Учреждение Российской академии наук
Институт вычислительной математики РАН

На правах рукописи

Данилов Александр Анатольевич

Технология построения неструктурированных сеток и монотонная дискретизация уравнения диффузии

05.13.18 – Математическое моделирование,
численные методы и комплексы программ

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата физико-математических наук

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

Москва – 2010

Содержание

Введение 4

Обзор методов построения сеток и комплексов программ 10

Глава 1. Построение двумерных треугольных сеток 20

1.1. Обозначения 20

1.2. Алгоритм продвигаемого фронта 24

1.3. Влияние вычислительных погрешностей 27

1.4. Конечность работы алгоритма 33

1.5. Скорость работы алгоритма продвигаемого фронта 40

1.6. Улучшение качества полученной сетки 49

1.7. Результаты экспериментов 54

1.8. Выводы к первой главе 59

Глава 2. Построение поверхностных сеток 60

2.1. Представление поверхности 61

2.2. Взаимодействие с геометрическим ядром САПР 64

2.3. Алгоритм продвигаемого фронта 67

2.4. Результаты экспериментов 71

2.5. Выводы к второй главе 73

Глава 3. Построение трёхмерных сеток 74

3.1. Алгоритм продвигаемого фронта 76

3.2. Устойчивый метод на основе тетраэдризации Делоне 84

3.3. Улучшение качества полученной сетки 90

3.4. Результаты экспериментов 92

3.5. Выводы к третьей главе 98

Глава 4. Сетки с многогранными ячейками и монотонная дискретизация уравнения диффузии 100
4.1. Уравнение диффузии 103
4.2. Монотонная нелинейная схема на основе метода конечных объёмов 104
4.3. Схема дискретизации и анализ её монотонности 114
4.4. Результаты экспериментов 117
4.5. Моделирование стационарного распределения интерферона в лимфатическом узле 128
4.6. Выводы к четвёртой главе 139

Заключение 140

Литература 141

Введение

При решении современных научных и инженерных задач широко применяется математическое моделирование. Существуют готовые системы автоматизации инженерных расчётов (CAE – Computer-aided engineering) для анализа и моделирования физических процессов на ЭВМ. Как правило, это большие закрытые коммерческие проекты, разрабатывающиеся на протяжении десятков лет группами в несколько сотен специалистов. С точки зрения пользователя системы CAE предоставляют удобный графический интерфейс для задания параметров и условий задачи, проведения расчётов, отображения и анализа результатов. Алгоритмы и методы, используемые для решения конкретной задачи, зачастую скрыты от пользователя.

Как правило, процесс моделирования физических процессов с использованием ЭВМ сводится к трём основным операциям: построение расчётных сеток, дискретизация дифференциального уравнения, решение системы алгебраических уравнений. В диссертационной работе мы детально рассмотрим проблемы построения сеток. Различные методы дискретизаций представлены в работах [54, 55, 58]. Методы решения алгебраических уравнений рассматриваются в работах [53, 56, 60].

Большинство современных комплексов CAE состоит из набора библиотек для графического интерфейса, визуализации, построения расчётных сеток, дискретизации дифференциальных уравнений и решения систем линейных и нелинейных уравнений. Разработку большинства низкоуровневых библиотек можно вынести за пределы всей системы CAE, что позволяет значительно упростить внедрение новых методов и алгоритмов для решения отдельных подзадач.

В настоящее время большинство таких библиотек доступны отдельно от систем CAE. Так, при разработке новых схем дискретизации исследователь

может использовать уже готовые библиотеки для построения расчётных сеток, для решения систем линейных уравнений и для визуализации результатов. Важными критериями при выборе конкретных библиотек с точки зрения разработчика являются надёжность, простота и стоимость или возможность свободного использования.

Автор диссертационной работы входит в коллектив исследователей, разрабатывающих пакет библиотек Ani3D с *открытым исходным кодом* – программный код библиотеки доступен для просмотра, изучения и изменения. В этот пакет входят библиотеки для построения и адаптации неструктурированных сеток, для построения дискретизаций на неструктурированных сетках, для решения систем линейных и нелинейных алгебраических уравнений. Также развивается и "двумерная" версия пакета – Ani2D. Программный код обеих библиотек распространяется свободно.

Автор работы является основным разработчиком библиотек для построения неструктурированных сеток. Конечной целью "трёхмерной" библиотеки является создание технологии надёжного построения гибридных сеток с многогранными ячейками для произвольных областей. В рамках диссертационной работы автором была разработана технология надёжного построения неструктурированных *тетраэдральных* сеток, которая уже включена в пакет Ani3D. Также автором диссертационной работы была разработана новая монотонная нелинейная схема дискретизации уравнения диффузии на неструктурированных сетках с многогранными ячейками.

Актуальность темы. Для решения прикладных трёхмерных задач в сложных областях возникает необходимость создания технологии построения расчётных сеток, методов дискретизации дифференциальных уравнений на них и способов решения полученных систем алгебраических уравнений. Задачам построения расчётных сеток для сложных геометрических областей уделяется большое внимание. Существующие методы построения тетраэдраль-

ных, призматических, гексаэдральных и многогранных сеток, как правило, требуют ручного вмешательства пользователя для получения результата, или применимы только для узкого класса геометрических областей. Перспективным направлением видится разработка методов построения гибридных сеток, состоящих из многогранных ячеек.

Дискретизация уравнений математической физики на сетках с многогранными ячейками является отдельной задачей. Во многих прикладных задачах важно соблюдение определённых физических свойств решения, например, неотрицательности концентрации в задачах диффузии. В последнее время особый интерес привлекают монотонные консервативные схемы дискретизации уравнений диффузии для сред с анизотропными свойствами.

Цель диссертационной работы. Целью диссертационной работы является разработка технологической цепочки для приближённого решения трёхмерной задачи диффузии в сложных областях, включающая создание технологии надёжного построения неструктурированных треугольных и тетраэдральных сеток и разработку новой монотонной нелинейной схемы дискретизации для трёхмерного численного моделирования диффузионных процессов на сетках с многогранными ячейками.

Научная новизна. В работе предложена технология автоматического надёжного построения тетраэдральных сеток для сложных областей на основе метода продвигаемого фронта. Проведён анализ влияния вычислительных погрешностей на алгоритмы при их реализации на ЭВМ, представлено теоретическое обоснование конечности работы предложенных алгоритмов. Предложена и исследована монотонная нелинейная схема на основе метода конечных объёмов для уравнения диффузии на неструктурированных сетках с многогранными ячейками.

Практическая значимость. Практическая значимость диссертационной работы заключается в создании и поддержке комплекса программ для

автоматического построения тетраэдральных сеток и для приближённого решения задач диффузии на многогранных сетках. Программы для построения поверхностных треугольных и тетраэдральных сеток включены в библиотеку программ Ani3D и находятся в свободном доступе. Решена практическая задача о стационарном распределении интерферона в лимфоузле.

На защиту выносятся следующие основные результаты:

1. Разработана технология надёжного построения неструктурированных тетраэдральных сеток, реализованы разные способы задания области, в том числе с помощью САПР.

2. Предложена и исследована новая монотонная нелинейная схема дискретизации уравнения диффузии на основе метода конечных объёмов на сетках с многогранными ячейками.

3. На основе предложенных методов разработана модель стационарного распределения интерферона в лимфоузле с учётом геометрических особенностей лимфоузла.

Апробация работы. Результаты диссертационной работы докладывались автором и обсуждались на научных семинарах Института вычислительной математики РАН, Института математического моделирования РАН, Вычислительного центра РАН, Механико-математического факультета МГУ им. М. В. Ломоносова, в Институте прикладной математики и информационных технологий, г. Павия (Италия), в Лос-Аламосской национальной лаборатории, г. Лос-Аламос (США) и на следующих научных конференциях: конференции “Тихоновские чтения” (МГУ, Москва, ноябрь 2007, октябрь 2009); конференции “Ломоносов” (МГУ, Москва, апрель 2008, апрель 2010); конференции “Ломоносовские чтения” (МГУ, Москва, апрель 2009, апрель 2010); конференция молодых учёных “Технологии высокопроизводительных вычислений и

компьютерного моделирования" (СПбГУ ИТМО, С.-Петербург, апрель 2009); конференция "Лобачевские чтения" (КГУ, Казань, ноябрь 2009); международные конференции "NUMGRID-2006" и "NUMGRID-2008" (ВЦ РАН, Москва, июнь 2006, июнь 2008); международная конференция "SIAM Geosciences 2009" (Лейпциг, Германия, июнь 2009); международный научный семинар "Computational Mathematics and Applications" (Технологический университет Тампере, Тампере, Финляндия, сентябрь 2009).

Публикации. Основные материалы диссертации опубликованы в 6 печатных работах, из них 3 статьи в рецензируемых журналах, входящих в перечень ВАК, [8, 9, 49].

Личный вклад автора. В совместной работе [9] вклад автора заключался в разработке нелинейной схемы дискретизации уравнения диффузии в трёхмерном пространстве, в программной реализации метода и в постановке численных экспериментов.

В совместной работе [43] вклад автора заключался в разработке методов построения треугольных и тетраэдральных неструктурированных сеток.

В совместной работе [44] вклад автора заключался в разработке методов взаимодействия с САПР при построении сеток, в программной реализации алгоритмов и проведении численных экспериментов.

Структура и объём диссертации. Диссертационная работа состоит из введения, обзора методов построения сеток и комплексов программ, четырёх глав, заключения и списка литературы из 73 наименований. Диссертационная работа содержит 32 рисунка и 15 таблиц. Общий объём диссертационной работы – 148 страниц.

Благодарности

Автор диссертационной работы выражает глубокую признательность научному руководителю Ю. В. Василевскому за продолжительную поддержку, ценные советы и плодотворное обсуждение вопросов. Автор благодарен Р. Гаримелле из Лос-Аламосской национальной лаборатории за идею использования библиотеки CGM в качестве интерфейса между генератором сеток и геометрическим ядром САПР и за возможность использования при программной реализации новой монотонной схемы на основе метода конечных объёмов библиотеки MSTK для работы с многогранными сетками. Автор также выражает благодарность К. Д. Никитину, И. В. Капырину, К. Н. Липникову, Д. А. Святскому и многим другим за помощь в обсуждении идей и методов. Особую благодарность автор выражает Г. А. Бочарову за постановку задачи о стационарном распределении интерферона в лимфоузле, за подбор необходимых параметров и за помощь в интерпретации полученных результатов.

Работа над диссертацией была частично поддержана грантами РФФИ 08-01-00159-а, 09-01-00115-а, программой Президиума РАН “Фундаментальные науки – медицине”, федеральной целевой программой “Научные и научно-педагогические кадры инновационной России”, а также грантом Upstream Research Center, ExxonMobil corp.

Обзор методов построения сеток и комплексов программ

В диссертационной работе будем рассматривать только конформные сетки, любые два элемента которых либо не имеют общих точек, либо имеют ровно одну общую вершину, либо одно общее ребро, либо одну общую грань. Важным примером неконформных сеток могут служить сетки типа восьмеричное дерево (см. Рис. 1),тем не менее их можно рассматривать как конформные сетки с многогранными ячейками.

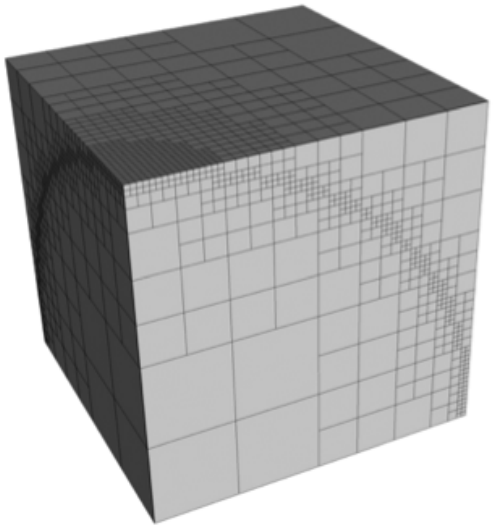


Рис. 1. Сетка типа восьмеричное дерево.

На плоскости можно выделить два класса сеток – треугольные и четырёхугольные. В трёхмерном пространстве выделим три класса сеток – тетраэдральные, треугольные призматические, и гексаэдральные. При фиксированном количестве вершин N_v сетки выполняются следующие оценки на количество элементов, N_e, и граней, N_f, в сетке. Для неструктурированных тетраэдральных сеток, используемых в приложениях, $N_e \approx 5.5N_V$, $N_f \approx 11N_V$. Для треугольных призматических сеток $N_e \approx 2N_V$, $N_f \approx 5N_V$, для гексаэдральных $N_e \approx N_V$, $N_f \approx 3N_V$.

С точки зрения количества элементов и граней гексаэдральные сетки являются наиболее удобными, так как получающиеся при дискретизации на них матрицы будут иметь меньше ненулевых значений. С точки зрения автоматического построения сеток для сложных областей, гексаэдральные сетки являются самыми неудобными. Существующие на сегодняшний день методы построения сеток, такие как метод отображений, метод продвигаемого фронта, методы заметания (sweeping) и замощения (paving) области, методы на основе восьмеричных деревьев (octree), применимы либо для достаточно узкого класса областей, либо требуют ручного вмешательства в процесс построения сетки.

Призматические сетки широко применяются в геофизических приложениях в задачах со слоистыми областями. В таких областях призматическую сетку можно представить как тензорное произведение неструктурированной треугольной сетки в плоскости OXY и одномерной сетки вдоль OZ. В Институте вычислительной математики РАН В. Н. Чугуновым разработан генератор призматических сеток [63]. Для построения начальной треугольной сетки используется разработанный автором диссертационной работы генератор треугольных сеток из библиотеки программ Ani2D. В будущем планируется включение этого призматического генератора в состав Ani3D.

Тетраэдральные сетки содержат большое количество ячеек и граней, что увеличивает количество ненулевых значений в матрице линеаризованной системы уравнений. С другой стороны, с помощью тетраэдральных сеток можно строить расчётные сетки в сколь угодно сложных областях. В этой диссертационной работе мы подробно остановимся именно на технологии построения тетраэдральных сеток.

Перспективным направлением видится создание автоматической надёжной технологии построения гибридных сеток для сложных областей, использующей в качестве ячеек гексаэдры, призмы, пирамиды и тетраэдры. Пред-

ставленная в диссертационной работе технология надёжного построения треугольных и тетраэдральных сеток является первым шагом в решении этой задачи.

Параллельно с разработкой методов построения гибридных сеток необходима разработка схем дискретизации на новых сетках. В диссертационной работе будет предложена новая монотонная схема дискретизации уравнения диффузии на основе метода конечных объёмов на сетках с многогранными ячейками.

Проблемам построения сеток уделяется большое внимание, постоянно выходят научные публикации, книги, сборники [37], проводятся ежегодные конференции. Наиболее известные международные конференции, посвящённые проблемам построения сеток, проходят в США – International Meshing Roundtable, и в России – "Численная геометрия, построение сеток и высокопроизводительные вычисления" (NUMGRID), результаты работ которой публикуются в специальных выпусках Журнала вычислительной математики и математической физики. Сделаем краткий обзор основных методов и наиболее известных комплексов программ для построения треугольных и тетраэдральных сеток. Краткий обзор имеющихся схем дискретизации уравнения диффузии будет сделан в Главе 4.

Построение треугольных сеток на плоскости

Существует два основных метода построения треугольной сетки на плоскости: 1) метод продвигаемого фронта [27, 30]; 2) метод построения триангуляции Делоне [2, 59]. Идея первого метода будет раскрыта в Главе 1, подробности использования второго метода хорошо представлены в [59]. Отметим, что оба метода могут быть использованы для построения конформной треугольной сетки внутри заданного многоугольника на заданном наборе вершин без

добавления новых вершин. На практике сетки строятся с добавлением новых вершин внутри области, и качество и свойства сетки зависят от способа выбора положения этих новых вершин [2, 27].

В ходе реализации и тестирования метода продвигаемого фронта в составе пакетов программ Ani2D и Ani3D автором был разработан набор эвристических алгоритмов для выбора положения новых вершин с целью увеличения надёжности и качества полученных сеток при сохранении простоты самого алгоритма. В Главе 1 проведён анализ влияния вычислительных погрешностей, доказана конечность работы алгоритма, и сделаны оценки на время работы алгоритма.

Отметим основные особенности комплекса программ для построения сеток, разработанного автором диссертационной работы и включённого в пакеты Ani2D и Ani3D. Граница задаётся либо дискретно, либо аналитически с помощью параметризующих функций, предоставленных пользователем; есть возможность контролировать желаемый размер треугольников во всей области, размер треугольников может выбираться автоматически на основе заданной дискретизации границы и заданной скорости увеличения размера треугольников при продвижении внутрь области. При использовании одинарной точности для хранения координат в памяти ЭВМ для корректной работы генератора достаточна двойная точность вычислений внутри генератора. Программа собирается в виде библиотеки, приведены примеры её использования. Исходный код открыт и свободно распространяется [65, 66].

Существует множество программных реализаций для построения треугольных сеток. Отметим один из наиболее известных надёжных генераторов треугольных сеток, генератор сеток Triangle, разработанный Дж. Р. Шевчуком [33]. Этот генератор сеток основан на методе построения триангуляции Делоне, и позволяет строить конформные триангуляции Делоне, триангуляции Делоне с ограничениями, и треугольные сетки с высоким качеством [36]. Гра-

ница области задаётся дискретно, нет возможности полностью контролировать желаемый размер элементов внутри области. Генератор может использовать арифметику с произвольной точностью для исключения ошибок округления при вычислениях [34], например, при вычислении знака определителя 3×3. Код программы может быть скомпилирован как готовый исполняемый файл, или собран в виде библиотеки для дальнейшего подключения к другим программам. Исходный код генератора открыт и свободно распространяется [64].

Построение тетраэдральных сеток в пространстве

Обычно процесс построения тетраэдральной сетки делится на два этапа: 1) построение поверхностной триангуляции на границе области; 2) построение тетраэдральной сетки внутри области.

В разных задачах геометрические формы моделей имеют разную сложность. Например, в случае моделирования физических процессов в однородной среде можно использовать область простой формы в виде шара или куба. В задачах обтекания используются более сложные модели, например, пересечение или объединение нескольких простых фигур. В инженерных задачах широко применяются системы автоматизации проектных работ (САПР) для создания геометрических моделей изделий. Не исключено и применение трёхмерного сканирования для получения компьютерной модели реально существующего объекта, а также использование картографических данных для создания моделей реального рельефа, например, в геофизических задачах. Универсального подхода не существует: в разных задачах удобен тот или иной способ задания геометрии модели.

В пакете программ Ani3D предусмотрен богатый выбор способов построения треугольных поверхностных сеток [49], включающий библиотеку для

улучшения качества имеющихся поверхностных триангуляций, разработанную совместно с А. В. Плёнкиным и А. В. Вершининым [43], и библиотеку для задания области на основе подходов конструктивной блочной геометрии, разработанную К. Д. Никитиным [57]. В диссертационной работе будет рассмотрен только способ построения треугольных поверхностных сеток для областей, заданных аналитически, или с помощью САПР [44]. Подробнее об этих методах будет рассказано в Главе 2.

При построении объёмной тетраэдральной сетки важным свойством метода является возможность сохранения заданного следа сетки на границе. В отличие от двумерных треугольных сеток, в трёхмерном случае существуют такие многогранные области, для которых невозможно построить конформную тетраэдральную сетку с заданным следом на границе без добавления новых вершин. Пример такой области приведён в [32], к нему мы вернёмся в разделе 3.1.2. Б. Джо в своей работе [17] показал, что для выпуклых многогранников добавление одной вершины позволяет построить конформную согласованную с границей сетку. П. Л. Джордж показал в [14] существование конформной согласованной сетки для произвольного многогранника при добавлении конечного числа вершин. Тем не менее задача построения сетки, согласованной с заданной границей, остаётся сложной практической задачей.

При построении тетраэдральных сеток могут использоваться методы, основанные на тех же идеях, что и в двумерном случае. Классический метод построения тетраэдризации Делоне строит сетку для выпуклой оболочки точек и не гарантирует сохранение границы области. Предложено несколько методов для сохранения границы: локальные модификации сетки [4], измельчение сетки [10], построение тетраэдризации Делоне с ограничениями (CDT – Constrained Delaunay triangulations) [35]. Алгоритм продвигаемого фронта, с другой стороны, не испытывает проблем с сохранением заданной границы, так как с неё и начинает [16]. Однако, этот алгоритм может столкнуться с

такой конфигурацией фронта, продвинуть которую он не в состоянии. В литературе встречаются и гибридные методы, основанные на комбинировании идей методов продвигаемого фронта и тетраэдризации Делоне [40].

В диссертационной работе предложена комбинация двух методов. Основной используемый метод – метод продвигаемого фронта, с его помощью строится бóльшая часть сетки. В качестве дополнительного метода используется упрощённая версия метода, предложенного П. Л. Джорджем в [15]. Подробнее об этих двух методах будет рассказано в Главе 3.

Разработанный и внедрённый в пакет Ani3D комплекс программ позволяет автоматически надёжно конформно строить тетраэдральные сетки внутри заданных многогранных областей с заданным следом сетки на границе. Как и в двумерном случае, есть возможность контролировать желаемый размер ячеек сетки во всей области. В случае отсутствия у пользователя информации о желаемом размере, размер может определяться автоматически на основании заданной дискретной триангуляции границы области и заданной пользователем скорости увеличения размера элементов при продвижении внутрь области. На первом этапе при построении большей части сетки методом продвигаемого фронта гарантируется корректная конечная работа генератора при неточных вычислениях чисел с плавающей точкой. Использование неточной арифметики уменьшает часть построенной области по сравнению с точной арифметикой. Тем не менее на практике больше 99% области разбивается с помощью алгоритма продвигаемого фронта. Устойчивый метод на основе тетраэдризации Делоне второго этапа может построить тетраэдры, которые из-за накопления вычислительных погрешностей могут быть вырожденными или вывернутыми. На третьем этапе качество ячеек сетки улучшается за счёт локальных операций модификации сетки. Для этого используется библиотека aniMBA из пакета Ani3D, разработанная К. Н. Липниковым и Ю. В. Василевским. Исходный код библиотек пакета Ani3D открыт и свобод-

но распространяется.

Перейдём к краткому обзору нескольких наиболее известных комплексов программ для автоматического построения неструктурированных тетраэдральных сеток. Рассмотрим следующие генераторы: TetGen [70], TetMesh (GHS3D) [71], NETGEN [68], CUBIT [67].

Пакет TetGen (основной разработчик – Х. Си) строит тетраэдральные сетки для произвольных многогранных областей, для этого используется метод построения и измельчения тетраэдризации Делоне. Граница области задаётся дискретно. При некоторых существенных ограничениях на поверхностную триангуляцию гарантируется возможность построения сетки, согласованной с заданной границей [12]. Желаемый размер ячеек сетки может полностью контролироваться пользователем. Программа может быть скомпилирована в виде исполняемого файла генератора или в виде библиотеки подключена к другим программам. Исходный код открыт и свободно распространяется.

Коммерческий пакет TetMesh, ранее известный как GHS3D, основным автором которого является П. Л. Джордж, включён в ряд CAE систем, таких как ANSYS и MSC.Nastran. Используемый метод основан на построении тетраэдризации Делоне. Граница области задаётся дискретно, гарантируется построение сетки, согласованной с заданной границей. Шаг сетки выбирается автоматически, также есть возможность его контролировать. Для построения начальных поверхностных триангуляций используются отдельные комплексы программ. Исходный код пакета закрыт. Предоставляются разовые лицензии для ознакомления.

Следующий пакет, NETGEN (автор – И. Шоберл), представляет собой готовый комплекс программ для автоматического построения треугольных поверхностных сеток и тетраэдральных сеток. Граница области задаётся с помощью методов конструктивной блочной геометрии, с помощью сплайнов

в формате STEP и IGES, или дискретно в формате STL. Построение сетки сводится к построению поверхностной сетки с помощью алгоритма продвигаемого фронта или метода триангуляции Делоне, а затем строится тетраэдральная сетка внутри области. Согласно документации бóльшая часть тетраэдральной сетки строится с помощью быстрого алгоритма построения тетраэдризации Делоне, а остальная часть достраивается с помощью локальных эвристик. Автору диссертационной работы неизвестно, имеется ли возможность задать фиксированную поверхностную сетку и с помощью пакета получить сетку, согласованную на границе с заданной триангуляцией, также неизвестно, имеется ли теоретическое обоснование надёжности используемых методов. Пользователь может ограничивать желаемый размер элементов глобально или локально вблизи точечных источников или внутри прямоугольных областей, что даёт частичный контроль над желаемым размером элементов сетки. Имеется возможность собрать код программы в виде библиотеки. Исходный код открыт и свободно распространяется.

Последний пакет из нашего обзора – пакет CUBIT, разрабатываемый группой разработчиков из Национальной лаборатории Сандии (США), является готовым коммерческим продуктом для построения разных типов сеток в различных областях. Граница области может быть задана как с помощью конструктивной блочной геометрии, так и с помощью сплайнов и фасеток. Проект активно развивает направление автоматического построения гексаэдральных сеток с помощью методов отображения и заметания области. Для построения тетраэдральных сеток используется генератор TetMesh, который нами уже был рассмотрен. Библиотека для подключения к своим программам распространяется отдельно под именем CAMAL, и также является коммерческим продуктом. Исходный код пакета закрыт. Предоставляются лицензии для некоммерческого использования в университетах на весь пакет CUBIT за исключением генератора тетраэдральных сеток TetMesh.

Отметим, что во всех рассмотренных примерах в качестве основного метода построения сетки используется метод построения тетраэдризации Делоне. По сравнению с рассмотренными комплексами программ библиотека для построения сеток из пакета Ani3D обладает как недостатками, так и преимуществами. К недостаткам можно отнести относительно медленную скорость работы по сравнению с коммерческими пакетами, отсутствие полноценного графического интерфейса. К преимуществам можно отнести широкие возможности для задания границы области и построения треугольных поверхностных сеток, построение тетраэдральных сеток, согласованных с заданной граничной триангуляцией, возможность контроля желаемого шага сетки в пространстве и возможность автоматического выбора шага сетки.

Глава 1

Построение двумерных треугольных сеток

В этой главе будет предложен алгоритм, строящий конформные треугольные сетки для областей, заданных дискретизациями своих границ. При построении сетки алгоритм сохраняет заданный след на границе. Предложенный алгоритм применим как к простым многоугольникам, так и к произвольным многокомпонентным многосвязным многоугольным областям, границы которых могут не быть одномерными многообразиями. Вопрос построения дискретной границы для произвольной области с криволинейными границами в этой главе обсуждаться не будет, однако, мы к нему вернёмся в разделе 2.1 следующей главы.

В этой главе будет проведён анализ влияния вычислительных погрешностей, анализ конечности и времени работы алгоритма, также будут предложены две модификации алгоритма, имеющие более низкие оценки сложности. Будут рассмотрены алгоритмы улучшения построенной сетки, а также проведены некоторые эксперименты.

1.1. Обозначения

Упорядоченную пару точек на плоскости будем называть *направленным отрезком*, и обозначать $\overrightarrow{AB}$. Направленный отрезок можно представить как вектор с компонентами $(x, y) = (x_B - x_A, y_B - y_A)$. Будем использовать следующее определение векторного произведения двух векторов:

$$(x_1, y_1) \times (x_2, y_2) = \det \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - y_1 x_2.$$

Для трёх точек A, B, C будем использовать следующую сокращённую запись: $[ABC] = \overrightarrow{CA} \times \overrightarrow{CB}$. Заметим, что площадь треугольника $\triangle ABC$ в таком случае легко выражается по формуле $S(\triangle ABC) = \frac{1}{2} |[ABC]|$. *Положительной полуплоскостью* относительно направленного отрезка $\overrightarrow{AB}$ будем называть геометрическое место точек X, для которых $[ABX] > 0$. Аналогично, *отрицательной* полуплоскостью будем называть геометрическое место точек, для которых $[ABX] < 0$.

Ломаной на плоскости назовём конечный набор отрезков, последовательно соединённых своими концами. При этом не будем запрещать двум последовательным отрезкам ломаной лежать на одной прямой. Если первая и последняя точки ломаной совпадают, то такую ломаную будем называть *замкнутой*. Ломаную, отрезки которой не пересекаются друг с другом, будем называть ломаной без самопересечений.

Многоугольником на плоскости назовём ограниченную открытую часть плоскости, граница которой состоит из одной или нескольких непересекающихся ломаных без самопересечений. *Простым многоугольником* назовём многоугольник, ограниченный одной замкнутой ломаной. На Рис. 1.1 показаны несколько примеров многоугольников. Вершинами многоугольника будем называть вершины его ломаных, а сторонами многоугольника – отрезки ломаных.

Стороны многоугольника можно разбить на два типа: внешние и внутренние. Пусть AB – сторона многоугольника $\mathcal{P}$, M – середина AB, $\Omega_\varepsilon(M)$ – ε-окрестность точки M. Рассмотрим $\mathcal{P}_\varepsilon = \mathcal{P} \cap \Omega_\varepsilon(M)$. Если при любом $\varepsilon > 0$ часть многоугольника $\mathcal{P}_\varepsilon$ лежит по обе стороны от отрезка AB, то будем называть AB *внутренней* стороной или *разрезом*. Если при некотором $\varepsilon > 0$ часть многоугольника $\mathcal{P}_\varepsilon$ лежит только по одну сторону от отрезка AB, то AB будем называть *внешней* стороной многоугольника $\mathcal{P}$. В последнем случае на отрезке AB можно ввести направление так, чтобы $\mathcal{P}_\varepsilon$ оказалась в положи-

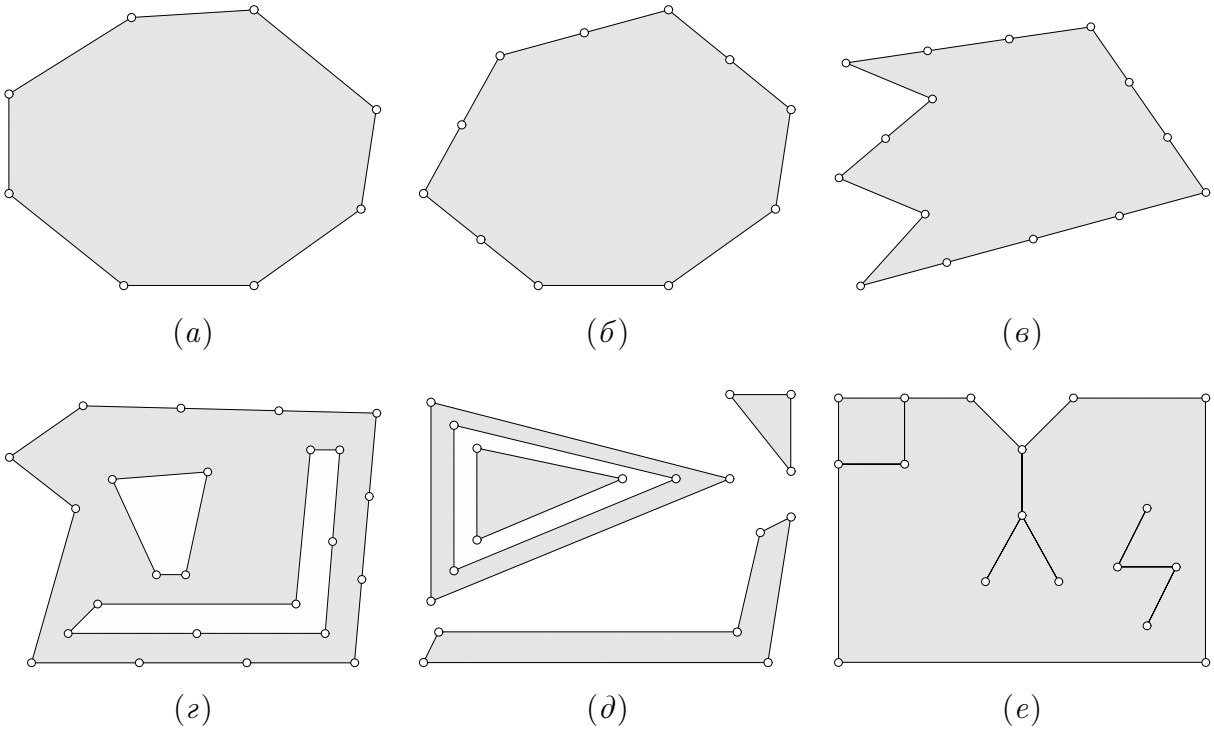
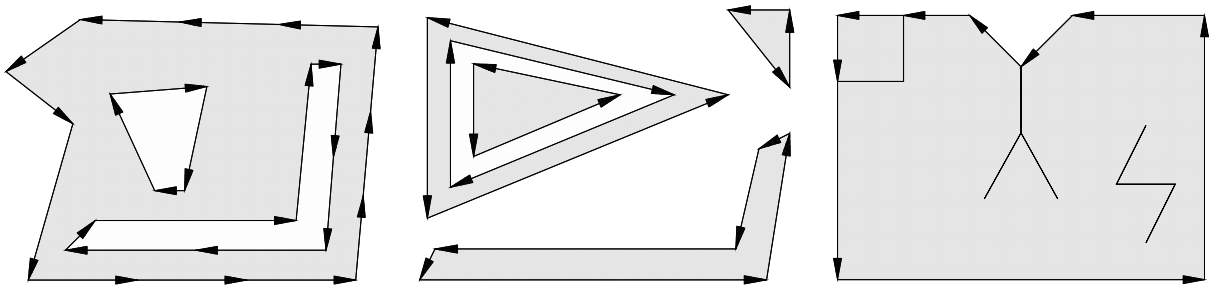


Рис. 1.1. Примеры многоугольников: (*а*) строго выпуклый, (*б*) нестрого выпуклый, (*в*) невыпуклый, (*г*) многосвязный, (*д*) многокомпонентный, (*е*) с внутренними разрезами.

тельной полуплоскости относительно $\overrightarrow{AB}$, будем называть это направление направлением обхода многоугольника против часовой стрелки. На Рис. 1.2 показаны направления обхода внешних сторон нескольких многоугольников. Заметим, что совокупность всех внешних сторон образует одну или несколько замкнутых ломаных.

Фронтом на плоскости будем называть набор направленных отрезков без самопересечений. Каждому многоугольнику $\mathcal{P}$ можно поставить в соответствие фронт $\mathcal{F}(\mathcal{P})$. Для этого введём на внешних сторонах многоугольника направление обхода против часовой стрелки, а для каждой внутренней стороны AB поставим в соответствие пару направленных отрезков $\overrightarrow{AB}$ и $\overrightarrow{BA}$. Совокупность всех этих направленных отрезков и образует фронт $\mathcal{F}(\mathcal{P})$. Будем называть фронт $\mathcal{F}$ *замкнутым*, если существует такой многоугольник $\mathcal{P}$, что $\mathcal{F} = \mathcal{F}(\mathcal{P})$.



(*а*) (*б*) (*в*)

Рис. 1.2. Обход многоугольника против часовой стрелки: (*а*) многосвязный многоугольник, (*б*) многокомпонентный многоугольник, (*в*) многоугольник с внутренними разрезами.

Утверждение 1.1.1. *Площадь многоугольника $\mathcal{P}$ может быть вычислена с помощью соответствующего замкнутого фронта $\mathcal{F} = \mathcal{F}(\mathcal{P}) = \cup_{i=1}^{N} \overrightarrow{A_i B_i}$ по следующей формуле:*

$$S(\mathcal{P}) = \frac{1}{2} \sum_{i=1}^{N} [A_i B_i O], \tag{1.1}$$

где O – начало координат.

Доказательство. Введём векторное поле $\mathbf{u} = (x, y)$, тогда $\mathrm{div}\mathbf{u} = 2$. И по формуле Грина

$$S(\mathcal{P}) = \frac{1}{2} \int\limits_{\mathcal{P}} \mathrm{div}\mathbf{u}\, \mathrm{d}x = \frac{1}{2} \int\limits_{\mathcal{F}} \mathbf{u} \cdot \mathbf{n}\, \mathrm{d}s,$$

где $\mathbf{n}$ – внешняя единичная нормаль к $\mathcal{F}$. Рассмотрим в $\mathcal{F}$ один из отрезков, $\overrightarrow{AB}$. Векторное поле на этом отрезке изменяется линейно, и поэтому интеграл по нему можно заменить на произведение значения подынтегрального выражения в центре отрезка на длину отрезка.

$$\int\limits_{\overrightarrow{AB}} \mathbf{u} \cdot \mathbf{n}\, \mathrm{d}s = \frac{x_A + x_B}{2}(y_B - y_A) + \frac{y_A + y_B}{2}(x_A - x_B) = x_A y_B - x_B y_A = [ABO].$$

Суммируя по всем отрезкам фронта, получаем (1.1). □

Периметром многоугольника назовём сумму длин отрезков соответству-

ющего фронта, иными словами это сумма длин внешних сторон плюс удвоенная сумма длин внутренних сторон.

