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

Препринты ИПМ • Препринт № 134 за 2018 г.

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

Суков С.А.

Метод сжатия топологии тетраэдральных сеток

Рекомендуемая форма библиографической ссылки: Суков С.А. Метод сжатия топологии тетраэдральных сеток // Препринты ИПМ им. М.В.Келдыша. 2018. № 134. 22 с. doi:10.20948/prepr-2018-134
URL: http://library.keldysh.ru/preprint.asp?id=2018-134

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

С.А.Суков

Метод сжатия топологии тетраэдральных сеток

Москва — 2018

Суков С.А.

Метод сжатия топологии тетраэдральных сеток

В работе рассматривается проблема сжатия топологии тетраэдральных сеток. Приводится описание двухшагового алгоритма компрессии четверок вершин тетраэдров и особенностей его программной реализации. Представлены результаты сжатия десяти дискретных моделей, содержащих до $134 \cdot 10^6$ элементов.

Ключевые слова: тетраэдральные сетки, сжатие данных

Sergey Alexandrovich Sukov

A compression method for a topology of tetrahedral meshes

This paper contains the description of sequential two-step tetrahedral mesh topology compression method and its software implementation. The results of compression of ten meshes containing up to $134 \cdot 10^6$ elements are presented.

Key words: tetrahedral meshes, data compression

Оглавление

1. Введение

Тетраэдральные сетки относятся к наиболее универсальному типу дискретных моделей трехмерных объектов со сложной формой границ, поэтому часто используются при проведении численных расчетов задач математической физики. Описание геометрии средней по размеру сетки, содержащей пятьдесят миллионов элементов, занимает гигабайт дискового пространства. А следовательно, для повышения эффективности копирования и хранения дискретной модели на удаленном сервере высокопроизводительной системы целесообразно применять методы компрессии данных.

В исходном виде описание тетраэдральной сетки (например, формат [1]) состоит из трех блоков данных: координатный блок, топологический блок и так называемая раскраска границы расчетной области. Узлам и элементам сетки присваиваются целочисленные индексы $id_v = 0,1,\dots,N_v - 1$ и $id_e = 0,1,\dots,N_e - 1$, где N_v и N_e – общее число вершин и тетраэдров соответственно. Координатный блок содержит тройки координат сеточных вершин, которые упорядочиваются по возрастанию индексов. Топологический блок состоит из четверок вершин тетраэдров, также упорядоченных по возрастанию индексов сеточных элементов. С точки зрения задачи вычисления геометрических параметров элемента порядок перечисления индексов его вершин не имеет значения, но может быть связан с форматом представления границы расчетной области. Раскраска содержит в себе метки находящихся на границе узлов и/или треугольников. Во время расчета каждой метке ставится в соответствие один из типов граничного условия. Ниже предполагается, что раскраска задается исключительно для поверхностных треугольников и состоит из N_b троек целых чисел: индекс тетраэдра, номер соответствующей треугольнику грани, метка границы.

Наибольший объем данных (порядка 75 %) занимает топологический блок. Коэффициент упаковки четверок индексов вершин методом сжатия без потерь (стандартным методом) находится в пределах 3 - 3.5 раз. Существенно лучшие результаты демонстрируют специализированные методы [2]. Их использование приводит к изменению индексации узлов, тетраэдров и порядка перечисления вершин в индексных списках. Поэтому, формально, специализированные методы не относятся к классу алгоритмов кодирования без потерь. Тем не менее, после сжатия и разархивирования данных топология сеточных элементов и отношений их связности полностью восстанавливается.

Упаковка четверок вершин тетраэдров делится на два этапа: базовое сжатие и кодирование потоков данных. Результат работы базового алгоритма (алгоритма первичного кодирования) состоит в переходе от явного представления элементов к описанию процедуры их последовательной генерации. Каждый новый тетраэдр задается коротким кодом и дополнительными параметрами одного из известных вариантов топологической связности между ним и ранее упакованными элементами. Далее следует

кодирование полученных таким образом потоков данных методом сжатия без потерь.

Известные алгоритмы сжатия координат, как правило, основываются на огрублении и квантовании исходных величин с контролируемой потерей точности. То есть гарантируется, что сдвиг сеточных вершин в результате огрубления координат не приведет к искажению топологии тетраэдров. В случае одновременной упаковки данных топологического и координатного блоков вместо координат вершины часто хранятся координаты вектора между ее фактической позицией и аналитически предсказанной точкой в пространстве. Например, позиция четвертой вершины тетраэдра регулярной сетки сравнительно точно определяется по расположению трех вершин противолежащей грани. Близкие к нулю целочисленные координаты векторов в дальнейшем эффективно упаковываются методом компрессии без потери точности. В случае сильно анизотропной сетки в сжатом виде хранятся координаты вектора между центром масс основания тетраэдра и четвертой вершиной. Координаты вектора вычисляются в связанной с основанием локальной системе координат. Центр локальной системы координат совпадает с центром масс треугольника, а ось абсцисс сонаправлена с вектором его площади. Считается, что данный алгоритм преобразования координат позволяет минимизировать значения y,z-компонент вектора.

Раскраска границы занимает незначительный объем дискового пространства. После приведения данных в соответствие с измененной индексацией сеточных примитивов блок упаковывается стандартным методом.

В настоящей работе представлены один из вариантов последовательного алгоритма сжатия топологии тетраэдральных сеток и его программная реализация. Проблема компрессии координат сеточных узлов не рассматривается. Координаты записываются в исходном виде. Приводятся результаты упаковки топологического блока регулярных и сгущающихся дискретных моделей, содержащих до 130 миллионов элементов.

2. Компрессия топологического блока

Раздел содержит описание алгоритма двухэтапного сжатия топологического блока данных тетраэдральной сетки и особенностей его программной реализации. За основу предлагаемого подхода взят метод, опубликованный в работе [3]. При схожей в целом логике процедуры кодирования данных алгоритмы отличаются выбором методов обхода тетраэдров, индексации сеточных примитивов и сжатия потоков данных.

2.1. Базовый алгоритм

Логика алгоритма первичного кодирования индексных списков совпадает с логикой генерации тетраэдральных сеток методом продвижения фронта [4]. В процессе сжатия элементы делятся на два подмножества: упакованные

тетраэдры и тетраэдры, ожидающие обработки. Фронту в данном случае соответствует триангулированная граница подмножества кодированных элементов. В начальный момент оно состоит из единственного тетраэдра, четыре грани которого формируют стартовый фронт. Далее в упакованное множество по одному добавляются тетраэдры, примыкающие к фронту одной или несколькими гранями. Добавление тетраэдра в кодированную область сопровождается соответствующей модификацией фронта. Процедура повторяется до завершения обработки всех сеточных ячеек. Промежуточный этап работы алгоритма проиллюстрирован на рис. 1. Текущая конфигурация фронта имеет заливку голубого цвета. Красным цветом выделен очередной кодируемый тетраэдр. Серым цветом обозначена триангуляция границы расчетной области.

