Содержание

1 Введение

Большинство методов численного расчета основаны на переходе от протяженной задачи к дискретной, т.е. к такой задаче, в которой не находят функции распределения той или иной физической величины, а находят лишь значения этой величины в некотором конечном наборе точек. При этом пространство, параметры которого расчитываются в задаче, разбивается на конечное число элементов, целиком заполняющих это пространство. Элементы, как правило, являются простыми фигурами и для них можно решить задачу. Для пространственной задачи элементы заключают в себе объем, например таким элементом может быть тетраэдр. Для плоских задач, либо пространственных задач, но тем или иным способом сводящихся к плоским, используются различные плоские полигональные фигуры.

Проблема оптимального разбиения пространства задачи является подчас оченьсложной. На каждый элемент разбиения могут накладываться довольно жесткие ограничения. К тому же, в пространстве задачи могут быть некие характерные области, где параметры меняются довольно резко. Например, в задачах механики твердого деформируемого тела, такие области образуются вблизи концентраторов напряжений и в них напряжения меняются чрезвычайно резко. Такие области требуют более частого разбиения.

В современных задачах подобного класса разбиение включает в себя огромное количество элементов, как правило не менее тысячи. Создавать вручную такие разбиения не представляется возможным, поэтому в дополнение к основным расчетным алгоритмам должны быть созданы и алгоритмы генерации разбиения.

Расчет тонких плит методом конечных элементов представляет из себя как раз такую задачу. В ней пространственная задача сводится к «псевдоплоской», разбиение производится в плоскости, а толщина является параметром каждого элемента. Для максимальной универсальности и упрощения алгоритма разумно выбрать треугольный конечный элемент — самый простой двумерный элемент. Поэтому сетка будет треугольной. На треугольники

сетки накладываются два основных ограничения: треугольник не должен быть слишком вытянутым (skinny triangle), т.е. его минимальный угол ограничен сверху; площадь треугольника не должна превышать определенного значения, ввиду того, что перемещения находятся только в вершинах и центре тяжести треугольника и есть шанс упустить максимальное перемещение (следовательно и напряжение) при достаточно больших элементах.

Таким образом, необходим алгоритм, разбивающий полигональную фигуру. При этом на эту фигуру должно накладываться минимум ограничений: не должно быть никаких ограничений, касающихся выпуклости; она может содержать отверстия,выделенные области (другие физ. характеристики, нагрузки и т.п.). Треугольники получаемой сетки должны удовлетворять двум перечисленным выше требованием. Желательно, чтобы алгоритм был адаптивным и требовал как можно меньшего участия человека.

1.1 Обзор существующих методов

Основными понятиями теории триангуляции и плоских треугольных сеток являются диаграммы Вороного (см. рис. 1) и триангуляция Делоне (Delaunay triangulation см. рис. 1). Диаграммами Воронова на плоскости для множества точек называют системы полигональных фигур, образуемых линиями, перпендикулярными отрезкам, соединяющими соседние точки и проходящим через середины этих отрезков. Триангуляция Делоне на плоскости — это множество не пересекающихся треугольников, в котором ни одна точка, не принадлежащая данному треугольнику не попадает в окружность, описанную вокруг этого треугольника. Все рассматриваемые методы оперируют этими понятиями в том или ином виде. Приведу лишь названия наиболее известных алгоритмов генерации сетки:

1. Radial sweep algorithm.
2. Recursive split algorithm (Алгоритм последовательного разбиения).

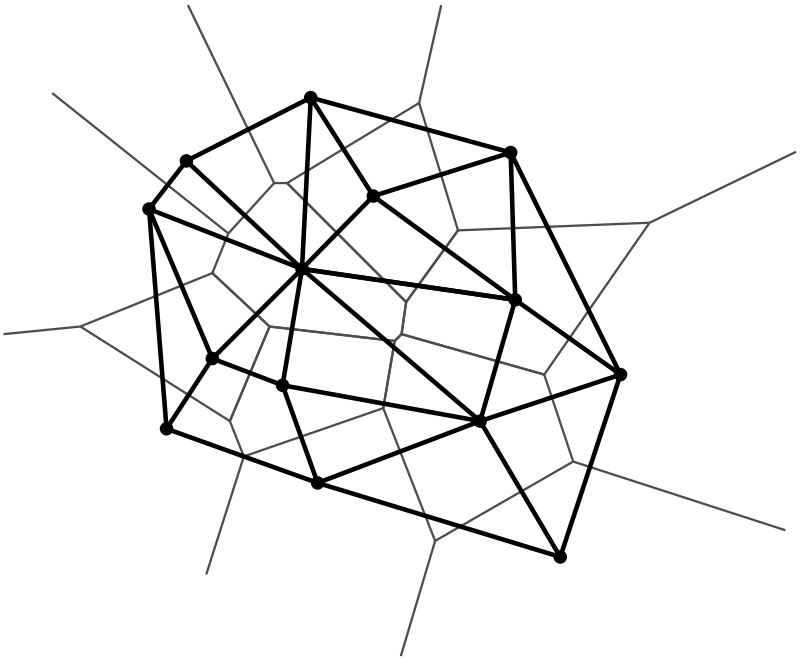

Рис. 1: Триангуляция Делоне (Delaunay triangulation).

3. Divide-and-conquer algorithm (Алгоритм деления-и-включения).
4. Step-by-step algorithm.
5. Modified hierarchical algorithm (Модифицированный иерархический алгоритм)
6. Incremental algorithm.
7. Incremental delete-and-build algorithm.

