Сопоставление разных методов решения одной задачи по методике проекта Algowiki*

А.В. Фролов[1], А.С. Антонов[2], Вл.В. Воеводин[2], А.М. Теплов[2]
ИВМ РАН[1], НИВЦ МГУ[2]

Авторы, в первую очередь на примере задачи решения трехдиагональных систем линейных уравнений, демонстрируют методику исследования параллельных свойств алгоритмов проекта Algowiki, предназначенных для решения одной задачи, и их сопоставления друг с другом. Исследование показывает, что при наполнении Открытой энциклопедии свойств алгоритмов авторам приходится изучать не только свойства самих алгоритмов, но и, в определенной степени, сложившееся к ним в вычислительном сообществе отношение. Результаты порой неожиданны.

Ключевые слова: Algowiki, параллельная сложность, последовательная сложность, параллелизм алгоритмов, алгоритмы решения трехдиагональных СЛАУ.

1. Введение

Открытая энциклопедия свойств алгоритмов[1,9], создаваемая в настоящее время, предполагает, что алгоритмы решения различных задач будут не просто проанализированы по своим структурным и вычислительным свойствам, но и, в случае если они решают одинаковые или сходные задачи, сопоставлены с "конкурентами". В настоящей работе авторы попытались изложить то, как у них получается при работе над Открытой энциклопедией свойств алгоритмов такое сопоставление. В предыдущих работах, кроме анонса самой энциклопедии, авторами уже разбирались некоторые выводы — например, полученные ими при анализе некоторых известных алгоритмов, в частности, метода Холецкого [16]. Однако ряд особенностей алгоритмов, видных лишь при сопоставлении разных методов решения одной и той же задачи, стал виден только по мере углубления работ над созданием Открытой энциклопедии свойств алгоритмов. Их авторы и хоте ли бы выделить и обрисовать в данной работе. В основном это сопоставление касается алгоритмов решения одной задачи — решения трехдиагональных систем линейных алгебраических уравнений[1] [8], но есть интересные результаты и сопоставления других алгоритмов.

В частности, оказалось, что сами по себе описания свойств алгоритмов в Открытой энциклопедии свойств алгоритмов необходимо включает в себя не только сами алгоритмы и их структуру, но и, в силу отсылки к известным их реализациям, взгляд вычислительного сообщества на их возможности, со всеми его ошибками и другими недостатками. Об этом также будет сказано в данной статье.

2. Сопоставление алгоритмов решения трехдиагональных СЛАУ

В данном разделе мы рассмотрим результаты сопоставления наиболее популярных, с точки зрения авторов, алгоритмов решения трехдиагональных СЛАУ. При этом авторы никоим образом не претендуют на полноту сопоставления — в противном случае в Открытой энциклопедии свойств алгоритмов уже давно были бы полностью описаны все применяемые для решения этой задачи методы. Речь пойдет о тех выводах, которые можно сделать по первым результатам исследования свойств выбранных алгоритмов.

* Результаты, приведенные в разделах 2.1-2.5,3.1, получены в Московском государственном университете имени М.В. Ломоносова за счет гранта Российского научного фонда (проект №14-11-00190). Работа выполнена с использованием ресурсов суперкомпьютерного комплекса МГУ им. М.В. Ломоносова [7].
[1] далее — СЛАУ

Возможно, кого-то удивит выбор в качестве задачи решение трехдиагональной СЛАУ — ведь сама по себе задача имеет довольно слабую арифметическую мощность: при $4n-2$ входных и n выходных данных самые простые алгоритмы имеют чуть больше требуемых арифметических операций. Однако, во-первых, тем и интереснее возможные подходы к преодолению "узких мест", в во-вторых, часть изучаемых методов может пригодиться, после адаптации, для решения блочно-трехдиагональных СЛАУ, где мощность операций может быть несколько (а может быть даже существенно) больше, чем в точечном случае.

2.1. Список алгоритмов

Коротко перечислим алгоритмы, которые полностью или частично были изучены и сопоставлены в плане решения трехдиагональных СЛАУ, и объясним, почему именно они попали в список.

Классическая *монотонная правая прогонка* [10] — самый первый из методов, которые вообще применяли к решению трехдиагональных СЛАУ, очевидный вариант метода исключения неизвестных. Все остальные алгоритмы решения авторы статей сравнивают, как правило, с этим первым, а также с тем, который сами считают самым быстрым из существующих и использующихся.

Естественно, что, полностью идентичная первому алгоритму по структуре, монотонная *левая прогонка* в число отдельно описанных не включалась.

Следующим был исследован *алгоритм Стоуна*[1] [12,19,20] — как алгоритм, в режиме точных вычислений эквивалентный монотонной прогонке в одной из ее версий (основанный на том же разложении трехдиагональной матрицы в произведение двух двудиагональных) и зачастую служащий на лекциях по параллельным вычислениям иллюстрацией "как распараллелить прогонку" и там же — поводом к сожалениям о том, что "так быстро, но неустойчиво".

Естественно, что изучение коснулось и метода *встречных прогонок,* чьи формулы почти являются комбинацией двух монотонных (правой и левой) прогонок и который в сравнении с ними дает выигрыш в 2 раза по критическому пути графа.

Судя по встречающимся в литературе публикациям, наиболее часто для параллельного решения трехдиагональных СЛАУ используется метод *циклической редукции* [8,20], который, как и схема Стоуна, имеет логарифмический критический путь. Циклическая редукция сама по себе является крайним случаем *редукции* простой, описанной в [8]. Поэтому вместе с самым популярным из методов было изучено и его "основание". Впрочем, первые результаты этого изучения простой редукции одним из авторов настоящей статьи вынесены в отдельную работу.

В числе других методов решения трехдиагональных СЛАУ, побывавших в поле зрения авторов, также и *последовательно-параллельный метод* [14,15], опирающийся на те же методы распараллеливания, что и схема Стоуна, и использующий то же самое разложение матрицы СЛАУ, что и последняя, и монотонная прогонка.

Кроме этого, можно упомянуть и метод окаймления, и метод дихотомии.

Резюмируя перечень исследованных методов, еще раз заметим, что не все они даже имеют краткое описание в Открытой энциклопедии свойств алгоритмов — по естественным причинам. Теперь перейдем к описанию того, какие характеристики алгоритмов прежде всего привлекали наше внимание.

