Ещё один метод распараллеливания прогонки с использованием ассоциативности операций*

А.В. Фролов

ИВМ РАН

Автором предложен новый метод распараллеливания прогонки на основе использования свойства ассоциативности операций перемножения матриц. В отличие от давно известной версии параллельного алгоритма Стоуна, основанного на приёме сдваивания, новый метод имеет те же характеристики устойчивости, что и последовательная прогонка. Предложена также и блочная модификация метода.

1. Введение

В данной статье автор предлагает новые способы решения системы линейных алгебраических уравнений[1] с трёхдиагональными матрицами, на той же идейной основе, что и известный метод сдваивания Стоуна, – с использованием ассоциативности матричного умножения. Однако использование последовательных фрагментов позволяет автору применить нормировку, что делает новый метод устойчивым.

2. Прогонка как метод решения СЛАУ специального вида.

Прогонка – последовательный алгоритм решения трёхдиагональной СЛАУ – является частным случаем общего метода исключения неизвестных, однако получила специальное название из-за распространённости задач такого типа в прикладных исследованиях. В виде отдельного алгоритма «открыта» несколькими исследователями независимо друг от друга (И.М. Гельфандом и О.В. Локуциевским в СССР, Л.Х. Томасом в США[2]). С этими названиями описана в большом количестве учебников по численным методам (например, в [5]), причём в разных вариантах.

Классическая схема прогонки практически не содержит возможностей для распараллеливания, максимум – по двум потокам вычислений. Однако решение трёхдиагональных СЛАУ довольно востребовано в различных вычислительных задачах при моделировании, поэтому задача «распараллеливания прогонки», т.е. параллельного решения исходной задачи, давно привлекает внимание многих исследователей.

В подавляющем большинстве учебников (например, в [5]) отмечается, что задача решения СЛАУ $Ax=b$ с трёхдиагональной матрицей A при решении с помощью прогонки эквивалентна последовательности двух задач. Это разложение матрицы в произведение, например, двухдиагональных матриц, а также решение СЛАУ с этими матрицами. Такая разбивка впоследствии даёт возможность использовать уже найденное разложение для более быстрого решения СЛАУ и с другими правыми частями. В дальнейшем мы увидим это как на примере рассмотрения алгоритма Стоуна, так и при конструировании собственного алгоритма.

2.1 Известные параллельные методы решения трёхдиагональных СЛАУ

Как только у исследователей появились в распоряжении вычислительные устройства с возможностью параллельной работы, так сразу было предложено несколько параллельных методов решения трёхдиагональных СЛАУ ([7–9]). Сравнение с позиций середины 70-х гг. XX

* Исследование выполнено при частичной финансовой поддержке гранта Российского научного фонда (проект N14-11-00190).

[1] далее для систем линейных алгебраических уравнений будем использовать сокращение СЛАУ.

[2] поэтому в англоязычной литературе, кроме названия «Tridiagonal matrix algorithm», используется и название «Thomas algorithm»

века трёх таких методов – циклической редукции, Бунемана и рекурсивного сдваивания – можно прочитать в работе автора последнего Х.Стоуна 1975г. [10]. Несмотря на общую привязку этого сравнения к архитектурам того времени, эти сравнения актуальны и теперь. Согласно им, алгоритм рекурсивного сдваивания (Стоуна) имеет лучшие характеристики по количеству требуемых вычислительных затрат. Однако читателям, должно быть, известно, что на практике применяют не его, а метод циклической редукции. Это связано с тем, что один из этапов метода рекурсивного сдваивания Стоуна (разложение матрицы в произведение двухдиагональных матриц) имеет гораздо худшую устойчивость.

Все перечисленные алгоритмы имеют логарифмическую (относительно размера задачи) длину критического пути, и придуманы в то время, когда специалисты по распараллеливанию ещё предполагали, что в будущем проблема эффективных пересылок между устройствами будет как-то решена. Эти методы также по-разному загружают доступные вычислительные устройства, что создаёт при их реализации дополнительные проблемы и уменьшает реальную эффективность.

Вышеизложенные проблемы подвигли автора статьи к работе по нахождению альтернативных способов распараллеливания прогонки. Автор преследовал две главных цели: разработать алгоритм, достаточно просто отображаемый на современные архитектуры, и при этом лишённый недостатков описанных выше методов, а также обеспечить его устойчивость. Для этого прежде всего им был изучен самый быстрый из перечисленных методов – метод рекурсивного сдваивания Стоуна, основанный на использовании ассоциативности операции перемножения матриц.

2.2 Основания метода Стоуна и возможные пути его изменения[1]

Итак, введём следующие обозначения. Пусть задача состоит в решении СЛАУ

$$Ax = b \tag{1}$$

где

$$A = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & & & & \\ \gamma_1 & \alpha_2 & \beta_2 & \cdots & & 0 & \\ 0 & \gamma_2 & \alpha_3 & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & \alpha_{n-2} & \beta_{n-2} & 0 \\ & 0 & & \cdots & \gamma_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & 0 & \gamma_{n-1} & \alpha_n \end{pmatrix} \tag{2}$$

– трёхдиагональная матрица, для которой выполняются условия устойчивости решения методом прогонки [5],

$$b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \\ b_n \end{pmatrix} \tag{3}$$

– вектор правой части. Если теперь матрицу A разложить в произведение двух треугольных

$$A = LU \tag{4}$$

где

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_1 & 1 & & \\ \vdots & & \ddots & \vdots \\ & 0 & \cdots & 1 \quad 0 \\ & & & l_{n-1} \quad 1 \end{pmatrix} \tag{5}$$

[1] В изложении метода Стоуна автор использует собственные обозначения, а не обозначения из статей самого Стоуна [9, 10]

$$U = \begin{pmatrix} u_1 & f_1 & \cdots & 0 \\ 0 & u_2 & & \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & u_{n-1} & f_{n-1} \\ & & 0 & u_n \end{pmatrix} \quad (6)$$

