

**МЕТОДЫ НЕПОЛНОЙ ФАКТОРИЗАЦИИ  
С ПОЛУСОПРЯЖЕННЫМИ НЕВЯЗКАМИ\*****В. П. Ильин<sup>1</sup>, С. Г. Пудов<sup>2</sup>**<sup>1</sup>*Институт вычислительной математики и математической геофизики СО РАН,  
г. Новосибирск**E-mail: ilin@sscc.ru*<sup>2</sup>*Конструкторско-технологический институт вычислительной техники СО РАН,  
г. Новосибирск**E-mail: pudov@dote.ru*

Рассматривается итерационное решение системы линейных алгебраических уравнений с несимметричными квадратными вещественными матрицами с помощью устойчивой модификации метода обобщенных сопряженных невязок и иерархического семейства алгоритмов неполного разложения матрицы на треугольные множители. Описываются особенности программной реализации алгоритмов на основе символической факторизации матриц, хранящихся в разреженном строчном формате. Приводятся результаты численных экспериментов для представительной серии модельных задач, демонстрирующих сравнительную эффективность предложенных методов.

**Введение.** Рассматривается итерационное решение системы линейных алгебраических уравнений

$$Au = f, \quad u, f \in R^N, \quad A \in R^{N, N}, \quad (1)$$

с квадратной несимметричной вещественной матрицей  $A$ . Предполагается, что матрица является положительно-определенной в смысле выполнения условий

$$(Au, u) \geq \delta \|u\|^2, \quad \delta > 0, \quad \|u\|^2 = (u, u), \quad (2)$$

для всех ненулевых вещественных векторов  $u = \{u_i\}$  размера  $N$ . Известно [1], что неравенства (2) обеспечивают для матрицы  $A$  положительную определенность ее симметричной части  $A_s = (A + A^t)/2$  ( $t$  – транспонирование матрицы), а также положительность вещественной части ее собственных чисел.

\* Работа выполнена при поддержке Российского фонда фундаментальных исследований (грант № 05-01-00487) и NWO-RFBR (грант № 047.016.008).

Предполагается, что матрица  $A$  является сильно разреженной и имеет высокий порядок, поэтому хранится в разреженном строчном формате [2], т. е. для каждой строки определено: количество ненулевых элементов, номера столбцов, в которых они расположены, и собственно значения этих ненулевых элементов. Такой подход позволяет существенно экономить память, но несколько увеличивает затраты времени на доступ к элементам матрицы. Матрицу  $A$  будем рассматривать в виде  $A = D - L - U$ , где  $D$ ,  $L$ ,  $U$  – диагональная, строго верхняя и нижняя треугольные матрицы соответственно.

Целью данной работы является исследование предложенного семейства методов предобусловливания для итерационных алгоритмов полусогранных невязок в подпространствах Крылова с различными управляющими параметрами на семействе вложенных сеток.

**Предобусловливание системы.** Вместо исходной системы (1) будем решать предобусловленную систему

$$\tilde{A}u = M^{-1}Au = M^{-1}f = \tilde{f} \quad (3)$$

с невырожденной матрицей  $M$  (предобусловливатель), которая имеет то же самое точное решение, что и исходная. Если матрица  $M$  будет «близка» к  $A$ , то новая матрица  $\tilde{A} = M^{-1}A$  будет близка к единичной матрице, что приведет к лучшей обусловленности новой системы, а значит, и более быстрому ее решению.

Таким образом, предобусловливающая матрица  $M$  должна быть близка к  $A$ , легко вычислима и легко обратима. В качестве  $M$  можно взять точное  $LU$ -разложение матрицы  $A$ , т. е.  $A = L_A U_A$ , где  $L_A$  и  $U_A$  – нижняя и верхняя треугольные матрицы. В этом случае формально достигается легкая обратимость, но такое разложение практически невозможно вычислить для реальных задач, поскольку в общем случае получаемые матрицы  $L_A$  и  $U_A$  будут иметь большую степень заполнения. Следовательно, нужно, чтобы матрица  $M$  также была сильно разреженной, что приводит к идее использовать семейство алгоритмов неполного  $LU$ -разложения  $ILU(q)$ , где  $q = 0, 1, 2, \dots$  – параметр, характеризующий уровень разложения (для симметричных матриц аналогичный алгоритм был предложен в [3]). Особенность данного подхода заключается в том, что предобусловливание производится на алгебраическом уровне с использованием портретов матриц, а не на уровне шаблонов соседств на прямоугольной сетке [4].