Все эти алгоритмы не предназначены непосредственно для генерации конечно-элементной сетки. Они подразумевают наличие опорных точек, которые будут потом узлами сетки. К сетке для метода конечных элементов предъявляется ряд особых требований. Во-первых, треугольники не должны быть сильно вытянутыми, т.к. наличие таких треугольников отрицательно сказывается на точности результатов расчета. Во-вторых, должен быть учтен характер работы конструкции и ее геометрия, т.е. в местах концентрации

напряжений сетка должна сгущаться. Все эти требования учтены в алгоритме генерации треугольных иррациональных конечно-элементных сеток Рапперта.

2 Алгоритм Рапперта

Этот алгоритм довольно новый, он был опубликован в 1994 году [1]. Его разработал Джим Рапперт еще будучи студентом в рамках проекта, поддерживаемым NASA. Алгоритм адаптирован для метода конечных элементов и перечисленные выше требования, предъявляемые к конечно-элементным сеткам удовлетворяются автоматически. К его преимуществам можно отнести и то, что алгоритм легко параметризуется и управляется.

Если говорить точнее, этот алгоритм не является алгоритмом генерации, а лишь алгоритмом улучшения качеств сетки (refinement). В нашем случае входными данными для него будут являться геометрия конструкции в виде замкнутых полигонов и первичное разбиение конструкции на треугольники. Для получения первичного разбиения при реализации алгоритма генерации сетки я использовал упрощенный вариант алгоритма step-by-step (см. п. 5) с использованием в качестве точек вершин полигонов исходной геометрии. В этом пункте, при описании алгоритма, будем считать, что первоначальное разбиение уже создано. Более подробно на генерации первоначального разбиения мы остановимся ниже (см. п. 5).

При описании своего алгоритма Рапперт вводит свою терминологию, мы также будем придерживаться этой терминологии.

Элемент по смыслу совпадает с конечным элементом. Это элементарная часть пространства на множество которых мы хотим разбить более сложную область этого пространства. В нашем случае элементом является треугольник.

Узел (node) — точка в которой сходятся грани и соприкасаются несколько элементов. В отличие от вершин элементов узел общий для всех соприкасающихся в одной точке элементов.

Сегмент (segment) — это отрезок, соединяющий две соседние точки лежащие на границе области разбиения, другими словами этот отрезок принадлежит контурам области.

Грань (edge) — это отрезок по которому граничат два соприкасающихся треугольника. Используя предыдущее определение можно сказать, что все множество отрезков, соединяющих вершины элементов, в любой момент работы алгоритма складывается из сегментов и граней.

Включенная точка (encroached point) — любая вершина текущей сетки, которая находится внутри окружности, радиусом которой является любой сегмент.

«**Неправильный треугольник**» (bad triangle) — это треугольник неудовлетворяющий глобальным условиям, которые накладываются на генерируемую сетку.

Вытянутый треугольник (skinny triangle) — треугольник, имеющий одну сторону, намного отличающуюся от других двух. То есть треугольник слишком вытянут вдоль некоторой прямой. На критериях «вытянутости» мы остановимся ниже.

Алгоритм опирается на две базовые процедуры: *a*) разбиение неправильного треугольника с введением нового узла и *b*) разбиение сегмента, принадлежащего границе области разбиения с введением нового узла. При разбиении треугольника происходит следующее:

1. Вычисляются координаты центра окружности, описанной вокруг треугольника, подлежащего разбиению.
2. В эту точку добавляется новый узел.
3. Удаляется треугольник, подлежащий разбиению и прилегающие и добавляются новые, как показано на рисунке 3.

Вторая базовая процедура состоит в следующем:

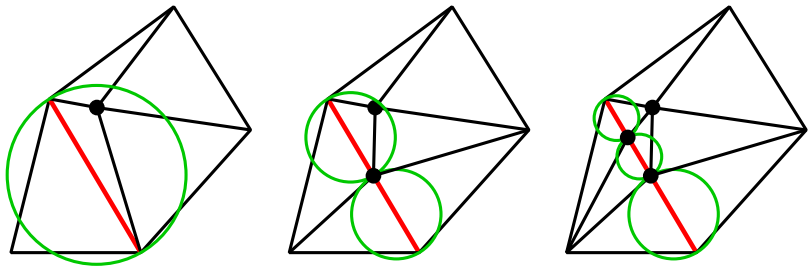

Рис. 2: Разбиение «неправильного» сегмента посередине и добавление двух новых треугольников.

1. Если в окружность, диаметром которой является сегмент, принадлежащий границе области разбиения попадает точка, не принадлежащая этому сегменту, то сегмент делится на две части.

2. Удаляется треугольник, которому принадлежал первоначальный сегмент и добавляются два новых треугольника(см. рис. 2).

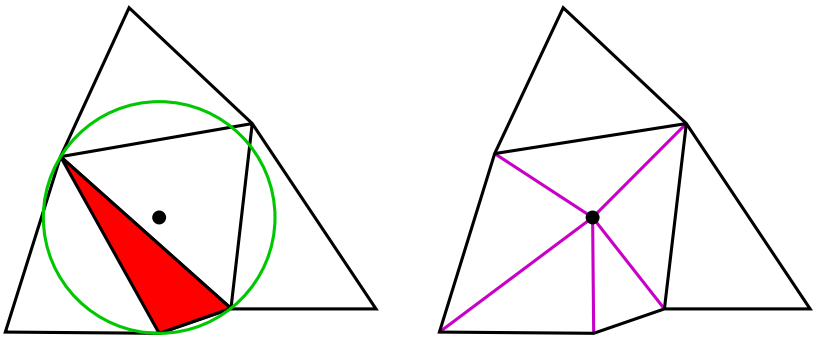

Рис. 3: Разбиение «неправильного» треугольника.

