Санкт-Петербургский государственный университет
Математическое обеспечение и администрирование информационных систем
Кафедра информатики

Слепцова Стелла Васильевна

Подсистема для разбиения сеток для программного комплекса ELCUT

Бакалаврская работа

Научный руководитель:
ст. преп. Симуни М. Л.

Рецензент:
ведущий программист ООО "Тор" Блюдзе М. Ю.

Санкт-Петербург
2017

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems

Computer science department

Stella Sleptsova

Subsystem for mesh partitioning for ELCUT software package

Bachelor's Thesis

Scientific supervisor:
senior lecturer Mikhail Simuni

Reviewer:
senior software engineer OOO "Tor" Mikhail Bliudze

Saint-Petersburg
2017

Содержание

Введение

Наиболее распространенным методом решений задач прикладной физики является метод конечных элементов [1]. Метод конечных элементов основан на замене дифференциальных и интегральных уравнений системой алгебраических уравнений.

Суть метода конечных элементов заключается в том, что данная область аппроксимируется сеткой при помощи разбиения области на несколько маленьких подобластей (например, на треугольники для двумерных областей или на тетраэдры для трехмерных областей), которые называются конечными элементами. Этот процесс называется дискретизацией.

Для более быстрых вычислений метода конечных элементов, полученную сетку надо разбить на несколько подобластей, называемых доменами. К этим доменам предъявлен ряд требований, например, приблизительно одинаковый размер доменов. Более подробно критерии описаны в разделе 2.1.

Задача построения разбиения, удовлетворяющего требованиям, является достаточно сложной. Существует много алгоритмов построения разбиения, а также реализующих такие алгоритмы коммерческих продуктов. Подробно они описаны в разделах 2.2 и 1 соответственно.

Данная работа была выполнена, как часть разработки новой версии программного комплекса ELCUT [2]. ELCUT — это среда для решений инженерных задач (подробнее в разделе 3.1) В своей работе ELCUT использует метод конечных элементов, следовательно, ему необходима подсистема разбиения сеток. Такая система в текущей версии ELCUT уже существует, однако построенные с ее помощью разбиения не всегда получаются высокого качества. Таким образом, возникла необходимость разработки новой версии подсистемы разбиения.

Постановка задачи

Целью данной работы является разработка подсистемы разбиения двумерных и трехмерных сеток для программного комплекса ELCUT. Для достижения поставленной цели были сформулированы следующие задачи.

1. Исследовать существующие алгоритмы разбиения сетки.
2. Выбрать подходящий класс алгоритмов.
3. Спроектировать и разработать подсистемы разбиения для двумерных и трехмерных сеток.
4. Проверить реализованные алгоритмы на различных расчетных сетках формата ELCUT.

1. Обзор пакетов разбиения сетки

В этом разделе рассматриваются наиболее распространенные программные пакеты для декомпозиции сеток и идеи, на которых они основаны.

Один из первых общедоступных программных пакетов под названием Chaco разработан Хендриксоном (Hendrickson) и Леландом (Leland) в начале 1990-х годов в Сандийской национальной лаборатории (Sandia National Labs) [3]. Этот пакет содержит реализации простых геометрических, инерционных алгоритмов и алгоритма Кернигана-Лина, которые описаны в разделе 2.2. Кроме того, Chaco, как и большинство программных пакетов, реализует многоуровневый подход, основные алгоритмы локального поиска и спектральные методы, которые в данной работе не затрагивались. Chaco доступен для скачивания по лицензии с открытым исходным кодом. Последующие программные пакеты, такие как METIS, Scotch и Jostle, переняли и усовершенствовали многие идеи, впервые использованные в Chaco.

Пожалуй, самым распространенным доступным программным пакетом является METIS [4]. Программный пакет METIS представляет собой набор последовательных программ для разбиения больших нерегулярных графов и сеток. METIS был разработан на кафедре компьютерных наук и инженерии (Department of Computer Science and Engineering) в университете Миннесоты (University of Minnesota) и имеет открытый исходный код, который непосредственно можно скачать с [4]. Алгоритмы, реализованные в METIS, основаны на многоуровневом методе разбиения графов, основная идея которого рассмотрена в разделе 2.2.3. Многие геометрические, комбинаторные, диффузионные и генетические алгоритмы входят в METIS, как подпрограммы для многоуровневого подхода. Среди семейств программ пакета METIS можно выделить такие инструменты, как kMETIS и hMETIS. kMETIS нацелен на большую скорость разбиения, а hMETIS, который применяется к гиперграфам, нацелен на оптимальный размер разреза. METIS предлагает набор программ, вызываемых из командной строки, а также программный интерфейс приложения (API), который может использоваться для различных программ на языках C/C++ и Fortran.

ParMETIS — библиотека, содержащая распараллеленные алгоритмы из

пакета METIS. Алгоритмы для декомпозиции графов в ParMETIS основаны на параллельном многоуровневом разбиении графа. Есть много статей, которые показывают временное преимущество пакета ParMETIS перед пакетом METIS. Например, в эксперименте из [5] показано, что общее время вычислений, которые ParMETIS проводит с 2 процессорами, имеет очевидные преимущества перед METIS, эффективность которых достигает $76,5\%$.

Также необходимо отметить такие пакеты, как Scotch и Jostle. Scotch — фреймворк для декомпозиции графа, который использует рекурсивное многоуровневое деление пополам и включает в себя, как последовательные и параллельные методы разбиения. Jostle — это программный пакет, предназначенный для разбиения неструктурированных сеток, основанный на многоуровневом алгоритме разбиения. Результаты экспериментов, сравнивающие различные программные пакеты декомпозиции графа, описаны в [6].

Рассматривался вопрос использования вышеупомянутых программных пакетов в систему ELCUT, однако они либо являются достаточно дорогими, либо могут быть применены только для образовательных целей. То есть, в коммерческих продуктах их использовать нельзя. Поэтому было принято решение самостоятельно реализовать подсистему для разбиения сеток для системы ELCUT.

2. Исследование алгоритмов разбиения сетки

2.1. Постановка задачи разбиения сетки

Рассмотрим более подробно задачу разбиения сетки. В качестве входных данных рассматриваются области, представленные в виде расчетных сеток. Расчетная сетка - это тройка (V, E, F), где V - множество вершин (узлов), E - множество ребер, F - множество граней. Расчетная сетка моделирует данную область, определяя форму объекта гранями. Подобласти, на которые разбивается данный объект, называются конечными элементами, из них и состоит расчетная сетка. Для представления данного объекта в виде расчетной сетки используется процесс триангуляции. Триангуляция заключается в разбиении геометрической области на симплексы: двумерные области разбиваются на треугольники, а трехмерные - на тетраэдры.

Задача разбиения сетки состоит в следующем. Необходимо разбить расчетную сетку на подобласти, называемые доменами, вдоль ребер так, чтобы количество конечных элементов в каждом домене не превосходило данного числа, задаваемого пользователем. При этом требуется учитывать следующие критерии оптимальности:

1. Количество конечных элементов в доменах должно быть приблизительно равным. Другими словами, необходимо минимизировать дисбаланс разбиения, где дисбалансом называется максимальное отклонение количества конечных элементов в доменах от среднего числа элементов всех доменов.

