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

Препринты ИПМ • Препринт № 87 за 2015 г.

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

Жуков В.Т., Краснов М.М., Новикова Н.Д., Феодоритова О.Б.

Численное решение параболических уравнений на локально-адаптивных сетках чебышевским методом

Рекомендуемая форма библиографической ссылки: Численное решение параболических уравнений на локально-адаптивных сетках чебышевским методом / В.Т.Жуков [и др.] // Препринты ИПМ им. М.В.Келдыша. 2015. № 87. 26 с.
URL: http://library.keldysh.ru/preprint.asp?id=2015-87

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

В.Т. Жуков, М.М. Краснов, Н.Д. Новикова,
О.Б. Феодоритова

Численное решение параболических уравнений на локально-адаптивных сетках чебышевским методом

Москва — 2015

Жуков В.Т., Краснов М.М., Новикова Н.Д., Феодоритова О.Б.

Численное решение параболических уравнений на локально-адаптивных сетках чебышевским методом

Рассматриваются некоторые аспекты решения параболических уравнений на декартовых локально-адаптивных сетках. Дано изложение основных элементов алгоритма, приведены результаты экспериментального исследования точности построенной схемы в модельных задачах с непрерывными и разрывными коэффициентами при использовании для интегрирования по времени явно-итерационной чебышевской схемы ЛИ-М. Демонстрируется работоспособность библиотеки сеточных операторов, обеспечивающей перенос программы на графические ускорители CUDA и не требующей от пользователя глубокого знания методов программирования для CUDA.

Ключевые слова: параболические уравнения, чебышевский метод, локально-адаптивные сетки, операторный метод программирования

Victor Timofeevich Zhukov, Mikhail Mikhailovich Krasnov, Natalia Dmitrievna Novikova, Olga Borisovna Feodoritova

Numerical solution of parabolic equations on locally-adaptive grids by Chebyshev method

Some aspects of the solution of parabolic equations on Cartesian locally adaptive grids are considered. We represent main features of the algorithm and describe some results of experimental study of accuracy the presented computational technique. For time integration it is used the explicit Chebyshev iterative scheme LI-M. The results of numerical experiments are given for model problems including case of continuous and discontinuous coefficients. We demonstrate performance of the grid operator library, which ensures the transfer of the program on graphics accelerators CUDA and does not require from a user deep knowledge of programming techniques for CUDA.

Key words: parabolic equations, Chebyshev method, locally-adaptive grid, operator programming method

Работа выполнена при финансовой поддержке Российского научного фонда (проект № 14-21-00025).

1. Введение

Среди различных подходов для решения задач математической физики значительный интерес проявляется к вычислительным моделям на адаптивных сетках с локальным измельчением.

Моделируемые задачи зачастую имеют сложную геометрию, локализованные области с разными физическими свойствами, и численное решение таких задач требует очень подробных сеток, что приводит к значительному росту вычислительных затрат. Выборочное измельчение элементов сетки может оказаться компромиссным решением. Возникает вопрос: насколько локально-адаптивные сетки удобны для реализации параллельных вычислений, ведь сейчас требуются коды для суперкомпьютеров с большим числом процессоров. И такие коды должны обеспечивать высокую параллельную эффективность, то есть, как принято говорить, масштабируемое моделирование на расчетных сетках с миллиардом (и более) узлов.

Данная работа посвящена исследованию одного подхода к численному решению параболических уравнений, достаточно перспективного в аспекте ультрапараллельных вычислений. Ключевое место этого подхода – использование для численного интегрирования по времени явно-итерационной схемы ЛИ-М [1]. Это схема разработана для решения параболических уравнений, опирается на многочлены Чебышева специальной конструкции и ранее прошла широкую практическую проверку, см. [1– 3]. В данной работе эта схема впервые используется на декартовых локально-адаптивных сетках (ЛАД–сетках). Заметим, что применение неявной схемы само по себе не гарантирует высокую точность описания эволюции решения во времени, и, кроме того, возникает необходимость решения систем линейных алгебраических уравнений, что для многомерных задач является серьезной проблемой. В особенности это относится к неструктурированным сеткам, в частности, к ЛАД–сеткам.

Сложность решения параболических уравнений принято характеризовать «параболическим» числом Куранта $cou \approx \tau \cdot \lambda_{\max}$, здесь $\lambda_{\max}$ – максимальное собственное значение разностного эллиптического оператора. Для 3D задач на регулярных сетках вычислительные затраты схемы ЛИ-М суть $C\sqrt{\tau \cdot \lambda_{\max}} \cdot h^{-3} \approx C\sqrt{\tau \cdot h^{-2}} \cdot h^{-3}$, и константа в этой оценке зависит только от задачи. На ЛАД-сетках оценка вычислительных затрат схемы ЛИ-М на одном шаге по времени имеет вид $C\sqrt{\tau \cdot \lambda_{\max}} \cdot J$, где J – число ячеек сетки, а $\lambda_{\max}$ определяется самой «плохой» ячейкой, то есть ячейкой, для которой отношение коэффициента диффузии к квадрату диаметра является максимальным. И это может оказаться определенным препятствием для использования схемы ЛИ-М на ЛАД-сетках, несмотря на сохранения ультрапараллельных возможностей. В данной работе этот аспект пока не изучается – главное внимание уделяется изложению основных элементов алгоритма, исследованию реальной точности

построенной ЛАД–ЛИ-М схемы в эволюционных задачах, где параболическое число Куранта относительно невелико. Дополнительно исследуется эффективность использования сеточной библиотеки [4], обеспечивающей перенос программы на графические ускорители CUDA и не требующей от пользователя глубокого знания методов программирования для CUDA.

Итак, для разрешения малых деталей геометрии расчетной области и, возможно, особенностей решения, используется прямоугольная адаптивная локально-измельченная сетка, которая требует среди прочего решения различных подзадач. К ним, например, относятся умение определять соседние ячейки вокруг заданной, аппроксимировать на такой сетке исходное дифференциальное уравнение и другие. Заметим, что круг интересующих нас задач в настоящий момент не предполагает динамического локального перестроения сеток.

Часть тестовых задач решена на базе операторной библиотеки [4], созданной для работы с сеточными функциями, определенными на произвольных трехмерных сетках. Библиотека спроектирована так, чтобы детали ее реализации были скрыты от пользователя, что позволяет эффективно реализовать ее на машинах с различной архитектурой (параллельных, гибридных и т.п.). Созданная библиотека приближает внешний вид программ к формулам в теоретических работах и относительно проста в использовании, в том числе при работе на графических платах с архитектурой CUDA.

2. Математическая постановка задачи

Рассмотрим параболическое уравнение

$$\frac{\partial u}{\partial t} = div\,(\kappa\ grad\ u) - a \cdot u + f\ . \qquad (1)$$

Будем искать решение этого уравнения в трехмерной области $\Omega = [x_0\ ;\ x_1] \times [y_0\ ;\ y_1] \times [z_0\ ;\ z_1]$, представляющей из себя прямоугольный параллелепипед, при условии, что на всех гранях расчетной области задан (для простоты изложения) нулевой поток по направлению внешней нормали к границе области. Функции $\kappa(r) \geq 0,\ f(r),\ a(r) \geq 0$ являются заданными, $r = (t,\ x, y, z) \in [t_0; T] \times \Omega$, $[t_0;\ T]$ – заданный интервал времени. В общем случае допускаются краевые условия первого – третьего родов, $\kappa(r)$ может быть тензором, зависящим от решения, то есть возможны различные обобщения.

Предполагается, что расчетная область может содержать зоны со сложной геометрией, отличной от прямоугольных параллелепипедов. Эти зоны могут иметь различные физические свойства, например, на границах зон тензор диффузии κ может быть разрывным и его скачок на поверхности разрыва может быть большим. Тогда в окрестности поверхности разрыва решение имеет большие градиенты, для сеточного воспроизведения которых обычно требуются подробные сетки.

3. Вычислительная постановка задачи

3.1. Локально-адаптивные сетки

При построении сетки в расчетной области мы считаем, что она состоит из прямоугольных параллелепипедов с возможным локальным подсеточным разрешением, диктуемым условиями задачи.

