2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

doi 10.26089/NumMet.v23r312 УДК 519.63

MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения первого порядка

О. Ю. Милюкова
Институт прикладной математики имени М. В. Келдыша РАН (ИПМ РАН),
Москва, Российская Федерация
ORCID: 0000-0001-7887-8700, e-mail: olgamilyukova@mail.ru

Аннотация: В работе рассматривается предобусловливатель блочного неполного обратного треугольного разложения первого порядка “по значению” BIIC-IC1 для решения систем линейных алгебраических уравнений с симметричной положительно определенной матрицей. Рассматривается способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1, при этом в предобусловливателе число блоков кратно числам используемых процессоров и используемых потоков. Предлагается способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1, в котором для применения OpenMP технологии используется специальное упорядочение узлов сетки внутри подобластей, соответствующих расчетам на процессорах. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем BIIC-IC1 с использованием MPI и гибридной MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse.

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

Для цитирования: Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения первого порядка // Вычислительные методы и программирование. 2022. **23**, № 3. 191–206. doi 10.26089/NumMet.v23r312.

MPI+OpenMP implementation of conjugate gradients method with preconditioner of the block incomplete inverse triangular decomposition of the first order

O. Yu. Milyukova
Keldysh Institute of Applied Mathematics, Moscow, Russia
ORCID: 0000-0001-7887-8700, e-mail: olgamilyukova@mail.ru

Abstract: The paper considers the preconditioner of the block incomplete inverse triangular decomposition of the first order “by value” BIIC-IC1 for solving systems of linear algebraic equations with a symmetric positively defined matrix. A method of using MPI+OpenMP technology for constructing and inverting the BIIC-IC1 preconditioner, in which the number of blocks in the preconditioner is a multiple of the number of used processors and used threads, is considered. A method is proposed for using the MPI+OpenMP technology to construct and invert the BIIC-IC1 preconditioner, in which a special ordering of grid nodes within subdomains corresponding to processor calculations is used to apply the OpenMP technology. The time of solving problems by

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

the conjugate gradient method with the BIIC-IC1 preconditioner using MPI and hybrid MPI + OpenMP technology is compared on the example of a model problem and a number of problems from the collection of sparse matrices SuiteSparse.

Keywords: implicit block preconditioning, incomplete Cholesky factorization, parallel preconditioning, conjugate gradient method.

For citation: O. Yu. Milyukova, "MPI+OpenMP implementation of conjugate gradients method with preconditioner of the block incomplete inverse triangular decomposition of the first order," Numerical Methods and Programming. **23** (3), 191–206 (2022). doi 10.26089/NumMet.v23r312.

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

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

с симметричной положительно определенной разреженной матрицей A 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, а также к ухудшению их обусловленности. Ввиду значительных вычислительных затрат решение задач с матрицами большого размера требует применения параллельных компьютеров.

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

$$\|b - Ax_k\| \leqslant \varepsilon \|b - Ax_0\|, \quad 0 < \varepsilon \ll 1. \tag{2}$$

Для предобусловливания используется блочное неполное обратное разложение Холецкого (Block Incomplete Inverse Cholesky) BIIC [1, 2] в сочетании с приближенным треугольным разложением с отсечением по параметру $0 < \tau \ll 1$ первого порядка IC1(τ) — предобусловливание блочного неполного обратного треугольного разложения первого порядка BIIC-IC1(τ) [3]. 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 [4]. Предобусловливание IC1(τ) имеет ограниченную область применимости для фиксированного значения τ [5], в ряде случаев для безотказности требует чрезмерного уменьшения параметра τ. Заметим, что модификация этого предобусловливания — предобусловливание RIC(τ) (Robust Incomplete Cholesky) [6, 7] обеспечивает безотказность предобусловленного метода сопряженных градиентов RIC(τ)-CG при любом значении τ. К недостаткам предобусловливания RIC(τ) следует отнести довольно завышенную величину возмущения, вносимого в диагональ матрицы A при построении этого предобусловливателя и, как следствие, недостаточно высокое качество предобусловливания [8].

Основная трудность распараллеливания алгоритма метода сопряженных градиентов с неявным факторизованным предобусловливателем, имеющим вид $H = (LL^T)^{-1} \approx A^{-1}$ или $H = (LDL^T)^{-1} \approx A^{-1}$, где L — нижнетреугольная матрица, а D — диагональная матрица, связана с рекурсивным характером вычислений при построении и обращении предобусловливателя.

В работах [9, 10] предложен подход, который позволяет избежать проблему распараллеливания рекурсивных вычислений при построении и обращении предобусловливателя при решении задачи на многопроцессорной вычислительной системе. В этих работах предложены параллелизуемые предобусловливатели, представляющие собой блочную версию предобусловливания неполного обратного треугольного разложения BIIC в сочетании с неполным треугольным разложением второго порядка IC2(τ) [7] — BIIC-IC2(τ) [9] и в сочетании с стабилизированным неполным треугольным разложением второго порядка IC2S(τ) [7] — BIIC-IC2S(τ) [10]. Для построения этих предобусловливателей специальным образом строятся блоки с налеганием, а внутри блоков используется приближенное треугольное разложение второго порядка IC2(τ) или IC2S(τ). Методы сопряженных градиентов с предобусловливанием BIIC-IC2(τ), BIIC-IC2S(τ) являются безотказными при любых значениях τ и были эффективно реализованы на параллельных архитектурах с распределенной памятью [9, 10].

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

Одним из способов преодоления основной трудности распараллеливания алгоритмов построения и обращения неявного факторизованного предобусловливателя, связанной с рекурсивным характером вычислений, является использование переупорядочений узлов сетки и соответствующих перестановок строк и столбцов матрицы, например использование переупорядочений, связанных с разбиением области расчета (DDO — Domain Decompositing Ordering) [11]. Применению такого подхода для крупнозернистого распараллеливания посвящено много работ, например [12–17]. В работе [18] предложено использовать упорядочение типа DDO для построения параллельного варианта метода стабилизированного треугольного разложения второго порядка сопряженных градиентов (IC2S(τ)-CG).

Проблеме использования высокоуровнего параллелизма (мелкозернистого или распараллеливания алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. В работах [19–22] было предложено использовать несколько итераций Якоби или блочного Якоби для решения треугольных систем при применении предобусловливания неполного треугольного разложения, что позволило получить высокий уровень параллелизма. В работе [23] предложен безытерационный способ применения MPI+OpenMP технологии при обращении неявного факторизованного предобусловливателя, т.е. решении двух треугольных систем (в том числе для случая решения СЛАУ (1)). В работе [24] предложен новый итерационный алгоритм вычисления предобусловливателей IC(0) (неполного треугольного разложения Холецкого без заполнения), IC(1), IC(2) (неполного треугольного разложения Холецкого с заполнением 1 и 2), в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно.

Заметим, что использование явных предобусловливателей позволяет эффективно применять MPI+OpenMP технологии для параллельного решения СЛАУ (1) предобусловленным методом сопряженных градиентов, например [25–27].

В работах [28, 29] предложены два безытерационных способа применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с IC(0) (неполного треугольного разложения Холецкого без заполнения) и IC1(τ). Один из них основан на переупорядочении узлов сетки типа DDO внутри каждой подобласти, соответствующей расчетам на своем процессоре, другой способ из этих работ основан на уменьшении шаблона разреженности матрицы A при построении предобусловливателя. В работе [30] предложены безытерационные способы применения MPI+OpenMP технологии при построении и обращении предобусловливателей блочного Якоби в сочетании с IC1(τ) и с IC2S(τ), при этом число блоков в блочном Якоби кратно числу используемых процессоров и числу используемых потоков.

В работе [3] впервые предложен безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателей BIIC-IC1(τ) и BIIC-IC2S(τ) на основе использования числа блоков в предобусловливателе, кратного числу используемых процессоров и числу используемых потоков. В работах [3, 31] предложены безытерационные способы применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ), в которых для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO. При этом в работе [3], в отличие от работы [31], при построении матрицы предобусловливания для некоторых ее строк осуществлялось отсечение по позициям. Если использовалось упорядочение узлов сетки типа DDO для применения OpenMP технологии при построении и обращении предобусловливателя, то OpenMP технологии применялись для большинства строк матрицы предобусловливателя. Заметим, что использование упорядочения узлов сетки типа DDO для мелкозернистого распараллеливания этапов построения и обращения предобусловливателя BIIC-IC2S(τ) нецелесообразно [3, 31].

В формуле (1) предполагается, что матрица A уже переупорядочена, а вместо A_P стоит A ($A = A_P = P\widetilde{A}P^T$), где P — матрица перестановки, а $\widetilde{A}$ — матрица коэффициентов исходной задачи. В настоящей работе применяются переупорядочения, уменьшающие среднюю ширину ленты матрицы, а именно предложенные в работе [32], являющиеся обобщением упорядочения [10]. 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.