то (1) будет заменена на

$$LUx = b \quad (7)$$

После этого сначала нужно решить систему

$$Ly = b \quad (8)$$

и после неё

$$Ux = y \quad (9)$$

Занявшись сначала более простым распараллеливанием решения двухдиагональных СЛАУ, видим, что для (8) получаются формулы

$$y_1 = b_1, \qquad y_k = b_k - l_{k-1}y_{k-1}, k = 2, \dots, n \quad (10)$$

после чего введением векторов

$$z_i = \begin{pmatrix} y_i \\ 1 \end{pmatrix} \quad (11)$$

достигается выражение

$$z_k = C_k z_{k-1}, k = 2, \dots, n \quad (12)$$

где

$$C_i = \begin{pmatrix} -l_{i-1} & b_i \\ 0 & 1 \end{pmatrix} \quad (13)$$

После подстановки в (12) этой же формулы с меньшими значениями индекса получаем

$$z_k = C_k C_{k-1} \dots C_2 z_1 \quad (14)$$

после чего нужно вычислить все такие частные произведения, что делается, например, методом сдваивания за $\log_2(n-1)$ ярусов.

Аналогично у Стоуна решается и СЛАУ с матрицей *U*. При этом выполнение сдваивания на данных этапах решения исходной задачи не вызывает роста вычислительной погрешности. Остаётся рассмотреть, как у Стоуна распараллеливается процесс разложения.

Если применить формулу Бине-Коши (см. например [1,2]) к вычислению ведущего главного минора[1] Δ_k матрицы *A* порядка *k*, то получим, что, учитывая (4), (5), (6),

$$\Delta_k = \prod_{i=1}^{k} u_i \quad (15)$$

и, вводя $\Delta_0 = 1$, имеем

$$u_k = \frac{\Delta_k}{\Delta_{k-1}} \quad (16)$$

а из формул перемножения матриц -

$$l_k = \frac{\gamma_k}{u_k} \quad (17)$$

и

$$f_k = \beta_k \quad (18)$$

Остаётся понять, как параллельно вычислить все ведущие главные миноры матрицы *A*. Из учебников и справочников (например, [1,2]) известно, что у трёхдиагональных матриц для ведущих главных миноров выполняются рекуррентные соотношения

$$\Delta_k = \alpha_k \Delta_{k-1} - \beta_{k-1}\gamma_{k-1}\Delta_{k-2} \quad (19)$$

что позволяет, введя вектора

$$r_k = \begin{pmatrix} \Delta_k \\ \Delta_{k-1} \end{pmatrix} \quad (20)$$

получить аналогичную (12) формулу

$$r_k = F_k r_{k-1}, k = 2, \dots, n \quad (21)$$

где

[1] У Стоуна – q_k, без отметки, что это ведущий главный минор. Доказательство формулы (16) он ведёт непосредственно по формулам вычисления элементов матриц, подобно тому, как это сделано здесь, в формулах (38) – (40), но не в блочном варианте.

$$F_k = \begin{pmatrix} \alpha_k & -\beta_{k-1}\gamma_{k-1} \\ 1 & 0 \end{pmatrix} \quad (22)$$

и

$$r_k = F_k F_{k-1} \dots F_2 r_1 \quad (23)$$

что схоже с (14) и даёт возможность применить тот же самый приём сдваивания, что и при решении двухдиагональных СЛАУ. Однако у этого этапа есть два существенных отличия. Первое – у произведения $C_k C_{k-1} \dots C_2$ нижняя строка остаётся той же, что и у всех сомножителей – $(0 \quad 1)$, а у произведения $F_k F_{k-1} \dots F_2$ нижняя строка та же, что верхняя у $F_{k-1} F_{k-2} \dots F_2$. Второе отличие, пожалуй, самое существенное. Дело в том, что если распараллеленные по Стоуну решатели двухдиагональных СЛАУ имеют тот же диапазон устойчивости, что и последовательные, то распараллеленный по Стоуну этап разложения матрицы имеет гораздо худшую устойчивость, чем *LU*-разложение по компактной схеме метода Гаусса. Дело в том, что, по [1,2], оценки эквивалентного возмущения у нераспараллеленного разложения определяются в основном ростом (или отсутствием роста) элементов полученного разложения. А вот при вычислении элементов матриц $F_k F_{k-1} \dots F_2$ рост промежуточных результатов будет гораздо больше. Это и понятно, если вспомнить, что ведущие главные миноры матрицы равны произведениям диагональных элементов получаемых матриц.

Именно поэтому алгоритм Стоуна, несмотря на то, что его часто рассказывают студентам на курсах по параллельным вычислениям, на практике не применяется. Вместе с тем нельзя полностью ставить на нём крест. Нередки случаи, когда после один раз выполненного разложения СЛАУ с уже разложенной матрицей решаются снова и снова, с новой правой частью. Тогда вполне разумным представляется, потратив большое время на это разложение обычным способом, затем использовать элементы метода Стоуна при решении новых потоков СЛАУ.

Кроме того, и эти «куски» метода можно модифицировать так, чтобы они лучше отображались на архитектуру вычислительных комплексов. Как нетрудно видеть по **Рис. 1**, алгоритм сдваивания имеет граф с довольно сложной структурой передач.

