УДК 004.422.63

ЭФФЕКТИВНЫЙ АЛГОРИТМ ОБЪЕДИНЕНИЯ СОГЛАСОВАННЫХ ПО УЗЛАМ ФРАГМЕНТОВ ПОВЕРХНОСТНЫХ СЕТОК

В. В. Лазарев
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Предложен новый алгоритм объединения согласованных по узлам фрагментов поверхностных сеток, построенных на геометрической модели в B-REP представлении. По сравнению с классическими алгоритмами объединения он имеет следующие отличия: 1) вместо рассмотрения всех узлов по отдельности рассматриваются их группы; 2) для нахождения совпадающих узлов используются не функция расстояния и ее минимальное значение, а связи между сеточными фрагментами, которые хранятся в структурах представления геометрической модели; 3) результат объединения не хранится в массивах, а представлен в функциональном виде — выражениями, которые по номеру узла вычисляют адрес этого узла в сеточном фрагменте, а по номеру ячейки — номера узлов этой ячейки.

Время выполнения разработанного алгоритма и объем потребляемой памяти почти не заметны. Эти параметры зависят только от количества сеточных фрагментов, но не зависят от количества узлов и ячеек в этих фрагментах.

Ключевые слова: составная сетка, алгоритм объединения поверхностных сеток, глобальная нумерация, функциональное представление множества, граничное представление геометрической модели (B-REP).

Введение

При решении большого класса задач используются так называемые составные сетки. Составная сетка получается разбиением области на подобласти и построением в каждой такой подобласти своей сетки (сеточного фрагмента). К составным можно отнести блочно-структурированные [1] и гибридные сетки [2], сетки с перекрытиями [3]. В блочно-структурированных сетках в каждой подобласти строится структурированная сетка, которая согласована по узлам с остальными. В гибридных сетках и сетках с перекрытиями используются комбинации структурированных и неструктурированных сеток. В общем случае составные сетки добавляют гибкость в построение сеток на сложных геометрических моделях и увеличивают степень структурированности для улучшения качества расчетов.

При реализации методов построения составных сеток перед разработчиком встает вопрос о выборе представления сетки в оперативной памяти. Существуют два основных представления — фрагментное и объединенное. У каждого из них свои преимущества и недостатки.

В ходе построения составной сетки происходят частые модификации сеточных фрагментов — добавления, удаления и редактирования, в связи с чем удобно использовать фрагментное представление сетки. В случае же ее представления в объединенном виде возникает фрагментация памяти из-за частого изменения размера массива, что приводит к невозможности выделения непрерывной области памяти большого размера при имеющихся для этого ресурсах.

Преимущество составной сетки, представленной в объединенном виде, состоит в том, что большинство алгоритмов — сглаживания, оптимизации, деформации — выполняются именно на таких

сетках. При фрагментном же представлении, прежде чем воспользоваться этими алгоритмами, сеточные фрагменты необходимо объединить в одно целое. Частые преобразования данных из одного представления в другое увеличивают задержку в получении конечного результата.

Предлагаемый в статье алгоритм объединения позволяет разработчику воспользоваться преимуществами фрагментного представления сетки и в любой момент переключаться на объединенную сетку быстро и без затрат по памяти. Для этого в алгоритме объединения используется предположение, что два смежных сеточных фрагмента были построены от одной и той же общей границы. Информация о смежности сеточных фрагментов и их общей границе хранится в структурах граничного представления геометрической модели [4, 5], с которой связана составная сетка. Это позволяет отказаться от функции расстояния между сеточными узлами, которая используется в классическом алгоритме объединения. Другая особенность алгоритма в том, что он представляет множество узлов и ячеек объединенной (результирующей) сетки в функциональном виде.

Исходными данными для алгоритма являются структура геометрической модели в граничном представлении (B-REP) и сетки, которые описаны в разд. 1 и 3 соответственно. В разд. 2 вводятся используемые множества, операции над ними и их функциональные представления. Подробное описание предложенного алгоритма приведено в разд. 4. В разд. 5 выполнена оценка производительности реализации алгоритма на языке программирования C++.

