УДК 519.612: 004.4

НОВЫЙ РЕШАТЕЛЬ ДЛЯ АЛГЕБРАИЧЕСКИХ СИСТЕМ РАЗРЕЖЕННЫХ ЛИНЕЙНЫХ УРАВНЕНИЙ С СИММЕТРИЧНОЙ ПОЛОЖИТЕЛЬНО ОПРЕДЕЛЕННОЙ МАТРИЦЕЙ

Е.А. Козинов, И.Г. Лебедев, С.А. Лебедев, А.Ю. Малова, И.Б. Мееров, А.В. Сысоев, С.С. Филиппенко

Нижегородский госуниверситет им. Н.И. Лобачевского

meerov@vmk.unn.ru

Поступила в редакцию 19.09.2012

Представлен новый решатель разреженных СЛАУ с симметричной положительно определенной матрицей. Описана поэтапная схема решения СЛАУ с использованием метода Холецкого. Выполнена последовательная программная реализация представленной схемы на основе супернодального подхода. Реализовано распараллеливание вычислений для систем с общей памятью. Приведены результаты экспериментов, дано их сравнение с результатами, полученными с помощью некоторых известных библиотек.

Ключевые слова: решение разреженных систем линейных уравнений, разреженные матрицы, супернодальный подход, разреженная алгебра.

Введение

Одной из актуальных задач алгебры разреженных матриц является решение систем линейных алгебраических уравнений $Ax = b$ с разреженной симметричной положительно определенной матрицей A. Разработка алгоритмов решения таких задач и их высокопроизводительная программная реализация, ориентированная на современные вычислительные архитектуры, представляет большой практический интерес. Для решения разреженных СЛАУ применяются как прямые, так и итерационные методы. Оба подхода имеют свои недостатки, вытекающие из внутренней природы методов. Так, например, прямые методы требуют серьезных объемов дополнительной памяти для представления промежуточных матриц, а итерационные методы могут демонстрировать достаточно медленную сходимость к решению [1]. На практике выбор метода во многом зависит от решаемой задачи.

На сегодняшний день в мире разработано большое количество специализированного программного обеспечения для решения больших разреженных СЛАУ – так называемые «решатели» СЛАУ. В соответствии с используемыми методами эти решатели подразделяются на прямые и итерационные, некоторые из них имеют высокопроизводительные реализации. В данной работе идет речь о разработке прямого решателя разреженных СЛАУ. Среди известных прямых решателей – MKL PARDISO, SuperLU, MUMPS, CHOLMOD и многие другие. Часть решателей ориентирована на последовательный режим работы, часть распараллелена для систем с общей памятью, часть ориентирована на работу в системах с распределенной памятью. Некоторые решатели работают только для симметричных положительно определенных матриц, другие – для матриц общего вида. Некоторые из решателей поддерживают режим работы out-of-core, используя жесткий диск в качестве «продолжения» оперативной памяти.

Постоянно обновляемый обзор прямых решателей от авторов SuperLU можно найти по следующей ссылке: http://crd.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf. Большой интерес представляет сравнительный анализ функциональности программного обеспечения, реализующего численные методы линейной алгебры, включая прямые и итерационные решатели разреженных СЛАУ, расположенный на странице Дж. Донгарра (J. Dongarra) http://www.netlib.org/ utk/people/JackDongarra/la-sw.html.

Коллективом авторов начата разработка собственной реализации высокопроизводительного прямого решателя разреженных СЛАУ с симметричной положительно определенной матрицей. Мотивация для создания своего инструмента состоит в потенциальной возможности разработки конкурентоспособного решателя, а также перспективах его использования в учебном процессе.

В работе приведены текущие результаты, дано их сравнение с результатами некоторых известных библиотек, а также определены пути дальнейшего развития. В качестве базы для сравнения выбраны один из лучших коммерческих прямых решателей MKL PARDISO, а также широко распространенные в академической среде разработки SuperLU и MUMPS.

1. Постановка задачи и метод решения

Пусть дана система линейных уравнений:

$$Ax = b. \quad (1)$$

Здесь A – разреженная симметричная положительно определенная матрица, b – плотный вектор, x – вектор неизвестных. Необходимо найти решение системы x.

