О методах решения большеразмерных задач строительной механики на многоядерных компьютерах

Д.т.н., с.н.с. С.Ю. Фиалко*,
Краковский технологический университет им. Тадеуша Костюшко

Ключевые слова: метод конечных элементов; прямые методы решения СЛАУ; метод сопряженных градиентов; предобусловливание; многоядерные компьютеры

Развитие современных программных комплексов и тенденция к высокой детализации расчетных моделей приводят к возрастанию размерности расчетных моделей зданий и сооружений до 1 000 000–8 000 000 уравнений. Такие задачи все чаще решаются на многоядерных настольных компьютерах, а не на кластерах, мощных рабочих станциях и компьютерных сетях. Поэтому появилась необходимость разработки современных прямых и итерационных методов решения систем линейных алгебраических уравнений (СЛАУ) с разреженными симметричными матрицами, возникающих при применении метода конечных элементов к задачам строительной механики. Эти методы должны учитывать специфику данных компьютеров, относящихся к архитектуре SMP (Symmetric Multiprocessing), когда несколько одинаковых процессоров взаимодействуют с оперативной памятью (ОП), разделяемой между ними. Такие компьютеры имеют ограниченный объем ОП и слабую пропускную способность системы памяти. Поэтому для большинства алгоритмов, используемых в инженерных и научных расчетах, ускоряемость вычислений при возрастании количества процессоров (speed up) существенно ограничена.

В данной работе мы представим краткое описание прямых методов для разреженных матриц, а также описание итерационного метода PSICCG (Parallel Sparse Incomplete Cholesky Conjugate Gradient), разработанного автором и внедренного в развивающуюся новую версию программного комплекса SCAD, а также приведем сопоставление производительности этих методов на нескольких примерах расчета реальных конструкций многоэтажных зданий.

Блочный многофронтальный метод подконструкций BSMFM (Block Substructure Multifrontal Method)

Этот метод детально описан в работе [1] и сочетает идею метода суперэлементов с автоматическим делением исходной конструкции на подконструкции, не требующим вмешательства пользователя. Декомпозиция конструкции на подсистемы осуществляется на основе методов упорядочения, значительно снижающих количество ненулевых элементов в факторизованной матрице. Как и в классическом многофронтальном методе [2, 3], разложение разреженной матрицы сводится к последовательности задач, решаемых для плотных прямоугольных матриц, что дает возможность применять матричные алгоритмы высокой производительности уровня 3 BLAS (Basis Linear Algebra Subroutines). Для увеличения производительности выполняется анализ разреженной матрицы с целью объединения узлов расчетной модели в группы везде, где это возможно, что приводит к увеличению размерности блока полностью собранных уравнений в плотных матрицах и к существенному увеличению производительности матричных алгоритмов.

Ведущей процедурой при разложении матрицы является матричное умножение – алгоритм dgemm (General Matrix Multiply), где первая буква d (double) означает, что операции выполняются с двойной точностью. Эта процедура многократно использует кэш процессора: данные, один раз считанные из оперативной памяти (ОП) в кэш процессора, многократно используются, причем загрузка в регистры процессора производится сразу из кэш, а не из медленной ОП. Поэтому система памяти оказывается не перегруженной, и алгоритм dgemm демонстрирует хороший speed up даже на обычных десктопах и ноутбуках. Алгоритм dgemm позволяет максимально использовать производительность процессора, поэтому прямые методы в современных программных комплексах так или иначе раскладывают матрицу на плотные прямоугольные блоки и используют матричное умножение вместо векторно-скалярных операций низкой производительности [1].

Многофронтальный метод может работать с любым упорядочением [4], а в случае нехватки объема оперативной памяти использовать диск, что позволяет решать большие задачи на компьютерах с ограниченным объемом ОП.

PARDISO – Parallel Direct Solver

Главным недостатком многофронтального метода является большое количество избыточных пересылок данных «память – память» и «память – диск». Поэтому в последнее время большую популярность получил метод PARDISO [5] из библиотеки процедур высокой производительности Intel Math Kernel Library (Intel MKL) [6]. PARDISO демонстрирует высокую производительность и хороший speed up, в этом отношении он значительно превосходит многофронтальный решатель. Однако пригоден этот метод только для решения задач, которые можно решать целиком в ОП. Хотя в руководстве Intel MKL написано, что PARDISO поддерживает режим OOC (Out of Core – использование дисковой памяти), на практике оказывается, что этот режим работает только для небольших задач, при этом производительность PARDISO в несколько раз ниже, чем многофронтального решателя. Для больших задач режим OOC заканчивается сообщением об ошибке [7, 8].

PARFES – Parallel Finite Element Solver

Сказанное выше послужило мотивацией для разработки конечно-элементного решателя PARFES [7, 9], который демонстрирует при работе в оперативной памяти (CM – Core Mode) производительность и ускоряемость, близкую к PARDISO, однако позволяет в случае дефицита ОП использовать диск, переключаясь в режимы OOC и OOC1. В режиме OOC метод показывает небольшое снижение производительности и speed up по сравнению с режимом CM. В режиме OOC1 PARFES выполняет большое количество обменов с диском, поэтому производительность и speed up существенно деградируют. Однако этот режим позволяет решать большие задачи на десктопах с малым объемом ОП и ноутбуках. В режиме OOC1 PARFES обычно требует меньшего объема ОП, чем многофронтальный метод.

