ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 109 за 2018 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Якобовский М.В., Григорьев С.К.

Алгоритм гарантированной генерации тетраэдральной сетки проекционным методом

Рекомендуемая форма библиографической ссылки: Якобовский М.В., Григорьев С.К. Алгоритм гарантированной генерации тетраэдральной сетки проекционным методом // Препринты ИПМ им. М.В.Келдыша. 2018. № 109. 18 с. doi:10.20948/prepr-2018-109
URL: http://library.keldysh.ru/preprint.asp?id=2018-109

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Российской академии наук

М.В. Якобовский, С.К. Григорьев

Алгоритм гарантированной генерации тетраэдральной сетки проекционным методом

Москва — 2017

Якобовский М.В., Григорьев С.К.

Алгоритм гарантированной генерации тетраэдральной сетки проекционным методом

Рассматривается алгоритм построения тетраэдральной сетки. Принципиальное отличие алгоритма от других алгоритмов тетраэдризации состоит в гарантированности построения за конечное время. Рассматриваются основные этапы работы алгоритма, приводится оценка вычислительной сложности.

Ключевые слова: неструктурированные сетки, генератор расчетных сеток, рациональные числа, триангуляция.

Mikhail Vladimirovich Yakobovskiy, Sergej Konstantinovich Grigorjev

Projection-based guaranteed mesh generation algorith

Considered tetrahedral mesh generation algorithm. Principal difference from other algorithms consists in guaranteed filling of volume for a finite time. Considered main stages of algorithm, assessment of computational complexity is given.

Key words: unstructured meshes, mesh generation, rational numbers, triangulation

Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект 15-29-07090 офи_м.

Оглавление

Условные обозначения

t_i – TOP-треугольник, с номером i.
b_i – BOT-треугольник, с номером i.
w_i – WALL-треугольник, с номером i.
T – множество TOP-треугольников.
B – множество BOT-треугольников.
W – множество WALL-треугольников.
N – количество треугольников поверхности.
N_T – число TOP-треугольников.
N_B – число BOT-треугольников.
N_W – число WALL-треугольников.
S – плоскость проецирования.
N' – число BOT-треугольников, оказавшихся рядом с рассматриваемым треугольником в кэше.
N'' – число треугольников, полученное в плоскости проецирования после построения триангуляции набора рёбер, сформированных под рассматриваемым треугольником.
M – общее количество сформированных треугольных призм.

Введение

При численном моделировании физических процессов методами механики сплошной среды широко используются геометрические расчетные сетки. Вопрос их построения в трехмерных областях сложной формы (рис. 1), заданных поверхностными триангуляциями, является на сегодняшний день одним из наиболее сложных с алгоритмической точки зрения. В дальнейшем, говоря о разбиении пространства на тетраэдры, будем, по аналогии с двумерным случаем, употреблять термин «триангуляция». Хороший обзор известных подходов к построению тетраэдральных сеток приведён в [1,2].

Методы построения триангуляции с ограничениями на основе критерия Делоне [11]. Их достоинством является возможность контроля размеров элементов сетки. Способов размещения узлов достаточно много. Некоторые из них, например метод с названием «упаковка пузырьков», позволяют получить априори высокое качество триангуляции. Недостатком таких методов являются крайне высокая чувствительность к точности машинных вычислений [2] и сложность адаптации к границам области [15]. Операции, используемые в этих методах (нахождение центра и радиуса описанной окружности и т.п.), дают на выходе иррациональные величины, что, вкупе с постоянным использованием подобных операций, ведет к неизбежному накоплению ошибок округления [2].

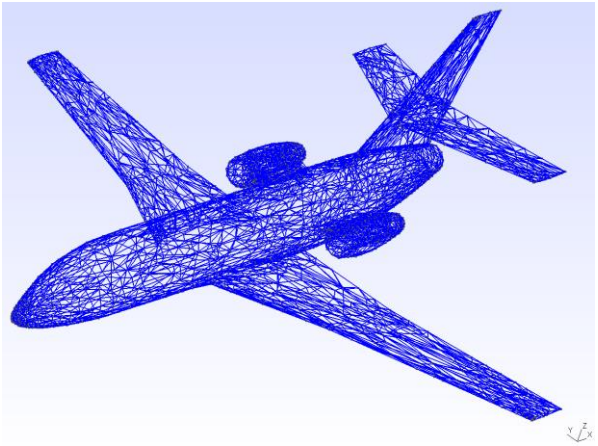


Рис.1. Пример сложной области.

Методы граничной коррекции [2] являются самыми быстрыми из итерационных методов, однако обладают существенными недостатками. Построение сеток в таких методах происходит в два этапа. На первом этапе производится триангуляция некоторой «супер-области», полностью включающей исходную область. На втором этапе все узлы полученной сетки, лежащие вне исходной области, удаляются. При этом узлы, лежащие у границы области, проецируются на саму границу области. Для компенсации геометрических искажений элементов вблизи границ применяют алгоритмы оптимизации сеток: локальное перестроение сетки [8], перестроение сетки [9]. Однако эти методы плохо применимы для областей с заданной триангуляцией границ, что является существенным недостатком [1].

