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

Препринты ИПМ • Препринт № 61 за 2023 г.

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

О.Ю. Милюкова

Параллельная реализация метода сопряженных градиентов с предобусловливателем IC1 на основе использования переупорядочения узлов сетки

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. Параллельная реализация метода сопряженных градиентов с предобусловливателем IC1 на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В.Келдыша. 2023. № 61. 28 с. https://doi.org/10.20948/prepr-2023-61
https://library.keldysh.ru/preprint.asp?id=2023-61

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

Параллельная реализация метода сопряженных градиентов с предобусловливателем IC1 на основе использования переупорядочения узлов сетки

Москва — 2023

Милюкова О.Ю.

Параллельная реализация метода сопряженных градиентов с предобусловливателем IC1 на основе использования переупорядочения узлов сетки

В работе предлагается способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя неполного треугольного разложения Холецкого первого порядка IC1(τ) для решения системы линейных алгебраических уравнений с произвольной симметричной положительно определенной матрицей на небольшом числе процессоров. Способы применения MPI и MPI+OpenMP технологии основаны на использовании упорядочений узлов сетки, согласованных с разбиением области расчета. При построении матрицы предобусловливания IC1(τ) с использованием MPI производится отсечение по позициям в некоторых ее строках. Применение OpenMP технологии при построении и обращении предобусловливателя осуществляется для большинства строк матрицы. Проводится сравнение времени решения задач методом сопряженных градиентов с предобусловливателем IC1(τ) с использованием MPI и гибридной MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse.

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

Olga Yurievna Milyukova

Parallel implementation of the conjugate gradient method with the IC1 preconditioner based on the use of grid node reordering

The paper proposes a method for using MPI+OpenMP technology for constructing and inverting a first-order incomplete triangular Cholesky expansion preconditioner IC1(τ) for solving system of linear algebraic equations with an arbitrary symmetric positive definite matrix on a small number of processors. 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. When constructing the preconditioning matrix IC1(τ) using MPI, cutting is performed at positions in some of its rows. The use of OpenMP technology in the construction and inversion of the preconditioner is carried out for most rows of the matrix. The time taken to solve problems using the conjugate gradient method with the IC1(τ) preconditioner using MPI and hybrid MPI+OpenMP technology is compared using the example of a model problem and a number of problems from the SuiteSparse collection of sparse matrices.

Keywords: incomplete Cholesky factorization, Domain Decomposition ordering, parallel preconditioning, conjugate gradient method

1. Введение

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

$$Ax = b \qquad (1.1)$$

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

$$\left\| b - Ax_k \right\| \le \varepsilon \left\| b - Ax_0 \right\|, \text{ где } 0 < \varepsilon << 1. \qquad (1.2)$$

Для предобусловливания будем использовать неполное треугольное разложение Холецкого с отсечением по параметру первого порядка IC1(τ) (Incomplete Cholesky). Здесь и далее $0 < \tau << 1$ – параметр отсечения. При этом используется факторизованная матрица предобусловливания

$$B \approx A, \; B = U^T U,$$

где U – верхнетреугольная матрица. Одним 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 [1]. Заметим, что предобусловливание IC1(τ) имеет ограниченную область применимости [2].

Основная трудность распараллеливания алгоритмов построения и обращения неявного предобусловливателя неполного треугольного разложения связана с рекурсивным характером вычислений. Одним из способов ее преодоления является использование переупорядочений узлов сетки и соответствующих перестановок строк и столбцов матрицы, например, использование упорядочений, связанных с разбиением области расчета DDO (Domain Decomposition ordering) [3]. 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 [4-8]. Однако в этих работах используются предобусловливатели, в которых в качестве критерия заполнения не используется параметр отсечения по значению, большая часть этих работ посвящена распараллеливанию вычислений при проведении расчетов задач на ортогональных сетках.

В работе [9] предлагается использовать упорядочение узлов сетки типа DDO для построения параллельного варианта метода стабилизированного неполного треугольного разложения второго порядка сопряженных градиентов (IC2S(τ)-CG) [10]. При этом производится отсечение по позициям для некоторых элементов матрицы предобусловливания.

В работах [11, 12, 13] 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 [14] в сочетании с неполным треугольным разложением второго порядка IC2(τ) [10] – BIIC-IC2(τ) [11], IC2S(τ) – BIIC-IC2S(τ) [12] и первого порядка IC1(τ) – BIIC-IC1(τ) [13]. Для построения этих предобусловливателей специальным образом строятся блоки с налеганием, а внутри блоков используется приближенное треугольное разложение IC2(τ), IC2S(τ), IC1(τ).

Проблеме использования высокоуровнего параллелизма (мелкозернистого или распараллеливания алгоритма на потоки) при построении и обращении неявного факторизованного предобусловливателя посвящен ряд работ. Среди них следует отметить работы, в которых для мелкозернистого распараллеливания используется переупорядочение узлов сетки типа DDO. В работах [15, 16] предложены два безытерационных способа применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с неполным треугольным разложением Холецкого без заполнения IC(0) [17], IC1(τ) и IC2S(τ). Один из них основан на переупорядочении узлов сетки типа DDO [9] внутри каждой подобласти, полученной для крупнозернистого распараллеливания.

В работе [13] предлагается использовать MPI+OpenMP технологии для параллельной реализации методов BIIC-IC1(τ)-CG и BIIC-IC2S(τ)-CG, причем число блоков в предобусловливателях кратно числу используемых процессоров и числу используемых потоков. В работе [18] рассматривается способ применения MPI+OpenMP технологии для параллельной реализации построения и обращения предобусловливателя BIIC-IC1(τ) [13] (способ 1) и предлагается способ 2 использования MPI+OpenMP технологии. В способе 2 для мелкозернистого распараллеливания применяется упорядочение узлов сетки типа DDO [9]. Расчеты показали, что при использовании способа 2 время счета тестовых задач с сильно разреженной матрицей было, как правило, немного меньше, чем при использовании способа 1. Заметим, что, как показали расчеты тестовых задач, использование переупорядочения узлов сетки типа DDO для мелкозернистого распараллеливания построения и обращения предобусловливателя BIIC-IC2S(τ) неэффективно.

В работе [19] предложены два способа применения MPI и MPI+OpenMP технологии для построения и обращения предобусловливателя IC(0) [17] в

предобусловленном методе сопряженных градиентов, которые отличаются способом вычисления матрицы предобусловливания. В этой работе упорядочение типа DDO [9] используется как для крупнозернистого, так и для мелкозернистого распараллеливания. В работе [20] предложен способ применения MPI+OpenMP технологии для параллельной реализации построения и обращения варианта предобусловливателя неполного разложения Холецкого без заполнения VIC [21]. При этом для крупнозернистого и мелкозернистого распараллеливания используется упорядочение узлов сетки типа DDO [9].

С помощью расчетов тестовых задач в работах [13, 15, 16, 18- 20] показано, что применение MPI+OpenMP технологии позволяет существенно ускорить вычисления по сравнению с применением только MPI для не слишком большого числа узлов суперкомпьютерной системы. В частности, для тестовых задач, рассмотренных в настоящей работе, в работах [15, 16] удалось существенно ускорить вычисления при использовании не более 10–20 процессоров, а в работах [13, 18-20] – не более 10 процессоров.