Прямые методы решения задачи (1), как правило, основаны на применении разложения Холецкого к матрице A в виде:

$$A = U^T U, \quad (2)$$

где U – верхнетреугольная матрица, называемая фактором. В этом случае решение системы сводится к последовательному решению двух треугольных систем:

$$U^T y = b, \quad Ux = y. \quad (3)$$

Одна из основных проблем при численном решении задачи (1) состоит в том, что при разложении матрицы A ее фактор перестает быть разреженным. Степень заполненности фактора можно уменьшить с помощью переупорядочения строк и столбцов исходной матрицы. При этом осуществляется переход к эквивалентной системе (4), где P – матрица перестановки:

$$\widetilde{A}(Px) = Pb, \quad \widetilde{A} = PAP^T. \quad (4)$$

Таким образом, при решении разреженной системы линейных уравнений с использованием метода Холецкого можно выделить следующие этапы:

1. Переупорядочение – вычисление матрицы перестановки P и переход к системе (4).
2. Символическое разложение – построение портрета матрицы U, выделение памяти для хранения ненулевых элементов.
3. Численное разложение – вычисление значений матрицы U и размещение их в выделенной памяти.
4. Обратный ход – решение треугольных систем уравнений (3).

Представленный решатель систем линейных уравнений основан на данной вычислительной схеме. Его программная реализация выполнена на языке C, для хранения разреженных матриц используется формат CRS. Допускается работа в одинарной и двойной точности.

2. Переупорядочение матрицы

Основные методы переупорядочения, позволяющие снизить заполненность фактора разреженной матрицы, можно разделить на две группы: подходы, основанные на методе вложенных сечений [2], и подходы, основанные на методе минимальной степени [3]. Метод вложенных сечений (nested dissection) был предложен А. Джорджем в 1973 г. [4]. Он основан на многократном разбиении графа матрицы при помощи разделителей. По сравнению с методом минимальной степени он позволяет достичь большей степени параллелизма на стадиях факторизации.

Реализация метода вложенных сечений была выполнена в соответствии с описанием, приведенным в [5]. Предложена и реализована модификация метода, в которой выполняется выбор уровня разделителя в структуре смежности графа на основе оценки качества будущего разбиения. Для устранения несбалансированности разбиений графа, приводящей к увеличению заполненности и неравномерной вычислительной нагрузке при параллельных вычислениях, реализован алгоритм улучшения разбиения Эшкрафта – Лю [6], представляющий собой адаптацию методов Кернигана – Лина [7] и Фидуччиа – Маттейсеса [8] для вершинного разделителя графа. Предложена его ускоряющая модификация. Кроме того, для сокращения времени работы переупорядочения применяется предварительное сжатие графа матрицы за счет объединения вершин с одинаковой структурой [9]. Этот прием особенно эффективен для матриц, полученных при численном решении дифференциальных уравнений методом конечных элементов. В данном решателе сжатие графа реализовано на основе работ [9, 10].

3. Символическая фаза

Во время символической фазы (в некоторых источниках – фаза анализа) подготавливаются структуры данных для факторизации, что наряду с упрощением дальнейших вычислений обеспечивает возможность повторения численной фазы для последовательности матриц с одинаковым шаблоном. Основной задачей этапа является получение структуры расположения ненулевых элементов. Кроме того, в символическую фазу часто включают подсчет количества ненулевых элементов в факторе, нахождение постперестановки матрицы (postordering), построение дерева исключения, выделение «супернодов» (supernode) для супернодального ме-

тода, а также построение других вспомогательных структур.

Для нахождения числа ненулевых элементов в факторе, а также для распараллеливания вычислений используется специальная структура – дерево исключения [11]. Оно отражает зависимость между строками матрицы: если вершины дерева не соединены ребром, то соответствующие им строки фактора могут вычисляться параллельно. В рассматриваемой реализации построение дерева исключения выполняется после нахождения перестановки по алгоритму, предложенному Лю [12]. Затем к переупорядоченной матрице дополнительно применяется постпереупорядочение в соответствии с алгоритмом, приведенным в [1]. Этот прием повышает структурированность портрета исходной матрицы и фактора, не ухудшая его заполненности, что позволяет ускорить численную фазу факторизации. Затем при обходе дерева исключения выполняется оценка общего числа ненулевых элементов в факторе согласно алгоритму, предложенному Гильбертом, Нг и Пейтоном [13], а также выделяется память для массива столбцовых индексов. Далее определяется портрет фактора матрицы U на основе модификации процедуры из [1].

4. Численная фаза

Во время численной фазы выполняется нахождение значений элементов верхнего треугольника. Это самая трудоемкая часть факторизации, так как для вычисления k-й строки фактора нужно знать значения в строках, содержащих ненулевой элемент в k-м столбце.

