ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2015. Т. 19, № 4 (70). С. 149–162 http://journal.ugatu.ac.ru

УДК 004.021

Исследование структурных свойств алгоритма разложения Холецкого: от давно известных фактов до новых выводов

А. В. Фролов[1], Вад. В. Воеводин[2], И. Н. Коньшин[3], А. М. Теплов[4]

[1] frolov@mail.inm.ras.ru, [2] vadim_voevodin@mail.ru, [3] igor.konshin@gmail.com, [4] alex-teplov@yandex.ru

[1,3] Федеральное государственное бюджетное учреждение науки Институт вычислительной математики Российской академии наук (ИВМ РАН)

[2,4] Научно-исследовательский вычислительный центр Московского государственного университета имени М.В.Ломоносова (НИВЦ МГУ)

[3] Федеральное государственное бюджетное учреждение науки Вычислительный центр им. А. А. Дородницына Российской академии наук (ВЦ РАН)

Поступила в редакцию 1.12.2015

Аннотация. В рамках создания Открытой энциклопедии свойств алгоритмов проведено исследование параллельной структуры алгоритма разложения Холецкого для симметричных положительно определенных матриц. На основе анализа графа алгоритма приводится описание ресурса параллелизма данного алгоритма, а также исследованы характеристики локальности обращений к памяти для данного алгоритма. Полученные в результате численных экспериментов на суперкомпьютере «Ломоносов» данные исследований масштабируемости базовых реализаций разложения Холецкого позволяют дать ряд рекомендаций по соотношению размеров задач и ресурсов, необходимых для их решения. Результаты теоретических исследований дают возможность, как объяснить сравнительно низкую производительность точечного варианта этого метода, так и по-новому взглянуть на некоторые решения, принятые в вычислительном сообществе более полувека назад, и пересмотреть их с позиций новых реалий при современном состоянии вычислительной техники.

Ключевые слова: параллельная структура алгоритма; метод Холецкого; отображение алгоритма на архитектуру вычислительных систем.

ВВЕДЕНИЕ

Разложение Холецкого впервые предложено французским офицером и математиком Андре-Луи Холецким в конце Первой Мировой войны, незадолго до его гибели в бою в августе 1918 г. Идея этого разложения была опубликована в 1924 г. его сослуживцем [1]. Потом использовано поляком Т. Банашевичем в 1938 г. [2–3]. В советской математической литературе называется также *методом квадратного корня* [4–6]; название связано с характерными операциями, отсутствующими в родственных разложениях Гаусса и Жордана.

Первоначально разложение Холецкого использовалось исключительно для плотных симметричных положительно определенных матриц. В настоящее время его использование гораздо шире. Оно может быть применено также, например, к комплексно-сопряженным матрицам. Для повышения производительности вычислений довольно часто применяется не точечная, а блочная версия разложения.

Для разреженных матриц разложение Холецкого также широко применяется в качестве

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2015». Результаты, приведенные в разделах 1, 2.1, 2.3–2.6, 3, получены в Московском государственном университете имени М.В. Ломоносова за счет гранта Российского научного фонда (проект №14-11-00190). Раздел 2.2 выполнен при поддержке РФФИ (грант № 13-07-00787). Работа выполнена с использованием ресурсов суперкомпьютерного комплекса МГУ им. М.В. Ломоносова [7].

основного этапа прямого метода решения линейных систем. В этом случае используют специальные упорядочивания для уменьшения ширины профиля исключения, а следовательно, и уменьшения количества арифметических операций. Другие упорядочивания используются для выделения независимых блоков вычислений при работе на параллельных системах.

Варианты разложения Холецкого нашли успешные применения и в итерационных методах для построения переобуславливателей разреженных симметричных положительно определенных матриц. В неполном треугольном разложении («по позициям») элементы переобуславливателя вычисляются только в заранее заданных позициях, например, в позициях ненулевых элементов исходной матрицы (так называемое разложение IC0). Для получения же более точного разложения применяется приближение, в котором фильтрация малых элементов производится «по значениям». В зависимости от используемого порога фильтрации можно получить более точное, хотя и более заполненное разложение. Существует и алгоритм разложения второго порядка точности [8]. В нем при таком же заполнении множителей разложения удается улучшить точность. Для такого разложения в параллельном режиме используется вариант аддитивного переобуславливания на основе разложения второго порядка [9].

В настоящей работе авторы попытались взглянуть на исходное разложение Холецкого с новых позиций нашего суперкомпьютерного века и в рамках работы над созданием Открытой энциклопедии свойств алгоритмов [10]. По этой причине большая часть настоящей работы выполнена в стиле и с последовательностью подачи материала Открытой энциклопедии. Кроме того, изложение существенно расширено и дополнено. Приведено описание конкретной версии разложения Холецкого – для плотных вещественных симметричных положительно определенных матриц, но структура для ряда других версий, например, для комплексного случая, почти такая же, различия состоят в замене большинства вещественных операций на комплексные.

1. ОПИСАНИЕ СВОЙСТВ И СТРУКТУРЫ АЛГОРИТМА

1.1. ОБЩЕЕ ОПИСАНИЕ АЛГОРИТМА

Разложение Холецкого используется для разложения положительно определенных эрмитовых (в вещественном случае – симметричных) матриц в виде $A=LL^*$ (L – нижняя треугольная матрица) или $A=U^*U$ (U – верхняя треугольная матрица; эти разложения эквивалентны друг другу по вычислениям и разные только по способу хранения данных). Он заключается в реализации формул (п.1.2) для элементов L. Распространено благодаря таким особенностям.

1) *Симметричность матрицы*, которая позволяет хранить и вычислять только чуть больше половины ее элементов, что почти вдвое экономит как необходимые для вычислений объемы памяти, так и количество операций в сравнении, например, с разложением по методу Гаусса. При этом альтернативное LU-разложение, использующее симметрию матрицы, все же несколько быстрее разложения Холецкого (там нет извлечения квадратных корней), но требует больше памяти, ограниченность которой оказалась решающей для вычислителей времен 50-х гг. XX в.

2) *Режим накопления:* значительную часть времени занимают вычисления скалярных произведений; благодаря ему разложение Холецкого имеет наименьшее эквивалентное возмущение из всех известных разложений матриц.

1.2. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ

Исходные данные: положительно определенная симметричная матрица A (элементы a_{ij}).

Вычисляемые данные: левая треугольная матрица L (элементы l_{ij}).

Формулы метода:

$$l_{11} = \sqrt{a_{11}},$$

$$l_{j1} = \frac{a_{j1}}{l_{11}}, \quad \text{при } j = 2,\dots, \text{n},$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}, \qquad \text{при } i = 2, \dots, n,$$

$$l_{ji} = \left(a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right) / l_{ii},$$

при $i = 2, \dots, n-1$ и $j = i+1, \dots, n$.

В ряде реализаций деление на диагональный элемент выполняется в два этапа: вычисление $1/l_{ii}$ и затем умножение на него всех (видоизмененных) a_{ji}. Здесь мы этот вариант алгоритма не рассматриваем. Заметим только, что он имеет худшие параллельные характеристики, чем представленный. Существует также блочная версия разложения, однако в данной работе разобран только точечный вариант.

1.3. ВЫЧИСЛИТЕЛЬНОЕ ЯДРО И МАКРОСТРУКТУРА АЛГОРИТМА

Вычислительное ядро последовательной версии разложения Холецкого можно составить из множественных (всего их $n(n-1)/2$) вычислений скалярных произведений строк матрицы:

$$\sum_{p=1}^{i-1} l_{ip} l_{jp}$$

в режиме накопления или без него, в зависимости от требований задачи. В отечественных реализациях, даже в последовательных, упомянутый способ представления не используется. Дело в том, что даже в этих реализациях разложения вычисление сумм

$$a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp},$$

в которых и встречаются скалярные произведения, ведутся не в порядке «вычислили скалярное произведение, а потом вычли его из элемента», а путем вычитания из элемента покомпонентных произведений, являющихся частями скалярных произведений. Поэтому следует считать вычислительным ядром разложения не вычисления скалярных произведений, а вычисления сумм

$$a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp}$$

в режиме накопления или без него.