В настоящей работе рассматривается предобусловливатель BIIC-IC1(τ) [3] для решения СЛАУ (1) методом сопряженных градиентов и безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ) — способ 1, в котором в предобусловливателе

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

BIIC-IC1(τ) используется число блоков, кратное числу используемых процессоров и числу используемых потоков [3]. Рассматривается новый безытерационный способ применения MPI+OpenMP технологии [31] для построения и обращения предобусловливателя BIIC-IC1(τ), в котором число блоков в предобусловливателе совпадает с числом используемых процессоров, а для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO — способ 2, впервые предложенный в работе [31]. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем BIIC-IC1(τ) с использованием MPI и MPI+OpenMP подходов на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse [33] и сравнение времени решения этих тестовых задач методом BIIC-IC1(τ)-CG с использованием двух рассматриваемых в работе способов применения MPI+OpenMP технологии.

2. Предобусловленный метод сопряженных градиентов. Решение СЛАУ (1) будем искать с помощью алгоритма предобусловленного метода сопряженных градиентов, который имеет следующий вид.

Алгоритм 1. Алгоритм предобусловленного метода сопряженных градиентов [34]

Algorithm 1. Algorithm of the preconditioned conjugate gradient method [34]

$\varepsilon \ll 1$, H — матрица предобусловливания ($H \approx A^{-1}$)

$r_0 = b - Ax_0, \quad p_0 = w_0 = Hr_0, \quad \gamma_0 = r_0^T p_0,$
for $k = 0, 1, \ldots$ **do**
  while $(r_k^T r_k) > \varepsilon^2 (r_0^T r_0)$ **do**
    $q_k = Ap_k, \quad \alpha_k = \gamma_k / (p_k^T q_k),$
    $x_{k+1} = x_k + \alpha_k p_k, \quad r_{k+1} = r_k - \alpha_k q_k, \quad z_{k+1} = Hr_{k+1},$
    $\gamma_{k+1} = r_{k+1}^T z_{k+1}, \quad \beta_k = \gamma_{k+1}/\gamma_k, \quad p_{k+1} = z_{k+1} + \beta_k p_k,$
  end while
end for

Этот алгоритм использует операции умножения разреженных матриц на вектор, операции вычисления скалярных произведений, элементарные векторные операции, а также вычисление $z_{k+1} = Hr_{k+1}$, $p_0 = w_0 = Hr_0$. Принципиальная возможность эффективной параллельной реализации всех операций, кроме вычисления Hr_{k+1}, Hr_0, не вызывает сомнений, даже при использовании большого числа процессоров и (или) применения OpenMP технологии.

3. Предобусловливатель блочного неполного обратного треугольного разложения первого порядка. Рассмотрим метод предобусловливания BIIC-IC1(τ), предложенный первоначально в работе [3]. Пусть матрица A переупорядочена и разбита на блоки, причем на блочной диагонали расположены p квадратных блоков размера $n_s \times n_s$, $1 \leqslant s \leqslant p$. Обозначим $k_s = n_1 + \ldots + n_s$, и пусть $m_s \geqslant n_s$ — размеры расширенных блоков. Определим прямоугольные матрицы

$$V_s = \left[e_{j_s(1)} | \ldots | e_{j_s(m_s - n_s)} | e_{k_{s-1}+1} | \ldots | e_{k_s} \right],$$

столбцы которых являются единичными n-векторами, где $k_{s-1} + 1, \ldots, k_s$ представляют собой индексы s-го блока, а $j_s(1), \ldots, j_s(m_s - n_s)$ являются индексами перекрытия, причем $j_s(.) \leqslant k_{s-1}$. Используя приближенное треугольное разложение первого порядка “по значению” для аппроксимации $m_s \times m_s$ матриц $A_s = V_s^T A V_s$:

$$A_s = \hat{U}_s^T \hat{U}_s - E_s,$$

где $\hat{U}_s$ — верхнетреугольные матрицы, $\|E_s\| = O(\tau)$ и $0 < \tau \ll 1$ — порог отсечения, определим предобусловливатель BIIC-IC1(τ) формулой

$$H = \sum_{s=1}^{p} V_s \hat{U}_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} \hat{U}_s^{-T} V_s^T, \quad \text{в которой } \hat{U}_s^{-T} = \left(\hat{U}_s^T \right)^{-1}. \tag{3}$$

В алгоритме построения матрицы $\hat{U}_s$ в предобусловливателе IC1(τ) для матрицы A_s размера $m_s \times m_s$ для каждого i выбор элементов e_{ij} матрицы погрешности E_s осуществляется так, чтобы “убрать” все “малые” значения:

$$v_{ij} = u_{ii} u_{ij} - e_{ij} = a_{ij} - \sum_{s=1}^{i-1} u_{si} u_{sj}, \quad j = i, \ldots, n.$$

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

Здесь a_{ij} — элементы матрицы A_s, u_{ij} — элементы матрицы $\hat{U}_s$. Перед построением матрицы предобусловливания IC1(τ) для матрицы A_s необходимо обеспечить, чтобы матрицы A_s были отмасштабированы. Алгоритм построения матрицы $\hat{U}_s$ приведен, например, в работах [3, 31].

Заметим, что при вычислении диагональных элементов u_{ii} матрицы $\hat{U}_s$ необходимо извлекать квадратный корень из числа, которое определяется в процессе вычисления элементов строк этих матриц с номерами, не превосходящими i. Это выражение может оказаться отрицательным. В этом случае следует уменьшить значение τ и использовать метод BIIC-IC1(τ)-CG с уменьшенным значением τ.

4. Алгоритмы параллельной реализации. Пусть матрица A переупорядочена, отмасштабирована и разбита на p блоков, где p — число используемых процессоров. В настоящей работе, не ограничивая общности, в предобусловливателе BIIC-IC1(τ) будем использовать налегание [1, 2, 9] с шириной $q = 1$. Перед вычислением матрицы предобусловливания в каждом процессоре с номером s ($1 \leqslant s \leqslant p$) строятся матрицы $A_s = V_s^T A V_s$ размера $m_s \times m_s$. Для этого осуществляются необходимые пересылки строк матрицы A.

Параллельная реализация вычисления предобусловливателя BIIC-IC1(τ) с применением только MPI не представляет труда [3]. Для вычисления элементов матрицы $\hat{U}_s$ в формуле (3) используется алгоритм из работ [3, 31], пересылок не требуется. Матрицы $\hat{U}_s^T$ находятся с помощью транспонирования матриц $\hat{U}_s$ в каждом процессоре с номером s, что не требует межпроцессорного обмена.

Обращение предобусловливателя $z = Hr$, где H определено в (3), происходит следующим образом. Перед началом вычислений элементы вектора r_s с индексами $j_s(1), \ldots, j_s(m_s - n_s)$ должны быть пересланы в процессор с номером s. Затем в каждом процессоре с номером s вычисляются

$$z_s = \hat{U}_s^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_s} \end{bmatrix} \hat{U}_s^{-T} r_s.$$

После вычисления z_s выполняются необходимые пересылки элементов вектора z_s с индексами $j_s(1), \ldots, j_s(m_s - n_s)$ в другие процессоры и завершается вычисление $z = Hr$.

Заметим, что можно не вычислять элементы матриц $\hat{U}_s^T$, а при обращении матриц $\hat{U}_s^T$ использовать способ обращения транспонированных матриц, первоначально предложенный в работе [10].

Рассмотрим способ 1 параллельной реализации с применением MPI+OpenMP технологии этапов построения и обращения предобусловливателя, предложенный первоначально в работе [3]. Разбиение всей области расчета при использовании этого способа производится сразу на pm подобластей, где p — число используемых процессоров, m — число используемых потоков в каждом процессоре. Процессор с номером s будет производить вычисления в "большой" подобласти с номером s, состоящей из подобластей с номерами $t = (s-1)m + 1, \ldots, sm$, полученными при разбиении. При использовании MPI+OpenMP технологии способом 1 число блоков в предобусловливателе BIIC-IC1(τ) равно pm. В программе необходимо задать число "внутренних" блоков с налеганием, равное m, и инициализировать число нитей, совпадающее с числом "внутренних" блоков.

Для построения предобусловливателя BIIC-IC1(τ) в каждом процессоре с номером s сначала создаются m матриц $A_t = V_t^T A V_t$, где $t = (s-1)m+1, \ldots, sm$, для этого совершаются необходимые пересылки строк матрицы A. Затем по матрице A_t в процессоре с номером s ($s = 1, \ldots, p$) строятся матрицы $\hat{U}_t$ ($t = (s-1)m+1, \ldots, sm$). При этом для каждого t вычисления элементов матриц $\hat{U}_t$ происходят в своем потоке, т.е. все рекурсивные вычисления происходят внутри потоков. На этапе построения матриц $\hat{U}_t$ в настоящей работе, так же как в работе [3], для циклов по $t1 = 1, \ldots, m$ использовалась директива `do` с опцией `schedule static`. При построении матриц A_t ($t = (s-1)m + 1, \ldots, sm$) в каждом процессоре с номером s ($s = 1, \ldots, p$) в настоящей работе, так же как в работе [3], используются OpenMP технологии с числом нитей m и директива `do` с опцией `schedule static`.