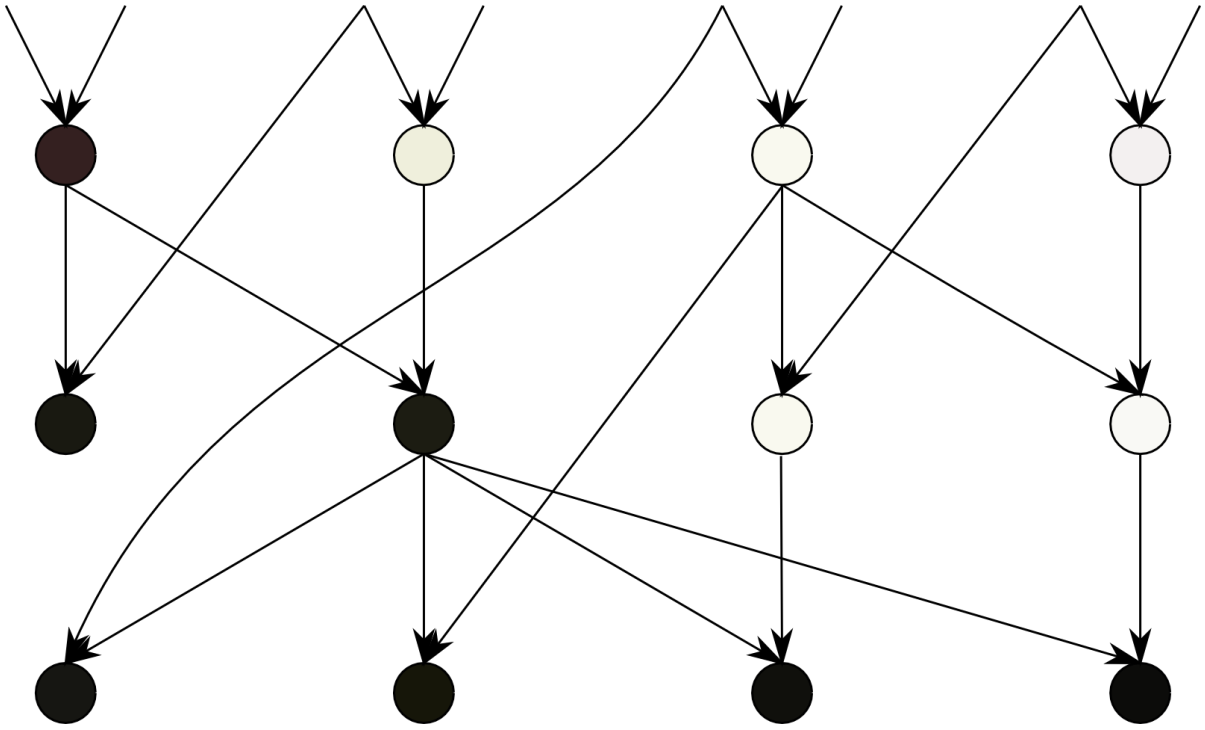

Рис. 1. Алгоритм сдваивания для вычисления всех частичных "произведений" для ассоциативных операций при *n=8*, чёрным обозначены операции, результаты которых нужны на выходе алгоритма

Другим важным его недостатком является большой *коэффициент избыточности*. Для метода сдваивания вычисление всех частных произведений будет стоить вместо *n-1 операций умножения матриц* $\frac{n}{2}\log_2 n$ таких же операций, так что коэффициент избыточности равен $\frac{\log_2 n}{2}$. Дополнительную добавку несёт замена операции типа *a+bc* на, хоть и урезанную благодаря

специфике матриц, но всё же занимающую больше ресурсов операцию перемножения матриц. Это накладывает жёсткие требования на количество устройств, нужных для того, чтобы хотя бы не проиграть последовательному алгоритму.

Пришедшая автору в голову идея для замены фрагментов метода Стоуна, связанных с решениями двухдиагональных СЛАУ – замена сдваивания на последовательно-параллельный метод организации вычислений. Как видно на **Рис. 12**, структура передач при его использовании существенно упрощается. Кроме этого, *коэффициент избыточности* существенно уменьшается по сравнению со сдваиванием и становится равным *2*. Это означает, что данную схему можно применять для ускорения соответствующих частей прогонки уже при небольшом количестве доступных устройств.

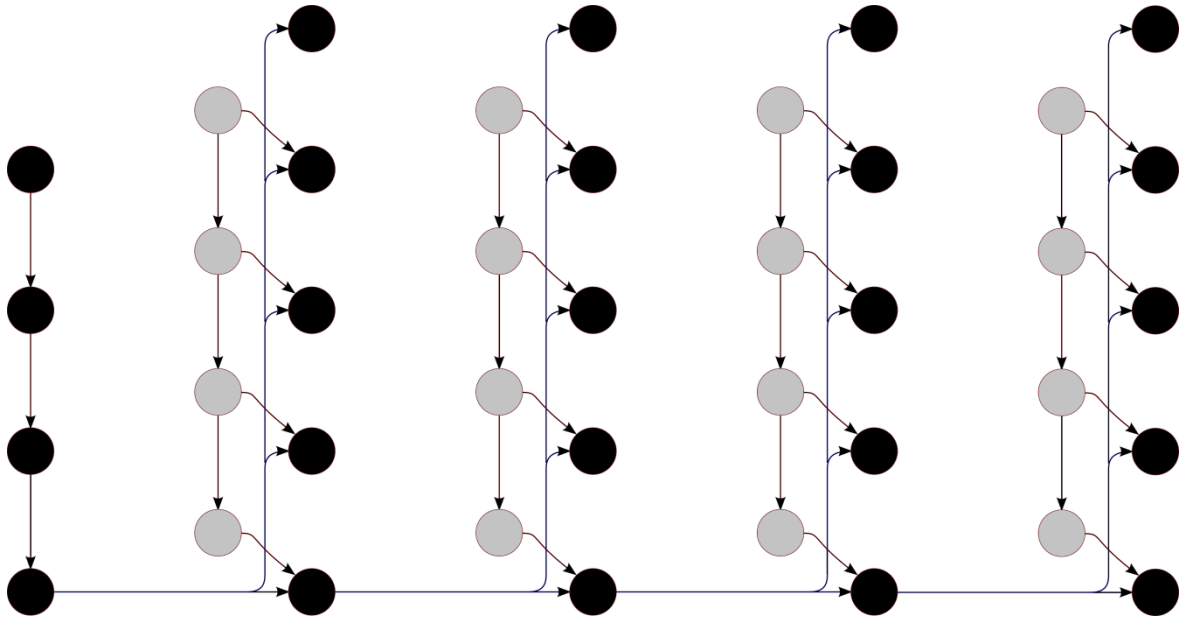

Рис. 2. Алгоритм последовательно-параллельного метода для вычисления всех частичных "произведений" для ассоциативных операций при *n=25*, чёрным обозначены операции, результаты которых нужны на выходе алгоритма

Есть и ещё важный момент, который следует подчеркнуть. Сам алгоритм сдваивания Стоуна нельзя применять по аналогии, если у нас СЛАУ не с трёхдиагональной, а с блочно-трёхдиагональной матрицей. Однако его фрагменты, связанные с решением двухдиагональных СЛАУ, вполне годятся для ускорения решения СЛАУ с блочно-двухдиагональными матрицами. Применение в последнем случае последовательно-параллельной схемы облегчит адаптацию этих частей к решению многих задач.