Тем не менее, в популярных зарубежных реализациях точечного варианта разложения Холецкого, в частности, в библиотеках LINPACK и LAPACK, основанных на BLAS, используются именно вычисления скалярных произведений в виде вызова соответствующих подпрограмм BLAS (конкретно – функции SDOT). На последовательном уровне это влечет за собой дополнительную операцию суммирования на каждый из $n(n+1)/2$ вычисляемый элемент матрицы L и некоторое замедление работы программы (о других следствиях рассказано ниже в п.2.7), и в данных вариантах ядром метода Холецкого будут вычисления этих скалярных произведений.

1.4. ОПИСАНИЕ СХЕМЫ РЕАЛИЗАЦИИ ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА

Последовательность операций алгоритма следующая:

1-й шаг

$$l_{11} = \sqrt{a_{11}},$$

2-й шаг

$$l_{j1} = \frac{a_{j1}}{l_{11}},$$

(при j=2,…, n).

Далее для всех i от 2 до n по нарастанию значений выполняются

3й шаг

$$l_{ii} = \sqrt{a_{ii} - \sum_{p=1}^{i-1} l_{ip}^2}$$

и 4й шаг ($i<n$, для всех j от i+1 до n)

$$l_{ji} = \left(a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp} \right) / l_{ii}.$$

После этого (если $i<n$) происходит переход к

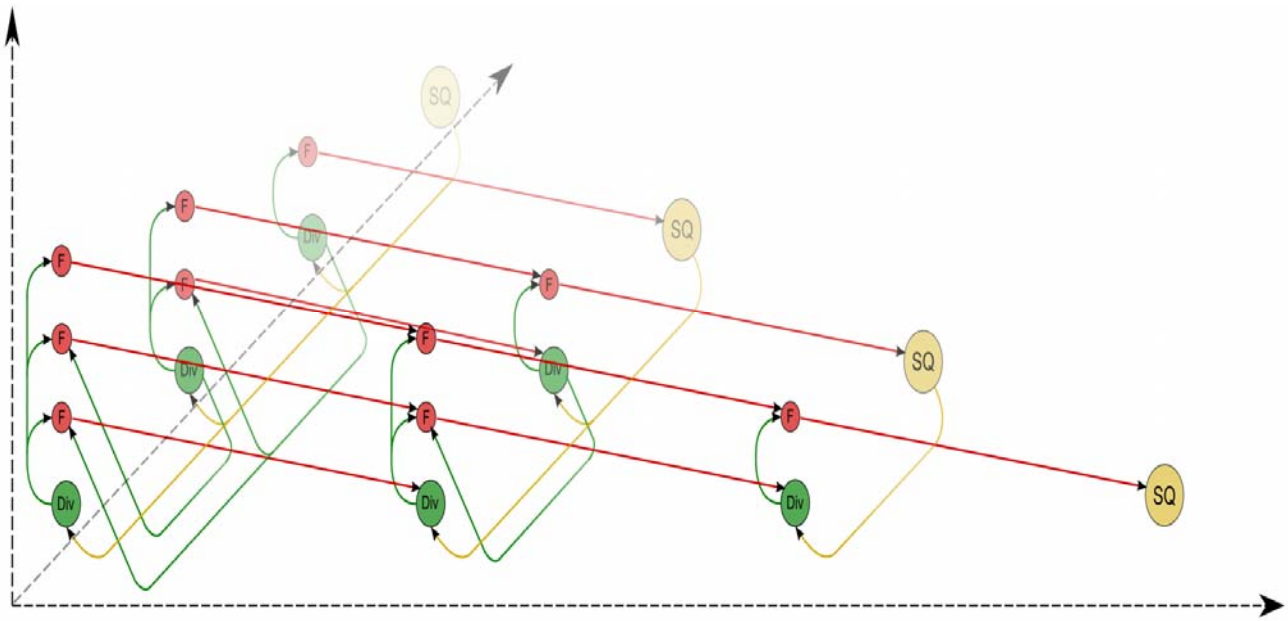

Рис. 1. Граф алгоритма без отображения входных и выходных данных. **SQ** – вычисление квадратного корня, **F** – операция исключения *a-bc*, **Div** – операция деления.

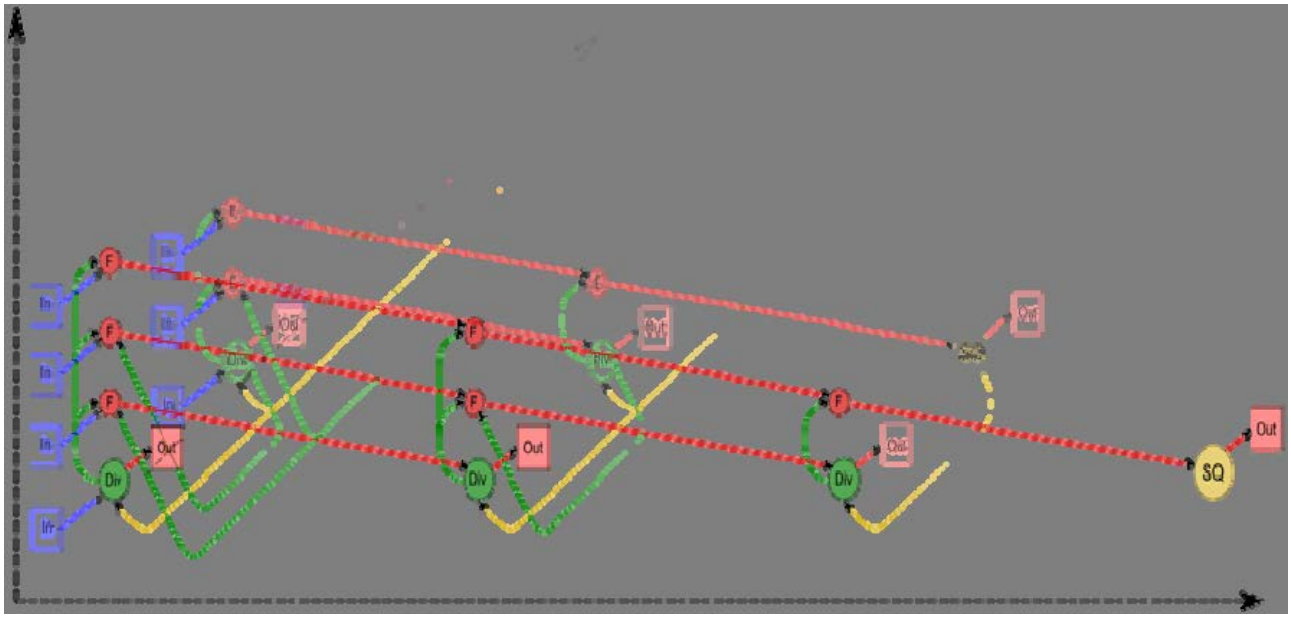

Рис. 2. Граф алгоритма с отображением входных и выходных данных. ***SQ*** – вычисление квадратного корня, ***F*** – операция исключения *a-bc*, ***Div*** – операция деления, ***In*** – входные данные, ***Out*** – результаты.

шагу 3 с бóльшим i.

Особо отметим, что вычисления сумм

$$a_{ji} - \sum_{p=1}^{i-1} l_{ip} l_{jp}$$

в обеих формулах 3-го и 4-го шагов производят в режиме накопления вычитанием из a_{ji} произведений $l_{ip}l_{jp}$ для p от 1 до $i-1$, с нарастанием p.

1.5. ПОСЛЕДОВАТЕЛЬНАЯ СЛОЖНОСТЬ АЛГОРИТМА

Для разложения Холецкого матрицы порядка n в последовательном (наиболее быстром) варианте требуется:

- n вычислений квадратного корня,
- $n(n-1)/2$ делений,
- по $(n^3-n)/6$ умножений и сложений (вычитаний) – основная часть алгоритма.

При этом использование режима накопления требует совершения умножений и вычитаний в режиме двойной точности (или использования функции вроде DPROD в Фортране), что еще больше увеличивает долю умножений и сложений/вычитаний во времени, требуемом для выполнения разложения Холецкого. Таким образом, при классификации по последовательной сложности разложение Холецкого относится к алгоритмам с кубической сложностью.