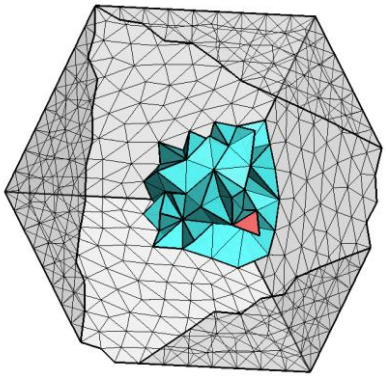


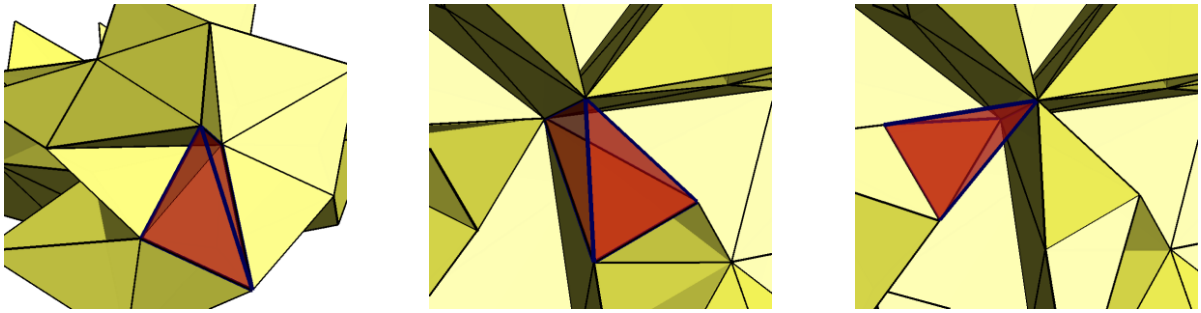
Рис. 1. Промежуточный этап работы алгоритма первичного кодирования.

Выбор активной грани фронта или основания следующего пакуемого тетраэдра происходит автоматически. Сжатый формат представления данных ограничивается информацией о том, каким образом построить тетраэдр на известном основании.

Кодируемый тетраэдр может опираться на фронт одной или несколькими гранями. В первом случае на фронт добавляются три боковые грани элемента, а его основание удаляется с фронта. Таким образом, размерность фронта увеличивается на два треугольника. Если при этом четвертая вершина тетраэдра отсутствует на фронте, то ее координаты записываются в сжатое представление вместе с сеточным элементом. При наличии нескольких общих граней у тетраэдра и фронта его размерность остается без изменений (две общие грани) или же сокращается (три или четыре общие грани). В процессе упаковки на фронте появляются и затем исчезают все внутренние сеточные грани.

Кодирование очередного тетраэдра без добавления новых вершин может сопровождаться появлением недвусвязных (non-manifold) конфигураций или ребер фронта, на которые одновременно опирается несколько пар граней (рис.2). Недвусвязная конфигурация возникает в ситуациях, когда на фронте уже есть общее ребро двух отсутствующих на нем боковых граней тетраэдра.

Алгоритм первичного сжатия гарантирует корректное завершение процедуры, если сетка в целом удовлетворяет критерию связности элементов по граням.



а) добавление на ребро второй пары граней

б) добавление на ребро третьей пары граней

в) добавление на ребро четвертой пары граней

Рис. 2. Примеры недвусвязных ребер.

Продвижение фронта от кодированного к неупакованному тетраэдру допускается при условии существования между ними общей грани. Поэтому сетки, состоящие из несвязных по граням подобластей (рис. 3), не могут быть упакованы обсуждаемым методом. Для их обработки необходима модификация алгоритма, предусматривающая случай компрессии нескольких тетраэдральных доменов с общими сеточными вершинами или ребрами.

а) два сеточных домена с общими ребрами

б) два сеточных домена с общей вершиной

Рис. 3. Примеры несвязных конфигураций расчетных областей.

Сжатое представление сетки делится на три потока данных: координатный поток, управляющий поток и индексный поток. Управляющий поток состоит из кодирующих символов $C_{P=0,1,\dots,6}$ (3 бита), обозначающих один из семи вариантов поведения алгоритма после выбора активной грани:

- C_0, C_1, C_2 – кодирование тетраэдра, опирающегося на фронт двумя и более гранями;
- C_3 – кодирование тетраэдра с добавлением новой вершины;
- C_4, C_5 – кодирование опирающегося на фронт только основанием тетраэдра без добавления новой вершины;
- C_6 – пропуск грани.

Первым элементом кодированной области выбирается тетраэдр с максимальным числом граней на глобальной границе. Вершинам тетраэдра присваиваются индексы от 0 до 3. Тройки их координат записываются в начало координатного потока в порядке возрастания новых индексов.

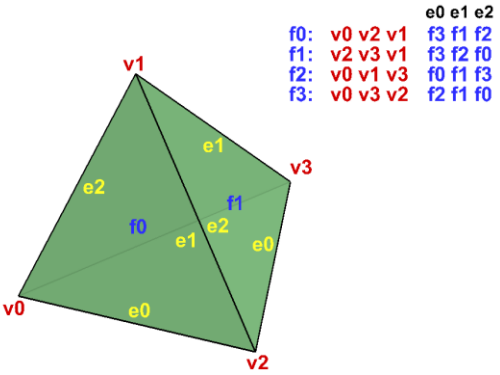
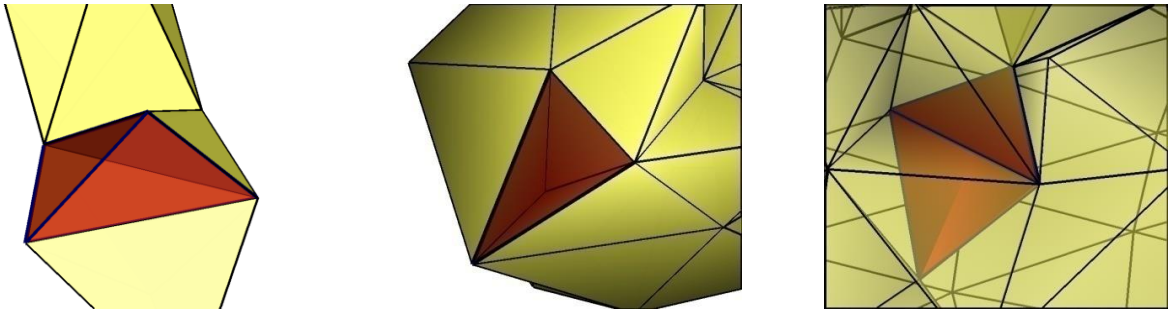


Рис. 4. Представление начальной конфигурации фронта.

Топология и граф связности фронтальных треугольников описываются ориентированными списками вершин граней и тройками индексов соседних по ребрам треугольников. Ориентация вершин, локальная индексация ребер и порядок перечисления соседей соответствуют обходу вершин треугольника в направлении по часовой стрелке со стороны множества кодированных тетраэдров. На рис. 4 показан пример представления начальной конфигурации фронта. Тип внутренней индексации фронтальных треугольников (индексы f_m) выбирается исходя из особенностей применяемого алгоритма хранения динамических списков.



