AlgoWiki: изучение разных вариантов одного численного метода и другие проблемы*

А.В. Фролов[1], А.С. Антонов[2]

Федеральное государственное бюджетное учреждение науки
Институт вычислительной математики им. Г.И. Марчука Российской академии наук[1],
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет имени М.В.Ломоносова»[2]

Как уже неоднократно отмечалось в работах авторов за предыдущие несколько лет, исследования классических методов в рамках подготовки их описаний для энциклопедии AlgoWiki периодически приводят к переоценке ряда их характеристик. В данной статье показаны примеры такой переоценки, полученной при исследованиях разных вариантов одного метода. Кроме этого, описаны также некоторые смежные проблемы, касающиеся измерений производительности.

Ключевые слова: последовательная сложность, параллельная сложность, спектральные задачи, метод Якоби, измерения производительности.

1. Введение

После начала работ над AlgoWiki [1,2] ряд авторов статей этой открытой энциклопедии свойств алгоритмов не только переносит в Wiki-формат информацию, найденную в учебниках и статьях и касающуюся очередного описываемого алгоритма или метода. Для того, чтобы статьи-описания соответствовали требованиям современности, желательно также провести самостоятельное исследование того, насколько эта сумма знаний соответствует требованиям суперкомпьютерной эпохи: нельзя ли найти что-то либо новое, либо хорошо забытое старое, ставшее теперь более интересным. Авторами настоящей статьи в ряде предыдущих работ были исследованы эти либо новые, либо забытые старые аспекты [3], касающиеся решения тех же задач, для которых предназначены классические алгоритмы, и полученные с помощью соотнесения данных задач с возможным распараллеливанием алгоритмов. Эти работы касались как сравнения разных методов решения одной задачи [4], так и исследования одного метода решения определённой задачи [5].

В данной статье излагаются разные аспекты исследования разных вариантов одного метода с помощью описанных в [6] приёмов. При этом разнообразие вариантов может быть связано как с простым выбором разных алгоритмов, реализующих один метод, так и с тем, что метод способен решать разные математические задачи. В качестве примера избран метод Якоби решения спектральных задач. Кроме этого, в статье рассмотрены и некоторые возникающие при наполнении AlgoWiki проблемы, связанные с замерами производительности вычислительных систем при реализации различных методов.

Последние проблемы связаны с тем, что, если на начальных стадиях работы над AlgoWiki исследования реальных замеров не очень большого числа описываемых алгоритмов проводились сравнительно небольшим коллективом, то в настоящее время такой подход уже не удовлетворяет возросшие требования к их количеству. Поэтому исследования скорости работы вычислительных систем на конкретных алгоритмах стали уже более массовыми. С одной стороны, это можно только приветствовать, поскольку малым числом исследователей интересные результаты сравнения замеров получить трудно. С другой же стороны, возникающие при этом проблемы нуждаются в специальном их выявлении, что и начато в настоящей статье.

* Работа выполняется при поддержке РФФИ, грант 19-07-01030. Результаты получены с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.

2. Исследование и сопоставление разных вариантов метода Якоби

Метод Якоби решения спектральных задач, основанный на двусторонних вращениях симметричных матриц, давно известен математикам и потому его описание входит во все классические учебники по вычислительной линейной алгебре, например в [7,8] (в [7] он назван методом вращений). При этом хорошо известно разнообразие его вариантов ещё с времён традиционной "последовательной эпохи" компьютеров. Оно связана как с разным выбором порядка выполнения двусторонних вращений, так и с использованием вспомогательных приёмов вроде барьеров. Кроме этих способов умножения количества вариантов, метод Якоби интересен и тем, что может быть применён как для вычисления собственных чисел симметричных матриц, так и для вычисления сингулярных чисел матриц несимметричных. Выбор этого метода для сопоставления разных его вариантов обусловлен и тем, что, с точки зрения т.н. "концепции неограниченного параллелизма" [6], он хорошо распараллеливается, что отмечено и в [8]. Тем не менее, это его заявленное качество, как видно из описания наиболее распространённых пакетов и библиотек, вовсе не привело к тому, что программисты его быстро распараллелили и включили в эти пакеты. Причиной этого не может быть медленность сходимости метода в сравнении с теми спектральными методами, что включены в эти библиотеки и пакеты: ведь метод Якоби точнее их, и в [8] даже приводятся примеры, когда наиболее близкие к нулю собственные числа при их вычислении этими быстрыми методами даже меняют знак, что невозможно для метода Якоби. Поэтому исследование вариантов метода Якоби назрело и необходимо.

2.1 Общие черты схем метода Якоби для разных вариантов

Напомним читателям главную математическую идею всех вариантов метода Якоби решения проблемы собственных значений симметричной матрицы.

