Федеральное государственное бюджетное учреждение науки
Институт вычислительной математики РАН

На правах рукописи

Терехов Кирилл Михайлович

Решение задач фильтрации и гидродинамики на адаптивных сетках типа восьмеричное дерево

05.13.18 – Математическое моделирование,
численные методы и комплексы программ

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
д. ф.-м. н.
Василевский Юрий Викторович

Москва – 2013

Содержание

Введение

При решении современных инженерных и научных задач одной из главных проблем является высокая точность методов при адекватной вычислительной сложности метода. Частично данную проблему решают методы высокого порядка, которые могут дать точное решение на более грубой сетке. Однако, такие методы являются более дорогими с вычислительной точки зрения, а использование грубых сеток, в свою очередь, не позволяет разрешить детали физических процессов. Для решения данной проблемы возможно два подхода: переход к массивно-параллельным вычислениям или к адаптации сетки к особенностям решения. В данной работе рассмотрены оба подхода.

Создание комплексов программ, которые могут выполнять расчеты на параллельных компьютерах является достаточно сложной и трудоемкой задачей. При переходе от последовательных программ к параллельным требуется не только добавить в последовательную программу обмены данных между процессорами, но и значительно перестроить всю структуру используемых данных. Для помощи в распараллеливании программ математического моделирования предназначена программная платформа, являющаяся основой для всех этапов параллельного расчета: построения сеток, аппроксимации физической задачи на построенных сетках, а также для решения систем линейных уравнений, получающихся в результате этой аппроксимации.

Рассматриваемая в первой главе технология параллельной работы с сеточной информацией входят в разрабатываемую программную платформу INMOST[1], которая состоит из методов работы с сеточными данными, методов решения систем линейных уравнений и методов визуализации. Данная платформа облегчает

[1] INMOST – Integrated Numerical Modeling Object-oriented Supercomputing Technologies

разработку параллельных программ для решения задач математической физики и лежит в основе нескольких программных кодов. Подробную информацию о платформе, описание параллельных алгоритмов, а так же задачи, при решении которых она использована можно найти в книге [1].

Детальный анализ подходов к хранению сеточной информации и иерархии связей между соседними элементами произвел Гаримелла в работе [2]. Исходя из анализа был выбран подход с оптимальным балансом между требуемой памятью и сложностью вычисления всех необходимых связей.

Существует ряд пакетов для работы с сетками, такие как MSTK[2][3], STK[3][4], MOAB[4][5, 6], FMDB[5][7], большинство из которых находится в разработке и по тем или иным причинам не удовлетворяют поставленным требованиям. Некоторые из пакетов не предназначены для работы с динамически адаптируемыми сетками; некоторые пакеты предлагают недостаточный параллельный функционал, например, поддержка всего одного слоя фиктивных элементов; некоторые на данный момент находятся в стадии активной разработки.

Основной подход при параллельном решении уравнений математической физики является метод декомпозиции расчетной сетки и метод перекрытия сеток слоями фиктивных элементов[8]. Метод декомпозиции расчетной сетки заключается в распределении исходной сетки между процессорами. Задача распределения элементов сетки между процессорами оптимальным образом, для равномерной загрузки вычислительных узлов, эквивалентна разрезанию связного графа и решается, например, посредством пакета Zoltan[9]. Получив от пакета решение задачи, алгоритм распределяет сетку между процессорами. Метод перекрытия сеток с по-

[2] MSTK - Mesh Toolkit, сеточный инструментарий

[3] STK – Sierra Toolkit

[4] MOAB – A Mesh-Oriented datABase, сеточно-ориентированная база данных

[5] FMDB – Flexible distributed Mesh DataBase, гибкая распределенная сеточная база данных

мощью фиктивных элементов заключается в дублировании на данном процессоре одного или нескольких слоев элементов, принадлежащих соседним процессорам.

Метод для параллельной работы с адаптивными сетками представлен в пакетах STK и FMDB. Он заключается в удалении слоя фиктивных элементов, перестроения сетки, а затем восстановления параллельного состояния новых сеточных элементов и слоев фиктивных элементов. Этот метод так же реализуем с помощью предложенных в данной работе алгоритмов, однако не является достаточно эффективным, поэтому не выносится на защиту.

При моделировании процессов разработки нефтяного месторождения, рассматриваемого во второй главе, широко используются неструктурированные сетки разных типов: гексаэдральные, призматические или гибридные, состоящие из ячеек разного типа. Данные сетки подпадают под определение конформных сеток с многогранными ячейками, для которых применима система хранения сеточной информации из первой главы.

Одним из ключевых аспектов решения задачи двухфазного заводнения резервуара с нефтью водой является корректное воспроизведение положения фронта насыщенности воды, которое непосредственным образом влияет на объемы добычи нефти и на момент прорыва воды в производящей скважине. Необходимость решать задачи с полным анизотропным тензором проницаемости $\mathbb{K}$, не $\mathbb{K}$-ортогональность сеток и ограничение памятью компьютера минимального шага сетки требуют более сложного подхода к решению данной задачи. В данной работе используется два подхода для решения данных проблем: полностью неявный нелинейный конечно-объемный метод и динамически адаптирующиеся сетки типа восьмеричное дерево.

Существует несколько подходов к дискретизации уравнений двухфазной филь-

трации воды и нефти по времени: IMPES[6][10–12] - условно устойчивый полунеявный метод, и полностью неявный метод[13], обладающий безусловной устойчивостью. В данной работе используется полностью неявный метод, чтобы избежать ограничения на шаг по времени при сгущении сетки.

Ранее был предложен подход решения задачи фильтрации на адаптивных сетках типа четверичное дерево (в плоскости) Сухиновым [14] и Саадом [15]. Одним из недостатков подходов, предложенных в данных работах, было применение полунеявной схемы. Устойчивость такой схемы, при агрессивном сгущении сетки, требует сильного ограничения шага по времени.

При измельчении и разгрублении сетки требуется переинтерполировать физические данные в новые образовавшиеся степени свободы. Интерполяция должна быть точной и обладать консервативностью. Пользуясь тем, что сгущение и разгрубление сеток типа восьмеричное дерево носит локальный характер, в работе используется локальная консервативная интерполяция[16].

Выбор критерия сгущения сетки влияет как на точность расчета так и на время работы модели на адаптивной сетке. Критерий сгущения указывает, где необходимо сгустить сетку, а где разгрубить. Таким образом неправильный выбор критерия может в результате привести как к слишком мелкой сетке и длительному времени работы, так и к очень грубой сетке и плохой точности. Один из подходов, редко используемый на практике, это метод апостериорной оценки ошибки, разработанный Бьетерманом и Бабушка [17, 18]. Другой подход основан на физических особенностях задачи, пример такого подхода для задачи двухфазного заводнения содержится в работах Сухинова и Саада [14, 15].

В данной работе в качестве индикатор определяется по модулю градиента насыщенности воды и давления нефти. Большой модуль градиента насыщенности

[6] IMPES - IMplicit Pressure Explicit Saturation, неявное давление, явная насыщенность

можно интерпретировать как четкий фронт между двумя фазами, а большой модуль давления означает особенности в скоростях, получающихся из уравнения Дарси.

При измельчении и разгрублении сетки требуется переинтерполировать физические данные в новые образовавшиеся степени свободы. Интерполяция должна быть точной и обладать консервативностью. Пользуясь тем, что сгущение и разгрубление сеток типа восьмеричное дерево носит локальный характер, в работе используется локальная консервативная интерполяция[16].

В данной работе используется нелинейная конечно-объемная схема, суть которой заключается в использовании монотонной нелинейной двухточечной аппроксимации потока. Данный метод был впервые применен для параболических уравнений на треугольных сетках К. Лепотье [19]. Этот подход был применен к широкому кругу задач [20–25] и используется в данной работе. Метод позволяет работать с не $\mathbb{K}$-ортогональными сетками и произвольными многогранными сетками.

Альтернативой данному подходу является многоточечная схема аппроксимации потока [26]. Многоточечная схема является линейной, аппроксимирует концентрации со вторым порядком, но является только условно устойчивой [27] и условно монотонной [28]. В данной работе дано сравнение двух методов аппроксимации.

Так как дискретная задача является нелинейной, для ее решения в работе используется итеративный метод Ньютона. Для этого метода необходимо найти матрицу частных производных по степеням свободы – Якобиан. В данной работе рассматривается три подхода к вычислению коэффициентов этой матрицы.

Корректность и эффективность предложенных методов для задачи двухфазного заводнения демонстрируется на ряде численных экспериментов.

В третьей главе рассматривается устойчивый низкодиссипативный метод решения уравнений Навье-Стокса, описывающих нестационарное течение вязкой несжимаемой жидкости. Сетки типа восьмеричное дерево завоевывают популярность в вычислительной механике и физике за счет своей простой прямоугольной структуры и вложенной иерархии. К примеру, такие динамически адаптируемые сетки были использованы в сочетании с конечно-объемными методами и методом разрывного Галеркина с применением к гиперболическим законам сохранения [29–32]. Благодаря быстрому динамическому перестроению, такие сетки естественным образом подходят для моделирования задач с подвижными границами и течений со свободными поверхностями [33–38].

Дискретизации для вязких и невязких уравнений течения жидкости уже были разработаны для динамических сеток типа восьмеричное дерево. Попинет [39] разработал конечно-объемную схему типа Годунова с использованием неразнесенных сеток, когда неизвестные компоненты скорости и давление расположены в центрах ячеек. Мин и Гибо [40, 41] разработали полу-лагранжев метод, так же для неразнесенных сеток, но с неизвестными в вершинах сетки. В этих работах был применен специальный метод для стабилизации ложных мод давления, типичных для неразнесенных сеток. В работах [34, 35, 38] была использована MAC[7] схема [42–44] с разнесенным расположением неизвестных, расширенная на сетки типа восьмеричное дерево.

Существует два преимущества разнесенного расположения неизвестных. Первое заключается в простом поэлементном наложении условия несжимаемости, выполнение которого эквивалентно сохранению массы. Второе заключается в устойчивости дискретизации по давлению, так как четные-нечетные ложные моды давления не могут быть представлены на такой сетке. Однако, такое расположение

[7] MAC – Marker and Cell

неизвестных усложняет построение схем высокого порядка, особенно в том случае, если рассматриваются разнесенные неизвестные на сетках типа восьмеричное дерево. К примеру, в работах [34, 35, 38] для конвекции был использован полу-лагранжев метод первого порядка.

В данной работе разрабатывается схема второго порядка точности, основанная на методе проекции Темама-Яненко-Шорина [45, 46]. Для линеаризованной [47, 48] конвекции используются конечно-разностные противопоточные схемы второго и третьего порядка с низкой численной вязкостью. Дискретизация основана на линейных и кубических интерполяциях по двум переменным, за счет чего шаблон дискретизации остается компактным, а матрица линейной системы при неявной дискретизации членов диффузии и конвекции остается разреженной. Для дискретизации задачи конвекции-диффузии по времени используется формула обратных разностей второго порядка[49]. За счет неявного шага конвекции удается избежать ограничения по Куранту на шаг по времени[50], так как данное ограничение оказывается довольно сильным при рассчете на адаптивных сетках.

После решения системы уравнений конвекции-реакции-диффузии, для проекции полученной скорости на бездивергентное пространство используется дискретное разложение Гельмгольца. При применении низкодиссипативной схемы обнаружилась ранее неизвестная проблема: на разнесенных сетках типа восьмеричное дерево дискретное разложение Гельмгольца является неустойчивым из-за ложных мод скорости, появляющихся на стыках грубой и мелкой сетки. Если вязкость жидкости или численная вязкость достаточно велика, то данные ложные моды подавляются, если же вязкость мала, то данные моды распространяются по всей области и понижают точность численного решения. Для стабилизации решения предлагается линейный низкочастотный фильтр для задачи конвекции, который в совокупности с методом аппроксимации градиента давления полностью

исключает появление данных ложных мод и значительно улучшает точность численного решения.

Разработанный метод верифицируется на ряде задач: аналитическое течение типа Бельтрами [51], течение в каверне с подвижной границей[52–54] и обтекание цилиндра [55, 56].

Описанный метод является частью разрабатываемой модели, используемой для вычисления течений вязкопластичной жидкости со свободной поверхностью [57, 58]. Данная модель была успешно применена к моделированию катастроф [59].

Данная работа разделена на три главы, каждая из которых касается вопросов эффективного решения задач на динамических сетках типа восьмеричное дерево.

В первой главе рассмотрен подход к хранению сеток общего вида. На основе данного подхода описан алгоритм динамического измельчения сеток типа восьмеричное дерево.

Во второй главе рассмотрена задача двухфазной фильтрации в пористой среде, а именно вытеснение нефти водой из пористого резервуара. В данной задаче вода поступает в нагнетательные скважины, а нефть извлекается из производящих скважин. Задача решается с помощью полностью неявного монотонного нелинейного конечно-объемного метода. Показана эффективность подхода с использованием нелинейной двухточечной аппроксимации потоков, на сетках типа восьмеричное дерево.

В третьей главе предложены низкодиссипативные дискретизации на разнесенных сетках типа восьмеричное дерево для решения уравнений Навье-Стокса. При применении низкодиссипативных схем была обнаружена неустойчивость, проявляющая себя в виде образования локальных дискретных бездивергентных мод в скоростях на стыках разных уровней сетки. Предложен подход к стабилизации

решения, среди которых наиболее эффективным оказался предложенный низкочастотный фильтр для конвекции, полностью исключающий появление данных бездивиргентных мод в скоростях.

Актуальность работы. При численном моделировании задач математической физики часто приходится сталкиваться с недостатком компьютерных ресурсов. Причиной тому является необходимость выполнять рассчет на достаточно мелкой сетке для разрешения ключевых физических эффектов. Существует два подхода решения данной проблемы. Первый подход заключается в переходе к параллельным вычислениям, что позволяет эксплуатировать большие компьютерные ресурсы. Второй подход заключается в использовании алгоритмов и методов, адаптирующихся к особенностям решения и позволяющих эффективно использовать ограниченные ресурсы компьютера.

При решении задачи вытеснения нефти водой в пористом резервуаре ключевыми факторами являются фронт раздела нефти и воды, а также скорости, получающиеся из закона Дарси. Качественное разрешение фронта требует мелкого разрешения сетки в небольшой расчетной области и является хорошим примером для применения динамических сеток. Одной из проблем данной задачи является невозможность в общем случае построить сетку, грани которой были бы ортогональны тензору проницаемости, что делает невозможным применение простых методов аппроксимации потоков концентрации через грань.

При изучении динамики течения вязкой несжимаемой нестационарной жидкости важными критериями является устойчивость, низкая численная вязкость, высокий порядок аппроксимации методов, возможность быстро решать прикладные задачи. Для быстрого эффективного решения подобных задач требуются динамические сетки, сгущающихся к особенностям задачи в сочетании с дешевыми

но точными методами аппроксимации уравнений.

Цель диссертационной работы. Целями диссертационной работы являются разработка методов хранения сеточной информации, позволяющих производить как параллельные расчеты, так и работать с динамическими сетками; разработка на ее основе полностью неявного нелинейного метода для задачи двухфазной фильтрации в пористой среде; разработка устойчивых низкодиссипативных схем для решения уравнений Навье-Стокса.

Научная новизна. В работе предложены и реализованы алгоритмы для хранения сеточной информации, позволяющей быстро динамически перестраивать сетки и производить параллельные вычисления. Предложен подход для динамического параллельного перестроения сеток.

Реализована полностью неявная монотонная нелинейная схема дискретизации потока для уравнений двухфазной фильтрации на неструктурированных конформных сетках с многогранными ячейками. Показана сходимость нелинейного метода дискретизации потока. Показано преимущество нелинейной дискретизации в сравнении с линейной двухточечной и многоточечной дискретизациями. Показана эффективность расчета на динамических сетках типа восьмеричное дерево. Протестирована эффективность параллельного решения задачи на фиксированной сетке.

Реализована экономичная технология моделирования нестационарных течений вязкой несжимаемой жидкости на основе адаптивных сеток типа восьмеричное дерево. Предложен и реализован конечно-разностный метод дискретизации линеаризованных уравнений конвекции-реакции-диффузии для сеток типа восьмеричное дерево и метод стабилизации ложных мод скорости, появляющихся на этапе проекции скорости на бездивергентное поле.

Практическая значимость. Практическая значимость диссертационной работы заключается в создании библиотеки для параллельной работы с сеточной информацией и решения систем линейных уравнений. Создание комплекса программ для численного моделирования процесса двухфазной фильтрации в пористой среде, описывающего разработку нефтегазовых месторождений и численного моделирования нестационарного течения вязкой несжимаемой жидкости.

На защиту выносятся следующие основные результаты:

1. Разработана структура данных и библиотека для работы с адаптивными динамическими сетками.

2. На основе предложенной структуры разработана численная модель двухфазной фильтрации в пористой среде.

3. Разработана экономичная технология, включающая численные методы, дискретизации и алгоритмы, для моделирования трехмерных нестационарных течений вязкой несжимаемой жидкости.

Апробация работы. Результаты диссертационной работы докладывались автором и обсуждались на научных семинарах ... Upstream Research Center of ExxonMobil corp. г.Хьюстон (США) и на следующих научных конференциях: конференция “Тихоновские чтения”, (МГУ, Москва, 2009 г.); конференция “Лобачевские чтения” (Казань, 2009 г.); 53-я научная конференция МФТИ (ИВМ РАН, 26 ноября 2010г.); международные конференции “Numerical geometry, grid generation and high performance computing (NUMGRID - 2010, NUMGRID - 2012)” (ВЦ РАН, Москва, 13 октября 2010г., 17 декабря 2012г.); международная конференция “Computational Methods in Applied Mathematics: CMAM-4” (Познань, Польша, июнь 2010); международная конференция “4th Workshop on Advanced

Numerical Methods for Partial Differential Equation Analysis" (Санкт-Петербург, 22 августа 2011г.); III Научно-практическая конференция "Суперкомпьютерные технологии в нефтегазовой отрасли. Математические методы, программное и аппаратное обеспечение" (МГУ, 28-30 ноября 2012г.).

Публикации. Основные материалы диссертации опубликованы в 13 печатных работах: Vassilevski 6 статей – в рецензируемых журналах, входящих в перечень ВАК, [57, 59–63] и 6 статей – в сборниках научных трудов [58, 64–68], одна монография [1].

Личный вклад автора. В совместной работе [60] заключался в параллелизации существующей модели общей циркуляции океана, данный опыт лег в основу библиотеки для работы с сеточными данными. В совместных работах [64, 65] вклад заключался в разработке ряда технологий для моделирования течения жидкости со свободной поверхностью, а именно в дискретизации операторов адвекции и решение задачи Пуассона, метода коррекции функции уровня по частицам, создании технологической цепочки для задания нетривиального течения в сложной геометрической области а так же в постановке и проведении численных экспериментов. В совместных работах [57–59] вклад заключался в разработке технологии моделирования течения вязкопластичной несжимаемой жидкости со свободной границей, а именно дискретизации оператора дивергенции от тензора напряжений, технологической цепочки для задания сложных геометрических сцен, верификации реализованного метода, постановке и проведении численных экспериментов со сходом оползня и разрушения дамбы. В совместных работах [61, 62] вклад заключался в реализации динамических сеток типа восьмеричное дерево и полностью неявного метода для решения задачи двухфазной фильтрации в пористой среде. Был предложен критерий сгущения, разгрубления, интерполяции и произведена верификация метода. В совместной работе [63, 68] вклад автора за-

ключался в разработке конечно-разностного неявного метода для решения задачи конвекции-реакции-диффузии. Так же автором была обнаружена неустойчивость и предложен и реализован метод стабилизации ложных мод скоростей. Проведен ряд численных экспериментов для апробации метода и сравнения с референтными значениями.

Структура и объем диссертации. Диссертационная работа состоит из введения, обзора используемой терминологии, трех глав, заключения и списка литературы из 79 наименований. Диссертационная работа содержит 39 рисунков и 18 таблиц. Общий объем диссертационной работы – 115 страниц.

Благодарности

В первую очередь автор выражает благодарность своим самым близким людям: жене Тереховой Юлии и сыну Терехову Льву. Своим родителям Тереховой Наталье и Терехову Михаилу. Автор диссертационной работы выражает глубокую признательность научному руководителю Ю. В. Василевскому за продолжительную поддержку, ценные советы и плодотворное обсуждение вопросов. Автор благодарен С. Ю. Малясову и В. Г. Дядечко из Upstream Research Center of ExxonMobil corp. за помощь в постановке задачи о практическом моделировании процесса двухфазной фильтрации в пористой среде. Автор также выражает благодарность М. А. Ольшанскому, И. В Капырину, А. А. Данилову, К. Д. Никитину, И. Н. Коньшину и многим другим за помощь в обсуждении идей и методов, используемых в диссертационной работе.

Работа над диссертацией была частично поддержана грантами РФФИ 09 - 05 - 01231, 09 - 01 - 12029 офи-м, 11 - 01 - 00971, 11 - 01 - 00767, 12 – 01 – 00283, 08 - 01 - 00159 - а, 09 - 01 - 00115 - а, 12 - 01 - 31275, 12 - 01 - 33084, программой Президиума РАН “Алгоритмы и математическое обеспечение для вычислительных

систем сверхвысокой производительности", федеральной целевой программой "Научные и научно-педагогические кадры инновационной России", федеральной целевой программой "Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России целевой программой "Research and educational human resources for innovative Russia 2009-2013 грантом Upstream Research Center of ExxonMobil corp, а так же проектом Росатома "Прорыв".

Обзор используемой терминологии

Введем необходимые понятия, которые будут использоваться в настоящей диссертационной работе.

В работе рассматриваются следующие классы расчетных сеток. Во второй главе схемы формулируются для *конформных* сеток, любые два элемента которых либо не имеют общих точек, либо имеют ровно одну общую вершину, либо одно общее целое ребро, либо одну общую целую грань. Сетки типа восьмеричное дерево (см. рис. 1), с формальной точки зрения не являются конформными. Однако, можно считать кубические ячейки сетки многогранниками с гранями, лежащими в одной плоскости, и рассматривать сетку типа восьмеричное дерево как конформную.

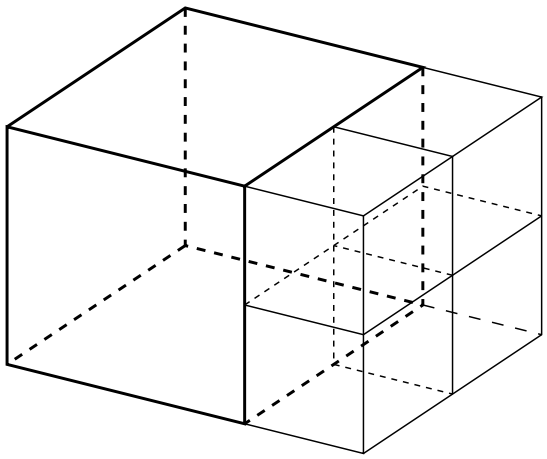

Рис. 1. Сетка типа восьмеричное дерево.

Каждая ячейка сетки является *ячейкой звездного типа* относительно центра масс, то есть каждая грань полностью видна из центра масс ячейки. Аналогично каждая грань является плоской *гранью звездного типа* относительно центра масс грани.

Скалярное поле насыщенности (вторая глава) или давления (вторая и третья

глава) считаем кусочно-постоянным на ячейках расчетной сетки. *Точки степени свободы* – точки, в которых задаются независимые значения неизвестной величины – в данном случае выбираются в центрах масс ячеек. В процессе решения задачи может возникать необходимость вычислять значения неизвестных величин в дополнительных точках, называемых *опорными точками*.

При описании свойств пористой среды в задаче фильтрации используется *тензорный коэффициент абсолютной проницаемости* $\mathbb{K}$, описывающий зависимость скорости фильтрационного потока от свойств среды без учета свойств жидкости. Для изотропных сред коэффициент $\mathbb{K}$ является скалярной величиной, для анизотропных – представляет собой симметричный положительно определенный тензор размерности 3×3. В задаче конвекции-диффузии тензору абсолютной проницаемости соответствует *тензор диффузии*. Для любого тензора диффузии существует система координат, в которой он принимает диагональный вид.

Вектор нормали к грани сетки, умноженный на тензор диффузии, называется *вектором конормали*. Сетка называется $\mathbb{K}$-*ортогональной*, если вектор, соединяющий точки степеней свободы соседних ячеек, сонаправлен вектору конормали к общей грани этих ячеек.

Глава 1

Система для работы с сеточными данными

1.1. Базовое ядро для операций с сетками

Базовое ядро для операций с сетками должно обеспечивать необходимый функционал для таких широко используемых для решения систем дифференциальных уравнений методов, как конечные разности, конечные объемы, конечные элементы. Эти методы требуют дискретизации области на, в общем случае, многогранные элементы. Такая дискретизация области называется сеткой. Поэтому представление сеточных данных и операций над ними играют важную роль в приближенном решении систем дифференциальных уравнений.

1.1.1. Сеточные элементы

Перечислим базовые сеточные элементы: узел, ребро, грань, ячейка. Узел содержит информацию о своем положении в пространстве, ребро состоит из двух вершин, грань состоит из ребер, ячейка из граней.

1.1.2. Иерархия связей сеточных элементов

Для восстановления геометрии элементов требуется узнать из каких элементов они состоят. Также нам потребуется не только геометрическая информация о данном элементе, но и информация о соседних элементах по отношению к данному. При отсутствии такой информации запрос соседних элементов по отношению к данному был бы алгоритмически сложной задачей.

Мы будем использовать иерархию связей изображенную на Рис. 1.1. Такая

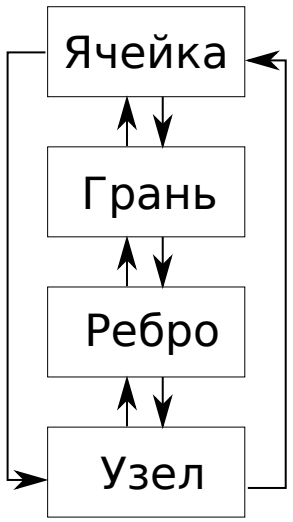

Рис. 1.1. Иерархия множеств.

иерархия позволит быстро запрашивать соседние элементы по отношению к данному, проходя вверх по иерархии, и элементы, из которых состоит данный элемент, проходя вниз по иерархии.

В итоге мы получим некоторый связный граф элементов. Мы не будем хранить направленность связей, однако, можно легко удостовериться в том, что подобную направленность можно восстановить исходя из иерархии элементов.

1.1.3. Множества

Чтобы работать с множествами самых различных элементов, введем в структуру базового ядра помимо узла, ребра, грани и ячейки понятие множества, которое может состоять из этих элементов. Определенное нами множество будет разрешать различные операции, характерные для множеств, такие как пересечение $\cap$, объединение $\cup$, разность $\setminus$ элементов.

Для удобства, мы будем запоминать принадлежность элемента множеству. Тогда при модификации или удалении элемента мы сможем скорректировать мно-

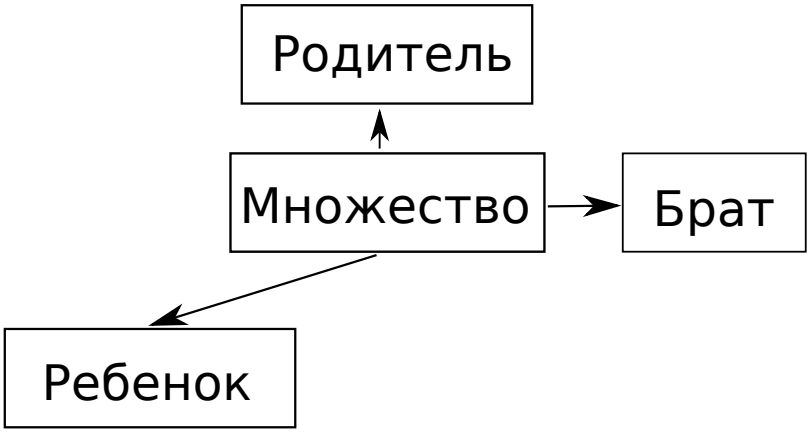

Рис. 1.2. Иерархия множеств.

жаства, которым он принадлежит, чтобы избежать ошибки в дальнейшем.

1.1.4. Иерархия множеств

Для реализации структуры типа дерево над множеством сеточных элементов введем иерархию множеств. Каждое множество может хранить три ссылки на другие множества(Рис. 1.2): на множество-отца, множество-брата и множество-ребенка. Множество-брат будет являться следующим по очереди ребенком для множества-отца данного множества. Множество-ребенок является самым первым множеством ниже по иерархии от данного. Такой механизм позволяет представить произвольную структуру типа дерево.

1.1.5. Сеточные данные

Один из ключевых моментов, необходимых для полного описания структуры базового ядра – это данные. Для узла данными будут его координаты. Для методов решения дифференциальных уравнений данными будут температура, скорость, давление, концентрация, тензор напряжений и т.д. Для составления систем линейных уравнений будет необходим глобальный идентификатор элемента, кото-

рый будет определять его строку/столбец в матрице.

Некоторые данные могут быть определены как на всех элементах, так и на некоторых, например, только на гранях. Некоторые типы данных, например, пометка о типе граничных условий, может стоять только на граничных гранях, поэтому нужен гибкий механизм возможностей задания данных. Данные могут быть определены на узлах, ребрах, гранях, ячейках, на множествах, на всей сетке. Данные могут представлять из себя целое число, действительное число или массив действительных или целых чисел. Мы будем различать данные фиксированной длины и переменной длины.

1.1.6. Ярлык данных

Для управления данными введем понятие ярлыка. Каждый ярлык определяется своим именем, содержит информацию, где и как искать данные для элемента каждого типа, для ячейки, грани, ребра, узла, либо для множества, либо для сетки. Ярлык обозначает, какому типу данных он соответствует: целочисленным, действительным или байтовым. Так же в ярлыке хранится информация, является ли он плотным, т.е. создан в каждом элементе данного типа, или разреженным, то есть принадлежит только некоторым элементам данного типа и является ли длина данных фиксированной или переменной. Каждый ярлык хранится в сетке и может быть запрошен по имени.

1.2. Операции модификации сетки

Для измельчения сетки нам потребуются следующие операции: поделить ребро пополам с помощью заданных вершин, заменить грань на ряд граней с заданными ребрами, заменить ячейку на ряд ячеек с заданными гранями. Для раз-

грубления ячейки нам потребуются обратные операции: объединить ряд ячеек, связанных гранями, объединить ряд граней, связанных ребрами, объединить два ребра, связанные одной вершиной.

Так как нам потребуется определить физические данные на новых сеточных элементах, данные операции будут разделены на два этапа. Будем различать с помощью ярлыка новые и старые сеточные элементы.

На первом этапе будут созданы новые сеточные элементы. Между новыми и старыми элементами зададим только одностороннюю связь от первых ко вторым. Тогда мы сможем запрашивать из новых элементов старые и проходить между старыми элементами по старой иерархии для запроса необходимых данных. Мы запомним обратную связь между старыми и новыми элементами с помощью ярлыка, так как при повторении операции над тем же старым элементом нам следует вернуть уже полученные новые элементы.