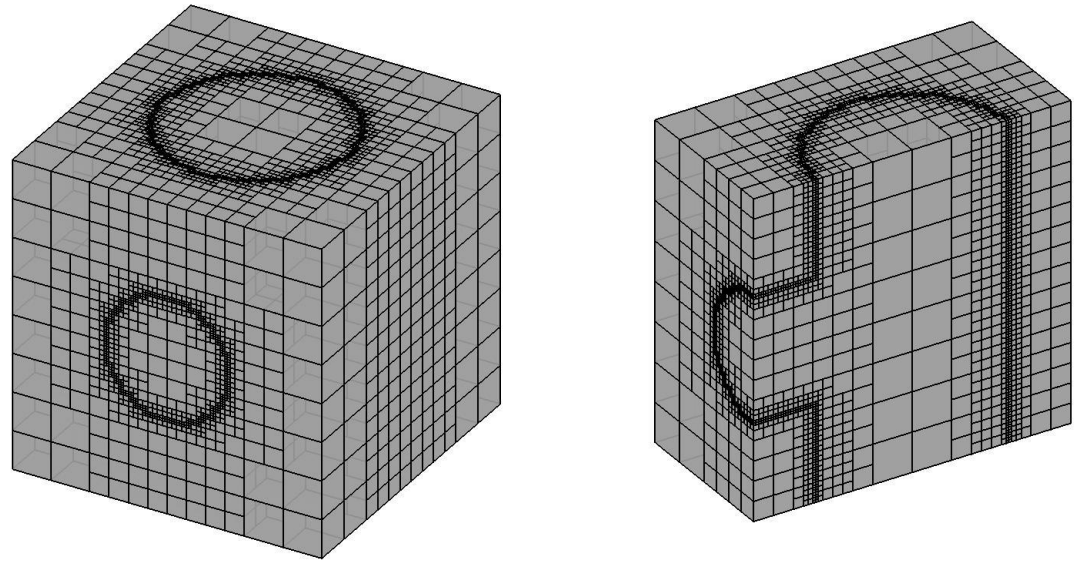
Сущность технологии построения локально-адаптивных сеток заключается в следующем. Начальная сетка является декартовой, и все ее ячейки являются прямоугольными параллелепипедами. Затем в соответствии с заданными критериями выделяются подобласти с особенностями геометрии или решения, и в этих подобластях строится более мелкая, по сравнению с исходной, сетка. Мы считаем для определенности, что выделяемая особенность задается какой-либо поверхностью. Если расчетная ячейка лежит в зоне влияния выделенной особенности, например, пересекается поверхностью, то такая ячейка делится на 8 равных ячеек. Далее, если необходимо, ячейки делятся еще раз, и так до достижения необходимой точности. Ячейки начальной сетки называются ячейками уровня 0, ячейки, получаемые измельчением уровня 0, называются ячейками уровня 1 и т.д.

На рис. 1 приведен пример 7-уровневой декартовой локально-адаптивной сетки, построенной в единичном кубе, внутрь которого помещен а) «тройник» – два пересекающихся цилиндрических стержня и б) шар радиуса R. Около поверхностей, помещенных внутрь куба (в нашем примере это – цилиндры и сфера), образованы раздробленные ячейки. Дополнительно можно увидеть фрагменты сетки в некотором вертикальном сечении области, где представлено внутреннее строение сетки в обоих случаях.

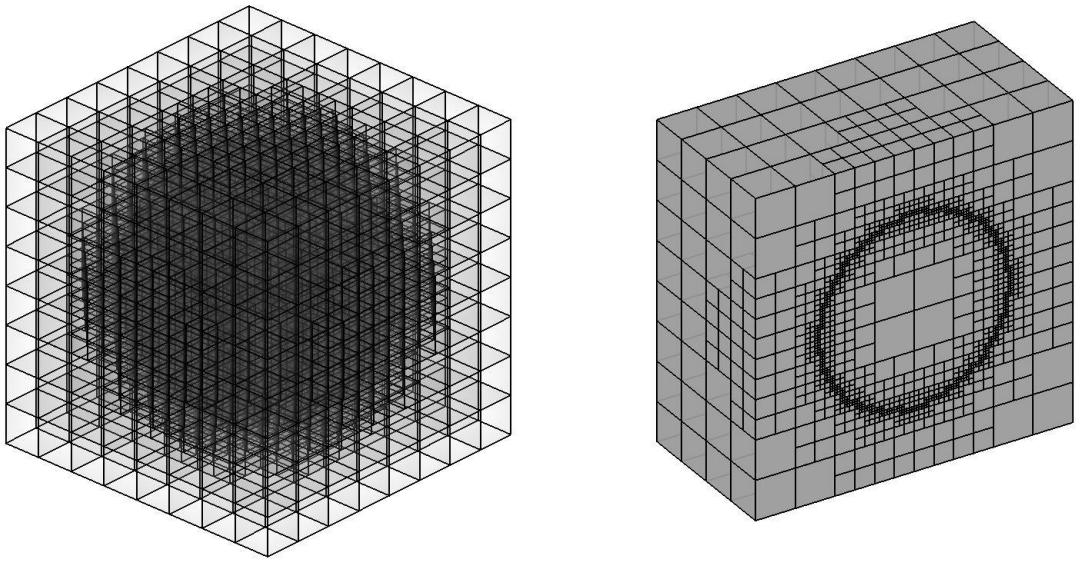
Критерии адаптации могут быть различными и определяться в соответствии с поставленными целями и имеющейся априорной информацией о задаче. Мы приводим здесь один из возможных вариантов. Критерий адаптации в случае «вложенного шара» (рис. 1б) имеет вид

$$D_a \cdot D_b < 0,$$
$$D_a = (x_a - x_0)^2 + (y_a - y_0)^2 + (z_a - z_0)^2 - R^2,$$
$$D_b = (x_b - x_0)^2 + (y_b - y_0)^2 + (z_b - z_0)^2 - R^2.$$

Здесь (x_a, y_a, z_a), (x_b, y_b, z_b) – координаты концов анализируемого отрезка, R – радиус вложенного шара, а (x_0, y_0, z_0) – координаты его центра. После окончания анализа всех ребер сетки исследуемого уровня переходим к анализу ячеек. Сначала рассматриваются все грани куба, и если хотя бы одно ребро грани оказалось разделенным, грань отмечается как кандидат для деления на четыре части. Кубическая ячейка разбивается на 8 частей, если любые две его противоположные грани отмечены для разделения. Все грани такого куба делятся на четыре части.



а)



б)

Рис. 1. Примеры локально-адаптивных сеток – а) два пересекающихся цилиндра б) сфера – внутри единичного куба; 7 уровней адаптации

При генерации сеток необходимо накладывать дополнительное ограничение: в окрестности каждой ячейки не должно быть ячеек, которые отличаются от неё по размерам более чем в два раза. Ниже на рис. 2 приведен пример недопустимой сеточной конфигурации.

