УДК 519.683.2

ПРОГРАММНАЯ СРЕДА ГЕНЕРАЦИИ, ПЕРЕСТРОЕНИЯ И РАЗДЕЛЕНИЯ СЕТОК И ПОСТРОЕНИЯ РАСЧЕТНЫХ МОДЕЛЕЙ ДЛЯ ПАРАЛЛЕЛЬНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ

С. П. Копысов[1], А. К. Новиков[1], А. Б. Пономарев[1], В. Н. Рычков[1]

Рассматривается программная среда, описывающая неструктурированные сетки с ячейками произвольной формы, тесно связанная с описанием геометрии области. Обсуждается реализация алгоритмов, позволяющих строить сетки, а также перестраивать и разделять их для параллельных вычислений. Работа выполнена при поддержке РФФИ (код проекта 06–07–89015).

Ключевые слова: параллельные распределенные вычисления, неструктурированные сетки, расчетные модели, объектно-ориентированное программирование.

Введение. При решении задач математического моделирования одной из основных проблем является приведение геометрической модели к расчетной. Сначала по описанию тела необходимо построить расчетную сетку, а затем поставить расчетные условия и разделить сетку для параллельных распределенных вычислений. Все эти этапы требуется выполнять интерактивно с последующей визуализацией результатов.

Современные программные решения (ICEM CFD, Pro/Engineer, SDRC I-DEAS, SolidWorks и др.) предоставляют комплексы специализированных программных средств для пре- и постпроцессинга в задачах механики жидкостей и газов, деформируемого твердого тела и др. Сеточные генераторы имеют в них прямые интерфейсы к CAD-системам, которые не просто позволяют построить любую расчетную модель, но и взаимодействуют с расчетными модулями.

Важные проблемы параллельных и распределенных вычислений связаны с необходимостью подготовки больших объемов начальных данных и обработки массивов полученной информации до и после операций на многопроцессорных системах. В задачах с большими объемами обрабатываемых данных резко возрастает вероятность появления ошибки в исходных данных. Кроме того, все больше задач моделирования требуют прямого обращения к геометрическим ядрам и функциям из пользовательского параллельного распределенного приложения [1 – 3]. Поэтому актуальным является построение CAE-системы, включающей в себя кроме обычных функций ряд специфических: подготовку и работу с распределенными данными, дополнительные функции проверки правильности введенных данных, оптимальную передачу и визуализацию больших объемов данных с возможностью фильтрации ненужной в данный момент информации и др.

В настоящей статье рассматривается среда, описывающая неструктурированные сетки с ячейками произвольной формы, тесно связанная с геометрией области. Соответствующие алгоритмы позволяют строить сетки, оптимизировать и адаптировать сетку в процессе решения в соответствии с особенностями конкретной задачи и разделять ее для параллельных вычислений.

Визуальное приложение, основанное на данной программной среде, дает простой интерфейс для задания граничных условий и редактирования расчетных сеток и интегрировано с расчетными модулями метода декомпозиции области [4, 5].

1. Структура программной среды. Структура программной среды представлена пятью логическими составляющими: геометрия, сетка, расчетная сетка, разделенная сетка и распределенная сетка (см. строки в таблице).

Каждая составляющая содержит компоненты ввода/вывода, а также компоненты объектно-ориентированной модели и визуализации (см. столбцы в таблице). В свою очередь, объектно-ориентированная модель разбивается еще на две составляющие: структуры данных и алгоритмы.

В данном подходе деление на уровни отражает принцип декомпозиции сложных задач на более простые компоненты. Каждый уровень допускает расширение своих возможностей за счет разработки модулей, реализующих новые решения.

[1] Институт прикладной механики УрО РАН, ул. Т. Барамзиной, д. 34, 426067, Ижевск; e-mail: kopysov@udman.ru

Структура визуального окружения FEStudio

	Ввод/вывод	Объектно-ориентированная модель		Визуализация
		Структуры данных	Алгоритмы	
Геометрия	Импорт/экспорт (IGES/STEP/STL); автоматизированная процедура проверки и восстановления геометрии	Твердые тела; поверхности; каркасные модели; воксели	Поверхностная триангуляция	Структура визуального приложения на основе Open CASCADE и MFC
Сетка	Формат файла; импорт/экспорт (CGNS); автоматизированная процедура проверки и восстановления сетки	2D/3D сетки; ячейки произвольной формы; уникальные индексы объектов; размещение объектов в памяти; свойства сеточных объектов; связь с геометрическими данными	Построение сеток: — поверхностных; — тетраэдральных; — шестигранных; — смешанных. Оптимизация: — сглаживание; — огрубление. Адаптивное перестроение сеток по шаблону	Расширение Open CASCADE
Расчетные данные	Расширение формата файла для расчетных данных; импорт/экспорт (CGNS); связь с геометрическими и сеточными данными	Расчетные данные: — граничные условия; — материальные характеристики. Расчетная геометрия. Расчетная сетка	Расстановка расчетных данных; перенос граничных условий с геометрии на сетку	Визуализация исходных данных; визуализация результатов
Разделенная сетка	Ввод/вывод разделенной сетки	Иерархия сеток: — подсетка есть сетка; — граф подсеток; — заместители сеточных объектов	Методы статического разделения: — разделение взвешенного узлового/элементного графа; — разделение на основе геометрической информации	Визуализация разделенной сетки
Распределенная сетка	Параллельный ввод/вывод	Взаимодействие с объектами инфраструктуры CORBA; SPMD-объекты; взвешенный граф МВС	Параллельное построение и перестроение сеток; методы динамического перераспределения; мониторинг загрузки	Визуализация удаленных данных; визуализация больших объемов данных

При построении программной среды стояла задача разработать по возможности платформонезависимое программное обеспечение с использованием следующих технологий: объектно-ориентированное проектирование (UML, Rational Rose [6]) и объектно-ориентированное программирование (Visual C++, MFC, CORBA [7, 8]). В таблице перечислены некоторые свойства компонентов, реализованные на соответствующем уровне. Составляющие геометрии и сетки определяют базовый уровень, каждый следующий логический уровень зависит от предыдущих. На рис. 1 показано деление на программные подсистемы и их зависимости (каждая подсистема реализована в виде статической библиотеки с указанием на зависимости), по горизонтали сетка разделяется и распределяется, по вертикали расширяется функциональность.

Подсистемы `Mesh`, `Partitioned`, `Prototype`, `Prototype_Partitioned` являются абстрактными, т.е. классы этих библиотек не содержат данных. Реализации этих классов находятся в соответствующих библиотеках `ModelingData`. Подсистемы распределенных данных зависят как от абстрактных подсистем, так

и от подсистем-реализаций. Стрелки снизу указывают, от каких подсистем необходимо устанавливать зависимости для расширения функциональности.

2. Модель геометрии. Создание геометрической модели расчетной области является первым этапом при построении расчетной сетки (см. первую строку таблицы). Как правило, геометрические модели строятся в CAD-системах и экспортируются в файл для дальнейшей работы с ним [9]. Для работы с геометрией на этапе генерации сетки и при выполнении операций над ней (уточнение, огрубление и др.) была использована открытая система геометрического моделирования Open CASCADE [10].

Open CASCADE — среда программирования на базе C++ — представляет собой набор компонентов для разработки специальных научно-технических и профессиональных приложений.

Open CASCADE предоставляет разработчикам свободный доступ к кодам исходных текстов десятков структур данных 3D-геометрии — от объемных примитивов до сложного поверхностного моделирования — и включает в себя алгоритмы моделирования (такие как булевы операции, удаление невидимых линий, сглаживание и снятие фасок).