2.2. Общий объем операций и обрабатываемых данных

Недавно на конференции "Суперкомпьютерные дни в России" Дж.Донгарра высказал справедливое мнение о том, что меньшее количество операций в алгоритме не всегда отражает его способность к более быстрому исполнению на суперкомпьютере. Этот тезис особенно ярко подтверждается именно при решении трехдиагональных СЛАУ. Среди всех используемых алгоритмов самое маленькое количество операций — именно у самого медленного последовательного алгоритма классической монотонной прогонки. Однако общий объем продолжает оставаться важной характеристикой алгоритма: при всех фиктивных "ускорениях" реальный

[1] в [12] — схема Стона.

выигрыш следует считать не в сравнении с однопроцессорной версией этого же метода, а с однопроцессорной версией метода с самым маленьким количеством операций, в данном конкретном случае — с монотонной прогонкой.

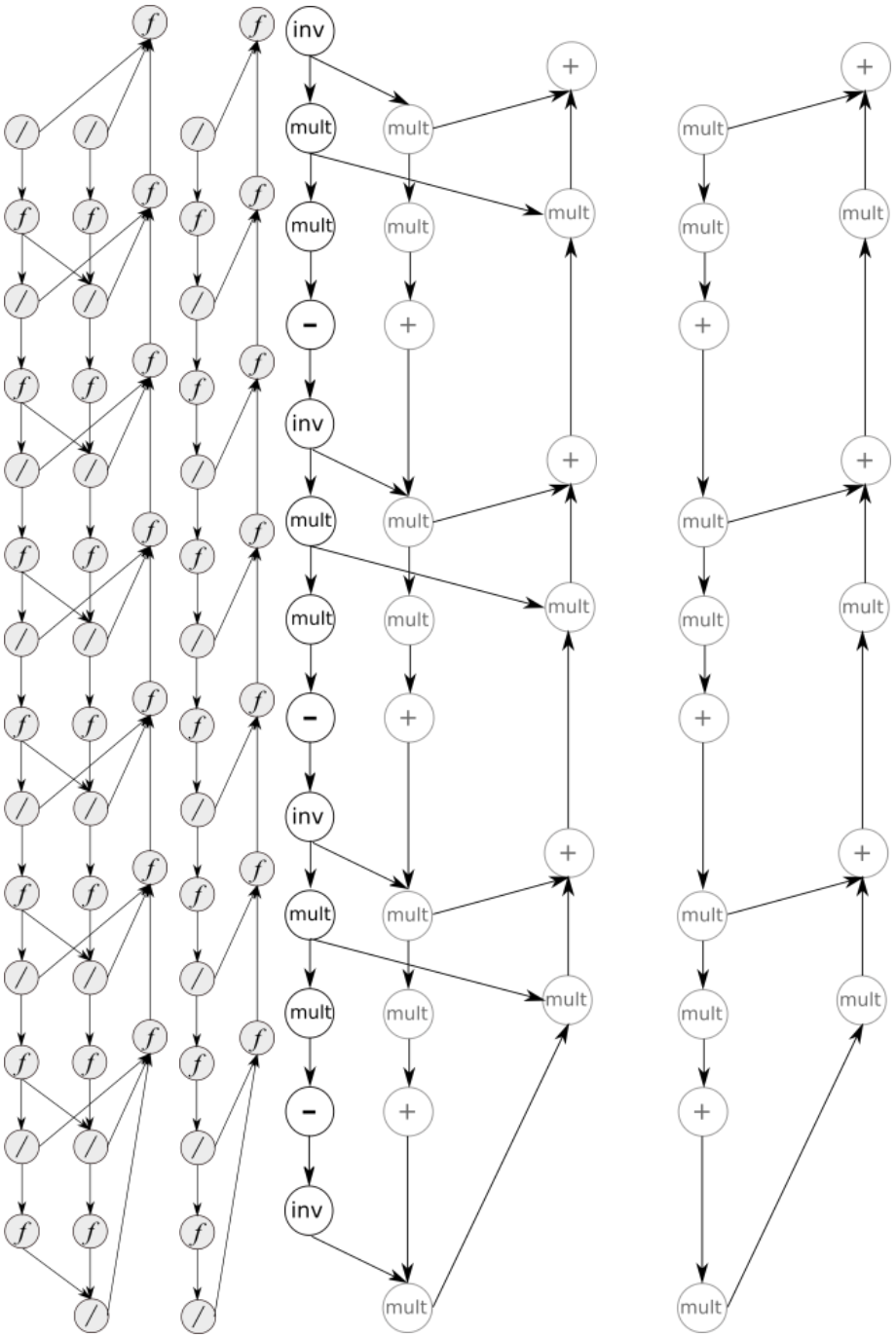

Рис. 1. Граф прогонки, повторной прогонки, а также их более детальные варианты с заменой деления на вычисление обратного числа и умножения.

Для рассматриваемой задачи ситуация по количеству требуемых вычислений такова, что только в алгоритме Стоуна, который и так по своим характеристикам неустойчив, количество арифметических операций не линейное (как у остальных алгоритмов) по размеру СЛАУ, а линейное с логарифмическим множителем. Собственно, такие методы, как алгоритм Стоуна, в реальности, даже если отвлечься от его неустойчивости, дают, да и в принципе могут дать лишь очень низкую реальную эффективность, потолок которой ограничен убывающей функцией $O(log_2^{-1}N)$. К потолкам реальной эффективности алгоритма мы еще вернемся ниже.

Общий объем обрабатываемых данных алгоритма неизбежно должен быть соотнесен с количеством арифметических операций. Собственно, для задачи решения трехдиагональной СЛАУ то, что объем этих данных по порядку почти равен объему вычислений, показывает, что от всех без исключения алгоритмов решения задачи не приходится ждать хотя бы сносной эффективности. Исключением может оказаться вариант, когда элементы матрицы и правой части СЛАУ вычисляются прикладным алгоритмом непосредственно там, где их будут использовать в качестве данных.

2.3. Особенность алгоритма - важность повторного использования

Особо отметим, что при изучении алгоритмов, решающих одну и ту же задачу, зачастую следует учитывать особенности их применения. В случае с трехдиагональными СЛАУ такой особенностью является то, что зачастую прикладникам нужно решение многих СЛАУ с разными правыми частями, но одной и той же матрицей.

