РАЗРАБОТКА НОВОГО РЕШАТЕЛЯ РАЗРЕЖЕННЫХ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

С.А. Лебедев, Е.А. Козинов

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача решения систем линейных уравнений с симметричной разреженной положительно определенной матрицей методом Холецкого. Описывается новая последовательная реализация численной фазы разложения Холецкого, построенная на основе мультифронтального метода. Приводятся результаты вычислительных экспериментов, показывающие сопоставимость выполненной реализации с рядом известных библиотек. Формулируются планы по распараллеливанию для систем с общей памятью.

Введение

Решение разреженных систем линейных уравнений лежит в основе многих вычислительных задач компьютерной алгебры и моделирования физических процессов. Типичный пример – решение уравнений в частных производных методами конечных разностей или конечных элементов.

На сегодняшний день в мире разработано большое количество специализированного программного обеспечения для решения больших разреженных СЛАУ – так называемые «решатели» СЛАУ. В данной работе идет речь о разработке прямого решателя разреженных СЛАУ. Среди известных прямых решателей – MKL PARDISO, SuperLU, MUMPS, CHOLMOD и многие другие. Постоянно обновляемый обзор прямых решателей от авторов SuperLU можно найти по следующей ссылке: http://crd.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf.

В работе приведены текущие результаты, дано их сравнение с результатами некоторых известных библиотек, а также определены пути дальнейшего развития.

Постановка задачи

Пусть дана система линейных уравнений:

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

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

Метод решения

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

$$A = U^T U, \tag{2}$$

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

$$U^T y = b, \tag{3}$$

$$Ux = y. \tag{4}$$

Особенностью процедуры разложения Холецкого для разреженной матрицы является то, что матрица обычно претерпевает заполнение, что на практике может привести

к неудовлетворительным требованиям по памяти. Степень заполненности матрицы можно уменьшить с помощью переупорядочивания ее строк и столбцов. Это соответствует нахождению матрицы перестановки P и переходу к эквивалентной системе (6):

$$\bar{A} = PAP^T, \tag{5}$$

$$\bar{A}(Px) = PB. \tag{6}$$

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

1. Переупорядочивание – вычисление матрицы перестановки P и переход к системе (6);
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), (4).

В данной работе идет речь о разработке численной фазы разложения Холецкого.

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

Мультифронтальный метод

Существует несколько методов численного разложения, и наибольшее распространение среди них получили ориентированный влево (leftlooking), ориентированный вправо (rightlooking) и мультифронтальный (multifrontal) методы [3]. В данной работе в качестве численного разложения использовался мултифронтальный метод. Проще всего мультифронтальный метод может быть описан в терминах дерева исключения.

Деревом исключения матрицы A называется дерево, множество вершин которого совпадает с множеством вершин графа матрицы (т.е. множеством строк), а множество ребер задается соотношением (7). Т.е. вершина i является потомком вершины j, если первый после диагонали ненулевой элемент в строке i расположен в столбце j:

$$e_{i,j} \in E \leftrightarrow j = \min_k\{a_{ik} \neq 0 \,\&\, i < k\}. \tag{7}$$

Основными структурами данных для мультифронтального метода являются множества матриц обновления (update matrix) и фронтальных матриц (frontal matrix).

Каждому узлу дерева ставится в соответствие матрица, которая называется фронтальной. Применяя операцию обновления 1 уровня (8), из фронтальной матрицы может быть получен соответствующий столбец фактора. Для того чтобы вычислить фронтальную матрицу, необходимо найти сумму матриц обновления всех детей узла в дереве. Матрица обновления в свою очередь получается из фронтальной матрицы после операции обновления 1 уровня.

$$A = \begin{pmatrix} d & v^t \\ v & C \end{pmatrix} = \begin{pmatrix} \sqrt{d} & 0 \\ \frac{v}{\sqrt{d}} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & C - \frac{vv^t}{d} \end{pmatrix} \begin{pmatrix} \sqrt{d} & \frac{v^t}{\sqrt{d}} \\ 0 & I \end{pmatrix}. \tag{8}$$

Более подробное описание мультифронтального метода можно найти в работе Лю [10].

Недостатком описанного подхода к разложению Холецкого является низкая производительность на матрицах больших размерностей из-за возникновения существенного количества кэш-промахов. Для решения этой проблемы применяется супернодальный (supernodal, «суперэлементный») подход. Он основан на использовании для алгоритма факторизации так называемых «супернодов» (supernode) – группы расположенных под-

ряд 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.