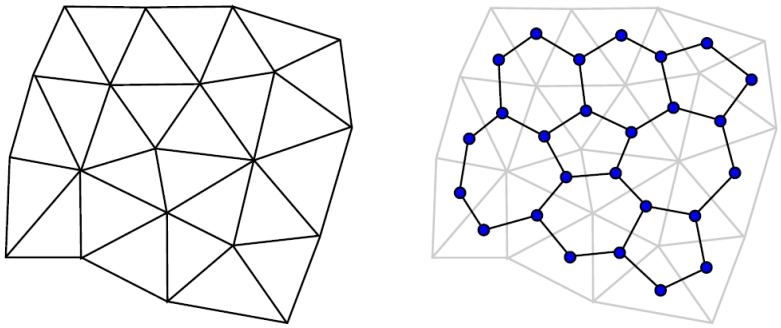
2. Минимизировать количество ребер, вдоль которых разбивается сетка.

Достичь оптимального разбиения, удовлетворяющего обоим критериям практически невозможно. Какой критерий является наиболее приоритетным, зависит от конкретной поставленной задачи и решается в разных случаях по-разному.

2.2. Алгоритмы разбиения сетки

В большинстве случаев задача разбиения расчетной сетки может быть реализована посредством декомпозиции графа. В отличие от задачи разбиения сетки в задаче декомпозиции графа веса имеют и ребра и вершины, при этом граф разбивается не вдоль ребер, из-за чего образуются пересекающиеся по ребрам домены, а на непересекающиеся подграфы. В этом случае, оптимальным разбиением считается разбиение с минимальным суммарным весом разрезанных ребер, которые соединяют подграфы, а также со сбалансированным весом вершин в подграфах.

Для того, чтобы свести задачу разбиения сетки к задаче разбиения графа, данную сетку аппроксимируют дуальным (двойственным) графом, так как расчетная сетка представляет собой планарный граф. Дуальным графом (рис. 1) планарного графа G называют граф G', в котором вершины G' соответствуют граням графа G, а ребра между вершинами G' существуют, только если соответствующие грани соединены общим ребром.



а) б)

Рис. 1: Построение из сетки (а) дуальный граф (б) (рис. из [7])

Таким образом, многие из методов, рассмотренных в данном обзоре, были сформулированы для задачи декомпозиции графа. Доказано, что задача разбиения графа является NP-трудной задачей [7], поэтому все известные алгоритмы основаны на эвристических методах.

Один из видов классификаций алгоритмов декомпозиции графов основывается на том, какие характеристики в них учитываются.

Алгоритмы, которые берут в расчет только номера вершин, являются

самыми простыми. К таким алгоритмам относятся алгоритмы рассеивания (scattered method), линейного распределения вершин (linear method) и случайного распределения (random method) [8,9].

Алгоритм линейного распределения заключается в разбиении диапазона номеров вершин на интервалы, где каждый интервал представляет собой один домен. В некоторых случаях вершины графа тесно сгруппированы в списке вершин, тогда данный метод может успешно получить домен с низким количеством разрезанных ребер.

В алгоритме рассеивания каждый домен также нумеруется. Далее вершины распределяются на различные части по модулю к номеру домена. Подобно линейному алгоритму, данный метод также может создавать хорошие разбиения для конкретных типов графов.

Случайным метод разбиения случайным образом распределяет вершины по различным доменам, то есть следующий домен выбирается случайно среди тех частей, где количество вершин меньше.

Преимуществом рассмотренных алгоритмов является быстрота выполнения. Но эти алгоритмы не используют инцидентность вершин, из-за чего количество разрезанных ребер получается большим. Поэтому такие алгоритмы применяются только для предварительного разбиения.

2.2.1. Геометрические алгоритмы

Более сложными алгоритмами разбиения графа являются алгоритмы, учитывающие геометрические сведения о графе, то есть основанные на координатой информации об узлах графа. Другими словами, геометрические методы применимы, если для узлов сетки существует координатная информация. Как правило, геометрические подходы очень быстры. Однако они склонны вычислять разбиения более низкого качества, чем методы, учитывающие связность узлов сетки.

Метод рекурсивной координатной бисекции

Наиболее простым примером геометрического подхода является метод, известный как метод рекурсивной координатной бисекции или рекурсивное

ортогональное деление пополам [7, 10].

В этом алгоритме вершины сортируются в соответствии с их координатами на каждой координатной оси. Далее для каждой оси координат рассматривается гиперплоскость, которая ей перпендикулярна и расположена так, что проходит через точку, являющейся средним значением отсортированных вершин на выбранной оси. По построению, каждая из этих гиперплоскостей будет делить граф пополам. Затем выбирается одна из гиперплоскостей, которая отсекает наименьшее количество ребер. Для полученных двух подграфов рекурсивно применяется вышеописанная методика. Пример выбора гиперплоскости для двумерного графа показан на рис. 2.

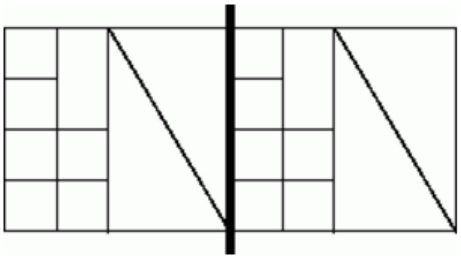


Рис. 2: Пример разбиения сетки методом координатной бисекции, где прямая, по которой делится граф, показана жирной линией (рис. из [10])

Таким образом, данный метод разбивает граф вдоль линии, которая перпендикулярна одной из координатных осей. Если повернуть граф из рис. 2 под острым углом, то разбиение будет менее оптимальным.

Метод рекурсивной инерциальной бисекции

В отличие от метода координатной бисекции, алгоритм инерциальной бисекции вместо выбора гиперплоскости, ортогональной некоторым фиксированным координатным осям, выбирает ось, вдоль которой область имеет наибольшую протяженность (рис. 3). Это позволяет свести к минимуму область сечения сетки и, следовательно, размер разреза. Математически это означает, что выбирается линия таким образом, чтобы сумма квадратов расстояния сетки до линии была минимизирована. Физическая интерпретация состоит в том, что эта линия является осью минимальной враща-

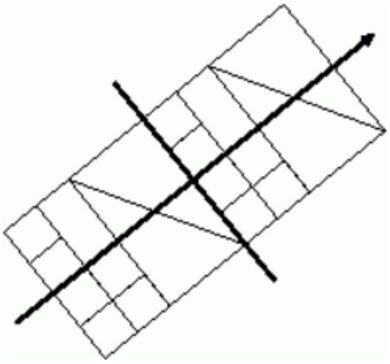


Рис. 3: Пример разбиения сетки методом инерциальной бисекции, где осью инерции является прямая со стрелкой (рис. из [10])

тельной инерции, откуда и следует ее название - ось инерции. Гиперплоскость бисекции, которая перпендикулярна построенной инерциальной оси, вычисляется так же, как и в методе координатной бисекции.

Метод с использованием кривой Гильберта

Идея метода состоит в преобразовании сетки в одномерный список [11]. Подходов, основанных на таком принципе, достаточно много. Одним из самых распространенных методов является метод заполнения вдоль фрактальной кривой (space-filling curve), которая проходит через центры конечных элементов сетки. На практике используют кривые Пеано и их частный случай - кривая Гильберта.

