ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 18 за 2023 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

О.Ю. Милюкова

MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем на основе использования переупорядочения узлов сетки

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В.Келдыша. 2023. № 18. 29 с. https://doi.org/10.20948/prepr-2023-18
https://library.keldysh.ru/preprint.asp?id=2023-18

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М. В. Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

О. Ю. Милюкова

MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем на основе использования переупорядочения узлов сетки

Москва — 2023

Милюкова О.Ю.

MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем на основе использования переупорядочения узлов сетки

В работе предлагаются способы применения MPI и MPI+OpenMP технологии для построения и обращения факторизованного предобусловливателя типа неполного треугольного разложения без заполнения для решения систем линейных алгебраических уравнений с произвольной симметричной положительно определенной матрицей. Способы применения MPI и MPI+OpenMP технологии основаны на использования упорядочений узлов сетки, согласованных с разбиением области расчета. Применение OpenMP технологии при построении и обращении предобусловливателя осуществляется для большинства строк матрицы. Проводится сравнение времени решения задач методом сопряженных градиентов с рассматриваемым предобусловливателем с использованием MPI и гибридной MPI+OpenMP технологии на примере модельных задач и ряда задач из коллекции разреженных матриц SuiteSparse.

Ключевые слова: неполное треугольное разложение, переупорядочение узлов сетки, параллельное предобусловливание, метод сопряженных градиентов

Olga Yurievna Milyukova

MPI+OpenMP implementation of the conjugate gradient method with a factorized preconditioner based on the use of grid node reordering

The paper proposes ways to use MPI and MPI+OpenMP technologies for constructing and inverting a factorized preconditioner of the type incomplete triangular decomposition for solving systems of linear algebraic equations with an arbitrary symmetric positive definite matrix. Methods of using MPI and MPI+OpenMP technologies are based on the use of grid node orderings consistent with the division of the calculation area. The use of OpenMP technology in the construction and inversion of the preconditioner is carried out for most rows of the matrix. Comparative timing results for the MPI+OpenMP and MPI implementations of the proposed preconditioning used with the conjugate gradient method for a model problems and the sparse matrix collections SuiteSparse are presented.

Keywords: incomplete triangular decomposition, Domain Decomposition ordering, parallel preconditioning, conjugate gradient method

1. Введение

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

$$Ax = b \tag{1.1}$$

с симметричной положительно определенной разреженной матрицей A общего вида

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

Проблема построения эффективных численных методов решения СЛАУ (1.1) 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.

В настоящей работе для решения СЛАУ (1.1) большого размера будем применять предобусловленный метод сопряженных градиентов (CG), итерации которого осуществляются до выполнения условия

$$\|b - Ax_k\| \le \varepsilon \|b - Ax_0\|, \text{ где } 0 < \varepsilon << 1. \tag{1.2}$$

Будем использовать неявный факторизованный предобусловливатель типа неполного треугольного разложения без заполнения, предложенный в работе [1], который будем называть вариантом неполного треугольного разложения Холецкого без заполнения IC(0) (Incomplete Cholesky) [2] и обозначать VIC (Version Incomplete Cholesky). В предобусловливании VIC используется факторизованная матрица предобусловливания

$$B \approx A, B = (D^{-1} + A^-)D(D^{-1} + A^+), \tag{1.3}$$

где A^- – строго нижнетреугольная часть матрицы A, A^+ – строго верхнетреугольная часть матрицы A. В предобусловливателе VIC элементы диагональной матрицы D определяются из условия совпадения диагональных элементов матриц A и B. Предобусловливание VIC имеет ограниченную область применимости. В частности доказано, что метод VIC-CG применим в случае положительных диагональных элементов, отрицательных внедиагональных элементов и наличии диагонального преобладания у симметричной положительно определенной матрицы A [3]. В этом случае все диагональные элементы матрицы D положительны.

Решение задач с матрицами большого размера требует применения параллельных компьютеров. Основная трудность распараллеливания алгоритмов построения и обращения неявного факторизованного предобусловливателя связана с рекурсивным характером вычислений. Одним из способов ее преодоления является использование переупорядочений узлов сетки и соответствующих перестановок строк и столбцов матрицы. При этом часто используются упорядочения, связанные с разбиением области расчета (DDO – Domain Decomposition ordering) [4]. Применению такого подхода для крупнозернистого распараллеливания, когда область расчета разбивается на подобласти и расчеты в каждой подобласти производятся на своем процессоре,

посвящено много работ, например [3, 5-11]. Однако большая часть этих работ посвящена распараллеливанию вычислений при проведении расчетов задач с использованием ортогональных сеток.

В работе [12] предлагается использовать упорядочение типа DDO для построения параллельного варианта метода неполного треугольного разложения второго порядка сопряженных градиентов (IC2S(τ)-CG) [13]. Здесь и далее $0<\tau<<1$ – параметр отсечения. При этом производится отсечение по позициям для некоторых элементов матрицы предобусловливания.

В работах [14, 15, 16] предложен альтернативный подход, который позволяет преодолеть 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 [17] в сочетании с неполным треугольным разложением второго порядка IC2(τ) [13] – BIIC-IC2(τ) [14], IC2S(τ) [13] – BIIC-IC2S(τ) [15] и первого порядка IC1(τ) [16] – BIIC-IC1(τ) [16]. Для построения этих предобусловливателей специальным образом строятся блоки с налеганием, а внутри блоков используется приближенное треугольное разложение IC2(τ), IC2S(τ) или IC1(τ). Заметим, что методы сопряженных градиентов с предобусловливателями IC2(τ), IC2S(τ), BIIC-IC2(τ) и BIIC-IC2S(τ), в отличие от методов с предобусловливателями VIC, IC(0), IC1(τ), BIIC-IC1(τ), являются безотказными для любого фиксированного значения τ.

Проблеме использования высокоуровневого параллелизма (мелкозернистого или распараллеливания алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. В работах [18 – 21] было предложено использовать несколько итераций Якоби или блочного Якоби для решения треугольных систем при применении предобусловливания неполного треугольного разложения, что позволило использовать высокий уровень параллелизма. В работе [22] 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 (1.1)). В работе [23] предложен новый итерационный алгоритм вычисления неполного треугольного разложения Холецкого без заполнения IC(0) [2], а также IC(1), IC(2) (неполного треугольного разложения Холецкого с заполнением 1 и 2), в котором все ненулевые элементы треугольных матриц могут быть вычислены асинхронно.

В работах [24 – 26] предложены два безытерационных способа применения MPI+OpenMP технологии построения и обращения предобусловливателей блочного Якоби в сочетании с IC(0) и IC1(τ) – предобусловливателей BJIC(0) и BJIC1(τ). Один из них основан на переупорядочении узлов сетки типа DDO, другой – на уменьшении шаблона разреженности матрицы A при построении

предобусловливателя. В работах [16, 27] предложен безытерационный способ применения MPI+OpenMP технологии для построения и обращения предобусловливателей BIIC-IC2S(τ) и BIIC-IC1(τ) на основе использования числа блоков в предобусловливателе, кратного числу используемых процессоров и числу используемых потоков. В работах [27, 28] предложены безытерационные способы применения MPI+OpenMP технологии для построения и обращения предобусловливателя блочного неполного обратного треугольного разложения первого порядка BIIC-IC1(τ), в котором для мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO [12]. В работе [28] с помощью расчета ряда тестовых задач показано, что применение переупорядочения узлов сетки типа DDO для мелкозернистого распараллеливания при реализации построения и обращения предобусловливателя BIIC-IC1(τ) в ряде случаев может привести к меньшему времени счета задачи, чем применение подхода, предложенного в работе [16]. В работе [29] предложен способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя IC(0) в предобусловленном методе сопряженных градиентов. В этой работе для параллельной реализации метода IC(0)-CG с применением только MPI используется упорядочение узлов сетки типа DDO [12], для применения OpenMP технологии тоже используется упорядочение узлов сетки типа DDO [12]. При применении MPI+OpenMP технологии для построения и обращения предобусловливателя на основе использования упорядочения узлов сетки типа DDO в работах [24-29] OpenMP технологии применялись для большинства строк матрицы. В работах [16, 24-28] с помощью расчетов тестовых задач показано, что применение MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для умеренного числа узлов суперкомпьютерной системы (порядка нескольких десятков).

В формуле (1.1) предполагается, что матрица A переупорядочена, а вместо A_P стоит A ($A = A_P = P\tilde{A}P^T$), где P – матрица перестановок, а $\tilde{A}$ – матрица коэффициентов исходной задачи. В настоящей работе применяются переупорядочения, уменьшающие среднюю ширину ленты матрицы, а именно предложенные в работе [30]. Подход, предложенный в этой работе, позволяет одновременно произвести разбиение односвязной области расчета на подобласти. В настоящей работе использовалась отмасштабированная матрица A, т. е. диагональные элементы матрицы A были равны единице. Это достигалось 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 предлагаются способы применения MPI+OpenMP технологии построения и обращения предобусловливателя VIC при решении СЛАУ (1.1) с произвольной симметричной положительно определенной

матрицей предобусловленным методом сопряженных градиентов. Способ применения MPI при построении и обращении предобусловливателя основан на использовании переупорядочения строк и столбцов матрицы в соответствие с использованием упорядочения узлов сетки всей области расчета типа DDO [12]. При применении MPI+OpenMP технологии проводится дополнительное переупорядочение строк и столбцов матрицы в соответствие с дополнительным переупорядочением внутренних узлов сетки, полученных в результате переупорядочения для применения MPI. При этом тоже используется упорядочение узлов сетки типа DDO [12]. В предложенном в настоящей работе способе применения MPI+OpenMP технологии для построения и обращения предобусловливателя VIC OpenMP технологии применяются для большинства строк матрицы. В работе проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем VIC с использованием MPI и MPI+OpenMP подходов на примере двух модельных задач и ряда задач из коллекции разреженных матриц SuiteSparse [31].