На втором этапе старые замененные элементы будут удалены, а в новых задана двусторонняя связь. При операции объединения на втором этапе так же будут удалены все элементы на один уровень ниже по иерархии, связывающие объединяемые элементы.

При объединении ячейки, мы будем создавать новые ребра и грани, из которых состоит объединенная ячейка. Тогда последующие операции объединения граней данной ячейки и ребер данной ячейки необходимо производить над множеством данных новых элементов.

1.3. Адаптивные сетки типа восьмиричное дерево

Для поддержания иерархии типа восьмеричное дерево мы будем использовать уже имеющуюся иерархию множеств. Множества данного дерева множеств

будут содержать максимум по 8 ячеек сетки. При измельчении ячейки, принадлежащей данному множеству, мы создаем новое множество-ребенка, к которому прикрепляем 8 новых мелких ячеек, а старую ячейку удаляем из данного множества. При разгрублении 8 ячеек, принадлежащих данному множеству, мы их удаляем из данного множества, а к множеству-отцу прикрепляем новую грубую ячейку.

Опишем измельчение ячейки. Если соседние ячейки меньше данной, то часть вершин уже может существовать. Мы сначала создадим не существующие вершины в серединах граней и ребер ячейки, а так же в центре самой ячейки. По новым вершинам разделим ребра, на которых они лежат. Новые вершины в центрах граней соединим ребрами с вершинами в центрах ребер. Вершину в центре ячейки соединим ребром с вершинами в гранях. По полученным ребрам создадим новые грани. Для тех новых граней, которые лежат на старых гранях выполним операцию замены граней. Затем мы можем создать новые ячейки и так же выполнить операцию замены ячеек.

При разгрублении ячейки, мы сначала выполним операцию объединения ячеек, в результате которой получим крупную ячейку с измельченными гранями. Мы объединим те (новые) грани грубой ячейки, оба соседа которых совпадают. Затем мы объединим те (новые) ребра, вершина между которыми соседствует только с четырьмя ячейками.

1.4. Параллельные алгоритмы

В следующих алгоритмах для параллельной работы с сеточными данными мы будем считать, что сетка будет статическая. Динамически адаптирующиеся параллельные сетки не будут рассматриваться в данной работе. Далее мы так

же будем считать, что передача данных между процессорами осуществляется с помощью интерфейса межпроцессорных коммуникаций MPI[1].

1.4.1. Свойства распределенных элементов

При решении систем дифференциальных уравнений одним из методов: конечных элементов, конечных разностей или конечных объемов, – ключевой операцией является получение соседних ячеек. Следовательно, при параллельном расчете необходимо создать приграничный слой фиктивных элементов вокруг принадлежащих данному процессору элементов. Для этого нам необходимо уметь вычислять множество граней, ребер и вершин, которые лежат между ячейками, принадлежащими двум разным процессорам, уметь упаковывать геометрические данные ячеек и пересылать их между процессорами. Так же нам понадобиться синхронизировать данные между элементами и эффективно балансировать и перераспределять сеточные элементы.

Введем несколько ярлыков данных, необходимых нам для дальнейшего расширения функционала, предназначенного для распределенных сеточных данных.

- “Состояние” –ярлык, которому соответствует состояние элемента:
  - собственный – элемент, которым владеет только данный процессор;
  - общий – элемент, которым владеет данный процессор, но копия элемента имеется у других процессоров;
  - фиктивный – копия элемента, которым данный процессор не владеет.
- “Владелец” – ярлык, которому соответствует число, обозначающее идентификатор процессора-владельца элемента.

[1] MPI – Message Passing Interface

- “Процессоры” – ярлык, которому соответствует отсортированный массив процессоров, указывающий, какие процессоры имеют копию данного элемента, помимо данного процессора. Такой же массив хранится в сетке для определения, с какими процессорами существуют связи в данной сетке.

1.4.2. Обмен данными

Будем считать, что известно некоторое множество $\mathcal{P}$ номеров процессоров, с которыми должен обмениваться данный процессор. Тогда неблокирующий обмен данными при известном размере принимаемого сообщения будет состоять из четырех шагов:

- для каждого $p \in \mathcal{P}$ данный процессор выставляет запрос $r \in \mathcal{R}$ на прием сообщения заданной длины в заданный буфер с помощью операции “MPI_Irecv”;
- для каждого $p \in \mathcal{P}$ данный процессор выставляют запрос $q \in \mathcal{Q}$ на отправку сообщения заданной длины из заданного буфера с помощью операции “MPI_Isend”;
- для запросов $\mathcal{R}$ данный процессор ожидает окончания какого-либо запроса на прием r с помощью операции “MPI_Waitsome” и обрабатывает соответствующие r принятые данные;
- данный процессор ожидает окончания посылки всех данных выполняя операцию “MPI_Waitall” над множеством $\mathcal{Q}$.

Если изначально размер сообщения не известен, то мы воспользуемся тем же алгоритмом для обмена размерами посылаемых сообщений.

Так как пересылки асинхронные, нам нельзя допустить, чтобы случайно были приняты неправильные данные, например, от предыдущих незавершенных пе-

ресылок к данному процессору. Для этого мы для каждой пары приема-посылки будем использовать свой tag, который будет вычисляться из номера посылающего процессора, принимающего процессора, общего числа процессоров и псевдо-произвольного числа, одинакового на всех процессорах.

1.4.3. Глобальная нумерация элементов

Поставим каждому элементу заданного типа в соответствие уникальный номер:

1. Посчитаем, сколько в сумме собственных и общих элементов заданного типа есть на данном процессоре.

2. С помощью операции “MPI_Scan” узнаем число, с которого следует начинать нумерацию элементов на данном процессоре.

3. Пронумеруем элементы.

1.4.4. Восстановление параллельного состояния сеточных элементов

Вполне вероятна ситуация, когда заранее информация о состоянии, владельце и процессорах недоступна, и мы имеем только геометрическую информацию о сетке. Тогда такую информацию следует восстановить.

Для начала, мы ограничим число процессоров, с которыми следует общаться данному процессору. Для этого пробежим по всем вершинам сетки, принадлежащей данному процессору и вычислим по координатам всех вершин границы окаймляющего данную сетку куба. Затем все процессоры могут обменяться границами своих кубов с помощью операции “MPI_Allreduce” и каждый процессор может вычислить по пересечению своего окаймляющего куба с чужыми множество соседних процессоров. В итоге получим временное множество $\mathcal{P}$.

Затем каждый процессор собирает свой массив, состоящий из координат узлов, сортирует его и обменивается (§ 1.4.2) им с другими процессорами. Пробегая по двум отсортированным массивам координат – своему и принятому от процессора p, при совпадении координат мы будем добавлять в массив, соответствующий ярлыку “процессоры” для узла с заданными координатами, номер процессора p.

Замечание 1.4.1. *Определим владельцем каждого узла процессор с наименьшим номером из соответствующего массива “процессоры”. Определим состояние узла:*

1. *Если массив процессоров пустой, то элемент собственный.*
2. *Если массив не пустой, но владеет элементом данный процессор, то элемент общий.*
3. *Иначе элемент фиктивный.*

Объединяя множества процессоров каждого элемента, мы получим минимальное множество всех процессоров $\mathcal{P}$, с которыми должен обмениваться данный процессор.

Для каждого ребра мы можем посчитать массив процессоров, которым он принадлежит, найдя пересечения множеств процессоров, которым принадлежат его узлы. Но, если два узла ребра принадлежат определенному процессору, само ребро не обязательно принадлежит этому процессору. Для определения множества ребер, которые принадлежат другому процессору, мы выполним следующие шаги. Пусть номер данного процессора p_0.

1. Найдем глобальную нумерацию для вершин (§ 1.4.3).

2. Составим массив пар $\mathbb{G}$, состоящих из локального и глобального номера ребра и отсортируем его по локальному номеру.

3. Соберем в множество $\mathcal{A}_p$ ребра, которые потенциально могут принадлежать соседнему процессору p по принадлежности этому процессору узлов данного ребра. Для каждого соседнего процессора p соберем следующий числовой массив $\mathbb{B}_{p_0,p}$:

   - количество потенциально принадлежащих процессору p ребер;
   - количество узлов, из которых состоят ребра;
   - глобальные номера этих узлов.

4. Процессор обменивается (§ 1.4.2) со всеми процессорами из $\mathcal{P}$ соответствующими числовыми массивами $\mathbb{B}_{p_0,p}$. При приеме $\mathbb{B}_{p,p_0}$:

   - возьмем глобальные номера узлов, из которых состоит принятое ребро, которое следует найти;
   - находим по глобальным номерам узлов сами узлы с помощью бинарного поиска по $\mathbb{G}$;
   - если все узлы найдены, мы можем по иерархии вверх найти само ребро;
   - если ребро найдено, то добавляем его в множество $\mathcal{A'}_p$

5. Когда вся принятая информация обработана, найдем пересечение множеств $\mathcal{C}_p = \mathcal{A}_{p_0,p} \cap \mathcal{A'}_{p_0,p}$.

6. Множества $\mathcal{C}_p \forall p \in \mathcal{P}$ является искомыми. Для всех ребер из $\mathcal{C}_p$ пометим, что они есть на процессоре p.

Мы можем поступить аналогично с гранями и ячейками, заменив в алгоритме везде вершины на ребра и грани соответственно. Далее мы можем определить “владельца” и “состояние” элемента как в Замечании 1.4.1.

Если изначально в сетке были слои фиктивных ячеек, то все эти слои заберет себе процессор с меньшим номером, что не всегда бывает удобно. Поэтому для определенности после выполнения описанной процедуры можно удалить все фиктивные ячейки. Способ удаления фиктивных ячеек будет описан в § 1.4.6.

1.4.5. Синхронизация и аккумуляция данных элементов

Имея всю необходимую информацию, описанную в § 1.4.1, мы можем определить процедуру обмена данными из общих ячеек одного процессора в фиктивные ячейки другого процессора. Назовем данную процедуру синхронизацией. Для синхронизации нам потербуется упаковывать и распаковывать данные.

Упаковка данных

Параметры:

- ярлык данных tag, которые следует упаковать;
- сортированное множество элементов $\mathcal{E}$, для которых следует упаковать данные;
- промежуточный байтовый массив $\mathbb{B}$, в который мы будем упаковывать данные.

Создадим два массива: один численный $\mathbb{N}$, другой байтовый $\mathbb{C}$.

В первых двух позициях численного массива мы в конце алгоритма запишем количество длину численного массива и длину байтового массива.

Пройдем по всем элементам множества $\mathcal{E}$, для элемента $e \in \mathcal{E}$:

- для разреженных данных, если данные, соответствующие tag присутствуют на e, добавим в $\mathbb{N}$ позицию элемента в $\mathcal{E}$, иначе перейдем к следующему элементу;
- поместим в массив $\mathbb{C}$ данные элемента e соответствующие tag;
- если данные имеют переменную длину, то поместим длину данных в $\mathbb{N}$.

Заполненные массивы запишем с помощью “MPI_Pack” в конец буфера $\mathbb{B}$.

Распаковка данных

Параметры:

- ярлык tag распаковываемых данных;
- отсортированное множество элементов $\mathcal{E}$, в которые следует распаковать данные;
- буфер $\mathbb{B}$, из которого следует распаковывать данные;

Так как мы запаковали первыми двумя значениями длины массивов $\mathbb{N}$ и $\mathbb{C}$, то, зная эти длины, мы можем извлечь данные массивы из $\mathbb{B}$ с помощью функции “MPI_Unpack”.

Если ярлык tag соответствует плотным данным, то пройдем по всем элементам множества $\mathcal{E}$, для элемента $e \in \mathcal{E}$ и скопируем данные нужной длины из текущей позиции в $\mathbb{C}$ в данные элемента e, а затем переместим текущую позицию на длину скопированных данных и перейдем к следующему элементу.

Если ярлык tag соответствует разреженным данным, мы сначала удалим все соответствующие ему данные из $\mathcal{E}$. Затем мы будем получать из $\mathbb{N}$ положение

следующего элемента $e \in \mathcal{E}$, для которого есть данные, длина которых записана следующим числом в $\mathbb{N}$. Мы скопируем эти данные из $\mathbb{B}$ в элемент e.

Эти алгоритмы позволяют в один и тот же буфер $\mathcal{B}$ поочередно упаковывать или распаковывать данные, соответствующие нескольким ярлыкам, если мы будем запоминать последнюю позицию в буфере.

Синхронизация данных

Опишем теперь алгоритм синхронизации данных между процессорами. Предположим, что разметка по процессорам, владельцу и состоянию всех элементов не нарушена. Тогда, если взять два процессора P_1 и P_2, количество фиктивных элементов у процессора P_2, на которых стоит пометка, что владелец P_1, должно совпадать с количеством общих элементов у процессора P_1, у которых в массиве процессоров присутствует процессор P_2. Чтобы множества соответствующих фиктивных и общих элементов совпадали на разных процессорах, нам следует одинаковым образом отсортировать эти два множества. При наличии глобальной нумерации можно отсортировать элементы по глобальным номерам. При отсутствии глобальной нумерации, мы можем отсортировать элементы по их барицентрам.

Пусть нам необходимо переслать данные, помеченные ярлыком tag. Пройдем по множеству соседних процессоров $\mathcal{P}$. Для каждого $P \in \mathcal{P}$:

1. проходим по всем элементам данного процессора, на которых определен ярлык, проверяем статус, владельца и массив процессоров данного элемента:

   - если элемент общий, и в его массиве процессоров присутствует процессор P, то помещаем его в множество общих элементов $\mathcal{S}_P$;

- если элемент фиктивный и его владельцем является процессор P, то помещаем его в массив фиктивных элементов $\mathcal{G}_P$.

2. Если массив общих элементов $\mathcal{S}_P$ не пуст, то отсортируем его и упакуем данные в буфер $\mathbb{B}_P$ для посылки процессору P.

Выполним обмен (§ 1.4.2) собранными буферами $\mathbb{B}_P \forall P \in \mathcal{P}$. При приеме данных от некоторого процессора P, отсортируем соответствующее множество элементов $\mathcal{G}_P$ и распакуем в него принятые данные.

Сортировать множества общих и фиктивных элементов при многократном выполнении синхронизации данных будет вычислительно сложно. Мы можем сделать это один раз и запомнить множества $\mathcal{G}_P$ и $\mathcal{S}_P$. При добавлении или удалении фиктивных и общих элементов, мы можем создать промежуточные множества $\mathcal{G}_P^*$ и $\mathcal{S}_P^*$, в которые будем добавлять удаленные или добавленные элементы, а затем воспользоваться операцией разности ($\mathcal{G}_P = \mathcal{G}_P \setminus \mathcal{G}_P^*$, $\mathcal{S}_P = \mathcal{S}_P \setminus \mathcal{S}_P^*$) или объединения элементов множеств ($\mathcal{G}_P = \mathcal{G}_P \cup \mathcal{G}_P^*$, $\mathcal{S}_P = \mathcal{S}_P \cup \mathcal{S}_P^*$) соответственно.

Аккумуляция данных

Алгоритм синхронизации данных переносит данные в одну сторону – из общих ячеек в фиктивные ячейки. В некоторых случаях может потребоваться обработать некоторым образом данные, помещенные как в общие, так и в фиктивные ячейки. Например, найти сумму или произведение значений, причем эта операция определяется заданной пользователем функцией. Назовем такую операцию “аккумуляцией данных”. Алгоритм работает аналогично алгоритму синхронизации данных, только посылаются данные из фиктивных ячеек, а принимаются в общие. При приеме вместо копирования данных в элемент вызывается функция пользователя, которая должна обработать принятые данные. Одной собственной

ячейке может соответствовать несколько фиктивных, тогда при приеме данных функция пользователя будет вызываться многократно. После того как данные были аккумулированы в общих ячейках, следует выполнить алгоритм синхронизации, чтобы те же данные появились в фиктивных ячейках.

Частичный обмен

Если нет необходимости обмениваться данными между всеми общими и фиктивными элементами, то можно ввести специальный ярлык $\mathbb{F}$, который будет служить фильтром над множеством общих и фиктивных элементов. Тогда, при упаковке и распаковке данных будем пропускать элементы, помеченные $\mathbb{F}$. Чтобы множества соответствующих фиктивных и общих элементов были не противоречивыми на разных процессорах, каждый процессор может пометить свои элементы через $\mathbb{F}$, данными которых он хочет обмениваться, а затем выполнить алгоритм аккумуляции данных над $\mathbb{F}$.

1.4.6. Удаление фиктивных элементов

Удалим заданные фиктивные ячейки. После удаления, возможно, остались фиктивные грани, не связанные с сеткой. Заведем новый ярлык $\mathbb{I}$ целочисленных разреженных данных произвольной длины для граней, с помощью которого мы будем помечать, удаляет ли данный процессор эти элементы.

1. пройдем по фиктивным граням, если число соседних ячеек равно нулю, то записываем в ярлык $\mathbb{I}$, что процессор с данным номером удаляет эту грань;

2. используем алгоритм аккумуляции данных (§ 1.4.5) над ярлыком $\mathbb{I}$, чтобы получить на собственных элементах процессора-владельца массив, который обозначает, какие процессоры удаляют эту грань;

3. убираем из массива процессоров для грани те процессоры, которые ее удаляют, если массив оказался пустым, то помечаем грань на процессоре-владельце как собственную;

4. все остальные процессоры, у которых данная грань является фиктивной, и количество соседних ячеек равно нулю, также удаляют эту грань.

Применим тот же самый алгоритм для ребер и узлов.

Данный алгоритм можно использовать также для удаления только части фиктивных элементов.

1.4.7. Упаковка и распаковка множества

Упаковка множества

Чтобы удаленный процессор мог восстановить в полном объеме геометрическую информацию о полученных элементах, необходимо передать удаленному процессору все элементы, лежащие в иерархии ниже по отношению к данному, а также связи по иерархии вниз. Связи по иерархии вверх удаленный процессор сможет восстановить сам. Данные элементов множества мы уже умеем упаковывать и передавать (§ 1.4.5). Считаем, что все упаковки и распаковки данных выполняются посредством функций “MPI_Pack” и “MPI_Unpack”. Каждая соответствующая упаковка записывается в конец буфера, а каждая распаковка сдвигает позицию в буфере.

Параметры алгоритма:

- множество упаковываемых элементов $\mathcal{E}$;
- буфер $\mathbb{B}$, в который совершается упаковка;
- номер процессора $P \in \mathcal{P}$, для которого совершается упаковка;

- список имен ярлыков $\mathcal{T}$, соответствующие данные которых следует передать.

Переберем множество элементов из $\mathcal{E}$ и соберем по отдельности множества вершин $\mathcal{N}_\mathcal{E}$, ребер $\mathcal{E}_\mathcal{E}$, граней $\mathcal{F}_\mathcal{E}$ и ячеек $\mathcal{C}_\mathcal{E}$.

Затем перебираем эти множества и добавим в соответствующие множества элементы, лежащие по иерархии ниже:

- для ячеек $C \in \mathcal{C}_\mathcal{E}$: грани в $\mathcal{F}_\mathcal{E}$, ребра в $\mathcal{E}_\mathcal{E}$, вершины в $\mathcal{N}_\mathcal{E}$;
- для граней $F \in \mathcal{F}_\mathcal{E}$: ребра в $\mathcal{E}_\mathcal{E}$ и вершины в $\mathcal{N}_\mathcal{E}$;
- для ребер $E \in \mathcal{E}_\mathcal{E}$: вершины в $\mathcal{N}_\mathcal{E}$.

Для вершин $N \in \mathcal{N}_\mathcal{E}$ упаковываем их количество и координаты всех вершин, если существует глобальная нумерация, то мы также упакуем глобальную нумерацию. Для ребер $E \in \mathcal{E}_\mathcal{E}$ (граней $F \in \mathcal{F}_\mathcal{E}$, ячеек $C \in \mathcal{C}_\mathcal{E}$) упаковываем их количество, затем массив длин, обозначающий по сколько связей по иерархии вниз имеет каждое ребро E (грань F, ячейка C), а затем массив позиций связей по иерархии вниз в упакованном ранее множестве вершин $\mathcal{N}_\mathcal{E}$ (ребер $\mathcal{E}_\mathcal{E}$, граней $\mathcal{F}_\mathcal{E}$ соответственно). Для удобства нахождения данных позиций, мы можем отсортировать множества $\mathcal{N}_\mathcal{E}$, $\mathcal{E}_\mathcal{E}$, $\mathcal{F}_\mathcal{E}$ по локальным идентификаторам элементов, а затем воспользоваться бинарным поиском.

В процессе упаковки помечаем каждый “собственный” элемент как “общий” и добавляем в массив “процессоров” элемента процессор P. Также каждый элемент, для которого владельцем не является процессор P, запишем в отдельное множество $\mathcal{Q}$.

Отсортируем собранное множество элементов $\mathcal{Q}$, для которых процессор P не является владельцем; для этих элементов мы будем записывать в буфер данные следующим образом:

- упакуем количество ярлыков передаваемых данных;
- для каждого ярлыка $t \in \mathcal{T}$ упакуем длину имени, имя, тип данных, являются ли данные плотные и разреженные, т.е. все данные, необходимые для создания ярлыка;
- воспользуемся ранее описанным алгоритмом из § 1.4.5 для упаковки данных для каждого ярлыка и собранного множества элементов $\mathcal{Q}$ в текущий буфер.

Некоторые данные, соответствующие ярлыкам "состояние", "владелец", массив "процессоров" не следует передавать. Добавим процессор P в множество процессоров $\mathcal{P}$.

Распаковка множества

Параметры алгоритма:

- множество $\mathcal{E}$, в которое будут записаны распакованные элементы;
- буфер $\mathcal{B}$, из которого распаковываются данные;
- номер процессора P, приславшего данные.

Создадим и отсортируем по координате, либо по глобальной нумерации множество всех вершин $\mathcal{N}$, имеющихся на данном процессоре. Создадим множество распакованных вершин $\mathcal{N}_\mathcal{E}$, ребер $\mathcal{E}_\mathcal{E}$, граней $\mathcal{F}_\mathcal{E}$ и ячеек $\mathcal{C}_\mathcal{E}$.

Распакуем количество вершин, затем их координаты, и если есть, глобальные номера. С помощью бинарного поиска по распакованным координатам, либо по глобальным номерам, проверим, существует ли вершина в $\mathcal{N}$. Если существует, то добавим вершину во множество $\mathcal{N}_\mathcal{E}$, иначе создадим и добавим вершину по полученным координатам.

Распакуем количество ребер (граней, ячеек), массив длин и номера вершин (ребер, граней) в множестве $\mathcal{N}_\mathcal{E}$ ($\mathcal{E}_\mathcal{E}$, $\mathcal{F}_\mathcal{E}$) из которых состоит ребро (грань, ячейка соответственно). Проверим, существует ли ребро (грань, ячейка), состоящее из заданных вершин (ребер, граней), если есть, то добавим его в $\mathcal{E}_\mathcal{E}$ ($\mathcal{F}_\mathcal{E}$,$\mathcal{C}_\mathcal{E}$), иначе создадим и добавим элемент в эти множества.

В процессе распаковки будем помечать каждый полученный несобственный элемент как "фиктивный". Будем также добавлять каждый полученный несобственный элемент в отдельное множество $\mathcal{Q}$.

Отсортируем множество несобственных элементов $\mathcal{Q}$ и распакуем количество полученных ярлыков данных. Для каждого ярлыка:

- распакуем длину имени и имя ярлыка и запросим у сетки сам ярлык, если ярлыка не существует, то мы можем его создать по всей полученной информации
- воспользуемся описанным ранее алгоритмом распаковки данных во множество несобственных элементов.

Добавим процессор P, приславший данные, в $\mathcal{P}$.

Замечание 1.4.2. *Мы могли бы избежать необходимости поиска вершин по координатам или по глобальной нумерации в множестве $\mathcal{N}$, если бы мы узнали локальную нумерацию вершин, присутствующих на удаленных процессорах, а затем использовали ее при упаковки и распаковки множества. Для этого каждый процессор может записать локальную позицию для "общих" и "фиктивных" элементов и свой номер процессора, а затем выполнить аккумуляцию этих данных, записывая принимаемые данные о локальной нумерации на удаленном процессоре в конец массива. Такой подход будет эффективным, если вершин на процессоре много, а обмены дешевые.*

1.4.8. Обмен элементами

Пусть у нас есть три процессора P_1, P_2, P_3, и пусть процессор P_1 передает процессору P_2 элементы. При этом процессор P_1 мог упаковать часть элементов, исходно принадлежащих процессору P_3 и передать их процессору P_2. В итоге процессор P_3 не будет знать о том, что процессор P_2 получил его элементы, и при синхронизации данных мы придем к тому, что на процессоре P_2 фиктивных элементов от процессора P_3 будет больше, чем на процессоре P_3 общих элементов с процессором P_2. Чтобы не возникла такая ситуация, необходимо, чтобы P_2 проинформировал P_3 о том, что он получил эти элементы.

Опишем алгоритм сборщика, который будет выполнять пересылки всех помеченных на пересылку элементов, а также выяснять, какие процессоры не были проинформированы о копиях элементов, и информировать их. Пусть номер данного процессора P_0 и пусть пользователь или программа пометили с помощью ярлыка $\mathbb{I}$ некоторые элементы сетки на пересылку некоторым процессорам. $\mathbb{I}$ соответствует целочисленным разреженным массивам переменной длины, элемент массива соответствует номеру процессора P, которому следует передать элемент.

1. Создадим массив связанных пар $(P, \mathcal{E})$, состоящих из номера процессора и множества посылаемых процессору элементов.

2. Перебирем все элементы сетки, если на элементе присутствует пометка о том, что следует переслать данный элемент другому процессору, то добавим этот элемент в соответствующую ему связную пару в массиве $(P, \mathcal{E})$.

3. Перебирем массив связных пар $(P, \mathcal{E})$, упакуем (§ 1.4.7) множество элементов $\mathcal{E}$ вместе с данными и перешлем (§ 1.4.2) его соответствующему процессору P.

4. При приеме множества, проверим, присутствует ли в их массиве процессоров данный процессор P_0. Если данный процессор отсутствует, то процессор-владелец не проинформирован о получении. Мы добавляем такие элементы в массив связных пар $(P', \mathcal{E}')$, где P' – процессор-владелец.

5. Мы повторно обменяемся собранными элементами $(P', \mathcal{E}')$, не упаковывая данные. В результате при приеме каждый процессор-владелец автоматически добавит процессор P_0 в массив "процессоров" принятых элементов.

6. Если все ярлыки данных были заранее синхронизированы, то в результате всех проделанных выше операций синхронизированы будут все ярлыки, кроме ярлыка, соответствующего массиву "процессоров", поэтому мы синхронизируем его.

В итоге мы описали механизм, который позволит пользователю или программе запрашивать создание фиктивных элементов в любой области сетки. Используя подобный функционал, пользователь сам бы мог создать несколько приграничных слоев фиктивных элементов или выполнить балансировку или перераспределение сетки. Для удобства кратко опишем также и эти алгоритмы.

1.4.9. Добавление слоев фиктивных элементов

Для добавления одного приграничного слоя фиктивных элементов требуется определить множества граней, ребер или вершин, обозначим их $\mathcal{S}_P$, $\mathcal{E}_P$? $\mathcal{N}_P$, соответственно, которые находятся на разделе между ячейками данного процессора P_0 и процессора P, для которого требуется создать приграничный слой фиктивных элементов, $P \in \mathcal{P}$. Ячейки, которые должны будут войти в приграничный слой фиктивных элементов для соседнего процессора, принадлежат данному процессору и являются соседними по отношению к этому множеству.

Если фиктивные слои не были созданы ранее, то найти множество $\mathcal{S}_P$ просто - достаточно пройти по всем элементам и добавить в $\mathcal{S}_P$ элементы, “фиктивные” и “общие” с процессором P.

Если фиктивные слои уже созданы и мы не хотим их удалять, то найти данное множество будет сложнее. Создадим на гранях временный ярлык $\mathbb{I}_1$, соответствующий целочисленным разреженным данным переменной длины. Пройдем по “общим” и “фиктивным” граням и добавим в ярлык глобальный идентификтор ячейки и номер процессора-владельца ячейки. Выполним аккумуляцию данных $\mathbb{I}_1$, при приеме данных будем добавлять в конец массива глобальный идентификатор и владельца, если такой пары в массиве нет.

Завершив аккумуляцию, пройдем опять по всем “общим” и “фиктивным” граням. Далее, если у грани есть одна соседняя ячейка и она “общая” или “собственная”, либо, если есть две соседнии ячейки и одна из них “фиктивная”, а вторая “общая” или “собственная”, и в массиве, соответствующем $\mathbf{I}_1$ есть две пары, то мы из пар возьмем значение процессора-владельца P', не равного данному и добавим грань в множество $\mathcal{S}_{P'}$.

Если нам требуется слой фиктивных элементов относительно граней, то мы уже получили искомые множества $\mathcal{S}_P$. Мы не можем получить из $\mathcal{S}_P$ множества $\mathcal{E}_P$ и $\mathcal{N}_P$ напрямую, взяв элементы вниз по иерархии, так как сетки на данном процессоре P_0 и на удаленном процессоре P могут не иметь пересечение граней, но иметь пересечение ребер или вершин.

Полученные нами множества граней $\mathcal{S}_P$, вместе с множеством граничных граней $\mathcal{S}_\Gamma$ окаймляет множество собственных ячеек данного процессора P_0, обозначим его $\mathcal{S}$. Если нам нужны множества ребер $\mathcal{E}_P$ или вершин $\mathcal{N}_P$, то эти ребра и вершины должны лежать ниже по иерархии по отношению к элементам из $\mathcal{S}$. Собирем все такие элементы в множество $\mathcal{D}$.

Пусть нас интересуют множества $\mathcal{E}_P$, для множества $\mathcal{N}_P$ действия аналогичны. Создадим на ребрах временный ярлык $\mathbb{I}_2$, соответствующий целочисленным разреженным данным переменной длины. Переберем элементы из $\mathcal{D}$ и добавим в массив, соответствующий $\mathbb{I}_2$ номер данного процессора P_0. Выполним аккумуляцию данных ярлыка $\mathcal{I}_2$ и при приеме данных будем добавлять полученное число в массив. Пройдем повторно по элементам $d \in \mathcal{D}$ и для всех $P' \neq P_0$ в массиве $\mathbb{I}_2$ на элементе d, добавим d в множество $\mathcal{E}_{P'}$. В итоге мы получим искомые множества.

Хотя алгоритм нахождения данных множеств является довольно сложным, его сложность аналогична сложности алгоритма удаления фиктивных элементов. Если фиктивные слои уже есть, то выгоднее будет изменить их, чем удалить и переслать заново.

