РЕАЛИЗАЦИЯ ПРЕДОБУСЛОВЛИВАТЕЛЯ ILU0 ДЛЯ БЛОЧНО-РАЗРЕЖЕННЫХ МАТРИЦ В БИБЛИОТЕКЕ PMLP/PARSOL

Д. А. Петров

ФГУП «РФЯЦ-ВНИИЭФ», г. Саров Нижегородской обл.

Известно, что использование решателей и предобусловливателей, ориентированных на конкретную задачу, позволяет получить серьезные выигрыши (относительно решателей и предобусловливателей общего назначения) в скорости нахождения решения систем линейных алгебраических уравнений (СЛАУ), с этой задачей связанной. Так, при решении уравнения Навье – Стокса по методике TVD комплекса ЛОГОС [1] возникают системы уравнений, в которых присутствуют одновременно несколько физических переменных. При этом переменные разной физической природы, как правило, связаны между собой соответствующими уравнениями. Поэтому имеет смысл рассматривать матрицы как блочно-разреженные: в большинстве случаев в них можно выделить небольшие (размера 5x5) блоки. Блоки могут быть неплотными, но в среднем в каждом их них значимыми являются примерно 90 % коэффициентов.

Также известно, что использование блочности – в данном случае плотных блоков коэффициентов матрицы – позволяет ускорить вычисления. Поэтому представляется разумным не только перевод предобусловливателей на их блочные аналоги, но и использование блочного формата хранения самих матриц. В рамках библиотеки линейных решателей PMLP/PARSOL [2] был реализован предобусловливатель неполного разложения ILU0 и некоторые его вариации, после чего было проведено их тестирование (в сочетании с итерационным решателем BiCGStab) на ряде конкретных примеров.

Выгоды по памяти от использования блочного формата хранения матриц

Чтобы численно оценить выигрыш в памяти при использовании блочности, рассмотрим следующий пример. Пусть в исходной разреженной матрице A количество ненулевых элементов равно nnz, и для ее хранения используется широко известный формат CSR. Тогда необходимый объем памяти можно приблизительно оценить как

$$V_1 \approx 8*nnz + 4*nnz = 12*nnz \text{ байт} \qquad (1)$$

(8 байтов/элемент для хранения значения элемента и 4 байта/элемент для хранения его столбцового индекса). Если использовать неполное ILU0 разложение для предобусловливания, то будет необходимо еще столько же памяти для хранения нижне- и верхнетреугольных матриц разложения

$$V_{\text{точ}D} \approx 24*nnz \text{ байт} - \text{общая память}. \qquad (2)$$

Теперь рассмотрим блочный формат хранения матриц BColsCSR, реализованный в библиотеке PMLP/PARSOL. Данный формат аналогичен блочному формату, используемому в Intel MKL [3]. Его основной идеей является то, что элементы матриц образуют квадратные блоки, которые сами могут рассматриваться как элементы. Для описания содержимого блока размера 5x5 вместо 25 столбцовых индексов необходим только один, для самого блока. Поэтому наша оценка памяти превращается в

$$V_{\text{блоч}D} \approx 2*(8*nnz + 4*nnz/25) = 16,32*nnz \text{ байт}. \qquad (3)$$

Также для минимизации затрат памяти для хранения предобусловливателя была реализована возможность использования одинарной точности (float) вместо двойной (double), даже если исходная матрица A хранится в double. Оценка общей памяти в этом случае будет выглядеть следующим образом:

$$V_{\text{блоч}F} \approx (8*nnz + 4*nnz/25) + (4*nnz + 4*nnz/25) = 12,32*nnz \text{ байт}, \qquad (4)$$

что уже практически в два раза меньше, чем $V_{\text{точ}D}$.

Стоит также отметить, что если для построения предобусловливателя будет требоваться больше памяти, чем для хранения исходной матрицы (например, для случаев ILU(k) с расширением портрета матриц), то использование одинарной точности в сочетании с блочностью даст еще большую выгоду по памяти.