Принцип работы алгоритма состоит в цикле определения «неправильных» треугольников, т.е. треугольников, не удовлетворяющих определенному критерию, и произведение над ним одной из базовых процедур. В оригинальном алгоритме, предложенным Джимом Раппертом [1] используется один критерий — минимальный угол в треугольниках сетки. В каждом треугольнике сетки определяется минимальный угол и, затем, сравниваются минимальные углы для

всех треугольников с целью нахождения минимального угла сетки. Другими словами, минимальный угол в сетке равен минимально возможному углу, образованному двумя сегментами, либо двумя гранями, либо сегментом и гранью, имеющими общий узел.

Для задачи генерации конечно-элементной сетки этого критерия недостаточно, дело в том, что функции, значения которых мы хотим получить в узлах сетки, не линейны, при этом они могут иметь области с большими значениями своих производных (области концентрации напряжений). При больших размерах конечного элемента, велик риск попросту упустить максимальные значения напряжений, поэтому был введен еще один критерий — условие максимальной площади треугольника. Сочетание этих двух ограничений гарантирует, что в сетке углы всех треугольников не менее значения, заданного пользователем, а площади всех треугольников не более другого значения, также задаваемого пользователем.

Математически доказано, что критерий минимального угла автоматически обеспечивает сгущение сетки вблизи мелких подробностей: отверстий, углов, вырезов, т.е. концентраторов напряжений. А критерий максимальной площади является дополнительным, регулирующим размер треугольников. Введение этого критерия было необходимо для адаптации этого алгоритма к методу конечных элементов. Этот дополнительный критерий снижает «эффективность» получаемого разбиения, т.к. введением этого критерия мы ограничиваем площадь треугольников сверху, тем самым увеличивая количество треугольников в получаемой сетке. Но задав минимальную площадь треугольника достаточно большой, можно сделать так, что фактически будет работать только критерий минимального угла.

Все описанные процедуры выполняются в определенной последовательности, они имеют разный приоритет. Общая схема работы алгоритма следующая:

1. Производится поиск включенной (encroached) точки перебором всех точек и сегментов *(только сегментов, но не граней)*.

2. Если такая точка найдена то над сегментом, который включает ее выполняется процедура деления сегмента пополам и производится возврат на шаг 1. Если включенных точек не было найдено, то алгоритм переходит на следующий шаг.

3. Производится поиск треугольника с минимальным углом.

4. Если минимальный угол меньше заданного параметра, то над треугольником, его содержащем производится процедура деления и производится возврат на шаг 1. В противном случае происходит переход на следующий шаг.

5. Производится поиск треугольника максимальной площадью.

6. Если максимальная площадь больше заданного параметра, то над таким треугольником производится процедура деления и производится возврат на шаг 1. В противном случае происходит переход на следующий шаг.

7. Все условия удовлетворены. Конец работы алгоритма

Автор, в своей работе [1], приводит строгие математические доказательства, гарантирующие правильную работу алгоритма при минимальном угле $\alpha < 20^\circ$. Мы не будем приводить здесь эти доказательства, т.к. нашей целью является дать практическое описание алгоритма и некоторые советы по его реализации. Все же рекомендуем обратиться к оригинальной статье Джима Рапперта, всем кто захочет реализовать его алгоритм.

3 Модификация алгоритма Рапперта

В работе для Journal of Algorithms [1] раскрыт только лишь чисто математический аспект проблемы, хотя вероятней всего алгоритм разрабатывался для решения прикладных задач и для генерации конечно-элементной сетки в том числе. К тому же, и математическая сторона была раскрыта не полностью, а только лишь с точки

зрения сходимости и некоторых особенностей. Ряд самых неочевидных подводных камней не поднимается на обсуждение. Видимо это объясняется тем, что это проект NASA и, вероятно, имеет какую-то степень закрытости.

В том варианте, в котором он был предложен автором, алгоритм не обладал хорошей сходимостью и устойчивостью. Опытным путем было установлено, что углы $\alpha > 25^\circ$ были недоступны для него. Для повышения качества работы алгоритма, помимо двух вышеперечисленных процедур была введена еще одна — процедура поворота диагонали. К сожалению автор опустил этот момент и не упомянул о нем, несмотря на то что, ее роль в работе алгоритма очень высока. Эта процедура, на самом деле, неявно присутствует в оригинальном алгоритме Рапперта (см.рис. 2, 3), но все же эту процедуру лучше выделить в отдельную и остановиться на ней подробно. Это упрощает и понимание работы алгоритма, и его программирование. На рисунках видно, что помимо непосредственного добавления новых треугольников, производятся и другие операции. Эти операции есть ничто иное как смена диагонали.

Суть ее заключается в следующем. Два треугольника, имеющие общую грань, образуют четырехугольник, разделенный диагональю. Внутри такого образования диагональ имеет два возможных положения. На рисунке 4 показаны эти положения — отрезки ac и bd. В общей системе триангуляции положение этой диагонали является локальной характеристикой для четырехугольника. Действительно, смена положения, кроме некоторых случаев, обсуждаемых позже, изменит характеристики только этих двух треугольников.