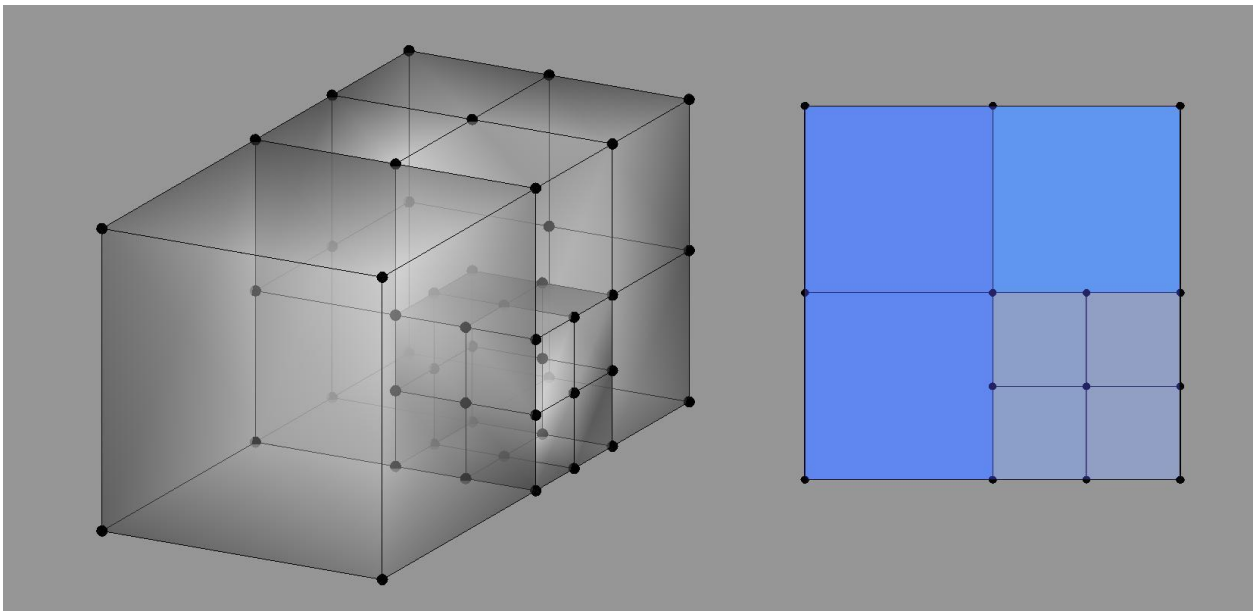


Рис. 2. Пример запрещенной декартовой локально-адаптивной сетки. Справа изображена общая грань между двумя ячейками 0-го уровня

Общая грань ячеек одного сеточного уровня не должна содержать более одного внутреннего узла. В примере на рис. 2 таких узлов оказалось 4. В этом случае включается процедура сглаживания сетки, задачей которой является проведение дополнительного разбиения (объединения) ячеек с целью ликвидации указанной ситуации. Из-за этого ограничения каждая ячейка может иметь от 4 до 24 соседей, и такая определенность значительно упрощает поиск соседей. Также не может быть одновременно больших и маленьких соседних ячеек.

Можно предположить, что в некоторых частях области возможно чрезмерное измельчение сетки, и в этом случае разумно ввести ограничение на минимальный размер ячейки. Можно зафиксировать отдельные ячейки, чтобы они не разбивались и не объединялись. В общем случае стратегия построения локальной адаптации многопараметрическая и требует дальнейшего развития.

3.2. Дискретизация

Рассмотрим уравнение теплопроводности в следующей форме:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(a_{xx}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(a_{yy}\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(a_{zz}\frac{\partial u}{\partial z}\right) - a \cdot u + f. \qquad (2)$$

Для получения дискретных уравнений используется метод конечных объемов. Сеточная функция задана в центрах ячеек, и ячейка консервативности совпадает с ячейкой сетки. Запишем дискретизацию уравнения (2) в виде

$$\frac{\hat{u}_i - u_i}{\tau} = \frac{1}{V_i} \sum_{face=1}^{6} \Delta^i_{face} S^i_{face} - a_i \, \hat{u}_i + f_i \, . \qquad (3)$$

Индекс i нумерует сеточную функцию в центре ячейки сетки, $\hat{u}_i$ – искомое значение функции на верхнем временном слое $\hat{t} = t + \tau$, S^i_{face} – площадь боковой грани ячейки, Δ^i_{face} – удельный диффузионный поток через боковую грань ячейки в направлении внешней нормали, V_i – объем i-ой ячейки сетки.

При дискретизации мы можем столкнуться с двумя геометрическими ситуациями. Назовем их в дальнейшем регулярным и нерегулярным случаями.

Сначала рассмотрим регулярный случай. Геометрия ячеек для этого варианта изображена на рис. 3. Обозначим исследуемую ячейку – “base”, а соседнюю с ней – “ne” (neighbor). Центр их общей грани “face” имеет индекс “f”.

Коэффициент диффузии D_f в центре грани рассчитывается как среднее геометрическое коэффициентов в центрах соседних ячеек D_{base} , D_{ne}

$$D_f = \frac{D_{base} \, D_{ne} (l_{base} + l_{ne})}{D_{base} l_{ne} + D_{ne} l_{base}} \, .$$

Здесь l_{base}, l_{ne} – расстояния между центром грани “face” и центрами соседних ячеек “base”, “ne” соответственно.

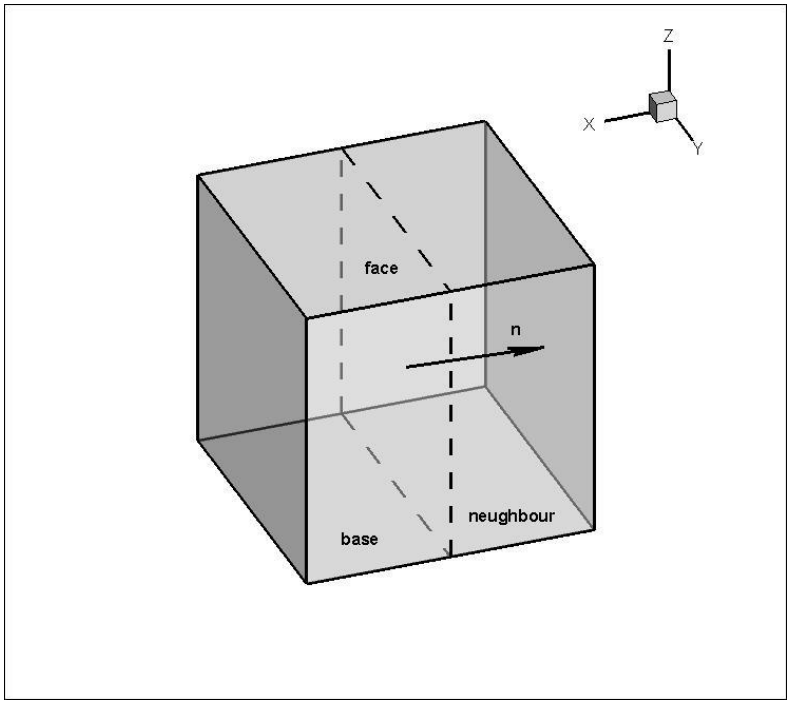


Рис. 3. Регулярный случай: соседние ячейки принадлежат одному сеточному уровню

Удельный диффузионный поток $\Delta^i_{face} = (grad\ u, \vec{n})$ аппроксимируем простейшим способом

$$\Delta^i_{face} = D_f \frac{u_{ne} - u_{base}}{0.5(l_{base} + l_{ne})} .$$

Если все соседние ячейки принадлежат одному сеточному уровню, то получаем простейшую семиточечную схему на ортогональных сетках. Разностные коэффициенты рассчитываются по формулам

$$A_{ne} = V_i \frac{D_f}{0.5(l_{base} + l_{ne})} S^i_{face} = \frac{D_f}{(L_{base} + L_{ne})L_{base}}, \quad A_{base} = -A_{ne} ,$$

где $L_{base} = 2l_{base}$, $L_{ne} = 2l_{ne}$ – размер ячейки в направлении нормали.

Несколько более сложным является нерегулярный случай, представленный на рис. 4.

