УДК 519.63

MPI+OpenMP РЕАЛИЗАЦИЯ МЕТОДА СОПРЯЖЕННЫХ ГРАДИЕНТОВ С ПРЕДОБУСЛОВЛИВАТЕЛЯМИ БЛОЧНОГО НЕПОЛНОГО ОБРАТНОГО ТРЕУГОЛЬНОГО РАЗЛОЖЕНИЯ ВТОРОГО И ПЕРВОГО ПОРЯДКА

О. Ю. Милюкова
(ИПМ им. М. В. Келдыша РАН, г. Москва)

Рассматривается новый предобусловливатель для решения систем линейных алгебраических уравнений с симметричной положительно определенной разреженной матрицей — предобусловливатель блочного неполного обратного треугольного разложения первого порядка *по значению*. Предлагается способ применения (MPI+OpenMP)-технологии для построения и обращения предобусловливателей блочного неполного обратного треугольного разложения второго и первого порядка по значению. При применении (MPI+OpenMP)-технологии число блоков в этих предобусловливателях кратно числу используемых процессоров и числу используемых потоков. Проводится сравнение времени решения задач с использованием исходной MPI-технологии и гибридной (MPI+OpenMP)-технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse. Сравнивается время решения этих задач методом сопряженных градиентов с предобусловливателями блочного неполного обратного треугольного разложения второго и первого порядка по значению.

Ключевые слова: разреженные матрицы, неявное блочное предобусловливание, неполное треугольное разложение Холецкого, параллельное предобусловливание, метод сопряженных градиентов.

Введение

Рассмотрим задачу приближенного решения системы линейных алгебраических уравнений (СЛАУ) большого размера

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

с симметричной положительно определенной разреженной матрицей общего вида

$$A = A^T > 0.$$

Проблема построения эффективных численных методов решения СЛАУ (1) сохраняет свою актуальность, так как во многих важных прикладных областях продолжают возникать новые постановки таких задач. При этом наблюдается тенденция к росту размера матриц n, увеличению их заполненности ненулевыми элементами, а также ухудшению обусловленности.

В настоящей работе для решения СЛАУ (1) большого размера применяется предобусловленный метод сопряженных градиентов (CG), итерации которого осуществляются до выполнения условия

$$\|\mathbf{b} - A\mathbf{x}_k\| \le \varepsilon \|\mathbf{b} - A\mathbf{x}_0\|, \text{ где } 0 < \varepsilon \ll 1. \tag{2}$$

Для предобусловливания используется блочное неполное обратное разложение Холецкого BIIC (Block Incomplete Inverse Cholesky) [1, 2] в сочетании с приближенным стабилизированным треугольным разложением второго порядка *по значению* IC2S (τ) ($0 < \tau \ll 1$) [3] — предобусловливание BIIC-IC2S (τ) [4]. В работе также используется новое предобусловливание — BIIC в сочетании с

приближенным треугольным разложением с отсечением по параметру $0 < \tau \ll 1$ первого порядка IC1 (τ) (см., например, [5]) — предобусловливание BIIC-IC1 (τ) [6]. Предобусловливание IC1 (τ) имеет ограниченную область применимости [7]. Предобусловливание BIIC-IC1 (τ) будем также называть блочным неполным обратным треугольным разложением первого порядка, а предобусловливание BIIC-IC2S (τ) — блочным неполным обратным треугольным разложением второго порядка.

В работе [8] в качестве предобусловливателя предложена и исследована блочная версия неполного обратного разложения Холецкого BIIC в сочетании с приближенным треугольным разложением второго порядка по значению IC2 (τ) [3] — BIIC-IC2 (τ). В BIIC-IC2S (τ) [4], в отличие от BIIC-IC2 (τ), для построения предобусловливателя внутри блока используется IC2S (τ)-разложение. Методы сопряженных градиентов с предобусловливателями BIIC-IC2 (τ) и BIIC-IC2S (τ) были эффективно реализованы на параллельных архитектурах с распределенной памятью [8, 4].

В настоящей работе основное внимание уделяется применению OpenMP-технологии для параллельной реализации алгоритмов метода сопряженных градиентов с предобусловливателями BIIC-IC2S (τ), BIIC-IC1 (τ) при проведении расчетов на многопроцессорной вычислительной системе. Далее параметр τ в названиях этих предобусловливателей может быть опущен.

Проблеме использования высокоуровнего параллелизма (мелкозернистое или распараллеливание алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. Неявный факторизованный предобусловливатель имеет вид $B = LL^T \approx A^{-1}$ или $B = LDL^T \approx A^{-1}$, где L — нижнетреугольная матрица, а D — диагональная матрица. Обращение предобусловливателя $B = LL^T$ сводится к решению двух треугольных систем для нахождения вектора поправки в предобусловленном методе сопряженных градиентов. В работах [9—12] было предложено использование нескольких итераций метода Якоби или блочного метода Якоби для решения треугольных систем при применении предобусловливания неполного треугольного разложения, что позволило использовать высокий уровень параллелизма. В работе [13] предлагается безытерационный способ применения (MPI+OpenMP)-технологии при обращении неявного факторизованного предобусловливателя, т. е. решении двух треугольных систем. В работе [14] предлагается новый итерационный алгоритм неполных IC(0)-, IC(1)-, IC(2)-разложений, в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно.

Заметим, что использование явных предобусловливателей позволяет эффективно применять (MPI+OpenMP)-технологию для параллельного решения СЛАУ (1) предобусловленным методом сопряженных градиентов (см., например, [15—17]).

В работах [18, 19] предложены два безытерационных способа применения (MPI+OpenMP)-технологии при построении и обращении неявного факторизованного предобусловливателя блочного метода Якоби в сочетании с IC(0) (неполным треугольным разложением Холецкого без заполнения) и IC1(τ), в которых обращение предобусловливателя сводится к решению двух треугольных систем. Эти способы основаны на переупорядочении узлов сетки типа DDO [20] внутри каждой подобласти или уменьшении шаблона разреженности матрицы A при построении предобусловливателя.

Безытерационный способ применения (MPI+OpenMP)-технологии построения и обращения предобусловливателей BIIC-IC1 и BIIC-IC2S, подробно рассматриваемый в настоящей работе, впервые был предложен в работе [6]. В этой работе был также предложен безытерационный способ применения (MPI+OpenMP)-технологии построения и обращения предобусловливателя BIIC-IC1, основанный на переупорядочении узлов сетки типа DDO внутри каждой подобласти, полученной при разбиении области расчета для параллельной реализации с использованием только MPI, и отсечении по структуре ничтожно малого числа элементов матрицы предобусловливания при ее построении.