Для триангуляции областей с заданной граничной триангуляцией применяют методы исчерпывания (в англоязычных источниках эти методы имеют название “advancing front”) [10]. Отправной точкой для этих методов является граничная триангуляция исходной области, причем не грубая триангуляция, а триангуляция, наиболее соответствующая требованиям разработчика, поскольку в дальнейшем никаких изменений этой триангуляции производиться не будет. На основе треугольников границы строятся отдельные тетраэдры (формируемый тетраэдр может включать в себя насколько треугольников границы), причем четвертой вершиной тетраэдра служит либо другая уже существующая вершина «фронта», либо дополнительный узел, помещаемый внутрь заданной области. Однако, несмотря на кажущуюся простоту, подобные алгоритмы имеют ряд существенных недостатков. Во-первых, затратность проверки правильности построения каждого нового тетраэдра, так как необходимо удостовериться в отсутствии пересечений с уже существующими тетраэдрами. Во-вторых, и это основная проблема, как правило, в результате работы алгоритма «фронт» формирует области все более сложной формы, которые невозможно триангулировать без добавления дополнительных точек внутрь области. В результате сходимость процесса оказывается под вопросом. Более того, размер наименьшего элемента

формируемой сетки неконтролируемо уменьшается, что крайне негативно отражается на возможности выполнения на такой сетке численного моделирования [2].

Существуют также методы, являющиеся комбинацией этих методов [12].

Упомянутые методы реализованы в большом количестве пакетов, таких как ANI3D [13], в котором также присутствуют методы модификации и перестроения сеток, пакет SALOME [14] и другие.

В рамках данной работы рассматривается прямой алгоритм построения триангуляции трехмерной области, отличный от всех вышеупомянутых. Преимущество рассматриваемого алгоритма состоит в его алгоритмических свойствах. Во-первых, метод является прямым, что гарантирует конечное число операций для исходной поверхности любой сложности. Во-вторых, метод обеспечивает возможность гарантированного построения трехмерной триангуляции на множестве вершин, координаты которых являются рациональными числами ограниченной разрядности. Важно отметить, что требуемая для проведения вычислений максимальная разрядность линейно зависит только от разрядности представления координат исходных вершин поверхностной триангуляции и не зависит от сложности исходной поверхности. В-третьих, алгоритм отличается высокой степенью внутреннего параллелизма.

Следует отметить два основных недостатка предлагаемого алгоритма. Первый заключается в том, что формирование триангуляции с вершинами, заданными в узлах рациональной решетки, не гарантирует существования топологически непротиворечивой триангуляции, возникающей после отображения вершин на решетку, соответствующую машинному представлению вещественных чисел, поскольку разрешение второй решетки ниже, чем первой. Второй, более важный недостаток, заключается в формировании триангуляции низкого, с точки зрения дальнейшего моделирования, качества. Таким образом, основное назначение формируемой триангуляции — предварительное разбиение трехмерной области на тетраэдры, например, с целью выявления топологии области или для упрощения этапа предварительного размещения узлов в областях сложной формы.

Алгоритм гарантированного заполнения объёма

В рамках алгоритма используется конечный замкнутый объём, ограниченный поверхностной триангуляцией, состоящий из правильно примыкающих друг к другу непересекающихся ориентированных треугольников (разрешены касания вершинами и рёбрами).

В качестве исходных данных алгоритм требует ориентированную поверхностную триангуляцию, не содержащую самопересечений. Алгоритм содержит четыре этапа:

- построение проекции и формирование на её основе призм;
- стыковка призм между собой;
- триангуляция боковых стенок призм;

- формирование тетраэдров на основе треугольных призм с триангулированными боковыми гранями путём добавления точки в геометрический центр такой призмы.

Рассмотрим каждый из этапов в отдельности.

Построение проекции и формирование призм

Для проведения ортогонального проецирования необходимо для начала выбрать ось проецирования. Пусть луч проецирования задан трёхмерным вектором проецирования $\overrightarrow{pr}$, которому ортогональна плоскость проецирования *S*.

Для каждого из треугольников вычислим нормаль к его плоскости. Использование арифметики рациональных чисел с произвольной разрядностью числителя и знаменателя позволяет точно определить такой вектор, избежав возникновения ошибок округления при расчёте. В силу ориентированности исходной поверхности можно разбить треугольники на три группы по признаку сонаправленности проекции вектора нормали к треугольнику на луч проецирования с вектором проецирования. Для определённости положим, что все вектора нормали направлены вовне заданной области (рис.2). Тогда если проекция вектора нормали противонаправлена с положительным вектором проецирования, то треугольник будет считаться TOP-треугольником ($\overrightarrow{n_2}$ и его проекция $\overrightarrow{n_{2_z}}$ на рис. 2), если сонаправлена, то BOT-треугольником ($\overrightarrow{n_3}$ и его проекция $\overrightarrow{n_{3_z}}$ на рис. 2), и если вектор нормали ортогонален оси проецирования, то такой треугольник будет считаться WALL-треугольником ($\overrightarrow{n_1}$ на рис. 2).

Важно, что использование арифметики рациональных чисел с произвольной разрядностью гарантирует отсутствие ошибок округления при вычислении, что позволяет ввести подобное разделение.