В формуле (1.1) предполагается, что матрица A уже 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 [22], являющиеся обобщением упорядочения [12]. Подход, предложенный в этих работах, позволяет одновременно произвести разбиение односвязной 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 предлагается способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя IC1(τ) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей на небольшом числе процессоров. В нем DDO упорядочение [9] используется как для крупнозернистого, так и для мелкозернистого распараллеливаний этапов построения и обращения предобусловливателя IC1(τ). При построении матрицы предобусловливания с использованием MPI производится отсечение по позициям в некоторых ее строках. Применение OpenMP технологии при построении и обращении предобусловливателя осуществляется для большинства строк матрицы. В работе проводится сравнение времени решения тестовых задач методом IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse [23], а также

сравнение скорости сходимости методов IC1(τ)-CG, IC(0)-CG, BIIC-IC1(τ) и BIIC-IC2S(τ) при решении ряда тестовых задач с использованием MPI и MPI+OpenMP технологии.

2. Предобусловленный метод сопряженных градиентов

Алгоритм предобусловленного метода сопряженных градиентов (см., например, [24]) 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 ($B \approx A$). Этот алгоритм 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 MPI+OpenMP реализации всех операций, кроме вычисления $B^{-1} r_{k+1}$, $B^{-1} r_0$, не вызывает сомнений.

3. Алгоритм построения предобусловливателя IC1(τ)

Перед построением матрицы предобусловливания IC1(τ) необходимо выполнить масштабирование матрицы A, как описано во введении. Матричная схема метода предобусловливания IC1(τ) имеет вид [1]

$$A = U^T U - E,$$

где U – верхнетреугольная матрица. Выбор элементов e_{ij} матрицы погрешности E осуществляется так, чтобы «убрать» все «малые» значения $v_{ij} = u_{ii} u_{ij} - e_{ij} = a_{ij} - \sum_{s=1}^{i-1} u_{si} u_{sj}$, где u_{ij} – элементы матрицы U, j = i, …, n. Будем использовать алгоритм 2 [13]. Напомним, что после масштабирования $a_{ii} = 1$.

Алгоритм 2

1. Инициализация вспомогательной диагональной матрицы:
   for $i=1,\ldots, n$

$d_i := 1$

end for

Цикл по строкам A для вычисления строк U :

for $i=1,...,n$

2. Инициализация вектора v при помощи i-й строки A:

for $j=i+1,..., n$

$v_j := a_{ij}$

end for

3. Сделать поправку к вектору v:

for $t=1,...,i-1$

for $j=i+1,...,n$

$v_j = v_j - u_{ti}u_{tj}$

end for

end for

4. Вычисление u_{ii}:

$u_{ii} := \sqrt{d_i}$

5. Вычисление элементов u_{ij} при $j>i$:

for $j=i+1, ..., n$

if $\left|v_j\right| \geq \tau u_{ii}$ then

$u_{ij} := v_j / u_{ii}$

else

$u_{ij} := 0$

end if

end for

6. Выполнение поправки к вспомогательной диагональной матрице:

for $j=i+1,..., n$

$d_j := d_j - u_{ij}^2$

end for

end for *(конец цикла по i)*

Здесь и далее d_i – элементы вспомогательной диагональной матрицы. При написании программы для реализации алгоритма 2 необходимо обеспечить доступ по столбцам к элементам матрицы U. При вычислении u_{ii} в п. 4 алгоритма 2 необходимо извлекать квадратный корень из числа, которое определяется в процессе вычислений. Это выражение может оказаться отрицательным. В этом случае следует для решения СЛАУ (1.1) уменьшить

значение параметра τ или использовать метод сопряженных градиентов с другим предобусловливателем.

4. Алгоритм реализации построения и обращения предобусловливателя IC1(τ) с использованием MPI

Пусть матрица A переупорядочена и отмасштабирована. Разобьем каким-либо образом произвольную (возможно, трехмерную) область расчета на p подобластей, где p – предполагаемое число используемых процессоров, выберем некоторую нумерацию подобластей и будем использовать упорядочение узлов сетки, предложенное в работе [9]. Для этого введем множество узлов разделителей – множество узлов сетки в подобластях, у которых имеются соседи из подобластей с бóльшим 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, бóльшими, 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 бóльшим номером. Остальные узлы разделителей назовем узлами разделителей третьего уровня.

Используя определение узлов разделителей первого, второго и третьего уровней и доказательство от противного, получим, что в шаблонах узлов разделителей первого уровня могут быть только внутренние узлы из подобластей с тем же или бóльшими номерами и узлы разделителей из той же подобласти и подобластей с меньшими номерами. В шаблонах узлов разделителей второго уровня могут быть внутренние узлы из подобластей с тем же или бóльшими номерами, узлы разделителей первого уровня из подобластей с тем же или бóльшими номерами, узлы разделителей более высокого уровня, чем первый, но только из рассматриваемой подобласти и подобластей с меньшими номерами. Используя доказательство от противного, можно доказать, что в шаблонах узлов разделителей первого уровня не может быть узлов разделителей первого уровня из других подобластей; в шаблонах узлов разделителей второго уровня не может быть узлов разделителей второго уровня из других подобластей.

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.

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 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.

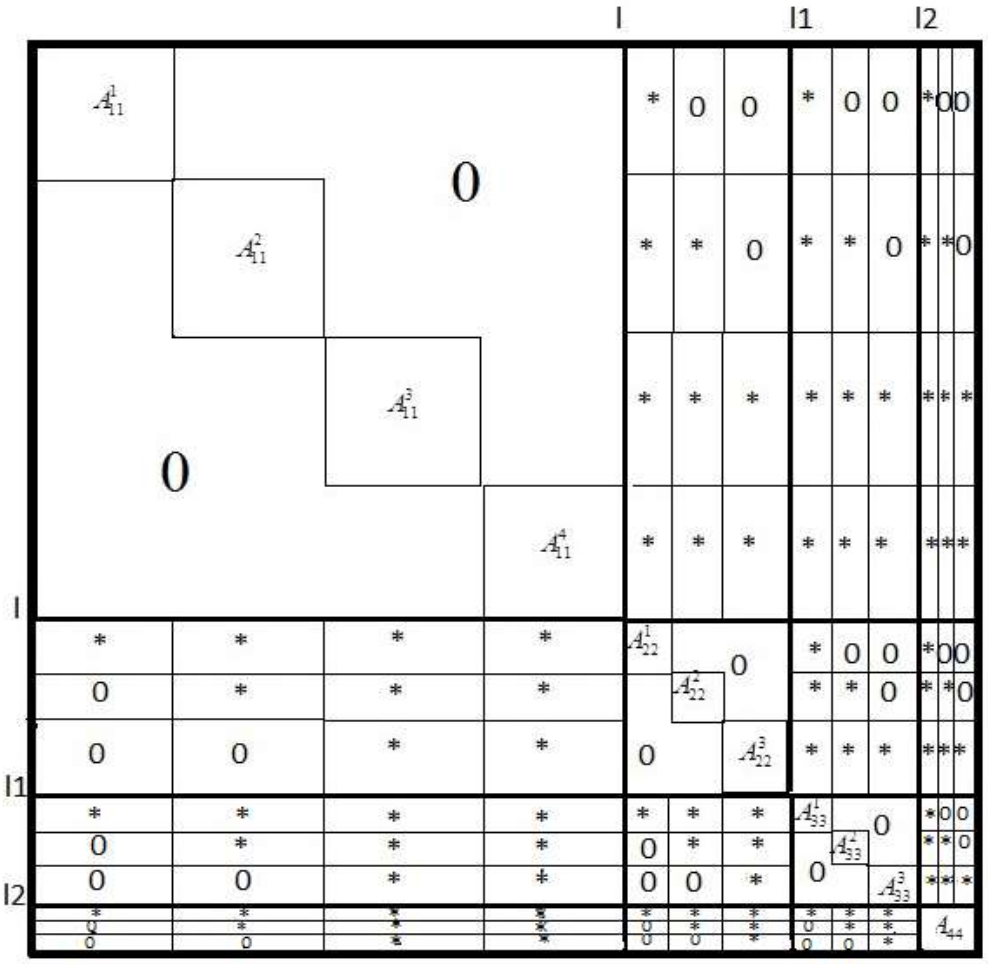

Рис. 1. Вид структуры разреженности матрицы *A*, полученной после перестановки строк и столбцов в результате переупорядочения