Для описания такого подхода введем понятие портрета разреженной матрицы  $A$ :  $P_A = \{(i, j) \mid a_{i,j} \neq 0\}$ . Без ограничения общности будем считать, что портрет матрицы симметричен, т. е. при  $a_{i,j} \neq 0$  пара индексов  $(i, j)$  также включается в  $P_A$ . Неполное  $LU$ -разложение матрицы  $A \approx M_q = L_q U_q$  производится в два этапа. На первом строится символьная  $ILU(q)$ -факторизация системы для заданного параметра  $q$ , т. е. находятся портреты треугольных матриц  $L_q$  и  $U_q$ . На втором этапе по найденным портретам вычисляются значения элементов этих матриц.

Опишем этап символьной факторизации, который проводится итеративно, шаг за шагом. При этом полагаем, что в треугольных матрицах  $L_0$  и  $U_0$ , кроме диагональных, ненулевые элементы присутствуют в тех же позициях, что и в матрице  $A$ . Другими словами,  $P_{L_0} = P_L \cup P_D$ ,  $P_{U_0} = P_U \cup P_D$ . Вычисляем портрет матрицы  $M_0 = L_0 U_0$  следующим образом: элемент  $(i, j) \in P_{M_0}$ , если  $\exists k: (i, k) \in P_{L_0}, (k, j) \in P_{U_0}$ . Полученный портрет, в силу наличия

диагональных элементов в матрицах  $L_0$  и  $U_0$ , включает в себя портрет матрицы  $A$ :  $P_{M_0} = P_{L_0 U_0} = P_A \cup P_{R_0}$ , где  $R_0$  – матрица, которая определяет различие между  $A$  и  $M_0$ . Предполагается, что  $P_{R_0} \neq 0$ , в противном случае получается полное  $LU$ -разложение матрицы  $A$ , которое соответствует значению параметра  $q=0$ , т. е. имеем символьную  $ILU(0)$ -факторизацию.

Опишем итерационный шаг построения  $ILU(q+1)$ -факторизации по имеющейся  $ILU(q)$ . Без потери общности будем считать  $q=0$ . Портреты треугольных матриц  $L_1$  и  $U_1$  строятся путем добавления к  $L_0$  и  $U_0$  позиций ненулевых элементов из матрицы  $R_0$ , т. е. все ненулевые позиции, присутствующие в верхней треугольной части  $R_0$ , добавляются к  $U_0$ , а позиции ненулевых элементов из нижней треугольной части  $R_0$  добавляются к  $L_0$ . Вычисляем портрет матрицы  $M_1 = L_1 U_1$ , причем полученный портрет в силу наличия диагональных элементов в матрицах  $L_1$  и  $U_1$  также включает в себя портрет матрицы  $A$ :  $P_{M_1} = P_{L_1 U_1} = P_A \cup P_{R_1}$ . В этом случае  $R_1$  – матрица, которая определяет различие между  $A$  и  $M_1$ .

При необходимости этот процесс можно повторять для получения следующих разбиений:  $ILU(2)$ ,  $ILU(3)$  и т. д. В пределе такой процесс должен дать полное  $LU$ -разложение исходной матрицы, т. е. возможность получения предобуславливающей матрицы  $M_n$ , «близкой» к  $A$  настолько, насколько это необходимо. Ограничением может быть объем необходимой памяти, время вычисления  $LU$ -разложения матрицы  $M_n$ , а также требования к устойчивости и точности такого разложения.

