Программное обеспечение для создания адаптивных сеток

А.Н. Семакин <arte-semaki@yandex.ru>
Московский Государственный Технический Университет им. Н. Э. Баумана
105005, Россия, Москва, 2-я Бауманская ул., д. 5, стр. 1

Аннотация. В этой статье мы представляем программный пакет для создания адаптивных конечно-разностных сеток через представление гидромеханических переменных разреженным рядом вейвлетов. Приводятся технические детали реализации. В частности, описаны используемые структуры данных и способ распараллеливания вычислений. Также представлены результаты решения некоторых физических задач с привлечением данного пакета.

Ключевые слова: программное обеспечение; вейвлеты; адаптивные сетки.

DOI: 10.15514/ISPRAS-2017-29(5)-15

Для цитирования: Семакин А.Н. Программное обеспечение для создания адаптивных сеток. Труды ИСП РАН, том 29, вып. 5, 2017 г., стр. 311-328. DOI: 10.15514/ISPRAS-2017-29(5)-15

1. Введение

Практически все физические процессы одновременно протекают в широком диапазоне пространственных масштабов. Например, перенос примесей в атмосфере зависит как от крупномасштабной циркуляции (~10 тыс. километров), так и мелкомасштабной конвекции (~100 метров или меньше). Характер многофазных течений определяется не только факторами, действующими в масштабе все области (геометрия границ, гравитация и т.д.), но и в значительной степени зависит от межфазных взаимодействий на границе раздела фаз, где происходит резкое изменение материальных свойств среды.

Для достижения высокой точности в ходе моделирования численные методы должны разрешать все вовлеченные пространственные масштабы, что требует наличия огромного объема вычислительных ресурсов. В результате подобные расчеты достаточно сложно и дорого проводить на регулярной основе. Однако, довольно часто не требуется рассчитывать всю рассматриваемую область с одинаково высоким уровнем точности. Обычно в каждый отдельный момент времени малые пространственные масштабы проявляются на нескольких относительно маленьких участках исходной расчетной области. В остальных

частях исходной области физические переменные изменяются плавно и могут быть рассчитаны на грубой сетке без потери точности. Эта особенность была положена в основу целого ряда методов расчета на адаптивных сетках.

В этой статье мы представляем программный пакет создания адаптивных сеток с помощью одного из существующих подходов — теории вейвлетов. Эта адаптивная техника позволяет нам концентрировать точки сетки в областях, где гидродинамические переменные (скорость, плотность, давление, концентрация химических компонентов и т.д.) претерпевают резкие изменения, и разрежать сетку в областях, где эти переменные изменяются в незначительной степени. Сама теория вейвлетов предоставляет математический инструмент для выделения таких областей. Среди характерных особенностей нашего программного пакета можно выделить: эффективный алгоритм разложения гидродинамических переменных в ряд по вейвлетам с быстрым вычислением вейвлет-коэффициентов, быстрый алгоритм адаптации сетки на основе полученного ряда вейвлетов, возможность применения конечно-разностных схем высокого порядка точности (3 и выше), способность конструировать одно-, двух- и трехмерные сетки.

Статья организована следующим образом. В разделе 2 описаны однородные сетки и три вида адаптивных сеток, показаны преимущества адаптивного измельчения сеток. В разделе 3 перечислены существующие программные пакеты по построению адаптивных сеток. В разделе 4 приведены некоторые теоретические положения метода построения адаптивных сеток с помощью вейвлетов. В разделах 5 и 6 представлены использующиеся в нашем программном пакеты структуры данных и способ распараллеливания вычислений. В разделе 7 приведены некоторые примеры численных расчетов.

2. Однородные и адаптивные сетки

Любой конечно-разностный метод преобразует дифференциальное уравнение, описывающее тот или иной физический процесс, в конечно-разностное уравнение, определенное на некоторой разностной сетке. Подавляющее большинство численных методов использует однородные сетки с постоянным интервалом между узлами. Этот интервал обычно выбирается так, чтобы решение разностного уравнения было как можно ближе к решению исходного дифференциального уравнения.

Однородная сетка обеспечивает равномерное разрешение всей расчетной области. Если гидромеханические переменные имеют резкие градиенты в какой-то одной небольшой части расчетной области, и для их расчета требуется достаточно мелкая сетка, то мы должны использовать эту мелкую сетку во всей области, несмотря на то, что для получения решения с приемлемой точностью в остальной части области можно задействовать более грубую сетку. Таким образом, использование однородной сетки приводит к неэффективному расходованию имеющихся вычислительных ресурсов.

Одним из возможных путей решения этой проблемы является переход от однородных сеток к адаптивным. Адаптивные сетки увеличивают плотность узлов в областях с резкими изменениями гидродинамических переменных и уменьшают там, где значения переменных меняются слабо. Кроме того, адаптивные сетки перестраиваются с течением времени вслед за конфигурацией потока. Эти свойства минимизируют требования к вычислительным системам, так как численные алгоритмы занимают не больше вычислительных ресурсов, чем им нужно для достижения заданной точности.

Существует несколько подходов к построению адаптивных сеток: криволинейные сетки, вложенные сетки и адаптивное измельчение сеток (АИС).

Криволинейные сетки строятся с помощью криволинейных систем координат, которые сгущают сетку вдоль одной или нескольких координатных направлений. В результате образуются хорошо разрешенные пространственные полосы, которые пересекают всю расчетную область в направлении, перпендикулярном соответствующей координате. Лишь небольшая часть каждой такой полосы (~20% или меньше) приходится на область резкого изменения гидромеханических переменных. Остальная часть (~80%) приходится на участки, подробное разрешение которых не требуется. Таким образом, хотя криволинейные сетки имеют существенно меньший размер по сравнению с однородными сетками, значительная доля узлов таких сеток используется крайне неэффективно.