Способ применения MPI при построении и обращении предобусловливателя IC1(τ) является аналогичным способу 1 применения MPI при построении и обращении предобусловливателей IC(0) [19] и IC2S(τ) [9]. Перед вычислением элементов матрицы U необходимо произвести переупорядочение строк и столбцов матрицы A в соответствии с новым упорядочением узлов сетки и положить $d_i := 0$ для $i = 1, 2, \ldots, n$.

Определение элементов матрицы U начинается с вычисления ее элементов в строках, соответствующих внутренним узлам всех подобластей. Используя метод математической индукции, можно доказать, что в шаблонах внутренних узлов в матрице U не могут присутствовать внутренние узлы из других подобластей. Поэтому вычисление элементов матрицы U в строках, соответствующих внутренним узлам, может происходить во всех процессорах одновременно. При этом используется алгоритм, аналогичный алгоритму 2, в котором циклы по i и по t происходят по внутренним узлам соответствующих подобластей.

Перед переходом к вычислениям элементов матрицы U в строках, соответствующих узлам разделителей первого уровня, следует вычислить вспомогательную матрицу V, элементы которой V_{ij} определяются с помощью алгоритма 3, аналогичного алгоритму 2 из работы [9].

Алгоритм 3

1. for $s=1,\ldots, p$
   Вычислить V_{ij}^s для $i=l+1,\ldots,n, j=i+1,\ldots,n$:

```
 for i=l+1,...,n
   for j=i+1,...,n
     if (g2(i) = g2(j)) ∧ (g1(i) ≠ g1(j)) then
        V_ij^s := 0
     else
        for t=i1(s), ..., i2(s)
          V_ij^s = V_ij^s + u_ti u_tj
        end for
     end if
   end for
 end for
end for (конец цикла по s)
```

2. Вычислить V_{ij} для $i=l+1,\ldots,n, j=i+1,\ldots,n$:

for $i=l+1,\ldots, n$,

for $j=i+1,\ldots,n$

$$V_{ij} = \sum_{s=1}^{p} V_{ij}^{s}$$

end for

end for

Здесь и далее $i_1(s)$, $i_2(s)$ – номера первой и последней строк, соответствующих внутренним узлам в подобласти с номером *s*, используются две целочисленные функции $g_1(i)$ и $g_2(i)$, определенные на множестве узлов разделителей: $g_1(i)=\hat{s}$, где $\hat{s}$ – номер подобласти, содержащей узел с номером i, $g_2(i)$ – номер уровня разделителей для узла с номером i. В отличие от алгоритма 4 из работы [19], в пункте 1 присутствует отсечение по позициям элементов матрицы *V*, что необходимо для лучшей параллелизуемости алгоритмов построения и обращения матрицы *U*. В работе [19] в этом нет необходимости, так как структура верхнетреугольного множителя матрицы предобусловливания IC(0) совпадает со структурой верхнетреугольной части матрицы *A*.

Кроме вычисления элементов матрицы *V* производится суммирование по всем процессорам вычисленных там неокончательных значений d_j для j, соответствующих номерам узлов разделителей. Параллельная реализация этих этапов аналогична их параллельной реализации в случае использования предобусловливателя IC(0) и описана в работе [19].

Остановимся на вычислении элементов матрицы *U* в строках, соответствующих узлам разделителей первого уровня. Так как $V_{ij}=0$ и $a_{ij}=0$ для i и j, соответствующих узлам разделителей первого уровня, находящимся в разных подобластях, то методом математической индукции можно доказать, что для этих значений ij $u_{ij}=0$. Следовательно, вычисление элементов матрицы *U* в узлах разделителей первого уровня может происходить во всех процессорах одновременно и независимо, каждый процессор будет производить вычисления в своей подобласти, используется алгоритм 4. Напомним, что перед началом вычисления матрицы *U* всем d_i было присвоено значение 0.

Алгоритм 4

for *s*=1, …, *p*

1. Инициализация вспомогательной диагональной матрицы:

for $i=\bar{i}_1(s),\ldots,\bar{i}_2(s)$

$d_i := d_i + 1$

end for
for $i = \bar{i}_1(s),..., \bar{i}_2(s)$

2. Инициализация вектора *v* при помощи *i*-ой строки *A* и *i*-ой строки *V*:

for *j=i+1, ..., n*
$v_j := a_{ij} + V_{ij}$
end for

3. Сделать поправку к вектору *v*:

for $t = \bar{i}_1(s), ..., i-1$
for *j=i+1, ..., n*
$v_j = v_j - u_{ti} u_{tj}$
end for
end for

4. Вычисление u_{ii}:

$u_{ii} := \sqrt{d_i}$

5. Вычисление элементов u_{ij} при *j>i :*

for *j=i+1, ..., n*
if $\left|v_j\right| \geq \tau u_{ii}$ then
$u_{ij} := v_j / u_{ii}$
else
$u_{ij} := 0$
endif
end for

6. Выполнение поправки к вспомогательной диагональной матрице:

for *j=i+1, ..., n*
$d_j := d_j - u_{ij}^2$
end for
end for (*конец цикла по i*)
end for (*конец цикла по s*)

В алгоритме 4 $\bar{i}_1(s)$, $\bar{i}_2(s)$ – номера первой и последней строк на разделителях первого уровня в подобласти с номером *s*.

Перед переходом к вычислению элементов матрицы *U* в строках, соответствующих узлам разделителей второго уровня, следует вычислить элементы вспомогательной матрицы $\overline{V}$, что осуществляется с использованием алгоритма, аналогичного алгоритму 3. Кроме того, нужно произвести

суммирование по всем процессорам вычисленных там значений Δd_j (изменений значений d_j, полученных на этапе 6 алгоритма 4) для j, соответствующих номерам узлов разделителей второго и третьего уровня, и добавить Δd_j к вычисленным ранее неокончательным значениям d_j.

Вычисление элементов матрицы U в строках, соответствующих узлам разделителей второго уровня, производится аналогично вычислению элементов этой матрицы в строках, соответствующих узлам разделителей первого уровня. При этом в п. 2 алгоритма 4 добавляется еще одно слагаемое: $\overline{V}_{ij}$, где $\overline{V}_{ij}$ – элементы матрицы $\overline{V}$. Затем вычисляются элементы вспомогательной матрицы $\tilde{V}$, что делается аналогично вычислению элементов матриц V, $\overline{V}$, и производится суммирование по всем процессорам вычисленных там значений Δd_j для номеров j, соответствующих узлам разделителей третьего уровня, и добавление Δd_j к вычисленным ранее неокончательным значениям d_j.