1. Представление геометрической модели

В современных CAD-системах твердотельного и поверхностного моделирования геометрическая модель задана в граничном представлении (рис. 1), которое заключает в себе набор поверхностей и информацию об их взаимных связях [4].

Для отслеживания связей поверхностей в рассмотрение вводятся топологические объекты — грани, циклы, ребра и вершины.

Вершина указывает на точку в пространстве — место углового стыка двух и более поверхностей, место излома кривой или стыка нескольких кривых.

Ребро указывает на участок кривой пересечения двух поверхностей. Оно задается двумя вершинами и направлено от первой ко второй. Ребро называется замкнутым, если оно начинается и заканчивается в одной и той же вершине. Одно и то же ребро может входить в несколько граней, но с разными направлениями.

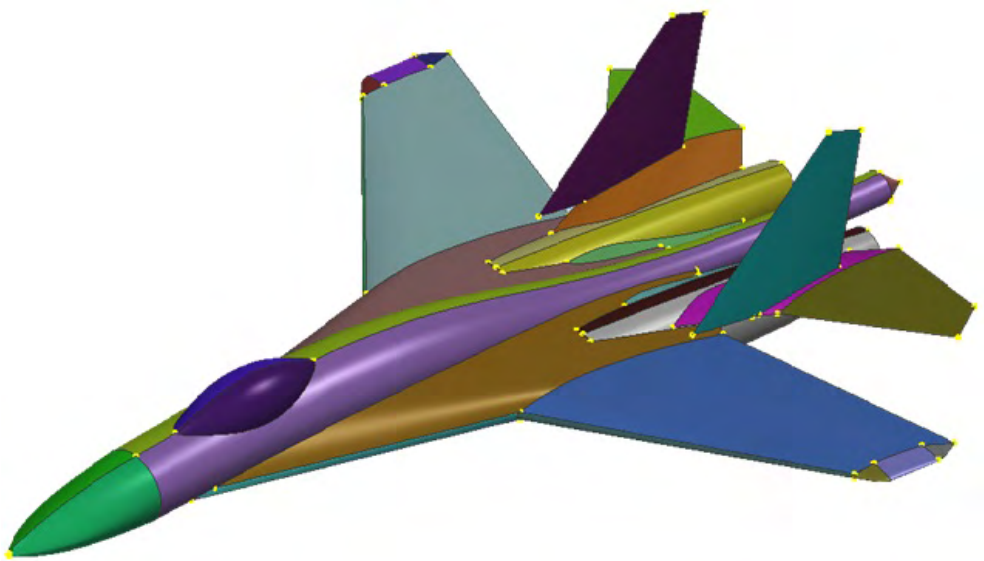

Рис. 1. Геометрическая модель самолета в граничном представлении из [6]

Цикл указывает на одну границу поверхности геометрической модели. Он задается набором стыкующихся в вершинах ребер. С ребром в цикле связан признак инверсного направления. Признак отрицателен, если направление ребра в цикле совпадает с его исходным направлением. В противном случае признак положителен. Одно и то же ребро не может входить в цикл более одного раза. Если в цикле больше одного ребра, то ни одно из этих ребер не может быть замкнутым. Данные ограничения на цикл налагаются алгоритмом объединения сеточных фрагментов.

Грань указывает на поверхность геометрической модели. Она задается одним или несколькими непересекающимися циклами. Первый цикл описывает внешнюю границу грани, а остальные, если они есть, — ее внутренние отверстия.

Множество всех вершин, ребер и граней обозначим соответственно через V, E, F. Через T обозначим объединенное множество вершин, ребер и граней, т. е. $T = V \cup E \cup F$. Введем понятие порядка топологического объекта из T: это целое число, приписанное топологическому объекту. Для вершин его значение равно нулю, для ребер — единице, для граней — двум.

2. Множества, операции над ними и их функциональное представление

Для описания алгоритма объединения сеток будут использованы множества и операции над ними. Все рассматриваемые множества конечные, а их элементы пронумерованы, начиная с нуля. Выражение $M(i)$ обозначает элемент с порядковым номером i множества M.