PSICCG – Parallel Sparse Incomplete Cholesky Conjugate Gradient

Главными недостатками итерационных методов по сравнению с прямыми являются: невозможность обнаружения геометрической изменяемости расчетной модели, замедление или отсутствие сходимости в случае плохо обусловленных задач, необходимость выполнять итерационный процесс с начала для каждой правой части и использование низкопроизводительных основных алгоритмов – умножение разреженной матрицы на вектор и решение дополнительной системы уравнений относительно предобусловливания. На рисунке 1 приведены типичные зависимости для алгоритма матричного умножения и алгоритма умножения плотной матрицы на вектор. Здесь $S_p = T_1/T_p$ – speed up, или отношение времени решения задачи на одном процессоре к времени ее решения на p процессорах.

Прямая 1 – идеальный speed up, проходит через точки [0, 0], [1, 1], …. (если задача решается на двух процессорах, то в идеале мы бы хотели решить ее в два раза быстрее, чем на одном, на трех – в три раза быстрее, и т. д.). Кривая 2 соответствует алгоритму умножения матрицы на вектор, а кривая 3 – алгоритму dgemm. В отличие от алгоритма dgemm, алгоритм умножения матрицы на вектор работает со скоростью медленной системы памяти, а не быстрого процессора, причем система памяти вследствие загруженности не успевает обслужить несколько процессоров. Поэтому этот алгоритм ускоряется до тех пор, пока не произойдет исчерпания пропускной способности системы памяти. Далее увеличение числа процессоров не приводит к возрастанию speed up [1].

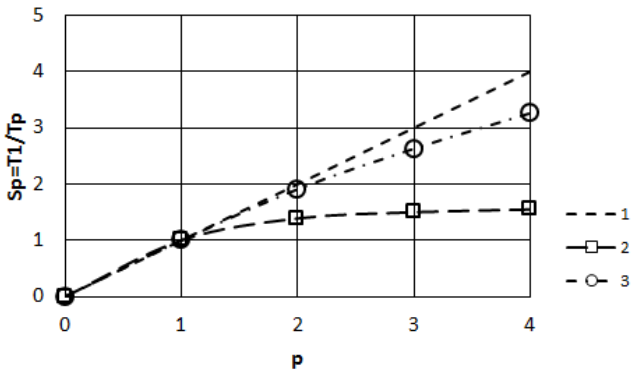

Рисунок 1. Ускоряемость (speed up) алгоритма матричного умножения (3) и алгоритма умножения матрицы на вектор (2) при возрастании количества процессоров *p*

Обзор известных библиотек высокопроизводительных процедур показал, что как в библиотеке Intel MKL, так и в библиотеке Фортрана IMSL для разреженных матриц отсутствуют версии параллельных процедур метода сопряженных градиентов и решения СЛАУ с нижней (верхней) треугольной матрицей [10, 11]. Это свидетельствует о том, что распараллеливание итерационных решателей на компьютерах архитектуры SMP – задача далеко не простая.

В предлагаемом методе распараллеливание основано на том, что подавляющее большинство реальных расчетных моделей подвержено действию нескольких загружений: собственный вес, длительные и кратковременные нагрузки, ветровая и снеговая нагрузки, нагрузки на часть этажа и т. д. При этом итерационный процесс для каждой правой части выполняется на отдельном потоке (процессоре), что обеспечивает распараллеливание всего вычислительного процесса, а не вилочно-подобного (fork-joint) распараллеливания на уровне ведущих процедур. В случае одной правой части выполняется распараллеливание только процедуры умножения матрицы на вектор.

Расчетные модели современных многоэтажных зданий (чаще всего именно такие объекты порождают СЛАУ высокого порядка) часто являются плохо обусловленными вследствие наличия множества конечных элементов тонких пластин и оболочек, сопряжения разнотипных элементов (стержней, пластин, оболочек и объемных элементов), наличия жестких связей, упругих опор и т. д. [12]. Кроме того, из-за сложности геометрии контуров не всегда удается выдержать оптимальные соотношения сторон и оптимальные углы для конечных элементов пластин и оболочек.

Эффективным методом борьбы с плохой обусловленностью является предобусловливание [13], суть которого состоит в переходе от заданной системы уравнений

$$\mathbf{Kx} = \mathbf{b}, \tag{1}$$

где **K** – симметричная разреженная матрица жесткости; **x** – вектор решения; **b** – правая часть, к системе

$$\mathbf{B}^{-1}\mathbf{Kx} = \mathbf{B}^{-1}\mathbf{b}, \tag{2}$$

где **B** – оператор предобусловливания – симметричная положительно определенная матрица значительно более редкой структуры, чем **K**, в силу чего вычислительная стоимость решения дополнительной системы уравнений $\mathbf{Bz}_k = \mathbf{r}_k$, где $\mathbf{r}_k = \mathbf{b} - \mathbf{Kx}_k$ значительно ниже, чем системы уравнений (1). Здесь $\mathbf{r}_k$ – вектор невязки на шаге итерации *k*. Если при этом $C\left(\mathbf{B}^{-1}\mathbf{K}\right) < C\left(\mathbf{K}\right)$, где *C*(…) – число обусловленности, то система (2) сходится за меньшее количество итераций, чем система (1). Если же при этом $\mathbf{B} = \mathbf{K}$, то приближенное решение сходится к точному за одну итерацию при любом стартовом векторе [13]. Последнее свойство служит ключом к построению эффективного предобусловливания.