Если блок A_{44} имеет блочно-диагональную структуру с нулевыми элементами вне блоков**,** то вычисление элементов матрицы U в строках, соответствующих узлам разделителей третьего уровня, производится аналогично. Если подматрица A_{44} не является блочно-диагональной с нулевыми элементами вне блоков, то при построении матрицы U будем производить отсечение элементов подматрицы A_{44} вне блоков по позициям.

На рис. 2 приведен вид структуры разреженности матрицы U.

Итак, вычисление элементов матриц U в строках, соответствующих внутренним узлам, производится с отсечением только по значению, а вычисление элементов этих матриц в строках, соответствующих узлам на разделителях, производится с отсечением по значению и по позициям. Вычисление элементов матрицы U во внутренних узлах, в узлах разделителей каждого уровня производится во всех процессорах одновременно.

Обращение матрицы предобусловливания IC1(τ) осуществляется так же, как обращение матрицы предобусловливания IC(0) в работе [19]. Перед обращением матрицы предобусловливания необходимо произвести переупорядочение элементов вектора r^{k-1}, где k – номер итерации в 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 первоначально в работе [12], подробно описанный в работе [19]. Будем вычислять элементы вектора $z = D_U \hat{w}^k$, где D_U – диагональная часть.

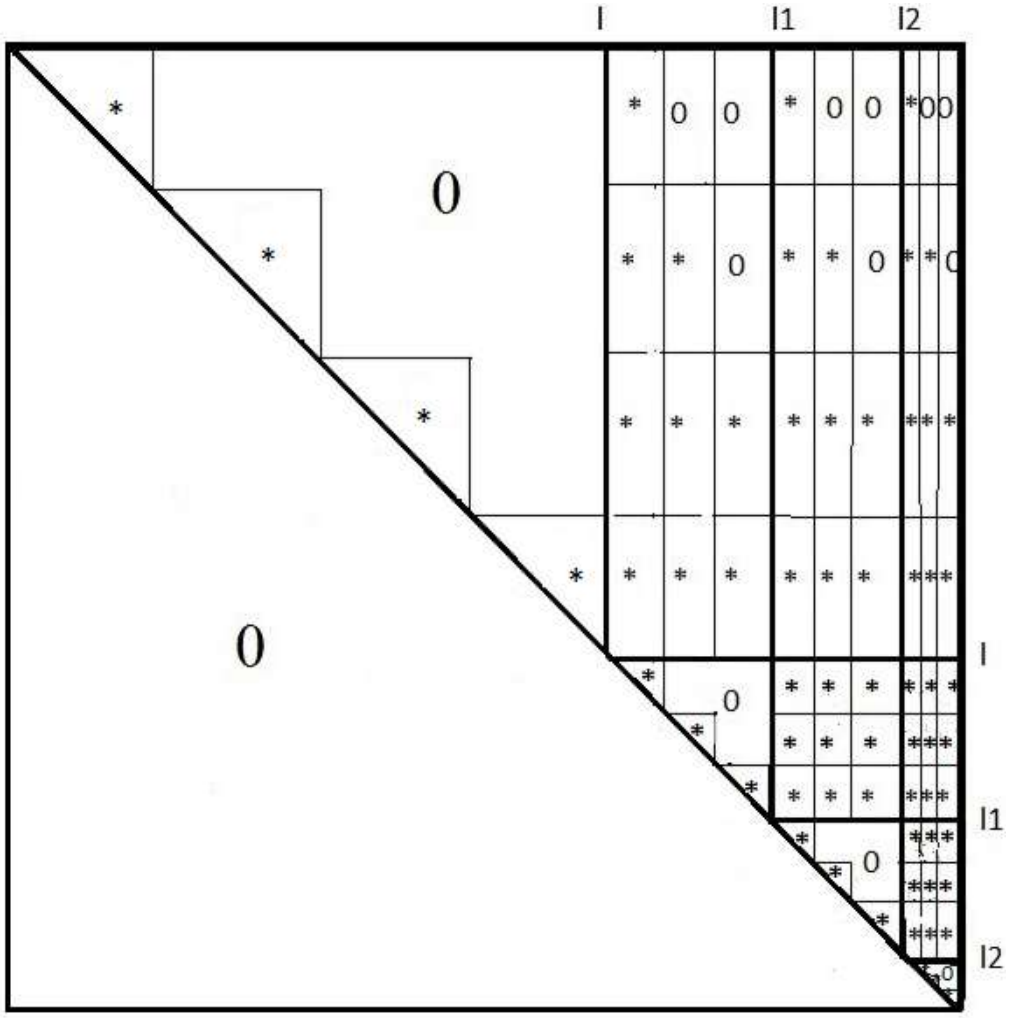

Рис. 2. Вид структуры разреженности матрицы *U*

матрицы *U*, а затем по элементам вектора *z* вычислять элементы вектора $\hat{w}^k$. 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 вид:

Алгоритм 5

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

3. 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*).

После этого осуществляется пересылка значений слагаемых z_j для индексов 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, аналогично описанному в работе [19].

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 осуществляются необходимые пересылки и необходимая коррекция значений z_j. Вычисление значений $\hat{w}_i^k$ в узлах разделителей второго и третьего уровня происходит аналогично их вычислению в узлах разделителей первого уровня и осуществляется во всех процессорах одновременно. Перед вычислением $\hat{w}_i^k$ в узлах разделителей третьего уровня осуществляются необходимые пересылки и необходимая коррекция значений z_j.

На этапе 2 обращения матрицы предобусловливания сначала с использованием матрицы U происходит вычисление значений w_i^k в узлах разделителей третьего уровня, затем второго и первого уровней, а далее во внутренних узлах подобластей. При вычислении w_i^k в узлах разделителей каждого уровня и во внутренних узлах все процессоры работают одновременно. 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. После завершения вычисления вектора w^k следует произвести переупорядочение его элементов.

5. Алгоритм реализации построения и обращения предобусловливателя IC1(τ) с использованием MPI+OpenMP

Способ применения MPI+OpenMP технологии при построении и обращении предобусловливателя IC1(τ) является аналогичным способу 1 применения MPI+OpenMP технологии при построении и обращении предобусловливателя IC(0) [19]. 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* (*s = 1, 2, ..., p*) 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 $\hat{l}_S = \lfloor n_S / m \rfloor$, где n_S – число узлов сетки в подобласти с номером *s*, первые *m*-1 «внутренних» подобластей содержат $\hat{l}_S$ узлов 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.

Произведем переупорядочение типа DDO [9] внутренних узлов сетки в подобластях, полученных при разбиении всей области расчета на *p* подобластей ($p \neq 1$), подробно описанное в предыдущем разделе. В результате будут определены «внутренние» узлы и узлы «разделителей». Здесь и далее кавычки перед и после слов внутренние и разделители означают, 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.

Определим $\bar{l}_S$ – минимальное число «внутренних» узлов при разбиении множества внутренних узлов 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*. 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, далее узлы «разделителей» второго уровня, 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 подобластей, введенного ранее. Затем идут узлы разделителей первого, второго и третьего уровней при упорядочении для крупнозернистого распараллеливания в установленном для крупнозернистого распараллеливания порядке. Таким образом, при использовании MPI+OpenMP технологии производится дополнительное переупорядочение только узлов сетки, являющихся внутренними при упорядочении для использования MPI.

