УДК 519.6

ПОСТРОЕНИЕ СЕТОК ТИПА ВОСЬМЕРИЧНОЕ ДЕРЕВО СО СКОЛОТЫМИ ЯЧЕЙКАМИ В НЕОДНОРОДНЫХ ОБЛАСТЯХ

А. Ю. Чернышенко[1]

Предложен алгоритм построения гексаэдральных сеток типа восьмеричное дерево в сложных областях, разбитых на непересекающиеся подобласти. В сетках допускаются сколы приграничных ячеек. Алгоритм скалывания основан на методе кубических марширующих квадратов (cubical marching squares) и методе марширующих кубов для неоднородных областей (multiple material marching cubes). Проводится анализ предложенного алгоритма, а также обсуждаются примеры полученных сеток. Работа частично поддержана проектами РФФИ (коды 11–01–00971, 12–01–33084, 12–01–31223) и ФЦП “Научные и научно-педагогические кадры инновационной России”, а также грантом компании ЭксонМобил и проектом “Прорыв” ГК “Росатом”.

Ключевые слова: сетки типа восьмеричное дерево, сколотые ячейки, гексаэдральные сетки, многогранные сетки.

Введение. Для решения прикладных трехмерных задач в сложных областях возникает необходимость построения качественных расчетных сеток. В трехмерном пространстве среди прочих выделяются три класса сеток — тетраэдральные, треугольные призматические и гексаэдральные. При фиксированном количестве вершин N_v сетки известны следующие оценки количества ячеек N_c и граней N_f в сетке: для неструктурированных тетраэдральных сеток $N_c \approx 5.5N_v$ и $N_f \approx 11N_v$, для треугольных призматических сеток $N_c \approx 2N_v$ и $N_f \approx 5N_v$, для гексаэдральных сеток $N_c \approx N_v$ и $N_f \approx 3N_v$. Получающиеся при дискретизации на гексаэдральных сетках шаблоны являются более компактными, а значит, и соответствующие матрицы имеют меньшее количество ненулевых элементов. Такие матрицы являются более экономичными с точки зрения требуемого объема памяти и вычислительной сложности решения системы уравнений. Использование сеток типа восьмеричное дерево позволяет строить гексаэдральные сетки, иерархически сгущающиеся к границам области или другим ее частям. Сгущение сетки к границе области за счет разбиения приграничных ячеек обеспечивает аппроксимацию криволинейной границы первого порядка. Для более точного приближения криволинейной границы области оказывается недостаточным использование только гексаэдральных сеток.

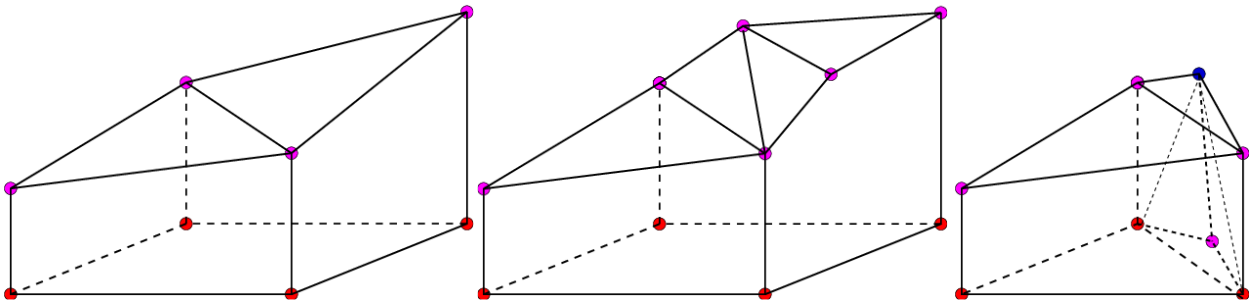


Рис. 1. Сколотые ячейки

В настоящей статье предлагается метод построения гексаэдральных сеток со сколотыми ячейками, которые позволяют приближать границу области со вторым порядком точности. Сколотая ячейка представляет собой часть кубической ячейки, полученную в результате ее среза поверхностной триангуляцией (рис. 1). Такая триангуляция порождается алгоритмом кубических марширующих квадратов (cubical marching squares, CMS [2]) в случае, если область является однородной. Этот алгоритм позволяет строить конформную триангуляцию для неравномерных сеток, в отличие от классического алгоритма марширующих кубов (marching cubes, MC [4]).

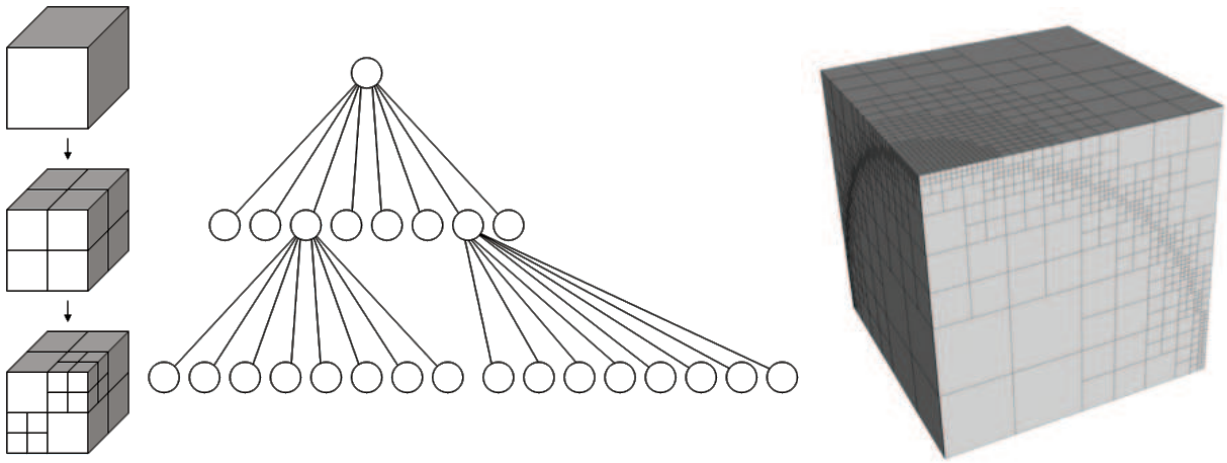
[1] Институт вычислительной математики РАН (ИВМ РАН), ул. Губкина, д. 8, 119333, Москва, аспирант; Институт проблем безопасного развития атомной энергетики РАН (ИБРАЭ РАН), Большая Тульская ул., д. 52, 115191, Москва, инженер; e-mail: chernyshenko.a@gmail.com

Во многих прикладных задачах, таких как построение сеток для численных моделей подземной гидродинамики или построение трехмерных сеток из двумерных медицинских изображений, область разбита на непересекающиеся подобласти с различными физическими свойствами, которые граничат между собой произвольным образом. Будем называть подобные области неоднородными. Построенная сетка должна правильно отражать наличие различных подобластей. Для построения подобных сеток в настоящей работе предложена модификация метода марширующих кубов для неоднородных областей (multiple material marching cubes, M^3C [3]) с более точным приближением границы. Эта модификация обладает преимуществами обоих методов, а значит, может быть применена для неравномерных сеток и неоднородных областей. В результате получается многогранная сетка типа восьмеричное дерево со сколотыми ячейками, которая состоит преимущественно из кубических ячеек. Многогранные ячейки порождены сколами кубических ячеек внешней и внутренних границ.

Сетка с многогранными ячейками является конформной, если любые два ее элемента либо не имеют общих точек, либо имеют ровно одну общую вершину, либо одно общее ребро, либо одну общую грань. Сетка типа восьмеричное дерево со сколотыми ячейками является конформной многогранной сеткой. Если область является неоднородной, то полученная сетка будет слабоконформной в том смысле, что некоторые ячейки могут иметь более одной общей грани.

Статья организована следующим образом. В разделе 1 описываются сетки типа восьмеричное дерево со сколотыми ячейками. В разделе 2 описывается алгоритм кубических марширующих квадратов для построения сколотых ячеек. В разделе 3 описаны алгоритмы для построения сколотых ячеек для неоднородных областей. В разделе 4 описывается общий алгоритм построения сеток типа восьмеричное дерево со сколотыми ячейками, а также проводится анализ на его конечность и вычислительную сложность. В разделе 5 приводятся примеры построенных сеток.

1. Сетки типа восьмеричное дерево со сколотыми ячейками. Опишем сетки типа восьмеричное дерево, а также методы скалывания кубических ячеек.



a) b)

Рис. 2. Восьмеричное дерево (a), пример сетки типа восьмеричное дерево (b)

1.1. Сетки типа восьмеричное дерево. Восьмеричное дерево (или восьмидерево) представляет собой древовидную структуру данных, элементы которой удовлетворяют следующим свойствам:

— каждый элемент имеет либо ровно восемь потомков, либо не имеет потомков; элемент, не имеющий потомков, называется *листом*, или листовым элементом;

— элемент не может быть потомком самого себя;

— каждый элемент, кроме исходного, называемого *корневым*, является потомком только одного элемента, называемого *родителем*;

— корневой элемент не имеет родителя, всегда существует и единственен.

Структура восьмидерева состоит из вершин, являющихся родителями и потомками, и ребер, обозначающих связи между родителями и их потомками. Сетка типа восьмеричного дерева представляет собой восьмидерево, вершинами которого являются кубики. Она может быть получена путем иерархического измельчения корневой кубической ячейки (рис. 2a). На каждом уровне измельчения родительская ячейка делится на восемь кубиков-потомков. Пример сетки типа восьмеричное дерево изображен на рис. 2b.

Введем дополнительное ограничение на такие сетки: размеры двух соседних ячеек могут отличаться не более чем в два раза. В таком случае измельчение сетки будет более плавным, что существенно при интерполяции сеточных данных и расчетах.