Обращение предобусловливателя BIIC-IC1(τ) при способе 1 применения MPI+OpenMP технологии осуществлялось с использованием формулы

$$z_t = \hat{U}_t^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_t} \end{bmatrix} \hat{U}_t^{-T} r_t, \quad t = (s-1)m + 1, \ldots, sm, \quad s = 1, \ldots, p. \tag{4}$$

Перед началом вычислений элементы векторов r_t для $t = (s-1)m+1, \ldots, sm$, необходимые для расчетов в процессоре с номером s и хранящиеся в памяти других процессоров, пересылаются в процессор с номером s. Для каждого $t = (s-1)m + 1, \ldots, sm$ вычисления элементов векторов z_t по формуле (4) происходят

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

в своем потоке. При вычислении элементов векторов z_t в настоящей работе, так же как в работе [3], использовалась директива `do` с опцией `schedule static`. Использовался способ обращения матриц $\hat{U}_t^T$, аналогичный способу обращения транспонированных матриц из работы [10]. При этом на этапе построения матрицы предобусловливания не нужно вычислять в явном виде матрицы $\hat{U}_t^T$. После вычисления z_t для всех $t = (s-1)m+1, \ldots, sm$, $s = 1, \ldots, p$, следует выполнить необходимые пересылки и завершить вычисление вектора z.

Рассмотрим новый способ — способ 2 применения MPI+OpenMP технологии построения и обращения предобусловливателя BIIC-IC1(τ), предложенный впервые в работе [31] и названный в ней способом 2. В этом способе в предобусловливателе BIIC-IC1(τ) используется p блоков с налеганием, где p — число процессоров ($p \neq 1$), а внутри каждого блока используется упорядочение узлов сетки расширенных подобластей типа DDO [18].

Так же, как при использовании способа 1, произведем разбиение всей области расчета сразу на pm подобластей, где p — число используемых процессоров, m — число используемых потоков. Процессор с номером s будет производить вычисления в "большой" подобласти с номером s, состоящей из подобластей с номерами $t = (s-1)m+1, \ldots, sm$, полученными при разбиении. При этом каждая нерасширенная подобласть с номером s, соответствующая расчетам на своем процессоре, уже получается разбитой на m "внутренних" подобластей, а при применении OpenMP технологии следует использовать m нитей. При разбиении расширенной подобласти будем использовать это разбиение узлов нерасширенной подобласти, а узлы налегания добавим в первую "внутреннюю" подобласть и расположим в начале первой "внутренней" подобласти. Такой способ разбиения в работе [31] назван разбиением 1.

Заметим, что в работе [31] был рассмотрен еще один способ разбиения области расчета — способ разбиения 2. При этом вся область расчета разбивалась на p подобластей так же, как при использовании только MPI. Затем каждая нерасширенная подобласть с номером s ($s = 1, \ldots, p$) разбивалась на m "внутренних" подобластей, где m — число используемых нитей (потоков) при применении OpenMP технологии, с приблизительно равным числом узлов сетки, причем разбиение нерасширенных подобластей осуществлялось в порядке следования узлов нерасширенных подобластей, установленном ранее. При разбиении расширенной подобласти узлы налегания добавлялись в первую "внутреннюю" подобласть и располагались в начале первой "внутренней" подобласти.

Произведем переупорядочение узлов сетки в расширенной подобласти. Будем использовать упорядочение типа DDO, предложенное в работе [18], а затем использованное для применения OpenMP технологии в работах [28, 29]. Введем множество узлов разделителей — множество узлов сетки во "внутренних" подобластях, у которых имеются соседи из "внутренних" подобластей с бóльшими номерами. Остальные узлы сетки во "внутренних" подобластях будем называть "внутренними". Множество узлов разделителей разобьем на три части. Узел разделителя назовем узлом разделителя первого уровня, если в шаблоне этого узла нет узлов разделителей из других подобластей с номерами бóльшими, чем номер рассматриваемой подобласти. Узел разделителя назовем узлом разделителя второго уровня, если в шаблоне этого узла нет узлов разделителей более высокого, чем первый, уровня, расположенных в подобластях с бóльшими номерами. Остальные узлы разделителей назовем узлами разделителей третьего уровня.

Определим $\bar{l}_s$ — минимальное число "внутренних" узлов при разбиении расширенной подобласти с номером s на "внутренние" подобласти. Предполагается, что $\bar{l}_s \neq 0$. Обозначим $M1 = m\bar{l}_s$. Установим следующий порядок следования узлов сетки расширенной подобласти с номером s ($s = 1, \ldots, p$). Сначала идут $\bar{l}_s$ "внутренних" узлов первой "внутренней" подобласти, затем $\bar{l}_s$ "внутренних" узлов второй "внутренней" подобласти и т.д., наконец, $\bar{l}_s$ "внутренних" узлов m-ой "внутренней" подобласти. Затем идут оставшиеся "внутренние" узлы "внутренних" подобластей в порядке следования номеров "внутренних" подобластей и в порядке следования узлов внутри "внутренних" подобластей. Затем идут узлы разделителей первого уровня, затем второго уровня, а затем третьего уровня. При этом для каждого уровня разделителей узлы следуют с сохранением порядка следования "внутренних" подобластей и порядка следования узлов внутри подобласти, введенного ранее.

Заметим, что узлы налегания в принципе могут попасть в множество узлов разделителей, в отличие от способа применения MPI+OpenMP технологии на основе переупорядочения типа DDO в работе [3]. В работе [3] узлы из области налегания не участвовали в переупорядочении, вычисления элементов строк верхнетреугольного множителя предобусловливания IC1(τ) начиналось со строк, соответствующих узлам налегания, причем без использования OpenMP технологии. При этом использовалось отсечение по позициям при построении матрицы предобусловливания в строках, соответствующих узлам из налегания. В

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

предложенном в настоящей работе способе использования MPI+OpenMp технологии никакого отсечения по позициям при построении предобусловливателя не производится.

Заметим, что, как показали расчеты тестовых задач методом сопряженных градиентов с предобусловливанием блочного Якоби в сочетании с IC1(τ) (BJIC1(τ)), применение OpenMP технологии для вычисления верхнетреугольного множителя предобусловливателя и обращения матрицы предобусловливателя BJIC1(τ) для всех "внутренних" узлов "внутренних" подобластей в подавляющем большинстве случаев нецелесообразно. Заметим, что можно использовать другое упорядочение узлов сетки типа DDO.

При способе 2 применения MPI+OpenMP технологии, рассматриваемого в настоящей работе, применение OpenMP технологии в каждом процессоре с номером s ($s = 1, \ldots, p$) при построении первых $M1$ строк верхнетреугольного множителя матрицы предобусловливания IC1(τ) при новом упорядочении узлов сетки используется цикл по $k2 = 1, \ldots, m$, внутри которого осуществляется вычисление элементов строк искомой матрицы с номерами $1, \ldots, M1$. При этом все рекурсивные вычисления происходят внутри потоков. Для выполнения цикла по $k2$ в настоящей работе использовалась директива `do` с опцией `schedule static`. Затем без использования OpenMP технологии производится вычисление элементов оставшихся строк этой матрицы. Если число узлов во всех "внутренних" подобластях достаточно велико, то для подавляющего большинства строк верхнетреугольного множителя матрицы предобусловливания IC1(τ) вычисление его элементов происходит с использованием OpenMP технологии. Затем с помощью транспонирования производится определение элементов нижнетреугольного множителя матрицы предобусловливания IC1(τ) при новом упорядочении. На этом этапе OpenMP технологии не применяются, так как это оказалось неэффективным.

При обращении предобусловливателя перед началом вычислений элементы вектора r_s, необходимые для вычисления z_s в процессоре с номером s и хранящиеся в памяти других процессоров, должны быть пересланы в процессор с номером s. Затем производится переупорядочение элементов вектора r_s. Использование OpenMP технологии на первом этапе обращения предобусловливателя при новом упорядочении (обращение нижнетреугольных матриц) происходит аналогично использованию OpenMP технологии при вычислении верхнетреугольного множителя матрицы предобусловливания IC1(τ). С использованием OpenMP технологии вычисляются $M1$ элементов искомого вектора с номерами $1, \ldots, M1 = m\bar{l}_s$ (при новом упорядочении), в настоящей работе использовалась директива `do` с опцией `schedule static`. На этапе обращения верхнетреугольных матриц вычисления происходят в обратном порядке, элементы искомого вектора с номерами $M1 = m\bar{l}_s, \ldots, 1$ (при новом упорядочении) вычисляются с использованием OpenMP технологии, в настоящей работе использовалась директива `do` с опцией `schedule static`. После вычисления z_s при новом упорядочении следует вернуться к первоначальному упорядочению элементов этого вектора, выполнить необходимые пересылки и завершить вычисление вектора z.