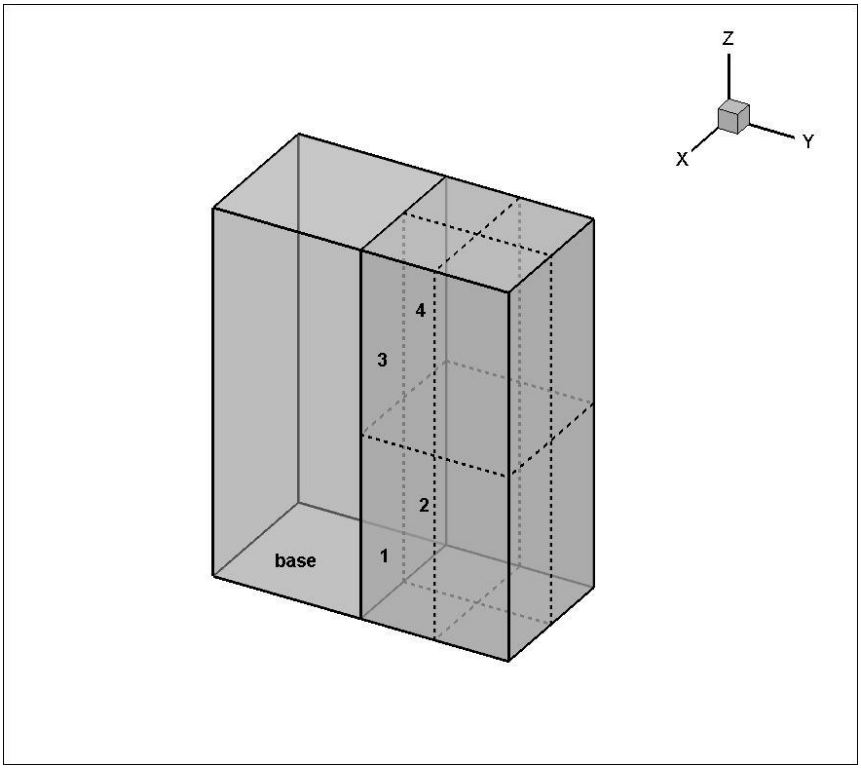


Рис. 4. Нерегулярный случай: соседние ячейки принадлежат разным сеточным уровням

В этом случае боковая грань, поток через которую нас интересует, разделяется на 4 равные по площади части (на рисунке они отмечены цифрами 1–4). Коэффициенты диффузии D_f на грани в этом случае считаются по следующим формулам:

$$DD = \frac{1}{4}\left(\frac{1}{D_1}+\frac{1}{D_2}+\frac{1}{D_3}+\frac{1}{D_4}\right), \quad D_{ne} = \frac{1}{DD}, \quad D_f = \frac{D_{base}\, D_{ne}(L_{base}+L_{ne})}{D_{base}L_{ne}+D_{ne}L_{base}}.$$

Как и прежде L_{base}, $L_{ne} = L_1 = L_2 = L_3 = L_4$ – размеры всех соседствующих ячеек.

Удельный диффузионный поток определяется как

$$\Delta^i_{face} = \frac{D_f}{L_{base}+L_{ne}}\left(\frac{u_1+u_2+u_3+u_4}{4}-u_{base}\right) = \frac{\Delta_1+\Delta_2+\Delta_3+\Delta_4}{4},$$

где

$$\Delta_1 = D_f\,\frac{u_1-u_{base}}{0.5(L_{base}+L_1)}, \qquad \Delta_2 = D_f\,\frac{u_2-u_{base}}{0.5(L_{base}+L_2)},$$

$$\Delta_3 = D_f\,\frac{u_3-u_{base}}{0.5(L_{base}+L_3)}, \qquad \Delta_4 = D_f\,\frac{u_4-u_{base}}{0.5(L_{base}+L_4)}.$$

В случае если ячейка граничит только с ячейками следующего уровня, шаблон схемы содержит 25 точек. Разностные коэффициенты считаются по формулам

$$A_1 = V_i\,\frac{D_f}{0.5(l_{base}+l_1)}\,\frac{S^i_{face}}{4} = \frac{D_f}{4(L_{base}+L_1)L_{base}},$$

$$A_2 = V_i\,\frac{D_f}{0.5(l_{base}+l_2)}\,\frac{S^i_{face}}{4} = \frac{D_f}{4(L_{base}+L_2)L_{base}},$$

$$A_3 = V_i\,\frac{D_f}{0.5(l_{base}+l_3)}\,\frac{S^i_{face}}{4} = \frac{D_f}{4(L_{base}+L_3)L_{base}},$$

$$A_4 = V_i\,\frac{D_f}{0.5(l_{base}+l_4)}\,\frac{S^i_{face}}{4} = \frac{D_f}{4(L_{base}+L_4)L_{base}},$$

$$A_{base} = -\,(A_1+A_2+A_3+A_4).$$

4. Схема интегрирования по времени

Для параболических уравнений универсальной является явная схема, но она используется крайне редко из-за известного ограничения на шаг по времени. Мы используем явно-итерационную схему ЛИ-М. Эта схема построена для решения параболических уравнений, она опирается на оптимальные свойства многочлена Чебышева специальной конструкции и ранее прошла широкую практическую проверку, см. [1– 3]. В данной работе эта схема впервые используется на декартовых локально-адаптивных сетках, и мы ее позиционируем как конкурента неявной схемы, особенно в контексте параллельных вычислений на компьютерах с экстрамассивным параллелизмом, в том числе гибридных с графическими сопроцессорами. Заметим, что применение неявной схемы само по

себе не гарантирует высокую точность описания эволюции решения во времени, и, кроме того, возникает необходимость решения систем линейных алгебраических уравнений, что для многомерных задач является серьезной проблемой. В особенности это относится к неструктурированным сеткам, в частности, к ЛАД–сеткам, нерегулярный многоточечный шаблон которых наследуется разреженной матрицей.

Запишем схему ЛИ-М для лаконичности объяснения в случае однородного уравнения

$$\frac{\partial u}{\partial t} + L_h u = 0. \qquad (4)$$

В этой схеме переход с нижнего слоя по времени t_j на верхний слой $t_{j+1} = t_j + \tau$ делается с помощью $\nu = 2p - 1$ явных шагов. В операторном виде этот переход можно записать как $u_{j+1} = S \cdot u_j$, где

$$S = (I - F_p^2) \cdot (I + \tau L_h)^{-1} \qquad (5)$$

является оператором послойного перехода, определяемым многочленом Чебышева $F_p(L_h)$ специальной конструкции [1] алгебраической степени

$$p = \left\lceil \frac{\pi}{4} \sqrt{\tau \lambda_{max} + 1} \right\rceil .$$

Вместо операторной записи схему ЛИ-М удобно объяснить в терминах декрементной функции

$$\rho_{\text{ЛИ–М}}(\lambda) = \frac{1 - F_p^2(\lambda)}{1 + \tau\lambda}, \qquad (6)$$

значения которой на спектре оператора L_h являются собственными значениями оператора перехода S; здесь $\lambda \in [0; \lambda_{max}]$ – отрезок вещественной оси, содержащий спектр оператора L_h. Многочлен F_p, определяющий эту функцию, обладает свойством оптимальности (наименьшего уклонения от 0) на отрезке, включающем спектр, и этот многочлен ограничен: $|F_p| \le 1$. На начальном участке спектра функция $\rho_{\text{ЛИ–М}}(\lambda)$ аппроксимирует декрементную функцию точного оператора перехода $exp(-\tau \cdot \lambda_h)$. Схема ЛИ-М требует задания верхней оценки максимального собственного значения λ_{max} дискретного оператора L_h; вычисление этой оценки не представляет труда и производится на основе теоремы Гершгорина о кругах. Схема ЛИ-М алгоритмически проста, не требует задания дополнительных настроечных параметров, может быть использована при произвольной сеточной дискретизации. Эта схема принадлежит к классу

явно-итерационных методов со скалярными параметрами и реализует переход с нижнего слоя по времени t на верхний слой $t+\tau$ за конечное число $\nu = 2p-1$ явных итераций. Алгоритм определяется величинами τ и λ_{max}; с их помощью однозначно определяется многочлен F_p, используемый для явно-итерационного конструирования дробно-рациональной аппроксимации (6) точного оператора перехода $exp(-\tau L_h)$. С другой стороны, у схемы ЛИ-М есть два принципиальных свойства, отличающих ее от неявной схемы, которая разрешается на верхнем слое каким-либо итерационным методом. Во-первых, схема ЛИ-М автоматически обеспечивает локальную консервативность с сохранением в качестве ячеек консервативности ячеек исходной сетки, используемых для записи разностной схемы. Во-вторых, при малых значениях $\tau < \tau_{exp}$, удовлетворяющих условию устойчивости явной схемы $\tau_{exp} \cdot \lambda_{max} < 2$, схема ЛИ-М автоматически переходит в явную схему, обеспечивающую на одном шаге минимум вычислительных затрат. Поэтому схему ЛИ-М целесообразно использовать в тех случаях, когда по разным причинам шаг интегрирования по времени τ может быть переменным, как большим, так и малым. При удвоении числа p схема ЛИ-М практически совпадает с неявной схемой.