1.6. ИНФОРМАЦИОННЫЙ ГРАФ

Опишем граф алгоритма аналитически, а также представим его в виде рисунка. Аналитическое описание примерно будет соответствовать описанию на языке Сигма [11], но вместо управляющих конструкций этого языка у нас будут словесные пояснения.

Граф алгоритма состоит из трех групп вершин, расположенных в целочисленных узлах трех областей разной размерности.

Первая группа вершин расположена в одномерной области, соответствующая ей операция вычисляет функцию SQRT. Естественно введенная единственная координата каждой из вершин i изменяется в диапазоне от 1 до n, пробегая при этом все целочисленные значения.

Аргументом этой функции является: при $i=1$ – элемент входных данных, а именно a_{11}; при $i>1$ – результат выполнения операции, соответствующей вершине из третьей группы, с координатами $i-1$, i, $i-1$.

Результат выполнения операции является выходным данным l_{ii}.

Вторая группа вершин расположена в двумерной области, и соответствующей ей операцией является a/b. Естественно введенные целочисленные координаты области таковы:

i – меняется в диапазоне от 1 до $n-1$; j – меняется в диапазоне от $i+1$ до n.

```
DO  I = 1, N
          S = A(I,I)
          DO  P = 1, I-1
                S = S - DPROD(A(I,P), A(I,P))
          END DO
          A(I,I) = SQRT (S)
          DO  J = I+1, N
                S = A(J,I)
                DO  P = 1, I-1
                      S = S - DPROD(A(I,P), A(J,P))
                END DO
                A(J,I) = S/A(I,I)
          END DO
END DO
```

Рис. 3. Классический вариант последовательной реализации разложения Холецкого

В качестве аргументов в этой операции используются:

как делимое при i=1 – элементы входных данных a_{j1}; как делимое при i>1 – результат выполнения операции, соответствующей вершине из третьей группы, с координатами i−1, j, i−1;

как делитель – результат срабатывания операции, соответствующей вершине из первой группы, с координатой i.

Результат выполнения операции является выходным данным l_{ji}.

Третья группа вершин расположена в трехмерной области, и соответствующей ей операцией является $a-bc$. Естественно введенные целочисленные координаты области таковы:

i – меняется в диапазоне от 2 до n; j – меняется в диапазоне от i до n; p – меняется в диапазоне от 1 до i−1.

В качестве аргументов этой операции используются:

как уменьшаемое в вычитании при p=1 – элемент входных данных a_{ji}; как уменьшаемое при p>1 – результат выполнения операции, соответствующей вершине из третьей группы, с координатами i, j,p−1;

как множимое – результат выполнения операции, соответствующей вершине из второй группы, с координатами p, i;

как множитель – результат выполнения операции, соответствующей вершине из второй группы, с координатами p, j;

Результат выполнения этой операции является «промежуточным данным» алгоритма.

Описанный граф, выполненный для случая n=4, изображен на рис. 1 и 2, аналогичные которым, но более детальные и цветные, есть на странице метода Холецкого сайта Открытой энциклопедии алгоритмов [10]. Здесь вершины первой группы обозначены SQ (на сайте они также выделены желтым цветом), вершины второй – Div (на сайте – зеленым цветом), третьей – F (на сайте – красным цветом). Вершины, соответствующие операциям, производящим выходные данные алгоритма, выполнены более крупно. Дублирующие друг друга дуги графа представлены как одна. На рис. 1 представлен граф алгоритма согласно классическому определению, а на рис. 2 к графу алгоритма добавлены вершины, соответствующие входным данным (обозначены In и на сайте они синего цвета) и выходным данным (обозначены Out и на сайте они розового цвета).

1.7. ОПИСАНИЕ РЕСУРСА ПАРАЛЛЕЛИЗМА АЛГОРИТМА

Для разложения Холецкого матрицы порядка n в параллельном варианте требуется последовательно выполнить следующие ярусы графа алгоритма:

- n ярусов с вычислением квадратного корня (единичные вычисления в каждом из ярусов),
- n−1 ярус делений (в каждом из ярусов количество делений линейно, в зависимости от яруса – от 1 до n−1),
- по n−1 ярусов умножений и сложений/вычитаний (в каждом из ярусов – количество операций квадратично, от 1 до $(n^2-n)/2$).

Таким образом, в параллельном варианте, в отличие от последовательного, вычисления квадратных корней и делений будут определять довольно значительную долю требуемого времени. При реализации на конкретных архитектурах наличие в отдельных ярусах ярусно-параллельной формы (ЯПФ) отдельных вычислений квадратных корней может породить и другие проблемы. Например, при реализации на современных вычислительных системах, основные вычисления (деления и тем более умножения и сложения/вычитания) могут быть конвей-

```
DO  I = 1, N
          A(I,I) = SQRT (A(I, I))
          DO  J = I+1, N
                A(J,I) = A(J,I)/A(I,I)
          END DO
          DO  K = I+1, N
                DO J = K, N
                      A(J,K) = A(J,K) - A(J,I)*A(K,I)
                END DO
          END DO
   END DO
```

Рис. 4. Последовательная реализация разложения Холецкого без накопления

еризованы, вычисления же квадратных корней из-за их изолированности могут привести к дисбалансу вычислений, а следовательно и простоям. Таким образом, общая экономия памяти в 2 раза, из-за которой разложение Холецкого предпочитают в случае симметричных задач тому же разложению Гаусса, в параллельном случае уже имеет место вовсе не по всем ресурсам, и главное – не по времени выполнения. При этом использование режима накопления требует совершения умножений и вычитаний в режиме двойной точности, а в параллельном варианте это означает, что практически все промежуточные вычисления для выполнения разложения Холецкого в режиме накопления должны быть двойной точности. В отличие от последовательного варианта это означает увеличение требуемой памяти почти в 2 раза.

При классификации по высоте ЯПФ разложение Холецкого относится к алгоритмам с линейной сложностью. При классификации по ширине ЯПФ его сложность будет квадратичной.

1.8. ОПИСАНИЕ ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ

Входные данные: плотная матрица A (элементы a_{ij}). Дополнительные предположения следующие:

A – симметричная матрица, т. е. $a_{ij}=a_{ji}$, где $i,j=1,\ldots, n$,

A – положительно определенная матрица, т. е. для любых ненулевых векторов x выполняется $x^TAx > 0$.

Объем входных данных: $n(n+1)/2$, т. к. в силу симметричности достаточно хранить только диагональ матрицы и над/поддиагональные элементы. В различных реализациях эта экономия хранения может быть выполнена различным образом: во многих старых библиотеках матрица A хранится в одномерном массиве такой длины по строкам своей нижней части.

Выходные данные: левая треугольная матрица L (элементы l_{ij}).

Объем выходных данных: $n(n+1)/2$, т. к. в силу треугольности достаточно хранить только ненулевые диагональные и поддиагональные элементы.

1.9. СВОЙСТВА АЛГОРИТМА

Соотношение последовательной и параллельной сложности в случае неограниченных ресурсов, как хорошо видно, является *квадратичным* (отношение кубической сложности к линейной). Однако вычислительная мощность алгоритма, определяемая как отношение числа операций к суммарному объему входных и выходных данных, является всего лишь *линейной*. При этом алгоритм почти полностью детерминирован, это гарантируется теоремой о единственности разложения. Использование другого порядка выполнения ассоциативных операций может привести к накоплению ошибок округления, хотя это влияние в используемых вариантах алгоритма не является настолько определяющим, как, например, отказ от использования режима накопления.

Дуги информационного графа, исходящие из вершин, соответствующих операциям квадратного корня и деления, образуют пучки так называемых *рассылок* линейной мощности (т. е. степень исхода этих вершин и мощность работы с этими данными является линейной функцией от порядка матрицы и координат этих вершин). При этом естественно наличие в этих пучках «длинных» дуг. Остальные дуги локальны.

Наиболее известной является компактная укладка графа – его проекция на треугольник матрицы, перевычисляемый укладываемыми операциями. При этом «длинные» дуги можно убрать, заменив более дальнюю пересылку комбинацией нескольких ближних (к соседям).