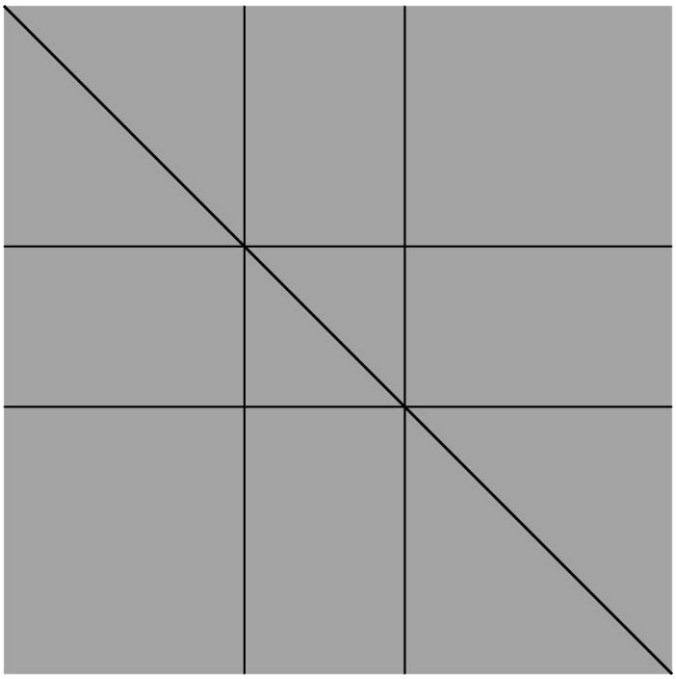

Рис. 1. Двустороннее вращение матрицы одним преобразованием вращения (Гивенса). Изменяемые элементы матрицы и диагональ.

Базовым преобразованием текущей версии матрицы во всех вариантах метода является выполнение над ней двустороннего вращения, то есть замена матрицы A на матрицу GAG^*, где G – хорошо известная вычислителям матрица Гивенса (вращений). На Рис.1 помечены чёрным,

кроме главной диагонали[1] матрицы, те элементы, что будут изменены после такой замены. При этом выбор параметра (угла) вращения в матрице Гивенса задан простым требованием: после двустороннего преобразования внедиагональные элементы на пересечении изображённых строк и столбцов обращаются в нули. Не приводя формул, которые есть в учебниках, заметим, что параметры матрицы вращения (Гивенса) однозначно определяются четырьмя (а в силу симметрии фактически тремя) элементами преобразуемой матрицы, стоящими на пересечении изменяемых строк и столбцов.

В [7] показано, что с каждым таким преобразованием сумма квадратов внедиагональных элементов уменьшается. Поэтому все варианты метода Якоби сводятся к последовательному применению базового преобразования (двустороннего вращения) к матрице, а выбор последовательностей пар строк/столбцов для таких преобразований и будет характеризовать эти варианты.

2.2 Выбор вариантов метода Якоби для сравнения

Количество вариантов метода Якоби, однако, настолько велико, что для сравнения их всех понадобилось бы очень много времени. Воспользуемся тем, что большую часть вариантов отбраковали как менее эффективные уже до нас, поэтому остановимся для сопоставления на следующих вариантах. Во-первых, нужно обязательно сравнить варианты для сингулярных и собственных чисел, как решающие разные задачи. И, во-вторых, сопоставим друг с другом наиболее распространённые циклические варианты с барьерами, при разном выборе порядка прохода полного цикла. Во всех исследованиях нужно попытаться понять, что же всё-таки помешало включить варианты метода Якоби в самые распространённые библиотеки и пакеты.

2.3 Преимущества и недостатки метода при нахождении сингулярных чисел

Напомним, в чём особенности метода Якоби в версии для нахождения сингулярных чисел. Как известно [8], его преимущество в том, что двустороннее вращение для этой задачи можно заменить односторонним.

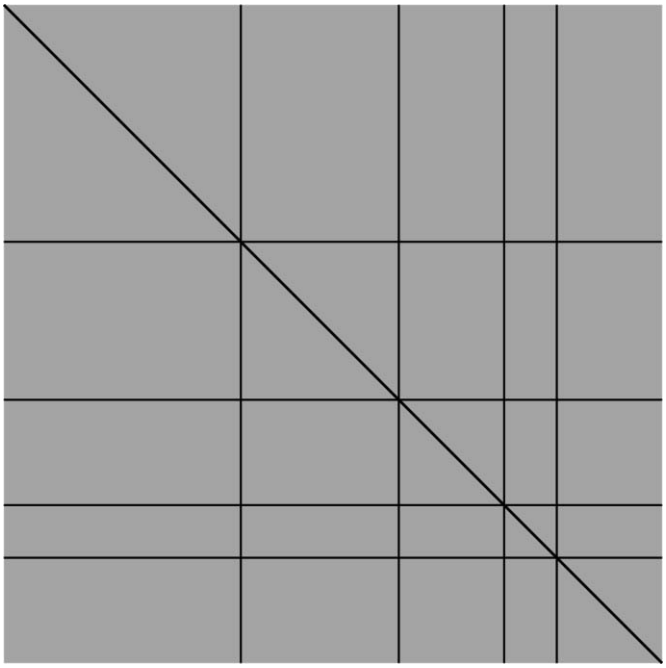

Рис. 2. Двустороннее вращение матрицы двумя преобразованиями Гивенса. Изменяемые элементы матрицы и её главная диагональ