Вычисление элементов вектора $q_k = Ap_k$, где k — номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, матрица A хранится в памяти в распределенном CRS-формате, с использованием MPI и MPI+OpenMP технологии достаточно хорошо изучено, описано, например в работах [26, 30]. MPI+OpenMP реализация вычислений векторных операций и скалярных произведений в алгоритме 1 тоже хорошо известна. При применении MPI+OpenMP подхода при умножении матрицы на вектор и при вычислении векторных операций и частичных сумм в скалярных произведениях в настоящей работе использовалась директива `do` с опцией `schedule static`.

5. Результаты расчетов. Программы, реализующие применение метода BIIC-IC1-CG для решения СЛАУ (1), были написаны на языке FORTRAN 90 с использованием MPI+OpenMP технологии. Здесь и далее параметр τ в названии предобусловливателя может быть опущен. Расчеты проводились на многопроцессорном вычислительном кластере K60, установленном в ЦКП ИПМ имени М. В. Келдыша РАН. В составе кластера K60 находятся 85 вычислительных узлов, содержащих 2380 ядер (по 28 ядер на узел). Каждый узел представляет собой двухпроцессорный сервер с процессорами Intel Xeon E5-2690 v4. Суммарная пиковая производительность K60 составляет 80,9 терафлопс. Тестирование и сравнение методов производилось с помощью решения модельной задачи — разностной задачи Дирихле для уравнения Пуассона в единичном квадрате на равномерной ортогональной сетке, причем $n = 1048576$. Использовалась стандартная 5-точечная аппроксимация лапласиана (имя матрицы — `5_1048576`). Для тестирования рассматриваемых параллельных методов использовались также некоторые матрицы из коллекции разреженных матриц SuiteSparse [33]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения: `apache2` — трехмерная конечно-разностная схема; `parabolic_fem` — уравнение диффузии-конвекции с постоянным переносом;

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

Таблица 1. Свойства некоторых матриц из коллекции разреженных матриц SuiteSparse
Table 1. Properties of some matrices from SuiteSparse's collection of sparse matrices

Матрица Matrix	n	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_0)$
`apache2`	715176	4817870	2	0	4	8	0.12+7
`parabolic_fem`	525825	3674625	0	1048576	3	7	0.20+6
`ecology2`	999999	4995991	1124	0	3	5	0.63+8
`boneS01`	127224	5516602	127222	2064830	12	67	0.13+8

`ecology2` — приложение теории электрических цепей к задаче передачи генов; `boneS01` — модель трубчатой кости.

В табл. 1 приведены некоторые свойства этих матриц, причем значения $Cond(A_0)$ взяты из работы [35]. Здесь $A_0 = (D_A)^{-1/2} A (D_A)^{-1/2}$ — матрица системы уравнений после масштабирования, Id — количество строк без диагонального преобладания, Ip — количество положительных внедиагональных элементов, NZA — число ненулевых элементов матрицы A, $nz_{\min}$, $nz_{\max}$ — минимальное и максимальное числа ненулевых элементов в строках матрицы A, n — число неизвестных в системе уравнений (1).

Модельную задачу далее будем называть задачей 1. Задачи с матрицами `parabolic_fem`, `apache2`, `ecology2` будем называть соответственно задачами 2, 3 и 4, задачу с более заполненной матрицей `boneS01` будем называть задачей 5.

Решалось уравнение (1) с матрицей $A = A_0$ с единичной правой частью ($b_i \equiv 1$). Начальное приближение было выбрано нулевым ($x_0 \equiv 0$), расчеты продолжались до выполнения условия (2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета на p подобластей при применении только MPI, а также на pm подобластей при применении MPI+OpenMP технологии использовался алгоритм [32]. При решении задач 1–4 методом BIIC-IC1(τ)-CG использовалось значение $\tau = 0.01$, а для задачи 5 с матрицей `boneS01` — $\tau = 0.005$, что было продиктовано требованием безотказности метода BIIC-IC1(τ)-CG.

В табл. 2–6 приведены число итераций и время вычислений методом BIIC-IC1-CG тестовых задач 1–5 при использовании для параллельной реализации MPI и MPI+OpenMP технологии способом 1 и способом 2. Под временем вычислений в табл. 2–6 подразумевается время счета в секундах итерационного процесса в сумме с временем вычисления предобусловливателя. В строках таблиц, соответствующих способу 1 применения MPI+OpenMP технологии, при расчетах элементы матриц $\hat{U}_s^T$ и $\hat{U}_t^T$ не находились в явном виде, использовался неявный способ обращения транспонированных матриц [10]. В строках таблиц, соответствующих способу 2 применения MPI+OpenMP технологии, элементы матриц $\hat{U}_s^T$ и $\hat{U}_t^T$ определялись с помощью транспонирования. При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 12, 16 нитей. В табл. 2–6 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей, которые приведены в таблицах в скобках. В табл. 2–6 приведены также коэффициенты ускорения счета на p процессорах (μ) благодаря использованию способов 1 и 2 применения MPI+OpenMP технологии.

Таблица 2. Число итераций и время вычислений методом BIIC-IC1-CG задачи с матрицей `5_1048576` на p процессорах без применения и с применением OpenMP технологии
Table 2. Number of iterations and counting time by the BIIC-IC1-CG method of the problem with the matrix `5_1048576` on p processors without and with the use of OpenMP technology

Способ Way	Технология Technology	$p = 2$	$p = 4$	$p = 8$	$p = 10$	$p = 16$
1	MPI	401, 19.55	401, 11.02	435, 6.67	427, 5.04	444, 3.86
	MPI+OpenMP	469, 6.13(12)	444, 6.33(4)	469, 4.05(3)	493, 4.17(3)	493, 4.16(3)
	μ	3.19	1.74	1.65	1.2	0.93
2	MPI	401, 18.94	401, 10.76	435, 6.4	427, 4.8	444, 3.77
	MPI+OpenMP	449, 5.99(12)	474, 4.51(6)	530, 3.9(3)	530, 4.18(4)	553, 3.57(3)
	μ	3.16	2.38	1.64	1.15	1.06

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

Таблица 3. Число итераций и время вычислений методом BIIC-IC1-CG задачи с матрицей `parabolic_fem` на p процессорах без применения и с применением OpenMP технологии
Table 3. Number of iterations and counting time by the BIIC-IC1-CG method of the problem with the matrix `parabolic_fem` on p processors without and with the use of OpenMP technology

Способ Way	Технология Technology	$p = 2$	$p = 4$	$p = 8$	$p = 10$	$p = 16$
1	MPI	415, 10.98	418, 5.79	453, 3.54	463, 3.09	353, 1.80
	MPI+OpenMP	454, 3.19(12)	453, 3.14(4)	454, 2.08(3)	497, 2.66(4)	509, 2.28(3)
	μ	3.44	1.84	1.57	1.16	0.79
2	MPI	415, 10.34	418, 5.7	440, 3.25	463, 3.00	453, 1.78
	MPI+OpenMP	526, 4.1(12)	533, 2.79(6)	521, 2.20(3)	586, 2.72(4)	574, 2.29(3)
	μ	2.52	2.04	1.48	1.1	0.79

Таблица 4. Число итераций и время вычислений методом BIIC-IC1-CG задачи с матрицей `apache2` на p процессорах без применения и с применением OpenMP технологии
Table 4. Number of iterations and counting time by the BIIC-IC1-CG method of the problem with the matrix `apache2` on p processors without and with the use of OpenMP technology

Способ Way	Технология Technology	$p = 2$	$p = 4$	$p = 8$	$p = 10$	$p = 16$
1	MPI	489, 16.69	501, 11.21	529, 5.36	613, 5.15	542, 2.97
	MPI+OpenMP	542, 6.78(8)	611, 6.89(8)	577, 4.3(3)	618, 5.29(3)	645, 4.5(3)
	μ	2.46	1.63	1.25	0.86	0.66
2	MPI	489, 16.48	501, 10.87	529, 5.20	540, 4.38	542, 2.99
	MPI+OpenMP	538, 5.7(12)	630, 4.58(6)	720, 4.25(3)	776, 4.28(4)	850, 3.94(3)
	μ	2.89	2.37	1.22	1.02	0.75