Базовая реализация [14] численной фазы разложения Холецкого состоит в последовательном рассмотрении всех строк исходной матрицы и проведении операции гауссова исключения для ненулевых элементов, лежащих левее главной диагонали. Недостатком такого подхода к разложению Холецкого является низкая производительность на матрицах больших размерностей из-за возникновения существенного количества кеш-промахов. Для решения этой проблемы распространены два подхода: супернодальный (supernodal, «суперэлементный») и мультифронтальный (multifrontal).

Супернодальный подход впервые был предложен Эшкрафтом, Гримсом, Льюисом, Пейтоном и Симоном [15] в 1987 г., а затем исследован Нг и Пейтоном [16, 17]. Он основан на использовании для алгоритма факторизации так называемых «супернодов» (supernode) – группы расположенных подряд столбцов, имеющих одинаковую структуру заполненности ниже плотного треугольного блока. Использование супернодов позволяет производить факторизацию поблочно с применением оптимизированных матричных операций BLAS третьего уровня для плотных матриц. Подобный подход используется в MKL PARDISO, SuperLU, CHOLMOD и др.

Мультифронтальный подход был впервые представлен Даффом и Рейдом [18, 19] в 1983 г., а затем развит Лю [20, 21], Амстоем и Даффом [22], Амстоем, Даффом, Костером и Л'Экселлентом [23]. Подход состоит в разбиении процесса факторизации на факторизацию больших плотных подматриц, называемых фронтальными [21, 20]. При этом используются те же механизмы повышения производительности, что и в супернодальном случае. Оперирование с фронтальными матрицами выполняется с использованием операций BLAS третьего уровня, что позволяет значительно сократить время факторизации по сравнению с базовыми подходами. Одним из наиболее известных решателей, реализующих данный подход, является MUMPS.

В статье при реализации численной части используется супернодальный подход. Для выделения блоков применяются так называемые ослабленные суперноды (relaxed supernode [24]) – группы строк, имеющих различие в структуре левее плотного треугольного блока не более чем на фиксированное число элементов. Выделение супернодов выполняется после символической части на основе алгоритмов, приведенных в [24].

При выполнении этой процедуры задается параметр огрубления, отвечающий за максимальное количество нулей, добавляемых в структуру матрицы. Ускорение достигается за счет укрупнения матриц, используемых в базовых операциях. При этом большие значения параметра могут приводить к неудовлетворительным требованиям по памяти. Таким образом, значение параметра необходимо подбирать под конкретную задачу и программно-аппаратное окружение.

Программная реализация численной фазы в представленном решателе состоит из двух этапов: перевод исходной матрицы в формат BCRS на основе выделенных супернодов; выполнение факторизации с использованием матричных операций BLAS третьего уровня. Реализация алгоритмов основана на работах [25–28]. Для выполнения операций с плотными матрицами использовались функции из библиотеки Intel MKL.

5. Результаты экспериментов. Последовательная версия

Для анализа производительности программного комплекса был проведен ряд экспериментов на симметричных положительно определенных матрицах из коллекции [30] университета Флориды. Характеристики тестовых матриц представлены в табл. 1.

Таблица 1

Характеристики тестовых матриц

Название матрицы	Порядок	Число ненулевых элементов	Заполненность, %
pwtk	217 918	11 634 424	0.000125
msdoor	415 863	20 240 935	0.000060
parabolic_fem	525 825	3 674 625	0.000008
tmt_sym	726 713	5 080 961	0.000005
ecology2	999 999	4 995 991	0.000003
G3_circuit	1 585 478	7 660 826	0.000002

Параметры тестовой инфраструктуры приведены в табл. 2.

Таблица 2

Параметры тестовой инфраструктуры

Процессор	2 четырехъядерных процессора Intel® Xeon E5520 (2.27 GHz);
Память	16 Gb
Операционная система	Windows 7, Linux open suse 11.2
Компилятор	Intel C++ Compiler из поставки Intel® Parallel Studio XE 2011

Все вычисления проводились в двойной точности. В дальнейшем для решателя авторов указано лучшее время работы из тестируемых модификаций. В качестве собственного метода переупорядочения для матриц pwtk, msdoor, G3_circuit использовались вложенные сечения с улучшением разделения модифицированным методом Эшкрафта – Лю, для матрицы tmt_sym – вложенные сечения с выбором уровня разделителя, для матриц parabolic_fem, ecology2 – базовый метод автоматических вложенных сечений. Для каждой матрицы был выполнен подбор параметра огрубления при выделении супернодов. Для выполнения матричных операций использовался BLAS библиотеки Intel MKL. Дополнительно, для анализа эффективности выполнения численной факторизации, были проведены эксперименты с переупорядочением, полученным с помощью широко распространенной библиотеки METIS [31] (использовалась версия 4.0.1).