В формуле (1) предполагается, что матрица A уже переупорядочена, а вместо $A_P = P\widetilde{A}P^T$ (P — матрица перестановки, $\widetilde{A}$ — матрица коэффициентов исходной задачи) стоит A. В настоящей работе применяются переупорядочения, уменьшающие среднюю ширину ленты матрицы, а именно предложенные в работах [21, 22], являющиеся обобщением упорядочения [4]. Подход, предложенный в этих работах, позволяет одновременно произвести разбиение области расчета на подобласти. Будем также предполагать, что матрица A отмасштабирована, т. е. ее диагональные элементы равны единице. Это достигается с использованием формулы $A_{SP} = D_{A_P}^{-1/2} A_P D_{A_P}^{-1/2}$, где D_{A_P} — диагональная

часть матрицы A_P. Далее вместо A_{SP} будем использовать обозначение A, предполагая, что переупорядочение и масштабирование уже выполнены.

Итак, в настоящей работе рассматривается безытерационный способ применения (MPI+OpenMP)-технологии для построения и обращения предобусловливателя BIIC-IC2S и нового предобусловливателя BIIC-IC1 для решения СЛАУ с симметричной положительно определенной матрицей. При этом в предобусловливателе BIIC вместо p используется pm блоков, где p — число процессоров, m — число потоков. OpenMP-технологии применяются для всех строк матрицы на этапах построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателями BIIC-IC2S и BIIC-IC1 с использованием MPI- и (MPI+OpenMP)-подходов на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse [23]. Время решения этих задач методом сопряженных градиентов с предобусловливателями BIIC-IC2S и BIIC-IC1 также сравнивается между собой.

Предобусловленный метод сопряженных градиентов

Пусть требуется решить СЛАУ (1). Алгоритм 1 предобусловленного метода сопряженных градиентов (см., например, [24]) имеет следующий вид:

$$
\begin{aligned}
&\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0; \quad \mathbf{p}_0 = \mathbf{w}_0 = H\mathbf{r}_0; \quad \gamma_0 = \mathbf{r}_0^T \cdot \mathbf{p}_0;\\
&\textit{для } k = 0, \ldots, \quad \textit{пока } \mathbf{r}_k^T \cdot \mathbf{r}_k > \varepsilon^2 \left(\mathbf{r}_0^T \cdot \mathbf{r}_0\right), \ \textit{выполнять}\\
&\quad \mathbf{q}_k = A\mathbf{p}_k; \qquad \alpha_k = \gamma_k / \left(\mathbf{p}_k^T \cdot \mathbf{q}_k\right);\\
&\quad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k; \qquad \mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k; \qquad \mathbf{z}_{k+1} = H\mathbf{r}_{k+1};\\
&\quad \gamma_{k+1} = \mathbf{r}_{k+1}^T \cdot \mathbf{z}_{k+1}; \qquad \beta_k = \gamma_{k+1}/\gamma_k; \qquad \mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k,
\end{aligned}
$$

где $0 < \varepsilon \ll 1$; H — матрица предобусловливания ($H \approx A^{-1}$). Этот алгоритм использует операции умножения разреженных матриц на вектор, вычисления скалярных произведений, элементарные векторные операции, а также вычисление $\mathbf{z}_{k+1} = H\mathbf{r}_{k+1}$, $\mathbf{p}_0 = \mathbf{w}_0 = H\mathbf{r}_0$. Принципиальная возможность эффективной параллельной реализации всех операций, кроме вычисления $H\mathbf{r}_{k+1}$, $H\mathbf{r}_0$, не вызывает сомнений даже при использовании большого числа процессоров и/или применении OpenMP-технологии.

Предобусловливатели блочного неполного обратного разложения Холецкого в сочетании с IC2S(τ) и IC1(τ)

Приведем краткое описание блочной версии алгоритма неполного обратного разложения Холецкого BIIC [1, 2]. Пусть матрица A размерами $n \times n$ отмасштабирована, переупорядочена и разбита на блоки, причем на блочной диагонали расположены p квадратных блоков размерами $n_s \times n_s$, $1 \le s \le p$. Для каждого s-го диагонального блока определим базисное множество индексов как $\{k_{s-1}+1, \ldots, k_s\}$, где $k_s = n_1 + \ldots + n_s$ ($k_0 = 0$; $k_p = n$), и введем "перекрывающиеся" множества индексов: $\{j_s(1), \ldots, j_s(m_s - n_s)\}$, $j_s(l) \le k_{s-1}$ ($l = 1, \ldots, m_s - n_s$). Для каждого s такое множество включает индексы, не превышающие k_{s-1}, которые оказываются наиболее существенно связанными с s-м базисным множеством индексов, например, в смысле графа смежности разреженной матрицы A. Здесь $m_s \ge n_s$, m_s — размер s-го расширенного блока, причем $m_1 = n_1$. Для каждого s-го диагонального блока определим прямоугольные матрицы

$$
V_s = \left(\mathbf{e}_{j_s(1)} \mid \ldots \mid \mathbf{e}_{j_s(m_s - n_s)} \mid \mathbf{e}_{k_{s-1}+1} \mid \ldots \mid \mathbf{e}_{k_s}\right), \tag{3}
$$

столбцы которых являются единичными n-векторами. Пусть $\overline{U}_s$ — правый множитель в точном треугольном разложении Холецкого расширенной диагональной $(m_s \times m_s)$-подматрицы:

$$A_s = V_s^T A V_s = \overline{U}_s^T \overline{U}_s, \qquad s = 1, \ldots, p. \tag{4}$$

Предобусловливатель блочного неполного обратного разложения Холецкого (ВИС) имеет вид [1, 2]

$$H = \sum_{s=1}^{p} V_s \overline{U}_s^{-1} \begin{pmatrix} 0 & 0 \\ 0 & I_{n_s} \end{pmatrix} \overline{U}_s^{-T} V_s^T. \tag{5}$$

В работах [1, 2] показано, что ВИС-предобусловливатель обладает свойством K-оптимальности.

В формуле (5) для каждого $s = 1, \ldots, p$ для аппроксимации $(m_s \times m_s)$-подматриц $A_s = V_s^T A V_s$ заменим точное разложение Холецкого (4) соответствующим приближенным стабилизированным треугольным разложением второго порядка по значению IC2S (τ) [3]:

$$V_s^T A V_s = U_s^T U_s + U_s^T R_s + R_s^T U_s - S_s.$$

Здесь U_s — верхнетреугольные матрицы; R_s — строго верхнетреугольные матрицы, $\|R_s\| = O(\tau)$; $\|S_s\| = O(\tau^2)$; $0 < \tau \ll 1$ — порог отсечения; матрицы V_s определены по формуле (3). Напомним, что перед построением матрицы предобусловливания IC2S (τ) необходимо выполнить масштабирование матрицы A (см. выше).

Определим предобусловливатель ВИС-IC2S формулой [4]

$$\widetilde{H} = \sum_{s=1}^{p} V_s U_s^{-1} \begin{pmatrix} 0 & 0 \\ 0 & I_{n_s} \end{pmatrix} U_s^{-T} V_s^T. \tag{6}$$

Алгоритм вычисления элементов матриц U_s и R_s приведен, например, в работах [3, 6]. При вычислении u_{ii} — диагональных элементов матрицы U_s (u_{ii} зависят от s) — необходимо извлекать квадратный корень из числа, которое определяется в процессе вычисления элементов строк этих матриц с номерами, не превышающими i (для каждого s). Доказано, что в случае вычисления предобусловливателя IC2S (τ) не может возникнуть ситуации, когда вычисленное подкоренное выражение отрицательно [3]. То есть метод сопряженных градиентов с предобусловливанием IC2S (τ) является безотказным для любой симметричной положительно определенной матрицы A. Метод сопряженных градиентов с предобусловливанием ВИС-IC2S (τ) тоже является безотказным.