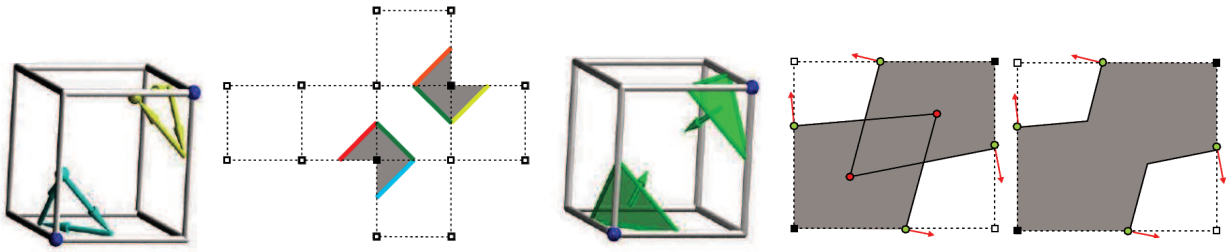
Грань сетки будем называть *листовой*, если две ячейки, разделяющие эту грань, являются листовыми и имеют одинаковый размер. Заметим, что листовая ячейка может иметь сложную нелистовую грань, если какая-либо соседняя ячейка находится на более глубоком уровне измельчения. Такую грань будем называть *переходной*.

Сетки типа восьмеричное дерево допускают быстрый доступ к любой ячейке, ее соседям и вершинам. Кроме того, они требуют небольших расходов памяти при хранении и динамическом перестроении.

Сетки типа восьмеричное дерево можно отнести к конформным сеткам, если рассматривать их как многогранные сетки, у которых некоторые грани (четыре грани, разделяющие одну переходную) могут лежать в одной плоскости.

1.2. Сетки со сколотыми ячейками. Для более точного приближения границы кубические ячейки скалываются. Скалывание кубической ячейки может осуществляться различными способами. В [7] рассмотрен алгоритм полигональных сечений (polygon clipping algorithm) [8], который используется в коммерческом генераторе сеток HEXPRESS [19]. Широко известен метод марширующих кубов (MC, marching cubes), который позволяет строить триангуляцию в кубической ячейке. При всей своей популярности, использование этого метода накладывает несколько существенных ограничений. Во-первых, в силу неоднозначности выбора триангуляции в кубической ячейке алгоритм может порождать топологически несвязные сетки. Во-вторых, классический метод MC не может воспроизводить резкие искривления границы. В [6] предлагается расширенный метод марширующих кубов (extended marching cubes), который сохраняет сложную геометрию границы на основе использования дополнительной информации в виде нормалей. В-третьих, метод MC может быть применен только к равномерным сеткам. Если размеры двух соседних ячеек отличаются, то получаемая триангуляция становится топологически несвязной и может содержать дырки и пустоты. В работах [9–12] предложены методы для восстановления топологической связности такой триангуляции на сетках типа восьмеричное дерево. Однако эти методы имеют существенные недостатки и сложности реализации, описанные в работе [2]. Там же предложен элегантный алгоритм кубических марширующих квадратов, который лишен всех описанных недостатков. Основная идея данного алгоритма изложена ниже.

2. Алгоритм cubical marching squares (CMS). Входными данными для рассматриваемого алгоритма является кубическая сетка, вершинам которой приписан знак характеристической функции области. На ребрах сетки отмечены точки пересечения с границей области и их нормали к границе, при этом на каждом ребре отмечено не более одной точки пересечения. Если на ребре имеется более одной точки пересечения, то все ячейки, разделяющие это ребро, следует измельчить. Как показано в [6], такой тип данных может быть получен из большинства используемых типов входных данных.



a) b)

Рис. 3. Развертка куба (a), разрешение неоднозначности (b). Рисунки из [2]

Алгоритм CMS основывается на следующих идеях:

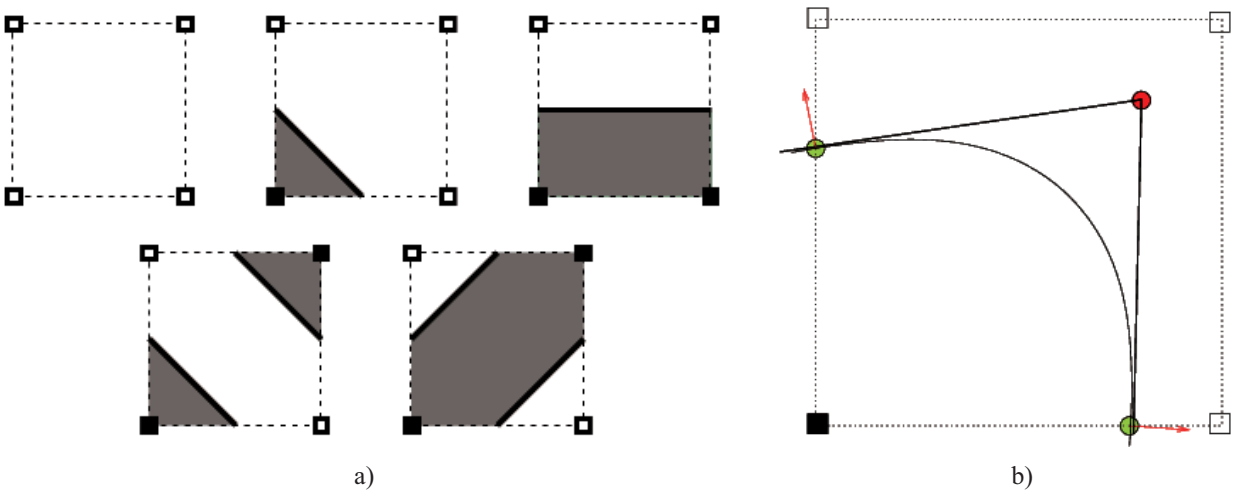
— марширующие кубы могут быть развернуты в марширующие квадраты;

— нормали могут быть использованы для разрешения неоднозначных ситуаций, а также для поиска особенностей криволинейной границы.

Как показано на рис. 3a, куб может быть развернут в шесть граней. Для каждой грани строится кусочно-линейный контур пересечения с поверхностью границы области с помощью алгоритма марширующих квадратов (MS, marching squares). Контуром будем называть набор ломаных. Если грани свернуть обратно в куб, то контуры будут образовывать замкнутые компоненты. Далее, эти компоненты триангулируются каким-либо способом. В случае равномерной кубической сетки полученная триангуляция будет

похожа на триангуляции метода марширующих кубов.

Рассмотрим процесс построения кусочно-линейного контура поверхности на листовой грани. Каждая грань имеет 4 вершины, каждой из которой приписан один из двух знаков. Таким образом, всего существует $2^4 = 16$ различных вариантов распределения знаков, которые с учетом симметрии и поворотов могут быть сведены к 5 вариантам, изображенным на рис. 4a. В случаях 4–5 возникает неоднозначность выбора пары точек, которые необходимо соединить отрезками. Эту неоднозначность можно решать одним из двух способов. В классическом подходе заранее выбирается приоритет связности одного из знаков характеристической функции. Однако такой подход не всегда отражает действительность. Более точным способом является использование нормалей к точкам. Так, для каждой точки на ребре можно провести соответствующую тангенциальную прямую в плоскости грани. В зависимости от пересечения таких прямых можно определить ту пару точек, которые нужно соединить (рис. 3b).



a) b)

Рис. 4. Марширующие квадраты (a) (рисунок из [2]), отслеживание искривлений границы с помощью нормалей (b)

Нормали также можно использовать для того, чтобы отслеживать искривления поверхности. Пусть имеется грань с двумя точками пересечения на ребрах. Тогда мы можем найти точку пересечения соответствующих тангенциальных прямых и использовать ее для построения более точного контура (рис. 4b).

Если грань переходная, то ее кусочно-линейным контуром является объединение контуров смежных с ней граней. Таким образом, любой элемент контура принадлежит ровно двум граням. Это автоматически означает отсутствие пустот и топологической несвязности триангуляции даже в случае, когда соседние грани разного размера.

После того как на всех гранях куба построены контуры, происходит объединение всех контуров. Затем из них выделяются компоненты, которые образуют циклы. Для этого достаточно сделать обход по контуру, начав с произвольной вершины и двигаясь к соседней. После этого на всех замкнутых контурах необходимо построить триангуляцию. Для этого можно использовать метод “разделяй и властвуй” (divide-and-conquer triangulation), описанный в [13]. Суть метода заключается в следующем. Рассмотрим замкнутую последовательность вершин $\mathcal{L}$, соединенных отрезками. Эта последовательность рекурсивно делится на две части вдоль *разделяющей прямой*, соединяющей две несоседние вершины из $\mathcal{L}$. Каждая новая часть делится вновь на две части, пока в каждой из частей не останется по три вершины. Так как вершины из $\mathcal{L}$ не обязательно принадлежат одной плоскости, то разделяющую прямую нужно выбрать так, чтобы полученные части не накладывались друг на друга. Кроме этого, разделяющая прямая не должна проходить по граням ячейки, так как на гранях не должны появляться новые отрезки. Для выбора разделяющей прямой используется *разделяющая плоскость* — плоскость, проходящая через разделяющую прямую и параллельная *средней нормали*. Если все вершины последовательности $\mathcal{L}$, находящиеся по разные стороны относительно той пары вершин, через которую проходит разделяющая прямая, находятся также по разные стороны относительно разделяющей плоскости, то полученные части не будут накладываться друг на друга и разделяющая плоскость является подходящей.

Для того чтобы найти среднюю нормаль, рассмотрим центр масс вершин последовательности. Соединив его с каждой из вершин, получим набор треугольников. Тогда средняя нормаль $\boldsymbol{n}$ вычисляется по

следующей формуле как средневзвешенная сумма единичных нормалей $\boldsymbol{n}_i$ к этим треугольникам:

$$\boldsymbol{n} = \frac{\sum \mathrm{mes}_i \cdot \boldsymbol{n}_i}{\sum \mathrm{mes}_i}. \tag{1}$$

Здесь mes_i — площадь i-го треугольника:

Триангуляция "разделяй и властвуй" может быть описана в виде следующего алгоритма.