Сложность решения параболических уравнений принято характеризовать «параболическим» числом Куранта $cou \approx \tau \cdot \lambda_{\max}$, здесь $\lambda_{\max}$ – максимальное собственное значение разностного эллиптического оператора. Для 3D задач на регулярных сетках вычислительные затраты схемы ЛИ-М суть $C\sqrt{\tau \cdot \lambda_{\max}} \cdot h^{-3} \approx C\sqrt{\tau \cdot h^{-2}} \cdot h^{-3}$, и константа в этой оценке зависит только от задачи. На ЛАД-сетках оценка вычислительных затрат схемы ЛИ-М имеет вид $C\sqrt{\tau \cdot \lambda_{\max}} \cdot J$, где J – число ячеек сетки, а значение $\lambda_{\max}$ определяется самой «плохой» ячейкой, то есть ячейкой, для которой отношение коэффициента диффузии к квадрату диаметра ячейки является максимальным. Как показывают расчеты, определенным препятствием для использования схемы ЛИ-М на ЛАД-сетках являются задачи, для которых использование неявной схемы имеет содержательное обоснование. К таким задачам относится, например, задача о прохождении температурной волны по нулевому фону: шаг по времени в такой задаче выбирается из условия правильного описания фронта тепловой волны, и тогда значение локального числа Куранта на фронте мало. В изотермической области значение числа Куранта может быть очень большим. И если в изотермической области из каких-либо посторонних соображений требуется измельчать сетку, то использование схемы ЛИ-М может привести к значительным вычислительным затратам. В данной работе этот аспект пока не изучается – главное внимание уделяется изложению основных элементов алгоритма, исследованию реальной точности построенной ЛАД-ЛИ-М в эволюционных задачах, где параболическое число Куранта относительно невелико. Дополнительно исследуется эффективность использования сеточной библиотеки [4], обеспечивающей перенос программы на графические

ускорители CUDA и не требующей от пользователя глубокого знания методов программирования для CUDA.

5. Библиотека *gridmath* в условиях локально-адаптивных сеток

При записи сеточных аппроксимаций дифференциальных операторов в компьютерных программах они превращаются в многострочные операторы с вложенными циклами и сложными выражениями. Существуют системы программирования, позволяющие записывать подобные выражения в упрощенном виде с использованием шаблонов (stencils), например, система Blitz++ (см. [5]). Однако эта система не решает другой важной задачи – облегчения переноса программ на новые параллельные архитектуры. В качестве примера укажем графические ускорители (например, CUDA) или новые процессоры Intel Xeon Phi. Существует ряд систем, облегчающий подобный перенос. Среди них можно отметить разработанную в ИПМ им. М.В. Келдыша систему DVM, системы OpenMP и OpenACC. С использованием этих и им подобных систем можно относительно легко преобразовать существующую последовательную программу, написанную на языках C/C++ или Fortran для работы на графических ускорителях или процессорах Intel Xeon Phi. Для этого в программе перед участками кода (например, циклами), которые должны исполняться параллельно, нужно вставить специальные инструкции компилятору, которые позволят ему сгенерировать параллельный код. В языках C и C++ это, как правило, делается с помощью так называемых прагм (#pragma dvm, #pragma omp или #pragma acc). В программах на языке Fortran используются псевдокомментарии (например, !$acc). Если скомпилировать такую программу обычным компилятором, который «не понимает» этих инструкций, то он их просто проигнорирует и сгенерирует последовательный код. Все это позволяет относительно легко перенести существующую программу на новые архитектуры. Если программа компилируется для гетерогенной архитектуры с отдельным ускорителем (например, с графическим ускорителем, таким как CUDA, или с процессором Intel Xeon Phi), то нужно иметь в виду, что у отдельного ускорителя своя оперативная память, и он может обрабатывать только данные, размещенные в ней. Поэтому компилятор, генерирующий параллельный код, как правило, хранит две копии данных – в памяти основной (host) системы и в памяти ускорителя, и при необходимости синхронизует их (копирует в ту или другую сторону). Синхронизацией данных (когда и что копировать) также можно управлять с помощью специальных инструкций (прагм или псевдокомментариев).

Операторная библиотека gridmath [4] стоит на стыке этих двух направлений. С одной стороны, она позволяет использовать сокращённую запись выражений с использованием операторов, аналогичных математическим, а с другой стороны, позволяет путем простой перекомпиляции генерировать код, исполняемый на различных архитектурах. В ней сеточная функция – это класс, для которого переопределены стандартные

математические операции. Кроме того, оператор (например, оператор Лапласа) превращен из абстрактного математического понятия в конкретный программный объект. Например, уравнение $u = \Delta(\mu\upsilon)$ с использованием библиотеки gridmath может быть записано так: u=laplas(mu*v).

В правой части оператора присваивания может стоять выражение любой сложности, результатом этого выражения будет сеточный вычислитель, который запомнит всю цепочку вычислений, не производя их. В момент присваивания вычислителя сеточной функции произойдет запуск вычислений, и для всех точек сеточной функции в левой части оператора присваивания будут произведены вычисления в соответствии с запомненной цепочкой.

Библиотека gridmath решает еще одну важную задачу. Как известно, на многих современных суперкомпьютерах (кластерах) стоят графические ускорители CUDA или ускорители Intel Xeon Phi. Например, в списке top500 самых мощных суперкомпьютеров за ноябрь 2015 года на первом месте стоит суперкомпьютер Tianhe-2 (Китай - NUDT), основанный на процессорах Intel Xeon Phi 31S1P, а на втором месте – Titan (США - Cray Inc.), основанный на графических ускорителях NVIDIA K20x. Многие российские суперкомпьютеры, например, самый мощный российский суперкомпьютер «Ломоносов» и вычислительный кластер К-100 в ИПМ им. М.В. Келдыша РАН, также содержат графические ускорители. Например, на К-100 на каждом из 64 узлов, помимо двух основных процессоров Intel Xeon X5670, стоят по три платы NVIDIA Fermi C2050 (по 448 GPU и 2,8 Gb памяти на каждом). Эффективное же использование этих весьма внушительных вычислительных мощностей затруднено, т.к. требует освоения большого объема новой и непривычной информации о методах программирования на них. Одной из целей при создании библиотеки ставилось облегчение использования подобных гетерогенных систем. В частности, все результаты, изложенные в данной статье, получены на кластере К-100.

Как показано выше, основной запуск вычислений в библиотеке производится в операторе присваивания вычислителя сеточной функции. То, как этот оператор реализован, полностью скрыто от пользователя и может производиться как последовательно, так и параллельно для точек сеточной функции в левой части. Порядок обхода точек – внутреннее дело библиотеки, и пользователь не должен закладываться на тот или иной порядок обхода точек. Все это дает библиотеке существенную гибкость. В частности, если компиляция осуществляется компилятором nvcc (CUDA), то, во-первых, все данные сеточных функций располагаются в памяти графического ускорителя, а не host-системы, и, во-вторых, оператор присваивания реализован путем вызова ядра (kernel) в графическом процессоре и осуществляется параллельно. В последовательном случае обход точек в случае трехмерной регулярной сетки осуществляется в трех вложенных циклах по трем осям. При работе на CUDA следует обратить внимание на то, что, в отличие от упомянутых во введении систем, нацеленных на распараллеливание циклов (DVM, OpenACC),

библиотека не хранит копию данных в памяти основного (host) процессора. Все данные сеточных функций хранятся только в памяти CUDA. Для хранения данных сеточных функций при компиляции «обычным» компилятором используется класс std::vector из стандартной библиотеки C++, а при компиляции с помощью nvcc используется класс thrust::device_vector из CUDA SDK.

Таким образом, при использовании библиотеки gridmath перенос программы на графические ускорители CUDA является относительно простой задачей и не требует от пользователя глубокого знания методов программирования для CUDA. В простейшем случае программу достаточно просто перекомпилировать. Немного сложнее получается при использовании MPI. В этом случае надо учитывать, что данные сеточных функций расположены в памяти CUDA, а MPI работает с памятью host-системы, поэтому нужно еще добавить копирование данных из CUDA в host-систему и обратно.