Таблица 5. Число итераций и время вычислений методом BIIC-IC1-CG задачи с матрицей `ecology2` на p процессорах без применения и с применением OpenMP технологии
Table 5. Number of iterations and counting time by the BIIC-IC1-CG method of the problem with the matrix `ecology2` on p processors without and with the use of OpenMP technology

Способ Way	Технология Technology	$p = 2$	$p = 4$	$p = 8$	$p = 10$	$p = 16$
1	MPI	697, 31.84	721, 17.34	740, 9.87	751, 8.25	790, 6.27
	MPI+OpenMP	809, 10.19(12)	790, 9.98(4)	809, 6.68(3)	846, 8.2(4)	854, 7.07(3)
	μ	3.12	1.74	1.48	1.01	0.88
2	MPI	697, 30.87	721, 16.64	740, 9.32	751, 7.96	790, 6.03
	MPI+OpenMP	768, 9.29(12)	843, 6.65(6)	878, 5.68(3)	952, 6.7(4)	852, 5.49(3)
	μ	3.32	2.5	1.64	1.19	1.1

Таблица 6. Число итераций и время вычислений методом BIIC-IC1-CG задачи с матрицей `boneS01` на p процессорах без применения и с применением OpenMP технологии
Table 6. Number of iterations and counting time by the BIIC-IC1-CG method of the problem with the matrix `boneS01` on p processors without and with the use of OpenMP technology

Способ Way	Технология Technology	$p = 2$	$p = 4$	$p = 8$	$p = 10$	$p = 16$
1	MPI	301, 10.09	348, 6.19	335, 3.28	376, 3.02	380, 2.04
	MPI+OpenMP	365, 3.4(12)	402, 3.2(12)	365, 2.44(3)	378, 2.99(3)	404, 2.32(3)
	μ	2.96	1.93	1.34	1.01	0.88
2	MPI	301, 10.24	348, 6.26	335, 3.49	376, 3.16	380, 2.1
	MPI+OpenMP	387, 7.89(4)	440, 4.95(6)	471, 3.48(3)	512, 3.76(3)	577, 3.55(3)
	μ	1.3	1.26	1.0	0.84	0.59

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

На рис. 1–5 приведены графики зависимостей времени счета тестовых задач методом BIIC-IC1-CG от числа процессоров p с использованием только MPI и MPI+OpenMP технологий. Сплошные линии соответствуют расчетам с использованием только MPI и разбиения области расчета на p подобластей с помощью алгоритма [32]. При этом сплошными линиями с квадратами (var1) представлены графики, полученные по результатам решения тестовых задач с использованием неявного способа обращения матриц $\hat{U}_s^T$, а сплошными линиями с треугольниками (var2) — с использованием вычисления матриц $\hat{U}_s^T$ в явном виде. Штриховыми линиями с квадратами, обозначенными var1,th, нарисованы графики зависимостей времени счета тестовых задач от числа процессоров p с использованием технологий MPI+OpenMP способом 1. Штриховыми линиями с треугольниками, обозначенными var2.1,th, показаны графики, полученные по результатам расчетов с использованием технологий MPI+OpenMP способом 2. На рис. 1–5 для сравнения приведены графики зависимостей времени счета тестовых задач от числа процессоров p с использованием способа 2 применения MPI+OpenMP технологий при использовании разбиения 2 из работы [31], кратко описанного в разделе 4 (штрих-пунктирные линии, перевернутые треугольники, обозначены var2.2,th). Все графики построены в логарифмическом масштабе.

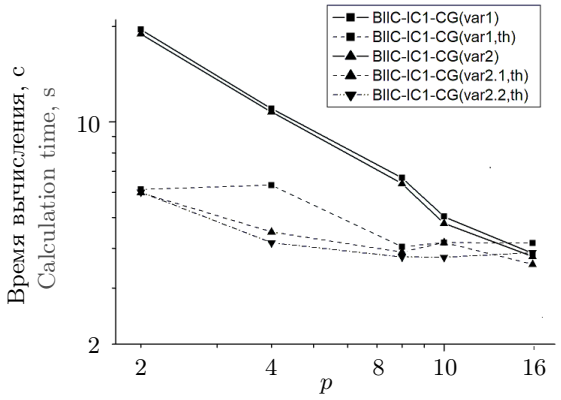

Рис. 1. Время вычисления задачи с матрицей `5_1048576` методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP

Fig. 1. Counting times of the problem with the matrix `5_1048576` by the BIIC-IC1-CG method using MPI and MPI+OpenMP

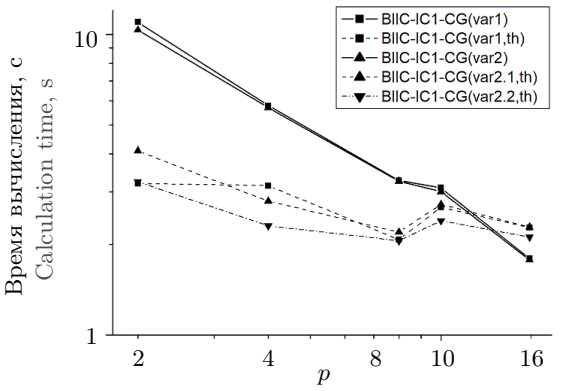

Рис. 2. Время вычисления задачи с матрицей `parabolic_fem` методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP

Fig. 2. Counting time of the problem with the matrix `parabolic_fem` by the BIIC-IC1-CG method using MPI and MPI+OpenMP

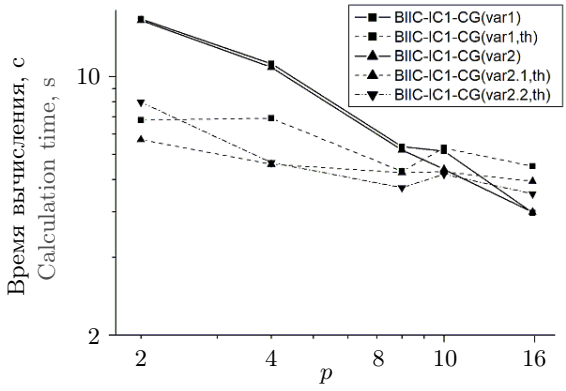

Рис. 3. Время вычисления задачи с матрицей `apache2` методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP

Fig. 3. Counting time of the problem with the matrix `apache2` by the BIIC-IC1-CG method using MPI and MPI+OpenMP

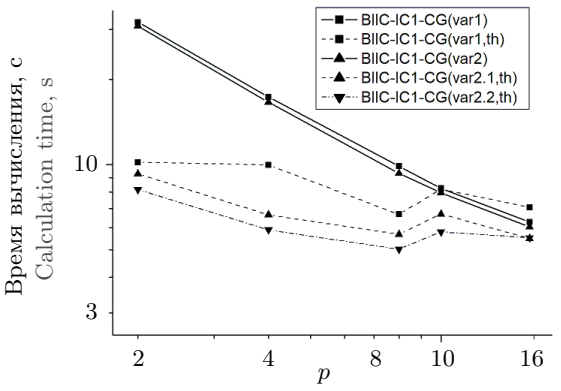

Рис. 4. Время вычисления задачи с матрицей `ecology2` методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP

Fig. 4. Counting time of the problem with the matrix `ecology2` by the BIIC-IC1-CG method using MPI and MPI+OpenMP

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

Как видно из табл. 2–6 и рис. 1–5, использование способа 1 применения MPI+OpenMP технологии для решения всех задач 1–5 позволяет значительно ускорить решение этих задач, по сравнению с использованием только MPI, при $p < 10$. При $p = 10, 16$ применение OpenMP технологии для решения тестовых задач методом BIIC-IC1-CG становится нецелесообразным.

Как видно из табл. 2–6 и рис. 1–4, использование способа 2 применения MPI+OpenMP технологии позволяет значительно ускорить вычисления по сравнению с использованием только MPI при решении задач с сильно разреженными матрицами: задач 1, 2, 4 при $p \leqslant 10$, а задачи 3 — при $p < 10$. Как правило, решение задач 1–4 с сильно разреженными матрицами с использованием способа 2 применения MPI+OpenMP технологии происходило быстрее, чем с использованием способа 1 применения MPI+OpenMP технологии.