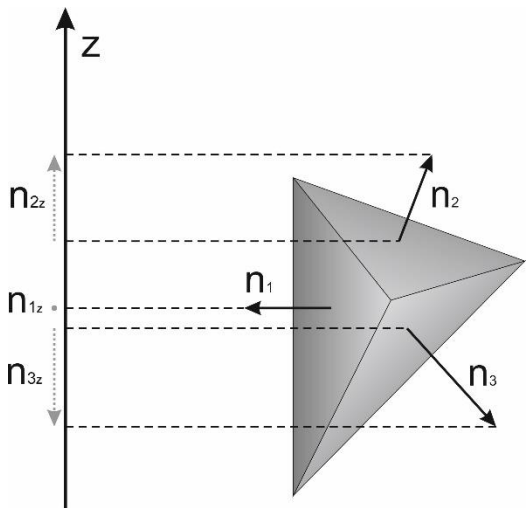


Рис. 2. Проекции векторов нормали на ось OZ

Для дальнейшего обсуждения введём ряд определений.

После введения указанной классификации треугольников сформируем проекцию всех BOT-треугольников на TOP-треугольники. На этом этапе алгоритма WALL-треугольники в рассмотрении не участвуют. Обозначим множество TOP-треугольников через *T*, множество BOT-треугольников через *B*, множество WALL-треугольников через *W*, общее количество треугольников поверхности через *N*, количество TOP-, WALL- и BOT-треугольников через N_T, N_W, N_B соответственно.

def. 1. Рассмотрим произвольный треугольник $t \in T$. Тогда область пространства, ограниченная плоскостью треугольника *t* сверху, ограниченная плоскостями, проходящими через рёбра t и параллельными лучу проецирования, и плоскостью проецирования *S*, расположенной заведомо ниже любого элемента поверхностной триангуляции, назовём *областью проецирования* треугольника *t*. (На рис. 3 для треугольника ABC показана область проецирования $ABCA'B'C'$).

def. 2. Будем говорить, что $\Delta\, DEF$ *пересекается с областью проецирования* $\Delta\, ABC$, если существует такая внутренняя точка $P \in \Delta\, DEF$, что решение уравнения $P + k * \overrightarrow{pr} = P'$ относительно k будет отрицательно, где P' — проекция точки P на рассматриваемый треугольник (см. рис. 3, точка P'' — проекция точек P и P' на плоскость проецирования *S*).

def. 3. Будем говорить, что *треугольник* DQR *принадлежит области проецирования треугольника* ABC, если для любой точки $P \in \Delta\, DQR$ выполняется условие 2 (На рис. 3 $\Delta\, DQR$ попадает полностью в область проецирования треугольника $\Delta\, ABC$).

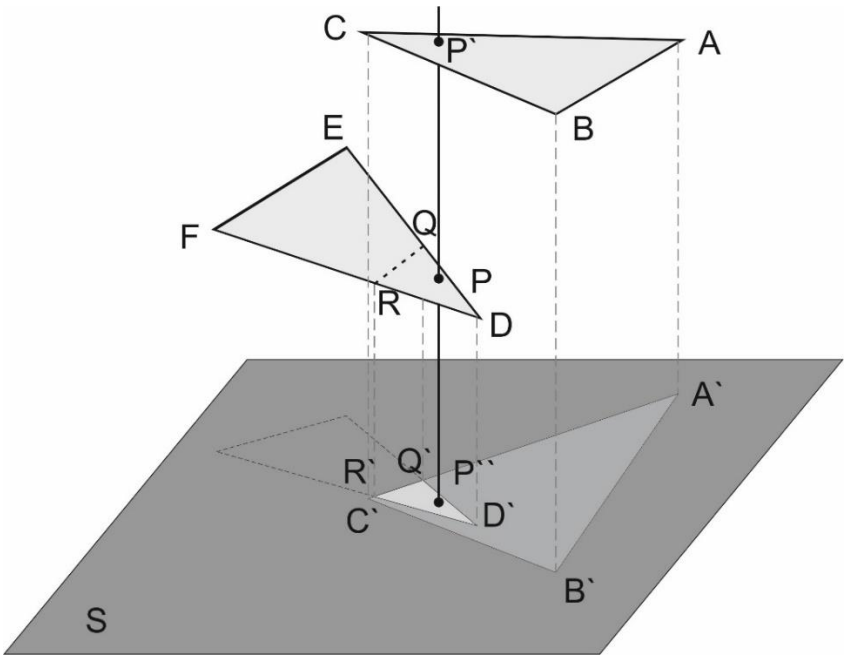


Рис. 3. Область проецирования треугольника ABC, плоскость проецирования S, проекция треугольника DEF в область проецирования, экранированная рёбрами исходного треугольника.

def. 4. Пусть заданы треугольники ABC и DEF, такие что ΔDEF пересекается с областью проецирования ΔABC. Тогда построение множества вершин и рёбер (отрезков рёбер) ΔDEF, попавших при проецировании на плоскость S в проекцию ΔABC, назовём операцией *экранирования треугольника DEF рёбрами треугольника ABC* (в [7] описана аналогичная операция «отсечение полигона», отличие в нашем случае состоит в сохранении данных об изначальном взаимном расположении объектов в трёхмерном пространстве).

На рис. 3 показано, как часть треугольника DEF, а именно треугольник DQR, принадлежит области проецирования треугольника ABC. При этом при построении проекции на плоскость проецирования S в проекцию $A'B'C'$ попадает только треугольник $D'Q'R'$. В результате экранирования треугольника DEF треугольником ABC формируется набор вершин D', Q', R' и набор инцидентных им рёбер. Заметим, что операция экранирования применяется только для треугольников, удовлетворяющих условию 2.

def. 5. Пусть заданы треугольники ABC, DQR и $A'B'C'$. Будем называть *экранированием поверхностью треугольника DQR* операцию удаления всех вершин и рёбер (отрезков рёбер) треугольника $A'B'C'$, попавших одновременно в область проецирования треугольников ABC и DQR.

