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

Препринты ИПМ • Препринт № 75 за 2024 г.

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

О.Ю. Милюкова

Сочетание числовых и структурных подходов в параллельном методе предобусловливания неполного треугольного разложения первого порядка

Рекомендуемая форма библиографической ссылки: Милюкова О.Ю. Сочетание числовых и структурных подходов в параллельном методе предобусловливания неполного треугольного разложения первого порядка // Препринты ИПМ им. М.В.Келдыша. 2024. № 75. 28 с.
https://doi.org/10.20948/prepr-2024-75
https://library.keldysh.ru/preprint.asp?id=2024-75

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

Сочетание числовых и структурных подходов в параллельном методе предобусловливания неполного треугольного разложения первого порядка

Москва — 2024

Милюкова О.Ю.

Сочетание числовых и структурных подходов в параллельном методе предобусловливания неполного треугольного разложения первого порядка

Рассматривается параллельный вариант метода неполного треугольного разложения Холецкого первого порядка сопряженных градиентов, в котором используется переупорядочение элементов матрицы коэффициентов, соответствующее переупорядочению узлов сетки, согласованному с разбиением области расчета. Построение неполного треугольного разложения производится с отсечением по значению во внутренних узлах подобластей, а также по значению и по позициям на разделителях при крупнозернистом распараллеливании. Получены достаточные условия безотказности метода сопряженных градиентов с этим предобусловливателем. Рассматриваются способы применения MPI и MPI+OpenMP технологии при построении и обращении предобусловливателя. Проводится сравнение времени решения задач рассматриваемым методом с использованием MPI и гибридной MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse. Проводится сравнение скорости сходимости этого метода со скоростью сходимости других известных методов на примере решения тестовых задач.

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

Olga Yurievna Milyukova

Combination of numerical and structural approaches in the parallel first-order incomplete triangular decomposition preconditioning method

A parallel version of the method of incomplete triangular Cholesky decomposition of the first order of conjugate gradients is considered, which uses a reordering of the coefficient matrix corresponding to a reordering of grid nodes consistent with the partition of the calculation domain. The construction of an incomplete triangular decomposition is carried out with cutting off by value at the internal nodes of subdomains and by value and position on the separators during coarse-grained parallelization. Sufficient conditions for failure-free performance of the conjugate gradient method with this preconditioner are obtained. Methods of using MPI and MPI+OpenMP technology in constructing and inverting a preconditioner are considered. The time taken to solve problems by the method under consideration using MPI and the hybrid MPI+OpenMP technology is compared using the example of the model problem and a number of problems from the SuiteSparse collection. The convergence rate of this method is compared with the convergence rate of other known methods using the example of solving test problems

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. Необходимость решения СЛАУ (1.1) большого размера возникает, например, при решении задач гидродинамики, радиационной газовой динамики, теплопроводности, диффузии электрического поля и др. Решение задач с матрицами большого размера требует применения параллельных вычислений.

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 параллельный вариант неполного треугольного разложения Холецкого с отсечением по параметру первого порядка – предобусловливателя IC1(τ) (Incomplete Cholesky), рассмотренный ниже. 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. Один из первых алгоритмов, в котором за основу построения матрицы предобусловливания берется точный алгоритм треугольной факторизации, а на его определенных этапах вносится отсечение возникающих элементов матриц малых относительно порога, зависящего от τ, опубликован в работе [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 упорядочений узлов сетки, 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 [4-7]. Однако в этих работах используются предобусловливатели, при построении которых применяется отсечение только по позициям.

В работе [8] предлагается использовать упорядочение узлов сетки типа DDO для построения параллельного варианта метода стабилизированного неполного треугольного разложения второго порядка сопряженных градиентов (IC2S(τ)-CG) [9]. При этом производится отсечение по позициям для некоторых элементов матрицы предобусловливания.

В работах [10, 11, 12] предложен альтернативный подход, который позволяет преодолеть проблему распараллеливания рекурсивных вычислений при построении и обращении предобусловливателя. В них предложены параллелизуемые предобусловливатели, представляющие собой блочную версию предобусловливания неполного обратного треугольного разложения BIIC(p,q) [13] в сочетании с неполным треугольным разложением второго порядка IC2(τ) [9] – BIIC(p,q)-IC2(τ) [10], IC2S(τ) – BIIC(p,q)-IC2S(τ) [11] и первого порядка IC1(τ) – BIIC(p,q)-IC1(τ) [12]. Здесь p – число блоков, q – размер налегания.