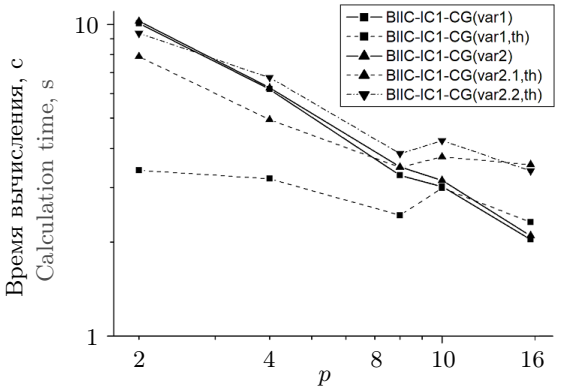

Рис. 5. Время вычисления задачи с матрицей `boneS01` методом BIIC-IC1-CG с использованием MPI и MPI+OpenMP

Fig. 5. Counting time of the problem with the matrix `boneS01` by the BIIC-IC1-CG method using MPI and MPI+OpenMP

Использование предложенного в настоящей работе способа 2 применения MPI+OpenMP технологии при решении задачи 5 (с более плотной матрицей и маленьким значением τ) методом BIIC-IC1(τ)-CG не позволило так же сильно, как в случае более разреженных матриц, уменьшить время счета по сравнению с временем счета с использованием только MPI. Причинами этого являются прежде всего большое число узлов сетки, попавших на разделители при разбиении на внутренние подобласти, и большое время вычисления матрицы предобусловливания. Ускорение счета благодаря использованию OpenMP технологии этим способом наблюдалось только при $p = 2, 4$ и разбиении всей области расчета на pm подобластей с помощью алгоритма [32]. Заметим, что использование предложенного способа 2 применения OpenMP технологии с разбиением 2 из работы [31] при решении задачи 5 методом BIIC-IC1(τ)-CG оказалось нецелесообразным. Разница в эффективности способа 2 применения MPI+OpenMP технологии при этих разбиениях может быть объяснена особенностями разбиения всей области расчета на pm подобластей с помощью алгоритма из работы [32], одной из задач которого является концентрация большей части ненулевых элементов матрицы в ее блочно-диагональной части, содержащей при таком разбиении не менее pm блоков. Заметим, что для сильно разреженных матриц эта проблема не является настолько актуальной.

Как видно из табл. 2–6, при использовании способов 1 и 2 применения MPI+OpenMP технологии по-разному происходит изменение числа итераций с ростом числа потоков.

На рис. 6, 7 приведены графики ускорения вычислений тестовых задач методом BIIC-IC1(τ)-CG с применением только MPI (сплошные линии) и MPI+OpenMP (штриховые линии) способами 1 и 2 по сравнению со счетом на двух процессорах с применением только MPI. При вычислении ускорения счета использовалось время счета задач на p процессорах с применением только MPI с разбиением области расчета на p подобластей с помощью алгоритма [32], а с применением MPI+OpenMP технологии использовалось время счета при разбиении области расчета на pm подобластей с помощью алгоритма [32].

Как видно из рис. 6, использование способа 1 применения MPI+OpenMP технологии для решения тестовых задач 1, 2, 4, 5 позволило получить сверхлинейное ускорение при $p < 10$, а задачи 3 — при $p < 8$. Как видно из рис. 7, использование способа 2 применения MPI+OpenMP технологии, предложенного в настоящей работе, для решения тестовых задач 1, 2, 4 позволило получить сверхлинейное ускорение при $p < 10$, задачи 3 — при $p < 8$, а задачи 5 только при $p = 2, 4$.

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 в каждом процессоре. В работах [3, 30] на примере решения модельной задачи, описанной в начале раздела 5, при различных значениях размерности матрицы n методом BIIC-IC2S(τ)-CG с нулевым налеганием (являющимся методом CG с предобусловливанием блочного Якоби в сочетании с IC2S(τ)) показано, что при фиксированном числе процессоров ускорение счета бла-

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

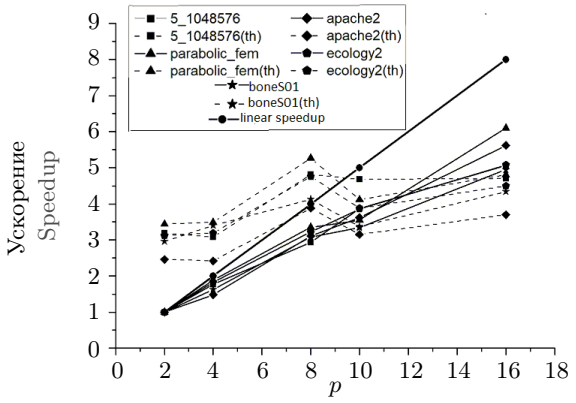

Рис. 6. Ускорение вычислений тестовых задач методом BIIC-IC1-CG при применении MPI и применении MPI+OpenMP технологии способом 1

Fig. 6. Accelerating the calculation of test problems by the method BIIC-IC1-CG when using MPI and when using MPI+OpenMP technology way 1

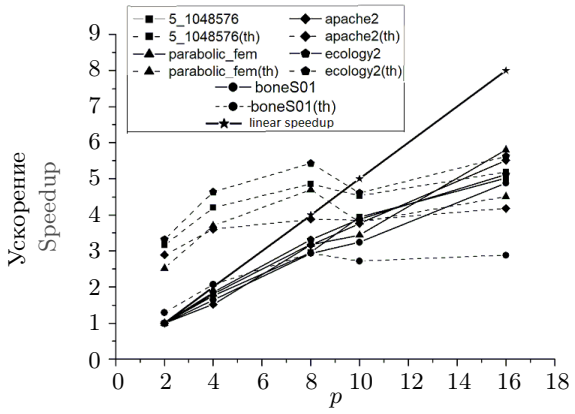

Рис. 7. Ускорение вычислений тестовых задач методом BIIC-IC1-CG при применении MPI и применении MPI+OpenMP технологии способом 2

Fig. 7. Accelerating the calculation of test problems by the method BIIC-IC1-CG when using MPI and when using MPI+OpenMP technology way 2

годаря использованию OpenMP технологии растет с ростом n. В настоящей работе в качестве тестовых матриц использовались матрицы относительно небольшого размера. При расчетах реальных физических задач размеры матриц, как правило, значительно больше. Следует ожидать, что потеря эффективности от применения OpenMP технологии при решении задач с большими размерами матриц наступит при значительно большем числе процессоров.

Используя результаты расчетов, приведенные в работах [3, 31], можно произвести сравнение времени счета задач с матрицами `5_1048576`, `parabolic_fem`, `apache2`, `ecology2`, а также необходимого числа итераций при решении этих тестовых задач методами BIIC-IC1(τ)-CG и BIIC-IC2S(τ)-CG при одинаковом значении τ. Решение задач с этими матрицами методом BIIC-IC2S(τ)-CG ($\tau = 0.01$) с использованием MPI и MPI+OpenMP технологии способом 1 происходило лишь немного быстрее, чем методом BIIC-IC1(τ)-CG ($\tau = 0.01$). При этом число итераций метода BIIC-IC1(τ)-CG было всего в 1.4–1.1 раза больше числа итераций метода BIIC-IC2S(τ)-CG при одинаковом числе блоков в этих методах, с увеличением числа блоков разница в числе итераций уменьшалась. Скорее всего, небольшая разница в числе итераций связана с тем, что основное влияние на скорость сходимости этих методов оказывает использование блоков с налеганием в предобусловливателях BIIC-IC2S(τ) и BIIC-IC1(τ).

Как показано в работе [31], решение задачи с матрицей `boneS01` методом BIIC-IC2S(τ)-CG с использованием MPI и MPI+OpenMP технологии способом 1 при $\tau = 0.005$ происходило гораздо медленнее, чем методом BIIC-IC1(τ)-CG при $\tau = 0.005$. Однако, решение этой задачи методом BIIC-IC2S(τ)-CG при $\tau = 0.01$, 0.02 почти всегда происходило быстрее, чем методом BIIC-IC1(τ)-CG при $\tau = 0.005$. При $\tau = 0.01$, 0.02 метод BIIC-IC1(τ)-CG не является безотказным. Таким образом, решение задачи с матрицей `boneS01` методом BIIC-IC2S(τ)-CG с использованием MPI и MPI+OpenMP технологии можно произвести лишь немного быстрее, чем методом BIIC-IC1(τ)-CG, благодаря безотказности BIIC-IC2S(τ)-CG при любых значениях τ. Заметим, что в работе [30] показано, что при решении задач с матрицами `cfd2` и `offshore` методом CG с предобусловливанием блочного Якоби в сочетании с IC1(τ) (BJIC1(τ)) время счета было значительно больше, чем при их решении методом BJIC2S(τ)-CG, из-за необходимости использования чрезмерно малого значения параметра τ в предобусловливателе BJIC1(τ) для безотказности метода BJIC1(τ)-CG.

Таким образом, если для обеспечения безотказности метода BIIC-IC1(τ)-CG не нужно использовать чрезмерно малые значения τ, то, учитывая простоту алгоритма построения предобусловливателя BIIC-IC1(τ) по сравнению с затратным алгоритмом построения предобусловливателя BIIC-IC2S(τ), для решения СЛАУ (1) может оказаться целесообразным использовать BIIC-IC1(τ)-CG вместо BIIC-IC2S(τ)-CG.