Например (рис. 3), треугольник DQR экранирует поверхностью область $D'Q'R'$ треугольника $A'B'C'$ относительно треугольника ABC, таким образом, от треугольника $A'B'C'$ остаётся многоугольник $A'B'C'R'D'Q'$.

В рамках дальнейшего рассмотрения алгоритма будем считать, что ось проецирования совпадает с осью *OZ*, а вектор проецирования $\overrightarrow{pr}$ противонаправлен положительному направлению оси *OZ*. Тогда плоскость проецирования S параллельна плоскости *OXY*. Таким образом, все точки исходной поверхности проецируются на эту плоскость путём замены их *z*-координаты на 0.

Построение проекции для каждого треугольника $t_i \in T, i = \overline{0, N_T - 1}$ требует поиска для обнаружения всех треугольников, оказавшихся близко к проекции Δt_i в плоскости S. В качестве алгоритма первичного грубого поиска целесообразно использовать двумерный геометрический кэш [3,4], формируемый в плоскости проецирования. Такой кэш представляет собой заведомо более простое разбиение плоскости, в нашем случае разбиение на прямоугольники. В качестве границы кэша используется ограничивающий проекцию расчетной области прямоугольник. Для каждой ячейки формируется набор ссылок на попавшие в неё треугольники. В силу регулярности разбиения плоскости номер ячейки кэша, содержащей вершину треугольника, может быть вычислен. Тогда каждый треугольник попадает во все ячейки кэша, в которые попадает ограничивающий его прямоугольник, номера этих ячеек можно получить за $O(1)$.

Выборка соседей треугольника состоит в получении всех уникальных элементов всех ячеек, в которые попал рассматриваемый треугольник. Эта

операция может быть выполнена при помощи сортировки слиянием. Обозначим количество таких элементов N'.

Построение проекции на t_i подразумевает, что на момент рассмотрения очередного треугольника $b_j \in B, j = \overline{0, N_B - 1}$ необходимо проверить следующие два утверждения: пересекается ли Δb_j и область проецирования Δt_i и существует ли линия видимости между частью поверхности Δb_j и частью поверхности Δt_i.

Возможно несколько случаев попадания Δb_j в область проецирования Δt_i:

- Δb_j принадлежит области проецирования Δt_i;
- Δb_j может пересечься с областью проецирования так, что ни точки, ни отрезки рёбер не попадут в область проецирования при построении проекции;
- Δb_j пересекается с областью проецирования, и в саму область попало несколько отрезков рёбер.

После получения всех треугольников, пересекающихся с областью проецирования рассматриваемого t_i, необходимо каждый из BOT-треугольников этого множества экранировать рёбрами t_i, а также экранировать поверхностями других BOT-треугольников. В результате формируется набор вершин и рёбер, лежащих в плоскости S в границах проекции t_i на S, соответствующий видимости элементов BOT-треугольников вдоль оси проецирования триангуляции под t_i.

Следующим шагом является построение триангуляции с ограничениями полученного набора вершин и рёбер, расположенных в плоскости S. Это можно сделать при помощи алгоритма плоского заметания [3,6], использовать алгоритм построения двумерной триангуляции Делоне с динамическим кэшированием поиска, или одним из методов, представленных в [3,4]. В результате получается некоторое множество треугольников Q', лежащее в плоскости проецирования S, которому соответствует множество треугольников Q, являющееся проекцией треугольников из Q' в плоскость треугольника t_i. Обозначим количество таких треугольников N''.

Каждому треугольнику из множества Q в плоскости треугольника t_i соответствует одна и только одна треугольная область на поверхности BOT-треугольника по построению. Каждая пара таких треугольников образует треугольную призму по построению. Отметим, что набор сформированных призм может содержать и вырожденные, вплоть до тетраэдров, случаи.

Для работы алгоритма используется структура LAY. Эта структура содержит набор точек, рёбер и треугольников, попавших в проекцию треугольника t_i на плоскость проецирования.

Алгоритм построения проекции и формирования призм выглядит следующим образом:

S1. [Цикл по i.] для $i = \overline{1, N_t}$, по всем TOP-треугольникам.

S2. [Найти ближайшие BOT-треугольники.] Сформировать для i-го TOP-треугольника множество геометрически близких BOT-треугольников. Количество таких соседей установим в N'. Очистить LAY.

S3. [Цикл по j.] Установить $j = 1$. После завершения цикла перейти на шаг **S8**.

S4. [Проверить попадание в область проецирования.] Проверить пересечение треугольника j с областью проецирования треугольника i. Если пересечение найдено, перейти к шагу **S5**, иначе перейти к шагу **S7.**

S5. [Экранировать рёбрами TOP-треугольника.] Выполнить экранирование j-го треугольника рёбрами i-го треугольника. Перейти к шагу **S6.**

S6. [Экранировать поверхностями BOT-треугольников, принадлежащих области проецирования.] Выполнить экранирование области, полученной в результате шага **S5**, поверхностями всех треугольников, содержащихся в LAY. Результат записать в LAY.