Треугольной сеткой $\mathcal{T}$ для многоугольной области $\mathcal{P}$ назовём разбиение этой области на непересекающиеся треугольники. Будем называть сетку *конформной*, если любые её два элемента либо не имеют общих точек, либо имеют одну общую вершину, либо имеют одно общее целое ребро. Будем говорить, что конформная сетка согласована с границей $\mathcal{P}$, если каждая сторона $\mathcal{P}$ является ребром какого-то треугольника в сетке. В частности у внутренних сторон многоугольника будет ровно два соседних треугольника, а у внешних – ровно один.

1.2. Алгоритм продвигаемого фронта

Будем решать задачу построения для заданной многоугольной области $\mathcal{P}$ конформной треугольной сетки, согласованной с границей $\mathcal{P}$. В этом разделе мы рассмотрим алгоритм продвигаемого фронта.

Текущим фронтом назовём некоторый замкнутый фронт, который будет являться границей многоугольной области, в которой нужно построить треугольную сетку. В начальный момент времени положим $\mathcal{P}^0 = \mathcal{P}, \mathcal{F}^0 = \mathcal{F}(\mathcal{P}^0), \mathcal{T}^0 = \varnothing$. Далее, на k-ом шаге будем строить новый треугольник $\triangle ABC \subset \mathcal{P}^k$, добавлять его в треугольную сетку, и вычитать из области, в которой сетка ещё не построена: $\mathcal{T}^{k+1} = \mathcal{T}^k \cup \{\triangle ABC\}$, $\mathcal{P}^{k+1} = \mathcal{P}^k \setminus \triangle ABC$, $\mathcal{F}^{k+1} = \mathcal{F}(\mathcal{P}^{k+1})$. Когда $\mathcal{F}^{k+1}$ станет пустым множеством, или что то же самое, $\mathcal{P}^{k+1} = \varnothing$, можно утверждать, что сетка $\mathcal{T} = \mathcal{T}^{k+1}$ полностью покроет $\mathcal{P}$.

Треугольник $\triangle ABC$ будем строить так, что одна из его сторон будет одним из отрезков текущего фронта $\mathcal{F}^k$. Тогда после вычитания из многоугольной области нового треугольника текущий фронт изменится незначительно: как минимум один отрезок из фронта будет удалён и не более чем два новых

будут добавлены.

Пусть $\overrightarrow{AB}$ – некоторый направленный отрезок текущего фронта. Чтобы $\triangle ABX$ лежал внутри $\mathcal{P}^k$, достаточно, чтобы $[ABX] > 0$ и $\triangle ABX$ не пересекал $\mathcal{F}^k$. При этом допускается, чтобы вершина X была вершиной фронта, и чтобы стороны треугольника совпадали с отрезками $\mathcal{F}^k$. Будем называть треугольник, удовлетворяющий этим условиям, *подходящим*. Такой выбор $\triangle ABX$ гарантирует, что после добавления его в сетку, она сохранит свойство конформности. Также при этом будет гарантироваться согласованность конечной сетки $\mathcal{T}$ с начальной областью $\mathcal{P}$.

Для определённости и из некоторых эвристических предположений будем выбирать на каждом шаге из фронта направленный отрезок $\overrightarrow{AB}$ с наименьшей длиной, и строить на нём новый треугольник. При построении $\triangle ABX$ выбор третьей вершины X не является однозначным. Мы можем руководствоваться как желаемым размером треугольника в этой области, так и какими-то другими эвристическими критериями. Будем рассматривать только вершины из некоторого конечного множества Σ. Назовём это множество вершин *кандидатами*. В это множество включим точку C, равноудалённую от точек A и B на некоторое расстояние s, а также добавим все вершины фронта, лежащие в положительной полуплоскости относительно направленного отрезка $\overrightarrow{AB}$.

Так как множество Σ конечно, то мы можем просто перебирать все треугольники $\triangle ABX$, $X \in \Sigma$ до тех пор, пока очередной из них не окажется подходящим. В разделе 1.4.1 будет доказано, что хотя бы один подходящий треугольник при таком выборе Σ найдётся.

Однако сложность такого алгоритма очень большая и на практике он будет неприменим. Рассмотрим некоторую выпуклую открытую окрестность Ω, содержащую точки A, B, C. Выберем из текущего фронта только те отрезки, которые имеют хотя бы одну общую точку с $\bar{\Omega}$, получившийся фронт может оказаться незамкнутым. Назовём его локальным фронтом, и обозначим $\mathcal{F}_\Omega$.

Любая подобласть Ω пересекается с текущим фронтом тогда и только тогда, когда она пересекается с локальным фронтом.

Построим множество $\Sigma_\Omega = \Sigma \cap \Omega$. В силу выпуклости Ω любой треугольник $\triangle ABX$, $X \in \Sigma_\Omega$ будет лежать в Ω. Для проверки пересечения этого треугольника с текущим фронтом достаточно проверить пересечение с локальным фронтом. В качестве области Ω можно выбрать круг с центром в C и радиуса $R > s$. Локальный перебор будет быстрее, но уже не будет гарантировать существование подходящего треугольника, поэтому надо предусмотреть возможность вернуться к полному перебору, если локальный перебор будет закончен неудачно.

Для доказательства конечности операций при работе алгоритма нужно сделать какую-то оценку на максимальное количество треугольников, которое мы с его помощью построим. Для этого нужно ввести какие-то ограничения на сами треугольники, на рёбра, или на вершины сетки. С практической точки зрения наиболее удобным является ограничение снизу на минимальное попарное расстояние между вершинами сетки. Для поддержания этого ограничения будем исключать из Σ точку C, если она лежит близко к вершинам фронта, или к уже построенной сетке. Для этого введём два параметра: h – ограничение на минимальное расстояние до текущего фронта, и r – ограничение на расстояние до вершин фронта. Более подробно об этом механизме будет рассказано в разделе 1.4.2.

С учётом всех предыдущих замечаний запишем алгоритм построения сетки, который назовём Алгоритм 1.1. Будем использовать следующие дополнительные обозначения. На каждом шаге вершины фронта $\mathcal{F}^k$ обозначим как $\{P_i^k\}$, а вершины построенной сетки $\mathcal{T}^k$ как $\{V_i^k\}$. Для некоторого $R > 0$ и точки X через $S_R(X)$ обозначим открытую R-окрестность точки X. Тогда Алгоритм 1.1 запишется следующим образом:

Алгоритм 1.1 Алгоритм продвигаемого фронта в двумерном случае

1: Положить $\mathcal{T}^0 = \varnothing$, $\mathcal{F}^0 = \mathcal{F}(\mathcal{P})$.
2: **для** $k = 0, 1, \ldots$ **начало цикла**
3: Выбрать $\overrightarrow{AB} \in \mathcal{F}^k$ с минимальной длиной.
4: Определить желаемую длину стороны элемента s.
5: Построить вершину C: $AC = BC = s$, $[ABC] > 0$.
6: Выбрать некоторое $R > s$, $h > 0$ и $r > 0$.
7: Построить локальный фронт $\mathcal{F}_R^k$: $\mathcal{F}_R^k \supset \mathcal{F}^k \cap S_R(C)$.
8: Определить множество кандидатов $\Sigma_R^k = \{P_i^k\} \cap S_R(C)$.
9: Если $\{P_i^k\} \cap S_r(C) = \varnothing$ и $\mathcal{F}^k \cap S_h(C) = \varnothing$, то добавить C к Σ_R^k.
10: **для всех** $X \in \Sigma_R^k$: **начало цикла**
11: **если** $\triangle ABX$ не пересекает $\mathcal{F}_R^k$, **тогда**
12: Добавить треугольник $T_k = \triangle ABX$, $\mathcal{T}^{k+1} = \mathcal{T}^k \cup \{T_k\}$.
13: Обновить фронт: $\mathcal{P}^{k+1} = \mathcal{P}^k \setminus T_k$, $\mathcal{F}^{k+1} = \mathcal{F}(\mathcal{P}^{k+1})$.
14: Если $\mathcal{F}^{k+1} = \varnothing$, то перейти к 20.
15: Перейти к 19.
16: **конец если**
17: **конец цикла**
18: Положить $R = \infty$, и перейти к 7.
19: **конец цикла**
20: Положить $\mathcal{T} = \mathcal{T}^{k+1}$.

В следующих разделах мы изучим устойчивость этого алгоритма к вычислительным погрешностям, его конечность и его сложность.

1.3. Влияние вычислительных погрешностей

При реализации алгоритма на ЭВМ особое внимание стоит уделить вычислительным погрешностям, возникающим при работе с вещественными чис-

лами. Использование специализированных библиотек для вычислений с произвольной точностью может решить проблему за счёт увеличения времени работы программы. В этом разделе мы проанализируем влияние вычислительных погрешностей на работу алгоритма, а также предложим несколько идей, которые позволят свести это влияние к минимуму.

1.3.1. Проверка пересечения треугольника с отрезком

Одной из основных операций, чувствительных к вычислительным погрешностям, является операция проверки пересечения треугольника с отрезком. Неправильное определение пересечения из-за вычислительных погрешностей может привести к неправильной работе всего алгоритма. Отметим, что существует только две ситуации получения неправильного результата:

1. Непересекающиеся треугольник и отрезок неверно восприняты как пересекающиеся;

2. Пересекающиеся треугольник и отрезок неверно восприняты как непересекающиеся.

С точки зрения Алгоритма 1.1 ошибки первого типа не являются критичными, пересечение треугольника и отрезка означает, что новый треугольник не пройдёт проверку пересечения с текущим фронтом, и соответствующая вершина-кандидат будет забракована. Ошибки второго типа в свою очередь могут привести к неверному прохождению проверки пересечения треугольника с текущим фронтом, что может привести к самопересекающемуся фронту или к неконформной сетке. Таким образом, ошибки первого типа лишь увеличивают перебор вершин-кандидатов, незначительно увеличивая общее время работы программы, а ошибки второго типа могут привести к неправильному результату работы программы.

Разработаем алгоритм проверки пересечения пары треугольник-отрезок, исключающий возникновение ошибок второго типа, и допускающий возникновение ошибок первого типа. В общем случае два плоских выпуклых объекта не пересекаются тогда и только тогда, когда существует прямая, отделяющая их друг от друга. В условиях нашего Алгоритма 1.1 также допускаются общие вершины и общие рёбра, например, отрезок и треугольник могут иметь общую вершину, или отрезок может быть стороной треугольника. В этих случаях конформность сетки нарушаться не будет, поэтому их нужно рассматривать как непересекающиеся.

Пусть на входе алгоритма у нас есть треугольник $\triangle ABC$ и отрезок PQ. Будем предполагать, что $[ABC] > 0$, так как только такие треугольники будут тестироваться в Алгоритме 1.1. Сначала определим, сколько общих вершин у треугольника и отрезка.

Рассмотрим случай, когда общих вершин нет. Треугольник и отрезок не пересекаются тогда и только тогда, когда либо треугольник лежит целиком в одной из полуплоскостей относительно прямой PQ, либо отрезок PQ лежит по другую сторону от треугольника относительно одной из его сторон. Таким образом, если пересечений нет, то выполнено хотя бы одно из следующих условий:

$$\left[\begin{array}{ll} [UVA] > 0, & [UVB] > 0, \quad [UVC] > 0; \\ [UVA] < 0, & [UVB] < 0, \quad [UVC] < 0; \\ [ABU] < 0, & [ABV] < 0; \\ [BCU] < 0, & [BCV] < 0; \\ [CAU] < 0, & [CAV] < 0. \end{array}\right.$$

Верно и обратное, если выполнено одно из этих условий, то пересечений нет. Эти условия можно записать в более удобной форме:

$$\left[\begin{array}{l} \operatorname{sign}([UVA]) + \operatorname{sign}([UVB]) + \operatorname{sign}([UVC]) = \pm 3; \\ \operatorname{sign}([ABU]) + \operatorname{sign}([ABV]) = -2; \\ \operatorname{sign}([BCU]) + \operatorname{sign}([BCV]) = -2; \\ \operatorname{sign}([CAU]) + \operatorname{sign}([CAV]) = -2. \end{array}\right.$$

Перейдём к случаю, когда у треугольника и отрезка есть одна общая вершина. В этом случае треугольник и отрезок не пересекаются в нашем смысле тогда и только тогда, когда отрезок PQ лежит в отрицательной полуплоскости относительно одной из сторон треугольника, при этом их общая вершина лежит на прямой, а вторая попадает в отрицательную полуплоскость. Эти условия можно записать в таком виде:

$$\left[\begin{array}{l} \operatorname{sign}([ABU]) + \operatorname{sign}([ABV]) = -1; \\ \operatorname{sign}([BCU]) + \operatorname{sign}([BCV]) = -1; \\ \operatorname{sign}([CAU]) + \operatorname{sign}([CAV]) = -1. \end{array}\right.$$

Если выполнено хотя бы одно из этих условий, то пересечения нет, и наоборот, если нет пересечения, то выполнено хотя бы одно из условий.

В последнем случае, когда обе вершины отрезка совпали с двумя вершинами треугольника, ребро PQ является стороной треугольника, и по нашей договорённости треугольник и отрезок считаются непересекающимися.

Предположим, что у нас есть функция $d(ABC)$, неточно вычисляющая знак выражения $[ABC]$. Но при этом, если $d(ABC) \neq 0$, то $d(ABC) = \operatorname{sign}([ABC])$, и $d(ABC) = 0$ в спорных ситуациях. Тогда проверку на пересечение можно выполнять с помощью Алгоритма 1.2.

Алгоритм 1.2 Проверка пересечения треугольника с отрезком

1: Найти общие вершины у треугольника ABC и отрезка PQ

2: **если** общих вершин нет, **тогда**

3: Если $d(UVA) + d(UVB) + d(UVC) = \pm 3$, то пересечения нет.

4: Если $d(ABU) + d(ABV) = -2$, то пересечения нет.

5: Если $d(BCU) + d(BCV) = -2$, то пересечения нет.

6: Если $d(CAU) + d(CAV) = -2$, то пересечения нет.

7: **конец если**

8: **если** одна общая вершина, **тогда**

9: Если $d(ABU) + d(ABV) = -1$, то пересечения нет.

10: Если $d(BCU) + d(BCV) = -1$, то пересечения нет.

11: Если $d(CAU) + d(CAV) = -1$, то пересечения нет.

12: **конец если**

13: **если** две общих вершины, **тогда**

14: Пересечения нет.

15: **конец если**

16: В остальных случаях считаем, что треугольник и отрезок пересекаются.

Отметим, что допустимая неточность вычисления $d(ABC)$ может привести только к ошибкам первого типа, и Алгоритм 1.2 исключает ошибки второго типа.

1.3.2. Вычисление знака определителя 2×2

Операция определения знака выражения $[ABC]$ является основной операцией, используемой как в Алгоритме 1.1, так и в Алгоритме 1.2. Предложим способ неточного определения знака $d(ABC)$ со следующим свойством: если $d(ABC) \neq 0$, то $d(ABC) = \text{sign}([ABC])$.

Выражение $[ABC]$ сводится к определителю 2×2:

$$[ABC] = \overrightarrow{CA} \times \overrightarrow{CB} = \det \begin{vmatrix} x_A - x_C & y_A - y_C \\ x_B - x_C & y_B - y_C \end{vmatrix} = \det \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc = D.$$

При вычислении элементов матрицы мы уже вносим некоторую относительную погрешность. В дополнение к этому в выражения ad и bc опять вносится относительная погрешность, и в конце в D вносится ещё одна погрешность. Суммарную абсолютную погрешность можно оценить сверху как $\varepsilon(|a| + |b| + |c| + |d| + 2|ad| + 2|bc|)$, где ε зависит от машинной точности. С учётом этой допустимой погрешности запишем алгоритм для определения $\operatorname{sign}([ABC])$.

Алгоритм 1.3 Вычисление $d(ABC)$

1: Вычислить $a = x_A - x_C$, $b = y_A - y_C$, $c = x_B - x_C$, $d = y_B - y_C$.
2: Оценить погрешность $r = \varepsilon(|a| + |b| + |c| + |d| + 2|ad| + 2|bc|)$.
3: Вычислить определитель матрицы $D = ad - bc$.
4: Если $D > r$, то **вернуть** 1.
5: Если $D < -r$, то **вернуть** -1.
6: Иначе **вернуть** 0.

В Алгоритме 1.3 величина D вычисляется с абсолютной погрешностью, не превосходящей r. Поэтому если $d(ABC) \neq 0$, то можно быть уверенным, что знак D вычислен правильно, и $d(ABC) = \operatorname{sign}([ABC])$.

На практике, если для хранения координат в памяти ЭВМ используется тип вещественных чисел с одинарной точностью, а вычисления величин a, b, c, d, D в Алгоритме 1.3 производятся с двойной точностью, то машинной точности хватит для точного вычисления как всех промежуточных величин, так и последней величины D. В этом случае можно положить $\varepsilon = 0$, и Алгоритм 1.3 будет вычислять $\operatorname{sign}([ABC])$ точно.

1.4. Конечность работы алгоритма

Проведём анализ Алгоритма 1.1, и покажем конечность его работы, то есть конечность числа операций. В предложенном алгоритме два явных вложенных цикла и один неявный за счёт перезапуска перебора в строке 18. Цикл перебора в строке 10 всегда конечен в силу конечности множества Σ_R^k. Необходимо показать, что после перезапуска перебора подходящий треугольник всегда будет найден, а также показать, что внешний цикл в строке 2 будет конечным.

1.4.1. Существование подходящего треугольника

Докажем, что поиск вершины-кандидата X в строке 10 Алгоритма 1.1 всегда найдёт хотя бы один подходящий треугольник после перезапуска перебора в строке 18. Сначала докажем вспомогательную лемму.

Лемма 1.4.1. *Пусть $\mathcal{P}$ – многоугольник с вершинами $P_1P_2\dots P_n$. P_1P_2 – сторона многоугольника $\mathcal{P}$, и угол при $P_1 \geq \pi$. Тогда существует такая диагональ P_1Q, целиком лежащая внутри $\mathcal{P}$, что угол $\angle P_2P_1Q < \pi$.*

Доказательство. Для определённости рассмотрим случай, когда P_1P_2 – внешняя сторона, и направление $\overrightarrow{P_1P_2}$ совпадает в направлением обхода $\mathcal{P}$ против часовой стрелки. Остальные случаи рассматриваются аналогично. Пусть Π^+ – положительная полуплоскость относительно $\overrightarrow{P_1P_2}$, $\Pi^+ = \{X : [P_1P_2X] > 0\}$. Рассмотрим множество вершин многоугольника $\mathcal{P}$, лежащих в этой полуплоскости, $\Sigma = \{P_i\} \cap \Pi^+$. Покажем, что Σ не пустое. На самом деле, предположим, что оно пустое. Выпустим из середины P_1P_2 перпендикуляр в сторону Π^+ до пересечения с границей $\mathcal{P}$. Многоугольник $\mathcal{P}$ ограничен, поэтому пересечение всегда существует. Пусть AB – ребро, которое мы пересекли.

Если $\Sigma = \varnothing$, то $A \notin \Pi^+$, $B \notin \Pi^+$, и, следовательно, $AB \cap \Pi^+ = \varnothing$, и точка пересечения с перпендикуляром не лежит в Π^+ – противоречие.

Пусть вершина $S = \arg\max_{X \in \Sigma} \angle P_2 P_1 X$. Выпустим луч $P_1 S$ до пересечения с границей P в точке C. Так как угол при вершине $P_1 \geq \pi$, а $\angle P_2 P_1 S < \pi$, то $C \neq P_1$. Пусть C лежит на стороне DE. Хотя бы одна из вершин D, E лежит в Σ, пусть $D \in \Sigma$. Если $C = D$, то диагональ $P_1 C$ лежит внутри $\mathcal{P}$ и угол $\angle P_2 P_1 C < \pi$, утверждение леммы верно. Иначе, CD – часть границы многоугольника, а $P_1 C$ целиком лежит внутри P (см. Рис. 1.3, *а*). Рассмотрим множество вершин $\mathcal{P}$, попавших в треугольник $\triangle P_1 C D$, $\Sigma_\triangle = \{P_i\} \cap \triangle P_1 C D$. $\Sigma_\triangle$ не пустое, так как $D \in \Sigma_\triangle$.

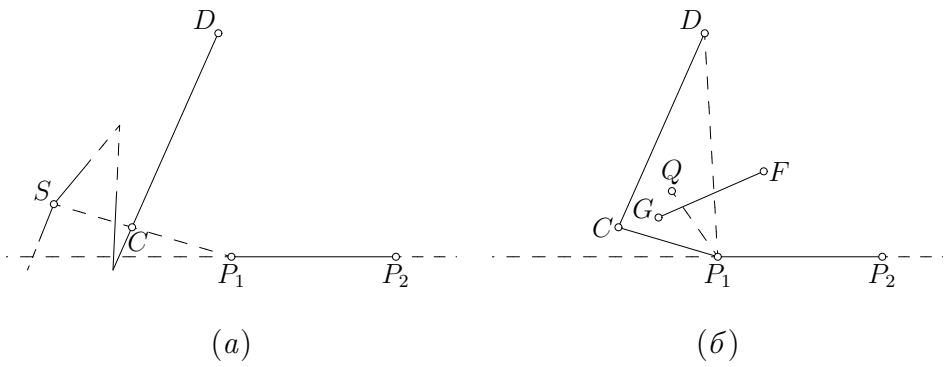


Рис. 1.3. К доказательству Леммы 1.4.1.

Пусть вершина $Q = \arg\max_{X \in \Sigma_\triangle} \angle P_2 P_1 X$, если максимальное значение угла достигается на нескольких вершинах, то выберем из них ту, которая ближе к P_1. Покажем, что Q удовлетворяет условию леммы. Действительно, предположим, что диагональ $P_1 Q$ пересекается со стороной многоугольника FG (см. Рис. 1.3, *б*). Заметим, что FG не пересекает CD и не пересекает $P_1 C$, при этом точка Q лежит в $\triangle P_1 C D$. Тогда одна из точек, пусть G, также лежит в $\triangle P_1 C D$ и либо $\angle P_2 P_1 G > \angle P_2 P_1 Q$, либо $\angle P_2 P_1 G = \angle P_2 P_1 Q$ и G лежит ближе к P_1 чем Q, что противоречит выбору Q. $\square$

С помощью этой леммы можно доказать следующую теорему.

Теорема 1.1. *Пусть $\mathcal{P}$ – многоугольник с вершинами $P_1P_2\dots P_n$. P_1P_2 – сторона многоугольника $\mathcal{P}$. Тогда существует такая третья вершина Q, что треугольник P_1QP_2 целиком лежит внутри $\mathcal{P}$.*

Доказательство. Если один из углов при P_1 или $P_2 \geq \pi$, то, используя Лемму 1.4.1, мы можем так мысленно разрезать диагональю многоугольник $\mathcal{P}$ на две части, что угол при соответствующей вершине станет меньше π. Пусть соседние вершины при P_1 и P_2 – A и B соответственно. Рассмотрим область Ω, ограниченную отрезком P_1P_2 и лучами P_1A, P_2B, $\Omega = \{X : [AP_1X] \geq 0, [P_1P_2X] > 0, [P_2BX] \geq 0\}$. Пусть $\Sigma = \{P_i\} \cap \Omega$. Так как стороны AP_1 и P_2B не пересекаются, то хотя бы одна из вершин A, B попадёт в Σ.

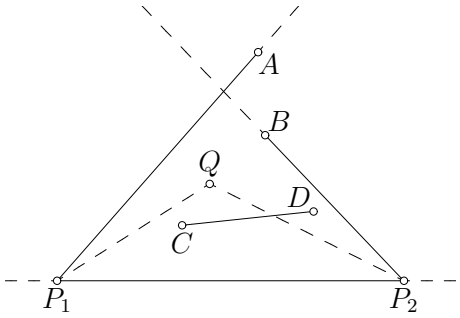


Рис. 1.4. К доказательству Теоремы 1.1.

Заметим, что величина $[P_1P_2X]$ с точностью до множителя и знака равна расстоянию от точки X до прямой P_1P_2. Выберем ближайшую к прямой P_1P_2 вершину из Σ, $Q = \arg\min_{X\in\Sigma}[P_1P_2X]$. Покажем, что треугольник P_1QP_2 целиком лежит внутри $\mathcal{P}$. Предположим, что это не так, и $\triangle P_1QP_2$ пересекает сторону CD многоугольника $\mathcal{P}$ (см. Рис. 1.4). Пусть для определённости C лежит ближе к прямой P_1P_2. Но тогда $[P_1P_2C] < [P_1P_2Q]$. При этом сторона CD не пересекает ни AP_1, ни P_1P_2, ни P_2B, то есть $C \in \Sigma$, что противоречит выбору Q. □

Наличие повторного выполнения перебора в строке 18 Алгоритма 1.1 делает перебор в строке 10 полным. Из Теоремы 1.1 с учётом предыдущих

замечаний о влиянии вычислительных погрешностей и возможности точного вычисления в Алгоритме 1.3 следует, что после повторного перебора подходящий треугольник всегда будет найден.

1.4.2. Конечность числа построенных треугольников

Покажем, что внешний цикл в строке 2 Алгоритма 1.1 будет конечным, то есть количество построенных треугольников конечно. Для этого мы сделаем оценку на количество треугольников через количество вершин, а количество вершин ограничим сверху за счёт конечности многоугольника $\mathcal{P}$ и ограничения снизу на расстояние между вершинами $\{V_i^k\}$.

Пусть d_k – минимальное из попарных расстояний между точками из $\{P_i^k\}$, а ρ_k – минимальное из попарных расстояний между точками из $\{V_i^k\}$, $\rho_0 = \infty$. При добавлении нового треугольника $\triangle ABX$ в $\mathcal{T}^k$ возможны две ситуации: $X \in \{P_i^k\}$ и $X = C$. В первом случае минимальное расстояние между вершинами сетки не уменьшится больше чем до минимального расстояния между вершинами фронта, и минимальное расстояние между вершинами фронта не уменьшится: $\rho_{k+1} \geq \min(\rho_k, d_k)$, $d_{k+1} \geq d_k$. Рассмотрим случай $X = C$. Так как $\triangle ABC$ не пересекает $\mathcal{F}^k$, то расстояние от C до $\{V_i^k\}$ не меньше расстояния от C до $\mathcal{F}^k$, которое по построению не меньше h, выбранного в строке 6. Расстояние от C до $\{P_i^k\}$ не меньше r. Соответственно, $\rho_{k+1} \geq \min(\rho_k, h)$, $d_{k+1} \geq \min(d_k, r)$. Предложим некоторый эвристический способ выбора параметров s, r и h. Обозначим $l = |AB|$ – длина отрезка AB, а h_c – высота в $\triangle ABC$ из вершины C.

Предположим, что у нас есть некоторая скалярная функция $s(x, y) : \mathcal{P} \to \mathbb{R}^+$, и мы хотим чтобы размер элементов в сетке менялся в соответствии с этой зависимостью. В этом случае будем говорить, что размер треугольников задаётся *пользовательской функцией* $s(x, y)$. Потребуем, чтобы

функция $s(x, y)$ была отделена от нуля на $\mathcal{P}$: $s(x, y) > s_{\min} > 0$. Возьмём $s = \max(\frac{5}{9}l, s(x, y))$. Такой выбор гарантирует существование точки C и ограничивает высоту треугольника снизу: $h_c \geq \sqrt{\frac{25}{81} - \frac{1}{4}}l = \frac{\sqrt{19}}{18}l$.

Если соответствующая функция $s(x, y)$ не задана, то будем определять желаемую длину треугольника на основе длины его основания и некоторого параметра, отвечающего за скорость увеличения размера треугольников: $s = \gamma l$, где $\gamma \geq 1$ – параметр, который задаётся пользователем. Будем говорить, что шаг сетки выбирается *автоматически*. Так как $\gamma \geq 1$, то точка C всегда существует, а $h_c = \sqrt{\gamma^2 - \frac{1}{4}}l \geq \frac{\sqrt{3}}{2}l$.

Возьмём параметр $h = \frac{1}{2}h_c$, а параметр r вычислим по следующей формуле:

$$r = \frac{\beta}{2}\left(s + s_0 + \sqrt{(s - s_0)^2 + \alpha^2}\right), \tag{1.2}$$

где $\beta > 0$, $s_0 \geq 0$, α – некоторые параметры. Отметим, что при $s > s_0$ величина $r > \beta s$, а при $s < s_0$ величина $r > \beta s_0$. Параметр α можно подобрать так, чтобы при $s = s_0$ было выполнено $r = s_0$. Ограничение $r > \beta s$ на практике позволяет избегать появления резких перепадов размеров соседних отрезков в новом фронте. А ограничение $r > \beta s_0$ позволяет ограничить снизу величину r.

При реализации алгоритма на ЭВМ в пакете Ani2D параметры выбираются одним из двух способов в зависимости от того, задаётся ли размер треугольников пользователем, или используется автоматический выбор размера. Примеры использования разных способов будут представлены в разделе 1.7.2. Рассмотрим оба случая отдельно.

Размер треугольников задаётся пользователем. Положим $\beta = \frac{1}{2}$, $s_0 = 0$, $\alpha = 0$. Тогда $r = \frac{1}{2}s$. По построению $s > s_{\min}$, следовательно, $r > \frac{1}{2}s_{\min}$. Тогда $d_k \geq \min(d_0, \frac{1}{2}s_{\min})$ для любого k. По построению $h \geq \frac{\sqrt{19}}{36}l \geq \frac{\sqrt{19}}{20}s$ и $s > s_{\min}$, поэтому $\rho_k \geq \min(d_0, \frac{\sqrt{19}}{20}s_{\min})$ для любого k.

Пусть теперь размер треугольников выбирается автоматически. Возьмём $\beta = \frac{1}{2}$, $l_0 = d_0$, $\alpha = 2s_0(1-\beta)$. Тогда $r > \frac{1}{2}d_0$, то есть $d_k > \frac{1}{2}d_0$ для любого k. По построению $h \geq \frac{\sqrt{3}}{4}l$, а $l \geq d_k$, поэтому $h > \frac{\sqrt{3}}{8}d_0$, следовательно, $\rho_k > \frac{\sqrt{3}}{8}d_0$ для любого k.

Мы только что показали, что существует такое $\delta > 0$, что $\rho_k > \delta$ для любого k. С помощью этого условия мы можем сделать оценку на максимальное количество вершин. Все вершины сетки $\{V_i^k\}$ лежат в замыкании $\mathcal{P}$ и попарное расстояние между ними больше δ. Пусть v_k – количество вершин в $\{V_i^k\}$. Покроем каждую вершину кругом с радиусом $\frac{\delta}{2}$, тогда круги не будут пересекаться, и покроют площадь равную $v_k\pi\frac{\delta^2}{4}$. Расширим исходный многоугольник $\mathcal{P}$ на δ, $\mathcal{P}_\delta = \{X : \mathrm{dist}(X, \mathcal{P}) < \frac{\delta}{2}\}$, см. Рис. 1.5. Тогда $\mathcal{P}_\delta$ полностью покроет круги. Оценим площадь $\mathcal{P}_\delta$. У каждой стороны многоугольника длиной l появляется прямоугольник, который даёт прирост площади не больше чем $\frac{\delta}{2}l$. Таким образом прирост площади за счёт прямоугольников составит $\frac{\delta}{2}p(\mathcal{P})$, где $p(\mathcal{P})$ – периметр многоугольника $\mathcal{P}$. Круговые сектора у вершин с внутренним углом α дают прирост площади не больше чем $(\pi - \alpha)\frac{\delta^2}{8}$. Причём эта оценка верна и для внутренних углов $> \pi$, в этом случае прирост становится отрицательным, но он полностью покрывается наложением двух прямоугольников, которые мы учли ранее. Из формулы на сумму углов многоугольника можно оценить вклад всех круговых секторов как $\pi\frac{\delta^2}{4}K(\mathcal{P})$, где $K(\mathcal{P})$ – количество компонент связности в $\mathcal{P}$. Имеем $S(\mathcal{P}_\delta) \leq S(\mathcal{P}) + \frac{\delta}{2}p(\mathcal{P}) + \pi\frac{\delta^2}{4}K(\mathcal{P})$. Отсюда можно сделать оценку на количество вершин: $v_k \leq \frac{4S(\mathcal{P})}{\pi\delta^2} + \frac{2p(\mathcal{P})}{\pi\delta} + K(\mathcal{P})$.

Заметим, что аналогичная оценка верна и на количество вершин в фронте $\mathcal{F}^k$:

$$N(\{V_i^k\}) \leq \frac{4S(\mathcal{P})}{\pi\delta^2} + \frac{2p(\mathcal{P})}{\pi\delta} + K(\mathcal{P}). \tag{1.3}$$

Треугольную сетку можно рассматривать как планарный граф [61]. С помощью формулы Эйлера можно доказать следующее утверждение.

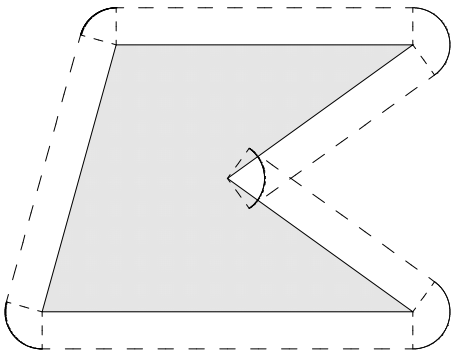


Рис. 1.5. Расширение многоугольника на $\frac{\delta}{2}$.

Утверждение 1.4.1. *Пусть на плоскости задана произвольная сетка с v вершинами, e рёбрами и n ячейками. Тогда верны следующие оценки:* $n \leq \max(0, 2v-5)$, $e \leq \max(2, 3v-6)$.

Доказательство. Будем рассматривать плоскую сетку как планарный граф, для которого верна формула Эйлера:

$$v - e + f = k + 1,$$

где v – количество вершин, e – количество рёбер, f – количество граней, включая внешнюю, а k – количество компонент связности графа. Тогда количество ячеек в сетке $n = f - 1$. Пусть $e \geq 3$, тогда у внешней грани не меньше трёх рёбер. Остальные грани являются ячейками сетки, поэтому в них тоже не меньше трёх рёбер в каждой. Каждое ребро лежит ровно в двух гранях, поэтому $2e \geq 3f$, то есть $f \leq \frac{2}{3}e$. Так как $k \geq 1$, то $v + f \geq 2 + e$. Вычитая из этого неравенства предыдущее, получаем $v \geq 2 + \frac{1}{3}e$, то есть $e \leq 3v - 6$, а $f \leq \frac{2}{3}e \leq 2v-4$. Из последнего получаем, что $n = f-1 \leq 2v-5$, утверждение доказано. Отметим, что эти оценки неулучшаемы. □

Из этого утверждения следует, что количество треугольников в $\mathcal{T}^k$ ограничено:

$$k \leq \frac{8S(\mathcal{P})}{\pi\delta^2} + \frac{4p(\mathcal{P})}{\pi\delta} + 2K(\mathcal{P}) - 5. \tag{1.4}$$

Эта оценка даёт ограничение на количество итераций во внешнем цикле Алгоритма 1.1, что в свою очередь доказывает конечность числа операций алгоритма.

1.5. Скорость работы алгоритма продвигаемого фронта

Проведём краткий анализ скорости работы предложенного Алгоритма 1.1. Мы уже получили оценку сверху на количество итераций внешнего цикла в строке 2 в (1.4). Обозначим через $N(\mathcal{F}^k)$ и $N(\mathcal{F}_R^k)$ количество отрезков в $\mathcal{F}^k$ и $\mathcal{F}_R^k$, соответственно. Выпишем все нетривиальные операции, которые используются в Алгоритме 1.1:

1. Выбор отрезка с минимальной длиной из $\mathcal{F}^k$ (строка 3);
2. Построение локального фронта $\mathcal{F}_R^k$ (строка 7);
3. Построение списка кандидатов Σ_R^k (строка 8);
4. Проверка пересечения треугольника с локальным фронтом (строка 11);
5. Обновление фронта (строка 13);

Операция 4 выполняется за $N(\mathcal{F}_R^k)$ операций проверки пересечения треугольника с отрезком (см. Алгоритм 1.2). Операция 3 может быть сделана за $2N(\mathcal{F}_R^k)$ операций проверки принадлежности вершин из $\mathcal{F}_R^k$ окрестности $S_R(C)$.