2. Метод сопряженных градиентов с предобусловливателем VIC

Пусть требуется решить СЛАУ (1.1). Алгоритм предобусловленного метода сопряженных градиентов (см., например, [32]) имеет следующий вид:

Алгоритм 1

$$r_0 = b - Ax_0,\ p_0 = w_0 = B^{-1}r_0,\ \gamma_0 = r_0^T p_0,$$

для k=0,… пока $(r_k^T r_k) \le \varepsilon^2 (r_0^T r_0)$ выполнять

$$q_k = Ap_k, \qquad \alpha_k = \gamma_k / (p_k^T q_k),$$

$$x_{k+1} = x_k + \alpha_k p_k, \quad r_{k+1} = r_k - \alpha_k q_k, \quad z_{k+1} = B^{-1} r_{k+1},$$

$$\gamma_{k+1} = r_{k+1}^T z_{k+1}, \qquad \beta_k = \gamma_{k+1} / \gamma_k, \qquad p_{k+1} = z_{k+1} + \beta_k p_k,$$

где $0 < \varepsilon << 1$, B – матрица предобусловливания ($B \approx A$).

В методе VIC-CG предобусловливатель VIC определяется по формуле (1.3), где элементы диагональной матрицы D определяются из условия совпадения диагональных элементов матриц A и B и вычисляются по формуле (см, например, [3])

$$d_i^{-1} = a_{ii} - \sum_{l<i} a_{il}^2 d_l \ , \tag{2.1}$$

где a_{ii}, a_{il} – элементы матрицы A. Для уменьшения числа арифметических действий при обращении матрицы предобусловливания будем использовать представление матрицы предобусловливания в виде

$$B = U^T U \text{, где } U = D^{-0.5} + D^{0.5} A^+ . \tag{2.2}$$

Таким образом, на этапе построения матрицы предобусловливания для матрицы A сначала нужно вычислить элементы диагональной матрицы D, а затем вычислить элементы матрицы U, определенной в формуле (2.2).

Заметим, что при вычислении $d_i^{0.5}$ необходимо извлекать квадратный корень из числа, которое определяется в процессе вычисления и может оказаться отрицательным. В этом случае рекомендуется для решения СЛАУ (1.1) использовать метод сопряженных градиентов с другим предобусловливателем.

Алгоритм 1 использует операции умножения разреженных матриц на вектор, операции вычисления скалярных произведений, элементарные векторные операции, а также вычисление $z_{k+1} = B^{-1} r_{k+1}$, $p_0 = w_0 = B^{-1} r_0$. Принципиальная возможность эффективной параллельной реализации всех операций, кроме вычисления $B^{-1} r_{k+1}$, $B^{-1} r_0$, не вызывает сомнений, даже при использовании большого числа процессоров и (или) применения OpenMP технологии.

3. Алгоритм параллельной реализации построения и обращения предобусловливателя с использованием MPI

Разобьем произвольную (возможно, трехмерную) область расчета на p подобластей каким-либо образом, выберем некоторую нумерацию подобластей и будем использовать упорядочение узлов сетки, описанное в работе [12]. Введем множество узлов разделителей – множество узлов сетки в подобластях, у которых имеются соседи из подобластей с большим номером. Остальные узлы сетки будем называть внутренними. Узел разделителя назовем узлом разделителя первого уровня, если в шаблоне этого узла нет узлов разделителей из других подобластей с номерами большими, чем номер рассматриваемой подобласти. Узел разделителя назовем узлом разделителя второго уровня, если в шаблоне этого узла нет узлов разделителей более высокого, чем первый уровень, расположенных в подобластях с большим номером. Остальные узлы разделителей назовем узлами разделителей третьего уровня.

Установим следующий порядок следования узлов сетки. Сначала идут все внутренние узлы подобластей в порядке следования номеров подобластей, причем сохраняется порядок следования узлов внутри каждой подобласти, введенный ранее. Затем идут узлы разделителей первого уровня в порядке следования номеров подобластей с сохранением порядка следования узлов внутри каждой подобласти, введенного ранее. Далее следуют узлы разделителей второго уровня в порядке следования номеров подобластей с сохранением ранее введенного порядка следования узлов внутри подобластей. И, наконец, идут узлы разделителей третьего уровня в порядке следования

номеров подобластей и с сохранением ранее введенного порядка следования узлов внутри каждой подобласти. Заметим, что это упорядочение является обобщением упорядочения DDO1 из работы [4].

В работе [29] приведен пример такого упорядочения узлов сетки для случая решения задачи Дирихле для уравнения Пуассона в квадратной области расчета области расчета при использовании для аппроксимации пятиточечного шаблона и разбиении области расчета на 4 подобласти.

На рис. 1 приведен вид структуры разреженности матрицы *A*, полученной после перестановки строк и столбцов в результате переупорядочения в случае разбиения области расчета на 4 подобласти. Использованы обозначения: *l* – число всех внутренних узлов сетки из всех подобластей, *l*1 – число всех внутренних узлов сетки из всех подобластей и всех узлов разделителей первого уровня из всех подобластей, *l*2 – число всех внутренних узлов сетки из всех подобластей и всех узлов разделителей первого и второго уровня из всех

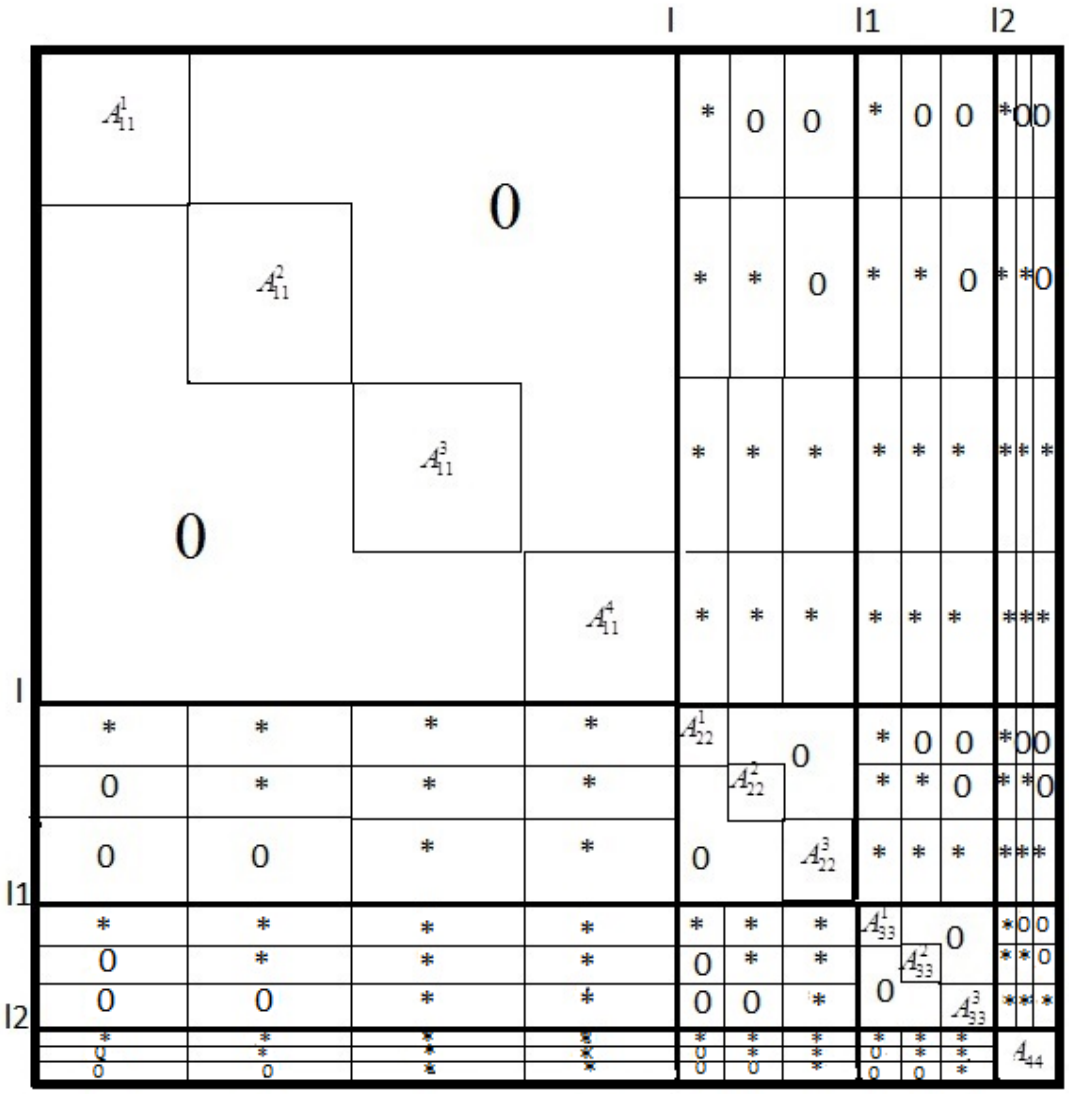

Рис. 1. Вид структуры разреженности матрицы *A*, полученной после перестановки строк и столбцов в результате переупорядочения.