При использовании MPI+OpenMP применение OpenMP технологии в каждом процессоре с номером s осуществляется при построении первых $M1$ строк матрицы U_S, где U_S – часть матрицы U, хранящаяся в процессоре с номером s. При этом используется цикл по $k2 = 1, \ldots, m$, внутри которого осуществляется вычисление элементов строк искомой матрицы с локальными номерами $1, \ldots, M1$, 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 матрицы U с использованием только MPI. Если число узлов во всех «внутренних» подобластях достаточно велико, то для подавляющего большинства строк матрицы U вычисление ее элементов происходит с использованием OpenMP технологии.

При применении MPI+OpenMP технологии 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. Создадим 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 $i = 1, \ldots, M1 = m\bar{l}_S$ (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 $i=M1+1, \ldots, il0(s)$. Затем с использованием матриц $U2_S$ вычисляются слагаемые элементов $z_j = \hat{w}_j U_{jj}$ вектора z для значений j, соответствующих узлам разделителей. 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. Вычисление $\hat{w}_i^k$ в узлах разделителей всех уровней осуществляется так же, как в случае применения только MPI.

На этапе обращения верхнетреугольных матриц вычисления происходят в обратном порядке с использованием матриц U_S. 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. Потом происходит вычисление w_i^k во внутренних узлах сетки. При этом элементы искомого вектора с локальными номерами $i = M1, \ldots, 1$ (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.

6. Результаты расчетов

Программы, реализующие применение метода IC1(τ)-CG для решения СЛАУ (1.1), были написаны на языке FORTRAN 90 с использованием MPI и MPI+OpenMP технологии. Расчеты проводились на многопроцессорном вычислительном кластере К60, установленном в ЦКП ИПМ им. М.В. Келдыша РАН. 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 пятиточечная аппроксимация лапласиана, имя матрицы **5_1048576**. Для тестирования рассматриваемых параллельных методов использовались также некоторые матрицы из коллекции разреженных

матриц SuiteSparse [23]. Перечислим имена используемых тестовых матриц и укажем 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;

tmt_sym – моделирование задач электромагнетизма.

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 [25], 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 Флоридской коллекции

Матрица	n	NZA	Id	Ip	$nz_{\min}$	$nz_{\max}$	$Cond(A_0)$
apache2	715176	4817870	2	0	4	8	0.12 E+7
parabolic_fem	525825	3674625	0	1048576	3	7	0.20 E+6
ecology2	999999	4995991	1124	0	3	5	0.63 E+8
tmt_sym	726713	5080961	724378	149642	3	9	0.25 E+9

Решалось уравнение $Ax = b$, где A – матрица, полученная после переупорядочения, связанного с разбиением на подобласти для использования MPI с помощью алгоритма [22], 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 $\varepsilon = 10^{-8}$.

В таблицах 2–6 приведены числа итераций и время счета методом IC1(τ)-CG тестовых задач при применении MPI и MPI+OpenMP технологии. Под временем вычислений в таблицах 2-6 подразумевается время счета в секундах итерационного процесса в сумме с временем вычисления предобусловливателя. При применении MPI+OpenMP технологии расчеты проводились с использованием 3, 4, 6, 8, 10 и 12 нитей. В таблицах 2–6 приведены оптимальные по числу нитей для каждого p с точки зрения времени вычислений результаты и соответствующие им значения числа использованных нитей, указанные в круглых скобках В таблицах 2–6 приведены курсивом коэффициенты μ ускорения счета задач благодаря использованию OpenMP технологии. В квадратных скобках в таблицах приведены числа итераций при решении соответствующих задач методом IC(0)-CG с применением только MPI способом 1 из работы [19].

Таблица 2

Числа итераций и время счета методом IC1(τ)-CG задачи с матрицей **5_1048576** на *p* процессорах без применения и с применением OpenMP технологии

p	3	4	5	6
MPI [it for IC(0)-CG]	310, 5.33 [1125]	314, 4.44 [1022]	332, 4.43 [1031]	327, 3.27 [1119]
MPI+OpenMP μ	336, 2.65(6) *2.01*	345, 2.11(6) *2.1*	357, 2.49(6) *1.77*	351, 2.19(6) *1.49*

Таблица 3

Числа итераций и время счета методом IC1(τ)-CG задачи с матрицей **ecology2** на *p* процессорах без применения и с применением OpenMP технологии

p	3	4	5	6
MPI [it for IC(0)-CG]	565, 10.53 [2041]	577, 9.25 [2136]	583, 6.95 [2181]	595, 7.39 [2022]
MPI+OpenMP μ	609, 4.31(8) *2.44*	603, 3.63(6) *2.55*	630, 3.72(6) *1.87*	623, 3.65(4) *2.02*

Таблица 4

Числа итераций и время счета методом IC1(τ)-CG задачи с матрицей **parabolic_fem** на *p* процессорах без применения и с применением OpenMP технологии

p	2	3	4	5	6
MPI [IC(0)-CG]	343, 3.9 [830]	342, 3.56 [874]	334, 2.5 [871]	343, 3.33 [918]	333, 2.82 [905]
MPI+OpenMP μ	372,1.27(12) *3.07*	356,1.44(6) *2.47*	351,0.99(6) *2.52*	346,1.62(4) *1.93*	352, 1.52(3) *1.85*

Таблица 5

Числа итераций и время счета методом IC1(τ)-CG задачи с матрицей **apache2** на *p* процессорах без применения и с применением OpenMP технологии

p	2	3	4	5	6
MPI [IC(0)-CG]	348,5.06 [953]	387,4.54 [1297]	362, 3.91 [873]	369, 4.48 [871]	381, 4.41 []927]
MPI+OpenMP μ	369, 1.99(8) *2.54*	394,2.27(8) *2.00*	374,1.72(10) *2.27*	370, 2.67(4) *1.68*	394, 2.84(3) *1.55*

Таблица 6

Числа итераций и время счета методом IC1(τ)-CG задачи с матрицей **tmt_sym** на p процессорах без применения и с применением OpenMP технологии

p	2	3	4	5	6
MPI [IC(0)-CG]	564, 9.6 [1511]	544, 8.16 [1575]	547, 6.08 [1542]	559, 6.58 [1543]	551, 5.91 [1533]
MPI+OpenMP μ	587,2.84(12) *3.38*	582,2.79(8) *2.92*	578, 2.4(6) *2.53*	592,3.77(8) *1.74*	583,2.43(4) *2.43*

Заметим, что применение OpenMP технологии при использовании MPI+OpenMP технологии не позволяет ускорить вычисление матрицы предобусловливания в методе IC1(τ)-CG, так же как и в методе IC(0)-CG. Ускорение решения СЛАУ происходит исключительно благодаря ускорению счета итерационного процесса, причем время счета итерационного процесса составляет основную часть времени решения СЛАУ с этими матрицами. Поэтому хорошая эффективность распараллеливания алгоритма будет наблюдаться главным образом в случае плохо обусловленных матриц.

