УДК 519.6

НЕКОТОРЫЕ ФОРМАТЫ ДАННЫХ ДЛЯ ПРЕДСТАВЛЕНИЯ ДВУМЕРНЫХ НЕСТРУКТУРИРОВАННЫХ СЕТОК ПРОИЗВОЛЬНОГО ВИДА

А. А. Воропинов
(РФЯЦ-ВНИИЭФ)

Рассматривается ряд форматов данных для представления двумерной или поверхностной неструктурированной сетки произвольного вида (ячейки — произвольные многоугольники, в узлах сходится произвольное количество ребер). В основу предлагаемых форматов положен принцип выделения основного элемента сетки. Для рассматриваемых форматов приводятся замеры требуемой памяти, оценки скорости работы с ними. На примере методики ТИМ-2D измеряется доля накладных расходов на выполнение алгоритмов получения информации о соседстве элементов сетки по отношению к расчету физических процессов.

Ключевые слова: неструктурированная сетка произвольного вида, формат данных.

Введение

Один из важных вопросов, возникающих в начале разработки любой методики, использующей разностные сетки, — тип сеток.

Для методик, использующих двумерные *неструктурированные* сетки (сетки произвольного вида), часто применяются подходы, связанные с введением ограничений на структуру сетки, — такие неструктурированные сетки будем называть *нерегулярными*. Один из самых распространенных подходов — использование нерегулярных сеток, состоящих из ячеек с одинаковым количеством узлов. Такой подход, как правило, применяется для треугольных и четырехугольных сеток. Треугольные сетки используются в некоторых счетных методиках, и особенно широко — для описания трехмерных поверхностей. В некоторых методиках расчеты могут проводиться как на треугольных, так и на четырехугольных сетках. Другое ограничение на сетку — фиксированное количество *соседей* в узле. Такого типа сетка используется, например, в методике ДМК [1], где внутренний узел нерегулярной сетки всегда окружают 3 ячейки и в узле сходятся 3 ребра.

Имеются и методики, которые не накладывают ограничений на структуру сетки. Примером такой методики является ТИМ-2D [2].

В зависимости от сетки, которая используется в методике, необходимо выбирать соответствующий формат данных для представления ее структуры (формат хранения). В данной работе рассматриваются форматы, пригодные для представления сеток произвольного вида (ячейки — произвольные многоугольники, в узлах сходится произвольное количество ребер).

Основное требование к формату хранения информации о структуре сетки — его полнота. То есть должна быть возможность получения информации о соседстве для любого элемента сетки. При этом одновременно нужно решать две во многом противоположные задачи:

1) данные о структуре сетки должны занимать как можно меньше оперативной памяти ЭВМ из расчета на одну счетную точку;
2) информацию о соседстве для элемента сетки нужно уметь получать с высокой скоростью.

В реальных условиях эти две задачи одновременно решить сложно, поэтому выбор формата необходимо делать, исходя из того, к каким типам данных приходится обращаться наиболее часто при проведении расчетов. Для форматов, рассматриваемых в настоящей работе, оценки даются именно по двум указанным характеристикам: объему требуемой памяти и скорости получения соседства элементов сетки.

При разработке формата хранения обязательно учитывается способ центрирования величин. В формате должен присутствовать тот элемент (ячейка, ребро, узел), к которому относятся величины, используемые счетными алгоритмами. В настоящей работе рассматривается ряд форматов данных, ориентированных на ячеечно-узловое центрирование (хотя некоторые форматы позволяют использовать и другое центрирование с небольшими модификациями). При этом в первую очередь речь идет о рабочих данных в оперативной памяти ЭВМ и об использовании форматов их хранения при реализации алгоритмов для получения соседства элементов сетки. При сохранении сетки на внешнем носителе используемые форматы часто могут быть существенно сокращены.

Настоящая работа во многом является продолжением работы [3], посвященной разработке формата хранения неструктурированной трехмерной сетки произвольного вида для методики ТИМ-3D [4], но уже применительно к двумерному случаю. В данной работе форматы рассматриваются более глубоко: как уже было сказано, помимо оценки требований к оперативной памяти, приводятся оценки скорости выполнения алгоритмов поиска соседства элементов на сетках различного типа.

1. Подходы к построению форматов хранения неструктурированной сетки

При построении форматов хранения неструктурированной сетки используются различные подходы.

Наиболее просто строятся форматы, содержащие минимальную информацию о сетке. Преимущество таких форматов заключается в том, что они оказываются весьма экономичными по объему требуемой памяти. Однако при проведении расчетов использовать такие форматы затруднительно: для некоторых элементов сетки для получения информации о соседстве может потребоваться выполнение операций глобального поиска, а это приводит к возрастанию сложности алгоритмов получения соседства с $O(N)$ или $O(N \log N)$ до $O(N^2)$. Использование таких форматов представляет интерес, когда при сохранении структуры сетки требуется уменьшить размер содержащего ее файла. Таким свойством обладает, например, формат, используемый для представления неструктурированной сетки в библиотеке ЕФР [5]. В библиотеке ЕФР используется *реберно-ячеечный* формат хранения, описанный в разд. 6, но не хранится информация об опорном ребре для узла. Как следствие, если использовать этот формат напрямую, то для получения информации о соседстве для узла необходимо выполнить поиск этого узла по всем ребрам области.