Вложенные сетки обычно состоят из основной разреженной сетки, покрывающей всю область, и нескольких дополнительных сеток меньшего размера и с большей плотностью точек, которые встроены в основную сетку на участках, требующих более подробного разрешения, чем основная часть области. Использование дополнительных встраиваемых сеток конечного размера убирает избыточное разрешение значительных участков исходной области, возникающее при использовании криволинейных систем координат, и, следовательно, еще больше уменьшает размер сетки. Обычно на практике используют набор из основной крупной сетки и одной или двух последовательно вложенных друг в друга малых сеток.

Как криволинейные, так и вложенные сетки имеют фиксированный размер, который задается вручную в начальный момент времени и не меняется во время расчетов. Это означает, что данные сетки не способны реагировать на динамические изменения пространственных масштабов и требуют их предварительной оценки. Также для построения этих сеток требуется предварительная информация о возможном распределении значений гидромеханических переменных в расчетной области.

В отличие от описанных выше подходов к построению адаптивных сеток методы АИС способны динамически адаптировать сетку к любым особенностям течения без какой-либо предварительной информации. Методы АИС автоматически измельчают или огрубляют сетку в зависимости от

текущего распределения значений гидромеханических переменных в расчетной области согласно заданному адаптационному критерию. Возможность оперативно изменять размер сетки позволяет методам АИС мгновенно реагировать на любые изменения в пространственных масштабах, поддерживая точность расчетов на заданном уровне.

3. Программные пакеты для АИС

В отличие от методов, работающих с сетками, структура и размеры которых известен до начала расчетов, АИС работает с динамически изменяющимися сетками, структура которых заранее неизвестна, что сильно усложняет задачу эффективного представления адаптивной сетки в памяти компьютера, а также последующей работы с различными ее частями.

Большая трудоемкость разработки программного кода под АИС привела к появлению и распространению ряда готовых программных продуктов для построения и управления адаптивными сетками. Такие пакеты обычно содержат систему переменных, функций и классов, написанных на одном из языков программирования высокого уровня. Их можно включить в состав собственного программного продукта и использовать, не разбирая подробно внутренние детали реализации, что существенно сокращает время написания собственных программ.

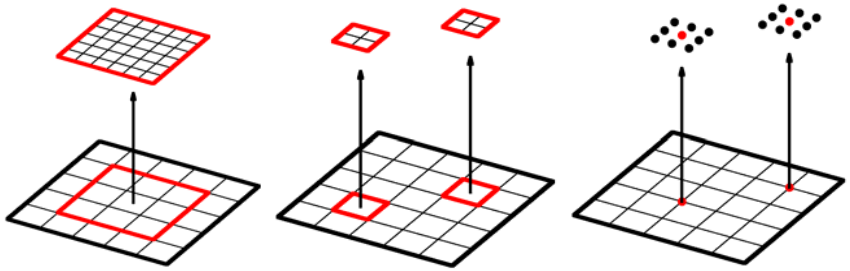
На данный момент в программных пакетах реализованы две стратегии измельчения сетки. Первая стратегия использует иерархию сеток, которые принадлежат разным уровням разрешения и покрывают последовательно уменьшающуюся часть исходной области (см. рис. 1а). В этом подходе измельчаются относительно большие блоки ячеек сетки, которые используются в качестве основных программных единиц при построении структур данных. Из-за регулярной формы сеток, входящих в иерархию, достаточно сложно эффективным образом охватить различные нерегулярные течения с теми или иными особенностями. В этом случае существенное количество точек будет приходиться на участки, не требующие подробного разрешения. Вследствие относительной простоты реализации подавляющая часть программных пакетов реализует именно этот подход построения адаптивных сеток: PARAMESH [1], AMRCLAW [2], AMROC [3], Enzo [4], HAMR [5], DAGH [6], AMR++ [7], SAMRAI [8], AMRCART [9], IAMR [10].

Вторая стратегия использует для измельчения отдельные ячейки сетки вместо их блоков. Каждая ячейка может быть поделена на несколько дочерних ячеек меньшего размера независимо от остальных (см. рис. 1б). В результате данная стратегия обладает большей гибкостью, но производит сетки нерегулярной формы, что усложняет их программную реализацию. Из-за этого лишь очень небольшое число пакетов реализует данную стратегию (например, Gerris [11]).

На рис. 2 показаны примеры адаптивных сеток, построенных по обеим стратегиям. Адаптивные сетки имеют два уровня разрешения и покрывают некое облако химического компонента, занимающее лишь часть расчетной

области. Сетка, основанная на блоках, состоит из двух блоков разного размера (рис. 2а). Большой блок разрешает всю расчетную область на первом уровне, а второй блок используется для разрешения облака на втором уровне. Хорошо видно, что часть ячеек второго блока лежит за пределами облака и, следовательно, используется нерационально. В свою очередь второй уровень адаптивной сетки, построенной путем измельчения ячеек, имеет нерегулярную структуру и содержит только те ячейки, которые разрешают само облако. Таким образом, данная сетка подробно разрешает только ту часть исходной области, которая нуждается в этом. В результате размер данной сетки существенно уменьшается по сравнению с блочной.