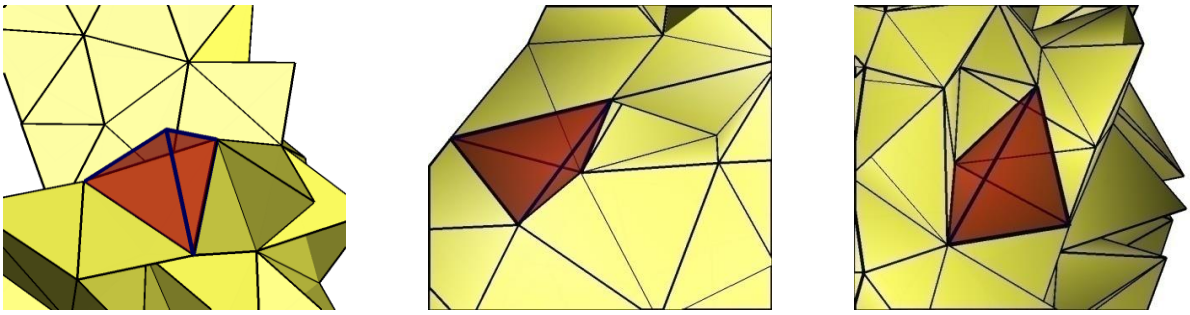
а) две грани на фронте б) три грани на фронте в) четыре грани на фронте

Рис. 5. Кодирование тетраэдра с двумя и более гранями на фронте.

Очередность использования треугольников в роли основания кодируемого тетраэдра эквивалентна последовательности добавления граней на фронт. Тетраэдр, две грани которого находятся на фронте (рис. 5 а), кодируется локальным индексом общего ребра у активного треугольника (коды C_0, C_1, C_2 для ребер e_0, e_1 и e_2 соответственно). Индекс четвертой вершины определяется аналитически. Таким образом, целочисленный индекс вершины заменяется

коротким управляющим символом, а размер кодированного представления тетраэдра составляет 3 бита. В случае, когда фронту одновременно принадлежат несколько боковых граней (рис. 5 б и 5 в), в управляющий поток записывается код любого из подходящих ребер основания.

Топология элемента, примыкающего к фронту исключительно основанием (рис. 6), задается одним из трех вариантов кодирования типа и позиции четвертой вершины. Символ C_3 указывает на случай добавления нового узла. Тройка координат вершины записывается в координатный поток. Новый индекс присваивается узлу автоматически в соответствии с очередностью добавления вершин.



а) кодирование с добавлением нового узла

б) кодирование с указанием локального индекса

в) кодирование с указанием глобального индекса

Рис. 6. Кодирование тетраэдра с одной гранью на фронте.

Индекс уже находящейся на фронте четвертой вершины тетраэдра записывается в индексный поток. При этом глобальный индекс вершины по возможности подменяется ее локальным индексом (id_v^l). Этот индекс вычисляется автоматически в соответствии с фиксированным алгоритмом обхода вершин фронта в окрестности основания. Если значение локального индекса удовлетворяет условию $id_v^l < 2^4$, то в управляющий поток записывается код C_4, а в индексный поток – id_v^l (4 бита). В противном случае в управляющий поток добавляется код C_5, а в индексный поток – глобальный индекс четвертой вершины тетраэдра (32 бита).

Пример работы алгоритма локальной индексации вершин показан на рис. 7. Начальная конфигурация окрестности включает активный треугольник (красная заливка) и его соседей по граням (желтая заливка). Вершины этих треугольников не индексируются, так как по постановке задачи не могут быть недостающим узлом тетраэдра. Далее в окрестность последовательно добавляются треугольники, ребра которых принадлежат ее границе. Если противолежащая ребру вершина треугольника не была ранее проиндексирована, то ей присваивается следующий по порядку локальный индекс. Расширение окрестности продолжается до тех пор, пока число проиндексированных узлов не превысит установленного ограничения.

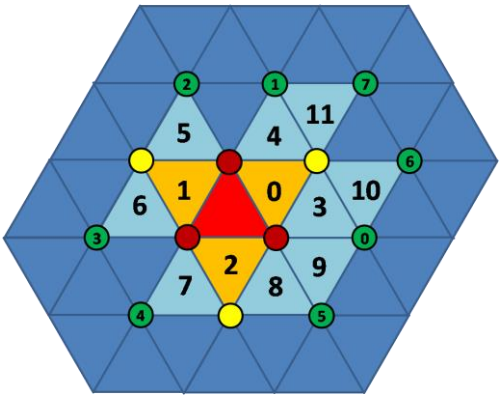


Рис. 7. Пример локальной индексации узлов в окрестности базового треугольника с ограничением $id_v^l < 2^3$.

Код C_6 обозначает активный треугольник на границе расчетной области. Такие треугольники не обрабатываются и остаются на фронте до окончания упаковки всех тетраэдров.

Блок-схемы алгоритмов первичного кодирования тетраэдров и соответствующего алгоритма декодирования показаны в приложении 1. Поведение декодера отличается отсутствием анализа вариантов взаимного расположения тетраэдра и фронта, а также выборочным преобразованием локальных индексов в глобальные индексы для отдельно стоящих на фронте четвертых вершин.

Финальная конфигурация фронта совпадает с границей расчетной области. Для каждого из ее треугольников известен индекс тетраэдра, гранью которого он является. И поскольку кодер и декодер выполняют одну и ту же последовательность действий, то индексация треугольников в обоих случаях совпадает. Таким образом, описание раскраски границ можно ограничить массивом N_b целочисленных меток поверхностей, упорядоченных по возрастанию индексов треугольников. Индексы тетраэдров и номера граней в исходном формате записи граничного блока восстанавливаются аналитически.

Наилучшее значение плотности записи данных топологического блока в потоковом виде будет равно $T^C = 3L_C/N_e$ бит на один тетраэдр (bpt), где $L_C = N_e - 1 + N_b$ – длина управляющего потока. Данная оценка соответствует идеальному случаю нулевой длины индексного потока, когда $N_{C_4} + N_{C_5} = 0$. Здесь и далее обозначение N_{C_P} используется для указания общего числа кодов C_P в управляющем потоке: $L_C = \sum_P N_{C_P}$.

На основании результатов проведенных вычислительных экспериментов можно утверждать, что доля символов кодирования тетраэдров с явным указанием индекса четвертой вершины составляет $\gamma_{L_C}^{C_{4,5}} = \frac{N_{C_4}+N_{C_5}}{L_C} \leq 0.15$. Внутри индексного потока доля случаев записи глобальных индексов равна $\gamma_{C_{4,5}}^{C_5} = \frac{N_{C_5}}{N_{C_4}+N_{C_5}} \leq 0.2$. Отношение числа граничных треугольников расчетной сетки к числу ее объемных элементов, как правило, ограничивается сверху

значением $\frac{N_b}{N_e} \leq 0.15$. Максимальные значения перечисленных параметров обычно относятся к сеткам небольшой ($N_e < 10^5$) размерности. В таком случае плотность записи данных должна находиться в пределах $T^C = \frac{3L_C+4N_{C_4}+32N_{C_5}}{N_e} \leq 5.11$ bpt.