Теперь в формуле (5) для каждого $s = 1, \ldots, p$ для аппроксимации $(m_s \times m_s)$-подматриц $A_s = V_s^T A V_s$ заменим точное разложение Холецкого (4) соответствующим приближенным треугольным разложением первого порядка по значению IC1 (τ):

$$V_s^T A V_s = \widehat{U}_s^T \widehat{U}_s - E_s,$$

где $\widehat{U}_s$ — верхнетреугольные матрицы; $\|E_s\| = O(\tau)$. Один из первых алгоритмов, в котором за основу построения матрицы предобусловливания берется точный алгоритм треугольной факторизации, а на его определенных этапах вносится отсечение возникающих элементов матриц, малых относительно порога, зависящего от τ, опубликован в работе [5]. Перед построением матрицы предобусловливания IC1 (τ) тоже необходимо обеспечить, чтобы матрицы $A_s = V_s^T A V_s$ были отмасштабированы.

Определим предобусловливатель ВИС-IC1 формулой

$$\widehat{H} = \sum_{s=1}^{p} V_s \widehat{U}_s^{-1} \begin{pmatrix} 0 & 0 \\ 0 & I_{n_s} \end{pmatrix} \widehat{U}_s^{-T} V_s^T. \tag{7}$$

Алгоритм вычисления элементов матрицы $\widehat{U}_s$ приведен, например, в работах [6, 19]. Метод сопряженных градиентов с предобусловливанием ВИС-IC1 (τ) имеет ограниченную область применимости, в ряде случаев для безотказности требуется чрезмерное уменьшение параметра τ.

Алгоритмы параллельной реализации

Пусть матрица A переупорядочена, отмасштабирована и разбита на p блоков. В настоящей работе, не ограничивая общности, в предобусловливателях BIIC-IC2S и BIIC-IC1 будем использовать перекрытие (*налегание*) с шириной $q = 1$. Перед вычислением матрицы предобусловливания на каждом процессоре с номером s ($1 \le s \le p$) строятся матрицы $A_s = V_s^T A V_s$ размерами $m_s \times m_s$; для этого осуществляются необходимые пересылки строк матрицы A.

Параллельная реализация вычисления предобусловливателей BIIC-IC2S, BIIC-IC1, определенных по формулам (6), (7), с использованием только MPI не представляет труда. Для вычисления элементов матриц U_s и $\widehat{U}_s$ используются алгоритмы из работ [3] или [19], пересылок не требуется. Так же, как в работе [4], матрицы U_s^T и $\widehat{U}_s^T$ не вычисляются.

Обращение предобусловливателя $\mathbf{z} = \widetilde{H}\mathbf{r}$, где $\widetilde{H}$ определено по (6), происходит следующим образом [4]. Перед началом вычислений элементы вектора $\mathbf{r}_s$ с индексами $j_s(1), \ldots, j_s(m_s - n_s)$ должны быть переданы на процессор с номером s. Затем на каждом процессоре с номером s вычисляются

$$\mathbf{z}_s = U_s^{-1} \begin{pmatrix} 0 & 0 \\ 0 & I_{n_s} \end{pmatrix} U_s^{-T} \mathbf{r}_s.$$

Используется способ обращения матриц U_s^T, предложенный в [4]. После вычисления $\mathbf{z}_s$ следует выполнить необходимые пересылки элементов вектора $\mathbf{z}_s$ с индексами $j_s(1), \ldots, j_s(m_s - n_s)$ на другие процессоры. Аналогично происходит обращение предобусловливателя BIIC-IC1.

Рассмотрим способ параллельной реализации этапов построения и обращения предобусловливателей с использованием (MPI+OpenMP)-технологии. Разобъем всю область расчета сразу на pm подобластей, где p — число используемых процессоров, m — число используемых потоков. На процессоре с номером s будут производиться вычисления в "большой" подобласти с номером s, состоящей из подобластей с номерами $t = (s-1)m + 1, \ldots, sm$, полученных при разбиении. При использовании (MPI+OpenMP)-технологии число блоков в предобусловливателях BIIC-IC2S и BIIC-IC1 равно pm.

Для построения предобусловливателей BIIC-IC2S, BIIC-IC1 на каждом процессоре с номером s сначала создаются m матриц $A_t = V_t^T A V_t$, где $t = (s-1)m + 1, \ldots, sm$. При этом выполняются необходимые пересылки строк матрицы A. Затем по отмасштабированным матрицам A_t на процессоре с номером s ($s = 1, \ldots, p$) с использованием алгоритмов из работ [3] или [19] строятся матрицы U_t или $\widehat{U}_t$ ($t = (s-1)m + 1, \ldots, sm$). При этом для каждого t вычисления элементов матриц U_t или $\widehat{U}_t$ выполняются в своем потоке, все рекурсивные вычисления при построении матриц U_t или $\widehat{U}_t$ выполняются внутри потоков. В программе необходимо задать число *внутренних* блоков с налеганием, равное m, и инициализировать число нитей, совпадающее с числом внутренних блоков. В настоящей работе на этапе построения предобусловливателей для циклов по $t1 = 1, \ldots, m$ используется директива `do` с опцией `schedule static`, внутри циклов по $t1$ осуществляется построение матриц U_t или $\widehat{U}_t$.

При построении матриц A_t ($t = (s-1)m + 1, \ldots, sm$) на каждом процессоре с номером s ($s = 1, \ldots, p$) тоже используются OpenMP-технологии с числом нитей m. В настоящей работе на этапе построения матриц A_t для внешних циклов по $t1 = 1, \ldots, m$ используется директива `do` с опцией `schedule static`.

Обращение предобусловливателя BIIC-IC2S с использованием формулы

$$\mathbf{z}_t = U_t^{-1} \begin{pmatrix} 0 & 0 \\ 0 & I_{n_t} \end{pmatrix} U_t^{-T} \mathbf{r}_t, \qquad t = (s-1)m + 1, \ldots, sm, \qquad s = 1, \ldots, p \tag{8}$$

происходит следующим образом. Перед началом вычислений элементы векторов $\mathbf{r}_t$ для $t = (s-1)m + 1, \ldots, sm$, необходимые для расчетов и хранящиеся в памяти других процессоров, должны быть переданы на процессор с номером s. Для каждого $t = (s-1)m + 1, \ldots, sm$ вычисления по формуле (8) выполняются в своем потоке. В настоящей работе при вычислении $\mathbf{z}_t$

для выполнения циклов по $t1 = 1, \ldots, m$ используется директива `do` с опцией `schedule static`. Применяется способ обращения матриц U_t^T, предложенный в [4]. После вычисления $\mathbf{z}_t$ для всех $t = (s-1)\,m+1, \ldots, sm$, $s = 1, \ldots, p$, следует выполнить необходимые пересылки. Аналогично происходит обращение предобусловливателя BIIC-IC1 с использованием (MPI+OpenMP)-технологии.