подобластей. Строки матрицы, содержащие блочно-диагональные части A_{11}^s, соответствуют внутренним узлам сетки и хранятся в процессоре с номером *s*. Строки матрицы, содержащие блочно-диагональные части A_{22}^s и A_{33}^s, соответствуют узлам сетки соответственно на разделителях первого и второго уровня из подобласти с номером *s* и хранятся в процессоре с номером *s*. Строки матрицы, соответствующие блочно-диагональной части A_{44}, соответствуют узлам сетки на разделителях третьего уровня и хранятся в соответствующих процессорах.

Перед вычислением элементов диагональной матрицы *D* с использованием формулы (2.1) и матрицы *U* с использованием формулы (2.2) необходимо построить нижнетреугольную и верхнетреугольную части, переупорядоченные в соответствии с новым упорядочением узлов сетки матрицы *A*.

Определение элементов диагональной матрицы *D* с использованием формулы (2.1) начинается с вычисления ее элементов в строках, соответствующих внутренним узлах всех подобластей. При этом используются значения элементов строго нижнетреугольной части A^- переупорядоченной матрицы *A*. Вычисление элементов матрицы *D* в строках, соответствующих внутренним узлам подобластей, происходит во всех процессорах одновременно и независимо. Затем осуществляется пересылка некоторых значений d_i, вычисленных в процессоре с номером *s*, в процессоры с номерами, меньшими *s*, если эти значения необходимы в этих процессорах для вычисления элементов матрицы *D* в строках, соответствующих узлам разделителей (*s*=1, 2, … *p*).

Далее с использованием формулы (2.1) вычисляются элементы матрицы *D* в строках, соответствующих узлам разделителей первого уровня. При этом все процессоры работают одновременно и независимо. Затем осуществляется пересылка некоторых значений d_i для номеров *i*, соответствующих узлам разделителей первого уровня, из процессора с номером *s* в процессоры с номерами, меньшими *s*, если эти значения необходимы в этих процессорах для вычисления элементов матрицы *D* в строках, соответствующих узлам разделителей второго и третьего уровней (*s*=1, 2, … *p*).

Далее с использованием формулы (2.1) вычисляются элементы матрицы *D* в строках, соответствующих узлам разделителей второго уровня. Затем осуществляется пересылка некоторых значений d_i, вычисленных в процессоре с номером *s*, для номеров *i*, соответствующих узлам разделителей второго уровня, в процессоры с номерами, меньшими *s* , если эти значения необходимы в них для вычисления элементов матрицы *D* в строках, соответствующих узлам разделителей 3 уровня (*s*=1,2,… *p*).

Далее с использованием формулы (2.1) вычисляются элементы матрицы D в строках, соответствующих узлам разделителей третьего уровня. Для

пересылок значений d_i на всех этапах используются операции MPI_Recv и MPI_ Send.

На следующем этапе вычисления матрицы предобусловливания осуществляется вычисление элементов верхнетреугольной матрицы U с использованием формулы (2.2). При этом все процессоры работают одновременно и независимо, пересылок не требуется.

Перед обращением матрицы предобусловливания необходимо произвести переупорядочение элементов вектора r^{k-1}, где k – номер итерации в предобусловленном методе сопряженных градиентов. Обращение матрицы предобусловливания состоит из двух этапов: $\hat{w}^k = U^{-T} r^{k-1}$ и $w^k = U^{-1}\hat{w}^k$. На первом этапе будем использовать алгоритм обращения транспонированной матрицы, предложенный в работе [15]. Так же, как в работах [15, 29], будем вычислять элементы вектора $z = D_U \hat{w}^k$, где D_U – диагональная часть матрицы U, а затем по элементам вектора z вычислять элементы вектора $\hat{w}^k$.

Сначала производится вычисление значений $\hat{w}_i^k$ во внутренних узлах подобластей. Алгоритм параллельной реализации вычисления элементов векторов z и $\hat{w}^k$, соответствующих внутренним узлам сетки подобластей, а также слагаемых элементов вектора z, соответствующих узлам разделителей, имеет следующий вид [29]:

Алгоритм 2.

1. Вычислить первоначальные значения z_j:

for $j=1,n$

$z_j = r_j^{k-1}$

end for

for $s=1,2,\ldots,p$

2. Вычислить z_j для $j=i_1(s), \ldots, i_2(s)$ и слагаемые элементов z_j для $j>l$:

for $i = i_1(s),...,i_2(s)$

$save = z_i / u_{ii}$

for $j=i+1,n$

$z_j = z_j - u_{ij} \times save$

end for

end for

2. Вычислить значение $\hat{w}_i^k$ для $i = i_1(s),...,i_2(s)$:

for $i=$ $i = i_1(s),...,i_2(s)$

$\hat{w}_i^k = z_i / u_{ii}$

end for

end for (*конец цикла по s*).

В алгоритме 2 i и j – глобальные индексы, $i_1(s)$, $i_2(s)$ – номера первой и последней строк, соответствующих внутренним узлам в подобласти с номером s. При вычислении с использованием алгоритма 2 все процессоры работают одновременно и независимо.

После этого осуществляется пересылка значений слагаемых z_j для j, соответствующих узлам сетки из соседних подобластей, расположенных на разделителях, в процессоры, в которых эти значения нужны для дальнейшего расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей первого уровня с использованием алгоритма, аналогичного алгоритму 5, что осуществляется во всех процессорах одновременно и независимо. Затем производится пересылка значений слагаемых z_j для j, соответствующих узлам сетки из соседних подобластей, в процессоры, в которых эти значения нужны для дальнейшего расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей второго уровня с использованием алгоритма, аналогичного алгоритму 5, и осуществляется пересылка значений слагаемых z_j для j, соответствующих узлам сетки из соседних подобластей, в процессоры, в которых эти значения нужны для дальнейшего расчета, а также сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах.

Далее происходит вычисление значений $\hat{w}_i^k$ в узлах разделителей третьего уровня. Если A_{44} имеет блочно-диагональную структуру с нулевыми элементами вне блоков, то вычисления на этом этапе происходят во всех процессорах одновременно и независимо (подробнее см. [12, 29]).

Для пересылок слагаемых z_j на всех этапах используются операции MPI_Recv и MPI_ Send.

На этапе 2 обращения матрицы предобусловливания сначала с использованием матрицы U происходит вычисление значений w_i^k в узлах разделителей третьего уровня, затем второго, первого уровня, а далее во внутренних узлах подобластей. При вычислении w_i^k в узлах разделителей второго и первого уровня и во внутренних узлах все процессоры работают

одновременно. Если A_{44} имеет блочно-диагональную структуру с нулевыми элементами вне блоков, то вычисления w_i^k в узлах разделителей третьего уровня происходят во всех процессорах одновременно и независимо. После расчета на разделителях каждого уровня происходит пересылка найденных значений w_i^k этого уровня в процессоры, в которых эти значения нужны для дальнейшего расчета. Для пересылок используются операции MPI_Recv и MPI_ Send.

После завершения вычисления вектора w^k следует произвести переупорядочение его элементов.

4. Алгоритм параллельной реализации построения и обращения предобусловливателя с использованием MPI+OpenMP

Рассмотрим способ применения MPI+OpenMP технологии при построении и обращении предобусловливателя VIC. Сначала произведем разбиение всей области расчета на p подобластей и переупорядочение всех узлов сетки типа DDO, как описано в предыдущем разделе. Затем в каждой получившейся подобласти с номером s разобьём внутреннюю часть подобласти, состоящую из внутренних узлов, полученных при переупорядочении, произведенном для MPI реализации, на m подобластей, где m – предполагаемое число используемых потоков, с приблизительно равным числом узлов сетки. В настоящей работе разбиение осуществлялось в порядке следования внутренних узлов подобластей, установленном ранее, следующим образом. Пусть $\hat{l}_s = [n_s / m]$, первые m-1 «внутренних» подобластей содержат $\hat{l}_s$ внутренних узлов, последняя «внутренняя» подобласть содержит $n_s - (m-1)\hat{l}_s$ внутренних узлов. Здесь и далее кавычки перед и после слова внутренняя означают, что речь идет о разбиении для мелкозернистого распараллеливания или распараллеливания на потоки.

Заметим, что можно производить разбиение всей области расчета сразу на pm подобластей каким-либо способом и обеспечить, чтобы каждый процессор производил вычисления в соответствующих m подряд идущих подобластях, как, например, в работе [16].