Как видно из таблиц 2–6, числа итераций при решении тестовых задач методом IC1(τ)-CG с применением MPI в несколько раз меньше, чем при их решении методом IC(0)-CG. Такая же картина наблюдается и при использовании MPI+OpenMP технологии для решения этих задач. Числа итераций при решении этих задач предложенным в настоящей работе методом на небольшом числе процессоров, как правило, меньше чисел итераций при их решении методом сопряженных градиентов с предобусловливателями BIIC-IC1(τ) и BIIC-IC2S(τ) (см. [13], [18]). В таблице 7 в качестве примера приведены числа итераций при решении 4 тестовых задач методами IC1(τ)-CG, BIIC-IC1(τ)-CG и BIIC-IC2S(τ)-CG на 4 процессорах. При этом для

Таблица 7

Числа итераций при решении тестовых задач на 4 процессорах методами IC(0)-CG, BIIC-IC1(τ)-CG, BIIC-IC2S(τ)-CG с использованием MPI и MPI+OpenMP технологии

имя матрицы	IC1(τ)-CG MPI, MPI+OpenMP	BIIC-IC1(τ)-CG MPI, MPI+OpenMP	BIIC-IC2S(τ)-CG MPI,MPI+OpenMP
5_1048576	314, 345 (6)	401, 444(4)	343, 421(6)
ecology2	577, 603(6)	721, 798(4)	608, 698(6)
parabolic_fem	334, 351(6)	418, 453(4)	323, 368(6)
apache2	362, 374(10)	501, 611(4)	418, 509(6)

MPI+OpenMP параллельной реализации используется число блоков, кратное числу используемых процессоров и числу используемых потоков, результаты взяты из работы [13]. В скобках приведено число используемых нитей.

На рисунках 3–7 приведены графики зависимости времени счета тестовых задач методом IC1(τ)-CG от числа процессоров p в логарифмическом масштабе с использованием только MPI (сплошные линии), а также с использованием MPI+OpenMP технологии (штриховые линии).

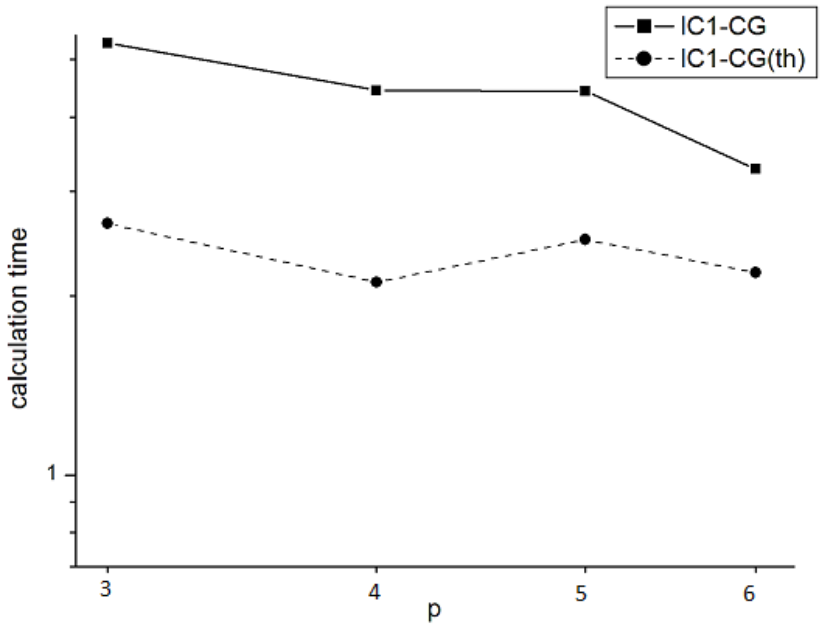

Рис. 3. Время счета задачи с матрицей **5_1048576** методом IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии

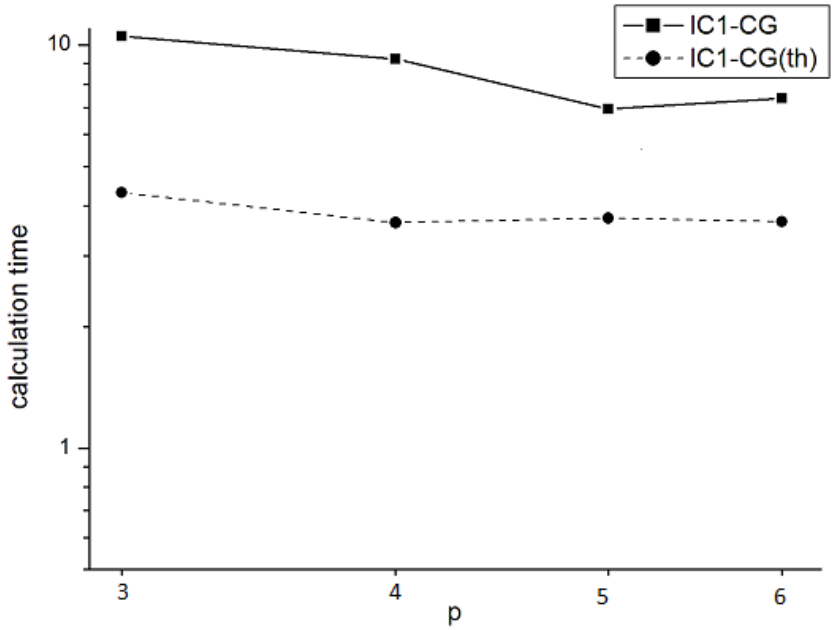

Рис. 4. Время счета задачи с матрицей **ecology2** методом IC1(τ) -CG с использованием MPI и MPI+OpenMP технологии

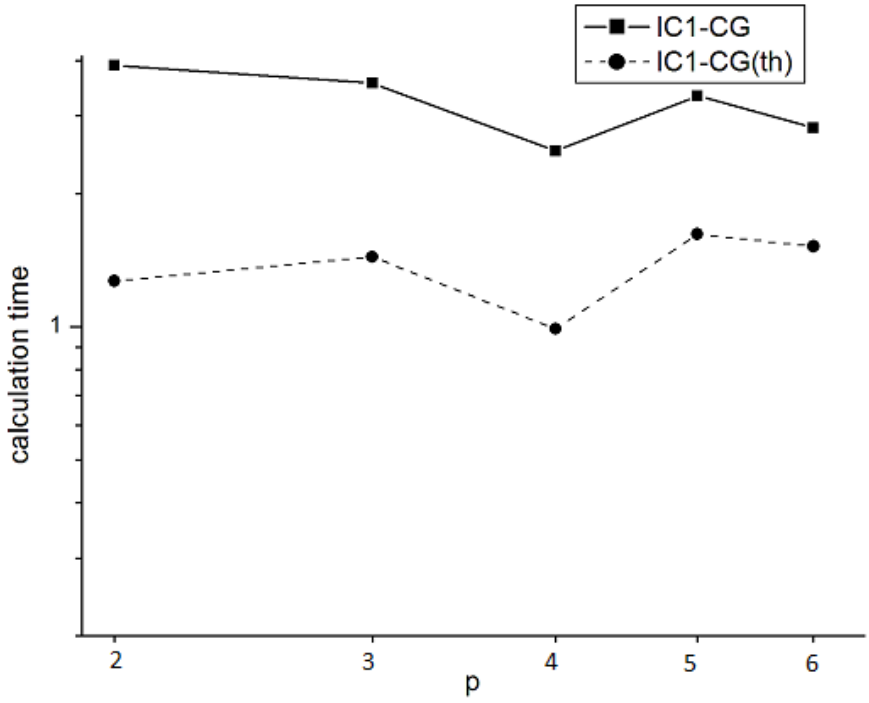

Рис. 5. Время счета задачи с матрицей **parabolic_fem** методом IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии

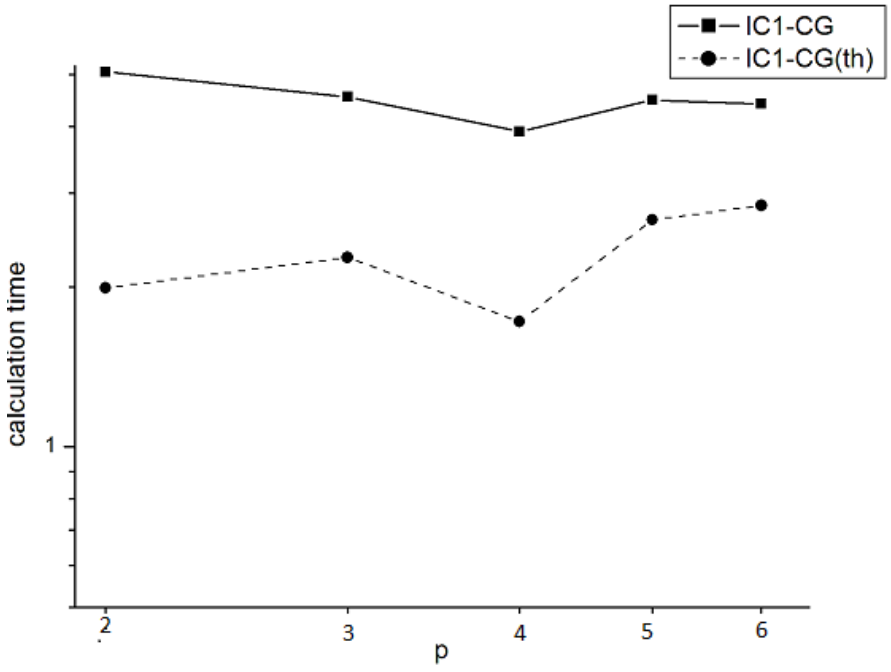

Рис. 6. Время счета задачи с матрицей **apache2** методом IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии

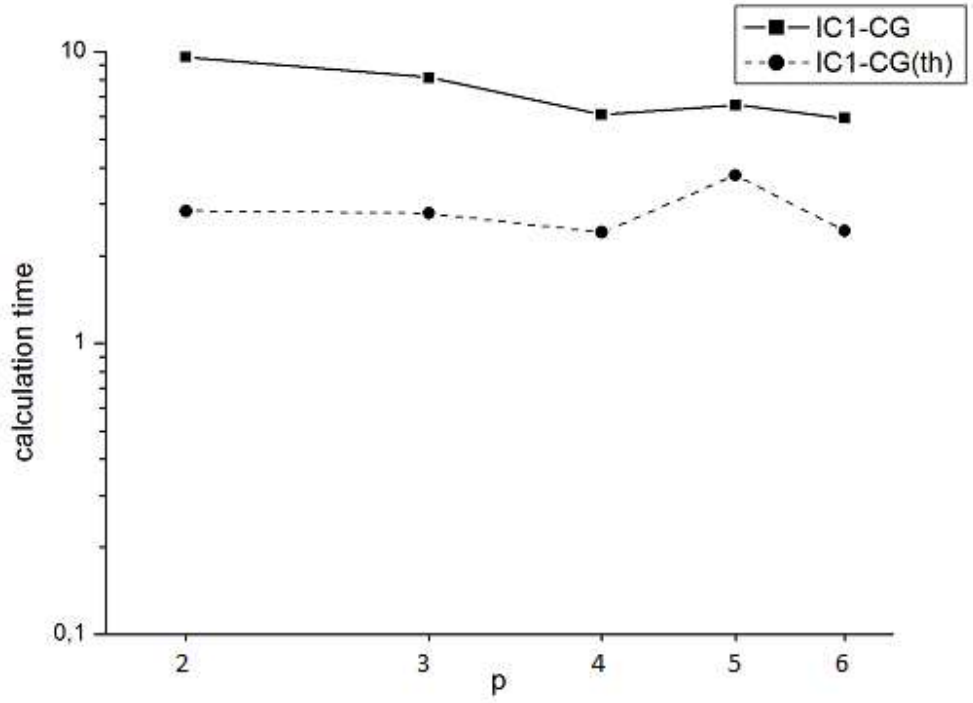

Рис. 7. Время счета задачи с матрицей **tmt_sym** методом IC1(τ)-CG с использованием MPI и MPI+OpenMP технологии

Как видно из таблиц 2–6 и рисунков 3–7, применение MPI+OpenMP технологии для решения всех тестовых задач позволило значительно ускорить их решение по сравнению с применением только MPI на небольшом числе процессоров.

На рисунках 8, 9 приведены графики зависимости ускорения счета от числа процессоров при решении тестовых задач с использованием только MPI и MPI+OpenMP технологии по сравнению со временем решения этих задач на 2 или 3 процессорах с использованием только MPI. Заметим, что на ускорение счета влияет также изменение числа итераций с ростом числа процессоров и числа потоков, которое может иметь немонотонный характер, что связано с особенностями разбиения области расчета.

Как видно из рисунков 8, 9, применение MPI+OpenMP технологии позволяет решать задачи с матрицами **5_1048576**, **ecology2** со сверхлинейным ускорением по сравнению с их решением с использованием только MPI на 3 процессорах при всех использованных значениях p. Как видно из рисунка 9, применение MPI+OpenMP технологии позволяет решать задачи с матрицами **apache2**, **parabolic_fem** со сверхлинейным ускорением по сравнению с их решением с использованием только MPI на 2 процессорах при $p < 5$, а задачу с матрицей **tmt_sym –** при всех использованных значениях p.

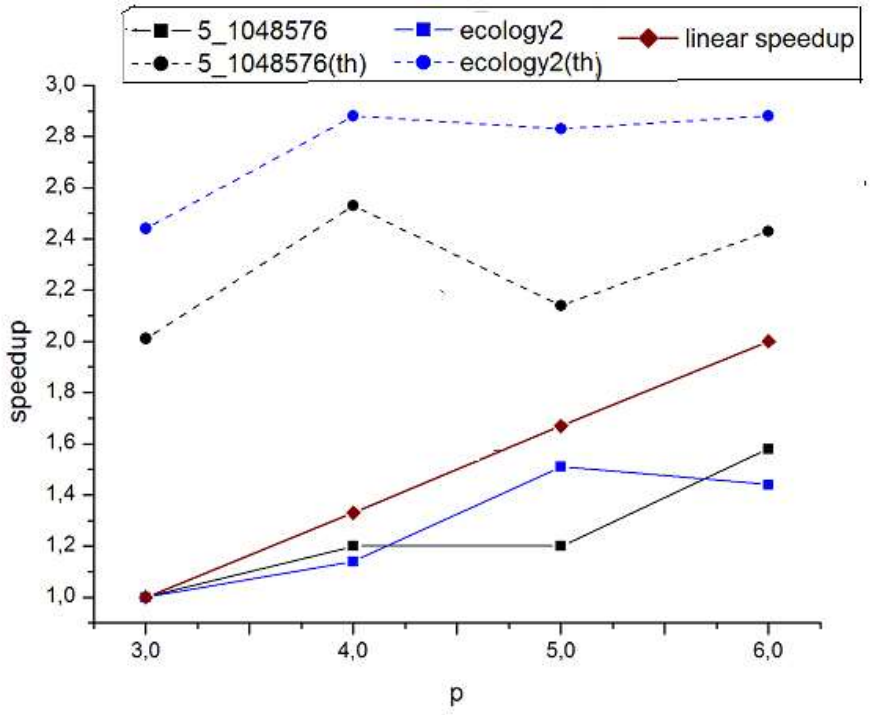

Рис. 8. Ускорение счета задач с матрицами **5_1048576**, **ecology2** при использовании MPI и MPI+OpenMP по сравнению со счетом на 3 процессорах с использованием MPI