Один из широко распространенных методов использует соседство между элементами сетки *соседних* размерностей, как в большую, так и в меньшую сторону. В трехмерном случае это выглядит следующим образом. Для ячейки (трехмерного объекта) хранятся грани (двумерные объекты). Для грани хранятся ячейки и ребра (одномерные объекты). Для ребра хранятся грани и узлы (объекты нулевой размерности). Условно такой подход можно назвать *связанными размерностями*. Преимуществом этого подхода является универсальность: сформированные форматы хорошо подходят для алгоритмов, использующих различное соседство. Недостатком таких форматов является большая избыточность и, как следствие, необходимый большой объем памяти. Форматом, близким к такому типу по набору информации, можно считать *реберно-ячеечно-узловой*, описанный в разд. 7 (хотя получен он по принципу *основного элемента*, описанному ниже).

Выделение *основного элемента* сетки — подход, преобладающий в данной работе. При таком подходе для одного, основного, элемента информация о соседстве сохраняется наиболее полно, а для других элементов хранится минимум информации — обычно только связь с основным элементом. При этом если в качестве основного выбирать тот элемент сетки, для которого построено большинство алгоритмов счетной методики, то это практически не скажется на скорости счета. Такой подход позволяет строить достаточно экономичные форматы хранения. Его недостатком является необходимость выполнения дополнительных операций поиска с целью получения информации о соседстве для *неосновных* элементов сетки.

При разработке формата необходимо учитывать различные дополнительные особенности сетки, которые могут иметь место, так, чтобы описание этих особенностей не требовало введения дополнительных признаков и дополнительных операций, "удорожающих" алгоритмы поиска соседства. Самым естественным ограничением на сетку является выбор направления обхода

ячейки. В данной работе использовался обход ячейки против часовой стрелки. Другое ограничение, часто накладываемое на формат, связано с описанием граничных элементов сетки. Такие ограничения могут, на первый взгляд, показаться искусственными, однако они позволяют упростить и часто ускорить работу алгоритмов поиска соседства. Для опознавания граничных элементов в формате вместо номера соответствующей ячейки хранится номер граничного условия. Признаком того, что это именно граничное условие, может быть, например, знак минус. При описании форматов указанные ограничения подробно описываются.

2. Типы использованных сеток

В данной работе рассматриваются форматы, предназначенные для представления неструктурированной двумерной сетки произвольного вида. Тем не менее на практике сетка произвольного вида часто близка по своей структуре и общим характеристикам к равномерной сетке одного из трех типов: треугольная, четырехугольная и шестиугольная. Сетка произвольного вида либо имеет один из таких типов с малым количеством особенностей, либо близка к одному из них в усреднении на одну точку, либо является *смешанной* (состоит из нескольких фрагментов сеток указанных типов). Поэтому для тестирования были использованы следующие типы сеток (рис. 1):

1. Треугольная сетка, в каждом внутреннем узле которой сходятся 6 ребер (рис. 1, *а*).
2. Четырехугольная сетка, в каждом внутреннем узле которой сходятся 4 ребра (рис. 1, *б*).
3. Шестиугольная сетка, в каждом внутреннем узле которой сходятся 3 ребра (рис. 1, *в*). Граничные ячейки в этой сетке содержат по 4—5 узлов.
4. Многоугольная сетка, представляющая собой *диаграмму Вороного* [6] (рис. 1, *г*). В каждом внутреннем узле этой сетки сходятся 3 ребра. Центры ячеек расставлялись случайным образом, поэтому результирующая сетка содержит ячейки с различным количеством узлов. В использованной сетке ячейки содержат от 3 до 14 узлов. Этот тип сетки использовался только для получения оценок скорости работы алгоритмов поиска соседства элементов сетки. По объему требуемой памяти эта сетка близка к шестиугольной.

Для проведения тестовых оценок скорости выполнения алгоритмов поиска соседства были использованы сетки, каждая примерно из 1 млн ячеек. Точные количественные характеристики этих сеток представлены в табл. 1.

При выполнении оценок требуемой памяти предполагается, что сетка не имеет особенностей, и не учитывается наличие границ. В реальных сетках число особенностей, нарушающих некоторую упорядоченность, как правило, невелико. Количество граничных точек также невелико, и для сеток, состоящих примерно из миллиона ячеек, их доля не превышает нескольких десятых процента.

Введем следующие обозначения: N — общее количество ячеек сетки; L — общее количество узлов; K — общее количество ребер; k — количество ребер, сходящихся во внутреннем узле (или количество ячеек, окружающих узел); i — количество узлов у ячейки (или количество соседних ячеек).