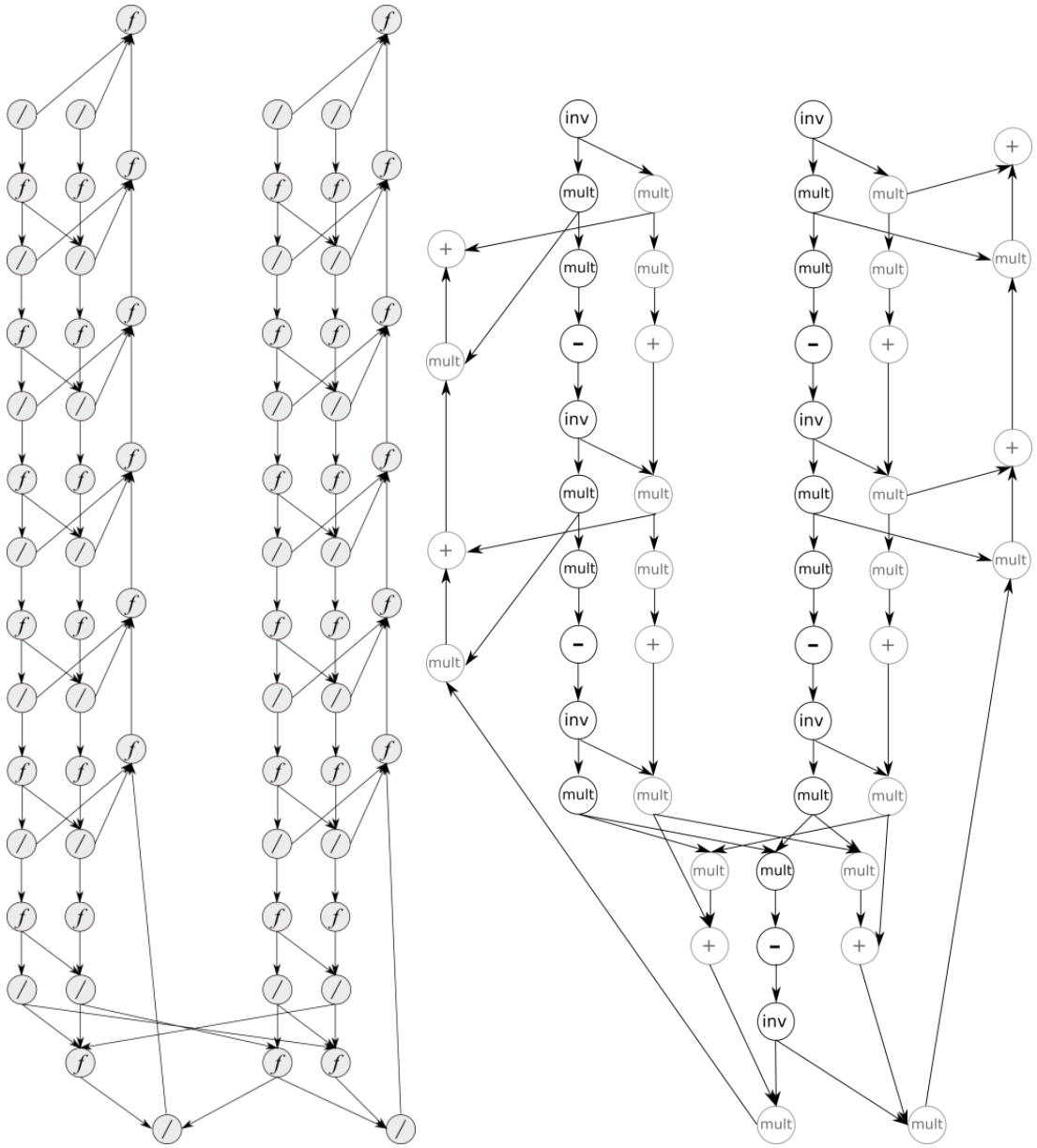

Рис. 2. Граф встречной прогонки и более детально выписанный вариант с заменой деления на вычисление обратного числа и умножения.

Естественно, что случай, когда все правые части доступны сразу, неинтересен, поскольку тут параллелизм будет не внутри алгоритма решения, а "внешний". Однако довольно часто правые части "поступают" для решателя последовательно. Поэтому для таких случаев интересно, как алгоритм решает новые СЛАУ с использованием расчетов от самого первого решения — тех расчетов, что связаны не с правой частью, а с матрицей. В Открытой энциклопедии свойств алгоритмов для такого случая описываются отдельно алгоритмы повторного решения.

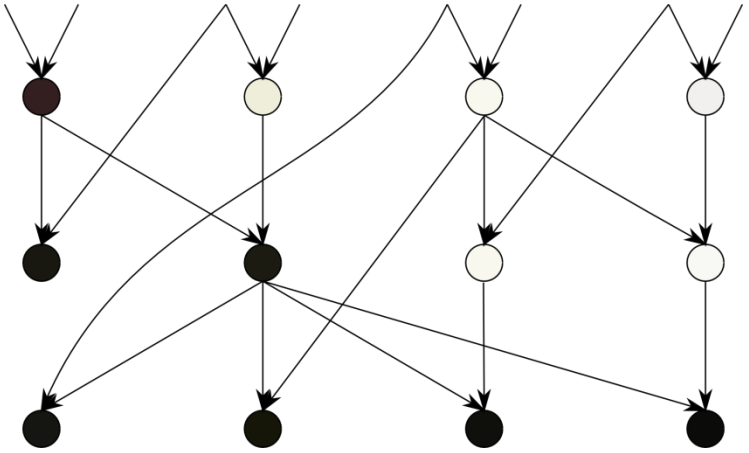

Рис. 3. Граф всех трех ступеней схемы Стоуна (вершины означают разные операции для разных ступеней).

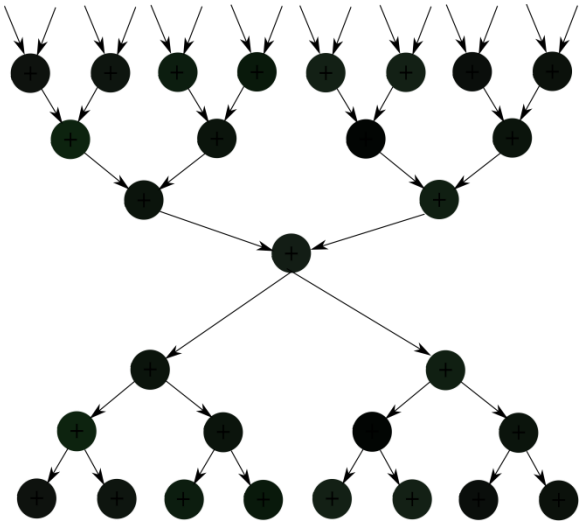

Рис. 4. Граф циклической редукции (вершины означают разные операции для первого и для последующих решений СЛАУ).