После построения символьной факторизации системы строится  $ILU$ -разложение матрицы  $A$ . Имеем  $P_{L_q U_q} = P_{L_q} \cup P_{U_q} \cup P_{R_{q+1}}$ , причем  $P_{L_q} \cup P_{U_q} \setminus P_A = \bigcup_{k=0}^q P_{R_k}$ .

$LU$ -разложение строится таким образом, чтобы

$$[L_q U_q]_{i,j} = a_{i,j}, \quad (i,j) \in P_A; \quad (4)$$

$$[L_q U_q]_{i,j} = 0, \quad (i,j) \in P_{L_q} \cup P_{U_q} \setminus P_A,$$

при этом выбирается  $[L_q]_{i,i} = 1$ ,  $i=1, \dots, n$ . Нахождение элементов матриц  $L_q$  и  $U_q$  можно производить построчно, начиная с левого верхнего угла. Поскольку  $[L_q]_{1,1} = 1$ , то из соотношений (4) можно найти элементы первой строки матрицы  $U_q$ , а именно

$$[U_q]_{1,j} = A_{1j}, \quad (1,j) \in P_A,$$

$$[U_q]_{1,j} = 0, \quad (1,j) \in P_{L_q} \cup P_{U_q} \setminus P_A.$$

Предположим, что строки матриц  $L_q$  и  $U_q$  с 1-й по  $(k-1)$ -ю уже найдены. Опишем процесс вычисления ненулевых элементов  $k$ -й строки:

$$[L_q]_{k,j} = \frac{1}{[U_q]_{j,k}} \left( a_{k,j} - \sum_{i=1}^{j-1} [L_q]_{k,i} [U_q]_{i,j} \right), \quad (k,j) \in P_A;$$

$$[L_q]_{k,j} = -\frac{\sum_{i=1}^{j-1} [L_q]_{k,i} [U_q]_{i,j}}{[U_q]_{j,k}}, \quad (k,j) \in P_{L_q} \cup P_{U_q} \setminus P_A;$$

$$[L_q]_{k,k} = 1;$$

$$[U_q]_{k,j} = a_{k,j} - \sum_{i=1}^{k-1} [L_q]_{k,i} [U_q]_{i,j}, \quad (k,j) \in P_A;$$

$$[U_q]_{k,j} = -\sum_{i=1}^{j-1} [L_q]_{k,i} [U_q]_{i,k}, \quad (k,j) \in P_{L_q} \cup P_{U_q} \setminus P_A.$$

**Метод обобщенных сопряженных невязок.** Предобусловленная система (3) решается с помощью модификации итерационного процесса, называемого методом обобщенных сопряженных невязок [5]:

$$\begin{aligned} r^0 &= f - Au^0; & p^0 &= r^0; \\ u^n &= u^{n-1} + \alpha_{n-1} p^{n-1}; & (5) \\ r^n &= r^{n-1} - \alpha_{n-1} Ap^{n-1}. \end{aligned}$$

Коэффициенты  $\alpha_k$  вычисляются из условия минимизации нормы вектора невязки  $r^n$  в подпространстве Крылова

$$K_n(r_0, A) = \text{span} \{p^0, p^1, \dots, p^{n-1}\} = \text{span} \{p^0, Ap^0, \dots, A^{n-1} p^0\}$$

по формуле

$$\alpha_n = (r^n, Ar^n) / (Ap^n, Ap^n), \quad (6)$$

а направляющие, или корректирующие, векторы  $p^n$  находятся в соответствии с [5] следующим образом:

$$p^n = r^n + \sum_{k=0}^{n-1} \beta_{n,k} p^k, \quad \beta_{n,k} = -(Ap^k, Ar^n) / (Ap^k, Ap^k). \quad (7)$$

В литературе (см., например, [6]) отмечается неустойчивость вычислений при больших  $n$  по формулам (7), представляющих собой алгоритм ортогонализации Грама – Шмидта векторов невязок. Для повышения устойчивости вычислений применим модифицированный метод Грама – Шмидта [7], описываемый вместо (7) формулами

$$p^n = r^{n,k} + \beta_{n,k} p^k, \quad \beta_{n,k} = -(Ap^k, Ar^{n,k}) / (Ap^k, Ap^k), \quad k=0,1,\dots,n-1.$$

Полученный при этом итерационный процесс будем называть методом полу-сопряженных невязок.