[1] На главной диагонали при этом изменяются только два элемента, находящиеся на пересечении с отмеченными столбцами и строками; она сама же выделена только для лучшей ориентации на рисунке.

Действительно, нахождение сингулярных чисел матрицы A эквивалентно нахождению собственных значений[1] матрицы A^TA. При явном применении метода Якоби будут выполняться двусторонние вращения для матрицы A^TA, которые эквивалентны односторонним вращениям справа для матрицы A: $J^TA^TAJ = (AJ)^TAJ$, где J – произведение матриц вращения (Гивенса).

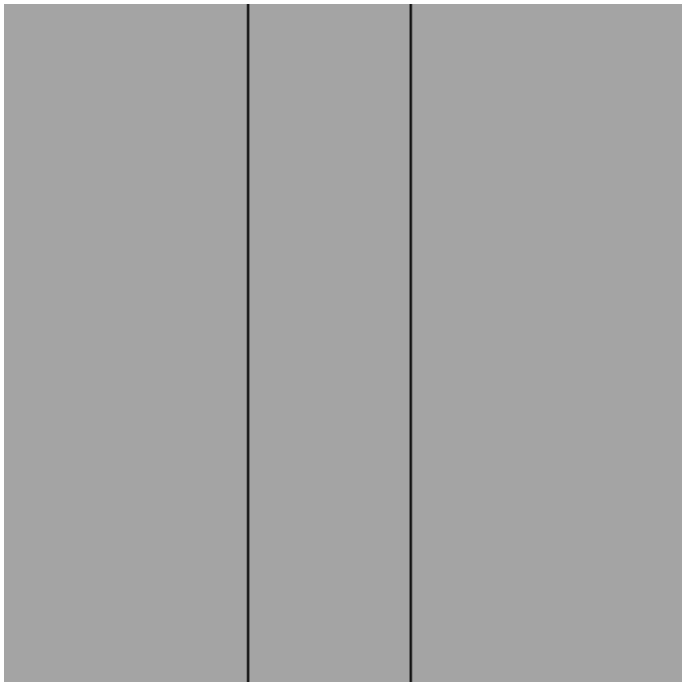

Рис. 3. Одностороннее вращение матрицы одним преобразованием Гивенса. Изменяемые элементы матрицы

На первый взгляд это даёт выигрыш неявному методу Якоби с односторонними вращениями в сравнении с явным его выполнением для матрицы A^TA. Действительно, при двусторонних вращениях на каждом шаге циклического метода мы имеем для каждого вращения картину, показанную на Рис. 1. Два любых двусторонних вращения дают картину с Рис. 2, и видно, что на элементах, не являющихся опорными для этих двух вращений, они "мешают" друг другу ещё в 4 местах. При выполнении же неявным методом односторонних вращений одно вращение даёт картину изменяемых элементов матрицы A, как на Рис. 3, а при одновременных двух вращениях – как на Рис. 4. Таким образом, при выполнении самих вращений на одном шаге циклического варианта неявного метода Якоби одностороннее вращение не требует пересылки данных, если отдельные вращения выполняются на разных процессорах вычислительной системы.

Кроме того, в [8] также указано, что при неявном выполнении метода Якоби якобы не нужно вычислять саму матрицу A^TA. Посмотрим, однако, что придётся заплатить за эти преимущества.

При явном выполнении двусторонних вращений матрицы размера $n \times n$ на каждом шаге циклического варианта метода вычисляется примерно $n/2$ матриц вращения (Гивенса)[2]. Эти матрицы вычисляются из примерно $2n$ элементов матрицы A^TA, причём для каждой из матриц количество арифметических операций конечно. Поэтому в отношении к параллельной сложности (к критическому пути графа алгоритма[3]) такое вычисление имеет примерно такой же вес, как и выполнение самих вращений.

[1] с последующим извлечением из них квадратных корней

[2] если точнее, то, конечно, не самих матриц, а характеризующих их параметров.