Полученная выше теоретическая оценка для плотности записи данных T^C соответствует коэффициенту сжатия $K^C = 25.1$ раза относительно записи данных в формате четверок целочисленных индексов (128 bpt), что уже многократно превосходит коэффициент сжатия алгоритмов кодирования без потерь. Однако следует учесть, что повышение эффективности упаковки во многом связано с преобразованием индексации сеточных узлов и элементов. Восстановление исходной индексации вершин и тетраэдров требует дополнительной записи целочисленных массивов суммарной длиной $N_v + N_e$ ячеек. А для восстановления начального порядка перечисления вершин необходимо хранить еще один массив длиной N_e байт. Таким образом, применение обсуждаемого подхода имеет практический смысл, когда кодирование топологии происходит до начала проведения вычислительных экспериментов или же сопровождается переупорядочиванием данных в массивах значений сеточных функций.

2.2. Сжатие потоков

Сформированные по итогам работы базового алгоритма кодирования потоки данных далее по отдельности упаковываются одним или комбинацией методов сжатия без потерь (рис. 8). Модификация координатного блока данных ограничивается упорядочиванием троек координат вершин в соответствии с их новой индексацией.

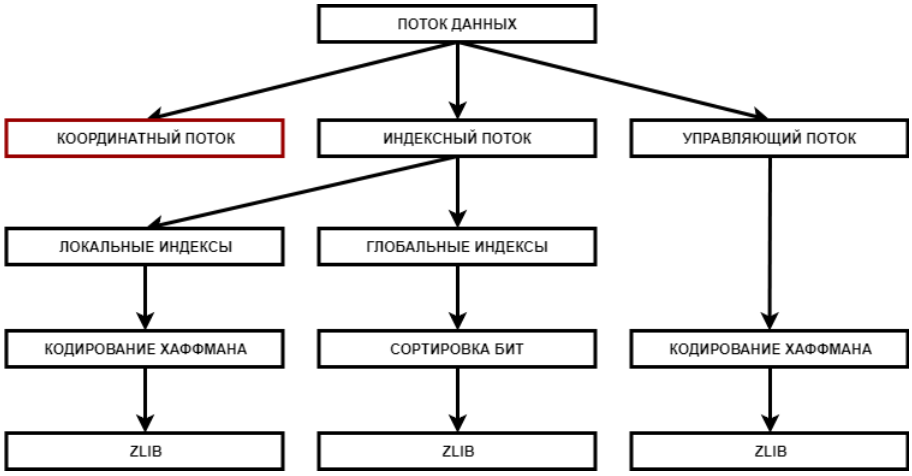


Рис. 8. Блок-схема процедуры кодирования потоков данных.

2.2.1. Оптимизация и сжатие управляющего потока

Длина управляющего потока многократно превосходит длину индексного потока. И, следовательно, коэффициент сжатия массива управляющих кодов имеет существенное влияние на значение коэффициента сжатия топологического блока в целом. Управляющий поток сжимается с применением комбинации двух подходов: метода префиксного кодирования Хаффмана [5] и процедур библиотеки подпрограмм zlib [6].

Идея алгоритма Хаффмана состоит в том, что фиксированные по длине представления символов C_P меняются на коды переменной длины. Символам с большей вероятностью появления в потоке ставится в соответствие короткий код. То есть существует вариант повышения эффективности сжатия за счет увеличения вероятностей появления в потоке ограниченного набора символов.

Количество символов C_3 всегда равно числу вершин сетки за вычетом четверки узлов стартового тетраэдра $N_{C_3} = N_v - 4$. Число кодов пропуска C_6 соответствует числу треугольников на границе расчетной области $N_{C_6} = N_b$. Вероятности появления в потоке кодов C_4 и C_5 зависят от стратегии выбора активного треугольника. Известные алгоритмы обхода имеют эвристический характер и, как правило, ставят целью минимизировать размер фронта. Поэтому остается один вариант оптимизации управляющего потока, который заключается в повышении вероятности появления какого-либо из символов (C_0, C_1 или C_2) среди случаев кодирования тетраэдра с использованием двух и более треугольников фронта.

Индексация ребер треугольника происходит в момент его добавления на фронт. Локальный индекс e_0 присваивается общему с основанием кодируемого тетраэдра ребру. Индексы e_1 и e_2 выдаются оставшимся ребрам в порядке их обхода по часовой стрелке. На практике данный алгоритм индексации характеризуется соотношением кодов $\gamma_{\mathrm{C}_{0,1,2}}^{\mathrm{C}_0} = \frac{N_{C_0}}{N_{C_0}+N_{C_1}+N_{C_2}} > 0.7$. С целью увеличения доли кода C_0 внутри группы трех символов в алгоритм кодирования добавляется процедура повторной индексации ребер с геометрическим предиктором.

Повторная индексация ребер активного треугольника происходит непосредственно в момент кодирования тетраэдра кодами из группы C_0, C_1, C_2. Предполагается, что тетраэдру должна принадлежать грань с минимальным среди соседей основания двугранным углом. Индекс e_0 присваивается общему с этой гранью ребру активного треугольника. Описанный прием повышает долю кода C_0 до 90%.

Преобразованный с использованием алгоритма Хаффмана управляющий поток далее сжимается средствами zlib, что при работе с большими сетками позволяет еще на 10-15% сократить размер записываемых данных.

2.2.2. Сжатие индексного потока и раскраски

Изначально внутри индексного потока чередуются локальные (4 бита) и глобальные (32 бита) индексы недостающих вершин тетраэдров. Для улучшения коэффициента компрессии поток делится на две составляющие, после чего каждая из них обрабатывается по отдельности.

Алгоритм упаковки массива локальных индексов идентичен алгоритму компрессии управляющего потока. Он состоит в кодировании данных методом Хаффмана с последующим сжатием результата средствами библиотеки zlib.

Массив глобальных индексов упаковывается функциями zlib после сортировки. В упорядоченном массиве данные группируются по битам, а биты внутри групп располагаются в порядке перечисления глобальных индексов. Целесообразность преобразования данных следует из особенностей алгоритмов индексации узлов и движения фронта. Равномерное по всем направлениям движение фронта сопровождается последовательной индексацией сеточных вершин. Отсюда можно предположить, что индексы в потоке будут упорядочены преимущественно по возрастанию. И, следовательно, в результате побитовой сортировки массив данных трансформируется к более предпочтительному для сжатия виду чередования продолжительных последовательностей одинаковых байт.

Аналогичный подход с сортировкой бит и сжатием инструментами zlib используется для упаковки массива целочисленных признаков поверхностных треугольников. Число выделенных граничных поверхностей обычно не превышает 10-20 единиц. То есть как минимум 27 нулевых бит целочисленных признаков после сортировки будут сжаты с максимальной эффективностью.

2.3. Ресурсоемкость, быстродействие и особенности программной реализации

Ресурсоемкость и быстродействие алгоритмов сжатия и разархивирования определяются в основном внутренними форматами представления сетки и фронта. Главное направление оптимизации алгоритма выбирается с учетом области его применения. Так, например, в задачах визуализации удаленных данных целью довольно часто ставится повышение быстродействия кодера на ограниченном объеме оперативной памяти [7], что приводит к ухудшению коэффициента компрессии данных. В настоящей работе предполагается использование алгоритма для сжатия топологии расчетных сеток. Поэтому акцент делается на максимизации коэффициента сжатия в отсутствие жестких ограничений на ресурсоемкость и быстродействие программной реализации.