6. Заключение. В работе рассмотрен предобусловливатель BIIC-IC1(τ) для решения СЛАУ (1) методом сопряженных градиентов и способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC-IC1(τ) с числом блоков, кратным числу используемых процессоров и числу используемых потоков — способ 1. Предложен способ применения MPI+OpenMP технологии

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

для построения и обращения предобусловливателя BIIC-IC1(τ) с числом блоков, кратным числу процессоров, в котором для мелкозернистого распараллеливания используется переупорядочение узлов сетки внутри расширенных подобластей типа DDO — способ 2. С помощью расчетов модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse показано, что использование способа 1 применения MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI при решении СЛАУ (1) методом BIIC-IC1(τ)-CG для не слишком большого числа узлов суперкомпьютерной системы. С помощью расчетов этих задач показано, что использование способа 2 применения MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для не слишком большого числа узлов суперкомпьютерной системы при решении задач с достаточно разреженными матрицами. При использовании способа 2 применения MPI+OpenMP технологии для решения задачи с более заполненной матрицей удалось получить небольшое ускорение счета по сравнению со счетом с применением только MPI для маленького числа процессоров, причем только при хорошем разбиении области расчета. Проведенный в работе анализ результатов расчетов тестовых задач показал, что, учитывая относительную простоту алгоритма построения предобусловливателя BIIC-IC1(τ), в ряде случаев для решения СЛАУ (1) может оказаться целесообразным использовать метод BIIC-IC1(τ)-CG вместо метода BIIC-IC2S(τ)-CG.

Список литературы

1. *Kaporin I.E.* New convergence results and preconditioning strategies for the conjugate gradient method // Numer. Linear Algebra Appl. 1994. **1**, N 2. 179–210.
2. *Капорин И.Е.* О предобусловливании метода сопряженных градиентов при решении дискретных аналогов дифференциальных задач // Дифференциальные уравнения. 1990. **26**, № 7. 1225–1236.
3. *Милюкова О.Ю.* MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1. Препринт № 48 ИПМ им. М.В. Келдыша РАН. М., 2021. doi 10.20948/prepr-2021-48.
4. *Munksgaard N.* Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Softw. 1980. **6**, N 2. 206–219.
5. *Tuff A.D., Jennings A.* An iterative method for large systems of linear structural equations // Int. J. Numer. Methods Eng. 1973. **7**, N 2. 175–183.
6. *Ajiz M.A., Jennings A.* A robust incomplete Choleski-conjugate gradient algorithm // Int. J. Numer. Methods Eng. 1984. **20**, N 5. 949–966.
7. *Kaporin I.E.* High quality preconditionings of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition // Numer. Linear. Algebra Appl. 1998. **5**, N 6. 483–509.
8. *Капорин И.Е.* Предобусловливание систем линейных алгебраических уравнений. Диссертация на соискание степени доктора физ.-мат. наук. М., 2011.
9. *Капорин И.Е., Коньшин И.Н.* Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Ж. вычисл. матем. и матем. физики. 2001. **41**, № 4. 515–528.
10. *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). Вып. 2. М.: ВЦ РАН, 2011. 32–49.
11. *Duff I.S., Meurant G.A.* The effect of ordering on preconditioned conjugate gradients // BIT Numer. Math. 1989. **29**. 625–657.
12. *Doi S.* On parallelism and convergence of incomplete LU factorizations // Appl. Numer. Math. 1991. **7**, N 5. 417–436. doi 10.1016/0168-9274(91)90011-N.
13. *Notay Y.* An efficient parallel discrete PDE solver // Parallel Computing. 1995. **21**. 1725–1748. doi 10.1016/0167-8191(96)80005-6.
14. *Milyukova O.Yu.* Parallel approximate factorization method for solving discreate elliptic equations // Parallel Comput. 2001. **27**, N 10. 1365–1379. doi 10.1016/S0167-8191(01)00092-8.
15. *Милюкова О.Ю.* Некоторые параллельные итерационные методы с факторизованными матрицами предобусловливания для решения эллиптических уравнений на треугольных сетках // Ж. вычисл. матем. и матем. физики. 2006. **46**, № 6. 1096–1113.
16. *Hysom D., Pothen A.* A scalable parallel algorithm for incomplete factor preconditioning // SIAM J. Sci. Comput. 2001. **22**, N 6. 2194–2215. doi 10.1137/S1064827500376193.

17. *Magolu monga Made M., van der Vorst H.A.* Spectral analysis of parallel incomplete factorizations with implicit pseudo-overlap // Numer. Linear Algebra Appl. 2002. **9**, N 1. 45–64. doi 10.1002/nla.247.
18. *Милюкова О.Ю.* Сочетание числовых и структурных подходов к построению неполного треугольного разложения второго порядка в параллельных методах предобусловливания // Ж. вычисл. матем. и матем. физ. 2016. **56**, № 5. 711–729. doi 10.7868/S0044466916050161.
19. *Anderson E.C., Saad Y.* Solving sparse triangular systems on parallel computers // Int. J. High Speed Comput. 1989. **1**. 73–96.
20. *Hammond S.W., Schreiber R.* Efficient ICCG on a shared memory multiprocessor // Int. J. High Speed Comput. **4**, N 01. 1992. 1–21. doi 10.1142/S0129053392000183.
21. *Wolf M.M., Heroux M.A., Boman E.G.* Factors impacting performance of multithreaded sparse triangular solve // Lecture Notes in Computer Science. Vol. 6449. Heidelberg: Springer, 2011. 2–44. doi 10.1007/978-3-642-19328-6_6.
22. *Chow E., Anzt H., Scott J., Dongarra J.* Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning // J. Parallel Distrib. Comput. 2018. **119**. 219–230. doi 10.1016/j.jpdc.2018.04.017.
23. *Chow E., Patel A.* Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comput. 2015. **37**, N 2. 169–193. doi 10.1137/140968896.
24. *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. Oxford: Harwell, 2018.
25. *Капорин И.Е., Милюкова О.Ю.* MPI+OpenMP параллельная реализация метода сопряженных градиентов с некоторыми явными предобусловливателями. Препринт № 8 ИПМ им. М.В. Келдыша РАН. М., 2018. doi 10.20948/prepr-2018-8.
26. *Капорин И.Е., Милюкова О.Ю.* MPI+OpenMP реализация метода сопряженных градиентов с факторизованными явными предобусловливателями // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. 2018. Вып. 4. 57–69.
27. *Chow E.* Parallel implementation and practical use of sparse approximate inverse preconditioners with a priori sparsity patterns // Int. J. High Performance Comput. Appl. 2001. **15**, N 1. 56–74.
28. *Милюкова О.Ю.* MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем. Препринт № 31 ИПМ им. М.В. Келдыша РАН. М., 2020. doi 10.20948/prepr-2020-31.
29. *Милюкова О.Ю.* MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1. Препринт № 83 ИПМ им. М.В. Келдыша РАН. М., 2020. doi 10.20948/prepr-2020-83.
30. *Милюкова О.Ю.* MPI+OpenMP реализация метода сопряженных градиентов с факторизованными неявными предобусловливателями // Математическое моделирование. 2021. **33**, № 10. 19–38. doi 10.20948/mm-2021-10-02.
31. *Милюкова О.Ю.* Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливанием блочного неполного обратного треугольного разложения IC1. Препринт № 2 ИПМ им. М.В. Келдыша РАН. М., 2022. doi 10.20948/prepr-2022-2.
32. *Капорин И.Е., Милюкова О.Ю.* Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов. Препринт № 37 ИПМ им. М.В. Келдыша РАН. М., 2017. doi 10.20948/prepr-2017-37.
33. *Davis T.A., Hu Y.* The university of Florida sparse matrix collection // ACM Trans. Math. Softw. 2011. **38**, N 1. 1:1–1:25. doi 10.1145/2049662.2049663.
34. *Axelsson O.* Iterative solution methods. New York: Cambridge Univ. Press, 1994.
35. *Капорин И.Е.* Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. **52**, № 2. 179–204.

Поступила в редакцию
20 мая 2022 г.

Принята к публикации
11 июля 2022 г.

Информация об авторе

Ольга Юрьевна Милюкова — д.ф.-м.н., вед. научн. сотр., Институт прикладной математики имени М. В. Келдыша РАН (ИПМ РАН), Миусская площадь, д. 4, 125047, Москва, Российская Федерация.