S7. [Условие завершения цикла по j.] Если $j \leq N'$, то установить $j = j + 1$ и перейти к шагу **S4**, иначе перейти к шагу **S8**.

S8. [Триангулировать проекцию.] Применить алгоритм двумерной триангуляции к LAY. В результате структура содержит N'' треугольников.

S9. [Цикл по k.] Выполнить шаг **S10** для $k = \overline{1, N''}$.

S10. [Сформировать призмы.] Для k-го треугольника из LAY восстановить его проекции в плоскости i-го треугольника и соответствующего ему BOT-треугольника. Полученные в результате 6 точек, представляющие собой два треугольника, сохранить как треугольную призму.

Стыковка призм

После формирования набора призм необходимо выяснить взаимное расположение призм между собой.

Существует несколько возможных вариантов соприкосновения призм (Рис.4):

- касание полностью гранью, т.е. совпадают 4 точки двух призм (рис. 4а);
- касание внутренними частями граней (рис. 4б);
- касание ребром (рис. 4в);
- касание частями рёбер (рис. 4г);
- касание частью ребра и гранью (рис. 4д).

Эти варианты касания призм приводят к необходимости решения задачи поиска двух соприкасающихся призм.

Грубый поиск соприкасающихся призм может использовать геометрический кэш, описанный выше. Однако так как призма является трёхмерным объектом, то целесообразно использовать трёхмерный геометрический кэш. Такой кэш формируется аналогичным образом, с тем

отличием, что ячейкой кэша является теперь не прямоугольник, а параллелепипед.

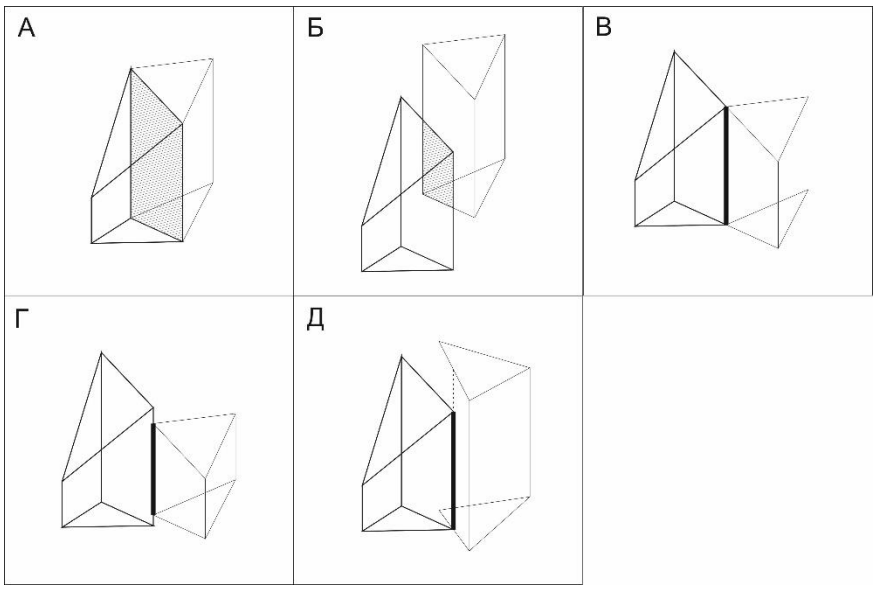


Рис. 4. Рассматриваемые варианты касания призм.

Так как все боковые рёбра призм параллельны оси проецирования и друг другу, то при рассмотрении возможности касания боковыми гранями достаточно проверять только одну из точек вертикальных рёбер на попадание в плоскость боковой грани.

После обнаружения касания любым из описанных выше способов необходимо добавить в топологию каждой призмы все новые точки и части рёбер, образующиеся при соприкосновении призм. Следует помнить, что при добавлении новых точек необходимо разбивать старые рёбра, на которые эти точки могли попасть.

Алгоритм стыковки боковых граней призм выглядит следующим образом:

S1. [Цикл по i.] для $i = \overline{1, M}$.

S2. [Найти ближайшие призмы.] Сформировать для i-ой призмы множество геометрически близких призм. Количество таких соседей установим в M'.

S3. [Цикл по j.] для $j = \overline{1, M'}$. После завершения цикла перейти на шаг **S8**.

S4. [Цикл по k.] Выполнить шаг **S5** для $k = \overline{1,3}$, по боковым граням i-ой призмы.

S5. [Цикл по l.] Выполнить шаг **S6** и, при необходимости, **S7** для $l = \overline{1,3}$, по вершинам верхнего основания j-ой призмы.

S6. [Проверка попадания в плоскость.] Проверить попадание l-ой вершины в плоскость k-ой боковой грани i-ой призмы. Результат (0 или 1)

записывается в свою ячейку *count*. Если есть попадание, выполнить шаг **S7**.

S7. [Добавление вертикальных рёбер в топологию.] Проверить попадание вертикального ребра *j*-ой призмы в боковую грань *i*-ой призмы. Если есть попадание, то добавить точку пересечения и отрезки ребра в топологию *i*-ой призмы.

S8. [Вычисляем количество касаний.] Суммируем значения массива *count* в переменную *sum*. Если $sum = 0$, перейти к шагу **S9**. Если $sum > 0$, перейти к шагу **S10**.

S9. [Проверка пересечения рёбер боковых граней с рёбрами основания.] Найти точки пересечения вертикальных рёбер *i*-ой призмы с рёбрами оснований *j*-ой призмы. Найденные точки добавляем в топологию *i*-ой призмы, разбивая соответствующее ребро на части.