При обмене приграничными слоями фиктивных элементов может встретиться топология, в которой процессоры, соседние с данным процессором, не обладают необходимым количеством слоев элементов. Чтобы алгоритмы пользователя корректно работали на многопроцессорной системе, данный алгоритм должен гарантировать запрошенные слои независимо от топологии сетки и топологии разбиения сетки на процессоры. Чтобы предоставить эти слои, соседний процессор для начала должен получить слой от своего соседа. Таким образом, создание границ должно быть выполнено итерационно: сначала каждый из процессоров обменивается первым слоем, затем каждый процессор обменивается вторым слоем, и т. д. Таким образом, алгоритм обмена слоями будет выглядеть следующим образом.

Пусть задано число N слоев, которое необходимо создать и тип элементов T, по которым следует найти соседние ячейки. Найдем множества $\mathcal{S}_P(T) \forall P \in \mathcal{P}$ элементов типа T по описанному ранее алгоритму. Создадим множество $\mathcal{L}_P^K$, которое будем обозначать множество элементов, образующих K-ый слой фиктивных элементов на данном процессоре для процессора P.

1. Для каждого $P \in \mathcal{P}$:

   а. перебирем все элементы $e \in \mathcal{S}_P$, и найдем все их соседние ячейки, владельцем которых не является процессор P;

   б. в каждой найденной ячейке, если в ее массиве процессоров P не присутствует, помечаем, что она должна быть послана процессору P, и кладем ее в множество $\mathcal{L}_P^1$.

2. Запускаем цикл по от $N-1$ до 0:

   а. вызываем алгоритм из § 1.4.8, чтобы он переслал множества $\mathcal{L}_P^{N-K}$;

   б. если $K > 0$, для каждого $P \in \mathcal{P}$:

      i. пометим, что мы уже посещали ячейки из $\mathcal{L}_P^{N-K}$;

      ii. для ячеек из $\mathcal{L}_P^{N-K}$ найдем множество элементы ниже по иерархии типа T, назовем его $\mathcal{M}$;

      iii. найдем соседние для элементов из $\mathcal{M}$ ячейки, которые не были отмечены и владельцем которых не является P;

      iv. пометим найденные ячейки на пересылку процессору P, и положим их в множество $\mathcal{L}_P^{N-K+1}$.

   в. Уменьшаем K и переходим к началу цикла.

В результате последовательно все процессоры гарантированно обменяются несколькими слоями фиктивных ячеек. В данном алгоритме нет необходимости хранить все множества $\mathcal{L}_P^K$, достаточно хранить одно множество с предыдущего шага. Если алгоритму передано число слоев, меньшее, чем было до этого, то можно удалить лишние фиктивные элементы, не трогая нужные. Тогда алгоритм

выше завершиться без обменов, а каждый процессор может пометить, какой элемент кому должен был бы быть отправлен. Выполнив аккумуляцию этих данных, мы узнали бы на всех процессорах, какие элементы не должны были быть отправлены и их следует удалить.

1.4.10. Перераспределение и балансировка сетки

Опишем теперь алгоритм перераспределения и балансировки сетки. Допустим, что при помощи внешнего пакета (к примеру, Zoltan [9]) было рассчитано новое распределение ячеек по процессорам. В данном алгоритме мы будем опираться на алгоритм § 1.4.8 для обмена элементами, после обмена алгоритм должен разметить новых владельцев всех элементов согласно полученному распределению и изменить статусы. Если известно, что до перераспределения мы имели N слоев через соседние ячейки типа T, то алгоритм должен оставить то же количество слоев после своей работы.

Пусть ярлык $\mathbb{I}_1$ обозначает данные о новом владельце каждого элемента сетки, а ярлык $\mathbb{I}_2$ соответствует новому массиву процессоров для каждого элемента сетки. Запишем информацию о новом владельце ячеек через ярлык $\mathbb{I}_1$ и синхронизируем его. Для граней, ребер и вершин рассмотрим новых владельцев для элементов выше по иерархии и выберем наименьший номер. Новый массив процессоров $\mathbb{I}_2$ для каждого элемента вычислим, объединив процессоры-владельцы всех элементов лежащих по иерархии вверх.

Если число слоев N не 0, нам следует найти окаймляющее множество элементов типа T для нового распределения процессоров. Сначала мы соберем множество граней, у которых размер массивов, соответствующих $\mathbb{I}_2$ равен 2, что будет означать, что с двух сторон от грани ячейки будут принадлежать разным процессорам. Если тип T не грань, мы рассмотрим все элементы типа T по иерархии

ниже у найденных граней, в результате чего получим искомые множества. Затем мы выполним алгоритм § 1.4.9, с определенными отличиями. Так как известно, что $N \geq 1$ слоев уже есть, то у нас нет необходимости в обменах, а вместо пометки на отправку процессору P, мы будем добавлять его в массив новых процессоров $\mathbb{I}_2$, если он там отсутствует и пропускать элемент, если присутствует.

Выполним аккумуляцию и синхронизацию на ячейках для ярлыка $\mathbb{I}_2$, а затем повторно вычислим $\mathbb{I}_2$ на гранях, ребрах и вершинах, , объединив процессоры-владельцы всех элементов лежащих по иерархии вверх и также выполним аккумуляцию на гранях, ребрах и вершинах.

В итоге мы получили новую разметку для все элементов сетки. Теперь нам надо переслать сами элементы. Для этого мы пометим на отправку элементы всем процессорам, которые есть в новом массиве процессоров и нет в старом. Затем выполним алгоритм из § 1.4.8, при упаковке будем удалять те элементы, в новом массиве процессоров которых нет данного процессора.

Выполнив обмен, мы заменим старый массив процессоров и старого владельца элемента на новые и обозначим элемент “фиктивным”, если владельцем элемента не является данный процессор, “общим”, если в массиве процессоров больше одного процессора, а все остальные “собственными”.

1.5. Параллельный расчет

Для тестирования параллельной эффективности, рассмотрим задачу, которая будет поставлена далее в § 2, с тензором проницаемости $\mathbb{K}$, изображенном в плоскости на Рис. 1.3, по оси Oz значение тензора проницаемости равно 10. Нагнетающая скважина присоединена к ячейке в одном углу, а производящая в другом. В отличии от предыдущих задач скважины расположены на разной высоте по оси

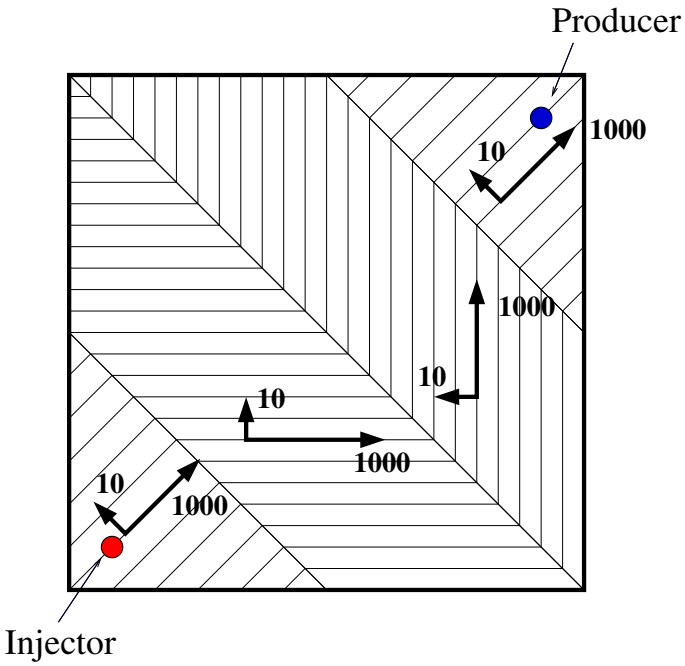

Рис. 1.3. Тензор проницаемости.

Oz. Сетка разрезана на параллелепипеды, размер которых определяется по числу процессоров $N_P = N_x \times N_y \times N_z$. Для тестирования мы возьмем две возмущенные сетки размеров $32 \times 32 \times 32$ с общим числом 51200 неизвестных и $64 \times 64 \times 64$ с общим числом 335872 неизвестных, обозначим их $\mathcal{A}$ и $\mathcal{B}$ соответственно. Для решения систем линейных уравнений был применен метод BiCGStab с предобуславливанием с помощью аддитивного метода Шварца с одним перекрытием. Эффективность расчета была проверена на двух параллельных машинах: кластере ИВМ РАН AltixXE310 и суперкомпьютере МГУ им. Ломоносова Bluegene/P. Время расчета приведено в Таблицах 1.1 и 1.2 соответственно.

Из таблиц видно, что на Bluegene/P метод демонстрирует ускорение, даже при 200 неизвестных на процессор. Причину уменьшения ускорения на AltixXE310 можно увидеть из Таблиц 1.3,1.4. Из таблицы 1.3 видно, что количество итераций увеличивается при 32 процессорах, из-за предобуславливателя и метода разбиения сетки по процессорам. В то же время время обменов 1.4 практически не меняется

Процессоров	Время работы на сетке A, сек	Время работы на сетке B, сек
16	84.15	–
32	52.09	–
64	23.40	176.513
128	13.06	94.29
256	8.46	62.34

Таблица 1.1. Время счета на Bluegene/P.

Процессоров	Время работы на сетке A, сек	Время работы на сетке B, сек
1	76.69	631.65
2	45.65	391.10
4	22.66	190.56
8	15.05	139.02
16	7.12	69.78
32	7.45	40.95
64	6.87	24.08

Таблица 1.2. Время счета на AltixXE310.

а решение системы дорожает.

Выводы по первой главе

Мы рассмотрели структуру для хранения связей для сеток общего вида и подход с помощью которого данную структуру можно использовать для адаптивных сеток типа восьмеричное дерево. Также мы рассмотрели алгоритмы, с помощью которых можно работать с сетками на большом числе процессоров. Эф-

Процессоров	Нелинейных итераций	Линейных итераций
1	39	384
2	39	385
4	38	376
8	39	388
16	40	395
32	50	508
64	37	370

Таблица 1.3. Число итераций на AltixXE310 на сетке A.

Процессоров	Время обмена, сек	Время решения системы, сек
1	0.39	0
2	0.34	0.0074
4	0.16	0.007
8	0.14	0.006
16	0.055	0.005
32	0.079	0.0046
64	0.133	0.0049

Таблица 1.4. Время одного решения системы и одного обмена на AltixXE310 на сетке A.

фективные алгоритмы для параллельной работы с динамическими адаптивными сетками выходит за рамки данной работы.

Мы рассмотрели скорость параллельного решения задачи на статических сетках, реализованного с помощью описанных алгоритмов. Результаты показывают перспективность подхода параллелизации и демонстрируют ускорение решения задачи при умеренном числе степеней свободы на процессор.

Глава 2

Численная модель двухфазной фильтрации в пористой среде

2.1. Математическая модель

В этой главе мы рассматриваем течение двух фаз не перемешиваемых жидкостей в пористой среде [69, 70]. Фазу, которая обладает большей смачиваемостью, назовем смачивающей фазой и обозначим индексом w. Другую, не смачивающую фазу обозначим индексом o. Здесь и далее S_α и p_α будут обозначать насыщенность и давление соответствующей фазы $\alpha = w, o$. Без потери общности будем называть фазу w водой, а фазу o нефтью.

Выпишем базовые уравнения двухфазного течения:

- Сохранение масс для каждой фазы:

$$\frac{\partial}{\partial t}\frac{\phi S_\alpha}{B_\alpha} = -\mathrm{div}\mathbf{u}_\alpha + q_\alpha, \quad \alpha = w, o. \tag{2.1}$$

- Закон Дарси:

$$\mathbf{u}_\alpha = -\lambda_\alpha \mathbb{K}\left(\nabla p_\alpha - \rho_\alpha g \nabla z\right), \quad \alpha = w, o. \tag{2.2}$$

- Обе жидкости заполняют все пустоты:

$$S_w + S_o = 1. \tag{2.3}$$

- Разница давлений между фазами определяется капилярным давлением $p_c = p_c(S_w)$:

$$p_o - p_w = p_c, \tag{2.4}$$

Здесь $\mathbb{K}$ – абсолютный тензор проницаемости, ϕ – пористость, g – гравитационный член, z – глубина, для фазы α: p_α – неизвестное давление, S_α – неизвестная насыщенность, $\mathbf{u}_\alpha$ – неизвестные скорости Дарси, ρ_α – неизвестная плотность, $\rho_{alpha,0}$ – известная плотность фазы на поверхности, $B_\alpha = \rho_{\alpha,0}/\rho_\alpha$ – коэффициент объемного расширения, μ_α – вязкость, $k_{r\alpha}$ – относительная фазовая проницаемость, $\lambda_\alpha = k_{r\alpha}/(\mu_\alpha B_\alpha)$ – мобильность, q_α – источник или сток.

Возьмем за основные неизвестные давление нефти p_o и насыщенность воды S_w. В дальнейшем мы так же будем использовать следующие зависимости: $k_{r\alpha} = k_{r\alpha}(S_w)$, $\mu_\alpha = \mu_\alpha(p_o)$, $B_\alpha = B_\alpha(p_o)$ и $\phi = \phi_0\,(1 + c_R(p_o - p_o^0))$, где c_R – константа сжимаемости породы.

На границах резервуара поставим условия непротекания (однородные условия Неймана). Скважины учитываются через источники или стоки в уравнении (2.1). Предполагается, что каждая скважина является вертикальной, подключена к центру ячейки. Будем считать, что в скважине нет капиллярного давления. Таким образом потоки нефти и воды в скважине зависят только от давления нефти. В данной работе используется формула для скважин, данная Писманом в [71]. Для ячейки T с центром $\mathbf{x}_T$, подключенной к скважине, имеем:

$$q_\alpha = \lambda_\alpha WI\Big(p_{\text{bh}} - p_o - \rho_\alpha g(z_{\text{bh}} - z)\Big)\delta(\mathbf{x} - \mathbf{x}_T), \tag{2.5}$$

где p_{bh} - данное забойное давление, $\delta(\mathbf{x}-\mathbf{x}_T)$ - дельта-функция Дирака, WI индекс скважины, который не зависит от свойств жидкости,но зависит от свойств среды и размеров ячейки h_x, h_y, h_z. Для $\mathbb{K}$-ортогональной шестигранной ячейки, $\mathbb{K} = diag\{K_x, K_y, K_z\}$, имеем

$$WI = \frac{2\pi h_z\sqrt{K_xK_y}}{\log(r/r_w) + s}, \quad r = 0.28\frac{((K_y/K_x)^{1/2}h_x^2 + (K_x/K_y)^{1/2}h_y^2)^{1/2}}{(K_y/K_x)^{1/4} + (K_x/K_y)^{1/4}}, \tag{2.6}$$

где r_w - радиус скважины, s - скин фактор скважины.

2.2. Полностью неявная дискретизация

Применим полностью неявную схему дискретизации по времени для уравнений сохранения массы (2.1):

$$\frac{\left(\frac{\phi S_\alpha}{B_\alpha}\right)^{n+1} - \left(\frac{\phi S_\alpha}{B_\alpha}\right)^{n}}{\Delta t^{n+1}} = -\operatorname{div}\mathbf{u}_\alpha^{n+1} + q_\alpha^{n+1}, \qquad \alpha = w, o. \tag{2.7}$$

Теперь можно определить нелинейную невязку для l аппроксимации значения неизвестных на шаге $n+1$ в ячейке T_i:

$$R_{\alpha,i} = \int\limits_{T_i} \left[\left(\frac{\phi S_\alpha}{B_\alpha}\right)^{l,i} - \left(\frac{\phi S_\alpha}{B_\alpha}\right)^{n,i} + \Delta t^{n+1} \left(\operatorname{div}\mathbf{u}_\alpha - q_\alpha\right)^{l,i} \right] \mathrm{d}x, \quad \alpha = w, o. \tag{2.8}$$

С другой стороны потребуем дискретного выполнения следующего условия для (2.7):

$$R_{\alpha,i} = 0, \quad \alpha = w, o \tag{2.9}$$

для всех ячеек T_i на каждом шаге по времени.

Из комбинации уравнений (2.2), (2.8), (2.9) получается нелинейная система, решаемая методом Ньютона:

$$J(x^l)\delta x^l = -R(x^l), \tag{2.10}$$

$$x^{l+1} = x^l + \delta x^l, \tag{2.11}$$

здесь l есть l шаг метода Ньютона, $x = (p_o S_w)^T$ – вектор неизвестных во всех сеточных ячейках, $R(x) = (R_w(x) R_o(x))^T$ – вектор нелинейных невязок во всех сеточных ячейках, и J – матрица частных производных (Якобиан):

$$J(x) = \begin{pmatrix} \frac{\partial R_w}{\partial p_o}(x) & \frac{\partial R_w}{\partial S_w}(x) \\ \frac{\partial R_o}{\partial p_o}(x) & \frac{\partial R_o}{\partial S_w}(x) \end{pmatrix}.$$

Будем считать, что метод Ньютона сошелся, если Эвклидова норма невязки вектора $R(x)$ меньше, чем $\varepsilon_{\mathrm{nwt}}$.

2.3. Конечно-объемный метод

Пусть $\Omega \in \Re^3$ - заданная многогранная область, а $\mathcal{T}$ – конформная многогранная сетка, заданная в области Ω состоящая из $N_{\mathcal{T}}$ ячеек с плоскими гранями. Предположим, что каждая ячейка T является звездной относительно своего барицентра $\mathbf{x}_T$, а каждая грань f – звездная относительно барицентра грани $\mathbf{x}_f$.

Обозначим за $\mathbf{q}$ полный поток консервативной неизвестной c, который удовлетворяет консервативному уравнению с источником g:

$$\operatorname{div} \mathbf{q} = g \quad \text{in} \quad \Omega. \tag{2.12}$$

Выведем конечно-объемный метод для неизвестных, расположенных в барицентрах ячеек. Интегрируя уравнение div по объему многогранной ячейки T и используя формулу Грина, получим:

$$\int\limits_{\partial T} \mathbf{q} \cdot \mathbf{n}_T \, \mathrm{d}s = \int\limits_T g \, \mathrm{d}x, \tag{2.13}$$

где $\mathbf{n}_T$ – внешняя нормаль к ∂T. Пусть f обозначает грань ячейки T, и $\mathbf{n}_f$ – соответствующий вектор нормали. Для каждой ячейки T мы считаем, что нормаль $\mathbf{n}_f$ направлена наружу. В остальных случаях мы будем определять ориентацию нормали $\mathbf{n}_f$. Для удобства и без потери общности будем считать, что $|\mathbf{n}_f| = |f|$, где $|f|$ обозначает площадь грани f. Таким образом, уравнение (2.13) принимает вид

$$\sum_{f \in \partial T} \mathbf{q}_f \cdot \mathbf{n}_f = \int\limits_T g \, \mathrm{d}x, \tag{2.14}$$

где $\mathbf{q}_f$ – осредненный поток неизвестной c через грань f.

Каждой ячейке T припишем по одной степени свободы C_T консервативной неизвестной c. Пусть C – вектор всех степеней свободны. Если две ячейки T_+ и T_-

имеют общую грань f, тогда аппроксимация потока с двухточечным шаблоном, или двухточечный поток, будет иметь следующий вид:

$$\mathbf{q}_f^h \cdot \mathbf{n}_f = D^+ C_{T_+} - D^- C_{T_-}, \tag{2.15}$$

где D^+ и D^- – некоторые коэффициенты. Для линейной двухточечной аппроксимации потока эти коэффициенты зафиксированы и равны. Для нелинейной двухточечной аппроксимации они могут отличаться и зависеть от консервативных неизвестных в ближайших ячейках. На границе с краевым условием типа Неймана значение потока $\mathbf{q}_f^h \cdot \mathbf{n}_f$ задано. Подставляя (2.15) в (2.14), получим систему, состоящую из $N_\mathcal{T}$ уравнений с $N_\mathcal{T}$ неизвестными C_T. Ключевым моментом конечно-объемного метода является определение коэффициентов дискретного потока (2.15).

Будем рассматривать только случай с непрерывным полем тензора проницаемости $\mathbb{K}$. Обозначим через T_+ и T_- две ячейки, для которых грань f является общей и предположим, что $\mathbf{n}_f$ направлена наружу из T_+. Пусть $\mathbf{x}_\pm$ (или $\mathbf{x}_{T_\pm}$), точка расположения степени свободы в $T_\pm$, совпадает с барицентром $T_\pm$. Пусть $C_\pm$ (или $C_{T_\pm}$) является дискретной консервативной неизвестной расположенной в $T_\pm$.

Стандартная линейная двухточечная аппроксимация потока выписывается следующим образом:

$$\mathbf{q}_f^h \cdot \mathbf{n}_f = \frac{\mathbb{K}\mathbf{n}_f \cdot \mathbf{t}_f}{|\mathbf{t}_f|^2}(C_{T_+} - C_{T_-}), \tag{2.16}$$

где $\mathbf{t}_f = \mathbf{x}_{T_+} - \mathbf{x}_{T_-}$. В случае $\mathbb{K}$-ортогональности сетки $\mathbb{K}\,\mathbf{n}_f$ и $\mathbf{t}_f$ колинеарны, и выражение (2.16) принимает форму центральных конечных разностей и приближает поток как минимум с первым порядком. В общем случае, линейная схема может не дать аппроксимации потока вовсе.

Детальное описание нелинейного двухточечного потока для трехмерных се-

ток можно прочитать в [20, 22]. Здесь мы коротко опишем метод для внутренних граней и диффузионных потоков.

Пусть $\mathcal{F}_T$ обозначает множество граней f многогранной ячейки T. Для каждой ячейки T зададим множество Σ_T ближайших точек коллокаций степеней свободы следующим образом. Во-первых, добавим к Σ_T саму точку $\mathbf{x}_T$. Затем для каждой внутренней грани $f \in \mathcal{F}_T$, добавим точку расположения $\mathbf{x}_{T'_f}$, где T'_f – ячейка, отличная от T, которая также содержит f. Наконец, для любой граничной грани $f \in \mathcal{F}_T$, добавим точку $\mathbf{x}_f$ (Рис. 2.1 слева). Обозначим $N(\Sigma_T)$ мощность множества Σ_T.

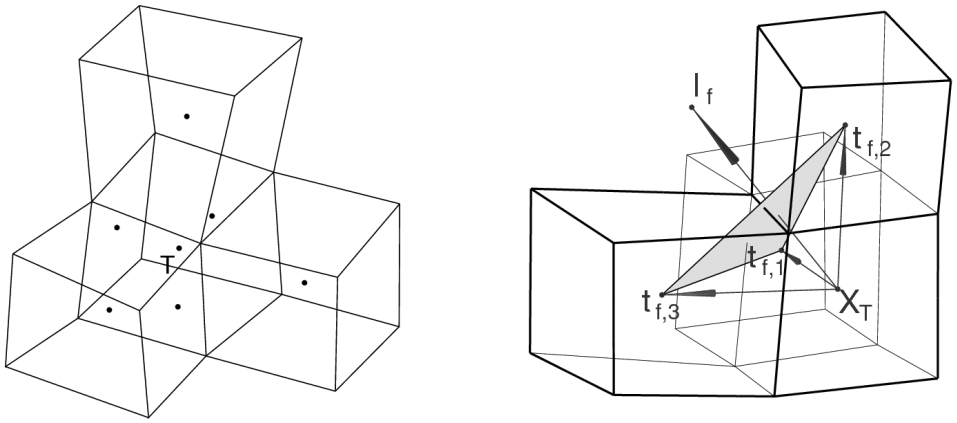

Рис. 2.1. Слева: пример множества Σ_T.Справа: Вектор конормали и триплет векторов.

Предположим, что для каждой пары ячейка-грань $T \to f$, $T \in \mathcal{T}$, $f \in \mathcal{F}_T$, существует три таких точки $\mathbf{x}_{f,1}$, $\mathbf{x}_{f,2}$, и $\mathbf{x}_{f,3}$ во множестве Σ_T, что выполняется следующее условие (см. Рис. 2.1 справа): вектор конормали $\boldsymbol{\ell}_f = \mathbb{K}(\mathbf{x}_f)\mathbf{n}_f$, начинающийся из $\mathbf{x}_T$, принадлежит трехгранному углу, образованному векторами

$$\mathbf{t}_{f,1} = \mathbf{x}_{f,1} - \mathbf{x}_T, \quad \mathbf{t}_{f,2} = \mathbf{x}_{f,2} - \mathbf{x}_T, \quad \mathbf{t}_{f,3} = \mathbf{x}_{f,3} - \mathbf{x}_T, \tag{2.17}$$

и

$$\frac{1}{|\boldsymbol{\ell}_f|}\boldsymbol{\ell}_f = \frac{\alpha_f}{|\mathbf{t}_{f,1}|}\mathbf{t}_{f,1} + \frac{\beta_f}{|\mathbf{t}_{f,2}|}\mathbf{t}_{f,2} + \frac{\gamma_f}{|\mathbf{t}_{f,3}|}\mathbf{t}_{f,3}, \tag{2.18}$$

где $\alpha_f \geq 0$, $\beta_f \geq 0$, $\gamma_f \geq 0$. В [20] дан простой и эффективный алгоритм поиска триплета, удовлетворяющего (2.18) с неотрицательными коэффициентами.

Коэффициенты α_f, β_f, γ_f вычисляются следующим образом:

$$\alpha_f = \frac{A_{f,1}}{A_f}, \qquad \beta_f = \frac{A_{f,2}}{A_f}, \qquad \gamma_f = \frac{A_{f,3}}{A_f}, \tag{2.19}$$

где

$$A_f = \frac{\left|\mathbf{t}_{f,1} \quad \mathbf{t}_{f,2} \quad \mathbf{t}_{f,3}\right|}{|\mathbf{t}_{f,1}||\mathbf{t}_{f,2}||\mathbf{t}_{f,3}|}, \; A_{f,1} = \frac{\left|\boldsymbol{\ell}_f \quad \mathbf{t}_{f,2} \quad \mathbf{t}_{f,3}\right|}{|\boldsymbol{\ell}_f||\mathbf{t}_{f,2}||\mathbf{t}_{f,3}|},$$
$$A_{f,2} = \frac{\left|\mathbf{t}_{f,1} \quad \boldsymbol{\ell}_f \quad \mathbf{t}_{f,3}\right|}{|\mathbf{t}_{f,1}||\boldsymbol{\ell}_f||\mathbf{t}_{f,3}|}, \; A_{f,3} = \frac{\left|\mathbf{t}_{f,1} \quad \mathbf{t}_{f,2} \quad \boldsymbol{\ell}_f\right|}{|\mathbf{t}_{f,1}||\mathbf{t}_{f,2}||\boldsymbol{\ell}_f|},$$

и $|\mathbf{a}\ \mathbf{b}\ \mathbf{c}| = |(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}|$.

Для диффузионного потока $\mathbb{K}_f \nabla c \cdot \mathbf{n}_f$ неотрицательные коэффициенты в (2.15) получаются следующим образом

$$D^{\pm} = \mu_{\pm}|\boldsymbol{\ell}_f|(\alpha_{\pm}/|\mathbf{t}_{\pm,1}| + \beta_{\pm}/|\mathbf{t}_{\pm,2}| + \gamma_{\pm}/|\mathbf{t}_{\pm,3}|). \tag{2.20}$$

Коэффициенты $\mu_{\pm}$ зависят от ближайших консервативных неизвестных:

$$\mu_+ = \frac{d_-}{d_- + d_+} \qquad \text{и} \qquad \mu_- = \frac{d_+}{d_- + d_+}$$

где

$$d_{\pm} = |\boldsymbol{\ell}_f| \left(\frac{\alpha_{\pm}}{|\mathbf{t}_{\pm,1}|} C_{\pm,1} + \frac{\beta_{\pm}}{|\mathbf{t}_{\pm,2}|} C_{\pm,2} + \frac{\gamma_{\pm}}{|\mathbf{t}_{\pm,3}|} C_{\pm,3} \right). \tag{2.21}$$

Если $d_{\pm} = 0$, положим $\mu_+ = \mu_- = \frac{1}{2}$.

Важное и удобное свойство нелинейной двухточечной аппроксимации состоит в том, что она сводится к стандартной линейной двухточеной аппроксимации на $\mathbb{K}$-ортогональных сетках.

Далее нам потребуется вариация коэффициентов $D^{\pm}$ в (2.20) при вычислении Якобиана. Сначала выпишем вариации для $d_{\pm}$ и $\mu_{\pm}$:

$$\Delta d_{\pm} = |\boldsymbol{\ell}_f| \left(\frac{\alpha_{\pm}}{|\mathbf{t}_{\pm,1}|} \Delta C_{\pm,1} + \frac{\beta_{\pm}}{|\mathbf{t}_{\pm,2}|} \Delta C_{\pm,2} + \frac{\gamma_{\pm}}{|\mathbf{t}_{\pm,3}|} \Delta C_{\pm,3} \right), \tag{2.22}$$

$$\Delta\mu_{\pm} = \frac{\Delta d_{\mp}}{d_{\mp} + d_{\pm}} - (\Delta d_{\mp} + \Delta d_{\pm})\frac{d_{\mp}}{(d_{\mp} + d_{\pm})^2}. \tag{2.23}$$

Затем, для вариации $D^{\pm}$ выпишем линейную комбинацию:

$$\Delta D^{\pm} = \Delta\mu_{\pm}\big(\alpha_{\pm}/|\mathbf{t}_{\pm,1}| + \beta_{\pm}/|\mathbf{t}_{\pm,2}| + \gamma_{\pm}/|\mathbf{t}_{\pm,3}|\big) = \sum_{T_i \in \Sigma_{T_*}} L_i^{\pm}\Delta C_i, \tag{2.24}$$

где $\Sigma_{T_*} := \Sigma_{T_+} \cup \Sigma_{T_-}$ и $L_i^{\pm} = L_i^{\pm}(C)$ – коэффициенты линейной комбинации, получаемые подстановкой (2.22) и (2.23) в (2.20).

2.4. Вычисление Якобиана

Вычислим матрицу частных производных – Якобиан – для уравнения (2.10) следующим образом. Разделим нелинейную невязку на две части: аккумуляцию и перенос, $R_{\alpha,i} = R_{\alpha,i}^{acc} + R_{\alpha,i}^{trans}$, где:

$$R_{\alpha,i}^{acc} = V_i\left[\left(\frac{\phi S_\alpha}{B_\alpha}\right)^{l,i} - \left(\frac{\phi S_\alpha}{B_\alpha}\right)^{n,i}\right] - \Delta t^{n+1} Q_\alpha^{l,i},$$

$$Q_\alpha^{l,i} = \int\limits_{T_i} q_\alpha^{l,i} dx,$$

$$R_{\alpha,i}^{trans} = \Delta t^{n+1}\int\limits_{T_i} \operatorname{div}\mathbf{u}_\alpha^l \,\mathrm{d}x, \qquad \alpha = w, o.$$

Не теряя общности, опустим индексы l и i. Сначала рассмотрим вариацию для аккумуляции:

$$\Delta R_{w,i}^{acc} = V_i\,\Delta\left(\frac{\phi S_w}{B_w}\right) - \Delta t^{n+1}\Delta Q_w,$$

$$\Delta R_{o,i}^{acc} = V_i\,\Delta\left(\frac{\phi S_o}{B_o}\right) - \Delta t^{n+1}\Delta Q_o,$$

где

$$\Delta\left(\frac{\phi S_w}{B_w}\right) = \frac{\phi}{B_w}\Delta S_w + S_w\left(\frac{\phi_0 c_R}{B_w} - \frac{\phi}{B_w^2}\frac{dB_w}{dp_o}\right)\Delta p_o,$$

$$\Delta\left(\frac{\phi S_o}{B_o}\right) = -\frac{\phi}{B_o}\Delta S_w + (1-S_w)\left(\frac{\phi_0 c_R}{B_o} - \frac{\phi}{B_o^2}\frac{dB_o}{dp_o}\right)\Delta p_o.$$

V_i - объем ячейки T_i.