Сложность операций 1, 2, и 5 зависит от используемых структур данных для хранения фронта. Будем использовать для хранения фронта кучу с минимальным по длине элементом в корне и дерево поиска для быстрого поиска локального фронта. В этом случае операция 1 становится тривиальной, сложность операции 2 будет в среднем пропорциональна $\log(N(\mathcal{F}^k))+N(\mathcal{F}_R^k)$. Операция 5 состоит из трёх операций добавления или удаления отрезка из

фронта, сложность каждой из них в среднем пропорциональна $\log(N(\mathcal{F}^k))$. Более подробно устройство дерева поиска обсудим в следующем разделе.

1.5.1. Быстрый поиск локального фронта

В этом разделе мы рассмотрим структуру данных, позволяющую быстро находить подмножество отрезков на плоскости, пересекающихся с заданной окрестностью. Предлагаемая структура основана на классическом четверичном дереве поиска. Зафиксируем некоторый ограничивающий квадрат для фронта $\mathcal{F}$, то есть такой квадрат, который полностью покрывает все отрезки фронта, пусть его сторона равна H. Будем строить четверичное дерево, у которого у каждой вершины либо нет потомков, либо их ровно четыре. Каждой вершине дерева будем ставить в соответствие подквадрат – некоторую часть ограничивающего квадрата. А именно, корень дерева будет соответствовать полному квадрату. Для каждой вершины с четырьмя потомками разделим соответствующий ей квадрат на четыре одинаковых квадрата и поставим их в соответствие потомкам. *Уровнем* вершины в графе назовём количество рёбер в пути от этой вершины до корня дерева, так у корня дерева уровень будет равен 0, а у его непосредственных потомков – 1.

Для каждого отрезка из $\mathcal{F}$ найдём его середину и длину l. По длине отрезка найдём такое наименьшее число $n \geq 0$, что $l \geq H2^{-n}$. Число n будем называть *уровнем* отрезка. К каждой вершине дерева припишем список отрезков из $\mathcal{F}$, у которых уровень совпадает с уровнем вершины в графе, и середина отрезка попадает в соответствующий подквадрат. По построению $\frac{1}{2}l < H2^{-n}$, то есть меньше стороны квадратов на уровне n в дереве поиска. Это означает, что этот отрезок фронта может пересекать только текущий квадрат и 8 его непосредственных соседей. В дереве будем хранить только те листья, в которых есть хотя бы один отрезок. Пусть у нас также имеется

оценка на минимальное попарное расстояние между вершинами фронта – d. Тогда длина каждого отрезка не меньше d, и максимальный уровень в дереве будет не больше $\log_2 \frac{H}{d}$. Количество вершин в фронте можно грубо оценить по (1.3), а количество отрезков в фронте – используя Утверждение 1.4.1.

Пусть теперь мы хотим найти локальный фронт, то есть те отрезки, которые пересекаются с окрестностью $S_R(C)$. Для этого мы делаем обход дерева, рассматривая только подквадраты, имеющие общие точки с $S_R(C)$, и их непосредственных соседей. В каждой такой вершине дерева проверяются приписанные ей отрезки. Рассмотрим множество квадратов, соответствующих вершинам на k-ом уровне дерева, которые участвуют в переборе. Эти квадраты полностью покрываются кругом, если его радиус увеличить на толщину двойного слоя квадратов, то есть кругом радиуса $R + 2\sqrt{2}H2^{-k}$. В этом круге мы можем оценить количество вершин фронта по формуле, аналогичной (1.3), и из результатов Утверждения 1.4.1 получить оценку на количество отрезков:

$$e_k \le \max\left(2, \frac{12(R + 2\sqrt{2}H2^{-k})^2}{d^2} + \frac{12(R + 2\sqrt{2}H2^{-k})}{d} - 3\right) < \\ < 12\frac{R^2}{d^2} + 48\sqrt{2}\frac{RH}{d^2 2^k} + 96\frac{H^2}{d^2 2^{2k}} + 12\frac{R}{d} + 24\sqrt{2}\frac{H}{d2^k} + 2 \quad (1.5)$$

На практике длина фронта, пересекающего круг радиуса $R + 2\sqrt{2}H2^{-k}$ пропорциональна его радиусу, а из ограничения на минимальную длину отрезков на уровне k можно сделать вывод, что в среднем на уровне k будет рассмотрено количество отрезков пропорциональное величине $\frac{R}{H}2^k + 1$. В сумме по всем уровням количество рассмотренных рёбер составит величину порядка $\frac{R}{H}2^{n+1} + n$, где n – максимальный уровень в дереве, и зависит от d. Сложность такого поиска можно оценить в среднем как величину пропорциональную $\frac{R}{d} + \log_2 \frac{H}{d}$. Количество операций, необходимые для обхода дерева, не превысят по порядку количество протестированных отрезков, так что их тоже можно включить в эту оценку.

Отметим, что в худшем случае сложность поиска локального фронта может составить величину, пропорциональную общему количеству отрезков во всём фронте, причём независимо от R. Пример конфигурации фронта, отображающей возможный из "худших" случаев, приведён на Рис. 1.6.

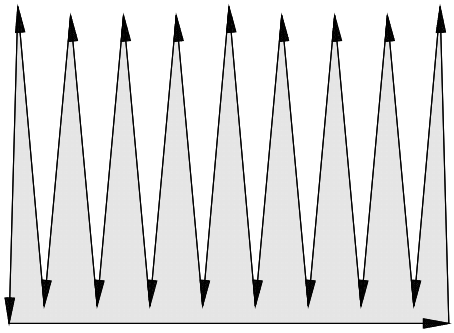


Рис. 1.6. Пример наихудшей конфигурации фронта.

Операция добавления отрезка в фронт по сложности пропорциональна глубине дерева. Операция поиска и удаления отрезков из фронта, сводится к поиску нужной вершины в дереве за время пропорциональное глубине дереве, а также поиску и удалению в обычном связном списке отрезков, соответствующих найденной вершине. Обычно длина списка небольшая на большинстве конфигураций фронтов, однако в худшем случае может достигать величины пропорциональной количеству отрезков во всём фронте, см. Рис. 1.6.

1.5.2. Анализ сложности алгоритма

Подытожим имеющиеся результаты. Пусть на входе задан многоугольник $\mathcal{P}$ с площадью S, периметром p, состоящий из K компонент связности, и полностью покрываемый квадратом со стороной H. Если пользователем задана функция желаемого размера треугольников $s(x, y)$, то пусть она отделена от нуля: $s(x, y) > s_{\min} > 0$.

В разделе 1.4.2 было показано, что существует параметр δ, с помощью которого можно оценить максимальное количество треугольников в области

(1.4). Оценим сверху теперь сложность одной итерации внешнего цикла в Алгоритме 1.1. Нас будет интересовать, как растёт сложность алгоритма при уменьшении шага сетки, и, соответственно, уменьшении δ.

Предположим, что подходящий треугольник был найден в локальном переборе. На практике R выбирается в пределах от s до $2s$, и длины отрезков фронта в этой окрестности сопоставимы с s. Оценка в среднем на количество отрезков в локальном фронте $N(\mathcal{F}_R^k)$ получается не зависящей от δ. Однако в худшем случае может составить величину пропорциональную $N(\mathcal{F}^k)$, которая в свою очередь в худшем случае достигает $\frac{4S}{\pi\delta^2}+\frac{2p}{\pi\delta}+K$. Поиск локального фронта в среднем пропорционален $N(\mathcal{F}_R^k)+\log_2\frac{H}{\delta}$.

Далее, в переборе для каждого отрезка локального фронта требуется проверить его пересечение с локальным фронтом. Сложность этой операции пропорциональна $N(\mathcal{F}_R^k)^2$, что в среднем является величиной, не зависящей от δ. Но в худшем случае сложность может достигать величины, пропорциональной $N(\mathcal{F}^k)^2$.

В случае рестарта в строке 18 проводится полный перебор, который соответствует худшей оценке для локального перебора.

Общее время работы алгоритма в среднем пропорционально $(\frac{4S}{\pi\delta^2}+\frac{2p}{\pi\delta}+K)\log_2\frac{H}{\delta}$. В худшем случае оно пропорционально $(\frac{4S}{\pi\delta^2}+\frac{2p}{\pi\delta}+K)^3$. Оценка средней скорости работы будет экспериментально подтверждена в разделе 1.7.1.

Покажем пример набора областей, для которых сложность алгоритма значительно хуже средней оценки. Рассмотрим область в виде расчёски, представленную на Рис. 1.6. Будем увеличивать в ней количество зубьев, N, при этом площадь будет оставаться примерно одной и той же, периметр будет пропорционален N. Параметр δ будет уменьшаться обратно пропорционально N. Количество отрезков в начальном фронте будет $2N+1$. На каждом шаге локальная окрестность будет почти целиком накрывать всю область,

поэтому сложность одной итерации внешнего цикла будет пропорциональна N^2. Всего будет построен $2N-1$ треугольник. Общая сложность работы составит величину пропорциональную N^3, что значительно хуже оценки в среднем порядка $N \log N$.

Вспомним конструктивное доказательство Теоремы 1.1, с помощью которого можно сразу находить подходящий треугольник и даже не проводить проверку пересечения его с фронтом. Одна итерация внешнего цикла в таком случае сводится к поиску ближайшей к отрезку вершины в положительной полуплоскости. За счёт использования деревьев поиска сложность этой операции в среднем будет пропорциональна $\log N$, и N в худшем случае, где N – количество вершин в текущем фронте. Если не накладывать ограничений на исходный многоугольник $\mathcal{P}$, то нам также может понадобится операция, описанная в Лемме 1.4.1, сложность которой в худшем случае пропорциональна N. Общая сложность нового алгоритма будет пропорциональна в среднем $N \log N$, и N^2 в худшем случае. Отметим, что минусом этой модификации алгоритма является невозможность контролировать размер элементов, а также низкое качество получаемых сеток.

1.5.3. Быстрая модификация алгоритма

В этом разделе мы модифицируем Алгоритм 1.1, и покажем, что при определённых условиях можно гарантировать работу алгоритма без полного перебора, сделаем оценки сложности получившегося алгоритма.

Введём некоторые дополнительные обозначения, которые мы будем использовать в этом разделе. Пусть $\mathcal{F}$ – замкнутый фронт для многоугольника $\mathcal{P}$. Обозначим через $d_{\min}$ и $d_{\max}$ – минимальное и максимальное значение длин отрезков фронта. Множество вершин фронта обозначим как $\{V_i\}$. Теперь рассмотрим все такие пары вершин (V_i, V_j), для которых отрезок V_iV_j

лежит внутри $\mathcal{P}$. Минимальное расстояние между вершинами среди всех таких пар обозначим как d_{int}.

Заметим, что если отрезок V_iV_j пересекался с фронтом, или лежал за пределами $\mathcal{P}$, то при дальнейшем продвижении фронта с помощью Алгоритма 1.1 этот отрезок никогда не станет отрезком фронта.

Зафиксируем некоторое $d > 0$. Предположим, что в каждый момент времени $d_{\min} \geq d$, $d_{\max} < 2d$, $d_{\text{int}} \geq d$. Покажем, что в этом случае мы можем продвигать фронт, поддерживая это свойство. В частности, в получившейся сетке стороны всех треугольников также будут лежать в пределах от d до $2d$.

Лемма 1.5.1. *Пусть задано некоторое значение $d > 0$, и многоугольник $\mathcal{P}$ с фронтом $\mathcal{F}$ с вершинами $\{V_i\}$, $d_{\min} \geq d$, $d_{\max} < 2d$, $d_{\text{int}} \geq d$, отрезок $\overrightarrow{AB}$ – минимальный по длине отрезок фронта $\mathcal{P}$. Тогда можно так выбрать точку X в положительной полуплоскости относительно $\overrightarrow{AB}$, что $d \leq AX < 2d$, $d \leq BX < 2d$, $\triangle ABX$ не пересекает $\mathcal{F}$, и после продвижения фронта будет выполнено $d_{\min} \geq d$, $d_{\max} < 2d$, $d_{\text{int}} \geq d$.*

Доказательство. В силу громоздкости доказательства, приведём только основную его идею, оставляя некоторые утверждения без строгого доказательства. Из условия $d_{\text{int}} \geq d$ следует, что внутри открытых d-окрестностей вершин A и B нет других вершин из фронта.

Сначала рассмотрим случай $|AB| \leq \sqrt{3}d$. В этом случае можно построить такую точку C в положительной полуплоскости относительно $\overrightarrow{AB}$, что $AC = BC = d$. И углы при основании треугольника будут не меньше $\frac{\pi}{6}$.

Если $|AB| > \sqrt{3}d$, то построим $\triangle ABC$ с углами при основании $\frac{\pi}{6}$. Тогда $AC = BC > d$, а так как $|AB| < 2d$, то $AC = BC < 2d$.

Можно показать, что при ограничениях $d_{\min} \geq d$, $d_{\max} < 2d$, $d_{\text{int}} \geq d$ $\triangle ABC$ либо пересекается с отрезком, выходящим из A или B, либо не пересекается с $\mathcal{F}$. Рассмотрим первый случай, пусть для определённости тре-

угольник пересекается с отрезком AX. Можно показать, что внутри $\triangle ABX$ нет вершин из $\mathcal{F}$, и BX не пересекает $\mathcal{F}$. в этом случае треугольник $\triangle ABX$, у которого как минимум две стороны из $\mathcal{F}$, удовлетворяет условиям леммы.

Во втором случае проверим, что отрезки CV_i, лежащие в $\mathcal{P}$, не меньше d. Если это так, то положим $X = C$ и условие леммы будет выполнено, при этом у $\triangle ABX$ только одна сторона лежит в $\mathcal{F}$.

Иначе, выберем ближайшую к C такую точку X. Можно показать, что внутри $\triangle ABX$ нет вершин фронта, и $\triangle ABX$ не пересекается никаким отрезком из фронта в силу ограничения на максимальную длину отрезков фронта. А значит $\triangle ABX$ удовлетворяет условиям леммы, и все углы треугольника не меньше $\frac{\pi}{6}$.

Отметим, что у построенного нами треугольника либо все углы не меньше $\frac{\pi}{6}$, либо хотя бы две его стороны лежали в $\mathcal{F}$. □

Модифицируем Алгоритм 1.1 с учётом Леммы 1.5.1. В строке 4 выберем s в зависимости от длины $|AB|$. В строке 6 возьмём $R = 2d$, $h = 0$, $r = d$. В строке 9 заменим проверку на проверку из Леммы 1.5.1: если $\{P_i^k : P_i^k \in S_r(C), CP_i^k \subset \mathcal{P}\} = \varnothing$, то добавить C к Σ_R^k. А строчку 18 удалим из алгоритма.

Теорема 1.2. *Пусть $\mathcal{P}$ – многоугольник с фронтом $\mathcal{F}$, для которого $d_{\min} > \frac{1}{2}d_{\max}$ и $d_{\text{int}} > \frac{1}{2}d_{\max}$. Тогда модифицированный алгоритм построит сетку для $\mathcal{P}$ за конечное время.*

Доказательство. Возьмём $d = \min(d_{\min}, d_{\text{int}})$. Тогда в начальном фронте $d_{\min} \geq d$, $d_{\max} < 2d$, $d_{\text{int}} \geq d$. Лемма 1.5.1 гарантирует существование подходящего треугольника в локальном переборе, и после продвижения фронта условие будет сохраняться. На каждом шаге добавляется треугольник со сторонами не меньше d и меньше $2d$.

Покажем, что количество таких треугольников внутри $\mathcal{P}$ ограничено. Так как в этот раз у нас нет ограничения на минимальное попарное расстояние между вершинами сетки, то рассуждения аналогичные рассуждениям из раздела 1.4.2 здесь не работают. В доказательстве Леммы 1.5.1 мы сделали замечание, что у нового треугольника может быть угол меньше $\frac{\pi}{6}$ только в том случае, когда две его стороны уже были в фронте. А значит при добавлении такого треугольника в сетку количество отрезков в фронте уменьшилось хотя бы на один. Каждый раз, когда количество отрезков в фронте увеличивается на один, мы добавляем треугольник с углами не меньше $\frac{\pi}{6}$. Мысленно покрасим такие треугольники в синий цвет, а треугольники с углами меньше $\frac{\pi}{6}$ в красный. Тогда количество красных треугольников, n_r, не может превышать количество синих, n_b, больше чем на количество отрезков в начальном фронте, n_f, то есть $n_r \leq n_b + n_f$.

Оценим площадь одного синего треугольника. Все его углы не меньше $\frac{\pi}{6}$, следовательно, они не больше $\frac{2\pi}{3}$. То есть синус любого его угла не меньше $\frac{1}{2}$, тогда площадь треугольника будет не меньше $\frac{1}{4}d^2$.

Тогда $n_b \leq \frac{4S}{d^2}$, где S – площадь $\mathcal{P}$. Пусть p – длина фронта, тогда $n_f \leq \frac{p}{d}$. Сделаем оценку на общее количество треугольников: $n \leq n_r + n_b \leq \frac{8S}{d^2} + \frac{p}{d}$. $\square$

Оценим вычислительную сложность нового алгоритма. По сравнению с первоначальной версией, в этом алгоритме используется только локальный перебор, а длины отрезков в фронте ограничены сверху. Поэтому все отрезки фронта будут лежать на двух последних уровнях четверичного дерева поиска. Из формулы (1.5) при $\delta = d$, $R = 2d$ получим, что $N(\mathcal{F}_R^k) \leq e_n + e_{n-1} < 12 \cdot 2^2 \cdot 2 + 48\sqrt{2} \cdot 2 \cdot (1+2) + 96 \cdot (1+4) + 12 \cdot 2 \cdot 2 + 24\sqrt{2} \cdot (1+2) + 2 \cdot 2 < C_0$. То есть $N(\mathcal{F}_R^k)$ ограничено константной, не зависящей ни от $\mathcal{P}$, ни от d. Таким образом выполнение одной итерации внешнего цикла занимает $C_1 + C_2 \log_2 \frac{H}{d}$ операций. А значит даже в худшем случае сложность всего

алгоритма составит величину пропорциональную $\left(\frac{8S}{d^2} + \frac{p}{d}\right) \log \frac{H}{d}$.

Мы показали, что новая модификация Алгоритма 1.1 работает за время пропорциональное $N \log N$, где N – количество треугольников. При этом длины сторон построенных треугольников лежат в пределах от d до $2d$, где d минимальная длина стороны или внутренней диагонали исходного многоугольника. Есть оценка на количество треугольников $N \leq \frac{8S}{d^2} + \frac{P}{d}$, где S – площадь многоугольника, P – периметр многоугольника. К сожалению, мы не можем сделать никаких дополнительных оценок на наихудшее качество треугольников в полученной сетке.

Новый алгоритм применим только к определённому классу областей и не позволяет контролировать желаемый размер элементов при построении. Тем не менее эта модификация имеет важный практический интерес в силу хорошей оценки на время работы даже в худшем случае. Возможность работы в локальном фронте может быть использована, например, для построения параллельной версии алгоритма, или для использования алгоритма на криволинейных поверхностях, о чём подробнее будет рассказано в Главе 2.

1.6. Улучшение качества полученной сетки

В этом разделе мы обсудим способы улучшения качества треугольной сетки. Будем использовать только операцию сдвига внутренних вершин, зафиксировав при этом связи между вершинами, и внешние вершины, лежащие на границе области.

Для измерения качества треугольника $\triangle ABC$ будем использовать следующую величину [28]:

$$q = C \frac{S}{a^2 + b^2 + c^2}, \tag{1.6}$$

где S – ориентированная площадь треугольника, $S = \frac{1}{2}[ABC]$, a, b и c – длины сторон треугольника, $C = 4\sqrt{3}$. Тогда для невырожденных треугольников

будет выполнено $0 < q \leq 1$. Если треугольник вырожден, то $q = 0$, а если он вывернут, то есть $[ABC] < 0$, то $q < 0$.

Пусть $\mathcal{V}$ – множество всех вершин в сетке $\mathcal{T}$, $\mathcal{V}_B$ – вершины лежащие на границе, неподвижные вершины, а $\mathcal{V}_I$ – внутренние вершины. Для каждой внутренней вершины $V \in \mathcal{V}_I$ определим множество её соседей $\Sigma(V)$. Количество соседей обозначим как $N(\Sigma(V))$.

Простейший способ улучшения сетки заключается в сдвиге каждой внутренней вершины в центр масс её соседей:

$$V_{\text{mass}} = \frac{1}{N(\Sigma(V))} \sum_{X \in \Sigma(V)} X.$$

В приведённой формуле все вершины имеют одинаковый вес. После этого все вершины одновременно сдвигаются в новые посчитанные значения.

Надо следить, чтобы сетка оставалась конформной, и треугольники не выворачивались. Для этого можно проверить, что в новой сетке для каждого треугольника $\triangle ABC$ было выполнено $[ABC] > 0$. Если какой-то из треугольников вывернулся, то нужно вернуться на шаг назад, к предыдущему конформному состоянию. Далее можно попробовать опять провести сдвиг вершин, но на этот раз сдвигать вершины не сразу в центр масс, а только на какую-то часть от этого расстояния: $V_{\text{new}} = kV_{\text{mass}} + (1 - k)V$, $0 \leq k \leq 1$. Можно каждый раз уменьшать k.

На практике этот подход хорошо работает для слабо возмущённых квазиравномерных сеток, но не даёт значительного улучшения качества сеток в случае, когда размер элементов сильно меняется. В последнем случае будем использовать метод улучшения сетки, основанный на псевдоминимизации некоторого функционала сетки, описанный в следующем разделе.

1.6.1. Псевдоминимизация функционала качества сетки

Рассмотрим следующий функционал:

$$F(\mathcal{V}_I) = \sum_{T \in \mathcal{T}} F_T(\mathcal{V}_I),$$

где функция F_T для треугольника зависит от его качества q:

$$F_T(\mathcal{V}_I) = q_T^{-n}.$$

С помощью параметра n можно регулировать насколько больше будет "штраф" за плохое качество треугольников. Эта функция растёт с уменьшением качества треугольников, и имеет разрыв при $q = 0$. Немного подправим функцию q так, чтобы не было разрыва [11, 47]. Для этого заменим в формуле S на

$$h(S) = \frac{1}{2}\left(S + \sqrt{S^2 + 4\delta^2}\right),$$

где δ некоторый параметр.

$$q^* = C\frac{h(S)}{a^2 + b^2 + c^2}.$$

Теперь функция q^* всегда положительная, и функция $F_T^*(\mathcal{V}_I) = {q_T^*}^{-n}$ не будет иметь разрывов. Это позволяет использовать её в том числе и для сеток с вывернутыми треугольниками. Будем так двигать внутренние вершины, чтобы минимизировать функционал, при этом будем постепенно уменьшать значение параметра δ. Большие значения для δ нужны в первую очередь для распутывания сетки с помощью функционала, в то время как при почти нулевом δ функционал используется для улучшения качества треугольников.

Задача минимизации функционала является довольно сложной задачей. Поэтому мы не будем его точно минимизировать, а будем использовать более простой подход. Каждую внутреннюю вершину будем сдвигать отдельно, зафиксировав при этом все остальные. В этом случае $F(\mathcal{V}_I) = F(x, y)$, где

(x, y) – координаты подвижной вершины. Будем сдвигать её в направлении, противоположном градиенту функционала. Значение градиента в точке можно достаточно просто вычислить аналитически.

Выпишем градиент $F_T(x, y)$ для треугольника с координатами (x, y), (x_1, y_1), и (x_2, y_2). В этом случае имеем:

$$S = \frac{1}{2}\begin{vmatrix} x - x_1 & y - y_1 \\ x - x_2 & y - y_2 \end{vmatrix} = \frac{1}{2}\left(x_2 y_1 - x_1 y_2 + x(y_2 - y_1) + y(x_1 - x_2)\right),$$

$$\frac{\partial S}{\partial x} = \frac{y_2 - y_1}{2}, \quad \frac{\partial S}{\partial y} = \frac{x_1 - x_2}{2},$$

$$H = \frac{1}{2}\left(S + \sqrt{S^2 + 4\delta^2}\right),$$

$$\frac{\partial H}{\partial x} = \frac{y_2 - y_1}{4} + \frac{S(y_2 - y_1)}{4\sqrt{S^2 + 4\delta^2}}, \quad \frac{\partial H}{\partial y} = \frac{x_1 - x_2}{4} + \frac{S(x_1 - x_2)}{4\sqrt{S^2 + 4\delta^2}},$$

$$L = a^2 + b^2 + c^2 =$$

$$= 2x^2 + 2x_1^2 + 2x_2^2 - 2xx_1 - 2xx_2 - 2x_1x_2 + 2y^2 + 2y_1^2 + 2y_2^2 - 2yy_1 - 2yy_2 - 2y_1y_2,$$

$$\frac{\partial L}{\partial x} = 4x - 2(x_1 + x_2), \quad \frac{\partial L}{\partial y} = 4y - 2(y_1 + y_2),$$

$$q^* = 4\sqrt{3}\frac{H}{L}, \quad f = \frac{1}{q^*} = \frac{L}{4\sqrt{3}H}, \quad \frac{\partial f}{\partial x} = \frac{\frac{\partial L}{\partial x}H - \frac{\partial H}{\partial x}L}{4\sqrt{3}H^2}, \quad \frac{\partial f}{\partial y} = \frac{\frac{\partial L}{\partial y}H - \frac{\partial H}{\partial y}L}{4\sqrt{3}H^2},$$

$$F_T = f^n, \quad \frac{\partial F_T}{\partial x} = nf^{n-1}\frac{\partial f}{\partial x}, \quad \frac{\partial F_T}{\partial y} = nf^{n-1}\frac{\partial f}{\partial y}.$$

Для плохих и вырожденных треугольников значения q^* будут находиться вблизи нуля, и значения F^* будут большие, а значит и модуль градиента функционала тоже может быть большим. Если сдвинуть вершину на большое расстояние, то могут появиться новые вырожденные треугольники и итерационный процесс не будет сходиться. Чтобы избежать этого и улучшить сходимость, предлагается уменьшить влияние больших значений модуля функционала.

Введём функцию, уменьшающую эффект от сдвига на большие расстояния ($r > r_0$) за счёт логарифмической зависимости:

$$f(r) = \begin{cases} r, & \text{если } r \le r_0; \\ r_0(1 + \log(r/r_0)), & \text{если } r > r_0. \end{cases}$$

Алгоритм, используемый для псевдоминимизации функционала представлен в Алгоритме 1.4.

Алгоритм 1.4 Псевдоминимизации функционала качества сетки

1: Выбрать начальное значение δ.
2: **начало цикла**
3: **для всех** $V \in \mathcal{V}_I$ **начало цикла**
4: Вычислить ∇F^*.
5: Вычислить расстояние $d = c \cdot f(|\nabla F^*|)$.
6: Сдвинуть V на расстояние d в сторону, противоположную направлению ∇F^*.
7: **конец цикла**
8: Уменьшить δ.
9: **конец цикла**

В программной реализации этого алгоритма в пакете Ani2D используются следующие параметры. $F_T^* = {q_T^*}^{-8}$, $N = 100$ – число итераций. Начальное значение $\delta = h^2/100$, где h – средняя длина ребра во всей области. На каждой итерации значение δ линейно уменьшается до значения $\delta = h^2/1000$. Параметры для вычисления $d = c \cdot f(|\nabla F^*|)$ используются следующие: $c = 0.25h^2/N$, $r_0 = 1/h$.

Результаты работы этого метода улучшения сетки будут показаны в разделе 1.7.3.

1.7. Результаты экспериментов

Приведём ряд приложений основного Алгоритма 1.1, предложенного в этой главе, и реализованного в библиотеке aniAFT из пакетов программ Ani2D и Ani3D. В первой части мы экспериментально оценим среднюю сложность работы алгоритма, во второй части приведём результаты использования разных методов задания желаемого шага сетки, в третьей части покажем результат работы предложенного в разделе 1.6 метода улучшения сетки для области с нерегулярным фронтом.

1.7.1. Скорость работы алгоритма продвигаемого фронта

Экспериментально измерим скорость работы алгоритма продвигаемого фронта на плоскости. В качестве области возьмём единичный квадрат, и будем строить на нём неструктурированные квазиравномерные сетки с шагом h. Уменьшая шаг сетки h, будем следить за количеством треугольников в сетке, N_T, и временем построения сетки, t. Результаты экспериментов представлены в Таблице 1.1.

Таблица 1.1. Скорость работы алгоритма продвигаемого фронта.

h	N_T	t, сек	$N_T/1$ сек	$t/(N_T \log N_T)$, мкс
0.01	19304	0.29	66566	1.52
0.005	128334	2.08	61699	1.38
0.0025	512956	8.96	57250	1.33
0.00125	2019486	37.67	53610	1.28

Из результатов расчётов видно, что время работы пропорционально величине $N_T \log N_T$. В таблицу также включены столбцы со скоростью построения (треугольники в секунду), и оценкой коэффициента перед $N_T \log N_T$.

1.7.2. Различные способы выбора шага сетки

В этом разделе мы проследим, как выбор локального шага сетки, или параметра s, при построении нового треугольника в Алгоритме 1.1 влияет на сетку и на качество сетки.

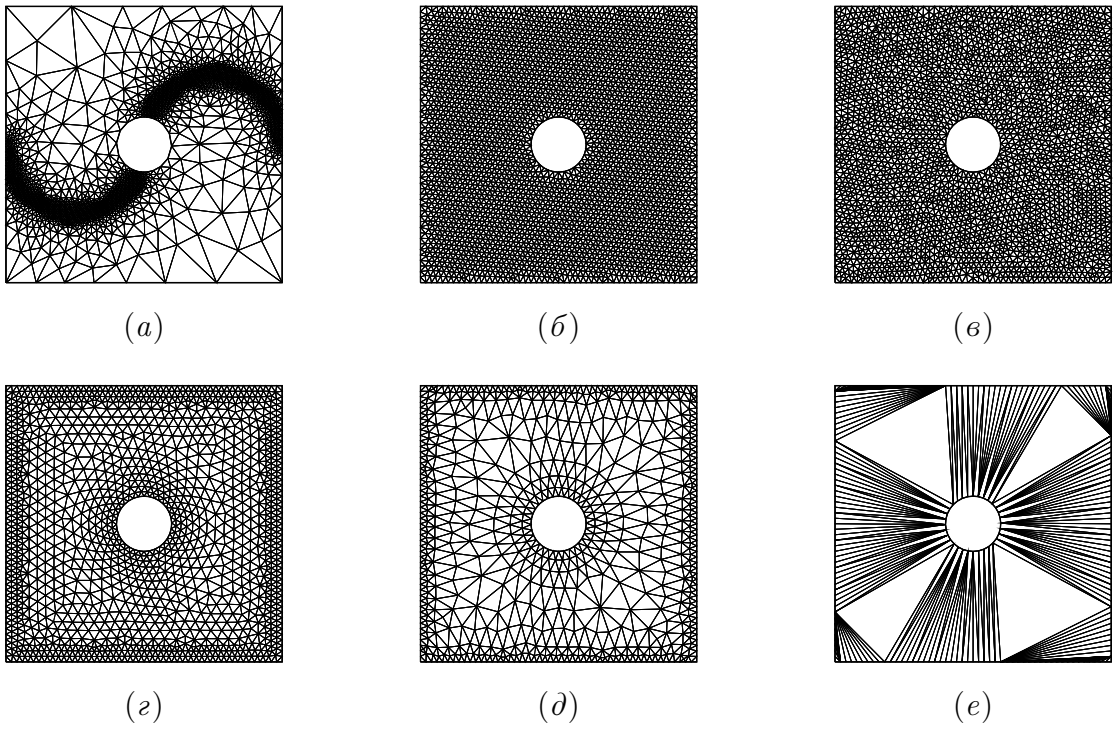
Рассмотрим область в виде единичного квадрата с вырезанным в центре кругом с радиусом 0.1. Реализованные в пакетах Ani2D и Ani3D алгоритмы позволяют задавать желаемый размер элементов сетки с помощью скалярной функции. С помощью этой же функции можно построить дискретную границу области, заданной аналитически. Подробнее об этом будет рассказано в разделе 2.1 следующей главы.

При построении сеток использовалось два варианта скалярных функций, отвечающих за желаемый размер треугольников. Первая уменьшалась вблизи двух полуокружностей, напоминающих по форме график функции $\sin(x)$. Вторая функция была тождественно равна константе на всей области. Полученные с их помощью сетки представлены на Рис. 1.7, *а* и 1.7, *б* соответственно.

Далее протестируем автоматический выбор шага при заданной дискретной границе области. В качестве дискретной границы возьмём границу, полученную при использовании дискретизации границы с постоянной функцией желаемого шага сетки. Протестируем работу алгоритма с автоматическим увеличением шага при разных значениях $\gamma = 1, 1.05, 1.25, 20$. Полученные сетки представлены на Рис 1.7. Выбор большой скорости увеличения шага сетки приводит на практике к очень грубым сеткам с минимальным количеством новых точек внутри области (см. Рис. 1.7, *е*).

В Таблице 1.2 собрана информация о количестве треугольников для построенных сеток, и о минимальном качестве треугольников в получившихся сетках. Качество треугольников вычисляется по формуле (1.6). Отметим, что

во всех рассмотренных примерах, кроме последнего, наихудшее качество не меньше 0.5.



$(а)$ $(б)$ $(в)$

$(г)$ $(д)$ $(е)$

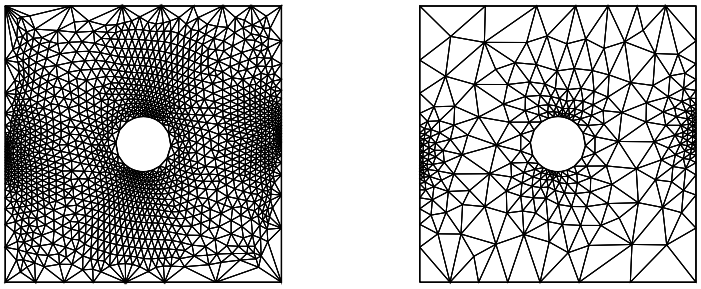
Рис. 1.7. Разные способы выбора шага сетки: $(а)$ заданная пользователем нетривиальная функция, $(б)$ заданная пользователем константная функция; $(в)$–$(е)$ автоматический выбор с $\gamma = 1, 1.05, 1.25, 20$, соответственно.

Таблица 1.2. Количество треугольников и наихудшее качество треугольников при разных способах выбора шага сетки.

	польз.	const	$\gamma = 1$	$\gamma = 1.05$	$\gamma = 1.25$	$\gamma = 20$
N_T	3636	6212	5386	2596	1730	232
$q_{\min}$	0.665	0.792	0.586	0.848	0.713	$0.448 \cdot 10^{-2}$

В дополнение к рассмотренным сеткам с равномерным следом на границе проверим алгоритм для неравномерной граничной дискретизации. В качестве такой дискретизации выберем след триангуляции, полученной с помощью непостоянной функции желаемого размера сетки (см. Рис. 1.7, $а$). Граничная

дискретизация имеет непостоянный, но плавно меняющийся шаг. Результаты автоматического выбора шага с параметрами $\gamma = 1.05$ и $\gamma = 1.25$ показаны на Рис. 1.8. Количество треугольников в получившихся сетках составило 2904 и 550 соответственно. Наихудшее качество – 0.280 и 0.645 соответственно.



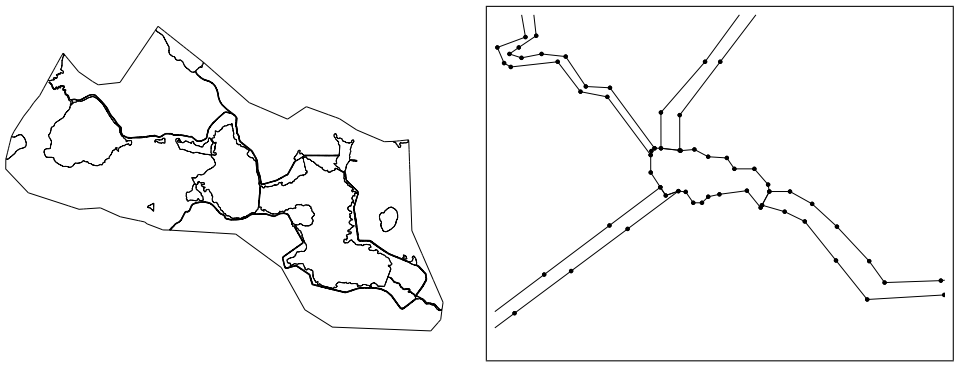
($а$) ($б$)

Рис. 1.8. Автоматический выбор шага сетки при неравномерной дискретизации границы: ($а$) $\gamma = 1, 1.05$, ($б$) $\gamma = 1, 1.25$.