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. ОСОБЕННОСТИ РЕАЛИЗАЦИИ ПОСЛЕДОВАТЕЛЬНОГО АЛГОРИТМА

В классическом варианте без перестановок суммирования алгоритм разложения Холецкого на Фортране можно записать так, как это показано на рис. 3. В этом варианте для реализации режима накопления переменная суммирования должна иметь двойную точность.

Для разложения Холецкого существует также блочная версия, которая отличается от точечной не тем, что операции над числами заменены на аналоги этих операций над блоками; ее построение основано на том, что практически все циклы точечной версии имеют тип SchedDo в терминах методологии, основанной на исследовании информационного графа [12] и, следовательно, могут быть расщеплены на составляющие. Тем не менее обычно блочную версию разложения Холецкого записывают не в виде программы с расщепленными и переставленными циклами, а в виде программы, подобной реализации точечного алгоритма, в которой вместо соответствующих скалярных операций присутствуют операции над блоками.

Для обеспечения локальности работы с памятью более эффективной представляется такая схема разложения Холецкого (полностью эквивалентная описанной), когда исходная матрица

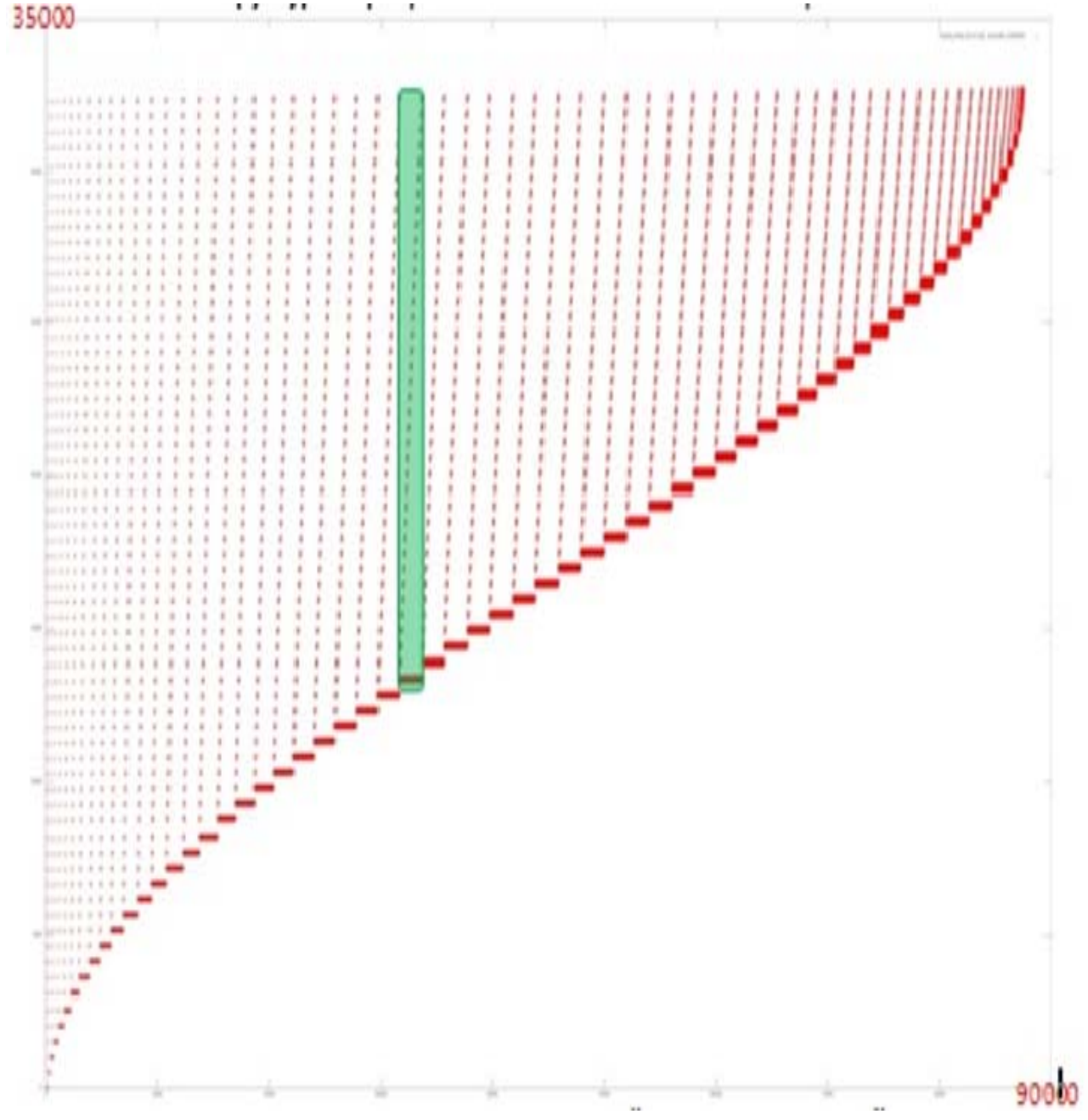

Рис. 5. Реализация разложения Холецкого: общий профиль обращений в память.

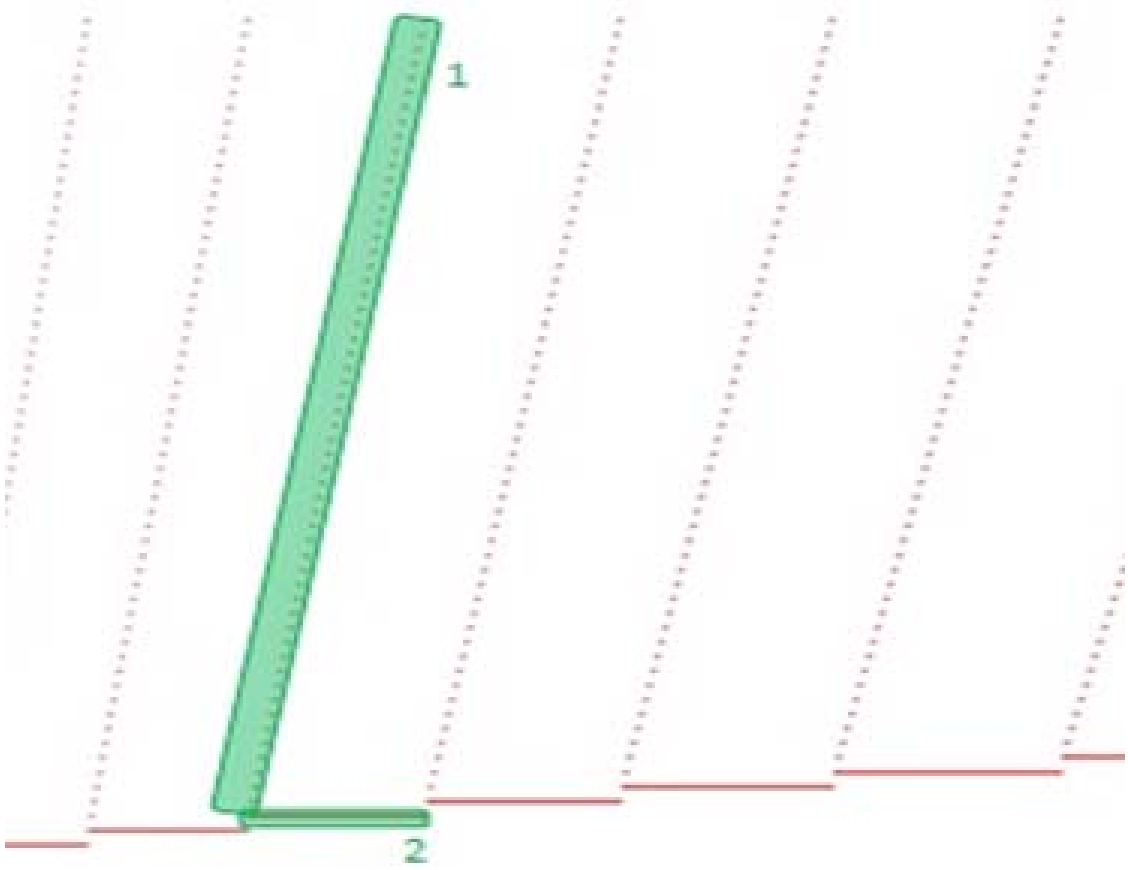

Рис. 6. Реализация разложения Холецкого: фрагмент профиля для нескольких итераций

и ее разложение хранятся не в виде нижней треугольной матрицы *L*, а в верхней треугольной матрицы *U*. Это связано с особенностью размещения массивов в языке Фортран, а также тем, что в этом случае вычисления скалярных произведений будут идти с выборкой идущих подряд элементов массива, увеличивая их локальность.