В работе [14] предложен параллельный предобусловливатель систем линейных алгебраических уравнений для решения нелинейного уравнения лучистой теплопроводности при использовании только MPI. В его основе лежит параллельное многоуровневое неполное LU разложение разреженных матриц в сочетании с неполным блочным разложением обратной матрицы. Суть метода в рекурсивном построении иерархии матриц дополнения Шура с последовательным уменьшением числа процессов.

Проблеме использования высокоуровневого 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 способы применения MPI+OpenMP технологии построения и обращения предобусловливателя блочного Якоби в сочетании с неполным треугольным разложением Холецкого без заполнения IC(0) [17] (BJIC(0)), IC1(τ) (BJIC1(τ)) и IC2S(τ) (BJIC2S(τ)), основанные на переупорядочении узлов сетки типа DDO [8] внутри каждой подобласти, полученной для крупнозернистого распараллеливания. В работе [18] предложен способ использования MPI+OpenMP технологии для построения и обращения предобусловливателя BIIC(p,q)-IC1(τ), в котором для мелкозернистого распараллеливания применяется упорядочение узлов сетки типа DDO [8]. 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 способы применения MPI и MPI+OpenMP технологии для построения и обращения предобусловливателя IC(0) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей, в которых упорядочение

типа DDO [8] используется как для крупнозернистого, так и для мелкозернистого распараллеливания.

В формуле (1.1) предполагается, что матрица A уже переупорядочена, а вместо A_P стоит A ($A = A_P = P\widetilde{A}P^T$), где P – матрица перестановок, а $\widetilde{A}$ – матрица коэффициентов исходной задачи. В настоящей работе применяется переупорядочение, уменьшающее среднюю ширину ленты матрицы, предложенное в работе [20], являющиеся обобщением упорядочения [11]. 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 [20] и масштабирование уже выполнены.

В настоящей работе рассматривается новый метод предобусловливания PIC1(τ) – параллельный вариант предобусловливания IC1(τ) [21], сочетающий отсечение по значению как в методе IC1(τ) и по позициям для ряда строк матрицы, соответствующих узлам разделителей при упорядочении узлов сетки типа DDO [8] для крупнозернистого распараллеливания. Получены достаточные условия безотказности метода PIC1(τ)-CG. Рассматриваются способы применения MPI и MPI+OpenMP технологии для построения и обращения предобусловливателя PIC1(τ) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей, основанные на использовании переупорядочений узлов сетки. При этом OpenMP технологии применяются для большинства строк матрицы. В работе проводится сравнение времени решения тестовых задач методом PIC1(τ)-CG с использованием MPI и MPI+OpenMP технологии на примере модельной задачи и ряда задач из коллекции разреженных матриц SuiteSparse [22]. В работе также проводится сравнение скорости сходимости методов PIC1(τ)-CG, IC(0)-CG, BJIC1(τ)-CG и BJIC2S(τ)-CG, BIIC(p,q)-IC1(τ)-CG и BIIC(p,q)-IC2S(τ)-CG на примере решения ряда тестовых задач.

2. Агоритм построения предобусловливателя IC1(τ)

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 $v_{ij} = u_{ii}u_{ij} - e_{ij} = a_{ij} - \sum_{s=1}^{i-1} u_{si}u_{sj}$ ($j>i$). Здесь u_{ij} – элементы матрицы U, $j=i,\ldots,n$. Перед построением матрицы предобусловливания IC1(τ) необходимо

выполнить масштабирование матрицы A, после масштабирования $a_{ii}=1$. Для вычисления элементов матрицы U будем использовать алгоритм 1 [12].

Алгоритм 1

1. Инициализация вспомогательной диагональной матрицы:

for $i=1,\ldots, n$

$d_i := 1$

end for

Цикл по строкам A для вычисления строк U:

for $i=1,\ldots,n$

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

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

$v_j := a_{ij}$

end for

3. Сделать поправку к вектору v:

for $t=1,\ldots,i-1$

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

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

end for

end for

4. Вычисление u_{ii}:

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

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

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

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

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

end for

end for *(конец цикла по i)*

Здесь и далее d_i – элементы вспомогательной диагональной матрицы. При вычислении u_{ii} в п. 4 алгоритма 1 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.