Для достижения приемлемого времени работы кодера необходимо хранить в оперативной памяти дуальный граф сетки, то есть граф связей тетраэдров по граням. Размерность дуального графа эквивалентна размерности массива четверок вершин тетраэдров ($4N_e$ целых чисел). Инициализация графа сопровождается распределением дополнительной памяти аналогичного

размера. Далее для хранения потоков данных, топологии фронта и прочих вспомогательных массивов и переменных запрашиваются уже меньшие объемы оперативной памяти. Таким образом, ресурсоемкость процедуры кодирования можно грубо оценить как $32N_e$ байт. Алгоритм декодирования характеризуется меньшей ресурсоемкостью, так как не учитывает связи тетраэдров по граням. Тем не менее при необходимости дуальный граф может быть восстановлен без распределения вспомогательных массивов и заметной потери производительности.

Актуальная конфигурация фронта хранится в виде двусвязного списка. Информационная часть элемента списка содержит структуру с параметрами одного фронтального треугольника. Начальная размерность массива данных выбирается равной размерности (числу треугольников) глобальной границы и при необходимости динамически увеличивается во время работы программы. Локальный индекс фронтального треугольника соответствует «физическому» индексу ячейки памяти, где хранится соответствующий элемент списка. То есть локальная индексация треугольников в общем случае не является непрерывной.

Для ускорения тестирования и обработки недвусвязных конфигураций вводится дополнительный массив структур со списками принадлежности фронтальных вершин треугольникам. Перестроение фронта, сочетающееся с появлением недвусвязной конфигурации, происходит без вычисления геометрических параметров. Корректная обработка случаев добавления на ребро третьей, четвертой и т.д. пар граней включает в себя определение двугранных углов.

Повторная индексация ребер при основании правильного тетраэдра сводится к вычислению и сортировке близких по значению двугранных углов. Поэтому реализация предиктора должна быть устойчива к различиям результатов вызова (вплоть до ошибок округления) стандартных математических функций в случае компиляции кода на разных платформах.

Каждый шаг алгоритма первичного кодирования с добавлением нового тетраэдра в потоковый формат имеет жесткую зависимость по данным от выполненных ранее шагов. Таким образом, распараллеливание базового алгоритма на уровне кодирования тетраэдров исключается. Незначительное повышение быстродействия программной реализации может быть получено только за счет параллельного выполнения витков внутренних циклов.

3. Результаты вычислительных экспериментов

В данном разделе представлены результаты сжатия десяти тетраэдральных сеток. С точки зрения проблемы упаковки топологического блока анизотропия тетраэдральной сетки, теоретически, не должна оказывать влияния на коэффициент сжатия топологии. Поэтому большинство дискретных моделей имеют регулярную структуру, то есть постоянный шаг по пространству. Условные обозначения и характерные параметры сеток приводятся в табл. 1.

Максимальная размерность сетки составляет $134 \cdot 10^6$ элементов при среднем значении на уровне $25 \cdot 10^6$ тетраэдров.

Таблица 1

Обозначение	Параметры сетки				
	N_v	N_e	N_b	$N_e : N_v$	$N_b : N_e$
NMF	3173	13331	3698	4.2	0.28
PRISM	53835	281491	27196	5.23	0.10
CUBE	190452	1083344	39332	5.69	0.04
CUBED	1483913	8666752	157328	5.84	0.02
SR002	365386	2103613	46368	5.76	0.02
SR016	2857568	16828904	185472	5.89	0.01
SR130	22636775	134631232	741888	5.95	0.01
SSF001	202017	1194400	11680	5.91	0.01
SSF009	1604273	9555200	46720	5.96	< 0.01
SSF076	12787105	76441600	186880	5.98	< 0.01

Сетка NMF представляет собой грубую дискретизацию объема многогранника с недвусвязной границей и используется в качестве примера возможности корректной обработки недвусвязной геометрии. Сетки CUBE и CUBED строятся внутри единичного куба и различаются пространственным шагом. Расчетная область для сетки PRISM выглядит как параллелепипед с вырезанной треугольной призмой. Последовательности сгущающихся сеток SSF и SR заполняют объем сферы. Сетки из группы SR характеризуются фиксированной плотностью узлов в пространстве. Плотность узлов сеток SSF линейно возрастает от границы к центру масс расчетной области. Визуализация расчетных областей тестовых сеток приводится в приложении 2.

3.1. Алгоритм первичного кодирования

Доли различных групп символов в управляющем потоке представлены в табл. 2. Здесь и далее средние значения параметров соответствуют средним арифметическим значениям десяти величин, а верхним подчеркиванием выделяются характеристики кодирования с геометрическим предиктором.

Как было отмечено выше, параметры $\gamma_{L_C}^{\mathrm{C}_3}$ и $\gamma_{L_C}^{\mathrm{C}_6}$ всегда имеют фиксированные значения, которые зависят от соотношений между числами узлов, тетраэдров и граничных треугольников сетки. Эффективность алгоритма первичного кодирования характеризуется близостью к нулю $\gamma_{L_C}^{\mathrm{C}_{4,5}}$. Доля случаев кодирования тетраэдра с явным указанием индекса четвертой вершины меняется в пределах от 3 до 6%. Значения $\gamma_{L_C}^{\mathrm{C}_{4,5}}$ и $\bar{\gamma}_{L_C}^{\mathrm{C}_{4,5}}$ по факту примерно равны, хотя последовательности кодирования тетраэдров с использованием геометрического предиктора и без него, естественно, отличаются. Повторная индексация ребер активных треугольников повышает долю кода C_0 в группе C_0,

C_1, C_2 в среднем с 79 до 97%. Таким образом, на символы C_0 приходится более 70% от общего числа кодов во всем управляющем потоке.

Таблица 2

Сетка	$\gamma_{L_C}^{C_{0,1,2}}$	$\gamma_{L_C}^{C_3}$	$\gamma_{L_C}^{C_{4,5}}$	$\gamma_{L_C}^{C_6}$	$\gamma_{C_{0,1,2}}^{C_0}$	$\bar{\gamma}_{C_{0,1,2}}^{C_0}$
NMF	0.55	0.19	0.04	0.22	0.82	0.97
PRISM	0.69	0.17	0.05	0.09	0.81	0.97
CUBE	0.74	0.17	0.06	0.04	0.83	0.97
CUBED	0.76	0.17	0.05	0.02	0.78	0.97
SR002	0.75	0.17	0.06	0.02	0.84	0.97
SR016	0.77	0.17	0.05	0.01	0.79	0.97
SR130	0.78	0.17	0.04	0.01	0.75	0.97
SSF001	0.77	0.17	0.05	0.01	0.79	0.97
SSF009	0.78	0.17	0.05	0.01	0.75	0.97
SSF076	0.80	0.17	0.03	<0.01	0.72	0.97
Среднее	0.74	0.17	0.05	0.04	0.79	0.97

3.2. Компрессия управляющего потока

В табл. 3 приводятся коэффициенты сжатия управляющего потока (K_M^C) тремя различными методами: прямое сжатие zlib ($M = Z$), метод Хаффмана ($M = H$) и комбинированный метод ($M = H\&Z$).