Первоначальная версия библиотеки написана для регулярных двух- и трехмерных прямоугольных сеток. С ее помощью успешно перенесен на CUDA многосеточный метод. Проведен сравнительный анализ эффективности многосеточного метода на различных архитектурах вычислительных систем. Результаты изложены в работе [4].

Затем встал вопрос о переносе библиотеки на нерегулярные сетки. Основная идея такого переноса состоит в том, чтобы некоторым образом упорядочить ячейки сетки, а затем, при присваивании выражения сеточной функции, делать их одномерный обход, последовательный (в одном цикле) или параллельный (для CUDA или OpenMP). Рассмотрим особенности, возникающие при переносе библиотеки на локально-адаптивные сетки, которые формируются на основе регулярных равномерных сеток. Физический шаг исходной регулярной сетки вдоль каждой из осей постоянный, но вдоль разных осей может быть разным. Сетка строится самой библиотекой или может быть загружена из файла.

При построении сетки исходная регулярная сетка рассматривается как нулевой уровень. До построения сетки формируются глобальные (общие для всех уровней) массивы физических координат по каждому из трех направлений. Первоначальное заполнение этих массивов простое – значение по i-му индексу равно шагу по данному направлению, помноженному на этот индекс. Эти массивы динамические – по мере построения сетки они расширяются. При делении ребра пополам может возникнуть новая точка с координатой, которой еще нет в массиве координат. В этом случае координата с заданным значением добавляется в конец массива координат. Это, в свою очередь, означает, что массивы координат не упорядочены по возрастанию (кроме начала этих массивов), значения координат могут идти в произвольном порядке.

Алгоритм построений каждого дополнительного уровня следующий. Изначально для строящегося уровня список ребер, граней и ячеек пустой. Рассматриваются все ребра сетки предыдущего уровня. У каждого ребра есть

две вершины. Если оказывается так, что одна из вершин ребра лежит внутри тела, а вторая – снаружи, то это ребро делится пополам, и получившиеся два коротких ребра добавляются к списку ребер строящегося уровня, а само ребро с предыдущего уровня помечается как поделенное. В описании каждого из объектов сетки (вершины, грани или ячейки) хранится индекс соответствующего объекта со следующего уровня. Если этот индекс равен -1 (недопустимое значение для индекса), то это означает, что данный объект сетки не поделен, иначе в нем хранится индекс первого из объектов, из которых состоит данный объект. После того как все ребра рассмотрены (и некоторые из них поделены), рассматриваются все грани сетки предыдущего уровня. Если оказывается, что для очередной грани хотя бы одно из четырех ребер было поделено на предыдущем шаге, то оставшиеся неподеленными ребра этой грани также делятся пополам, а сама грань делится на 4 маленьких равных грани, которые добавляются к списку граней строящегося уровня, а грань предыдущего уровня помечается как поделенная. Затем аналогичная процедура повторяется для ячеек. Если хотя бы одна из шести граней ячейки была поделена на предыдущем шаге, то оставшиеся неподеленными грани также делятся (вместе со своими ребрами), а сама ячейка делится на 8 маленьких равных ячеек, которые добавляются к списку ячеек строящегося уровня, а ячейка предыдущего уровня помечается как поделенная.

После построения очередного уровня в том случае, если номер построенного уровня больше 1, осуществляется сглаживание предыдущих уровней в обратном порядке (вначале сглаживается пред-предыдущий уровень, затем пред-пред-предыдущий и т.д.). Сглаживание делается для того, чтобы с двух сторон одной грани не было ячеек «через уровень», т.е. это должны быть или ячейки одного уровня, или соседних уровней. Алгоритм сглаживания следующий. Просматриваются все ячейки сглаживаемого уровня. Если какая-то ячейка не была поделена, а хотя бы одна из ее граней поделена дважды (поделена грань и хотя бы одна из ее дочерних граней), то ячейка делится (вместе со всеми своими гранями и ребрами).

Формат внутреннего представления информации обо всех элементах сетки (ребрах, гранях, ячейках) одинаковый и содержит следующие поля: int child; int boundary_type; int map[6]. Поле child задает индекс первого дочернего объекта следующего уровня или -1. Поле boundary_type содержит признак граничного элемента. 0 – внутренний элемент, 1 – граничный с граничным условием типа Дирихле, 2 – граничный с граничным условием типа Неймана. В поле map хранится информация об объектах того же уровня, составляющих данный элемент (вершинах для ребра, ребер для граней, граней для ячеек). Для ребер хранятся координаты (глобальные индексы) двух вершин. Для граней хранятся индексы 4 ребер, окружающих данную грань. Последние два элемента массива map в этом случае не используются. Для ячеек хранятся индексы 6 граней, окружающих данную ячейку.

Для сеточных функций, определенных на локально-адаптивных сетках, доступны операции сложения, вычитания, умножения (и прочие) со скалярами и вычисляемыми объектами, для работы с сеточными функциями созданы сеточные операторы, значительно упрощающиеся запись программ.

6. Вычислительные результаты

Для демонстрации работоспособности схемы на локально-адаптивных сетках приведем результаты некоторых предварительных численных экспериментов для модельных задач. В качестве основных сеток в расчетах используем равномерные сетки. Будем называть такие сетки регулярными. Для них на гладких решениях достигается второй порядок аппроксимации и точности по пространству. ЛАД-схема может работать на произвольных неравномерных сетках, но этот ресурс повышения реальной точности (сгущение регулярной сетки) здесь не изучается.

В зоне адаптивного измельчения сетки порядок локальной аппроксимации ЛАД-схемы по пространству снижается до первого. Вне зоны влияния сгущения порядок локальной аппроксимации остается вторым. Шаблон схемы в зоне сгущения уже не является 7-точечным. При вычислении потоков на интерфейсах ячеек, принадлежащих разным сеточным уровням (см. рис. 4), используется достаточно грубая аппроксимация, но она обеспечивает нужную знакоопределенность (положительность) коэффициентов разностной схемы (см. раздел 3).

Для оценки реальной точности проведем расчеты на последовательности регулярных сеток с числом шагов N по каждому координатному направлению. Для каждой регулярной сетки вводится L дополнительных сеточных уровней для обеспечения сгущения сетки в окрестности особенности, задаваемой в приведенных примерах некоторой поверхностью. Точность схемы будем контролировать в заданный момент времени либо по норме ошибки $\delta = \| u - u_h \|_2 / \| u_h \|_2$ приближенного решения u_h, если известно точное решение $u(t,x,y,z)$, либо по норме невязки разностного уравнения

$$\| div\ (\kappa\ grad\ u) + a \cdot u - f - u_t \|, \qquad (7)$$

где производная по времени вычисляется как разделенная разность сеточных значений на двух последовательных временных слоях, а дискретный оператор по пространству вычисляется на сеточной функции, равной полусумме приближенных решений на этих временных слоях

Задача 1. Решается нестационарное уравнение теплопроводности в кубе $[0;\ 1]^3$ на промежутке времени $t \in [0,1]$. Внутрь исходного куба помещается цилиндр радиуса 0.27 с вертикальной осью $x = 0.5, y = 0.5$. Внутри и вне цилиндра коэффициент диффузии κ полагается одинаковым и равным $\kappa = 1$. Параметр a полагается постоянным и равным нулю.

Возьмем точное решение в виде

$$u(t;x,y,z)=e^{-\pi^2 t}\ \cos\ \pi x\ \ \cos\ \pi y\ \ \cos\ \pi z\ ,$$

тогда правая часть $f(t,x,y,z)\equiv 0$. Начальные данные возьмем из точного решения.

В этой задаче нет априорной необходимости в сеточной адаптации. Однако мы такую адаптацию проведем, предположив наличие цилиндра внутри куба, построив второй и третий сеточный уровень с адаптацией к поверхности цилиндра.