3. Новый метод распараллеливания главной части прогонки - разложения матрицы.

Алгоритм сдваивания Стоуна неустойчив из-за возможного роста результатов при выполнении промежуточных вычислений, связанных с ведущими главными минорами матрицы. Однако нам не нужны сами эти миноры, нужны лишь отношения соседних. Посмотрим снова на формулы, связывающие их, – нельзя ли как-то избежать роста результатов промежуточных вычислений?

3.1 Идея введения нормировки в промежуточных вычислениях

Посмотрим на формулы (21) – (23) снова. Выполним в (21) подстановку не до конца, как в (23), а до некоторого значения $i<k$:

$$r_k = F_k F_{k-1} \dots F_{i+1} r_i \qquad (24)$$

Обозначим произведения $F_k F_{k-1} \dots F_{i+1}$:

$$F_k F_{k-1} \dots F_{i+1} = G_{ki} \quad (25)$$

Обозначим элементы верхней строки этой матрицы как s_{ki} и t_{ki}. Тогда из вида матрицы F_k видно, что

$$G_{ki} = \begin{pmatrix} s_{ki} & t_{ki} \\ s_{k-1\,i} & t_{k-1\,i} \end{pmatrix} \quad (26)$$

($s_{ii} = 1, t_{ii} = 0$). Поэтому

$$\Delta_k = s_{ki}\Delta_i + t_{ki}\Delta_{i-1} \quad (27)$$

$$\Delta_{k-1} = s_{k-1\,i}\Delta_i + t_{k-1\,i}\Delta_{i-1} \quad (27')$$

Поделив (27) на (27'), получаем из (16)

$$u_k = \frac{\Delta_k}{\Delta_{k-1}} = \frac{s_{ki}\Delta_i + t_{ki}\Delta_{i-1}}{s_{k-1\,i}\Delta_i + t_{k-1\,i}\Delta_{i-1}} = \frac{s_{ki}u_i + t_{ki}}{s_{k-1\,i}u_i + t_{k-1\,i}} \quad (28)^1$$

и таким образом мы уже избавились от одного из источников больших промежуточных результатов – самих миноров. Остаётся избавиться от вычислений с большими элементами промежуточных матриц s_{ki} и t_{ki}. Если посмотреть на (28) внимательно, то окажется, что вместо 2мерных строк (s_{ki} t_{ki}) и ($s_{k-1\,i}$ $t_{k-1\,i}$) в ней могут фигурировать эти же строки, домноженные на любой выбранный нами нормировочный коэффициент. Вкупе с идеей последовательно-параллельного метода это даёт следующую схему.

3.2 Схема нового метода разложения трёхдиагональной матрицы

Весь диапазон натуральных чисел *от 1 до n* разбивается на *q* промежутков – *от 1 до k_1, от k_1+1 до k_2, ..., от k_{q-1}+1 до k_q=n.* На первом промежутке все значения u_i вычисляются последовательно, по стандартным формулам разложения на 2 диагональные матрицы. На остальных промежутках выполняется следующая процедура.

Пусть рассматривается *j+1*-й промежуток. Тогда уже известны значения

$$s_{q_j q_j} = 1, \qquad t_{q_j q_j} = 0, \quad (29)$$

$$S_{q_j+1\,q_j} = \alpha_{q_j+1}, \qquad T_{q_j+1\,q_j} = -\beta_{q_j}\gamma_{q_j} \quad (29')$$

Для всех значений *i от* $q_j + 2$ *до* q_{j+1} теперь вычисляем

$$S_{i\,q_j} = \alpha_i - \beta_{i-1}\gamma_{i-1}s_{i-2\,q_j} / S_{i-1\,q_j} \quad (30)$$

$$T_{i\,q_j} = (\alpha_i T_{i-1\,q_j} - \beta_{i-1}\gamma_{i-1}t_{i-2\,q_j}) / S_{i-1\,q_j} \quad (30')$$

$$s_{i-1\,q_j} = 1 \quad (31)^2$$

$$t_{i-1\,q_j} = T_{i-1\,q_j} / S_{i-1\,q_j} \quad (31')$$

Формулы (30) – (31') как раз включают в себя нормировку – их выполнение приводит к единице один из коэффициентов в знаменателе дроби в (28). После того, как выполнены вычисления на всех промежутках, на них (кроме первого) ещё неизвестны значения u_i. Их мы для каждого *j+1-го* промежутка вычисляем параллельно по формулам

$$u_i = \frac{S_{iq_j}u_{q_j} + T_{iq_j}}{u_{q_j} + t_{i-1\,q_j}} = S_{iq_j} + \frac{T_{iq_j} - S_{iq_j}\,t_{i-1\,q_j}}{u_{q_j} + t_{i-1\,q_j}} \quad (32)$$

Естественно, что каждое значение $u_{q_{j+1}}$ можно вычислить только после вычисления u_{q_j}. В результате, если разбиение на интервалы будет проведено равномерно, структура алгоритма разложения будет иметь почти тот же вид, что и на **Рис. 12,** с тем, однако, отличием, что «тяжести» операций будут разными. Это отражено на **Рис. 13**.