Критерием остановки итераций является выполнение условия

$$(r^n, r^n) < \varepsilon^2(f, f), \quad (8)$$

где  $\varepsilon \ll 1$  – задаваемая точность вычислений.

При этом векторы  $r^n$ , определяемые соотношениями (5)–(7), являются правыми полусопряженными [5], т. е. выполняются условия

$$(Ar^k, r^n) = \begin{cases} 0 & \text{при } k < n, \\ \sigma_n = (Ar^n, r^n) \neq 0 & \text{при } k = n, r^n \neq 0, \end{cases}$$

а получаемые линейно независимые векторы  $p^n$  удовлетворяют условиям

$$(A^t A p^k, p^n) = (A p^k, A p^n) = \rho_n \delta_{k,n}, \quad \rho_n = (A p^n, A p^n), \quad (9)$$

где  $\delta_{k,n}$  – символ Кронекера. Таким образом, приведенные формулы описывают длинную рекурсию, т. е. новый направляющий вектор  $p^n$  должен вычисляться через все предыдущие, причем кроме  $p^k$  для нахождения  $\beta_{n,k}$  требуется знание  $A p^k$ , что приводит к необходимости хранить все эти векторы.

Отметим, что если  $A$  – симметричная, положительно-определенная матрица, то  $\beta_{n,k} = 0$ ,  $k = 0, \dots, n-1$ , а это приводит к тому, что формулы (5)–(7) описывают метод минимальных (сопряженных) невязок [6].

Рассмотренный метод обобщенных сопряженных невязок по своим итерационным свойствам аналогичен более известному методу обобщенных минимальных невязок [6]. Однако в последнем для нахождения коэффициентов  $\beta_{n,k}$  необходимо решать на каждой итерации вспомогательную систему уравнений со специально формируемой матрицей Хессенберга.

**Особенности программной реализации.** Первый момент касается расчетов по формулам (7), которые при больших  $n$  численно неустойчивы [7]. С алгебраической точки зрения формулы (7) преобразуют линейно независимые векторы  $r^0, r^1, \dots, r^n$  в обладающие свойством (9) векторы  $p^0, p^1, \dots, p^n$  с помощью  $A^t A$ -ортогонализации Грама – Шмидта. Поэтому при реализации они могут быть заменены формулами устойчивой модифицированной ортогонализации Грама – Шмидта из [7]:

$$\begin{aligned} p^{n,0} &= r^{n-1}; & A p^{n,0} &= A r^{n-1}; \\ \beta_{n,l} &= (A p^{n,l}, A p^l) / (A p^l, A p^l); \\ p^{n,l+1} &= p^{n,l} + \beta_{n,l} p^l; \\ A p^{n,l+1} &= A p^{n,l} + \beta_{n,l} A p^l; \\ p^k &= p^{n,k}; \\ A p^k &= A p^{n,k}. \end{aligned} \quad (10)$$

Отметим, что при абсолютно точных вычислениях величины  $\beta_{n,k}$  и  $p^n$ , реализуемые по формулам (7) и (10), совпадают.

Второй момент касается затрат ресурсов памяти и процессора: с ростом  $n$  увеличивается как объем требуемой оперативной памяти (уже отмечалось, что необходимо сохранять по  $2n$  векторов длины  $N$  на каждой  $n$ -й итерации), так и объем вычислений, поскольку направляющий вектор с номером  $n$  находится через все предыдущие. Здесь для экономии памяти возможны два варианта.