S10. [Построение пересечения боковых граней.] Проверить попадание отрезков оснований *j*-ой призмы на боковую грань *i*-ой призмы. Добавить соответствующие элементы в топологию *i*-ой призмы.

При этом построение всех точек пересечения между рёбрами рассматриваемых призм делается на основании неразбитых рёбер (т.е. обе призмы рассматриваются как 6 точек), а точки при добавлении разбивают уже существующие отрезки рёбер (т.е. при добавлении точки в топологию призмы рассматриваются все рёбра, содержащиеся в этой призме).

После завершения этой процедуры для всех призм производится введение в топологию всех WALL-треугольников. Так как эти треугольники параллельны оси проецирования, то все их точки и рёбра могут так или иначе попасть только на боковые грани призм. Соответственно, для каждого WALL-треугольника необходимо найти все призмы, с элементами которых он может пересечься, после чего способом, аналогичным со стыковкой боковых граней, добавить все точки и части рёбер к топологии призм.

Как и на предыдущем этапе, здесь также необходимо проводить первичный грубый поиск соседей. Для этого можно использовать и двумерное кэширование призм, однако так как призмы могут располагаться друг над другом, то целесообразнее использовать трёхмерный кэш.

Триангуляция боковых стенок

После добавления всех элементов изначальной сетки в призмы и стыковки всех призм между собой необходимо триангулировать боковые грани призм. В результате работы предыдущего этапа алгоритма на гранях призм сформировался некоторый набор точек и рёбер. На этом этапе необходимо таким образом построить двумерную триангуляцию каждой грани, чтобы для

любых двух соседних призм триангуляции общей области касания на гранях совпадали. Это возможно сделать двумя способами.

Первый способ предполагает уникальную обработку каждой грани. Таким образом, каждое ребро будет добавлено в топологию один раз. Второй способ заключается в создании инвариантного алгоритма двумерной триангуляции.

Первый способ хорош тем, что каждая грань будет обработана уникально, и, следовательно, уменьшает объём выполненной работы. Второй способ, хотя и удваивает объём выполняемой работы, позволяет обрабатывать каждую из призм по-отдельности.

Заполнение построенных призм тетраэдрами

После триангуляции всех граней призм задача разбиения на тетраэдры сводится к добавлению в центр масс призмы точки. Таким образом, каждая призма заполняется некоторым набором тетраэдров. Так как призмы заполняют весь объём исходной области по построению, то заполнение призм тетраэдрами решает задачу тетраэдризации исходной области.

Применение рациональных чисел произвольной разрядности

Ни на одном из этапов алгоритма не содержится никакие операции, результатом которых могут являться иррациональные числа. Это позволяет использовать арифметику рациональных чисел с произвольной разрядностью (далее – рациональная арифметика).

Использование рациональной арифметики на всех этапах использования алгоритма позволяет избежать погрешности округления при вычислениях, что позволяет существенно упростить реализацию алгоритма, но требует отдельного рассмотрения процесса возврата от рациональной арифметики к вещественным числам.

Существенным недостатком применения подобной арифметики является увеличение затрат на выполнение элементарных операций.

Оценка вычислительной сложности

Вычислительная сложность алгоритма состоит из суммы вычислительных сложностей всех этапов алгоритма.

Рассмотрим наихудший случай.

Пусть N – количество треугольников исходной поверхности, M – количество призм, формируемых во время первого этапа. Число призм M зависит в основном от поверхностной триангуляции исходной области. Отметим, что на практике $N < M < N^2$.

В силу достаточно сильной зависимости вычислительной сложности алгоритма от топологии области, для каждого этапа существует свой собственный худший случай, дающий наихудшую оценку вычислительной сложности.

Для первого этапа в наихудшем случае вычислительная сложность будет иметь оценку $O(N^2)$. Пример такой области изображен на рис. 5.

Для второго этапа в наихудшем случае вычислительная сложность так же будет оцениваться как $O(M^3)$. Отметим, что такая оценка достигается только для случая, показанного на рис. 5. Случай, когда M пропорционально N^2, однако, даст более низкую оценку вычислительной сложности.

Заметим, что, хотя в [5] указано, что построение пересечения двух полиэдров имеет квадратичную оценку вычислительной сложности по числу вершин полиэдра, в рассматриваемом случае эта оценка является постоянной величиной, т.к. все призмы при проведении стыковки состоят из 6 вершин (включая возможно повторяющиеся вершины в вырожденных случаях).