Алгоритм 1. Триангуляция "разделяй и властвуй".

Вход: $\mathcal{L}$ (Последовательность вершин)
Присвоить N — количество вершин в $\mathcal{L}$
Если N = 3 **то**
    Создать треугольник из вершин $\mathcal{L}$
    вернуть
конец если
Вычислить среднюю точку AP
Вычислить среднюю нормаль $\boldsymbol{n}$
для $i = 1, 2, \ldots, N-2$ **начало цикла**
    для $j = i+2, \ldots, N$ **начало цикла**
        Положить D_{ij} — диагональ, соединяющая i и j вершины
        Построить разделяющую плоскость DP из D_{ij} и $\boldsymbol{n}$
        Разделить $\mathcal{L}$ на $\mathcal{L}_{\mathrm{left}}$ и $\mathcal{L}_{\mathrm{right}}$ с помощью D_{ij}
        если $\mathcal{L}_{\mathrm{left}}$ and $\mathcal{L}_{\mathrm{right}}$ лежат в разных полупространствах относительно DP **то**
            Триангуляция($\mathcal{L}_{\mathrm{left}}$)
            Триангуляция($\mathcal{L}_{\mathrm{right}}$)
        конец если
    конец цикла
конец цикла

Наличие нормалей в точках на ребрах позволяет также отслеживать особенности области внутри ячейки. Для поиска особенности области используется идея, предложенная в [6]. Обозначим точки на ребрах через $\boldsymbol{s}_i$, единичные нормали в этих точках обозначим через $\boldsymbol{n}_i$. Вычислим открытый угол конуса, натянутого на $\boldsymbol{n}_i$, по формуле $\theta := \min\limits_{i,j} \left(\boldsymbol{n}_i^{\mathrm{T}} \boldsymbol{n}_j\right)$.

Если угол θ меньше некоторой пороговой величины θ_s, то считается, что область имеет особенность. Пусть $\boldsymbol{n}_0$ и $\boldsymbol{n}_1$ — нормали, образующие наибольший угол, и пусть $\boldsymbol{n}^* = \boldsymbol{n}_0 \times \boldsymbol{n}_1$. Тогда оценим максимальное отклонение нормалей $\boldsymbol{n}_i$ от плоскости, натянутой на $\boldsymbol{n}_0$ и $\boldsymbol{n}_1$. Другими словами, если величина $\phi := \max_i \left|\boldsymbol{n}_i^{\mathrm{T}} \boldsymbol{n}^*\right|$ больше, чем некоторая пороговая величина ϕ_s, то считается, что область имеет особую точку в ячейке. На практике в [6] используются следующие значения: $\theta_s = 0.9$ и $\phi_s = 0.7$. Искомую точку $\boldsymbol{p}$ будем искать как решение системы

$$[\ldots, \boldsymbol{n}_i, \ldots]^{\mathrm{T}} \boldsymbol{p} = \left[\ldots, {\boldsymbol{n}_i}^{\mathrm{T}} \boldsymbol{s}_i, \ldots\right]. \tag{2}$$

В общем случае эта система может быть переопределенной или недоопределенной. Чтобы избежать рассмотрения различных случаев, система решается псевдообращением матрицы с использованием сингулярного разложения, а значение $\boldsymbol{p}$ определяется методом наименьших квадратов.

Если имеется особая точка $\boldsymbol{p}$ внутри ячейки, то триангуляция строится методом "веер треугольников" с центром в точке $\boldsymbol{p}$. Если точка $\boldsymbol{p}$ не попадает внутрь ячейки, то считаем, что ячейка не имеет особой точки.

Следует заметить, что теоретически возможна ситуация, когда контур достаточно искривленный, но особая точка не найдена, а метод "разделяй и властвуй" не определяет подходящую плоскость. В таком случае триангуляцию можно построить методом "веер треугольников" с использованием центра масс точек $\boldsymbol{s}_i$ в качестве центра триангуляции.

Кроме того, метод поиска особой точки позволяет также разрешать неоднозначность триангуляции кубической ячейки. Такие неоднозначности могут возникать, когда невозможно определить, нужно ли соединять две замкнутые компоненты контура или нет (рис. 5а). Разрешить указанные ситуации можно аналогичным способом, как и в случае неоднозначности на грани. Для этого в каждой из компонент ищется особая точка описанным выше способом. Затем строится конусообразная поверхность с центром в этой

точке. Если полученные поверхности пересекаются, то компоненты должны быть связными. Тогда триангуляция, соединяющая эти компоненты, топологически будет цилиндрической поверхностью (рис. 5b). В противном случае для каждой из компонент строятся независимые триангуляции (рис. 5c).

Рассмотрим кубическую ячейку C с гранями F. Тогда метод CMS может быть представлен в виде следующего алгоритма.

Алгоритм 2. Алгоритм кубических марширующих квадратов.

для $i = 1, 2, \ldots, 6$ **начало цикла**
    Получить контур S_i на грани F_i
конец цикла
Объединить все контуры S_i и выделить замкнутые компоненты $\mathcal{L}$
Присвоить NL — количество замкнутых компонент
для $k = 1, 2, \ldots, NL$ **начало цикла**
    Проверить наличие особой точки для $\mathcal{L}_k$
конец цикла
если имеется неоднозначность триангуляции **то**
    Разрешить неоднозначность
конец если
для $k = 1, 2, \ldots, NL$ **начало цикла**
    Триангуляция($\mathcal{L}_k$)
конец цикла

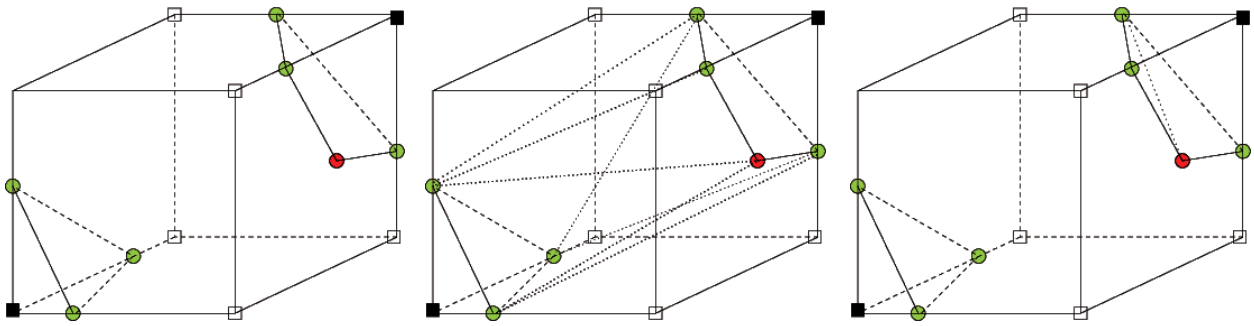


Рис. 5. Неоднозначность триангуляции

3. Случай неоднородной области. Традиционный метод марширующих кубов, а также упомянутые ранее модификации могут быть применены только в однородных областях. На практике существует множество задач, в которых область разбита на непересекающиеся подобласти с различными физическими свойствами. Эту область может представлять тело человека, состоящее из различных органов, либо различные геологические слои, которые допускают выклинивания и разломы. Подобласти могут располагаться произвольным образом; в таком случае одна кубическая ячейка может принадлежать более чем двум подобластям. Если применить алгоритм MC для каждой подобласти итерационно, то полученные треугольные интерфейсы будут образовывать пустоты и сетка получится топологически несвязной. В работе [18] предлагается модификация метода марширующих тетраэдров на случай неоднородной области. Так как кубическая ячейка может быть разделена на тетраэдры, то этот алгоритм может быть применен к гексаэдральной сетке, однако при этом порождается сравнительно большое количество треугольников для одной кубической ячейки. В работе [17] предложен алгоритм генерации тетраэдральных и гексаэдральных сеток для неоднородных областей с использованием метода дуальных контуров (dual contouring [16]). В работе [3] предложена модификация метода марширующих кубов на случай неоднородной области. Этот алгоритм позволяет создать топологически связную конформную триангуляцию на кубической сетке в случае неоднородной области, при этом используется двумерный метод марширующих квадратов для неоднородных областей (multiple material marching squares, M^3S). Далее опишем методы марширующих квадратов и кубов для неоднородных областей, а затем опишем модификацию этих алгоритмов.

3.1. Метод марширующих квадратов для неоднородных областей. Рассмотрим квадратную сетку в области, разделенной интерфейсами на непересекающиеся подобласти. Для каждой вершины сетки мы можем определить индекс подобласти, в которой она находится. Если вершина располагается на

интерфейсе двух или более подобластей, то ей присваивается индекс подобласти с наибольшим приоритетом, который определяется заранее. Каждое ребро имеет две вершины. Если индексы подобластей в этих вершинах различны, то на середине ребра создается точка, которая считается точкой пересечения ребра с соответствующим интерфейсом. Точку, которую мы считаем точкой пересечения интерфейса с ребром ячейки, будем называть *интерфейсной* точкой на ребре.