Есть и другой вариант точечной схемы: использовать вычисляемые элементы матрицы *L* в качестве аргументов непосредственно «сразу после» их вычисления. Такая программа будет выглядеть так, как показано на рис. 4.

Как видно, в этом варианте для реализации режима накопления одинарной точности мы должны будем объявить двойную точность для массива, хранящего исходные данные и результат. Подчеркнем, что информационный граф алгоритма обеих схем остается неизменным (как в п. 1.6), за исключением замены умножения на функцию DPROD.

2.2. ОПИСАНИЕ ЛОКАЛЬНОСТИ АЛГОРИТМА

На рис. 5 представлен профиль обращений в память для рассмотренной реализации разложения Холецкого. Профиль представляет собой последовательность виртуальных адресов, записанных в том порядке, в котором происходят обращения в память в последовательной реализации программы [13]. В таком профиле содержатся данные для оценки локальности, которая кардинальным образом может влиять на общую эффективность выполнения программ [14].

В программе задействован только 1 массив, поэтому обращения в профиле происходят только к элементам этого массива. Программа состоит из одного основного этапа, который, в свою очередь, состоит из последовательности подобных итераций. Пример одной итерации выделен контуром.

Видно, что на каждой *i*-й итерации используются все адреса, кроме первых k_i, при этом с ростом *i* увеличивается значение k_i. Также можно заметить, что число обращений в память на каждой итерации растет примерно до середины работы программы, после чего уменьшается вплоть до завершения работы. Это позволяет говорить о том, что данные в программе используются несколько неравномерно: многие итерации, особенно в начале выполнения программы, используют большой объем данных, что влечет ухудшение локальности. Но все же основным фактором, влияющим на локальность

работы с памятью, является строение итерации. Рассмотрим фрагмент профиля, соответствующий нескольким первым итерациям.

Исходя из рис. 6, видно, что каждая итерация состоит из двух различных фрагментов, выделенных контуром и обозначенных номерами *1* и *2*. Фрагмент *1* – последовательный перебор (с некоторым шагом) всех адресов, начиная с некоторого начального адреса. При этом к каждому элементу происходит мало (а возможно и всего одно) обращений. Такой фрагмент обладает достаточно неплохой пространственной локальностью, так как шаг по памяти между соседними обращениями невелик, но плохой временно́й локальностью, поскольку данные редко используются либо вообще не используются повторно.

Фрагмент *2* устроен значительно лучше с точки зрения локальности, в нем выполняется большое число обращений подряд к одним и тем же данным, что обеспечивает более высокую степень как пространственной, так и временно́й локальности по сравнению с фрагментом *1*.

После рассмотрения фрагмента профиля на рис. 6 можно оценить общую локальность двух фрагментов на каждой итерации. Однако стоит рассмотреть более подробно, как устроен каждый из фрагментов. Рис. 7, на котором представлена часть одной итерации общего профиля, позволяет отметить достаточно интересный факт: строение каждого из фрагментов на самом деле заметно сложнее, чем это выглядит на рис. 6. В частности, каждый шаг фрагмента 1 состоит из нескольких обращений к соседним элементам, причем выполняется не последовательный перебор. Кроме этого, фрагмент номер 2 на самом деле состоит из повторяющихся итераций, и видно: каждый шаг фрагмента 1 соответствует одной итерации фрагмента 2 (выделено контуром на рис. 7). Это говорит о том, что для точного понимания локальной структуры профиля необходимо его рассмотреть на уровне отдельных обращений.

Стоит особо отметить, что выводы на основе рис. 7 просто дополняют общее представлении о строении профиля обращений; при этом сделанные на основе рис. 6 выводы относительно общей локальности двух фрагментов остаются справедливыми.

В целом можно отметить, что профиль разложения Холецкого обладает достаточно высокой как пространственной, так и временно́й локальностью, что обусловлено удачным по работе с памятью внутренним строением итераций.

2.3. ВОЗМОЖНЫЕ СПОСОБЫ И ОСОБЕННОСТИ РЕАЛИЗАЦИИ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА

Как нетрудно видеть по структуре графа алгоритма, вариантов распараллеливания алгоритма довольно много. Например, во втором варианте (см. п. 2.1 и рис. 4) все

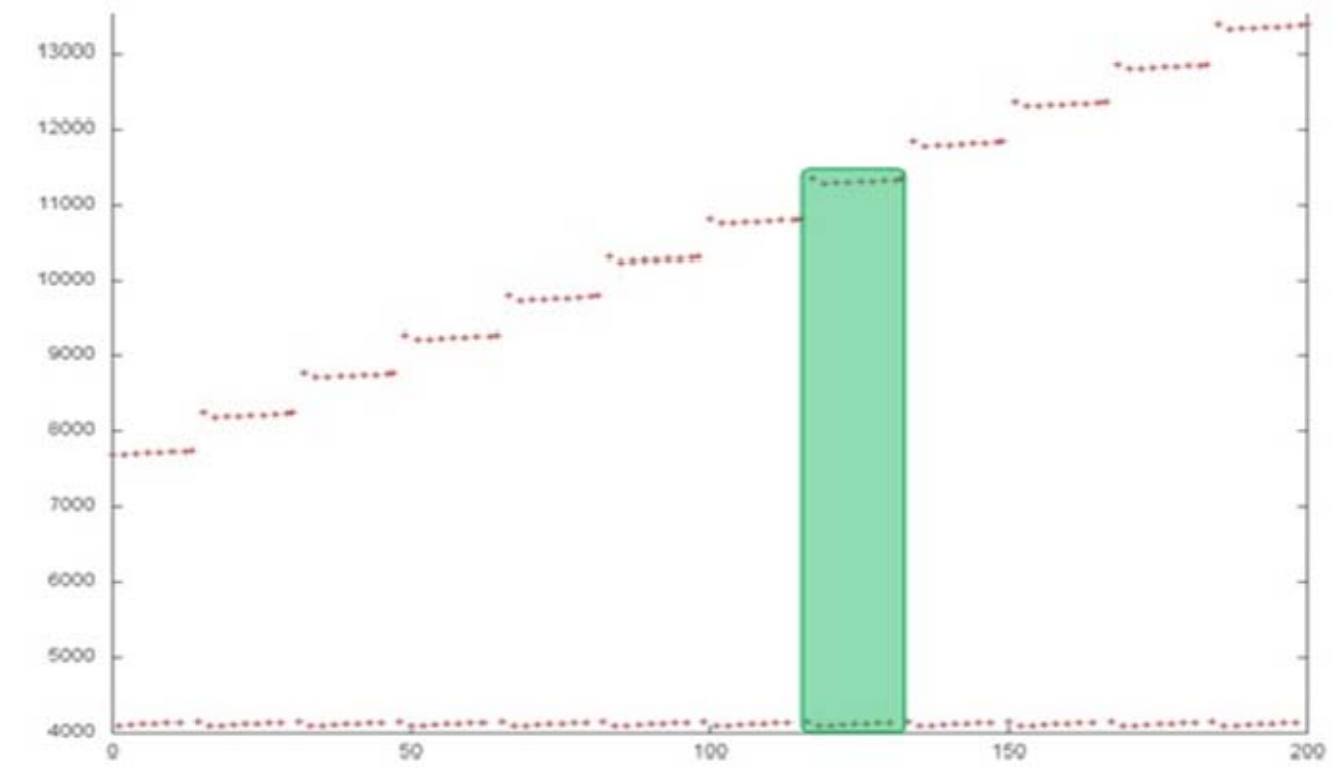

Рис. 7. Реализация разложения Холецкого: фрагмент профиля для части одной итерации