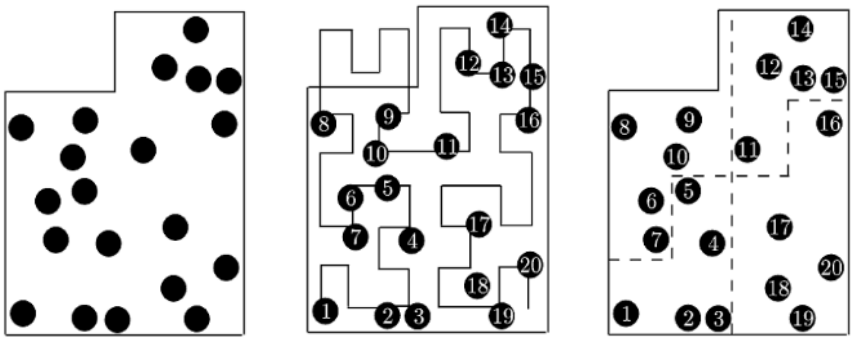
Кривая Гильберта - это непрерывная фрактальная кривая, которая полностью заполняет некоторый гиперпараллелепипед [11]. Кривая Гильберта для двумерной сетки строится следующим образом:

- сперва строится наименьший по площади прямоугольник (гиперпараллелепипед), содержащий все вершины сетки.
- строится кривая Гильберта первого порядка. То есть, сначала прямоугольник разбивается на 4 одинаковых блока, после чего линия, образующая участок кривой Гильберта, заполняет 4 ячейки и имеет форму буквы U.

- Далее каждый последующий уровень кривой Гильберта делит каждую из ячеек прямоугольника на 4 блока, для которых аналогично строится кривая Гильберта первого порядка.

- Затем, используя отображение двумерной области на одномерный список ячеек, проводится декомпозиция области.

Пример декомпозиции двумерной расчетной сетки на 4 домена при помощи кривой Гильберта показан на рис.4. Для трехмерных сеток правила построения кривых сохраняются, но добавляется третья размерность, а также поворот линии, которая формирует кривую.



а) б) в)

Рис. 4: Декомпозиция сетки (а) с использованием кривой Гильберта (б) на 4 домена (в) (рис. из [11])

Качество декомпозиции с использованием кривых Гильберта обычно хуже , чем при использовании других геометрических подходов. Однако благодаря своей быстроте построения разбиения, этот метод присутствует во многих пакетах декомпозиции графов.

2.2.2. Комбинаторные методы

При вычислении разбиения, геометрические методы пытаются сгруппировать вершины, которые расположены рядом друг с другом, не учитывая их связь по ребрам [12]. В отличие от них, комбинаторные методы группируют вершины, учитывая их связи по ребрам, независимо от того, находятся ли эти вершины близко друг к другу в пространстве. Другими словами, комбинаторные подходы вычисляют разбиение только на основе

информации смежности графа. Они не рассматривают координаты вершин. По этой причине, полученные разбиения имеют более низкие граничные показатели. Однако комбинаторные методы медленнее, чем методы геометрического разделения и, как правило, их трудно распараллеливать.

Алгоритм Кернигана-Лина

В начале работы данного алгоритма [12] предполагается, что граф уже имеет предварительное разбиение, которое улучшается на каждом шаге итерации. Улучшение заключается в обмене вершинами между полученными частями разбиения. Перемещение основано на понятии выигрыша, которое вычисляется для каждой перемещаемой вершины. Выигрыш от перемещения рассчитывается, как сумма весов ребер, инцидентными вершинами которого являются рассматриваемая вершина и вершины из другого второго подмножества минус сумма весов ребер, в котором одна из инцидентных вершин, отличная от данной, является вершиной из рассматриваемого подмножества.

То есть, разработанный Керниганом (Kernighan) и Лином (Lin), алгоритм стремится улучшить исходное (возможно, случайное) разбиение графа, перемещая вершины из одного подмножества в другое с целью уменьшения количества разрезанных ребер. Общая работа алгоритма выглядит следующим образом:

1. Выбираем начальное разбиение графа.
2. В качестве текущего разбиения выбираем лучшее.
3. Вычисляем выигрыши для каждой вершины.
    3.1. Выбираем пару с наибольшим выигрышем для перемещения, учитывая при этом критерий балансировки.
    3.2. Обмениваем выбранные вершины и отмечаем их, как перемещенные.
    3.3. Пересчитываем выигрыши.
    3.4. Запоминаем лучшее разбиение.

4. До тех пор, пока все вершины не помечены, выполняем шаги 3.1-3.4.

5. Со всех вершин снимаем пометки и переходим к шагу 2.

6. Останавливаем алгоритм при достижении критерии останова. Они могут быть разными. Алгоритм останавливается, например, если лучшее разбиение не может улучшить текущее.

Одним из основных недостатков данного алгоритма является его время работы. Наихудшее время выполнения одной итерации составляет $O(n^2 \log n)$. Однако существует много вариаций алгоритма Кернигана-Лина с лучшими временными показателями, с которыми можно ознакомиться в источниках [8, 12].

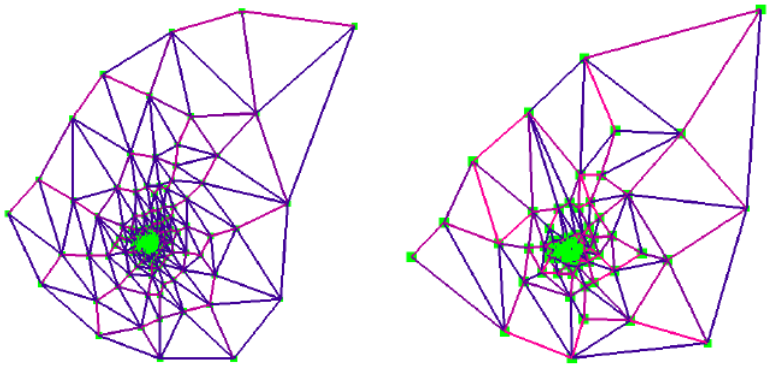
2.2.3. Многоуровневые иерархические методы

В иерархических алгоритмах применяются различные методы [8]. Но можно выделить три основных этапа всех алгоритмов:

1. Последовательное огрубление графа (рис. 5).

2. Разделение самого последнего огрубленного графа.

3. Восстановление графа с отображением разбиения на каждом шаге.



а) б)

Рис. 5: Исходный граф (а) и результат его первого огрубления (б) (рис. из [8])

Огрубление графа осуществляется последовательным уменьшением числа вершин. В основе этого этапа лежит понятие стягивания ребра: две смежные вершины стягиваемого ребра заменяются одной вершиной, вес которой равен сумме весов заменяемых вершин. Для разбиения полученного огрубленного графа используются различные методы декомпозиции. В частности, на этом этапе хорошо себя зарекомендовали геометрические алгоритмы (при существовании координатной информации).