Некоторые способы построения предобусловливания оговорены в работе [13]. В данной работе для построения предобусловливания используется неполная факторизация Холецкого, реализованная в технологии разреженных матриц [4]. При этом $\mathbf{B} = \mathbf{H} \cdot \mathbf{H}^{\mathrm{T}}$, где $\mathbf{H}$ – неполный фактор Холецкого. В процессе неполной факторизации пренебрегается малыми по величине элементами:

$$H_{ij} < \psi H_{ii} H_{jj}, \tag{3}$$

где ψ – параметр отбрасывания, $0 \le \psi \le 1$, $i, j \in [1, N]$. При отбрасывании элемента H_{ij} производится поправка диагональных элементов с целью обеспечения положительной определенности неполного фактора $\mathbf{H}$ [14, 15], что имеет решающее значение для гарантированной сходимости.

Чем параметр ψ ближе к нулю, тем меньшее количество итераций требуется для получения решения с заданной точностью, но тем больший объем оперативной памяти необходим для удержания неполного фактора $\mathbf{H}$ и тем большее время будет затрачено для выполнения прямых-обратных подстановок в процессе итерации. Для того чтобы параметр ψ можно было брать как можно меньшим, в данной работе применяется технология разреженных матриц.

Прежде всего, производится упорядочение узлов исходной конечно-элементной модели, которое значительно уменьшает количество ненулевых элементов в факторизованной матрице. При этом уменьшается и количество отбрасываемых элементов в процессе неполной

факторизации, поэтому следует ожидать, что в среднем при таком подходе в неполном факторе $\mathbf{H}$ набегает меньшая погрешность, чем при отсутствии упорядочения.

На основе многочисленных тестов установлено, что лучшие результаты получаются при упорядочении алгоритмом минимальной степени [16], поскольку плотность матрицы увеличивается только в самом конце разложения и отбрасывание малых элементов в начале матрицы в большей степени сказывается на значениях удерживаемых элементов, расположенных в конце матрицы, в меньшей – на значениях удерживаемых элементов в средней части матрицы, что ограничивает накопление ошибок, связанных с отбрасыванием. Другие популярные методы упорядочения (метод вложенных сечений, METIS [17, 18, 19], метод фактор-деревьев и др. [4]) такой особенностью не обладают и обычно приводят к худшим свойствам предобусловливания (см. пример 1). Идея используемого нами параллельного алгоритма неполной факторизации представлена в работе [20].

После окончания неполной факторизации производится вторичное отбрасывание малых внедиагональных элементов матрицы **H** [14]. Под малыми понимаются элементы $\mathbf{H}_{ij}^2 < \psi_1 \mathbf{H}_{ii}\mathbf{H}_{jj}$, где $0 \le \psi \le \psi_1 \le 1$. Таким образом, при неполной факторизации параметр ψ берется как можно меньшим, что позволяет лучше приблизить значения удерживаемых коэффициентов матрицы **H** ($\mathbf{H}_{ij}^2 \ge \psi_1 \mathbf{H}_{ii}\mathbf{H}_{jj}$) к значениям соответствующих им коэффициентов матрицы **L** – нижней треугольной матрицы полного разложения Холецкого. Вторичное отбрасывание и сжатие данных позволяет освободить часть оперативной памяти и ускорить прямые-обратные подстановки на этапе итераций при незначительном снижении способности предобусловливания ускорять сходимость.

Распараллеливание метода на этапе итераций осуществляется следующим образом. Сначала создается очередь задач *Q*, в которой каждой правой части *s* ϵ [1, *nrhs*] (*nrhs* – их количество) соответствует свое задание – для заданной нагрузки $\mathbf{b}^s$ определить вектор решения $\mathbf{x}^s$.

Затем в параллельном регионе поток *ip* (*ip* ϵ [0; *np*-1*]*) получает монопольный доступ к очереди заданий *Q*, выбирает ближайшее задание *s* и сразу же удаляет его из очереди. После этого любой другой поток получает право монопольно завладеть очередью. Поток *ip* запускает итерационный процесс для загружения *s*. Когда вектор решения будет получен с заданной точностью

$$\left(\frac{\left\| \mathbf{b}^s - \mathbf{K}\mathbf{x}_k^s \right\|_2}{\left\| \mathbf{b}^s \right\|_2} \wedge \frac{\left\| \mathbf{b}^s - \mathbf{K}\mathbf{x}_k^s \right\|_\infty}{\left\| \mathbf{b}^s \right\|_\infty} \right) < tol,$$

где *tol* – верхняя граница ошибки, определяемая в двух нормах; *k* – номер итерации, поток *ip* монопольно овладеет структурой данных, хранящей векторы решений, и поместит в нее полученное решение для правой части *s*. После этого поток *ip* освобождается и подхватывает следующее задание. И так до тех пор, пока все задания в очереди *Q* не будут выполнены. В результате осуществляется параллельное итерирование *np* правых частей.

Численные результаты

Рассмотрим примеры реальных задач, взятых из коллекции SCAD Soft. Названия задач сохранены такими, как их представили авторы соответствующих проектов. Исследование проводилось на компьютерах:

1. Ноутбук DELL XPS L502X с 4-ядерным процессором Intel® Core™ i7-27600QM CPU 2.4/3.4 GHz, RAM DDR3 8 GB, OS Windows 7 (64 bit) Professional, SP1.
2. Десктоп AMD Phenom™ II x4 995 3.2 GHz, RAM DDR3, 16 GB, OS Windows Vista™ Business (64-bit), SP2.
3. Рабочая станция с 16-ядерным процессором AMD Opteron 6276 2.3/3.200 GHz, RAM DDR3 64 GB, OS Windows Server 2008 R2 Enterprise, SP1.