Также был проведен ряд экспериментов на указанных тестовых матрицах с использованием библиотек MKL PARDISO из пакета Intel MKL [32], SuperLU v. 4.1 [33], MUMPS v. 4.10 [34]. Для выполнения матричных операций BLAS в MUMPS использовался пакет Intel MKL. Для библиотеки SuperLU эксперименты проводились на той же аппаратной платформе под операционной системой Linux open suse 11.2, в других случаях использовалась операционная система Windows 7.

Далее для пакета SuperLU приведено время работы при одном из встроенных алгоритмов перестановки, дающем меньшее время работы. Для матриц pwtk, parabolic_fem, tmt_sym, ecology2 это приближенный столбцовый метод минимальной степени (COLAMD), для матрицы msdoor – множественный метод минимальной степени (MMD). Отметим, что на матрице G3_circuit программа завершила работу из-за внутренней ошибки работы с памятью. В библиотеке MKL PARDISO для получения перестановки использовался встроенный METIS. Для библиотеки MUMPS приведены результаты работы с автоматическим выбором одного из встроенных переупорядочивателей (для всех тестовых матриц было выбрано переупорядочение из библиотеки PORD).

Рассмотрим текущие результаты последовательной версии решателя в сравнении с последовательной версией MKL PARDISO и однопоточными версиями SuperLU и MUMPS (табл. 3). Как видно из таблиц, последовательная версия решателя показывает лучшие результаты, чем SuperLU, но отстает от MKL PARDISO и MUMPS.

Анализ результатов экспериментов последовательных версий решателей подтверждает, что время работы определяется стадиями переупорядочения и численной фазой разложения Холецкого. При этом время работы численной факторизации существенно зависит от заполнения фактора, которое, в свою очередь, определяется структурой исходной матрицы и алгоритмом переупорядочения.

Сравним размеры факторов матриц, полученных на тестовых задачах рассматриваемыми переупорядочивателями: собственным, METIS, PORD, SuperLU (табл. 4).

В сравнении с SuperLU, на всех тестовых примерах в собственной реализации получен меньший размер фактора. На матрице msdoor преимущество составляет 7%, на остальных матрицах – в среднем в два раза. Это в дальнейшем позволило увеличить выигрыш по времени при выполнении численной факторизации.

Таблица 3

Сравнение времени работы последовательной версии решателя с некоторыми сторонними решателями. Время приведено в секундах

Матрица	Решатель	Решатель с METIS	SuperLU	MKL PARDISO	MUMPS
pwtk	13.50	10.84	79.81	5.85	5.20
msdoor	50.29	14.19	80.47	6.68	5.85
parabolic_fem	8.71	16.21	37,59	6.41	7.02
tmt_sym	17.81	22.27	76.89	8. 39	12.64
ecology2	33.63	30.00	38.00	10. 28	10.50
G3_circuit	261.77	92.44	Ошибка	24.82	41.03

Таблица 4

Сравнение числа элементов в факторах матриц

Матрица	Собственная реализация	METIS	SuperLU	MKL PARDISO	MUMPS с PORD
pwtk	57 150 602	48 516 239	108 904 188	52 006 621	48 988 990
msdoor	103 929 030	52 699 437	111 245 898	57 129 312	54 793 824
parabolic_fem	26 872 725	26 283 509	52 361 272	28 279 494	28 496 994
tmt_sym	44 622 448	30 153 729	88 165 169	32 246 984	36 156 598
ecology2	33 203 230	37 735 953	68 677 613	40 087 694	32 956 605
G3_circuit	194 640 384	97 636 819	Ошибка	102 935 902	130 972 526

Собственная реализация переупорядочения показывает результаты, близкие по качеству к библиотекам METIS и PORD на большинстве тестовых матриц. На матрице parabolic_fem преимущество представленной реализации составляет около 5%, на матрице ecology2 – до 17% (METIS в составе MKL PARDISO). Наибольший проигрыш в размере фактора получен на матрицах msdoor, G3_circuit – до двух раз. Отметим, что время работы собственного переупорядочения на матрицах размером до 10^6 меньше, чем у METIS, в среднем в 2.4 раза. На матрицах большего размера METIS работает быстрее реализованного переупорядочивателя, в среднем в 5.5 раза. Это связано с тем, что реализованный в представленном решателе алгоритм сжатия не позволил уменьшить размеры графов матриц.

Таблица 3 показывает, что использование METIS вместо собственного переупорядочения в решателе авторов позволило сократить общее время решения на 4 матрицах из 6 (в среднем в 2.2 раза). На матрицах msdoor и G3_circuit, где собственное переупорядочение показывает худшие результаты, ускорение составляет в среднем 3.2 раза. На матрицах pwtk, ecology2 ускорение составило в среднем 19%.