Таблица 3

Сетка	Однократная индексация			Индексация с предиктором		
	K_Z^C	K_H^C	$K_{H\&Z}^C$	$\bar{K}_Z^C$	$\bar{K}_H^C$	$\bar{K}_{H\&Z}^C$
NMF	1.22	1.39	1.40	1.36	1.67	1.71
PRISM	1.31	1.49	1.52	1.57	1.89	2.01
CUBE	1.40	1.61	1.66	1.73	2.00	2.20
CUBED	1.39	1.60	1.65	1.84	2.09	2.37
SR002	1.45	1.68	1.74	1.83	2.08	2.36
SR016	1.41	1.63	1.69	1.89	2.13	2.45
SR130	1.40	1.62	1.67	1.99	2.20	2.59
SSF001	1.40	1.63	1.68	1.85	2.13	2.42
SSF009	1.39	1.62	1.67	1.97	2.20	2.56
SSF076	1.42	1.61	1.68	2.12	2.25	2.74
Среднее	1.38	1.59	1.64	1.82	2.06	2.34

Вне зависимости от применяемого метода сжатия коэффициент упаковки для потока с повышенной долей символа C_0 ожидаемо оказывается выше $\bar{K}_M^C > K_M^C$. Сравнение эффективности использования алгоритмов Хаффмана и zlib по отдельности дает соотношение $K_H^C/K_Z^C \approx \bar{K}_H^C/\bar{K}_Z^C \approx 1.14$. Кроме того, из полученных результатов следует, что применение комбинации методов сжатия целесообразно в основном для управляющего потока с предикторной

индексацией, когда $\overline{K}_{H\&Z}^{C}/\overline{K}_{Z}^{C} \approx 1.14$. Использование комбинации методов для компрессии потока, полученного на основе алгоритма с однократной индексацией ребер активных треугольников, в среднем дает выигрыш лишь около 3%.

3.3. Компрессия индексного потока

Осредненные соотношения между случаями кодирования локальным и глобальным индексом отдельно стоящей на фронте четвертой вершины тетраэдра и характеристики распределения локальных индексов на отрезке $[0, 15]$ приводятся в табл. 4.

Таблица 4

$\gamma_{C_4}^{C_5}$	id_v^l		
	0÷3	4÷7	8÷15
0.01	0.73	0.22	0.05

Значения параметров слабо зависят от алгоритма индексации ребер треугольников на этапе первичного кодирования и размерности сетки. Число случаев задания четвертой вершины 32-битным целочисленным индексом ограничивается 1-2% от числа переходов к локальной индексации с ограничением $id_v^l < 2^4$. При этом порядка 75% локальных индексов удовлетворяют условию $id_v^l < 2^2$, а в 95% случаев $id_v^l < 2^3$.

Таблица 5

Сетка	Локальные индексы			Глобальные индексы	
	K_Z^{iL}	K_H^{iL}	$K_{H\&Z}^{iL}$	K_Z^{iG}	K_{OZ}^{iG}
NMF	1.15	1.19	1.15	1.60	2.35
PRISM	1.35	1.41	1.42	1.15	2.35
CUBE	1.33	1.40	1.41	1.34	2.35
CUBED	1.33	1.40	1.42	1.37	2.35
SR002	1.35	1.43	1.45	1.32	2.35
SR016	1.33	1.40	1.42	1.36	2.35
SR130	1.33	1.40	1.41	1.38	2.35
SSF001	1.33	1.39	1.41	1.50	2.35
SSF009	1.33	1.40	1.41	1.46	2.35
SSF076	1.34	1.41	1.43	1.43	2.35
Среднее	1.32	1.38	1.39	1.39	2.35

Коэффициент компрессии выделенного массива локальных индексов при использовании комбинации двух методов сжатия равен $K_{H\&Z}^{iL} = 1.39$ раза (табл. 5). Эффект от второго этапа упаковки алгоритмом zlib составляет в среднем только 1%. Поэтому применение zlib в данном случае можно назвать избыточным. Из представленных результатов видно, что сортировка бит в

массиве глобальных индексов позволяет повысить коэффициент сжатия данных на 69% до $K_{OZ}^{iG} = 2.35$ раза.

3.4. Компрессия топологического блока в целом

Значения параметров сжатия сеточной топологии различными подходами представлены в табл. 6. Плотность записи данных в потоковом представлении составляет в среднем 3.37 bpt, что соответствует коэффициенту сжатия 37.95 раза. Последующая компрессия управляющего и индексного потоков поднимает значение коэффициента сжатия до 60.89 ÷ 82.85 раза в зависимости от алгоритма индексации ребер активных треугольников. Осредненный коэффициент компрессии четверок вершин тетраэдров исключительно средствами zlib оказывается как минимум в 11.9 раза ниже.

Таблица 6

	Сетка	Метод сжатия			
		zlib	Алгоритм первичного кодирования	Двухэтапный алгоритм	Двухэтапный алгоритм с предиктором
Плотность записи данных (bpt)	NMF	40.49	4.20	3.04	2.54
	PRISM	42.98	3.50	2.31	1.79
	CUBE	43.90	3.35	2.04	1.59
	CUBED	39.69	3.28	2.01	1.45
	SR002	44.43	3.30	1.92	1.46
	SR016	39.57	3.26	1.96	1.39
	SR130	37.50	3.21	1.94	1.30
	SSF001	39.51	3.26	1.97	1.42
	SSF009	37.56	3.21	1.94	1.31
	SSF076	37.52	3.16	1.89	1.20
	Среднее	40.32	3.37	2.10	1.55
Осредненный коэффициент сжатия (раз)		3.17	37.95	60.89	82.85

Наилучший показатель сжатия получен для сетки SSF076, содержащей $76 \cdot 10^6$ тетраэдров. Плотность записи данных в данном случае равна 1.20 bpt (1.89 bpt без геометрического предиктора), что эквивалентно коэффициенту компрессии топологического блока 106.67 раза (67.72 раза без предиктора).

Проанализировав результаты в табл. 6, можно сделать вывод, что при разработке базового алгоритма кодирования важно делать акцент на достижении максимальной однородности потоков, так как однородный поток ожидаемо эффективнее сжимается методами компрессии без потерь. В представленных примерах регуляризация управляющего потока с использованием геометрического предиктора повышает коэффициент

компрессии самого потока в 1.43 раза и в 1.36 раза улучшает показатель упаковки топологического блока данных в целом.

3.5. Быстродействие программной реализации

Характерные времена и параметры быстродействия разработанного программного обеспечения, зафиксированные в процессе последовательного сжатия и разархивирования сетки SR130, даны в табл. 7. Запуск программ выполнялся на стационарном ПК (процессор Intel Core i7 6700 K 4.0 GHz, 64 ГБ оперативной памяти).

Существенное отличие между временами работы кодера и декодера объясняется учетом процедуры генерации дуального графа, на которую приходится 59% от суммарного времени упаковки данных. Повторная индексация ребер активных треугольников с сортировкой примыкающих граней фронта по двугранному углу замедляет первый этап кодирования данных с переходом к потоковому представлению и процедуру обратного преобразования в декодере примерно в 1.5 раза.