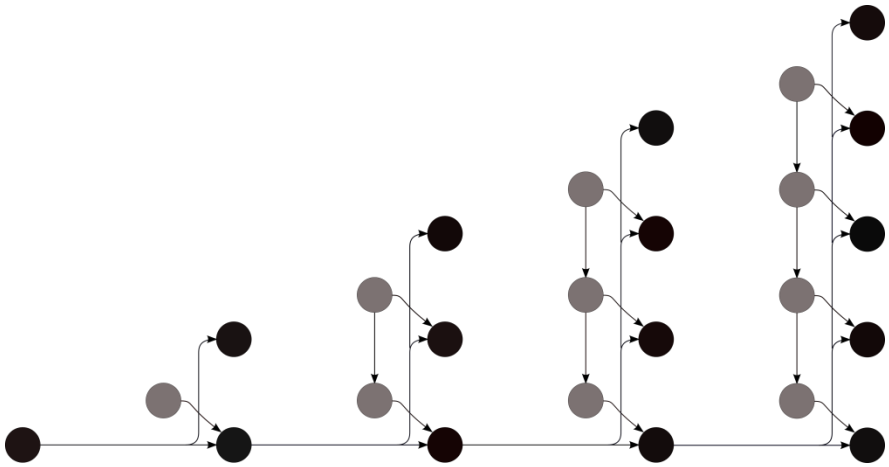

Рис. 5. Граф последовательно-параллельного метода.

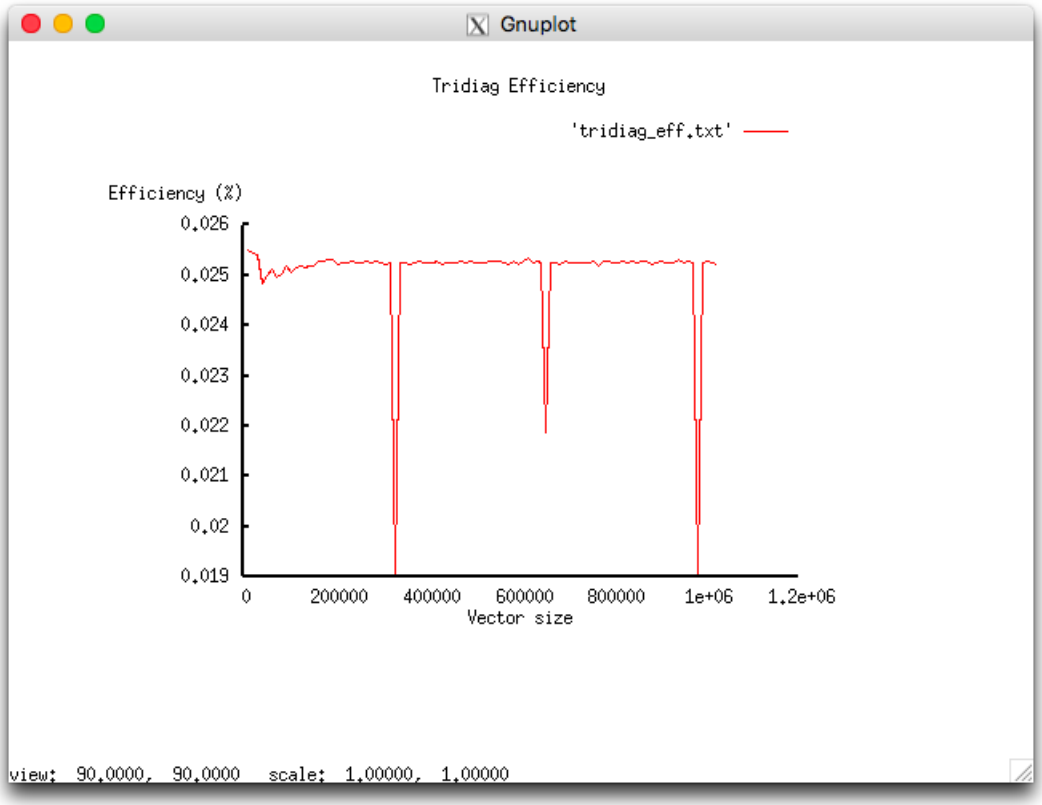

Рис. 6. График эффективности прогонки в зависимости от размера задачи.

Среди выбранных алгоритмов по этому признаку можно выделить две группы. В одной из них обработку данных, связанную с матрицей или с правой частью, можно разделить даже на отдельные алгоритмы. Скажем, в монотонной прогонке, методе Стоуна или последовательно-параллельном методе, как основанных на LU-разложении, это разделение получается естественно: вот — разложение, а вот — алгоритмы решения двухдиагональных СЛАУ. В другой группе такого естественного деления нет, поэтому для повторного решения СЛАУ приходится просто вычеркивать те вычисления, что уже проделаны. В ряде случаев для оптимизации повторных решений СЛАУ можно чуть "утяжелить" первое решение: например, для прогонок вместо выполнения деления на одно и то же выражение, зависящее только от матрицы, логично сразу на первом этапе вычислить обратное этому выражению, а при повторных решениях умножать на него. Аналогичные примеры есть и в других алгоритмах.

Бывает так, что повторные решения настолько важны прикладнику, что он идет на неоптимальность первого решения, лишь бы упростить схемы распараллеливания следующих.

2.4. Информационный граф и его характеристики

При сравнении информационных графов разных алгоритмов решения трехдиагональной СЛАУ получилась естественная классификация по порядку длины их критического пути (приведем только наиболее распространенные методы):

- линейный: все разновидности прогонки
- логарифмический: циклическая редукция, схема Стоуна
- "корень квадратный": простая редукция, последовательно-параллельный метод, метод окаймления.