Для скважин введем дополнительные переменные и производные:

$$\mathcal{D}_\alpha = p_{bh} - p_o - \frac{\rho_{\alpha,0}}{B_\alpha} g(z_{bh} - z),$$

$$\frac{d\mathcal{D}_\alpha}{dp_o} = -1 + \frac{\rho_{\alpha,0}}{B_\alpha^2}\frac{dB_\alpha}{dp_o} g(z_{bh} - z), \qquad \frac{d\lambda_\alpha}{dS_w} = \frac{dk_{r\alpha}}{dS_w}\frac{1}{B_\alpha \mu_\alpha},$$

$$\frac{d\lambda_\alpha}{dp_o} = -k_{r\alpha}\left(B_\alpha \frac{d\mu_\alpha}{dp_o} + \mu_\alpha \frac{dB_\alpha}{dp_o}\right)/(B_\alpha\mu_\alpha)^2, \quad \alpha = w, o.$$

Вариация для скважин выписывается следующим образом: для производящих скважин

$$\Delta Q_\alpha = WI\left[\mathcal{D}_\alpha \frac{d\lambda_\alpha}{dS_w}\Delta S_w + \left(\lambda_\alpha \frac{d\mathcal{D}_\alpha}{dp_o} + \frac{d\lambda_\alpha}{dp_o}\mathcal{D}_\alpha\right)\Delta p_o\right]$$

для нагнетающих скважин

$$\begin{aligned}\Delta Q_w = WI\Bigg[&\mathcal{D}_w\left(\frac{d\lambda_w}{dS_w} + \frac{d\lambda_o}{dS_w}\right)\Delta S_w + \\ &+ \left((\lambda_w + \lambda_o)\frac{d\mathcal{D}_w}{dp_o} + \mathcal{D}_w\left(\frac{d\lambda_w}{dp_o} + \frac{d\lambda_o}{dp_o}\right)\right)\Delta p_o\Bigg], \\ \Delta Q_o = 0.&\end{aligned}$$

Теперь, рассмотрим перенос, состоящий из потоков Дарси

$$R_{\alpha,i}^{trans} = \Delta t^{n+1}\int\limits_{\partial T_i}(\mathbf{u}_\alpha \cdot \mathbf{n})\,\mathrm{d}s \approx \Delta t^{n+1}\sum_{f\in\partial T_i}\mathbf{u}_{\alpha,f}^h \cdot \mathbf{n}_f. \tag{2.25}$$

Мы применяем двухточечную аппроксимацию потока для каждого поля: p_o, p_c, z и обозначаем соответствующие коэффициенты потоков через $D_{p_o}^\pm$, $D_{p_c}^\pm$, $D_z^\pm$, а значения поля в барицентре ячейки $\mathbf{x}_{T_\pm}$ через $p_o^\pm$, $p_c^\pm$, $S_w^\pm$, $z^\pm$. Таким образом,

$$\mathbf{u}^h_{w,f}\cdot\mathbf{n}_f = -\left(\frac{k_{rw}}{\mu_w B_w}\right)_f\Big(D^+_{p_o}p^+_o - D^-_{p_o}p^-_o\Big)+ \\ +\left(\frac{k_{rw}}{\mu_w B^2_w}\right)_f\Big[\rho_{w,0}\,g\Big(D^+_z z^+ - D^-_z z^-\Big)\Big]+ \\ +\left(\frac{k_{rw}}{\mu_w B_w}\right)_f\Big(D^+_{p_c}p^+_c - D^-_{p_c}p^-_c\Big), \tag{2.26}$$

$$\mathbf{u}^h_{o,f}\cdot\mathbf{n}_f = -\left(\frac{k_{ro}}{\mu_o B_o}\right)_f\Big(D^+_{p_o}p^+_o - D^-_{p_o}p^-_o\Big)+ \\ +\left(\frac{k_{ro}}{\mu_o B^2_o}\right)_f\Big[\rho_{o,0}\,g\Big(D^+_z z^+ - D^-_z z^-\Big)\Big]. \tag{2.27}$$

Здесь в выражении $k_{r\alpha} = k_{r\alpha}(\widetilde{S}_w)$, член $\widetilde{S}_w$ обозначает значение насыщенности воды, взятое против потока на грани f, а в выражениях $B_\alpha = B_\alpha(\widetilde{p}_o)$, $\mu_\alpha = \mu_\alpha(\widetilde{p}_o)$, член $\widetilde{p}_o$ обозначает давление нефти, взятое против потока на грани f.

Введем дополнительные переменные и производные:

$$\lambda_{g,\alpha} = \frac{k_{r\alpha}}{\mu_w B^2_w}, \qquad \frac{d\lambda_{g,\alpha}}{d\widetilde{S}_w} = \frac{d\lambda_\alpha}{d\widetilde{S}_w}/B_w,$$

$$\frac{d\lambda_{g,\alpha}}{d\widetilde{p}_o} = \left(\frac{d\lambda_\alpha}{d\widetilde{p}_o}B_w - \lambda_\alpha\frac{dB_w}{d\widetilde{p}_o}\right)/B^2_w, \quad \alpha = w, o,$$

$$\mathcal{D}_1 = D^+_{p_o}p^+_o - D^-_{p_o}p^-_o,$$

$$\mathcal{D}_2 = D^+_{p_c}p^+_c - D^-_{p_c}p^-_c,$$

$$\mathcal{D}_{3,\alpha} = \rho_{\alpha,0}\,g\Big(D^+_z z^+ - D^-_z z^-\Big).$$

Используя (2.26) и (2.27), получим следующее выражение для вариации потока для каждой из двух фаз:

$$\begin{aligned}\Delta(\mathbf{u}^h_{w,f}\cdot\mathbf{n}_f) = &\left[\Big(\frac{d\lambda_w}{d\widetilde{S}_w}\Big)(-\mathcal{D}_1+\mathcal{D}_2)+\frac{d\lambda_{g,w}}{d\widetilde{S}_w}\mathcal{D}_{3,w}\right]\Delta\widetilde{S}_w+\\ &+\left[\Big(\frac{d\lambda_w}{d\widetilde{p}_o}\Big)(-\mathcal{D}_1+\mathcal{D}_2)+\frac{d\lambda_{g,w}}{d\widetilde{p}_o}\mathcal{D}_{3,w}\right]\Delta\widetilde{p}_o-\\ &-\lambda_w\Big(D^+_{p_o}\Delta p^+_o-D^-_{p_o}\Delta p^-_o\Big)+\\ &+\lambda_w\Big(D^+_{p_c}\Big(\frac{dp_c}{dS_w}\Big)^+\Delta S^+_w-D^-_{p_c}\Big(\frac{dp_c}{dS_w}\Big)^-\Delta S^-_w\Big)-\\ &-\lambda_w\Big(\Delta D^+_{p_o}p^+_o-\Delta D^-_{p_o}p^-_o\Big)+\ \lambda_w\Big(\Delta D^+_{p_c}p^+_c-\Delta D^-_{p_c}p^-_c\Big),\end{aligned}\tag{2.28}$$

$$\begin{aligned}\Delta(\mathbf{u}^h_{o,f}\cdot\mathbf{n}_f) = &\left[(\frac{d\lambda_o}{d\widetilde{S}_w})(-\mathcal{D}_1+\mathcal{D}_2)+\frac{d\lambda_{g,o}}{d\widetilde{S}_w}\mathcal{D}_{3,o}\right]\Delta\widetilde{S}_w+\\ &+\left[(\frac{d\lambda_o}{d\widetilde{p}_o})(-\mathcal{D}_1+\mathcal{D}_2)+\frac{d\lambda_{g,o}}{d\widetilde{p}_o}\mathcal{D}_{3,o}\right]\Delta\widetilde{p}_o\ -\\ &-\lambda_o\Big(D^+_{p_o}\Delta p^+_o-D^-_{p_o}\Delta p^-_o\Big)+\ \lambda_o\Big(\Delta D^+_{p_o}p^+_o-\Delta D^-_{p_o}p^-_o\Big).\end{aligned}\tag{2.29}$$

Можно подойти с нескольких сторон к вычислению приближения к вариации переноса (2.28) и (2.29): коэффициенты $D^\pm_{p_o}, D^\pm_{p_c}, D^\pm_z$ можно предположить зафиксированными на каждом шаге Ньютона, или можно продифференцировать их по давлению и насыщенности в соседних ячейках. В первом случае коэффициенты будут зафиксированы $\Delta D^\pm_{p_o} = \Delta D^\pm_{p_c} = \Delta D^\pm_z = 0$ и разница между линейной и нелинейной двухточечной аппроксимации потока заключается только в том, как мы вычисляем коэффициенты $D^\pm_{p_o}, D^\pm_{p_c}, D^\pm_z$, при этом шаблон матрицы – Якобиана будет тот же. Вычислительная цена каждого умножения Якобиана на вектор будет одинаковой для линейного и нелинейного метода. Если коэффициенты не

фиксировать, то

$$\Delta D^{\pm}_{p_o} = \sum_{T_j \in \Sigma_{T_*}} L^{\pm}_{p,j}\, \Delta p^j_o, \tag{2.30}$$

$$\Delta D^{\pm}_{p_c} = \sum_{T_j \in \Sigma_{T_*}} L^{\pm}_{p_c,j} \left(\frac{dp_c}{dS_w}\right)^j \Delta S^j_w, \tag{2.31}$$

$$\Delta D^{\pm}_{z} = 0, \tag{2.32}$$

где $L^{\pm}_{p_o,j}$ и $L^{\pm}_{p_c,j}$ это коэффициенты, вычисляемые по формуле (2.24) для переменных p_o и $p_c(S_w)$, соответственно. В результате получим менее разреженный Якобиан и более дорогую операцию умножения вектора на Якобиан. Вычисление предобуславливателя так же становится более дорогим. Для уменьшения вычислительной сложности можно ввести барьер, по которому отфильтровываются малые значения в Якобиане и в результате получается более разреженная матрица.

2.5. Сравнение линейной и нелинейной двухточечной аппроксимации потока

Рассмотрим простой пример, с одной нагнетательной и одной производящей скважинами. Нагнетаемая вода вытесняет нефть к производящей скважине и заполняет собой среду. На примере мы хотим продимонстрировать влияние дискретизации на поведение фронта воды. А именно, мы сравниваем кривые зависимости производительности воды и нефти от времени и момент прорыва воды в производящей скважине.

Возьмем резервуар со следующими размерами в футах $\Omega = [-50, 50] \times [-50, 50] \times [4010, 4020]$. Будем рассматривать последовательность сгущающихся сеток. Скважины находятся внутри области. Сетки будем сгущать таким образом, чтобы центр каждой скважины всегда был расположен в той же позиции и

совпадал с центром одной из ячеек.

Нагнетательная скважина расположена в позиции (-40,-40) футов, производящая - в (40,40) футов. Обе скважины имеют по одному подключению к резервуару. В качестве условия на скважине задается забойное давление. Для нагнетательной скважины зададим $p_{bh,inj} = 4100$ psia, а для производящей $p_{bh,pr} = 3900$ psia. Здесь psia обозначает абсолютное значения давления в фунтах-силы на квадратный дюйм. Индекс скважины рассчитывается согласно (2.6) с радиусом скважины $r_w = 5 \cdot 10^{-4}$ фут и скин-фактором $s = 0$.

В численных экспериментах мы используем следующие свойства породы и жидкостей: вязкости μ_α и фактор объемного расширения B_α заданы в Таблице 2.1, а плотности вычисляются через $\rho_\alpha = \rho_{\alpha,0}/B_\alpha$, где $\rho_{w,0} \approx 4.331 \cdot 10^{-1} psi/$фут и $\rho_{o,0} \approx 3.898 \cdot 10^{-1} psi/$фут. Коэффициент сжимаемости породы c_R равен $10^{-6} psi^{-1}$. Зависимости капиллярного давления p_c от насыщенности воды S_w и относительные проницаемости $k_{r\alpha}$ представлены на Рис. 2.2.

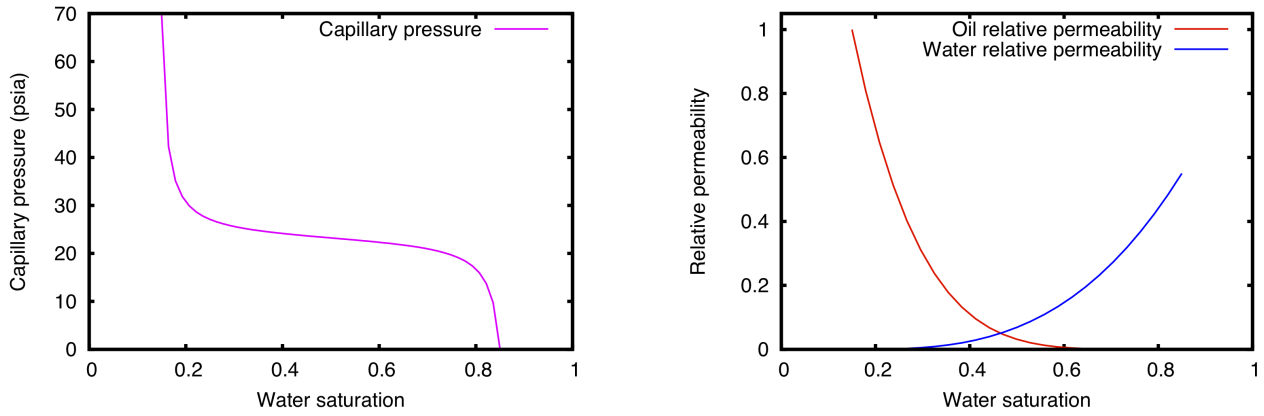

Рис. 2.2. Зависимость капиллярного давления от S_w (слева) и относительные проницаемости для воды и нефти (справа).

В этом разделе мы продемонстрируем преимущество нелинейного двухточеч-

p (psia)	B_o (bbl/STB)	B_w (bbl/STB)	μ_o (cp)	μ_w (cp)
3900	1.0030285	1.0131740	90.582	0.5151
4000	1.0019665	1.0129084	96.015	0.5179
4100	1.0009032	1.0126377	101.719	0.5207

Таблица 2.1. Свойства сжимаемости жидкости.

ного потока. Для простоты мы опускаем влияние гравитации и решаем псевдодвумерную проблему, используя $N \times N \times 1$ сетки с одним слоем ячеек. Проведем анализ сходимости решения на последовательности прямоугольных сеток и последовательности возмущенных сеток. Возмущенные сетки мы получим из прямоугольных сеток с шагом сетки h следующим образом. Каждый внутренней узел с координатами (x, y), не смежный с ячейкой, в которой находится скважина, сдвинем в положение $(\tilde{x}, \tilde{y})$:

$$\tilde{x} := x + \gamma\,\xi_x h, \qquad \tilde{y} := y + \gamma\,\xi_y h, \tag{2.33}$$

где ξ_x и ξ_y – случайные переменные со значениями от -0.5 до 0.5 и $\gamma \in [0, 1]$ – степень возмущения. Выберем $\gamma = 0.6$, чтобы избежать запутывания сетки. Следует подчеркнуть, что возмущение производится на каждом уровне сгущения. Примеры ортогональных и возмущенных сеток приведены на Рис. 2.3.

В первом эксперименте мы сравниваем результаты моделирования с линейной и нелинейной двухточечной аппроксимацией потока на последовательности сеток, состоящих из $15 \times 15 \times 1$, $45 \times 45 \times 1$, $135 \times 135 \times 1$ и $405 \times 405 \times 1$ ячеек. Максимальный шаг по времени $dt_{max} = 27$, 9, 3 и 1 дней, соответственно. В каждом тесте мы стартуем с шага $dt = 0.005$ дней, а затем вычисляем dt на каждом шаге по времени по следующей формуле

$$\alpha = \sqrt{\frac{dt_{max} - dt}{dt_{max}}}, \quad dt := \alpha dt + (1 - \alpha) dt_{max}.$$

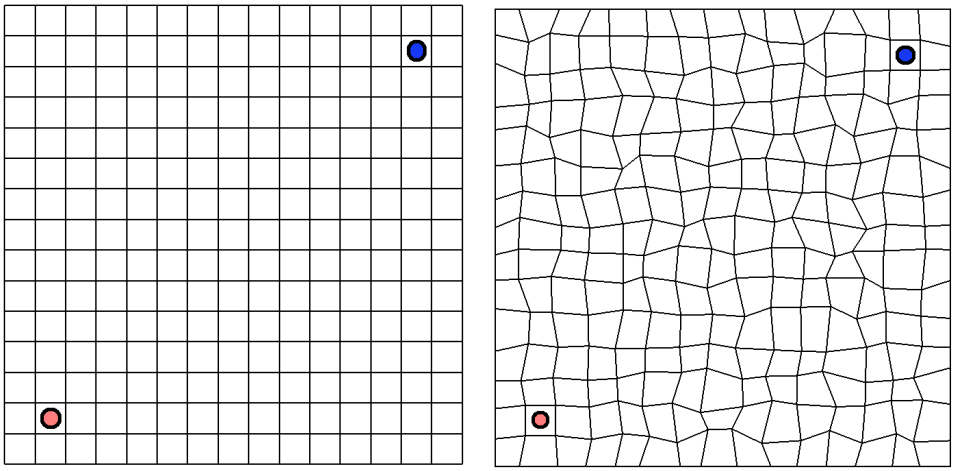

Рис. 2.3. Примеры ортогональной (слева) и возмущенной (справа) сеток. Красный круг обозначает нагнетательную скважину, синий - производящую.

Модельное временя равнялось 250 дням.

Зададим абсолютный тензор проницаемости $\mathbb{K} = diag(1000, 100, 50)$. Так как ортогональная сетка является $\mathbb{K}$-ортогональной, то линейная и нелинейная двухточечная аппроксимация потока совпадают. Поэтому здесь и далее мы будем использовать решение, полученное с помощью линейной двухточечной аппроксимации потока на самой мелкой прямоугольной сетке, как эталонное.

Рис. 2.5 показывает идентичное поведение кривых добычи нефти в зависимости от времени, полученное обоими методами на ортогональных сетках и нелинейным методом на возмущенных сетках. С другой стороны, конечно-объемная схема с линейной аппроксимации потока *расходится* на последовательности возмущенных сеток. Это можно увидеть на Рис. 2.4.

Таким образом, конечно-объемная схема с линейной аппроксимацией потока дает неверный результат из-за потери свойства аппроксимации, в то время как нелинейная двухточечная аппроксимация демонстрирует сходимость показателя производительности нефти в зависимости от времени.

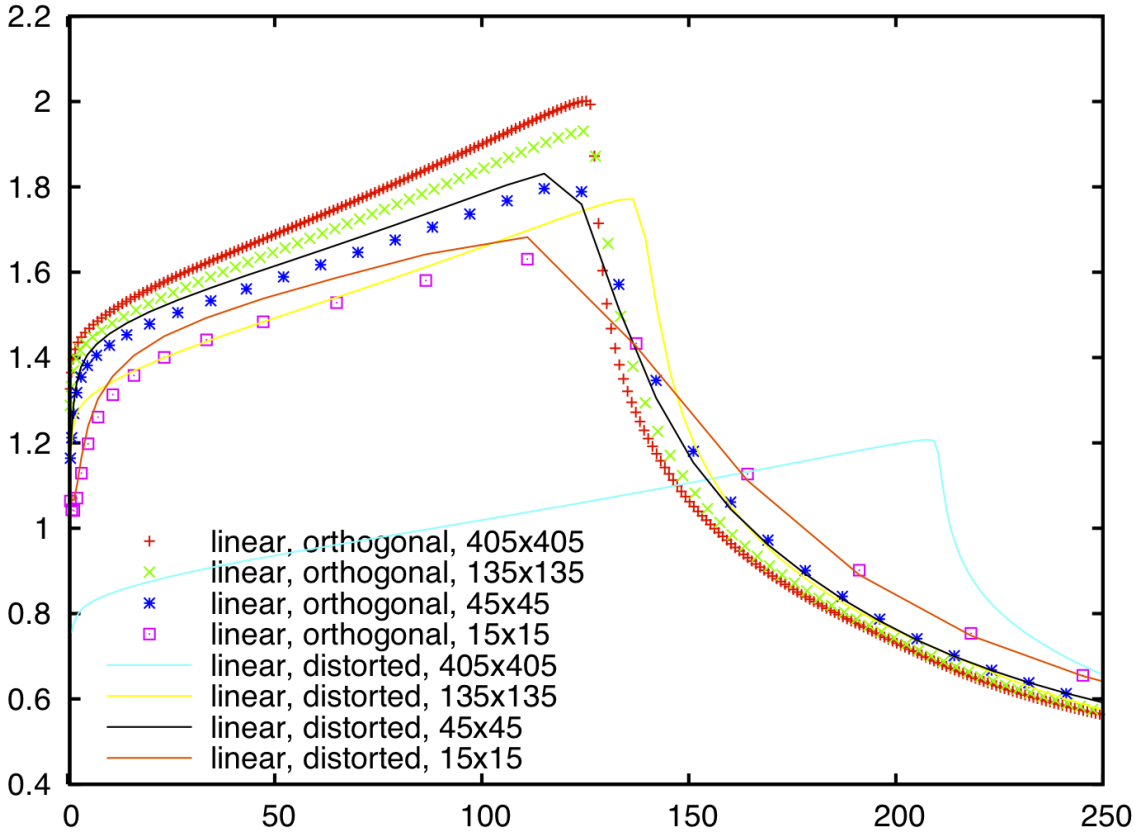

Рис. 2.4. Зависимость производительности нефти от времени. Линейная аппроксимация потока на ортогональных сетках (точки) и на возмущенных сетках (линии).

2.6. Решения задачи на динамических сетках типа восьмеричное дерево

Динамические сетки типа восьмеричное дерево строятся следующим образом. Рассмотрим грубую ортогональную сетку $M \times N \times K$, которая определяет самый грубый уровень восьмеричного дерева. Затем каждая ячейка этой сетки сама превращается в восьмеричное дерево. Конечная сетка состоит из множества объединенных восьмеричных деревьев. Далее возьмем грубую сетку разме-

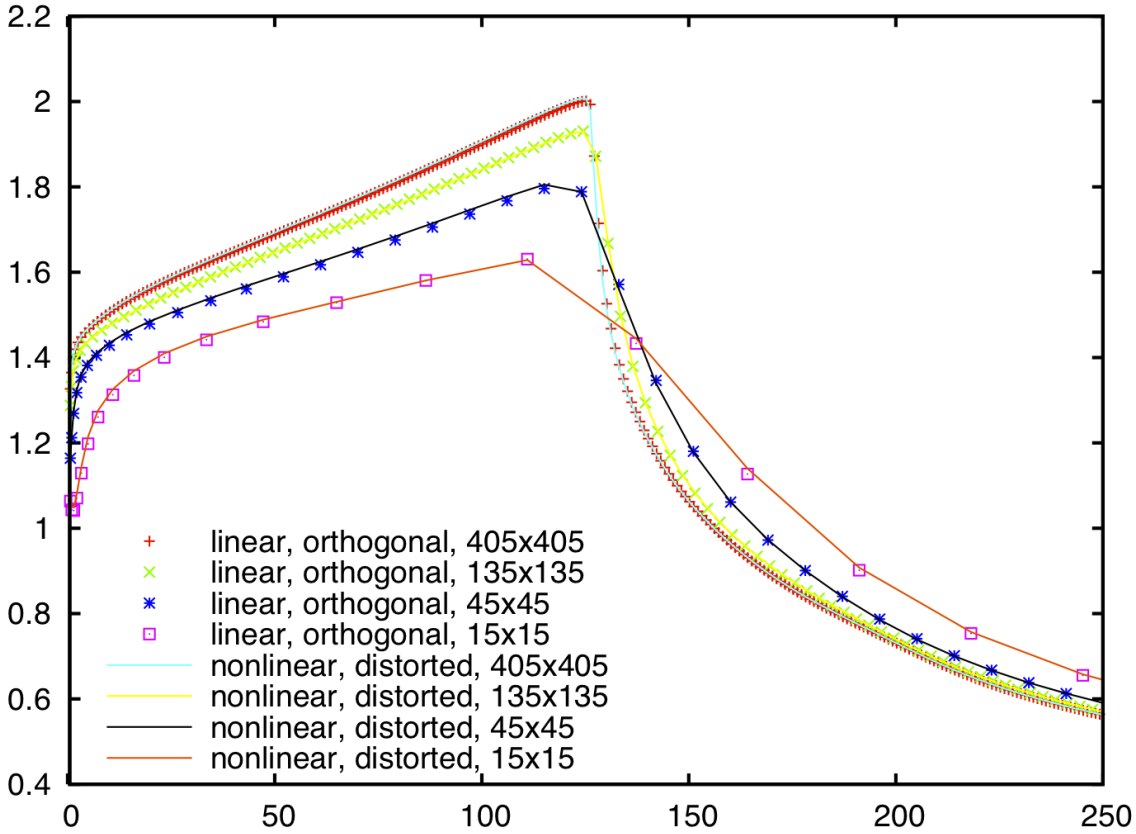

Рис. 2.5. Зависимость производительности нефти от времени. Линейная двухточечная аппроксимация потока на прямоугольных сетках (точки) и нелинейная на возмущенных (линии)

ров $M = N = 5$, $K = 1$ и предположим, что размер любых двух соседних ячеек в локально измельченной сетке не может отличаться более чем в двое. При дискретизации мы будем считать сетку типа восьмеричное дерево как конформную многогранную сетку, где каждая ячейка может состоять из 24 соседей, а каждая грань граничит не более, чем с двумя ячейками. В данном тесте мы не будем измельчать ячейки в направлении оси Oz. На Рис. 2.6 продемонстрирована сетка типа восьмеричное дерево и поле насыщенности воды в момент времени $t = 100$ дней, для численного примера, представленного ниже.

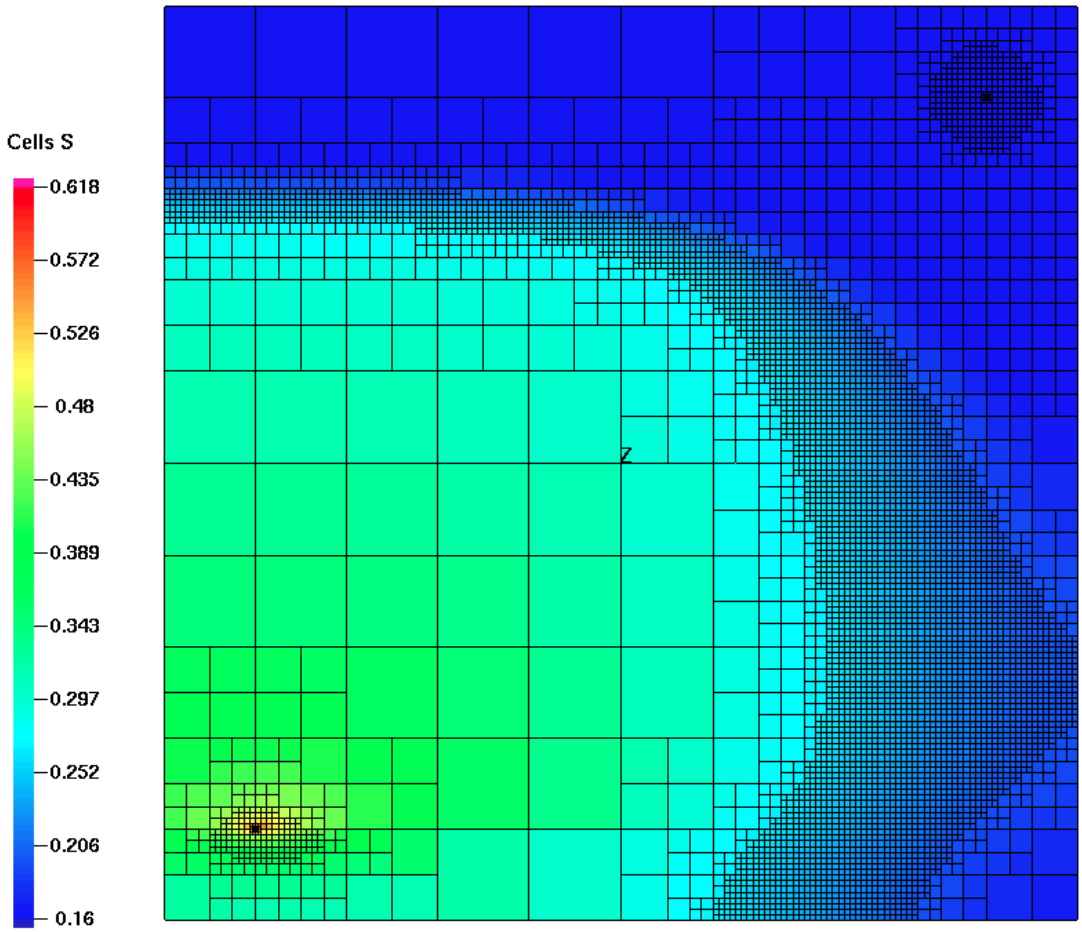

Рис. 2.6. Пример сетки типа восьмеричное дерево, цветом изображено поле насыщенности воды, в момент $t = 100$ дней.

Далее дадим правила сгущения и разгрубления ячеек сетки. Интерполяция функции с одной сетки типа восьмеричное дерево на другую основана на предположении, что одноуровневое локальное сгущение и разгрубление может только удвоить или поделить пополам размер ячейки, и размеры соседних ячеек не могут отличаться более, чем в два раза. Для интерполяции будем использовать модификацию консервативного метода наименьших квадратов (WLSQR) [16]. Предположим, что в ячейки T_0 и в соседних ячейках T_i кусочно-постоянная функция u определена по значениям u_0, u_i. Предположим так же, что ячейка T_0 будет разбита на 8 ячеек и нам необходимо проинтерполировать u в эти ячейки. Мы

находим линейную функцию $P(x, y, z) = ax + by + cz + d$, которая удовлетворяет $\int_{T_0} P(x, y, z)dT = |T_0|u_0$ (консервативности) и $\int_{T_i} P(x, y, z)dT \approx |T_i|u_i$ (аппроксимации). Из уравнения консервативности мы можем зафиксировать d, а требование аппроксимации удовлетворяется за счет определения коэффициентов a, b, c методом наименьших квадратов. Затем мы можем определить u как значение $P(x, y, z)$ в барицентрах восьми новых ячеек.