Описание мелкоблочных предобусловливателей

Здесь и далее под мелкоблочным (в отличие, например, от блочного предобусловливателя Якоби) мы будем понимать предобусловливатели, чьи матрицы хранятся в блочном формате, а вычисления проводятся на уровне блоков размера 5x5. Так, например, при нахождении очередной блочной строки разложения ILU0 выполняются операции умножения блоков с вычитанием, аналогичные операции

$$a_{ij} = a_{ij} - \sum_{k<i} (a_{ik} * a_{kj}) \qquad (5)$$

для традиционного, «точечного» случая этого разложения. Вместо элемента a_{ij} рассматривается блок B_{ij}, и эта операция превращается в операцию вычитания матриц с умножением (каждый блок можно считать матрицей размера 5x5). Соответственно, аналогом деления элемента матрицы на другой элемент будет служить операция умножения блока на обратный блок.

Первым из реализованных предобусловливателей стал блочный предобусловливатель BlockILDU0. Он представляет собой вариацию неполного разложения ILU0, где дополнительно выделена блочная диагональ *D*. Таким образом, вместо системы уравнений

$$L*U*x = r \tag{6}$$

будет решаться система

$$L^{'}*D*U^{'}*x = r. \tag{7}$$

При этом производится обращение блочной диагональной матрицы *D*, чтобы вместо решения системы

$$D*z = r \tag{8}$$

сразу выполнять умножение

$$z = D^{-1}*r. \tag{9}$$

При реализации такого предобусловливателя стоит учесть два следующих факта: во-первых, блочные подматрицы *L* и *U* обладают блочно-единичной диагональю, а значит, эту диагональ можно не хранить (но помнить о ней при решении треугольных систем уравнений). А во-вторых – все операции с блоками надо оптимизировать под используемый формат. Например, если ориентация элементов в блоке идет по столбцам, то и в операциях надо учитывать этот фактор.

Результаты тестирования показали (см. табл. 1), что по времени решения и числу итераций полученные предобусловливатели BlockILDU0 (речь идет сразу о двух его реализациях, с одинарной и двойной точностью хранимых матриц) достаточно неплохо сравнимы со стандартным предобусловливателем ILU0. Однако на одном из примеров наблюдалось резкое увеличение числа итераций в блочном варианте предобусловливателя, объяснить которое было достаточно сложно.

Вторым вариантом блочного предобусловливателя стал ILU0_BCSR – предобусловливатель, в котором сочетаются точечное разложение исходной матрицы на верхне- и нижнетреугольную (т. е. шаблоны матриц сохраняются на уровне элементов, а не блоков) и блочный формат хранения и выполняемых операций. Алгоритм построения треугольных подматриц практически полностью совпадает со стандартным алгоритмом неполного ILU0 разложения, за исключением того, что все операции выполняются в матричном виде над блоками, а не над элементами. Кроме того, поскольку мы хотим сохранить шаблоны элементов внутри блоков, в результате этих операций не должно оказаться больше ненулевых элементов, чем это было изначально.

Блочная диагональ в случае ILU0_BCSR уже не будет единичной в полном смысле этого слова. Она будет присутствовать как в треугольной подматрице *L*, так и в подматрице *U*, но в блоках на диагонали для *L* на месте элементов $a_{ii}, i = 1...n$ будут стоять единицы.

Отдельным случаем неполного блочного ILU0 предобусловливателя выступает мелкоблочный предобусловливатель Якоби (BlockMinijac). Названный так из-за своей схожести с блочным предобусловливателем Якоби, этот вариант отличается от него размером блока. Если в блочном предобусловливателе Якоби размер блока велик (совпадает с диагональным блоком локальной на каждом процессоре части матрицы, относящейся к локальной подобласти сетки) и сильно разрежен, то в мелкоблочном варианте он – маленький, плотный и соответствует диагональному блоку коэффициентов, относящихся к одной ячейке сетки, в которой ищется несколько (в нашем случае обычно 5) неизвестных. Фактически, мелкоблочный предобусловливатель Якоби – это предобусловливатель Якоби (обратная диагональ) для случая блочной разреженной структуры коэффициентов матрицы.

Плотные блоки диагонали D_i заменяются на обратные, и вместо блочно-диагональной подматрицы *D* мы храним обратную к ней матрицу D^{-1}, также блочно-диагональную. Это позволяет на каждой итерации выполнять умножение матрицы на вектор вместо решения системы уравнений вида (8).

Несмотря на кажущуюся простоту, в некоторых случаях мелкоблочный предобусловливатель Якоби проявил себя достаточно неплохо.