Поскольку каждая вершина огрубленного графа является объединением двух вершин предыдущего огрубления, то веса вершин и разрезанных ребер, полученных доменов, сохраняются. Для нахождения оптимального разбиения на этапе восстановления графа, обычно применяются различные упрощенные методы декомпозиции, которые основываются на предыдущем разбиении.

3. Проектирование и реализация

3.1. Требования к подсистеме разбиения сеток в системе ELCUT

ELCUT представляет собой набор программ, предназначенный для инженерного моделирования различных физических задач методом конечных элементов. Требования к подсистеме разбиения в ELCUT состоят в следующем. Подсистема должна принимать на вход двумерную или трехмерную триангулированную расчетную сетку, а также значение, равное максимальному количеству конечных элементов в каждом домене. На выход подсистема должна вернуть информацию о полученных доменах. В настоящее время программа использует текстовые файлы для получения входной информации и для вывода выходной информации. Однако в дальнейшем предполагается реализация доступа непосредственно к структурам данных системы ELCUT. Также необходимо отметить, в текущей версии предполагается, что сбалансированность является более важным критерием, чем оптимальность разреза, за исключением тех случаев, когда обрабатываемая сетка состоит из несвязных компонент. Такой выбор основан на экспериментах с различными сетками, которые показали, что при данном подходе разбиение получается достаточно хорошим с точки зрения скорости работы метода конечных элементов. Учитывая именно эти требования, выбирается алгоритм для подсистемы разбиения сеток для программного комплекса ELCUT.

3.2. Выбор алгоритма и его основные этапы

Выбор подходящего класса алгоритмов основывался на выше приведенных исследованиях, а также на требованиях системы ELCUT. Выбор пал на геометрические алгоритмы по следующим причинам:

- Входные данные, представляющие собой триангулированные расчетные сетки, содержат координатную информацию узлов, что позволяет рассмотреть в качестве подходящего класса алгоритмов геометрические подходы.

- Системе ELCUT важна быстрота вычисления разбиения, а также экономное использование памяти, что и являются достоинствами геометрических алгоритмов.

- Кроме этого, геометрические методы отличаются от других алгоритмов меньшим дисбалансом. Поэтому они в основном используются, когда сбалансированность разбиения более важна, что и требуется системе ELCUT.

Среди различных геометрических алгоритмов, был выбран алгоритм инерциальной бисекции, так как на основании проделанного выше исследования можно сделать вывод, что данный метод более прост в реализации, и, в тоже время, полученное разбиение более оптимально, в отличие от других алгоритмов.

В реализованном алгоритме можно выделить следующие этапы, которые будут подробно разобраны ниже:

1. Вычисляется направление наибольшей протяженности сетки, которая для двумерной сетки носит название оси инерции.

2. Далее проводится поиск "центра" сетки.

3. Строится прямая и плоскость для двумерных и трехмерных сеток соответственно, проходящая через "центр" и перпендикулярная оси инерции.

4. Далее сетка разбивается на две подсетки, для которых рекурсивно применяются предыдущие шаги до тех пор, пока не выполнится критерий останова. Однако, если одна или обе подсетки разбиваются на несвязные сетки, то каждая компонента связности представляет собой отдельную подсетку. Подробно разбиение на связные компоненты описано в разделе 3.6.1.

3.3. Поиск направления наибольшей протяженности

Для поиска момента инерции применяется используемая в статистике регрессионная модель зависимости некоторой переменной от других пере-

менных. Для двумерных сеток была выбрана парная линейная регрессия, которая строит линейную зависимость между двумя переменными. А для трехмерных сеток используется метод главных компонент.

3.3.1. Метод линейной регрессии

Каждый узел сетки формата ELCUT представляется в виде координат точки в двумерной декартовой системе координат. Рассматривая их, как множество пар $(x_i, y_i), i = 1, .., n,$ где $n -$ количество всех точек сетки, построим одностороннюю стохастическую зависимость переменной y от одной переменной x: $y = f(x)$, которая для парной линейной регрессии имеет вид:

$$y = a + bx. \tag{1}$$

Нахождение коэффициентов a и b проводится с помощью метода наименьших квадратов. Он заключается в поиске такой прямой, у которой сумма квадратов расстояния от сетки до прямой минимизирована. Коэффициенты вычисляются по следующим формулам, которые взяты из источника по регрессионному анализу [13]:

$$a = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} (x_i)^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n \sum_{i=1}^{n} (x_i)^2 - \left(\sum_{i=1}^{n} x_i \right)^2} \tag{2}$$

$$b = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} (x_i)^2 - \left(\sum_{i=1}^{n} x_i \right)^2} \tag{3}$$

Таким образом, поиск оси инерции заключается в поиске коэффициентов a и b, время вычисления которых линейно зависит от количества узлов сетки.

До этого момента мы искали зависимость y от x, то есть рассматривали переменную y как зависимую, а x - как объясняющую. Однако во многих случаях, параметр y может не зависеть от x. Сетка, характеризующая данное явление, показана на рис. 6. Прямая вида $x = const$ не выражается по формуле (1). Для такого специфического случая, проводится поиск

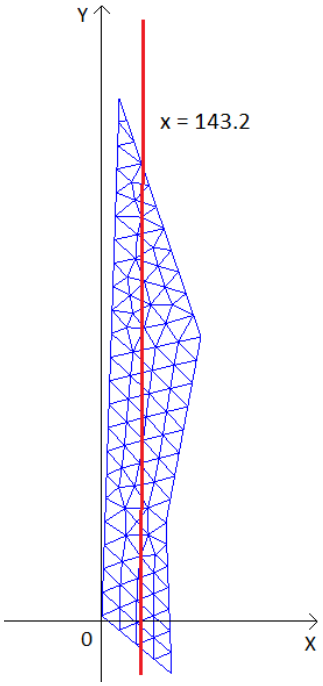


Рис. 6: Вспомогательный вывод программы: для данной сетки направление наибольшей протяженности лучше всего выражается через уравнение $x = 143.2$

сопряженной регрессионной прямой. То есть находится прямая, которая выражается по следующей формуле:

$$x = a' + b'y. \tag{4}$$

Коэффициенты a' и b' находятся аналогично коэффициентам a и b по формулам (2) и (3).

Для того, чтобы выбрать по какой формуле находить момент инерции воспользуемся средней ошибкой аппроксимации, которая показывает качество уравнения регрессии. То есть, для каждого уравнения, вычисленного по формулам (1) и (4) найдем среднюю ошибку аппроксимации. Она представляет собой среднее отклонение фактического значения от теоретического. То есть, чем меньше средняя ошибка аппроксимации, тем лучше качество построенной модели.