Рассмотрим сначала "крайности". Самая простая монотонная прогонка в варианте без замены деления и с заменой представлена на Рис. 1. Хорошо видны и практически безнадежная для распараллеливания структура, и то, что замена деления избавляет от делений повторную прогонку. Похожий граф алгоритма и у другой версии прогонки - встречной (см. Рис.2, в нем мы уже не показываем граф повторной встречной прогонки). Он как бы получен из двух графов монотонной прогонки, соединенных "переходником" в месте поворота дуг графа алгоритма снизу вверх.

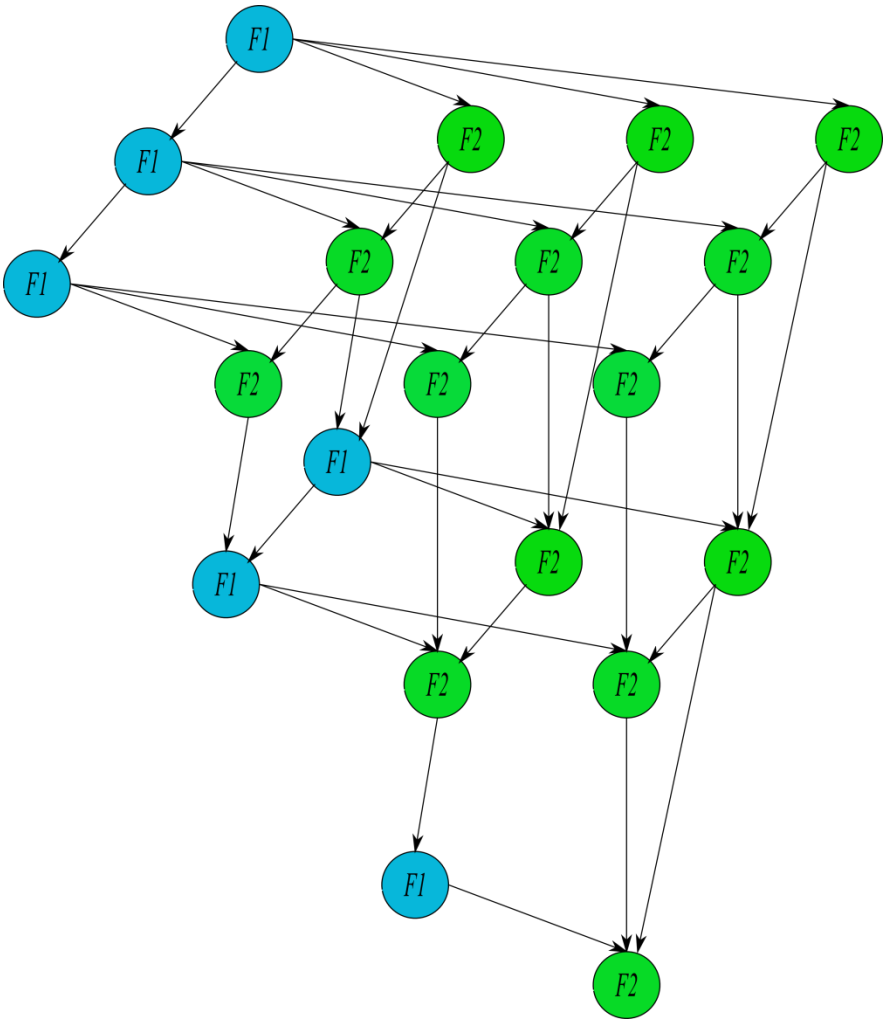

Рис. 7. Граф *QR*-разложения методом Гивенса при n=4.

Совсем другой граф алгоритма у схемы Стоуна (см. Рис.3). Как уже отмечалось в [14,15], кроме своей неустойчивости (на первой ступени) этот алгоритм еще и обладает большим количеством избыточных операций, что делает довольно проблематичной эффективное использование даже его устойчивой второй части.

Казалось бы, циклическая редукция (см. Рис. 4) с ее небольшой избыточностью лишена этого недостатка. Однако если мы внимательно взглянем на ширину ярусно-параллельной

формы[1] (равную n/2), то увидим, что такую ширину во всей ЯПФ имеют только первый и последний ярусы. Поэтому, несмотря на небольшую избыточность вычислений, ее эффективное использование также затруднено естественными ограничениями эффективности сверху порядка $O(log_2^{-1}N)$.

В связи с выводами по графам предыдущих алгоритмов сравнительно хорошими получаются небольшая избыточность и сбалансированность последовательно-параллельных алгоритмов, у которых критический путь пропорционален корню квадратному размера СЛАУ (например, Рис.5). Хотя вычислительные характеристики уже описанного в [14] оказались не так хороши по устойчивости, однако исследование графов алгоритмов показало, что последовательно-параллельное исполнение метода простой (нециклической) редукции [8] может быть одним из самых эффективных, и при этом у него давно исследованы характеристики устойчивости, практически общие с обычными прогонками.

2.5. Локальность вычислений алгоритма

При изучении алгоритмов одним из важных параметров является локальность вычислений. Как видно из структуры графов алгоритма, наиболее хорошая локальность по пространству наблюдается у прогонки во всех ее вариантах. Однако при такой хорошей локальности по пространству она не очень локальна по времени. Действительно, как только длина загруженных данных превышает размеры кэша, прогонка просто должна притормаживать, поскольку начинаются "промахи кэша". Вместе с этой "Сциллой переполнения размера кэша" в прогонках, к сожалению, в наличии и "Харибда излишней локальности", которая заключается в том, что вычисляемые данные требуются операциям прямо сразу после вычислившей их. Это немедленное требование не позволяет современному оборудованию проявить свою суперскалярность. Это тоже должны учитывать разработчики программ, пользующихся прогонкой. В результате даже на однопроцессорных узлах прогонка обычно дает мизерную эффективность (см. Рис.6).

3. Другие задачи

Помимо рассмотренных нами алгоритмов решения трехдиагональных СЛАУ, еще ряд алгоритмов привлек наше внимание — в основном, в связи с тем, что потенциальная (по структуре алгоритма) распараллеливаемость некоторых из них совершенно не соответствует степени их распространенности. Самым интересным примером среди них является сопоставление *методов Гивенса* (в отечественной литературе — также *метод вращений*) и *Хаусхолдера* (в отечественной литературе — также *метод отражений*) для нахождения *QR*-разложения матрицы.