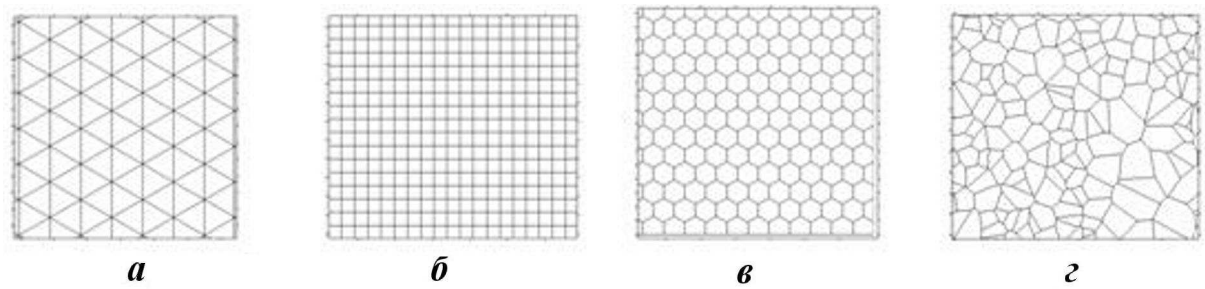

Рис. 1. Фрагменты сеток, использованных для тестирования: *а* — треугольной; *б* — четырехугольной; *в* — шестиугольной; *г* — многоугольной

Таблица 1

Характеристики использованных сеток

Тип сетки	Количество ячеек	Количество ребер	Количество узлов
Треугольная	1 005 519	1 509 100	503 582
Четырехугольная	1 000 000	2 002 000	1 002 001
Шестиугольная	1 000 500	3 001 501	2 001 002
Диаграмма Вороного	1 000 867	3 002 602	2 001 736

В зависимости от типа сетки эти величины принимают следующие значения:

- для треугольной сетки $L = \dfrac{N}{2}$, $K = \dfrac{3}{2}N$, $k = 6$, $i = 3$;
- для четырехугольной сетки $L = N$, $K = 2N$, $k = 4$, $i = 4$;
- для шестиугольной сетки $L = 2N$, $K = 3N$, $k = 3$, $i = 6$.

Оценки выполнялись с усреднением на одну ячейку, т. е. требуемая память на всю сетку зависит от количества ячеек N.

Поскольку в данной работе рассматриваются форматы хранения для сеток, состоящих из ячеек с произвольным количеством узлов и ребер в узле, то при их организации почти всегда требуются списковые массивы переменной длины. В свою очередь, для организации таких массивов использован подход, описанный в работе [3].

3. Формат хранения по узлам

Формат хранения *по узлам* получен путем обобщения формата методики ДМК [1] на случай сеток с произвольным количеством ребер, сходящихся в узлах (в методике ДМК для нерегулярной сетки количество ребер, сходящихся в узле, фиксировано и равно 3). Данный формат хорошо подходит для методики, использующей узловое центрирование, а также для *узловых* алгоритмов.

В формате хранения по узлам основным элементом сетки является узел. Элементами формата являются узлы и ячейки.

Для ячейки хранится номер одного из узлов (опорный узел).

Для узла V (рис. 2) хранятся:

- список соседних узлов (соединенных ребрами с рассматриваемым узлом) $V_1, V_2, \ldots, V_k$;
- список ячеек, окружающих узел (содержащих этот узел), $C_1, C_2, \ldots, C_k$.

Узлы и ячейки в списках упорядочены в порядке их обхода против часовой стрелки, при этом между ними устанавливается соответствие по правилу *узел "догоняет" ячейку* в порядке обхода (см. рис. 2, *а*).

Для граничных узлов в качестве первого выбирается правый соседний граничный узел, если

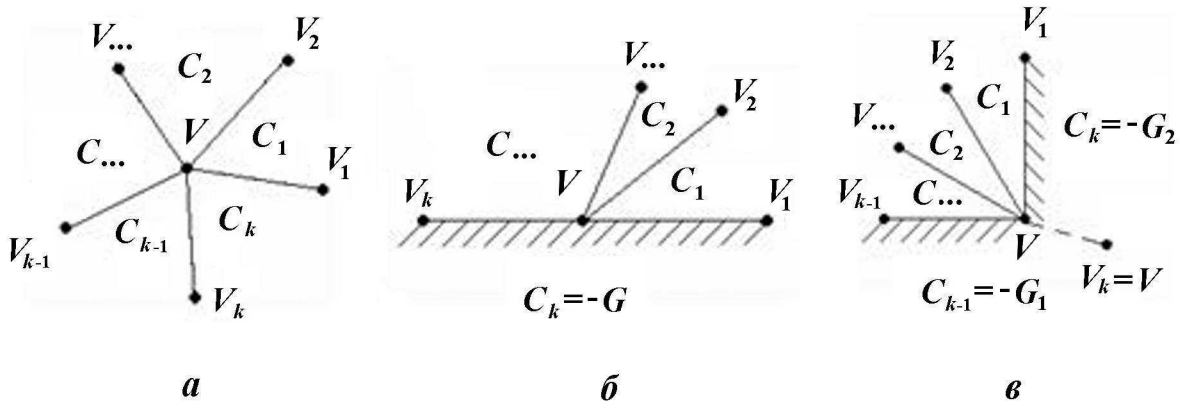

Рис. 2. Информация для узла в формате хранения по узлам: *а* — для внутреннего; *б* — для граничного; *в* — для углового

смотреть со стороны границы (см. рис. 2, *б*). Номер граничного условия хранится в последнем элементе списка соседних ячеек.

Для угловых узлов в последнем элементе списка соседних узлов хранится номер самого узла (рис. 2, *в*). В результате граничные условия хранятся в последних двух элементах списка соседних ячеек.

На трехмерный случай формат напрямую не обобщается. Подобный трехмерный формат с введением элемента *угол* предложен в работе [3].

Как видно из описания, в данном формате хранятся 1 величина для ячейки и $2k$ величин для узла. Но поскольку хранение соседства элементов для узла требует организации списка, то необходимо $2k+2$ величины на каждый узел. Таким образом, объем памяти для формата хранения по узлам выражается следующей формулой: $M = N + (2k+2)\,L$.