Этой возможностью можно воспользоваться для локального улучшения свойств сетки. Для этого необходимо выяснить еще одно обстоятельство — критерий, который бы давал ответ, улучшится ли локальная характеристика сетки при смене положения диагонали, т.е. дал ответ, надо или не надо производить эту операцию над каждым конкретным четырехугольника.

Вариантов такого критерия можно придумать множество. Но нами было опробовано три варианта: длина диагонали, отношение

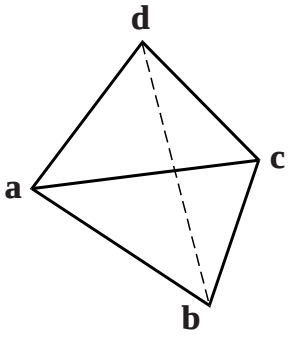

Рис. 4: Два треугольника, имеющие общую грань, образуют четырехугольник, разделенный диагональю. Внутри такого образования диагональ имеет два возможных положения.

площадей до и после смены диагонали и все тот же критерий минимального угла. Все эти критерии эмпирические и ни один из них не исследовался теоретически, хотя они и были тщательно испытаны на практике. Но этот список был ограничен лишь в силу того, что автор не ставил себе целью провести исследовательскую работу, было необходимо лишь создать работающую программу. Поэтому, этот список открыт и может пополняться.

К каждому возможному варианту необходимо предъявлять ряд требований. Во-первых, это адекватность этого критерия, т.е. улучшает ли он свойства сетки на самом деле, и если да, то с какими оговорками. Может случится, что способ хорош внешне, прост, не трудоемок, но работает при ряде ограничений. Эти ограничения и могут свести на нет все преимущества предлагаемого метода. Во-вторых, необходимо учитывать машинную трудоемкость выбираемого метода. И в-третьих, эффективность его будет зависеть еще и от сложности алгоритма, реализующего предлагаемый метод.

Первый, возможно самый очевидный критерий — это длина диагонали(см. рис. 4). Действительно, зрительно, диагональ ac кажется гораздо более удачной диагонали bd для минимального угла двух треугольников и поменяв эту диагональ мы, вероятно, добьемся улучшения свойств сетки. У этого способа, к тому же, есть преимущества: он не трудоемок в вычислительном плане и его просто запрограммировать. Но есть у него и критический недостаток — он

неадекватен. На рисунке 5 показан случай, где использование критерия минимальной длины диагонали может привести к серьезному ухудшению свойств сетки.

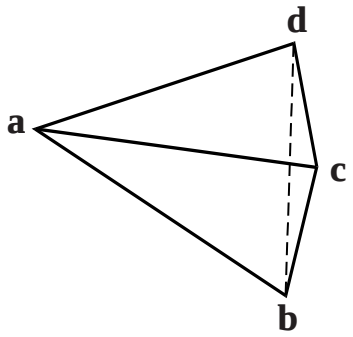

Рис. 5: Случай, в котором использование критерия минимальной длины диагонали, может привести к серьезному ухудшению свойств сетки.

Диагональ *bd* короче диагонали *ac*, поэтому, по критерию минимальной длины диагонали, диагональ *ac* надо поменять на *bd*. Но это приведет к появлению очень тонкого треугольника, а их появление крайне нежелательно, а часто и вовсе становится критическим. Этот случай уже не так просто формализуется, и запрограммировав его, легко убедиться, что это дополнительное условие сильно «утяжеляет» алгоритм, сводя на нет все его преимущества.

Разумеется, что и на практике, в силу своей неадекватности, ничего хорошего этот критерий не показал и от него пришлось отказаться.

Следующий очевидный критерий можно получить сравнивая минимальные углы в двух треугольниках до и после смены диагонали, для предсказания эффективности этого действия (см. рис. 6). Такой критерий является абсолютно адекватным, т.к. он совпадает с основным глобальным критерием сетки — критерием минимального угла и это его существенное преимущество.

Применение такого критерия действительно кажется заманчивым в силу его объективности, но у него есть один специфический недостаток — трудоемкость. Для него необходимо вычислить 6 углов, а по объему операций трудоемкость этого превосходит более

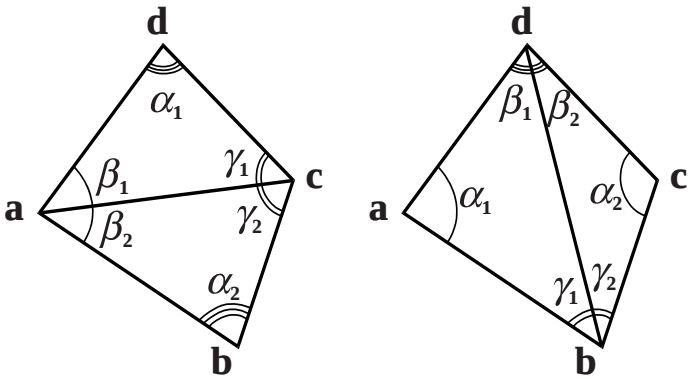

Рис. 6: Очевидный критерий — сравнение минимального угла в кластере, образуемого двумя треугольниками, до и после смены диагонали.

чем в 10 раз вычисление двух длин отрезков. Поэтому было решено попробовать и другое решение.