3.3.2. Метод главных компонент

Метод главных компонент представляет собой процедуру понижения размерности данных, который используется для сокращения большого на-

бора переменных до другого небольшого набора, по-прежнему содержащего большую часть информации [14]. Метод главных компонент строит новые координатные оси, называемые главными компонентами. Первая главная компонента будет растягиваться в направлении, вдоль которой выборочная дисперсия максимальна (рис. 7). Вторая главная компонента ор-

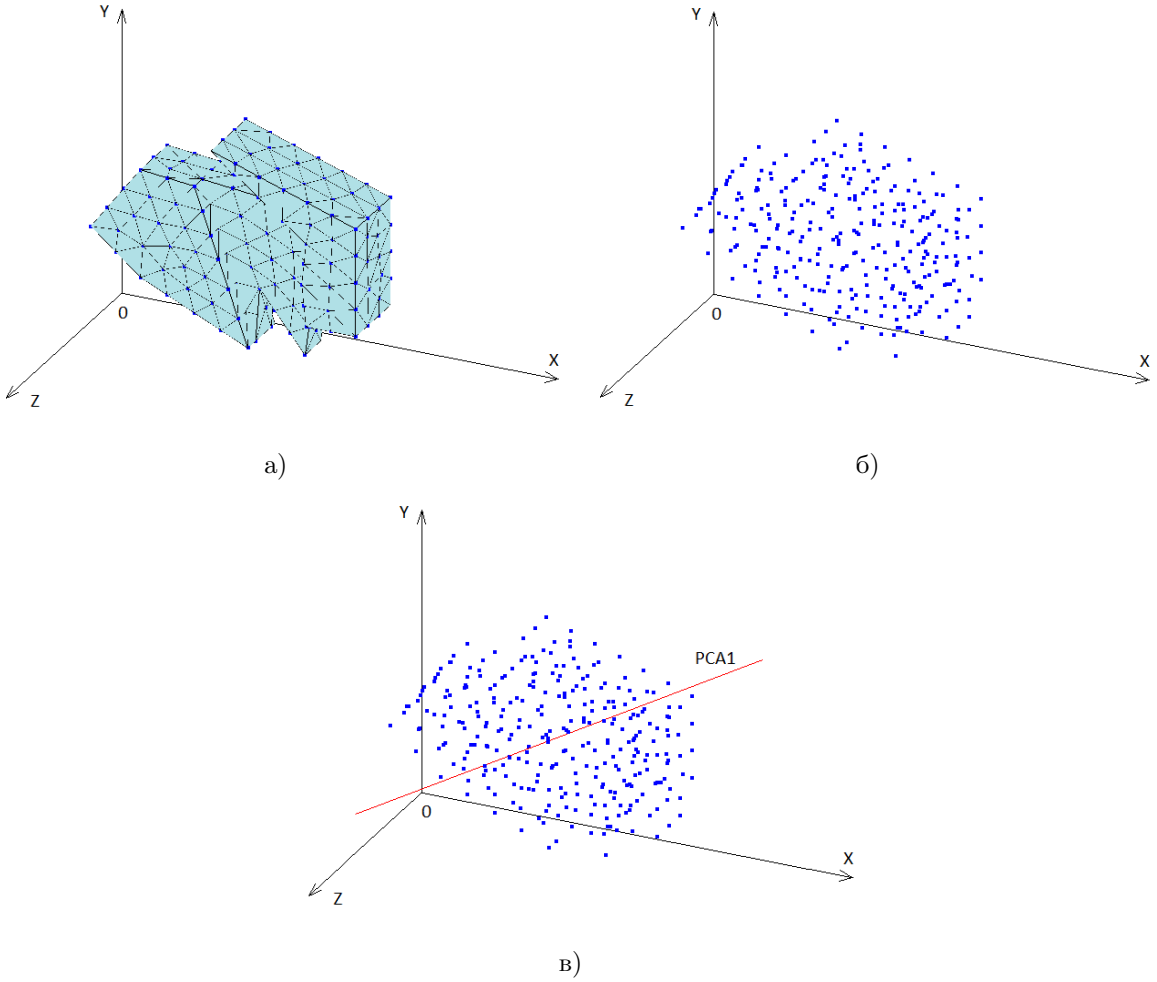


Рис. 7: Вспомогательный вывод программы: трехмерная сетка (а), узлы этой сетки (б) и первая главная компонента PCA1 (в)

тогональна первой и имеет направление, у которого второе распределение дисперсии вдоль этой прямой максимально. Следующие главные компоненты строятся аналогичными образом. Таким образом, цель анализа главных компонент заключается в том, чтобы объяснить максимальную величину дисперсии при наименьшем числе основных компонентов.

В качестве направления наибольшей протяженности для трехмерной сетки возьмем первую главную компоненту, так как она вдоль своего направления сохраняет наибольшее количество информации. В системе был

реализован алгоритм NIPALS (Nonlinear Iterative Partial Least Squares) [14], который итеративно продолжает поиск компоненты пока не будет достигнута задаваемая точность. В реализованном алгоритме разбиения сетки эта точность по умолчанию установлена 10^{-13}, которую при необходимости может изменить пользователь. Также важно обратить внимание на следующее. Алгоритм NIPALS сходится всегда, однако в некоторых случаях сходимость может быть очень медленной. Например, если исходные точки расположены на поверхности сферы, то никакая главная компонента не может понизить размерность. Для таких случаев расчет может быть прекращен досрочно. Чтобы прекратить процесс, в реализованном алгоритме установлено максимальное количество итераций, которое по умолчанию имеет значение 1000, так как для большинства наборов данных требуется не более 200 итераций для достижения сходимости. Более подробно изучить алгоритм NIPALS можно из источника [14].

3.4. Алгоритмы поиска центра сетки

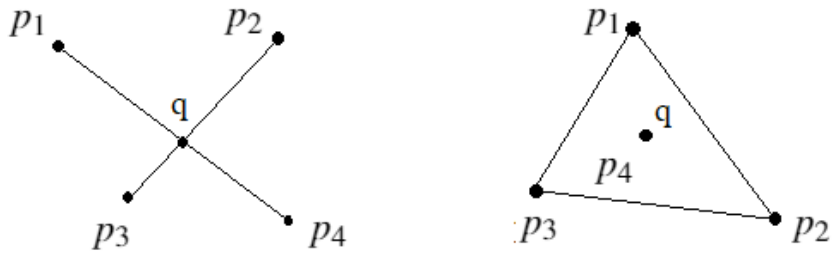
Для поиска "центра" мы ищем центральную точку множества вершин сетки. Центральной точкой множества P, состоящего из n точек в $\mathbb{R}^d$ называется такая точка c из $\mathbb{R}^d$, что каждая гиперплоскость, проходящая через нее разделяет P на два подмножества, размер каждого из которых не более, чем $nd/(d+1)$ [15]. Сначала был реализован простой метод (центр тяжести). Однако в некоторых случаях он дает неточные результаты. Поэтому был реализован более точный алгоритм нахождения точки Радона.

3.4.1. Точка Радона

Рассматриваемая эвристика находит приблизительную центральную точку, используя некоторый набор случайных узлов сетки и точку Радона для малых множеств. Пусть P - это множество из n точек в $\mathbb{R}^d$. Тогда точка $q \in \mathbb{R}^d$ называется точкой Радона множества P, если P можно разбить на два непересекающихся подмножества P_1 и P_2 таких, что q лежит в пересечении выпуклой оболочки P_1 и выпуклой оболочки P_2 [15]. Существует теорема Радона, которая гласит, что любое множество точек из $\mathbb{R}^d$, состо-

ящее более чем из $d+1$ точек имеет точку Радона. Более подробно теорию можно узнать из [15].