1. По достижении номером текущей итерации некоторого значения  $n = sm_1$ ,  $s = 1, 2, \dots$ , кратного заданному числу  $m_1$ , производить полный перезапуск алгоритма, т. е. использовать текущее решение в качестве начального приближения и повторять программу заново с нулевого шага (5), когда очередной вектор невязки определяется не из рекуррентных соотношений, а непосредственно по формуле  $r^n = f - Au^n$ .

2. При проведении  $n$ -й итерации обеспечить  $A$ -ортогональность вектора  $r^n$  не всем предыдущим невязкам  $r^k$ , а только  $m_2$  последним  $r^{n-1}, \dots, r^{n-m_2}$ , при этом следует, начиная с итерации  $m_2$ , делать сдвиг направляющих векторов. Тогда самый старый вектор «выкидывается» из памяти, а на его место помещается следующий. Таким образом, новое приближение после достижения алгоритмом своего «потолка» строится по последним  $m_2$  векторам.

В любом случае скорость сходимости алгоритма замедлится по сравнению с первоначальным вариантом и будет существенно снижаться при уменьшении значений  $m_1$  и  $m_2$ , поскольку минимизация нормы невязки осуществляется на подпространствах Крылова меньшей размерности. Далее максимально возможное количество одновременно хранимых в памяти направляющих векторов  $p^k$ ,  $k = 0, \dots, n-1$ , будем называть уровнем алгоритма.

Для построения портретов матриц  $L_k$  и  $U_k$  была разработана специальная программа на языке C++, которая по исходной матрице строит соответствующие портреты  $L_k$  и  $U_k$  для  $k = 0, 1, 2, 3$ . Выбор языка C++ обусловлен возможностью использования ассоциативных контейнерных типов данных `set` и `map` стандартной библиотеки шаблонов STL (Standard Template Library).

Основная программа реализована на языке Фортран. Кроме исходной матрицы она считывает также портреты матриц  $L_k$  и  $U_k$ , вычисляет их элементы методом полусопряженных невязок с устойчивой ортогонализацией Грама – Шмидта (9). Программа позволяет организовывать работу как в режиме перезапуска, так и в режиме сдвига векторов, а также в любой их комбинации. Отдельно задается уровень алгоритма (`level`), максимальное количество итераций ( $n_{\max}$ ) и количество итераций (`restart`) между перезапусками. Это позволяет гибко организовывать процесс вычислений.

Например, пусть  $n_{\max} = \text{restart} = 1000$ ,  $\text{level} = 100$ . В этом случае перезапуск не происходит, а на каждой итерации с номером больше 100 происходит сдвиг направляющих векторов. Если  $n_{\max} = 1000$ , а  $\text{restart} = \text{level} = 100$ , то после каждой 100-й итерации программа перезапускается, а сдвиг не происходит. И наконец, если  $n_{\max} = 1000$ ,  $\text{restart} = 500$ ,  $\text{level} = 100$ , то на 100-й итерации алгоритм заполняет векторами всю выделенную память, после чего на последующих итерациях до 500-й производится сдвиг направляющих векторов. На 500-й итерации происходит полный перезапуск алгоритма, и все начинается заново (если только задача не будет решена раньше).

**Результаты численных экспериментов.** Эффективность разработанных методов показана на примере решения несимметричных систем сеточ-

Т а б л и ц а 1

$N = 32$					$N = 64$				
$p = q = r$	$ILU(0)$	$ILU(1)$	$ILU(2)$	$ILU(3)$	$p = q = r$	$ILU(0)$	$ILU(1)$	$ILU(2)$	$ILU(3)$
-16	45	42	42	42	-16	105	99	98	98
-8	60	58	58	58	-8	127	123	122	122
-4	70	67	66	66	-4	138	133	131	131
0	79	74	72	72	0	147	140	138	137
4	84	78	76	75	4	159	150	146	144
8	82	77	74	74	8	160	151	147	145
16	78	74	72	71	16	157	148	144	143

ных семиточечных уравнений, аппроксимирующих задачу Дирихле для линейного диффузионно-конвективного уравнения

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + p \frac{\partial u}{\partial x} + q \frac{\partial u}{\partial y} + r \frac{\partial u}{\partial z} = f(x, y, z), \quad (11)$$

$$(x, y, z) \in \Omega, \quad u|_{\Gamma} = g(x, y, z),$$

в единичном кубе  $\Omega = [0, 1]^3$  на различных кубических сетках с шагом  $h = 1/N$ .