4. Формат хранения по ребрам

Формат *по ребрам* получен на основе формата методики МЕДУЗА [1]. В методике МЕДУЗА используется представление соседства по треугольникам. Для каждого треугольника хранятся 3 ребра и 3 узла, а также ряд дополнительных величин. С другой стороны, для ребра не хранятся номера треугольников, которые им разделяются.

В формате хранения по ребрам основным элементом является ребро. Элементами формата являются ячейки, ребра и узлы.

Для ячейки хранится номер одного из ребер этой ячейки.

Для ребра E (рис. 3, *а*) хранятся:

- номера двух узлов V_1, V_2, которые соединяются этим ребром;
- номера двух ячеек C_1, C_2, которые разделяются этим ребром;
- номера двух соседних ребер E_1, E_2, выходящих из узлов этого ребра и являющихся ребрами ячеек C_1, C_2, которые разделяются данным ребром.

При этом вводится следующая упорядоченность пар (см. рис. 3, *а*):

- со стороны первой ячейки C_1 пара узлов V_1, V_2 упорядочена против часовой стрелки (со стороны второй ячейки C_2 узлы V_1, V_2 упорядочены по часовой стрелке);
- ребра E и E_1 являются ребрами ячейки C_1 и имеют общий узел V_1;
- ребра E и E_2 являются ребрами ячейки C_2 и имеют общий узел V_2.

Таким образом, ребра E_1 и E_2 являются соседями ребра E в направлении против часовой стрелки в узлах V_1 и V_2 соответственно.

Для граничного ребра вместо номера первой ячейки всегда хранится номер граничного условия. Таким образом, ребро E_1 будет следующим граничным ребром в порядке обхода против часовой стрелки со стороны расчетной области (рис. 3, *б*).

Для узла хранится номер одного из ребер, сходящихся в данном узле. Для граничного узла в качестве опорного выбирается правое граничное ребро, если смотреть со стороны границы (рис. 3, *в*).

Формат не обобщается на трехмерный случай. Его условным трехмерным аналогом можно считать формат *ребро-грани*, рассмотренный в работе [3].

Как видно из описания, в данном формате хранится по 6 величин на ребро и по 1 величине

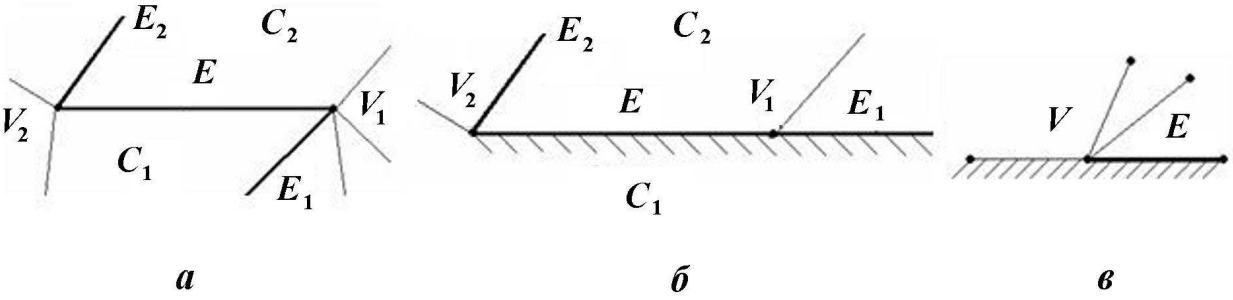

Рис. 3. Информация в формате хранения по ребрам: *а* — для внутреннего ребра; *б* — для граничного ребра; *в* — для граничного узла

на узел и ячейку. Отметим, что это единственный формат из рассматриваемых в данной работе, не требующий организации списков. Таким образом, объем памяти для формата хранения по ребрам выражается следующей формулой: $M = N + 6K + L$.

5. Формат хранения по ячейкам

Этот формат получен на основе формата хранения, который использовался в первой версии библиотеки ЕФР [5]. Доработки формата связаны с хранением ячейки для узла, чтобы избежать операции глобального поиска для определения узлового соседства.

В формате хранения *по ячейкам* основным элементом является ячейка. Элементами формата являются ячейки и узлы.

Для ячейки хранятся списки (рис. 4):

- узлов ячейки в порядке их обхода против часовой стрелки;
- соседних ячеек в порядке обхода против часовой стрелки;

Эти списки упорядочены по правилу *узел догоняет ячейку.*

Для узла хранится номер одной из ячеек (опорной), окружающих узел. Для граничных узлов в качестве опорной ячейки выбирается ячейка, имеющая граничное ребро, выходящее из рассматриваемого узла направо, если смотреть со стороны границы (рис. 5).

На трехмерный случай для сеток произвольного вида формат не обобщается. Однако в случае нерегулярных сеток из ячеек фиксированной формы (например, тетраэдры или топологические кубы) возможно использование подобного формата.