Сравнивается производительность следующих методов: PSICCG, PARFES, BSMFM и ICCG0 [14] – классического итерационного метода сопряженных градиентов с предобусловливанием неполной факторизации Холецкого «по позиции».

Для метода ICCG0 структура матрицы **B** совпадает со структурой матрицы жесткости **K**, поскольку при неполной факторизации удерживаются только те элементы, которые расположены в тех же позициях, что и ненулевые элементы матрицы жесткости. Для того чтобы сходимость была гарантирована, необходимо обеспечить положительную определенность неполного фактора **H**. Поэтому матрица **B** представляется как **D**+1/(1+γ)·**S,** где **D** = Diag(**K**); **S** = **K** – **D**; γ – малый параметр, вначале полагаемый равным нулю [14]. Если в процессе неполной факторизации на главной диагонали появляется отрицательный элемент, это означает потерю положительной определенности матрицы **H**. Тогда параметр γ увеличивается (уменьшаются внедиагональные элементы матрицы **B**) и факторизация повторяется сначала. И так до тех пор, пока не добьемся положительной определенности матрицы **H**. Распараллеливание метода ICCG0 осуществляется так же, как и PSICCG.

Пример 1. Расчетная модель Ком_1 (2 546 400 уравнений, 4 загружения, рис. 2) содержит 3 и 4-узловые оболочечные конечные элементы, а также конечные элементы пространственной рамы. Толщины оболочечных элементов лежат в пределах от 0.2 м до 0.5 м. Сечения стержневых элементов меняются от 0.2 × 0.2 м до 1.5×0.5 м.

В таблице 1 приведена зависимость количества ненулевых элементов в факторизованной матрице **L** (Nonzero(**L**)), в неполном факторе **H** (Nonzero(**H**)), количества отброшенных элементов (Rejected), времени решения задачи и количества итераций для всех правых частей от метода упорядочения. Здесь RCM – обратный алгоритм Катхилла–Макки [4], MMD – улучшенная версия алгоритма минимальной степени [16]. Алгоритм METIS был упомянут ранее. На рисунках 3, 4, 5 приведены портреты факторизованной матрицы при различных методах упорядочения. Поскольку матрица симметричная, то показана только верхняя половина. Результаты подтверждают, что наиболее эффективным из рассмотренных является упорядочение алгоритмом MMD.

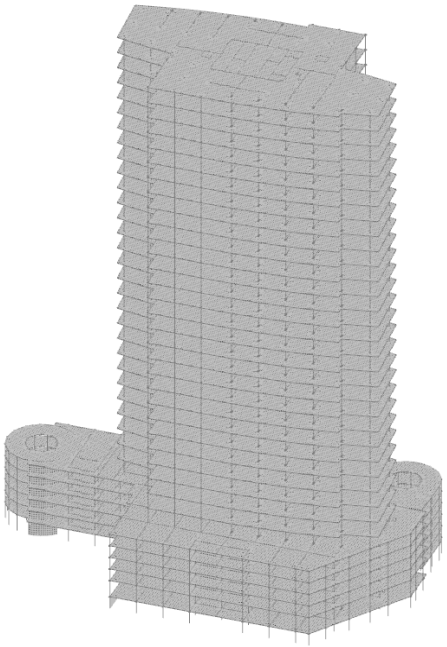

Рисунок 2. Расчетная модель Kom_1 (2 546 400 уравнений)

Таблица 1. Зависимость времени решения и количества итераций от метода упорядочения. Задача Ком_1 (2 546 400 уравнений) на компьютере с процессором Intel® Core™ i7-27600QM CPU 2.4 GHz, RAM DDR3 8 GB, количество процессоров 4, ψ=10^{-10}, $ψ_1 = 10^{-7}$, tol = 10^{-4}

Метод	Nonzero(L)	Nonzero(H)	Rejected	Время решения, с	Количество итераций
RCM	1 712 207 056	132 838 948	1 579 368 108	548	1 241
METIS	982 167 276	129 234 434	852 932 842	247	657
MMD	974 895 924	119 539 725	855 356 199	232	539

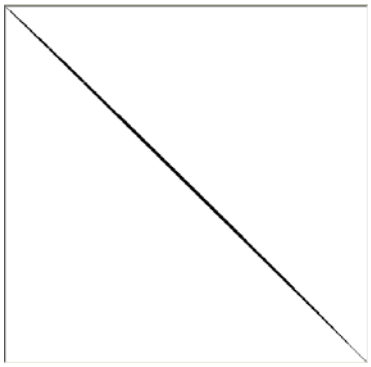

Рисунок 3. Упорядочение алгоритмом RCM

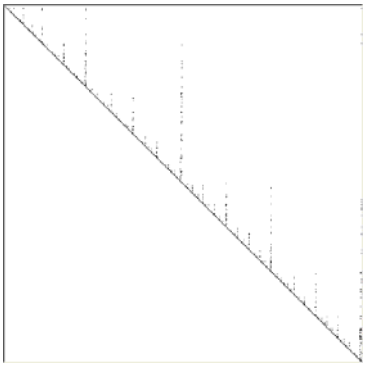

Рисунок 4. Упорядочение алгоритмом METIS

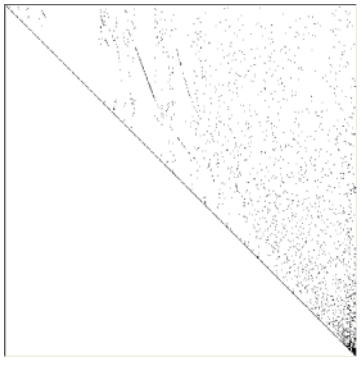

Рисунок 5. Упорядочение алгоритмом MMD

В таблицах 2, 3 приведено сопоставление параметров решения этой задачи на компьютерах с процессорами Intel® Core™ i7-27600QM и AMD Phenom™ II x4 995 для различных методов.