В следующем разделе мы рассмотрим использование автоматического выбора шага для начального фронта с сильно неравномерным шагом на границе.

1.7.3. Сложный начальный фронт

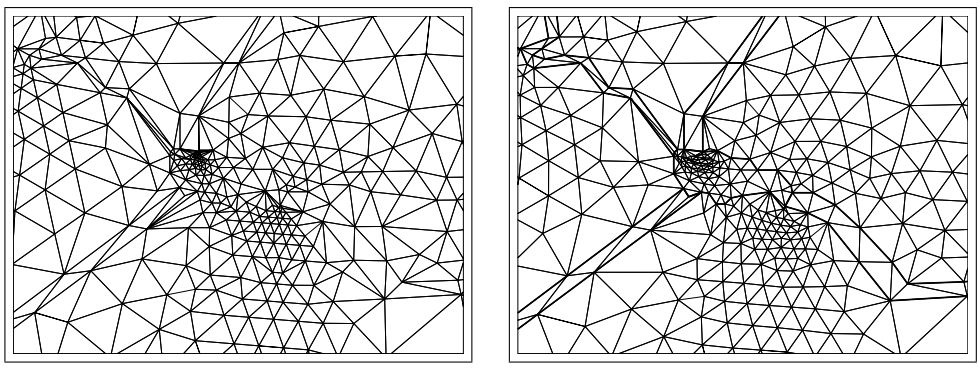
Рассмотрим пример достаточно сложного и нерегулярного начального фронта. На Рис. 1.9, $а$ показан весь начальный фронт, а на Рис. 1.9, $б$ увеличена малая подобласть всей области. Начальный фронт предоставлен И. В. Капыриным и представляет собой границы Теченского каскада водоёмов и окружающих каналов. Отметим, что шаг дискретизации на границе меняется неравномерно, и вообще говоря нужна дополнительная предобработка фронта для получения высококачественной сетки. Тем не менее попробуем построить треугольную сетку с помощью автоматического выбора шага сетки. Возьмём $\gamma = 1.25$.



($а$) ($б$)

Рис. 1.9. Дискретная границы области. ($а$) вся область целиком, ($б$) увеличенная часть.

Начальный фронт состоит из 2743 вершин и 5187 направленных отрезков. Напомним, что внутренние разрезы порождают пару разнонаправленных отрезков. После построения сетки с помощью алгоритма продвигаемого фронта была получена сетка из 14807 вершин и 29279 треугольников. Часть сетки, соответствующая Рис. 1.9, $б$ показана на Рис. 1.10, $а$, на Рис. 1.10, $б$ показана сетка после применения алгоритма улучшения качества сетки, описанного в разделе 1.6.



($а$) ($б$)

Рис. 1.10. Треугольная сетка. ($а$) после алгоритма продвигаемого фронта, ($б$) после алгоритма улучшения качества сетки.

В Таблице 1.3 приведено распределение качества треугольников в сетке до и после улучшения сетки.

Таблица 1.3. Распределение качества треугольников до (Д) и после (П) улучшения треугольной сетки.

	$q_{\min}$	< 0.1	0.1–0.3	0.3–0.5	0.5–0.7	0.7–0.9	0.9–1.0
Д	$3.6 \cdot 10^{-3}$	27	461	802	1334	6453	20202
П	$5.2 \cdot 10^{-2}$	5	339	477	794	5234	22430

Из таблицы следует, что качество наихудшего элемента увеличилось на порядок, и количество треугольников, для которых $q < 0.1$, теперь равно пяти, при общем количестве треугольников около 30 тысяч. Отметим, что качество наихудших треугольников в этом примере ограничено неравномерным начальным фронтом.

1.8. Выводы к первой главе

Разработан, реализован и протестирован алгоритм построения конформных согласованных с заданной границей сеток для произвольных многокомпонентных многосвязных многоугольных областей. Доказана конечность числа операций и продемонстрирована устойчивость алгоритма к вычислительным погрешностям при реализации алгоритма на ЭВМ. Получены оценки в среднем и в худшем случае на количество операций алгоритма продвигаемого фронта. Предложены две модификации алгоритма с лучшими оценками на сложность работы, но обладающие недостатками при их практическом использовании. Разработан, реализован и протестирован алгоритм улучшения качества сетки на основе псевдоминимизации функционала качества сетки. Экспериментально подтверждена оценка на сложность алгоритма продвигаемого фронта в среднем. Представлены некоторые примеры работы алгоритмов, реализованных автором в библиотеке aniAFT из пакета Ani3D.

Глава 2

Построение поверхностных сеток

Технология построения треугольных *поверхностных* сеток, то есть сеток на границе трёхмерных областей, необходима для дальнейшего построения неструктурированных тетраэдральных сеток. Также поверхностные треугольные сетки могут использоваться при решении квазидвумерных задач на криволинейных поверхностях. Примерами таких задач могут служить задачи деформации тонкостенных конструкций, или решение уравнений мелкой воды на поверхности сферы или геоида.

В этой главе мы покажем, что двумерный алгоритм продвигаемого фронта может быть расширен на случай криволинейных поверхностей, обсудим способы построения дискретных криволинейных границ, способы взаимодействия с системами САПР для получения необходимой информации о сложных областях. Будет проведён анализ конечности и сложности алгоритмов, представлено несколько примеров работы реализованных алгоритмов.

Поверхностью назовём двумерное многообразие в трёхмерном пространстве. *Простой* поверхностью назовём такую поверхность, которая является образом гомеоморфного отображения единичного квадрата.

Кривой на поверхности или в пространстве назовём одномерное многообразие на поверхности или в пространстве, соответственно. *Простой* кривой назовём кривую, являющуюся образом гомеорфизма единичного отрезка.

Будем по возможности использовать обозначения из раздела 1.1 и для поверхностей. В качестве векторного произведения двух направленных отрезков на поверхности будем использовать следующую величину: $[ABC] = \left(\overrightarrow{CA} \times \overrightarrow{CB}, \mathbf{n}_C\right)$, где $\mathbf{n}_C$ – внешняя единичная нормаль к поверхности в точке C, а $(\cdot, \cdot)$ – скалярное умножение. Отметим, что для плоскости новое обозначе-

ние в точности совпадёт с определением из Главы 1. Также можно отметить, что такая величина соответствует значению $[A'B'C]$ в касательной плоскости, где A' и B' – проекции A и B соответственно.

2.1. Представление поверхности

Сначала рассмотрим простую поверхность, которую можно представить, как двумерное многообразие – образ гомеоморфного отображения единичного квадрата. То есть должна быть определена такая параметризующая функция, которая переводит точку из квадрата в параметрическом пространстве в точку на поверхности.

Построение такого гомеоморфизма для произвольной поверхности является трудной задачей. На практике можно построить гомеоморфизм из некоторой ограниченной области в параметрическом пространстве $\Omega \subset \mathbb{R}^2$.

Разбиение границы Ω при гомеоморфизме порождает разбиение границы простой поверхности, которое можно рассматривать как дискретную границу поверхности. Используя эту границу как фронт, можно применить алгоритм продвигаемого фронта, локально рассматривая поверхность как плоскость.

Сложная поверхность, как правило, может быть разрезана на несколько простых поверхностей. Зафиксировав дискретизацию на разрезах, можно для каждой простой поверхности построить согласованную с этой дискретизацией триангуляцию, тогда общая треугольная сетка для всей сложной поверхности будет конформной.

Составим алгоритм действий на основе описанной выше идеи. Вся поверхность разбивается на несколько простых поверхностей Σ_i, будем называть их *криволинейными гранями*. Каждая криволинейная грань параметризуется гладкой функцией.

$$\Sigma_i = \left\{(x, y, z) | (x, y, z) = f_i(u, v), (u, v) \in \Omega_i \subset \mathbb{R}^2\right\}.$$

Под гладкостью функции будем подразумевать существование и непрерывность производных по u, v: $f_i \in C^1(\Omega_i)$.

Границу криволинейной грани разобьём на несколько простых кривых, Γ_k, которые будем называть *криволинейными рёбрами*. Введём параметризацию на криволинейных рёбрах.

$$\Gamma_k = \left\{(x, y, z) | (x, y, z) = g_k(t), t \in [t_k^0, t_k^1]\right\}.$$

Нам также понадобится отображения u_{ik}, v_{ik} из параметрического пространства кривой в параметрическое пространство поверхности:

$$g_k(t) = f_i(u_{ik}(t), v_{ik}(t)), t \in [t_k^0, t_k^1].$$

Отметим, что для произвольных f_i, g_k вычисление этих отображений может оказаться сложной задачей. Поэтому на практике такой подход применим для достаточно простых областей.

При реализации этого метода на ЭВМ в пакете программ Ani3D был использован другой подход. Пусть у нас задана параметризация криволинейных рёбер в параметрическом пространстве поверхности отдельно для каждой криволинейной грани. Пусть грань параметризована функцией $f_i(u, v)$, ребро Γ_k можно рассматривать как кривую в параметрическом пространстве (u, v). Например, если мы используем отображение $u \to (u, v_{ik}(u))$, тогда в $\mathbb{R}^3$ кривая будет параметризована следующим образом:

$$\Gamma_k = \{(x, y, z) | (x, y, z) = f_i(u, v_{ik}(u)), u \in [u_i^0, u_i^1]\}.$$

Такие параметризации кривой Γ_k могут быть разными для разных криволинейных граней. На практике можно допускать незначительное отклонение в пределах допустимой погрешности, что упрощает построение параметризаций и расширяет класс областей, для которых такой подход применим.

Для построения поверхностных сеток на криволинейных гранях сначала строится дискретизация криволинейных рёбер. Рассмотрим параметризованную простую кривую:

$$(x, y, z) = g(u) = f(u, v(u)), u \in [u_0, u_1].$$

Будем строить на ней точки:

$$P_i = g(\tau_i), \quad i = 0, 1, \ldots, n.$$

Значения параметров $\tau_i \in [u_0, u_1]$ выбираются с помощью метода бисекции в соответствии с желаемым расстоянием между точками P_i.

После построения дискретизации границы криволинейной грани, мы можем применить аналог алгоритма продвигаемого фронта для построения треугольной сетки. Так как соседние грани имеют одну и ту же дискретизацию их общего ребра, и построенные на них треугольные сетки согласованы с этой дискретизацией, то составленная из них общая сетка будет конформной.

Выпишем основные этапы построения поверхностной сетки. Предположим, что поверхность разбита на несколько непересекающихся простых поверхностей – криволинейных граней. Криволинейная граница каждой грани разбита на несколько простых кривых – криволинейных рёбер. Будем рассматривать только конформные разбиения, то есть такие, в которых у двух соседних граней общие криволинейные рёбра.

На первом этапе построим дискретизацию для всех криволинейных рёбер. Для этого на кривой линии расставляются точки, и кривая приближается ломаной. Длина отрезков регулируется желаемым размером элементов сетки.

На втором этапе для каждой криволинейной грани составляется дискретная граница из имеющихся рёбер. Для точек на рёбрах восстанавливаются значения параметров (u, v) в параметрическом пространстве грани. С помощью алгоритма продвигаемого фронта строится поверхностная триангуля-

ция, согласованная с дискретной границей. Запишем Алгоритм 2.1.

Алгоритм 2.1 Построение поверхностной сетки

1: **для всех** криволинейных рёбер: **начало цикла**
2: Выбрать одну из параметризаций простой кривой.
3: Построить дискретизацию кривой с помощью метода бисекций.
4: **конец цикла**
5: **для всех** криволинейных граней: **начало цикла**
6: Составить дискретизацию границы из дискретизаций рёбер.
7: Применить метод продвигаемого фронта для построения сетки.
8: **конец цикла**
9: Объединить все построенные сетки в одну общую сетку.

Согласованность поверхностных сеток граней с дискретизациями рёбер гарантирует сохранение конформности общей сетки.

2.2. Взаимодействие с геометрическим ядром САПР

Информация о границе области может быть получена с помощью геометрического ядра САПР [42]. Большинство из существующих САПР предлагают интерфейс для взаимодействия со своим внутренним геометрическим ядром, позволяя получать информацию о топологии и геометрии области. У каждой системы САПР свой интерфейс для взаимодействия, и общих стандартов в этой области пока нет. Некоторые САПР с открытым исходным кодом хорошо документированы. Одним из примеров такой САПР может служить открытая система Open CASCADE Technology [69]. Однако, в большинстве случаев закрытость исходного кода и отсутствие документации по интерфейсам в свободном доступе усложняет разработку интерфейсов с САПР.

Перспективным направлением видится использование промежуточных

библиотек, предоставляющих общий унифицированный интерфейс для разработчика и поддерживающих взаимодействие с разными САПР. Во время разработки коммерческого пакета CUBIT Tool Suite, включающего в себя в том числе и собственное геометрическое ядро ACIS, разработчики стали добавлять новые интерфейсы к другим САПР. Для этого была создана специальная прослойка, предоставляющая общий интерфейс для взаимодействия с САПР. Позднее эта часть кода была вынесена из коммерческого проекта в виде отдельной открытой библиотеки Common Geometry Module (CGM) [72]. На тот момент были разработаны интерфейсы к геометрическим ядрам систем ACIS и Pro/ENGINEER. Открытость исходного кода позволила новым разработчикам использовать и совершенствовать эту библиотеку. В 2006 году автор диссертационной работы приступил с созданию интерфейса между геометрическим ядром Open CASCADE и библиотекой CGM. Базовая версия этого интерфейса позволила использовать CGM вместе с САПР Open CASCADE в пакете Ani3D. В настоящее время новой группой разработчиков ведётся дальнейшее развитие проекта CGM под новым именем, CGMA [73]. В этой версии основной упор делается на улучшение взаимодействия с геометрическим ядром Open CASCADE.

CGMA предлагает универсальный интерфейс для общения с геометрическими ядрами САПР. При этом вся информация делится на две части:

- Топологическая информация о модели: составные части модели и их топологические отношения между собой.

- Геометрическая информация: координаты, размеры, параметры, и параметризующие функции для кривых и поверхностей.

В большинстве ядер САПР используется граничное представление моделей (boundary representation, B-Rep или BREP). Такой метод представления ис-

пользуется и в библиотеке CGMA. Модель состоит из частей, образующих древовидную иерархию. Рассмотрим их от самых простых к сложным.

1. Точка, для которой заданы её координаты в пространстве (x, y, z).

2. Ребро – часть гладкой параметризованной кривой, ограниченная точками. Для ребра определена некоторая параметризация $t \leftrightarrow (x, y, z)$, а также ограничивающие её точки и значения параметра t в этих точках.

3. Петля – связный и замкнутый набор рёбер. Не несёт никакой геометрической информации и является чисто топологическим объектом.

4. Грань – часть гладкой параметризованной поверхности, ограниченная петлёй. Рёбра петли должны лежать на поверхности грани. Грань задана параметризацией поверхности $(u, v) \leftrightarrow (x, y, z)$.

5. Оболочка – связный и замкнутый набор граней. Также, как и петля, является чисто топологическим объектом.

6. Тело – часть пространства, ограниченная оболочкой.

7. Набор тел, представляющих модель в целом.

Как отмечалось ранее, для построения трёхмерной треугольной поверхностной сетки на каждой грани строится своя триангуляция, причём они конформно соединяются на рёбрах. Для этого сначала строится дискретизация рёбер. Рёбра аппроксимируются ломанными с заданным пользователем шагом в пространстве. После этого для каждой грани строится триангуляция с помощью алгоритма продвигаемого фронта [13]. В качестве начального фронта выступает дискретизация рёбер.

Особое внимание стоит уделить криволинейным поверхностям. Геометрические ядра САПР часто создают грани с периодичной параметризацией, а также с параметризацией, имеющей особые точки. Например, Open

CASCADE задаёт боковую поверхность цилиндра одной гранью с периодичной параметризацией, а боковую поверхность конуса – гранью с особой точкой в вершине конуса. Более того, поверхность шара задаётся одной гранью с периодичной параметризацией и двумя особыми точками в полюсах, сама грань при этом ограничивается лишь одним ребром, соединяющим два полюса. Алгоритм продвигаемого фронта должен быть соответствующим образом модифицирован, чтобы допускать такие особенности. Примеры работы алгоритма с геометрическими моделями САПР будут приведены в разделе 2.4.2.

2.3. Алгоритм продвигаемого фронта

Алгоритм 1.1 продвигаемого фронта естественно расширяется на случай криволинейных поверхностей. В этом разделе мы лишь отметим основные особенности поверхностного алгоритма продвигаемого фронта.

2.3.1. Геометрические особенности построения сеток на криволинейных поверхностях

В разделе 1.3.1 был предложен Алгоритм 1.2 для проверки пересечения треугольника с отрезком. В этом алгоритме использовалась только функция определения знака выражения $[ABC]$. Покажем, как получить знак выражения $[ABC]$ на поверхности. Построим точку $D = C + \mathbf{n}_C$, таким образом $\overrightarrow{CD}$ будет являться внешней нормалью к поверхности в точке C. По нашему определению $[ABC] = \left(\overrightarrow{CA} \times \overrightarrow{CB}, \overrightarrow{CD}\right) = \det \begin{vmatrix} x_A - x_C & y_A - y_C & z_A - z_C \\ x_B - x_C & y_B - y_C & z_B - z_C \\ x_D - x_C & y_D - y_C & z_D - z_C \end{vmatrix}$.

Знак определителя $D = aei - ceg + bfg - afh + cdh - bdi$ матрицы 3×3 вычисляется по алгоритму аналогичному Алгоритму 1.3. Абсолютную погрешность можно оценить величиной $r = \varepsilon(|a|+|b|+|c|+|d|+|e|+|f|+|g|+$

$|h|+|i|+|aei|+|ceg|+|bfg|+|afh|+|cdh|+|bdi|$). Отметим, что при использовании для координат типа вещественных чисел с одинарной точностью и вычислении определителя и промежуточных значений с двойной точностью запаса точности может не хватить для получения точного ответа. Для точных вычислений необходима тройная или четверная точность. Отметим, что некоторые современные архитектуры ЭВМ позволяют проводить вычисления с четверной точностью, и некоторые компиляторы могут использовать программную реализацию четверной точности на обычных процессорах.

Алгоритм 1.2 с учётом замечания о вычислении $[ABC]$ на поверхности можно применять для проверки пересечения треугольника и отрезка на поверхности. При этом на самом деле мы будем проверять пересечение проекции треугольника и проекции отрезка на касательную плоскость, поэтому этот метод применим только в локальных окрестностях, в которых поверхность не сильно отличается от касательной плоскости.

Для вычисления значения $[ABC]$ нужно уметь строить нормаль к поверхности. Пусть поверхность параметризована следующим образом: $(u, v) \mapsto (x, y, z)$, и в некоторой окрестности точки C существуют и непрерывны первые производные компонент x, y, z по параметрам u, v. Тогда направление нормали можно вычислить по следующей формуле:

$$\mathbf{n}_C = r\left(\frac{D(y,z)}{D(u,v)}, \frac{D(z,x)}{D(u,v)}, \frac{D(x,y)}{D(u,v)}\right), \tag{2.1}$$

где $\frac{D(y,z)}{D(u,v)} = \det\begin{vmatrix} y_u & y_v \\ z_u & z_v \end{vmatrix}$, и $\frac{D(z,x)}{D(u,v)}$, $\frac{D(x,y)}{D(u,v)}$ определяются аналогично, r – нормирующий множитель.

В формуле (2.1) все три компоненты могут одновременно обратиться в ноль. В этом случае для определения нормали можно выбрать производные вдоль других направлений в параметрическом пространстве. Условно запишем это так: $\mathbf{n}_C = r\left(\frac{D(y,z)}{D(u-v,u+v)}, \frac{D(z,x)}{D(u-v,u+v)}, \frac{D(x,y)}{D(u-v,u+v)}\right)$. В случае, если точка

C является особой точкой параметризации, можно отступить на небольшое расстояние от точки C, и вычислить нормаль в соседней точке.

Ещё одной важной операцией, которая легко выполняется на плоскости, является построение на заданном ребре AB равнобедренного треугольника с боковыми сторонами длины s. Для нахождения положения вершины C будем искать такую точку в параметрическом пространстве (u_C, v_C), что $AC = BC = s$ в трёхмерном пространстве. Пусть вершина A параметризована точкой (u_A, v_A). Будем искать параметрическую точку для C в виде $(u_C, v_C) = (u_A + r\cos\phi, v_A + r\sin\phi)$, где $r > 0$, $\phi \in [0, 2\pi]$. При фиксированном ϕ с помощью метода бисекции можно подобрать r так, что $AC = s$ с некоторой точностью. В зависимости от результата сравнения стороны BC с желаемой длиной s будем увеличивать или уменьшать с помощью метода бисекции ϕ до тех пор, пока не добьёмся $BC = s$ с некоторой точностью.

Использование двух вложенных методов бисекции приводит к большому количеству необходимых вычислений для построения равнобедренного треугольника. Поэтому на практике поверхностный метод продвигаемого фронта работает медленнее метода на плоскости. В следующем разделе мы проведём краткий анализ сложности поверхностной модификации метода.

2.3.2. Конечность работы алгоритма

При анализе алгоритма продвигаемого фронта нужно уделить внимание двум моментам: существованию подходящего треугольника на каждом шаге и ограниченности количества шагов. К сожалению, результат Теоремы 1.1 не переносится на случай поверхностного многоугольника. Более того, при достаточно большой кривизне поверхности и больших длинах отрезков в фронте подходящего треугольника на заданном ребре может не существовать.

На практике, при условии, что кривизна поверхности ограничена, и дли-

ны отрезков фронта достаточно малы, подходящий треугольник находится. Более того, он находится среди кандидатов в некоторой небольшой окрестности рассматриваемого ребра. При работе в локальной окрестности поверхность можно отобразить на касательную плоскость и использовать аналог Алгоритма 1.2 для поиска пересечений треугольника с локальным фронтом.

При использовании Алгоритма 1.1 мы можем поддерживать оценку снизу на минимальное попарное расстояние между вершинами поверхностной сетки. Рассуждения из раздела 1.4.2 применимы в этом случае с точностью до кривизны поверхности и Эйлеровой характеристики поверхности, которая для простых поверхностей равна Эйлеровой характеристике плоскости.

Анализ сложности алгоритма продвигаемого фронта из раздела 1.5.2 применим и в поверхностном случае. Несмотря на то, что конечность алгоритма не доказана в общем случае, на практике при работе с поверхностями с ограниченной кривизной и достаточно мелким шагом сетки сложность алгоритма в среднем пропорциональна $N \log N$, где N – общее количество треугольников.

При дополнительных ограничениях на кривизну поверхности и на шаг дискретизации рёбер границы криволинейной грани можно рассмотреть аналог модификации алгоритма продвигаемого фронта, предложенной в разделе 1.5.3. В предположении, что верна лемма, аналогичная Лемме 1.5.1, можно сформулировать следующую теорему.

Теорема 2.1. *Пусть Ω – криволинейная грань с границей Γ. Пусть главные кривизны поверхности не превышают k на всей Ω. Тогда существует такое $d = d(k)$, что при выборе $h < d$ и дискретизации Γ ломаной с шагом h предложенная модификация алгоритма построит конформную согласованную триангуляцию за конечное время.*

Доказательство теоремы следует из раздела 1.5.3. Время работы такой

модификации алгоритма будет пропорционально в худшем случае $N \log N$, где N – количество построенных треугольников.

2.4. Результаты экспериментов

В этом разделе мы проведём два эксперимента, в первом мы экспериментально оценим сложность алгоритма продвигаемого фронта для криволинейных поверхностей, заданных аналитически. А во втором примере покажем возможности использования интерфейса с геометрическим ядром САПР при построении поверхностной сетки.

2.4.1. Скорость работы алгоритма на криволинейной поверхности

Экспериментально измерим скорость работы алгоритма продвигаемого фронта на поверхности. В качестве области возьмём единичную сферу. Поверхность сферы разобъём на две полусферы, рассмотрим только верхнюю полусферу. Зададим параметризацию полусферы аналитически с помощью отображения: $(u, v) \mapsto (u, v, \sqrt{1 - u^2 - v^2})$, $u^2 + v^2 \leq 1$.

Будем строить квазиравномерную сетку с шагом h. Уменьшая шаг сетки h, будем следить за количеством треугольников в сетке, N_T, и временем построения сетки, t. Результаты экспериментов представлены в Таблице 2.1.

Таблица 2.1. Скорость работы алгоритма продвигаемого фронта на поверхности.

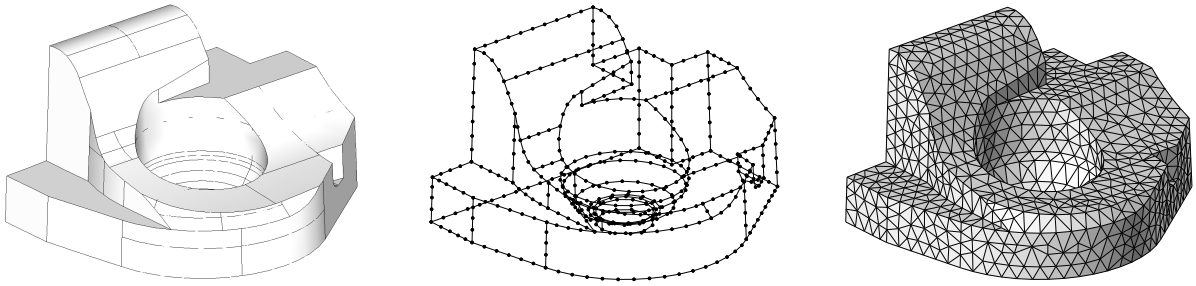
h	N_T	t, сек	$N_T/1$ сек	$t/(N_T \log N_T)$, мкс
0.1	1298	0.49	2649	121
0.05	5460	2.18	2505	107
0.025	22152	9.36	2367	97
0.0125	89730	40.34	2224	91

Из результатов расчётов видно, что время работы пропорционально величине $N_T \log N_T$. В таблицу также включены столбцы со скоростью построения (треугольники в секунду), и оценкой коэффициента перед $N_T \log N_T$.

2.4.2. Взаимодействие с геометрическим ядром САПР

Продемонстрируем совместную работу генератора поверхностных треугольных сеток и геометрического ядра САПР OpenCASCADE. В качестве примера рассмотрим модель `29_misc1` с сайта OpenCASCADE [69]. BREP-модель состоит из 63 вершин, 96 криволинейных рёбер, и 36 криволинейных граней (см. Рис. 2.1, *а*).

Сначала строится дискретизация всех криволинейных рёбер BREP-модели. Построенная дискретизация изображена на Рис. 2.1, *б*, она состоит из 461 вершины и 494 отрезков. Далее для каждой из 36 криволинейных граней запускается алгоритм продвигаемого фронта на поверхности. В зависимости от размера криволинейной грани количество построенных на них треугольников колеблется от 3 до 641 треугольника. В сумме в итоговой поверхностной триангуляции 2594 треугольника и 1299 вершин. Полученная поверхностная триангуляция изображена на Рис. 2.1, *в*.



(*а*) (*б*) (*в*)

Рис. 2.1. Модель, заданная в САПР: (*а*) BREP-модель, (*б*) дискретизация криволинейных рёбер, (*б*) поверхностная квазиравномерная треугольная сетка.

Отметим, что скорость работы алгоритма, использующего интерфейс

с ядром САПР, сильно зависит от скорости вычисления параметризующих функций внутри САПР. Так, в рассмотренном примере грань с наибольшим количеством треугольников была частью плоскости, и триангуляция для неё была построена за 2.11 секунд, в то время как для другой криволинейной грани было потрачено 4.54 секунд для треугольной сетки, состоящей всего из 25 треугольников. Общее время построения поверхностной сетки составило 14 секунд.

2.5. Выводы к второй главе

Разработана, реализована и протестирована технология построения поверхностных треугольных сеток для областей, заданных аналитически с помощью параметризующих функций, или заданных в САПР. Проведён краткий анализ влияния вычислительных погрешностей, конечности и сложности работы алгоритма. При определённых ограничениях на максимальную кривизну поверхности и на шаг дискретизации теоретически обоснована конечность и правильность работы алгоритма. Экспериментально подтверждена оценка на сложность алгоритма продвигаемого фронта на поверхности. Представлены примеры совместной работы рассмотренного алгоритма и геометрического ядра САПР при построении треугольной поверхностной сетки для области, заданной в САПР.

Глава 3

Построение трёхмерных сеток

В этой главе будет изучено расширение алгоритма продвигаемого фронта для трёхмерного пространства. Будут обсуждены вопросы его надёжности с точки зрения неточных вычислений и с точки зрения возможности продвижения фронта, конечность алгоритма и сложность работы алгоритма.

Алгоритм продвигаемого фронта не гарантирует построение тетраэдральной сетки для всей области, поэтому будет предложен второй, устойчивый алгоритм, основанный на идее П. Л. Джорджа [15]. Будет проведён краткий анализ конечности второго метода и В конце главы представлены примеры работы алгоритмов, реализованных в библиотеке aniAFT из пакета Ani3D.

Введём понятия и обозначения, аналогичные используемым в Главе 1.

Для четырёх точек A, B, C, D будем использовать следующую сокращённую запись:

$$[ABCD] = \det \begin{vmatrix} x_A - x_D & y_A - y_D & z_A - z_D \\ x_B - x_D & y_B - y_D & z_B - z_D \\ x_C - x_D & y_C - y_D & z_C - z_D \end{vmatrix}.$$

Ориентированным треугольником в пространстве назовём треугольник с некоторым зафиксированным с точностью до чётной перестановки порядком обхода его вершин.

Положительным полупространством относительно ориентированного треугольника ABC будем называть геометрическое место точек X, для которых $[ABCX] > 0$. Аналогично, *отрицательным* полупространством будем называть геометрическое место точек, для которых $[ABCX] < 0$.

Триангуляцией в пространстве назовём конечный набор треугольников, конформно связный через общие рёбра. При этом два соседних треугольника

могут лежать в одной плоскости. Если каждое ребро триангуляции принадлежит ровно двум треугольникам, то такую триангуляцию будем называть *замкнутой*. Триангуляцию, треугольники которой не пересекаются друг с другом, будем называть триангуляцией без самопересечений.

Многогранником в пространстве назовём ограниченную открытую часть пространства, граница которой состоит из одной или нескольких непересекающихся триангуляций без самопересечений. *Простым многогранником* назовём многогранник, ограниченный одной замкнутой триангуляцией. Вершинами многогранника будем называть вершины его триангуляций, а гранями многогранника – треугольники триангуляций.

По аналогии с рассуждениями из раздела 1.1 грани многогранника делятся на внешние и внутренние. Причём на внешних гранях можно ввести ориентацию так, что многогранник будет лежать в положительном полупространстве относительно ориентированного треугольника. Эту ориентацию будем называть положительной.

Фронтом в пространстве будем называть набор ориентированных треугольников без самопересечений. Каждому многограннику $\mathcal{P}$ можно поставить в соответствие фронт $\mathcal{F}(\mathcal{P})$. Для этого введём на внешних гранях $\mathcal{P}$ положительную ориентацию, а для каждой внутренней грани ABC поставим в соответствие пару противоположно ориентированных треугольников ABC и CBA. Совокупность всех этих ориентированных треугольников образует фронт $\mathcal{F}(\mathcal{P})$. Будем называть фронт $\mathcal{F}$ *замкнутым*, если существует такой многогранник $\mathcal{P}$, что $\mathcal{F} = \mathcal{F}(\mathcal{P})$.

Тетраэдральной сеткой $\mathcal{T}$ для многогранной области $\mathcal{P}$ назовём разбиение этой области на непересекающиеся тетраэдры. Будем называть сетку *конформной*, если любые её два элемента либо не имеют общих точек, либо имеют одну общую вершину, либо имеют одно общее целое ребро, либо одну общую целую грань. Будем говорить, что конформная сетка согласована

с границей $\mathcal{P}$, если каждая грань $\mathcal{P}$ является гранью какого-то тетраэдра в сетке. В частности, у внутренних граней многогранника будет ровно два соседних тетраэдра, а у внешних – ровно один.

3.1. Алгоритм продвигаемого фронта

Алгоритм 1.1 продвигаемого фронта расширяется на случай трёхмерного пространства. В этом разделе мы отметим основные особенности трёхмерного алгоритма продвигаемого фронта.

Отличительной особенностью трёхмерного пространства является существование таких фронтов, для которых не существует подходящих тетраэдров. По этой причине в алгоритм добавляется возможность помечать грани, для которых не нашлось подходящих тетраэдров, и пропускать их при дальнейшем переборе.

В конце работы мы получим сетку $\mathcal{T}$ для многогранной подобласти $\mathcal{P}_{\text{mesh}} \subseteq \mathcal{P}$ и фронт для неразбитой многогранной подобласти $\mathcal{P}_{\text{out}} = \mathcal{P} \setminus \mathcal{P}_{\text{mesh}}$.

Общая последовательность действий представлена в Алгоритме 3.1. Повторим основные действия, которые выполняются при продвижении фронта.

Из фронта выбирается непомеченная грань ABC с наименьшей площадью, и строится вершина D в соответствии с выбранным параметром s.

При реализации алгоритма в пакете Ani3D был использован следующий эвристический метод выбора s и положения вершины D. Пусть a, b, c – длины сторон треугольника ABC. Сначала строится центр масс M треугольника ABC, затем от M по нормали к ABC на расстоянии $\sqrt{\frac{2}{3}}s$ откладывается вершина D. Параметр s выбирается на основании функции желаемого размера элементов, предоставленной пользователем, или вычисляется автоматически по формуле $s = \gamma\sqrt{\frac{a^2+b^2+c^2}{3}}$, где $\gamma \geq 1$ – некоторый параметр, отвечающий за скорость автоматического разгрубления сетки.

Алгоритм 3.1 Алгоритм продвигаемого фронта в двумерном случае

1: Положить $\mathcal{T}^0 = \emptyset$, $\mathcal{F}^0 = \mathcal{F}(\mathcal{P})$.
2: **для** $k = 0, 1, \ldots$ **начало цикла**
3: Выбрать непомеченный $\triangle ABC \in \mathcal{F}^k$ с минимальной площадью.
4: Если непомеченных треугольников не осталось, то перейти к 22.
5: Определить желаемую длину стороны элемента s.
6: Построить вершину D с учётом s и $[ABCD] > 0$.
7: Выбрать некоторое $R_0 > s$, $h > 0$ и $r > 0$.
8: **для** $R \in \{R_0, \infty\}$: **начало цикла**
9: Построить локальный фронт $\mathcal{F}_R^k$: $\mathcal{F}_R^k \supset \mathcal{F}^k \cap S_R(D)$.
10: Определить множество кандидатов $\Sigma_R^k = \{P_i^k\} \cap S_R(D)$.
11: Если $\{P_i^k\} \cap S_r(D) = \emptyset$ и $\mathcal{F}^k \cap S_h(D) = \emptyset$, то добавить D к Σ_R^k.
12: **для всех** $X \in \Sigma_R^k$: **начало цикла**
13: **если** $ABCX$ не пересекает $\mathcal{F}_R^k$, **тогда**
14: Добавить тетраэдр $T_k = ABCX$, $\mathcal{T}^{k+1} = \mathcal{T}^k \cup \{T_k\}$.
15: Обновить фронт: $\mathcal{P}^{k+1} = \mathcal{P}^k \setminus T_k$, $\mathcal{F}^{k+1} = \mathcal{F}(\mathcal{P}^{k+1})$.
16: Перейти к 21.
17: **конец если**
18: **конец цикла**
19: **конец цикла**
20: Пометить $\triangle ABC$, и перейти к 3
21: **конец цикла**
22: Положить $\mathcal{T} = \mathcal{T}^k$, $\mathcal{P}_{\text{out}} = \mathcal{P}^k$, $\mathcal{F}_{\text{out}} = \mathcal{F}^k$.