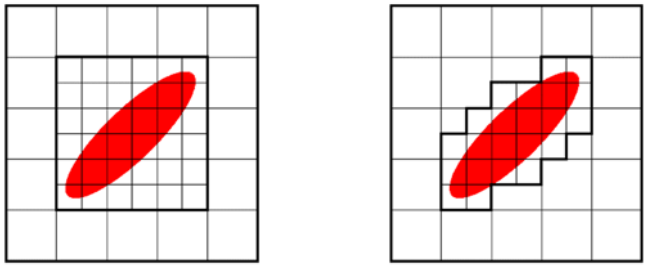
Наш программный пакет использует третью стратегию измельчения сетки — измельчение сетки на основе узлов (рис. 1в). Когда возникает необходимость, любой узел сетки на данном уровне разрешения может породить до 8 дочерних узлов в двухмерном случае и до 26 дочерних узлов в трехмерном случае на более высоком уровне разрешения. При измельчении ячеек мы получаем только 4 или 8 дочерних ячеек в двухмерном и трехмерном случаях, соответственно (сравните рис. 1б и 1в). В результате для обеспечения той же мелкости наша сетка требует меньшего числа уровней разрешения, что влечет за собой меньшее время доступа к каждому отдельному узлу сетки.



а) б) в)

Рис. 1.Способы измельчения сетки: (а) на основе блоков, (б) на основе ячеек, (в) на основе узлов.

Fig. 1. Approaches to the grid refinement: (a) block-based grid, (b) cell-based grid, and (c) point-based grid.



а) б)

Рис. 2 Примеры адаптивных сеток с двумя уровнями разрешения: (а) на основе блоков, (б) на основе ячеек.

Fig. 2. Examples of the adaptive grids with two levels of resolution: (a) block-based grid and (b) cell-based grid.

4. Адаптация сетки с помощью вейвлетов

Рассмотрим реализованный в нашем пакете способ построения адаптивных сеток на основе теории вейвлетов.

В любой момент времени t_0 достаточно гладкая функция $f(\mathbf{x})$ может быть разложена в ряд по вейвлетам [12]:

$$f(\mathbf{x}) = \sum_{\mathbf{k}} b_{\mathbf{k}} \varphi_{\mathbf{k}}(\mathbf{x}) + \sum_{l=1}^{\infty} \sum_{\mathbf{k}} \mathbf{D}_{l,\mathbf{k}} \mathbf{\Psi}_{l,\mathbf{k}}(\mathbf{x}), \quad (1)$$

где l — уровень разрешения, $\mathbf{k}$ — положение в пространстве, $\varphi_{\mathbf{k}}(\mathbf{x})$ — масштабирующая функция, $\mathbf{\Psi}_{l,\mathbf{k}}(\mathbf{x}) = \{\psi^n_{l,\mathbf{k}}(\mathbf{x}), n = 1,...,\alpha\}$ — вектор вейвлетов и $\mathbf{D}_{l,\mathbf{k}} = \{d^n_{l,\mathbf{k}}, n = 1,...,\alpha\}$ — вектор коэффициентов вейвлетов. Значения α, $\mathbf{k}$ и $\mathbf{x}$ зависят от размерности пространства: для трехмерного пространства $\alpha = 7$, $\mathbf{k} = (i, j, k)$, $\mathbf{x} = (x, y, z)$, для двухмерного пространства $\alpha = 3$, $\mathbf{k} = (i, j)$, $\mathbf{x} = (x, y)$ и для одномерного пространства $\alpha = 1$, $\mathbf{k} = (i)$, $\mathbf{x} = (x)$. Для ряда (1) разработано множество различные семейств вейвлетов и масштабирующих функций. Мы используем вейвлеты семейства Deslauriers-Dubuc [13]. Выбор данного семейства обусловлен возможностью как построения самих вейвлетов, так и вычисления их коэффициентов в разложении (1) с помощью специальной достаточно простой интерполяционной схемы [14].

В ряде (1) масштабирующие функции $\varphi_{\mathbf{k}}(\mathbf{x})$ используются для представления основных крупномасштабных особенностей поведения функции $f(\mathbf{x})$. Вейвлеты $\psi^n_{l,\mathbf{k}}$ описывают поведение функции на мелких масштабах. Чем выше уровень разрешения l, тем меньше вклад вейвлетов в функцию $f(\mathbf{x})$.

Поскольку вклад вейвлета $\psi^n_{l,\mathbf{k}}$ в ряд (1) определяется величиной его коэффициента $d^n_{l,\mathbf{k}}$, мы можем сжать представление функции путем отсечения вейвлетов, чьи коэффициенты меньше некоторого порога ε. Порог ε выбирается достаточно малым, чтобы удалить из ряда (1) только те вейвлеты, которые несут лишь несущественную информацию о поведении функции. В результате получаем разреженный ряд вейвлетов:

$$f^{\varepsilon}(\mathbf{x}) = \sum_{\mathbf{k}} b_{\mathbf{k}} \varphi_{\mathbf{k}}(\mathbf{x}) + \sum_{l=1}^{J} \sum_{|d|>\varepsilon} \mathbf{D}_{l,\mathbf{k}} \mathbf{\Psi}_{l,\mathbf{k}}(\mathbf{x}), \quad (2)$$

где $f^{\varepsilon}(\mathbf{x})$ — это аппроксимация функции $f(\mathbf{x})$ с ошибкой, задаваемой порогом ε, J — конечный максимальный уровень разрешения, который зависит от величины ε. Разреженный ряд (2) содержит лишь небольшое число значимых вейвлетов.

Из способа построения вейвлетов семейства Deslauriers-Dubuc вытекает взаимно-однозначное соответствие между вейвлетами и точками физического пространства, т.е. каждый вейвлет ассоциируется с определенной точкой. Поэтому разреженный ряд (2) задает множество точек пространства, которые образуют сетку со сгущениями в областях быстрого изменения значений функции.