Будем считать, что интерфейсной точке приписываются оба индекса подобластей соответствующих концов отрезка. Квадрат имеет четыре вершины, каждая из которых имеет определенный индекс подобласти. Таким образом, всего имеется $4^4 = 256$ вариантов распределения индексов подобластей и построения кусочно-линейного контура на этой грани. Эти варианты могут быть сгруппированы по следующим правилам:

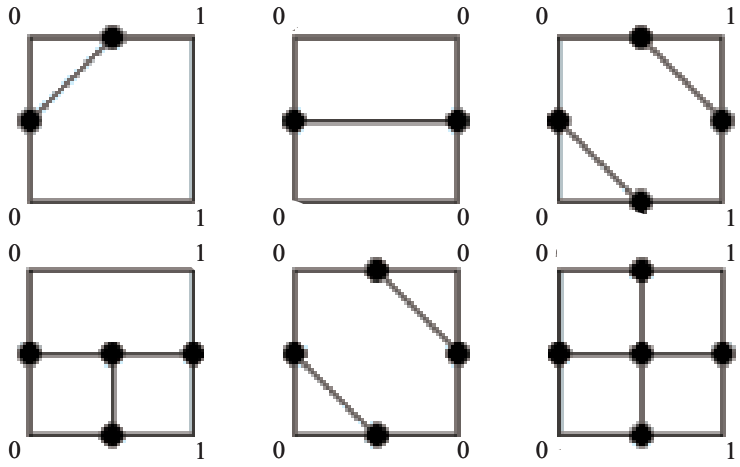


Рис. 6. Метод марширующих квадратов для неоднородных областей. Рисунок из [3]

— если количество различных индексов не более двух, то контур строится по обычному методу марширующих квадратов (рис. 6), верхние картинки;

— если количество различных индексов равно трем, причем две вершины с одинаковым индексом расположены по диагонали, то контур строится так, что эти вершины будут соединены (рис. 6), нижняя средняя картинка;

— в остальных случаях на грани создается центральная точка, которая соединяется с интерфейсными точками на ребрах (рис. 6), нижние левая и правая картинки; эту центральную точку будем называть интерфейсной точкой на грани.

Пусть F — квадратная грань, имеющая ребра E_i и вершины V_i. Метод M^{3}S может быть описан следующим алгоритмом.

Алгоритм 3. Алгоритм марширующих квадратов для неоднородных областей (M^{3}S).

Присвоить NP = 0 (количество интерфейсных точек на ребрах)
для $i = 1, 2, \ldots, 4$ **начало цикла**
  если E_i имеет различные индексы на концах, **то**
    NP = NP + 1
    Построить точку P_{NP} на середине E_i
  конец если
конец цикла
если NP = 2 **то**
  Соединить точки P_1 и P_2
конец если
если NP = 3 **то**
  Построить интерфейсную точку I_F на грани F
  для $i = 1, 2, \ldots, NP$ **начало цикла**
    Соединить точки P_i и I_F
  конец цикла
конец если
если NP = 4 **то**
  Определить NM — количество различных индексов в вершинах V_i
  если NM = 2, **то**
    Разрешить неоднозначность и выбрать пары точек для соединения (см. рис. 4b)
    Соединить эти точки
  конец если
  если NM = 3 **то**
    Соединить 2 пары точек так, чтобы вершины с одинаковым индексом остались связанными

Построить интерфейсную точку I_F на грани F
для $i = 1, 2, \ldots, NP$ **начало цикла**
Соединить точки P_i и I_F
конец цикла
конец если
конец цикла

3.2. Метод марширующих кубов для неоднородных областей. Каждому элементу кусочно-линейного контура (т.е. отрезку) соответствует ровно два индекса подобластей, которые отрезок разделяет.

После того как на каждой грани куба построены кусочно-линейные контуры, куб рассматривается в свернутом виде. Следует отметить, что на гранях и ребрах куба больше не будут добавляться точки или отрезки. Таким образом, как и в случае обычного метода MC, общая грань любых двух соседних ячеек всегда будет иметь один и тот же контур. Это означает, что ячейки будут граничить конформно. Объединив все контуры, будем рассматривать его компоненты, которые разделяют одни и те же пары подобластей. Некоторые из таких компонент могут оказаться незамкнутыми. Этот случай возможен, когда компонента содержит интерфейсную точку на грани. Такие компоненты необходимо замкнуть и после этого построить соответствующую триангуляцию во всех замкнутых компонентах. Каждая связная компонента имеет ровно два индекса подобластей, которые имеются у всех ее элементов, и отделяет соответствующие подобласти друг от друга.

В зависимости от количества интерфейсных точек на гранях выделяются следующие случаи:

— если куб не имеет интерфейсных точек на гранях, то все контуры образуют независимые замкнутые компоненты; тогда строится триангуляция методом “разделяй и властвуй”;

— если куб имеет ровно две интерфейсных точки на гранях (ровно одну куб иметь не может [5]), то некоторые компоненты формируют незамкнутые полилинии с концами в этих интерфейсных точках; соединив их, сделаем все незамкнутые компоненты замкнутыми; для триангуляции таких компонент также используется метод “разделяй и властвуй”;

— если имеется более двух интерфейсных точек на гранях, то создается точка в центре ячейки (по аналогии будем называть ее интерфейсной точкой ячейки); эта точка соединяется отрезками со всеми интерфейсными точками на гранях, после чего все компоненты становятся связными; каждая компонента, содержащая интерфейсную точку ячейки, триангулируется с помощью метода “веер треугольников” (triangle fan) с центром в этой точке; если компонента не содержит интерфейсную точку ячейки, то она триангулируется методом “разделяй и властвуй”.

Обозначим через C кубическую ячейку, имеющую грани F_i. Тогда метод M^3C может быть описан следующим алгоритмом.

Алгоритм 4. Алгоритм марширующих кубов для неоднородных областей.

Присвоить $NI_F = 0$ (количество интерфейсных точек на гранях)
для $i = 1, 2, \ldots, 6$ **начало цикла**
Применить алгоритм $\mathrm{M^3S}$ для F_i
если создана интерфейсная точка I_{F_i} на грани F_i **то**
$NI_F = NI_F + 1$
конец если
конец цикла
если $NI_F = 2$ **то**
Соединить две интерфейсные точки на гранях
конец если
если $NI_F > 2$ **то**
Построить интерфейсную точку O в центре C
для $i = 1, 2, \ldots, 6$ **начало цикла**
если Существует I_{F_i}; **то**
Соединить I_{F_i} и O
конец если
конец цикла
конец если
для $i = 1, 2, \ldots$ **начало цикла**
Найти $\mathcal{L}$ — очередную замкнутую компоненту контура
если компонента $\mathcal{L}$ пуста, **то**

перейти на Выход
конец если
если $\mathcal{L}$ содержит O **то**
Построить триангуляцию “веер треугольников” для $\mathcal{L}$
иначе
Построить триангуляцию “разделяй и властвуй” для $\mathcal{L}$
конец если
конец цикла
Выход

Таким образом, можно получить триангуляцию контуров, разделяющую различные подобласти, без добавления новых точек и отрезков на гранях и ребрах, а только внутри ячейки.

Авторы метода M^3C используют только середины ребер в качестве интерфейсных точек на ребрах, а также центр грани и ячейки в качестве интерфейсной точки на грани и в ячейке соответственно. Они объясняют это тем, что в случае использования точных значений точек пересечения могут возникнуть потенциально неправильные ситуации и нереалистичные поверхности. Такой выбор, действительно, упрощает алгоритм, но в то же время использование лишь середин ребер вносит ошибку аппроксимации границы порядка $h/2$, где h — размер ребра ячейки, а значит, сетка приближает криволинейную границу с первым порядком точности. В настоящей статье предложена модификация этого алгоритма, в которой используются более точные значения точек пересечения с ребрами и гранями, при этом учтены возникающие вследствие этого сложности.

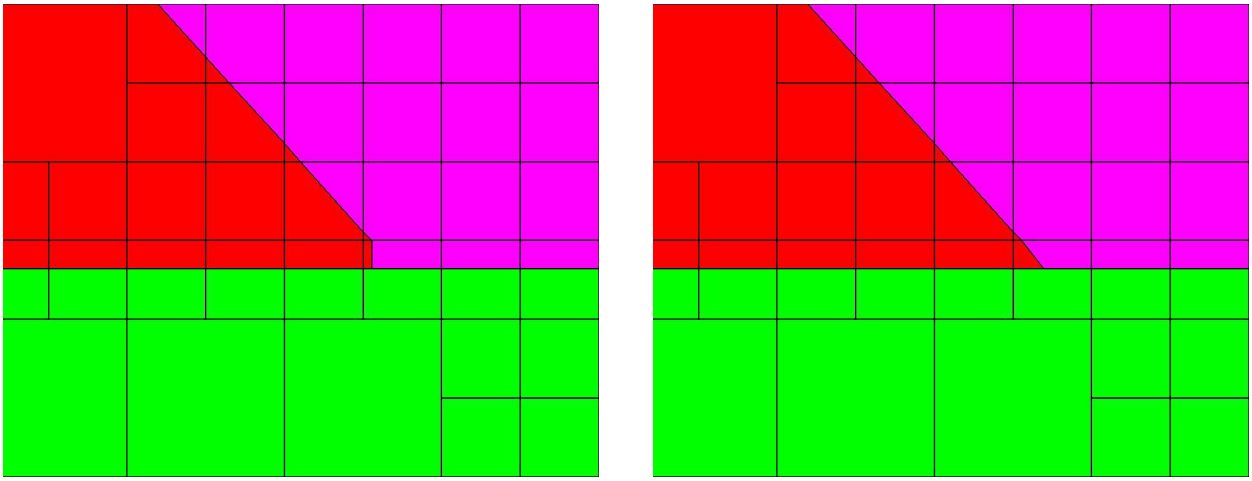


Рис. 7. Появление “зубчика” (слева) при неточном поиске интерфейсной точки на грани

3.3. Модификация метода марширующих кубов для неоднородных областей. Интерфейсная точка на ребре вычисляется с помощью метода бисекции по характеристической функции либо линейной интерполяцией функции расстояния со знаком. Как и в методе CMS, на каждом ребре имеется не более одной интерфейсной точки. Необходимо измельчить все ячейки, содержащие ребра с более чем одной интерфейсной точкой. Отметим, что возможность применения такого подхода на сетках типа восьмеричное дерево позволяет избегать возникновения неправильных и неоднозначных ситуаций. В качестве интерфейсной точки на грани вместо центральной точки используется более точное значение. Для ее вычисления применяется идея, аналогичная поиску особой точки области внутри ячейки. Обозначим интерфейсные точки на ребрах через $\boldsymbol{s}_i$, а проекции нормалей в этих точках на плоскость грани через $\boldsymbol{n}_i$. Искомую точку $\boldsymbol{p}$ будем искать как точку пересечения тангенциальных элементов точек $\boldsymbol{s}_i$. Другими словами, точка $\boldsymbol{p}$ удовлетворяет системе (2).