Произведем переупорядочение внутренних узлов сетки в подобластях, полученных при разбиении всей области расчета на p подобластей, где p – число процессоров ($p \neq 1$). Для переупорядочения внутренних узлов в подобластях будем использовать упорядочение типа DDO [12], которое было подробно описано в предыдущем разделе и используется для переупорядочения узлов сетки всей области расчета для крупнозернистого распараллеливания. В результате будут определены «внутренние» узлы и узлы «разделителей». Здесь

и далее кавычки перед и после слов «внутренние» и «разделители» означает, что речь идет о внутренних узлах и узлах разделителей при упорядочении для мелкозернистого распараллеливания. Множество узлов «разделителей» разобьем на множество узлов «разделителей» первого, второго и третьего уровней аналогично тому, как описано в предыдущем разделе при построении переупорядочения для крупнозернистого распараллеливания.

Определим $\bar{l}_s$ – минимальное число «внутренних» узлов при разбиении множества внутренних узлов (необязательно таком, как описано выше) из 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 подобласти с номером s (s= 1, …, 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-й «внутренней» подобласти. Затем идут оставшиеся «внутренние» узлы «внутренних» подобластей в порядке следования номеров «внутренних» подобластей и в порядке следования узлов внутри «внутренних» подобластей, введенном ранее. Затем идут узлы «разделителей» первого уровня, полученные при упорядочении множества внутренних узлов подобласти с номером s, затем узлы «разделителей» второго уровня при этом упорядочении, а затем узлы «разделителей» третьего уровня. При этом для каждого уровня «разделителей» узлы следуют с сохранением порядка следования «внутренних» подобластей и порядка следования узлов внутри подобласти, введенного ранее.

Затем идут узлы разделителей первого, второго и третьего уровней при упорядочении для крупнозернистого распараллеливания. При этом сохраняется порядок следования узлов на разделителях, введенный для крупнозернистого распараллеливания.

Таким образом, при использовании MPI+OpenMP технологии производится дополнительное переупорядочение только узлов сетки, являющихся внутренними при упорядочении для использования только MPI.

При использовании MPI+OpenMP технологии для вычисления диагональных элементов d_i диагональной матрицы D по формуле (2.1) применение OpenMP технологии в каждом процессоре с номером s (s=1, …, p) осуществляется при вычислении первых $M1$ диагональных элементов части матрицы D, вычисляемой в этом процессоре. При этом используется цикл по $k2$=1, …, m, внутри которого осуществляется вычисление элементов d_i с локальными номерами 1, …, $M1$ (при новом упорядочении). 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 остальных диагональных элементов части диагональной матрицы D, вычисляемой в соответствующем процессоре, с использованием только MPI,

как описано выше. Если число узлов во всех «внутренних» подобластях достаточно велико, то для подавляющего большинства элементов диагональной матрицы D их вычисление происходит с использованием MPI+OpenMP технологии.

Применение OpenMP технологии при построении матриц U_S (частей матрицы U, вычисляемых и используемых в процессорах с номером s (s=1, 2, …, p) по формуле, аналогичной (2.2), осуществляется для всех строк этих матриц. При этом в настоящей работе использовалась директива **do** с опцией **schedule static**.

Обращение предобусловливателя VIC с использованием MPI+OpenMP технологии происходит аналогично обращению предобусловливателя IC(0) с использованием MPI+OpenMP технологии [29]. Перед началом итерационного процесса в каждом процессоре с номером s строится матрица $L1_S$ размера $il0(s)\times il0(s)$ транспонированная к матрице $U1_S$, где матрица $U1_S$ содержит первые $il0(s)$ строк и первые $il0(s)$ столбцов матрицы U_S. Здесь $il0(s)$ – количество внутренних узлов в подобласти с номером s. Создаются также матрицы $U2_S$, элементы которых совпадают с элементами из первых $il0(s)$ строк матрицы U_S с номерами столбцов с индексами, соответствующими узлам разделителей при упорядочении для крупнозернистого распараллеливания.

При обращении предобусловливателя, состоящем из двух этапов, указанных выше, перед началом вычислений в каждом процессоре с номером s производится переупорядочение элементов вектора r_s^{k-1} – части вектора r^{k-1}, хранящейся в процессоре с номером s. Первый этап обращения матрицы предобусловливания ($\hat{w}^k = U^{-T} r^{k-1}$) начинается с обращения нижнетреугольных матриц $L1_S$. В каждом процессоре с применением OpenMP технологии вычисляются $M1$ элементов $\hat{w}_i^k$ с локальными индексами i=1, … $M1 = m\bar{l}_s$ (при новом упорядочении), в настоящей работе использовалась директива **do** с опцией **schedule static**. Далее без применения OpenMP технологии производится вычисление элементов $\hat{w}_i^k$ с локальными номерами i=$M1$+1, …, $il0(s)$. Затем с использованием матрицы $U2_S$ вычисляются слагаемые элементов $z_j = U_{jj}\hat{w}_j^k$ вектора z для значений j, соответствующих узлам разделителей. При этом используется алгоритм, аналогичный алгоритму 5.

После этого осуществляется пересылка значений слагаемых z_j для индексов j, соответствующих узлам сетки из соседних подобластей на разделителях, в процессоры, в которых эти значения нужны для дальнейшего

расчета, и сложение в соответствующем процессоре слагаемых z_j, посчитанных в разных процессорах. Для пересылок используются операции MPI_Recv и MPI_ Send.

Затем происходит вычисление значений $z_j = U_{jj}\hat{w}_j^k$ и для индексов j, соответствующих узлам разделителей всех уровней при упорядочении, введенном для крупнозернистого распараллеливания, и вычисление $\hat{w}_i^k$ в узлах разделителей всех уровней. Это осуществляется так же, как в случае применения только MPI, и описано в предыдущем разделе.

На этапе обращения верхнетреугольных матриц вычисления происходят в обратном порядке с использованием матриц U_S (s=1, 2, …, p). Сначала происходит вычисление значений w_i^k в узлах разделителей третьего уровня, затем второго и первого уровня, что осуществляется так же, как при использовании только MPI. После расчета на разделителях каждого уровня происходит пересылка найденных значений w_i^k этого уровня в процессоры, в которых эти значения нужны для дальнейшего расчета. Для пересылок используются операции MPI_Recv и MPI_ Send. Потом происходит вычисление w_i^k во внутренних узлах сетки (полученных при упорядочении для MPI распараллеливания). При этом элементы искомого вектора с локальными номерами i=M1, …,1 (при новом упорядочении) вычисляются с использованием OpenMP технологии. В настоящей работе при этом использовалась директива **do** с опцией **schedule static**. После вычисления вектора w^k при новом упорядочении следует вернуться к первоначальному упорядочению для элементов этого вектора.

Вычисление элементов вектора $q_k = Ap_k$, где k – номер итерации в алгоритме 1 предобусловленного метода сопряженных градиентов, если матрица A хранится в памяти в распределенном CRS-формате, содержит верхний и нижний треугольники, с использованием MPI+OpenMP хорошо изучено, подробно описано, например, в работе [26]. MPI+OpenMP реализация вычислений векторных операций и скалярных произведений в алгоритме 1 тоже хорошо известна. При применении MPI+OpenMP технологии для умножения матрицы на вектор и для вычислений 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 VIC-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI+OpenMP технологии. Расчеты проводились на многопроцессорном вычислительном кластере К60, установленном в ЦКП ИПМ им. М.В. Келдыша РАН. Исследование эффективности использования MPI и MPI+OpenMP технологии производилось с помощью решения двух модельных задач: разностных задач Дирихле для уравнения Пуассона в единичном квадрате на равномерной ортогональной сетке с использованием стандартной 5-точечной аппроксимации лапласиана. В модельной задаче 1 n=1048576 (имя матрицы **5_1048576**), в модельной задаче 2 n=262144 (имя матрицы **5_262144**). Для тестирования эффективности применения MPI и MPI+OpenMP технологии при решении СЛАУ (1.1) методом VIC-CG использовались также матрицы из коллекции разреженных матриц SuiteSparse [31]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:

apache2 – трехмерная конечно-разностная схема;
parabolic_fem – уравнение диффузии-конвекции с постоянным переносом;
ecology2 – приложение теории электрических цепей к задаче передачи генов;
apache1 – трехмерная конечно-разностная схема;
g2_circuit – моделирование интегральных схем.

В таблице 1 приведены некоторые свойства этих матриц, причем значения $Cond(A_0)$, где $A_0 = (D_A)^{-1/2} A (D_A)^{-1/2}$ – матрица системы уравнений после масштабирования, взяты из работы [33], 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 (1.1).

Таблица 1

Свойства некоторых матриц из коллекции разреженных матриц SuiteSparse

Матрица	N	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_S)$
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
apache1	80800	542184	0	0	4	7	0.10+7
g2_circuit	150102	726674	0	28000	2	6	0.23+6

Решение СЛАУ (1.1) с матрицами относительно небольшого размера с именами **5_262144**, **apache1**, **g2_circuit** производилось для сравнения необходимого числа итераций и времени счета этих задач методами VIC-CG и IC(0)-CG (см. работу [29]).

Решалось уравнение $Ax = b$, где A – матрица, полученная после переупорядочения и масштабирования исходной матрицы, с единичной правой частью ($b_i \equiv 1$), начальное приближение $x_0 = 0$. Счет продолжался до выполнения условия (1.2), где $\varepsilon = 10^{-8}$. Для разбиения области расчета на p подобластей использовался алгоритм [30].