Таблица 2. Параметры решения задачи Ком_1 (2 546 400 уравнений) на компьютере с процессором Intel® Core™ i7-27600QM CPU 2.4 GHz, RAM DDR3 8 GB, $\psi = 10^{-10}$, $\psi_1 = 10^{-7}$, $tol = 10^{-4}$

Метод	Количество потоков	Режим	Время решения, с	Количество итераций для всех правых частей
PARFES	4	OOC	479	–
BSMFM	4	OOC	1 845	–
PSICCG	4	CM	247	657
ICCG0	4	CM	4 035	24 395

Лучший результат показал итерационный метод PSICCG. Малое количество итераций при принятых значениях параметра отбрасывания свидетельствует о том, что предложенный способ построения предобусловливания оказался эффективным. Классический метод ICCG0 показал значительно большее количество итераций, а продолжительность решения оказалась в 16 раз больше, чем при использовании метода PSICCG. Прямые методы использовали режим OOC, который снизил их производительность. При этом PARFES оказался в 3.9 раза быстрее многофронтального метода.

Таблица 3. Параметры решения задачи Ком_1 (2 546 400 уравнений) на компьютере с процессором AMD Phenom™ II x4 995 3.2 GHz, RAM DDR3, 16 GB, $\psi=10^{-10}$, $\psi_1 = 10^{-7}$, $tol = 10^{-4}$

Метод	Количество потоков	Режим	Время решения, с	Количество итераций для всех правых частей
PARFES	4	CM	211	–
BSMFM	4	OOC	1 778	–
PSICCG	4	CM	364	657
ICCG0	4	CM	5 559	24 395

На компьютере с процессором AMD Phenom™ II x4 995 объем ОП оказался в 2 раза больше, чем в предыдущем случае, что позволило PARFES работать в режиме CM и показать лучший результат. На рисунке 6 приведена ускоряемость метода PSICCG при увеличении количества загруженных ядер.

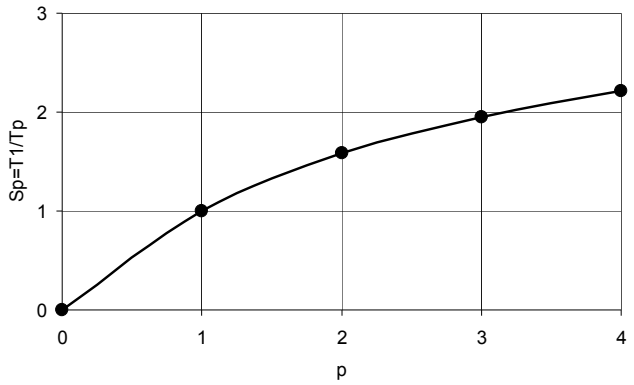

Рисунок 6. Ускоряемость решения при увеличении количества потоков на компьютере с процессором AMD Phenom™ II x4 995 3.2 GHz, RAM DDR3, 16 GB

Пример 2. Расчетная модель многоэтажного офисного здания Atrium_4_1 (7 328 394 уравнения, 5 загружений, рис. 7) включает оболочечные треугольные 6-узловые, четырехугольные 8-узловые, треугольные и четырехугольные транзитные конечные элементы, а также стержневые конечные элементы и конечные элементы специального назначения – упругие опоры, моделирующие работу упругого основания. Толщины оболочечных элементов лежат в пределах от 0.13 м до 0.6 м, что приводит к разбросу жесткостей до 100 раз.

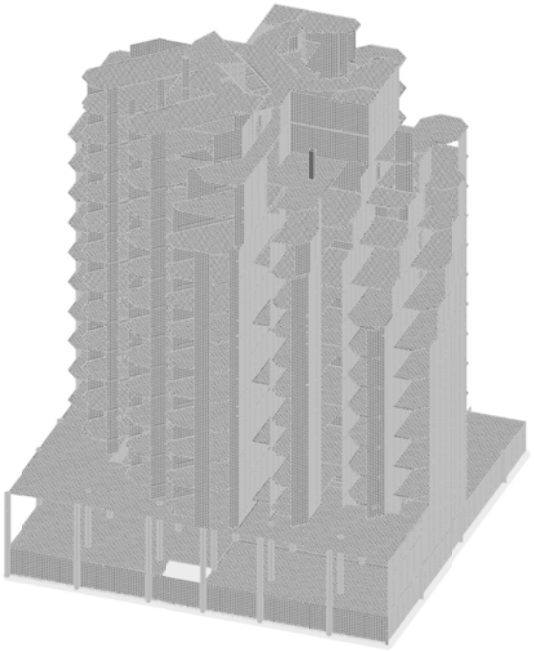

Рисунок 7. Расчетная модель Atrium 4_1 (7 328 394 уравнений)

Решение данной задачи на компьютере с процессором Intel® Core™ i7-27600QM оказалось неэффективным вследствие недостаточного количества оперативной памяти (8 GB). Для метода PSICCG $\psi = 10^{-5}$, что не позволяет обеспечить быструю сходимость метода. Продолжительность решения на четырех потоках составила 23 219 с, а для достижения сходимости при $tol = 10^{-4}$ было выполнено 25 703 итерации.