Используемые далее множества являются подмножествами следующих множеств: $\mathbb{Z}_0$ – натуральных чисел, дополненных нулем; $\mathbb{A}$ – адресов ячеек памяти; T – объектов топологии; произведений $T \times \mathbb{Z}_0$ или $\mathbb{Z}_0^n$ $(n \geq 3)$. Конечные множества могут быть представлены двумя способами – массивом элементов или функцией. Массив элементов может быть заменен таблично заданной функцией. Таким образом, используемые далее множества имеют функциональное представление. Функциональное представление множества дает правило вычисления элемента множества по его номеру.

Функция f, задающая множество, – это функция одного целочисленного аргумента с областью определения $[0, N-1] \in \mathbb{Z}_0$, где N — мощность этого множества. Мощность обозначается как $|f|$.

Благодаря функциональному способу задания множества более экономично используется оперативная память. Такой способ применяется, например, в определении массива ячеек матрично-структурированной сетки. Для вычисления узлов ячейки по ее номеру необходимы только размеры этой сетки по направлениям I, J.

Будем использовать четыре операции над конечными множествами. Результатом выполнения этих операций являются множества, которые представляются в функциональном виде:

1. $f_1 = Reverse(f)$ — изменение последовательности элементов множества на обратный. Результирующее множество f_1 имеет следующее функциональное представление:

$$f_1(i) = f(|f| - i - 1).$$

   При этом $|f_1| = |f|$.

   Например, $Reverse(\{a, b, c, d, e\}) = \{e, d, c, b, a\}$.

2. $f_1 = ExcludeBounds(f)$ — исключение из множества первого и последнего элементов. Результирующее множество f_1 может быть представлено в виде

$$f_1(i) = f(i+1), \text{ причем } |f_1| = |f| - 2.$$

   Таким образом, $ExcludeBounds(\{a, b, c, d, e\}) = \{b, c, d\}$.

3. $f = f_1 \cup f_2$ — объединение множеств в предположении, что все их элементы уникальны. Результирующее множество f имеет следующее функциональное представление:

$$f(i) = \begin{cases} f_1(i), & \text{если} \quad i < |f_1|; \\ f_2(i - |f_1|), & \text{если} \quad i \geq |f_1|; \end{cases} \qquad |f| = |f_1| + |f_2|.$$

   Например, $\{a, b, c\} \cup \{d, e, f\} = \{a, b, c, d, e, f\}$.

4. $f = f_1 \circ f_2$ — выборка подмножества элементов. Результирующее множество f имеет следующее функциональное представление:
$$f(i) = f_1(f_2(i)),$$
причем $|f| = |f_2|$, $0 \le \max(f_2) < |f_1|$. Например, $\{a,b,c,d,e,f\} \circ \{2,4,0\} = \{c,e,a\}$.

3. Представление сеточных фрагментов

Составной сеткой назовем совокупность сеточных фрагментов для топологических объектов из T. Сеточный фрагмент — это самодостаточная сетка, связанная с объектом топологии из T — вершиной, ребром или гранью.

Сеточные фрагменты для вершины и ребра задаются одномерными массивами узлов. Для случая с вершиной в массиве содержится только один элемент.

Особый интерес будет представлять *сеточный фрагмент для грани*. Это поверхностная сетка, которая в ячеечно-узловом представлении задается массивами узлов, ячеек и четырьмя вспомогательными множествами.

Элементом массива узлов в простом случае может быть пара (для двумерного пространства) или тройка (для трехмерного пространства) вещественных чисел — декартовы координаты узла. Массив узлов далее рассматривается как множество $nodes \in \mathbb{A}$.*

Элементами массива ячеек являются списки номеров узлов этих ячеек. Массив ячеек далее рассматривается как множество $cells \in \mathbb{Z}_0^n$, где $n \ge 3$.