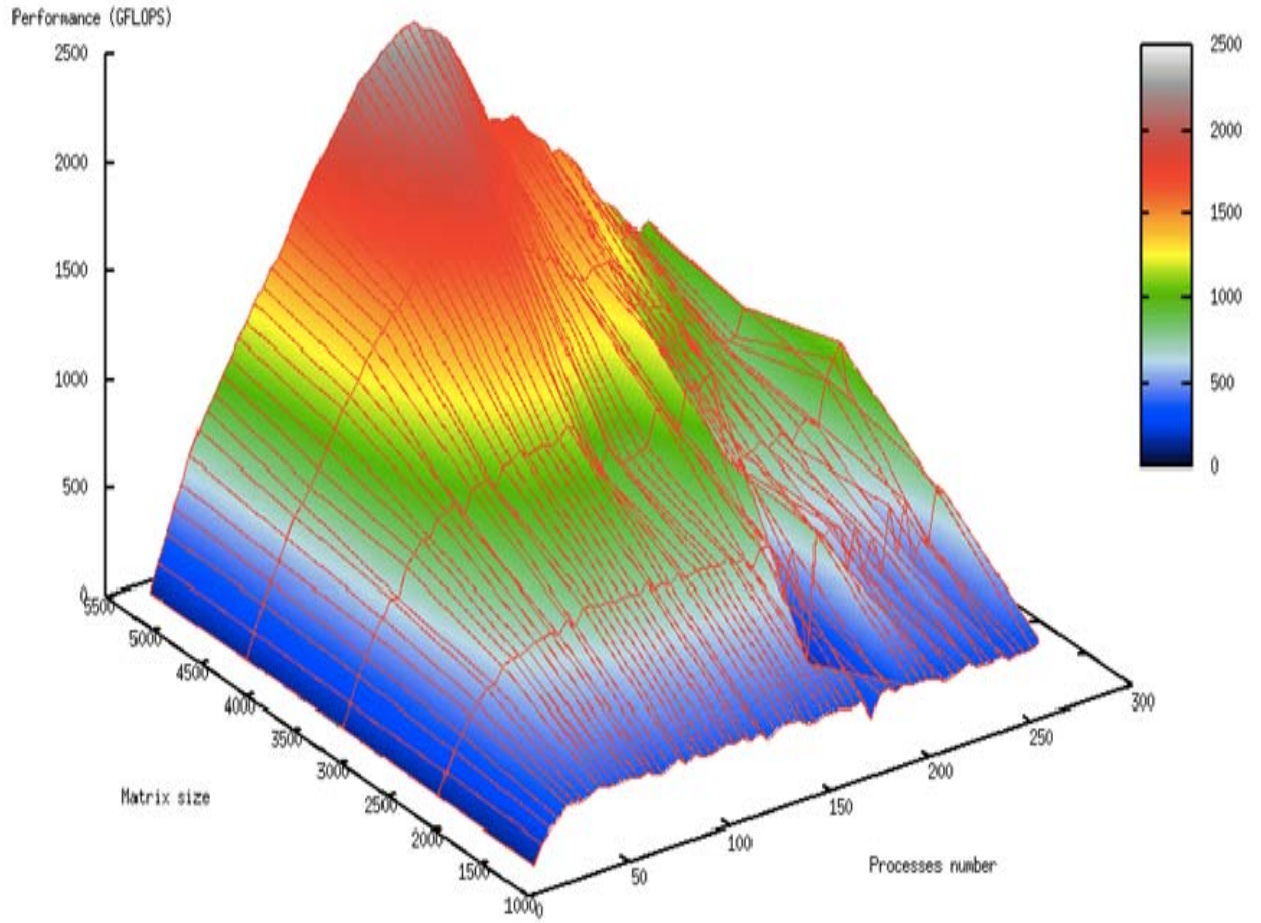

Рис. 8. Производительность параллельной реализации разложения Холецкого в зависимости от числа процессоров (ось направо) и размерности матрицы (ось налево)

внутренние циклы параллельны, в первом (см. рис. 3) – параллелен цикл по j. Тем не менее простейшее распараллеливание «в лоб» вызовет такое количество пересылок между процессорами с каждым шагом внешнего цикла, которое почти сопоставимо с количеством арифметических операций. Поэтому перед размещением операций и данных между процессорами вычислительной системы предпочтительно разбиение всего пространства вычислений на блоки, с сопутствующим разбиением обрабатываемого массива. Многое зависит от конкретного типа вычислительной системы. Присутствие конвейеров на узлах многопроцессорной системы делает рентабельным параллельное вычисление нескольких скалярных произведений одновременно. В принципе для получения большего ускорения возможно и использование так называемого «скошенного» параллелизма.

2.4. МАСШТАБИРУЕМОСТЬ АЛГОРИТМА И ЕГО РЕАЛИЗАЦИИ

Исследование масштабируемости параллельной реализации разложения Холецкого проводилось на суперкомпьютере «Ломоносов» Суперкомпьютерного комплекса Московского университета. Набор и границы значений изменяемых параметров запуска реализации алгоритма:

- число процессоров [4 : 256] с шагом 4;
- размер матрицы [1024 : 5120].

В результате проведенных экспериментов был получен следующий диапазон эффективности реализации алгоритма:

- минимальная эффективность реализации 0,11%;
- максимальная эффективность реализации 2,65%.

На рис. 8 и 9 приведены графики производительности и эффективности выбранной реализации разложения Холецкого в зависимости от изменяемых параметров запуска.

По результатам численных экспериментов были построены осредненные оценки масштабируемости выбранной реализации разложения Холецкого.

По числу процессов: -0,000593. Работа характеризуется крайне низкой общей эффективностью работы приложения с максимумом в 2,65%, и значение эффективности на рассмотренной области значений быстро доходит до десятых долей процента, когда накладные расходы начинают сильно превалировать над вычислениями.

По размеру задачи: 0,06017. При увеличении размера задачи эффективность возрастает. Эффективность возрастает тем быстрее, чем большее число процессов используется для выполнения. Это подтверждает предположение о том, что размер задачи сильно влияет на эффективность выполнения приложения. Оценка показывает, что с ростом размера задачи эффективность на рассмотренной области значений параметров запуска сильно увеличивается.

2.5. ВЫВОДЫ ДЛЯ КЛАССОВ АРХИТЕКТУР

Как видно по показателям SCALAPACK на суперкомпьютерах, обмены при большой размерности n хоть и уменьшают эффективность расчетов, но слабее, чем неоптимальность организации расчетов внутри одного узла. Поэтому, видимо, сначала следует оптимизировать не блочный алгоритм, а используемые на отдельных процессорах точечный вариант разложения, перемножения матриц и др. Ниже содержится информация о возможном направлении такой оптимизации. В отношении же архитектур спецпроцессоров вполне показателен тот момент, что разработчики – наполнители специализированных библиотек – пока что не докладывают об успешных и эффективных реализациях точечного варианта разложения Холецкого на этих вычислительных устройствах. Это связано со следующим свойством информационной структуры алгоритма: если операции деления или вычисления выражений $a-bc$ являются не только массовыми, но и параллельными, и потому их вычисления сравнительно легко выстраивать в конвейеры, то операции извлечения квадратных корней являются узким местом алгоритма – отведенное на эту операцию оборудование неизбежно будет простаивать большую часть времени.

Аналогичная ситуация уже складывается и на многопроцессорных системах: степень распараллеливания при большом числе узлов начинает приближаться к предельной, и удельный вес вычисления квадратных корней, этого узкого места алгоритма, растет. В еще большей степени это происходит в случае применения разложения Холецкого для разреженных матриц.

Поэтому для эффективной реализации решения задач с плотными симметричными положительно определенными матрицами следует использовать не классический алгоритм разложения Холецкого, а его давно известную модификацию без извлечения квадратных корней – разложение матрицы в произведение LDL^T (версия компактной схемы Гаусса для LU-разложения). Собственно, в ряде современных публикаций это уже было предложено [15].