Проанализируем время работы последовательных версий рассматриваемых библиотек. Решатель авторов на всех тестовых матрицах работает быстрее, чем SuperLU, в среднем в 3.5 раза. Наибольшее преимущество (более 4.3 раза) можно видеть на матрицах pwtk, parabolic_fem, tmt_sym. При использовании перестановки из METIS в решателе авторов опережение SuperLU составляет в среднем 4 раза. Наибольшее преимущество, полученное за счет улучшения качества перестановки, можно видеть на матрицах pwtk, msdoor (в 7.3 и 5.7 раза соответственно). При этом, как было отмечено, SuperLU не работает на самой большой из рассматриваемых матриц.

По сравнению с MKL PARDISO на всех матрицах, кроме msdoor и G3_circuit, отставание решателя авторов не превышает 3.3 раза и в среднем составляет 2.3 раза. На указанных матрицах из-за значительной разницы в размерах фактора решатель авторов проигрывает в среднем в 9 раз. Использование перестановки из METIS в решателе авторов позволило сократить максимальное отставание от MKL PARDISO до 3.7 раза (на G3_circuit). Данная версия на матрицах размером до $6{\cdot}10^5$ проигрывает MKL PARDISO в среднем в 2.2 раза, на матрицах большего размера – в среднем в 3.1 раза.

Характер сравнения с MUMPS аналогичен сопоставлению с MKL PARDISO. На матрицах, где размер полученных факторов был близок, отставание решателя авторов не более 3.2 раза. Наибольшее отставание – на матрице msdoor – в 8.6 раза. При использовании METIS в решателе авторов отставание от MUMPS не превышает 2.7 раза на всех тестовых матрицах.

6. Параллельная версия

Авторами была выполнена реализация параллельной версии решателя для систем с об-

щей памятью на основе супернодального подхода. На данном этапе работы распараллелена численная фаза разложения Холецкого, как наиболее трудоемкий этап вычислений.

В реализации авторов схема параллельных вычислений формируется на основе модификации дерева исключения, построенного по BCRS-портрету матрицы (разреженная матрица из плотных блоков). В этом случае дерево исключения показывает, над какими группами строк операции могут выполняться независимо. Для минимизации накладных расходов при организации параллелизма и равномерной загрузки потоков по дереву исключения строится так называемое дерево распараллеливания. Для этого объединяются некоторые вершины дерева исключения, принадлежащие одному и тому же поддереву. В полученной конструкции вычисления, соответствующие отдельному узлу, могут выполняться независимо. Число поддеревьев, на которое можно разбить дерево распараллеливания, является параметром алгоритма и подбирается в зависимости от задачи.

Выполнение параллельных вычислений осуществляется на основе Windows-потоков и шаблона параллельного программирования «мастер – рабочий». Распределение вычислений между «потоками-рабочими» происходит согласно дереву распараллеливания. «Поток-мастер» назначает обработку независимых вершин «потокам-рабочим», начиная с листьев дерева. Очередная вершина становится доступной для вычислений и назначается свободному «потоку-рабочему» после завершения вычислений для всех ее потомков.

7. Результаты экспериментов. Параллельная версия

Результаты экспериментов с параллельной версией разработанного решателя при работе в 8 потоков представлены в табл. 5. Для каждой тестовой задачи проводился подбор параметров: число Windows-потоков, используемых при численной факторизации, и число выделяемых поддеревьев в дереве распараллеливания. Использовались те же методы переупорядочивания, что и в последовательной версии.

Как видно из результатов экспериментов, ускорение численной фазы относительно последовательной реализации в среднем составляет 2.3 раза. Наибольшее ускорение достигается на матрице pwtk. При этом общее ускорение небольшое, в среднем в 1.5 раза. Это обусловлено тем, что была распараллелена только численная фаза разложения, занимающая в среднем 54.8% от общего времени работы в последовательной версии и 37.8% – в параллельной версии решателя. Большую часть оставшегося времени занимает переупорядочение. При использовании перестановки из METIS на половине матриц наблюдается рост ускорения численной части, в среднем оно составляет также 2.3 раза. Общее ускорение незначительно возрастает на матрицах большего размера. При этом время работы решателя с перестановкой из METIS сократилось для 4 матриц из 6 (в среднем в 2.1 раза). Наибольший выигрыш наблюдается, как и в последовательном случае, на матрицах msdoor и G3_circuit – в 3 раза.