Программная реализация

На основе описанной мультифронтальной схемы выполнена последовательная программная реализация решателя СЛАУ. Программная реализация выполнена на языке C. Для хранения матрицы разреженных матриц выбран формат CSR. Вычисления проводились в двойной точности.

При реализации численной части использовался супернодальный мультифронтальный метод с некоторыми модификациями [10], которые позволяют повысить производительность. Выделение супернодов выполняется после символической части на основе алгоритмов, приведенных в [1]. Реализация алгоритмов основана на работах [4, 6]. Для выполнения операции с плотными матрицами использовались функции из библиотеки Intel MKL [7].

Результаты экспериментов

Для анализа производительности программного комплекса был проведен ряд экспериментов на матрицах из коллекции [13] университета Флориды. В таблице 1 приведены характеристики выбранных матриц. Все они являются симметричными положительно определенными.

Таблица 1. Характеристики тестовых матриц

Матрица	Порядок	Число ненулевых элементов	Заполненность, %
pwtk	217 918	5 926 171	0,0125
msdoor	415 863	10 328 399	0,0060
parabolic_fem	525 825	2 100 225	0,0008
tmt_sym	726 713	2 903 837	0,0005
G3_circuit	1 585 478	4 623 152	0,0002
ecology2	999 999	2 997 995	0,0003

Параметры тестовой инфраструктуры приведены в таблице 2.

Таблица 2. Параметры тестового окружения

Процессор	2 четырехъядерных процессора Intel® Xeon E5520 (2.27 GHz)
Память	16 Gb
Операционная система	Windows
Среда разработки	Microsoft Visual Studio 2008
Компилятор	Intel® Parallel Studio XE 2013

В таблице 3 приведены результаты запусков последовательной версии решателя на тестовых матрицах, выделено время работы каждого этапа.

Таблица 3. Время работы последовательной реализации (в секундах)

Матрица	Число ненулевых элементов в факторе	Символическая часть	Численная часть
pwtk	49 426 260	1,06	5,39
msdoor	54 620 729	1,45	5,29
parabolic_fem	25 554 689	0,71	2,91
tmt_sym	30 095 144	0,91	3,81
G3_circuit	104 315 886	2,72	18,25
ecology2	36 283 143	1,13	5,07

Также был проведен ряд экспериментов на тех же тестовых матрицах с использованием следующих библиотек:

- MKL PARDISO: Intel® Math Kernel Library (в составе Intel® Parallel Studio XE 2013).
- MUMPS Version 4.10.0 [11].

Результаты экспериментов приведены в таблице 4. Для всех решателей использовались перестановки, полученные при использовании разработанного в ННГУ переупорядочивателя [14, 15].

MKL PARDISO при выполнении разложения Холецкого использует супернодальный метод, ориентированный влево, а MUMPS – супернодальный мультифронтальный метод.

Таблица 4. Сравнение работы численной фазы на тестовых матрицах

Матрица	Решатель авторов	MUMPS	MKL
pwtk	7,57	6,23	7,42
msdoor	8,28	6,91	8,21
parabolic_fem	4,42	4,93	4,34
tmt_sym	5,68	6,76	5,67
ecology2	7,35	9,22	7,01
G3_circuit	23,15	23,58	21,56

Как видно из таблицы 4, последовательная версия решателя показывает результаты, сравнимые с MUMPS и MKL. Т.к. наиболее трудоемким этапом вычислений является численная факторизация, время работы всего решателя в значительной мере определяется полученным числом ненулевых элементов в факторе и временем работы численной фазы, которое зависит от качества работы метода перестановки. Реализация авторов работает незначительно быстрее реализации из библиотеки MUMPS на 3 из 6 матриц (parabolic_fem, tmt_sym, ecology2), при этом отстает от Intel MKL в среднем всего лишь на 4%.

Заключение

Основным результатом работы является программная реализация мультифронтального метода. Реализованный алгоритм позволяет выполнять численное разложение разреженных положительно определенных матриц более эффективно по сравнению с предыдущим [14] и показывает сравнительное время работы с рядом широко распространенных прямых решателей.

Основным направлением дальнейшего развития является распараллеливание представленного алгоритма на системах с общей памятью (в том числе и ускорители Intel Xeon Phi), т.к. именно численная часть разложения занимает большую часть времени и требует значительных ресурсов памяти.