Таблица 7

Процедура		Алгоритм индексации ребер	
		Без предиктора	С предиктором
Кодер	Генерация дуального графа	116 с	
	Первичное кодирование	76.2 с	113.5 с
	Кодирование потоков	2.7 с	3.2 с
	Дополнительные операции	0.8 с	1 с
	Всего	195.7 с	233.7 с
	Быстродействие	687 947 тет/с	576 086 тет/с
	Быстродействие (без учета графа)	1 689 225 тет/с	1 143 851 тет/с
Декодер	Декодирование потоков	1.3 с	1 с
	Восстановление топологии	62.5 с	96 с
	Дополнительные операции	0.8 с	0.8 с
	Всего	64.6 с	97.8 с
	Быстродействие	2 084 075 тет/с	1 376 597 тет/с

Число проверок недвусвязных конфигураций в среднем равно $0.7 \cdot L_C$. В 7.5% ($0.0525 \cdot L_C$) случаев тестируемое ребро классифицируется как недвусвязное после модификации фронта. Но только 0.6% ($0.000315 \cdot L_C$) из них изначально являются недвусвязными ребрами, обработка которых связана с необходимостью вычисления двугранных углов. Максимальное число пар граней, одновременно примыкающих к фронтальному ребру, в подавляющем большинстве случаев равно трем (четыре пары только для сетки NMF).

Максимальная размерность фронта при упаковке сетки SR130 составляет 1.9% от общего числа узлов сетки и 0.3% от числа ее треугольных граней.

Наибольший объем выделяемой оперативной памяти приходится на этап генерации дуального графа и составляет 7.2 гигабайта.

4. Заключение

В настоящей работе описан специализированный двухэтапный алгоритм компрессии топологического блока данных тетраэдральных сеток. Представленные результаты демонстрируют возможность эффективной обработки многомиллионных дискретных моделей с использованием разработанных на его основе алгоритмических и программных средств. В продолжение исследований по данной тематике автором предполагается разработка методов компрессии координатного блока данных с последующей адаптацией подходов на случай кодирования гибридных сеток с элементами типа тетраэдр, треугольная призма, четырехугольная пирамида и гексаэдр.

5. Список литературы

1. Электронный ресурс: https://web.stanford.edu/class/me469b/handouts/gambit_write.pdf.
2. Maglo A., Lavoué G., Dupont F., Hudelot C. 3D mesh compression: Survey, comparisons, and emerging trends. ACM Computing Surveys (CSUR) 47 (3), 44
3. Gumhold S., Guthe S., Straßer W. 1999. Tetrahedral mesh compression with the cutborder machine. In Proceedings of the conference on Visualization (VIS ’99). 51–58.
4. Суков С.А. Методы генерации тетраэдральных сеток и их программные реализации // Препринты ИПМ им. М.В.Келдыша. 2015. № 23. 22 с. URL: http://library.keldysh.ru/preprint.asp?id=2015-23
5. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003. - 384 с.
6. Электронный ресурс: zlib. http://zlib.net/
7. Isenburg M., Lindstrom P., Gumhold S., Shewchuk J. 2006. Streaming compression of tetrahedral volume meshes. In Proceedings of Graphics.