Рассмотрим текущие результаты параллельной версии решателя в сравнении с параллельными версиями SuperLU, MUMPS и MKL PARDISO при работе в 8 потоков (табл. 6). Размеры факторов матриц не приведены, поскольку для MUMPS использовался тот же переупорядочиватель, что и в последовательной версии, для библиотек SuperLU и MKL PARDISO разница в размерах факторов матриц в сравнении с последовательной версией составляет не более 1.5%.

Как видно из таблицы, решатель авторов работает быстрее, чем SuperLU, на матрицах меньшего размера и на всех матрицах проигрывает MKL PARDISO и MUMPS.

На матрицах размером до 10^6 решатель авторов опережает SuperLU при использовании собственного переупорядочивателя или METIS, в среднем выигрыш составляет 1.5 раза. Наибольшее отставание решателя авторов достигается на матрице ecology2 – 3.8 раза при собственном переупорядочении, 3 раза – при использовании METIS. В среднем отставание от SuperLU незначительное и составляет 1.5 раза.

В сравнении с MKL PARDISO отставание решателя авторов не превышает одного порядка на матрицах размером до $1.5 \cdot 10^6$ (в среднем – в 5.3 раза). Сокращение размеров факторов матриц и, как следствие, времени работы при использовании METIS в решателе авторов позволило уменьшить отставание от MKL PARDISO. На матрицах размером до $1.5 \cdot 10^6$ оно составляет в среднем 4.3 раза, наибольшее отставание – на G3_circuit – в 8.1 раза.

В параллельной версии решателя удалось сократить отставание разработанного решателя от MUMPS. На матрице parabolic_fem решатель авторов обгоняет MUMPS на 5%. Отставание на других тестовых задачах не превышает 6.5 раза (на G3_circuit). При использовании METIS в решателе авторов отставание не превышает 2.7 раза (на ecology2).

Таблица 5

Работа параллельной версии решателя

Матрица	Решатель			Решатель с METIS		
	Время работы, с	Ускорение численной части	Общее ускорение	Время работы, с	Ускорение численной части	Общее ускорение
pwtk	6.26	3.76	2.16	5.23	4.38	2.07
msdoor	28.33	2.09	1.78	9.57	2.30	1.48
parabolic_fem	6.48	2.18	1.35	12.34	1.80	1.31
tmt_sym	13.76	2.03	1.29	17.16	1.86	1.30
ecology2	30.18	2.25	1.11	24.33	1.63	1.23
G3_circuit	204.11	1.64	1.28	64.44	1.97	1.43

Таблица 6

Сравнение времени работы параллельной версии решателя с другими решателями. Время в секундах

Матрица	Решатель	Решатель с METIS	SuperLU	MKL PARDISO	MUMPS
pwtk	6.26	5.23	13.59	2.04	3.30
msdoor	28.33	9.57	18.68	2.96	4.57
parabolic_fem	6.48	12.34	7.38	2.54	6.80
tmt_sym	13.76	17.16	14.77	3.39	11.84
ecology2	30.18	24.33	8.01	4.19	10.16
G3_circuit	204.11	64.44	Ошибка	7.91	31.35

Сравнение ускорений параллельных версий решателей относительно их последовательных (однопоточных) версий приведено на рис. 1.

Как видно из результатов экспериментов, значительную масштабируемость (в среднем в 5 раз) показывает только SuperLU. При этом время работы данного решателя в последовательной версии значительно отстает от других рассматриваемых библиотек и решателя авторов. Среднее ускорение MKL PARDISO составляет 2.6 раза. Ускорение решателя авторов имеет характер, аналогичный ускорению библиотеки MUMPS, однако, при отставании во времени работы, масштабируемость решателя авторов немного выше, чем у MUMPS.

Заключение

В данной статье представлен новый решатель разреженных СЛАУ с симметричной положительно определенной матрицей. На данный момент авторами реализована последовательная версия на основе супернодального подхода. Также разработана базовая параллельная версия решателя для систем с общей памятью, для которой было выполнено распараллеливание численной фазы факторизации.