В таблицах 2 – 8 приведены числа итераций и время счета методом VIC-CG тестовых задач при использовании MPI и MPI+OpenMP технологии. Под временем вычислений в таблицах 2 – 8 подразумевается время счета в секундах итерационного процесса в сумме с временем вычисления предобусловливателя. При этом время вычисления предобусловливателя состоит из времени создания переупорядоченных верхнетреугольной и нижнетреугольной частей матрицы *A*, вычисления диагональной матрицы *D*, вычисления верхнетреугольной матрицы *U*. При использовании MPI+OpenMP технологии время вычисления предобусловливателя дополняется временем вычислением матриц $L1_S$.

При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 10, 12 и 16 нитей. В таблицах 2 – 8 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей, обозначенные *Th*. В таблицах 2 – 8 приведены курсивом коэффициенты μ ускорения счета задач благодаря использованию OpenMP технологии, а также коэффициенты η ускорения счета по сравнению со счетом с использованием только MPI на минимальном числе процессоров, указанном в таблицах (на двух или трех процессорах). В таблицах 2 – 8 приведены также значения линейного ускорения для соответствующего числа процессоров, обозначенные ηl.

Таблица 2

Числа итераций и время счета задачи с матрицей **5_1048576** на *p* процессорах без применения и с применением OpenMP технологии

P ηl	3	4 *1.33*	6 *2*	8 *2.66*	10 *3.33*	16 *5.33*
MPI η	1125,11.67	1022, 7.72 *1.51*	1119, 5.87 *1.99*	1034,4.35 *2.68*	1031,3.51 3.32	1122,3.34 *3.49*
MPI+ OpenMP μ, η	*Th*=8 1186, 5.60 *2.08*	*Th*=6 1186, 4.09 *1.88, 2.85*	*Th*=4 1173, 3.65 *1.61, 3.20*	*Th*=3 1164,3.64 *1.19, 3.21*	*Th*=4 1113,3.36 *1.04, 3.47*	Th=3 1202,3.98 *0.84, 2.93*

Таблица 3

Числа итераций и время счета задачи с матрицей **parabolic_fem**
на *p* процессорах без применения и с применением OpenMP технологии

P η *l*	2	3 *1.5*	4 *2*	6 *3*	8 *4*	10 *5*	16 *8*
MPI η	897,8.25	931,7.48 *1.1*	949,4.6 *1.79*	933,3.24 *2.54*	936,2.5 *3.3*	946,2.24 *3.68*	963,1.54 *5.35*
MPI+ OpenMP μ, η	*Th*=12 999,2.49 *3.31*	*Th*=10 1070,3.01 *2.48, 2.74*	*Th*=6 1050,2.04 *2.25, 4.04*	*Th*=4 1005,1.85 *1.75, 4.45*	*Th*=3 1005,1.74 *1.43, 4.74*	*Th*=4 1006,1.75 *1.28, 4.71*	*Th*=3 1011,1.62 *0.95, 5.09*

Таблица 4

Числа итераций и время счета задачи с матрицей **apache2**
на *p* процессорах без применения и с применением OpenMP технологии

P η *l*	2	3 *1.5*	4 *2*	6 *3*	8 *4*	10 *5*	16 *8*
MPI η	953,11.10	1297,9.88 *1.12*	873,7.4 *1.5*	937,4.07 *2.73*	948,2.97 *3.74*	1294,3.46 *3.21*	956,1.96 *5.66*
MPI+ OpenMP μ, η	*Th*=10 1135,4.73 *2.34*	*Th*=8 1373,4.84 *2.04, 2.29*	*Th*=6 1016,3.25 *2.27, 3.41*	*Th*=4 1039,2.61 *1.56, 4.25*	*Th*=3 1019,2.39 *1.24, 4.64*	*Th*=4 1367,3.32 *1.04, 3.34*	*Th*=3 1069,2.52 *0.78,4.4*

Таблица 5

Числа итераций и времена счета задачи с матрицей **ecology2**
на *p* процессорах без применения и с применением OpenMP технологии

P η *l*	3	4 1.75	6 2	8 2.66	10 3.33	16 5.33
MPI η	2039,20.58	2080,13.98 *1.47*	2022,11.06 *1.86*	2081,7.55 *2.73*	2055,6.54 *3.14*	2092,5.31 *3.88*
MPI+ OpenMP μ, η	*Th*=8 2126,9.39 *2.19*	*Th*=6 2136,6.66 *2.1, 3.09*	*Th*=4 2116, 6.78 *1.63, 3.03*	*Th*=3 2156,5.82 *1.3, 3.53*	*Th*=4 2141,6.1 *1.07, 3.37*	*Th*=3 2122,6.04 *0.88, 3.41*

Заметим, что неожиданно большое число итераций при решении задач с матрицами **apache2** (на трех и на 10 процессорах) и **apache1** (на двух процессорах) связано с особенностями разбиения области расчета на подобласти методом [30].

Как видно из таблиц 2-5, эффективность распараллеливания при решении соответствующих задач с использованием только MPI, вычисляемая по формуле $\eta / \eta l \times 100\%$, составляла более 64% при $p \le 16$.

В таблицах 6 – 8 в скобках приведено время счета задач с матрицами **5_262144**, **apache1**, **g2_circuit** без учета времени создания переупорядоченных верхнетреугольной и нижнетреугольной частей матрицы A и вычисления матриц $L1_S$. Это сделано для сравнения времени счета этих задач с временем их счета при использовании предобусловливателя IC(0), приведенного в работе [29]. В работе [29] в качестве времени счета предобусловливателя используется время вычисления его верхнетреугольного множителя без учета времени построения верхнегреугольной части переупорядоченной матрицы *A*.

Таблица 6

Числа итераций и времена счета задачи с матрицей **5_262144** на *p* процессорах без применения и с применением OpenMP технологии

P ηl	2	3 *1.5*	5 2.5	8 *4*
MPI η	456,1.20(1.17)	516,1.02(1.00) *1.18*	569,0.71(0.68) *1.69*	578,0.49(0.44) *2.45*
MPI+OpenMP μ, η	*Th*=12 574,0.61(0.52) *1.97*	*Th*=8 612,0.63(0.56) *1.62, 1.9*	*Th*=4 593,0.51(0.44) *1.39, 2.35*	*Th*=3 617,0.42(0.33) *1.17, 2.86*

Таблица 7

Числа итераций и времена счета задачи с матрицей **apache1** на *p* процессорах без применения и с применением OpenMP технологии

P ηl	2	3 *1.5*	5 *2.5*	8 4
MPI η	1395,1.34(1.32)	455,0.37(0.35) 3.62	534,0.27(0.25) *4.96*	639, 0.21(0.19) *6.38*
MPI+ OpenMP μ, η	Th=10, 1448,0.66(0.62) *2.03*	Th=4 706, 0.34(0.30) *1.08, 3.94*	Th=3 756,0.33(0.28) *0.81, 4.06*	Th=3 886,0.27(0.23) *0.77, 4.96*

Таблица 8

Числа итераций и времена счета задачи с матрицей **g2_circuit** на *p* процессорах без применения и с применением OpenMP технологии

P ηl	2	3 *1.5*	5 2.5	8 *4*
MPI η	747,1.2(1.18)	756,0.97(0.95) *1.24*	757,0.64(0.62) *1.87*	758,0.46(0.43) 2.61
MPI+ OpenMP μ, η	Th=8 829,0.71(0.64) *1.69*	Th=4 817,0.61(0.56) *1.59, 1.96*	Th=3 825,0.51(0.46) *1.25, 2.35*	Th=3 821, 0.40(0.34) *1.15, 3.0*

Как видно из таблиц 6-8 и таблиц 2, 4, 5 из работы [29], при одинаковом числе процессоров и одинаковом числе нитей числа итераций, полученные в расчетах, как правило, совпадают с числами итераций при решении задач методом IC(0)-CG. Время счета, приведенное в скобках, не сильно отличается от времени счета, приведенного работе [29]. Однако алгоритм построения предобусловливателя в методе из настоящей работы значительно проще, чем алгоритм построения предобусловливателя IC(0), приведенный в работе [29]. Кроме того, метод, рассмотренный в настоящей работе, при написании программы, использованной в настоящей работе, потребовал значительно меньше оперативной памяти, чем метод IC(0)-CG при реализации, использованной в работе [29].

Как видно из таблиц 6 – 8, эффективность распараллеливания при решении соответствующих задач с использованием только MPI, вычисляемая по формуле $\eta / \eta l \times 100\%$, составляла более 61% при $p \le 8$.

На рисунках 2 – 5 приведены графики зависимости времени счета задач с матрицами **5_1048576**, **parabolic_fem**, **apache2**, **ecology2** методом VIC-CG от числа процессоров *p* в логарифмическом масштабе с использованием только MPI при разбиении на *p* подобластей с помощью алгоритма [30] (сплошные линии), а также с использованием MPI+OpenMP технологии (штриховые линии).