Действие вычисления площади по трудоемкости сравнимо с нахождением одного угла. Поэтому третьим критерием является сравнение площадей двух треугольников, точнее не сравнение самих площадей, а сравнение с единицей отношения площадей до и после смены диагонали. Сравниваться должна величина отношения площадей до и после смены диагонали. Другими словами, величина этого отношения должна быть меньше 1, если она получается больше 1, то сравнивается ее обратная величина. Более близкое к единице значение отношений означает более равномерное распределение площадей. Этот критерий теоретически не обоснован, он больше интуитивный, эмпирический, но тем не менее, на практике работает довольно хорошо.

Но все же, сравнивая качество работы второго и третьего критерия было решено использовать второй. Из-за своего совпадения с глобальным критерием качества сетки он показывает более стабильную работу всего алгоритма в целом.

Выше было упомянуто о существовании исключения, при котором нарушается локальность изменения положения диагонали внутри четырехугольника. На рисунке 7 показан такой случай.

Как видно, новая диагональ выходит за границы кластера, пересекая грани уже существующих соседей. Из-за этого получа-

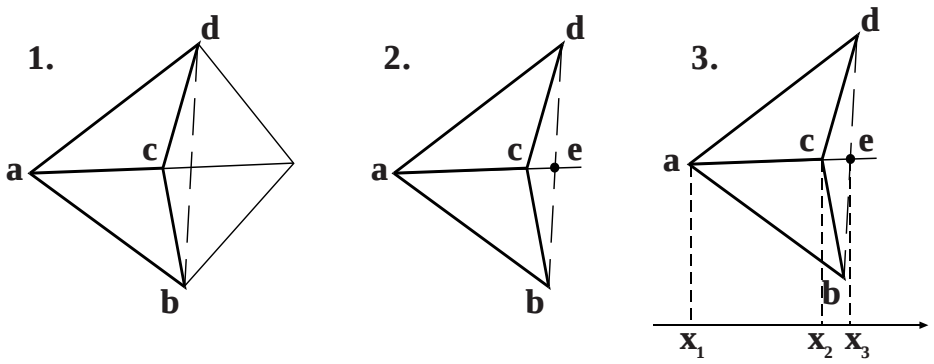

Рис. 7: Случай нарушения локальности при смене диагонали.

ется пересечение площадей с соседними треугольниками, что совсем недопустимо, т.к. вызывает нарушение непрерывности пространства, описываемого треугольниками. Поэтому, в дополнение к основному критерию, необходимо добавить еще один, запрещающий производить смену диагонали в подобных ситуациях. Самыми простыми являются два. Первый показан на рисунке, когда отыскивается точка пересечения прямых, на которых лежат существующая и новая диагональ и проверяется, лежит ли точка пересечения на этих отрезках. Но самым эффективным является второй, когда сравниваются суммы площадей двух треугольников, т.е. площади четырехугольников до и после смены диагоналей. Если суммы совпадают то, новая диагональ не является мнимой. Этот способ особенно эффективен с использованием критерия по площади, т.к. они совместно используют четыре площади треугольников.

Теперь необходимо установить в каком порядке включать эту процедуру. Самым простым решением было проверять циклом каждую грань триангуляции на каждом шаге и, если необходимо, производить смену диагонали. Такой подход и был осуществлен в первой версии рабочего алгоритма, и надежно работал. Но известно, что в треугольной сетке граней больше чем треугольников и узлов в 1,5–1,6, поэтому тотальная проверка всех граней стала бы трудоемкой.

Но по сути тотальной проверки производить не надо т.к. если, например, на предыдущем шаге грань i не нуждалась в смене и,

если изменения в сетке, сделанные в этот шаг, не коснулись треугольников, которым принадлежит грань i, то, очевидно, что и на текущем шаге она не нуждается в смене.

Исходя из этого была применена другая технология — технология распространяющегося «вируса смены диагонали». Суть ее заключается в следующем. После выполнения одной из базовых процедур точно известно, какие треугольники сетки подверглись изменениям, т.е. известна область, которая подверглась изменениям. Грани в этой области могут нуждаться в перемене своего направления и эти грани становятся «зараженными». Они помещаются в очередь для проверки — линейный массив индексов. Каждая грань в очереди проверяется и если проверяемая грань нуждалась в изменении, то все остальные грани двух треугольников, содержащих проверяемую грань также становятся зараженными и помещаются в очередь. Другими словами, меняя диагональ, мы изменяем область, состоящую из этих треугольников. Границами этой области будут остальные четыре грани этих двух треугольников. Каждая такая грань как граница измененной области нуждается в проверке. Если же грань не нуждается в изменениях, то она удаляется из очереди. Такой процесс продолжается до тех пор, пока очередь не станет пустой.

Такой подход существенно ускорил работу алгоритма в целом. Например, в одном контрольном примере, для случая с тотальной проверкой время работы на Celeron 266 (333) составило около 18 секунд, а для второго случая время составило менее 10 секунд — сокращение трудоемкости более чем на 45%.

4 Дополнительные замечания к реализации алгоритма.

Кроме добавления в базовый алгоритм еще одной процедуры, необходимо было усовершенствовать и базовые алгоритмы. Как было сказано выше в базовом алгоритме есть ряд недочетов, которые потребовали своего устроения.

4.1 Правильная геометрия фигуры.