Заметим, что для случая $p = 1$ выше описан способ параллельной реализации этапов построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1 для m блоков с налеганием с применением OpenMP-технологии.

Вычисление элементов вектора $\mathbf{q}_k = A\mathbf{p}_k$, где k — номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, выполняется следующим образом [15, 16]. Матрица A хранится в памяти в распределенном CRS-формате, содержит верхнюю и нижнюю треугольные матрицы. Сначала производится пересылка значений компонент вектора $\mathbf{p}_k$, необходимых для вычисления части вектора $\mathbf{q}_k$ на рассматриваемом процессоре с номером s ($s = 1, \ldots, p$), хранящихся в памяти других процессоров. В случае применения OpenMP-технологии для вычисления $(\mathbf{q}_k)_i$ (компонент вектора $\mathbf{q}_k$) на каждом процессоре для своей группы индексов i для выполнения цикла по i в настоящей работе используется директива `do` с опцией `schedule static`. Заметим, что, как показали расчеты задач методом CG с предобусловливанием Якоби в работах [15, 16], использование параметров `dynamic`; `guided`; `guided`, `6` в этой опции, как правило, не позволяет ускорить вычисления по сравнению с использованием параметра `static`.

MPI-реализация вычислений векторных операций и скалярных произведений в алгоритме 1 хорошо известна. При применении (MPI+OpenMP)-подхода для вычислений векторных операций и частичных сумм в скалярных произведениях в настоящей работе использовалась директива `do` с опцией `schedule static`.

Результаты расчетов

В настоящем разделе метод сопряженных градиентов с предобусловливанием BIIC-IC2S будем называть BIIC2S-CG, метод сопряженных градиентов с предобусловливанием BIIC-IC1 будем называть BIIC1-CG. Программы, реализующие применение методов BIIC2S-CG, BIIC1-CG для решения СЛАУ (1), были написаны на языке FORTRAN 90 с использованием (MPI+OpenMP)-технологии. Расчеты проводились на многопроцессорном вычислительном кластере K60, установленном в ЦКП ИПМ им. М. В. Келдыша РАН. В состав кластера K60 входят 85 вычислительных узлов, содержащих 2 380 ядер (по 28 ядер на модуль). Каждый узел представляет собой двухпроцессорный сервер с процессорами Intel Xenon E5-2690 v4. Суммарная пиковая производительность кластера — 80,9 Тфлопс.

Тестирование и сравнение методов производились с помощью решения модельной задачи — разностной задачи Дирихле для уравнения Пуассона в единичном квадрате на ортогональной сетке с $n = 1\,048\,576$. Использовалась стандартная 5-точечная аппроксимация лапласиана (имя матрицы `5_1048576`). Для тестирования рассматриваемых параллельных методов использовались также некоторые матрицы из коллекции разреженных матриц SuiteSparse [23]. Перечислим имена используемых тестовых матриц и укажем источники их происхождения:

`apache2` — трехмерная конечно-разностная схема;

`parabolic_fem` — уравнение диффузии-конвекции с постоянным переносом;

`ecology2` — приложение теории электрических цепей к задаче передачи генов;

`boneS01` — модель трубчатой кости;

`cfd2` — вычислительная гидродинамика (уравнение давления).

В табл. 1 приведены некоторые свойства этих матриц: NZA — число ненулевых элементов матрицы A; Id — количество строк без диагонального преобладания; Ip — количество положительных внедиагональных элементов; $nz_{\min}$, $nz_{\max}$ — минимальное и максимальное числа ненулевых элементов в строках матрицы A; значения $Cond\,(A_0)$, где $A_0 = (D_A)^{-1/2}\,A\,(D_A)^{-1/2}$ — матрица системы уравнений после масштабирования, взяты из работы [25].

Таблица 1

Свойства некоторых матриц из коллекции разреженных матриц SuiteSparse

Матрица A	n	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond\,(A_0)$
`apache2`	715 176	4 817 870	2	0	4	8	$0{,}12 \cdot 10^7$
`parabolic_fem`	525 825	3 674 625	0	1 048 576	3	7	$0{,}20 \cdot 10^6$
`ecology2`	999 999	4 995 991	1 124	0	3	5	$0{,}63 \cdot 10^8$
`boneS01`	127 224	5 516 602	127 222	2 064 830	12	67	$0{,}13 \cdot 10^8$
`cfd2`	123 440	3 085 406	123 440	936 464	8	30	$0{,}15 \cdot 10^7$

Модельную задачу далее будем называть задачей 1. Задачу с матрицей `parabolic_fem` будем называть задачей 2, задачи с матрицами `apache2`, `ecology2` — соответственно задачами 3, 4, а с матрицами `boneS01`, `cfd2` — задачами 5, 6.

Решалось уравнение $A\mathbf{x} = \mathbf{b}$, где A — отмасштабированная матрица ($A = A_0$) с единичной правой частью ($b_i \equiv 1$), начальное приближение $\mathbf{x}_0 \equiv 0$, счет продолжался до выполнения условия (2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета на p подобластей, а также на pm подобластей (при применении (MPI+OpenMP)-технологии) применялся алгоритм [21]. Использовалось налегание шириной $q = 1$. В качестве параметра отсечения при решении всех задач методом BIIC2S-CG использовалось значение $\tau = 0{,}01$. При решении задач 1—4 методом BIIC1-CG использовалось значение $\tau = 0{,}01$, а задачи 5 — $\tau = 0{,}005$, что было продиктовано требованием безотказности метода BIIC1-CG при решении этой задачи. При решении задачи 6 методом BIIC1-CG для безотказности метода необходимо использовать чрезмерно малое значение τ, поэтому решение этой задачи данным методом не проводилось.

В табл. 2—6 приведены число итераций и время счета методами BIIC2S-CG и BIIC1-CG тестовых задач 1—5 при применении для параллельной реализации MPI- и (MPI+OpenMP)-технологий. В табл. 7 приводятся результаты расчетов методом BIIC2-CG тестовой задачи 6 с применением этих же технологий. При применении (MPI+OpenMP)-технологии расчеты проводились с использованием 3, 4, 6, 8, 12, 16 нитей. В табл. 2—7 приведены результаты, оптимальные по числу нитей для каждого числа процессоров p с точки зрения времени вычислений, и соответствующие им значения числа использованных нитей. Приведены также коэффициенты ускорения счета μ благодаря применению OpenMP-технологии на том же числе процессоров. Под временем вычислений в табл. 2—7 подразумевается время счета итерационного процесса в сумме со временем вычисления предобусловливателя.

На рис. 1—6 приведены зависимости времени счета t_p тестовых задач от числа процессоров в логарифмическом масштабе с использованием MPI- и (MPI+OpenMP)-технологий для методов BIIC2S-CG и BIIC1-CG.