Множество значимых вейвлетов описывает только текущее поведение функции в данный момент времени. Однако физические процессы развиваются со временем. Это проявляется в постоянном появлении новых деталей в пространственном распределении функции f, которые отсутствуют в данный момент времени и, следовательно, не могут быть описаны значимыми вейвлетами. Чтобы учесть возможные изменения в поведении функции, в разреженный ряд (2) необходимо добавить дополнительные вейвлеты, чьи коэффициенты в данный момент времени меньше порога ε, но могут пересечь его в ближайшем будущем. Вместе значимые и дополнительные вейвлеты образуют множество активных вейвлетов, которые используются при построении адаптивной сетки.

5. Структура данных

Важнейшей деталью любого программного пакета, производящего адаптивные сетки, является структура данных, используемая для представления этой сетки. Выбор структуры данных задает порядок работы с данными, способ их хранения в памяти компьютера и алгоритм доступа. Фактически правильный выбор структуры данных определяет степень эффективности программного пакета.

Структура данных — это совокупность некоторых однотипных элементов, хранящихся в памяти компьютера, и набор операций для работы с этими

элементами. Выделяют два подхода к созданию структур данных — статический и динамический.

Статическая структура данных остается неизменной все время работы с ее элементами. Она имеет фиксированный размер и размещается в памяти компьютера в момент запуска программы. В результате данная структура способна предоставить быстрый доступ к любому своему элементу. Однако такие операции как добавление и удаление новых элементов требуют переформирования сразу всей структуры данных, что занимает достаточно большой объем времени работы CPU. Примером статических структур данных служит массив.

Динамическая структура данных — это структура, которая может изменять свой размер во время работы программы. Каждый ее элемент размещается в памяти только тогда, когда появляется новая порция данных. Все неактивные элементы своевременно удаляются из памяти. Поэтому динамическая структура данных содержит только актуальную информацию и занимает минимально возможный объем памяти. В отличие от статической структуры здесь операции добавления и удаления элементов выполняются предельно быстро, но динамическое размещение в памяти очень сильно замедляет доступ к каждому отдельному элементу. Примерами динамических структур данных являются деревья и списки.

Однородная сетка — это неизменное множество узлов, занимающих одни и те же позиции в пространстве все время расчетов. Размер и положение однородной сетки задаются заранее и не могут быть изменены. Единственная операция, которая выполняется с сеткой — это доступ к ее узлам. Поэтому для однородной сетки наиболее подходящей является статическая структура данных.

Адаптивная сетка — это множество узлов переменного размера. Такая сетка постоянно добавляет и удаляет узлы, следуя за эволюцией моделируемого физического процесса. Любой численный алгоритм, основанный на адаптивной сетке, использует все основные операции (прямой доступ, вставка и удаление) при работе с ее узлами. Также адаптивная сетка имеет меняющуюся со временем нерегулярную форму.

Если адаптивную сетку представлять статической структурой данных регулярной формы (например, массивом), то эта структура должна содержать избыточное число элементов, чтобы учесть нерегулярную форму адаптивной сетки и минимизировать число операций вставки/удаления. Необходимое число элементов намного больше (примерно на 3-4 порядка), чем предельное число узлов сетки, которые реально используются в расчетах. Поэтому статические структуры данных крайне неэффективны для описания адаптивных сеток, поскольку программа забирает существенно больший объем памяти, чем реально использует.

Более эффективная форма представления адаптивных сеток — это динамическая структура данных. Ее свойства полностью отвечают всем

требованиям эффективного представления адаптивной сетки. Динамическая структура данных может иметь нерегулярную форму, и каждый ее элемент может быть быстро добавлен или удален при необходимости. Эти свойства позволяют нам размещать в памяти столько элементов данных, сколько мы имеем узлов в сетке, и легко модифицировать саму сетку во время расчетов. Единственный недостаток такого представления — это медленный доступ к узлам.

Эффективная реализация адаптивных алгоритмов подразумевает следующие требования к динамической структуре данных: 1) быстрый доступ к любому элементу структуры, 2) быстрое удаление старых и добавление новых элементов в структуре, 3) линейная зависимость размера памяти, занимаемой структурой данных, от размера адаптивной сетки, 4) возможность эффективного распараллеливания вычислительного алгоритма.

Как было упомянуто выше, наиболее подходящим выбором для представления адаптивной сетки является динамическая структура данных. Любую адаптивную сетку можно представить в виде дерева или связного списка. Каждая из этих структур имеет свои преимущества и недостатки. Дерево предоставляет быстрый доступ к своим элементам, а связный список более подходит для организации параллельных вычислений.

В нашей работе мы представили адаптивную сетку в виде комбинации дерева и связного списка (см. рис. 3), что позволяет воспользоваться их преимуществами и взаимно нейтрализовать их недостатки. Так, мы используем дерево для быстрого доступа к необходимым узлам сетки и связный список для параллельных вычислений. В последнем случае связный список делится на несколько дочерних списков, которые распределяются между параллельными процессорами.

Дерево — это структура данных, составленная из элементов, содержащих указатели на другие элементы. В дереве, использующемся для представления одномерной сетки, каждый элемент имеет не более 2 указателей на элементы, представляющие узлы сетки на следующем уровне разрешения. В двухмерном и трехмерном случаях каждый элемент имеет до 8 или 26 указателей на дочерние элементы с более высокого уровня разрешения, соответственно. Положение дочерних элементов в трехмерном случае показано на рис. 4.