Первый недочет состоял в том, что алгоритм не работал правильно с симметричными или правильными геометрическими формами, которые часто возникают из-за того, что область которую необходимо разбить на элементы является правильной. На рисунке 8 видно, при попытке разбить равнобедренный треугольник *bcd* с прямым углом по базовой процедуре образуются треугольники нулевой площади (см. рис. 8). В данном случае процедура разобьет треугольник на три: *dce*, *bec*, *bde*. Центр описанной окружности треугольника *bcd* находится точно на отрезке *bd*, поэтому треугольник *bde* имеет нулевую площадь. При попытке вычислить координаты центра его описанной окружности происходит деление на ноль.

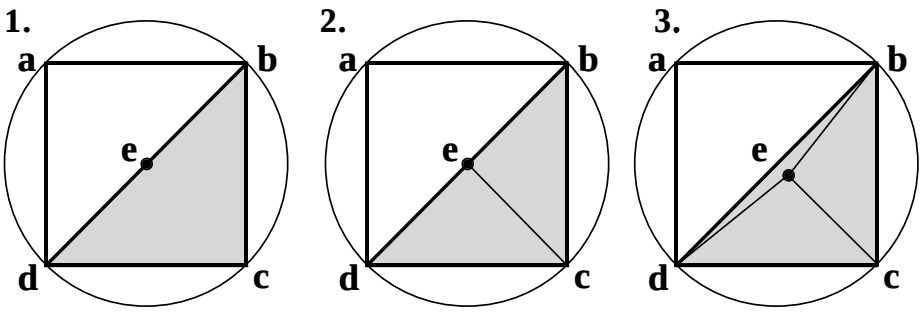

Рис. 8: Правильная геометрия фигуры приводит к генерации треугольников нулевой площади.

Для ликвидации таких случаев, и в первую, и во вторую базовую процедуру были введены небольшие относительные поправки к точным координатам середины отрезка и центра описанной окружности, не нарушающие принципы работы алгоритма. Теперь точки стали добавляться не точно в середины сегментов и центры окружностей, а в точки, расположенные вблизи них. Эти поправки добавляет какую-то степень иррациональности в правильную геометрию, и ликвидирует проблему.

4.2 Очень вытянутые треугольники с удаленным центром.

Второй недочет заключается в том, что не учитывается возможность существования таких треугольников, у которых центр описанной вокруг них окружности находится далеко от самого треугольника. В таком случае, при делении треугольников с наименьшим углом в первую очередь, возникает следующая ситуация (см. рис. 9). При первом разделении такого треугольника добавляется точка в центр описанной вокруг него окружности, но так как точка находится далеко от самого треугольника, то сам треугольник не претерпевает никаких изменений и он по-прежнему остается первым в очереди для разбиения.

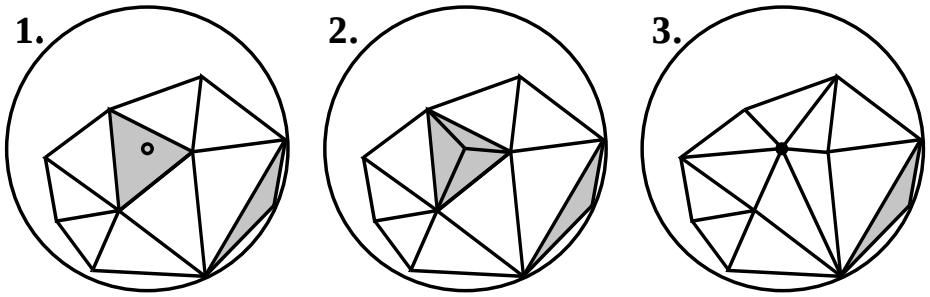

Рис. 9: Вытянутый треугольник, центр описанной окружности которого находится за пределами треугольников, с которыми он граничит.

При следующей попытке разбиения опять вводится новая точка с теми же координатами, что и на предыдущем шаге. В итоге получаются треугольники нулевой площади и грани нулевой длины. Для ликвидации подобных ситуаций была введена проверка на принадлежность вновь вводимой точки к одному из соседних треугольников.

Принадлежность к треугольнику устанавливается следующим способом. Для треугольника, зная вершины 1, 2 и 3 треугольника проверяется с какой стороны от грани 1-2 находится проверяемая точка, далее аналогично для граней 2-3 и 3-1. Если для всех граней точка находится с одной стороны, то точка находится внутри треугольника. Математически это определяется довольно просто. Достаточно записать каноническое уравнение прямой на которой

лежит грань, подставить в него координаты проверяемой точки и

$$ax_1 + by_2 + c > 0 \qquad \text{или} \qquad ax_1 + by_2 + c < 0$$

сравнить полученные знаки для каждой грани. Если знаки одинаковые, то и точка находится с одной стороны от всех отрезков.

4.3 Малые углы между гранями.

Существует еще одна любопытная подробность, которая даже упомянута у Рапперта. При существовании в исходной геометрии острых углов, т.е. если существует малый угол между соседними сегментами, происходит зацикливание на процедуре деления сегмента (см. рис. 10).