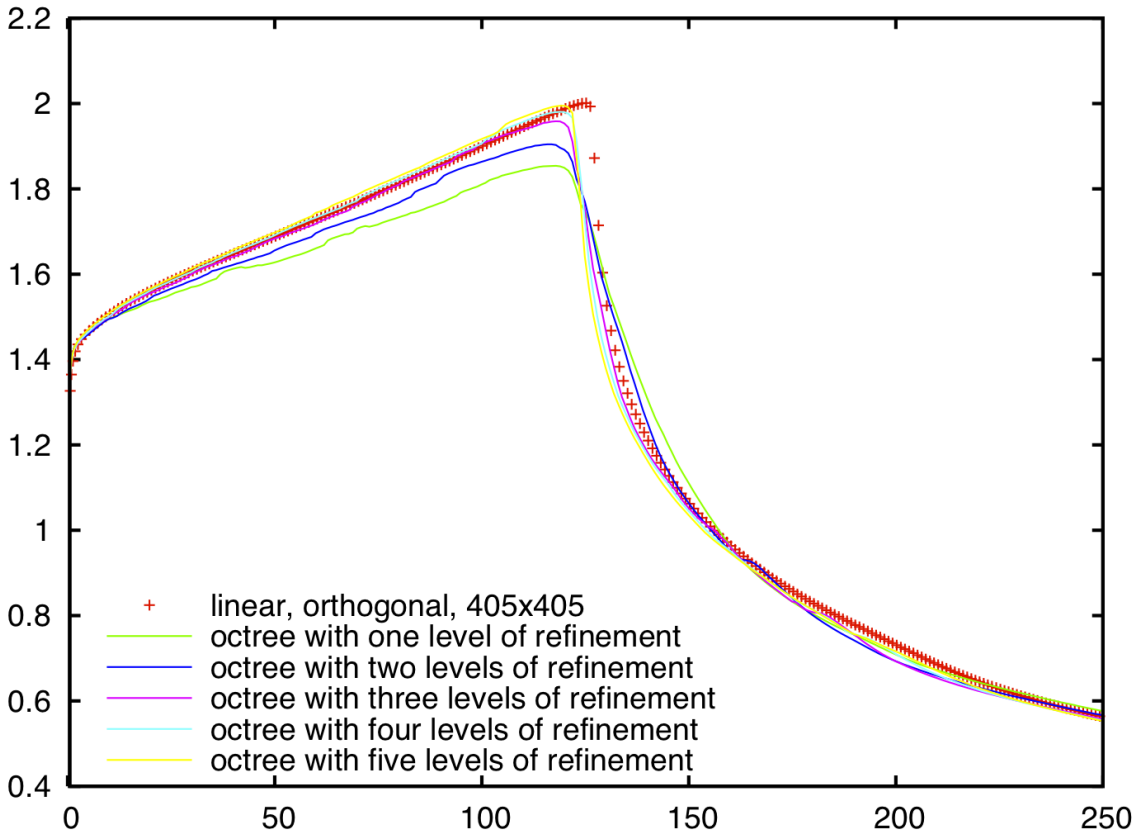

Рис. 2.7. Производительность нефти в зависимости от времени на сетках типа восьмеричное дерево с разным уровнем сгущения.

Вернемся к задаче из § 2.5 с диагональным тензором $\mathbb{K} = diag(1000, 100, 50)$. Так как ортогональная сетка является $\mathbb{K}$-ортогональной, и нелинейная аппрок-

симация потоков совпадает с линейной на $\mathbb{K}$-ортогональной сетке, то мы можем использовать решение задачи на самой мелкой сетке $405 \times 405 \times 1$ с шагом по времени 1 день как эталонное. Далее в тестах мы будем использовать шаг в 1 день для всех рассматриваемых сеток. В следствии анизотропии тензора проницаемости $\mathbb{K}$ фронт воды движется быстрее вдоль оси Ox чем вдоль оси Oy. Адаптивная сетка должна отслеживать фронт воды по величине градиента насыщенности. Чтобы уменьшить численную вязкость, мы также применяем локальное сгущение в области с высоким градиентом давления. А именно, если $|\nabla S_w| > tol_{S_w}$, то сетка сгущается до самого мелкого уровня l, а если $|\nabla p_o| > tol_{p_o}$, то сетка сгущается к уровню $l-1$, где $tol_{S_w} = 0.25$, $tol_{p_o} = 0.0005$. Дополнительно, потребуем, чтобы в точке, содержащей скважину, сетка имела уровень l.

На рис. 2.7 изображены кривые производительности нефти в зависимости от времени на самой мелкой регулярной сетке и на сетках типа восьмеричное дерево с разным уровнем сгущения l. Очевидно, что решения на сетках типа восьмеричное дерево сходятся к эталонному.

Далее, сравним эффективность использования динамических сеток типа восьмеричное дерево. Итерации метода Ньютона прерываются, когда норма нелинейной невязки становится меньше 10^{-9}, а итерации метода BiCGStab и с предобуславливателем ILU(1) прерываются, когда норма линейной невязки становится меньше 10^{-12}. Существенная разница в числе неизвестных (Рис. 2.8) между самой мелкой эталонной сеткой и адаптивными сетками типа восьмеричное дерево приводит к гораздо меньшему числу линейных итераций на каждом шаге, см. Рис. 2.9.

Уменьшение числа неизвестных на сетках типа восьмеричное дерево приводит к ускорению решения задачи, см. Таблицу 2.2. По результатам теста можно видеть, что дополнительные вычисления, возникающие на динамических адаптив-

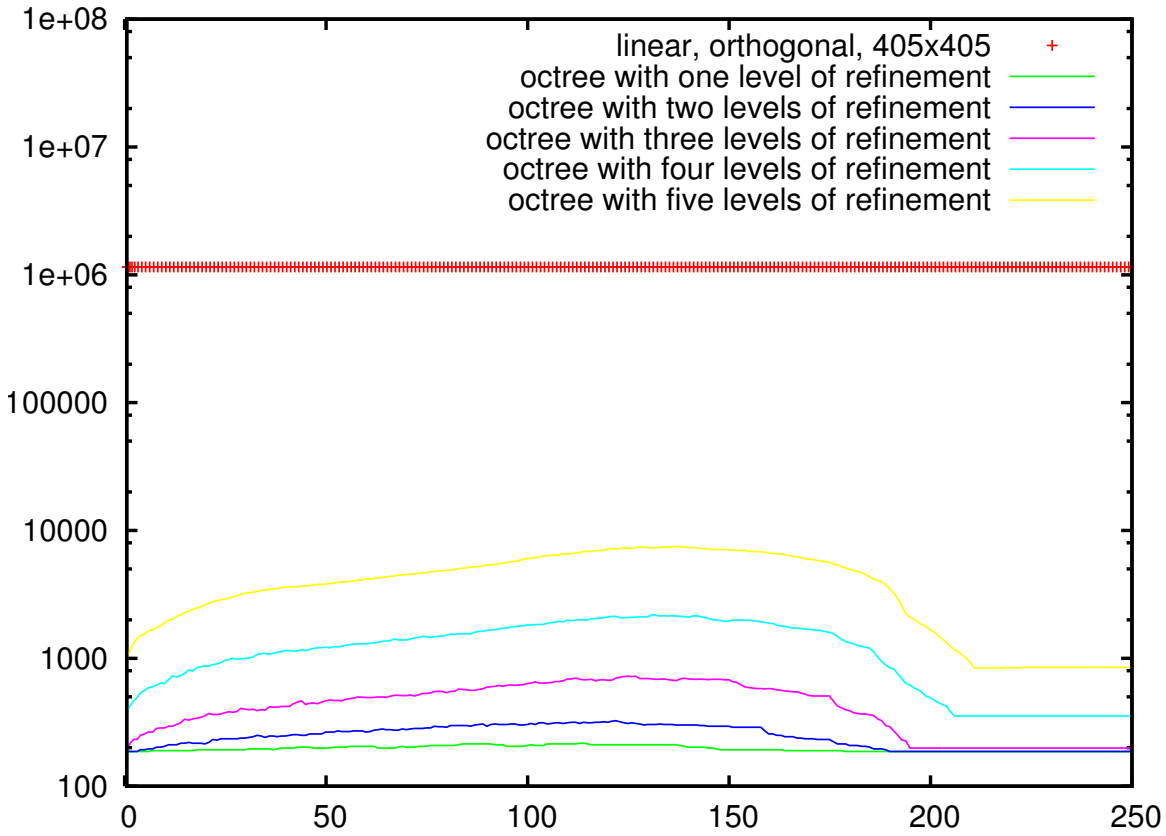

Рис. 2.8. Количество ячеек на каждом шаге в сетках типа восьмеричное дерево с разным уровнем сгущения.

ных сетках: необходимость интерполировать данные при сгущениях и разгрублениях сетки, возросшая плотность Якобиана и необходимость пересчитывать триплеты для нелинейного метода в большинстве ячеек, - не мешают получить значительное ускорение. Численные эксперименты были поставлены на процессоре Intel Xeon X5650 с частотой 2.67 Ггц.

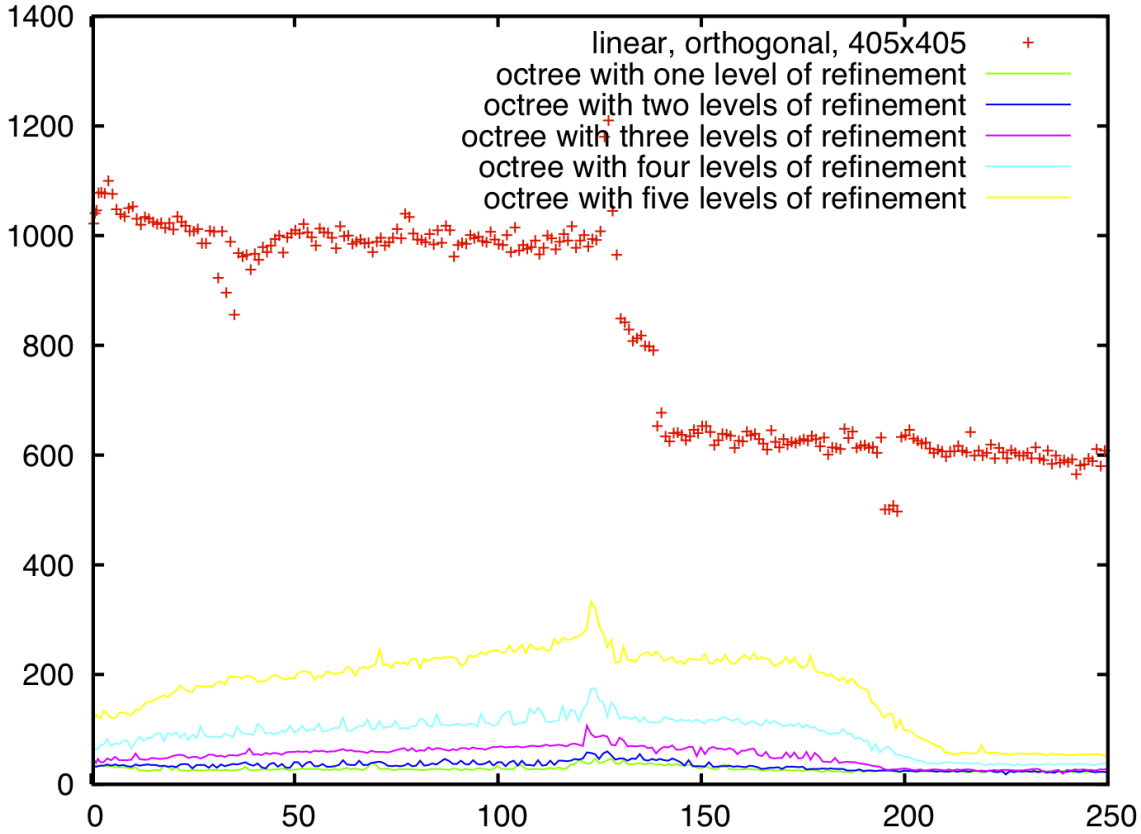

Рис. 2.9. Количество линейных итераций на каждом шаге на сетках типа восьмеричное дерево с разным уровнем сгущения.

2.7. Вычисление вариации нелинейной аппроксимации потока

В следующем тесте мы хотим проанализировать эффективность решения задачи в зависимости от приближения вариации нелинейной двухточечной аппроксимации потока в (2.28) и (2.29). Повернем ось тензора проницаемости $\mathbb{K} = diag(1000, 100, 50)$ на 45° вокруг оси Oz, вдоль плоскости Oxy и посчитаем ту же задачу на возмущенной сетке. Зафиксируем сетку $135 \times 135 \times 1$. Мы не будем

Сетка	Время, сек	T_{ref}/T_k
Эталонная ортогональная сетка		
405×405	T_{ref}=83837	1
Адаптивная грубая сетка $5 \times 5 \times 1$		
1 уровень	$T_1 = 39.2$	2139
2 уровня	$T_2 = 47.6$	1764
3 уровня	$T_3 = 104$	808
4 уровня	$T_4 = 361$	233
5 уровней	$T_5 = 2105$	40

Таблица 2.2. Время счета на эталонной сетке и на адаптивных сетках типа восьмеричное дерево с разными уровнями сгущения.

делать сравнение с линейной двухточечной аппроксимацей потока, так как с этой аппроксимацией задача сойдется к неверному решению. В таблице 2.3 представленно полное время решения задачи в секундах, число линейных итераций (метод BiCGStab с предобуславливателем ILU(1), остановка итераций при невязке менее 10^{-12}), и время решения, затраченное на итерацию, для трех подходов. В первом подходе используется (2.28)-(2.29) с полной вариацией (2.30), что приводит к наименее разреженному Якобиану (в таблице обозначено как корректный Якобиан). Второй подход заключается в фиксировании коэффициентов $\Delta D_f^{\pm} = 0$ в (2.28)-(2.29), что приводит к упрощенному Якобиану, шаблон которого всегда совпадает с шаблоном, получаемым при линейной аппроксимации потока. В третьем подходе мы будем отбрасывать все элементы матрицы корректного Якобиана, которые меньше барьера – 10^{-7}. Второй подход оказывается не эффективным из-за большого числа нелинейных и линейных итераций, хотя цена каждой итерации выходит такая же как при линейной аппроксимации потока. Второй и третий

подходы имеют примерно одинаковую скорость сходимости, но третий оказывается эффективней на 10% из-за того, что решается система с более разреженной матрицей.

Метод	Полное время	Линейных итераций	Время одной итерации
корректный Якобиан	1063.4	17274	0.0615
упрощенный Якобиан	15329.77	371372	0.0412
барьер 10^{-7}	926.16	17227	0.0537

Таблица 2.3. Время решения задачи в § 2.7 при разных способах аппроксимации Якобиана.

Вывод ко второй главе

Во второй главе мы показали, что задачу двухфазного заводнения пористого резервуара с нефтью можно решить с той же точностью при гораздо меньшем числе неизвестных на адаптивных сетках типа восьмеричное дерево. Одна из не рассмотренных в данной работе проблем, которая так же может понизить эффективность решения задачи на динамических сетках – это необходимость апскейлинга гетерогенных физических свойств на грубые ячейки сетки.

Глава 3

Численная модель течения вязкой несжимаемой жидкости

3.1. Математическая модель

Основу модели составляют уравнения Навье-Стокса, описывающие нестационарное течение вязкой несжимаемой жидкости в безразмерной форме:

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = \mathbf{0} \quad & \text{in } \Omega\times(0,T), \\ \operatorname{div}\mathbf{u} = 0 \quad & \text{in } \Omega\times[0,T), \\ \mathbf{u}|_{t=0} = \mathbf{u}_0 \quad & \text{in } \Omega, \\ \mathbf{u}|_{\Gamma_1} = \mathbf{g}, \quad \left(\nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\mathbf{n}\right)\Big|_{\Gamma_2} = \mathbf{0}, & \end{aligned} \tag{3.1}$$

где $\mathbf{u}$, p - неизвестные скорость жидкости и кинематическое давление, ν - кинематическая вязкость, Γ_2 - граница с условием свободного вытока, $\mathbf{n}$ - вектор-нормаль к Γ_2, а Γ_1 - остальная граница $\partial\Omega$. Чтобы решение было единственное, потребуем задания начального условия $\mathbf{u}(\mathbf{x}, t=0) = \mathbf{u}_0$, $p(\mathbf{x}, t=0) = p_0$, для $\mathbf{x}\in\Omega$. Где Ω - область с кусочно-гладкой границей.

При разнесенном расположении неизвестных скорости и давления на кубических сетках степени свободы давления располагаются в центрах ячеек, а компоненты скоростей располагаются в гранях таким образом, что каждая грань содержит компоненту, направленную вдоль нормали к этой грани. Мы условимся, что размер двух соседних ячеек сетки типа восьмеричное дерево не может отличаться более чем в двое. Если грань является общей для ячеек разных размеров, тогда большая ячейка имеет четыре грани, в центре каждой из которых находится своя

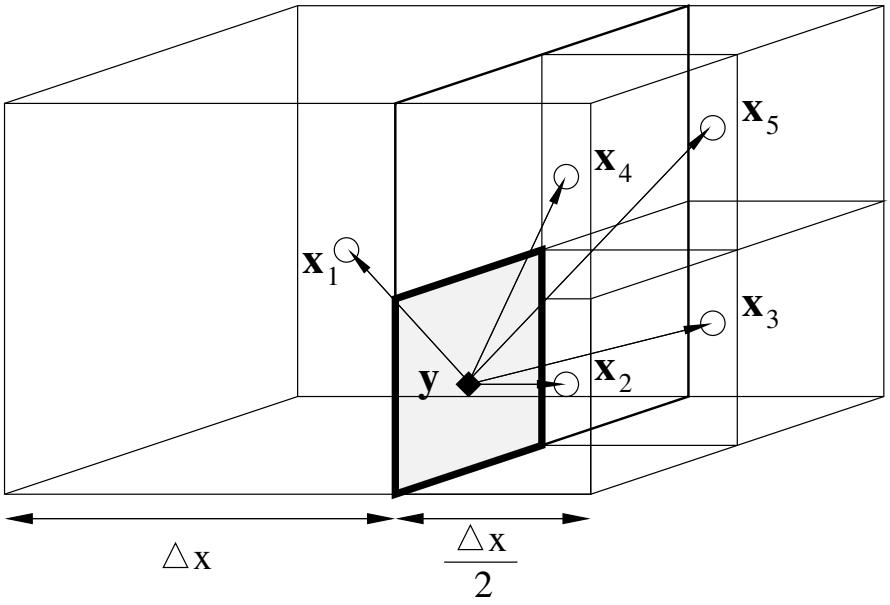

Рис. 3.1. Шаблон дискретизации $\partial p/\partial x$.

степень свободы.

3.2. Разложение Гельмгольца

3.2.1. Оператор дивергенции скорости

Начнем с описания конечно-разностной аппроксимации дивергенции скорости $\operatorname{div}\mathbf{u}$. Воспользуемся формулой Гаусса в центре $\mathbf{x}_V$ ячейки V

$$\int_V \operatorname{div}\mathbf{u}\,\mathrm{d}\mathbf{x} = \int_{\partial V} \mathbf{u}\cdot\mathbf{n}\,\mathrm{d}\mathbf{s}, \tag{3.2}$$

где $\mathbf{n}$ - единичная нормаль к грани ячейки, направленная наружу. Пусть $\mathcal{F}(V)$ - множество всех граней F ячейки V, то есть $\partial V = \cup_{F\in\mathcal{F}(V)}F$, и $\mathbf{x}_F$ обозначает центр $F \in \mathcal{F}(V)$. Определим оператор дивергенции следующим образом

$$(\operatorname{div}_h \mathbf{u}_h)(\mathbf{x}_V) = |V|^{-1} \sum_{F\in\mathcal{F}(V)} |F|(\mathbf{u}_h\cdot\mathbf{n})(\mathbf{x}_F). \tag{3.3}$$

Благодаря разнесенному расположению степеней свободы скорости, потоки $(\mathbf{u}_h \cdot \mathbf{n})(\mathbf{x}_F)$ легко аппроксимируются.

3.2.2. Оператор градиента давления

Один из способов определить дискретный градиент, - определить его как сопряженный к дискретной дивергенции. Мы определим ∇_h иначе, основываясь на разложении Тейлора. Для каждой внутренней грани мы припишем компоненту $\nabla_h p$ следующим образом. Выпишем компоненту p_x для компоненты скорости u. Так как мы условились, что в нашей сетке типа восьмиричное дерево размер соседних ячеек не может отличаться более чем в двое, то возможно всего два геометрических случая. Если с двух сторон от грани обе ячейки равного размера, то будем использовать аппроксимацию центральными разностями. В случае, если обе ячейки разного размера, то аппроксимация p_x в центре грани $\mathbf{y}$ изображена на Рис. 3.1: Рассмотрим центры $\mathbf{x}_1, \ldots, \mathbf{x}_5$ пяти ближайших ячеек, изображенных на рисунке, и разложим в ряд Тейлора значение давления $p(\mathbf{x}_i)$ по отношению к $p(\mathbf{y})$:

$$p(\mathbf{x}_i) = p(\mathbf{y}) + \nabla p(\mathbf{y}) \cdot (\mathbf{x}_i - \mathbf{y}) + O(|\mathbf{x}_i - \mathbf{y}|^2).$$

Опуская члены второго порядка, мы получим переопределенную систему, решая которую методом наименьших квадратов получим следующее выражение для x-компоненты градиента [72]:

$$p_x(\mathbf{y}) \approx \frac{1}{3\Delta}(p_2 + p_3 + p_4 + p_5 - 4p_1). \tag{3.4}$$

Замечание 3.2.1. *Суперпозиция дискретного градиента и дискретной дивергенции в общем случае приводит к несимметричной матрице дискретизации оператора Лапласа. Такая система эффективно решается итерационным мето-*

дом в подпространстве Крылова с многосеточным предобуславливателем, результаты приведены в § 3.6.

В данном случае, чтобы матрица стала симметричной, положительно определенной, достаточно помножить каждую строку матрицы и соответствующий ей элемент из правой части на отрицательный объем ячейки [73]. Однако, предложенный в § 3.5 метод задания граничных условий для давления все равно приведет к несимметричной матрице.

Во многих алгоритмах, использующих метод расщепления для решения уравнений Навье-Стокса, описывающих течение нестационарной несжимаемой вязкой жидкости, ключевым шагом является дискретное разложение Гельмгольца данной сеточной векторной функции, такое, что $\int_{\partial\Omega} \mathbf{f} \cdot \mathbf{n} = 0$:

$$\begin{cases} \mathbf{f} = \mathbf{u} + \nabla p, \\ \operatorname{div} \mathbf{u} = 0, \\ \mathbf{u} \cdot \mathbf{n}|_{\partial\Omega} = \mathbf{f} \cdot \mathbf{n}|_{\partial\Omega}, \end{cases} \iff \begin{cases} -\operatorname{div} \nabla p = \operatorname{div} \mathbf{f}, \\ \left.\dfrac{\partial p}{\partial \mathbf{n}}\right|_{\partial\Omega} = 0, \\ \mathbf{u} = \mathbf{f} - \nabla p. \end{cases} \tag{3.5}$$

Ряд численных экспериментов показал, что дискретное разложение Гельмгольца на сетках типа восьмеричное дерево с разнесенным расположением неизвестных не устойчиво: при разложении гладкой функции $\mathbf{f}$, может появиться ошибка в $\mathbf{u}$ на границах сгущения. Эта ошибка для двумерной сетки схематически изображена на Рис. 3.2 и заключается в появлении локальных дискретных бездивергентных мод скорости.

В результате данные бездивергентные моды при проекции скорости на бездивергентное пространство могут систематически накапливаться, снижая тем самым точность решения и отвечая за перенос кинетической энергии в область между сгущениями.

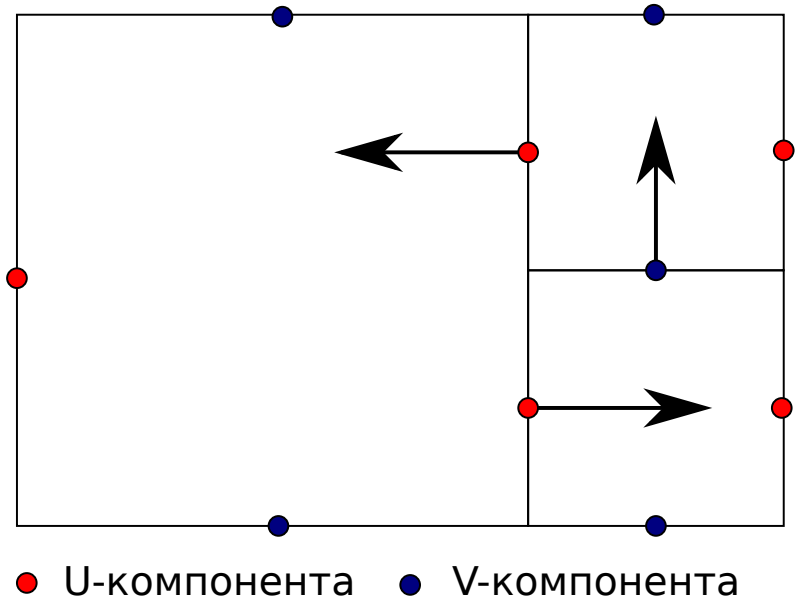

Рис. 3.2. Схематическое изображение локальной дискретной бездивергентной моды.

Покажем появление ошибки на двумерном примере с одним уровнем сгущения. Рассмотрим сетку с локальным сгущением, изображенную на Рис. 3.3 справа.

Функция $\mathbf{f}$ такова, что (3.5) имеет следующее решение:

$$
\begin{aligned}
u &= \sin\left(\frac{2\pi(e^x-1)}{e-1}\right)\left(1-\cos\left(\frac{2\pi(e^{ay}-1)}{e^a-1}\right)\right)\frac{1}{2\pi}\frac{e^x}{(e-1)},\\
v &= \left(1-\cos\left(\frac{2\pi(e^x-1)}{e-1}\right)\right)\sin\left(\frac{2\pi(e^{ay}-1)}{e^a-1}\right)\frac{a}{2\pi}\frac{e^{ay}}{(e^a-1)},\\
p &= a\cos\left(\frac{2\pi(e^x-1)}{e-1}\right)\cos\left(\frac{2\pi(e^{ay}-1)}{e^a-1}\right)\frac{e^{a+1}}{(e-1)(e^a-1)},
\end{aligned}
\tag{3.6}
$$

где $a = 0.1$, $\mathbf{u} = (u, v)^T$.

Из Рис. 3.3 можно видеть, что ошибка в скорости сосредоточена на границе между разными уровнями сгущения сетки. В данной ошибке доминируют локально-бездивергентные моды. Далее мы применим низкочастотный фильтр для подавления данных мод, который опирается на свойства оператора ∇_h, описанного выше.

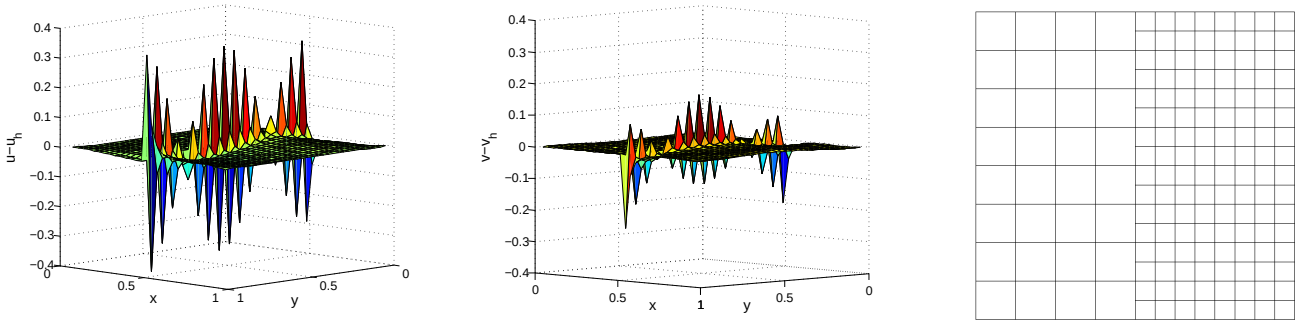

Рис. 3.3. Ошибка при решении (3.5) на сетке с локальным сгущением. Ошибка u-компоненты – слева, ошибка v-компоненты – по середине, справа изображена сетка при $h = 1/8$.

3.2.3. Низкочастотный фильтр

Определим низкочастотный фильтр G, исключающий локальные бездивергентные моды на границах между разными уровнями сгущения сетки Γ_{cf}:

$$G \circ u(\mathbf{x}) = \begin{cases} \dfrac{1}{4}\displaystyle\sum_{i=1}^{4} u(\mathbf{x}_i) & \text{если } \mathbf{x} \in \Gamma_{cf}, \\ \\ u(\mathbf{x}) & \text{иначе.} \end{cases}$$

Здесь Γ_{cf} обозначает объединение всех граней, которые граничат с ячейками разных размеров; $\mathbf{x}_i$ - четыре центра степеней свободы скорости, граничащих с одной и той же большой ячейкой. Фактически данный фильтр объединяет для оператора конвекции четыре степени свободы в одну.

Заметим, что выполняется следующее соотношение $G \circ \nabla = \nabla$, тогда для построенного фильтра, действующего на оператор конвекции верно соотношение:

$$\frac{\partial u}{\partial t} + G \circ [(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p] = \nu \Delta \mathbf{u}. \tag{3.7}$$

Таким образом, для невязкой жидкости, $\mathbf{u}$, вычисленное на следующем шаге не будет содержать ложных бездивергентных мод, если их не было на предыдущем.

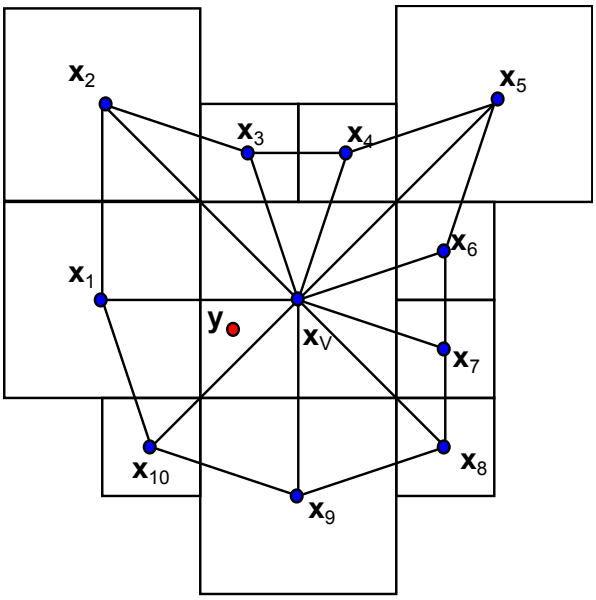

Рис. 3.4. Значение $u(\mathbf{y})$ определяется линейной интерполяцией в треугольнике $\mathbf{x}_V$, $\mathbf{x}_1$,$\mathbf{x}_{10}$ с известными значениями $u(\mathbf{x}_V)$, $u(\mathbf{x}_1)$,$u(\mathbf{x}_{10})$.