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

References

1. I. E. Kaporin, "New Convergence Results and Preconditioning Strategies for the Conjugate Gradient Method," Numer. Linear Algebra Appl. **1** (2), 179–210 (1994).
2. I. E. Kaporin, "A Preconditioned Conjugate-Gradient Method for Solving Discrete Analogs of Differential Problems," Differ. Uravn. **26** (7), 1225–1236 (1990) [Differ. Equ. **26** (7), 897–906 (1990)].
3. O. Yu. Milyukova, *MPI+OpenMP Parallel Implementation of Conjugate Gradient Method with Preconditioner of Block Partial Inverse Triangular Decomposition of IC2S and IC1*, Preprint No. 48 (Keldysh Institute of Applied Mathematics, Moscow, 2021). doi 10.20948/prepr-2021-48.
4. N. Munksgaard, "Solving Sparse Symmetric Sets of Linear Equations by Preconditioned Conjugate Gradients," ACM Trans. Math. Softw. **6** (2), 206–219 (1980).
5. A. D. Tuff and A. Jennings, "An Iterative Method for Large Systems of Linear Structural Equations," Int. J. Numer. Methods Eng. **7** (2), 175–183 (1973).
6. M. A. Ajiz and A. Jennings, "A Robust Incomplete Choleski-Conjugate Gradient Algorithm," Int. J. Numer. Methods Eng. **20** (5), 949–966 (1984).
7. I. E. Kaporin, "High Quality Preconditionings of a General Symmetric Positive Definite Matrix Based on Its $U^TU + U^TR + R^TU$-Decomposition," Numer. Linear Algebra Appl. **5** (6), 483–509 (1998).
8. I. E. Kaporin, *Preconditioning of Systems of Linear Algebraic Equations.* Doctoral Dissertation in Mathematics and Physics (Moscow State Univ., Moscow, 2011).
9. I. E. Kaporin and I. N. Kon'shin, "Parallel Solution of Symmetric Positive Definite Systems Based on Decomposition into Overlapping Blocks," Zh. Vychisl. Mat. Mat. Fiz. **41** (4), 515–528 (2001) [Comput. Math. Math. Phys. **41** (4), 481–493 (2001)].
10. I. E. Kaporin and O. Yu. Milyukova, "Massively Parallel Preconditioned Conjugate Gradient Algorithm for the Numerical Solution of Linear Algebraic Equations," in *Optimization and Applications* (Comput. Center Ross. Akad. Nauk, Moscow, 2011), Issue 2, pp. 32–49.
11. I. S. Duff and G. A. Meurant, "The Effect of Ordering on Preconditioned Conjugate Gradients," BIT Numer. Math. **29**, 635–657 (1989).
12. S. Doi, "On Parallelism and Convergence of Incomplete LU Factorizations," Appl. Numer. Math. **7** (5), 417–436 (1991). doi 10.1016/0168-9274(91)90011-N.
13. Y. Notay, "An Efficient Parallel Discrete PDE Solver," Parallel Comput. **21** (11), 1725–1748 (1995). doi 10.1016/0167-8191(96)80005-6.
14. O. Yu. Milyukova, "Parallel Approximate Factorization Method for Solving Discrete Elliptic Equations," Parallel Comput. **27** (10), 1365–1379 (2001). doi 10.1016/S0167-8191(01)00092-8.
15. O. Yu. Milyukova, "Parallel Iterative Methods Using Factorized Preconditioning Matrices for Solving Elliptic Equations on Triangular Grids," Zh. Vychisl. Mat. Mat. Fiz. **46** (6), 1096–1113 (2006) [Comput. Math. Math. Phys. **46** (6), 1044–1060 (2006)]. doi 10.1134/S096554250606011X.
16. D. Hysom and A. Pothen, "A Scalable Parallel Algorithm for Incomplete Factor Preconditioning," SIAM J. Sci. Comput. **22** (6), 2194–2215 (2001). doi 10.1137/S1064827500376193.
17. M. Magolu monga Made and H. A. van der Vorst, "Spectral Analysis of Parallel Incomplete Factorizations with Implicit Pseudo-Overlap," Numer. Linear Algebra Appl. **9** (1), 45–64 (2002). doi 10.1002/nla.247.
18. O. Yu. Milyukova, "Combination of Numerical and Structured Approaches to the Construction of a Second-Order Incomplete Triangular Factorization in Parallel Preconditioning Methods," Zh. Vychisl. Mat. Mat. Fiz. **56** (5), 711–729 (2016) [Comput. Math. Math. Phys. **56** (5), 699–716 (2016)]. doi 10.1134/S096554251605016X.
19. E. C. Anderson and Y. Saad, "Solving Sparse Triangular Systems on Parallel Computers," Int. J. High Speed Comput. **1**, 73–96 (1989).
20. S. W. Hammond and R. Schreiber, "Efficient ICCG on a Shared Memory Multiprocessor," Int. J. High Speed Comput. **4** (01), 1–21 (1992). doi 10.1142/S0129053392000183.
21. M. M. Wolf, M. A. Heroux, and E. G. Boman, "Factors Impacting Performance of Multithreaded Sparse Triangular Solve," in *Lecture Notes in Computer Science* (Springer, Heidelberg, 2011), Vol. 6449, pp. 32–44. doi 10.1007/978-3-642-19328-6_6.
22. E. Chow, H. Anzt, J. Scott, and J. Dongarra, "Using Jacobi Iterations and Blocking for Solving Sparse Triangular Systems in Incomplete Factorization Preconditioning," J. Parallel Distrib. Comput. **119**, 219–230 (2018). doi 10.1016/j.jpdc.2018.04.017.
23. E. Chow and A. Patel, "Fine-Grained Parallel Incomplete LU Factorization," SIAM J. Sci. Comput. **37** (2), 169–193 (2015). doi 10.1137/140968896.

2022, **23** (3), 191–206. doi 10.26089/NumMet.v23r312

24. S. Cayrols, I. Duff, and F. Lopes, *Parallelization of the Solve Phase in a Task-based Cholesky Solver Using a Sequential Task Flow Model*, Technical Report RAL-TR-2018-008 (Harwell, Oxford, 2018).
25. I. E. Kaporin and O. Yu. Milyukova, *MPI+OpenMP Parallel Implementation of Explicitly Preconditioned Conjugate Gradient Method*, Preprint No. 8 (Keldysh Institute of Applied Mathematics, Moscow, 2018). doi 10.20948/prepr-2018-8.
26. I. E. Kaporin and O. Yu. Milyukova, "MPI+OpenMP Parallel Implementation of the Conjugate Gradient Method with Factorized Explicit Preconditioners," Voprosy Atomn. Nauki Tekhniki, Ser.: Mat. Mod. Fiz. Proc. Issue 4, 57–69 (2018).
27. E. Chow, "Parallel Implementation and Practical Use of Sparse Approximate Inverse Preconditioners with a Priori Sparsity Patterns," Int. J. High Perform. Comput. Appl. **15** (1), 56–74 (2001). doi 10.1177/109434200101500106.
28. O. Yu. Milyukova, *MPI+OpenMP Parallel Implementation of Conjugate Gradient Method with Factorized Preconditioner*, Preprint No. 31 (Keldysh Institute of Applied Mathematics, Moscow, 2020). doi 10.20948/prepr-2020-31.
29. O. Yu. Milyukova, *MPI+OpenMP Parallel Implementation of Conjugate Gradient Method with the Block Jacobi Preconditioner Combined with IC1*, Preprint No. 83 (Keldysh Institute of Applied Mathematics, Moscow, 2020). doi 10.20948/prepr-2020-83.
30. O. Yu. Milyukova, "MPI+OpenMP Parallel Implementation of Conjugate Gradient Method with Factored Implicit Preconditioners," Mat. Model. **33** (10), 19–38 (2021) doi 10.20948/mm-2021-10-02.
31. O. Yu. Milyukova, *Approaches MPI+OpenMP Implementation of Conjugated Gradient Method with Pre-conditioner of Block Incomplete Inverse Triangular Decomposition of IC1*, Preprint No. 2 (Keldysh Institute of Applied Mathematics, Moscow, 2022). doi 10.20948/prepr-2022-2.
32. I. E. Kaporin and O. Yu. Milyukova, *Incomplete Inverse Triangular Factorization in Parallel Algorithms of Preconditioned Conjugate Gradient Methods*, Preprint No. 37 (Keldysh Institute of Applied Mathematics, Moscow, 2017). doi 10.20948/prepr-2017-37.
33. T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection," ACM Trans. Math. Softw. **38** (1), 1:1–1:25 (2011). doi 10.1145/2049662.2049663.
34. O. Axelsson, *Iterative Solution Methods* (Cambridge Univ. Press, New York, 1994).
35. I. E. Kaporin, "Using Chebyshev Polynomials and Approximate Inverse Triangular Factorizations for Preconditioning the Conjugate Gradient Method," Zh. Vychisl. Mat. Mat. Fiz. **52** (2), 179–204 (2012) [Comput. Math. Math. Phys. **52** (2), 169–193 (2012)]. doi 10.1134/S0965542512020091.

Received
May 20, 2022

Accepted for publication
July 11, 2022

Information about the author

Olga Yu. Milyukova — Dr. Sci., Leading Scientist, Keldysh Institute of Applied Mathematics, Miusskaya ploshchad' 4, 125047, Moscow, Russia.