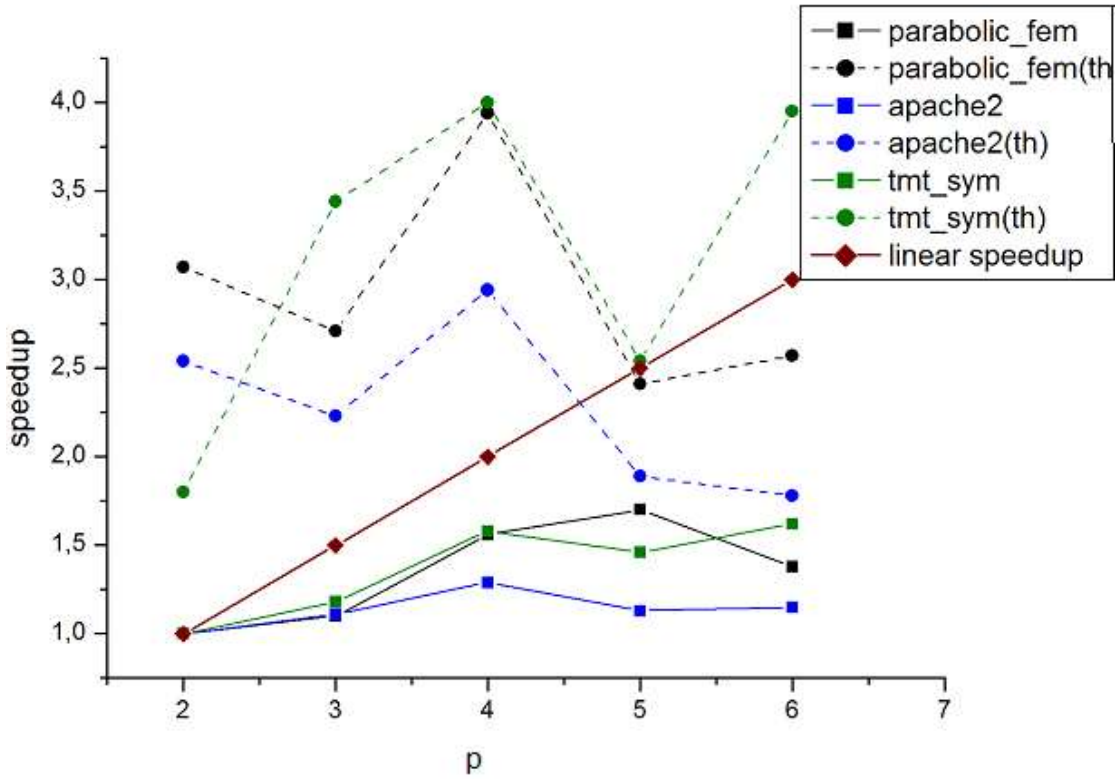

Рис. 9. Ускорение счета задач матрицами **parabolic_fem**, **apache2**, **tmt_sym** при использовании MPI и MPI+OpenMP по сравнению со счетом на 2 процессорах с использованием MPI

7. Заключение

В работе предложен способ применения MPI+OpenMP технологии для построения и обращения предобусловливателя IC1(τ) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей методом сопряженных градиентов на небольшом числе процессоров. 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 построении предобусловливателя IC1(τ) в строках матрицы, соответствующих узлам разделителей, производится отсечение по параметру и по позициям. При применении MPI+OpenMP технологии проводится дополнительное переупорядочение строк и столбцов матрицы в соответствии с дополнительным переупорядочением внутренних узлов сетки. При построении и обращении матрицы предобусловливания OpenMP технологии применяются для большинства строк матрицы.

С помощью расчетов модельной задачи и четырех задач из коллекции SuiteSparse на небольшом числе процессоров показано, что использование OpenMP технологии позволяет значительно ускорить вычисления. Использование предобусловливателя IC1(τ) позволяет сильно уменьшить число итераций предобусловленного метода сопряженных градиентов по сравнению с использованием предобусловливателя IC(0) при параллельной реализации на основе упорядочения типа DDO. Число итераций при решении 4 тестовых задач методом IC1(τ)-CG при параллельной реализации с использованием DDO на небольшом числе процессоров меньше или сопоставимо с числом итераций при их решении методом сопряженных градиентов с предобусловливателями BIIC-IC1(τ) и BIIC-IC2S(τ).

Список литературы

1. Munksgaard N. Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients // ACM Trans. Math. Software. 1980. № 6. P. 206-219.

2. Tuff A.D., Jennings A. An iterative method for lerge systems of linear structural equations // J. Numer. Methods Eng. 1973. № 7. P. 175-183.

3. Duff I. S., Meurant G. A. The effect of ordering on preconditioned conjugate gradients // BIT. 1989. V. 29. P. 625-657.

4. Doi S. On parallelism and convergence of incomplete LU factorizations //Applied Numerical Mathematics: Transactions of IMACS. 1991. V.7. № 5. P. 417–436.

5. Notay Y. An efficient parallel discrete PDE solver // Parallel Computing. 1995. V. 21. P. 1725-1748.

6. Milyukova O.Yu. Parallel approximate factorization method for solving discreate elliptic equations // Parallel Computing. 2001. V. 27. № 10. P. 1365-1379.

7. Милюкова О.Ю. Некоторые параллельные итерационные методы с факторизованными матрицами предобусловливания для решения эллиптических уравнений на треугольных сетках // Ж. вычисл. матем. и матем. физ. 2006. Т. 46. № 6. С. 1096-1112.
8. Magolu Monga Made M., van der Vorst H.A. Spectral analysis of parallel incomplete factorizations with implicit pseudo-overlap // Numer. Linear Algebra Appl. 2002. V. 9. № 1. P. 45–64.
9. 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. № 5. С. 711-729.
10. 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. № 6. P. 483-509.
11. Капорин И.Е., Коньшин И.Н. Параллельное решение симметричных положительно-определенных систем на основе перекрывающегося разбиения на блоки // Ж. вычисл. матем. и матем. физ. 2001. Т. 41. № 4. С. 515–528.
12. 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.
13. 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.
14. Kaporin I.E. New convergence results and preconditioning strategies for conjugate gradient method // Numer. Linear Algebra and Appl. 1994. V. 1. № 2. P. 179-210.
15. 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.
16. Милюкова О.Ю. MPI+OpenMP реализация метода сопряженных градиентов с факторизованными неявными предобусловливателями // Математическое моделирование. 2021. Т. 33. № 10. С. 19-39.
17. 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.
18. 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.

19. Милюкова О.Ю. Способы MPI+OpenMP реализации метода сопряженных градиентов с предобусловливателем IC(0) на основе использования переупорядочения узлов сетки // Препринты ИПМ им. М.В.Келдыша. 2023. № 35. 32 с. https://doi.org/10.20948/prepr-2023-35.
20. 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.
21. 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.
22. 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. https://doi.org/10.20948/prepr-2017-37.
23. Davis T., Hu Y.F. University of Florida sparse matrix collection // ACM Trans. on Math.~Software. 2011. V. 38. № 1.
24. Axelsson O. Iterative solution methods. – New York: Cambridge Univ. Press. 1994.
25. Капорин И.Е. Использование полиномов Чебышева и приближенного обратного треугольного разложения для предобусловливания метода сопряженных градиентов // Ж. вычисл. матем. и матем. физики. 2012. Т. 52. № 2. С. 179-204.

Оглавление