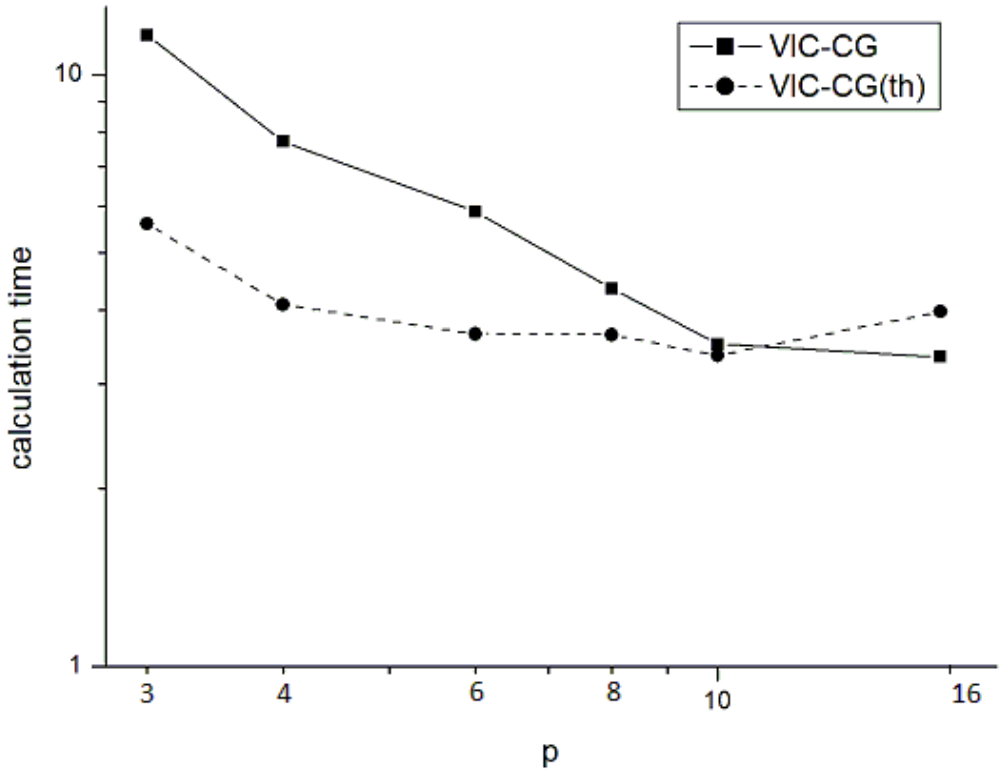

Рис. 2. Время счета задачи с матрицей **5_1048576** методом VIC-CG с использованием MPI и MPI+OpenMP технологии.

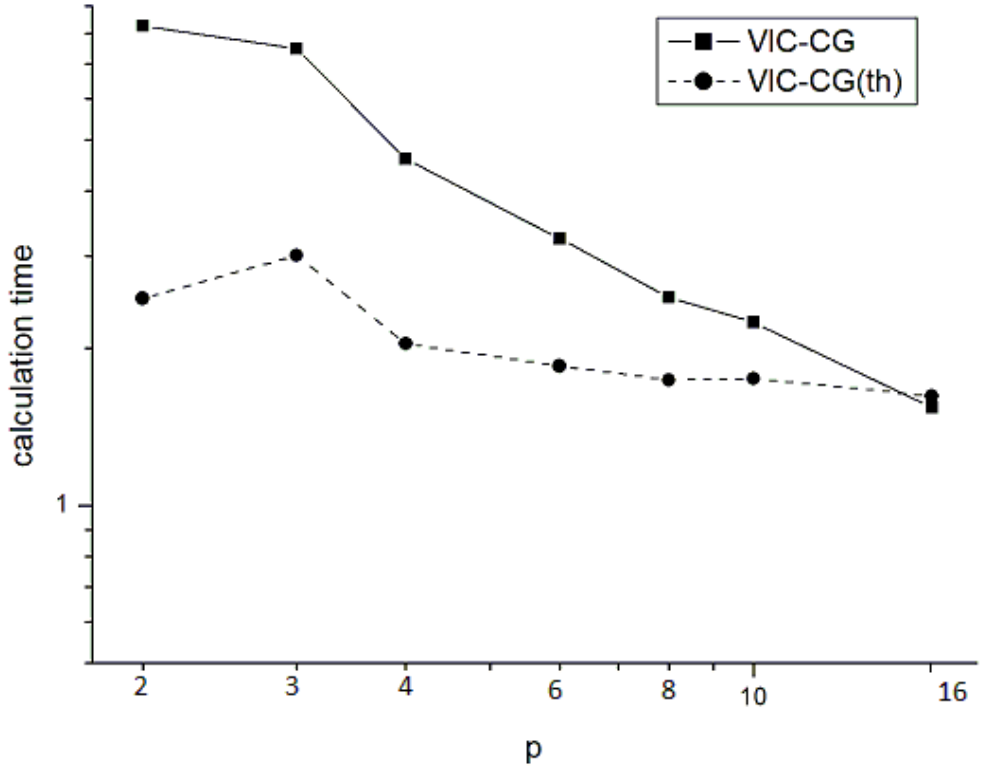

Рис. 3. Время счета задачи с матрицей **parabolic_fem** методом VIC-CG с использованием MPI и MPI+OpenMP технологии.

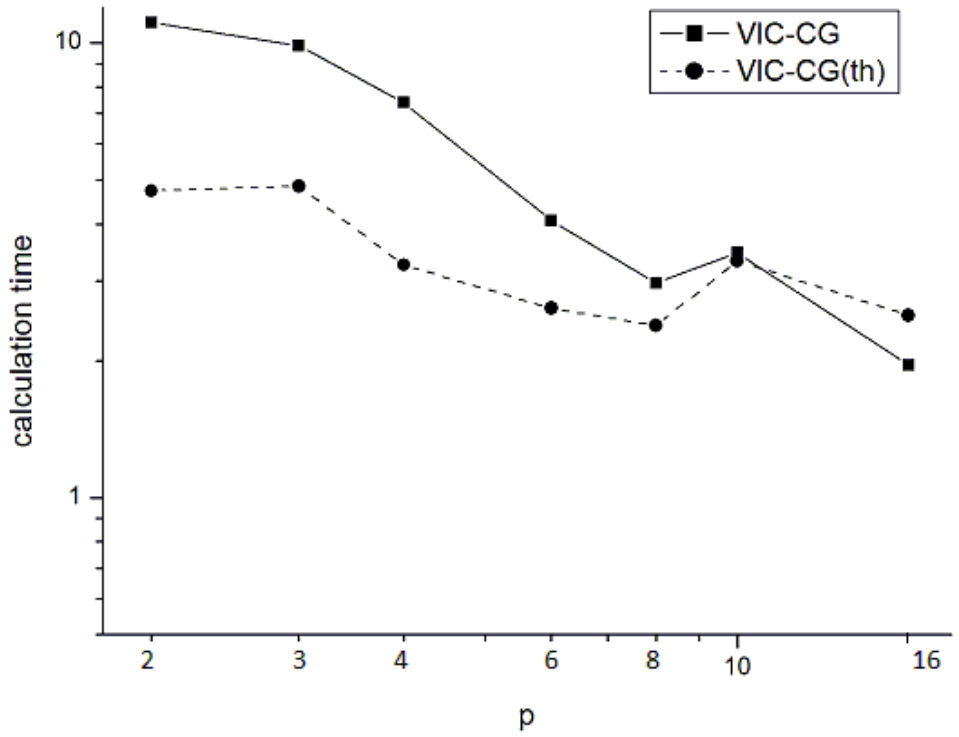

Рис. 4. Время счета задачи с матрицей **apache2** методом VIC-CG с использованием MPI и MPI+OpenMP технологии.

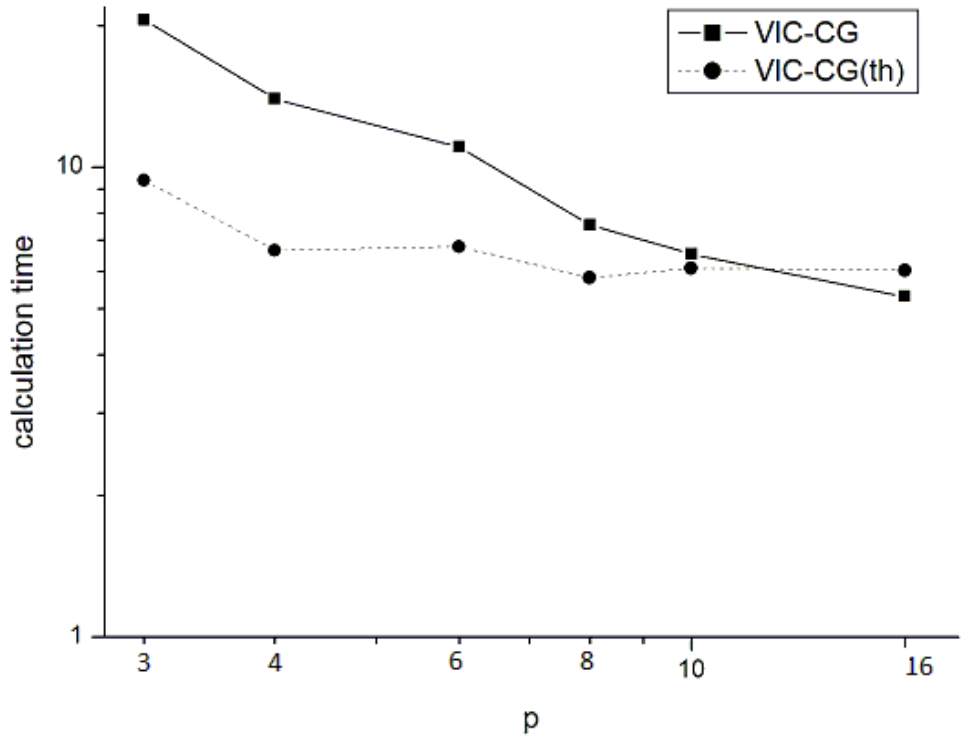

Рис. 5. Время счета задачи с матрицей **ecology2** методом VIC-CG с использованием MPI и MPI+OpenMP технологии.