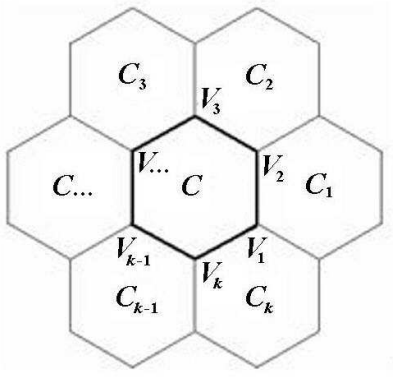

Рис. 4. Информация для ячейки в формате хранения по ячейкам

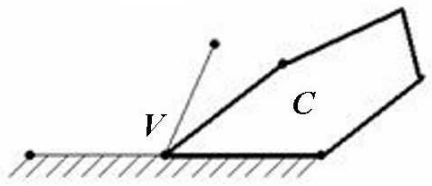

Рис. 5. Выбор опорной ячейки для граничного узла

Как видно из описания, в данном формате хранятся 1 величина для узла и $2i$ величин для ячейки. Но поскольку хранение соседства для ячейки требует организации списка, то необходимо $2i + 2$ величины на каждую ячейку. Таким образом, объем памяти для данного формата выражается формулой $M = (2i + 2)\, N + L$.

6. Реберно-ячеечный формат хранения

Реберно-ячеечный формат хранения является двумерным аналогом формата по граням, используемого для представления трехмерной сетки в методике ТИМ-3D [3]. Данный формат первоначально использовался для представления двумерной сетки и поверхностной сетки произвольного вида в рамках программы ТИМ-РНД [7], а затем с некоторыми доработками стал применяться в методике ТИМ-2D [2].

В реберно-ячеечном формате хранения основным элементом является ребро, для ячеек хранится полный список ребер (отсюда и название формата). Элементами формата являются ячейки, ребра и узлы.

Для ячейки хранится список ребер, перечисленных в порядке их обхода против часовой стрелки (рис. 6, *а*).

Для ребра хранится пара ячеек, которые разделяются ребром, и пара узлов, которые этим ребром соединяются. Со стороны первой ячейки узлы упорядочены против часовой стрелки (рис. 6, *б*).

Для ребра, лежащего на границе области (рис. 6, *в*), вместо номера первой ячейки всегда хранится номер граничного условия.

С целью локализации операций поиска соседства для узла хранится номер одного из сходящихся в узле ребер, так называемого опорного ребра. Если узел находится на границе области,

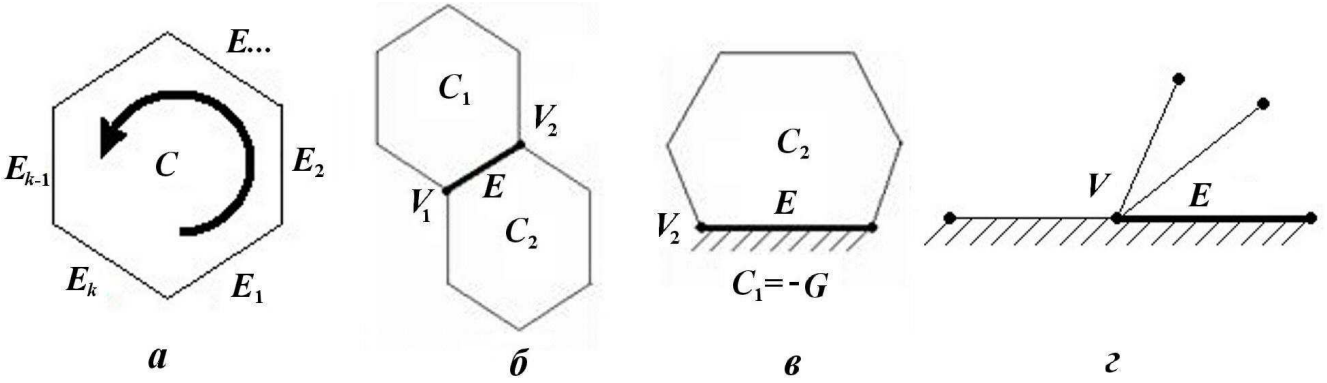

Рис. 6. Структурная информация в реберно-ячеечном формате: *а* — для ячейки; *б* — для внутреннего ребра; *в* — для ребра, лежащего на границе; *г* — опорное ребро для граничного узла

то в качестве опорного выбирается правое граничное ребро, если смотреть со стороны границы (рис. 6, *г*).

Данный формат взят за основу для представления неструктурированной многоугольной сетки в библиотеке ЕФР [5] (в библиотеке ЕФР не хранится опорное ребро для узла).

Хотя формат лучше подходит для алгоритмов, использующих ячеечное или реберное соседство, однако он достаточно просто расширяется информацией о соседстве для узлов. Это расширение превращает формат в *реберно-ячеечно-узловой*, который рассматривается в разд. 7.

Одним из основных преимуществ реберно-ячеечного формата является естественное обобщение на трехмерный случай, результатом которого является формат хранения по граням, используемый для методики ТИМ-3D [3]. При обобщении используется аналогия между ребром в двумерном случае и гранью в трехмерном случае.

Как видно из описания, в данном формате хранятся 1 величина для узла, 4 величины для ребра и i величин для ячейки. Но поскольку хранение соседства для ячейки требует организации списка, необходимо $i+2$ величины на каждую ячейку. Таким образом, объем памяти для реберно-ячеечного формата хранения выражается формулой $M = (i+2)\,N + 4K + L$.