Результаты решения данной задачи на компьютере с процессором AMD Phenom™ II x4 995 приведены в таблице 4, а на рабочей станции AMD Opteron 6276 3,200/2,3 GHz – в таблице 5. Во всех случаях для прямых методов количество потоков *np* принимается равным количеству ядер процессора *n_cores*, а для итерационных – *np = min{n_cores, nrhs}*.

Таблица 4. Параметры решения задачи Atrium_4_1 (7 328 394 уравнения) на компьютере с процессором AMD Phenom™ II x4 995 3.2 GHz, RAM DDR3, 16 GB, $\psi=10^{-10}$, $\psi_1 = 10^{-7}$, $tol = 10^{-4}$

Метод	Количество потоков	Режим	Время решения, с	Количество итераций для всех правых частей
PARFES	4	OOC	17 445	–
BSMFM	Не решено – мало ОП			
PSICCG	4	CM	2 713	859
ICCG0	4	CM	57 769	72 146

Продолжительность решения задачи методом PARFES оказалась в 6.4 раза большей, чем для итерационного метода PSICCG, поскольку при объеме ОП 16 GB PARFES работал в режиме OOC1. Многофронтальному методу не хватило оперативной памяти. Сходимость классического итерационного метода ICCG0 оказалась очень медленной. Это позволяет сделать вывод о том, что данная задача является плохо обусловленной. Несмотря на это, предложенный способ построения предобусловливания в методе PSICCG позволил получить быструю сходимость.

Таблица 5. Параметры решения задачи Atrium_4_1 (7 328 394 уравнения) на компьютере с процессором AMD Opteron 6276 3,200/2,3 GHz, RAM DDR3 64 GB, $\psi=10^{-12}$, $\psi_1 = 10^{-7}$, $tol = 10^{-4}$

Метод	Количество потоков	Режим	Время решения, с	Количество итераций для всех правых частей
PARFES	16	CM	2 568	–
BSMFM	16	OOC	9 935	–
PSICCG	5	CM	2 006	852
ICCG0	5	CM	59 526	72 146

На компьютере с процессором AMD Opteron 6276 все методы, кроме многофронтального, работают в режиме оперативной памяти. Тем не менее, PSICCG решил эту задачу быстрее, чем PARFES. Многофронтальный метод потребовал в 3.9 раза большее время решения, чем PARFES.

На рисунке 8 приведена ускоряемость метода PSICCG при увеличении количества загруженных ядер.

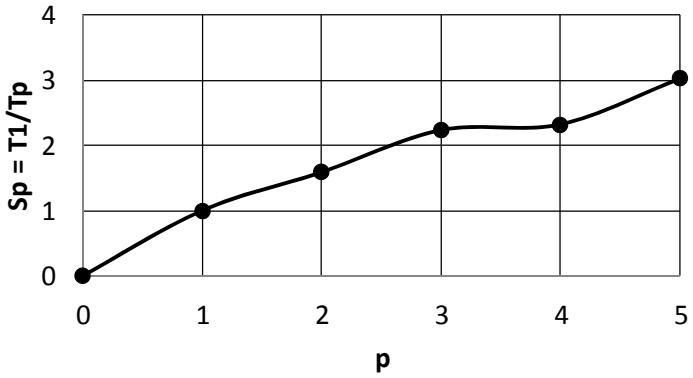

Рисунок 8. Ускоряемость решения при увеличении количества потоков на компьютере с процессором AMD Opteron 6276 3,200/2,3 GHz, RAM DDR3 64 GB

Заключение

Для рассмотренных большеразмерных расчетных моделей предлагаемый итерационный метод PSICCG, использующий неполную факторизацию Холецкого в технологии разреженных матриц, продемонстрировал устойчивую сходимость. При достаточно большом объеме оперативной памяти PSICCG позволил принять достаточно малую величину параметра отсечения ψ, что обеспечило быструю сходимость. Об этом свидетельствует малое количество итераций даже для плохо обусловленных задач, для которых классический метод ICCG0 сходится очень медленно.

Ускоряемость метода при увеличении количества потоков следует признать удовлетворительной, поскольку ведущие процедуры (умножение разреженной матрицы на вектор и решение СЛАУ относительно предобусловливания) относятся к процедурам низкой производительности, поэтому в архитектуре SMP их ускоряемость существенно ограничена пропускной способностью системы памяти (рис. 1).

При сравнении времени решения рассмотренных задач методом PSICCG и прямым методом PARFES оказалось, что если факторизованная матрица жесткости располагается в оперативной памяти, то обычно PARFES дает более быстрое решение (табл. 3). Однако для очень больших задач (табл. 5) метод PSICCG может оказаться более эффективным. При дефиците оперативной памяти PARFES переходит в режим OOC или OOC1. Тогда в ряде случаев применение итерационного метода PSICCG также оказывается более эффективным (табл. 2, 4).