На практике полученная точка может не принадлежать соответствующей грани. Это может быть вызвано неточностью входных данных, а именно нормалей к интерфейсам, а также наличием особенностей на поверхностях интерфейсов. Кроме того, это происходит в случае, когда одно ребро ячейки пересекает более чем один интерфейс. Как было отмечено ранее, в таких случаях можно иерархически разбить эту ячейку и, соответственно, грань. Потенциальная интерфейсная точка на грани будет принадлежать одной из дочерних граней, а значит, можно попробовать там ее найти, при этом можно сделать несколько уровней измельчения. Следует заметить, что можно процесс измельчения применить для поиска интер-

фейсной точки на грани, но при этом итоговую сетку оставить без изменений, если измельчение сетки нежелательно. Можно воспользоваться альтернативным способом выбора интерфейсной точки на грани с использованием, например, центра масс точек $\boldsymbol{s}_i$. При неточном поиске интерфейсной точки на грани на сетке визуально может присутствовать "зубчик" в окрестности пересечения интерфейсов (рис. 7).

Определенную сложность представляет то, что некоторые интерфейсные точки на ребрах могут совпадать с вершинами граней. В некоторых случаях это может привести к тому, что на гранях появляются дополнительные точки или отрезки на этапе замыкания контуров или построения триангуляции контуров. Следует отметить, что для каждой пары соседних ячеек алгоритм должен создавать одинаковые контуры на их общей грани, иначе получаемая триангуляция будет не конформной. Поэтому если на грани появляется новая точка или отрезок, то их следует учитывать одновременно со стороны обеих ячеек, разделяющих эту грань.

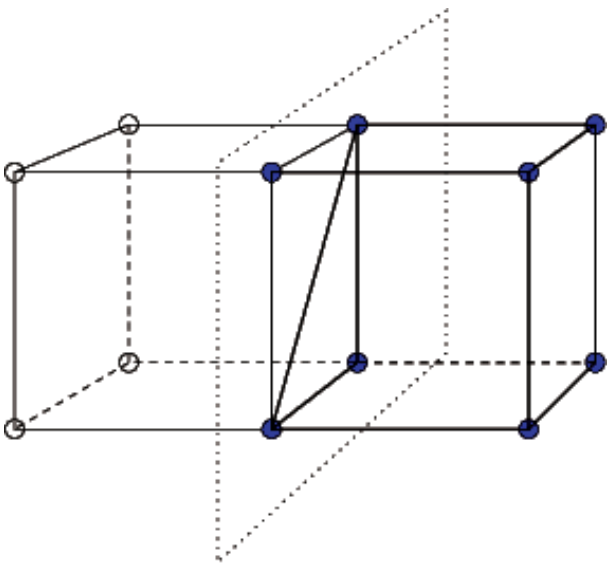


Рис. 8. Пример триангуляции на граничной грани

Рассмотрим простой пример, в котором одна из граней приграничной ячейки располагается на плоской границе (рис. 8). Все вершины данной ячейки имеют одинаковый индекс подобласти; следовательно, на ней не будет строиться триангуляция. Однако если рассмотреть ячейку, соседнюю с ней по грани на границе, то в ней строится триангуляция, которая полностью попадает на общую грань этих ячеек. Объем части этой ячейки, попавшей в область, равен нулю, поэтому мы ее отбрасываем, но при этом оставляем триангуляцию. Таким образом, мы добавили триангуляцию в рассматриваемую приграничную ячейку. Это приводит к тому, что след сетки на поверхности границы области, а также на интерфейсах представляет собой триангуляцию, поэтому внешне сетка выглядит как поверхностная триангуляция, хотя и состоит из кубических и сколотых ячеек.

Рассмотрим две соседние ячейки и их общую грань после применения к ней алгоритма $\mathrm{M}^3\mathrm{S}$. Заметим, что если со стороны одной из ячеек на грани нужно добавить отрезок, вершины которого уже имеются на грани, то это можно сделать без каких-либо дополнительных изменений со стороны второй ячейки. Действительно, добавление нового отрезка лишь разделяет на части грань, соответствующую одной подобласти.

Когда некоторые интерфейсные точки на ребрах попадают в вершины грани, интерфейсные точки на грани также могут совпасть с вершиной грани либо попасть на ее ребро. Действительно, для примера рассмотрим грань, в которой две соседние вершины имеют индекс подобласти 1, две другие — индексы 2 и 3 соответственно (рис. 9). Если интерфейсная точка на ребре BC совпадет с C, а интерфейсная точка на ребре AD совпадет с D, то, очевидно, интерфейсная точка на грани $ABCD$ будет лежать на ребре CD. Если интерфейсная точка на ребре CD совпадет, скажем, с вершиной C, то тогда интерфейсная точка на грани также совпадет с вершиной C (рис. 9b).

В большинстве случаев попадание интерфейсной точки грани на ребро или в вершину грани не приводит к изменениям алгоритма, однако в некоторых ситуациях это может привести к появлению новых точек или отрезков на гранях. Такая ситуация может иметь место после того, как мы проведем дополнительные отрезки, чтобы контуры стали замкнутыми. Изучим такие ситуации подробнее.

Рассмотрим две соседние грани F и G одной кубической ячейки и их общее ребро R. Обозначим за I_F интерфейсную точку на грани F. Пусть оказалось, что $I_F \in R$. Если концы ребра R имеют одинаковый индекс, то на нем нет интерфейсной точки, а значит, I_F будет новой точкой на этом ребре. Об этой точке должны "узнать" остальные ячейки, разделяющие ребро R. Для этого проводится постпроцессинг, который проходит по всем треугольникам, содержащим ребро R, и разделяет их на два треугольника отрезком, соединяющим точку I_F и третью вершину треугольника, не лежащую на ребре R. Для остальных граней, содержащих R, точка I_F просто увеличивает число вершин грани на единицу. Далее, в зависимости от расположения интерфейсной точки на грани G рассмотрим несколько вариантов.

1. Если у G нет интерфейсной точки, то на грани G не будут построены дополнительные отрезки.

2. Пусть I_G — интерфейсная точка на грани G и $I_G \in R$. Так как на одном ребре может быть не более

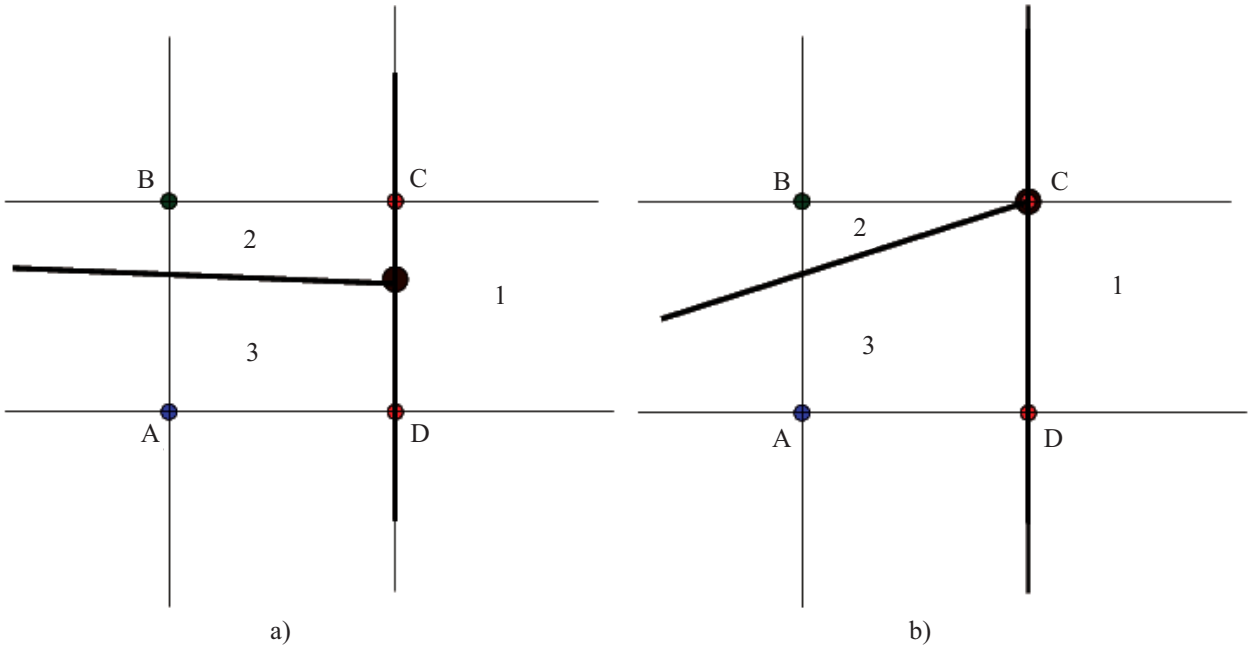


Рис. 9. Попадание интерфейсной точки на грани на ребро и в вершину

одной интерфейсной точки, то $I_F = I_G$, а значит, общее количество интерфейсных точек на гранях ячейки уменьшится на единицу. Если новое количество центров граней стало равно одному, то это эквивалентно ситуации, когда их число равно нулю, т.е. все компоненты контуров являются замкнутыми (рис. 10a). Остальные ситуации соответствуют описанным выше случаям.

3. Пусть $I_G \notin R$. Если концы отрезка R имеют разные индексы, то на нем есть интерфейсная точка, в которую и попала точка I_F. Если общее количество интерфейсных точек на гранях в ячейке равно двум, то, согласно алгоритму, мы должны соединить отрезком интерфейсные точки I_G и I_F. Однако точка I_F уже была соединена с I_G (как интерфейсная точка на ребре с интерфейсной точкой на грани), а значит, отрезок $I_G I_F$ не является новым для грани G (рис. 10b). Если концы ребра R имеют одинаковый индекс, то на нем нет интерфейсной точки. Если нужно соединить точки I_G и I_F, то отрезок $I_G I_F$ будет новым для грани G (рис. 10c). Этот отрезок нужно учитывать со стороны соседней по грани G ячейки. Как было сказано ранее, это не приводит к изменениям алгоритма для этой ячейки, однако нужно учесть наличие новой точки на ребре с помощью постпроцессинга, описанного выше.