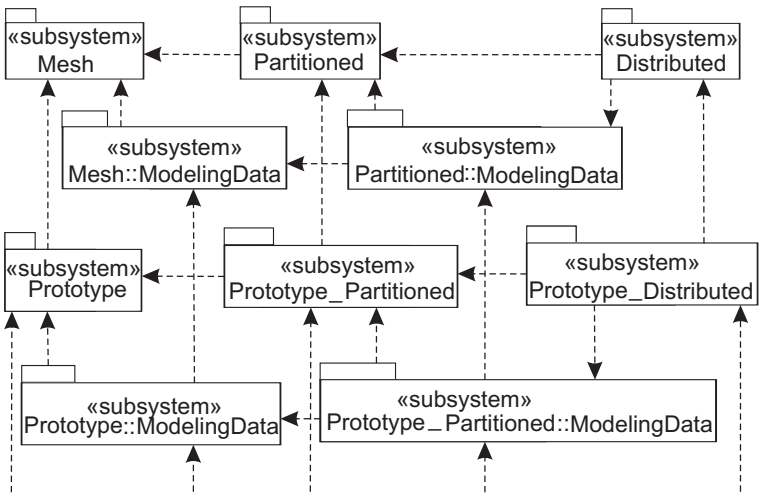


Рис. 1. Зависимости подсистем

Использование классов топологических форм объектов (таких как твердое тело, геометрическая грань, ребро, вершина) позволяет реализовать различные сеточные генераторы по криволинейным поверхностям и другие операции, требующие уточнения сетки по геометрическому описанию области.

3. Модель сетки. Выбор структуры для представления сетки оказывает существенное влияние на теоретическую трудоемкость алгоритмов, а также на скорость конкретной реализации сеточной модели. Выбрана структура данных, которая задает связанность объектов вида узлы–ребра–грани–ячейки. Данная структура является максимальной, так как задаются все возможные типы объектов. Объем требуемой оперативной памяти для хранения сетки больше, чем в других подходах, но это оправдывается широким классом задач, для решения которых такая модель может использоваться (задачи со свободными границами — контактные, сопряженные задачи; задачи оптимизации геометрии и топологии по напряженно-деформированному состоянию и др.). В одних алгоритмах некоторые из связей объектов могут не использоваться, в других алгоритмах операции с ребрами или гранями могут возникать нечасто, и поэтому ребра могут представляться косвенно. В таких же алгоритмах, как, например, p-версия метода конечных элементов, основные вычисления связаны с ребрами и гранями.

3.1. Базовые классы. Модуль Foundation Classes содержит классы, которые не относятся напрямую к модели сетки и другим моделям программной среды, но содержат важные данные, которые также используются и в разработках других программных продуктов.

Класс `Class` является базовым для всех классов модели сетки и совместно с шаблонным классом `handle` следит за ссылками на динамически созданные объекты, а также очищает память, когда все ссылки заканчиваются. Использование класса `handle` вместо звездочки (`handle<Object>` вместо `Object*`) облегчает использование, в особенности, такой сложной модели, как описанная ниже модель сетки.

Класс `Vector` представляет собой некоторую точку в трехмерном пространстве. Его использование позволяет выполнять над векторами различные математические операции (сложение, вычитание, умножение на скаляр, векторное произведение и др.). Класс `Vector` является динамическим массивом и помимо итераторов имеет оператор (`operator [](int)`). Классы `List` и `Set` являются сбалансированными бинарными деревьями, причем `List` — неупорядоченная последовательность объектов, а `Set` — упорядоченная последовательность объектов. Класс `Map` является картой соответствия одного объекта другому. В сложных комплексах библиотек иногда требуется “склеивать” разные списки однотипных объектов; для этого вводится специальный итератор (`PartitionedIterator`), позволяющий пробегать последовательно по объектам нескольких списков.

3.2. Объектно-ориентированная модель расчетной неструктурированной сетки. Основу ба-

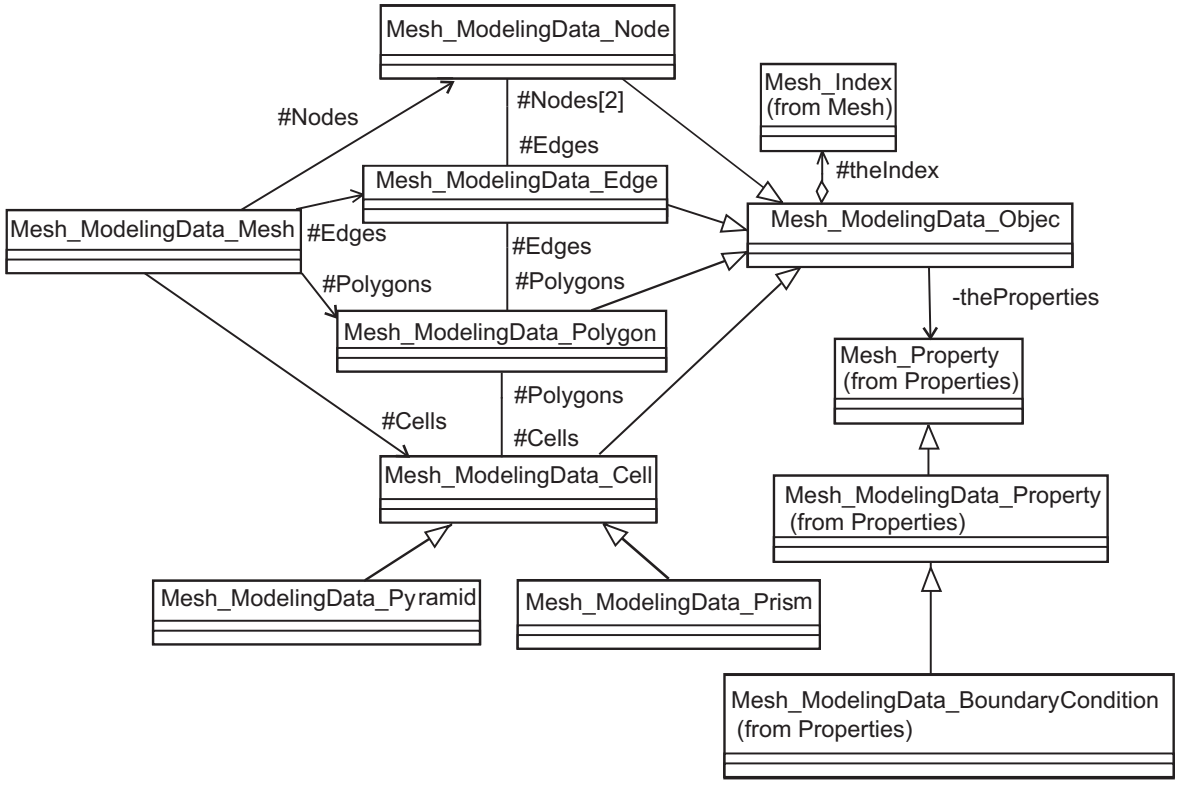


Рис. 2. Диаграмма классов модели сетки

зовой объектно-ориентированной модели сетки (вторая строка таблицы) составляют два класса (рис. 2): `Object` и `Mesh` (в диаграмме имена классов начинаются с префикса `Mesh_ModelingData`, префиксами обозначается принадлежность класса к той или иной логической подсистеме).

Класс `Object` является базовым для всех сеточных объектов. `Object` содержит функции по восстановлению древовидной структуры сетки (связности между объектами разных уровней), по установке и получению свойств объектов, по вычислению центра масс (`Vector CentreOfMass()`). Большинство из этих функций реализуются только в классах-наследниках, описанных ниже.

Классы `Node`, `Edge`, `Polygon`, `Cell` представляют объекты соответствующих уровней (размерностей): узел, ребро, грань, ячейка (рис. 2). Между соседними уровнями существует двунаправленная связь — ребро содержит ссылки на узлы, которыми оно образовано, а узел — на соответствующие ребра. Связи “вниз” (от ребра к узлу и т.д.) создаются и заполняются при инициализации объекта. Наличие связи “вверх” (от узла к ребру и т.д.) объясняется необходимостью определения соседних элементов в различных операциях над сетками. Связи “вверх” представляют собой динамически изменяющиеся массивы (`Sequencese::Vector`) и заполняются по мере создания объектов. Отметим также, что координаты узла выделяются динамически, за счет этого реализуются как 3D- и 2D-сетки, так и смешанные сетки.

Из всего многообразия форм ячеек в трехмерном случае выделены два типа — пирамида и призма. Эти классы реализуют наиболее часто используемые типы ячеек в методе конечных элементов — тетраэдры, шестигранники, пятигранные пирамиды и призмы. Отличия между различными элементами заключаются только в процедуре их инициализации и функциях доступа к данным, поскольку требуется особая сортировка объектов внутри элемента. При необходимости без значительных затрат времени могут быть реализованы ячейки любой формы (например, многогранники Вороного).