Сейчас остановимся на точках Радона для двумерного и трехмерного случаев. Пусть $P = \{p_1, p_2, p_3, p_4\}$, где $p_i \in \mathbb{R}^2$. Четыре точки на плоскости образуют три пары отрезков, концы которых образованы из этих точек. Если одна из пар отрезков пересекается, то в качестве точки Радона берется точка пересечения. Если ни одна пара отрезков не пересекается, то точкой Радона будет одна из четырех точек P, которая находится внутри треугольника, образованного из остальных трех точек (рис. 8). В трехмерном случае

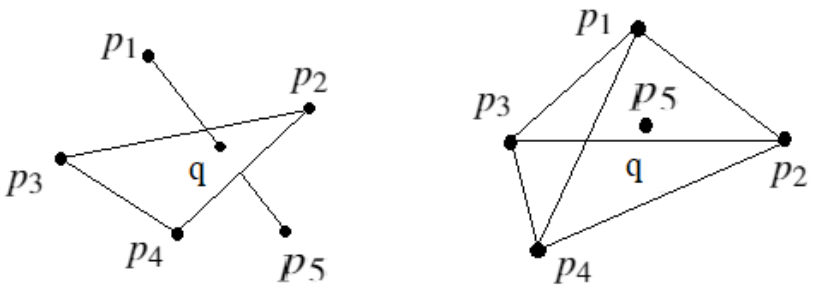


а) б)

Рис. 8: Точкой Радона q в $\mathbb{R}^2$ является либо пересечение отрезков (а), либо одна из точек, находящаяся внутри треугольника (б)

рассмотрим пять точек. То есть, пусть $P = \{p_1, p_2, p_3, p_4, p_5\}$, где $p_i \in \mathbb{R}^3$. Пять точек в пространстве образуют десять множеств пар, состоящих из отрезка и треугольника, которые образованы из данных пяти точек. Если в одной паре отрезок пересекает соответствующий ей треугольник, то точкой Радона будет точка пересечения. Иначе, точкой Радона является одна из рассматриваемых точек, которая находится внутри тетраэдра, образованного из остальных четырех точек (рис. 9).

Эвристический алгоритм нахождения центральной точки сетки заключается в следующем. Пусть N - количество узлов данной сетки. Выберем M случайных узлов, где M является степенью числа 4 (5) для двумерной сетки (для трехмерной). Для каждых четырех (пяти) точек найдем ее точку Радона. Тогда из M точек останется $M/4(M/5)$ точек Радона, для которых аналогичным образом найдем точки Радона. Продолжая этот процесс, в



а) б)

Рис. 9: Точкой Радона q в $\mathbb{R}^3$ является либо пересечение отрезка и треугольника (а), либо одна из точек, находящаяся внутри тетраэдры (б)

итоге получим одну точку Радона, которая и будет "центром" сетки. Достоинством данного алгоритма является то, что для сеток, состоящих из большого количества узлов, можно рассмотреть намного меньше точек. В реализованном алгоритме M по умолчанию имеет значение 4096 в $\mathbb{R}^2$ и 3125 в $\mathbb{R}^3$, которые при необходимости может изменить пользователь.

3.5. Минимизация дисбаланса разбиения

Чтобы дисбаланс разбиения был минимизирован, необходимо разделить текущую подобласть на две равные по количеству конечных элементов части. Другими словами, гиперплоскость бисекции должна располагаться так, чтобы полученные при делении подобласти были более одинаковыми. Следовательно, задача сходится к нахождению такой гиперплоскости.

Для улучшения поиска гиперплоскости бисекции был применен следующий алгоритм, который заключается в "улучшении" начальной гиперплоскости. Напомним, что начальная гиперплоскость проходит через "центр" сетки перпендикулярно оси инерции. После построения начальной гиперплоскости, строятся новые k гиперплоскостей, которые расположены параллельно относительно начального и проходят через k равноудаленных точек. Эти k равноудаленных точек расположены на прямой наибольшей протяженности, где средней точкой является проекция "центра" сетки на ось инерции. А удалены эти точки на расстояние равное половине среднего значения длины всех ребер сетки. Среди построенных k гиперплоскостей

выбирается гиперплоскость с наилучшим разбиением. Под наилучшим разбиением понимается разбиение, у которого минимален дисбаланс. Значение k по умолчанию равно 15. Пример выбора оптимальной гиперплоскости приведен на рис. 10.

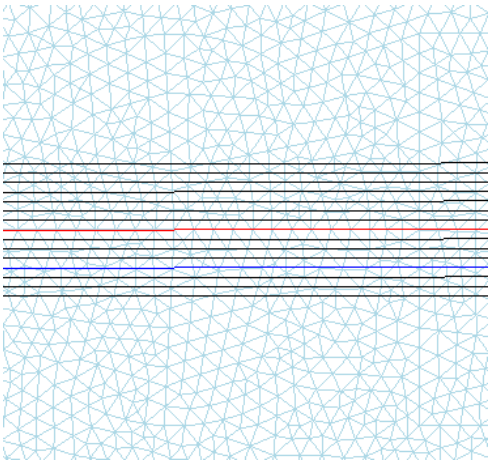
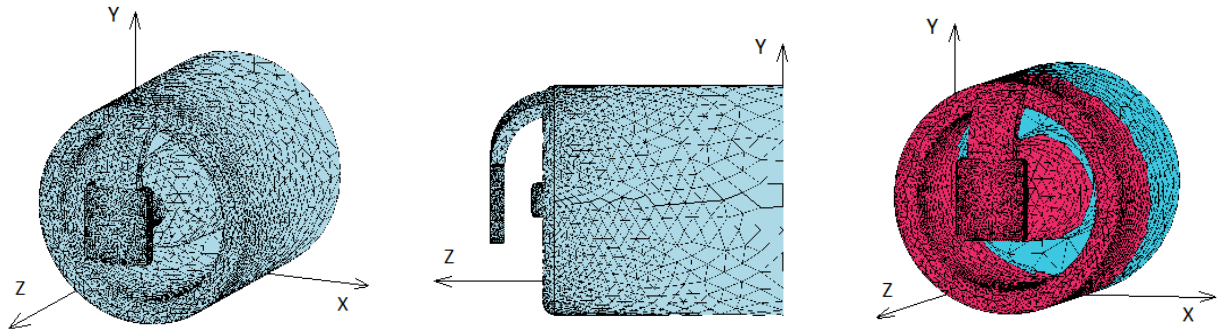


Рис. 10: Вспомогательный вывод программы: 15 гиперплоскостей в $\mathbb{R}^2$, где красным цветом обозначена гиперплоскость, проходящая через "центр" сетки, а синим - гиперплоскость, дающая наименьший дисбаланс разбиения