7. Реберно-ячеечно-узловой формат хранения

Данный формат хранения получается расширением реберно-ячеечного формата, описанного в предыдущем разделе, путем дополнительного хранения полного списка ребер, сходящихся в узле. Этот формат, хотя и сформирован по принципу основного элемента, но уже может рассматриваться и как формат на основе связанных размерностей. Такой формат позволяет строить достаточно экономичные алгоритмы, использующие соседство для любого элемента сетки.

Реберно-ячеечно-узловой формат аналогичен реберно-ячеечному по информации для ячеек и ребер и отличается наличием полного списка ребер для узла.

Для узла хранится список ребер, сходящихся в узле, в порядке их обхода против часовой стрелки (рис. 7, *а*). Если узел находится на границе области, то в качестве первого выбирается правое граничное ребро, если смотреть со стороны границы (рис. 7, *б*). Таким образом, граничные ребра всегда являются первым и последним в списке граничного узла.

Недостатком формата является то, что он является самым неэкономичным с точки зрения объема требуемой памяти. Однако благодаря высокой избыточности для этого формата оказываются достаточно быстрыми алгоритмы, построенные по любому из трех элементов сетки.

Как видно из описания, в данном формате хранятся 4 величины для ребра, i величин для ячейки и k величин для узла. Но поскольку хранение соседства для ячейки и для узла требует организации списков, то нужно $i + 2$ величины на каждую ячейку и $k + 2$ величины на узел. Таким образом, объем памяти для реберно-ячеечно-узлового формата хранения выражается формулой $M = (i+2)\,N + 4K + (k+2)\,L$.

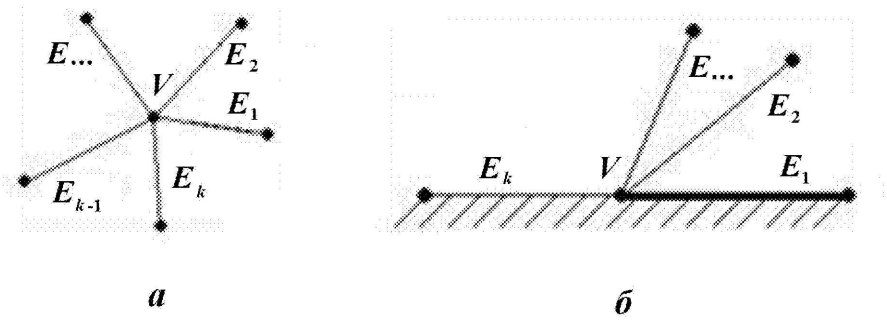

Рис. 7. Информация для узла в реберно-ячеечно-узловом формате: *а* — для внутреннего; *б* — для граничного

8. Сравнение форматов хранения по объему требуемой памяти и скорости поиска соседства элементов сетки

Данные по оценкам затрат памяти для различных форматов хранения сведены в табл. 2. Из таблицы видно, что по объему требуемой памяти наиболее экономичными оказываются форматы, в которых не вводится элемент *ребро* — это форматы по узлам и по ячейкам. С другой стороны, наличие ребер в формате позволяет реализовывать алгоритмы с использованием величин, отнесенных к ребрам. Форматы по ребрам и реберно-ячеечный близки между собой, хотя для формата по ребрам требуется немного меньше памяти (на 1 величину на ячейку). Самым неэкономичным оказался реберно-ячеечно-узловой формат, требующий почти на треть больше величин на одну ячейку по сравнению с реберно-ячеечным форматом. Между остальными форматами разброс составляет не более 3—6 величин в зависимости от типа сетки.

Также необходимо отметить, что даже неэкономичный формат хранения требует объема памяти всего для 30 целых величин на каждую ячейку сетки. При расчете сложных приближений с несколькими десятками величин на одну ячейку такие расходы на описание структуры сетки не являются значительными. Поэтому в целом можно сказать, что объем памяти не является критичным требованием к формату хранения, за исключением тех случаев, когда необходимо работать с очень большими сетками с малым количеством счетных величин на вычислительной системе с ограниченным объемом памяти.

При реализации счетных алгоритмов могут потребоваться операции получения информации о соседстве самых различных элементов сетки: от простейших, таких как узлы ячейки, до сложных, таких как соседи нескольких уровней. При оценке форматов были использованы базовые операции, которые наиболее часто используются в алгоритмах и на основе которых могут строиться и более сложные. Поскольку форматы разрабатывались с ориентацией на ячеечно-узловое центрирование величин, то и выбранные операции поиска соседства элементов сетки также ориентированы на соседство для ячеек и узлов.

При поиске соседства элементов сетки исполь-

Таблица 2

Количество величин для описания неструктурированной сетки в расчете на одну ячейку для различных форматов хранения

Формат хранения	Треугольная сетка	Четырехугольная сетка	Шестиугольная сетка
По узлам	8	11	17
По ребрам	10,5	14	21
По ячейкам	8,5	11	16
Реберно-ячеечный	11,5	15	22
Реберно-ячеечно-узловой	15	20	30

пользуются алгоритмы, получающие следующие списки:

1) узлов ячейки;
2) соседних ячеек для ячейки;
3) соседних узлов для узла;
4) ячеек, окружающих узел.