Под граничными узлами сеточного фрагмента для грани понимаются узлы, лежащие на цикле грани. Внутренние узлы — это узлы, не являющиеся граничными. Информация о граничных и внутренних узлах задается в виде четырех множеств:

1. $BN \in \mathbb{Z}_0$. Содержит номера во множестве $nodes$ граничных узлов, упорядоченных по направлению обхода цикла и заданной последовательности циклов в грани. $|BN|$ дает количество граничных узлов.
2. $IN \in \mathbb{Z}_0$. Перечисляет номера внутренних узлов во множестве $nodes$ в порядке их возрастания.
3. $BNN \in \mathbb{Z}_0$. Множество имеет следующий функциональный вид:
$$BNN(i) = \begin{cases} x, & \text{если} \quad BN(x) = i; \\ -1, & \text{если} \quad \nexists x : BN(x) = i. \end{cases}$$
4. $INN \in \mathbb{Z}_0$. Множество имеет следующий функциональный вид:
$$INN(i) = \begin{cases} x, & \text{если} \quad IN(x) = i; \\ -1, & \text{если} \quad \nexists x : IN(x) = i. \end{cases}$$

Как $|BNN|$, так и $|INN|$ дает количество узлов в сеточном фрагменте для грани.

На рис. 2 представлен пример сеточного фрагмента с номерами узлов во множестве $nodes$. Стрелкой показаны начало и направление цикла. Для данной сетки $|BN| = 7$, $|IN| = 3$, $|BNN| = |INN| = 10$. Множества содержат следующие элементы:

$$BN = \{4,0,1,9,7,5,3\}; \quad IN = \{2,6,8\};$$
$$BNN = \{1,2,-1,6,0,5,-1,4,-1,3\}; \quad INN = \{-1,-1,0,-1,-1,-1,1,-1,2,-1\}.$$

В общем случае нет необходимости в задании всех четырех множеств. Множества IN, BNN, INN могут быть вычислены по BN, но на практике для большинства случаев существует простой

*Далее для простоты будем отождествлять адрес узла с самим узлом.

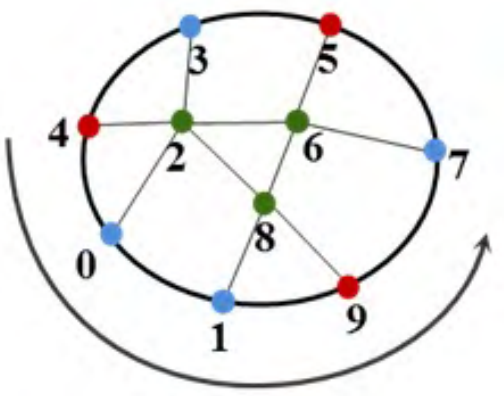

Рис. 2. Пример сеточного фрагмента с номерами узлов во множестве *nodes*

функциональный способ их задания. Для этого нужно заметить, что большинство методов построения неструктурированных сеток от заданной границы добавляют новые узлы (внутренние) в конец массива узлов. Таким образом, в массиве узлов сначала располагаются граничные узлы в порядке, заданном пользователем, а затем — внутренние узлы. Следовательно, если обозначить через n_b, n_i соответственно количество граничных и внутренних узлов в сеточном фрагменте, то

$$BN\,(i) = i; \quad |BN| = n_b; \quad IN\,(i) = n_b + i; \quad |IN| = n_i;$$

$$BNN\,(i) = \begin{cases} i, & \text{если} \quad i < n_b; \\ -1, & \text{если} \quad i \geq n_b; \end{cases} \qquad |BNN| = n_b + n_i;$$

$$INN\,(i) = \begin{cases} -1, & \text{если} \quad i < n_b; \\ i - n_b, & \text{если} \quad i > n_b; \end{cases} \qquad |INN| = n_b + n_i.$$

Для каждого сеточного фрагмента задаются свои множества *nodes*, *cells*, BN, IN, BNN, INN. Далее запись $f|_t$ будет означать, что множество f относится к сеточному фрагменту для топологического объекта t.