3.6. Особенности реализации

3.6.1. Связность разбиения

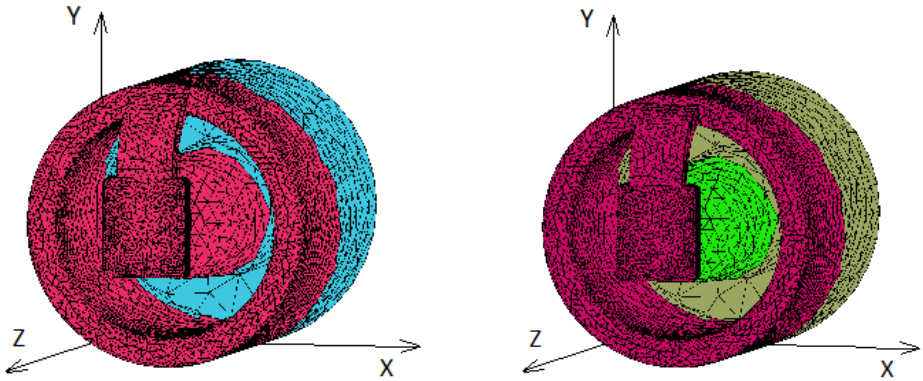
При делении текущей области на две части, возникает проблема со связностью полученных подобластей. Проблема заключается в том, что области могут иметь специфичную форму, из-за чего область разбивается не на две части, а на несколько. Пример такой сетки представлен на рис. 11. Как видно из этого рисунка, область разбилась на две части, где одна из частей (розового цвета) состоит из двух несвязных подобластей. Для таких случаев необходимо каждую компоненту связности представить в виде отдельной области. Чтобы найти такие связные области, в реализованном алгоритме используется поиск компонент связности. То есть, после разбиения каждая связная область рассматривается, как отдельный домен.



а) б) в)

Рис. 11: Пример работы программы: трехмерная сетка (а), ее вид сбоку (б) и разбиение (каждый домен имеет уникальный цвет) на две части (в)

Таким образом, отметим, что область из-за особой формы сетки не обязательно делится на две части. Пример работы алгоритма с учетом связности доменов представлен на рис 12.



а) б)

Рис. 12: Примр работы программы: разбиение на два домена без учета связности (а) и с учетом связности (б)

3.6.2. Распараллеливание алгоритмов

Как говорилось ранее, геометрические алгоритмы дают большие возможности для распараллеливания. В реализованных алгоритмах, средствами OpenMP [16] распараллелен главный метод рекурсивного разделения области на две части. Для этого используется шаблон Fork-Join, который в

основном применяется для рекурсивной декомпозиции. Распараллеленную часть алгоритма можно описать следующим образом. Каждый рекурсивный вызов создает задачу (task), которая загружается в пул задач. Затем группой потоков задачи одновременно запускаются из пула. Временное преимущество, которое дает распараллеливание представлена в результатах в разделе 4.

4. Результаты экспериментов

В рамках данной работы была разработана и реализована подсистема для разбиения сеток для системы ELCUT. В нее вошли два алгоритма: алгоритм для разбиения двумерных сеток, а также алгоритм для разбиения трехмерных сеток. Оба алгоритма были реализованы средствами языка программирования C++, библиотеки OpenGL для графического представления разбиения и библиотеки OpenMP для распараллеливания алгоритмов.

Была осуществлена проверка реализованных алгоритмов на различных расчетных сетках формата ELCUT, а также проведено сравнения нераспараллеленных версий алгоритмов с распараллеленными версиями.

Вычисления проводились на ноутбуке DELL Inspiron N5110 с процессором 2.2 GHz Intel Core i7-2670QM, с ОЗУ 8 ГБ, на операционной системе Windows 10.

Ниже представлены две таблицы, показывающие результаты работ программ для двумерных и трехмерных сеток.

Таблица 1: Результаты работ последовательной и параллельной версий алгоритмов разбиения двумерных расчетный сеток.

Информация о сетке			Макс. кол-во треугольников в каждом домене	Среднеквадратичное отклонение конечных элементов	Время работы последов. алгоритма, сек	Время работы параллельного алгоритма, сек
Название	Кол-во вершин	Кол-во треугольников				
dBig	4315	8334	600	≈ 50	0.506	0.353
upBig	4948	9481	700	≈ 100	0.58	0.395
dMagn4	10120	19924	2000	≈ 150	3.01	2.2
upHuge	477785	951438	70000	≈ 9000	76.262	49.373
bigTriang5	745431	1485922	300000	≈ 40000	100.366	74.391
bigB	874676	1743504	400000	≈ 90000	119.072	99.394

Стоит отметить, что из-за эвристического метода нахождения точки Радона, в котором точки выбираются случайным образом, алгоритм дает разные разбиения для одной и той же сетки. Поэтому для каждой сет-

ки была проведена серия экспериментов. Чтобы определить на сколько в среднем отклоняются количества конечных элементов в каждом домене от их среднего значения, для каждого эксперимента была оценена мера его приближения к среднему, как среднеквадратичное отклонение. Так как проводилось несколько экспериментов, в таблицах для каждой сетки представлены средние значения среднеквадратичного отклонения.

В таблице 1 представлены результаты работы программы для двумерных расчетных сеток. Некоторые графические результаты работы программы представлены на следующих рисунках.

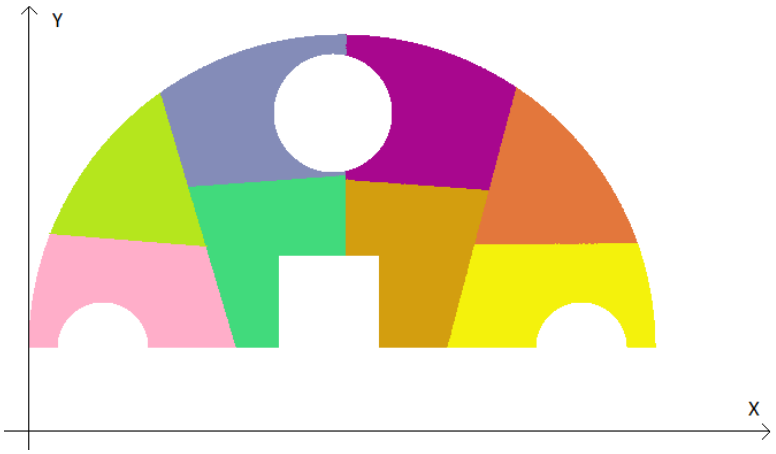


Рис. 13: Графический результат работы программы разбиения двумерной сетки "bigTriang5" из таблицы 1

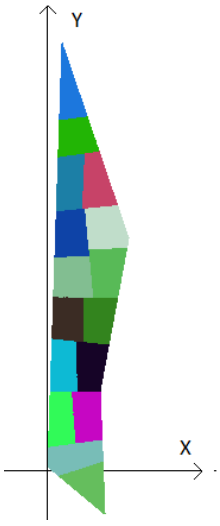


Рис. 14: Графический результат работы программы разбиения двумерной сетки "upHuge" из таблицы 1

Таблица 2: Результаты работ последовательной и параллельной версий алгоритмов разбиения трехмерных расчетный сеток .

Информация о сетке			Макс. кол-во тетраэдров в каждом домене	Среднеквадратичное отклонение конечных элементов	Время работы последов. алгоритма, сек	Время работы параллельного алгоритма, сек
Название	Количество вершин	Количество тетраэдров				
bent_wire	1040	4727	800	≈ 1	0.81	0.542
ellipse	3378	18582	5000	≈ 1	1.2	0.71
spark_plug	9581	37184	13000	≈ 3000	1.521	0.912
arcing_horns	16292	62624	20000	≈ 5000	1.954	1.409

В таблице 2 приведены результаты работ последовательной и параллельной версии программы разбиения для трехмерных расчетных сеток. Графические результаты разбиения сеток "spark_plug" и "arcing_horns" представлены на рисунках 15 и 16 соответственно.