Таблица 2

Число итераций и время счета (в секундах) методами BIIC2S-CG и BIIC1-CG задачи с матрицей `5_1048576` на p процессорах без использования и с использованием OpenMP-технологии

p	BIIC2S-CG						BIIC1-CG					
	MPI		MPI+OpenMP			μ	MPI		MPI+OpenMP			μ
	Число итераций	Время счета	Число итераций	Время счета	Число нитей		Число итераций	Время счета	Число итераций	Время счета	Число нитей	
2	342	17,4	421	5,73	12	3,04	401	19,55	469	6,13	12	3,19
4	343	9,93	421	4,62	6	2,15	401	11,02	444	6,33	4	1,74
8	375	5,96	421	3,8	3	1,57	435	6,67	469	4,05	3	1,65
10	374	4,5	414	4,63	3	0,97	427	5,04	493	4,17	3	1,2
16	386	3,52	442	3,9	3	0,9	444	3,86	493	4,16	3	0,93

Таблица 3

Число итераций и время счета (в секундах) методами BIIC2S-CG и BIIC1-CG задачи с матрицей `parabolic_fem` на p процессорах без использования и с использованием OpenMP-технологии

p	BIIC2S-CG						BIIC1-CG					
	MPI		MPI+OpenMP			μ	MPI		MPI+OpenMP			μ
	Число итераций	Время счета	Число итераций	Время счета	Число нитей		Число итераций	Время счета	Число итераций	Время счета	Число нитей	
2	296	8,79	368	3,22	12	2,73	415	10,98	454	3,19	12	3,44
4	323	5,22	368	2,63	6	1,98	418	5,79	453	3,14	4	1,84
8	346	2,94	368	2,03	3	1,45	440	3,27	454	2,08	3	1,57
10	359	3,24	399	3,06	3	1,05	463	3,09	497	2,66	4	1,16
16	357	1,59	424	2,33	3	0,68	353	1,80	509	2,28	3	0,79

Таблица 4

Число итераций и время счета (в секундах) методами BIIC2S-CG и BIIC1-CG задачи с матрицей `apache2` на p процессорах без использования и с использованием OpenMP-технологии

p	BIIC2S-CG						BIIC1-CG					
	MPI		MPI+OpenMP			μ	MPI		MPI+OpenMP			μ
	Число итераций	Время счета	Число итераций	Время счета	Число нитей		Число итераций	Время счета	Число итераций	Время счета	Число нитей	
2	403	15,85	509	5,63	12	2,81	489	16,69	542	6,78	8	2,46
4	418	10,5	509	4,71	6	2,23	501	11,21	611	6,89	8	1,63
8	451	5,17	509	3,86	3	1,34	529	5,36	577	4,3	3	1,25
10	461	4,61	560	5,28	3	0,87	540	4,60	618	5,29	3	0,86
16	481	2,99	581	3,99	3	0,75	542	2,97	645	4,5	3	0,66

Таблица 5

Число итераций и время счета (в секундах) методами BIIC2S-CG и BIIC1-CG задачи с матрицей `ecology2` на p процессорах без использования и с использованием OpenMP-технологии

p	BIIC2S-CG						BIIC1-CG					
	MPI		MPI+OpenMP			μ	MPI		MPI+OpenMP			μ
	Число итераций	Время счета	Число итераций	Время счета	Число нитей		Число итераций	Время счета	Число итераций	Время счета	Число нитей	
2	560	27,55	544	7,49	16	3,68	697	31,84	809	10,19	12	3,12
4	608	14,8	698	6,71	6	2,21	721	17,3	790	9,98	4	1,74
8	623	8,32	698	5,45	3	1,53	740	9,87	809	6,68	3	1,48
10	634	7,19	714	7,72	3	0,93	751	8,25	846	8,2	4	1,01
16	677	5,48	737	5,48	3	1,0	790	6,27	854	7,07	3	0,88

Как видно из табл. 2—7 и рис. 1—6, применение (MPI+OpenMP)-технологии позволяет значительно быстрее, чем при применении только MPI, выполнять решение всех тестовых задач методом BIIC2S-CG при $p < 10$. При $p = 10$ и 16 применение OpenMP-технологии для решения тестовых задач методом BIIC2S-CG становится нецелесообразным.

Как видно из табл. 2—6 и рис. 1—5, применение (MPI+OpenMP)-технологии для решения задач 1—5 позволяет значительно ускорить их решение методом BIIC1-CG по сравнению с использованием только MPI при $p < 10$. При $p = 10$ и 16 применение OpenMP-технологии для решения тестовых задач методом BIIC1-CG становится нецелесообразным.

Таблица 6

Число итераций и время счета (в секундах) методами BIIC2S-CG и BIIC1-CG задачи с матрицей `boneS01` на p процессорах без использования и с использованием OpenMP-технологии

p	BIIC2S-CG					μ	BIIC1-CG					μ
	MPI		MPI+OpenMP				MPI		MPI+OpenMP			
	Число итераций	Время счета	Число итераций	Время счета	Число нитей		Число итераций	Время счета	Число итераций	Время счета	Число нитей	
2	301	12,28	366	3,62	12	3,39	301	10,09	365	3,4	12	2,96
4	324	6,89	366	3,26	6	2,11	348	6,19	402	3,2	12	1,93
8	332	3,91	366	2,69	3	1,45	335	3,28	365	2,44	3	1,34
10	350	3,27	380	3,21	3	1,02	376	3,02	378	2,99	3	1,01
16	357	2,19	390	2,5	3	0,88	380	2,04	404	2,32	3	0,88

Таблица 7

Число итераций и время счета (в секундах) методом BIIC2S-CG задачи с матрицей `cfd2` на p процессорах без использования и с использованием OpenMP-технологии

p	MPI		MPI+OpenMP			μ
	Число итераций	Время счета	Число итераций	Время счета	Число нитей	
2	296	8,61	436	3,15	12	2,73
4	333	5,14	436	2,76	6	1,86
8	386	3,20	436	2,36	3	1,35
10	393	2,61	462	2,61	3	1,0
16	428	2,09	515	2,23	3	0,94

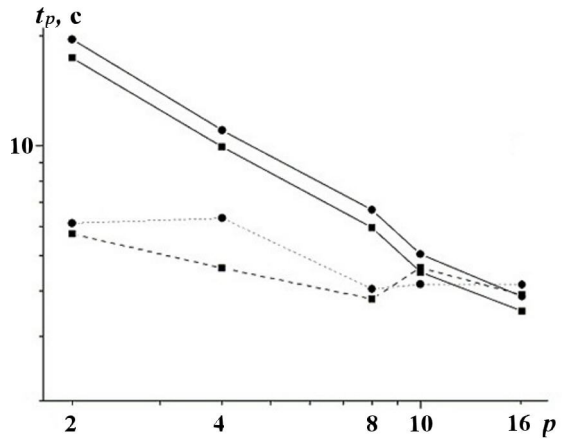

Рис. 1. Времена счета задачи с матрицей `5_1048576` методами BIIC2S-CG и BIIC1-CG с использованием технологий MPI и MPI+OpenMP: —•— — BIIC1-CG, MPI; - -•- - — BIIC1-CG, MPI+OpenMP; —■— — BIIC2S-CG, MPI; - -■- - — BIIC2S-CG, MPI+ +OpenMP