Мы не применяем фильтр к вязким членам, так как они подавляют возникающие бездивергентные моды.

Легко проверить, что фильтр не испортит свойство сохранения моментов, если таковое выполняется.

3.3. Дискретизация конвекции и диффузии

3.3.1. Интерполяция

Определим значение компонент скорости $\mathbf{u}(\mathbf{y})$ в произвольной точке $\mathbf{y}$. Пусть $\mathbf{y}$ принадлежит ячейке V, и нас интересует компонента $u(\mathbf{y})$. Рассмотрим плоскость $\mathcal{P}$, такую что $\mathbf{y} \in \mathcal{P}$ и $\mathcal{P} \perp Ox$. Пусть $\mathbf{x}_V \in \mathcal{P}$ - ортогональная проекция центра V на $\mathcal{P}$ и $\mathbf{x}_k$, $k = 1, \ldots, m$, $m \leq 12$, проекции центров ячеек, имеющих общую грань или ребро с V. Значения $u(\mathbf{x}_V)$ и $u(\mathbf{x}_k)$ определим через линейную интерполяцию со степеней свободы скорости u. Тогда $u(\mathbf{y})$ можно определить с

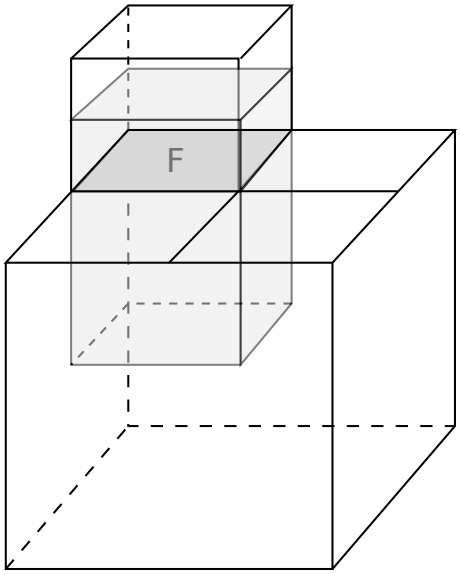

Рис. 3.5. Контрольный объем для грани F.

помощью линейной интерполяции между значениями u в вершинах треугольника, содержащего $\mathbf{y}$, как показано на Рис 3.4.

3.3.2. Оператор конвекции

Мы будем рассматривать оператор конвекции в дивергентной форме $\partial u\mathbf{u}/\partial x + \partial v\mathbf{u}/\partial y + \partial w\mathbf{u}/\partial z$. Рассмотрим компоненту скорости u с центром $\mathbf{x}_F$ лежащей на грани F. Определим кубический контрольный объем V' следующим образом. Данный объем V' имеет в сечении грань F и его границы задаются переносом F в нормальном направлении в стороны соседних с F ячеек на половину расстояния, равного размеру соседних ячеек. V' изображен на Рис. 3.5

Рассмотрим дискретизацию тангенсальной производной по отношению к грани F, $\partial vu/\partial y$, в центре грани $\mathbf{x}_F$. В дискретной форме выпишем производную следующим образом:

$$\frac{\partial vu}{\partial y} \approx \frac{v_R U_R F_R - v_L U_L F_L}{|V'|}, \tag{3.8}$$

где v_R, v_L, U_R, U_L - аппроксимация компонент скорости в центрах $\mathbf{x}_R$, $\mathbf{x}_L$ противо-

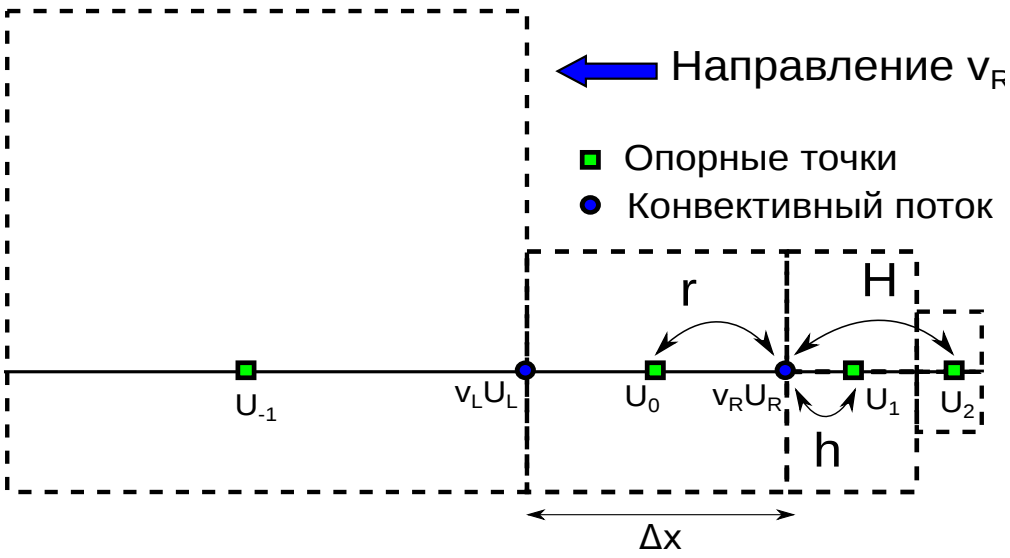

Рис. 3.6. Опорные точки для схемы аппроксимации конвекции. Иллюстрация для тангенсальной производной к грани, на которой расположена степень свободы скорости.

положных ребер грани F, перпендикулярных оси Oy; F_R и $F_L \in \mathcal{F}(V')$ - площади соответствующих граней контрольного объема V'. Рассмотрим приближение конвективного потока $v_R U_R$ в точке $\mathbf{x}_R$.

Если размер двух соседних с F ячеек совпадает, мы можем усреднить v_R с двух ближайших степеней свободы компоненты скорости v; если размер ячеек различается, то мы можем определить $v_R = v(\mathbf{x}_R)$ с помощью процедуры, описанной в § 3.3.1.

Для определения U_R возьмем четыре 'опорных' точки ($\mathbf{x}_{-1}$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_0 := \mathbf{x}$). Отметим, что $\mathbf{x}_{-1}$, $\mathbf{x}_1$, и $\mathbf{x}_2$ не обязательно являются узлами сетки. Значения u_{-1}, u_1, и u_2 (Рис 3.6) в этих точках вычисляются с помощью интерполяций.

Если опорная точка принадлежит ячейке *меньшей*, чем ячейка для $\mathbf{x}_0$ (точки $\mathbf{x}_1$ и $\mathbf{x}_2$ на Рис. 3.6), тогда используется линейная интерполяция между *двумя* узлами с ближайших граней.

Если ячейка принадлежит ячейке *большей*, чем ячейка для $\mathbf{x}_0$ (точка $\mathbf{x}_{-1}$ на Рис.3.6), тогда мы поступим так же, как в § 3.3.1 при определении $u(\mathbf{y})$ в точке

$\mathbf{y} = \mathbf{x}_{-1}$, с тем отличием, что мы вместо линейной интерполяции по треугольнику, содержащему $\mathbf{y}$, построим для всех проекций $\mathbf{x}_\mathbf{V}$, $\mathbf{x}_k$, $k = 1, \ldots, m$ с помощью взвешенного метода наименьших квадратов полином второго порядка Q_2 от двух переменных y и z, и положим $u_{-1} := Q_2(\mathbf{x}_{-1})$. В качестве весов для метода наименьших квадратов мы возьмем $e^{-8|\mathbf{x}_{-1}-\mathbf{x}_k|/s(V)}$, где $s(V)$ размер ячейки V.

При применении полинома первого порядка так же достигается второй порядок сходимости на аналитическом тесте, однако норма ошибок L_∞ и L_2 получается хуже, поэтому мы далее не будем рассматривать такой подход.

Определив значения $\{u_i\}$, $i = -1, \ldots, 2$ и v_R вычислим U_R в $\mathbf{x}$. Если $v_R > 0$, мы будем использовать опорные точки u_{-1}, u_0, u_1, иначе, если $v_R < 0$ мы будем использовать u_0, u_1, u_2. Пусть $v_R < 0$. Пользуясь обозначениями Рис. 3.6, определим U_R следующим образом:

$$\begin{aligned} U_R = D^{-1}[u_0(hH^2 - h^2H) + u_1(rH^2 + r^2H) - u_2(hr^2 + h^2r)] \\ + \lambda\triangle x^2(u_0(H - h) - u_1(H + r) + u_2(r + h))], \end{aligned} \tag{3.9}$$

где $D = (r + h)(H - h)(H + r)$, а параметр λ задает семейство схем против потока второго порядка аппроксимации. Данный метод определения конвективного потока аналогичен схеме QUICK [74] при $\lambda = 0$. При $\lambda = -1$ на равномерной сетке получается схема из [75], одновременно сохраняющая момент и кинетическую энергию. Мы далее будем использовать схему QUICK. Если мы находимся возле границы, опорной точки $\mathbf{x}_2$ может не существовать, тогда мы будем использовать точку $\mathbf{x}_{-1}$. Найдем U_L аналогично, рассматривая значение v_L и опорные точки $(\mathbf{x}_{-2}, \mathbf{x}_{-1}, \mathbf{x}_1, \mathbf{x}_0 := \mathbf{x})$.

Аппроксимацию производной компоненты u скорости в нормальном направлении, $\partial uu/\partial x$, можно получить таким же образом, через $\partial uu/\partial x \approx (u_R U_R F_R - u_L U_L F_L)/|V'|$. Определив u_R и u_L линейной интерполяцией в грани контрольного

объема, мы затем определим U_R и U_L интерполяцией против потока по формуле (3.9).

Построенный нами оператор конвекции не сохраняет импульс на стыках между грубыми и мелкими ячейками. Различные модификации оператора, позволяющие сохранить импульс, приводят к тому, что на аналитической задаче происходит падение точности сходимости метода до первого порядка, поэтому мы оставим оператор в этом виде.

Замечание 3.3.1. *Схемы против потока априори не выполняют закон сохранения кинетической энергии. Известно, что построение схем решений уравнений Эйлера, сохраняющих кинетическую энергию, накладывает ряд сильных ограничений: на методы интерполяции[76, 77], на дискретный оператор дивергенции [78], на дискретизацию по времени [75, 77], на постановку граничных условий [79, 80], требуется выполнение определенной симметрии от дискретных операторов [81]. Нарушать закон сохранения энергии может так же метод расщепления [77].*

Автору данной работы известно всего две работы, в которых одновременно дискретно сохраняется момент, масса и кинетическая энергия [77, 82] при решении трехмерных уравнений Эйлера, описывающих течение невязкой несжимаемой жидкости. Обе работы используют полностью неявный метод решения уравнений Навье-Стокса.

Построение схемы для сеток типа восьмеричное дерево с учетом всех описанных ограничений может стать бесдиссипативной альтернативой схеме, предложенной в данной работе.

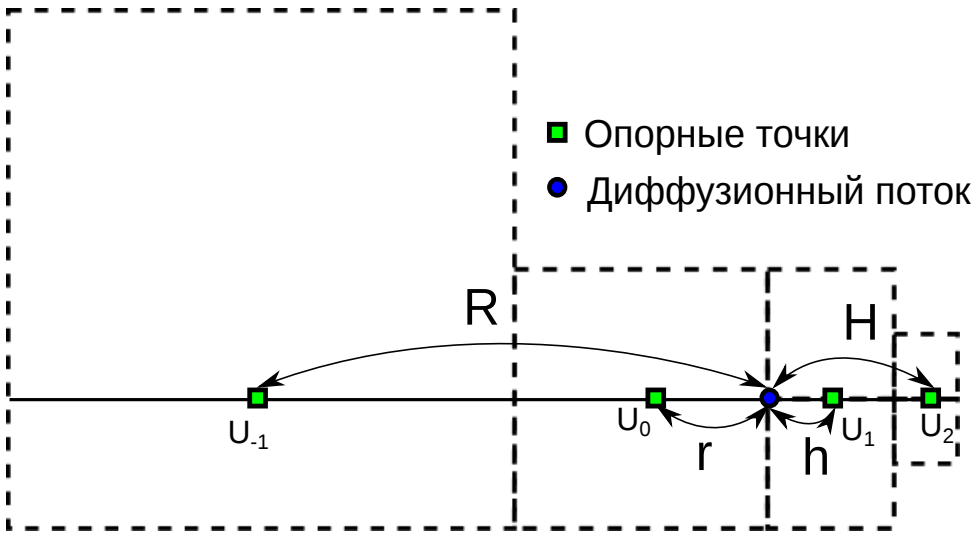

Рис. 3.7. Опорные точки для аппроксимации диффузионного потока.

3.3.3. Оператор диффузии

Обозначим

$$(\Delta_h u)(\mathbf{x}) = -|V'|^{-1} \sum_{F' \in \mathcal{F}(V')} |F'|(\nabla_h u \cdot \mathbf{n})(\mathbf{y}_{F'}). \tag{3.10}$$

Чтобы приблизить диффузионный поток в центре грани $\mathbf{y}_{F'}$, где $F' \in \mathcal{F}(V')$, мы возьмем четрые опорные точки $(\mathbf{x}_{-1}, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$ как показано на Рис. 3.7. Значения скорости u_{-1}, u_0, u_1, и u_2 приписываются к опорным точкам таким же образом, как и для конвективных членов выше в § 3.3.2. Пользуясь обозначениями как на Рис. 3.7, выпишем аппроксимацию диффузионного потока $(\nabla u \cdot \mathbf{n})$ с третьим порядком:

$$\begin{aligned}(\nabla u \cdot \mathbf{n}) \approx D^{-1}\big[&(h^2H^3 + h^3R^2 - H^3R^2 + h^2R^3 - H^2R^3 - h^3H^2)u_0 \\ &+ (H^3R^2 + r^3R^2 + H^2R^3 - r^2R^3 - H^3r^2 - H^2r^3)u_1 \\ &+ (h^3r^2 + h^2r^3 - h^3R^2 - r^3R^2 - h^2R^3 + r^2R^3)u_{-1} \\ &+ (h^3H^2 - h^2H^3 - h^3r^2 + H^3r^2 - h^2r^3 + H^2r^3)u_2\big],\end{aligned}$$

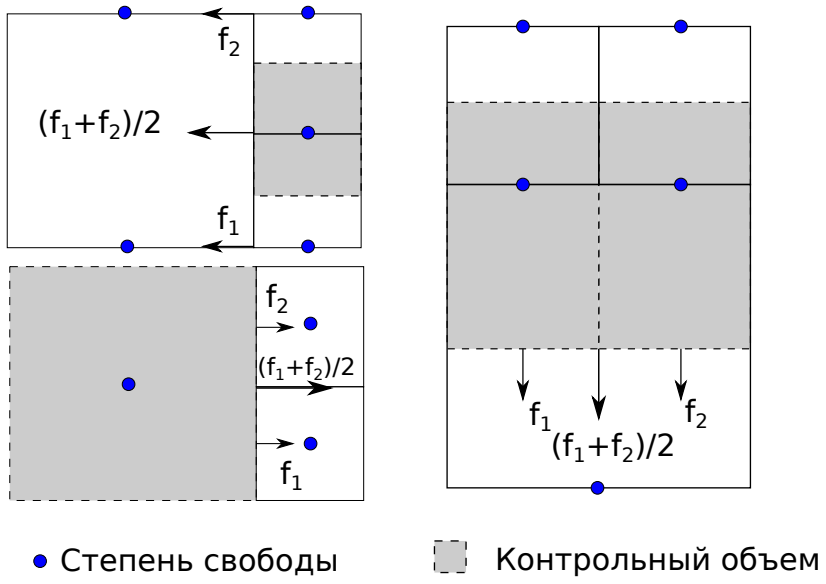

Рис. 3.8. Слева: осреднение тангенсального потока диффузии; Справа: осреднение нормального потока диффузии.

где $D = (H-h)(h+r)(H+r)(h+R)(H+R)(R-r)$. Если нельзя найти опорную точку $\mathbf{x}_2$ возли границы, то мы можем использовать точку $\mathbf{x}_{-2}$.

Чтобы уравнять потоки через грани контрольного объема для построенного сеточного оператора диффузии, нам необходимо усреднить тангенсальный диффузионный поток для ячеек большего размера при изменении шага сетки с двух степеней свободы в тангенсальном направлении (Рис. 3.8 слева, снизу), с двух степеней свободы в нормальном направлении (Рис. 3.8 слева, сверху). Нормальный диффузионный поток необходимо усреднить с четырех степеней свободы в нормальном направлении, что схематически изображено в плоскости на Рис. 3.8 (справа).

3.4. Интегрирование по времени

Для интегрирования по времени мы используем полунеявную схему расщепления (известную так же как проекционная схема [45]). Имея аппроксимации $\mathbf{u}^n$,

p^n к $\mathbf{u}(t), p(t)$, найдем аппроксимации $\mathbf{u}^{n+1}$, p^{n+1} к $\mathbf{u}(t+\Delta t^n), p(t+\Delta t^n)$ в несколько шагов. Сначала предскажем значение скорости $\widetilde{\mathbf{u}^{n+1}}$, решив уравнение конвекции-диффузии-реакции с фильтром, действующим на конвективный член,

$$\begin{cases} \dfrac{\alpha\widetilde{\mathbf{u}^{n+1}} + \beta\mathbf{u}^n + \gamma\mathbf{u}^{n-1}}{\triangle t^n} + \\ G \circ ((\mathbf{u}^n + \xi(\mathbf{u}^n - \mathbf{u}^{n-1})) \cdot \nabla\widetilde{\mathbf{u}^{n+1}}) - \\ \qquad\qquad \nu\Delta\widetilde{\mathbf{u}^{n+1}} = -\nabla p^n, \\ \widetilde{\mathbf{u}^{n+1}}|_{\Gamma_1} = \mathbf{g}, \quad \left.\dfrac{\partial\widetilde{\mathbf{u}^{n+1}}}{\partial\mathbf{n}}\right|_{\Gamma_2} = 0. \end{cases} \tag{3.11}$$

Здесь $\xi = \triangle t^n/\triangle t^{n-1}$, $\alpha = 1 + \xi/(\xi+1)$, $\beta = -(\xi+1)$, $\gamma = \xi^2/(\xi+1)$. Далее, спроектируем $\widetilde{\mathbf{u}^{n+1}}$ на бездивергентное пространство, чтобы получить $\mathbf{u}^{n+1}$:

$$\begin{cases} \alpha(\mathbf{u}^{n+1} - \widetilde{\mathbf{u}^{n+1}})/\triangle t^n - \nabla q = 0, \\ \qquad\qquad \operatorname{div}\mathbf{u}^{n+1} = 0, \\ \mathbf{n}\cdot\mathbf{u}^{n+1}|_{\Gamma_1} = \mathbf{n}\cdot\mathbf{g}, \quad q|_{\Gamma_2} = 0. \end{cases} \tag{3.12}$$

Задача (3.12) сводится к уравнению Пуассона для q:

$$\begin{cases} -\Delta q = \alpha \operatorname{div}\widetilde{\mathbf{u}^{n+1}}/\triangle t^n, \\ q|_{\Gamma_2} = 0, \quad \left.\dfrac{\partial q}{\partial\mathbf{n}}\right|_{\Gamma_1} = 0. \end{cases} \tag{3.13}$$

Получим новую скорость:

$$\mathbf{u}^{n+1} = \widetilde{\mathbf{u}^{n+1}} + \triangle t^n\nabla q/\alpha, \tag{3.14}$$

и новое давление:

$$p^{n+1} = p^n + q/\alpha + \nu\operatorname{div}\widetilde{\mathbf{u}^{n+1}}, \tag{3.15}$$

где прибавка $\nu\operatorname{div}\widetilde{\mathbf{u}^{n+1}}$ введена в [83] для компенсации ошибки расщепления вблизи границы. В [84] была рассмотрена аналогичная схема интегрирования по времени с фиксированным шагом и было показано, что схема имеет второй порядок

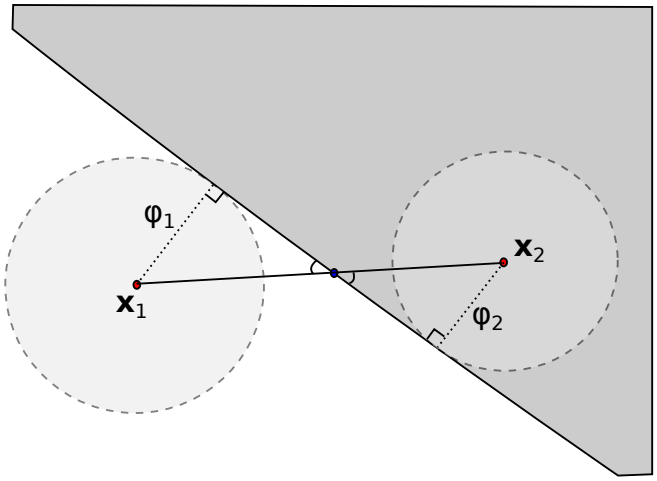

Рис. 3.9. Аппроксимация положения границы.

по скорости и в лучшем случае $\frac{3}{2}$ по давлению. Детальный анализ ошибки для схемы расщепления с неявным конвективным членом дан в [85], показано, что схема имеет второй порядок сходимости по скорости. При наличии условий вытекания, получение устойчивого проекционного метода является нерешенной проблемой [86, 87]. Если мы возьмем $\nu \frac{\partial \widetilde{\mathbf{u}^{n+1}}}{\partial \mathbf{n}}\Big|_{\Gamma_2} = p^n \mathbf{n}$ в (3.11), в [86] доказано, что метод расщепления имеет порядок $\frac{3}{2}$. Однако, численные эксперименты показывают неустойчивость метода, если ν недостаточно велико, поэтому мы не будем использовать подобный подход.

3.5. Расчетная область и граничные условия

Для обозначения криволинейной границы мы можем воспользоваться функцией уровня φ, заданной в вершинах ячеек. Ноль данной функции будет обозначать положение границы, положительное значение $\varphi > 0$ - внутреннюю область, отрицательное $\varphi < 0$ - внешнюю. Если значение данной функции будет обозначать расстояние до границы со знаком, то зная значение функции в двух точках

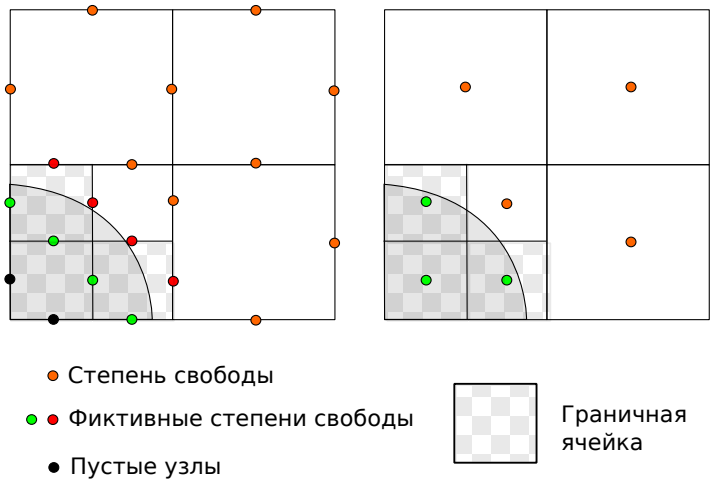

Рис. 3.10. Задание граничных ячеек и фиктивных степеней свободы. Значения в ложных степенях свобода определяются (зеленый) в тангенсальном направлении по отношению к грани и (красный) нормальном направлении по отношению к грани.

$\varphi_1 = \varphi(\mathbf{x}_1)$, $\varphi_2 = \varphi(\mathbf{x}_2)$, можно приблизить расстояние до границы вдоль луча $(\mathbf{x}_1, \mathbf{x}_2)$ по формуле (Рис. 3.9):

$$h = \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|\varphi_1}{\varphi_1 - \varphi_2}. \tag{3.16}$$

Определим дискретную расчетную область Ω_h. Будем считать ячейку V внутренней, если в центрах $\mathbf{x}_F$ всех ее граней $F \in \mathcal{F}(V)$ верно $\varphi(\mathbf{x}_F) \geq 0$. Ячейку будем считать граничной, если хоть одна ячейка, соседняя к данной через ребро или грань является внутренней. Все остальные ячейки будем считать пустыми. Для задания граничных условий мы можем воспользоваться концепцией фиктивных степеней свободы. Если две соседние с F ячейки являются внутренними, то мы определим в $\mathbf{x}_F$ степень свободы. Если одна соседняя с F ячейка является граничной, а другая внутренней, ты мы определим в $\mathbf{x}_F$ фиктивную степень свободы, которую мы будем определять через степень свободы в нормальном направлении по отношению к грани F. Если обе соседних с F ячейки являются граничными,

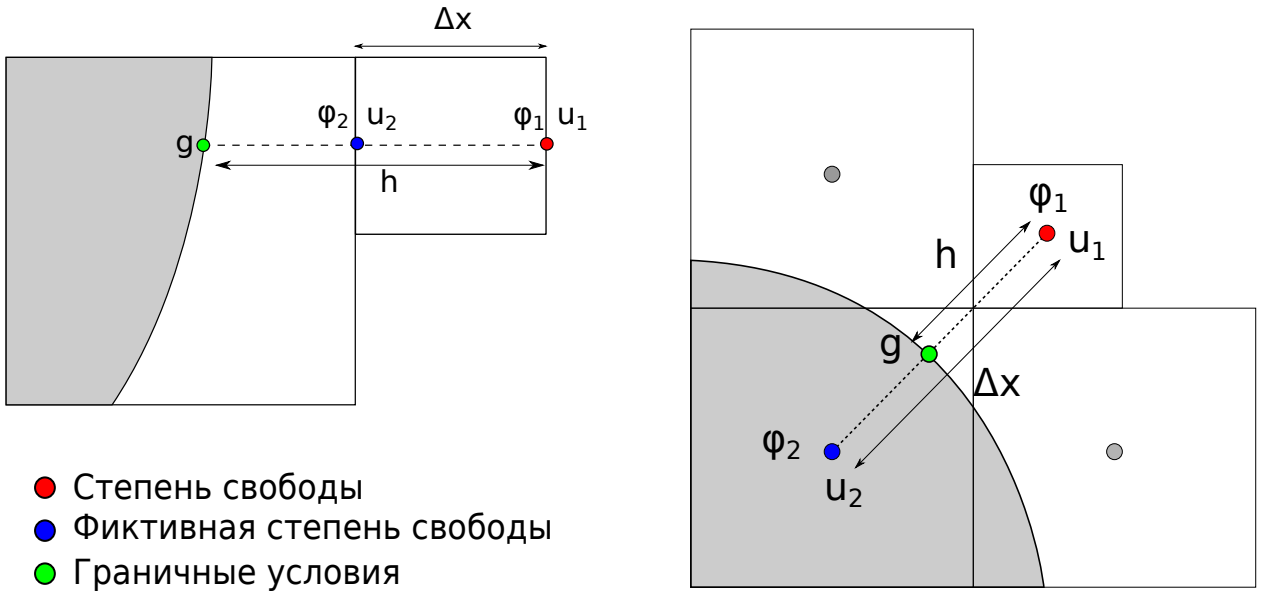

Рис. 3.11. Слева: аппроксимация скорости за криволинейной границей в нормальном направлении; Справа: аппроксимация скорости за криволинейной границей в тангенсальном направлении по отношению к грани.

то мы определим в $\mathbf{x}_F$ фиктивную степень свободы, которую будем определять через (возможно, фиктивную) степень свободы в тангенсальном направлении по отношению к грани F. Пример изображен на Рис. 3.10.

Мы рассмотрим только условия типа Дирихле, так как нам не прийдется сталкиваться в дальнейшем одновременно с криволинейными границами и условиями типа Неймана. Используя обозначения на Рис. 3.11 (слева), мы можем определить значение фиктивной степени свободы u_2 со вторым порядком по следующей формуле:

$$u_2 = (1 - \frac{\triangle x}{h})u_1 + \frac{\triangle x}{h}g, \tag{3.17}$$

где φ_1 и φ_2 определяются через линейную интерполяцию с узлов ячейки, а h определяется по формуле (3.16). Аналогичным образом определяется значение для фиктивных степеней свободы в тангенсальном направлении по отношению к F. При неоднозначности в определении ближайшей степени свободы (возмож-

но, фиктивной), мы можем взять (фиктивную) степень свободы по диагонали, как изображено на Рис. 3.11 (справа). Заметим, что в нормальном направлении мы всегда будем использовать интерполяцию, что соответствует методу [88], а в тангенсальном - экстраполяцию значений, что соответствует методу [89]. При экстраполяции, если $h < \triangle x/10$, то мы возьмем $h = \triangle x/10$.

Мы можем избежать экстраполяций совсем, поместив все ложные степени свободы внутрь области, рассмотрим этот подход далее в задаче обтекания цилиндра с круглым сечением. Такой подход имеет смысл, так как мы избежим неустойчивости при отсутствии гладкости течения вблизи границы.

Далее, чтобы решить уравнение Пуассона (3.13) во внутренних ячейках, нам необходимо задать граничные условия для поправки к давлению q. Мы можем определить ∇q_2 для u_2 через ∇q_1 для u_1 по следующей формуле:

$$\nabla q_2 = (1 - \frac{\triangle x}{h})\nabla q_1. \tag{3.18}$$

Подставив выражения из (3.17) и (3.18) в (3.15), получим, что значение скорости фиктивной степени свободы после проекции на бездивергентное пространство так же соответствует интерполяции (3.17).

В данном методе никак не учитываются законы сохранения, тем не менее он является популярным благодаря своей простоте [89]. Условия типа Неймана так же реализуемы в рамках данной концепции. В данной работе для задания условий Неймана на вытоке мы будем пользоваться методом первого порядка.

Так как построенная схема (3.11)–(3.15) аппроксимирует исходную систему уравнений на следующем уровне по времени, то все граничные условия мы будем задавать неявно.

Таблица 3.1. Ошибка на регулярной сетке.