Дискретизация задачи проводилась с помощью монотонной схемы экспоненциального типа [8]. Функции  $f$  и  $g$  в (11) брались из условия, чтобы точное решение  $u(x, y, z)$  было тождественно равно единице. В качестве начального приближения выбиралась функция  $u^0(x, y, z) = x^2 + y^2 + z^2$ , а точность итераций (8) полагалась  $\varepsilon = 10^{-7}$ .

В табл. 1 приводится количество итераций для численного решения задачи (11) при различных значениях  $p, q, r$  и для различных параметров предобусловливания. Число шагов в каждом направлении  $N = 32$  и  $N = 64$ . Все эксперименты проводились без перезапусков и сдвигов.

**Заключение.** Результаты проведенных экспериментов позволяют сделать следующие выводы.

1. Для модельной задачи число итераций уменьшается незначительно при переходе от предобусловливания  $ILU(0)$  к  $ILU(3)$ .

2. Общее время решения задачи и время построения предобусловливателя в виде неполного разложения матрицы представлены в табл. 2, где  $N$  – размерность задачи,  $n$  – размер матрицы,  $v$  – количество ненулевых элементов в верхнетреугольной матрице. В столбцах  $ILU(k)$  данные приведены в формате: время на  $ILU$ -факторизацию (с)/общее время решения задачи (с)/количество итераций. Эксперименты проводились на системе из четырех процессо-

Т а б л и ц а 2

$N$	$n$	$\nu$	$ILU(0)$	$ILU(1)$	$ILU(2)$	$ILU(3)$
32	297910	864900	0,26/2,39/84	0,5/2,9/78	1,1/4,3/76	3,7/8,5/75
64	250047	738234	2,3/94/159	5,3/93/150	11,5/117/146	34/121/144

ров Itanium2 (1,5 ГГц, кэш 4 Мбайт, 12 Гбайт оперативной памяти) под управлением операционной системы RedHat Enterprise Linux v.3.

3. При дроблении сетки в 2 раза число итераций увеличивается также примерно в 2 раза.

Это позволяет рекомендовать данный метод предобусловливания для небольших значений  $k = 0, 1, 2$ , когда реализация каждой итерации наиболее экономична.

#### СПИСОК ЛИТЕРАТУРЫ

1. **Фаддеев Д. К., Фаддеева В. Н.** Вычислительные методы линейной алгебры. М.: Физматгиз, 1960.
2. **Писсанецки С.** Технология разреженных матриц. М.: Мир, 1988.
3. **Gustafsson I. A.** Modified incomplete Cholesky (MIC) methods // Preconditioning Methods: Analysis and Applications /Ed. D. J. Evans. N. Y.: Gordon and Breach Science Publishers, 1983. P. 265.
4. **Ильин В. П.** Методы неполной факторизации для решения алгебраических систем. М.: Физматлит, 1995.
5. **Eisenstat S. C., Elman H. C., Schultz M. H.** Variational iterative methods for nonsymmetric systems of linear equations // SIAM Journ. Numer. Anal. 1983. **20**, N 3. P. 345.
6. **Saad Y.** Iterative Methods for Sparse Linear Systems. N. Y.: PWS Publishing, 1996.
7. **Ильин В. П.** Численный анализ. Ч. 1. Новосибирск: Изд-во ИВМиМГ СО РАН, 2004.
8. **Andreeva M. Yu., Il'in V. P., Itskovich E. A.** Two solvers for nonsymmetric SLAE // Bull. of the Novosibirsk Computing Center. Ser. Numer. Anal. 2004. N 12. P. 1.

*Поступила в редакцию 3 ноября 2006 г.*