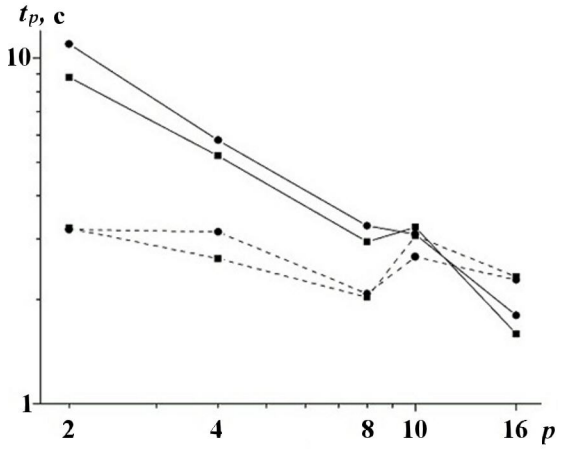

Рис. 2. Времена счета задачи с матрицей `parabolic_fem` методами BIIC2S-CG и BIIC1-CG с использованием технологий MPI и MPI+OpenMP: —•— — BIIC1-CG, MPI; - -•- - — BIIC1-CG, MPI+ +OpenMP; —■— — BIIC2S-CG, MPI; - -■- - — BIIC2S-CG, MPI+OpenMP

Заметим, что алгоритм построения матрицы предобусловливания BIIC-IC1 значительно проще, чем матрицы предобусловливания BIIC-IC2S, и время вычисления предобусловливателя BIIC-IC1, как правило, меньше, чем время вычисления предобусловливателя BIIC-IC2S, особенно для случая не очень сильно разреженных матриц. Однако ожидается, что скорость сходимости итерационного

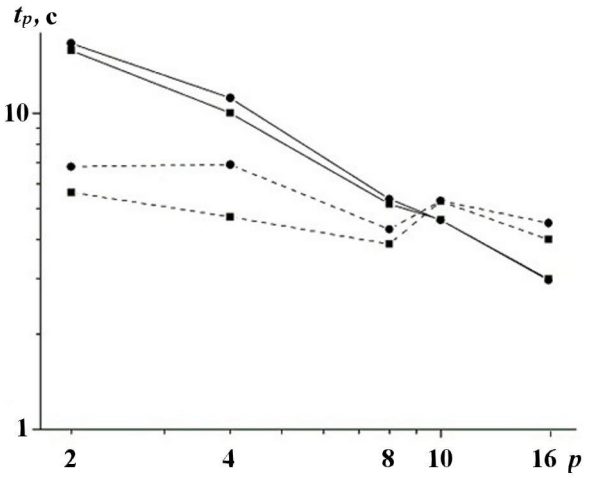

Рис. 3. Времена счета задачи с матрицей `apache2` методами BIIC2S-CG и BIIC1-CG с использованием технологий MPI и MPI+OpenMP: —●— — BIIC1-CG, MPI; - -●- - — BIIC1-CG, MPI+OpenMP; —■— — BIIC2S-CG, MPI; - -■- - — BIIC2S-CG, MPI+ +OpenMP

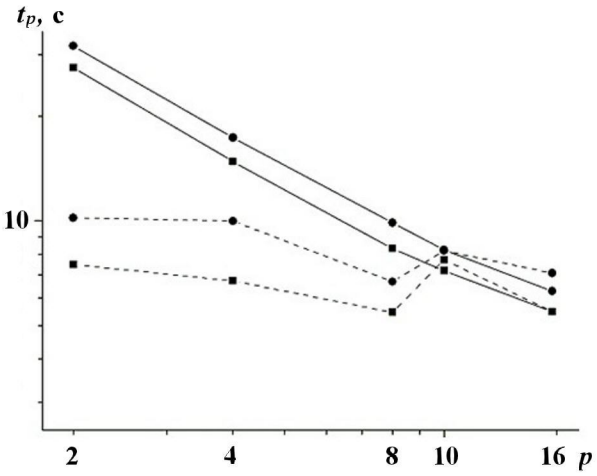

Рис. 4. Времена счета задачи с матрицей `ecology2` методами BIIC2S-CG и BIIC1-CG с использованием технологий MPI и MPI+OpenMP: —●— — BIIC1-CG, MPI; - -●- - — BIIC1-CG, MPI+OpenMP; —■— — BIIC2S-CG, MPI; - -■- - — BIIC2S-CG, MPI+ +OpenMP

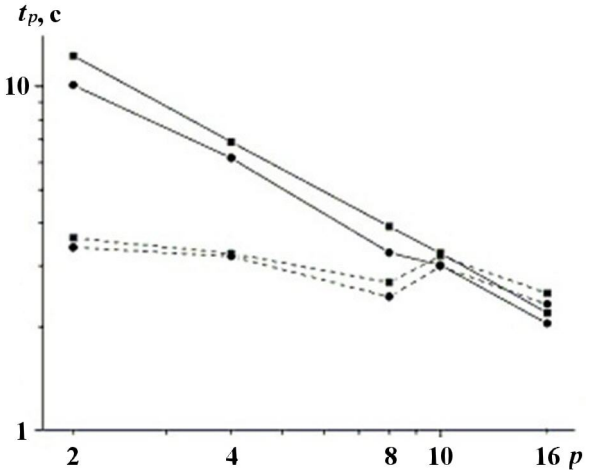

Рис. 5. Времена счета задачи с матрицей `boneS01` методами BIIC2S-CG и BIIC1-CG с использованием технологий MPI и MPI+OpenMP: —●— — BIIC1-CG, MPI; - -●- - — BIIC1-CG, MPI+OpenMP; —■— — BIIC2S-CG, MPI; - -■- - — BIIC2S-CG, MPI+ +OpenMP

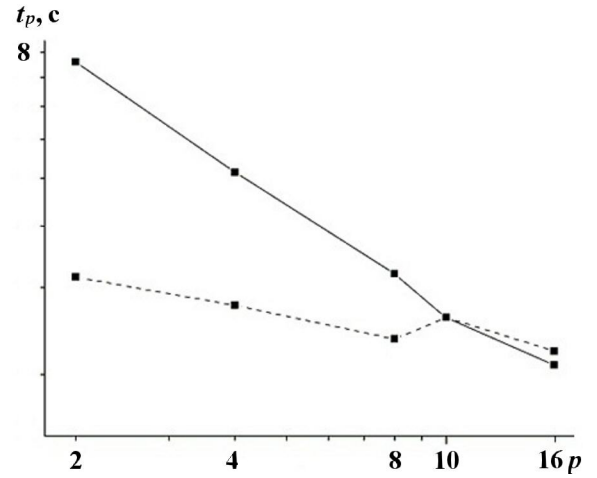

Рис. 6. Времена счета задачи с матрицей `cfd2` методом BIIC2S-CG с использованием технологий MPI (—■—) и MPI+OpenMP (- -■- -)

процесса в методе BIIC2S-CG должна быть выше, чем в BIIC1-CG. Кроме того, метод BIIC2S-CG является безотказным.