Оценим длину рёбер AD, BD и CD. При $\gamma \geq 1$ высота тетраэдра $s_0 \geq \frac{2a^2+2b^2+2c^2}{9}$. Рассмотрим ребро AD: $|AD|^2 = |AM|^2 + |MD|^2$. Длина AM выражается из формулы для медианы треугольника: $|AM| = \frac{2}{3}\sqrt{\frac{2b^2+2c^2-a^2}{4}}$,

$|AM|^2 = \frac{2b^2+2c^2-a^2}{9}$. Таким образом, получаем, что

$$|AD|^2 \geqslant \frac{2b^2+2c^2-a^2}{9} + \frac{2a^2+2b^2+2c^2}{9} = \frac{a^2+4b^2+4c^2}{9}.$$

Если стороны треугольника ABC были не меньше d, то и $|AD|$ будет не меньше d. Аналогичные оценки верны для BD и CD.

После выбора положения вершины-кандидата D мы рассматриваем локальную окрестность, и для каждого тетраэдра-кандидата проверяем его пересечение с локальным фронтом. Если все кандидаты оказались неподходящими, то проводится полный перебор всего фронта. Если и в этом случае подходящего тетраэдра найдено не было, то грань ABC помечается как уже рассмотренная, и перебор начинается с начала для следующей грани.

В отличие от Алгоритма 1.1 участок Алгоритма 3.1 с номерами строчек 3–20 может выполниться несколько раз для построения одного тетраэдра. При каждом выполнении этого участка соответствующая грань либо удаляется из фронта, либо помечается. В конце работы алгоритма помеченные грани и удалённые из фронта грани будут являться гранями конечной сетки $\mathcal{T}$. Поэтому количество выполнений этого участка алгоритма ограничено сверху не количеством тетраэдров в конечной сетке, а количеством граней.

Более детально рассмотрим остальные отличия трёхмерного алгоритма продвигаемого фронта от двумерного аналога, рассмотренного в Главе 1.

3.1.1. Проверка пересечения тетраэдра с треугольником

Построим алгоритм, аналогичный Алгоритму 1.2, для алгоритма продвигаемого фронта в трёхмерном пространстве. Как и в разделе 1.3.1, будем делать проверки на основе знака определителя матрицы 3×3.

В общем случае два выпуклых объекта не пересекаются тогда, и только тогда, когда существует плоскость, разделяющая их. В условиях Алгоритма 3.1 тетраэдр и треугольник могут также иметь одну общую вершину, или

одно общее ребро, или треугольник может быть одной из граней тетраэдра.

Выделим из задачи проверки пересечения отдельную подзадачу проверки пересечения треугольника с отрезком. В разделе 2.3.1 мы уже рассматривали аналог этой задачи для поверхности. Теперь рассмотрим общий случай. Будем искать такую плоскость, которая разделяет треугольник и отрезок с учётом предыдущего замечания об общих вершинах. Перебор всех возможных таких плоскостей представлен в Алгоритме 3.2 (см. стр. 80). В алгоритме используется функция $d(ABCD)$, вычисляющая знак выражения $[ABCD]$. Реализация такой функции уже обсуждалась в разделе 2.3.2.

Проверка пересечения треугольника с отрезком используется в Алгоритме 3.3 (см. стр. 81) проверки пересечения тетраэдра и треугольника. Сначала мы проверяем, есть ли пересечения у границы тетраэдра с границей треугольника, а потом проверяем особые случаи, такие как треугольник, целиком лежащий внутри тетраэдра.

Предложенная комбинация Алгоритмов 3.2 и 3.3 для проверки пересечения использует функцию $d(ABCD)$, вычисляющую знак выражения $[ABCD]$. Пусть про функцию $d(ABCD)$ известно, что при $d(ABCD) \neq 0$ выполнено $d(ABCD) = \text{sign}([ABCD])$. Тогда предложенные алгоритмы исключают возникновение ошибок второго типа. То есть, пересекающиеся на самом деле тетраэдр и треугольник не будут ошибочно восприняты как непересекающиеся. Это гарантирует замкнутость и отсутствие самопересечений при продвижении фронта.

3.1.2. Конечность работы алгоритма продвигаемого фронта

Проведём анализ Алгоритма 3.1, и покажем конечность числа операций. В предложенном алгоритме три явных вложенных цикла и один неявный за счёт перезапуска перебора в строке 20. Цикл перебора в строке 12 всегда

Алгоритм 3.2 Проверка пересечения треугольника с отрезком в трёхмерном пространстве

1: Найти общие вершины у треугольника ABC и отрезка PQ.
2: **если** общих вершин нет, **тогда**
3: Определить положение вершин P и Q относительно плоскости ABC.
4: **если** P и Q лежат в одном полупространстве, **тогда**
5: Пересечения нет.
6: **иначе если** P и Q лежат в разных полупространствах, **тогда**
7: Вычислить $k_A = d(PQBC), k_B = d(PQCA), k_C = d(PQCB)$.
8: Если среди k_A, k_B, k_C есть и 1 и -1, то пересечения нет.
9: **иначе если** P и Q лежат в плоскости ABC, **тогда**
10: Проверить, что одна из прямых PQ, AB, BC или CA разделяет треугольник и отрезок в разные полуплоскости.
11: Если такая прямая нашлась, то пересечения нет.
12: **конец если**
13: **конец если**
14: **если** одна общая вершина, **тогда**
15: Если вторая вершина отрезка не лежит в плоскости треугольника, то пересечения нет.
16: Иначе проверить что одна из прямых AB, BC, CA разделяет треугольник и отрезок в разные полуплоскости.
17: Если такая прямая нашлась, то пересечения нет.
18: **конец если**
19: **если** две общих вершины, **тогда**
20: Пересечения нет.
21: **конец если**
22: В остальных случаях считаем, что треугольник и отрезок пересекаются.

Алгоритм 3.3 Проверка пересечения тетраэдра с треугольником

1: Для каждой грани тетраэдра $ABCD$ и ребра треугольника PQR проверить пересечение.
2: Для треугольника PQR и каждого ребра тетраэдра $ABCD$ проверить пересечение.
3: Найти общие вершины у тетраэдра $ABCD$ и треугольника PQR.
4: **если** общих вершин нет, **тогда**
5: Если три точки P, Q, R лежат внутри $ABCD$, то есть пересечение.
6: **конец если**
7: **если** одна общая вершина, **тогда**
8: Если две оставшиеся точки треугольника PQR лежат внутри $ABCD$, то есть пересечение.
9: **конец если**
10: **если** две общих вершины, **тогда**
11: Если оставшаяся точка треугольника PQR лежит внутри $ABCD$, то есть пересечение.
12: **конец если**
13: В остальных случаях считаем, что пересечения нет.

конечен в силу конечности множества Σ_R^k. Цикл в строке 8 делает не больше двух итераций.

В трёхмерном случае существуют такие конфигурации фронта, для которых не существует подходящего тетраэдра с четвёртой вершиной из множества вершин-кандидатов фронта. Пример такой конфигурации фронта, для которой дальнейшее продвижение невозможно, представлен на Рис. 3.1. Именно по этой причине в Алгоритме 3.1, в строке 20, грани, не имеющие подходящих тетраэдров, помечаются и пропускаются при дальнейшем построении сетки. Каждая грань помечается не более одного раза, и количество

граней в $\mathcal{F}^k$ конечно, поэтому неявный цикл в строке 20 конечен.

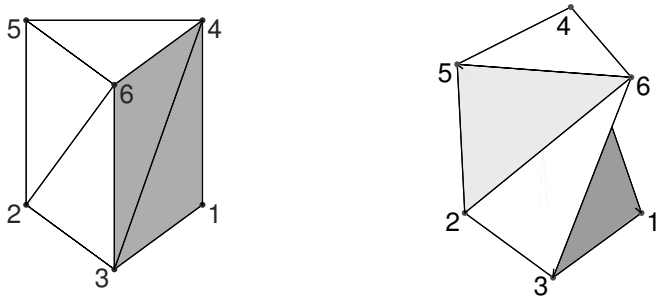


Рис. 3.1. Примеры конфигурации фронта, для которых дальнейшее продвижение невозможно: призмы Шёнхардта. На рисунке ребро 1–5 скрыто.

Покажем, что при определённом выборе параметров в Алгоритме 3.1 можно ограничить максимально возможное количество построенных тетраэдров. Рассуждения из раздела 1.4.2 полностью переносятся на трёхмерный случай. Мы можем ограничить снизу расстояние между вершинами сетками: $\rho_k > \delta > 0$ при любом k. Верна и аналогичная (1.3) оценка на максимальное количество вершин, которое можно уместить в области с объёмом V, площадью поверхности S, и состоящую из K компонент связности:

$$v_k \leq \frac{6V}{\pi\delta^3} + \frac{3S}{\pi\delta^2} + K. \tag{3.1}$$

По аналогии с Утверждением 1.4.1 можно доказать следующие грубые оценки на количество рёбер, граней и элементов в трёхмерных сетках.

Утверждение 3.1.1. *Пусть в трёхмерном пространстве задана произвольная сетка с v вершинами, e рёбрами, f гранями и n ячейками. Тогда верны следующие оценки: $n \leq \max(0, \frac{v(v-3)}{2} - 1)$, $f \leq \max(0, v(v-3))$, $e \leq \frac{v(v-1)}{2}$.*

Доказательство. Используем следующее расширение формулы Эйлера:

$$v - e + f - r = k - 1,$$

где v – количество вершин, e – количество рёбер, f – количество граней, r – количество областей, включая внешнюю, а k – количество компонент

связности. Тогда количество ячеек в сетке $n = r - 1$. Пусть $f \geq 4$, тогда у внешней области не меньше четырёх граней. Остальные области являются ячейками сетки, поэтому у них тоже не меньше четырёх граней в каждой. Каждая грань принадлежит ровно двум областям, поэтому $2f \geq 4r$, то есть $r \leq \frac{1}{2}f$. Складывая это неравенство с формулой Эйлера, получаем $\frac{1}{2}f \leq e - v + k - 1$.

Оценим количество рёбер в сетке. В каждой компоненте связности количество рёбер не превосходит количества различных пар вершин. Пусть в компоненте связности с номером i содержится $v_i \geq 1$ вершин. Тогда $e \leq \sum_{i=1}^{k} \frac{v_i(v_i-1)}{2}$. Максимальное количество рёбер при фиксированном количестве вершин и компонент связности достигается в случае одной большой компоненты из $v - k + 1$ вершин и $k - 1$ компонент по одной вершине, то есть $e \leq \frac{(v-k+1)(v-k)}{2} \leq \frac{v(v-1)}{2}$.

Обозначим для удобства $d = v - k + 1$, тогда $\frac{1}{2}f \leq e - d$ и $e \leq \frac{d(d-1)}{2}$. То есть $f \leq 2e - 2d \leq d(d-3) \leq v(v-3)$ и $n = r - 1 \leq \frac{1}{2}f - 1 \leq \frac{v(v-3)}{2} - 1$, утверждение доказано.

Отметим, что эти оценки весьма “грубые”, и на практике в неструктурированной тетраэдральной сетке количество граней приближённо равно $11v$, а количество тетраэдров приближённо равно $5.5v$. □

Из этого утверждения следует, что количество тетраэдров, и, следовательно, число итераций внешнего цикла в Алгоритме 3.1 ограничено.

Анализ сложности алгоритма продвигаемого фронта из раздела 1.5.2 применим и в трёхмерном случае. Оценка сверху на количество операций в худшем случае составит величину пропорциональную N_F^3, где N_F – количество граней в конечной сетке, которое ограничено согласно оценке на количество вершин (3.1) и Утверждению 3.1.1. Оценка в среднем на количество операций составляет величину, пропорциональную $N_F \log(N_F)$. Оценка сред-

ней скорости работы будет экспериментально подтверждена в разделе 3.4.1.

Мы показали, что Алгоритм 3.1 сделает конечное число операций. Однако он не гарантирует построение конформной тетраэдральной сетки для всей области $\mathcal{P}$. На практике алгоритм продвигаемого фронта разбивает более 90% объёма области, а неразбитая часть области, $\mathcal{P}_{\text{out}}$, состоит из некоторого количества отдельных лакун – многогранников с малым количеством граней. Для области $\mathcal{P}_{\text{out}}$ применяется второй метод, речь о котором пойдёт в следующем разделе.

3.2. Устойчивый метод на основе тетраэдризации Делоне

В этом разделе мы предложим метод построения тетраэдральной сетки, согласованной с заданным фронтом на основе тетраэдризации Делоне. Общая идея этого метода была предложена П.-Л. Джорджем в [15]. Мы рассмотрим упрощённую версию этого метода, которую можно применить для разбиения лакун, оставленных алгоритмом продвигаемого фронта [48]. Этот метод применим для произвольных замкнутых фронтов, однако не позволяет контролировать ни размер, ни качество построенных тетраэдров.

Построение сетки делится на три этапа. На первом этапе строится сетка для выпуклой оболочки множества точек заданного фронта. На втором этапе сетка измельчается и восстанавливается геометрия области. На третьем этапе восстанавливается согласованность сетки с заданными фронтом.

3.2.1. Тетраэдризация Делоне

Тетраэдризацией Делоне для конечного множества точек $\mathcal{V} = \{V_1, \ldots, V_n\}$ называется такое конформное разбиение выпуклой оболочки множества то-

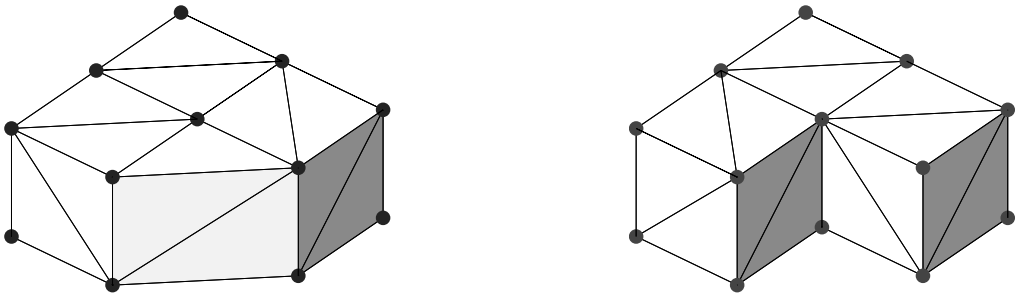
чек на тетраэдры $\mathcal{T} = \{T_1, \ldots, T_m\}$, что для любого тетраэдра T_i строго внутри описанной вокруг него сферы нет точек из $\mathcal{V}$.

В силу Утверждения 3.1.1 количество тетраэдров, m, ограничено квадратичной зависимостью от числа вершин, n.

Существует множество способов построения тетраэдризации Делоне, мы не будем останавливаться на деталях, отсылая читателя к [59]. Отметим лишь, что самый простой итеративный способ построения по сложности в худшем случае пропорционален m^2, и в среднем пропорционален $m^{\frac{3}{2}}$.

3.2.2. Восстановление геометрии области

На этом этапе у нас есть тетраэдральная сетка $\mathcal{T}$ и заданный фронт $\mathcal{F}$ на одном наборе вершине $\mathcal{V}$. Пример сетки и фронта показан на Рис. 3.2.



($а$) ($б$)

Рис. 3.2. Две сетки на одном наборе вершин: ($а$) тетраэдральная сетка для выпуклой оболочки, ($б$) фронт $\mathcal{F}$.

Будем измельчать сетку $\mathcal{T}$, добавляя новые вершины на рёбрах и гранях, так, чтобы рёбра и грани фронта $\mathcal{F}$ либо целиком, либо в виде своих разбиений были представлены в новой сетке. Детали предлагаемого алгоритма представлены в Алгоритме 3.4.

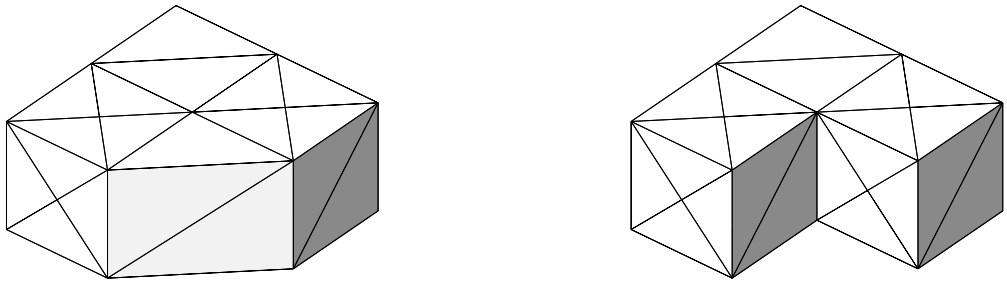
Алгоритм 3.4 Восстановление геометрии области

1: **процедура** ОБРАБОТКА_РЕБРА(V_1, V_2)
2: Найти тетраэдр с вершиной V_1, пересекающий V_1V_2.
3: **если** тетраэдр найден, **тогда**
4: Построить точку пересечения V на грани тетраэдра.
5: Разбить соответствующие тетраэдры в $\mathcal{T}$.
6: ОБРАБОТКА_РЕБРА(V, V_2).
7: **конец если**
8: **конец процедуры**
9: **процедура** ОБРАБОТКА_ГРАНЕЙ(V_1, V_2)
10: **для всех** граней $F \in \mathcal{F}$: **начало цикла**
11: **если** грань F пересекает ребро V_1V_2, **тогда**
12: Построить точку пересечения V.
13: Разбить соответствующие тетраэдры в $\mathcal{T}$.
14: ОБРАБОТКА_ГРАНЕЙ(V_1, V).
15: ОБРАБОТКА_ГРАНЕЙ(V, V_2).
16: Выйти из процедуры.
17: **конец если**
18: **конец цикла**
19: **конец процедуры**
20: **для всех** рёбер V_1V_2 из $\mathcal{F}$: **начало цикла**
21: ОБРАБОТКА_РЕБРА(V_1, V_2).
22: **конец цикла**
23: **для всех** рёбер V_1V_2 из $\mathcal{T}$: **начало цикла**
24: ОБРАБОТКА_ГРАНЕЙ(V_1, V_2).
25: **конец цикла**
26: Удалить тетраэдры, лежащие за пределами $\mathcal{F}$.

Первым проходом проверяются пересечения рёбер фронта $\mathcal{F}$ с гранями $\mathcal{T}$. Для каждого ребра из $\mathcal{F}$ пересекающие его тетраэдры из $\mathcal{T}$ разбиваются таким образом, чтобы ребро либо целиком, либо в виде разбиения было представлено в сетке. Второй проход аналогичен, проверяются пересечения рёбер сетки $\mathcal{T}$ с гранями $\mathcal{F}$. Для каждой грани из $\mathcal{F}$ пересекающие её тетраэдры из $\mathcal{T}$ разбиваются таким образом, чтобы грань либо целиком, либо в виде разбиения была представлена в сетке.

Каждый раз при разбиении тетраэдров в $\mathcal{T}$ мы создаём точку пересечения, лежащую на $\mathcal{F}$. При этом разбиваются только элементы из $\mathcal{T}$. В конце $\mathcal{F}$ останется без изменений, а новая сетка $\mathcal{T}$ своими гранями будет полностью покрывать грани $\mathcal{F}$.

Удаляя тетраэдры из $\mathcal{T}$, лежащие за границами многогранника $\mathcal{P}$, мы восстановим геометрию области. На Рис. 3.3 показана измельчённая сетка $\mathcal{T}$ до и после удаления тетраэдров.



($а$) ($б$)

Рис. 3.3. Восстановление геометрии области: ($а$) измельчённая сетка, ($б$) сетка с правильной геометрией области.

Полученная сетка $\mathcal{T}$ будет являться конформной сеткой для многогранной области $\mathcal{P}$, однако она не будет согласована с его границей $\mathcal{F}$. На границе $\mathcal{T}$ будут лежать новые точки пересечения, которые мы только что построили (сравните Рис. 3.2, $б$ и Рис. 3.3, $б$), алгоритм удаления этих точек с границы будет представлен в следующем разделе.

3.2.3. Восстановление следа сетки на границе

На завершающем этапе с границы сетки удаляются лишние вершины. Это достигается за счёт сдвига их внутрь области, и заполнения образовавшихся “вмятин” конформными тетраэдральными сетками.

Все лишние точки делятся на точки, лежащие на рёбрах $\mathcal{F}$, и на точки, лежащие на гранях $\mathcal{F}$. Избавление от точек в обоих случаях происходит одним и тем же способом. Рассмотрим для определённости первый случай.

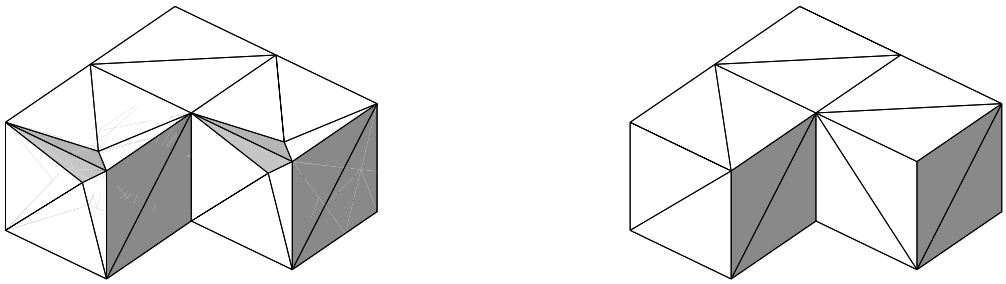
Пусть P – точка на граничном ребре, которую надо убрать с поверхности. Предположим для простоты изложения, что фронт $\mathcal{F}$ является простым. С незначительными дополнениями предлагаемый алгоритм применим и в общем случае.

Рассмотрим множество Σ_P тетраэдров из $\mathcal{T}$ с вершиной P. Граница этого множества, $\Omega_P = \partial\Sigma_P$, состоит из треугольников двух типов: лежащих на границе $\mathcal{P}$, и лежащих внутри $\mathcal{P}$. При сдвиге вершины P внутрь $\mathcal{P}$ на границе области образуются “вмятины”. Эти вмятины на самом деле являются объединением двух конусов с вершиной P и многоугольными основаниями. Каждый конус может быть конформно разбит на тетраэдры, для этого многоугольные основания разбиваются диагоналями на треугольники.

Можно показать, что внутри Σ_P всегда есть непустое открытое множество точек, куда может быть сдвинута вершина P при сохранении невырожденности тетраэдров сетки $\mathcal{T}$. На Рис. 3.4 представлена сетка $\mathcal{T}$ с вмятинами и с заделанными вмятинами.

Второй случай, когда вершина лежит внутри грани $\mathcal{F}$ рассматривается аналогично, единственное отличие – вместо двух конусов вмятина будет являться одним конусом с многоугольной границей.

Полный алгоритм действий на этом этапе представлен в Алгоритме 3.5. Сначала рассматриваются вершины на рёбрах $\mathcal{F}$, а потом на гранях $\mathcal{F}$.



($а$) ($б$)

Рис. 3.4. Восстановление следа сетки на границе: ($а$) сетка с образованными вмятинами, ($б$) сетка с правильным следом на границе.

Алгоритм 3.5 Восстановление следа сетки на границе

1: **для всех** вершин P на рёбрах $\mathcal{F}$ и гранях $\mathcal{F}$: **начало цикла**
2: Построить множество соседних тетраэдров Σ_P.
3: Из $\Omega_P = \partial\Sigma_P$ выделить грани, лежащие на $\mathcal{F}$: $\Omega'_P = \Omega_P \cap \mathcal{F}$.
4: Переразбить Ω'_P на треугольники, исключив вершину P.
5: Добавить в $\mathcal{T}$ тетраэдры из триангуляции Ω'_P и вершины P.
6: Сдвинуть P внутрь Σ_P, восстановив невырожденность тетраэдров.
7: **конец цикла**

Предложенный метод гарантирует построение сетки, согласованной с заданным замкнутым фронтом, при условии, что все вычисления проводятся точно. Качество построенных тетраэдров никак не ограничивается снизу. На практике из-за накопления вычислительных погрешностей метод может построить вырожденные или вывернутые тетраэдры. По этой причине в качестве третьего этапа необходимо применение алгоритмов, позволяющих значительно улучшить качество сетки и избавиться от вырожденных элементов.

3.2.4. Конечность работы алгоритма

Проведём краткий анализ конечности числа операций в предложенном методе, не вдаваясь в оценки вычислительной сложности.

На первом этапе мы строим тетраэдризацию Делоне $\mathcal{T}$. Количество вершин в ней совпадает с количеством вершин в $\mathcal{F}$. При наличии оценки на количество вершин в сетке $\mathcal{T}$ с помощью Утверждения 3.1.1 мы получаем оценки на количество рёбер, граней и тетраэдров.

На втором этапе при восстановлении геометрии в Алгоритме 3.4 используются три основных операции. В строке 21 для каждого ребра из $\mathcal{F}$ новых вершин в $\mathcal{T}$ может быть добавлено не более чем количество тетраэдров в $\mathcal{T}$ на момент перед выполнением строки 21. Количество рёбер в $\mathcal{F}$ конечно, и следовательно, ограничено количество новых вершин.

В строке 24 для каждого ребра из $\mathcal{T}$ новых вершин в $\mathcal{T}$ добавляется не более, чем количество граней в $\mathcal{F}$. Отметим, что новые рёбра при добавлении в $\mathcal{T}$ будут лежать на $\mathcal{F}$, и тем самым сами не будут порождать новых вершин.

Оставшиеся операции, удаление внешних тетраэдров в строке 24 Алгоритма 3.4 и сдвиг лишних точек с границы области в Алгоритме 3.5, не добавляют новых вершин в $\mathcal{T}$, и тем самым будут работать конечное время.

На практике предложенный алгоритм на основе тетраэдризации Делоне используется только для локализованных лакун с небольшим количеством вершин и граней. Количество построенных с его помощью тетраэдров невелико, и время работы делает незначительный вклад в общее время работы всей цепочки построения тетраэдральных сеток. В разделе 3.4.2 будут проведены эксперименты для оценки распределения времени работы между алгоритмом продвигаемого фронта и предложенным алгоритмом.

3.3. Улучшение качества полученной сетки

Для построения тетраэдральной сетки используется комбинация трёх алгоритмов. Первый алгоритм продвигаемого фронта используется для построения большей части сетки с возможностью контролирования шага сетки и

качества тетраэдров. Второй алгоритм на основе тетраэдризации Делоне используется для разбиения оставшейся части области. Третий, заключительный алгоритм используется для улучшения качества сетки.

Отметим, что для достижения хорошего качества сетки с использованием только первых двух алгоритмов, как правило, необходима хорошая поверхностная сетка – начальный фронт. Хороший начальный фронт, подходящим образом подобранная функция, отвечающая за желаемый размер тетраэдров, или правильно подобранный параметр автоматического разгрубления являются ключевыми факторами для получения качественной сетки.

В рамках полностью автоматической и надёжной технологии построения неструктурированных тетраэдральных сеток необходимо дополнительное улучшение качества сетки с помощью третьего алгоритма.

В этой работе мы не будем обсуждать какой-то конкретный алгоритм для улучшения тетраэдральных сеток. Отметим лишь основные идеи, которые могут быть использованы.

Самые простые методы использует сдвиг вершин сетки без изменения топологической структуры. Примерами таких методов могут служить представленные в разделе 1.6.1 алгоритмы. Использование локальных модификаций топологии сетки, таких как разбиение ребра новой вершиной, стягивание ребра в одну вершину и пр., в дополнение к сдвигу вершин как правило позволяет значительно улучшить качество сетки.

В пакет библиотек Ani3D входит библиотека для анизотропной адаптации сеток, AniMBA. Используемые в ней методы позволяют улучшать качество ячеек построенной сетки. Использование библиотеки AniMBA рекомендуется как завершающий шаг в цепочке построения тетраэдральной сетки. В разделе 3.4.3 будут проведены эксперименты по измерению качества сетки на разных этапах всей цепочки.

3.4. Результаты экспериментов

В этом разделе будет представлено несколько примеров работы рассмотренных выше алгоритмов. Для вычисления качества элементов сетки мы будем использовать следующую формулу для качества тетраэдра $ABCD$:

$$q_T = 36\sqrt{2}\frac{V}{\sum l_{ij}^3},$$

здесь $V = \frac{1}{6}[ABCD]$, l_{ij} – длины рёбер тетраэдра. Величина q_T для невырожденных тетраэдров лежит в интервале $(0, 1]$. Для равностороннего тетраэдра $q_T = 1$. Для вырожденных или вывернутых тетраэдров $q_T \leq 0$.

Алгоритм продвигаемого фронта всегда строит тетраэдры с $q_T > 0$. При вычислении качества тетраэдра, построенного с алгоритмом на основе тетраэдризации Делоне, из-за неточных вычислений может получиться $q_T \leq 0$.

По отношению ко всей сетке нас будет в первую очередь интересовать наихудшее качество тетраэдров $q_{\min}$ и распределение качества по всей сетке. Для лучшего отображения результатов будем использовать логарифмическую шкалу.

Эксперименты проводились на разных машинах при различных условиях. Однако тестовые запуски в пределах одного эксперимента проводились в одинаковых условиях. Все эксперименты проводились с использованием библиотеки программ Ani3D.

3.4.1. Скорость работы алгоритма продвигаемого фронта

Экспериментально измерим скорость работы алгоритма продвигаемого фронта. В качестве области возьмём единичный куб. На поверхности куба с помощью алгоритма продвигаемого фронта, описанного в разделе 2.3, строится квазиравномерная сетка с шагом h. Полученная поверхностная триангуляция используется как начальный фронт для трёхмерного алгоритма про-

двигаемого фронта для построения квазиравномерной сетки с шагом h.

Уменьшая шаг сетки h, будем следить за количеством тетраэдров в сетке, N_T, и временем построения сетки, t. Результаты экспериментов представлены в Таблице 3.1.

Таблица 3.1. Скорость работы алгоритма продвигаемого фронта в трёхмерном пространстве.

h	N_T	t, сек	$N_T/1$ сек	$t/(N_T \log N_T)$, мкс
0.1	3544	0.48	7383	16.6
0.05	27988	4.81	5819	16.8
0.025	204734	43.35	4723	17.3
0.0125	1613980	406.75	3968	17.6

Из результатов расчётов видно, что время работы пропорционально величине $N_T \log N_T$. В таблицу также включены столбцы со скоростью построения (тетраэдры в секунду), и оценкой коэффициента перед $N_T \log N_T$.

3.4.2. Комбинация алгоритма продвигаемого фронта и устойчивого алгоритма

В этом разделе мы рассмотрим пример последовательной работы алгоритма продвигаемого фронта и устойчивого алгоритма на основе тетраэдризации Делоне. Первый используется для построения большей части сетки, второй используется для разбиения оставшихся лакун.

Рассмотрим начальную поверхностную триангуляцию трёхмерной отсканированной модели дракона, изображённой на Рис. 3.5, *a*. Она состоит из 54296 вершин и 108582 треугольников. Все вершины сетки лежат в плоских срезах, параллельных плоскости xy. В каждом срезе вершины соединены отрезками, образуя контур сечения модели плоскостью. Контуры соседних

срезов соединены треугольниками и образуют конформную поверхностную треугольную сетку. Отметим, что эта начальная сетка содержит достаточно большое количество треугольников с плохим качеством.

($а$) ($б$)

Рис. 3.5. Работа алгоритма продвигаемого фронта на модели дракона: ($а$) начальный фронт, ($б$) конечный фронт.

На первом этапе мы применяем метод продвигаемого фронта с автоматическим увеличением шага сетки при продвижении внутрь области. Параметр γ, отвечающий за скорость увеличения шага сетки, выбран $\gamma = 1.3$. Алгоритм продвигаемого фронта построил сетку для части области, занимающей 99.67% объёма всей области. Всего построен 250461 тетраэдр, в конечном фронте осталось 1718 треугольников, они изображены на Рис. 3.5, $б$. Распределение качества тетраэдров полученной сетки представлено в первой строке Таблицы 3.2. В столбцах таблицы показано количество тетраэдров, качество которых превышает соответствующее значение, и которые не вошли в предыдущие столбцы. Наихудшее качество сетки составляет $q_{\min} = 3.698 \cdot 10^{-7}$, что

обуславливается плохим качеством треугольников в начальном фронте.

Полученный фронт автоматически разбивается на несвязные лакуны, и для каждой из них автоматически запускается устойчивый метод. Лакуны имеют очень плохую форму, и поэтому устойчивый метод порождает очень плохие элементы. Распределение качества тетраэдров в конечной сетке приведено во второй строке Таблицы 3.2. Наихудшее качество сетки теперь составляет $q_{\min} = 1.232 \cdot 10^{-14}$. Итоговая тетраэдральная сетка содержит 71241 вершину, 108582 боковых граней, полностью совпадающих с заданным начальным фронтом, и 254764 тетраэдра.

Для улучшения качества полученной сетки была использована библиотека aniMBA из пакета Ani3D, при этом боковые грани сетки были зафиксированы. Распределение качества тетраэдров в новой сетке показано в третьей строке Таблицы 3.2. Качество тетраэдральной сетки в этом примере ограничивается плохим качеством начальной поверхностной триангуляции. В новой сетке 75665 вершин, 108582 боковых грани, и 267475 тетраэдров. Отметим, что при предварительном улучшении качества начальной поверхностной сетки можно получить тетраэдральную сетку с лучшим качеством [8].

Таблица 3.2. Распределение качества тетраэдров на разных этапах построения сетки для модели дракона: (Ф) алгоритм продвигаемого фронта; (Д) устойчивый алгоритм на основе тетраэдризации Делоне; (С) улучшение сетки с сохранением заданного следа на границе (библиотека aniMBA из Ani3D).

	$q_{\min}$	N_T	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
Ф	$3.7 \cdot 10^{-7}$	250461	248359	1923	166	11	1	0	1
Ф+Д	$1.2 \cdot 10^{-14}$	254764	249674	3428	1020	340	138	71	93
Ф+Д+С	$1.9 \cdot 10^{-7}$	267475	258449	8269	718	30	6	0	3

Аналогичные эксперименты были проведены для большого количества

разных областей, приведём здесь лишь несколько показательных примеров. Первый пример, `rbox1`, – полученная экспортом из САПР триангуляция поверхности для параллелепипеда со скруглёнными краями, качество треугольников очень плохое, имеются резкие перепады размеров треугольников на поверхности. Второй пример, `drag3`, – рассмотренная в этом разделе модель дракона. Третий пример, `Lymph`, – модель лимфоузла, отношение наибольшего и наименьшего размеров треугольников в поверхностной сетке достигает 10^3, к этой модели мы вернёмся в разделе 4.5.

В Таблице 3.3 для каждой модели приведено распределение объёма области, для которой построена сетка с помощью метода продвигаемого фронта и устойчивого метода. Приведено количество тетраэдров, построенных этими методами, и время, которое потребовалось для выполнения каждого из двух методов.

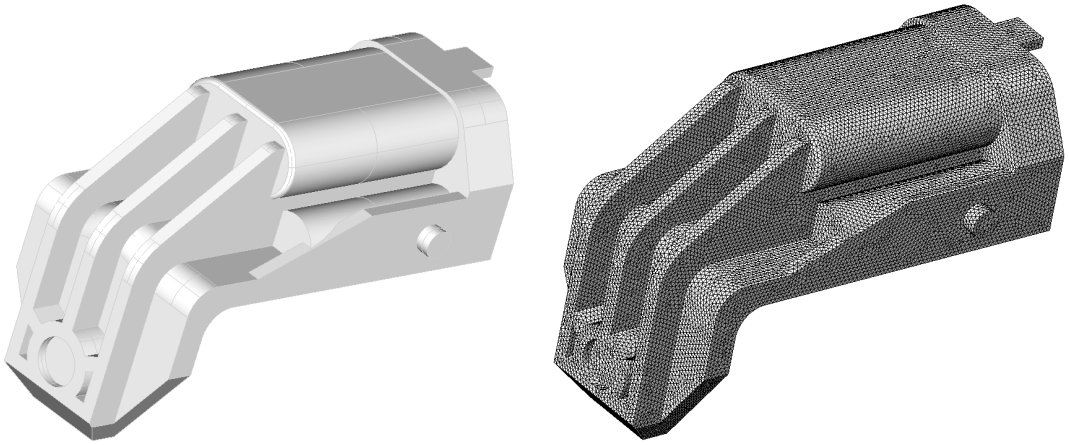
Таблица 3.3. Распределение объёма области, количества тетраэдров, и времени работы между алгоритмом продвигаемого фронта и устойчивым алгоритмом на основе тетраэдризации Делоне.

	продвигаемый фронт			устойчивый метод		
модель	%	N_T	t, сек	%	N_T	t, сек
`rbox1`	97.25	265	0.18	2.75	171	0.03
`drag3`	99.67	250461	101.70	0.33	4303	0.76
`Lymph`	99.94	437874	405.08	0.06	13624	6.82

Отметим, что устойчивый метод используется лишь в незначительной части всей области, и занимает малую часть времени по сравнению с алгоритмом продвигаемого фронта.