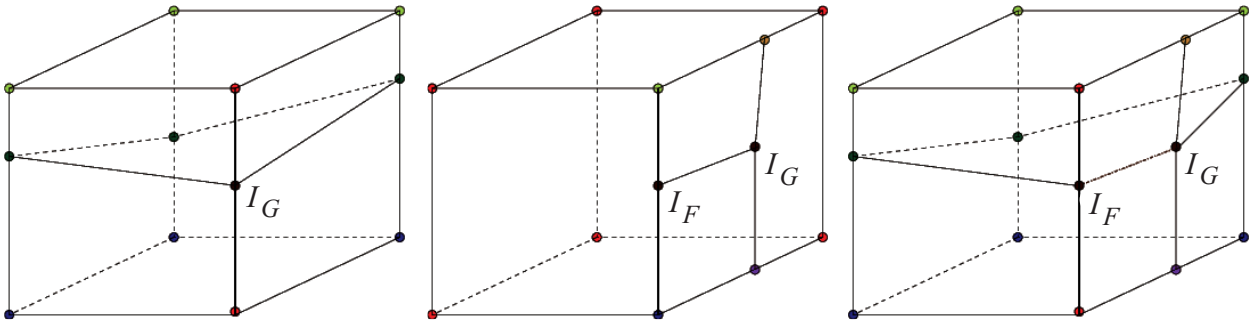


Рис. 10. Попадание на ребро интерфейсной точки на грани

Случай, когда интерфейсная точка на грани попадает в вершину, аналогичен случаю попадания на ребро, при этом новые точки на ребрах не появляются.

Интерфейсная точка ячейки также выбирается более точно. Она считается аналогично интерфейсной точке на грани по формуле (2), при этом рассматриваются все интерфейсные точки на ребрах. На практике возникают случаи, когда полученная точка не принадлежит рассматриваемой ячейке. Тогда, как и в аналогичном случае для интерфейсной точки на грани, можно воспользоваться иерархическим измельчением ячейки либо, для простоты, можно выбрать альтернативный вариант, например, принять за интерфейсную точку ячейки центр масс интерфейсных точек на гранях.

На положение интерфейсной точки ячейки также влияет попадание на ребра и в вершины ячейки интерфейсных точек на гранях. Когда на одну грань попадают три или более интерфейсных точек на грани, интерфейсной точкой ячейки автоматически становится интерфейсная точка этой грани. Случаи появления новых отрезков на грани аналогичны рассмотренным ранее.

4. Общий алгоритм. Заметим, что описанная выше модификация метода M^{3}C может быть использована на неравномерных сетках типа восьмеричное дерево. Отличия в алгоритме могут возникнуть лишь на переходных гранях. Возможна ситуация, когда переходная грань ячейки имеет более одной интерфейсной точки на грани. В таком случае ячейку следует разбить. Отметим, что при этом минимальный размер ячеек сетки не изменяется, так как соседние по переходной грани ячейки имеют меньший размер.

Таким образом, метод CMS является частным случаем этой модификации метода M^{3}C. Соответственно, полученная модификация имеет преимущества обоих методов, а именно возможность работы на неравномерных сетках и возможность работы в неоднородных областях. Будем называть эту модификацию методом кубических марширующих квадратов для неоднородных областей (multiple material cubical marching squares, MMCMS).

Опишем общий алгоритм построения сетки типа восьмеричное дерево со сколотыми ячейками для неоднородной области Ω. Будем считать, что входными данными является функция, которая для любой точки возвращает индекс принадлежащей ей подобласти, а также нормаль к интерфейсу, если точка находится вблизи интерфейса. Такую функцию можно восстановить из различных типов входных данных. Так, например, если область задана поверхностной триангуляцией, то искомую функцию можно получить с помощью алгоритма отслеживания лучей (ray tracing). Следует отметить, что вершины такой триангуляции не переносятся на итоговую сетку. Таким образом, если необходимо перенести данные из них на сетку, то необходим процесс интерполяции данных. Если точка находится вне области Ω, то ей присваивается фиктивный индекс, обозначающий внешнюю подобласть.

Сначала построим кубическую сетку типа восьмеричное дерево. Сетка может сгущаться к границам области и интерфейсам, вблизи особенностей и резких изменений области, а также к другим участкам. При этом сгущение может быть неравномерным для разных участков границы и интерфейсов. Кроме этого, сетку необходимо сгустить в следующих случаях. Рассмотрим две соседние ячейки разного размера. Если существует ребро большей ячейки, середина которого имеет отличный от его концов индекс подобласти, то все ячейки, разделяющие это ребро, должны быть измельчены. Кроме того, следует измельчать ячейки, имеющие ребро, которое пересекается с интерфейсами более чем в одной точке. В противном случае будет учтена только одна из точек пересечения и часть области будет потеряна.

В каждой вершине сетки известен индекс подобласти, которой она принадлежит. Для всех ячеек, у которых число различных индексов в вершинах больше одного, применяется метод MMCMS, описанный ранее. В результате некоторые ячейки будут сколоты либо разбиты на части с соответствующими индексами подобластей. Ячейки, имеющие фиктивный индекс, удаляются из сетки.

Построение сетки типа восьмеричное дерево со сколотыми ячейками для области Ω может быть представлено в виде следующего алгоритма.

Алгоритм 5. Алгоритм построения сетки типа восьмеричное дерево со сколотыми ячейками.

Построить сетку типа восьмеричное дерево для Ω
Сделать дополнительное измельчение сетки
для $V \in \mathcal{V}$ **начало цикла** ($\mathcal{V}$ — вершины сетки)
    Присвоить вершине V индекс подобласти, в которой она находится
конец цикла
1: **для** $C \in \mathcal{C}$ **начало цикла** ($\mathcal{C}$ — ячейки сетки)
    если Число различных индексов в C больше 1 **то**
        $NI_F = 0$ (Число интерфейсных точек на гранях)
        для $i = 1, 2, \ldots, 6$ **то**
            Применить модифицированный алгоритм M^{3}S для грани F_i
            если созданы более одной интерфейсной точки на переходной грани **то**
                Разбить ячейку C
                перейти к метке 1
            конец если
            если создана интерфейсная точка I_{F_i} на грани F_i **то**
                если I_{F_i} не совпадает с другими интерфейсными точками **то**
                    $NI_F = NI_F + 1$

конец если
конец если
конец цикла
конец если
если $NI_F = 2$ **то**
Соединить две интерфейсные точки на гранях
конец если
если $NI_F > 2$ **то**
Построить интерфейсную точку O ячейки C
если точка O совпала с одной из интерфейсных точек на грани **то**
Удалить точку O
иначе
для $i = 1, 2, \ldots, 6$ **начало цикла**
если Существует I_{F_i} **то**
Соединить I_{F_i} и O
конец если
конец цикла
конец если
конец если
для $i = 1, 2, \ldots$ **начало цикла**
Найти $\mathcal{L}$ — очередную замкнутую компоненту контура
если $\mathcal{L}$ пуста **то**
перейти к метке 2
конец если
если $\mathcal{L}$ содержит O **то**
Построить триангуляцию “веер треугольников” для $\mathcal{L}$
иначе
Построить триангуляцию “разделяй и властвуй” для $\mathcal{L}$
конец если
конец цикла
2: Разделить ячейку C на части с помощью триангуляции
конец цикла
Сделать постпроцессинг (Описан в разделе 3.3)
Удалить ячейки с фиктивным индексом

Следует отметить, что после того как мы поделили кубическую ячейку на отдельные ячейки, сетка формально перестает быть восьмеричным деревом, так как некоторые ячейки имеют отличное от 8 число потомков. Кроме того, отметим, что ориентация получаемых треугольников, вообще говоря, не является одинаковой даже в случае одной подобласти. Поэтому, в случае необходимости согласованной ориентации, можно сделать соответствующее переупорядочивание вершин треугольников.

Вспомним, что сетку типа восьмеричное дерево можно рассматривать как конформную многогранную сетку. Сетка со сколотыми ячейками тоже принадлежит классу конформных многогранных сеток, если область не разделяется на подобласти. Если область разделена на подобласти, то некоторые соседние ячейки могут иметь более одной общей грани. Действительно, когда кубическая ячейка разбивается на части методом MMCMS, граница между этими частями представляет собой набор из нескольких треугольников (рис. 11). В этой связи сетку со сколотыми ячейками будем называть *слабо конформной* в описанном выше смысле.

4.1. Конечность алгоритма. Проведем анализ конечности описанного алгоритма.

Процесс построения сетки типа восьмеричное дерево описан в различной литературе [1]. Этот процесс конечен, если ограничена глубина измельчения дерева.

Далее, мы имеем цикл по вершинам сетки, который конечен в силу ограниченности числа вершин в сетке. Рассмотрим цикл по ячейкам. Для каждой ячейки в цикле по граням применяется модифицированный алгоритм M^{3}S. В нем необходимо найти интерфейсную точку на грани. Как было сказано ранее, она может быть найдена решением линейной системы либо, возможно, с помощью измельчения сетки или альтернативным выбором точки на грани. Так как можно ограничить число уровней измельчения сетки при этом поиске, то этот процесс конечен. В некоторых ячейках нужно найти интерфейсную точку ячейки. Аналогично случаю интерфейсной точки на грани, процесс ее поиска является конечным.