Во всех рассмотренных случаях PARFES оказался более производительным и менее требовательным к ресурсам ОП, чем многофронтальный метод.

Таким образом, современные программные комплексы, реализующие метод конечных элементов, должны содержать как прямые, так и итерационные методы.

Работа выполнена при поддержке Национального научного центра Польши (Narodowy Centrum Nauki, Polska) на основе решения DEC-2011/01/B/ST6/00674. Автор приносит глубокую благодарность коллективу SCAD Soft за предоставление обширной коллекции реальных задач.

Литература

1. Фиалко С.Ю. Прямые методы решения систем линейных уравнений в современных МКЭ-комплексах. М.: СКАД СОФТ, 2009. 160 с.

2. Amestoy P.R., Duff I.S., L'Excellent J.Y. Multifrontal parallel distributed symmetric and unsymmetric solvers // Computer Methods. In Applied Mechanics and Engineering. 2000. Vol. 184. Pp. 501–520.
3. Gould N.I.M., Hu Y., Scott J.A. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations // Technical report RAL-TR-2005-005. Rutherford Appleton Laboratory. 2005.
4. Джордж А., Лю Дж. Численное решение больших разреженных систем уравнений. М.: Мир, 1984. 333 с.
5. Schenk O., Gartner K. Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems // Parallel Computing. 2002. Vol. 28. Pp. 187–197.
6. Intel® Math Kernel Library Reference Manual. Document Number: 630813-029US. [Электронный ресурс]. URL: http://www.intel.com/software/products/mkl/docs/WebHelp/mkl.htm. (дата обращения: 29.04.2012).
7. Fialko S. PARFES: A method for solving finite element linear equations on multi-core computers // Advances in Engineering Software. 2010. Vol. 40. No.12. Pp. 1256–1265.
8. Pardo D., Myung Jin Nam, Carlos Torres-Verdín,·Michael G. Hoversten, Iñaki Garay. Simulation of marine controlled source electromagnetic measurements using a parallel fourier hp-finite element method // Computers & Geosciences. 2011. Vol. 15. No.1. Pp. 53–67.
9. Fialko S. Parallel finite element solver for multi-core computers [Электронный ресурс] // Federated conference on computer science and information systems, September 9–12, 2012. Wrocław, Poland. IEEE Xplore Digital Library, INSPEC Accession Number: 13137537. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6354298&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D6354298. (дата обращения: 22.08.2013).
10. Intel® Math Kernel Library for Windows* OS User's Guide. Document Number: 315930-032US. Intel® MKL 11.0 - Windows* OS. Managing Performance and Memory. Improving Performance with Threading. Threaded Functions and Problems [Электронный ресурс]. URL: http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/mkl_userguide_win/index.htm. (дата обращения: 22.08.2013).
11. IMSL® Fortran Library Features. Fortran Library Documentation. [Электронный ресурс]. URL: http://www.roguewave.com/support/product-documentation/imsl-numerical-libraries/fortran-library.aspx. (дата обращения: 29.04.2012).
12. Perelmuter A.V., Fialko S.Y. Problems of computational mechanics relate to finite-element analysis of structural constructions // International Journal for Computational Civil and Structural Engineering. 2005. Vol. 2. No.1. Pp. 72–86.
13. Фиалко С.Ю. Сопоставление прямых и итерационных методов решения больших конечно-элементных задач строительной механики // В кн. Перельмутер А.В., Сливкер В.И. Расчетные модели сооружений и возможность их анализа. Киев: Сталь, 2002. С. 552–569.
14. Suarjana M., Kincho Law H. A robust incomplete factorization based on value and space constraints // International Journal for Numerical Methods In Engineering. 1995. Vol. 38. Pp. 1703–1719.
15. Jennings A. Development of an ICCG algorithm for large sparse systems // In: Evans D.J. (Ed.), Preconditioned methods. Theory and Applications. Gordon and Breach: Science Publishers, Inc., 1983. Pp. 425–438.
16. George A., Liu J.W.H. The evolution of the minimum degree ordering algorithm // SIAM Review. 1989. Vol. 31. No.1. Pp. 1–19.
17. METIS – Serial graph partitioning and fill-reducing matrix ordering. [Электронный ресурс]. URL: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview (дата обращения: 29.04.2012).
18. Karypis G., Kumar V. METIS: unstructured graph partitioning and sparse matrix ordering system // Technical report. Department of Computer Science, University of Minnesota, Minneapolis. 1995.
19. Karypis G., Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs // Technical Report TR 95-035. Department of Computer Science, University of Minnesota, Minneapolis. 1995.
20. Fialko S. Parallel sparse incomplete Cholesky conjugate gradient solver for multi-core computers // In book of abstracts of the 38th Solid Mechanics Conference Warsaw, Poland, August 27 – 31, 2012. Institute of Fundamental Technological Research, Polish Academy of Sciences. Warsaw, 2012. Pp. 50–51.

Сергей Юрьевич Фиалко, г. Краков, Польша
Тел. моб.: +48 510693187; эл. почта: sfialko@riad.pk.edu.pl

Фиалко С.Ю. О методах решения большеразмерных задач строительной механики на многоядерных компьютерах

doi: 10.5862/MCE.40.13

About analysis of large problems of structural mechanics on multi-core computers

S.Yu. Fialko,
Tadeusz Kościuszko Cracow University of Technology, Kraków, Poland
+48 510693187; e-mail: sfialko@riad.pk.edu.pl

Key words

finite element method; sparse direct solvers; preconditioned conjugate gradient method; incomplete Cholesky factorization; multi-core computers

Abstract

The sparse direct solvers as well as iterative methods are considered for analysis of large-scale finite element problems of structural mechanics on multi-core computers. The parallel preconditioned conjugate gradient method, based on sparse incomplete Cholesky factoring “by value” approach, is presented.