3.1. Графы *QR*-разложения квадратной матрицы

Изучение методов Гивенса и Хаусхолдера являлось и остается одной из важных задач при наполнении Открытой энциклопедии свойств алгоритмов. При этом один из авторов, занимающийся содержательным наполнением страниц проекта, достаточно давно изучал структуру этих алгоритмов еще в конце 80-х — начале 90-х гг. XX века, в связи с работами по отображению и записи графов алгоритмов [13]. Поэтому, руководствуясь соображением "начинать описания с более параллельного алгоритма", он начал описания методов нахождения *QR*-разложения матрицы не с метода Хаусхолдера (отражений), а с метода Гивенса (вращений).

Действительно, при подсчете критического пути графа алгоритма оказывается, что у метода Гивенса (вращений) он *линеен* по размеру матрицы, а в методе Хаусхолдера (отражений) - *квадратичен*. Это хорошо видно на рисунках 7 и 11, где изображены графы обоих алгоритмов.

В методе Гивенса для устойчивого вычисления параметров вращения используется довольно сложный, но все же скалярный, не зависящий от размеров задачи, условный алгоритм (на рисунке 7 он заменен вершинами с названием F1). Его можно, например, записать фортран-подпрограммой вида

[1] далее ЯПФ

```
       SUBROUTINE PARAMS (X, Y, C, S)
     Z = MAX (ABS(X), ABS(Y))
     IF (Z.EQ.0.) THEN
    C OR (Z.LE.OMEGA) WHERE OMEGA - COMPUTER ZERO
     C = 1.
     S = 0.
     ELSE IF (Z.EQ.ABS(X))
     R = Y/X
     RR = R*R
     RR2 = SQRT(1+RR)
     X = X*RR2
     Y = (1-RR2)/R
     C = 1./RR2
     S = -C*R
     ELSE
     R = X/Y
     RR = R*R
     RR2 = SQRT (1+RR)
     X = SIGN(Y, X)*RR
     Y = R - SIGN(RR,R)
     S = SIGN(1./RR2, R)
     C = S*R
     END IF
     RETURN
 END
```

Рис. 8. Вариант вычисления параметров поворота в методе вращений (Гивенса).

в то время, как сами вращения реализуются довольно просто, например, как

```
     SUBROUTINE ROT2D (C, S, X, Y)
     ZZ = C*X - S*Y
     Y = S*X + C*Y
     X = ZZ
     RETURN
     END
```

Рис. 9. Вариант вычисления поворота в методе вращений (Гивенса).

после чего последовательная программа метода Гивенса может быть записана, например, как

```
     DO I = 1, N-1
              DO J = I+1, N
     CALL PARAMS (A(I,I), A(J,I), C, S)
     DO K = I+1, N
     CALL ROT2D (C, S, A(I,K), A(J,K))
     END DO
     END DO
       END DO
```

Рис. 10. Вариант последовательного фортран-фрагмента, реализующего метод вращений (Гивенса).

В методе же Хаусхолдера (отражений) большинство операций графа имеют более простой вид, поскольку почти все длинные ветви в нем — скалярные произведения, а параллельные "куски" — сложение векторов "с весом". Только "узкие места" — более сложны, они-то и вычисляют окончательные параметры преобразований отражения.

После того, как содержательное наполнение страницы о методе Гивенса было уже близко к середине, редакторы Открытой энциклопедии свойств алгоритмов начали искать распространенную его параллельную реализацию, чтобы посмотреть ее вычислительные характеристики при счете на суперкомпьютере. Однако это оказалось не таким быстрым делом, как предполагалось. Если в старых пакетах программ для последовательных ЭВМ фон-неймановского типа подпрограмм метода вращений довольно много, то в самых известных общедоступных пакетах параллельных программ QR-разложения методом Гивенса обнаружить не удалось.

Таким образом, авторы оказались перед вопросом: а почему, собственно, "более параллельный" (по своей структуре) алгоритм в эпоху параллельных суперкомпьютеров оказался среди пакетов параллельных программ позади "менее параллельного"? При обсуждении этого

вопроса в кругу редакторов Открытой энциклопедии свойств алгоритмов высказывались разные версии. Скажем, версия о том, что предпочтение Хаусхолдеру отдано по количеству операций (главный член количества операций в методе Гивенса в полтора раза больше, чем в методе Хаусхолдера) не согласуется с тем, что во многих "последовательных" пакетах, где количество операций важнее, чем в "параллельных", метод вращений (Гивенса) как раз есть. С этим же не согласуется версия о более сложной операции вычисления параметров вращения, которую-де сложно "распараллеливать" — эта операция давно реализована в "последовательных" пакетах, а распараллеливать нужно не ее саму, а ее вызовы в разных местах. Вопрос о вычислительной погрешности отпал после обращения к таблице методов из [2] — у методов Гивенса и Хаусхолдера одинаковая точность...