3.4.3. Взаимодействие с геометрическим ядром САПР

В этом разделе мы продемонстрируем совместную работу методов построения поверхностных треугольных и объёмных тетраэдральных сеток. В качестве примера рассмотрим модель `62_shaver1` с сайта САПР OpenCASCADE [69]. BREP-модель состоит из 266 вершины, 403 криволинейных рёбер, и 153 криволинейных граней (см. Рис. 3.6, *а*). С помощью метода продвигаемого фронта на поверхности, описанного в Главе 2, была построена квазиравномерная треугольная сетка из 38335 вершин и 76666 треугольников (см. Рис. 3.6, *б*).



(*а*) (*б*)

Рис. 3.6. Модель, заданная в САПР: (*а*) BREP-модель, (*б*) поверхностная квазиравномерная треугольная сетка.

Для построения тетраэдральной сетки использовалась комбинация трёх методов: алгоритм продвигаемого фронта, устойчивый алгоритм на основе тетраэдризации Делоне, и метод улучшения качества сетки из библиотеки aniMBA. После первого метода была построена сетка с 206562 тетраэдрами для части области, занимающей 99.84% от общего объёма модели. В фронте осталось 830 треугольных граней, которые были переданы на вход второму методу, который построил ещё 1476 тетраэдров. Минимальное качество полу-

ченной сетки составило $q_{\min} = 1.044 \cdot 10^{-10}$.

После сглаживания сетки с помощью третьего метода качество сетки значительно улучшилось, и минимальное качество одного тетраэдра составило $q_{\min} = 1.780 \cdot 10^{-2}$. Итоговая сетка содержит 60204 вершины, 76666 граничных грани, полностью совпадающих с гранями, построенными поверхностным генератором сеток, и 229554 тетраэдра. На Рис. 3.7 показан срез тетраэдральной сетки.

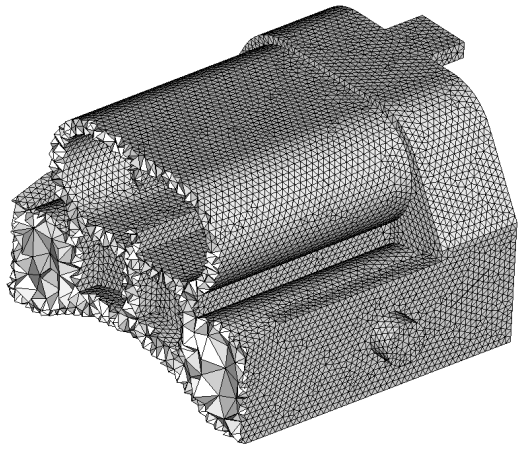


Рис. 3.7. Разрез тетраэдральной сетки для области, заданной в САПР.

Общее время работы составило 12 минут 30 секунд. Распределение времени по этапам следующее: построение поверхностной сетки – 8 минут 12 секунд, построение объёмной сетки – 1 минута 53 секунды, улучшение сетки – 10 секунд. Оставшееся время было потрачено на проверку корректности полученной сетки и разные вспомогательные операции.

3.5. Выводы к третьей главе

Разработана, реализована и протестирована технология надёжного построения конформных тетраэдральных сеток, согласованных с заданной дискретной границей, для многокомпонентных многосвязных многогранных областей. Доказана конечность числа операций и корректность работы всей

цепочки при условии использования точных вычислений. Доказана корректность работы алгоритма продвигаемого фронта при использовании неточных вычислений. Метод, основанный на тетраэдризации Делоне, при использовании неточных вычислений может породить почти вырожденные и вырожденные тетраэдры, поэтому необходимо использование последующего улучшения качества сетки. Экспериментально подтверждена оценка сложности работы алгоритма продвигаемого фронта в среднем. Проведено исследование распределения работы при построении сетки между двумя используемыми методами. Приведён пример работы алгоритмов, реализованных автором в библиотеке aniAFT из пакета Ani3D.

Глава 4

Сетки с многогранными ячейками и монотонная дискретизация уравнения диффузии

В трёхмерном моделировании используются разные типы расчётных сеток (тетраэдральные, гексаэдральные, призматические, сетки типа восьмеричное дерево), возможно применение гибридных сеток с разными типами ячеек. Как правило, все подобные типы сеток попадают в класс конформных сеток с многогранными ячейками. В настоящее время востребованы простые консервативные схемы дискретизации, применимые к анизотропным тензорам диффузии на произвольных конформных сетках. В диссертационной работе предложен новый метод конечных объёмов, сохраняющий неотрицательность решения.

Широко известные консервативные линейные схемы на неструктурированных сетках, такие как метод конечных объёмов с многоточечной аппроксимацией потока (MPFA – Multipoint flux approximation), смешанные конечные элементы (MFE – Mixed finite element) и миметические дискретизации (MFD – Mimetic finite difference), имеют второй порядок точности, но не являются монотонными для анизотропных диффузионных тензоров. Метод конечных объёмов со степенями свободы в центрах ячеек и линейной двухточечной аппроксимацией диффузионного потока является монотонным, но не имеет даже первого порядка точности для анизотропных диффузионных задач или на неструктурированных сетках. Тем не менее именно этот метод наиболее распространён в моделировании течений в пористых средах в силу своей технологической простоты и монотонности. Ограничения на монотонность ко-

нечно-объёмных схем MPFA рассмотрены в [1, 29]. Условия, необходимые для выполнения дискретного принципа максимума в схемах с кусочно-линейными конечными элементами, накладывают значительные ограничения на расчётные сетки [7, 21, 31]. Другой класс монотонных схем для произвольных сеток состоит из нелинейных методов. Эти методы гарантируют неотрицательность решения для уравнения Пуассона [6] и даже для общего уравнения диффузии [24–26, 41, 45, 50].

Развивая подходы [26, 41, 50], мы предлагаем новую монотонную схему на основе метода конечных объёмов с нелинейной двухточечной аппроксимацией потока. Первоначальная идея принадлежит К. ЛеПотье [23], который предложил монотонную схему дискретизации параболических уравнений методом конечных объёмов на треугольных сетках. Спустя два года схема и анализ её монотонности были расширены на случай стационарного уравнения диффузии с полными анизотропными тензорами на треугольных, или скалярными коэффициентами диффузии на сетках с многоугольными ячейками регулярной формы [25]. В то же время схема была расширена для конформных тетраэдральных сеток в [45, 50], монотонность схемы была доказана для полных анизотропных тензоров. Позже вывод шаблона нелинейной двухточечной аппроксимации потока был модифицирован на основе идеи разложения векторов конормали [41], которая изначально была предложена для линейного метода конечных объёмов в [51]. С новым подходом схема дискретизации стала применимой для широкого класса сеток с ячейками звёздного типа и для полных тензоров диффузии.

Во всех этих монотонных нелинейных схемах конечных объёмов для вычисления потока используются дискретные значения решения в центрах ячеек (первичные неизвестные) и в узлах сетки (вспомогательные неизвестные). Вспомогательные неизвестные находились из первичных на основе интерполяции. Выбор схемы интерполяции оказывает большое влияние на точность

нелинейной схемы [25, 41]. Определённый метод интерполяции может оказаться эффективным для одной задачи и неточным для другой. Была разработана новая схема, не требующая интерполяции решения в узлы сетки, для полного анизотропного тензора на неструктурированных многоугольных двумерных сетках [26]. Однако, этот метод был неприменим для *разрывных* диффузионных тензоров на произвольных сетках, так как требовал доразбиения некоторых ячеек.

В диссертационной работе представлено расширение подхода из [26] для случая трёхмерных конформных сеток с многогранными ячейками и разрывных диффузионных тензоров. Новый метод точен для линейных и кусочно-линейных решений, поэтому есть основания ожидать второй порядок сходимости. Предложенная схема может потребовать интерполяции для некоторых вспомогательных неизвестных, однако бóльшая часть этих неизвестных интерполируется на основе *физических* принципов, например, на основании непрерывности диффузионного потока на гранях сетки. Шаблон оператора интерполяции на гранях сетки является двухточечным, а коэффициенты зависят как от самих неизвестных, так и от некоторых вспомогательных значений на гранях. В некоторых случаях могут понадобиться вспомогательные значения в рёбрах граней, которые получаются с помощью арифметического осреднения значений из соседних граней. Такой выбор интерполяционной схемы использует физически обоснованные значения на гранях и прост в реализации. В диссертационной работе представлен теоретический анализ только монотонности метода, тем не менее, численные эксперименты показывают второй порядок сходимости в дискретной L_2 норме.

Предложенная схема с двухточечной аппроксимацией потока привлекательна с технологической точки зрения из-за компактного шаблона на многогранных сетках. Отметим, что для кубических сеток и диагонального тензора диффузии шаблон совпадёт с традиционным семиточечным шаблоном. Ос-

новные накладные расходы нелинейных методов связаны с необходимостью использования двух вложенных итераций для решения системы нелинейных алгебраических уравнений. Внешние итерации выполняются методом Пикара, который гарантирует неотрицательность решения на каждом шаге. Внутренние итерации выполняются стабилизированным методом бисопряжённых градиентов для решения системы линейных уравнений.

В этой главе будет предложена новая монотонная нелинейная схема конечных объёмов для решения стационарного уравнения диффузии на конформных сетках с многогранными ячейками. Будет проведён теоретический анализ монотонности метода, а также проведены эксперименты для подтверждения монотонности и оценки порядка сходимости предложенной схемы.

4.1. Уравнение диффузии

Пусть Ω – трёхмерная многогранная область, граница которой состоит из двух частей, $\Gamma = \Gamma_N \cup \Gamma_D$, причём множество Γ_D замкнуто и непусто, $\Gamma_D = \bar{\Gamma}_D$, $\Gamma_D \neq \emptyset$. Рассмотрим задачу диффузии для неизвестной концентрации c:

$$\begin{aligned} -\mathrm{div}(\mathbb{K}\nabla c) &= g \quad \text{в} \quad \Omega \\ c &= g_D \quad \text{на} \quad \Gamma_D \\ -\mathbf{n}\cdot\mathbb{K}\nabla c &= g_N \quad \text{на} \quad \Gamma_N, \end{aligned} \tag{4.1}$$

здесь $\mathbb{K}(\mathbf{x}) = \mathbb{K}^{\mathrm{T}}(\mathbf{x}) > 0$ – полный анизотропный диффузионный тензор, g – внешние источники, и $\mathbf{n}$ – внешний нормальный вектор.

Рассмотрим конформную сетку $\mathcal{T}$ из многогранников с плоскими гранями. Будем предполагать, что каждая ячейка является трёхмерной ячейкой звёздного типа по отношению к центру масс ячейки, то есть из центра масс каждая грань ячейки видна полностью. Также будем предполагать, что каждая грань ячейки в свою очередь является плоской двумерной ячейкой звёздного типа по отношению к её центру масс. Ограничение на плоские грани

введено для упрощения изложения, в разделе 4.4.6 будут также приведены результаты численных экспериментов на сетках с неплоскими гранями. Обозначим количество ячеек в сетке как $N_{\mathcal{T}}$, а количество внешних граней – $N_{\mathcal{B}}$. Будем предполагать, что сетка $\mathcal{T}$ является связной по граням, то есть не может быть разбита на две подсетки, не имеющих общих граней. Также будем предполагать, что тензор $\mathbb{K}(\mathbf{x})$ в пределах одной ячейки меняется незначительно, однако он может значительно отличаться при переходе через грань от одной ячейки к другой, и, в частности, менять направления своих главных осей. В таких случаях мы будем говорить, что тензор является разрывным на соответствующей грани.

Обозначим через $\mathcal{F}_I$ и $\mathcal{F}_B$ множества внутренних и внешних граней, соответственно. Пусть подмножество $\mathcal{F}_J \subset \mathcal{F}_I$ содержит все грани, на которых тензор является разрывным. Множество $\mathcal{F}_B$ в свою очередь разбивается на два подмножества $\mathcal{F}_B^D$ и $\mathcal{F}_B^N$, в которых накладываются граничные условия типа Дирихле и Неймана, соответственно. Мощность (количество элементов) множеств $\mathcal{F}_*$ обозначим как $N_{\mathcal{F}_*}$. Для произвольной ячейки T обозначим множества её граней и рёбер как $\mathcal{F}_T$ и $\mathcal{E}_T$, соответственно, а множество рёбер грани f обозначим как $\mathcal{E}_f$.

4.2. Монотонная нелинейная схема на основе метода конечных объёмов

Обозначим через $\mathbf{q} = -\mathbb{K}\nabla c$ диффузионный поток, который удовлетворяет уравнению баланса:

$$\operatorname{div} \mathbf{q} = g \quad \text{в} \quad \Omega. \tag{4.2}$$

Выпишем схему конечных объёмов с нелинейной двухточечной дискретизацией диффузионного потока. Интегрируя уравнение (4.2) по многогранной

ячейке T и используя формулу Грина, мы получаем:

$$\int\limits_{\partial T} \mathbf{q} \cdot \mathbf{n}_T \, \mathrm{d}s = \int\limits_{T} g \, \mathrm{d}x, \tag{4.3}$$

здесь $\mathbf{n}_T$ обозначает единичную внешнюю нормаль к ∂T. Для грани f ячейки T соответствующий нормальный вектор обозначим как $\mathbf{n}_f$. В дальнейшем, если в контексте участвует только одна ячейка T, то будем считать, что $\mathbf{n}_f$ соответствует внешнему по отношению к T нормальному вектору. Во всех остальных случаях мы будем явно указывать ориентацию $\mathbf{n}_f$. Будет удобно считать, что $|\mathbf{n}_f| = |f|$, где $|f|$ – площадь грани f. В таком случае уравнение (4.3) запишется в следующем виде:

$$\sum_{f \in \partial T} \mathbf{q}_f \cdot \mathbf{n}_f = \int\limits_{T} g \, \mathrm{d}x, \tag{4.4}$$

где $\mathbf{q}_f$ – средняя плотность диффузионного потока для грани f.

Каждой ячейке T мы поставим в соответствие одну степень свободы C_T для концентрации c. Вектор, составленный из всех дискретных концентраций C_T обозначим как C. Если две ячейки T_+ и T_- имеют общую грань f, то двухточечная дискретизация потока для этой грани записывается в следующем виде:

$$\mathbf{q}_f^h \cdot \mathbf{n}_f = M_f^+ C_{T_+} - M_f^- C_{T_-}, \tag{4.5}$$

где нормаль $\mathbf{n}_f$ направлена из ячейки T_+ в ячейку T_-, а M_f^+ и M_f^- – некоторые коэффициенты. В случае линейного метода конечных объёмов эти коэффициенты фиксированы и равны друг другу. В случае нелинейного метода конечных объёмов коэффициенты могут быть различными и могут зависеть от значений концентрации в окружающих ячейках. Для внешних граней с граничными условиями типа Дирихле, $f \in \Gamma_D$, поток представляется в форме похожей на (4.5) с явным значением для одной из концентраций. Для задачи Дирихле, $\Gamma_D = \partial\Omega$, подставляя (4.5) в (4.4), мы получим систему из

$N_{\mathcal{T}}$ уравнений с $N_{\mathcal{T}}$ неизвестными C_T. Граничные условия типа Дирихле и Неймана подробно рассматриваются в разделе 4.2.3

4.2.1. Обозначения

Для каждой ячейки $T \in \mathcal{T}$ определим точку коллокации $\mathbf{x}_T$ в центре масс ячейки T. Для каждой грани $f \in \mathcal{F}_B \cup \mathcal{F}_I$ обозначим её центр масс как $\mathbf{x}_f$, и поставим в соответствие точки коллокации $\mathbf{x}_f$ для граней $f \in \mathcal{F}_B \cup \mathcal{F}_J$. Также определим точки коллокации в центрах рёбер $\mathbf{x}_e$ для рёбер $e \in \mathcal{E}_f$, $f \in \mathcal{F}_B \cup \mathcal{F}_J$.

Точки коллокации на гранях и на рёбрах будем называть *вспомогательными* точками коллокации. Они введены для удобства записи математических выражений в дальнейшем изложении, и не войдут в конечную систему алгебраических уравнений, хотя и будут влиять на коэффициенты системы. Точки коллокации в центрах масс ячеек мы будем называть *первичными* точками коллокации, дискретные значения концентрации именно в этих точках образуют вектор неизвестных в системе алгебраических уравнений.

Для каждой ячейки T определим множество Σ_T соседних точек коллокации следующим образом. Сначала в Σ_T добавляется точка $\mathbf{x}_T$. Далее, для каждой грани $f \in \mathcal{F}_T \setminus (\mathcal{F}_B \cup \mathcal{F}_J)$ добавляется точка коллокации $\mathbf{x}_{T'_f}$ ячейки T'_f, соседствующей с T через грань f. И наконец, для остальных граней $f \in \mathcal{F}_T \cap (\mathcal{F}_B \cup \mathcal{F}_J)$ мы добавляем точки коллокации $\mathbf{x}_f$. Мощность множества (или количество элементов) Σ_T обозначим как $N(\Sigma_T)$.

Аналогичным образом для каждой грани $f \in \mathcal{F}_B \cup \mathcal{F}_J$ ячейки T определим множество $\Sigma_{f,T}$ соседних точек коллокации. Начиная с $\Sigma_{f,T} = \{\mathbf{x}_f, \mathbf{x}_T\}$, мы добавляем те точки Σ_T, которые являются центрами масс ячеек или граней, соседних с f. Мощность множества $\Sigma_{f,T}$ обозначим как $N(\Sigma_{f,T})$.

Предположим, что для каждой пары ячейка-грань $T \to f$, $T \in \mathcal{T}$,

$f \in \mathcal{F}_T$, существуют три точки $\mathbf{x}_{f,1}$, $\mathbf{x}_{f,2}$, и $\mathbf{x}_{f,3}$ из множества Σ_T, такие, что выполнено следующее условие (см. Рис. 4.1):

Вектор конормали $\ell_f = \mathbb{K}(\mathbf{x}_f)\mathbf{n}_f$, отложенный от $\mathbf{x}_T$, лежит в трёхгранном угле, образованном векторами

$$\mathbf{t}_{f,1} = \mathbf{x}_{f,1} - \mathbf{x}_T, \quad \mathbf{t}_{f,2} = \mathbf{x}_{f,2} - \mathbf{x}_T, \quad \mathbf{t}_{f,3} = \mathbf{x}_{f,3} - \mathbf{x}_T, \tag{4.6}$$

и

$$\frac{1}{|\ell_f|}\ell_f = \frac{\alpha_f}{|\mathbf{t}_{f,1}|}\mathbf{t}_{f,1} + \frac{\beta_f}{|\mathbf{t}_{f,2}|}\mathbf{t}_{f,2} + \frac{\gamma_f}{|\mathbf{t}_{f,3}|}\mathbf{t}_{f,3}, \tag{4.7}$$

где $\alpha_f \geq 0$, $\beta_f \geq 0$, $\gamma_f \geq 0$.

Коэффициенты α_f, β_f, γ_f могут быть вычислены следующим способом:

$$\alpha_f = \frac{D_{f,1}}{D_f}, \qquad \beta_f = \frac{D_{f,2}}{D_f}, \qquad \gamma_f = \frac{D_{f,3}}{D_f}, \tag{4.8}$$

где

$$D_f = \frac{\left|\mathbf{t}_{f,1} \;\; \mathbf{t}_{f,2} \;\; \mathbf{t}_{f,3}\right|}{|\mathbf{t}_{f,1}||\mathbf{t}_{f,2}||\mathbf{t}_{f,3}|}, \; D_{f,1} = \frac{\left|\ell_f \;\; \mathbf{t}_{f,2} \;\; \mathbf{t}_{f,3}\right|}{|\ell_f||\mathbf{t}_{f,2}||\mathbf{t}_{f,3}|},$$
$$D_{f,2} = \frac{\left|\mathbf{t}_{f,1} \;\; \ell_f \;\; \mathbf{t}_{f,3}\right|}{|\mathbf{t}_{f,1}||\ell_f||\mathbf{t}_{f,3}|}, \; D_{f,3} = \frac{\left|\mathbf{t}_{f,1} \;\; \mathbf{t}_{f,2} \;\; \ell_f\right|}{|\mathbf{t}_{f,1}||\mathbf{t}_{f,2}||\ell_f|},$$

и $|\mathbf{a}\ \mathbf{b}\ \mathbf{c}|$ означает определитель матрицы 3×3, составленной из векторов $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$.

Тройку векторов $(\mathbf{t}_{f,1}, \mathbf{t}_{f,2}, \mathbf{t}_{f,3})$ будем для краткости называть триплетом.

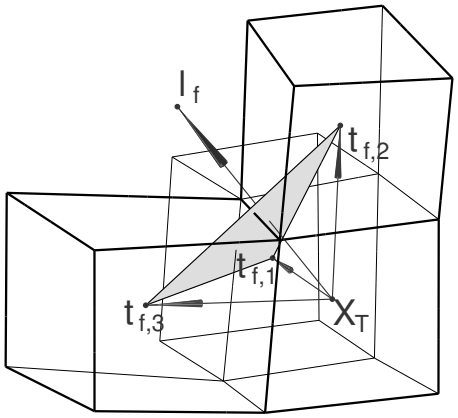


Рис. 4.1. Вектор конормали и векторы триплета.

Аналогично, предположим, что для каждой пары грань-ячейка $f \to T$, $f \in \mathcal{F}_B \cup \mathcal{F}_J$, $T : \ f \in \mathcal{F}_T$ существуют три точки $\mathbf{x}_{f,1}$, $\mathbf{x}_{f,2}$, и $\mathbf{x}_{f,3}$ из множества $\Sigma_{f,T}$, такие, что вектор $\ell_{f,T} = -\mathbb{K}_T(\mathbf{x}_f)\mathbf{n}_f$, отложенный от $\mathbf{x}_f$, лежит в трёхгранном угле, образованном триплетом

$$\mathbf{t}_{f,1} = \mathbf{x}_{f,1} - \mathbf{x}_f, \quad \mathbf{t}_{f,2} = \mathbf{x}_{f,2} - \mathbf{x}_f, \quad \mathbf{t}_{f,3} = \mathbf{x}_{f,3} - \mathbf{x}_f, \tag{4.9}$$

и выполнены условия (4.7), $\alpha_f \geq 0$, $\beta_f \geq 0$, $\gamma_f \geq 0$.

Предложим простой алгоритм для поиска триплета, удовлетворяющего (4.7) с неотрицательными коэффициентами. Алгоритм рассматривает случай пары ячейка-грань $T \to f$. Случай пары грань-ячейка $f \to T$ получается из Алгоритма 4.1 (см. стр. 109) заменой $\mathbf{x}_T$ и Σ_T на $\mathbf{x}_f$ и $\Sigma_{f,T}$, соответственно.

В общем случае для произвольной многогранной сетки необходимого триплета может не существовать. В этом случае множество соседних точек коллокации расширяется. Для случая пары ячейка-грань $T \to f$ расширение проводится следующим образом. Сначала для каждой грани $f \in \mathcal{F}_T \cap (\mathcal{F}_J \cup \mathcal{F}_B)$ мы добавляем к Σ_T точки коллокации $\mathbf{x}_e$, $e \in \mathcal{E}_f$. Далее, для граней $f \in \mathcal{F}_T \setminus (\mathcal{F}_J \cup \mathcal{F}_B)$ мы добавляем к Σ_T точки коллокации $\mathbf{x}_{T'_e}$ ячеек T'_e, имеющих общее ребро $e \in \mathcal{E}_f$, и таких, которые могут быть соединены с T кусочно-линейным путём $\{\mathbf{x}_{T'_e}, \ldots, \mathbf{x}_T\}$, проходящим через центры масс граней. Причём этот путь должен проходить только через ячейки с общим ребром e, и не пересекать грани из $\mathcal{F}_J$. Для случая пары грань-ячейка $f \to T$ мы добавляем к $\Sigma_{f,T}$ точки коллокации рёбер $\mathbf{x}_e$, $e \in \mathcal{E}_f$.

4.2.2. Двухточечная дискретизация диффузионного потока

Пусть f – внутренняя грань, $f \in \mathcal{F}_I$. Обозначим через T_+ и T_- две соседние с f ячейки, будем считать, что нормаль к грани $\mathbf{n}_f$ является внешней к T_+. Пусть $\mathbf{x}_\pm$ (или $\mathbf{x}_{T_\pm}$) – точки коллокации соответствующих ячеек $T_\pm$, а $C_\pm$ (или $C_{T_\pm}$) – дискретные значения концентрации в $T_\pm$.

Алгоритм 4.1 Алгоритм поиска триплета для пары ячейка-грань $T \to f$

1: Построить множество точек на единичной сфере $\bar{\mathbf{x}}_{\ell_f} = \mathbf{x}_T + |\ell_f|^{-1}\ell_f$, $\bar{\mathbf{x}}_i = \mathbf{x}_T + \bar{\mathbf{t}}_i$, $\bar{\mathbf{t}}_i = \mathbf{t}_i/|\mathbf{t}_i|$, $i = 1, \ldots, N(\Sigma_T)$.
2: Переупорядочить $\bar{\mathbf{x}}_i$ в соответствии с увеличением расстояния $|\bar{\mathbf{x}}_i\bar{\mathbf{x}}_{\ell_f}|$.
3: **для** $k = 3, \ldots, N(\Sigma_T)$ **начало цикла**
4: **для** $j = 2, \ldots, k-1$ **начало цикла**
5: **для** $i = 1, \ldots, j-1$ **начало цикла**
6: Вычислить коэффициенты (4.8) для векторов $\bar{\mathbf{t}}_i$, $\bar{\mathbf{t}}_j$, $\bar{\mathbf{t}}_k$.
7: **если** все коэффициенты неотрицательны, **тогда**
8: **если** коэффициенты не превосходят 1, **тогда** перейти к 16.
9: **иначе** добавить триплет $\{i, j, k\}$ в множество Σ_T^*.
10: **конец если**
11: **конец если**
12: Если все коэффициенты неотрицательные, то прервать цикл.
13: **конец цикла**
14: **конец цикла**
15: **конец цикла**
16: Выбрать из Σ_T^* триплет $\{i, j, k\}$ с минимальным значением $\max\{\alpha_f, \beta_f, \gamma_f\}$.
17: Положить $\mathbf{t}_{f,1} = |\mathbf{x}_i - \mathbf{x}_T|\bar{\mathbf{t}}_i$, $\mathbf{t}_{f,2} = |\mathbf{x}_j - \mathbf{x}_T|\bar{\mathbf{t}}_j$, $\mathbf{t}_{f,3} = |\mathbf{x}_k - \mathbf{x}_T|\bar{\mathbf{t}}_k$.

Начнём со случая $f \notin \mathcal{F}_J$, и обозначим $\mathbb{K}_f = \mathbb{K}(\mathbf{x}_f)$. Для простоты обозначим $T = T_+$. Используя предыдущие обозначения, определение производной по направлению,

$$\frac{\partial c}{\partial \ell_f}|\ell_f| = \nabla c \cdot (\mathbb{K}_f\, \mathbf{n}_f),$$

и предположение (4.7), мы можем записать

$$\mathbf{q}_f \cdot \mathbf{n}_f = -\frac{|\ell_f|}{|f|}\int\limits_f \frac{\partial c}{\partial \ell_f}\,\mathrm{d}s = -\frac{|\ell_f|}{|f|}\int\limits_f \left(\alpha_f \frac{\partial c}{\partial \mathbf{t}_{f,1}} + \beta_f \frac{\partial c}{\partial \mathbf{t}_{f,2}} + \gamma_f \frac{\partial c}{\partial \mathbf{t}_{f,3}}\right)\,\mathrm{d}s. \quad (4.10)$$

Заменив производные по направлениям на конечные разности, мы получим

$$\int_f \frac{\partial c}{\partial \mathbf{t}_{f,i}}\,\mathrm{d}s = \frac{C_{f,i} - C_T}{|\mathbf{x}_{f,i} - \mathbf{x}_T|}\,|f| + O(h_T^2), \quad i = 1,2,3, \tag{4.11}$$

где h_T – диаметр ячейки T. Используя конечно-разностное представление (4.11), мы преобразуем формулу (4.10) к виду:

$$\mathbf{q}_f^h \cdot \mathbf{n}_f = -|\ell_f| \left(\frac{\alpha_f}{|\mathbf{t}_{f,1}|}(C_{f,1} - C_T) + \frac{\beta_f}{|\mathbf{t}_{f,2}|}(C_{f,2} - C_T) + \frac{\gamma_f}{|\mathbf{t}_{f,3}|}(C_{f,3} - C_T) \right). \tag{4.12}$$

В этой формуле участвуют четыре, а не два значения дискретных концентраций. Чтобы получить двухточечную дискретизацию диффузионного потока, рассмотрим ячейку T_- и выведем аналогичную формулу для того же потока через грань f. Чтобы отличать формулы для ячеек T_+ и T_-, добавим к коэффициентам индексы $\pm$, а индекс f для простоты опустим. Так как $\mathbf{n}_f$ является внутренней нормалью для T_-, то знак во второй формуле изменится на противоположный. В общем виде (4.12) запишется следующим образом:

$$\mathbf{q}_\pm^h \cdot \mathbf{n}_f = \mp|\ell_f| \left(\frac{\alpha_\pm}{|\mathbf{t}_{\pm,1}|}\,(C_{\pm,1} - C_\pm) + \frac{\beta_\pm}{|\mathbf{t}_{\pm,2}|}\,(C_{\pm,2} - C_\pm) + \frac{\gamma_\pm}{|\mathbf{t}_{\pm,3}|}\,(C_{\pm,3} - C_\pm) \right), \tag{4.13}$$

где $\alpha_\pm$, $\beta_\pm$ и $\gamma_\pm$ вычисляются по формулам (4.8), и $C_{\pm,i}$ соответствуют концентрациям в точках $\mathbf{x}_{\pm,i} \in \Sigma_{T_\pm}$.

Запишем новую дискретизацию диффузионного потока как линейную комбинацию $\mathbf{q}_\pm^h \cdot \mathbf{n}_f$ с неотрицательными весами $\mu_\pm$:

$$\begin{aligned}
\mathbf{q}_f^h \cdot \mathbf{n}_f &= \mu_+\,\mathbf{q}_+^h \cdot \mathbf{n}_f + \mu_-\,\mathbf{q}_-^h \cdot \mathbf{n}_f \\
&= \mu_+|\ell_f| \left(\tfrac{\alpha_+}{|\mathbf{t}_{+,1}|} + \tfrac{\beta_+}{|\mathbf{t}_{+,2}|} + \tfrac{\gamma_+}{|\mathbf{t}_{+,3}|} \right) C_+ - \mu_-|\ell_f| \left(\tfrac{\alpha_-}{|\mathbf{t}_{-,1}|} + \tfrac{\beta_-}{|\mathbf{t}_{-,2}|} + \tfrac{\gamma_-}{|\mathbf{t}_{-,3}|} \right) C_- \\
&- \mu_+|\ell_f| \left(\tfrac{\alpha_+}{|\mathbf{t}_{+,1}|} C_{+,1} + \tfrac{\beta_+}{|\mathbf{t}_{+,2}|} C_{+,2} + \tfrac{\gamma_+}{|\mathbf{t}_{+,3}|} C_{+,3} \right) \\
&+ \mu_-|\ell_f| \left(\tfrac{\alpha_-}{|\mathbf{t}_{-,1}|} C_{-,1} + \tfrac{\beta_-}{|\mathbf{t}_{-,2}|} C_{-,2} + \tfrac{\gamma_-}{|\mathbf{t}_{-,3}|} C_{-,3} \right).
\end{aligned} \tag{4.14}$$

Наложим условие на веса так, чтобы слагаемые в двух последних строчках (4.14) сократились, оставляя двухточечную дискретизацию для диффузионного потока. Второе условие на веса накладывается из соображения аппроксимации настоящего потока. Эти условия приводят нас к системе уравнений

$$\begin{cases} -\mu_+ d_+ + \mu_- d_- &= 0, \\ \mu_+ + \mu_- &= 1, \end{cases} \tag{4.15}$$

где

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

Так как коэффициенты $d_\pm$ зависят не только от геометрии, но и от значений концентрации, то и веса $\mu_\pm$ также зависят. Таким образом, получившаяся двухточечная дискретизация является *нелинейной*.

Решение системы (4.15) записывается в явном виде. Если $C \geq 0$, то $d_\pm \geq 0$. Если $d_\pm = 0$, положим $\mu_+ = \mu_- = \frac{1}{2}$. Иначе,

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

Из этого следует, что и веса $\mu_\pm$ являются неотрицательными. Подставляя эти выражения в (4.14), мы получим формулу для двухточечной дискретизации диффузионного потока (4.5) с коэффициентами

$$M_f^\pm = \mu_\pm |\ell_f| (\alpha_\pm / |\mathbf{t}_{\pm,1}| + \beta_\pm / |\mathbf{t}_{\pm,2}| + \gamma_\pm / |\mathbf{t}_{\pm,3}|). \tag{4.16}$$

Если какое-то из значений $C_{+,i}$, $(C_{-,i})$, $i = 1, 2, 3$, определено в той же точке коллокации, что и C_-, (C_+), тогда соответствующий коэффициент перед $C_{+,i}$, $(C_{-,i})$, выносятся из двух последних строчек (4.14), и включается во вторую строчку (4.14). Соответственно, коэффициенты $d_\pm$, $\mu_\pm$ и $M_f^\pm$ пересчитываются. В частности, для оператора Лапласа на регулярных кубических сетках такой подход приведёт к классическому линейному шаблону 6-1-1-1-1-1-1.

Перейдём к случаю $f \in \mathcal{F}_J$, в котором $\mathbb{K}_+(\mathbf{x}_f)$ и $\mathbb{K}_-(\mathbf{x}_f)$ отличаются. Добавим в рассмотрение вспомогательную точку коллокации в $\mathbf{x}_f$ и выпишем две двухточечных дискретизации для каждой ячейки T_+ и T_- независимо:

$$(\mathbf{q}_f^h \cdot \mathbf{n}_f)_+ = N^+ C_+ - N_f^+ C_f, \tag{4.17}$$

$$-(\mathbf{q}_f^h \cdot \mathbf{n}_f)_- = N^- C_- - N_f^- C_f. \tag{4.18}$$

Неотрицательные коэффициенты N^+, N_f^+, N^-, N_f^- вычисляются аналогично коэффициентам из (4.16) для внешних по отношению к $T_\pm$ конормалей к грани f, $\ell_\pm = \mp \mathbb{K}_\pm(\mathbf{x}_f)\,\mathbf{n}_f$. Эти коэффициенты зависят от значений дискретных концентраций в точках коллокации из $\Sigma_{T_\pm}$, $\Sigma_{f,T_\pm}$. Член C_f можно исключить из (4.17)-(4.18), используя непрерывность диффузионного потока,

$$C_f = (N^+ C_+ + N^- C_-)/(N_f^+ + N_f^-). \tag{4.19}$$

Полученная двухточечная дискретизация диффузионного потока (4.5) будет иметь следующие коэффициенты

$$M_f^\pm = N^\pm N_f^\mp / (N_f^+ + N_f^-). \tag{4.20}$$

Если $N_f^\pm = 0$, то положим $M_f^\pm = N^\pm/2$ и $C_f = (C_+ + C_-)/2$.

4.2.3. Дискретизация граничных условий

Сначала рассмотрим граничные условия типа Дирихле на грани $f \in \mathcal{F}_B^D$. Положим

$$C_f = \bar{g}_{D,f} = \frac{1}{|f|} \int\limits_f g_D \,\mathrm{d}s, \tag{4.21}$$

и для каждого ребра $e \in \mathcal{E}_f$

$$C_e = \bar{g}_{D,e} = \frac{1}{|e|} \int\limits_e g_D \,\mathrm{d}x. \tag{4.22}$$

Пусть T – ячейка с гранью f. Будем рассматривать грань f как фиктивную ячейку с нулевым объёмом. Заменив C_+ и C_- на C_T и C_f, а Σ_{T_+}, Σ_{T_-} на Σ_T, $\Sigma_{f,T}$ соответственно, получим

$$\mathbf{q}_f^h \cdot \mathbf{n}_f = M_f^+ C_T - M_f^- C_f, \tag{4.23}$$

где коэффициенты $M_f^\pm$ вычисляются по (4.16).