Класс `Property` позволяет приписывать к сеточному объекту произвольные свойства. Так, путем наследования реализовано граничное условие, содержащее номер поверхности, к которой принадлежит объект (т.е. грань). Для хранения свойств в объекте выделяется динамический список его свойств. В классе `Property` определены функции для сравнения двух свойств на совпадение по типу (а также сортировке по типу для упорядоченного хранения в объекте) и для сравнения однотипных свойств по значению параметров.

На основе класса `Property` реализуются различные расчетные данные (рис. 3). Среди них можно

выделить нагрузку (в зависимости от приложения к объекту она интерпретируется как узловая, поверхностная или объемная), материал объекта сетки (свойства конечного элемента) и различные граничные условия: заданное перемещение, жесткое закрепление, условие симметрии и т.д.

Класс `Mesh` служит контейнером для всех сеточных объектов и свойств и следит за тем, чтобы все объекты присутствовали в единственном экземпляре. Создание новых объектов и свойств производится через вызов соответствующих функций сетки, которые проверяют наличие такого объекта в списке: если объект уже создан, функция возвращает уже имеющийся, в противном случае новый объект добавляется в список. Единственность объектов достигается введением уникального индекса.

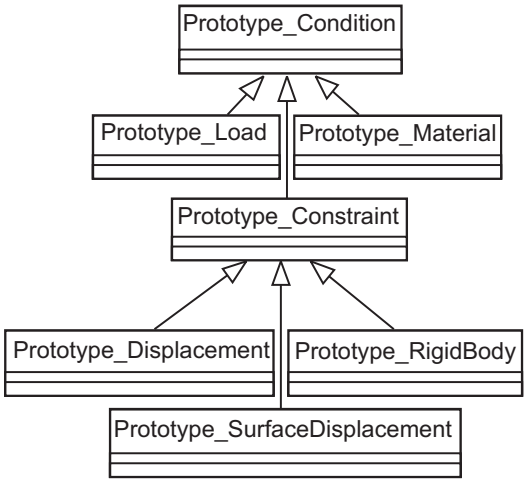


Рис. 3. Диаграмма классов расчетной модели сетки

Класс `Index` (рис. 3) представляет индекс объекта и формируется на основе его геометрического местоположения. При вычислении индекса координаты центра масс объекта нормируются по габаритам исходного тела (нормировка переводит координаты в отрезок от -1 до 1, 0 переходит в 0). Нормированные значения умножаются на максимальное целое MAX, записываемое в индексе, и округляются до целого (при использовании трех байт на хранение одной координаты будем иметь $\mathrm{MAX} = 2^{8*3-1} = 8\,388\,608$, при этом точность округления по каждой координате составляет $\mathrm{MAX}^{-1} \approx 10^{-7}$ от размеров геометрии).

Функции по созданию, добавлению, удалению и поиску сеточных объектов позволяют реализовывать различные алгоритмы построения сетки.

В интерфейс класса сетки добавлены функции по работе с геометрией для нахождения проекции точек на соответствующие геометрические ребра или поверхности. Классы описания геометрии Open CASCADE также широко используются для генерации и перестроения сеток.

3.3. Сеточные генераторы. Построение неструктурированной 3D-сетки для сложных геометрических областей в данной программной среде реализовано в три этапа: анализ CAD геометрии и построение реберной сетки, т.е. аппроксимация топологических криволинейных ребер (каркасной модели) отрезками; построение поверхностной сетки методом сжатия текущей границы (развивающегося фронта) с адаптацией к геометрии; генерация объемной сетки — алгоритм Делоне с ограничениями.

Особое внимание уделено построению сеток для многосвязных областей. Проблема заключается в том, что при объединении тел в одно теряется граница между подобластями, в то время как независимое построение на каждой из подобластей приводит к несогласованной сетке. Модификация описанного выше алгоритма дает возможность построения сетки на многосвязных областях и выполнения сеточной генерации параллельно на многопроцессорной системе. Процесс построения сеток основан на представлении тела в системе геометрического моделирования OpenCASCADE [10].

Шаг 1. Создается объединенный объект `TopoDS::Compound`, в котором задаются все твердые тела, для которых требуется построить сетку. На объекты накладывается дополнительное условие: все поверхности, по которым смежны объекты, должны быть идентичны.

Шаг 2. Из объединенного объекта извлекаются все твердые тела.

Шаг 3. Для каждого из твердых тел создаются объекты представления геометрии и сетки. Каждый объект представления геометрии инициализируется, и выполняется этап анализа геометрии. На этом этапе задаются локальные шаги сетки, при этом общие ребра и грани обрабатываются один раз для всех подобластей.

Шаг 4. Генерируется реберная сетка. Реберная сетка — это кусочно-линейная аппроксимация геометрических ребер. Перед генерацией реберной сетки создаются списки соответствия каждому ребру граней с указанием ориентации. При генерации реберной сетки все геометрические ребра обрабатываются один раз, а сетка, построенная по ним, записывается во все объекты представления сетки, связанные с соответствующим ребром.

Шаг 5. Независимо для каждой подобласти производится построение поверхностной сетки.

Шаг 6. Поверхностные сетки согласуются; для этого ищутся общие геометрические грани между подобластями, затем сетка с одной из подобластей копируется на смежную по соответствующей грани. В функции копирования граней производится поиск идентичных узлов, чтобы при удалении поверхностной

сетки на одной из подобластей не удалить элементы реберной сетки. После удаления лишних узлов и треугольников происходит копирование координат узлов из смежной области и создание копий поверхностных элементов.

Шаг 7. Генерация объемных сеток на каждой из подобластей. Этот этап также выполняется независимо для каждой подобласти и завершает построение сетки на многосвязной области. За счет того, что на предыдущем этапе поверхностные сетки были согласованы, дополнительно согласовывать объемные сетки не надо.

В программной среде предусмотрено подключение внешних библиотек для построения сеток; такая возможность была реализована на примере генератора сеток GRUMMP [11].

3.4. Перестроение сеток. В числе процедур, реализованных для перестроения сеток, можно выделить такие, как преобразование тетраэдральной сетки в шестигранную (разделение тетраэдра на четыре гексаэдра), регулярное деление тетраэдров, адаптивное перестроение сетки (например, деление по большой стороне). Все эти методы объединяет то, что на вход им подаются тетраэдры, каждый из которых разбивается по определенному принципу, и соседние неконформные элементы вновь подаются на вход соответствующего алгоритма.

Отметим, что при разбиении тетраэдра появляются новые узлы. Как правило, новые узлы ставятся в середины исходных ребер, треугольников и элементов, при этом описание исходной геометрии ухудшается, поскольку новые точки уже не принадлежат границе области. Использование классов Open CASCADE позволяет уточнить положение новых узлов, и описание геометрии при этом значительно улучшается.