4. Вывод функционального представления результата объединения сеточных фрагментов

Назовем сетку, полученную объединением сеточных фрагментов, результирующей. Она так же, как и сеточные фрагменты, состоит из множества узлов и ячеек. Но в отличие от сеточных фрагментов для результирующей сетки эти множества представлены в функциональном виде. В данном разделе показан вывод этих представлений.

4.1. Функциональное представление множества узлов. Множество узлов $nodes \in \mathbb{A}$ результирующей сетки содержит адреса этих узлов в памяти. Для узлов результирующей сетки память не выделяется. Узлы, или их адреса, берутся из исходных множеств $nodes|_{t \in T}$.

Граничные узлы у смежных сеточных фрагментов дублируются — один и тот же узел входит в несколько сеточных фрагментов. Задачей алгоритма объединения является выбор из нескольких одинаковых узлов *правильного*. Под правильным понимается узел, принадлежащий сеточному фрагменту для топологического объекта наименьшего порядка. Чтобы реализовать это условие, введем множество внутренних узлов $innerNodes|_{t \in T} \in \mathbb{A}$ сеточного фрагмента:

$$innerNodes|_{t \in T} = \begin{cases} nodes|_t\,, & \text{если} \quad t \in V; \\ ExcludeBounds\,(nodes|_t)\,, & \text{если} \quad t \in E; \\ nodes|_t \circ IN|_t\,, & \text{если} \quad t \in F. \end{cases}$$

Таким образом, внутренним узлом для вершины будет сам узел-сетка, для ребра — все узлы, исключая крайние, для грани — все внутренние узлы, определенные множеством IN и выделенные операцией выборки.

Узлы результирующей сетки — это объединение внутренних узлов всех сеточных фрагментов. С использованием операции объединения, введенного в разд. 2, данный факт можно выразить в функциональном виде как

$$nodes = \bigcup_{t \in T} innerNodes|_t\,.$$

Заметим, что это выражение исключает дублируемые узлы.

4.2. Глобальная нумерация узлов. Глобальная нумерация для узла сеточного фрагмента определяет его номер в результирующей сетке. Зададим для грани f такую нумерацию множеством $l2g|_{f\in F} \in \mathbb{Z}_0$. Прежде чем определить это множество, определим $GN|_{t\in T} \in \mathbb{Z}_0$ — глобальную нумерацию внутренних узлов сеточного фрагмента для топологического объекта и $bounds|_{f\in F} \in \in Z_0$ — глобальную нумерацию узлов на границе сеточного фрагмента грани.

Последовательность узлов в результирующей сетке определяется последовательностью топологических объектов во множестве T и неявно задает глобальную нумерацию внутренних узлов $GN|_{t\in T}$ сеточных фрагментов как

$$GN|_{T(j)}(i) = i + \sum_{k=0}^{k<j} \left| innerNodes|_{T(k)} \right|.$$

Множество $bounds|_f$ определяется как объединение множеств $l2g|_c$, заданных для циклов грани:

$$bounds|_{f\in F} = \bigcup_{c\in loops(f)} l2g|_c\,.$$

Здесь $loops(f)$ определяет множество циклов грани f. В свою очередь, $l2g|_c$ для цикла можно рассматривать как последовательное (для всех его ребер) объединение множеств GN, определяемых для одной из вершин ребра и для этого ребра с учетом признака направления:

$$l2g|_c = \bigcup_{(e,inv)\in Edges(c)} \left(GN|_{BegVertex(e,inv)} \cup GN|_{(e,inv)} \right).$$

Здесь $Edges(c)$ определяет множество пар — ребер e и признаков inv их направления в цикле c; $BegVertex\,(e, inv)$ возвращает первую вершину ребра e при отрицательном inv и вторую вершину — при положительном. Множество GN для вершины введено выше, а для ребра с учетом признака направления определяется как

$$GN|_{(e,inv)} = \begin{cases} GN|_e\,, & \text{если} \quad inv \quad \text{ложно;} \\ Reverse\,(GN|_e)\,, & \text{если} \quad inv \quad \text{истинно.} \end{cases}$$

На основе введенных выше вспомогательных множеств множество $l2g|_{f\in F}$ определяется как

$$l2g|_{f\in F}(i) = \begin{cases} bounds|_f \left(BNN|_f(i) \right), & \text{если} \quad BNN|_f(i) \neq -1; \\ GN|_f \left(INN|_f(i) \right), & \text{если} \quad BNN|_f(i) = -1. \end{cases}$$

Если узел внутренний, то его номер в результирующей сетке определяется значением из $GN|_f$, в противном случае — значением из $bounds|_f$. Тип узла, а также его порядковый номер в списке граничных или внутренних узлов определяются через $BNN|_f$ и $INN|_f$.