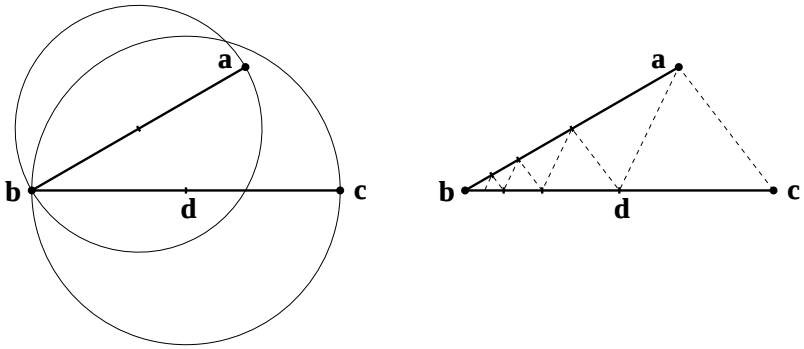

Рис. 10: При появлении острых углов между сегментами, базовый алгоритм Рапперта «зацикливается» на процедуре деления сегмента пополам.

Для ликвидации подобных ситуаций автор алгоритма предложил эвристическое правило. Он предложил в таких случаях делить сегмент не посередине, а на ближайшей точке пересечения этого сегмента и концентрических окружностей с центрами в вершине угла (см. рис. 11). Радиус каждой последующей окружности ровно вдвое больше радиуса предыдущей, а первая окружность берется произвольно, но достаточно малой, например $r_0 = 0.01$. То есть радиус каждой i-ой окружности равен $r_i = r_0 \cdot 2^i$.

Безусловно, эту схему надо применять только там, где это действительно необходимо. Здесь программисту тоже дается свобода

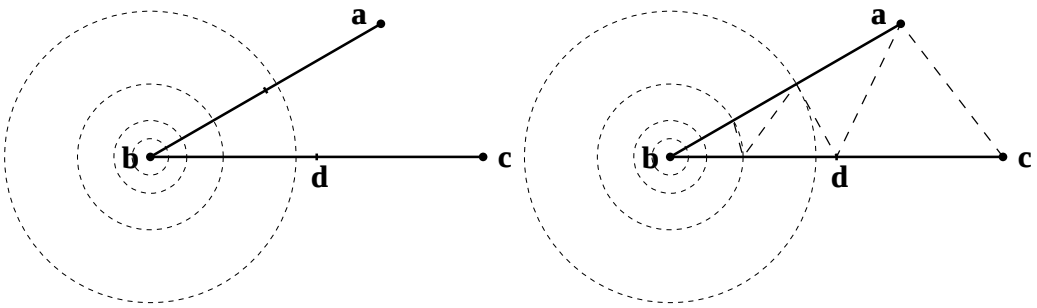

Рис. 11: Способ концентрических окружностей.

в выборе схемы проверки такой необходимости. Можно все же предложить такую схему: сначала проверяется, является ли угол достаточно малым; затем проверяется площадь треугольника, подлежащего разбиению. Если площадь меньше площади, задаваемой как параметр для алгоритма, т.е. критерия максимальной площади, умноженного на какой-то коэффициент, который можно подобрать $(0 < k \leq 1)$, то тогда необходимо использовать способ концентрических окружностей.

Если в нашей конструкции к вершине такого угла приложена сосредоточенная сила, то в окрестности этой точки возникают резкие изменения напряжений и деформаций. Задав коэффициент k достаточно малым, мы добьемся более густого разбиения в таких углах.

Все эти усовершенствования коснулись самого алгоритма улучшения сетки. Они объективно позволили улучшить стабильность работы и качество генерируемой сетки, но в нем все же есть небольшие недостатки. Но они не критичны, самым заметным из них является появление неоправданных сгущений сетки. Такие сгущения появляются не всегда, но если и появляются то их можно ликвидировать изменив параметры (минимальный угол или максимальная площадь) и сгенерировав заново сетку. А при разумных установках этих параметров, алгоритм всегда сходится, т.е. достигает этих параметров. Безусловно при более тщательном его исследовании можно добиться еще лучших результатов.

5 Алгоритм построения первичного разбиения.

Ранее говорилось об алгоритме генерации сетки, но по сути все, о чем говорилось выше, является не алгоритмом генерации сетки, а лишь алгоритмом ее улучшения (Delaunay Refinement Algorithm). Для него входными данными является уже готовое разбиение. Первоначальное разбиение полигональной фигуры производится другим алгоритмом. Входными данными для него являются массивы точек (координаты, нагрузка, закрепление) и граней полигонов, в терминологии Рапперта [1] — сегментов, соединяющих эти точки. Выходные данные — это массив вершин без изменений, массив граней треугольников и массив собственно треугольников, заполняющих весь объем конструкции.

Схематично этот алгоритм можно описать так:

1. Выбирается начальный сегмент (любой) на внешнем полигоне исходной геометрии. (см. рис. 12)
2. Из всех вершин выбирается такая, из которой лучи, соединяющие эту вершину и концы текущего сегмента образуют максимальный угол.
3. Добавляется полученный таким образом треугольник и текущим сегментом становится одна из созданных сторон этого треугольника.
4. Если существует подходящий текущий сегмент, то на шаг 2.

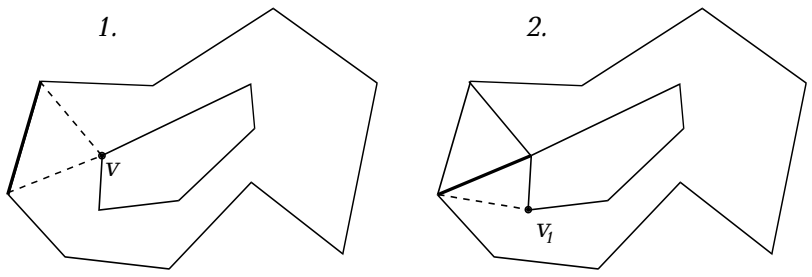

Рис. 12: Принцип генерации начального разбиения.