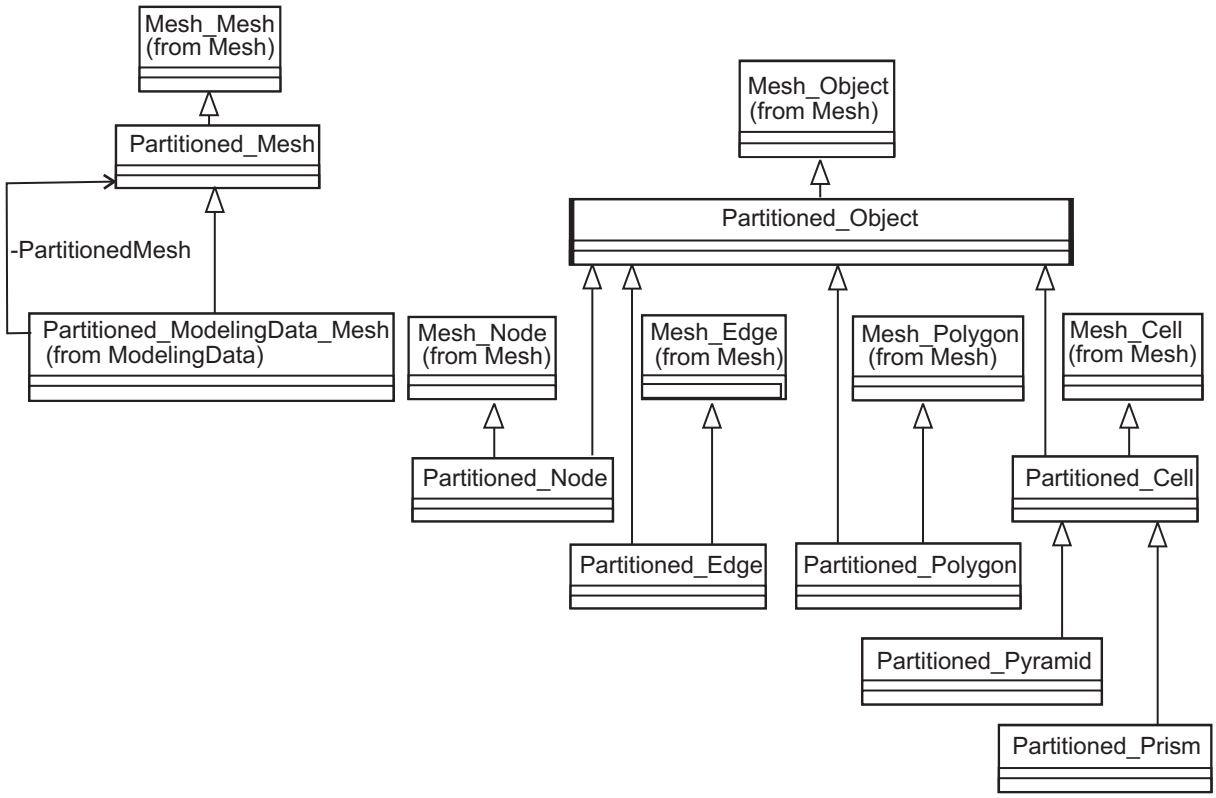


Рис. 4. Диаграмма классов разделенной сетки

4. Модель разделенной сетки. Модель разделенной сетки (четвертая строка таблицы) базируется на модели сетки (рис. 1), введенной выше. Классы разделенной модели наследуют соответствующие им классы базовой модели. На рис. 4 представлена модель разделенной сетки.

Важной особенностью разделенной модели является то, что и сетка, и подсетка представляются одним и тем же классом. Класс `Mesh` ссылается сам на себя. Данный подход позволяет рассматривать подсетки как самостоятельные сетки, что, в свою очередь, позволяет применять к подсеткам те же алгоритмы и использовать их в тех же расчетах, что и разделенные сетки, без необходимости внесения изменений в программный код. Примером использования этой особенности является алгоритм адаптивного деления тетраэдров, описанный ниже.

4.1. Методы разделения сеток. Для разработки эффективных решений задач как при последовательном, так и при параллельном подходе необходимы алгоритмы, которые обеспечивают эффективное разделение неструктурированных сеток.

Для разделения расчетной сетки существует множество алгоритмов, которые можно условно разделить на две группы: геометрические и графовые алгоритмы разделения сетки [12].

После применения какого-либо алгоритма разделения, на выходе которого имеется разделение вершин графа (т.е. разделение узлов или элементов сетки), необходимо определить, в каких подсетках должны находиться другие объекты сетки (ребра, грани, элементы или грани, ребра, узлы соответственно узлового и дуального графов). Этот этап является обязательным перед шагом распределения сетки по вычислительным узлам многопроцессорной вычислительной системы. В процессе разделения объектов сетки выделены следующие шаги.

Подготовка объектов. Определяются граничные и внутренние объекты, которые должны быть созданы в подсетках, формируются списки объектов для пересылки в подсетки.

Рассылка объектов. Выполняется последовательная рассылка объектов. В первую очередь в подсетках создаются клиенты на граничные объекты. Затем создаются локальные узлы согласно передаваемому списку координат. Вслед за узлами создаются ребра, затем грани и ячейки. Выполнение каждого последующего шага должно производиться только при условии завершения предыдущего; другими словами, ребра не должны создаваться раньше, чем будут созданы клиенты для граничных объектов и узлы, грани — не ранее, чем создадутся ребра, ячейки — после создания всех граней. Отличительной особенностью такого способа является то, что после непосредственных рассылок данных процессы выполняются на подсетках параллельно.

5. Модель распределенной сетки. Обеспечение возможности распределения подсеток и сеточных объектов требует расширения модели разделенной сетки и создания новой подсистемы, описывающей распределенную сетку (см. пятую строку таблицы), элементы которой находятся в различных адресных пространствах (в том числе и на различных вычислительных узлах многопроцессорной вычислительной системы). Введение распределенной сетки вносит дополнительные требования. С одной стороны, необходимо обеспечить коммуникации между сеткой и подсетками, а также объектами, находящимися в разных подсетках, с другой — необходимо сделать вносимые изменения прозрачными для используемых алгоритмов, чтобы избежать необходимости их повторной реализации.

В распределенной модели введено понятие остова сетки, которое обеспечивает единую точку входа и синхронизации для всех алгоритмов, применяемых к сетке. Остов сетки берет на себя управление подсетками и организацию коммуникаций между ними.

При разделении сетки возникают граничные объекты, т.е. объекты, принадлежащие двум или более подсеткам. При получении подсетками копии граничных объектов возникает проблема синхронизации этих копий. В больших сетках, число узлов в которых достигает пяти-шести порядков, синхронизация граничных объектов значительно снижает параллельную эффективность.

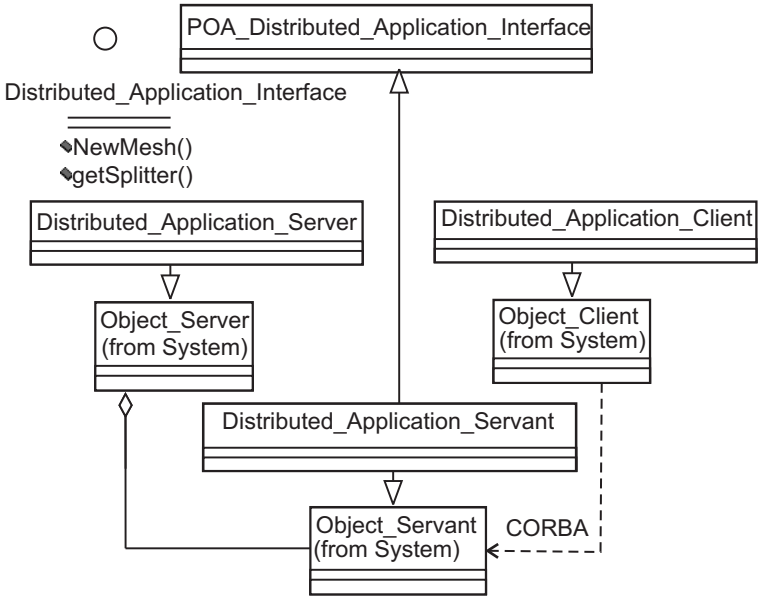


Рис. 5. Набор классов для взаимодействия между клиентом и удаленными серверами

В рассматриваемой модели распределенной сетки граничные объекты хранятся в остове, а в подсетках создаются заместители, которые внешне ничем не отличаются от обычных сеточных объектов, однако их внутренняя реализация построена по-другому. Заместители обеспечивают доступ к реальному объекту посредством коммуникаций, которые в распределенной модели реализуются на основе применения технологии CORBA [8]. Такой подход позволяет сделать граничные объекты прозрачными для используемых алгоритмов и, следовательно, не нарушать общие требования,

предъявляемые к распределенной модели.

Для обеспечения управления распределенной сеткой построен набор сервисных классов, которые обеспечивают доступ к остову сетки, а также позволяют остову получать доступ к подсеткам, расположенным на других узлах вычислительной системы. На рис. 5 приведена диаграмма CORBA-интерфейса (`Distributed_Application_Interface`) и его абстрактной реализации, содержащей некоторые вспомогательные методы, но не включающей в себя реализаций методов, описанных в интерфейсах, — все они создаются разработчиком.

Взаимодействие с удаленными серверами производится посредством вызова методов клиентов, связанных с этими серверами. Связь клиента с сервантом обеспечивается силами CORBA, сервант же имеет прямой доступ к серверу, поскольку находится в том же адресном пространстве.