4.3. Локальная нумерация ячеек. Локальная нумерация для номера ячейки результирующей сетки определяет пару: сеточный фрагмент для грани, которому ячейка принадлежит, и ее номер в этом сеточном фрагменте. Зададим такую нумерацию множеством $c2c \in F \times \mathbb{Z}_0$. Заметим, что $|c2c| = N$ — количество ячеек в результирующей сетке.

Введем вспомогательное множество $s \in \mathbb{Z}_0$ номеров ячеек результирующей сетки, которые являются первыми ячейками множеств $cells|_{F(j)}$ сеточных фрагментов для граней:

$$s(i) = \begin{cases} 0, & \text{для} \quad i = 0; \\ \left(\sum\limits_{j=0}^{i-1} \left| cells|_{F(j)} \right| \right) - 1, & \text{для} \quad 0 < i \leq |T|\,. \end{cases}$$

Элементы множества строго возрастают в силу того, что $\left|cells|_{F(j)}\right| > 0$ — количество ячеек в сеточном фрагменте всегда больше нуля.

Тогда множество $c2c$ определяется как

$$\mathrm{c2c}\,(j) = \{F\,(i)\,, j - s\,(i)\}\,,$$

где i удовлетворяет условию $s\,(i) < j < s\,(i+1)$; $|c2c| = \sum\limits_{f \in F} \left|cells|_f\right|$.

4.4. Функциональное представление множества ячеек. Множество ячеек $cells \in \mathbb{Z}_0^n$ результирующей сетки содержит ячейки как списки номеров узлов этих ячеек во множестве $nodes$ (см. подразд. 4.1). Способ вычисления номеров узлов результирующей сетки по номеру ячейки выглядит следующим образом:

1. Номер ячейки результирующей сетки i преобразуется в номер ячейки i' в сеточном фрагменте для грани f. Это преобразование задается локальной нумерацией ячеек, а ее результат содержится в $c2c$.
2. Из множества $cells|_f$ берутся номера узлов $(i_0, i_1, \ldots)$ ячейки i' сеточного фрагмента f.
3. Номера узлов сеточного фрагмента f преобразуются в номера узлов результирующей сетки. Это преобразование задается глобальной нумерацией узлов, а ее результат содержится в $l2g|_f$.

В функциональном виде это представляется как

$$cells\,(i) = \left(l2g|_f\,(i_0)\,, l2g|_f\,(i_1)\,, \ldots\right).$$

Здесь $(i_0, i_1, \ldots) = cells|_f\,(i')$, где $(f, i') = c2c\,(i)$, $|cells| = |c2c|$.

5. Оценка производительности алгоритма

Алгоритм на основе введенных в предыдущих разделах множеств и операций над ними реализован на языке программирования C++ с использованием объектно-ориентированного подхода, классов и полиморфизма подтипов. Программа собрана компилятором Microsoft Visual Studio 2013 под конфигурацией Release со стандартными настройками для 64-разрядной платформы. Для запуска использован персональный компьютер с процессором Intel Core i5-2400, 3.10 ГГц и 16 Гбайт оперативной памяти.

Оценка производительности проведена на трех геометрических моделях, состоящих из 103, 188 и 94 граней. Для каждой модели построены две различные сетки, отличающиеся количеством сеточных элементов. Выполнены измерения трех параметров — времени объединения и скорости доступа к узлам и ячейкам. Доступ выполняется последовательно по всему массиву и повторяется 100 раз.