Связный список представляет собой группу элементов, выстроенных в некоторой последовательности. Каждый элемент содержит данные и ссылку на следующий элемент списка.

Для представления дерева и связного списка в одномерном случае мы определили производный тип данных type Point ... end type Point. Каждый элемент содержит индексы (i,l), левый и правый указатели на дочерние элементы в дереве и указатель на следующий элемент в связном списке. Индексы элемента определяют его место в дереве (l — уровень, на котором находится элемент в дереве, считая от корня, i — индекс позиции элемента на

уровне l) и положение в пространстве соответствующего узла сетки как $x = i / 2^l$. Левый указатель ссылается на ближайший узел сетки с уровня разрешения $l+1$, чья пространственная координата меньше, чем координата данного элемента. Правый указатель ссылается на ближайший узел сетки с уровня разрешения $l+1$, чья пространственная координата больше, чем координата данного элемента. Каждый элемент типа Point одновременно имеет указатели, необходимые для его включения как в дерево, так и в связный список, т.е. мы не дублируем элементы — обе структуры данных состоят из одних и те же элементов. На рис. 3 каждый узел сетки однозначно идентифицируется парой (i,l), которая определяет его положение как в дереве, так и в связном списке.

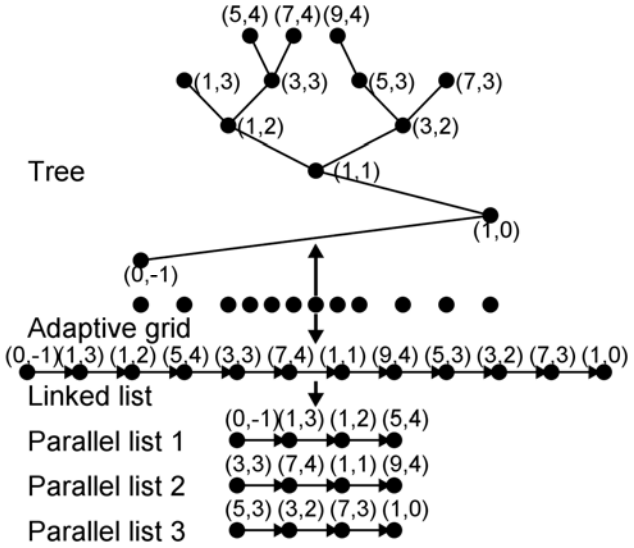


Рис. 3 Представление адаптивной сетки в виде дерева и связного списка.

Fig. 3. A representation of the adaptive grid in the form of a tree and a linked list.

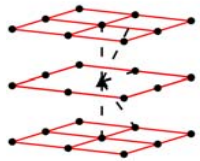


Рис. 4 Некоторый элемент (треугольник) со своими дочерними элементами (круги) в трехмерном дереве.

Fig. 4. A node (triangle) and its child nodes (circles) in the three-dimensional tree data structure

Двумерные и трехмерные сетки отличаются от одномерной одной и двумя дополнительными размерностями, соответственно. Здесь элементы характеризуются тремя или четырьмя индексами — (i, j, l) в двухмерном случае и (i, j, k, l) в трехмерном случае. Данные индексы определяются из пространственных координат соответствующих узлов сетки $x = i / 2^l$, $y = j / 2^l$ и $z = k / 2^l$. Также каждый элемент содержит 8 или 26 указателей на дочерние элементы на уровне $l+1$ в дереве и один указатель на следующий элемент в связном списке.

Для дискретизации пространственных производных наш пакет формирует конечно-разностные шаблоны до 8 порядка точности включительно. Для этого специальная процедура ищет элемент дерева, ближайший к данному, и принимает расстояние до него в пространстве за шаг шаблона. Далее процедура добавляет в шаблон все необходимые соседние элементы, расстояние до которых кратно шагу шаблона. В случае двух и трех измерений шаблоны формируются вдоль каждого направления. Заметим, что шаблоны в разных направлениях могут иметь разные шаги. В типе Point шаблон представлен массивом указателей на соответствующие элементы.

Для работы со структурами данных нам также нужны внешние ссылки на корень дерева и на первый элемент связного списка. Листинг программы для одномерного и трехмерного случаев приведен ниже:

1. Program listing for the 1D case in FORTRAN

```
type Child
        type(Point), pointer :: ref
end type Child
type Point
        integer :: i, l
        type(Child), allocatable, dimension(:) :: d_dx
        type(Child), dimension(1:2) :: links_ to_the_children_in_the_tree
        type(Point), pointer :: link_to_the_next_node_in_the_list
end type Point
type(Point), pointer :: head_of_the_linked_list
type(Point), pointer :: root_of_the tree
```

2. Program listing for the 3D case in FORTRAN

```
type Child
        type(Point), pointer :: ref
end type Child
type Point
        integer :: i, j, k, l
        type(Child), allocatable, dimension(:) :: d_dx,d_dy,d_dz
```



```
        type(Child), dimension(1:26) :: links_ to_the_children_in_the_tree
        type(Point), pointer :: link_to_the_next_node_in_the_list
end type Point
type(Point), pointer :: head_of_the_linked_list
type(Point), pointer :: root_of_the tree
```

6. Параллельная реализация

Любой численный метод, использующий адаптивные сетки, выполняется в два шага: построение адаптивной сетки и численное решение модельных уравнений. На обоих шагах выполняется последовательность некоторых операций в каждой точке адаптивной сетки. Эти операции берут текущие значения гидромеханических переменных в ближайших точках сетки и дают новое значение для данной точки:

$$c_k^{new} = f(c_1^{old}, c_2^{old}, ..., c_n^{old}), k = 1, 2, ..., n, \quad (3)$$

где c_k^{new} и c_k^{old} — новое и текущее значения гидромеханической переменной в точке k. Так как результаты вычисления c_k^{new} в данной точке не зависят от аналогичных результатов в других точках, каждая операция может выполняться в параллельном режиме.