После этого находятся все замкнутые компоненты контура. Этот процесс конечен, так как каждая вершина контура имеет ограниченное число соседей. Докажем, что процесс построения триангуляции замкнутой последовательности вершин конечен. Очевидно, что метод "веер треугольников" конечен, так как требует лишь соединения центра триангуляции с остальными несоседними вершинами. Метод "разделяй и властвуй" требует поиска подходящей разделяющей плоскости. Как указано в [13], для произвольных последовательностей вершин существуют примеры, в которых разделяющая плоскость не будет найдена данным алгоритмом. Однако в [14] показано, что такие случаи могут возникать лишь в более сложных последовательностях, чем в методе M^3C. В любом случае докажем, что если не удалось построить триангуляцию методом "разделяй и властвуй", то ее можно построить альтернативным способом.

Рассмотрим случай, когда внутри ячейки не создается интерфейсная точка. Если не удалось построить триангуляцию методом "разделяй и властвуй", то она строится методом "веер треугольников", где в качестве центра берется особая точка ячейки либо центр масс интерфейсных точек. Если ячейка имеет внутри интерфейсную точку, то все контуры, проходящие через нее, триангулируются этим же способом. Остался случай, когда ячейка имеет интерфейсную точку, но контур не проходит через нее. Докажем, что если контур не проходит через интерфейсную точку ячейки, то он не проходит также через какую-либо интерфейсную точку на грани. Предположим, что интерфейсная точка грани принадлежит последовательности, и рассмотрим две ее соседние точки. Ни одна из них не является интерфейсной точкой ячейки, так как контур не проходит через нее. В то же время ни одна из них не является другой интерфейсной точкой грани, так как такие точки не могут быть соединены в данном случае. Следовательно, оба ее соседа — это интерфейсные точки на ребрах, причем на этой же грани. Этим точкам приписано ровно по два индекса подобластей, которые они разделяют, причем эти пары индексов не совпадают. Однако тогда эти две точки не могут принадлежать одному контуру, так как контуру соответствует только одна пара индексов. Таким образом, контур не проходит через интерфейсные точки на гранях и, следовательно, не проходит через грани, на которых есть такая интерфейсная точка. Так как имеется интерфейсная точка внутри ячейки, то число интерфейсных точек на гранях не меньше трех. Следовательно, число граней, на которых нет интерфейсных точек, не больше трех.

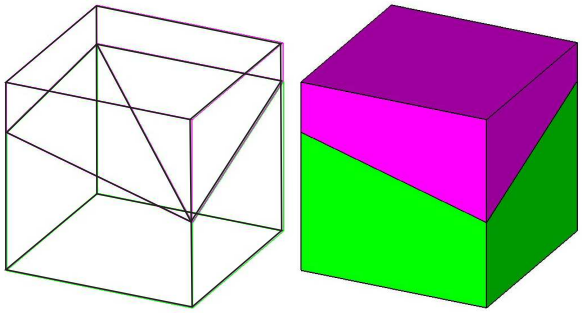


Рис. 11. Кубическая ячейка, поделенная на две части триангуляцией

Рассмотрим случай, когда таких граней ровно три и контур проходит через них. Очевидно, что они имеют одну общую вершину, иначе контур был бы незамкнутым. Для наглядности обозначим последовательно вершины через $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m, C_1, \ldots, C_k$, причем A_1, B_1, C_1 — интерфейсные точки на ребрах (рис. 12). Рассмотрим первую вершину, не совпадающую с A_1, B_1, C_1. Не умаляя общности можно считать, что это A_2. Соединим вершину A_2 со всеми остальными вершинами последовательности, не лежащими с ней на одной грани: $B_2, \ldots, B_m, C_1, \ldots, C_k$. Далее, соединив B_2 (может совпадать с C_1) с оставшимися вершинами $A_3, \ldots, A_n$, мы получим подходящую триангуляцию последовательности.

Если таких граней две, то рассмотрим их общее ребро. Так как контур замкнутый, то он пересекает это ребро в двух точках (как минимум одна из них совпадает с вершиной ребра), которые разделяют контур на две части. Соединив последовательно пары вершин из разных частей, получим искомую триангуляцию.

Таким образом, если алгоритм "разделяй и властвуй" не находит подходящую разделяющую плоскость и не может быть выполнен, то всегда можно построить триангуляцию альтернативным способом. Однако на практике оказывается достаточным использовать алгоритм "разделяй и властвуй".

4.2. Вычислительная сложность алгоритма. Будем считать, что область задается функцией $f : \mathbb{R}^3 \to \mathbb{Z}$, возвращающей индекс подобласти, которой принадлежит точка (x, y, z). В зависимости от типа входных данных, вызов этой функции может потребовать большие вычислительные затраты, поэтому будем учитывать ее сложность cf.

Максимальная сложность построения сетки типа восьмеричное дерево — порядка $O(n \ln n)$, где n — число вершин дерева.

Пусть имеется сетка типа восьмеричное дерево, N_v — число вершин и N_c — число ячеек. Для оценки вычислительной сложности алгоритма рассмотрим следующие основные операции при построении сетки: поиск интерфейсной точки на ребре и нормали к ней; поиск интерфейсной точки на грани и в ячейке; построение триангуляции замкнутого контура.

Рассмотрим ребро ячейки, концы которого имеют разные индексы. Для нахождения интерфейсной

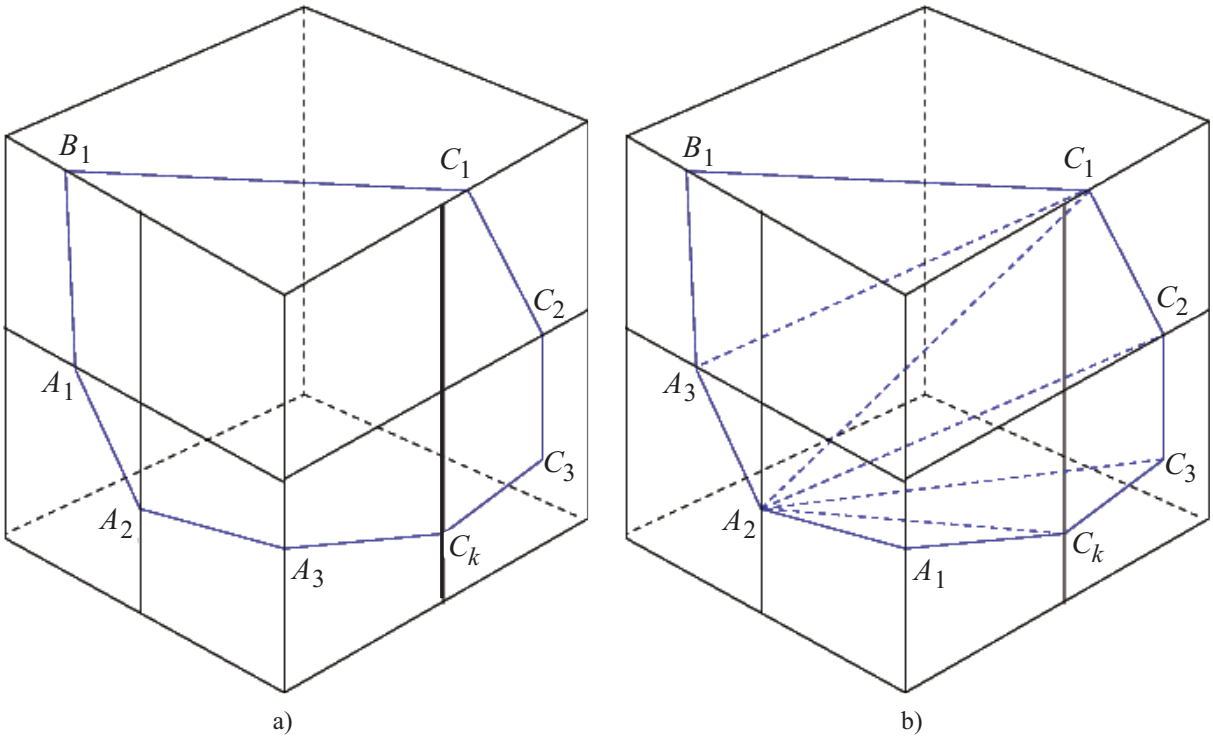


Рис. 12. Замкнутый контур (a), его триангуляция (b)

точки на ребре необходимо сделать nb шагов метода бисекции, где nb зависит от желаемой точности сетки. На каждом шаге один раз вызывается функция f; следовательно, общая сложность равна $nb \cdot cf$. В гексаэдральной сетке типа восьмеричное дерево имеется следующая оценка количества ребер: $N_r \approx 3N_v$. Предположим, что на каждом ребре может находиться интерфейсная точка, тогда сложность поиска всех интерфейсных точек пропорциональна $nb \cdot cf \cdot 3N_v = O(N_v \cdot cf)$. Для вычисления нормали, в зависимости от типа входных данных, используется один из способов, описанных в [6]. Так, например, если область задается поверхностной триангуляцией, то используется нормаль к ближайшему треугольнику. Для поиска интерфейсной точки на грани необходимо решить систему линейных уравнений порядка 3×3 или 4×4. Если требуется сделать дополнительно не более k измельчений сетки, то необходимо решить еще k подобных систем. Для поиска интерфейсной точки ячейки необходимо решить систему линейных уравнений порядка до 12×12 либо дополнительно k систем, если необходимо измельчать сетку.

При построении триангуляции контура методом “разделяй и властвуй” имеется три вложенных цикла, каждый из которых ограничен количеством N точек в контуре. На каждой из 6 граней кубической ячейки может находиться не более 8 отрезков одной компоненты контура; следовательно, число N не превосходит 48, хотя существует более точная оценка. Заметим, что применение алгоритма MMCMS является локальным и его сложность для одной ячейки не зависит от общего числа ячеек. Таким образом, общая сложность алгоритма пропорциональна $c_1 N_v + c_2 N_v \ln N_v$, причем значения c_1 и c_2 зависят от cf.