Разнородность операций делает не столь тривиальной задачу разбиения выполняемых операций по разным устройствам; тем не менее, эта задача существенно проще, чем организация пересылок в методах, использующих сдваивание. Оценим количество операций на каждом этапе. Вычисления $-\beta_{k-1}\gamma_{k-1}$ будем считать выполненными заранее.

[1] последний переход получается делением числителя и знаменателя на Δ_{i-1}

[2] деление $S_{i-1\,q_j}$ на себя не показано

1. Вычисление разложения на первом из промежутков известно, если длина промежутка равна *m*, то это по *m-1* операций деления, умножения и сложения/вычитания. Все эти операции следуют друг за другом последовательно.

2. Вычисления коэффициентов на остальных промежутках, если их длина равна *m*, займёт m-2 параллельных операций:

одна из них – умножение, деление, потом вычитание/сложение,

вторая – два параллельных умножения, вычитание/сложение, потом деление,

третья – деление.

Видно, что вычисление коэффициентов в общем сложнее. В него можно добавить и вычисление числителя последней дроби из (32), он не зависит от значений u_{q_j}. Тогда для вычислений на 2м этапе самым длинным является вычисление по «низу картинки» – это операция типа сложение, потом деление и потом снова сложение, повторяемая *q-1* раз. Параллельно этой операции нужно выполнить ещё *m-2* таких же, а также ещё одну, состоящую только из деления и сложения.

Приведённые формулы опираются на формулы (19) для миноров и поэтому, как и сам метод сдваивания Стоуна, казалось бы, неприменимы для блочно-трёхдиагональных матриц.

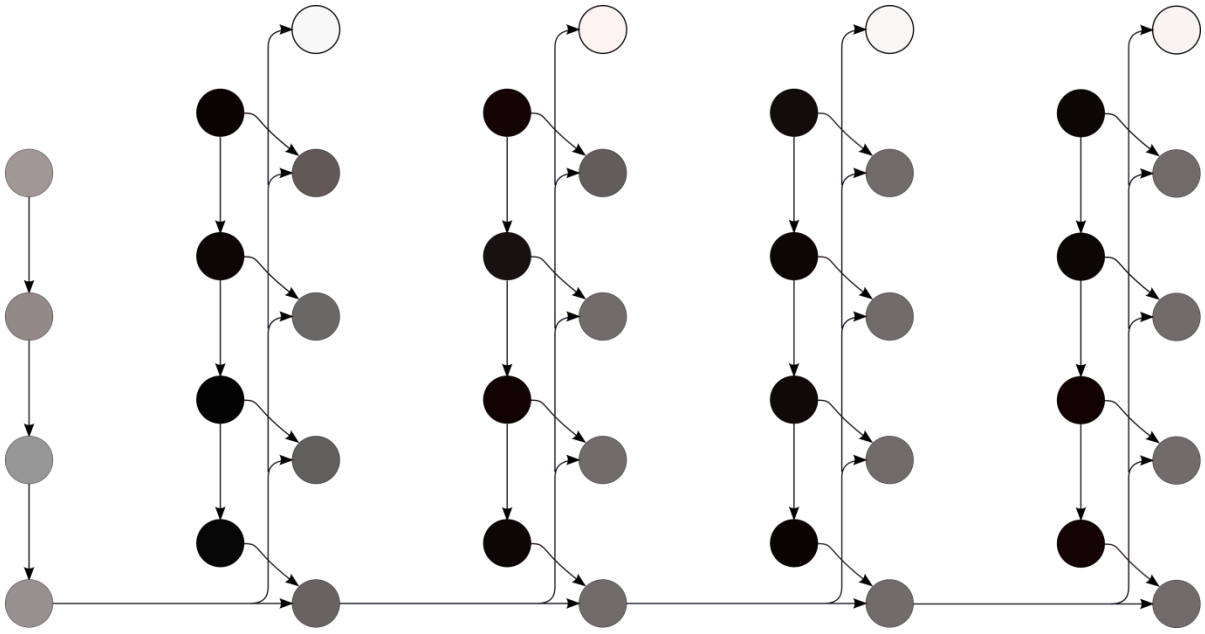

Рис. 3. Алгоритм нового метода для вычисления разложения трёхдиагональной матрицы при *n=25* и равномерном разбиении на 5 промежутков, разная интенсивность соответствует разным операциям

4. Блочная версия

Пусть теперь нам нужно выполнить разложение блочно-трёхдиагональной матрицы

$$A = \begin{pmatrix} A_1 & B_1 & 0 & & & & \\ C_1 & A_2 & B_2 & \cdots & & 0 & \\ 0 & C_2 & A_3 & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & A_{n-2} & B_{n-2} & 0 \\ & 0 & & \cdots & C_{n-2} & A_{n-1} & B_{n-1} \\ & & & & 0 & C_{n-1} & A_n \end{pmatrix} \quad (33)$$

где A_k, B_k, C_k – квадратные блоки одинакового размера. Для разложения

$$A = LU \quad (34)$$

где

$$L = \begin{pmatrix} E & 0 & \cdots & 0 \\ L_1 & E & & \\ \vdots & \ddots & & \vdots \\ & & E & 0 \\ 0 & \cdots & L_{n-1} & E \end{pmatrix} \quad (35)$$

$$U = \begin{pmatrix} U_1 & R_1 & \cdots & 0 \\ 0 & U_2 & & \\ \vdots & \ddots & & \vdots \\ & & U_{n-1} & R_{n-1} \\ 0 & \cdots & 0 & U_n \end{pmatrix} \quad (36)$$

существуют формулы, являющиеся частью блочной прогонки:

$$U_1 = A_1, \quad L_k = C_k U_k^{-1}, \; R_k = B_k, \; U_k = A_k - L_{k-1}R_{k-1} = A_k - C_{k-1}U_{k-1}^{-1}B_{k-1} \quad (37)$$