Анализ результатов вычислительных экспериментов показал, что последовательная версия решателя авторов на всех тестовых задачах опережает по скорости решатель SuperLU (в сред-

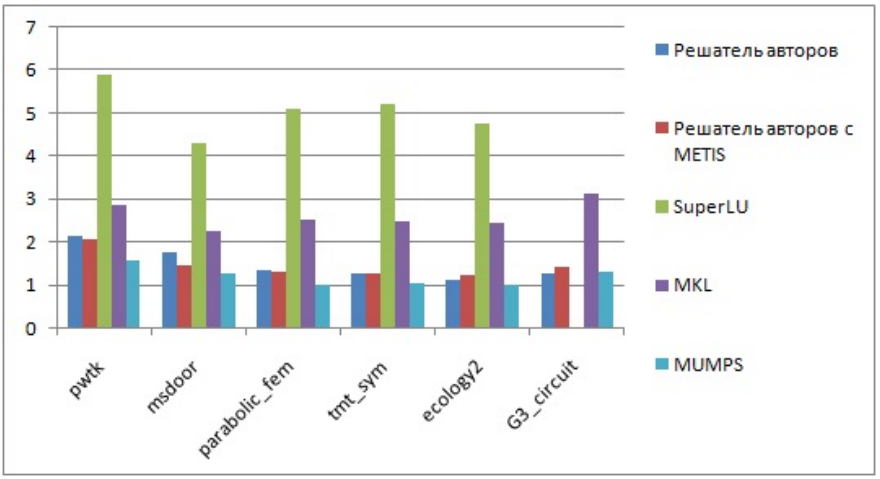

Рис. 1. Сравнение ускорений рассматриваемых решателей

нем в 3.5 раза), но проигрывает решателям MUMPS и MKL PARDISO (в среднем в 4.5 раза). Параллельная версия решателя опережает SuperLU на матрицах порядка до 10^6 и отстает на всех тестовых матрицах от MKL PARDISO и MUMPS. Использование в решателе авторов переупорядочения из библиотеки METIS позволяет сократить время работы решателя в среднем в 2.1 раза и уменьшить отставание по времени от других библиотек.

Таким образом, текущая версия решателя при работе в последовательном режиме сопоставима по скорости с ведущими коммерческими и академическими разработками. Прототип параллельной версии показывает работоспособность, но нуждается в дальнейшем развитии и оптимизации. Решатель используется в ряде НИР на факультете ВМК ННГУ, а также в учебном процессе в магистратуре в рамках курса «Параллельные численные методы».

Основные направления дальнейшей работы состоят в повышении производительности путем разработки новых алгоритмов, переноса части реализации на GPU и другие перспективные архитектуры.

Подготовлено в лаборатории «Информационные технологии» ITLab ННГУ при поддержке Минобрнауки РФ, соглашение 14.B37.21.0878.

Список литературы

1. Davis T.A. Direct methods for sparse linear systems. Philadelphia: SIAM, 2006. 217 p.

2. George A., Liu J.W.H. An Automatic Nested Dissection Algorithm for Irregular Finite Element Problems // SIAM J. on Numerical Analysis. 1978. Vol. 15, No 5. P. 1053–1069.

3. Tinney W., Walker J. Direct solutions of sparse network equations by optimally ordered triangular factorization // Proceedings of the IEEE. 1967. Vol. 55, No 11. P. 1801–1809.

4. George A. Nested dissection of a regular finite element mesh // SIAM J. on Numerical Analysis. 1973. Vol. 10, No 2. P. 345–363.

5. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. М.: Мир, 1984.

6. Ashcraft C., Liu J.W.H. A partition improvement algorithm for generalized nested dissection // Technical Report BCSTECH-92-020 Boeing computer Services. 1994.

7. Kernighan B.W., Lin S. An efficient heuristic procedure for partitioning graphs // The Bell System Technical J. 1970. Vol. 29. P. 291–307.

8. Fiduccia C.M., Mattheyses R.M. A linear time heuristic for improving network partitions // 19th IEEE Design Automation Conference, Proceedings. IEEE Press Piscataway, 1982. P. 175–181.

9. Ashcraft C. Compressed graphs and the minimum degree algorithm // SIAM J. on Scientific Computing. 1995. Vol. 16, No 6. P. 1404–1411.

10. Hendrickson B., Rothberg. Improving the runtime and quality of nested dissection ordering // SIAM J. on Scientific Computing. 1999. Vol. 20, No 2. P. 468–489.

11. Liu J.W.H. The role of elimination trees in sparse factorization // SIAM J. on Matrix Analysis and Applications. 1990. Vol. 11, No 1. P. 134–172.

12. Liu J.W.H. A compact row storage scheme for Cholesky factors using elimination trees // ACM Trans. Math. Software. 1986. Vol. 12, No 2. P. 127–148.

13. Gilbert J.R., Ng E.G., Peyton B.W. An efficient algorithm to compute row and column counts for sparse Cholesky factorization // SIAM J. Matrix Anal. Appl. 1994. Vol. 15, No 4. P. 1075–1091.

14. Писсанецки С. Технология разреженных матриц. М.: Мир,1988.