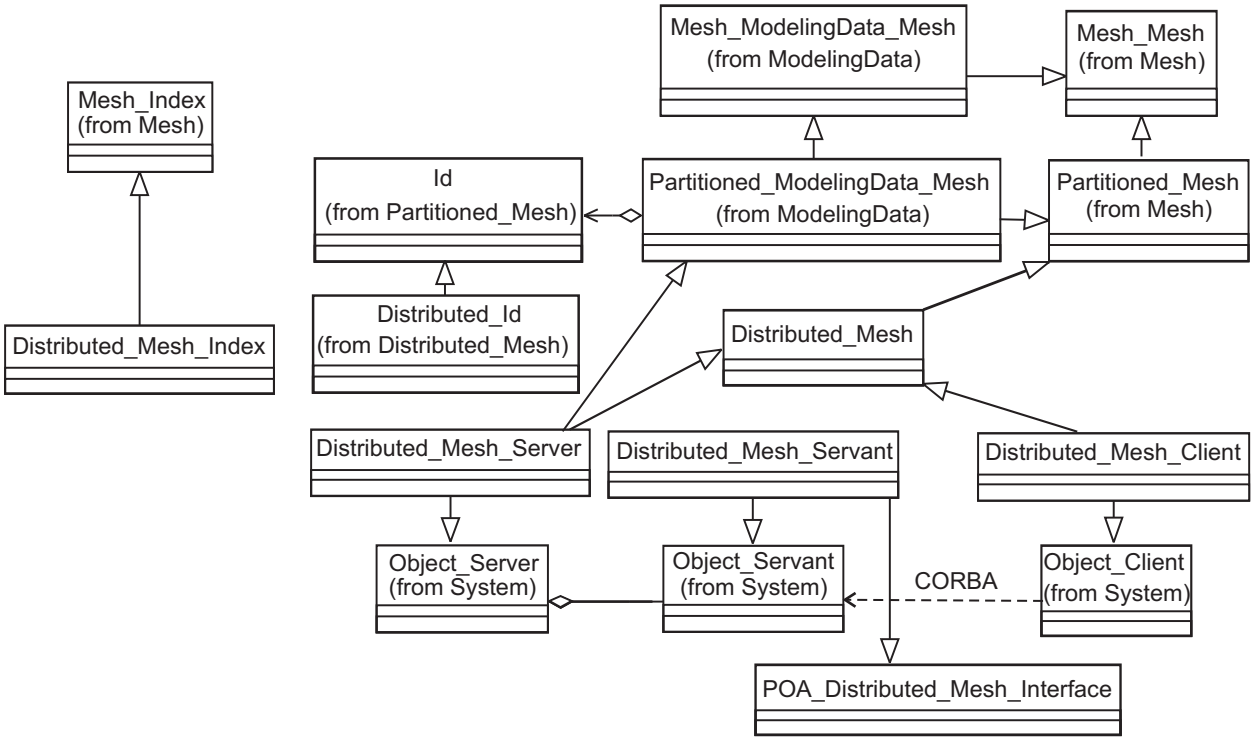


Рис. 6. Модель классов для удаленного доступа между остовом сетки и подсетками

Необходимость введения клиентов и серверов требует проведения декомпозиции модели, на базе которой строится распределенная модель. В нашем случае базовой является модель разделенной расчетной сетки, поэтому классы, ее составляющие, должны быть рассмотрены с точки зрения удаленного доступа к ним. В первую очередь необходимо обеспечить удаленный доступ к подсеткам из остова и к остову из подсеток. На рис. 6 представлена модель классов для организации двунаправленного удаленного доступа. Модель разделена на уровни. На первом уровне отображены классы расчетной сетки, которые легли в основу распределенной сетки: это — интерфейс обычной расчетной сетки (`Mesh`), его реализация (`ModelingData_Mesh`) и класс `Index`, служащий для представления индексов объектов сетки (узлов, ребер, граней и ячеек).

На втором уровне отображены классы модели разделенной сетки, являющиеся непосредственными предками классов распределенной сетки. Помимо описанных выше классов на втором уровне отображен также и класс `Id`, описывающий уникальный идентификатор сетки. Поскольку стандарт CORBA является платформонезависимым, для передачи данных по сети требуются специальные классы, обеспечивающие унифицированное представление данных без уточнения деталей реализации на конкретной платформе. Для работы с такими вспомогательными классами спроектированы адаптеры, которые обеспечивают механизмы по конвертации данных в их CORBA-представления и обратно. Идентификатор сетки и индекс объекта передаются по сети, поэтому для работы с ними реализованы адаптеры, изображенные на третьем уровне в модели. Это классы `Distributed_Id` и `Distributed_Index`.

Основой распределенной модели являются три класса, отображенные на четвертом уровне: “клиент”, “сервант” и “сервер”. Следует отметить, что класс “клиент” не содержит в себе никаких данных и потому

наследуется от абстрактного класса `Mesh`. Класс "сервер" же, наоборот, представляет собой сетку, поэтому должен содержать все ее данные, на основе которых он унаследован от класса `ModelingData_Mesh`, реализующего абстрактный класс `Mesh`. Следует отметить, что разделение "клиента" и "сервера" на уровне классов не означает четкого разграничения их функций: и остов, и любая из подсеток могут являться как "сервером", так и "клиентом". В первую очередь, отдельное рассмотрение классов "клиента" и "сервера" необходимо для того, чтобы на уровне программного кода в любой момент времени можно было определить, какую роль играет сетка.

Пятый уровень содержит базовые системные классы, реализующие стандартное поведение "клиента", "сервера" и "серванта".

На шестом уровне отображен CORBA-интерфейс распределенной сетки. В являющемся абстрактным классе `POA_Distributed_Mesh_Interface` реализованы лишь коммуникационные и сервисные методы, необходимые для получения доступа непосредственно к CORBA-объекту, обеспечивающему связь с удаленным сервером. Реализация методов удаленного доступа к сетке содержится в соответствующем классе серванта.

Для создания экземпляра подсетки клиент-приложение делает запрос сервер-приложению, вызывая метод `NewMesh()` у серванта, ссылка на который хранится внутри клиента. Промежуточное программное обеспечение, предоставляемое CORBA, обрабатывает запрос и посылает сообщение требуемому серверу. Сервер создает внутри себя экземпляр класса `Distributed_Mesh_Server` и возвращает ссылку на CORBA-объект, с помощью которого клиент сможет связаться с этим сервером. Сервант приложения, к которому было обращение, создает экземпляр класса `Distributed_Mesh_Client` и возвращает ссылку на него клиенту приложения. В дальнейшем работа с сеткой, находящейся на удаленном сервере, протекает аналогично работе с обычной сеткой, все необходимые преобразования аргументов и вызовы методов берут на себя клиент и сервант сетки, связь же с удаленным сервером поддерживается средствами CORBA.

Интерфейс удаленного взаимодействия остова и подсетки является важной, но не единственной составляющей распределенной модели. Необходимо также обеспечить взаимодействие объектов сетки между собой и сеткой. Классы, обеспечивающие удаленный доступ к объектам, изображены на рис. 7. Взаимодействие с удаленным объектом является двусторонним. Поскольку в модели клиенты не являются полноценными объектами, а только лишь их заместителями, взаимодействие с ними организовано по другому интерфейсу.

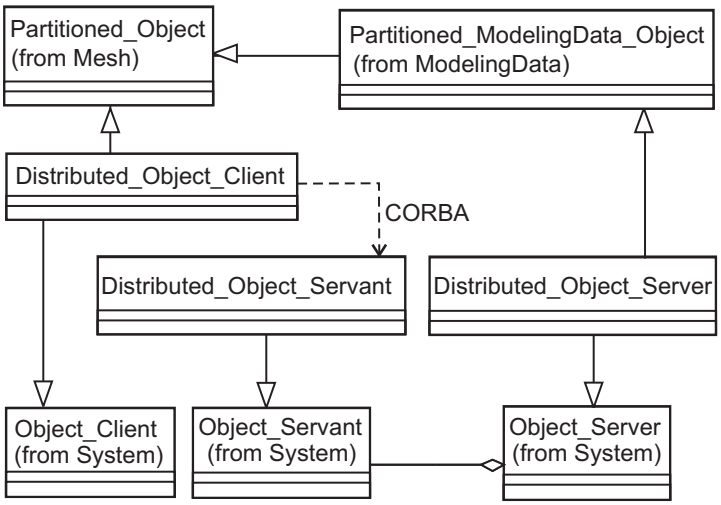


Рис. 7. Диаграмма классов, обеспечивающих удаленный доступ к объекту

На рис. 8 приведена диаграмма классов, отображающая связь между понятиями распределенного и удаленного объектов. Как видно из диаграммы, удаленным объектом считается клиент для объекта, находящегося в остове (в случае граничного объекта) или в другой подсетке. В свою очередь, сам клиент является сервером (в терминах данной модели — сервером удаленного объекта, `Distributed_RemoteObject_Server`). Введение понятия удаленного объекта и совмещение ролей клиента и сервера у заместителя объекта позволило обеспечить двустороннее взаимодействие, которое является необходимым для правильного функционирования программной среды.