и в общем случае введение связи между произведениями разных U_k невозможно по простой причине – умножение блоков, в отличие от умножения чисел, некоммутативно. Ситуация, однако, меняется, если блоки B_k окажутся скалярными матрицами, т.е. $B_k = \beta_k E$. Эти матрицы коммутируют с любыми другими, поэтому последняя из формул (37) может быть переписана в виде

$$U_k = A_k - C_{k-1}B_{k-1}U_{k-1}^{-1} \quad (38)$$

а после домножения этого равенства справа на произведение $U_{k-1}U_{k-2} \dots U_1$ это даст нам

$$Q_k = A_k Q_{k-1} - C_{k-1}B_{k-1}Q_{k-2} \quad (39)$$

где

$$Q_k = U_k U_{k-1} \dots U_1, \qquad Q_1 = U_1, \quad Q_0 = E \quad (40)$$

Теперь наличие двучленной рекурсии позволяет нам повторить приём объединения блоков в блочные «вектора»:

$$W_k = \begin{pmatrix} Q_k \\ Q_{k-1} \end{pmatrix} \quad (41)$$

и получить аналогичную (12) формулу

$$W_k = F_k W_{k-1}, k = 2, \dots, n \quad (42)$$

где

$$F_k = \begin{pmatrix} A_k & -C_{k-1}B_{k-1} \\ E & 0 \end{pmatrix} \quad (43)$$

Теперь мы можем повторить рассуждения, аналогичные формулам из 3.1. Выполним в (42) подстановки не до конца, а до некоторого значения $i<k$:

$$W_k = F_k F_{k-1} \dots F_{i+1} W_i \quad (44)$$

обозначим произведения $F_k F_{k-1} \dots F_{i+1}$:

$$F_k F_{k-1} \dots F_{i+1} = G_{ki} \quad (45)$$

обозначим блоки верхней «строки» этой матрицы как S_{ki} и T_{ki}. Тогда из вида матрицы F_k видно, что

$$G_{ki} = \begin{pmatrix} S_{ki} & T_{ki} \\ S_{k-1\,i} & T_{k-1\,i} \end{pmatrix} \quad (46)$$

($S_{ii} = E, T_{ii} = 0$). Поэтому

$$Q_k = S_{ki}Q_i + T_{ki}Q_{i-1} \quad (47)$$

$$Q_{k-1} = S_{k-1\,i}Q_i + T_{k-1\,i}Q_{i-1} \quad (47')$$

Учитывая (40), получаем

$$U_k = Q_k Q_{k-1}^{-1} = (S_{ki}Q_i + T_{ki}Q_{i-1})(S_{k-1\,i}Q_i + T_{k-1\,i}Q_{i-1})^{-1} \quad (48)$$

$$U_k = (S_{ki}Q_i + T_{ki}Q_{i-1})E(S_{k-1\,i}Q_i + T_{k-1\,i}Q_{i-1})^{-1} \quad (49)$$

$$U_k = (S_{ki}Q_i + T_{ki}Q_{i-1})Q_{k-1}^{-1}Q_{i-1}(S_{k-1\,i}Q_i + T_{k-1\,i}Q_{i-1})^{-1} \quad (50)$$

$$U_k = (S_{ki}Q_i Q_{i-1}^{-1} + T_{ki})(Q_{i-1}^{-1})^{-1}(S_{k-1\,i}Q_i + T_{k-1\,i}Q_{i-1})^{-1} \quad (51)$$

$$U_k = (S_{ki}Q_i Q_{i-1}^{-1} + T_{ki})((S_{k-1\,i}Q_i + T_{k-1\,i}Q_{i-1})Q_{i-1}^{-1})^{-1} \quad (52)$$

$$U_k = (S_{ki}Q_i Q_{i-1}^{-1} + T_{ki})(S_{k-1\,i}Q_i Q_{i-1}^{-1} + T_{k-1\,i})^{-1} \quad (53)$$

$$U_k = (S_{ki}U_i + T_{ki})(S_{k-1\,i}U_i + T_{k-1\,i})^{-1} \quad (54)$$

и, таким образом, нам не нужно вычислять произведения матриц. От роста же матриц S_{ki} и T_{ki} можно попытаться избавиться с помощью нормировки. К сожалению, она не может быть впол-

не подобна нормировке в (3.2), и вопрос об её устойчивости остаётся открытым. Приведём возможную схему.

Весь диапазон натуральных чисел *от 1 до n* разбивается на *q* промежутков – *от 1 до* k_1*, от* k_1+1 *до* k_2*, ..., от* $k_{q-1}+1$ *до* $k_q=n$. На первом промежутке все значения U_i вычисляются последовательно, по стандартным формулам (37). На остальных промежутках выполняется следующая процедура.

Пусть рассматривается *j+1*-й промежуток. Тогда уже известны значения

$$S_{q_j q_j} = E, \qquad T_{q_j q_j} = 0, \tag{55}$$

$$\hat{S}_{q_j+1\, q_j} = A_{q_j+1}, \qquad \hat{T}_{q_j+1\, q_j} = -C_{q_j} B_{q_j} \tag{55'}$$

Для всех значений i *от* $q_j + 2$ *до* q_{j+1} теперь вычисляем

$$\delta_i = \frac{1}{\left\| \hat{S}_{i-1\, q_j} \right\|} \tag{56}$$

$$\hat{S}_{iq_j} = \delta_i (A_i \hat{S}_{i-1\, q_j} - C_{i-1} B_{i-1} S_{i-2\, q_j}) \tag{57}$$

$$\hat{T}_{iq_j} = \delta_i (A_i \hat{T}_{i-1\, q_j} - C_{i-1} B_{i-1} T_{i-2\, q_j}) \tag{57'}$$

$$S_{i-1\, q_j} = \delta_i \hat{S}_{i-1\, q_j} \tag{58}$$

$$T_{i-1\, q_j} = \delta_i \hat{T}_{i-1\, q_j} \tag{58'}$$

Формулы (30) – (31') как раз включают в себя нормировку. После того, как выполнены вычисления на всех промежутках, на них (кроме первого) ещё неизвестны значения U_i. Их мы для каждого *j+1-го* промежутка вычисляем параллельно по формулам

$$U_i = \left(\hat{S}_{iq_j} U_{q_j} + \hat{T}_{iq_j} \right) (S_{i-1\, q_j} U_{q_j} + T_{i-1\, q_j})^{-1} \tag{59}$$

Естественно, что каждое значение $U_{q_{j+1}}$ можно вычислить только после вычисления U_{q_j}. В результате, если разбиение на интервалы будет проведено равномерно, структура алгоритма разложения будет иметь тот же вид, что и на **Рис. 13**. Естественно, на этот раз вершинам графа будут соответствовать более сложные операции.