Полученные измерения представлены в табл. 1, из которой можно сделать следующие выводы. Время объединения сетки зависит от количества сеточных фрагментов и не зависит от количества сеточных элементов в них. Средняя скорость доступа составляет 30 млн узлов в секунду и 7 млн ячеек в секунду. Последовательный доступ непосредственно к элементам массива в ОЗУ дает скорости 350—400 млн узлов и 180—230 млн ячеек в секунду. Вычисление адреса узла и описания ячеек по предложенным в статье выражениям медленнее соответственно в 12—14 и 25—33 раза по сравнению с доступом в память по индексу массива. Скорость доступа может быть ускорена вычислением выражений параллельно, так как предложенные структуры не имеют побочных эффектов.

Оценка объема потребляемой памяти выполнена на геометрической модели с 250 гранями. Объединяемая фрагментная сетка на этой модели состояла из разного количества сеточных элементов. Результаты выполнения алгоритма для этих входных данных представлены в табл. 2. Заметим, что количество узлов/ячеек и количество дублируемых узлов в первом и втором столбцах таблицы меньше действительных значений в 1 000 раз.

Таблица 1

Время объединения и доступа к элементам сетки в зависимости от задачи

Количество граней	Количество узлов/ячеек	Время объединения сетки, с	Время стократного доступа ко всем узлам/ячейкам, с
103	39 603/51 364	0,002	0,133/0,821
103	308 129/393 254	0,002	1,07/6,312
188	248 72/36 480	0,003	0,091/0,553
188	43 184/67 968	0,003	0,147/0,906
94	25 913/40 404	0,002	0,083/0,528
94	171 103/245 733	0,001	0,552/3,332

Таблица 2

Измеренные параметры функции объединения в зависимости от количества сеточных элементов в результирующей сетке

Количество узлов/ячеек	Количество дублируемых узлов	Время объединения сетки, с	Потребляемая память до объединения, Мбайт	Память после объединения, Мбайт	Приращение памяти, Кбайт
8,3/17,5	3,7	0,005	51,141	51,441	308
15,9/32,6	8,7	0,005	51,578	51,848	276
26,2/53,4	12,3	0,005	52,840	52,984	148
57,3/115,5	18,6	0,005	52,687	53,176	504
97,3/195,6	24,8	0,005	54,816	55,227	420
168,6/338,2	34,7	0,006	58,328	58,691	372
253,1/507,1	44,6	0,005	63,176	63,660	496
377,2/755,4	56,7	0,005	71,066	71,336	276
501,4/100,4	66,6	0,005	75,508	75,715	212
1 240,3/2 481,5	107,0	0,005	112,969	113,402	443

Количество ячеек в сетке росло от 17 тыс. до 2,5 млн. При этом количество дублируемых узлов составило от 4 до 107 тыс. Как видно из табл. 2, время объединения не зависит от размеров сетки и занимает около 5 мс. Память, потребляемая за счет функции объединения, не коррелирует с размерами сетки и меняется случайным образом — от 150 до 500 Кбайт. По мнению автора, это связано с малостью измеряемого значения и логики действий менеджера памяти C++. В ходе выполнения функции объединения память запрашивается малыми порциями и может быть выделена в уже существующей странице памяти — в областях, возникших из-за ее фрагментации. Также память может быть выделена в новой странице, но страница будет использована частично. Отсутствие корреляции между размерностью сетки и потребляемой памятью показывает для алгоритма объединения одинаковую эффективность по памяти при увеличении количества сеточных элементов.

Заключение

Предложен новый алгоритм объединения поверхностных сеток, который вместо функции расстояния использует предположение, что смежные сеточные фрагменты построены от одной и той же границы. При этом объединенная сетка представляется в функциональном виде — функциями узлов и ячеек. Такие особенности алгоритма позволяют объединять фрагментную сетку эффективно как по времени, так и по памяти. Время выполнения алгоритма и потребляемая им память не зависят от количества сеточных элементов, а зависят только от количества сеточных фрагментов. Недостатком функционального представления является необходимость вычислений при доступе к элементам сетки. Для преодоления этого недостатка можно воспользоваться двумя способами — па-

раллельным доступом к элементам сетки или кэшированием сетки в оперативной памяти на время использования.

