East West High Speed Flow Field Conference*, pages 314–320, Beijing, China, 2005.

43. Abalakin I.V., Dervieux A., Kozubskaya T.K. *A vertex centered high order MUSCL scheme applying to linearised Euler acoustics*. INRIA report RR4459, April 2002.

44. Dervieux A. Debiez, C. Mixed element volume MUSCL methods with weak viscosity for steady and unsteady flow calculation. *Computer and Fluids*, 29:89–118, 1999.

45. С.Н. Болдырев, Е.И. Леванов, М.В. Якобовский. *Обработка и хранение нерегулярных сеток большого размера на многопроцессорных системах.* В сб. "Сеточные методы для краевых задач и приложения". Материалы четвертого всероссийского семинара. Казань, 13-16 сентября 2002 г., с.33-39.

46. J.C.Hardin, J.R.Ristorcelli, and C.K.W.Tam. *Proceedings of ICASE/LaRC Workshop on Benchmark Problems in Computational Aeroacoustics (CAA), Hampton, Virginia, October 24-26, 1994.* В сб. "Сеточные методы для краевых задач и приложения". Материалы четвертого всероссийского семинара. Казань, 13-16 сентября 2002 г., с.33-39., 1995.

47. Tam C.K.W., and Ju H. A computational and experimental study of slit resonators. *AIAA paper 2003-3310*, 2003.

48. Tam C.K.W., and Shen H. Direct computational of nonlinear acoustic pulses using high order finite difference schemes. *AIAA paper 93-4325*, 1993.

49. Tam C.K.W., and Webb J. C. Dispersion-Relation-Preserving Schemes

for Computational Aeroacoustics. *Journal of Computational Physics*, Vol. 107:pp. 262–281, 1993.

50. Abalakin, I.V., Dervieux A., and Kozubskaya T.K. A vertex centered high order MUSCL scheme applying to linearised Euler acoustics. *INRIA report RR4459*, April 2002.

51. Debiez, C., and Dervieux A. Mixed element volume MUSCL methods with weak viscosity for steady and unsteady flow calculation. *Computer and Fluids*, Vol. 29:89–118, 1999.

52. Gourvitch N., Roge G., Abalakin I., Dervieux A., Kozubskaya T. A tetrahedral-based superconvergent scheme for aeroacoustics . *INRIA report RR5212*, May 2004.

53. Alexandre Joel Chorin. Numerical Solution of the Navier-Stokes Equations. *Journal of Computational Physics*, 22:745–762, 1968.

54. N. N. Yanenko. *The Method of Fractional Steps*. Springer-Verlag, 1971.

55. A. J. Chorin. *A Mathematical Introduction to Fluid Mechanics*. Springer-Verlag, 1993.

56. William D. Henshaw. A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids. *Journal of Computational Physics*, 113(1):13–25, 1994.

57. N. Anders Petersson. Stability of Pressure Boundary Conditions for Stokes and Navier-Stokes Equations. *Journal of Computational Physics*, 172:40–70, 2001.

58. J. Kim and P. Moin. Application of a Fractional-Step Method to Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 123:308–323, 1985.

59. S. Xin and P. Le Quéré. Direct numerical simulations of two-dimensional chaotic natural convection in a differentially heated cavity of aspect ratio 4. *Journal of Fluid Mechanics*, 304:87–118, 1995.

60. R. W. Hockney. A Fast Direct Solution of Poisson's Equation Using Fourier Analysis. *Journal of the Association for Computing Machinery*, 12:95–113, 1965.

61. P.N.Swarztrauber. The Methods of Cyclic Reduction, Fourier Analysis and the FACR Algorithm for the Discrete Solution of Poisson's Equation on a Rectangle. *SIAM Review*, 19:490–501, 1977.

62. P.J.Davis. *Circulant Matrices.* Chelsea Publishing, New York, 1994.

63. Robert W. Ramirez. *The FFT, Fundamentals and Concepts.* Prentice-Hall, Inc., 1985.

64. Yousef Saad. *Iterative Methods for Sparse Linear Systems.* 2000.

65. M. Frigo and S.G. Johnson. *The fastest Fourier transform in the west.* Tech. Rep, MIT-LCS-TR-728, 1997.

66. Matteo Frigo and Steven G. Johnson. *The Design and Implementation of FFTW3.* Proceedings of the IEEE 93 (2), 216–231 (2005). Invited paper, Special Issue on Program Generation, Optimization, and Platform Adaptation.

67. F. X. Trias, M. Soria, C. D. Pérez-Segarra, and A. Oliva. Direct Numerical Simulation of Turbulent Flows on a low cost PC Cluster. In *Parallel Computational Fluid Dynamics*. Elsevier, May 2005.