Рассмотрение всех сеточных объектов в распределенной модели с точки зрения обеспечения удаленного доступа к ним позволяет избежать модификаций используемых алгоритмов, поскольку внешне такие объекты ничем не отличаются от локальных аналогов (например, из модели разделенной сетки), а все необходимые преобразования и коммуникации скрыты во внутренней реализации.

5.1. Перераспределение сеточных объектов. Динамическое перераспределение сеточных объектов применяется при балансировке нагрузки между вычислительными узлами. При перераспределении сеточных объектов, помимо загруженности вычислительных узлов, немаловажную роль играют затраты на выполнение непосредственно пересылки данных сеточных объектов [13]. Выделено три стратегии

перераспределения объектов.

1. Перемещение сеточного объекта со всеми свойствами. Эта стратегия является наиболее затратной из всех рассматриваемых, поскольку необходимо переместить все данные, и выбирается в качестве базовой в построенной модели.

2. Перемещение сеточного объекта с созданием некоторых (или всех) его свойств заново. Применяется в том случае, если восстановить данные свойства проще и быстрее, чем переслать их по коммуникационной сети. Для реализации этой стратегии у класса, реализующего конкретное свойство, в функции перемещения на подобласть необходимо вместо реального перемещения указать уничтожение свойства с созданием его на другой стороне и запустить его инициализацию.

3. Удаленный доступ через заместителя без реального перемещения объекта. Данная стратегия является самой экономной по времени перераспределения, но самой затратной в дальнейших вычислениях. Она может использоваться в том случае, когда пересылка данных всего объекта очень длительна и прогнозируется повторное перемещение объекта. В этом случае используются заместитель объекта, который создастся автоматически при попытке доступа к объекту из другой подсетки.

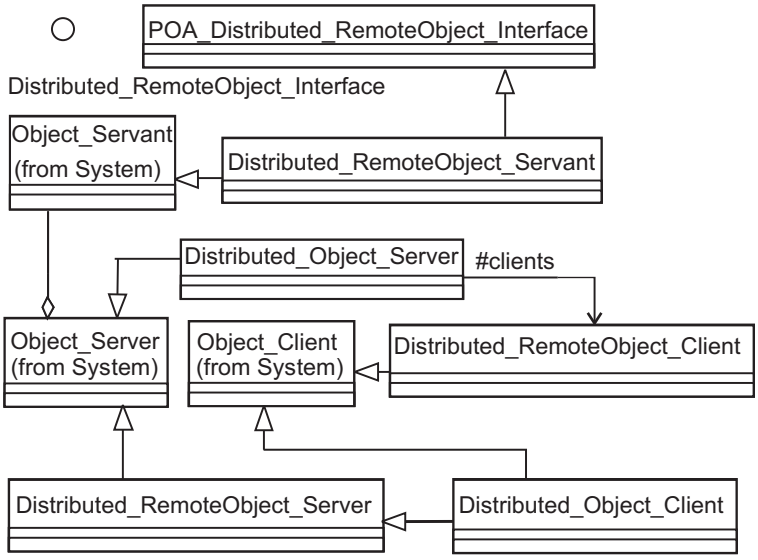


Рис. 8. Связь понятий распределенного и удаленного объектов

6. Визуализация. Компоненты визуализации (четвертый столбец таблицы) делятся на четыре части: визуализация геометрии, сетки, расчетных данных и разделения сетки на подсетки. Каждая из частей является потомком предыдущей и наследует все ее функциональные возможности.

Как уже отмечалось, Open CASCADE имеет модули структуры приложения (Application Framework) и визуализации (Visualization) для построения визуальных систем; на их основе, с использованием MFC реализована визуальная оболочка [14] (визуальный редактор расчетных моделей FEStudio), содержащая окно отображения геометрии и древовидной структуры документа и окно установки свойств объектов. Библиотека визуального редактора содержит следующие классы: `Application`, `Document`, `MDIChildForm`, `View`, `Browser`, `Properties`, `TreeView`.

Класс `Application` обеспечивает работу с шаблонами документов. Класс `Document` отвечает за открытие, модификацию, сохранение и импорт/экспорт стандартных файлов Open CASCADE, содержащих геометрию. Класс `View` отображает объект геометрически, `TreeView` — структурно, в виде дерева. Связь между визуальным объектом Open CASCADE, классом `AIS_InteractiveObject` и соответствующей строкой дерева, описанной структурой `TREEITEM`, осуществляется классом `TreeInteractiveObject`, инкапсулирующим эти два типа данных и осуществляющим их интерактивное взаимодействие.

В библиотеке визуализации сетки реализованы интерфейсы для импорта/экспорта некоторых сеточных форматов (GRUMMP [11], CGNS [15] и др.). В меню приложения введены пункты по режимам отображения сеточных объектов: возможно вывести узлы, ребра, граничные полигоны, ячейки как все вместе, так и по отдельности. Естественно, отображение всех объектов сетки весьма затратно, поэтому режим “по умолчанию” — отображение всех граничных поверхностей.

Библиотека содержит функции по сбору статистических данных по сетке: длины ребер, площади полигонов, объемы ячеек, плоские и двугранные углы, радиусы вписанных и описанных окружностей и сфер. Все данные выводятся в виде гистограмм.

Для визуализации расчетных данных в соответствующей библиотеке введены наследники классов отображения сеточных объектов с расширенными функциями `Compute`. Задание граничных условий на соответствующих сеточных объектах осуществляется при помощи окна свойств. После выделения необходимых объектов в окне геометрической визуализации или дереве на соответствующей граничному условию вкладке задаются параметры расчетных данных и всем этим объектам приписываются указанные свой-

ства. Установленные граничные условия отображаются схематически в окне отображения геометрии при каждом объекте. Для улучшения восприятия расчетных данных введен параметр масштабирования их относительно габаритов расчетной области.

Отображение подсеток с разнесением осуществляется введением в библиотеку наследника класса `AIS_InteractiveObject`.

7. Примеры построения расчетных моделей. Предлагаемая программная среда используется: при создании расчетной сетки и модели; в ходе вычислений при параллельной адаптации сетки к решению и к геометрии области; для визуализации полученных сеток и результатов. Рассмотрим применение этой программной среды для параллельного построения больших неструктурированных сеток, которое является первым этапом решения задачи проекционно-сеточными методами, во многом определяющим дальнейшие вычисления. При параллельной реализации методов декомпозиции области построение сетки необходимо рассматривать, с одной стороны, как первоначальное распределение данных и вычислительной нагрузки процессоров, а с другой — как основу для алгоритмов, построенных на распределенных сетках и связанных с модификацией сетки (перестроение, динамическая балансировка нагрузки и пр.).

Актуальность параллельного построения сетки обусловлена возможностью распределения затрат оперативной памяти (особенно для больших трехмерных неструктурированных сеток) и сокращения затрат на передачу данных между последовательным генератором сетки и параллельным приложением.

Традиционно построение и разделение сетки осуществляются независимо друг от друга, но с выполнением операций ввода/вывода и с накладными расходами на пересылку данных, которые порой превышают 80 % полного времени построения, разделения и размещения большой сетки на распределенной памяти параллельной системы. Новый подход, объединяющий построение и разделение сетки, минимизирует ввод/вывод и накладные расходы пересылок данных, устраняя избыточные операции. Результаты показывают, что такой подход применительно к неструктурированным сеткам дает ускорение в несколько раз по сравнению с традиционным.

Методы параллельного построения неструктурированных сеток можно классифицировать по:

1) предварительному разделению расчетной области на подобласти: разделение области на уровне геометрии и разделение грубой сетки на подсетки;

2) используемому последовательному методу построения: триангуляции Делоне, метод сжатия границ и т.д.;

3) очередности построения: сначала триангулируются окрестности границ между подобластями сетки (разделители), затем подобласти; подобласти и далее разделители; сетка строится и разделяется одновременно во всей области.