Конечно, скалярность блоков B_k – довольно сильное ограничение для того, чтобы можно было распараллелить блочную прогонку. На деле можно ограничиться менее сильным ограничением – их невырожденностью. Действительно, пусть все наддиагональные блоки B_k матрицы

$$A = \begin{pmatrix} A_1 & B_1 & 0 & & & & \\ C_1 & A_2 & B_2 & \cdots & & 0 & \\ 0 & C_2 & A_3 & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & A_{n-2} & B_{n-2} & 0 \\ & 0 & & \cdots & C_{n-2} & A_{n-1} & B_{n-1} \\ & & & & 0 & C_{n-1} & A_n \end{pmatrix} \tag{60}$$

невырождены. Тогда вместо матрицы *A* мы можем выполнить предложенным способом разложение матрицы *DA*, где

$$D = diag\,\{B_1^{-1}, B_2^{-1}, \ldots, B_{n-1}^{-1}, E\} \tag{61}$$

Действительно, матрица

$$DA = \begin{pmatrix} B_1^{-1}A_1 & E & 0 & & & & \\ B_2^{-1}C_1 & B_2^{-1}A_2 & E & \cdots & & 0 & \\ 0 & B_3^{-1}C_2 & B_3^{-1}A_3 & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & B_{n-2}^{-1}A_{n-2} & E & 0 \\ & 0 & & \cdots & B_{n-1}^{-1}C_{n-2} & B_{n-1}^{-1}A_{n-1} & E \\ & & & & 0 & C_{n-1} & A_n \end{pmatrix} \tag{62}$$

удовлетворяет заявленному требованию скалярности наддиагональных блоков и потому может быть разложена предложенным методом. К сожалению, ограничение невырожденности для блоков, составляющих хотя бы одну из побочных диагоналей, всё же довольно сильно ограничивает область применимости блочной версии последовательно-параллельного метода разложения блочно-трёхдиагональной матрицы.

После этого вместо систем вида (1) можно будет либо решать СЛАУ вида

$$DAx = Db \quad (63)$$

либо домножить матрицу L из полученного LU-разложения матрицы DA на матрицу D^{-1} слева и работать с полученным LU-разложением.

Остаётся отметить, что решения СЛАУ с полученными из разложения блочно-двухдиагональными матрицами распараллеливаются по схеме, принципиально не отличающейся от схемы, получающейся из формул (10) – (14), с заменой скалярных операций на блочные, и с возможностью упрощения по последовательно-параллельной схеме, как на **Рис. 12.** В отличие от блочной схемы разложения, при этом распараллеливании не нужно применять специальные методы для обеспечения устойчивости – она у распараллеленной схемы та же, что и у последовательных блочных версий.

5. Возможные параллели между методом и другими схемами

Автор при разработке последовательно-параллельного метода для разложения трёхдиагональной матрицы, изложенного в части 3, прежде всего руководствовался целью распараллелить нахождение того же LU-разложения трёхдиагональной матрицы, что получается последовательной схемой, получаемой из компактной схемы метода Гаусса применительно к трёхдиагональным матрицам. В условиях точных вычислений предложенный метод эквивалентен как указанной версии компактной схемы метода Гаусса, так и методу рекурсивного сдваивания Стоуна. Структура расположения ненулевых элементов матрицы A дублируется аналогичной структурой разложения (см. **Рис. 14**).

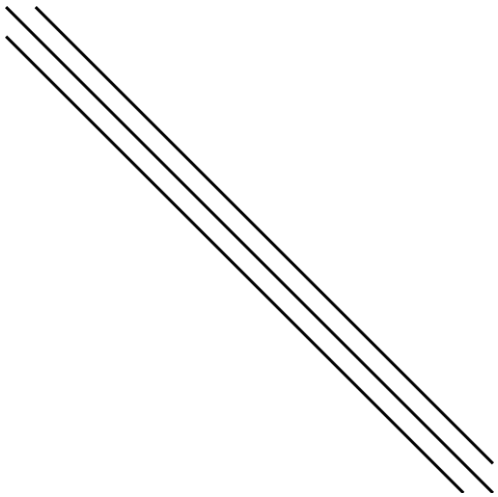

Рис. 4. Структура расположения ненулевых элементов исходной матрицы. В нижнем треугольнике она дублируется структурой матрицы L, в верхнем – структурой матрицы U.

Другие алгоритмы, возможно, похожие по схеме, дают другие разложения. Например, один из читателей увидел схожесть предложенного метода и следующего: «перестановки такие, что в трехдиагональной матрице разделители уходят в конец матрицы; возникает набор не связанных блоков и окаймление; такую матрицу можно устойчиво факторизовать в параллельном режиме» (возможный пример такого с одним разделением приведён на **Рис. 15**).