Сравним результаты расчетов на последовательности регулярных и адаптивных сеток с одним и двумя сеточными уровнями. На рис. 5 приведены в логарифмическом масштабе графики относительной погрешности ошибки δ на конечный момент времени $t=0.2$ в зависимости от шага основной регулярной сетки. Шаг по времени выбирается из условия сохранения параболического числа Куранта неизменным при переходе на более подробную сетку. Для регулярных сеток с числом шагов $N=8,\ 16,\ 32,\ 64$ по каждому направлению шаг по времени задается как $\tau=0.05/(\lg_2 N-2)$. Такой выбор объясняется желанием хорошо проинтегрировать уравнение по времени; при таком выборе степень многочлена Чебышева, используемого в конструкции (5) схемы ЛИ-М, в данном конкретном случае равна 5.

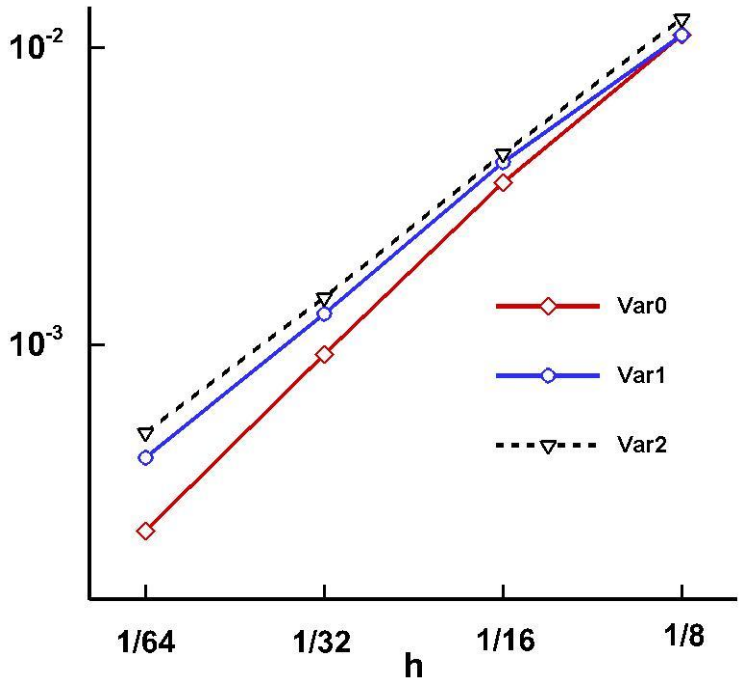


Рис. 5. Сходимость сеточных приближений на регулярных сетках (Var0) и ЛАД-сетках (Var1, Var2) с одним и двумя сеточным уровнями соответственно

Из графиков следует, что для регулярного случая Var0 ошибка с уменьшением шага убывает со скоростью $O(h^2)$ и локальное измельчение сетки в окрестности боковой поверхности цилиндра уменьшает скорость сходимости, но очень незначительно.

Задача 2. В условиях задачи 1 возьмем разрывные коэффициент диффузии и начальные данные: внутри и вне цилиндра коэффициент диффузии κ полагается равным $\kappa = 0.01$ и $\kappa = 1$ соответственно, начальная функция равна 1 внутри и 0 – вне цилиндра. Отрезок интегрирования $[0;\ 0.002]$.

Профиль приближенного решения, полученный на очень подробной сетке, приведен на рис. 6 (маркеры узлов показаны с пропусками).

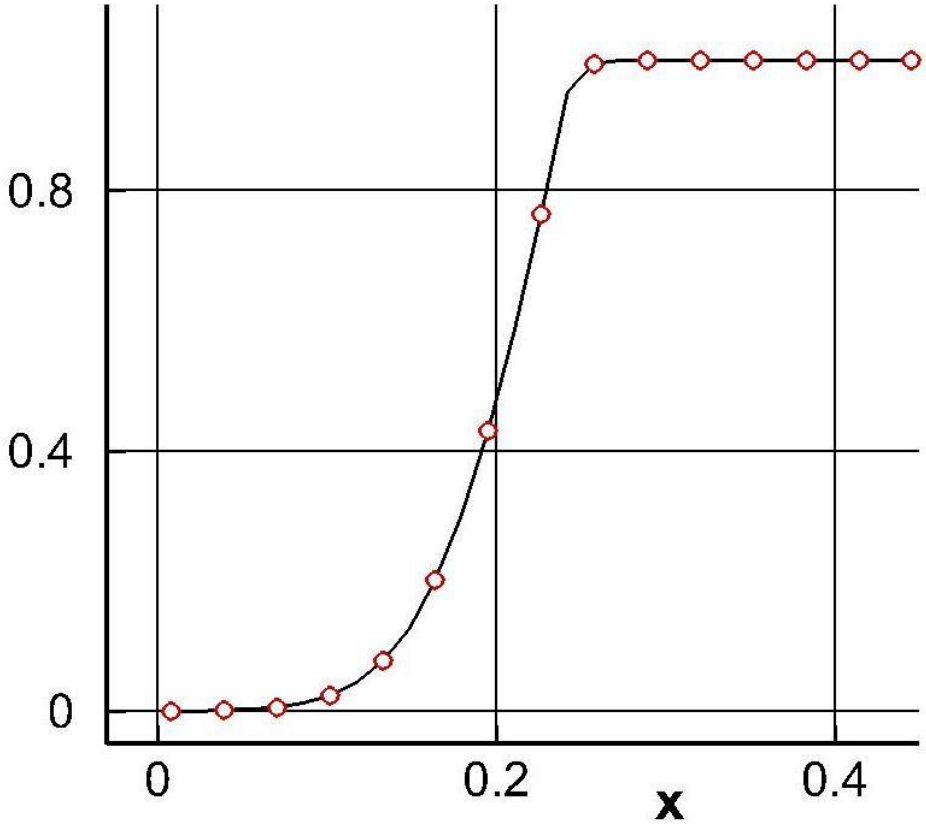


Рис. 6. Профиль приближенного решения в центральном сечении на регулярной сетке

Для регулярных сеток с числом шагов $N = 16,\ 32,\ 64$ по каждому направлению шаг по времени задается как $\tau = 0.001/(\lg_2 N - 2)$.

На рис. 7 приведены в логарифмическом масштабе графики нормы невязки (7) на конечный момент времени $t = 0.002$ в зависимости от шага основной регулярной сетки.

Анализируя рисунки 6 и 7, видим, что для гладкой задачи 1 ЛАД-сетки не слишком ухудшают порядок сходимости, но для негладкой задачи 2 измельчение сетки в области резкого изменения решений приводит к заметному уменьшению нормы невязки.

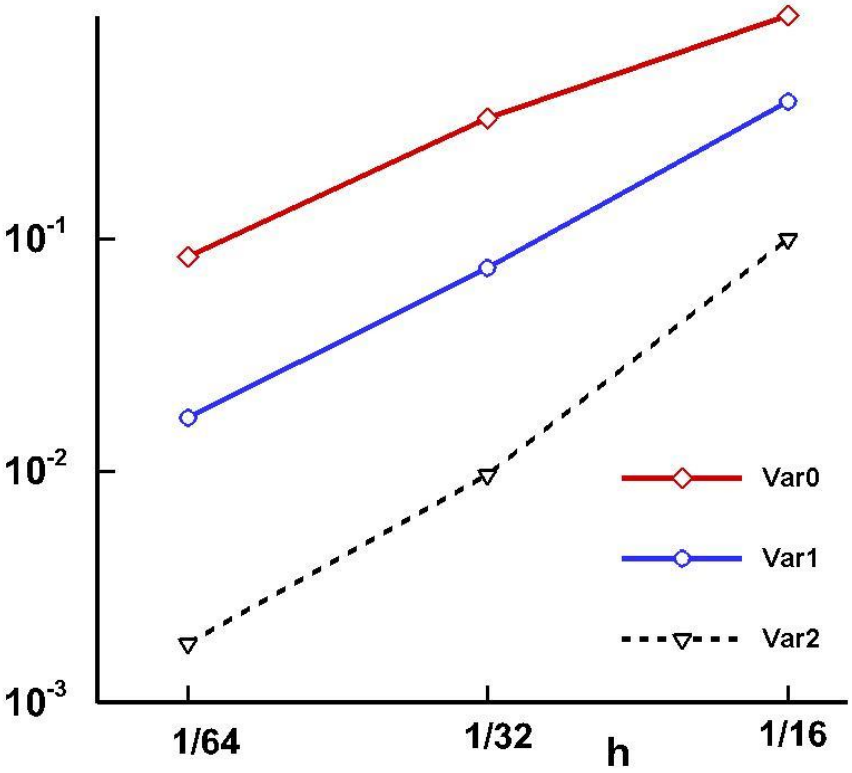


Рис. 7. Сходимость нормы невязки сеточных приближений на регулярных сетках (Var0) и ЛАД-сетках (Var1, Var2) с одним и двумя сеточным уровнями соответственно. Задача 2

Мы пока не делаем окончательных выводов, требуется дальнейшее изучение ЛАД-схемы на более широком круге задач с оценкой вычислительных затрат (времени счета, памяти). Приведем для примера рост числа узлов сетки при использовании локальной адаптации в задаче с цилиндром. Для основных сеток $32\times32\times32$ и $64\times64\times64$ (с полным числом узлов 32768 и 262144) включение локальной адаптации с одним дополнительным уровнем увеличивает число узлов на 19456 и 79872 соответственно.