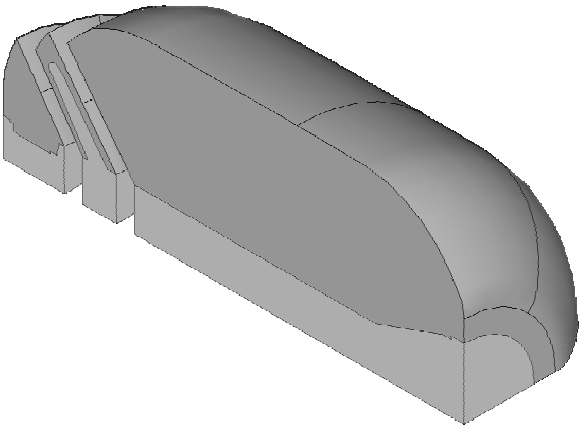


Рис. 9. Геометрическое деление области

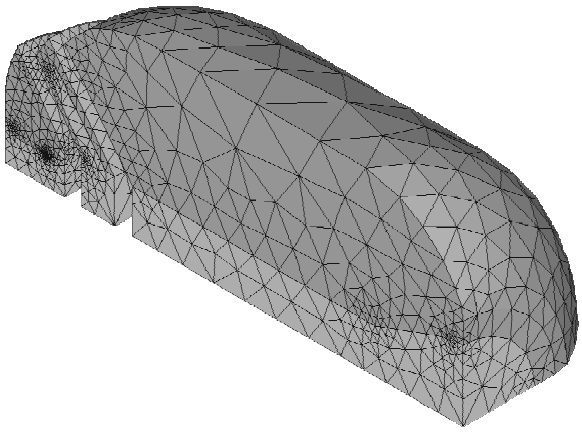


Рис. 10. Сетка, построенная для шести геометрических подобластей

При построении сеток используются, как правило, распараллеливание по данным и распределение вычислений по процессорам [16, 17]. Область разделяется на подобласти без перекрытия подобластей так, чтобы построение в подобластях выполнялось на различных процессорах сбалансированно. Далее будет рассмотрено параллельное построение сетки, основанное на разделении геометрической модели области и грубой триангуляции.

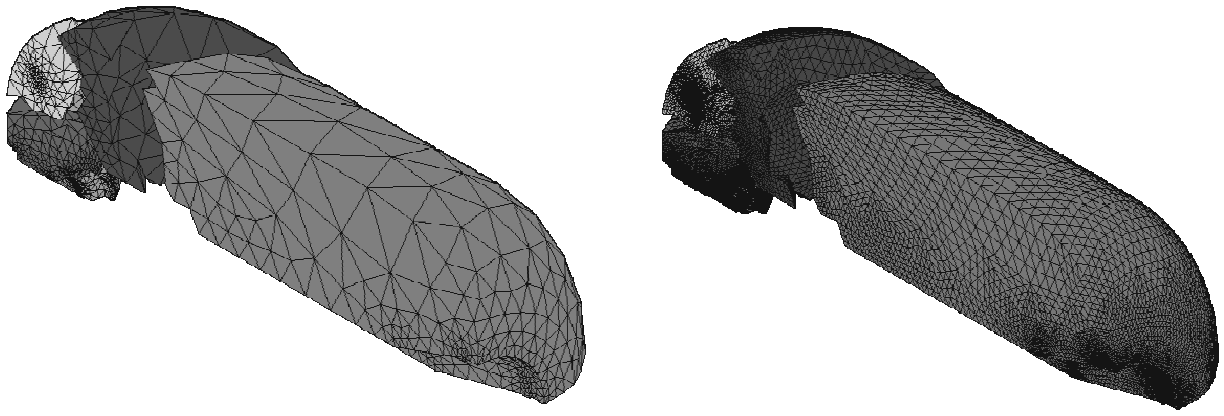
7.1. Параллельное построение сетки на геометрически разделенной области. Геометрия

области является критической частью построения сетки и других этапов моделирования, основанных на дискретизации. Прежде всего это относится к трехмерным сеткам. Алгоритмы и программы построения сеток сами должны непосредственно взаимодействовать с твердотельной геометрической моделью, в которой тела обычно строятся. Выделим общие принципы, позволяющие рационально построить и разделить геометрическую модель: необходимо стремиться к представлению области набором из минимально возможного числа разнотипных подобластей (если подобласти, входящие в состав рассматриваемой области, одинаковы в геометрическом и физическом смысле, то вычисления в численных алгоритмах значительно упрощаются); топология и размеры подобластей должны соответствовать возможностям параллельных сеточных генераторов; следует стремиться к введению иерархической системы подобластей; подобласти необходимо выбирать таким образом, чтобы вычислительные затраты на построение сетки и решения задач в подобластях были сбалансированы, а границы между ними минимальны.

На рис. 9 показана двухсвязная область, которая на основании априорной информации о сгущениях сетки делится в CAD-системе на шесть подобластей. Представленный пример иллюстрирует применение разделения геометрической модели в случае, когда расчетная область состоит из различных по физическим свойствам тел/сред. Используя алгоритм построения сетки на многосвязных областях, описанный выше, на распределенной вычислительной системе построена сетка (рис. 10), состоящая из 17382 тетраэдров, со следующим делением по подобластям (сверху вниз, слева направо; N — число узлов, M — число тетраэдров):

$N_1 = 930$	$N_2 = 1700$	$N_3 = 942$	$N_4 = 439$	$N_5 = 1088$	$N_6 = 290$
$M_1 = 3165$	$M_2 = 5777$	$M_3 = 3225$	$M_4 = 1294$	$M_5 = 3216$	$M_6 = 705$

Сетка распределена существенно неравномерно. Максимальное отклонение от равномерного распределения (далее дисбаланс) для элементов — 99 %. Это приводит к неэффективному использованию вычислительных ресурсов при параллельном счете и требует дополнительной балансировки распределенной сетки. Поэтому построение сетки с использованием разделения геометрии применяется: для получения трехмерных сверхбольших сеток в случае, когда размещение даже грубой сетки в памяти одного вычислительного процессора невозможно; в случаях, когда расчетная область состоит из нескольких связанных твердых тел (такие области характерны при рассмотрении процессов в неоднородных средах и решении сопряженных задач). После построения сетки выполняется перераспределение сеточных объектов в соответствии с выбранной стратегией перераспределения (п. 5.1).



а) б)

Рис. 11. Разделенная и дважды регулярно перестроенная сетка

7.2. Использование параллельного регулярного деления для построения больших сеток. Для получения больших распределенных неструктурированных сеток также используются грубые сетки. Существующая грубая сетка разделяется и параллельно считывается на распределенной вычислительной системе. В грубой сетке элементы имеют достаточно крупный размер, поэтому многие детали исходного тела не отражаются в расчетной сетке, что отрицательно сказывается на точности расчетов. Учет геометрии области при уточняющем (измельчающем) перестроении сетки обеспечивает необходимую точность

представления сеткой исходного тела. Несколькими последовательными запусками алгоритмов измельчения можно добиться достаточного для проведения расчетов соответствия сетки и исходной геометрии.

В зависимости от используемого метода перестроения изменяются величины внутренних углов ячеек. Сохранение качества ячеек обеспечивается регулярным делением всех ячеек, а его улучшение соответственно сменой граней/ребер по лемме Делоне, движением узлов, делением тупых углов, объединением ячеек с малыми углами.