Как видно из таблиц 2, 4, 5 и рис. 2, 4, 5, применение MPI+OpenMP технологии для решения задач с матрицами **5_1048576**, **apache2**, **ecology2** позволяет заметно ускорить решение этих задач, по сравнению с использованием только MPI при p<10. Как видно из таблицы 3 и рис. 3, применение MPI+OpenMP технологии для решения задачи с матрицей **parabolic_fem** позволяет заметно ускорить решение по сравнению с использованием только MPI при значениях $p<16$. Как видно из таблиц 2, 5, применение MPI+OpenMP технологии позволяет производить решение задач с матрицами **5_1048576**, **ecology2** с сверхлинейным ускорением при $p \le 10$, а задач с матрицами **parabolic_fem, apache2** – при p<10.

На рисунках 6 – 8 приведены графики зависимости времени счета задач с матрицами **5_262144**, **apache1**, **g2_circuit** методами VIC-CG и IC(0)-CG от числа процессоров p в логарифмическом масштабе с использованием только MPI и разбиения на p подобластей с помощью алгоритма [30] (сплошные линии), а также с использованием MPI+OpenMP технологии (штриховые линии). При этом под временем счета этих задач методом VIC-CG подразумевается время счета без учета времени вычисления переупорядоченных верхнетреугольной и нижнетреугольной частей матрицы A и определения элементов матрицы $L1_S$, приведенное в таблицах 6-8 в скобках.

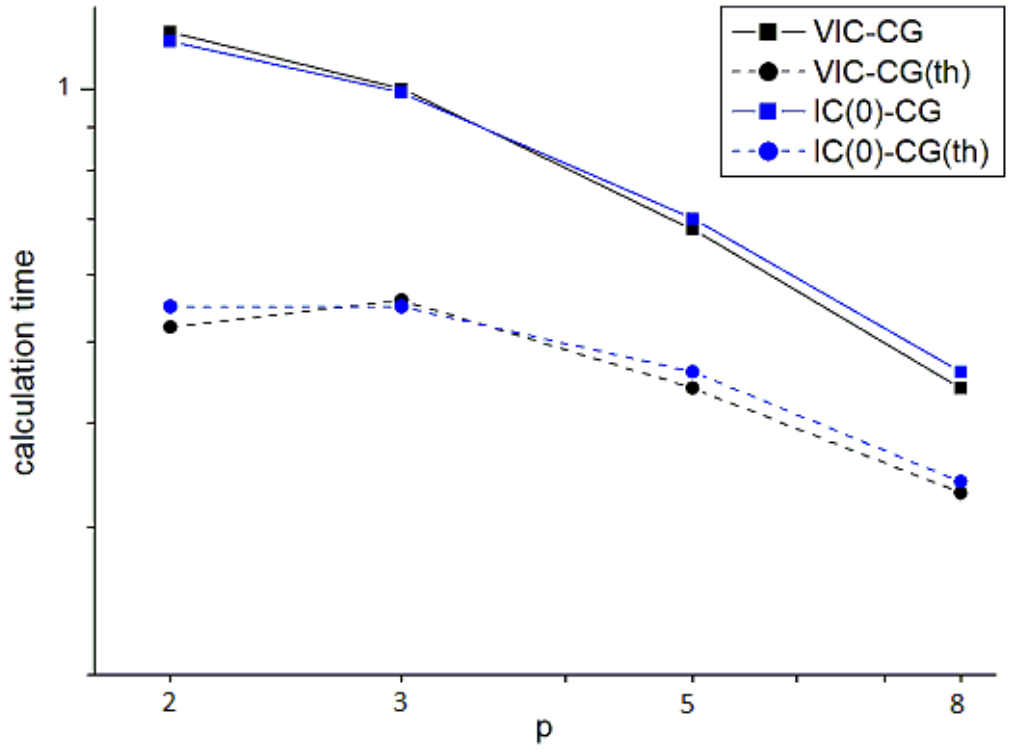

Рис. 6. Время счета задачи с матрицей **5_262144** методами VIC-CG и IC(0)-CG с использованием MPI и MPI+OpenMP технологии.

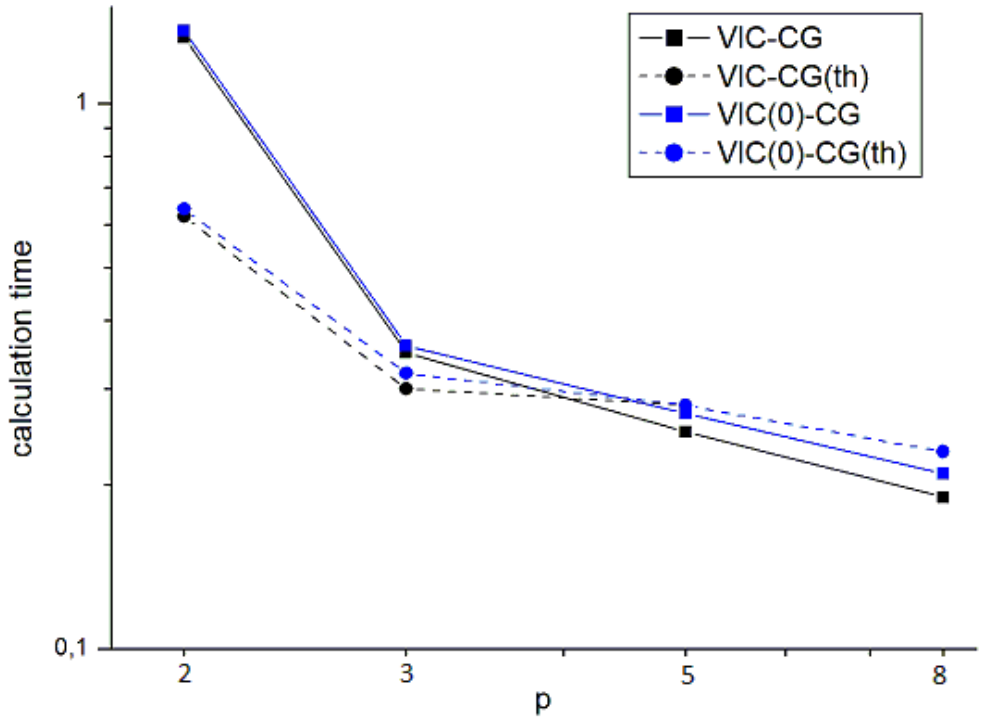

Рис. 7. Время счета задачи с матрицей **apache1** методами VIC-CG и IC(0)-CG IC(0)-CG с использованием MPI и MPI+OpenMP технологии.

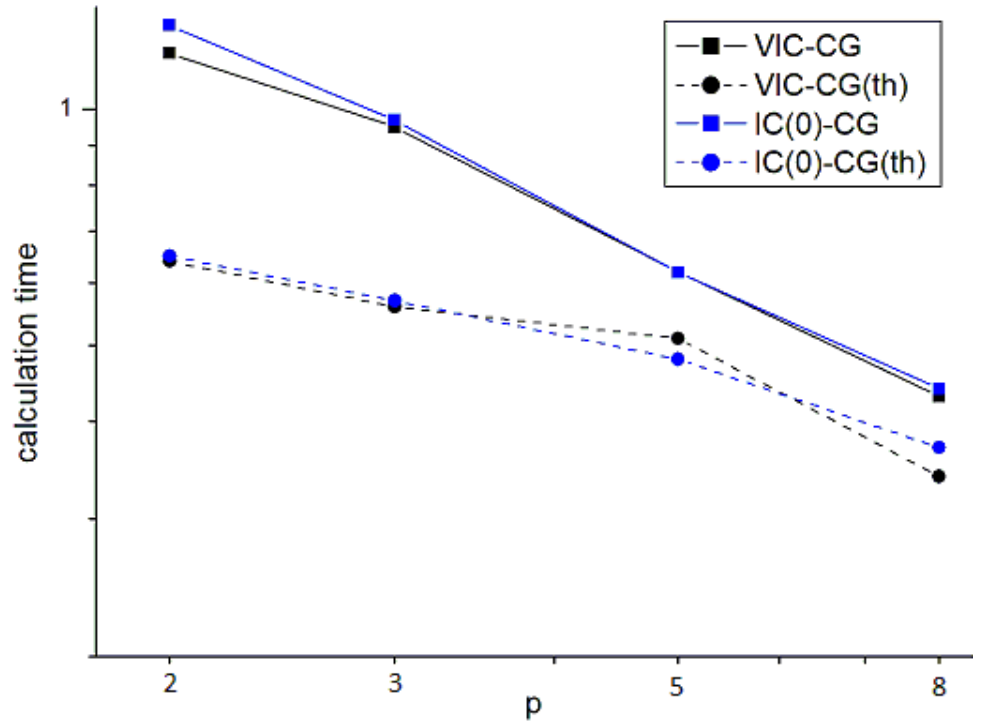

Рис. 8. Время счета задачи с матрицей **g2_circuit** методами VIC-CG и IC(0)-CG с использованием MPI и MPI+OpenMP технологии.