В случае граничных условий типа Неймана для внешней грани $f \in \mathcal{F}_B^N$ поток вычисляется по формуле

$$\mathbf{q}_f^h \cdot \mathbf{n}_f = \bar{g}_{N,f}|f|, \tag{4.24}$$

где $\bar{g}_{N,f}$ – среднее значение g_N на грани f.

4.2.4. Восстановление решения в вспомогательных точках коллокации

Коэффициенты $M_f^\pm$ в (4.16), (4.20), (4.23) могут зависеть от значений дискретного решения C_f и C_e в вспомогательных точках коллокации $\mathbf{x}_f$, $f \in \mathcal{F}_B \cup \mathcal{F}_J$, и $\mathbf{x}_e$, $e \in \mathcal{E}_f$. В свою очередь дискретная система в методе конечных объёмов включает в себя только концентрации C_T для первичных точек коллокации. Значения C_f, C_e, $f \in \mathcal{F}_B^D$, $e \in \mathcal{E}_f$, вычисляются по (4.21), (4.22) из условий Дирихле. Значения C_f, $f \in \mathcal{F}_J$, восстанавливаются из (4.19). Оставшиеся значения, C_f, C_e, $f \in \mathcal{F}_B^N$, $e \in \mathcal{E}_f$, $e \notin \Gamma_D$, и C_e, $e \in \mathcal{E}_f$, $e \notin \Gamma_D$, $f \in \mathcal{F}_J$, необходимо восстановить из имеющихся данных.

Значения концентрации на гранях с граничным условием типа Неймана восстанавливаются с помощью (4.23) и (4.24). Коэффициенты $M_f^\pm$ могут зависеть от значений решения в первичных точках коллокации, от значений C_f, $f \in \mathcal{F}_J \cup \mathcal{F}_B$, и C_e, $e \in \mathcal{E}_f$. Концентрация C_f на гранях f, $f \in \mathcal{F}_J \cup \mathcal{F}_B$, восстанавливается из значений в ячейках на основе *физических* принципов, таких

как непрерывность диффузионного потока или известность значения диффузионного потока. При этом для восстановления значения концентрации на грани C_f нам в свою очередь может понадобиться значение концентрации на ребре C_e в точке коллокации $\mathbf{x}_e$, $e \in \mathcal{E}_f$, $f \in \mathcal{F}_J \cup \mathcal{F}_B^N$, $e \notin \Gamma_D$. Для каждого такого ребра предлагается использовать в качестве C_e среднее арифметическое C_f всех граней $f \in \mathcal{F}_B^N \cup \mathcal{F}_J$ с общим ребром e.

4.3. Схема дискретизации и анализ её монотонности

Для каждой ячейки $T \in \mathcal{T}$ запишем уравнение (4.4) в виде

$$\sum_{f \in \mathcal{F}_T} \chi(T, f)\, \mathbf{q}_f^h \cdot \mathbf{n}_f = \int_T f \,\mathrm{d}x, \tag{4.25}$$

где $\chi(T, f) = sign(\mathbf{n}_f \cdot \mathbf{n}_T(\mathbf{x}_f))$. Подставляя формулу для двухточечной дискретизации диффузионного потока (4.5) с неотрицательными коэффициентами, полученными по (4.16) и (4.20), используя уравнения (4.21)-(4.22) и (4.23)-(4.24) для исключения значений концентрации на внешних гранях, а также используя арифметическое осреднение восстановленных на гранях концентраций для рёбер $e \in \mathcal{E}_f$, $f \in \mathcal{F}_B^N \cup \mathcal{F}_J$, $e \notin \Gamma_D$, мы получим нелинейную систему из $N_{\mathcal{T}}$ уравнений

$$\mathbb{M}(C)C = G(C). \tag{4.26}$$

Матрица $\mathbb{M}(C)$ может быть представлена как матрица, полученная из совокупности 2×2 блоков

$$\mathbb{M}_f(C) = \begin{pmatrix} M_f^+(C) & -M_f^-(C) \\ -M_f^+(C) & M_f^-(C) \end{pmatrix} \tag{4.27}$$

для внутренних граней, и 1×1 блоков $\mathbb{M}_f(C) = M_f^+(C)$ для внешних граней $f \in \mathcal{F}_B^D$ (подробнее см. Алгоритм 4.2). Правая часть системы $G(C)$ составля-

ется из внешних источников и граничных данных:

$$G_T(C) = \int\limits_T g \,\mathrm{d}x + \sum_{f \in \mathcal{F}_B^D \cap \mathcal{F}_T} M_f^-(C)\bar{g}_{D,f} - \sum_{f \in \mathcal{F}_B^N \cap \mathcal{F}_T} |f|\bar{g}_{N,f}, \qquad \forall T \in \mathcal{T}. \quad (4.28)$$

При условии $g \geq 0$, $g_D \geq 0$ и $g_N \leq 0$, компоненты вектора G будут также неотрицательными. Для решения нелинейной системы (4.26) используется метод Пикара (см. Алгоритм 4.2, стр. 116).

Система линейных уравнений на шаге 8 с несимметричной матрицей $\mathbb{M}_k = \mathbb{M}(C^k, C_f^k, C_e^k)$ и правой частью $G^k = G(C^k, C_f^k, C_e^k)$ решается с помощью стабилизированного метода бисопряжённых градиентов (BiCGStab) [38] с использованием предобуславливателя ILU второго порядка [19]. Итерации метода BiCGStab останавливаются, когда относительная норма невязки системы линейных уравнений становится меньше ε_{lin}.

Докажем следующую теорему.

Теорема 4.1. *Пусть $g \geq 0$, $g_D \geq 0$, $g_N \leq 0$ и $\Gamma_D \neq \emptyset$ в (4.1). Если $C^0 \geq 0$ и линейные системы в методе Пикара решаются точно, то $C^k \geq 0$ для $k \geq 1$.*

Покажем, что матрица $\mathbb{M}_k$ монотонна при условии $C^k \geq 0$. Из построения следует, что вспомогательные неизвестные $C_f^k \geq 0$, $C_e^k \geq 0$, и коэффициенты $M_f^{\pm}(C^k)$ положительны. Таким образом, все диагональные элементы матрицы $\mathbb{M}_k$ положительны, а все внедиагональные элементы неположительны. Из структуры блока для грани (4.27) следует, что сумма элементов по столбцам матрицы $\mathbb{M}_k$ неотрицательна. Более того, для ячеек, имеющих внешние грани с граничными условиями типа Дирихле, соответствующая сумма по столбцу будет строго положительной. Для связной сетки, матрицы $\mathbb{M}_k$ и $\mathbb{M}_k^{\mathrm{T}}$ неразложимы, так как их направленные графы сильно связны [46]. При этих условиях из широко известного результата алгебры [39] следует, что матрица $\mathbb{M}_k^{\mathrm{T}}$ является M-матрицей, и все элементы матрицы $(\mathbb{M}_k^{\mathrm{T}})^{-1}$

положительны. Так как операции инверсии и транспонирования коммутируют, $(\mathbb{M}_k^T)^{-1} = (\mathbb{M}_k^{-1})^T$, заключаем, что $\mathbb{M}_k$ монотонна. Теорема доказана.

Алгоритм 4.2 Сборка и решение нелинейной системы уравнений (4.26)

1: Для каждой пары ячейка-грань $T \to f$, $f \in \mathcal{F}_T$, и для каждой пары грань-ячейка $f \to T$, $f \in \mathcal{F}_J \cup \mathcal{F}_B$, найти триплеты $\mathbf{t}_{f,1}$, $\mathbf{t}_{f,2}$, $\mathbf{t}_{f,3}$, удовлетворяющие условиям (4.6)-(4.7) и (4.9), (4.7), соответственно.

2: Выбрать начальные векторы $C^0 \in \Re^{N_\mathcal{T}}$ и $C_f^0 \in \Re^{N_{\mathcal{F}_J}+N_{\mathcal{F}_B^N}}$ с неотрицательными значениями, и малый параметр $\varepsilon_{\text{non}} > 0$.

3: Вычислить значения концентрации C_e^0 в вспомогательных точках коллокации на рёбрах с помощью (4.22) или с помощью арифметического осреднения соседних значений C_f^0.

4: **для** $k = 0, 1, \ldots,$ **начало цикла**

5: Составить общую матрицу $\mathbb{M}_k = \mathbb{M}(C^k, C_f^k, C_e^k)$ из блоков для граней $\mathbb{M}_f(C^k, C_f^k, C_e^k)$, используя (4.16) для $f \in \mathcal{F}_B^D \cup \mathcal{F}_I \setminus \mathcal{F}_J$, и (4.20) для $f \in \mathcal{F}_J$.

6: Вычислить вектор правой части $G^k = G(C^k, C_f^k, C_e^k)$ из (4.28).

7: Остановиться, если $\|\mathbb{M}_k C^k - G^k\| \le \varepsilon_{\text{non}} \|\mathbb{M}_0 C^0 - G^0\|$

8: Решить $\mathbb{M}_k C^{k+1} = G^k$.

9: Вычислить значения концентрации C_f^{k+1} в вспомогательных точках коллокации на гранях $f \in \mathcal{F}_J \cup \mathcal{F}_B$, используя (4.19), (4.21), (4.23)-(4.24) и значения C^{k+1}, C_f^k, C_e^k.

10: Вычислить значения концентрации C_e^{k+1} в вспомогательных точках коллокации на рёбрах с помощью (4.22) или с помощью арифметического осреднения соседних значений C_f^{k+1}.

11: **конец цикла**

Из положительности диагональных и неположительности внедиагональных элементов следует, что $\mathbb{M}_k$ является также и M-матрицей.

Предложенный метод конечных объёмов по построению точен для кусочно-линейных решений, и имеет второй порядок аппроксимации. Таким образом, можно ожидать от метода второй порядок сходимости для концентрации C, и как минимум первый порядок сходимости для потока.

4.4. Результаты экспериментов

Будем использовать L_2-нормы для оценки ошибок дискретизации для концентрации c и для диффузионного потока $\mathbf{q}$:

$$\varepsilon_2^c = \left[\frac{\sum\limits_{T\in\mathcal{T}} \left(c(\mathbf{x}_T) - C_T\right)^2 |T|}{\sum\limits_{T\in\mathcal{T}} \left(c(\mathbf{x}_T)\right)^2 |T|} \right]^{1/2} \quad \text{и} \quad \varepsilon_2^q = \left[\frac{\sum\limits_{f\in\mathcal{F}_I\cup\mathcal{F}_B} \left((\mathbf{q}_f - \mathbf{q}_f^h)\cdot\mathbf{n}_f\right)^2 |V_f|}{\sum\limits_{f\in\mathcal{F}_I\cup\mathcal{F}_B} \left(\mathbf{q}_f\cdot\mathbf{n}_f\right)^2 |V_f|} \right]^{1/2},$$

где $|V_f|$ – характерный объём грани f. Более точно, $|V_f|$ – среднее арифметическое объёмов ячеек с гранью f. Нелинейные итерации прекращались при достижении относительной нормы невязки $\varepsilon_{\text{non}} = 10^{-9}$. Критерием остановки решения системы линейных уравнений было выбрано достижение относительной нормы невязки $\varepsilon_{\text{lin}} = 10^{-12}$.

Были рассмотрены три класса многогранных сеток для единичного куба $[0,1]^3$: гексаэдральные, призматические и тетраэдральные. Все сетки предполагаются квази-равномерными.

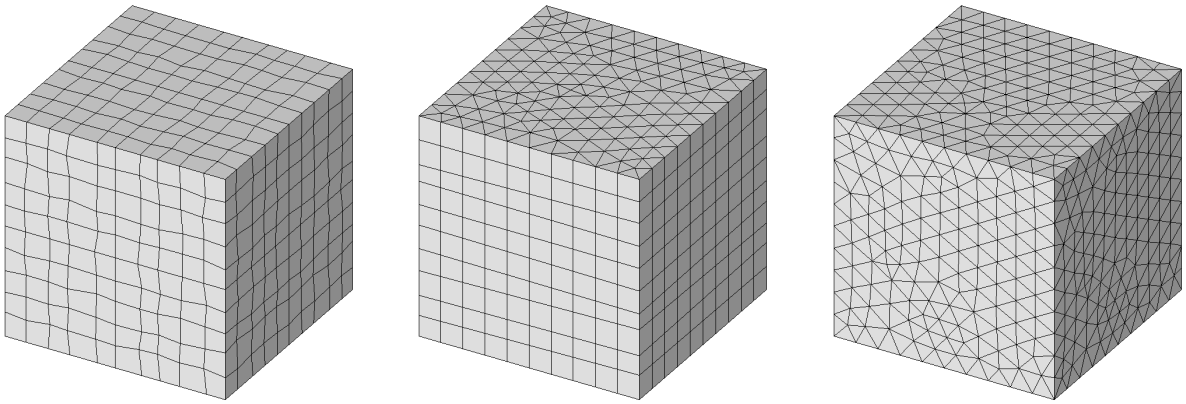
Гексаэдральные сетки были получены деформацией из регулярных кубических сеток с помощью сдвижки узлов. В каждой плоскости $x = 0.5$, $y = 0.5$ и $z = 0.5$ вершины сетки сдвигались в случайном направлении внутри соответствующей плоскости. Положение остальных вершин в сетке однозначно восстанавливалось из условия планарности граней. Расстояние и направление, в котором сдвигались вершины, выбиралось случайно. Максимальное

расстояние, на которое сдвигались вершины, не превышало $0.3h$, где h – размер элементов в исходной кубической сетке.

Призматические сетки были получены как тензорное произведение двумерной квази-равномерной неструктурированной треугольной xy-сетки и одномерной z-сетки, размер элементов сетки – h. Дополнительно, плоскости, параллельные xy, были наклонены в произвольные стороны так, чтобы при этом не возникало пересечений, а высота каждой призматической ячейки была в пределах между $0.75h$ и $1.25h$.

Тетраэдральные сетки были квази-равномерными неструктурированными с размером элементов h.

Типичные примеры сеток всех трёх классов показаны на Рис. 4.2.



($а$) ($б$) ($в$)

Рис. 4.2. Примеры сеток: ($а$) гексаэдральная, ($б$) треугольная призматическая, ($в$) тетраэдральная.

При необходимости сетки подстраивались таким образом, чтобы отражать необходимые разрывы диффузионного тензора, или граничные условия внутри единичного куба.

4.4.1. Монотонность схемы

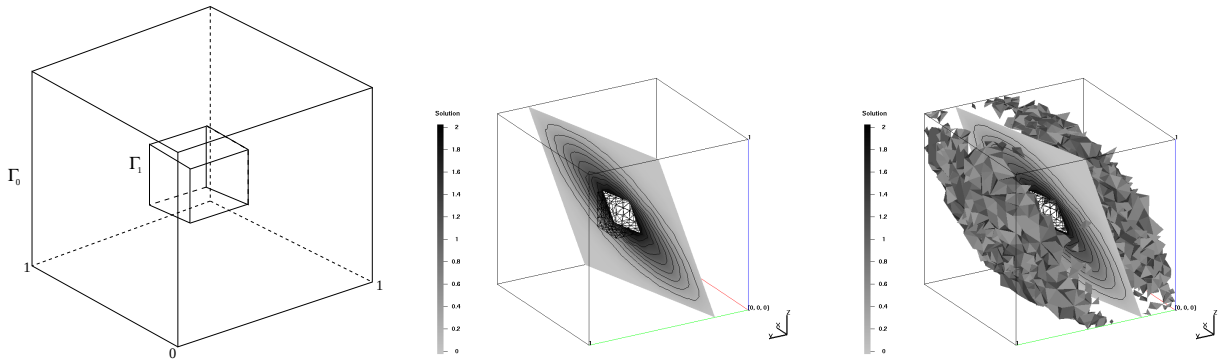
Численные эксперименты в этом разделе проверяют предположение Теоремы 4.1. Будут рассмотрены два теста с сильно анизотропными тензорами диффузии, в обоих случаях сеточное решение получается неотрицательным, однако дискретный принцип максимума при этом может нарушаться.

Задача Дирихле

Рассмотрим задачу (4.1) заданную на единичном кубе с кубической дыркой, $\Omega = (0,1)^3/[0.4,0.6]^3$. Граница области Ω состоит из двух несвязных частей, внутренней Γ_0 и внешней Γ_1 (см. Рис. 4.3, *а*). Положим $\Gamma_N = \emptyset$, $g = 0$, $g_D = 2$ на Γ_0, $g_D = 0$ на Γ_1, в качестве диффузионного тензора $\mathbb{K}$ выберем сильно анизотропный полный тензор

$$\mathbb{K} = R_z(-\theta_z)R_y(-\theta_y))R_x(-\theta_x)\mathrm{diag}(k_1,k_2,k_3)R_x(\theta_x)R_y(\theta_y)R_z(\theta_z) \qquad (4.29)$$

где $k_1 = 100, k_2 = 10, k_3 = 1$, $\theta_x = \pi/3$, $\theta_y = \pi/4$, $\theta_z = \pi/6$, и $R_a(\alpha)$ – матрица поворота на угол α в плоскости, ортогональной оси oa.



(*а*) (*б*) (*в*)

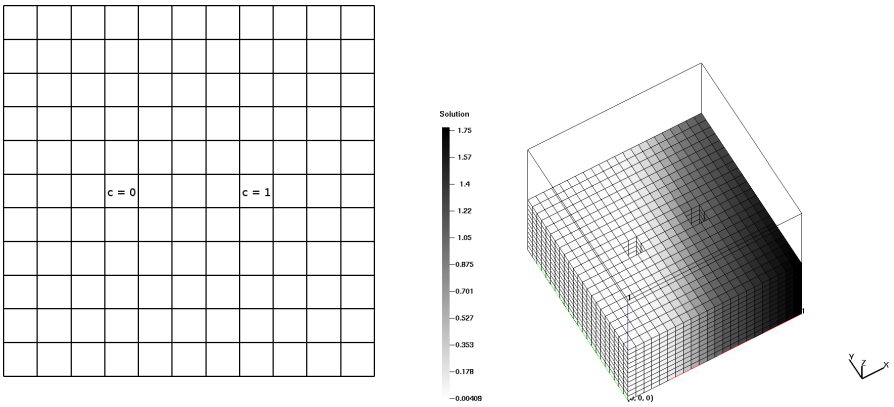
Рис. 4.3. Тест на монотонность, задача Дирихле. (*а*) эскиз области. Сечения полученных решений с помощью (*б*) нелинейного метода конечных объёмов, (*в*) смешанных конечных элементов. На рисунке показаны элементы, значения в которых меньше чем -10^{-3}. Тетраэдральная сетка с $h = 1/20$.

Согласно принципу максимума для эллиптических уравнений в частных производных, решение задачи должно лежать между 0 и 2. Сеточное решение, полученное с помощью предлагаемого нелинейного метода конечных объёмов, на всех рассмотренных классах сеток является неотрицательным во всей области Ω (см. Рис. 4.3, *б*). Помимо прочего, полученное решение не нарушает и дискретный принцип максимума. Отметим, что дискретизация на основе смешанных конечных элементов Равьера-Тома низшего порядка порождают большие области с отрицательным решением (см. Рис. 4.3, *в*). Аналогичное наблюдение верно и в двумерном случае задачи (4.1) для дискретизаций MFE и MPFA [25, 26].

Однородные условия типа Неймана на границе

Этот пример продемонстрирует, что сеточное решение нелинейного метода конечных объёмов может нарушать дискретный принцип максимума даже на регулярных кубических сетках.

Рассмотрим трёхмерный аналог задачи, описанной и исследованной в [1]. Возьмём единичный куб с двумя вертикальными дырками P_1, P_2, $\Omega = (0,1)^3 \setminus (P_1 \cup P_2)$, где $P_i = S_i \times (0,1)$, $i = 1,2$, $S_1 = [3/11, 4/11] \times [5/11, 6/11]$, $S_2 = [7/11, 8/11] \times [5/11, 6/11]$. Граница области разбита на внешнюю часть Γ_N, на которой накладываются однородные условия типа Неймана, и две внутренних части $\Gamma_{D,1}$, $\Gamma_{D,2}$, на которых накладываются условия типа Дирихле: $g_D(\mathbf{x}) = 0$, $\mathbf{x} \in \Gamma_{D,1}$, $g_D(\mathbf{x}) = 1$, $\mathbf{x} \in \Gamma_{D,2}$ (см. Рис. 4.4, *а*). Анизотропный тензор задан формулой (4.29) с коэффициентами $k_1 = k_3 = 1, k_2 = 10^{-3}$, $\theta_x = \theta_y = 0$, $\theta_z = 67.5°$. Согласно принципу максимума для эллиптических уравнений, точное решение должно лежать между 0 и 1, и не должно иметь экстремумов на границе Γ_N.



(a) $(б)$

Рис. 4.4. Тест на выполнение дискретного принципа максимума: (a) эскиз области (плоскость xy), $(б)$ полученное решение ($h = 1/22$).

Таблица 4.1. Максимальное значение концентрации для задачи с однородными граничными условиями типа Неймана на внешней границе.

h	$1/11$	$1/22$	$1/44$	$1/88$
$\max C$	2.163	1.765	1.136	1.024

Решение, полученное с помощью нелинейного метода конечных объёмов, показано на Рис. 4.4, $б$. Оно неотрицательно в соответствии с Теоремой 4.1, но имеет забросы выше 1 около границы. Величина забросов быстро уменьшается по мере измельчения сетки, см. Таблицу 4.1.

4.4.2. Сходимость метода Пикара

Основные накладные расходы в нелинейном методе конечных объёмов состоят в итерационном решении нелинейной системы алгебраических уравнений. Метод Пикара гарантирует неотрицательность решения на каждой итерации.

Вернёмся к первой задаче, описанной в разделе 4.4.1, и проследим за количеством итераций Пикара, необходимых для уменьшения относительной

невязки до 10^{-3}, 10^{-6} и 10^{-9}. В качестве начального вектора выбирался вектор, составленный из единиц. Системы линейных уравнений решались до падения нормы невязки на 12 порядков. В Таблице 4.2 представлено количество итераций N_{it} для разных типов сеток и разных критериев остановки.

Таблица 4.2. Количество итераций Пикара для разных типов сеток и критериев остановки ε_{non}, $10^{-3}/10^{-6}/10^{-9}$.

$h\backslash$ Тип сетки	гексаэдральная	призматическая	тетраэдральная
1/10	6/18/30	7/25/43	6/23/42
1/20	7/30/54	8/37/69	9/39/74
1/40	7/48/95	9/61/127	8/67/137

Наблюдается быстрая скорость сходимости на первых итерациях, незначительный рост N_{it} при $h \to 0$ для $\varepsilon_{\text{non}} = 10^{-3}$, обратная зависимость N_{it} от h при $\varepsilon_{\text{non}} \to 0$ и слабая зависимость N_{it} от типа сетки.

4.4.3. Сходимость на гладком решении

В этом разделе мы проведём исследование скорости сходимости предложенной схемы на гладком решении. Пусть $\Omega = (0,1)^3$, $\Gamma_D = \partial\Omega$, и g получена подстановкой в уравнение (4.1) точного решения

$$c(x,y,z) = \frac{1}{3\pi^2}\sin(\pi x)\sin(\pi y)\sin(\pi z). \tag{4.30}$$

Граничные условия Дирихле, g_D, совпадают со следом $c(x,y,z)$ на Γ_D. Рассмотрим два варианта следующего анизотропного диффузионного тензора

$$\mathbb{K} = k(x,y,z)\cdot \operatorname{diag}(1,10,100).$$

В первом случае возьмём $k(x,y,z) \equiv 1$, и получим постоянный диффузионный тензор. Во втором случае возьмём $k(x,y,z) = 1 + 0.25\cos(x+y-z)$, и

получим непрерывный диффузионный тензор без резких скачков коэффициентов и направлений собственных векторов. В обоих случаях $\mathcal{F}_J = \varnothing$.

Результаты сходимости представлены в Таблицах 4.3, 4.4.

Таблица 4.3. Результаты сходимости для задачи с гладким решением и постоянным тензором диффузии.

Тип сетки	гексаэдральная		призматическая		тетраэдральная	
h	ε_2^C	ε_2^q	ε_2^C	ε_2^q	ε_2^C	ε_2^q
1/10	6.79e-3	1.39e-2	8.31e-3	6.85e-3	2.42e-2	5.72e-2
1/20	1.69e-3	4.45e-3	2.05e-3	2.26e-3	5.45e-3	3.05e-2
1/40	3.77e-4	1.51e-3	5.34e-4	8.09e-4	1.44e-3	1.47e-2
порядок сходимости	2.09	1.60	1.98	1.54	2.04	0.98

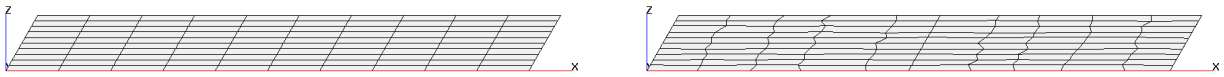
Таблица 4.4. Результаты сходимости для задачи с гладким решением и переменным непрерывным тензором диффузии.

Тип сетки	гексаэдральная		призматическая		тетраэдральная	
h	ε_2^C	ε_2^q	ε_2^C	ε_2^q	ε_2^C	ε_2^q
1/10	6.91e-3	1.38e-2	8.43e-3	6.86e-3	2.43e-2	5.73e-2
1/20	1.72e-3	4.42e-3	2.08e-3	2.26e-3	5.46e-3	3.04e-2
1/40	3.84e-4	1.49e-3	5.42e-4	8.02e-4	1.45e-3	1.45e-2
порядок сходимости	2.08	1.61	1.98	1.55	2.03	0.99

Отметим второй порядок сходимости скалярной неизвестной C в дискретной L_2 норме, и как минимум первый порядок сходимости нормальных составляющих диффузионного потока на гранях сетки.

4.4.4. Сходимость на анизотропных сетках

Рассмотрим задачу (4.1) с единичным тензором диффузии на сильно анизотропных сетках. Идея теста взята из [1]. Рассмотрим сжатие единичного куба вдоль оси OZ в k раз, $k = 10, 100$, и сдвиговый наклон его yz граней на угол $\frac{\pi}{6}$. Рассмотрим два типа квазиравномерных сеток в начальном единичном кубе, невозмущённые регулярные, и возмущённые, описанные в начале раздела 4.4. После предложенного аффинного преобразования мы получим анизотропные сетки с отношением анизотропии 0.1 (см. Рис. 4.5) и 0.01.



(a) $(б)$

Рис. 4.5. Анизотропные сетки с $h = 1/10$, отношение сторон 0.1, угол наклона – $\frac{\pi}{6}$. (a) невозмущённая сетка, $(б)$ возмущённая сетка.

Определим точное решение $c(x, y, z) = \cosh(\pi x)\cos(\pi z)$, порождающее нулевую правую часть и ненулевые граничные условия типа Дирихле.

Результаты сходимости представлены в Таблице 4.5. Порядок сходимости оценивался по двум последним строчкам.

Асимптотический второй порядок сходимости неизвестной C наблюдается только для сеток с анизотропным отношением 0.1. Для большой анизотропии сетки наблюдаемая скорость сходимости ниже второго порядка. В случае невозмущённых сеток наблюдается асимптотическое стремление к первому порядку, в случае возмущённых сеток сходимость демонстрирует асимптотическую зависимость h^β, $1 \leq \beta < 2$, $h \to 0$. Отметим, что сходимость для диффузионного потока в обоих случаях не меньше первого порядка.

Таблица 4.5. Результаты сходимости для задачи с анизотропными гексаэдральными сетками.

	Невозмущённые сетки				Возмущённые сетки			
	отношение 0.1		отношение 0.01		отношение 0.1		отношение 0.01	
h	ε_2^C	ε_2^q	ε_2^C	ε_2^q	ε_2^C	ε_2^q	ε_2^C	ε_2^q
1/10	1.02e-3	1.70e-1	1.55e-5	1.77	1.15e-3	1.47e-1	4.76e-4	2.03
1/20	4.35e-4	6.11e-2	1.17e-5	6.32e-1	5.00e-4	5.45e-2	2.04e-4	1.14
1/40	1.35e-4	2.03e-2	8.10e-6	2.24e-1	1.48e-4	1.97e-2	6.89e-5	4.60e-1
1/80	3.60e-5	6.58e-3	4.96e-6	8.08e-2	3.82e-5	6.40e-3	2.41e-5	1.97e-1
порядок	1.91	1.63	0.71	1.47	1.95	1.62	1.51	1.22

4.4.5. Сходимость с разрывным тензором диффузии

В этом разделе мы рассмотрим сходимость к решению задачи с разрывным тензором диффузии. Пусть область $\Omega = (0,1)^3$ разбита на две непересекающиеся подобласти $\Omega^{(1)} = \Omega \cap \{x < 0.5\}$, $\Omega^{(2)} = \Omega \cap \{x > 0.5\}$, с общей гранью в плоскости $x = 0.5$. Пусть тензор диффузии $\mathbb{K}$ испытывает разрыв на этой грани, $\mathbb{K}(\mathbf{x}) = \mathbb{K}^{(i)}$ при $\mathbf{x} \in \Omega^{(i)}$, где

$$\mathbb{K}^{(1)} = \begin{pmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{и} \quad \mathbb{K}^{(2)} = \begin{pmatrix} 10 & 3 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Спектральное разложение тензоров диффузии $\mathbb{K}^{(i)} = (W^{(i)})^T \Lambda^{(i)} W^{(i)}$ демонстрирует резкие скачки как в собственных значениях, так и в направлениях собственных векторов $\mathbb{K}(\mathbf{x})$:

$$\Lambda^{(1)} = \mathrm{diag}\{4, 2, 1\}, \qquad \Lambda^{(2)} \approx \mathrm{diag}\{10.908, 0.092, 1\},$$

$$W^{(1)} \approx \begin{pmatrix} 0.707 & 0.707 & 0 \\ -0.707 & 0.707 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad W^{(2)} \approx \begin{pmatrix} 0.957 & 0.290 & 0 \\ -0.290 & 0.957 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

В качестве точного решения задачи (4.1) с $\Gamma_D = \partial\Omega$ рассмотрим

$$c(\mathbf{x}) = \begin{cases} 1 - 2y^2 + 4xy + 2y + 6x, & \mathbf{x} \in \Omega^{(1)}, \\ 3.5 - 2y^2 + 2xy + x + 3y, & \mathbf{x} \in \Omega^{(2)}. \end{cases}$$

Численные эксперименты проводились на гексаэдральных, призматических и тетраэдральных сетках, при построении граница между подобластями, плоскость $x = 0.5$, разрешалась гранями сетки точно. Изолинии решения в плоскости xy показаны на Рис. 4.6. Результаты сходимости представлены в Таблице 4.6 (см. стр. 127), порядок сходимости оценивался по двум последним строчкам.

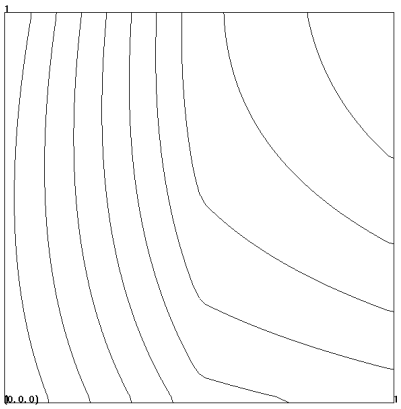


Рис. 4.6. Изолинии решения в плоскости xy для задачи с разрывным тензором диффузии.

Отметим, что наличие разрыва у диффузионного тензора не влияет на скорость сходимости, мы наблюдаем второй порядок сходимости по концентрации и первый по потокам.

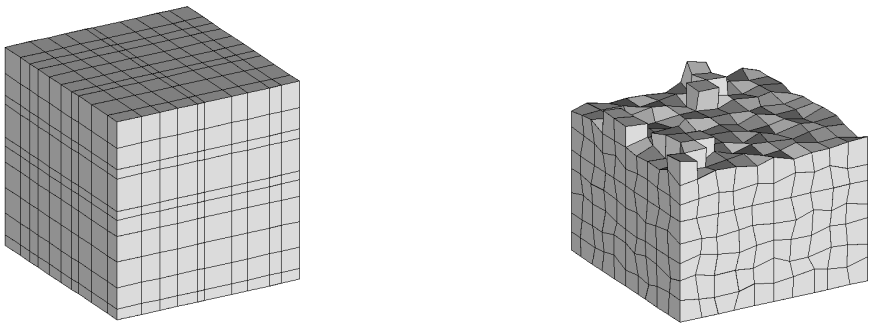
4.4.6. Сходимость на гексаэдральных сетках

Вернёмся к тесту, который мы проводили в разделе 4.4.3, и рассмотрим ту же задачу с постоянным тензором диффузии ещё на двух типах гекса-

Таблица 4.6. Результаты сходимости для задачи с разрывным тензором диффузии.

Тип сетки	гексаэдральная		призматическая		тетраэдральная	
h	ε_2^C	ε_2^q	ε_2^C	ε_2^q	ε_2^C	ε_2^q
1/10	3.38e-4	5.02e-3	2.99e-4	2.73e-3	4.68e-4	7.54e-3
1/20	8.64e-5	2.09e-3	1.37e-4	2.09e-3	1.67e-4	4.09e-3
1/40	2.18e-5	7.85e-4	3.45e-5	6.35e-4	4.62e-5	1.95e-3
порядок сходимости	1.99	1.41	1.99	1.71	1.85	1.07

эдральных сеток: $\mathbb{K}$-ортогональные сетки, и сетки с неплоскими гранями. Первый тип сеток получается из регулярной кубической сдвигом внутренних плоскостей на случайные расстояния, при этом все ячейки остаются прямоугольной формы, и их грани при этом ортогональны собственным векторам диффузионного тензора. Сетки второго типа получаются из регулярной кубической за счёт случайного сдвига вершин сетки, при этом грани ячеек получаются неплоскими. Изображения сеток приведены на Рис. 4.7.



($а$) ($б$)

Рис. 4.7. Гексаэдральные сетки с $h = 1/10$: ($а$) $\mathbb{K}$-ортогональная, ($б$) разрез сетки с неплоскими гранями.

Результаты сходимости представлены в Таблице 4.7. В таблицу включены результаты из Таблицы 4.3 раздела 4.4.3.

Таблица 4.7. Результаты сходимости для задачи с гладким решением и постоянным тензором диффузии на гексаэдральных сетках.

Тип граней	плоские		$\mathbb{K}$-ортогональные		неплоские	
h	ε_2^C	ε_2^q	ε_2^C	ε_2^q	ε_2^C	ε_2^q
1/10	6.79e-3	1.39e-2	1.74e-2	8.82e-3	7.88e-3	2.40e-2
1/20	1.69e-3	4.45e-3	4.01e-3	2.85e-3	2.03e-3	1.02e-2
1/40	3.77e-4	1.51e-3	1.17e-3	8.55e-4	5.35e-4	4.45e-3
1/80	9.74e-5	4.66e-4	2.85e-4	2.76e-4	1.31e-4	1.94e-3
порядок	2.04	1.63	1.98	1.67	1.97	1.21

Отметим, что второй порядок сходимости по концентрации наблюдается даже для сеток с неплоскими гранями. Порядок сходимости по потокам во всех случаях выше первого. Отдельно отметим, что для $\mathbb{K}$-ортогональных сеток достаточно сделать всего две итерации Пикара для получения решения.

4.5. Моделирование стационарного распределения интерферона в лимфатическом узле

Применим разработанные методы и алгоритмы для моделирования стационарного распределения интерферона в лимфатических узлах. Решение этой задачи является первым шагом в моделировании имунных процессов в лимфоузлах.

Развитие иммунных реакций происходит в системе лимфатических узлов и других вторичных лимфоидных органах. Лимфатические узлы представляют собой органы, пространственная организация и структура которых имеют своей целью поддержание оптимальных условий для развития иммунных реакций. В частности, в лимфоузлах происходит взаимодействие вирусов

с антиген-презентирующими клетками с последующей презентацией вирусных антигенов Т-лимфоцитам соответствующей специфичности. Эту функцию выполняют макрофаги и обычные дендритные клетки, а их защита от вирусов обеспечивается интерфероном , секретируемым плазмацитоидными дендритными клетками. Однако, пространственные закономерности распределения интерферона в лимфатическом узле в зависимости от числа плазмацитоидных дендритных клеток, скорости деградации интерферона и других параметров пока недоступны эмпирическому анализу. Единственным средством исследования данных аспектов в настоящий момент является математическое моделирование на основе уравнений конвекции-реакции-диффузии с учётом геометрии лимфоузла. Целью данного раздела является исследование зависимости стационарного распределения интерферона в лимфоузле при различных численностях секретирующих его дендритных клеток. Для этого будут использоваться разработанные методы и алгоритмы для построения сеточной модели лимфоузла и для решения стационарного уравнения диффузии.