[3] Как и в [6], критическим путём здесь называем самый длинный из путей в графе алгоритма, где все вершины соответствуют однократному исполнению арифметической операции (для вычислительных методов обычно операции с плавающей запятой). Напомним, что длина критического пути естественным образом ограничивает снизу возможную высоту ярусно-параллельной формы графа, поэтому является

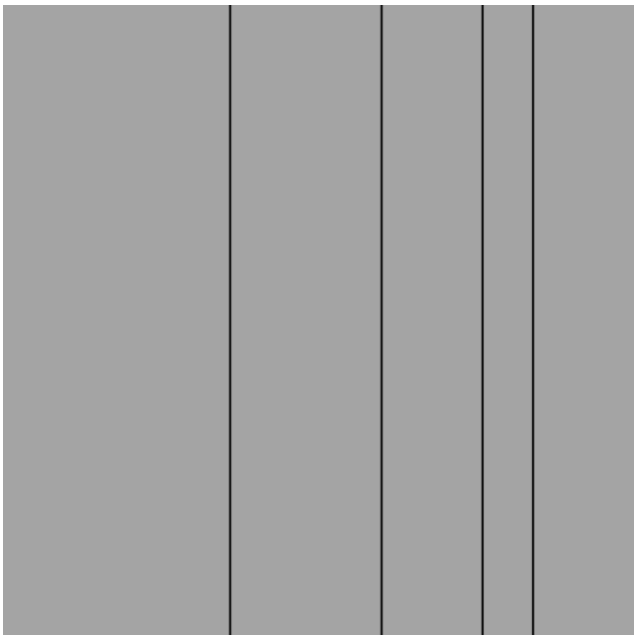

Рис. 4. Одностороннее вращение матрицы двумя преобразованиями Гивенса

А вот при выполнении неявного метода Якоби ситуация в отношении параллельной сложности изменится кардинальным образом. Ведь для вычисления параметров каждой из матриц Гивенса необходимы три элемента матрицы[1] A^TA. Если мы работаем с ней неявным образом, то эти элементы нужно вычислять после каждого шага циклического метода из элементов матрицы[2] A, то есть вычислять скалярные произведения её столбцов.

Последовательная сложность вычисления каждого скалярного произведения линейна по размеру матриц n, а параллельно их можно выполнить уж никак не быстрее, чем за $log_2 n$ шагов. При этом следует помнить, что версии алгоритмов с логарифмической параллельной сложностью обычно имеют низкую практическую эффективность из-за простаивания многих функциональных устройств вычислительной системы, что уже неоднократно отмечалось в наших предыдущих работах [1,3–5].

Если же мы сначала вычислим матрицу A^TA и будем преобразовывать её, то на каждом шаге двусторонние вращения могут быть выполнены параллельно за конечное количество шагов, особенно если вспомнить о перестановочности умножения на матрицу слева и справа и соответственно скорректировать вычисления в отношении каждого из элементов матрицы A^TA. Одновременно можно выполнять и односторонние вращения матрицы A – нужность этого математик, реализующий алгоритм, определит в зависимости от того, нужны ли также и вычисления сингулярных векторов.

Поэтому для хорошего распараллеливания метода Якоби в приложении к вычислению сингулярных чисел следует вывод: то, что виделось преимуществом в свете общего количества операций, при распараллеливании метода становится скорее помехой. Поэтому даже при решении проблемы сингулярных значений, как мы полагаем, от явных двусторонних вращений никуда не уйти, если нужно максимально быстро распараллелить метод Якоби. Далее при рассмотрении вариантов мы будем считать, что имеем дело только с двусторонними вращениям, то есть с явной версией метода Якоби.

"Пересечение" разных двусторонних вращений (см. Рис. 2), конечно, несколько затрудняет распараллеливание. Однако при наличии достаточных ресурсов можно распределить элементы матрицы по столбцам, таким образом, дублируя внедиагональные элементы и их

теоретической оценкой времени, нужного для выполнения алгоритма в концепции неограниченного параллелизма.

[1]естественно, не начальной, а текущей преобразованной версии этой матрицы

[2]также не начальной, а текущей преобразованной версии этой матрицы

перевычисление[1]. При этом, однако, не избежать на каждом таком шаге пересылки данных о параметрах матриц Гивенса. Поэтому логично хранить на всех устройствах также и всю диагональ, пересылая вместе с параметрами матриц Гивенса также и новые значения диагональных элементов. Как видно, параллельная схема довольно усложняется, и это еще не все нюансы. Кроме передач небольшого числа элементов, нужных для вращений, требуется понять, как пересылать для выполнения вращений сами столбцы – ведь от шага к шагу пары столбцов/строк во вращениях меняются.

2.4 Разные варианты прохода пар индексов в циклическом варианте с барьерами

Теперь рассмотрим, что есть в существующих версиях циклического метода Якоби с барьерами для выбора порядка прохождения одного цикла, то есть всех $\frac{n(n-1)}{2}$ пар индексов для матриц Гивенса, задающих собой двусторонние вращения. Далее мы будем предполагать, что *n* чётно (для нечётного *n* прохождение будет таким же, как для на единицу большего, просто с индексом *n+1* вращения выполняться не будут).