Как видно из табл. 2—5 и рис. 1—4, решение задач 1—4 методом BIIC2S-CG с применением MPI и MPI+OpenMP в подавляющем большинстве случаев происходит быстрее, чем методом BIIC1-CG. Для задач 1—4 число итераций метода BIIC2S-CG меньше числа итераций метода BIIC1-CG как без использования, так и при использовании OpenMP-технологии. Решение задачи 5 с более заполненной матрицей `boneS01` методами BIIC2S-CG и BIIC1-CG выполнялось с разными значениями τ. При этом, как видно из табл. 6 и рис. 5, решение этой задачи методом BIIC2S-CG с применением MPI и MPI+OpenMP происходит немного медленнее, чем методом BIIC1-CG.

На рис. 7 приведены графики ускорения счета задач ν методами BIIC2S-CG и BIIC-CG с применением MPI- и (MPI+OpenMP)-технологий по сравнению с расчетами на двух процессорах.

Как видно из рис. 7, *а*, при решении задач 1—4, 6 при $p \le 8$, а задачи 5 — при $p < 8$ методом BIIC2S-CG с применением (MPI+OpenMP)-технологии наблюдается сверхлинейное ускорение. Как видно из рис. 7, *б*, при решении задач 1, 2, 4, 5 при $p \le 8$, а задачи 4 — при $p < 8$ методом BIIC1-CG с применением (MPI+OpenMP)-технологии также наблюдается сверхлинейное ускорение.

Уменьшение эффекта от использования OpenMP-технологии с увеличением числа процессоров объясняется, в частности, уменьшением числа строк матрицы, приходящихся на каждый процессор, т. е. уменьшением вычислительной работы на каждом процессоре. В работе [6] проведено исследование влияния размера матрицы на эффективность использования OpenMP-технологии с помощью решения модельных задач — разностных задач Дирихле для уравнения Пуассона в единичной квадратной области на равномерной ортогональной сетке с возрастающим числом узлов. Решение этих задач выполнялось методом BIIC2S-CG без использования налегания ($q = 0$). В этом случае имеет место предобусловливание блочным методом Якоби в сочетании с IC2S(τ) (BJIC2(τ)). Расчеты продемонстрировали, что с увеличением размера матрицы эффективность использования OpenMP-технологии для фиксированного числа процессоров возрастает.

В настоящей работе в качестве тестовых матриц использовались матрицы относительно небольшого размера. При расчетах реальных физических задач размеры матриц, как правило, значительно больше. Можно ожидать, что потеря эффективности от применения OpenMP-технологии при решении задач наступит при значительно большем числе процессоров.

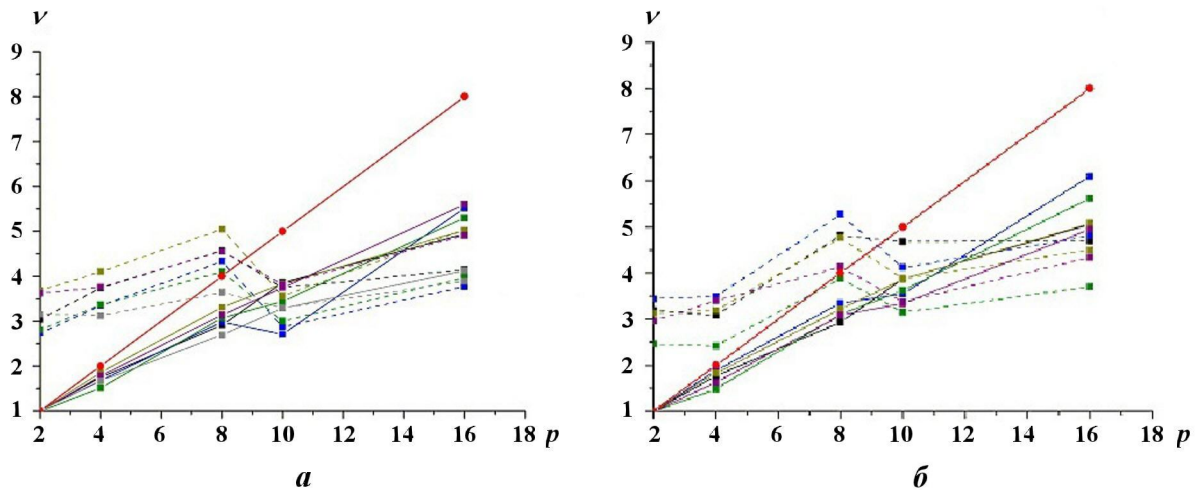

Рис. 7. Ускорения счета задач при использовании методов BIIC2S-CG (*а*) и BIIC21-CG (*б*) с применением технологий MPI (——) и MPI+OpenMP (- - -): ■ — `5_1048576`; ■ — `parabolic_fem`; ■ — `apache2`; ■ — `ecology2`; ■ — `boneS01`; ■ — `cfd2`; ● — идеальное ускорение

Заключение

В работе рассмотрен новый предобусловливатель BIIC-IC1 для решения СЛАУ (1) методом сопряженных градиентов. Предложен способ применения (MPI+OpenMP)-технологии для построения и обращения предобусловливателей BIIC-IC2S и BIIC-IC1 с числом блоков, кратным числу процессоров и числу используемых потоков. С помощью расчетов модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse показано, что использование (MPI+OpenMP)-технологии для реализации метода сопряженных градиентов с предобусловливателями BIIC-IC2S(τ) и BIIC-IC1(τ) позволяет существенно ускорить вычисления по сравнению с применением только MPI для не слишком большого числа узлов суперкомпьютерной системы. При этом наблюдается сверхлинейное ускорение расчетов. Как показали расчеты, в подавляющем большинстве случаев

решение тестовых задач методом сопряженных градиентов с предобусловливанием BIIC-IC2S(τ) происходит быстрее, чем с предобусловливанием BIIC-IC1(τ), как при использовании MPI, так и при применении (MPI+OpenMP)-технологии.