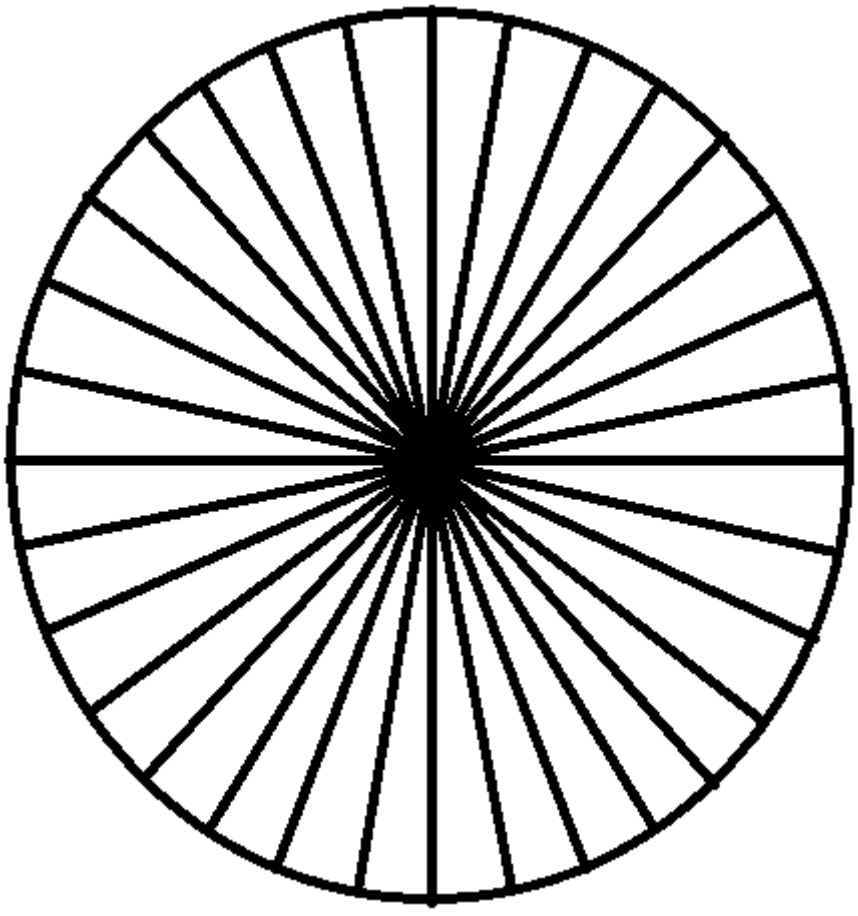


Рис. 5. Наихудший случай для второго этапа.

Для третьего этапа наихудшим будет случай, когда на одну боковую грань одной призмы попадёт N – 2 вершины. Тогда вычислительная сложность в наихудшем случае будет составлять $O(N \log N)$.

Однако каждый из упомянутых выше случаев будет наихудшим только для одной из частей алгоритма. Кроме того, существует такая поверхность, которая объединит наихудшие случаи для первого и второго этапов. Общая

вычислительная сложность для всего алгоритма в наихудшем случае может быть оценена как $O(M^3)$.

Вычислительная сложность четвёртого этапа алгоритма постоянна и составляет $O(M)$ во всех случаях.

Наихудшая вычислительная сложность для всего алгоритма будет равна максимальной из вычислительных сложностей этапов, следовательно, худшая вычислительная сложность из возможных составляет $O(M^3)$.

Рассмотрим наилучший случай.

В этом случае поверхность тела представляет собой квазиравномерную сетку, в которой для каждого TOP-треугольника существует только один, и притом полностью попавший в область видимости, BOT-треугольник. Кроме того, распределение количества треугольников в пространстве также равномерно, и в топологии структуры отсутствуют WALL-треугольники.

В этом случае известно, что под каждым TOP-треугольником сформируется только одна призма, алгоритм двумерной триангуляции использован не будет. Следовательно, вычислительная сложность алгоритма сведётся к вычислительной сложности поиска соседних треугольников. Тогда, в силу квазиравномерного распределения треугольников в рамках расчетной области, а так же того, что под каждым из TOP-треугольников сформировалась только одна призма, можно считать, что вычислительная сложность первого этапа будет равна $O\left(c_1 \frac{N}{2}\right)$, где c_1 – константа, обозначающая число треугольников, находящихся рядом с рассматриваемым треугольником в кэше. Число сформированных призм так же будет равно $\frac{N}{2}$.

Аналогичным образом такая же оценка получается и на втором этапе: квазиравномерное распределение треугольников в пространстве приводит в таком случае и к квазиравномерному распределению призм, следовательно, число призм, находящихся рядом в блоках кэша, также будет невелико и приблизительно одинаково для всех призм, а значит, вычислительная сложность будет составлять $O\left(c_2 \frac{N}{2}\right)$, где c_2 – константа, обозначающая число призм, находящихся рядом с рассматриваемой призмой в кэше.

В силу отсутствия WALL-треугольников призмы буду касаться друг друга гранью полностью, т.е. на предыдущем этапе не сформируется никаких дополнительных точек, и, следовательно, вычислительная сложность этапа будет равна $O\left(\frac{N}{2}\right)$.

Таким образом, можно сказать, что в лучшем случае вычислительная сложность алгоритма является линейной и равна $O\left(\frac{N}{2}\right)$.

Для второго этапа наихудшим случаем будет следующий случай, представленный на рис. 5.