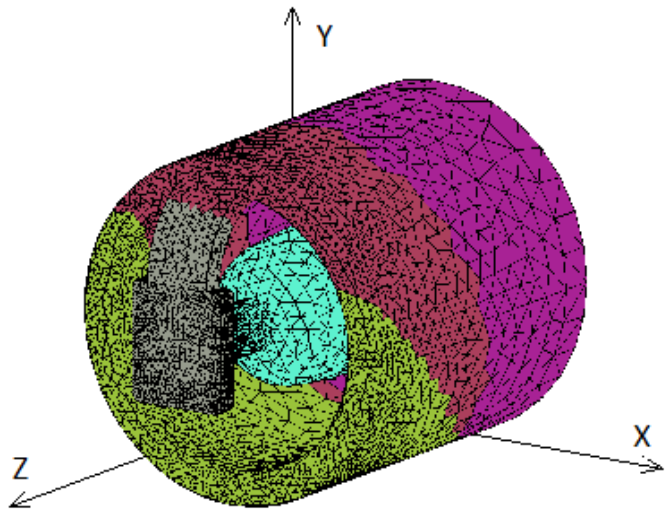


Рис. 15: Графический результат работы программы разбиения трехмерной сетки "spark_plug" из таблицы 2

Как видно из таблиц 1 и 2, параллельная версия алгоритма дает относительно быстрое вычисление разбиения, из чего рекомендуется использовать именно эту версию алгоритма.

На основании вышеизложенных результатов сделан вывод: разбиения достаточно хорошие, поэтому алгоритмы могут быть рекомендованы для

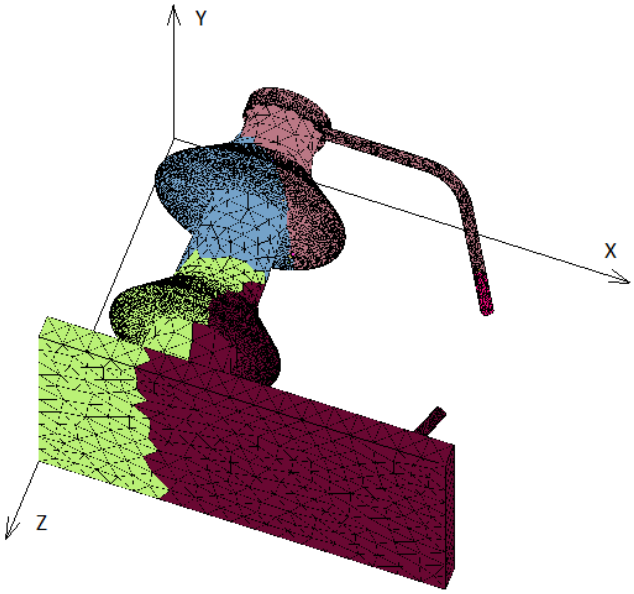


Рис. 16: Графический результат работы программы разбиения трехмерной сетки "arcing_horns" из таблицы 2

интеграции в систему ELCUT. Однако требуется дальнейшая доработка алгоритмов построения плоскости бисекции, с использованием возможности изменения угла для 3d.

Заключение

В рамках данной работы были достигнуты следующие результаты.

1. Проведено исследование алгоритмов разбиения сетки, базирующихся на геометрических, комбинаторных и иерархических методах, на основании которых был выбран подходящий алгоритм для разбиения двумерных и трехмерных сеток для системы ELCUT.

2. Разработана и реализована подсистема для разбиения двумерных и трехмерных сеток, и описаны ее основные элементы.

3. Проведены эксперименты с применением реализованных алгоритмов для различных двумерных и трехмерных сеток и сделаны выводы о возможности их использования в системе ELCUT.

Список литературы

[1] Dhatt G., Touzot G., Lefrancois E. Finite Element Method. Great Britain: ISTE Ltd, 2012. doi: 10.1002/9781118569764.ch5

[2] ECLUT.— URL: http://elcut.ru/

[3] Padua D. (Ed.). Encyclopedia of Parallel Computing. New York: Springer, 2011. doi:10.1007/978-0-387-09766-4

[4] Family of Graph and Hypergraph Partitioning Software | Karypis Lab.— URL: http://glaros.dtc.umn.edu/gkhome/views/metis

[5] Zhang L., Zhang G., Liu Y., Pan H. Mesh Partitioning Algorithm Based on Parallel Finite Element Analysis and Its Actualization. Mathematical Problems in Engineering, 2013, Vol. 2013, No. 751030. Available at: https://www.hindawi.com/journals/mpe/2013/751030/

[6] Wyrzykowski R., Dongarra J., karczewski K., Wasniewski J. (Eds.). Parallel Processing and Applied Mathematics. 7th International Conference. Poland: Gdansk, PPAM 2007, september 9-12.

[7] Korosec P., Silc J., Robic B. Mesh Partitioning: A Multilevel Ant-Colony-Optimization Algorithm. IEEE Proceedings of the International Parallel and Distributed Processing Symposium, 2003 doi:10.1109/IPDPS.2003.1213278

[8] Головченко Е. В. Декомпозиция расчетных сеток для решения задач механики сплошных сред на высокопроизводительных вычислительных системах: диссертация на соискание ученой стенепи кандидата физико-математических наук РАН институт прикладной математики. Москва, 2014.

[9] Preis R., Diekmann R. PARTY — a software library for graph partitioning. Advances in Computational Mechanics with Parallel and Distributed Processing. Advances in Computational Mechanics with Parallel and Distributed Processing, 1997, P. 63-71. Available

at: https://www.researchgate.net/publication/2736581_PARTY_-_A_software_library_for_graph_partitioning

[10] НОУ ИНТУИТ | Лекция | Параллельные методы на графах.— URL: http://www.intuit.ru/studies/courses/1156/190/lecture/4960?page=5

[11] Волков К.Н., Дерюгин Ю.Н., Емельянов В.Н., Карпенко А.Г., Козелков А.С., Тетерина И.В. Методы ускорения газодинамических расчетов на неструктурированных сетках. Москва: Изд-во ФИЗМАТЛИТ, 2013.

[12] Karypis G., Kumar V., Schloegel K. Graph Partitioning for High-Performance Scientific Simulations. Technical Report 00-018, University of Minnesota, 2000. Available at: https://www.researchgate.net/publication/2482474_Graph_Partitioning_for_Hi

[13] Фёрстер Э., Рёнц Б. [Frster E., Rnz B.] Методы корреляционного и регрессионного анализа: пер. с нем. В.М. Ивановой: Москва, Изд-во Финансы и статистика, 1983.

[14] Kim H. Esbensen, Guyot D., Westad F., Lars P. Houmoller. Multilevel Data Analysis – In Practice, 5th edition. Esbjerg: Alborg University, 2004.

[15] Clarkson K.L., Eppstein D., Miller G.L., Sturtivant C., S.-H. Teng. Approximating center point with iterated Radon points. 9th ACM Symp. Computational Geometry, P. 91-98, 1993. Available at: http://dl.acm.org/citation.cfm?id=161004

[16] Home – OpenMP.— URL: http://www.openmp.org/