Список литературы

1. *Kaporin I. E.* New convergence results and preconditioning strategies for conjugate gradient method // Numer. Linear Algebra and Appls. 1994. Vol. 1, No 2. P. 179—210.
2. *Капорин И. Е.* О предобусловливании метода сопряженных градиентов при решении дискретных аналогов дифференциальных задач // Дифференциальные уравнения. 1990. Т. 26, № 7. С. 1225—1236.
   Kaporin I. E. O predobuslovlivanii metoda sopryazhyennykh gradientov pri reshenii diskretnykh analogov differentsialnykh zadach // Differentsialnye uravneniya. 1990. T. 26, № 7. S. 1225—1236.
3. *Kaporin I. E.* High quality preconditionings of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition // Numer. Lin. Alg. Appl. 1998. Vol. 5. P. 483—509.
4. *Капорин И. Е., Милюкова О. Ю.* Массивно-параллельный алгоритм предобусловленного метода сопряженных градиентов для численного решения систем линейных алгебраических уравнений // Сб. трудов отдела проблем прикладной оптимизации ВЦ РАН / Под ред. В. Г. Жадана. М.: ВЦ РАН, 2011. С. 132—157.
   Kaporin I. E., Milyukova O. Yu. Massivno-parallelnyy algoritm predobuslovlennogo metoda sopryazhyennykh gradientov dlya chislennogo resheniya system lineynykh algebraicheskikh uravneniy // Sb. trudov otdela problem prikladnoy optimizatsii VTs RAN / Pod red. V. G. Zhadana. M.: VTs RAN, 2011. S. 132—157.
5. *Munksgaard N.* Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Software. 1980. No 6. P. 206—219.
6. *Милюкова О. Ю.* MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1: Препринт № 48. М.: ИПМ им. М. В. Келдыша РАН, 2021. https://doi.org/10.20948/prepr-2021-48.
   Milyukova O. Yu. MPI+OpenMP realizatsiya metoda sopryazhyennykh gradientov s predobuslovlivatelem blochnogo nepolnogo obratnogo treugolnogo razlozheniya IC2S i IC1: Preprint № 48. M.: IPM im. M. V. Keldysha RAN, 2021. https://doi.org/10.20948/prepr-2021-48.
7. *Tuff A. D., Jennings A.* An iterative method for large systems of linear structural equations // J. Numer. Methods Eng. 1973. No 7. P. 175—183.
8. *Капорин И. Е., Коньшин И. Н.* Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Журнал вычисл. мат. и мат. физ. 2001. Т. 41, № 4. С. 515—528.
   Kaporin I. E., Konshin I. N. Parallelnoe reshenie simmetrichnykh polozhitelno opredelyennykh system na osnove perekryvayushchegosya razbieniya na bloki // Zhurnal vychisl. mat. i mat. fiz. 2001. T. 41, № 4. S. 515—528.
9. *Anderson E. C., Saad Y.* Solving sparse triangular systems on parallel computers // Int. J. High Speed Computing. 1989. Vol. 1. P. 73—96.
10. *Hammond S. W., Schreiber R.* Efficient ICCG on a shared memory multiprocessor // Ibid. 1992. Vol. 4. P. 1—21.
11. *Wolf M. M., Heroux M. A., Boman E. G.* Factors impacting performance of 535 multithreaded sparse triangular solve // Proc. 9th Int. Conf. on High Performance Computing for Computational Science VECPAR'10. Berlin, Heidelberg: Springer-Verlag, 2011. P. 32—44.
12. *Chow E., Anzt H., Scott J., Dongarra J.* Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning // J. Parallel and Distributed Computing. 2018. No 119. P. 219—230.

13. *Chow E., Patel A.* Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comp. 2015. Vol. 37. P. 169—193.

14. *Cayrols S., Duff I., Lopes F.* Parallelization of the Solve Phase in a Task-Based Cholesky Solver Using a Sequential Task Flow Model. Technical Report RAL-TR-2018-008. UK, Science & Technology Facilities Council, 2018.

15. *Капорин И. Е., Милюкова О. Ю.* MPI+OpenMP параллельная реализация метода сопряженных градиентов с некоторыми явными предобусловливателями: Препринт № 8. М.: ИПМ им. М. В. Келдыша РАН, 2018. https://doi.org/10.20948/prepr-2018-8.
Kaporin I. E., Milyukova O. Yu. MPI+OpenMP parallelnaya realizatsiya metoda sopryazhyennykh gradientov s nekotorymi yavnymi predobuslovlivatelyami: Preprint № 8. M.: IPM im. M. V. Keldysha RAN, 2018. https://doi.org/10.20948/prepr-2018-8.

16. *Капорин И. Е., Милюкова О. Ю.* MPI+OpenMP реализация метода сопряженных градиентов с факторизованными явными предобусловливателями // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2018. Вып. 4. С. 57—69.
Kaporin I. E., Milyukova O. Yu. MPI+OpenMP realizatsiya metoda sopryazhyennykh gradientov s faktorizovannymi yavnymi predobuslovlivatelyami // Voprosy atomnoy nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskikh protsessov. 2018. Vyp. 4. S. 57—69.

17. *Chow E.* Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns // Int. J. High Performance Comp. Appl. 2001. Vol. 15. No 1. P. 56—74.

18. *Милюкова О. Ю.* MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем: Препринт № 31. М.: ИПМ им. М. В. Келдыша РАН, 2020. https://doi.org/10.20948/prepr-2020-31.
Milyukova O. Yu. MPI+OpenMP realizatsiya metoda sopryazhyennykh gradientov s faktorizovannym predobuslovlivatelem: Preprint № 31. M.: IPM im. M. V. Keldysha RAN, 2020. https://doi.org/10.20948/prepr-2020-31.

19. *Милюкова О. Ю.* MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1: Препринт № 83. М.: ИПМ им. М. В. Келдыша РАН, 2020. https://doi.org/10.20948/prepr-2020-83.
Milyukova O. Yu. MPI+OpenMP realizatsiya metoda sopryazhyennykh gradientov s predobuslovlivatelem blochnogo Yakobi IC1: Preprint № 83. M.: IPM im. M. V. Keldysha RAN, 2020. https://doi.org/10.20948/prepr-2020-83.

20. *Duff I. S., Meurant G. A.* The effect of ordering on preconditioned conjugate gradients // BIT. 1989. Vol. 29. P. 625—657.

21. *Капорин И. Е., Милюкова О. Ю.* Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов: Препринт № 37. М.: ИПМ им. М. В. Келдыша РАН, 2017. https://doi.org/10.20948/prepr-2017-37.
Kaporin I. E., Milyukova O. Yu. Nepolnoe obratnoe treugolnoe razlozhenie v parallelnykh algoritmakh predobuslovlennogo metoda sopryazhyennykh gradientov: Preprint № 37. M.: IPM im. M. V. Keldysha RAN, 2017. https://doi.org/10.20948/prepr-2017-37.

22. *Kaporin I. E.* Reordering and splitting of sparse matrices into overlapping blocks for massively parallel preconditioning of iterative methods // Int. Conf. "NUMGRID-2012". Moscow, A. A. Dorodnicyn Computing Center RAS. Dec. 17—18, 2012.

23. *Davis T., Hu Y. F.* University of Florida sparse matrix collection // ACM Trans. on Math.-Software. 2011. Vol. 38, No 1 // http://www.cise.ufl.edu/research/sparse/matrices.

24. *Axelsson O.* Iterative solution methods. New York: Cambridge Univ. Press, 1994.

25. *Капорин И. Е.* Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Журнал вычисл. мат. и мат. физ. 2012. Т. 52, № 2. С. 1—26.

Kaporin I. E. Ispolzovanie polinomov Chebysheva i priblizhyennogo obratnogo treugolnogo razlozheniya dlya predobuslovlivaniya methoda sopryazhyennykh gradientov // Zhurnal vychisl. mat. i mat. fiz. 2012. T. 52, № 2. S. 1—26.

Статья поступила в редакцию 09.08.21.