The efficiency of block substructure multifrontal method BSMFM, PARFES (Parallel Finite Element Solver), proposed iterative method PSICCG (Parallel Sparse Incomplete Cholesky Conjugate Gradient) and conventional ICCG0 method are compared. PARFES is proved much more effective than multifrontal method. When the amount of RAM is insufficient, the sparse direct methods produce lots of I/O operations, therefore, an iterative method is often more preferable.

References

1. Fialko S.Yu. *Priamye metody resheniya sistem lineynykh uravneniy v sovremennykh MKE-kompleksakh* [Direct method for solving linear equation system in FEM-complex]. Moscow: SKAD SOFT, 2009. 160 p. (rus)
2. Amestoy P.R., Duff I.S., L’Excellent J.Y. Multifrontal parallel distributed symmetric and unsymmetric solvers . *Computer Methods. In Applied Mechanics and Engineering*. 2000. Vol. 184. Pp. 501–520.
3. Gould N.I.M., Hu Y., Scott J.A. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. *Technical report RAL-TR-2005-005*. Rutherford Appleton Laboratory. 2005.
4. Dzhordzh A., Liu Dzh. *Chislennoye resheniye bolshikh razrezhennykh sistem uravneniy* [Numerical solution of big sparse set of equations]. Moscow: Mir, 1984. 333 p. (rus)
5. Schenk O., Gartner K. Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems. *Parallel Computing*. 2002. Vol. 28. Pp. 187–197.
6. *Intel® Math Kernel Library Reference Manual. Document Number: 630813-029US.* [Online source]. URL: http://www.intel.com/software/products/mkl/docs/WebHelp/mkl.htm. (accessed: April 29, 2012).
7. Fialko S. PARFES: A method for solving finite element linear equations on multi-core computers. *Advances in Engineering Software*. 2010. Vol. 40. No.12. Pp. 1256–1265.
8. Pardo D., Myung Jin Nam, Carlos Torres-Verdín, Michael G. Hoversten, Iñaki Garay. Simulation of marine controlled source electromagnetic measurements using a parallel fourier hp-finite element method. *Computers & Geosciences*. 2011. Vol. 15. No.1. Pp. 53–67.
9. Fialko S. *Parallel finite element solver for multi-core computers. Federated conference on computer science and information systems, September 9–12, 2012. Wrocław, Poland. IEEE Xplore Digital Library, INSPEC Accession Number: 13137537* [Online source]. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6354298&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D6354298. (accessed: August 22, 2013).
10. *Intel® Math Kernel Library for Windows* OS User's Guide. Document Number: 315930-032US. Intel® MKL 11.0 - Windows* OS. Managing Performance and Memory. Improving Performance with Threading. Threaded Functions and Problems* [Online source]. URL: http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/mkl_userguide_win/index.htm. (accessed: August 22, 2013).
11. *IMSL® Fortran Library Features. Fortran Library Documentation* [Online source]. URL: http://www.roguewave.com/support/product-documentation/imsl-numerical-libraries/fortran-library.aspx . (accessed: April 29, 2012).

12. Perelmuter A.V., Fialko S.Y. Problems of computational mechanics relate to finite-element analysis of structural constructions. *International Journal for Computational Civil and Structural Engineering*. 2005. Vol. 2. No.1. Pp. 72–86.

13. Fialko S.Yu. V kn. Perelmuter A.V., Slivker V.I. *Raschetnye modeli sooruzheniy i vozmozhnost ikh analiza* [In: Perelmuter A.V., Slivker V.I. Design models of structures and possibility of its analysis]. Kiev: Stal, 2002. Pp. 552–569. (rus)

14. Suarjana M., Kincho Law H. A robust incomplete factorization based on value and space constraints. *International Journal for Numerical Methods In Engineering*. 1995. Vol. 38. Pp. 1703–1719.

15. Jennings A. Development of an ICCG algorithm for large sparse systems. In: Evans D.J. (Ed.), *Preconditioned methods. Theory and Applications*. Gordon and Breach: Science Publishers, Inc., 1983. Pp. 425–438.

16. George A., Liu J.W.H. The Evolution of the Minimum Degree Ordering Algorithm. *SIAM Review*. 1989. Vol. 31. No.1. Pp. 1–19.

17. *METIS – Serial graph partitioning and fill-reducing matrix ordering* [Online source]. URL: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview (accessed: April 29, 2012).

18. Karypis G., Kumar V. *METIS: unstructured graph partitioning and sparse matrix ordering system. Technical report.* Department of Computer Science, University of Minnesota, Minneapolis. 1995.

19. Karypis G., Kumar V. *A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035.* Department of Computer Science, University of Minnesota, Minneapolis. 1995.

20. Fialko S. Parallel sparse incomplete Cholesky conjugate gradient solver for multi-core computers. *In book of abstracts of the 38th Solid Mechanics Conference Warsaw, Poland, August 27–31, 2012.* Institute of Fundamental Technological Research, Polish Academy of Sciences. Warsaw, 2012. Pp. 50–51.

Full text of this article in Russian: pp. 116–124