Результаты тестирования по времени решения, числу итераций и потребляемой памяти

В табл. 1 приводятся результаты тестирования реализованных предобусловливателей на нескольких СЛАУ из верификационного набора тестовых примеров. Все системы решались согласно критерию точности $\|Ax-b\|_2 \le 10^{-6}\|b\|_2$.

Во всех случаях в качестве итерационного решателя использовался решатель метода бисопряженных градиентов со стабилизацией (BiCGStab). Параллельность выполнения на нескольких процессорах достигается за счет использования предобусловливателя Якоби и выбранных предобусловливателей в качестве внутренних.

Как видно из таблицы, эффективность предобусловливателей во многом определяется матрицей СЛАУ. В системах, требующих небольшого числа итераций для получения решения, основной выигрыш во времени работы достигается за счет более быстрого построения предобусловливателей в блочном формате. Однако здесь, для экономии места, детали измерений не приводятся.

В табл. 2 приведены результаты измерения потребляемой оперативной памяти для распределенной на четыре процессора матрицы M1. Все результаты приблизительные, поскольку выделяемая память рассчитывается для всего тестового комплекса, но, тем не менее, по ним можно судить об относительной эффективности реализованных предобусловливателей.

Приведенные значения памяти рассчитываются для всех четырех процессоров сразу. Из таблицы можно видеть, что использование мелкоблочных предобусловливателей с одинарной точностью позволяет достигнуть сокращения памяти примерно на 30–35 %, что является довольно неплохим результатом.

Таблица 1

Результаты тестирования предобусловливателей (время решения и число итераций)

Название и характеристики СЛАУ	Предобусловливатель	Количество процессоров	Число итераций	Общее время решения, с
M05 n = 475000 nnz = 15.567.500	ILU0	1	199	24,07
	BlockILDU0F		203	23,11
	BlockILDU0		194	22,81
	ILU0F_BCSR		201	25,90
	ILU0_BCSR		204	27,10
	BlockMinijac		653	26,33
A01_block n = 475.000 nnz = 15.567.500	ILU0	1	20	3,10
	BlockILDU0F		68	8,00
	BlockILDU0		56	6,90
	ILU0F_BCSR		20	2,86
	ILU0_BCSR		20	3,13
	BlockMinijac		93	6,67
M1 n = 2.181.550 nnz = 19.141.625	ILU0	4	4	2,50
	BlockILDU0F		4	1,26
	BlockILDU0		4	1,45
	ILU0F_BCSR		4	1,39
	ILU0_BCSR		4	1,58
	BlockMinijac		14	1,78

Примечание: n – число неизвестных (число строк), nnz – полное число ненулевых элементов в матрице, ILU0 – стандартный предобусловливатель неполного разложения, BlockILDU0 – блочный предобусловливатель ILDU0 разложения, BlockILDU0F – блочный предобусловливатель ILDU0 разложения с одинарной точностью, ILU0_BCSR – блочный предобусловливатель со стандартным ILU0 разложением, ILU0F_BCSR – блочный предобусловливатель со стандартным ILU0 разложением и одинарной точностью, BlockMinijac – мелкоблочный предобусловливатель Якоби.

Таблица 2

Результаты тестирования предобусловливателей (используемая оперативная память)

Предобусловливатель	Используемая оперативная память, Мб
ILU0	1705
BlockILDU0F	1105
BlockILDU0	1403
ILU0F_BCSR	1144
ILU0_BCSR	1487
BlockMinijac	867

Литература

1. Дерюгин Ю. Н., Зеленский Д. К., Козелков А. С. и др. Многофункциональный пакет программ ЛОГОС для расчета задач гидродинамики и тепломассопереноса на многопроцессорных ЭВМ: базовые технологии и алгоритмы: Доклад XII Международный семинар «Супервычисления и математическое моделирование», г. Саров, 2010.

2. Артемьев А. Ю., Бартенев Ю. Г., Басалов В. Г. и др. Библиотека решателей разреженных линейных систем // Труды РФЯЦ-ВНИИЭФ. Саров: РФЯЦ-ВНИИЭФ, 2004. Вып. 7. С. 80–95.

3. Intel® Math Kernel Library версии 10.1 – Документация. – Электронный ресурс: www.intel.com /cd/software/products/emca/rus/358888.html