С одной стороны, несложно найти такое решение "задачи о распределении игр в турнире", в котором в "каждом туре" все индексы будут разбиты на $^{n}/_{2}$ пар, и поэтому полный цикл будет проходиться за *n-1* "тур". Однако при таком распределении индексов, как несложно убедиться, почти при всех значениях *n* "расписание турнира" будет таково, что в коммуникационной сети маршруты передачи будут довольно часто меняться. Между тем, как известно [6], даже постоянные пересылки между устройствами параллельной вычислительной системы лучше выполнять по одной и той же схеме: при этом её коммутаторы не требуют постоянной перестройки сети.

В связи с этим обращает на себя внимание исследование, выполненное в [9], в котором приведен такой порядок вращений, при котором топология передач сохраняется от "тура" к "туру", и при этом полный цикл проходится всего на 1 "тур" дольше. Он показан на Рис. 5, на котором каждому "туру" соответствует одна строка, серыми прямоугольниками объединены индексы, с которыми производится одно преобразование (двустороннее вращение). Видно, что на нечётных "турах" проводят по $^{n}/_{2}$ двусторонних вращений, а вот на чётных - на одно двустороннее вращение меньше (не участвуют во вращениях те индексы, что по краям).

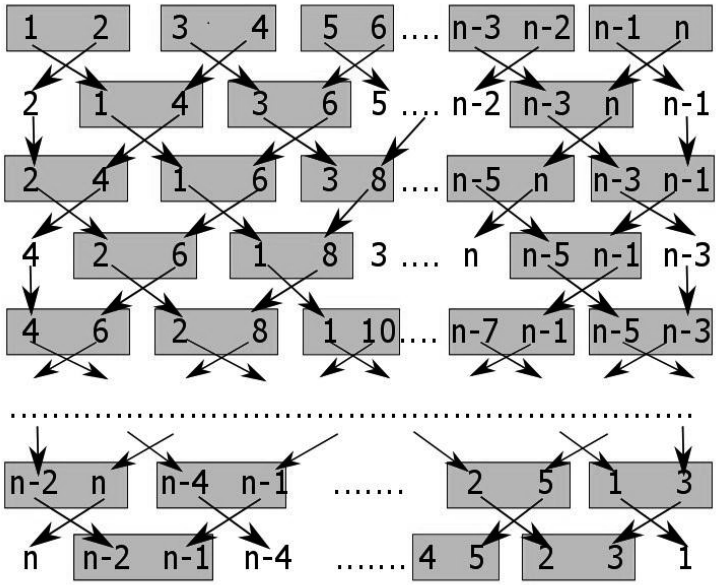

Рис. 5. Порядок двусторонних преобразований Гивенса, предложенный в [9]

[1] В такой схеме, однако, нужно специально проследить, чтобы порядок вычислений симметричных элементов был одинаков, иначе алгоритм, скорее всего, потеряет устойчивость.

Авторы полагают, что описанный комплекс причин, затрудняющих адаптацию, вполне убедительно показывает, почему метод Якоби – ни в явном, ни в неявном виде – не включён в наиболее известные пакеты вроде SCALAPACK.

3. Проблемы, возникающие при измерениях производительности

Ещё в 2016г. в [4] отмечалось, что при измерении производительности программ, реализующих описываемые в энциклопедии AlgoWiki алгоритмы, нужно быть внимательным, действительно ли реальной программой выполняется заявленное количество операций. Уже тогда был пример, когда в QR-разложении методом Хаусхолдера тестовая матрица оказалась настолько простой структуры, что ни о каком кубическом количестве операций не было и речи, и поэтому тот тест явно был непригоден для отображения его результатов на странице метода. Осенью 2018г. авторам поступил на рассмотрение для энциклопедии большой поток тестов[1], проведённых по нескольким методам, реализованным в пакете SCALAPACK. В связи с этим были выявлены сходные проблемы, обнаруживать которые, однако, было существенно труднее.