вязкость	$\nu = 10^{-5}$				$\nu = 1$			
шаг сетки h	1/16	1/32	1/64	1/128	1/16	1/32	1/64	1/128
шаг по времени $\triangle t$	1/20	1/40	1/80	1/160	1/20	1/40	1/80	1/160
$\|\mathbf{u} - \mathbf{u}_h\|_{L^\infty}$	2.3e-3	6.1e-4	1.8e-4	4.7e-5	3.1e-2	7.2e-3	1.8e-3	4.9e-4
$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$	7.1e-4	1.0e-4	2.5e-5	6.3e-6	1.3e-2	2.3e-3	5.8e-4	1.5e-4
$\|p - p_h\|_{L^\infty}$	9.3e-3	2.3e-3	6.3e-4	1.6e-4	8.4e-1	4.9e-1	2.2e-1	8.8e-2
$\|p - p_h\|_{L^2}$	2.3e-3	1.5e-3	4.2e-4	1.1e-4	4.3e-1	1.6e-1	7.6e-2	3.2e-2

3.6. Численные эксперименты

3.6.1. Пример с известным аналитическим решением

Чтобы проверить точность схемы, мы рассмотрим известное точное решение для уравнений Навье-Стокса, выведенное Эшером и Стейнманом [51]. Данная задача была предложена в качестве трехмерного аналога двумерной задачи о вихре Тейлора. Для заданных параметров a, d и кинематической вязкости ν, точное решение системы (3.1) в кубе $\Omega = [-1, 1]^3$ выглядит следующим образом:

$$
\begin{aligned}
u &= -a\left(e^{ax}\sin(ay+dz) + e^{az}\cos(ax+dy)\right)e^{-\nu d^2 t} \\
v &= -a\left(e^{ay}\sin(az+dx) + e^{ax}\cos(ay+dz)\right)e^{-\nu d^2 t} \\
w &= -a\left(e^{az}\sin(ax+dy) + e^{ay}\cos(az+dx)\right)e^{-\nu d^2 t} \\
p &= -\frac{a^2}{2}(e^{2ax} + e^{2ay} + e^{2az} + 2\sin(ax+dy)\cos(az+dx)e^{a(y+z)} \\
&\quad +2\sin(ay+dz)\cos(ax+dy)e^{a(z+x)} \\
&\quad +2\sin(az+dx)\cos(ay+dz)e^{a(x+y)})e^{-2\nu d^2 t}.
\end{aligned}
$$

Зафиксируем $a = \pi/4, d = \pi/2$. Ошибка, как и в [51], измеряется в момент времени $t = 0.1$.

Таблица 3.2. Зависимость ошибки от количества уровней локального сгущения.

вязкость	$\nu = 0$				$\nu = 1$			
шаг сетки $h_{\max}$	1/16	1/16	1/16	1/16	1/16	1/16	1/16	1/16
шаг сетки $h_{\min}$	1/16	1/32	1/64	1/128	1/16	1/32	1/64	1/128
	Без фильтра							
$\|\mathbf{u} - \mathbf{u}_h\|_{L^\infty}$	2.0e-3	1.9e-2	2.1e-2	2.0e-2	1.2e-3	1.3e-3	1.8e-3	1.7e-4
$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$	4.3e-4	1.1e-3	1.3e-3	1.4e-3	3.1e-4	3.3e-4	3.8e-4	4.0e-4
$\|p - p_h\|_{L^\infty}$	9.4e-3	2.5e-2	2.5e-2	2.4e-2	4.6e-2	4.6e-2	4.7e-2	4.8e-2
$\|p - p_h\|_{L^2}$	6.1e-3	4.9e-3	4.4e-3	4.2e-3	1.3e-3	1.3e-3	1.4e-2	1.4e-2
	С фильтром							
$\|\mathbf{u} - \mathbf{u}_h\|_{L^\infty}$	2.0e-3	2.3e-3	3.2e-3	3.1e-3	1.2e-3	1.3e-3	1.7e-3	1.7e-3
$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$	4.3e-4	5.2e-4	6.2e-4	6.7e-4	3.1e-4	3.3e-4	3.8e-4	4.0e-4
$\|p - p_h\|_{L^\infty}$	9.4e-3	1.1e-2	1.1e-2	1.1e-2	4.6e-2	4.6e-2	4.7e-2	4.7e-2
$\|p - p_h\|_{L^2}$	6.1e-3	5.5e-3	5.0e-3	4.7e-3	1.3e-3	1.3e-3	1.3e-2	1.3e-2

Сначала мы продемонстрируем порядок сходимости метода на последовательности равномерных сгущающихся сеток с дроблением шага по времени. Результат показан в Таблице 3.1. Метод показывает второй порядок как по скорости, так и по давлению при малой вязкости. При большой вязкости порядок сходимости по давлению падает до первого.

В следующих двух численных экспериментах мы продемонстрируем влияние фильтра на сходимость метода на последовательности неравномерных сеток с локальным сгущением. Зафиксируем шаг по времени $\triangle t = 0.01$ и будем сгущать сетку внутри сферы радиуса 0.5 с центром $(0, 0, 0)$. Размер самой большой и самой маленькой ячейки задан через $h_{\max}$ и $h_{\min}$ соответственно. Случай $h_{\max} = h_{\min}$ соответствует равномерной сетке. В Таблице 3.2 продемонстрированы два варианта теста, при $\nu = 0$ (предел Эйлера) и $\nu = 1$ (диффузия доминирует). Из

Таблица 3.3. Сходимость метода на последовательности сеток со сгущением при $\nu = 0.01$ с применением фильтра.

шаг сетки $h_{\max}$	1/8	1/16	1/32	1/64
шаг сетки $h_{\min}$	1/32	1/64	1/128	1/256
шаг по времени dt	1/50	1/100	1/200	1/400
$\|\mathbf{u} - \mathbf{u}_h\|_{L^\infty}$	6.3e-3	2.2e-3	6.7e-4	1.6e-4
$\|\mathbf{u} - \mathbf{u}_h\|_{L^2}$	2.1e-3	5.1e-4	1.2e-4	4.3e-5
$\|p - p_h\|_{L^\infty}$	5.4e-2	1.1e-2	2.9e-3	7.0e-4
$\|p - p_h\|_{L^2}$	2.3e-2	5.5e-3	1.4e-3	4.9e-4

результатов видно, что при отсутствии или малой диффузии фильтр подавляет возникающие ложные бездивергентные моды, а при отсутствии фильтра норма ошибки L^∞ возрастает в 10 раз. Тем не менее фильтр не может не влиять на точность аппроксимации метода, поэтому мы видим ухудшение нормы ошибки при применении фильтра и сгущении сетки. С другой стороны, при доминирующей диффузии не возникают ложные моды скорости и фильтр не влияет на точность решения.

Таблица 3.3 демонстрирует второй порядок сходимости для скорости и почти второй для давления на последовательности сеток с локальным сгущением.

3.6.2. Трехмерная каверна

Рассмотрим тестовый эксперимент с течением внутри трехмерной каверны с подвижной верхней крышкой. Условия задачи проиллюстрированы на Рис. 3.12. Мы ищем стационарное решение уравнений (3.1) в $\Omega = (0,1)^3$, с заданным $\mathbf{u} = (1,0,0)^T$ при $z = 1$ и $\mathbf{u} = (0,0,0)^T$ на остальной границе. Течение в каверне демонстрирует множество важных механических феноменов [52].

Нас интересуют стационарные решения для чисел Рейнольдца $Re = 100, 400, 1000$.

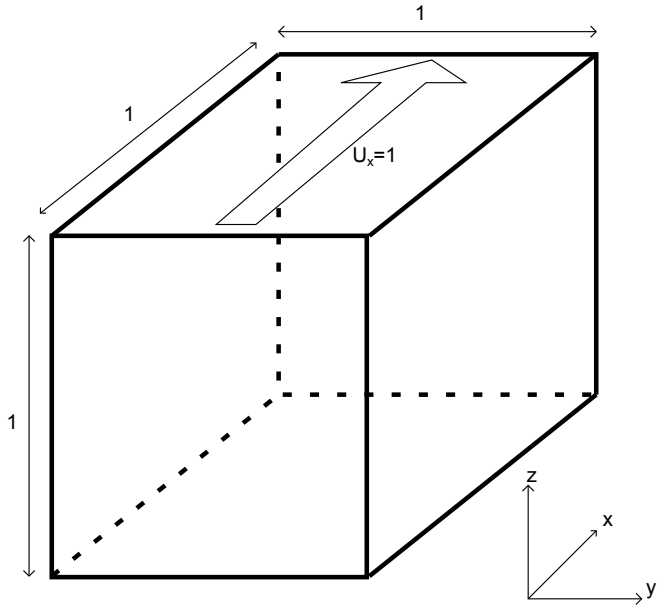

Рис. 3.12. Схематическое изображение условий теста с каверной.

Мы будем применять метод (3.11)–(3.15) до достижения стационарного состояния. Для решения задачи мы возьмем сетку с самым грубым уровнем $h_{max} = 1/32$ и сгустим сетку к границе области (пять слоев сетки с $h = 1/64$ и 2 слоя с $h_{min} = 1/256$). В результате мы получим 983256 степеней свободы давления и 2936724 степеней свободы скорости. На Рис. 3.13 (справа) изображена компонента скорости u вдоль линии $((0.5, 0.5, z), \ \ 0 \leq z \leq 1)$ при стационарном состоянии. Полученные результаты совпадают с результатами из [54].

На Рис 3.14 представлены контуры y-компоненты вихря $\nabla \times \mathbf{u}$ на плоскости $y = 0.5$. Поле скоростей в различных плоскостях изображены на Рис. 3.15. Воспроизведенная данным методом структура течения совпадает со структурой течения в [53, 54]. Помимо главного вихря, метод так же воспроизводит вторичный вихрь вверх по потоку при $Re = 1000$ and $Re = 400$, нижние вихри у стены для Re и верхние вихри у стены при $Re = 1000$ и $Re = 400$; видны завихренности вниз по потоку для $Re = 1000$ и $Re = 400$. Все эти структуры гладко переходят между

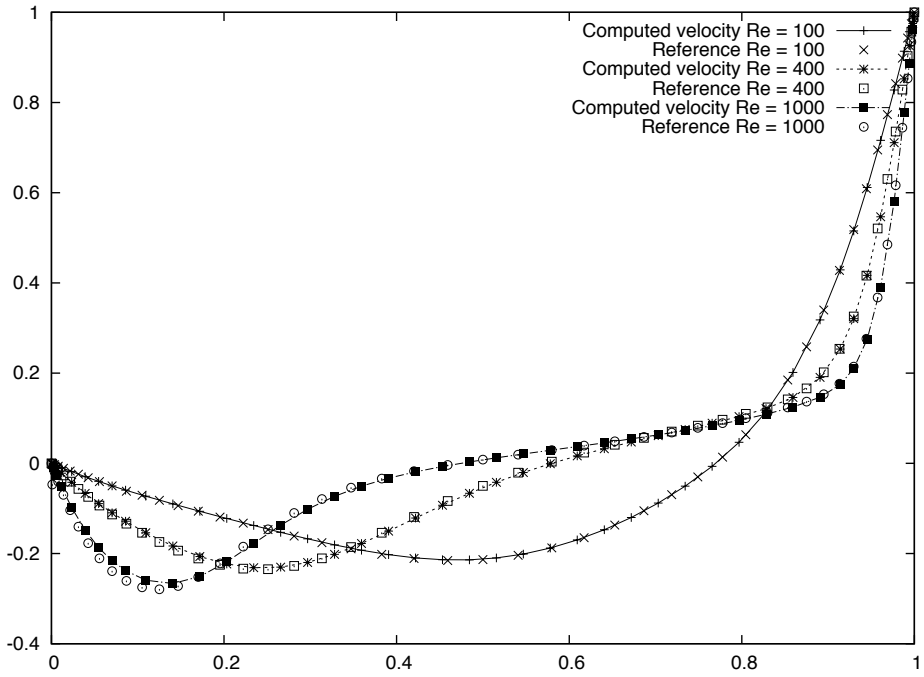

Рис. 3.13. u-компоненты скорости вдоль прямой $((0.5, 0.5, z),\ \ 0 \leq z \leq 1)$ в сравнении с данными из [54]

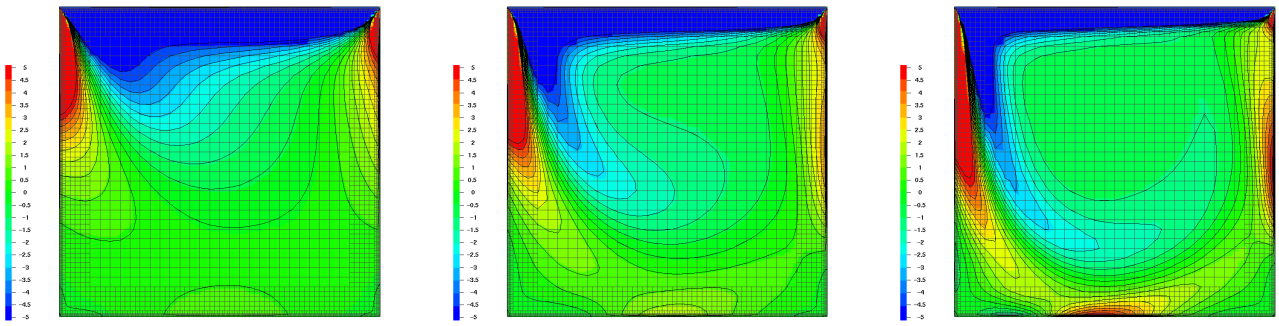

Рис. 3.14. y-компонента вихря в плоскости $y = 0.5$ для числа $Re = 100$ (слева), $Re = 400$ (посередине), $Re = 1000$ (справа).

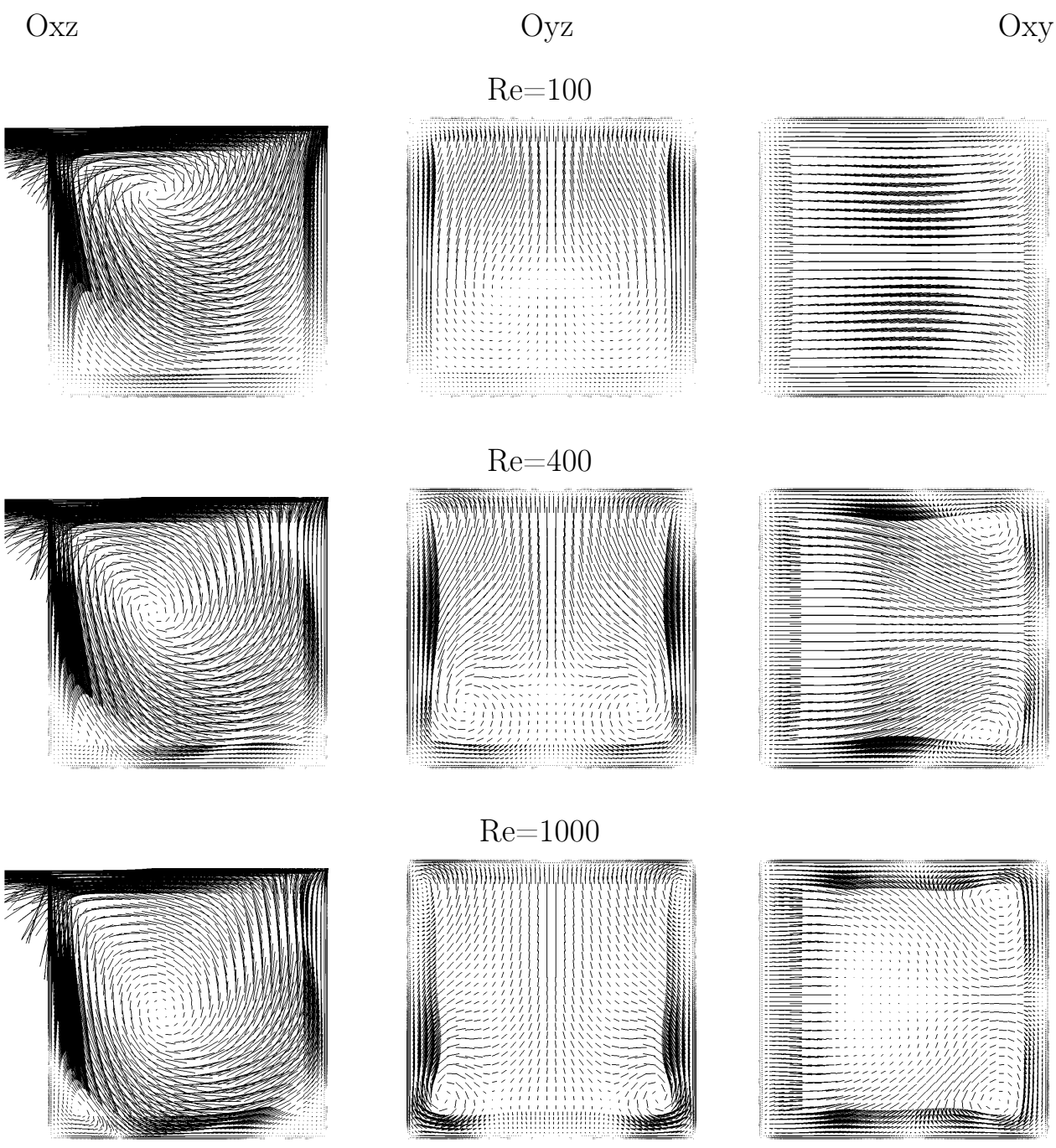

Рис. 3.15. Поле скоростей в плоскостях $y = 0.5$, $x = 0.5$ и $z = 0.5$ при числах $Re = 100, 400, 1000$.

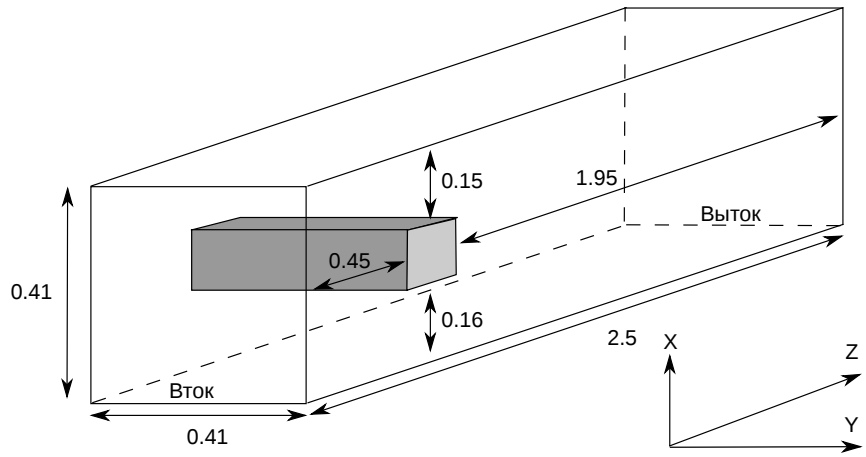

Рис. 3.16. Условия задачи обтекания цилиндра с квадратным сечением.

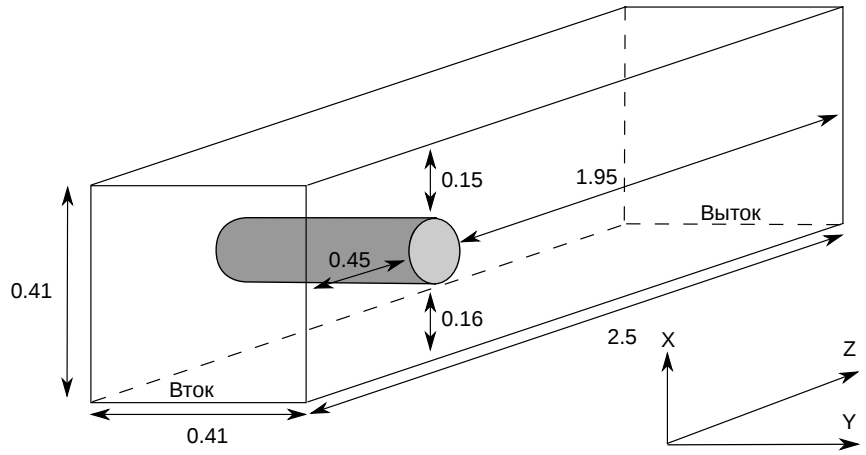

Рис. 3.17. Условия задачи обтекания цилиндра с круглым сечением.

разными уровнями сгущения сетки. Способность метода корректно предсказать вторичные структуры течения показывают низкую численную вязкость схемы.

3.6.3. Обтекание цилиндра

В последнем численном эксперименте мы рассмотрим трехмерное обтекание цилиндра в канале. Мы рассмотрим два вида цилиндра - с квадратным и круглым сечением. Данная тестовая задача была поставлена Шафером и Тureком в [55]. Эта же задача рассмотрена в [56, 90].

Геометрия расчетной области представлена на Рис. 3.16 и Рис. 3.17. На границах канала и цилиндра стоят условия условия прилипания и непротекания $\mathbf{u} = 0$. Параболический профиль скорости задан на втоке:

$$\mathbf{u} = (0, 0, 16\widetilde{U}xy(H-x)(H-y)/H^4)^T \quad \text{on } \Gamma_{\text{inflow}},$$

где $H = 0.41$, а $4\widetilde{U}/9$ - характеристическая скорость. Параметр кинематической вязкости ν равен 10^{-3}, диаметр цилиндра $D = 0.1$. Число Рейнольдса определяется по формуле $Re = 4\nu^{-1}D\widetilde{U}/9$. В [55] было предложено по три задачи для каждого типа цилиндра:

- Задача Z1,Q1: стационарное течение при $Re = 20$ ($\widetilde{U} = 0.45$);
- Задача Z2,Q2: нестационарное переодическое течение при $Re = 100$ ($\widetilde{U} = 2.25$);
- Задача Z3,Q3: нестационарное течение с переменным числом Рейнольдца $\widetilde{U} = 2.25\sin(\pi t/8)$.

Где Z - задача обтекания цилиндра с круглым сечением, а Q – с квадратным. Начальное условие $\mathbf{u} = 0, \mathbf{p} = 0$ при $t = 0$.

Нас интересует следующая статистика:

- Разница $\Delta p = p(\mathbf{x}_2) - p(\mathbf{x}_1)$ между значениями давления в точках $\mathbf{x}_1 = \{0.2, 0.205, 0.55\}$ and $\mathbf{x}_2 = \{0.2, 0.205, 0.45\}$.
- Коэффициент лобового сопротивления, заданный следующим интегралом по поверхности цилиндра S:

$$C_{\text{drag}} = \frac{2}{DH\widetilde{U}^2}\int\limits_S \left(\nu\frac{\partial(\mathbf{u}\cdot\mathbf{t})}{\partial\mathbf{n}}n_x - pn_z\right)ds, \tag{3.19}$$

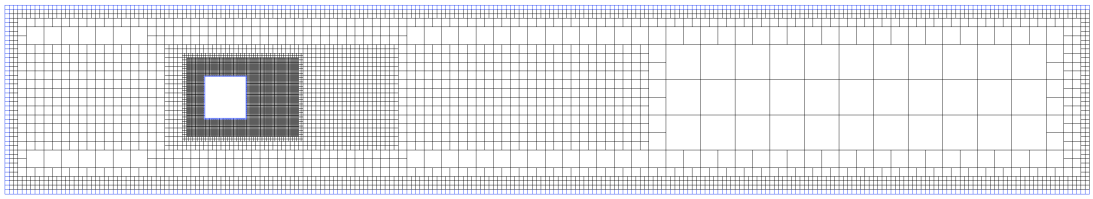

Рис. 3.18. Срез сетки $y = 0.205$ при $h_{max} = 1/32$ и $h_{min} = 1/1024$.

где $\mathbf{n} = (n_x, n_y, n_z)^T$ – вектор нормали к поверхности цилиндра, направленный внутрь Ω и $\mathbf{t} = (-n_z, 0, n_x)^T$ – тангенсальный вектор.

- Коэффициент подъемной силы, заданный следующим интегралом:

$$C_{\text{lift}} = -\frac{2}{DH\widetilde{U}^2} \int\limits_S \left(\nu \frac{\partial(\mathbf{u} \cdot \mathbf{t})}{\partial \mathbf{n}} n_z + p n_x \right) ds. \tag{3.20}$$

- Если в задачах Z2,Q2 достигнут переодический режим, тогда нас интересует число Струхаля $Df\widetilde{U}^{-1}$, где f – частота отрыва вихрей.

В задачах Z3,Q3, характеристическая скорость в C_{drag} и C_{lift} берется для момента времени $t = 4$.

Для более точного вычисления коэффициентов лобового сопротивления и подъемной силы мы заменим поверхностные интегралы в (3.19) и (3.20) на интеграл по всей области [56, 91].

Предположим, $\mathbf{u} = (u, v, w)^T$ и p являются решением (3.1). Тогда, применяя прием интегрирования по частям, получим следующие равенства:

$$\begin{aligned} C_{\text{drag}} &= \widetilde{C} \int\limits_\Omega \left[\left(\frac{\partial w}{\partial t} + (\mathbf{u} \cdot \nabla) w \right) \varphi + \nu \nabla w \cdot \nabla \varphi - p \partial_z \varphi \right] \mathrm{d}\mathbf{x} \\ C_{\text{lift}} &= \widetilde{C} \int\limits_\Omega \left[\left(\frac{\partial u}{\partial t} + (\mathbf{u} \cdot \nabla) u \right) \varphi + \nu \nabla u \cdot \nabla \varphi - p \partial_x \varphi \right] \mathrm{d}\mathbf{x}, \end{aligned} \tag{3.21}$$

для любых $\varphi \in H^1(\Omega)$, таких, что $\varphi|_S = 1$, $\varphi|_{\partial\Omega/S} = 0$. Здесь $\widetilde{C} = \frac{2}{DH\widetilde{U}^2}$.

Таблица 3.4. Количество степеней свободы для скорости и давления для различных сеток. N_{CD} и N_{PP} среднее число итераций решения уравнения конвекции-диффузии-реакции и давления задачи Q1, соответственно.

h_{min}	h_{max}	$\mathbf{u}$ d.o.f.	p d.o.f.	N_{CD}	N_{PP}
1/256	1/256	1246359	416150	23	30
1/512	1/256	1402593	467110	27	37
1/1024	1/256	2707497	897330	47	41
1/1024	1/32	1969827	645393	45	44

Если решение уравнений Навье-Стокса достаточно гладкое, тогда использование формулы интегрирования по объему (3.21) дает более точные значения коэффициентов лобового сопротивления и подъемной силы по сравнению с (3.19) и (3.20). Хотя в задаче обтекания цилиндра решение не является гладким, оказалось, что метод (3.21) дает более точные результаты.

Будем вычислять (3.21) следующим образом. Равенства (3.21) верны для любого φ, удовлетворяющего $\varphi|_S = 1$ и $\varphi|_{\partial\Omega/S} = 0$. Определим φ в центрах ячеек, и рассмотрим дискретно гармоническую функцию ($\mathrm{div}_h \nabla_h \varphi = 0$). Все производные (3.21) приблизим со вторым порядком.

Для численного решения задач Z1–Z3 и Q1–Q3 будем использовать последовательность сеток с локальным сгущением. В Таблице 3.4 дано число степеней свободы для каждой сетки для задачи Q1. Срез сетки $h_{min} = 1/32$ и $h_{max} = 1/1024$ для задачи обтекания цилиндра с квадратным сечением изображен на Рис. 3.18.

Систему линейных уравнений в задаче конвекции-диффузии-реакции на шаге (3.11) будем решать с помощью метода бисопряженных градиентов со стабилизацией и предобуславливателем неполной факторизации ILU(0). Систему (3.13) будем решать тем же методом с предобуславливателем ILU(1). Потребовавшееся

Таблица 3.5. Количество шагов по времени до достижения момента $T = 8$ в задачах Z2, Z3.

h_{min}	h_{max}	Z2, шагов по времени	Z3, шагов по времени
1/256	1/256	547	282
1/512	1/256	1031	528
1/1024	1/256	2014	1036
1/1024	1/32	2002	1033

в среднем число итераций до достижения нормы невязки линейной системы 10^{-13} показано в Таблице 3.4. Эти данные соответствуют задаче Q1.

Для задачи Q1 и Z1 мы возьмем шаг $\triangle t = 0.1$. Для задач Q3,Z3 мы возьмем шаг $\triangle t^n = \max\{0.1, 5h_{min}(\max|\mathbf{u}^n|)^{-1}\}$. Данное ограничение на шаг по времени, эквивалентное условию $CFL = 5$, необходимо нам для точности метода. Для задач Q2,Z2 мы возьмем шаг $\triangle t^n = \max\{0.1, 4h_{min}(\max|\mathbf{u}^n|)^{-1}\}$. Во всех случаях нас будет интересовать расчет до момента $T = 8$ модельного времени. В результате задачи Q1,Z1 завершаться через 80 шагов модели, количество произведенных шагов на различных сетках с локальным сгущением для задач Z2,Z3 приведено в Таблице 3.5, для задач Q2,Q3 количество шагов аналогично.

В [55] Шафер и Турек собрали данные на основе моделей, использующих самые различные методы дискретизации. На основе этих данных приведены референтные интервалы, в которые должен сходиться метод. Браак и Рихтер [56] использовали конечно-элементный метод третьего порядка и локально сгущающиеся сетки на основе апостериорной оценки ошибки, чтобы получить точные значения C_{drag}, C_{lift} и Δp для задач Q1 и Z1, к которым сходится метод. В [56] для получения сходимости были использованы сетки с общим числом степеней свободы 10^8. Мы не будем пытаться воспроизвести точные значения, а продемонстрируем, что воспроизводимые нашим методом значения находятся в некоторой

Таблица 3.6. Задача Q1: Сходимость коэффициентов тяговой и подъемной силы и изменение давления на последовательности сеток с локальным сгущением по сравнению с референтными значениями.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	Δp
1/256	1/256	7.726	0.07122	0.1717
1/512	1/256	7.683	0.06814	0.1724
1/1024	1/256	7.706	0.06829	0.1745
Braack & Richter		7.767	0.06893	0.1757
Schäfer & Turek		7.5-7.7	0.06-0.08	0.172-0.18
1/1024	1/32	7.716	0.0678	0.1749

Таблица 3.7. Задача Z1: Сходимость коэффициентов лобового сопротивления и подъемной силы и изменения давления на последовательности сеток с локальным сгущением по сравнению с референтными значениями с экстраполяцией значений в граничные условия.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	Δp
1/256	1/256	5.9707	0.00882	0.159
1/512	1/256	6.0281	0.00888	0.164
1/1024	1/256	6.0699	0.00905	0.167
Braack & Richter		6.18533	0.0094	0.171
Schäfer & Turek		6.05-6.25	0.008-0.01	0.165-0.175
1/1024	1/32	6.0715	0.00892	0.1665

Таблица 3.8. Задача Z1: Сходимость коэффициентов лобового сопротивления и подъемной силы и изменения давления на последовательности сеток с локальным сгущением по сравнению с референтными значениями с интерполяцией значений в граничные условия.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	Δp
1/256	1/256	6.1405	0.0151	0.167
1/512	1/256	6.15091	0.0141	0.169
1/1024	1/256	6.1841	0.0107	0.170
Braack & Richter		6.18533	0.0094	0.171
Schäfer & Turek		6.05-6.25	0.008-0.01	0.165-0.175
1/1024	1/32	6.1966	0.0111	0.169

окрестности возле точных референтных значений и сходятся к ним при локальном сгущении сетки. Мы так же постараемся воспроизвести значения, используя грубую сетку с пятью уровнями сгущения как к цилиндру, так и к границе области, представленной на Рис. 3.18, ($h_{max} = 1/32$, $h_{min} = 1/1024$).