Здесь используется технология, похожая на «вирус смены диагонали». Первым действием является нахождение первого «правильного» треугольника. Таким образом появляются две первых грани (не сегмент, сегментом называется такая грань, которая принадлежит к границам первоначальных полигональных областей — очертаний). Далее треугольники образуются путем правильного добавления к грани еще двух граней соединяющих концы этой грани и определенную точку, принадлежащую сегменту (границе области), добавляя в сетку еще две грани, и т.д.

При выборе третьей вершины для формирования треугольника возникает множество вопросов. Во-первых, необходим критерий, позволяющий легко избежать пересечений вновь создаваемых граней с уже существующими. Во-вторых, как ограничить треугольники от распространения за пределы полигональной области. В-третьих, как правильно сгенерировать первый треугольник. И в-четвертых, как учесть то обстоятельство, что в конструкции присутствуют как отверстия так и заполненные области описываемые одинаково.

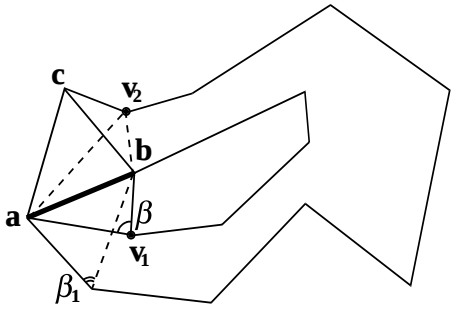

Рис. 13: Выбор третьей точки для построения нового треугольника.

Критерием для оптимального выбора третьей точки треугольника является угол, под которым видна грань из этой точки, т.е. угол образуемый двумя лучами, выходящими из выбираемой точки и проходящими через концы грани. Для каждой грани выбирается такая точка, для которой этот угол максимален. Такой выбор критерия позволил сэкономить время на том, что нет необходимости проверять вновь создаваемые грани на пересечение с уже существующи-

ми. Для того, чтобы сразу исключить проверки, подобные проверке точки, перед перебором всех точек устанавливается с какой стороны лежит третья точка треугольника, которому принадлежит грань *ab* (точка *c*, см. рис. 13). Каждая точка проверяется на то, с какой стороны от грани *ab* она находится. Другими словами, эта проверка исключает пересечение площадей с уже существующими треугольниками. Я не находил точного обоснования этой особенности, но, тем не менее, такой подход надежно работает.

Чтобы треугольники не распространялись за пределы конструкции, вводится простое правило: треугольники не формируются от сегментов а только от граней уже существующих треугольников. На грани также накладывается ограничение, треугольники формируются только от граней, которые принадлежат только одному треугольнику. Эти правила нарушаются только в одном случае — при генерации первого треугольника.

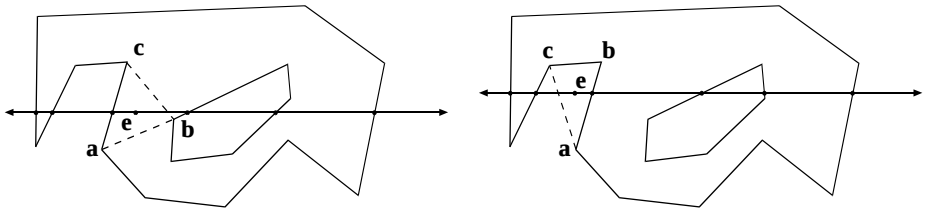

Рис. 14: Определение принадлежности вновь создаваемого треугольника контуру.

Первый треугольник генерируется от любого сегмента внешнего полигона, а третья точка выбирается также как и для грани. Для первой проверяемой точки устанавливается принадлежность центра треугольника, образуемого этой точкой и сегментом, телу конструкции (см. рис. 14), т.е. лежит ли треугольник внутри контура. Для этого, через этот центр проводится прямая, параллельная оси *X* и находятся все ее точки пересечения с сегментами. Далее эти точки сортируются по мере возрастания координаты *X* и формируются интервалы. Каждый нечетный интервал принадлежит контуру.

Все вышеперечисленные процедуры и формируют алгоритм генерации сетки.

6 Выводы

Описанный алгоритм является завершенным алгоритмом генерации треугольной сетки для плоских полигональных фигур. Этот алгоритм дает теоретически подтвержденные гарантии качества получаемой с его помощью сетки. Прежде всего это касается ограничения сверху «вытянутости» треугольников, т.е. отношения длин их максимальной и минимальной грани.

Приведенный алгоритм адаптирован для построения конечно-элементных сеток, но благодаря своей простоте, может быть легко приспособлен для других задач.

Для него можно выделить несколько путей дальнейшего развития. Прежде всего, это расширение алгоритма для работы в трехмерном пространстве и генерации трехмерных сеток для многогранников. Алгоритм легко может быть усовершенствован для повторной триангуляции областей уже имеющейся сетки, где была получена большая погрешность при расчете, с целью сгустить сетку в такой области.

Список литературы

[1] Jim Ruppert: *A Delaunay Refinement Algorithm for Quality 2-Dimentional Mesh Generation*, NASA Ames Research Center, Submission to Journal of Algorithms, 1994.