Matrix	Cores	Time (sec)	SpeedUp	Perfomance (GFLOPS)
4096	1	6.865000	1.000000	6.673414
4096	2	3.576000	1.919743	12.811237
4096	4	1.971000	3.483004	23.243523
4096	8	1.059000	6.482531	43.260609
4096	16	0.834000	8.231415	54.931636
4096	32	0.627000	10.948963	73.066961
4096	64	0.750000	9.153333	61.083979
8192	1	113.308000	1.000000	3.234581
8192	2	70.605000	1.324816	5.190905
8192	4	47.573000	2.381771	7.704031
8192	8	31.179000	3.623653	11.810998
8192	16	15.005000	7.551350	24.425450
8192	32	4.302000	26.338447	85.193834
8192	64	2.098000	54.007626	174.692029
16384	1	1445.902000	1.000000	2.027821
16384	2	662.908000	2.181150	4.422983
16384	4	453.624000	3.187446	6.463571
16384	8	303.797000	4.912895	9.4056909
16384	16	147.894000	9.777007	19.826024
16384	32	60.754000	23.799289	48.260707
16384	64	26.527000	54.506804	110.530064
32768	1	18757.072000	1.000000	1.250528
32768	2	9378.536000	2.000000	2.501056
32768	4	4689.268000	4.000000	5.002113
32768	8	2344.634000	8.000000	10.004226
32768	16	1172.317000	16.000000	20.010452
32768	32	642.553000	30.221478	36.504768
32768	64	275.257000	67.143851	85.215809

Рис. 6. Таблица, полученная от одного из тестеров на QR-разложениях

Основная трудность была связана с тем, что, если после срабатывания программы, реализовавшей метод Хаусходера, количество выполненных операций можно было прямо посчитать по полученной на выходе структуре данных, то в программах, реализовавших решения спектральных задач, такой возможности не было. Ситуация была осложнена тем, что данные спектральные алгоритмы, являясь по своей сути итерационными, считаются западными исследователями прямыми методами, поскольку для большинства матриц известна оценка количества требуемых для сходимости итераций. Между тем, при замене компьютерной арифметики на точную и снижении барьеров точности до нуля, эти методы, очевидно, не

[1] Тесты были получены в результате выполнения заданий студентами магистратуры факультета ВМК МГУ. Несмотря на критические отзывы о них, следует отметить, что выявленные при этом проблемы могут присутствовать и при тестировании более опытными исследователями.

сойдутся почти никогда, исключая только примеры совсем "хороших" матриц. Поэтому и в классификации AlgoWiki их называют "прямыми" не иначе как в кавычках.

Данные алгоритмы исследованы на сходимость достаточно хорошо, и в учебниках вроде [7,8] для их разновидностей указано или количество итераций, приходящееся на определение каждого собственного числа, или просто даётся оценка в арифметических операциях, в зависимости от размера матрицы *n*. Однако не следует забывать о формулировках этих оценок: они выполняются для большинства матриц, но вовсе не для всех. Этот факт заставляет, например, в выборе сдвигов для QR-итераций учитывать эти особые случаи, чтобы получение новых собственных значений не остановилось. Аналогичные исключения обычно существуют и для других алгоритмов – как прямых, так и итерационных. Различия между ними в том, что для настоящих прямых алгоритмов исключения могут дать только более быстрое окончание алгоритма, а вот итерационные методы могут сходиться как быстрее, так и медленнее, чем для большинства матриц.

Ещё одной проблемой, которая может повлиять на адекватность измерений производительности, является то, что для получения более общих результатов исследователи могут использовать случайно сгенерированные матрицы. Программы такого рода обычно состоят из генерации матриц с помощью генераторов случайных чисел, вызова тестируемой подпрограммы и сравнения результатов её срабатывания с эталонными. Для получения же замеров с разными размерами матриц и количеством вычислительных ресурсов часто используют скрипты, где эти размеры и ресурсы задают как параметры. На Рис. 6 приведена сводная таблица замеров такого рода, причём производительность посчитана, как теоретическая оценка количества операций, поделённая на время исполнения. При этом бросаются в глаза "суперлинейные ускорения" для размеров матрицы 16384 (сравните 1 и 2 ядра) и 32768 (на 64 ядрах ускорение более 67). Между тем, эффекты от более быстрого/медленного исполнения на конкретных матрицах должны подсказать тестирующему, что для вычисления "ускорений" одинаковым у матрицы должен быть не только размер, но и все элементы. Это, конечно, усложнило бы написание скриптов для замеров. Однако возможные колебания времени/производительности, снятые с правильных замеров, могли бы указать нам на какие-то проблемы в адаптации (переполнения кэша, проблемы с пересылками и т.п.). В случае же, когда эти колебания связаны просто с особенностями конкретных матриц, возможность такого анализа будет упущена.