2.6. СУЩЕСТВУЮЩИЕ РЕАЛИЗАЦИИ АЛГОРИТМА

Точечный вариант разложения Холецкого реализован как в основных библиотеках отечественных организаций, так и в западных пакетах LINPACK, LAPACK, SCALAPACK и др.

При этом в отечественных реализациях обычно выполнены стандартные требования к методу с точки зрения ошибок округления, то есть реализован режим накопления, и обычно нет лишних операций. Правда, анахронизмом в наше время выглядит то, что ряд старых реализаций использует для экономии памяти упаковку матриц A и L в одномерный массив. При реальных вычислениях на современных вычислительных системах данная упаковка только создает дополнительные накладные расходы. Однако реализации, не использующие такую упаковку, вполне отвечают требованиям современности в отношении вычислительной точности метода.

Реализация точечного варианта разложения Холецкого в современных западных пакетах обычно обладает другими недостатками, имеющих один источник: все эти реализации, по сути, происходят из одной и той же в LINPACK, а та использует пакет BLAS. Основная слабость – даже не наличие лишних операций, о котором уже упоминалось, а то, что в используемых BLAS скалярное произведение реализовано без режима накопления. Это перечеркивает имеющиеся для разложения Холецкого наименьшие среди разложений оценки эквивалентного возмущения, поскольку они выведены как раз в предположении использования режима накопления при вычислении скалярных произведений. Поэтому тот, кто использует реализации разложения Холецкого без режима накопления, серьезно рискует не получить желаемую вычис-

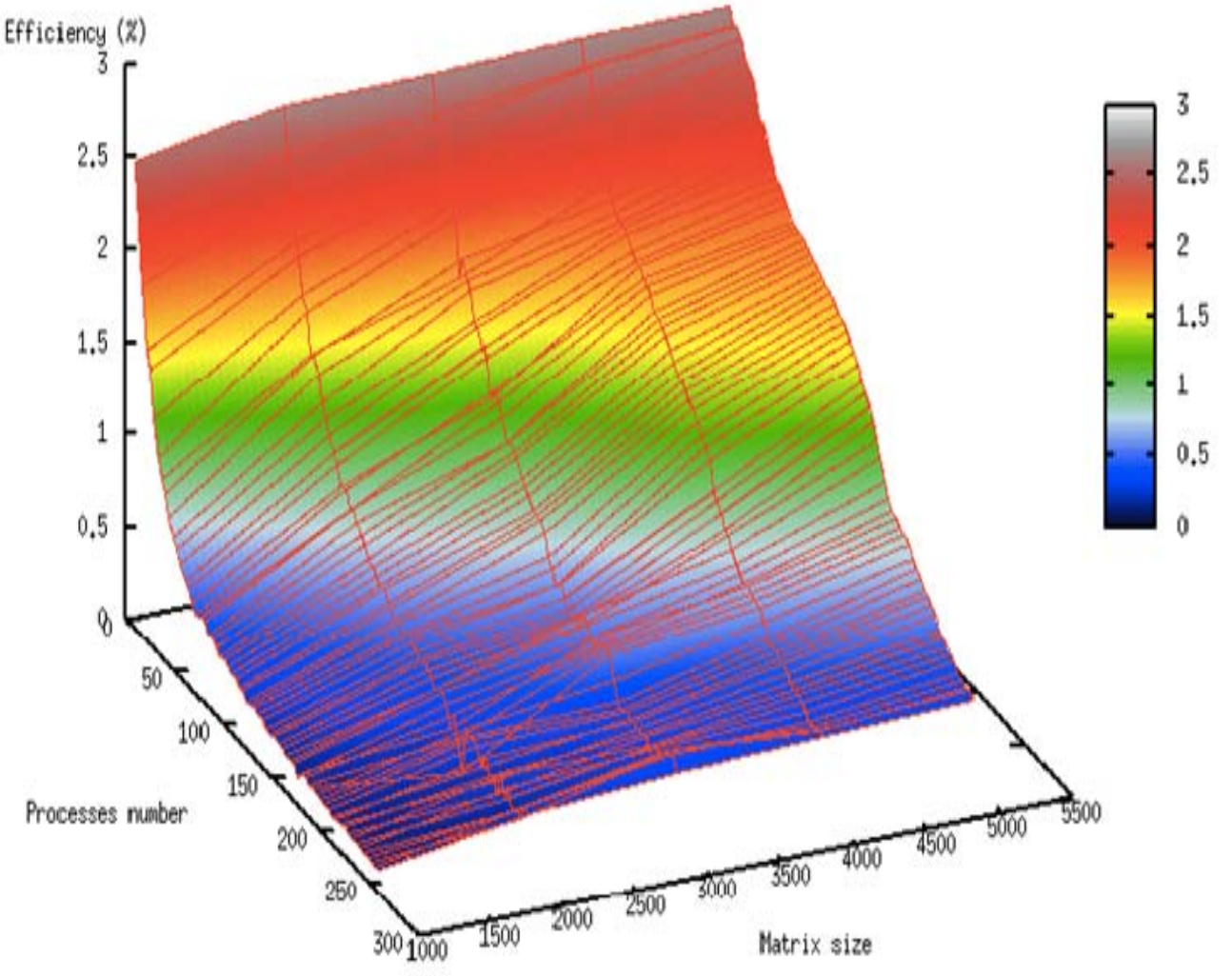

Рис. 9. Эффективность параллельной реализации разложения Холецкого в зависимости от числа процессоров (ось налево) и размерности матрицы (ось направо)

лительную точность, либо ему придется использовать подпрограммы двойной точности.

В крупнейших библиотеках алгоритмов до сих пор предлагается именно разложение Холецкого, а более быстрый алгоритм LU-разложения без извлечения квадратных корней используется только в особых случаях (например, для трехдиагональных матриц), в которых количество диагональных элементов уже сравнимо с количеством внедиагональных.

ВЫВОДЫ

В настоящей работе на примере разложения Холецкого – одного из наиболее популярных прямых методов – опробована методика всестороннего анализа свойств алгоритмов на основе анализа графа вычислений, оценки локальности, свойств ярусно-параллельной формы и исследования эффективности конкретной реализации, что позволяет сделать следующие выводы.

Как и для большинства известных алгоритмов, узкими местами метода являются не только пересылки между устройствами вычислительной системы, но и работа на отдельном узле. Поэтому необходимо изучение масштабируемости и локальности вычислений и данных.

Популярность метода, оправданная для систем с очень маленькой памятью, сейчас устарела: для решения тех же задач вместо разложения Холецкого следует использовать компактную схему LU-разложения (в версии LDL^T– разложения с учетом симметрии матрицы).

На настоящий момент, пока в пакетах программ это дополнение не произведено и для решения задач используется разложение Холецкого, во избежание простоев оборудования следует правильно выбирать количество используемых устройств суперкомпьютера.

СПИСОК ЛИТЕРАТУРЫ

1. **Commandant Benoit**. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues (Procédé du Commandant Cholesky) // Bulletin Géodésique 2 (1924), 67-77. [Commandant Benoit. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues (Procédé du Commandant Cholesky) (in French) // Bulletin Géodésique 2 (1924), pp. 67-77]

2. **Banachiewicz T.** Principles d'une nouvelle technique de la méthode des moindres carrês // Bull. Intern. Acad. Polon. Sci. A., 1938, 134-135 [T. Banachiewicz, Principles d'une nouvelle technique de la méthode des moindres carrês, (in French), Bull. Intern. Acad. Polon. Sci. A., 1938, pp. 134-135]

3. **Banachiewicz T.** Méthode de résoltution numérique des équations linéaires, du calcul des déterminants et des inverses et de réduction des formes quardatiques // Bull. Intern. Acad. Polon. Sci. A., 1938, 393-401. [T. Banachiewicz, Méthode de résoltution numérique des équations linéaires, du calcul des déterminants et des inverses et de réduction des formes quardatiques, (in French),// Bull. Intern. Acad. Polon. Sci. A., 1938, 393-401]

4. **Воеводин В. В.** Вычислительные основы линейной алгебры. М.: Наука, 1977. 304 с. [V. V. Voevodin, Computational base of linear algebra, (In Russian). Moscow, Nauka, 1977]

5. **Воеводин В. В., Кузнецов Ю. А.** Матрицы и вычисления. М.: Наука, 1984. 320 с. [V. V. Voevodin, Yu. A. Kuznetzov, Matrices and Computations, (In Russian). Moscow, Nauka, 1984]

6. **Фаддеев Д. К., Фаддева В. Н.** Вычислительные основы линейной алгебры. М.-Л.: Физматгиз, 1963. 735 с. [D. K. Faddeev, V. N. Faddeva, Computational base of linear algebra, (In Russian). Moscow, Fizmatgiz, 1963.]