Результаты решения задачи Q1,Z1 показаны в Таблицах 3.6 и 3.7, соответственно. В § 3.5 обсуждалась возможность интерполяции значений граничных условий вместо экстраполяции, что приводит к результатам, приведенным в 3.8. При экстраполяции граничных условий все референтные значения для задачи Z1 на сетке с наилучшим локальным сгущением были воспроизведены с погрешностью 3%–4%. При интерполяции значений, коэффициент тяговой силы и разность давления воспроизводится точнее, а коэффициент подъемной силы воспроизводится хуже, хотя демонстрирует хорошую сходимость к референтному значению. Примерно та же ситуация будет наблюдаться в остальных тестах. Мы остановимся на методе с интерполяцией граничных условий, так как он является более устойчивым.

Далее мы перейдем к нестационарному течению с переменным числом Рей-

Таблица 3.9. Задача Q3: Коэффициент лобового сопротивления и подъемной силы и разница давления.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	$\Delta p(T = 8)$
1/256	1/256	6.038	0.3497	-0.1461
1/512	1/256	5.178	0.0381	-0.1284
1/1024	1/256	4.655	0.0168	-0.1367
Schafer & Turek		4.3–4.5	0.01–0.05	-0.14 – -0.12
1/1024	1/32	4.658	0.0172	-0.1374

Таблица 3.10. Задача Z3: Коэффициент лобового сопротивления и подъемной силы и разница давления.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	$\min(C_{\text{lift}})$	$\Delta p(T = 8)$
1/256	1/256	3.0856	0.0316	-0.0016	-0.119
1/512	1/256	3.1760	0.00278	-0.00723	-0.112
1/1024	1/256	3.2354	0.00266	-0.00964	-0.114
Bayraktar et al.		3.2978	0.0028	-0.011	–
Schafer & Turek		3.2–3.3	0.002–0.004	–	-0.14 – -0.12
1/1024	1/32	3.2462	0.00230	-0.00950	-0.116

Таблица 3.11. Задача Q2: Коэффициэнт лобового сопротивления и подъемной силы и число Струхаля. Референтные значения для задачи Q2 являются не точными.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	St
1/256	1/256	6.204	0.07631	$*$
1/512	1/256	5.222	0.04407	0.326
1/1024	1/256	4.679	0.02697	0.297
Schafer & Turek		4.32–4.67	0.015–0.05	0.27–0.35
1/1024	1/32	4.671	0.02666	0.306

Таблица 3.12. Задача Z2: Коэффициэнт тяговой и подъемной силы и число Струхаля.

h_{min}	h_{max}	$\max(C_{\text{drag}})$	$\max(C_{\text{lift}})$	St
1/256	1/256	2.703	0.0071	0.3201
1/512	1/256	3.079	-0.0068	0.3014
1/1024	1/256	3.229	-0.0088	0.2887
Schafer & Turek		3.29-3.31	-0.011– -0.008	0.29–0.35
1/1024	1/32	3.239	-0.0087	0.2896

нольдса. Для задачи Z3 в работе Байрактар, Миерка и Турек [90] рассмотрели сходимость на основе трех методов к значениям C_{drag} и C_{lift}, где для получения значений так же были использованы сетки с общим числом степеней свободы до 10^8. Результаты расчетов для этих задач приведены в Таблицах 3.9 и 3.10 для задач Q3 и Z3 соответственно.

Задачи Q2,Z2 являются более сложными, чем четыре предыдущих. В литературе на данный момент отсутствует исследование на сходимость к результату, не зависящему от шага сетки. В [55] не дан референтный интервал для задачи Q2, поэтому мы приведем в таблице 3.11 в качестве референтных значений максимум и минимум значений, получены с помощью разных методов на самых точных

сетках. Особенность задачи обтекания цилиндра с квадратным сечением Q2 заключается в особенностях геометрии: углы цилиндра разрушают регулярность решения (3.1). В добавок, считается, что для данных задач $Re = 100$ близко к критическому числу Рейнольдса, когда происходит переход от стационарного течения в нестационарное переодическое течение. Можно определить простой критерий проверки метода: воспроизводится ли устойчивое переодическое течение, включая отрыв вихря и образование переодической вихревой дорожки Кармана.

Высокоточное вычисление коэффициентов лобового сопротивления и подъемной силы становится сложной задачей, в решении которой нам поможет локальное сгущение сетки в окрестности цилиндра. С другой стороны, сложностью в задаче Z2 является воспроизведение течения вблизи криволинейной границы, так как мы используем для аппроксимации простой метод из § 3.5. Результаты расчетов для задач Q2 и Z2 приведены в Таблицах 3.11 и 3.12 соответственно.

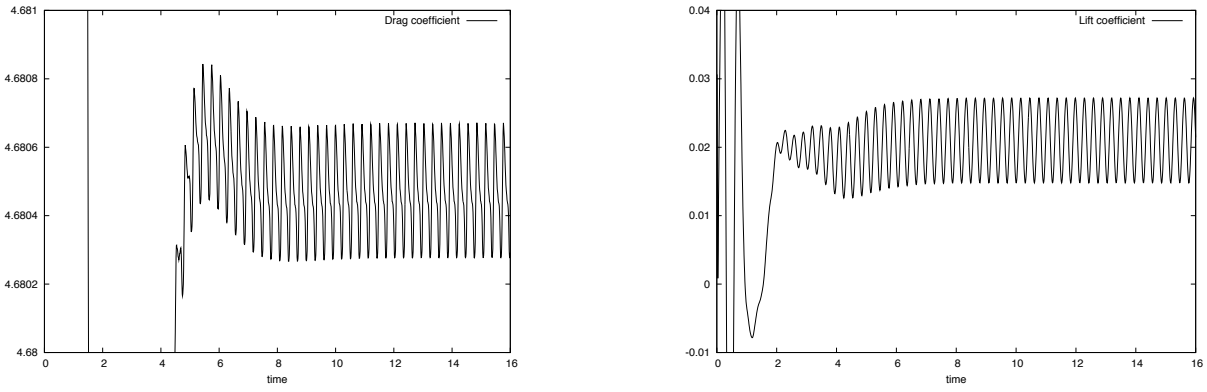

Рис. 3.19. Временная зависимость коэффициентов лобового сопротивления (слева) и подъемной силы (справа) в задаче Q2, на сетке $h_{max} = 1/32$, $h_{min} = 1/1024$.

Трехмерная структура течения в задаче Q2 при числе $Re = 100$ проиллюстрирована на Рис. 3.20 и Рис. 3.21, 3.22, где изображены линии тока, изоповерхности давления и изообъем завихренности $|\mathbf{w}| = 20$, где $\mathbf{w} = \nabla \times \mathbf{u}$, окрашенный

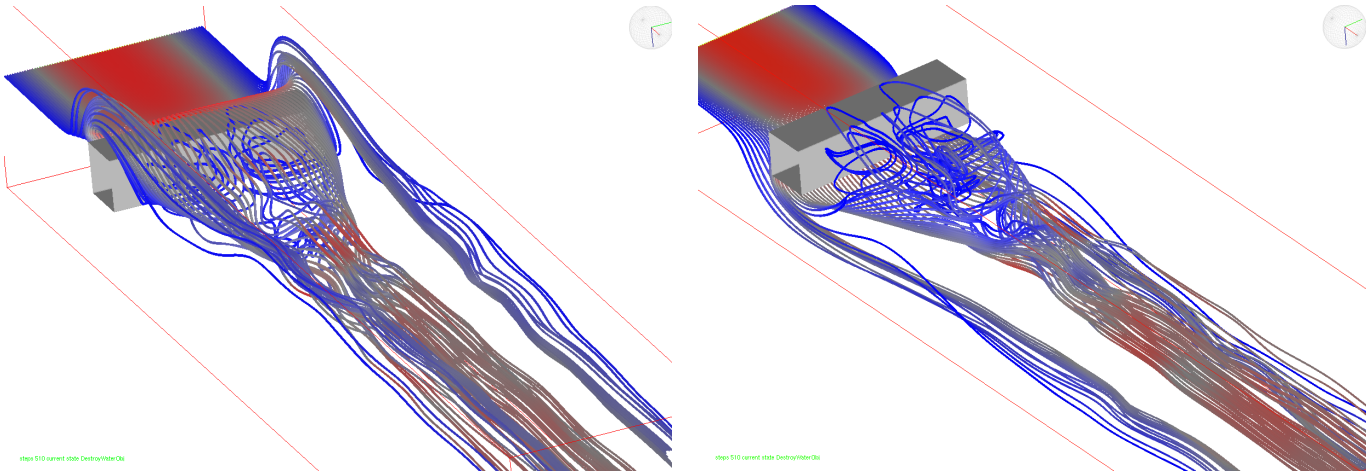

Рис. 3.20. Задача Q2 (Re=100): линии тока.

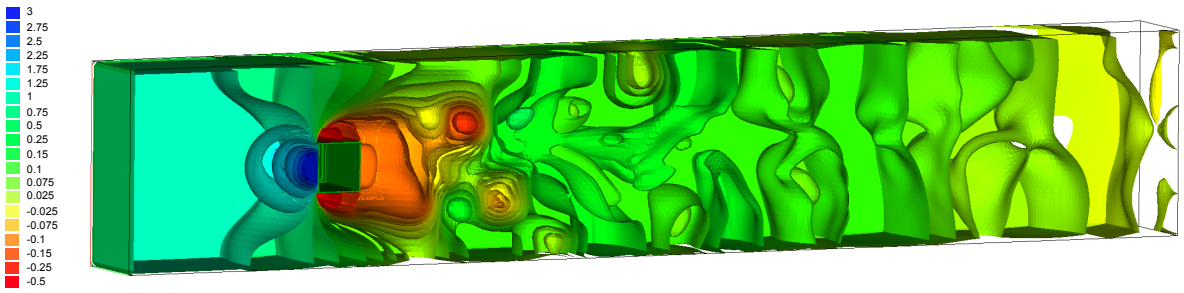

Рис. 3.21. Задача Q2 (Re=100): изоповерхности давления. На сетке $h_{\max} = 1/256$ и $h_{\min} = 1/1024$ в момент времени $t = 16$.

по значению модуля скорости, $|\mathbf{u}|$.

Выводы к третьей главе

В данной главе были предложены методы дискретизации и стабилизации уравнений Навье-Стокса на сетках типа восьмеричное дерево. На ряде численных экспериментов показана низкая численная вязкость метода. Данный метод может быть в последствии использован при расчетах на адаптивных сгущающихся сетках типа восьмеричное дерево. Для этого необходимо рассмотреть возможные критерии сгущения и разгрубления сеток и консервативные методы интерполяций скоростей и давления.

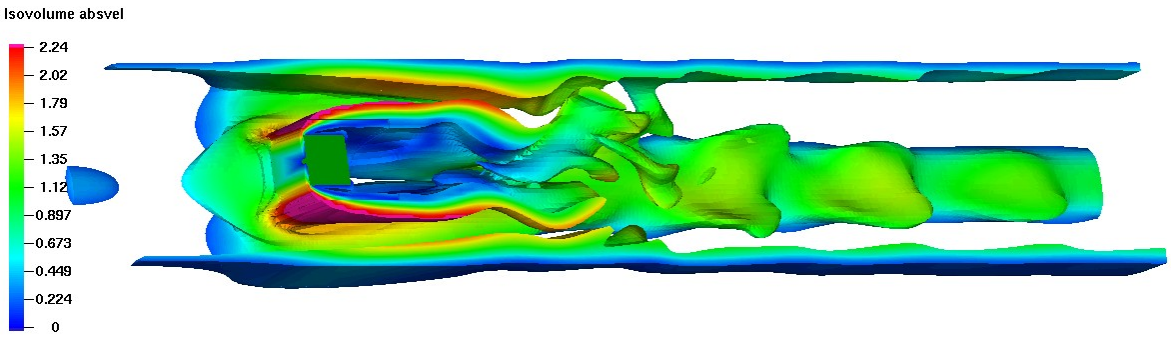

Рис. 3.22. Задача Q2 (Re=100): изообъем завихренности $|\mathbf{w}| = 20$ окрашенный по модулю скорости. На сетке $h_{\max} = 1/256$ и $h_{\min} = 1/1024$ в момент времени $t = 16$.

Заключение

Приведем основные результаты диссертации, представленные по главам.

В первой главе предложена структура данных для хранения сеток общего вида на параллельных компьютерах и ее применения для работы с адаптивными динамическими сетками типа восьмеричное дерево в последовательном режиме.

Во второй главе был предложен подход к решению задачи двухфазного заводнения с помощью полностью неявного нелинейного конечно-объемного метода на динамически адаптируемых сетках типа восьмеричное дерево. Использование нелинейных двухточечных шаблонов позволяет получать решение на сетках типа восьмеричное дерево вне зависимости от их $\mathbb{K}$-ортогональности. Предложенный критерий сгущения сеток основан на определении области с большим градиентом насыщенности воды и давления нефти. Показано, что выбранный критерий ведет к минимальной потери точности и большому выигрышу в скорости расчета.

В третьей главе предложена низкодиссипативная дискретизация уравнений Навье-Стокса для разнесенных сеток типа восьмеричное дерево. Предложен метод подавления локальных бездивергентных мод, портящих решение на стыках между мелкими и грубыми ячейками сетки. Предложенные дискретизации имеют второй порядок сходимости по времени и пространству.

Литература

1. Василевский Ю., Коньшин И., Копытов Г., Терехов К. INMOST - Программная платформа и графическая среда для разработки параллельных численных моделей на сетках общего вида. Москва: Издательство Московского Университета, 2012. С. 144.

2. R.V.Garimella. Mesh Data Structure Selection for Mesh Generation and FEA Applications // International Journal of Numerical Methods in Engineering. 2002. Vol. 55, no. 4. Pp. 451–478.

3. Garimella R. V. MSTK: A Flexible Infrastructure Library for Developing Mesh-based Applications // Proceedings of the 13th International Meshing Roundtable, Williamsburg, VA. 2004.

4. Edwards H. C., Williams A. B., Sjaardema G. D. et al. SIERRA Toolkit Computational Mesh Conceptual Model: Tech. Rep. SAND2010-1192: Sandia National Labratories, 2010.

5. Tautges T. J., Meyers R., Merkley K. et al. MOAB: A Mesh-Oriented Database: Tech. Rep. SAND2004-1592. Albuquerque, NM: Sandia National Laboratories, 2004. — April.

6. Tautges T. J. MOAB-SD: Integrated Structured and Unstructured Mesh Representation // Engineering With Computers. 2004. Vol. 20. Pp. 286–293.

7. Seol E. Flexible distributed Mesh DataBase for parallel automated adaptive analysis: Ph.D. thesis / Rensselaer Polytechnic Institute. 2005.

8. Bertsekas D. P., Tsitsiklis J. N. Parallel and Distributed Computation: Numerical Methods. Prentice-Hall, 1989. P. 730.

9. Devine K., Boman E., Heaphy R. et al. Zoltan Data Management Services for Parallel Dynamic Applications // Computing in Science and Engineering. 2002. Vol. 4, no. 2. Pp. 90–97.

10. Sheldon J., Zondek B., Cardwell W. One-dimensional incompressible, non-capillary, two-phase fluid flow in a porous medium // SPE AIME. 1959. Vol. 216. Pp. 290–296.

11. Stone H., Garder J. Analysis of gas-cap or dissolved-gas reservoirs // T. SPE AIME. 1961. Vol. 222. Pp. 92–104.

12. Ascher U., Ruuth S., Wetton T. Implicit-Explicit Methods for Time-Dependent Partial Differential Equations // SIAM J. Numer. Anal. 1995. Vol. 32, no. 3. Pp. 797–823.

13. Douglas J., Peaceman D., H.H.Rachford. A Method for Calculating Multi-Dimensional Immiscible Displacement // Am. Inst. Min. Metall. Pet. Eng. 1959. Vol. 216. Pp. 297–308.

14. Александрович С. А. Математическое моделирование процессов переноса примесей в жидкостях и пористых средах: Кандидатская диссертация / Институт математического моделирования РАН. 2009.

15. Saad M., Zhang H. Front tacking for two-phase flow in reservoir simulation by adaptive mesh // Numerical Methods for Partial Differential Equations. 1997. Vol. 13, no. 6. Pp. 673–697.

16. Furst J. A weighted least square scheme for compressible flows // Flow, Turbulence and Combustion. 2006. — september. Vol. 76, no. 4. Pp. 331–342.

17. Babuska I., Rheinboldt W. A posteriori error analysis of finite element solutions of one dimensional problems // SIAM J. Numer. Anal. 1981. Vol. 18. Pp. 565–589.

18. Bieterman M., Babuska I. The finite element method for parabolic equations, I: A posteriori error estimation, II: A posteriori error estimation and adaptive approach // Numerische Mathematik. 1982. Vol. 40. Pp. 339–371 and 373–406.

19. LePotier C. Schema volumes finis monotone pour des operateurs de diffusion fortement anisotropes sur des maillages de triangle non structures // C. C. Acad. Sci. Paris, 2005. Vol. 341. Pp. 787–792.

20. Danilov A., Vassilevski Y. A monotone nonlinear finite volume method for diffusion equations on conformal polyhedral meshes // Russ. J. Numer. Anal. Math. Modelling. 2009. Vol. 24, no. 3. Pp. 207–227.

21. Sheng Z., Yuan A. Monotone finite volume schemes for diffusion equations on polygonal meshes // J. Comput. Phys. 2008. Vol. 227. Pp. 6288–6312.

22. Nikitin K., Vassilevski Y. A monotone finite folume method for advection-diffusion equations on unstructured polyhedral meshes in 3D // Russ. J. Numer. Anal. Math. Modelling. 2010. Vol. 25, no. 4. Pp. 335–358.

23. Lipnikov K., Svyatskiy D., Vassilevski Y. Interpolation-free monotone finite volume method for diffusion equations on polygonal meshes // J. Comp. Phys. 2009. Vol. 228, no. 3. Pp. 703–716.

24. Lipnikov K., Svyatskiy D., Vassilevski Y. A monotone finite volume method for

advection-diffusion equations on unstructured polygonal meshes // J. Comp. Phys. 2010. Vol. 229. Pp. 4017 – 4032.

25. Vassilevski Y., Danilov A., Kapyrin I., Nikitin K. Application of Nonlinear Monotone Finite Volume Schemes to Advection-Diffusion Problems // Finite Volumes for Complex Applications VI Problems and Perspectives, Springer Proceedings in Mathematics. 2011. Vol. 4. Pp. 761–769.

26. Aavatsmark I., Eigestad G., Mallison B., Nordbotten J. A compact multipoint flux approximation method with improved robustness // Numer. Meth. Partial Diff. Equations. 2008. Vol. 24, no. 5. Pp. 1329–1360.

27. Klause R. A., Winther R. Convergence of Multipoint Flux Approximations on Quadrilateral Grids // Numer. Methods Partial Differential Equations. 2006. Vol. 22. Pp. 1438 – 1454.

28. Nordbotten J. M., Aavatsmark I., Eigestad G. T. Monotonicity of control volume methods // Numer. Math. 2007. Vol. 106, no. 2. Pp. 255–288.

29. Flaherty J. E., Loy R. M., Shephard M. S. et al. Adaptive Local Refinement with Octree Load Balancing for the Parallel Solution of Three-Dimensional Conservation Laws // Journal of Parallel and Distributed Computing. 1997. Vol. 47. Pp. 139–152.

30. Remacle J.-F., Flaherty J. E., Shephard M. S. An adaptive discontinuous galerkin technique with an orthogonal basis applied to compressible flow problems // SIAM Review. 2003. Vol. 45. Pp. 53–72.

31. Murman S. M. Compact upwind schemes on adaptive octrees // J. Comput. Phys. 2010. Vol. 229. Pp. 167–1180.

32. Strain J. Tree Methods for Moving Interfaces // J. Comput. Phys. 1999. Vol. 151. Pp. 616–648.

33. Sochnikov V., Efrima S. Level set calculations of the evolution of boundaries on a dynamically adaptive grid // Int. J. Numer. Meth. Engng. 2003. Vol. 56. Pp. 1913–1929.

34. Losasso F., Gibou F., Fedkiw R. Simulating water and smoke with an octree data structure // ACM Transactions on Graphics (TOG). 2004. Vol. 23.

35. Losasso F., Fedkiw R., Osher S. Spatially adaptive techniques for level set methods and incompressible flow // Computers & Fluids. 2006. Vol. 35. Pp. 995–1010.

36. Popinet S. An accurate adaptive solver for surface-tension-driven interfacial flows // J. Comput. Phys. 2009. Vol. 228. Pp. 5838–5866.

37. Fuster D., Agbaglah G., Josserand C. et al. Numerical simulation of droplets, bubbles and waves: state of the art // Fluid Dyn. Res. 2006. Vol. 41. P. 065001.

38. Nikitin K., Vassilevski Y. V. Free surface flow modelling on dynamically refined hexahedral meshes // Rus. J. Numer. Anal. Math. Model. 2008. Vol. 23. Pp. 469–485.

39. Popinet S. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries // J. Comput. Phys. 2003. Vol. 190. Pp. 572–600.

40. Min C., Gibou F. A second order accurate level set method on non-graded adaptive cartesian grids // J. Comput. Phys. 2007. Vol. 225. Pp. 300–321.

41. Gibou F., Min C., Ceniceros H. Finite Difference Schemes for Incompressible Flows

on Fully Adaptive Grids // International Series of Numerical Mathematics. 2006. Vol. 154. Pp. 199–208.

42. Лебедев В. И. Разностные аналоги ортогональных разложений, основных дифференциальных операторов и некоторых краевых задач математической физики // ЖВМиМФ. 1964. Т. 4, № 3. С. 449–465.

43. Лебедев В. И. Разностные аналоги ортогональных разложений, основных дифференциальных операторов и некоторых краевых задач математической физики // ЖВМиМФ. 1964. Т. 4, № 4. С. 649–659.

44. Harlow F., Welch J. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface // Phys. Fluids. 1965. Vol. 8. Pp. 2182–2189.

45. Chorin A. Numerical solution of the Navier-Stokes equations // Math. Comp. 1968. Vol. 22. Pp. 745–762.

46. Gresho P. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. I - Theory // Int. J. Numer. Meth. Fluids. 1990. Vol. 11. Pp. 587–620.

47. Oseen F. Neuere Methoden und Ergebnisse in der Hydrodynamik. Leipzig: Akademische Velagsgesellschaft, 1927.

48. Olshanskii M., Vassilevski Y. Pressure Schur complement preconditioners for the discrete Oseen problem // SIAM J. Sci. Comput. 2007. Vol. 29, no. 6. Pp. 2686–2704.

49. Ascher U. M., Petzold L. R. Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. Philadelphia: SIAM, 1998.

50. Courant R., Friedrichs K., Lewy H. Über die partiellen Differenzengleichungen der mathematischen Physik // Mathematische Annalen. 1928. Vol. 100. Pp. 32–74.

51. Ethier C., Steinman D. Exact fully 3d Navier-Stokes solutions for benchmarking // Int. J. Numer. Meth. Fluids. 1994. Vol. 19. Pp. 369–375.

52. Shankar P. N., Deshpande M. D. Fluid mechanics in the driven cavity // Annu. Rev. Fluid Mech. 2000. Vol. 32. Pp. 93–136.

53. Zunic Z., Hribersek M., Skerget L., Ravnik J. 3D driven cavity flow by mixed boundary and finite element method // European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006 / Ed. by E. O. P. Wesseling, J. Periaux. 2006.

54. K.L.Wong, A.J.Baker. A 3D incompressible Navier-Stokes velocity-vorticity weak form finite element algorithm // Int. J. Numer. Meth. Fluids. 2002. Vol. 38. Pp. 99–123.

55. Schäfer M., Turek S. Benchmark computations of laminar flow around a cylinder // Notes numer fluid mech. 1996. Vol. 52. Pp. 547–566.

56. Braack M., Richter T. Solutions of 3D Navier–Stokes benchmark problems with adaptive finite elements // Computers & Fluids. 2006. Vol. 35. Pp. 372–392.

57. Nikitin K. D., Olshanskii M. A., Terekhov K. M., Vassilevski Y. V. A numerical method for the simulation of free surface flows of viscoplastic fluid in 3D // J. Comput. Math. 2011. Vol. 29. Pp. 605–622.

58. Nikitin K., Olshanskii M., Terekhov K., Vassilevski Y. Numerical modelling of viscoplastic free surface flows in complex 3D geometries // Proceedings of Euro-

pean Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2012. Vienna, Austria: September 10-12, 2012.

59. Vassilevski Y., Nikitin K., Olshanskii M., Terekhov K. CFD technology for 3D simulation of large-scale hydrodynamic events and disasters // Rus. J. Numer. Anal. Math. Model. 2012. Vol. 27, no. 4. Pp. 399–412.

60. Terekhov K., Volodin E., Gusev A. Methods and efficiency estimation of parallel implementation of the sigma-model of general ocean circulation // Rus. J. Numer. Anal. Math. Model. 2011. Vol. 26, no. 2. Pp. 189–208.

61. Terekhov K., Vassilevski Y. Two-phase water flooding simulations on dynamic adaptive octree grids with two-point nonlinear fluxes // Rus. J. Numer. Anal. Math. Model. 2013. Vol. 28, no. 3. Pp. 267–288.

62. Nikitin K., Terekhov K., Vassilevsky Y. A monotone nonlinear finite volume method for diffusion equations and multiphase flows // Geoscience. 2013. Vol. ???, no. ??? Pp. ???–???

63. Olshanskii M., Terekhov K., Vassilevski Y. An octree-based solver for the incompressible Navier-Stokes equations with enhanced stability and low dissipation. // Computers and Fluids. 2013. Vol. ???, no. ??? Pp. ???–???

64. Никитин К. Д., Сулейманов А. Ф., Терехов К. М. Технология моделирования течений со свободной поверхностью в реалистичных сценах // Труды Математического центра им. Н.И. Лобачевского. 2009. Т. 39. С. 305–307.

65. Nikitin K. D., Olshanskii M. A., Terekhov K. M., Vassilevski Y. V. Preserving distance property of level set function and simulation of free surface flows

on adaptive grids // Численная геометрия, построение расчетных сеток и высокопроизводительные вычисления. 2010. Pp. 25–32.

66. Терехов К. Параллельная реализация модели общей циркуляции океана // Сборник тезисов лучших дипломных работ 2010. ВМИК МГУ, Москва: МАКС ПРЕСС, 2010.

67. Nikitin K., Olshanskii M., Suleimanov A. et al. Free surface flow modelling with dynamically refined octree meshes // Abstracts of international conference, CMAM-4. Bedlow, Poland: 2010.

68. Olshanskii M., K.M. Terekhov Y. V. An octree-based solver for the incompressible Navier-Stokes equations. // Proceedings of the Second China-Russia Conference on Numerical Algebra with Applications. Rostov-on-Don: 2013. Pp. ???–???

69. Aziz K., Settari A. Petroleum Reservoir Simulation. London: Applied Sci. Publ. Ltd, 1979.

70. Peaceman D. W. Fundamentals of Numerical Reservoir Simulation. New York: Elsevier, 1977.

71. Peaceman D. W. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation // SPEJ. 1978. Pp. 183–194.

72. Nikitin K., Vassilevski Y. Free surface flow modelling on dynamically refined hexahedral meshes // Russ. J. Numer. Anal. Math. Modelling. 2008. Vol. 23, no. 5. Pp. 469–485.

73. Doug E., Nguyen D., Gibou F., Fedkiw R. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows // Pro-

ceedings of EFEDSM 2003 4th ASME JSME Joint Fluids Engineering Conference. Honolulu, Hawaii, USA: July 6-11, 2003.

74. Leonard B. A stable and accurate convective modelling procedure based on quadratic upstream interpolation // Computer Methods in Applied Mechanics and Engineering. Vol. 19, no. 1. Pp. 59–98.

75. Lilly D. K. On the Computational Stability of Numerical Solutions of Time-Dependent Non-Linear Geophysical Fluid Dynamics Problems // Monthly Weather Review. 1965. Vol. 93, no. 1. Pp. 11–26.

76. van't Hof B., Veldman A. E. Mass, momentum and energy conserving (MaMEC) discretizations on general grids for the compressible Euler and shallow water equations // J. Comp. Phys. 2012. Vol. 231. Pp. 4723–4744.

77. Ham F., Lien F., Strong A. A Fully Conservative Second-Order Finite Difference Scheme for Incompressible Flow on Nonuniform Grids // J. Comp. Phys. 2002. Vol. 177. Pp. 117–133.

78. Morinishi Y., Lund T., Vasilyev O., P.Moin. Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow // J. Comp. Phys. 1998. Vol. 143. Pp. 90–124.

79. Droge M., Verstappen R. A new symmetry-preserving Cartesian-grid method for computing flow past arbitrary shaped objects // Int. J. Number. Meth. Fluids. 2005. Vol. 47. Pp. 979–985.

80. Arakawa A. Computational Design for Long-Term Numerical Integration of the Equations of Fluid Motion: Two-Dimensional Incompressible Flow. Part I // J.Comp.Phys. 1997. Vol. 135. Pp. 103–114.

81. McLachlan R. I. Spatial Discretization Of Partial Differential Equations With Integrals. 1998.

82. Mullen P., Crane K., Pavlov D. et al. Energy-preserving integrators for fluid animation // ACM Trans. Graph. 2009. Vol. 28, no. 3. Pp. 38:1–38:8.

83. Brown D. L., Cortez R., Minion M. L. Accurate Projection Methods for the Incompressible Navier-Stokes Equations // J. Comp. Phys. 2001. no. 168. Pp. 464–499.

84. Guermond J. L., Minev P., Shen J. An overview of projection methods for incompressible flows // Comput. Meth. Appl. Mech. Engrn. 2006. Vol. 195. Pp. 6011–6045.

85. Sun H., He Y., Feng X. On Error Estimates of the Pressure-Correction Projection Methods For The Time-Dependent Navier-Stokes Equations // Int. J. Numer. Anal. and Model. 2011. Vol. 8, no. 1. Pp. 70–85.

86. Guermond J. L., Minev P., Shen J. Error Analysis of Pressure-Correction Schemes for the Time-Dependent Stokes Equations with Open Boundary Conditions // SIAM J. Numer. Anal. 2005. Vol. 42. P. 239–258.

87. Angot P., Cheaytou R. Vector penalty-projection method for incompressible fluid flows with open boundary conditions // Proceedings of 19th Conference on Scientific Computing, Algoritmy. 2012. Pp. 219–229.

88. Fadlun E., Verzicco R., Orlandi P., Mohd-Yusof J. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations // J.Comp.Phys. 2000. Vol. 161. Pp. 30–60.

89. Tseng Y.-H., Ferziger J. H. A ghost-cell immersed boundary method for flow in complex geometry // J. Comp. Phys. 2003. Vol. 192. Pp. 593–623.

90. Bayraktar E., Mierka O., Turek S. Benchmark computations of 3D laminar flow around a cylinder with CFX, OpenFOAM and FeatFlow // Int. J. of Comp. Sc. and Eng. 2012. Vol. 7. Pp. 253–266.

91. John V. Higher order finite element methods and multigrid solvers in a benchmark problem for 3D Navier-Stokes equations // Int. J. Numer. Meth. Fluids. 2002. Vol. 40. Pp. 775–98.