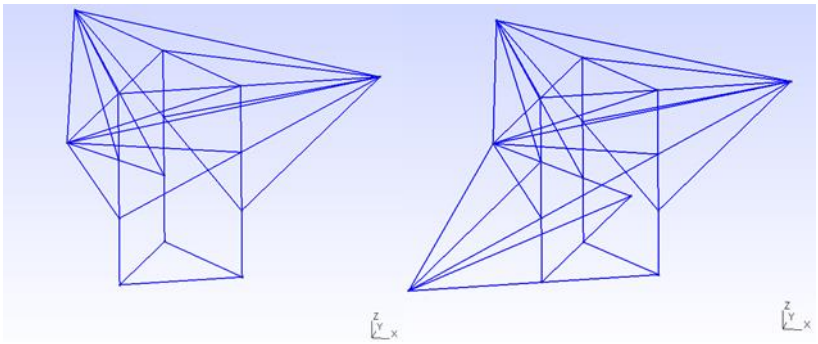
Это коническая поверхность, все BOT-треугольники которой имеют одну общую точку в центре описанной вокруг многоугольника основания

окружности, а все TOP-треугольники имеют центр в вершине конуса. В отсутствие подобных структур возможно реализовать алгоритмы кеширования [3] так, чтобы привести вычислительную сложность к $O\left(N\sqrt{N}\right)$ (см. сложность извлечения из кэша в [3]). Если же реализовывать трёхмерный кэш, то вычислительная сложность может упасть до $O(m^3 * N)$, где m – размерность трёхмерного кэша.

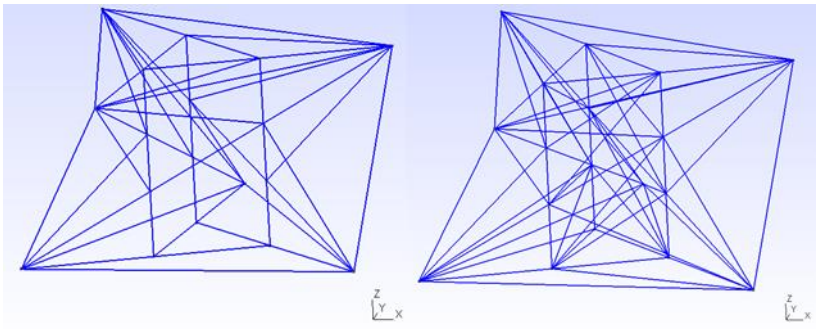
Для последнего, третьего этапа формирования триангуляции, кроме случая, когда большая часть точек попадает на одну призму, вычислительная сложность может быть оценена как $O(N)$.

Примеры работы алгоритма

Наглядно продемонстрировать работу алгоритма можно на примере разбиения призмы Шёнхардта (многогранника Шёнхардта) (рис. 6).



а) б)



в) г)

Рис. 6. Заполнение призмы Шёнхардта.

Рис. 6 иллюстрирует работу алгоритма заполнения призмы Шёнхардта. На рис. 6а показаны триангуляция верхнего основания призмы, а также набор призм, сформированных под этим основанием. На рис. 6б, рис. 6в показано добавление новых призм в общую топологию, с построением точек пересечения между рёбрами, а также вершин призм, попавших на рёбра других призм. На рис. 6г показан этап построения триангуляции боковых граней призм.

Литература

1. Галанин М.П., Щеглов И.А. Разработка и реализация алгоритмов трехмерной триангуляции сложных пространственных областей: прямые методы // Препринты ИПМ им. М.В. Келдыша. 2006. № 10,
2. Галанин М.П., Щеглов И.А. Разработка и реализация алгоритмов трехмерной триангуляции сложных пространственных областей: итерационные методы // Препринты ИПМ им. М.В. Келдыша РАН. № 9, 2006.
3. Скворцов А.В. Триангуляция Делоне и ее применение. – Томск: Томский университет, 2002. 128с.
4. Скворцов А.В., Мирза Н.С. Алгоритмы построения и анализа триангуляции. – Томск: Изд-во Том. ун-та, 2006. – 168 с.
5. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение: Пер. с англ. – М.: Мир, 1989. – 478 с. ISBN 5-03-001041-6, 0-387-96131-3
6. Ивановский С.А., Симончик С.К. Алгоритмы вычислительной геометрии. Пересечение отрезков: метод заметания плоскости // Компьютерные инструменты в образовании. - СПб.: Изд-во ЦПО "Информатизация образования", 2007, N4, С. 18-33.
7. Ласло М. Вычислительная геометрия и компьютерная графика на С++. – М.:БИНОМ, 1997 ISBN: 0-13-290842-5, 5-7889-008-8
8. Borouchaki H., Hecht F., Saltel E., George P. L. Reasonably efficient Delaunay based mesh generator in 3 dimensions // Proc. 4th Internat. Meshing Roundtable. 1995. P. 3–14.
9. Du Q., Wang D. Recent progress in robust and quality Delaunay mesh generation // J. Comput. and Appl. Math. 2006. V. 195. P. 8–23.
10. Ito Y., Shih A., Soni B. Reliable isotropic tetrahedral mesh generation based on an advancing front method // Proc. 13th Internat. Meshing Roundtable. 2004. P. 95–106.
11. Shewchuk J.R. Constrained Delaunay tetrahedralizations and provably good boundary recovery // Proc. 11th Internat. Meshing Roundtable. 2002. P. 193–204.
12. Danilov A.A. Unstructured tetrahedral mesh generation technology // Ж. вычисл. матем. и матем. физ., 50:1 (2010), 146–163; Comput. Math. Math. Phys., 50:1 (2010), 139–156.
13. Пакет для работы с трёхмерными сетками ANI3D http://sourceforge.net/projects/ani3d/

14. The Open Source Integration Platform for Numerical Simulation SALOME http://www.salome-platform.org/
15. Гаранжа В.А., Кудрявцева Л.Н. Построение трехмерных сеток Делоне по слабоструктурированным и противоречивым данным // Ж. вычисл. матем. и матем. физ., 2012, том 52, номер 3, 499–520.