Вопрос об эффективной реализации параллельного алгоритма не решен до конца и представляет большой практический интерес [12, 2, 9]. Использование только высокопроизводительных библиотек, таких как BLAS, не дает приемлемых результатов из-за непоследовательного обращения к памяти и как следствие возникновения большого числа кэш-промахов, а также в связи с малыми размерами матриц, обрабатываемых функциями BLAS. Поэтому основной идеей распараллеливания является использование дерева исключения, при этом каждый узел дерева представляет собой отдельную вычислительную задачу, а узлы, не имеющие общих потомков, могут обрабатываться параллельно. При таком подходе можно рассматривать две формы балансировки – статическую и динамическую.

Одним из первых алгоритмов, использующих статическую балансировку, был алгоритм Гейста и Нг [5]. Основной идеей этого алгоритма является выделение некоторого уровня в дереве, т.е. множества вершин, таких, что для всех поддеревьев в корне с вершиной из этого множества объем вычислений примерно одинаков. В настоящее время алгоритмы, использующие статическую балансировку, так или иначе основаны на алгоритме Гейста–Нг.

При динамической балансировке используется схема мастер–рабочий, в которой мастер назначает узлы дерева рабочим в соответствие с их нагрузкой. Однако при таком простом подходе появляется большое число обращений потока в память другого потока, поэтому, как и при статической балансировке, потоку предпочтительнее назначать некоторое поддерево дерева исключения. Поиск наилучших структур данных и оптимального способа динамической балансировки продолжается до сих пор [8].

Исследование возможности создания эффективных реализаций рассмотренных алгоритмов для распараллеливания численной фазы на общую память является направлением дальнейшей работы.

Работа выполнена в лаборатории ННГУ-Intel «Информационные технологии». Авторы благодарят И.Б. Меерова, А.Ю. Пирову, А.В. Сысоева за полезные обсуждения и внимание к работе.

Литература

1. 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.
2. Chowdhury I., L'Excellent J.-Y. Some experiments and issues to exploit multicore parallelism in a distributed-memory parallel sparse direct solver. Research report RR-4711, INRIA and LIP-ENS Lyon, Oct. 2010.
3. Davis T.A. Direct methods for sparse linear systems. – Philadelphia: SIAM, 2006. – 217 p.
4. Demmel J.W., Eisenstat S.C., Gilbert J.R., Li X.S., Liu J.W.H. A supernodal approach to sparse partial pivoting // SIAM J. Matrix Anal. Appl. – 1999. – Vol. 20, No. 3. – P. 720–755.
5. Geist A., Ng. E.G. Task scheduling for parallel sparse Cholesky factorization // Int J. Parallel Programming, 18:291(314). 1989.
6. Hogg J.D. Efficient sparse Cholesky factorization – URL.: http://www.maths.ed.ac.uk/~s0455378/EfficientCholesky.pdf.
7. Intel Math Kernel Library Reference Manual. URL: [http://software.intel.com/sites/products/documentation/hpc/mkl/mklman.pdf].
8. L'Excellent J.-Y. Multifrontal method: Parallelism, Memory Usage and Numerical Aspects. Charge de recherché, Inria, Sept. 2012.
9. L'Excellent J.-Y., Sid-Lakhdar M. Introduction of shared-memory parallelism in a distributed-memory multifrontal solver. Research report №8227, Project-Team ROMA, Feb. 2013.
10. Liu J.W.H. The multifrontal method for sparse matrix solution: Theory and practice // SIAM Review. – 1992. Vol. 34, No. 1. – P. 82–109.
11. 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]
12. Posey S. GPU progress in sparse matrix solvers for applications in computational mechanics // European seminar on computing, Pilzen, Chech Republic, June, 2012.
13. The University of Florida Sparse Matrix Collection – URL.: www.cise.ufl.edu/research/sparse/matrices/.

14. Козинов Е.А., Лебедев И.Г., Лебедев С.А., Малова А.Ю., Мееров И.Б., Сысоев А.В., Филиппенко С.С. Новый решатель для алгебраических систем разреженных линейных уравнений с симметричной положительно определенной матрицей // Вестник Нижегородского университета им. Н.И. Лобачевского. – Н. Новгород: Изд-во ННГУ, 2012. – №5(2) – С. 376-384.
15. Пирова А.Ю. Разработка нового переупорядочивателя симметричных разреженных матриц // Материалы XIII всероссийской конференции «Высокопроизводительные параллельные вычисления на кластерных системах». Н.Новгород: Изд-во ННГУ, 2013. В печати.