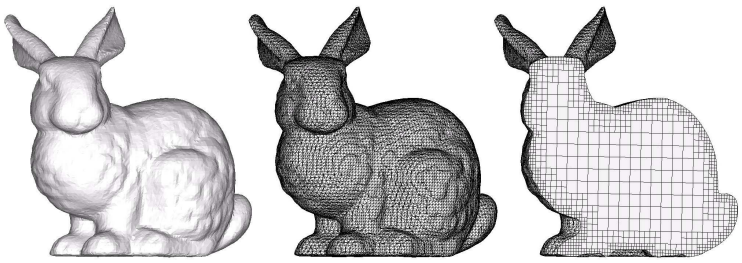


Рис. 13. Сетка для модели кролика, ее срез

5. Примеры сеток. В этом разделе представлены некоторые примеры сеток, построенных с помощью описанного алгоритма. Все примеры были построены с помощью собственного генератора, написанного на языке C++.

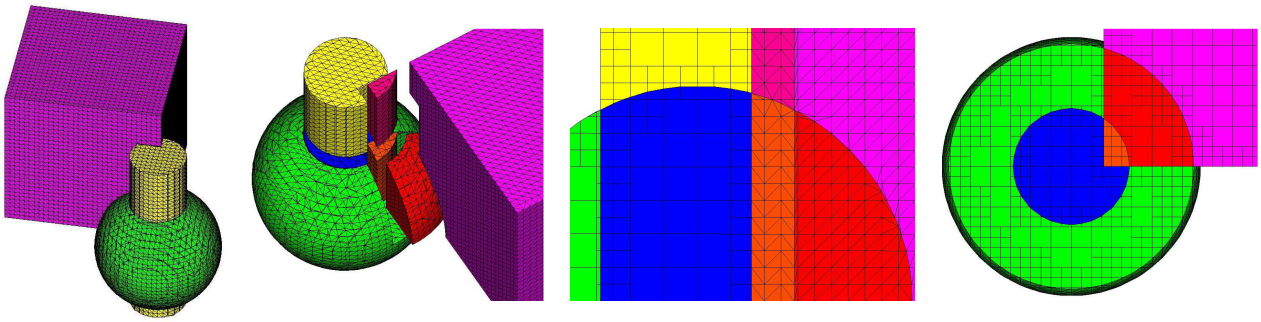
5.1. Однородная область. Рассмотрим сетку для модели кролика [20] (рис. 13). Входными данными является поверхностная триангуляция области, с помощью которой была построена характеристическая функция области. В полученной сетке 49 997 ячеек, 68 244 вершины и 191 059 граней. Сетка сгущается к границе области.

5.2. Неоднородная область. Будем рассматривать область, состоящую из пересечения примитивов: шара, параллелепипеда и цилиндра. Будем рассматривать области пересечения фигур как отдельные подобласти.

Таблица 1

Параметры сетки для области из пересечения примитивов

Уровень	Время, сек.	# ячеек	# сколотых ячеек	# граней	# вершин
1	0.75	22 073	7792	88 645	29 786
2	3.25	99 206	31 422	391 859	134 526
3	14.62	421 894	119 306	1 724 179	604 021



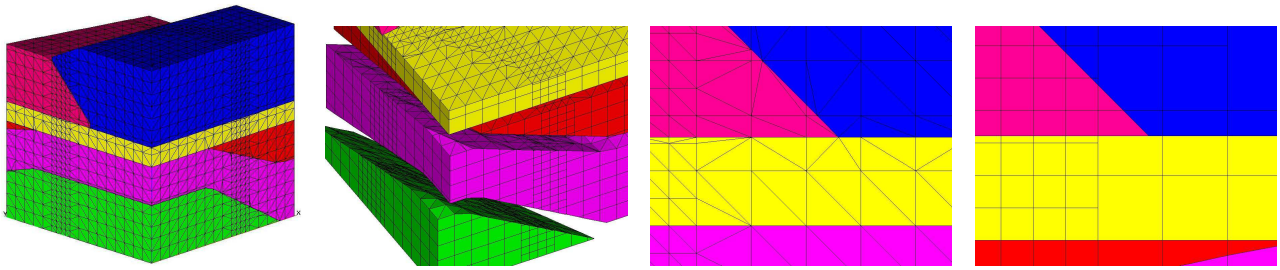
a) b) c) d)

Рис. 14. Сетка для пересечения примитивов (a), ее внутренняя структура (b), увеличения у вертикального (c) и горизонтального (d) срезов

Таблица 2

Параметры сетки для области из 6 слоев

Уровень	Время, сек.	# ячеек	# сколотых ячеек	# граней	# вершин
1	1.14	8293	3370	34 354	11 159
2	6.38	54 846	13 422	202 180	66 394
3	38.96	199 991	96 978	768 287	439 839



a) b) c) d)

Рис. 15. Сетка для 6 слоев (a), ее срез (b). Увеличения сетки в окрестности интерфейсной точки на грани (c, d)

Область задается аналитическими функциями, поэтому сложность cf пренебрежительно мала. Построим последовательность сеток, которые на каждом шаге имеют более глубокий уровень измельчения приграничных ячеек. В данном примере используются следующие ранее описанные параметры: $nb = 10$ и $k = 1$. В табл. 1 представлены параметры полученных сеток, а также время их построения. Из таблицы видно, что при увеличении числа вершин примерно в 4.5 раз общее время построения сетки увеличивается примерно в 4.5 раз. В этом примере основное время работы алгоритма занимает построение сколотых ячеек. На рис. 14 изображена сетка на первом уровне измельчения, а также некоторые ее срезы. Разные подобласти изображены разным цветом.

Далее, рассмотрим область, состоящую из шести геологических слоев. Область задается в виде поверхностной триангуляции каждого слоя, поэтому сложность cf в данном случае существенна. Построенная сетка сгущается к границе области и интерфейсам. Кроме этого, имеется дополнительное измельчение сетки вдоль некоторого фронта (рис. 15). Для проверки вычислительной сложности генератора была также построена последовательность сеток с разными уровнями измельчения и измерено время работы алгоритма. Результаты представлены в табл. 2. Из таблицы можно видеть, что время работы алгоритма растет пропорционально росту количества вершин. На рис. 15 изображена построенная сетка с первым уровнем измельчения, а также ее внутренняя структура. На рис. 15c и рис. 15d можно увидеть интерфейсные точки на гранях.

Заключение. В настоящей статье представлен метод построения сеток типа восьмеричное дерево со сколотыми ячейками. Сетки состоят преимущественно из кубических ячеек, а также из многогранных ячеек. Алгоритм может быть применен как для однородных, так и для неоднородных областей. Полученная сетка принадлежит классу конформных многогранных сеток в случае, если область является однородной. Если область разбита на подобласти, то сетка является слабо конформной. Проведен анализ конечности и вычислительной сложности данного алгоритма, а также приведены примеры сеток.

СПИСОК ЛИТЕРАТУРЫ

1. *Livnat Y., Shen H., Johnson C.R.* A near optimal isosurface extraction algorithm using the span space // IEEE Trans. Vis. Comp. Graphics. 1996. **2**. 73–84.
2. *Ho C.-C., Wu F.-C., Chen B.-Y., Chuang Y.-Y., Ouhyoung M.* Cubical marching squares: adaptive feature preserving surface extraction from volume data // Proc. EUROGRAPHICS. 2005. **24**, N 3. 537–545.
3. *Wu Z., Sullivan J.M.* Multiple material marching cubes algorithm // Int. J. Numer. Meth. Engng. 2003. **58**. 189–207.
4. *Lorensen W.E., Cline H.E.* Marching cubes: a high resolution 3d surface construction algorithm // ACM SIGGRAPH. 1987. **21**. 163–169.
5. *Wu Z.* Accurate and efficient three-dimensional mesh generation for biomedical engineering applications // Ph.D. Dissertation. Worcester Polytechnic Institute, 2001.
6. *Kobbelt L.P., Botsch M., Schwanecke U., Seidel H.-P.* Feature sensitive surface extraction from volume data // Proc. ACM SIGGRAPH. New York: ACM Press, 2001. 57–66.
7. *Bretonnet L., Li Y., Hirsch Ch.* 3D Navier–Stokes cutcell solver for octree meshes. Academy Colloquium on Immersed Boundary Methods. Amsterdam, 2009.
8. *Sutherland I.E., Hogdman G.W.* Reentrant polygon clipping // Comm. of the ACM on Graphics and Image Processing. 1974. **17**, N 1. 32–42.
9. *Hu S.R., Hen C.Z., Ankanhalli K.M.* Adaptive marching cubes // The Visual Computer. 1995. **11**, N 4. 202–217.
10. *Hekhar S.R., Fayyad E., Yagel R., Cornhill J.F.* Octree-based decimation of marching cubes surfaces // Proc. of IEEE Visualization. San Francisco, 1996. 335–342.
11. *Lopes A., Brodlie K.* Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing // IEEE Transactions on Visualization & Computer Graphics. 2003. **9**, N 1. 16–29.
12. *Wilhelms J., Gelder A.V.* Octrees for faster isosurface generation // ACM Transactions on Graphics. 1992. **11**, N 3. 201–227.
13. *Schroeder W.J., Zarge J.A., Lorensen W.E.* Decimation of triangle meshes // Comput. Graph. 1992. **26**. 65–70.
14. *Golodetz S.* Seeing things differently // Overload. 2007. **87**. 4–9.
15. *Shephard M.S., Georges M.K.* Three-dimensional mesh generation by finite octree technique // Int. J. for Numerical Methods in Engineering. 1991. **32**. 709–749.
16. *Ju T., Losasso F., Schaefer S., Warren J.* Dual contouring of Hermite data // Proc. of SIGGRAPH. 2002. **21**, N 3. 339–346.
17. *Zhang Y., Hughes T.J., Bajaj C.L.* An automatic 3D mesh generation method for domains with multiple materials // Comput. Methods Appl. Mech. Eng. 2010. **199**. 405–415.
18. *d'Otreppe V., Boman R., Ponthot J.-P.* Generating smooth surface meshes from multi-region medical images // Int. J. Numer. Meth. Biomed. Engng. 2011. **28**, N 6. 642–660.
19. http://www.numeca.com/
20. http://www-graphics.stanford.edu/data/3Dscanrep/

Поступила в редакцию
24.04.2013