15. Ashcraft C., Grimes R.G., Lewis J.G. et al. Progress in sparse matrix methods for large linear systems on vector supercomputers // International J. of High Performance Computing Applications. 1987. Vol. 1, No 4. P. 10 – 30.

16. Ng E.G., Peyton B.W. A supernodal Cholesky factorization algorithm for shared-memory multiprocessors // SIAM J. on Scientific Computing. 1993. Vol. 14, No 4. P. 761–769.

17. Ng E.G., Peyton B.W. Block sparse Cholesky factorization on advanced uniprocessor computers // SIAM J. on Scientific Computing. 1993. Vol. 14, No 5. P. 1034–1056.

18. Duff I.S., Reid J.K. The multifrontal solution of indefinite sparse symmetric linear equations // ACM Trans. Math. Software. 1983. Vol. 9, No 3. P. 302–325.

19. Duff I.S., Reid J.K. The multifrontal solution of unsymmetric sets of linear systems // SIAM Journal on Scientific and Statistical Computing. 1984. Vol. 5. P. 633–641.

20. Liu J.W.H. The multifrontal method for sparse matrix solution: Theory and practice // SIAM Review. 1992. Vol. 34, No 1. P. 82–109.

21. Liu J.W.H. The multifrontal method and paging in sparse Cholesky factorization // ACM Trans. Math. Softw. 1989. Vol. 15. P. 310–325.

22. Amestoy P.R., Duff I.S. Vectorization of a multiprocessor multifrontal code // International Journal of Supercomputer Applications. 1989. Vol. 3. P. 41–59.

23. Amestoy P.R., Duff I.S., L'Excellent J.-Y., Koster J. A fully asynchronous multifrontal solver using distributed dynamic scheduling // SIAM J. on Matrix Analysis and Applications. 2001. Vol. 23, No 1. P. 15–41.

24. Ashcraft C., Grimes R. The influence of relaxed supernode partitions on the multifrontal method // ACM Trans. Math. Software. 1989. Vol. 15, No 4. P. 291–309.

25. Demmel J.W., Eisenstat S.C., Gilbert J.R. et al. A supernodal approach to sparse partial pivoting // SIAM J. Matrix Anal. Appl. 1999. Vol. 20, No 3. P. 720–755.

26. Hogg J.D. Efficient sparse Cholesky factorization. URL: http://www.maths.ed.ac.uk/~s0455378/EfficientCholesky.pdf.

27. Hogg J.D. Elimination Trees and Up-/Downdating of Sparse Cholesky Factorizations. URL: http://www.maths.ed.ac.uk/~s0455378/ETreesUpDown.pdf.

28. Hogg J.D., Reid J.K., Scott J.A. A DAG-based Sparse Cholesky Solver for Multicore Architectures // Tech. report RAL-TR-2009-004, Rutherford Appleton Laboratory. – 2009.

29. Liu J.W.H., Ng E.G., Peyton B.W. On finding supernodes for sparse matrix computations // SIAM J. on Matrix Analysis and Applications. 1990. Vol. 14, No 1. P. 242–252.

30. The University of Florida Sparse Matrix Collection. URL: www.cise.ufl.edu/research/sparse/ matrices/.

31. Karipis G. METIS. A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices // Technical report, University of Minnesota, Department of Computer Science and Engineering. 2011. URL: http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf.

32. Intel Math Kernel Library Reference Manual. URL: http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf.

33. Li X.S., Demmel J.W., Gilbert J.R. et al. SuperLU Users' guide // Technical report LBNL-44289. 2011. URL: http://crd-legacy.lbl.gov/~xiaoye/SuperLU/superlu_ug.pdf.

34. MUltifrontal Massively Parallel Solver (MUMPS 4.10.0) User's guide // Technical report ENSEEINT-IRIT. 2011. URL: http://mumps.enseeiht.fr/doc/ userguide_4.10.0.pdf.

NOVEL LINEAR EQUATIONS SYSTEMS SOLVER FOR SPARSE SYMMETRIC POSITIVE-DEFINITE MATRIX

E.A. Kozinov, I.G. Lebedev, S.A. Lebedev, A.Yu. Malova, I.B. Meyerov, A.V. Sysoyev, S.S. Filippenko

A new solver is presented for sparse SLAE with a symmetric positive definite matrix. A phased scheme of sparse SLAE solution based on the Cholesky method is described. A supernodal approach has been used for the phased scheme implementation. Parallel computing for shared memory systems has been implemented and the experimental results are given and compared with the results of some other well-known solvers.

Keywords: solution to sparse SLAE (systems of linear algebraic equations), sparse matrices, supernodal approach, sparse algebra.