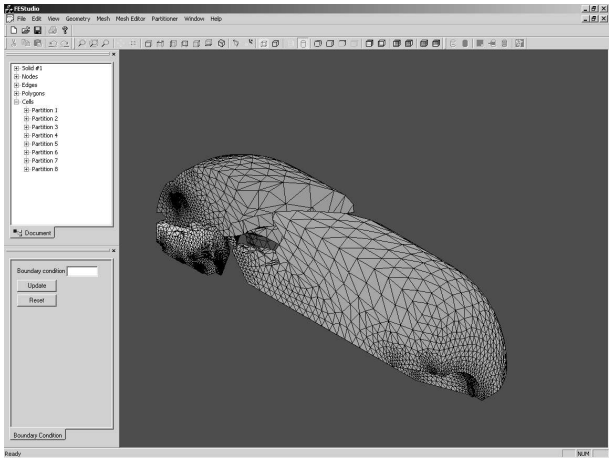


Рис. 12. Визуализация разделенной сетки

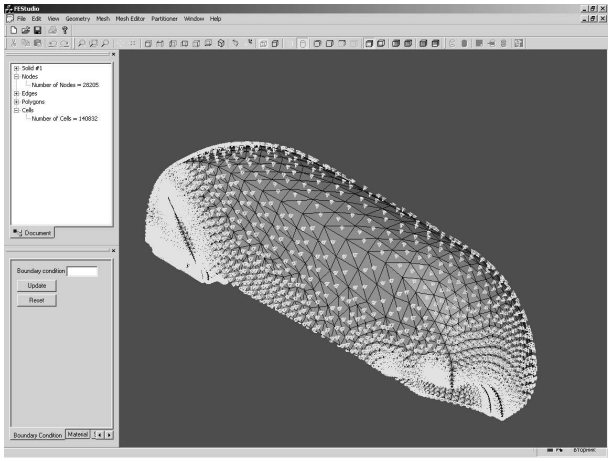


Рис. 13. Граничные условия

Важным аспектом является сохранение соотношений между количествами элементов в подсетках. Алгоритмы, используемые для разделения сетки, стремятся к тому, чтобы после деления количество сеточных объектов (например, ячеек) было приблизительно равным на всех подсетках. При регулярном делении всех ячеек эти соотношения остаются теми же, так что при нескольких последовательных измельчениях отношение разности между числом ячеек в двух произвольных подсетках к числу ячеек всей сетки сохраняется. Таким образом, перестроение сетки не приводит к неравномерной загрузке различных узлов кластера, а наоборот — сохраняет баланс между количествами элементов в разных подсетках. Вместе с увеличением числа общих узлов или ребер разделителя уменьшается отношение числа общих узлов к числу узлов внутри подсетки. Отметим, что согласованность подсеток обеспечивается автоматически.

На рис. 11 приведен пример сетки, разделенной на восемь частей, а затем дважды измельченной с применением параллельного регулярного деления (рис. 11 б); исходная сетка состоит из 17 604 тетраэдров, распределение по подобластям: $M_{\min} = 2136$, $M_{\max} = 2266$ и $N_{\min} = 549$, $N_{\max} = 653$; конечная сетка содержит 1 126 656 тетраэдров, распределение по подобластям: $M_{\min} = 136\,704$, $M_{\max} = 145\,024$ и $N_{\min} = 25\,767$, $N_{\max} = 27\,237$. Дисбаланс вычислительной нагрузки для конечной сетки по числу тетраэдров составляет примерно 3 %, по числу узлов — 3.3 %.

Визуализация разделенной сетки предполагает разнесение подсеток. На рис. 12 представлено рабочее окно FEStudio. Процесс создания расчетной модели предполагает указание граничных условий. Пример задания граничных условий Дирихле показан на рис. 13.

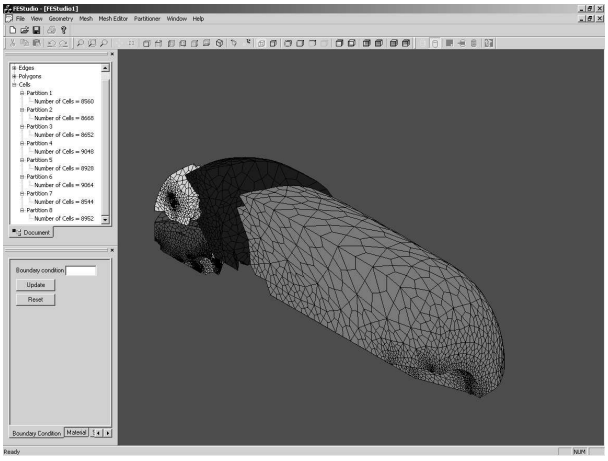


Рис. 14. Разделенная сетка из шестигранников ($M = 70416$, $N = 83263$)

На основе тетраэдральных сеток строятся сетки из шестигранных конечных элементов. Шестигранники строятся делением тетраэдра плоскостями, проходящими через середины его ребер и центр масс. В результате деления тетраэдра получается четыре шестигранника. На рис. 14 представлена сетка из шестигранных элементов, построенная на основе распределенной тетраэдральной сетки (рис. 11 а). Для сетки из шестигранников сохраняется дисбаланс вычислительной нагрузки исходной тетраэдральной сетки по числу элементов (3 %). По числу узлов дисбаланс

составляет 3.2 %.

Заключение. Основная идея данной работы — построение и использование программной среды в параллельных распределенных вычислениях. Построена модель неструктурированной сетки и на ее основе построена модель разделенной и распределенной сетки. Все вычисления на сетке выполняются с учетом геометрической модели, а расчетные данные связываются как с геометрией, так и сеткой.

Построенная среда и визуальное приложение используются в практике вычислений в Институте прикладной механики УрО РАН для решения сопряженных задач механики деформированного твердого тела и газовой динамики.

СПИСОК ЛИТЕРАТУРЫ

1. *Tautges T.J.* CGM: A geometry interface for mesh generation, analysis and other applications // Engineering With Computers. 2001. **17**. 299–314.
2. *Четверушкин Б.Н., Гасилов В.А., Поляков С.В., Якобовский М.В. и др.* Пакет прикладных программ GIMM для решения задач гидродинамики на многопроцессорных вычислительных системах // Матем. моделирование. 2005. **17**, № 6. 58–74.
3. *Ильин В.П.* Геометрическое и функциональное моделирование в задачах математической физики // Вычислительные технологии. 2001. **6**. 315–321.
4. *Копысов С.П., Красноперов И.В., Рычков В.Н.* Объектно-ориентированный метод декомпозиции области // Вычислительные методы и программирование. 2003. **4**, № 1. 176–193.
5. *Копысов С.П., Красноперов И.В., Рычков В.Н.* Реализация объектно-ориентированной модели метода декомпозиции области на основе параллельных распределенных компонентов CORBA // Вычислительные методы и программирование. 2003. **4**, № 1. 194–206.
6. *Ларман К.* Применение UML и шаблонов проектирования. М.: Вильямс, 2001.
7. *Черносвитов А.* Visual C++ 6 и MFC. Курс MCSD для профессионалов. СПб.: Питер, 2000.
8. *Цимбал А.* Технология CORBA для профессионалов. СПб.: Питер, 2001.
9. *Kopyssov S.P., Rychkov V.N., Ponomaryov A.B.* The integration of CAD-systems and generators of unstructured 3D mesh // Proc. of the Workshop on Grid Generation: Theory and Applications. Moscow, 2002. 218–229.
10. Open CASCADE, simulation integrator (http://www.opencascade.com).
11. *Olivier-Gooch C.* GRUMMP Version 0.2. User's Guide. Department of Mechanical Engineering, University of British Columbia. Vancouver, 2001.
12. *Копысов С.П., Новиков А.К.* Параллельные алгоритмы адаптивного перестроения и разделения неструктурированных сеток // Матем. моделирование. 2002. **14**, № 9. 91–96.
13. *Копысов С.П., Пономарев А.Б., Рычков В.Н., Зубцовский С.Н.* Расчетные неструктурированные сетки для распределенных вычислений // Сибирская школа-семинар по параллельным вычислениям. Томск: Изд-во Том. ун-та, 2005. 19–25.
14. *Копысов С.П., Пономарев А.Б., Рычков В.Н.* Открытое визуальное окружение для взаимодействия с геометрическими ядрами, генерации/перестроения/разделения сеток и построения расчетных моделей // Прикладная геометрия, построение расчетных сеток и высокопроизводительные вычисления / Ред. Ю.Г. Евтушенко, М.К. Керимов, В.А. Гаранжа. М.: ВЦ РАН, 2004. **2**. 154–164.
15. *Rumsey C.L., Poirier D., Bush R.H., Towne C.E.* CFD General Notation System. A User's Guide to CGNS. Overview and Entry-Level Document. Mid-Level Library (http://www.cgns.org).
16. *Löhner R.* A parallel advancing front grid generation scheme // AIAA-00-1005. 2000.
17. *Chrisochoides N.* Parallel mesh generation // Numerical Solution of Partial Differential Equations on Parallel Computers. Berlin: Springer-Verlag, 2005.

Поступила в редакцию
22.11.2006