Как видно из таблиц 6, 8 и рис. 6, 8, применение MPI+OpenMP технологии для решения задач с матрицами **5_262144**, **g2_circuit** позволяет заметно ускорить решение этих задач по сравнению с использованием только MPI при всех использованных значениях p. Как видно из таблицы 7 и рис. 7, применение MPI+OpenMP технологии для решения задачи с матрицей **apache1** позволяет заметно ускорить решение задачи по сравнению с использованием только MPI только при p=2, 3 из-за чрезмерного увеличения числа итераций в результате переупорядочения при выбранном способе разбиения для использования OpenMP технологии и относительно небольшого размера матрицы. Из рис. 7-9 видно, что время счета задач с матрицами **5_262144**, **apache1**, **g2_circuit** методами VIC-CG и IC(0)-CG мало отличается.

Числа итераций метода VIC-CG при решении задач с матрицами **5_1048576**, **parabolic_fem**, **apache2**, **ecology2** на 4, 8, 16 процессорах при применении MPI и MPI+OpenMP с использованием предложенного в настоящей работе подхода были в подавляющем большинстве случаев меньше чисел итераций метода BJIC(0)-CG, приведенных в работе [24].

Числа итераций метода VIC-CG при решении задач с матрицами **5_1048576**, **parabolic_fem**, **apache2**, **ecology2** на 4, 8, 10, 16 процессорах при применении MPI и MPI+OpenMP с использованием предложенного в настоящей работе подхода значительно больше чисел итераций метода BIIC1(τ)-CG, BIIC2S(τ)-CG, приведенных в работах [16, 28]. Однако алгоритм вычисления матрицы предобусловливания в методе VIC-CG гораздо проще, чем в методах BIIC1(τ)-CG, BIIC2S(τ)-CG. Заметим, что время вычисления предобусловливателя в методе BIIC2S(τ)-CG относительно велико по сравнению с временем счета итерационного процесса.

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 работе [26] на примере решения модельной задачи 1, описанной в начале раздела 6, при различных значениях размерности матрицы n методом CG предобусловливанием блочного Якоби в сочетании с IC2S(τ) показано, что при фиксированном числе процессоров ускорение счета благодаря 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 надеяться, что потеря эффективности от применения OpenMP технологии при решении задач с большими размерами матриц наступит при значительно большем числе процессоров.

Заметим, что при использовании MPI и MPI+OpenMP плохо масштабировался алгоритм вычисления нижнетреугольной и верхнетреугольной частей переупорядоченной матрицы A. Ускорение счета обеспечивалось главным образом за счет ускорения счета итерационного

процесса. Если матрица A имеет большие размеры, а число итераций предобусловленного метода сопряженных градиентов невелико, то можно ожидать недостаточного эффекта распараллеливания.

6. Заключение

В работе предложены способы применения MPI+OpenMP технологии построения и обращения предобусловливателя VIC при решении СЛАУ (1.1) предобусловленным методом сопряженных градиентов с произвольной симметричной положительно определенной матрицей.

Способ применения MPI при построении и обращении предобусловливателя основан на использовании переупорядочения строк и столбцов матрицы в соответствии с использованием упорядочения узлов сетки всей области расчета типа DDO. При применении MPI+OpenMP технологии проводится дополнительное переупорядочение строк и столбцов матрицы в соответствии с дополнительным переупорядочением внутренних узлов сетки, полученных в результате переупорядочения для применения MPI. При построении и обращении матрицы предобусловливания OpenMP технологии применяются для большинства строк матрицы.

С помощью расчетов двух модельных задач и ряда задач из коллекции SuiteSparse на небольшом числе процессоров показано, что применение MPI+OpenMP технологии позволяет значительно ускорить вычисления для небольшого числа процессоров. Время решения небольших тестовых задач на небольшом числе процессоров методом VIC-CG было близко к времени их решения методом IC(0)-CG.

Список литературы

1. Kershow D. The Incomplete choleski-conjugate gradient method for the iterative solution of systems of linear equations // J.Comp. Phys. 1978. V 26. P. 43-65.
2. Meijering J.A., van der Vorst H.A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix // Math. Comp. 1977. V.31. P. 148-162.
3. Милюкова О.Ю. Параллельные итерационные методы с факторизованной матрицей предобусловливания для решения эллиптических уравнений. Диссерт. на соиск. степ. д-ра физ.-мат. наук. Москва. 2004. 219 с.
4. Duff I.S., Meurant G.A. The effect of ordering on preconditioned conjugate gradients // BIT. 1989. V. 29. P. 625-657.
5. Doi S. On parallelism and convergence of incomplete LU factorizations //Applied Numerical Mathematics: Transactions of IMACS. 1991. V. 7. № 5. P. 417–436.
6. Notay Y. An efficient parallel discrete PDE solver // Parallel Computing. 1995. V. 21. P. 1725-1748.
7. Milyukova O.Yu. Parallel approximate factorization method for solving discreate elliptic equations // Parallel Computing. 2001. № 27. P. 1365-1379.

8. 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. Т. 46. № 6. С. 1096-1112.
9. Hysom D., Pothen A. A scalable parallel algorithm for incomplete factor preconditioning // SIAM J. Sci. Comput. 2001. V. 22. P. 2194-2215.
10. Karypis G., Kumar V. Parallel threshold-based ILU factorization // in Proceedings of the ACM/IEEE Conference on Supercomputing. ACM, New York, IEEE, Washington. DC. 1997.
11. 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. P. 45–64.
12. Милюкова О.Ю. Сочетание числовых и структурных подходов к построению неполного треугольного разложения второго порядка в параллельных методах предобусловливания // Журн. вычисл. матем. и матем. физ. 2016. Т. 56. N5. С. 711-729.
13. 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. V. 5. P. 483-509.
14. 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. Т. 41. № 4. С. 515–528.
15. 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.: Из-во ВЦ РАН. 2011. С. 132-157.
16. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателями блочного неполного обратного треугольного разложения второго и первого порядка // ВАНТ. Серия Математическое моделирование физических процессов. 2022. Вып. 1. С. 48-61.
17. Kaporin I.E. New convergence results and preconditioning strategies for conjugate gradient method // Numer. Linear Algebra and Appls. 1994. V. 1. N 2. P. 179-210.
18. Anderson E. C., Saad Y. Solving sparse triangular systems on parallel computers // International J. of High Speed Computing. 1989. V. 1. P. 73–96.
19. Hammond S. W., Schreiber R. Efficient ICCG on a shared memory multiproces530sor, International J. High Speed Computing 4. 1992. P. 1–21.
20. Wolf M. M., Heroux M. A., Boman E. G. Factors impacting performance of 535 multithreaded sparse triangular solve // in: Proceedings of the 9th International Conference on High Performance Computing for Computational Science. VECPAR'10. Springer-Verlag. Berlin. Heidelberg. 2011. P. 32-44.

21. Chow E., Anzt H., Scott J., Dongarra, J. Using Jacobi iterations and blocking for solving sparse triangular systems in incomplete factorization preconditioning // Journal of Parallel and Distributed Computing. 2018. N. 119. P. 219-230.
22. Chow E., A. Patel A. Fine-grained parallel incomplete LU factorization // SIAM J. Sci. Comput. 2015. V. 37. P. 169-193.
23. 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. Science & Technology Facilities Council. UK. 2018. 27 P.
24. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованным предобусловливателем // Препринты ИПМ им. М.В. Келдыша. 2020. № 31. 22 с. https://doi.org/10.20948/prepr-2020-31.
25. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного Якоби IC1 // М.: Препринты ИПМ им. М.В. Келдыша. 2020. № 83. 28 с. https://doi.org/10.20948/prepr-2020-83.
26. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованными неявными предобусловливателями // Математическое моделирование. 2021. Т. 33. № 10. С.19-39.
27. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем блочного неполного обратного треугольного разложения IC2S и IC1 // Препринты ИПМ им. М.В. Келдыша РАН № 48. Москва. 2021 г. 32 с. https://doi.org/10.20948/prepr-2021-48.
28. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателями блочного неполного обратного треугольного разложения первого порядка // Ж. вычисл. мат. и програм. 2022. Т. 23. Вып. 3. С. 191-206. doi 10.26089/NumMet.v23r312
29. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с предобусловливателем IC(0) на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В.Келдыша. 2022. № 63. 32 с. https://doi.org/10.20948/prepr-2022-63.
30. Капорин И.Е., Милюкова О.Ю. Неполное обратное треугольное разложение в параллельных алгоритмах предобусловленного метода сопряженных градиентов // Препринты ИПМ им. М.В.Келдыша. 2017. №37. 28 с. doi:10.20948/prepr-2017-37.
31. Davis T., Hu Y.F. University of Florida sparse matrix collection / ACM Trans. on Math.~Software. 2011. V. 38, N. 1.
http://www.cise.ufl.edu/research/sparse/matrices.
32. Axelsson O. Iterative solution methods. New York: Cambridge Univ. Press, 1994.
33. Капорин И.Е. Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. Т. 52. № 2. С.1-26.

Оглавление