7. **Kaporin I.E.** High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition // Numer. Lin. Algebra Appl. (1998) Vol. 5, No. 6, 483-509. [I.E.Kaporin, High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition // Numer. Lin. Algebra Appl. (1998) vol. 5, no. 6, pp. 483-509.]

8. **Капорин И.Е., Коньшин И.Н.** Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Ж. вычисл. матем. и матем. физ., 2001, Т, 41, N. 4, С. 515–528. [I.E.Kaporin, I.N.Konshin, Parallel solving of symmetric positive definite linear algebraic systems on the block overlapping base, (In Russian) // J. Vychisl. Matem. I Matem. Phys. (2001) vol.41, no.4, pp. 515-528]

9. **Воеводин Вл., Жуматий С., Соболев С., Антонов А., Брызгалов П., Никитенко Д., Стефанов К., Воеводин Вад.** Практика суперкомпьютера «Ломоносов» // Открытые системы, 2012, N 7, С. 36-39. [Vl.Voevodin, S Zhumatiy,S.Sobolev,A.Antonov,P.Bryzgalov,D.Nikitenko,K.Stefanov,Vad.Voevodin, Lomonosov Supercomputer Practice, (In Russian) // Otkrytye Sistemy, 2012, no.7, pp. 36-39]

10. Открытая энциклопедия свойств алгоритмов [Электронный ресурс]. URL: http://algowiki-project.org (дата обращения 07.04.2015) [Open Encyclopedia of Algorithms' Properties [online]. Available: http://algowiki-project.org]

11. **Фролов А. В.** Принципы построения и описание языка Сигма. Препринт ОВМ АН N 236. М.: ОВМ АН СССР, 1989, 26 с.[A. V. Frolov, Design principles and description of Sigma language. (In Russian) Preprint 236. Moscow: OVM AN SSSR, 1989]

12. **Воеводин В. В.** Математические основы параллельных вычислений. М.: Изд. Моск. ун-та, 1991. 345 с. [V. V. Voevodin, Parallel Computatation mathematical bases, (In Russian), Moscow: MGU, 1991]

13. **Воеводин Вад. В.** Визуализация и анализ профиля обращений в память // Вестник Южно-Уральского государственного университета. Серия Математическое моделирование и программирование. 2011. Т.17, №234. с.76–84. [Vad.V.Voevodin, Memory access profile analysis and visualization // Vestnik YuUGU, Matematicheskoe modelirovanie I programmirovanie (2011) Vol.17, No.234, 76-84]

14. **Воеводин Вл. В., Воеводин Вад. В.** Спасительная локальность суперкомпьютеров // Открытые системы. 2013. № 9. С. 12–15. [Vl.V.Voevodin, Vad.V.Voevodin, Helpful locality of supercomputers // Otkrytye Sistemy, 2013, no.9, pp. 12-15]

15. **Krishnamoorthy A., Menon D.** Matrix Inversion Using Cholesky Decomposition. 2013. eprint arXiv:1111.4144 [Электронный ресурс]. URL:http://adsabs.harvard.edu/cgi-bin/bib_query?arXiv:1111.4144 (дата обращения 07.12.2014) [A. Krishnamoorthy, D. Menon, Matrix Inversion Using Cholesky Decomposition. 2013 [online]. Available: http://adsabs.harvard.edu/cgi-bin/bib_query?arXiv:1111.4144]

ОБ АВТОРАХ

ФРОЛОВ Алексей Вячеславович, ст. науч. сотр. ИВМ РАН. Дипл. инж.-физ., автоматика и электроника (МФТИ, 1987). Канд. физ.-мат. наук по выч. мат. и мат. и программному обеспечению выч. машин, комплексов, сист. и сетей (ИВМ РАН, 1990). Доц. по каф. теории вероятностей и мат. статистики (РУДН, 1997). Линейная алгебра, параллельные выч., теор. распарал., методики распарал., практическое распарал. алгоритмов.

ВОЕВОДИН Вадим Владимирович, науч. сотр. лаб. параллельных инф. технол. НИВЦ МГУ им. М. В. Ломоносова. Дипл. мат-к, с. программист (МГУ им. М. В. Ломоносова, 2003). Канд. физ.-мат. наук (Иссл. задачи отображения программ и алгоритмов на архитектуру). Иссл. в обл. эффективности программ, локальности данных.

КОНЬШИН Игорь Николаевич, науч. сотр. ВЦ РАН, ст. науч. сотр. ИВМ РАН. Дипл. инж.-физ., аэродинамика и термодинамика (МФТИ, 1985). Канд. физ.-мат. наук по выч. мат. (ВЦ РАН, 2009). Выч. мат., линейная алгебра, парал. выч.

ТЕПЛОВ Алексей Михайлович, мл. науч. сотр. лаб. параллельных инф. технол. НИВЦ МГУ им. М. В. Ломоносова. Дипл. прикл. мат-к, сист. программист, прикл. мат. и информатика (МГУ им. М. В. Ломоносова, 2011). Масштабируемость, эффективность парал. программ, инструменты анализа парал. программ.

METADATA

Title: Study of structural properties of Cholesky decomposition: from well-known facts to new conclusions.

Authors: A. V. Frolov[1], Vad. V. Voevodin[2], I. N. Konshin[1,3], A. M. Teplov[2]

Affiliation:

[1] Institute of Numerical Mathematics of Russ. Acad. Sci. (INM RAS), Moscow, Russia.

[2] Research Computing Center of Moscow State University (RCC MSU), Moscow, Russia.

[3] Dorodnicyn Computing Centre (CC RAS), Moscow, Russia.

Email: [1] frolov@mail.inm.ras.ru.

Language: Russian.

Source: Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 19, no. 4 (70), pp. 149-162, 2015. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Abstract: As a part of Open Encyclopedia of Algorithms' Properties (AlgoWiki) development the properties of parallel algorithm of Cholesky decomposition for symmetric positive definite matrices has been studied. Based on algorithm graph analysis a description of this algorithm parallel properties is given as well as locality properties of memory access are analyzed. Resulted from numerical experiments on Lomonosov supercomputer research data of a scalability for Cholesky decomposition straightforward implementations allows to give several suggestions as regards dimension ratio of tasks and resources required for their solution. Theoretical studies give an opportunity both to explain relatively low performance of this method pointwise implementation and to reassess some approaches accepted by the scientific society more than half a century ago as well as to re-evaluate them from the contemporary computer facilities point of view.

Key words: Parallel algorithm structure; Cholesky decomposition; mapping of algorithm to computer architecture.

About authors:

FROLOV, Alexey Vyacheslavovich, Senior Researcher, INM RAS. Dipl. Engineer-Physicist, Automation and Electronics (MIPT, 1987). Cand. of Phys.&Math. Sci. (INM RAS, 1990).

VOEVODIN, Vadim Vladimirovich, research associate in Parallel information technologies laboratory. Specialist in mathematics and system programming (MSU, 2003).

KONSHIN, Igor Nikolaevich, Researcher, CC RAS, Senior Researcher, INM RAS. Dipl. Engineer-Physicist, Aerodynamics and Thermodynamics (MIPT, 1985). PhD in Computational Mathematics (CC RAS, 2009). Computational mathematics, linear algebra, parallel computing.

TEPLOV, Alexey Mikhailovich, Jr. Researcher LPIT RCC MSU. Dipl. Specialist in Mathematics and System Programming, Applied Mathematics and Computer Science (Moscow State Univ., 1995).