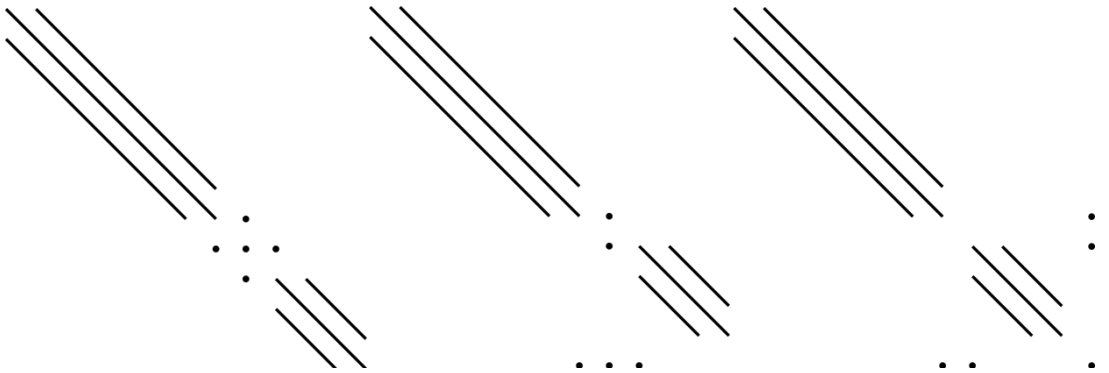

Рис. 5. Выполнение перестановок в окаймлении, ассоциации с которым могут возникнуть у читателя статьи. Выполнено только одно разделение матрицы на фрагменты

На **Рис. 16** показана структура ненулевых элементов получаемого разложения.

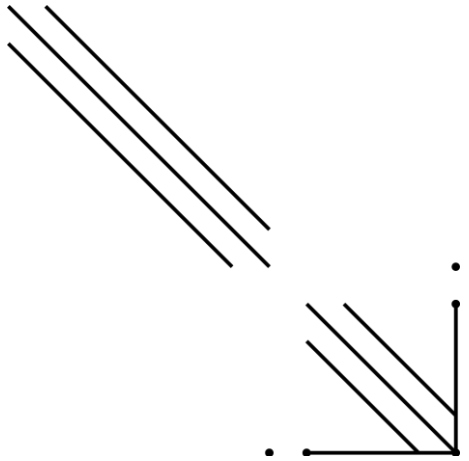

Рис. 6. Структура расположения ненулевых элементов разложения «переставленной» матрицы. В нижнем треугольнике она дублируется структурой матрицы *L*, в верхнем – структурой матрицы *U*.

Хорошо видно появление дополнительных ненулевых элементов. При разбиении перестановками верхнего левого трёхдиагонального блока исходной матрицы будут возникать аналогичные структуры. Кроме этого, таким методом, в отличие от авторского, вычисляется *LU*-разложение не для исходной матрицы *A*, а для матрицы PAP^T, где *P* – матрица перестановок .Поэтому такая схема явно неэквивалентна предложенному автором методу, в котором вычисляемое разложение не имеет таких дополнительных элементов. При этом, однако, подмеченная читателем схожесть в том, что основными частями разложения и там, и там является параллельная обработка отдельных трёхдиагональных фрагментов исходной матрицы, плюс некоторое добавление. Однако и обработка фрагментов, и добавления – разные в разных методах. К слову, несмотря на дополнительное «заплывание» структуры разложения в методе такого окаймления тот, пожалуй, более пригоден для распараллеливания разложения блочно-трёхдиагональной матрицы в тех случаях, когда у нас нет невырожденности блоков хотя бы на одной из её побочных диагоналей.

6. Заключение

Переход от приёма сдваивания к варианту последовательно-параллельного выполнения для использования ассоциативности операций позволил сконструировать такой метод, где, благодаря использованию приёма нормировки, характерного для последовательных операций [2], возможно стало сочетать параллельность с устойчивостью разложения трёхдиагональной матрицы. Аналогичный приём, который автор предлагает использовать и при решении двухдиагональных СЛАУ, получающихся после разложения, также позволяет упростить схему организа-

ции вычислений и снять жёсткие требования на количество устройств, позволяя запрограммировать эффективное параллельное исполнение даже при сравнительно небольшом уровне параллелизма вычислительной системы.

У читателей, возможно, возникнут вопросы по поводу апробации метода и его сравнению с другими методами решения этой же задачи. На апробацию у автора просто пока не было времени (контуры метода намечены в апреле, а в нынешнем виде он записан в середине мая 2015г.), но в ближайшем будущем запланировано заняться и апробацией метода, и детальным его сравнением с другими.

Кроме получения непосредственно новых схем вычисления, автор рекомендует читателю присмотреться к последовательно-параллельной схеме вычислений там, где схемы сдваивания не дают возможности построить устойчивые алгоритмы. Не исключено, что подобные замены могут помочь и в других аналогичных случаях.

Литература

1. Воеводин В.В. Вычислительные основы линейной алгебры. М.: Наука, 1977.
2. Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления. М.: Наука, 1984.
3. Воеводин В.В. Математические основы параллельных вычислений // М.: Изд. Моск. ун-та, 1991.
4. Открытая энциклопедия свойств алгоритмов. URL: http://algowiki-project.org (дата обращения: 28.05.2015).
5. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. М.: Наука, 1978.
6. Фролов А.В. Использование последовательно-параллельного метода для распараллеливания алгоритмов с ассоциативными операциями // Представлена в качестве доклада на первую объединенную международную конференцию "Суперкомпьютерные дни в России", Москва, 28-29 сентября 2015 г.
7. Buneman O. A Compact Non-iterative Poisson Solver // Rep. 294, Inst. for Plasma Res., Stanford U., Stanford, Calif., 1969.
8. Buzbee B.L., Golub G.H., Nielson C.W. On Direct Methods for Solving Poisson's Equations // SIAM J. Numer. Anal., Vol. 7, No. 4 (Dec. 1970), P. 627-656.
9. Stone H.S. An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations // J. ACM, Vol. 20, No. 1 (Jan. 1973), P. 27-38.
10. Stone H.S. Parallel Tridiagonal Equation Solvers // ACM Trans. on Math. Software, Vol. 1, No. 4 (Dec. 1975), P. 289-307.

Yet another tridiagonal matrix algorithm parallelizing method

Alexey Frolov

Keywords: Thomas algorithm, tridiagonal matrix algorithm, parallelizing

New tridiagonal matrix algorithm parallelizing method is described in this article.

This method uses matrix multiplication associativity, and is stable for standard data of Thomas algorithm.