Данные с оценками скорости работы алгоритмов поиска соседства приведены в табл. 3, 4. Символом △ обозначены столбцы с замерами для треугольной сетки, □ — для четырехугольной, 6 — для шестиугольной, В — для диаграммы Вороного. В таблицах подчеркнуты минимальные значения; жирным шрифтом выделены значения, которые от минимальных отличаются не очень сильно (примерно в 2 раза, если минимальное значение больше 100 и примерно в 3 раза, если минимальное значение меньше 100).

При тестировании алгоритмов сначала производился поиск соседства для всех элементов сетки в порядке их нумерации (см. табл. 3), а затем 1 миллион раз — для элемента сетки со случайным номером (см. табл. 4).

По результатам тестирования необходимо отметить значительное замедление при случайном обращении к операциям поиска соседства. Это демонстрирует, что при последовательном обращении к элементам сетки достаточно интенсивно используется КЭШ-память современных процессоров, что значительно ускоряет работу алгоритмов.

Анализируя полученные данные, можно отметить, что в форматах хранения по узлам и по ячейкам, в которых не вводится элемент *ребро*, быстро получается информация о соседстве для основного элемента и медленно — для неосновного. Форматы с введением элемента *ребро* оказались более сбалансированными. Наиболее быстрым по времени работы алгоритмов оказался формат хранения *по ребрам* — для всех типов сеток алгоритмы, использующие этот формат, демонстрируют минимальные или близкие к ним времена. Немного уступает этому формату реберно-ячеечно-узловой формат хранения. Реберно-ячеечный формат хорош для алгоритмов ячеечного соседства, но для алгоритмов узлового соседства по полученным временам близок к формату хранения по ячейкам.

Особое внимание надо обратить на результаты, полученные для сетки диаграммы Вороного, в сравнении с результатами на шестиугольной сетке. Поскольку сетка диаграммы Вороного строилась со случайной расстановкой центров, то результирующая сетка сильно "перемешана" — имеется соседство между элементами с большой разницей в номерах. Такое перемешивание может возникать, например, в процессе расчета из-за использования локальных перестроек сетки. Из-за перемешанности элементов сетки при работе алгоритмов поиска соседства слабо используются преимущества КЭШ-памяти современных процессоров. В результате алгоритмы на этой сетке работают значительно медленнее даже при последовательном обращении (ср. табл. 3 и 4).

Проведенное тестирование было бы не полным без сравнения представленных алгоритмов со счетными алгоритмами, которые производят полезные вычисления. Это необходимо сделать, чтобы определить долю накладных расходов на получение соседства элементов сетки при выполнении расчетов. Такое тестирование проводилось на алгоритмах расчета газовой динамики методики ТИМ-2D [8].

В методике ТИМ-2D используется реберно-ячеечный формат хранения. Был сделан замер времени выполнения одного счетного шага на сетках, указанных в табл. 5.

Замерялось время выполнения блока расчета узловых величин (расчет уравнения движения). В этом блоке используется информация о соседстве для узла, при этом выполняется процедура, которая одновременно получает номера ячеек, ребер и узлов, окружающих данный узел. Результаты замеров доли накладных расходов на получение соседства элементов сетки при расчете узловых величин в методике ТИМ-2D приведены в табл. 6.

Второй блок, для которого выполнялись измерения, — расчет ячеечных величин (расчет уравнения энергии). В этом блоке получается информация об узлах ячейки. Результаты замеров доли накладных расходов на получение соседства элементов сетки при расчете ячеечных величин в методике ТИМ-2D приведены в табл. 7.

Необходимо отметить, что приближение газовой динамики является одним из самых "дешевых" в расчетах по методике ТИМ-2D. При решении задач с использованием приближений, требующих большего объема вычислений, доля накладных расходов на работу операций поиска соседства элементов сетки не превышает нескольких процентов и практически не влияет на общую скорость счетных алгоритмов.

Таблица 3

Время выполнения алгоритмов поиска соседства элементов сетки для различных форматов хранения при последовательном обращении

Алгоритм поиска	Формат хранения																			
	По узлам				По ребрам				По ячейкам				Реберно-ячеечный				Реберно-ячеечно-узловой			
	△	□	6	B	△	□	6	B	△	□	6	B	△	□	6	B	△	□	6	B
Узлы ячейки	203	219	344	1578	**31**	**47**	**47**	484	**31**	**31**	**63**	**78**	**78**	**94**	**109**	312	**63**	**78**	**31**	**281**
Соседние ячейки	219	219	344	1594	**31**	**16**	**31**	438	**31**	**47**	**31**	**47**	**31**	**31**	**31**	281	**78**	63	**31**	297
Ячейки, окружающие узел	**16**	**31**	**63**	**63**	**16**	**16**	**16**	297	156	188	328	1656	94	125	188	1875	**16**	**31**	109	406
Соседние узлы для узла	**16**	**47**	**78**	**63**	**16**	**16**	**16**	281	188	250	438	1797	94	125	250	1891	**31**	**63**	**63**	375

Таблица 4

Время выполнения алгоритмов поиска соседства элементов сетки для различных форматов хранения при случайном обращении