Предложенный алгоритм позволяет разработчику хранить составную сетку в оперативной памяти как набор сеточных фрагментов и в любой момент переходить к объединенной сетке. Такой подход уменьшает фрагментацию памяти при частых изменениях составной сетки, что свойственно при ее построении и редактировании. При этом алгоритмы, разработанные для единой сетки, можно использовать для составной без дублирования оперативной памяти.

Предложенный алгоритм используется для объединения блочных сеток в препроцессоре пакета программ "Логос" [8]. По сравнению с описанным в статье он расширен на случай объединения объемных структурированных сеток. Также объединение используется для подготовки сеточных данных к построению объемной сетки протягиванием [7].

Список литературы

1. *Weatherill N. P., Forsey C. R.* Grid generation and flow calculations for complex aircraft geometries using a multi-block scheme // AIAA 17th Fluid Dynamics, Plasma Dynamics, and Laser Conference. Snowmass, CO. June 25—27, 1984.

2. *Lohner R., Luo H., Spiegel S.* Hybrid grid generation method for complex geometries // AIAA. 2010. Vol. 48, No 11. P. 2639—2646.

3. *Benek J. A., Donegan T. L., Suhs N. E.* Extended chimera grid embedding scheme with application to viscous flow // AIAA Paper. 1987. № 87-1126.

4. *Голованов Н. Н.* Геометрическое моделирование. М.: Физматлит, 2002.
   Golovanov N. N. Geometricheskoe modelirovanie. M.: Fizmatlit, 2002.

5. Industrial Automation Systems and Integration. Product Data Representation and Exchange. Part 42: Integrated Generic Resource: Geometric and Topological Representation. https://www.iso.org/standard/78579.html.

6. Sukhoi Su-27 | 3D CAD Model Library | GrabCAD. https://grabcad.com/library/su-27-2.

7. *Гиниятуллина А. Г., Лазарев В. В., Мартенс Р. В.* Построение сетки протягиванием поверхностных ячеек в отдельном объеме // XV науч.-тех. конф. "Молодежь в науке": сб. тез. Саров, 25—27 октября, 2016. С. 9.
   Giniyatullina A. G., Lazarev V. V., Martens R. V. Postroenie setki protyagivaniem poverkhnostnykh yacheek v otdelnom obyeme // XV nauch.-tekh. konf. "Molodezh v nauke": sb. tez. Sarov, 25-27 oktyabrya, 2016. S. 9.

8. *Лазарев В. В.* Распараллеливание и оптимизация построения блочных расчетных сеток в препроцессоре пакета программ ЛОГОС // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2018. Вып. 1. С. 54—63.
   Lazarev V. V. Rasparallelivanie i optimizatsiya postroeniya blochnykh raschetnykh setok v preprotsessore paketa programm LOGOS // Voprosy atomnoy nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov. 2018. Vyp. 1. S. 54—63.

Статья поступила в редакцию 17.01.20.

AN EFFICIENT ALGORITHM FOR MERGING NODE-MATCHED FRAGMENTS OF SURFACE GRIDS / V. V. Lazarev (FSUE "RFNC-VNIIEF", Sarov, Nizhniy Novgorod region).

A new algorithm is proposed for merging node-matched fragments of surface grids constructed based on a geometric model in the B-REP representation. As opposed to classical merging algorithms, this algorithm: 1) considers groups of nodes rather than each individual node; 2) identifies matching nodes based on the connections between grid fragments, which are stored in geometric model representation structures, rather than based on the distance function and its minimum value; 3) instead of storing the result of merging in arrays, represents it in the form of functions, which calculate the node address in the grid fragment based on the node index and the node indexes in a cell, based on the cell index.

The processor time and the memory consumed by the algorithm are negligibly small. They depend only on the number of grid fragments and do not depend on the number of nodes or cells in the fragments

Keywords: composite grid, surface grid merging algorithm, global indexing, functional representation of a set, boundary representation of a geometric model (B-REP).