Адаптивная сетка как множество точек служит базисом для параллельной реализации численных методов. С помощью отдельной процедуры множество узлов сетки делится на подмножества одинакового размера, которые в дальнейшем могут быть переданы параллельным процессорам для обработки. Тогда последовательность операций, составляющая численный алгоритм, будет выполняться каждым параллельным процессором на выделенном ему подмножестве точек сетки. Например, сетка на рис. 5 состоит из 12 узлов. Эти узлы распределяются между 3 процессорами так, что каждый процессор имеет 4 узла. Далее процессоры выполняют последовательно операции 1 и 2 только на принадлежащих им точках, т.е. одновременно обрабатываются 3 узла. Для каждого узла время выполнения отдельной операции одинаково. Также узлы могут обрабатываться в любой последовательности. Поэтому распределение узлов между процессорами произвольно. Например, на рис. 5 узлы распределены по процессорам последовательно слева направо: первый процессор обрабатывает узлы 1–4, второму принадлежат узлы 5–8, а третьему достаются последние узлы 9–12. Если бы мы распределили узлы в другой последовательности, общее время работы не изменилось бы.

7. Примеры расчетов

Мы применили разработанный программный пакет к решению нескольких физических задач: движение в последовательно сужающемся и расширяющемся потоке (рис. 6), перенос облака примеси через Тихий океан из Китая в США (рис. 7) и подъем капли толуола в воде (рис. 8).

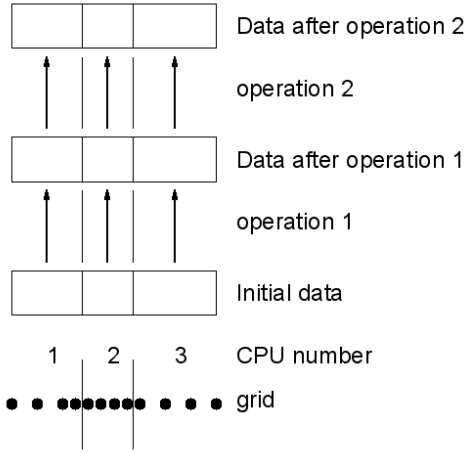


Рис. 5 Распределение точек сетки между процессорами.

для проведения параллельных вычислений

Fig. 5. Distribution of grid nodes among CPUs for the parallel implementation.

Движение в сужающемся-расширяющемся потоке и в атмосфере описывается одним уравнением адвекции массовой доли некоторого модельного вещества, которая использовалась в качестве входной переменной программного пакета для построения адаптивной сетки. Движение капли толуола в воде описывается полной системой уравнений гидромеханики (уравнение неразрывности, уравнение импульса и уравнение адвекции характеристической функции). Адаптивная сетка в этом случае строилась по двум переменным — характеристической функции и кинетической энергии.

Рис. 6–8 показывают, что адаптивная сетка, построенная с помощью разработанного программного пакета, подробно разрешает участки, где гидромеханические переменные претерпевают резкие изменения (внутри облака примеси и капли), и использует крупный шаг между узлами в остальной области. Адаптивная сетка имеет минимально возможный размер (на 3–5 порядков меньший, чем эквивалентная ей однородная сетка), при этом она достаточно велика, чтобы учитывать все возможные динамические изменения в потоке и не вносить каких-либо искажений. Например, численные эксперименты показали, что постоянная перестройка сетки в ходе решения приводит к крайне незначительной ошибке в сохранении массы, не превосходящей по ходу расчетов 0.02%.

Детальный анализ двухмерного движения облака примесей над Тихим океаном может быть найден в [15]. Анализ трехмерного движения облака примесей в атмосфере и движения капли толуола в воде будет опубликован в последующих статьях.

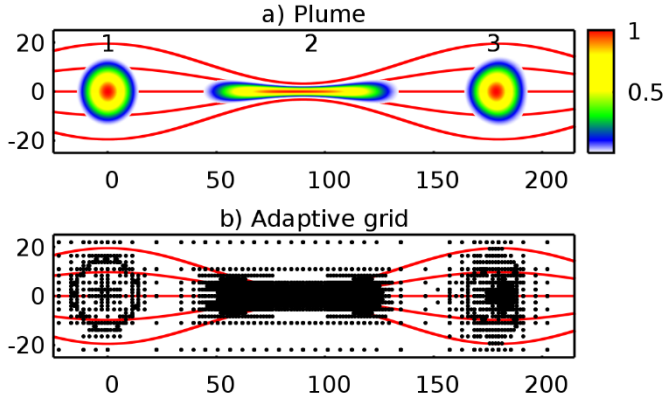


Рис. 6 Движение облака примесей (a) и эволюция соответствующей адаптивной сетки (b) в сужающемся-расширяющемся потоке для различных моментов времени (1), (2), (3).

Fig. 6. Evolution of (a) a plume and (b) the corresponding adaptive grid in the convergent-divergent flow at different time moments (1), (2), (3).