Алгоритм поиска	Формат хранения																			
	По узлам				По ребрам				По ячейкам				Реберно-ячеечный				Реберно-ячеечно-узловой			
	△	□	6	B	△	□	6	B	△	□	6	B	△	□	6	B	△	□	6	B
Узлы ячейки	875	859	1047	1828	**422**	**422**	**594**	750	**297**	**297**	**328**	**328**	**484**	**469**	**531**	**625**	**484**	**469**	**516**	**625**
Соседние ячейки	891	859	1062	1859	**266**	**422**	**516**	734	**203**	**188**	**219**	**281**	469	453	**531**	641	**359**	438	**500**	625
Ячейки, окружающие узел	**148**	**297**	**1312**	**656**	**242**	**422**	**1000**	**452**	602	859	3500	2232	742	1078	4248	2812	**258**	**453**	**906**	**906**
Соседние узлы для узла	**102**	**203**	**812**	**438**	**219**	**266**	**1000**	**500**	633	906	3652	2438	742	1047	4188	2750	**250**	**438**	**688**	**718**

Таблица 5

Время расчета счетного шага в методике ТИМ-2D

Тип сетки	Расчет узловых величин	Расчет ячеечных величин
Треугольная	1181	2445
Четырехугольная	1402	2736
Шестиугольная	2373	3511
Диаграмма Вороного	6116	5184

Таблица 6

Доля накладных расходов на получение соседства элементов сетки в методике ТИМ-2D при расчете узловых величин

Тип сетки	Время расчета узловых величин	Время на получение соседства для узла	Доля накладных расходов, на получение соседства, %
Треугольная	1181	203	17,19
Четырехугольная	1402	281	20,04
Шестиугольная	2373	0469	19,76
Диаграмма Вороного	6116	2266	37,05

Таблица 7

Доля накладных расходов на получение соседства элементов сетки в методике ТИМ-2D при расчете ячеечных величин

Тип сетки	Время расчета узловых величин	Время на получение соседства для узла	Доля накладных расходов, на получение соседства, %
Треугольная	2445	0344	14,07
Четырехугольная	2736	0391	14,29
Шестиугольная	3511	0562	16,01
Диаграмма Вороного	5184	2047	39,48

Заключение

В статье рассмотрен ряд форматов хранения, пригодных для представления неструктурированной многоугольной сетки произвольного вида. Форматы сформированы на основе подхода с выделением основного элемента. Для всех рассмотренных форматов даны оценки с точки зрения требуемой памяти и скорости работы базовых алгоритмов поиска соседства элементов сетки.

Оптимальным из рассмотренных оказался формат хранения по ребрам. Однако его существенным недостатком является отсутствие прямого обобщения на трехмерный случай.

В целом необходимо отметить, что в большинстве случаев доля накладных расходов на работу с форматом хранения не превышает 20 % даже для небольших по объему вычислений приближений. Объем требуемой памяти для большинства форматов колеблется в пределах 8—22 величин на одну ячейку в зависимости от типа сетки и также не является существенным.

В этих условиях при выборе формата хранения на первый план выходят дополнительные характеристики, такие как наличие "прозрачного" обобщения на трехмерный случай. Из всех рассмотренных форматов таким свойством обладает только реберно-ячеечный формат хранения. Именно этот формат используется в многомерной методике ТИМ для описания неструктурированных двумерной и трехмерной сеток произвольного вида. Близость двумерного и трехмерного форматов позволяет разрабатывать алгоритмы сразу для обоих случаев.

Список литературы

1. *Sofronov I. D., Rasskazova V. V., Nesterenko L. V.* The use of nonregular nets for solving two-dimensional nonstationary problems in gas dynamics // Numerical Methods in Fluid Dynamics / Ed. by N. N. Yanenko, Ju. I. Shokin. Moscow: Mir Publishers, 1984. P. 82—121.
2. *Соколов С. С., Воропинов А. А., Новиков И. Г. и др.* Методика ТИМ-2D для расчета задач механики сплошной среды на нерегулярных многоугольных сетках с произвольным количеством связей в узлах // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2006. Вып. 4. С. 29—43.
3. *Воропинов А. А., Соколов С. С., Панов А. И., Новиков И. Г.* Формат для описания нерегулярной многогранной сетки произвольной структуры в методике ТИМ // Там же. 2007. Вып. 3—4. С. 55—63.
4. *Соколов С. С., Панов А. И., Воропинов А. А. и др.* Методика ТИМ расчета трехмерных задач механики сплошных сред на неструктурированных многогранных лагранжевых сетках // Там же. 2005. Вып. 3. С. 37—52.
5. *Волгин А. В., Тарасов В. И., Красов А. В., Кузнецов М. Ю.* Библиотека ЕФР для универсального представления расчетных данных // Труды РФЯЦ-ВНИИЭФ. 2007. Вып. 11. С. 130—135.
6. *Voronoi G.* Nouvelles applications des parametres continus a la theorie des formes quadratiques. Deuxieme Memorie: Recherches sur les paralleloedres primitifs // J. Reine und Angew. Math. 1908. No 134. P. 198—287.
7. *Воропинов А. А.* Развитие методики построения нерегулярной многогранной сетки на основе полистового заполнения // Молодежь в науке: Сб. докл. IV науч.-тех. конф. Саров: РФЯЦ-ВНИИЭФ, 2006. С. 31—36.
8. *Воропинов А. А., Соколов С. С., Новиков И. Г.* Метод расчета задач газодинамики и упругопластичности на нерегулярных многоугольных сетках в методике ТИМ-2D // Молодежь в науке: Сб. докл. V науч.-тех. конф. Саров: РФЯЦ-ВНИИЭФ, 2007. С. 46—55.

Статья поступила в редакцию 26.02.10.