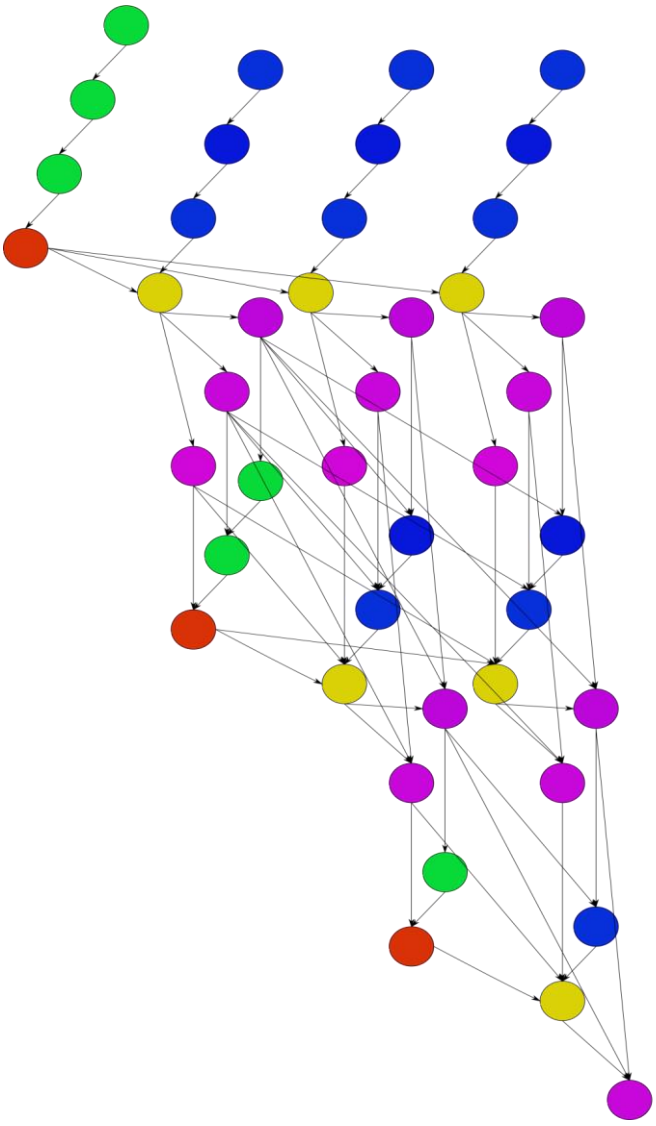

Рис. 11. Граф *QR*-разложения методом Хаусхолдера при n=4.

Однако после изучения вопроса, какого именно вида параллелизм в методе Гивенса, все стало ясно. Дело в том, что для реализации параллелизма метода вращений в полном объеме (с линейным временем счета по размеру матрицы) нужно реализовать т.н. "скошенный параллелизм", т.е. переписывать "естественные" (по размерностям матрицы) циклы в более сложную структуру. Из тех алгоритмов, что редакторами проекта были просмотрены в пакетах, нет ни одного, где бы переписали алгоритм "наискосок".

А вот метод отражений (Хаусхолдера) в этом плане для распараллеливания выгодно отличается двумя вещами — во-первых, тем, что никакого "скошенного параллелизма" в нем нет, и

во-вторых, тем, что основными массовыми операциями там являются скалярное произведение и сложение вектора с умноженным на скаляр другим вектором. Обе операции реализованы и оптимизированы в базовых пакетах, поэтому все пишут через их вызовы, из пакета в пакет, от версии к версии.

3.2. Выводы, касающиеся особенности реализации параллельного алгоритма

Как нетрудно видеть по приведенному примеру, зачастую программирующие на суперкомпьютерах продолжают предпочитать простоту компоновки возможности получения больших вычислительных скоростей. В принципе, понятно что использование так называемого «скошенного» параллелизма, как и другие сложные приемы распараллеливания, тратят время программиста, поэтому при решении нужной ему прикладной задачи он не пользуется ими, ибо суммарно он во времени не выиграет. Однако когда подобной логикой руководствуются программирующие алгоритмы для пакетов программ, такая ситуация оказывается идущей в ущерб интересам всего сообщества пользователей суперкомпьютеров. Поэтому обнаружение случаев, подобных описанному с методом Гивенса, должно вызывать работы по их устранению.

4. Заключение

В настоящей работе на примерах алгоритмом решения задач решения трехдиагональных СЛАУ и QR-разложения матриц показаны некоторые выводы, к которым приходят исследователи при анализе и сопоставлении разных алгоритмов, решающих аналогичные задачи, при их включении в Открытую энциклопедию свойств алгоритмов. В частности, отмечаются не только свойства самих алгоритмов, но и свойства "общественного мнения" вычислительного сообщества, работающего над этими алгоритмами. В целом можно констатировать, что работа над Открытой энциклопедией свойств алгоритмов стимулирует применение ряда старых известных подходов к тем областям вычислений, в которых они еще не были применены, а также открывает "белые пятна" в даже, казалось бы, изведанной области алгоритмов и параллельных вычислений.

Литература

1. Антонов А.С., Воеводин Вад.В., Воеводин Вл.В., Теплов А.М., Фролов А.В.. Первая версия Открытой энциклопедии свойств алгоритмов // Вестник УГАТУ. Серия управление, вычислительная техника и информатика. Том 19, N 2(68), 2015., С.150-159.
2. Воеводин В.В. Вычислительные основы линейной алгебры. М.: Наука, 1977.
3. Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления. М.: Наука, 1984.
4. Воеводин В.В. Математические основы параллельных вычислений// М.: Изд. Моск. ун-та, 1991. 345 с.
5. 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.
6. Воеводин Вл. В., Воеводин Вад. В. Спасительная локальность суперкомпьютеров // Открытые системы. — 2013. — № 9. — С. 12–15.
7. 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.
8. Ильин В.П., Кузнецов Ю.И. Трехдиагональные матрицы и их приложения. М.: Наука. Главная редакция физико-математической литературы, 1985г., 208 с.
9. Открытая энциклопедия свойств алгоритмов. URL: http://algowiki-project.org (дата обращения: 1.12.2015).

10. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. М.: Наука, 1978.
11. Фаддеев Д.К., Фаддеева В.Н. Вычислительные основы линейной алгебры. М.-Л.: Физматгиз, 1963.
12. Фаддеева В.Н., Фаддеев Д.К. Параллельные вычисления в линейной алгебре 1,2. // Кибернетика, 1977. №6. С. 28-40; 1982. №3. С. 18-31.
13. Фролов А.В.. Принципы построения и описание языка Сигма. Препринт ОВМ АН N 236. М.: ОВМ АН СССР, 1989.
14. Фролов А.В. Еще один метод распараллеливания прогонки с использованием ассоциативности операций // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва). – М.: Изд-во МГУ, 2015. с. 151-162.
15. Фролов А.В. Использование последовательно-параллельного метода для распараллеливания алгоритмов с ассоциативными операциями // Суперкомпьютерные дни в России: Труды международной конференции (28-29 сентября 2015 г., г. Москва). – М.: Изд-во МГУ, 2015. с. 176-184.
16. Фролов А.В., Воеводин Вад.В., Коньшин И.Н., Теплов А.М. Исследование структурных свойств алгоритма разложения Холецкого: от давно известных фактов до новых выводов // Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (31 марта - 2 апреля 2015 г., г. Екатеринбург). Челябинск: Издательский центр ЮУрГУ, 2015. С. 320-331.
17. Buneman O. A Compact Non-iterative Poisson Solver // Rep. 294, Inst. for Plasma Res., Stanford U., Stanford, Calif., 1969.
18. 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.
19. 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.
20. Stone H.S. Parallel Tridiagonal Equation Solvers // ACM Trans. on Math. Software, Vol. 1, No. 4 (Dec. 1975), P. 289-307.