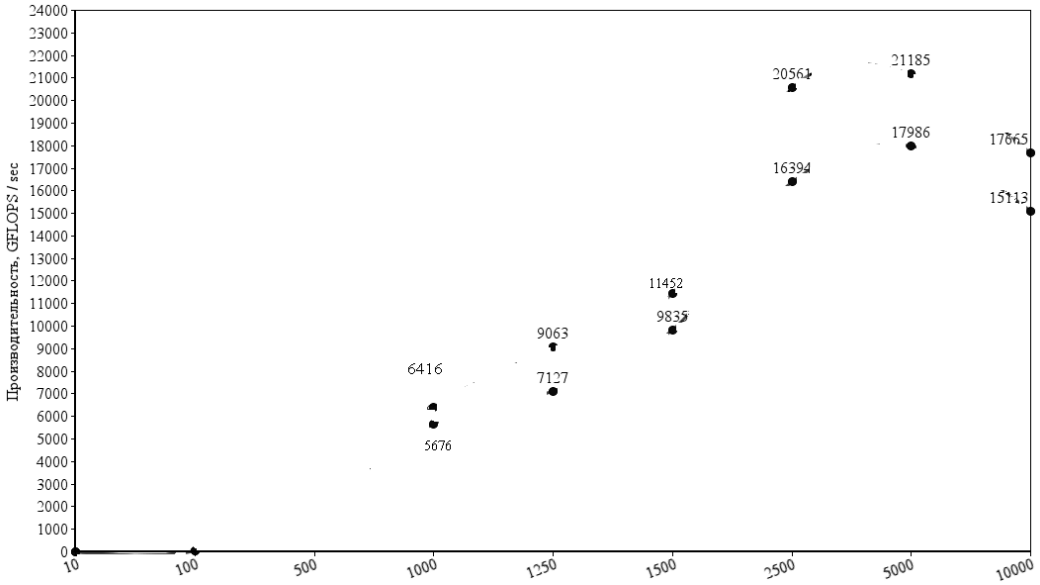

Рис. 7. График, полученный от одного из тестеров на спектральной задаче для графических ускорителей

Рассмотрим ещё один пример. На Рис. 7 приведён график производительностей одной из спектральных задач в зависимости от размера матрицы. Верхние точки соответствуют расчётам на двух графических ускорителях, нижние – на одном. Производительность опять была

вычислена как отношение теоретического количества операций к потраченному времени. Видно, что, хотя колебания такой вычисленной производительности и происходят примерно на одной размерности, однако относительные величины этих колебаний разные. Связано ли это с разной сходимостью метода на конкретных матрицах или же это характеризует работу на разном количества ускорителей – эти эффекты трудно разделить, пока обрабатываемые матрицы генерируются случайным образом в разных запусках программы.

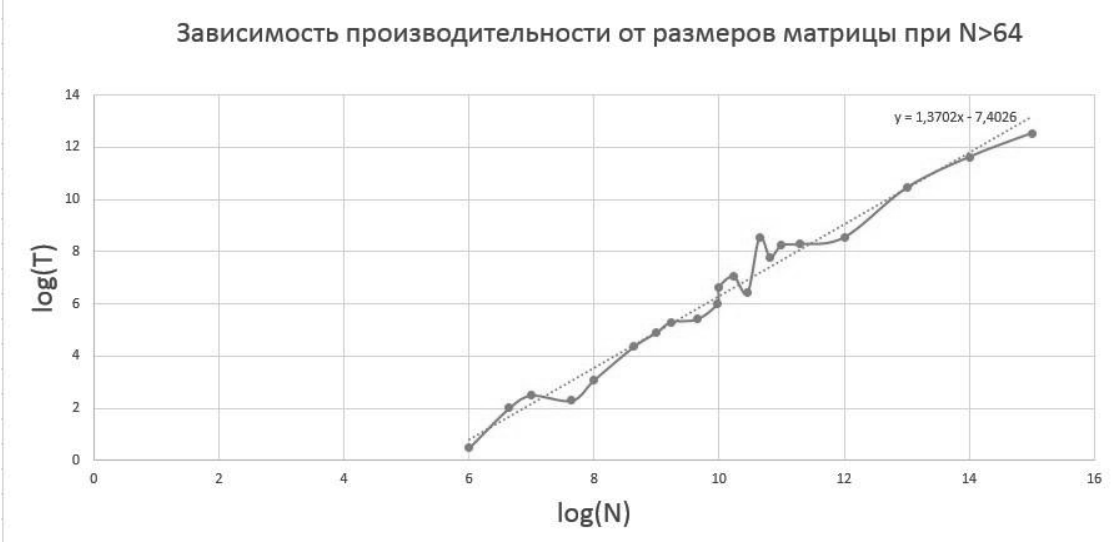

Рис. 8. График, полученный от одного из тестеров на спектральной задаче для кластерной архитектуры

На Рис. 8 приведён график производительности для решения задачи собственных значений в логарифмической[1] шкале, и тоже видны заметные колебания. Поскольку здесь видно уклонение как вверх, так и вниз, то логично предположить, что они тоже связаны не с тем, как программа взаимодействует с архитектурой, а с чем-то другим вроде разной скорости сходимости на конкретных матрицах.