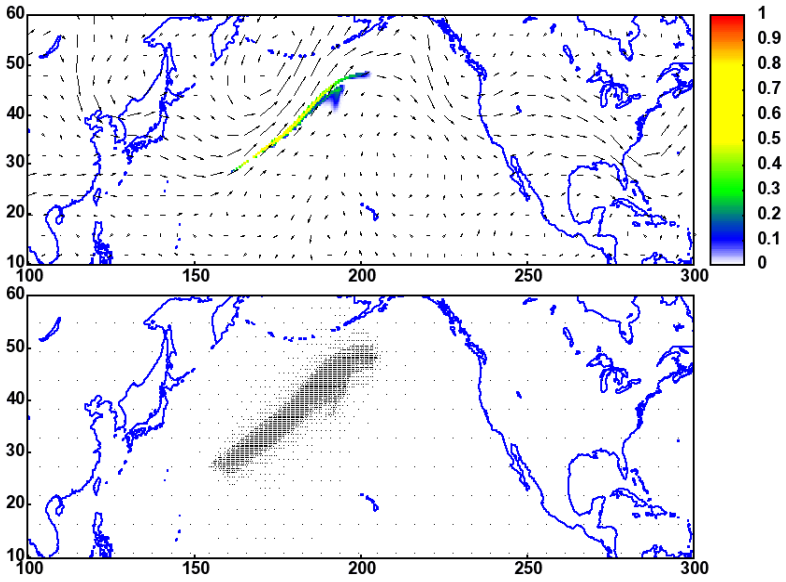


Рис. 7 Перенос облака примесей над Тихим океаном 1-ого мая 2001 года (верхняя панель) и соответствующая адаптивная сетка (нижняя панель).

Fig. 7. Pollution plume transport over the Pacific Ocean in the free troposphere on May 1, 2001 (the upper panel) and the corresponding adaptive grid (the lower panel).

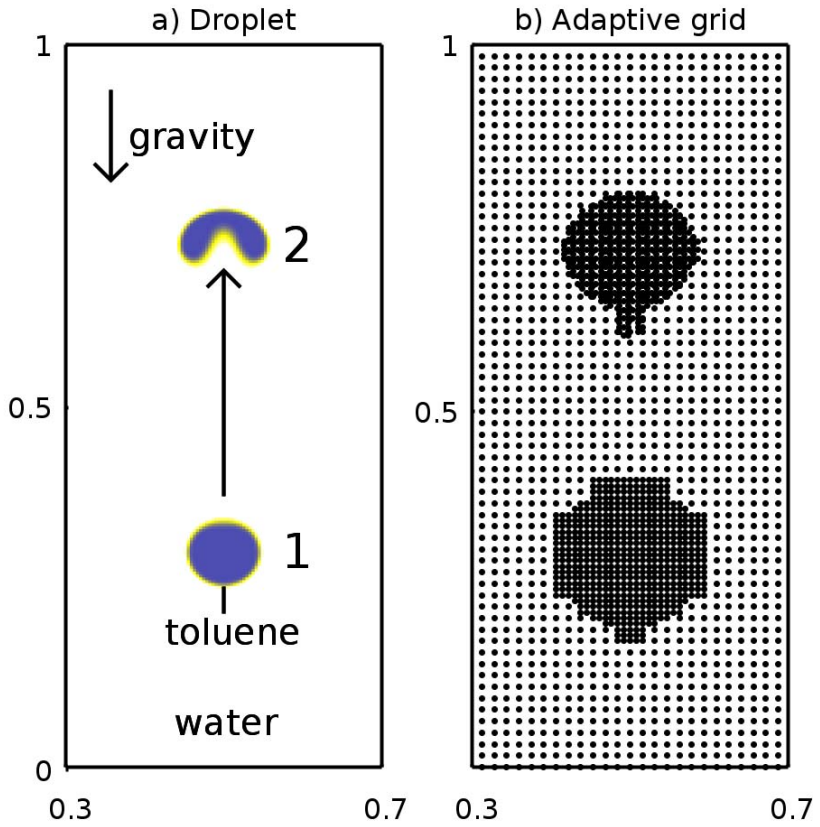


Рис. 8 Подъем капли толуола в воде. Здесь капля и соответствующая адаптивная сетка показаны для начального (1) и некоторого промежуточного (2) моментов времени.

Fig. 8. Evolution of a toluene droplet rising in water. Here the droplet and the corresponding adaptive grid are shown for the initial time moment (1) and some intermediate time moment (2) in the left and right panels, respectively.

8. Заключение

Мы разработали программный пакет для создания одно-, двух- и трехмерных адаптивных сеток. Адаптация основана на теории вейвлетов и осуществляется через разложение одной или более гидромеханических переменных в ряд вейвлетов. В компьютерной памяти после построения адаптивная сетка представляется в виде комбинации двух разных динамических структур данных

— дерева и связного списка. Совместное использование двух структур данных позволяет поддерживать соответствие между пространственной структурой сетки и ее программным представлением, при необходимости получать быстрый доступ к любой точке сетки и эффективно распараллеливать вычисления.

Программный пакет был применен к моделированию нескольких физических проблем. Было показано, что адаптивная сетка, построенная нашим пакетом, позволяет получать численное решение хорошего качества и при этом поддерживать размер сетки на минимально приемлемом уровне.

Список литературы

[1]. MacNeice P., Olson K. M., Mobarry C., deFainchtein R., Packer C. PARAMESH: A parallel adaptive mesh refinement community toolkit. Comput. Phys. Commun., 126, 2000, pp. 330–354.

[2]. Berger M. J., LeVeque R. J. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. SIAM J. Numer. Anal., 35, 1998, pp. 2298–2316.

[3]. Deiterding R., Radovitzky R., Mauch S. P., Noels L., Cummings J. C., Meiron D. I. A virtual test facility for the efficient simulation of solid material response under strong shock and detonation wave loading. Eng. Comput., 22, 2006, pp. 325–347.