One problem solving different methods' comparison according to the criteria of the Algowiki project

A.V. Frolov[1], A.S. Antonov[2], Vl.V. Voevodin[2], A.M. Teplov[2]

INM RAS[1], RCC MSU[2]

The authors demonstrate the methodology of the study the parallel properties of the algorithms Algowiki project designed to solve one problem, and their comparison with each other, primarily by the solving tridiagonal systems of linear equations. Research shows that when filling an Open encyclopedia properties of the algorithms the authors have to study not only the properties of the algorithms themselves, but also, to a certain extent, established to them in the computing community. The results are sometimes unexpected.

Keywords: Algowiki, algorithms parallelizm, tridiagonal solvers, parallel complexity, serial complexity.

References

1. Antonov A.S., Voevodin Vad.V., Voevodin Vl.V., Teplov A.M., Frolov A.V.. Pervaya versiya Otkrytoy entsiklopedii svoystv algoritmov [First Version of Algorithms' Properties' Open Encyclopedia] // Vestnik UGATU. Seriya upravlenie, vychislitel'naya tekhnika i informatika [Bulletin of UGATU. Series: Control, Compruting Technique & Informatics]. Tom 19, N 2(68), 2015., S.150-159.
2. Voevodin V.V. Vychislitel'nye osnovy lineynoy algebry [Computing Basics of Linear Algebra]. M.: Nauka, 1977.
3. Voevodin V.V., Kuznetsov Yu.A. Matritsy i vychisleniya [Matrices & Computing]. M.: Nauka, 1984.
4. Voevodin V.V. Matematicheskie osnovy parallel'nykh vychisleniy [Mathematics' Basics of Parallel Computing]. M.: Izd. Mosk. un-ta, 1991. 345 s.
5. Voevodin Vad. V. Vizualizatsiya i analiz profilya obrashcheniy v pamyat' [Memory Accesses Visualization & Profile Analysis]// Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya Matematicheskoe modelirovanie i programmirovanie [Bulletin of South Ural State University. Series: Mathematical Modeling, Programming & Computer Software]. — 2011. — T. 17, № 234. — S. 76–84.
6. Voevodin Vl. V., Voevodin Vad. V. Spasitel'naya lokal'nost' superkomp'yuterov [Helpful supercomputers' locality]// Otkrytye sistemy [Open Systems]. — 2013. — № 9. — S. 12–15.
7. Voevodin Vl., Zhumatiy S., Sobolev S., Antonov A., Bryzgalov P., Nikitenko D., Stefanov K., Voevodin Vad. Praktika superkomp'yutera «Lomonosov» [Lomonosov Supercomputer Practice] // Otkrytye sistemy [Open Systems], 2012, №7, S. 36-39.
8. Il'in V.P., Kuznetsov Yu.I. Trekhdiagonal'nye matritsy i ikh prilozheniya [Tridiagonal Matrices & Applications]. M.: Nauka. Glavnaya redaktsiya fiziko-matematicheskoy literatury, 1985g., 208 s.
9. Otkrytaya entsiklopediya svoystv algoritmov [Algorithms' Properties' Open Encyclopedia]. URL: http://algowiki-project.org (accessed: 1.12.2016).
10. Samarskiy A.A., Nikolaev E.S. Metody resheniya setochnykh uravneniy [Grid Equations Solving Methods]. M.: Nauka, 1978.
11. Faddeev D.K., Faddeeva V.N. Vychislitel'nye osnovy lineynoy algebry [Computing Basics of Linear Algebra]. M.-L.: Fizmatgiz, 1963.

12. Faddeeva V.N., Faddeev D.K. Parallel'nye vychisleniya v lineynoy algebre [Concurrent computing in Linear Algebra] 1,2. // Kibernetika [Cybernetics], 1977. №6. S. 28-40; 1982. №3. S. 18-31.
13. Frolov A.V.. Printsipy postroeniya i opisanie yazyka Sigma [Sigma Language Design Principles & Description]. Preprint OVM AN №236. M.: OVM AN SSSR, 1989.
14. Frolov A.V. Eshchye odin metod rasparallelivaniya progonki s ispol'zovaniem assotsiativnosti operatsiy [Yet another Tomas Algorithm Parallelizing Method with Operathions Associativity Using]// Superkomp'yuternye dni v Rossii: Trudy mezhdunarodnoy konferentsii (28-29 sentyabrya 2015 g., g. Moskva) [Russian Supercomputer Days: Proceedings of the International Scientific Conference (Moscow, Russia, September, 28-29, 2015]. Moscow, MSU Publishing, 2015. P. 151-162.
15. Frolov A.V. Ispol'zovanie posledovatel'no-parallel'nogo metoda dlya rasparallelivaniya algoritmov s assotsiativnymi operatsiyami [Sequential-Parallel Method's Using for Algorithms Containing Associative Operations Parallelizing]// Superkomp'yuternye dni v Rossii: Trudy mezhdunarodnoy konferentsii (28-29 sentyabrya 2015 g., g. Moskva) [Russian Supercomputer Days: Proceedings of the International Scientific Conference (Moscow, Russia, September, 28-29, 2015]. Moscow, MSU Publishing, 2015. P. 176-184.
16. Frolov A.V., Voevodin Vad.V., Kon'shin I.N., Teplov A.M. Issledovanie strukturnykh svoystv algoritma razlozheniya Kholetskogo: ot davno izvestnykh faktov do novykh vyvodov [Cholesky Algorithm Structure Properties' Investigation: from Old Facts to New Conclusions]// Parallel'nye vychislitel'nye tekhnologii (PaVT'2015): trudy mezhdunarodnoy nauchnoy konferentsii (31 marta - 2 aprelya 2015 g., g. Ekaterinburg) [Parallel Computational Technologies (PCT'2015): Proceedings of the International Scientific Conference (Ekaterinburg, Russia, March, 31 – April, 2, 2015)]]. Chelyabinsk, Publishing of the South Ural State University, 2050. P. 320–331.
17. Buneman O. A Compact Non-iterative Poisson Solver // Rep. 294, Inst. for Plasma Res., Stanford U., Stanford, Calif., 1969.
18. 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.
19. 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.
20. Stone H.S. Parallel Tridiagonal Equation Solvers // ACM Trans. on Math. Software, Vol. 1, No. 4 (Dec. 1975), P. 289-307.