Гораздо более адекватными являются в плане исследования адаптации программы пакета к архитектуре вычислительной системы такие замеры, где производительность измеряется как средняя от серии запусков (с возможным отбрасыванием самых быстрых и медленных). При такой организации тестирования возможные колебания графиков скорее всего укажут именно на критические размеры для адаптации по кэшу, пересылкам данных или т.п. Тем не менее, даже для таких замеров вероятность того, что сходимость задач для конкретных матриц как-то отличается от других, существует, хотя и более низкая, чем в замерах без усреднения.

Наиболее же кардинальным способом решить проблему неверной оценки является простой подсчет количества операций с помощью счётчиков, изменения которых вставляется в исследуемую программу после срабатывания таких процедур, в которых оценить количество операций можно точно. Это потребовало бы более досконального разбирательства в тексте исследуемой программы на предмет того, что делается в её частях и вызываемых функциях, и поэтому доступно только опытным и искушённым в программировании исследователям, но даже для них потребовало бы много сил и времени. Поэтому обычным выходом для тестирующих авторы всё же считают усреднение по группе запусков.

Описанные проблемы влекут за собой ещё одно важное следствие. В случаях, когда исследователями взамен старого "прямого", а в действительности итерационного, метода предлагается новый (например, [3,10,11]), следует, помимо теоретических оценок по сходимости, обязательно подкрепить их также и статистически, выполнив большую серию замеров на современных вычислительных системах. Без таких измерений для потенциальных пользователей нового метода будет не вполне ясно, насколько такой метод можно будет адаптировать на современных параллельных вычислительных системах разной архитектуры, поскольку нужные для этого оценки вычислительных затрат будут отсутствовать.

[1] в двоичных логарифмах

4. Заключение

В данной работе показаны две стороны работы исследователя, готовящего материал к AlgoWiki.

Во-первых, сопоставление разных вариантов одного вычислительного метода может по-новому осветить проблемы этого метода – как представляющиеся уже решенными, так и нерешенные. В частности, как уже отмечалось в предыдущих работах, возможен пересмотр таких взглядов на метод, которые считают одни сложности важнее других.

Во-вторых, проведение замеров для оценки степени адаптации методов к различным вычислительным системам содержит в себе ряд "подводных камней", которые могут помешать выполнить измерения производительности надлежащим образом. В частности, для группы итерационных методов, считающихся на Западе "прямыми", использование известных статистически средних данных по количеству операций в приложении к конкретным матрицам может привести к ошибочным результатам, в том числе и если данные генерируются случайным образом. Приведены рекомендации во избежание этих ошибок.

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. Открытая энциклопедия свойств алгоритмов. URL: http://algowiki-project.org (дата обращения: 30.03.2018).
3. Фролов А.В., Антонов А.С., Фролов Н.А. AlgoWiki: о некоторых характеристиках новых алгоритмов // Суперкомпьютерные дни в России: Труды международной конференции (24-25 сентября 2018 г., г. Москва). – М.: Изд-во МГУ, 2018. С. 43-49.
4. Фролов А.В., Антонов А.С., Воеводин Вл.В., Теплов А.М. Сопоставление разных методов решения одной задачи по методике проекта Algowiki // Параллельные вычислительные технологии (ПаВТ’2016): труды международной научной конференции (г. Архангельск, 28 марта – 1 апреля 2016 г.). Челябинск: Издательский центр ЮУрГУ, 2016. С. 347–360.
5. Фролов А.В., Антонов А.С. AlgoWiki: опыт исследования ряда алгоритмов // Параллельные вычислительные технологии – XII международная конференция, ПаВТ'2018, г. Ростов-на-Дону, 2–6 апреля 2018 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2018. – с.366–375.
6. Воеводин В.В. Математические основы параллельных вычислений. М.: Изд. Моск. ун-та, 1991. 345 с.
7. Воеводин В.В. Вычислительные основы линейной алгебры. М.: Наука, 1977. 304 с.
8. Деммель Дж. Вычислительная линейная алгебра. Теория и приложения. Пер. с англ. – М.: Мир, 2001. 430 с.
9. Zhou B. B., Brent R. P. On parallel implementation of the one-sided Jacobi algorithm for singular value decompositions // Proceedings Euromicro Workshop on Parallel and Distributed Processing, 401-408, 1995. DOI: 10.1109/EMPDP.1995.389182.
10. Braman K., Byers R., Mathias R. The multishift QR algorithm, I: Maintaining well-focused shifts and level 3 performance // SIAM J. Matrix Anal. Appl., 23(4): 929–947, 2002. DOI: 10.1137/s0895479801384573.
11. Braman K., Byers R., Mathias R. The multishift QR algorithm, II: Aggressive early deflation // SIAM J. Matrix Anal. Appl., 23(4):948–973, 2002. DOI: 10.1137/s0895479801384585.