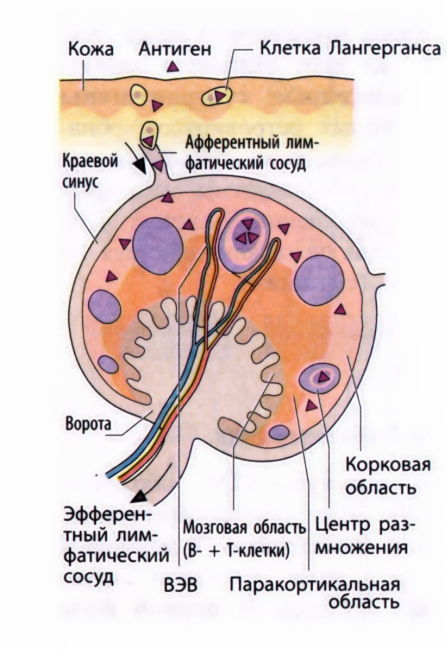


Рис. 4.8. Строение лимфатического узла (рисунок из [62]).

Хотя морфология лимфатического узла весьма сложна (см Рис. 1, [62]), с функциональной точки зрения его строение можно представить в виде обобщённой структуры, в основе которой лежит концепция базовых модулей или блоков, изложенной в [18]. Пространственно, макрофаги и дендритные клетки локализуются в областях краевого синуса под афферентными лимфатическими сосудами, по которым эти клетки и вирусы поступают в лимфоузел (см Рис. 4.8). Далее, клетки, презентирующие антиген, перемещаются в корковую область, где и выполняют свою функцию индукции антиген-специфических иммунных реакций. Для анализа распределения интерферона в лимфатическом узле воспользуемся результатами работы [3], в которой по экспериментальным данным были оценены важнейшие кинетические параметры реакции активации и синтеза интерферона-альфа в ответ на инфекцию коронавирусами у мышей и в частности скорость синтеза интерферона одной активированной дендритной клеткой ($4.4 \cdot 10^{-4}$ пг/час/клетку) и скорость распада интерферона (0.012 – 0.12 1/час). В качестве оценки коэффициента диффузии интерферона примем величину $1.6 \cdot 10^{-3}$ см^2/час, характеризующую диффузию молекул миоглобина, имеющих близкий к интерферону молекулярный вес [20]. Известно также, что области лимфоузла, в которых расположены Т-лимфоциты (корковая и паракортикальная области), обладают очень низкой гидродинамической проводимостью [22]. Для учёта этого факта мы полагали, что коэффициент диффузии в соответствующих областях меньше на 2 порядка, чем приведённая выше оценка. Промежуточными, по величине коэффициента диффузии, являются области (лимфоидные фолликулы), в которых расположены В-лимфоциты. В них коэффициент диффузии снижен в 10 раз по сравнению с базовым. Поскольку интерферон обладает низкой молекулярной массой, то он переносится также по системе кондуитов.

Запишем формальную постановку задачи стационарного распределения интерферона в лимфоузле. В качестве лимфоузла будем рассматривать сфе-

рическую область с несколькими внутренними подобластями. Пусть $c(\mathbf{x})$ – установившаяся концентрация интерферона в лимфоузле. Предположим, что единственным источником интерферона являются плазмацитоидные дендритные клетки, число которых может варьироваться в широких пределах: от 1 до 10^5 клеток. Мы будем рассматривать стационарную задачу, и будем предполагать, что конвективные силы влияют на распределение интерферона незначительно, а основными процессами, влияющими на распределение, являются диффузия и реакция.

Запишем уравнение для распределения концентрации интерферона:

$$-\operatorname{div}(\mathbb{K}\nabla c(\mathbf{x})) + \alpha c(\mathbf{x}) = \sum_{k=1}^{M} \rho\delta(\mathbf{x} - \mathbf{x}_k), \tag{4.31}$$

где $\mathbb{K}$ – тензорный коэффициент диффузии, $\mathbb{K} = \mathbb{K}^{(i)}$ в подобласти Ω_i, $i = 1, 2, 3$, определённой ниже, M – количество клеток-источников, $\mathbf{x}_k$ – положение клеток-источников, ρ – скорость синтеза интерферона одной клеткой, α – скорость поглощения интерферона.

Область Ω состоит из трёх подобластей: Ω_1, включающая краевой синус, трабекулярные синусы и кондуиты; Ω_2, состоящая из нескольких лимфоидных фолликулов; и Ω_3 – остальная часть, соответствующая корковой и паракортикальной областям. Диаметр лимфоузла – 2мм, толщина внешней оболочки – 0.1мм. Диаметр кондуитов порядка 0.0005мм, их форма имеет разветвлённую структуру, начинаются кондуиты во внешней оболочке, и заканчиваются в центральной зоне лимфоузла. Трабекулярные синусы имеют диаметр 0.05мм и длину 0.5мм, для простоты также будем считать, что они цилиндрической формы. Фолликулы будем рассматривать сферической формы с диаметром 0.2мм. Геометрическая модель лимфоузла вместе с расчётной сеткой представлена на Рис. 4.10.

Граничные условия – однородные условия Неймана (условие непротекания). На интерфейсе между Ω_1 и Ω_3, кроме поверхностей трабекулярных

синусов, также наложим условие непротекания. На оставшихся интерфейсах происходит скачок диффузионных коэффициентов, будем использовать условия сшивки, задающие непрерывность концентрации и непрерывность диффузионного потока [5, 52].

Положим $\mathbb{K}^{(1)} = \mathbb{K}^*$, $\mathbb{K}^{(2)} = 0.1\mathbb{K}^*$, $\mathbb{K}^{(3)} = 0.01\mathbb{K}^*$. Зафиксируем параметры задачи [3]: $\mathbb{K}^* = k\mathbb{I}, k = 0.16\text{мм}^2/\text{час}$, $\rho = 4.4 \cdot 10^{-4}\text{пг}/\text{час}$, $\alpha = 0.012/\text{час}$. Отметим, что выбранное значение α соответствует естественной деградации интерферона. Если учитывать ещё и поглощение интерферона клетками, то параметр α может быть увеличен до величины порядка $\alpha = 0.12$.

В процессе обсуждения постановки задачи с Г. А. Бочаровым также рассматривалось дополнительное неоднородное граничное условие в нижней части лимфоузла. Предполагалось, что в нижней части интерферон выносится вместе с лимфой через эфферентный лимфатический сосуд со скоростью потока жидкости $v \sim 0.1\text{мм}/\text{сек}$. С учётом этого поток интерферона через единицу площади составит $v \cdot c$. При численном моделировании было замечено, что в силу отсутствия конвективных потоков внутри области, при больших значениях v и при большой площади стока значения концентрации вблизи стока становятся отрицательными. Также было замечено, что накладывание такого условия на выток влияет на концентрацию только в маленькой локальной области вблизи стока. Поэтому было решено пренебречь этим условием при решении поставленной стационарной задачи реакции-диффузии.

4.5.1. Построение расчётных сеток

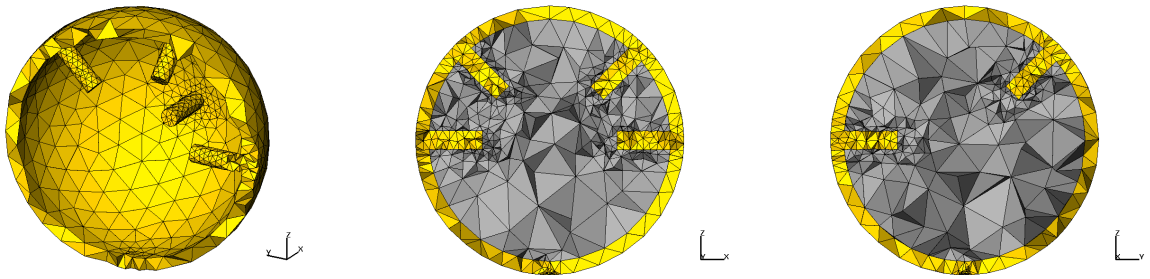
Для построения расчётных сеток для модели лимфоузла используются методы, разработанные и изложенные в Главах 2 и 3. Область Ω_1 задаётся в САПР OpenCASCADE с помощью объединения, пересечения и разности примитивных фигур: шаров и цилиндров. Пользователь может легко выби-

рать и менять размеры и положение фигур. Выбор системы OpenCASCADE обусловлен возможностью дальнейшего усложнения геометрической области и использования более сложных криволинейных фигур. Область Ω_2 задаётся как набор шаров заданного диаметра. Область Ω_3 является частью пространства, ограниченной внутренними границами Ω_1 и границами Ω_2.

Построение сетки происходит в три этапа. Сначала с помощью алгоритма продвигаемого фронта строится поверхностная треугольная сетка на границах областей $\{\Omega_i\}$. При этом шаг сетки на поверхности выбирается неравномерным, вблизи кондуитов и синусов шаг меньше, чем вдали от них на внешней поверхности лимфоузла. Такой подход обусловлен большим перепадом характерных геометрических размеров в области Ω, для сравнения, диаметр лимфоузла на 4 порядка больше диаметра кондуита.

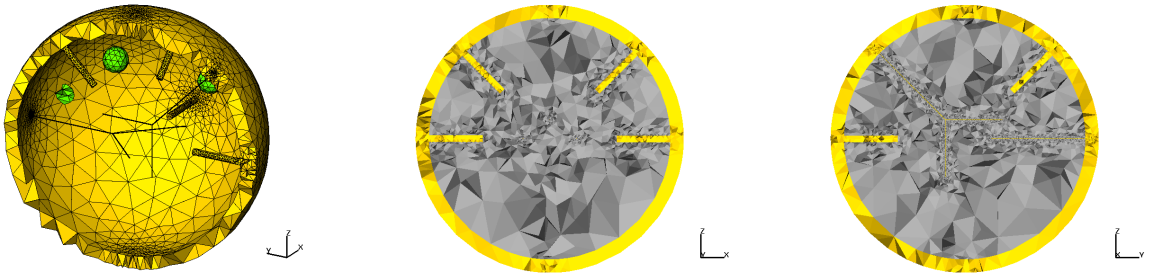
После построения поверхностной сетки строится тетраэдральная сетка с помощью алгоритмов, описанных в Главе 3. После построения тетраэдральной сетки её качество улучшается с помощью методов библиотеки aniMBA из пакета Ani3D.

Для решения задачи были построены две сетки: одна упрощённая – для тестирования метода, и одна полная – для проведения расчётов. В первой сетке были увеличены диаметры синусов, а кондуиты и фолликулы были исключены из рассмотрения для упрощения геометрической модели. Во второй модели присутствует четыре сферических фолликула, шесть синусов цилиндрической формы, и два “Y”-образных скрещивающихся кондуита с диаметром 0.005мм, который в 10 раз больше модельных значений и выбран таким для упрощения геометрической модели, а также для более наглядной визуализации. Изображения построенных сеток приведены на Рис. 4.9 для простой модели, и на Рис. 4.10 для полной модели.



(*а*) (*б*) (*в*)

Рис. 4.9. Сетка для упрощённой геометрической модели лимфоузла: (*а*) разрез области Ω_1, (*б*) сечение сетки плоскостью xz, (*в*) сечение сетки плоскостью yz. Жёлтым цветом показана область Ω_1, серым – область Ω_3.



(*а*) (*б*) (*в*)

Рис. 4.10. Сетка для полной геометрической модели лимфоузла: (*а*) разрез областей Ω_1 и Ω_2, (*б*) сечение сетки плоскостью xz, (*в*) сечение сетки плоскостью yz. Жёлтым цветом показана область Ω_1, зелёным – область Ω_2, серым – область Ω_3.

Для упрощённой модели геометрическая модель, созданная в САПР состояла из 22 узлов, 30 криволинейных рёбер, и 16 криволинейных граней. С помощью алгоритма продвигаемого фронта на поверхности была построена треугольная сетка, состоящая из 2138 вершин и 6824 треугольников. При построении с помощью алгоритма продвигаемого фронта тетраэдральной сетки было построено 18114 тетраэдров и 80 граней осталось в конечном фронте. Образованные лакуны были разбиты с помощью второго алгоритма, основанного на тетраэдризации Делоне, и общее количество тетраэдров составило 18206. Наихудшее качество тетраэдров на этом этапе составило

$q_{\min} = 3.26 \cdot 10^{-5}$. После улучшения сетки с помощью библиотеки aniMBA из пакета ani3D наихудшее качество увеличилось до $q_{\min} = 1.25 \cdot 10^{-2}$, а количество тетраэдров уменьшилось до 17448.

Полная геометрическая модель была задана в САПР с помощью 50 узлов, 62 криволинейных рёбер и 30 криволинейных граней. Построенная треугольная поверхностная сетка состояла из 24720 вершин и 95732 треугольников. С помощью комбинации двух методов была построена тетраэдральная сетка, состоящая из 103891 вершин и 619691 тетраэдров. Наихудщее качество тетраэдров составило $q_{\min} = 3.45 \cdot 10^{-12}$. После улучшения сетки с помощью библиотеки aniMBA из пакета ani3D наихудшее качество увеличилось до $q_{\min} = 7.78 \cdot 10^{-2}$, количество тетраэдров уменьшилось до 594898.

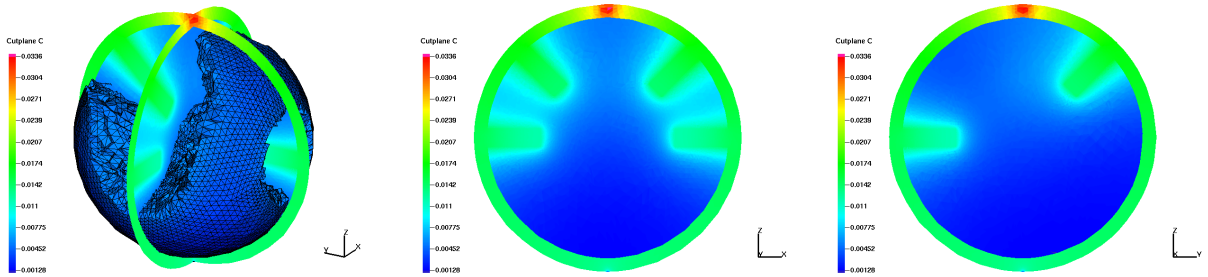
4.5.2. Проверка сходимости на упрощённой модели

Перед началом расчётов на полной модели лимфоузла проведём экспериментальную проверку сходимости решения на упрощённой модели с одним точечным источником, находящимся в верхней части лимфатического узла в центе краевого синуса. Рассмотрим набор иерархически измельчающихся сеток. Начальная сетка для упрощённой геометрической модели состоит из 17448 тетраэдров. В этой сетке каждый тетраэдр разбивается на 8 тетраэдров за счёт разбиения каждого ребра на два и каждой грани на четыре треугольника. Такой процесс мы будем называть равномерным иерархическим разбиением. Отметим, что получившаяся сетка также будет конформной. К полученной сетке можно опять применить равномерное иерархическое разбиение. Каждый раз характерный размер ячеек сетки уменьшается в два раза, а количество ячеек увеличивается в 8 раз.

При проверке сходимости решения будем предполагать, что решение, получившееся на самой мелкой сетке является точным. Вычислим ошибки в дис-

кретной L_2 норме для решений, полученных на грубых сетках, и по скорости падения ошибок при измельчении шага сетки оценим скорость сходимости решения. При проведении экспериментов на персональном компьютере ограниченный объём оперативной памяти не позволяет работать с очень большими сетками, поэтому в нашем эксперименте набор иерархических сеток состоял из трёх сеток, самая мелкая из которых содержала более миллиона ячеек.

На Рис. 4.11 приведены сечения решения задачи с одним источником, полученного на самой мелкой сетке, а также область лимфоузла, для которой значение концентрации оказалось меньше $5 \cdot 10^{-3}$. В Таблице 4.8 приведены результаты расчётов.



$(а)$ $(б)$ $(в)$

Рис. 4.11. Распределение интерферона в упрощённой модели лимфоузла (пг/л). $(а)$ сечение решения плоскостями xz и yz и область лимфоузла, в которой значение концентрации ниже $5 \cdot 10^{-3}$, $(б)$ сечение решения плоскостью xz, $(в)$ сечение решения плоскостью yz.

Таблица 4.8. Оценка скорости сходимости для упрощённой модели лимфоузла. N_T – число тетраэдров в сетке, ε_2^C – дискретная L_2 норма ошибки.

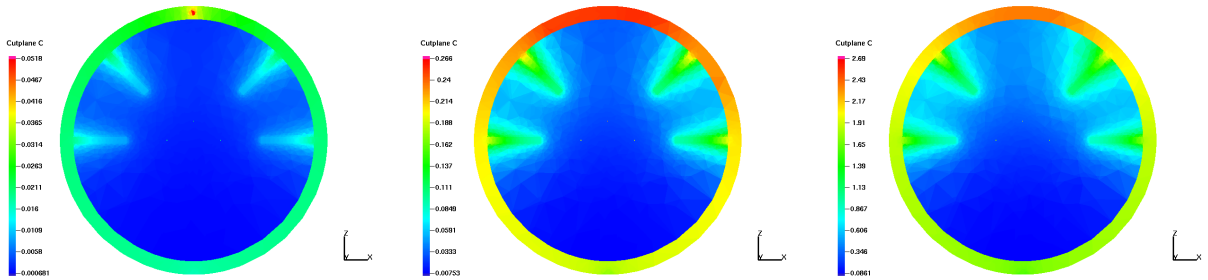
	N_T	ε_2^C
1	17448	2.44e-04
2	139584	6.15e-05
3	1116672	—
порядок		1.99

На упрощённой модели мы наблюдаем второй порядок сходимости. Аналогичные эксперименты проводились и при разном количестве источников интерферона, и при разном значении параметра α, отвечающего за скорость деградации интерферона. Во всех случаях наблюдался второй порядок сходимости решения. После экспериментальной проверки метода на упрощённой модели перейдём к полной модели.

4.5.3. Решение задачи для полной модели

При решении задачи для полной модели использовалась полная геометрическая модель с фолликулами и кондуитами. Число источников интерферона менялось, а сами источники распределялись случайно в верхней половине лимфоузла. Для этого использовалась неравномерная плотность вероятности, достигающая максимальных значений в верхней части краевого синуса.

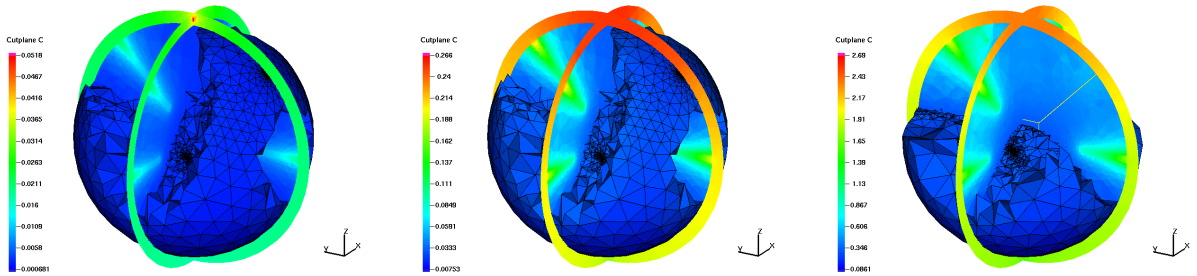
На Рис. 4.12 представлены результаты моделирования стационарного распределения интерферона в лимфатическом узле создаваемого 1, 10 и 100 активированными дендритными клетками для сечения плоскостью xz. Стационарное распределение оказывается существенно неоднородным. При этом разброс концентрации составляет 2 порядка.



($а$) ($б$) ($в$)

Рис. 4.12. Распределение интерферона в полной модели лимфоузла (пг/л) при разном количестве дендритных клеток. Сечение плоскостью xz. Количество плазмацитоидных дендритных клеток: ($а$) 1, ($б$) 10, ($в$) 100.

Как было показано в работе [3], пороговая концентрация интерферона, обеспечивающая защиту макрофагов и дендритных клеток составляет, соответственно, 0.1 пг/мл и 1 пг/мл. Результаты расчётов показывают, что для обеспечения защиты макрофагов во всём лимфатическом узле достаточно активировать порядка 100 дендритных клеток. На Рис. 4.13 показаны области лимфоузла, в которых концентрация интерферона ниже 0.003, 0.03 и 0.3 пг/мл для 1, 10 и 100 плазмацитоидных дендритных клеток соответственно.

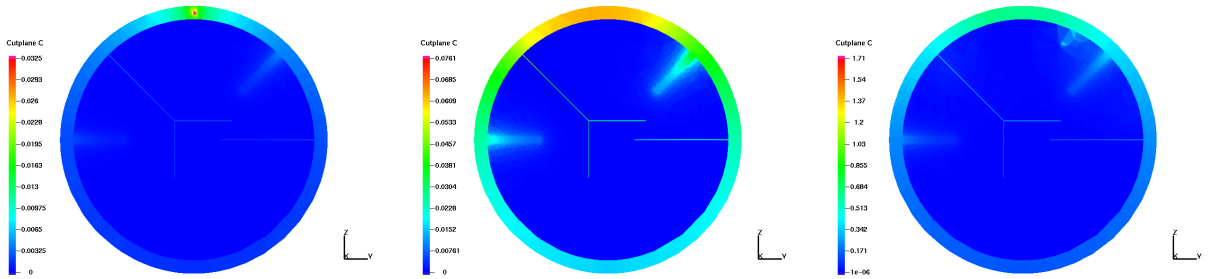


(*а*) (*б*) (*в*)

Рис. 4.13. Распределение интерферона в полной модели лимфоузла (пг/л) при разном количестве дендритных клеток. Области лимфоузла, в которых значение концентрации ниже некоторого порогового значения. Количество плазмацитоидных дендритных клеток и порог концентрации: (*а*) $M = 1$, $c < 0.003$, (*б*) $M = 10$, $c < 0.03$, (*в*) $M = 100$, $c < 0.3$.

Расчёты показывают, что система весьма чувствительна к скорости потери интерферона (вследствие деградации или поглощения клетками). Увеличение её в 10 раз ($\alpha = 0.12$) существенно уменьшает стационарный уровень интерферона в среде. На Рис. 4.14 показаны результаты моделирования распределения интерферона сечения плоскостью yz. Видно, что в системе кондуитов концентрация интерферона является относительно высокой. В нижних областях лимфатического узла значительно снижается стационарный уровень интерферона.

Таким образом, впервые описано стационарное распределение молекул интерферона в лимфатическом узле и исследована его зависимость от ба-



(*а*) (*б*) (*в*)

Рис. 4.14. Распределение интерферона в полной модели лимфоузла (пг/л) при высокой скорости его деградации. Сечение плоскостью yz. Количество плазмацитоидных дендритных клеток: (*а*) 1, (*б*) 10, (*в*) 100.

зовых параметров синтеза и деградации. Показано, что это распределение является существенно неоднородным и предсказаны области лимфоузла с относительно высоким уровнем защиты. Эти результаты, хотя и требуют дальнейшего уточнения на основе нестационарных моделей, вносят новое понимание в организацию иммунных процессов в лимфоузлах, в первую очередь, при хронических инфекциях, когда имеет место некоторая перманентная активация плазмацитоидных дендритных клеток, синтезирующих интерферон.

4.6. Выводы к четвёртой главе

Разработана и протестирована новая монотонная нелинейная схема конечных объёмов для решения стационарного уравнения диффузии на конформных сетках с многогранными ячейками. Проведён анализ монотонности предложенной схемы. Экспериментально подтверждена монотонность схемы, второй порядок сходимости по концентрациям и первый порядок сходимости по потокам для полных анизотропных тензоров диффузии на произвольных конформных сетках с многогранными ячейками. С помощью предложенного метода дискретизации уравнения диффузии решена практическая задача о стационарном распределении интерферона в лимфоузле.

Заключение

В диссертационной работе получены следующие результаты.

Разработана технология надёжного построения треугольных и тетраэдральных сеток. В работе проведён анализ влияния вычислительных погрешностей при реализации алгоритмов на ЭВМ, представлено теоретическое обоснование конечности числа операций предложенных алгоритмов. Программы для построения треугольных, поверхностных, и тетраэдральных сеток включены в пакет библиотек Ani3D и находятся в свободном доступе.

Разработанная технология является первым шагом на пути к созданию технологии автоматического построения конформных неструктурированных гибридных сеток с многогранными ячейками.

Для дискретизации уравнения диффузии на неструктурированных конформных сетках предложена новая монотонная схема на основе метода конечных объёмов. Проведено экспериментальное исследование скорости сходимости предложенной схемы на разных типах сеток. С использованием разработанных методов и алгоритмов решена практическая задача о стационарном распределении интерферона в лимфоузле.

Литература

1. Aavatsmark I., Eigestad G., Mallison B., Nordbotten J. A compact multipoint flux approximation method with improved robustness // Num. Meth. for Part. Diff. Eqs. 2008. Vol. 24, no. 5. Pp. 1329–1360.

2. Baker T. J. Shape Reconstruction and Volume Meshing for Complex Solids // Int. J. Numer. Meth. Engng. 1991. Vol. 32. Pp. 665–675.

3. Bocharov G., Züst R., Cervantes-Barragan L. et al. A Systems Immunology Approach to Plasmacytoid Dendritic Cell Function in Cytopathic Virus Infections // PLoS Pathogens. 2010.

4. Borouchaki H., Hecht F., Saltel E., George P.-L. Reasonably efficient Delaunay based mesh generator in 3 dimensions // Proc. of 4th Int. Meshing Roundtable. 1995. Pp. 3–14.

5. Brezzi F., Fortin M. Mixed and Hybrid Finite Element Methods. New York: Springer-Verlag, 1991.

6. Burman E., Ern A. Discrete maximum principle for galerkin approximations of the laplace operator on arbitrary meshes // Comptes Rendus Mathematique. 2004. Vol. 338, no. 8. Pp. 641–646.

7. Ciarlet P. G., Raviart P.-A. Maximum principle and uniform convergence for the finite element method // Comput. Methods Appl. Mech. Engrg. 1973. Vol. 2. Pp. 17–31.

8. Danilov A. Unstructured tetrahedral mesh generation technology // Ж. Выч. Мат. и Мат. Физ. 2010. Т. 50, № 1. С. 146–163.

9. 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.

10. Du Q., Wang D. Recent progress in robust and quality Delaunay mesh generation // J. of Comp. and App. Math. 2006. Vol. 195. Pp. 8–23.

11. Escobar J., Rodríguez E., Montenegro R., González-Yuste J. Simultaneous untangling and smoothing of tetrahedral meshes // Comp. Meth. in Appl. Mech. and Eng. 2003. Vol. 192. Pp. 2775–2787.

12. Gärtner K., Si H., Fuhrmann J. Boundary conforming delaunay mesh generation // Ж. Выч. Мат. и Мат. Физ. 2010. Т. 50, № 1. С. 44–59.

13. George P.-L., Borouchaki H. Delaunay Triangulation and Meshing. Application to Finite Elements. Paris: Hermes, 1998.

14. George P.-L., Borouchaki H. Maillage simplicial d'un polyèdre arbitraire // C. R. Acad. Sci. Paris, 2004. Vol. 338. Pp. 735–740.

15. George P.-L., Borouchaki H., Saltel E. 'Ultimate' robustness in meshing an arbitrary polyhedron // Int. J. Numer. Meth. Eng. 2003. Vol. 58. Pp. 1061–1089.

16. Ito Y., Shih A., Soni B. Reliable isotropic tetrahedral mesh generation based on an advancing front method // Proc. of 13th Int. Meshing Roundtable. 2004. Pp. 95–106.

17. Joe B. Three-dimensional boundary-constrained triangulations // Proc. of 13th IMACS World Congress. 1992. Pp. 215–222.

18. Junt T., Scandella E., Ludewig B. Form follows function: lymphoid tissue microarchitecture in antimicrobial immune defence // Nature Reviews Immunology. 2008. Vol. 8. Pp. 764–775.

19. Kaporin I. E. High quality preconditioning of a general symmetric positive definite matrix based on its $u^t u + u^t r + r^t u$-decomposition // Numer. Linear Algebra Appl. 1998. Vol. 5, no. 6. Pp. 483–509.

20. Keener J., Sneyd J. Mathematical Physiology. New York: Springer-Verlag. P. 766.

21. Korotov S., Křížek M., Neittaanmäki P. Weakened acute type condition for tetrahedral triangulations and the discrete maximum principle // Math. Comp. 2001. Vol. 70, no. 233. Pp. 107–119.

22. Lammermann T., Sixt M. The microanatomy of T-cell responses // Immunological Reviews. 2008. Vol. 221. Pp. 26–43.

23. 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.

24. LePotier C. Finite volume scheme satisfying maxcimum and minimum principles for anisotropic diffusion operators // Finite Volumes for Complex Applications / Ed. by R. Eymard, J.-M. Hérard. 2008. Pp. 103–118.

25. Lipnikov K., Svyatskiy D., Shashkov M., Vassilevski Y. Monotone finite volume schemes for diffusion equations on unstructured triangular and shape-regular polygonal meshes // J. Comp. Phys. 2007. Vol. 227. Pp. 492–512.

26. 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.

27. Löhner R., Parikh P. Generation of Three-Dimensional Unstructured Grids

by the Advancing Front Method // Int. J. Numer. Meth. Fluids. 1988. Vol. 8. Pp. 1135–1149.

28. López E., Nigro N., Storti M. Simultaneous untangling and smoothing of moving and fixed grids // Int. J. Numer. Meth. Engng. 2000. Vol. 10. Pp. 1–6.

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

30. Peraire J., Peiro J., Formaggia L. et al. Finite Element Euler Calculations in Three Dimensions // Int. J. Numer. Meth. Engng. 1988. Vol. 26. Pp. 2135–2159.

31. Santos V. R. On the strong maximum principle for some piecewise linear finite element approximate problems of nonpositive type // J. Fac. Sci. Univ. Tokyo Sect. IA Math. 1982. Vol. 29, no. 2. Pp. 473–491.

32. Schönhardt E. Über die Zerlegung von Dreieckspolyedern in Tetraeder // Mathematische Annalen. 1928. Vol. 98. Pp. 309–312.

33. Shewchuk J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator // Applied Computational Geometry: Towards Geometric Engineering / Ed. by M. C. Lin, D. Manocha. Berlin: Springer-Verlag, 1996. Pp. 203–222.

34. Shewchuk J. R. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates // Discrete & Comp. Geom. 1997. Vol. 18, no. 3. Pp. 305–363.

35. Shewchuk J. R. Constrained Delaunay tetrahedralizations and provably good boundary recovery // Proc. of 11th Int. Meshing Roundtable. 2002. Pp. 193–204.

36. Shewchuk J. R. Delaunay Refinement Algorithms for Triangular Mesh Generation // Comp. Geom.: Theory and Appl. 2002. Vol. 22, no. 1–3. Pp. 21–74.

37. Ushakova O. V. Advances in Grid Generation. New York: Nova Science Publishers, Inc, 2007.

38. van der Vorst H. A. BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems // SIAM J. Sci. Stat. Comput. 1992. Vol. 13, no. 2. Pp. 631–644.

39. Varga R. S. Matrix Iterative Analysis. Englewood Cliffs, NJ, USA: Prentice-Hall, 1962.

40. Yang Y., Yong J., Sun J. An algorithm for tetrahedral mesh generation based on conforming constrained Delaunay tetrahedralization // Computers & Graphics. 2005. Vol. 29. Pp. 606–615.

41. Yuan A., Sheng Z. Monotone finite volume schemes for diffusion equations on polygonal meshes // J. Comp. Phys. 2008. Vol. 227, no. 12. Pp. 6288–6312.

42. Боровиков С. Н., Иванов И. Э., Крюков И. А. Построение тетраэдризации Делоне с ограничениями для тел с криволинейными границами // Ж. Выч. Мат. и Мат. Физ. 2005. Т. 45, № 8. С. 1407–1423.

43. Василевский Ю. В., Вершинин А. В., Данилов А. А., Плёнкин А. В. Технология построения тетраэдральных сеток для областей, заданных в САПР // Матричные методы и технологии решения больших задач / Под ред. Е. Тыртышникова. М.: ИВМ РАН, 2005. С. 21–32.

44. Василевский Ю. В., Данилов А. А. Взаимодействие с САПР для построения расчётных сеток в сложных областях // Труды Математического центра им. Н.И. Лобачевского. 2009. Т. 39. С. 5–12.

45. Василевский Ю. В., Капырин И. В. Две схемы расщепления для нестационарной задачи конвекции-диффузии на тетраэдральных сетках // Ж. Выч. Мат. и Мат. Физ. 2008. Т. 48, № 8. С. 1429–1447.

46. Воеводин В. В., Кузнецов Ю. А. Матрицы и вычисления. М.: Наука, 1984.

47. Гаранжа В. А., Капорин И. Е. Регуляризация барьерного вариационного метода построения разностных сеток // Ж. Выч. Мат. и Мат. Физ. 1999. Т. 39, № 9. С. 1489–1503.

48. Данилов А. А. Построение тетраэдральных сеток для областей с заданными поверхностными триангуляциями // Численные методы, параллельные вычисления и информационные технологии. М.: МГУ, 2008. С. 119–130.

49. Данилов А. А. Способы построения трёхмерных поверхностных триангуляций и тетраэдральных сеток // Научно-технический вестник СПбГУ ИТМО. 2010. Т. 65, № 1. С. 87–92.

50. Капырин И. В. Семейство монотонных методов численного решения трёхмерных задач диффузии на неструктурированных тетраэдральных сетках // Доклады Академии Наук. 2007. Т. 614, № 5. С. 588–593.

51. Колдоба А. В., Повещенко Ю. А., Попов Ю. П. Об одном алгоритме решения уравнения теплопроводности на неортогональных сетках // Дифференциальные уравнения. 1985. Т. 21, № 7. С. 1273–1276.

52. Ладыженская О. А. Краевые задачи математической физики. М.: Наука, 1973.

53. Лебедев В. И., Агошков В. И. Операторы Пуанкаре-Стеклова и их приложения в анализе. М.: ОВМ АН СССР, 1983.

54. Марчук Г. И. Методы вычислительной математики. М.: Наука, 1989.

55. Марчук Г. И., Агошков В. И. Введение в проекционно-сеточные методы. М.: Наука, 1981.

56. Марчук Г. И., Кузнецов Ю. А. Итерационные методы и квадратичные функционалы // Методы вычислительной математики. Новосибирск: Наука, 1975.

57. Никитин К. Д. Технология построения поверхностных регулярных триангуляций для областей, составленных из примитивов. Дипломная работа, Мех.-мат. ф-т. МГУ им. М.В.Ломоносова, Москва, 2007.

58. Самарский А. А. Теория разностных схем. М.: Наука, 1982.

59. Скворцов А. В. Триангуляция Делоне и её применение. Томск: Изд-во Том. ун-та, 2002.

60. Тыртышников Е. Е. Методы численного анализа. М.: Издательский центр "Академия", 2007.

61. Харари Ф. Теория графов. М.: Мир, 1973.

62. Цинкернагель Р. Основы иммунологии. Пер. с нем. М.: Мир, 2008. С. 135.

63. Чугунов В. Н. Алгоритм построения конформной квази-иерархической треугольной сетки, слабо δ-аппроксимирующей заданные ломаные // Ж. Выч. Мат. и Мат. Физ. 2009. Т. 49, № 5. С. 874–878.

64. Электронный ресурс: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator. `http://www.cs.cmu.edu/~quake/triangle.html`.

65. Электронный ресурс: Advanced Numerical Instruments 2D. `http://sourceforge.net/projects/ani2d/`.

66. Электронный ресурс: Advanced Numerical Instruments 3D. http://sourceforge.net/projects/ani3d/.

67. Электронный ресурс: CUBIT. http://cubit.sandia.gov/.

68. Электронный ресурс: Netgen Mesh Generator. http://sourceforge.net/projects/netgen-mesher/.

69. Электронный ресурс: Open CASCADE Technology. http://www.opencascade.org/.

70. Электронный ресурс: TetGen: A Quality Tetrahedral Mesh Generator. http://tetgen.berlios.de/.

71. Электронный ресурс: TetMesh-GHS3D. http://www.distene.com/build/meshing.html.

72. Электронный ресурс: The Common Geometry Module. http://cubit.sandia.gov/cgm.html.

73. Электронный ресурс: The Common Geometry Module, Argonne (CGMA). http://trac.mcs.anl.gov/projects/ITAPS/wiki/CGM.