Задача 3. В условиях задачи 1 сравним результаты расчетов на сетках с разным числом сеточных уровней в обычной и CUDA-реализациях (на графическом сопроцессоре-ускорителе) схемы с помощью библиотеки сеточных операторов gridmath [4].

Таблица 1

Зависимость времени счета от уровней адаптации

Основная сетка	gridmath	gridmath +CUDA
3 уровня адаптации		
$32\times32\times32$	1.92	1.21
$64\times64\times64$	39.38	21.69
4 уровня адаптации		
$16\times16\times16$	1.64	1.05
$32\times32\times32$	28.97	15.90
$64\times64\times64$	490.74	258.97
5 уровней адаптации		
$64\times64\times64$	28.60	15.59
$32\times32\times32$	454.34	237.04

Можно видеть (см. табл. 1), что CUDA-реализация (получаемая простой перекомпиляцией кода) дает ускорение примерно в два раза. Не очень большое ускорение на CUDA объясняется спецификой задачи. CUDA дает существенное ускорение, если удается сделать так, чтобы потоки с последовательными номерами обращались к соседним ячейкам памяти, что в случае локально-адаптивных сеток не выполняется. Основное время уходит на построение массива граничных ячеек, которые собираются по всей области.

Задача 4. Рассмотрим куб $[0;\,1]^3$, внутрь которого помещен цилиндр радиуса $R=0.05$. Внутри цилиндра коэффициент диффузии и начальная функция равны 10^{-4} и 1, вне цилиндра – 1 и 0 соответственно. Задача решается при краевом условии Неймана на отрезке $t\in[0;\,0.002]$.

Рассмотрим в качестве основной сетку $32\times32\times32$. Ниже этот вариант без адаптации обозначен Var0. Обозначение Var1 означает расчет на сетке $32\times32\times32$ с одним дополнительным уровнем адаптации, Var2 соответствует сетке с двумя уровнями адаптации, Var3 – с тремя уровнями. Все варианты сосчитаны с постоянным шагом по времени, одним и тем же для всех сеток, определяемым соотношением $\tau=30/\lambda_{\max}$ со значением $\lambda_{\max}$ для основной сетки. В рассматриваемом случае использование локально-адаптивных сеток позволяет более точно описать приближенное решение и потоки на границе тонкого стержня. Для данной задачи мы ограничимся информацией об основных вычислительных характеристиках задачи и их ростом при увеличении числа уровней адаптации (см. табл. 2).

Таблица 2

Основные характеристики расчета задачи 4

	Var0	Var1	Var2	Var3
N	32768	35456	50688	10265
nnz	223232	244280	361416	763208
iterLIM	45	80	160	320
$\lambda_{\max}$	12288	49152	196608	786432
T/T_0	1	2	6	25

Здесь N – число расчетных точек в задаче (размер сеточной функции), *nnz* – число ненулевых элементов в «матрице» разностного оператора. Мы не формируем матрицу явно, и операция применения разностного оператора к сеточной функции осуществляется процедурно; трудоемкость этой операции (массовой в рассматриваемом чебышевском методе ЛИ-М) и определяется параметром *nnz*. Остальные параметры имеют следующие значения: *iterLIM* – полное число шагов схемы ЛИ-М (каждый из которых по трудоемкости эквивалентен шагу явной схемы), $\lambda_{\max}$ – максимальное собственное значение разностного оператора, определяющее вместе с шагом τ число шагов ЛИ-М на одном временном шаге, T/T_0 – отношение времени счета, затраченного на расчет рассматриваемого варианта, ко времени счета варианта Var0.

Если расчет проводить на сетке $256\times256\times256$, что соответствует варианту без адаптации, но с минимальным пространственным шагом, взятым из варианта Var3, то полный расчет по явной схеме займет около 5 часов. Вариант Var3 считается 6 секунд.

7. О параллельной реализации

Многопоточная параллельная реализация схемы ЛИ-М на локально-адаптивных сетках для графических сопроцессоров обсуждена в разделе 5 данной работы. Рассмотрим возможности параллельной реализации схемы ЛИ-М на ЛАД-сетках на многопроцессорной системе с помощью MPI-технологии. На регулярных сетках с простейшей геометрической декомпозицией сетки на заданное число блоков с виртуальными топологиями «линейка процессоров», «двумерная и трехмерная решетки процессоров» схема ЛИ-М легко распараллеливается и обеспечивает почти идеальное масштабирование. Для ЛАД-сеток возникает препятствие к идеальному масштабированию в виде возможной неравномерной загрузки процессоров. Для оптимизации загрузки процессоров для неструктурированных сеток служит METIS – пакет программного обеспечения для разбиения неструктурных графов, разбиения

сеток [6], см. также сайт http://glaros.dtc.umn.edu/. Этот пакет для ЛАД-сеток нами опробован. В дальнейшем возможно его применение для обеспечения масштабирования схемы ЛИ-М на ЛАД-сетках на многопроцессорной системе.

8. Заключение

В данной работе приведены основные элементы алгоритмической и программной реализации схемы ЛИ-М, примененной к численному интегрированию параболических уравнений с разрывными коэффициентами на локально-адаптивных сетках. Исследование реальной точности построенной схемы в эволюционных задачах показывает, что с помощью ЛАД-сеток можно повысить точность схемы в окрестности поверхностей разрыва коэффициентов исходного дифференциального уравнения без заметного ухудшения точности. Приведены результаты использования для решения параболических уравнений на локально-адаптивных сетках социализированной библиотеки сеточных операторов, обеспечивающей перенос программы на графические ускорители CUDA и не требующей глубокого знания методов программирования для CUDA. Для повышения эффективности расчетов эволюционных задач на локально-адаптивных сетках требуются дальнейшие исследования по представлению особенностей исходной задачи, выработке критериев степени измельчения сетки и другие.

Библиографический список

1. Жуков В.Т. О явных методах численного интегрирования для параболических уравнений // Математическое моделирование, 2010, т. 22, № 10, С. 127-158.
2. Жуков В.Т., Новикова Н.Д., Феодоритова О.Б. О решении эволюционных уравнений многосеточным и явно-итерационным методами // Ж. вычисл. матем. и матем. физ., 2015, т. 55, № 8. – С. 1305–1319.
3. Жуков В.Т., Краснов М.М., Новикова Н.Д., Феодоритова О.Б. Параллельный многосеточный метод: сравнение эффективности на современных вычислительных архитектурах // Программирование, 2015, т. 41, №. 1. С. 21-31.
4. Краснов М.М. Операторная библиотека для решения трёхмерных сеточных задач математической физики с использованием графических плат с архитектурой CUDA // Математическое моделирование, 2015, т. 27, № 3. С. 109-120.
5. The Blitz++ library. URL: http://blitz.sourceforge.net/
6. Karypis G. and Kumar V. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs // SIAM Journal on Scientific Computing, 1999, vol. 20, №. 1, pp. 359—392.

Оглавление