[4]. Bryan G. L. et al. Enzo: An adaptive mesh refinement code for astrophysics. 2013, arXiv:1307.2265.

[5]. Neeman H. Autonomous hierarchical adaptive mesh refinement for multiscale simulations. Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1996.

[6]. Parashar M., Browne J. C. On partitioning dynamic adaptive grid hierarchies. Proceedings of the 29th Annual Hawaii International Conference on System Sciences, 1996.

[7]. Quinlan D. Adaptive mesh refinement for distributed parallel architectures. Ph.D. dissertation, University of Colorado at Denver., 1993.

[8]. Hornung R. D., Kohn S. R. Managing application complexity in the SAMRAI object-oriented framework. Concurr. Comp. Pract. E., 14, 2002, pp. 347–368.

[9]. Walder R., Folini D. A-MAZE: A code package to compute 3D magnetic flows, 3D NLTE radiative transfer, and synthetic spectra. Thermal and Ionization Aspects of Flows from Hot Stars: Observations and Theory, ASP Conference Series, 204, 2000, pp. 281–284.

[10]. Almgren A. S., Bell J. B., Colella P., Howell L. H., Welcome M. L. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. J. Comp. Phys., 142, 1998, pp. 1–46.

[11]. Popinet S., Rickard G. A tree-based solver for adaptive ocean modeling. Ocean Model., 16, 2007, pp. 224–249.

[12]. Holmstrom M. Solving hyperbolic PDEs using interpolating wavelets. SIAM J. Sci. Comput., 21, 1999, pp. 405–420.

[13]. Donoho D. L. Interpolating wavelet transforms. Department of Statistics. Stanford University, Tech. Rep., 1992.

[14]. Deslauriers G., Dubuc S. Symmetric iterative interpolation processes. Constr. Approx., 5, 1989, pp. 49–68.

[15]. Semakin A. N., Rastigejev Y. Numerical simulation of global-scale atmospheric chemical transport with high-order wavelet-based adaptive mesh refinement algorithm. Mon. Wea. Rev., 144, 2016, pp. 1469–1486.

Software for adaptive grid construction

A.N. Semakin <arte-semaki@yandex.ru>
Bauman Moscow State Technical University,
5, Second Bauman st., Moscow, 105005, Russia.

Abstract. In this paper, we present a software package for the construction of an adaptive finite-difference grid by expanding physical variables into a sparse wavelet series. Some technical details of the program implementation are given. In particular, we describe data structures used for the grid representation in computer memory and tools for organizing parallel calculations on the adaptive grid. Also the results of several numerical simulations involving the developed package are shown.

Keywords: software; wavelets; adaptive grids.

DOI: 10.15514/ISPRAS-2017-29(5)-15

For citation: Semakin A.N. Software for adaptive grid construction. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 5, 2017. pp. 311-328 (in Russian). DOI: 10.15514/ISPRAS-2017-29(5)-15

References

[1]. MacNeice P., Olson K. M., Mobarry C., deFainchtein R., Packer C. PARAMESH: A parallel adaptive mesh refinement community toolkit. Comput. Phys. Commun., 126, 2000, pp. 330–354.
[2]. Berger M. J., LeVeque R. J. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. SIAM J. Numer. Anal., 35, 1998, pp. 2298–2316.
[3]. Deiterding R., Radovitzky R., Mauch S. P., Noels L., Cummings J. C., Meiron D. I. A virtual test facility for the efficient simulation of solid material response under strong shock and detonation wave loading. Eng. Comput., 22, 2006, pp. 325–347.
[4]. Bryan G. L. et al. Enzo: An adaptive mesh refinement code for astrophysics. 2013, arXiv:1307.2265.
[5]. Neeman H. Autonomous hierarchical adaptive mesh refinement for multiscale simulations. Ph.D. dissertation, University of Illinois at Urbana-Champaign, 1996.
[6]. Parashar M., Browne J. C. On partitioning dynamic adaptive grid hierarchies. Proceedings of the 29th Annual Hawaii International Conference on System Sciences, 1996.
[7]. Quinlan D. Adaptive mesh refinement for distributed parallel architectures. Ph.D. dissertation, University of Colorado at Denver., 1993.
[8]. Hornung R. D., Kohn S. R. Managing application complexity in the SAMRAI object-oriented framework. Concurr. Comp. Pract. E., 14, 2002, pp. 347–368.
[9]. Walder R., Folini D. A-MAZE: A code package to compute 3D magnetic flows, 3D NLTE radiative transfer, and synthetic spectra. Thermal and Ionization Aspects of Flows from Hot Stars: Observations and Theory, ASP Conference Series, 204, 2000, pp. 281–284.

[10]. Almgren A. S., Bell J. B., Colella P., Howell L. H., Welcome M. L. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. J. Comp. Phys., 142, 1998, pp. 1–46.
[11]. Popinet S., Rickard G. A tree-based solver for adaptive ocean modeling. Ocean Model., 16, 2007, pp. 224–249.
[12]. Holmstrom M. Solving hyperbolic PDEs using interpolating wavelets. SIAM J. Sci. Comput., 21, 1999, pp. 405–420.
[13]. Donoho D. L. Interpolating wavelet transforms. Department of Statistics. Stanford University, Tech. Rep., 1992.
[14]. Deslauriers G., Dubuc S. Symmetric iterative interpolation processes. Constr. Approx., 5, 1989, pp. 49–68.
[15]. Semakin A. N., Rastigejev Y. Numerical simulation of global-scale atmospheric chemical transport with high-order wavelet-based adaptive mesh refinement algorithm. Mon. Wea. Rev., 144, 2016, pp. 1469–1486.