Приложение 1. Блок-схемы

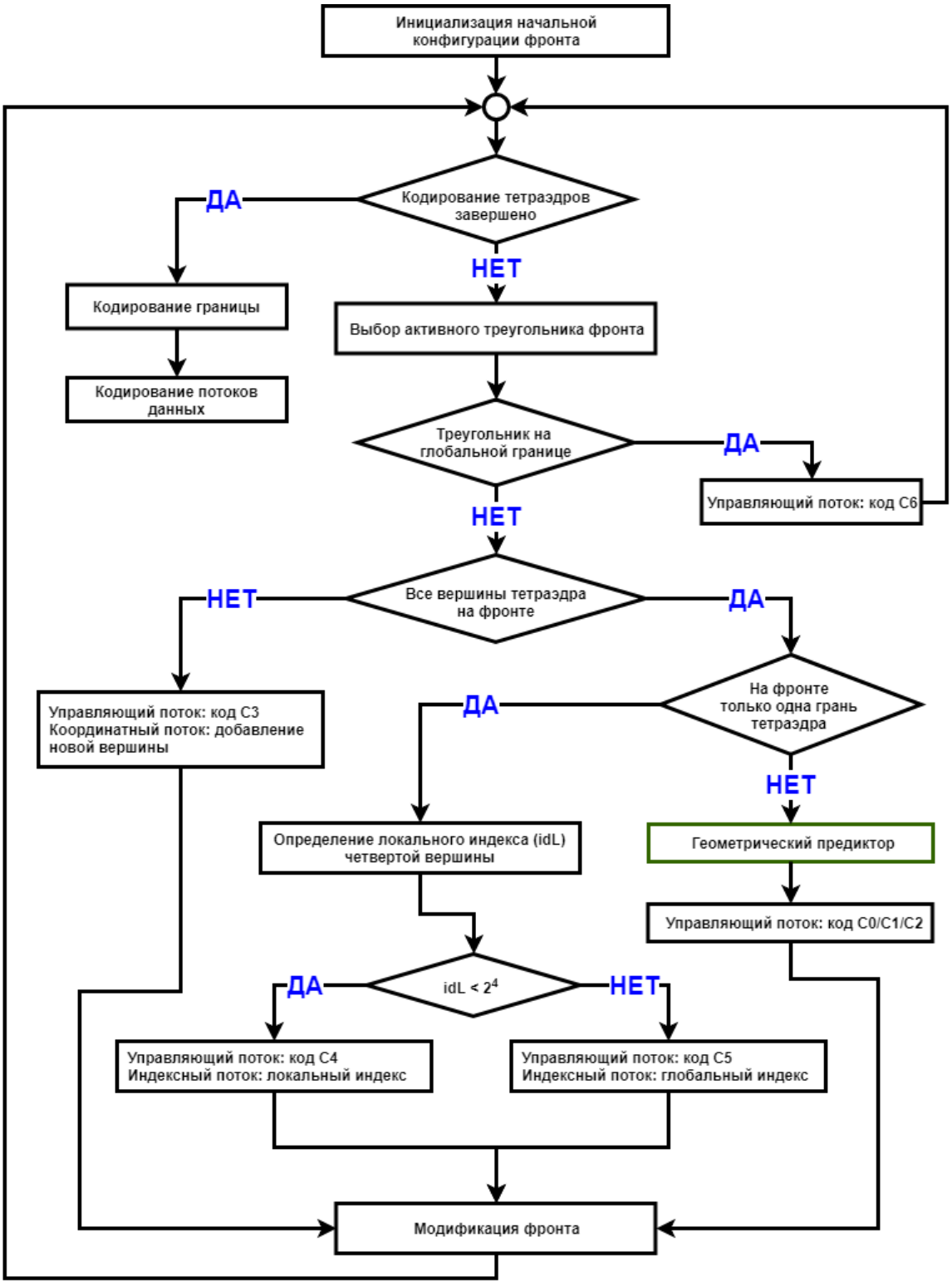


Рис. 1. Блок-схема алгоритма первичного кодирования

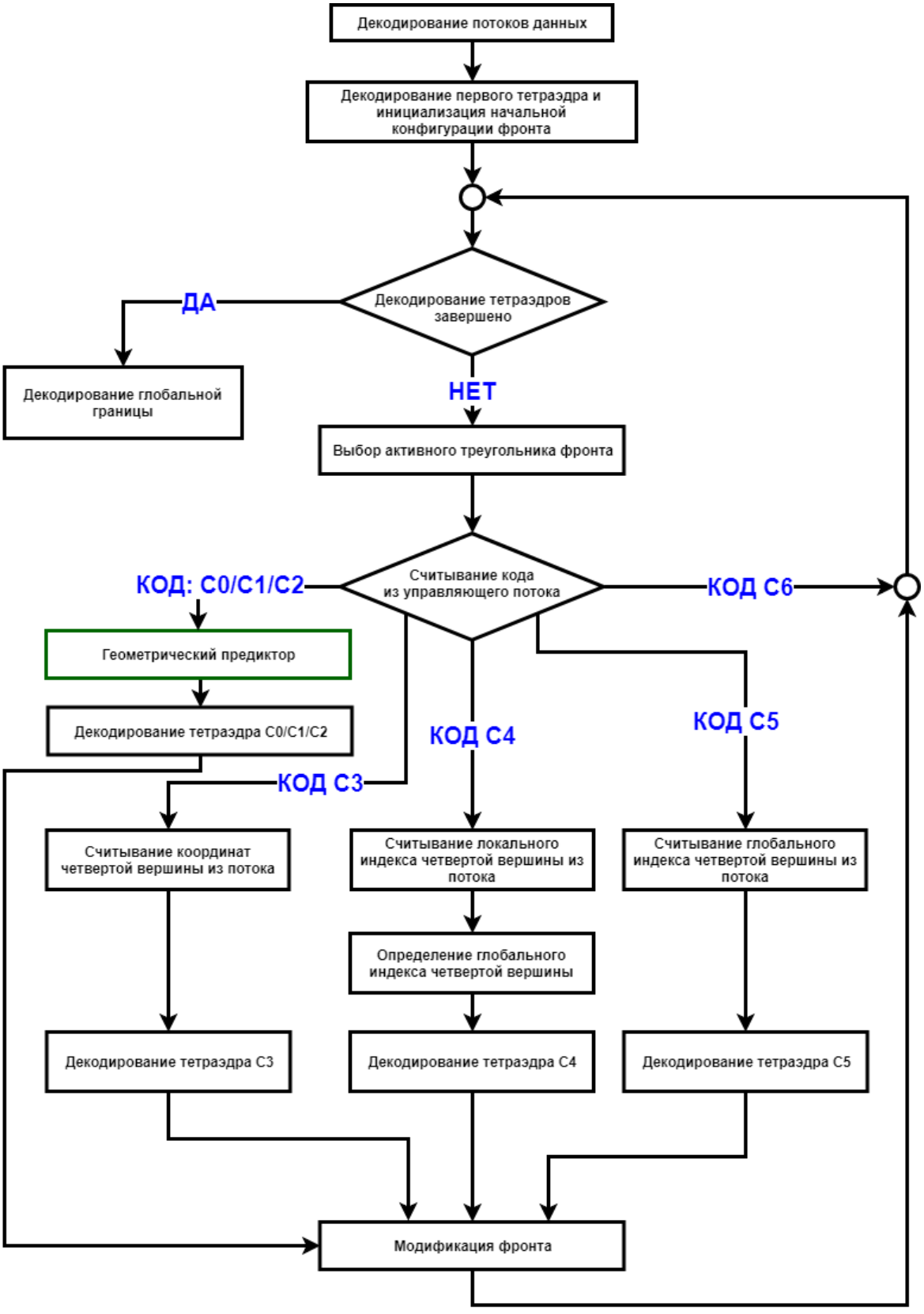
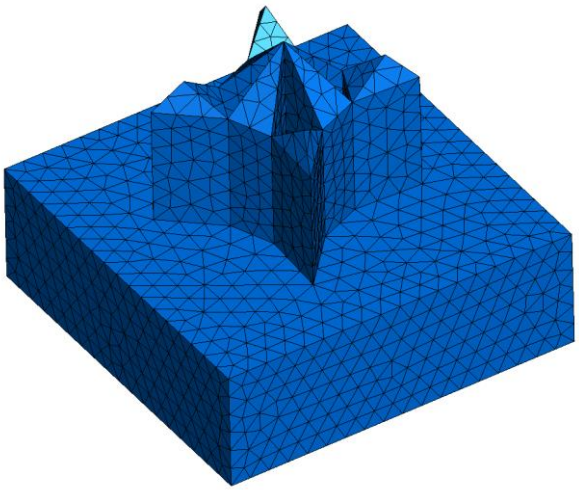
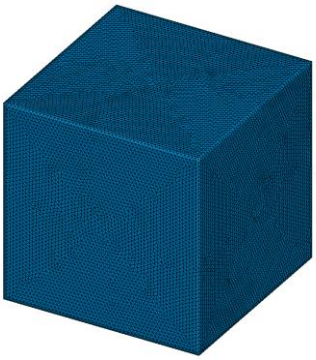


Рис. 2. Блок-схема декодирования потоков

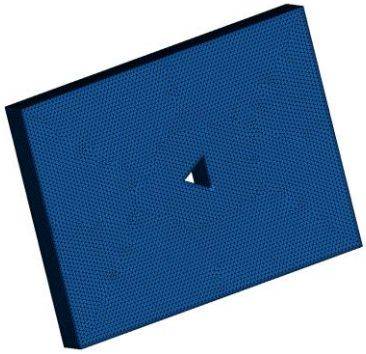
Приложение 2. Тестовые сетки



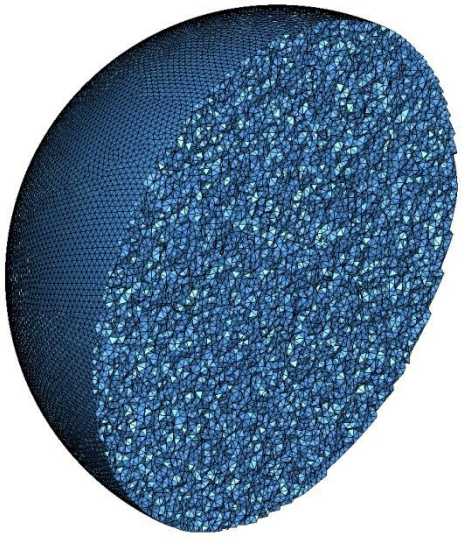
а) сетка NMF



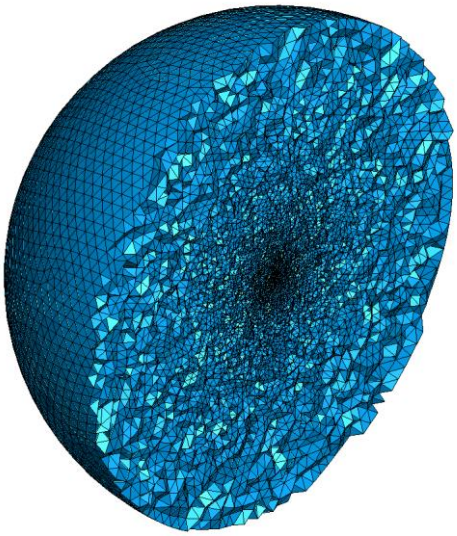
б) сетки CUBE и CUBED



в) сетка PRISM



г) сетки группы SR



д) сетки группы SSF

Рис. 1. Визуализация расчетных областей