3. Алгоритмы реализации построения и обращения предобусловливателя PIC1(τ) с использованием MPI и MPI+OpenMP технологии

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 типа DDO, предложенное в работе [8]. 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 и всех узлов разделителей первого и второго уровня из всех

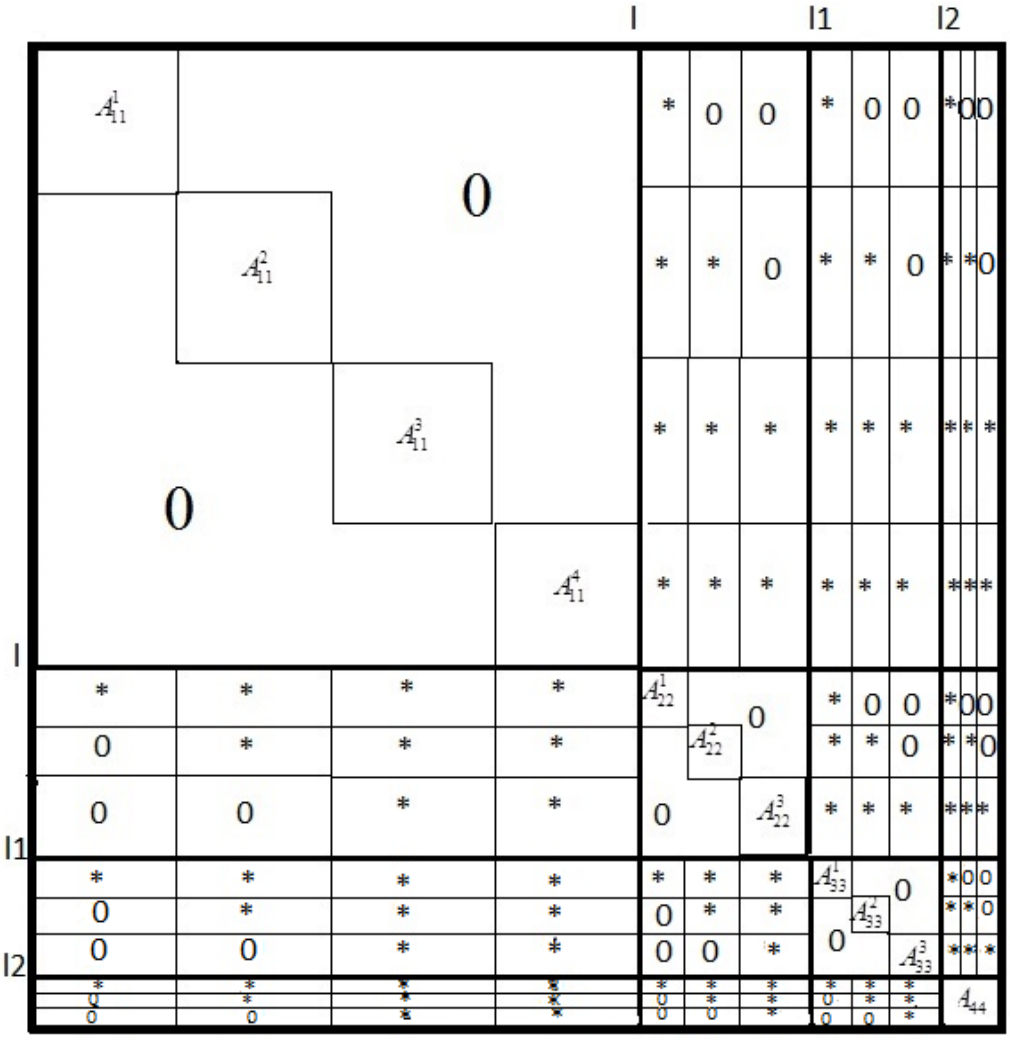

Рис. 1. Пример структуры разреженности матрицы *A*, полученной после перестановки столбцов и строк в результате переупорядочения

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.

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, …,n.

С помощью рассуждений от противного 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 1, 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 2, аналогичного алгоритму 2 из работы [8].

Алгоритм 2

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 цикло 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*. Заметим, что в пункте 1 алгоритма 2, так же как в алгоритме 2 в работе [8], происходит отсечение по позициям элементов матрицы *V*, что необходимо для лучшей параллелизуемости алгоритмов построения и обращения матрицы *U.* В работе [19] в этом нет необходимости из-за совпадения структур верхнетреугольного множителя матрицы предобусловливания IC(0) и верхнетреугольной части матрицы *A*.

Затем производится суммирование по всем процессорам вычисленных в них неокончательных значений d_j для *j*, соответствующих номерам узлов разделителей. Параллельная реализация вычисления элементов матрицы *V* и суммирования по всем процессорам вычисленных в них неокончательных значений d_j аналогична параллельной реализации этих этапов при построении предобусловливателя IC(0) [19] и параллельной реализации соответствующих этапов в работе [8] подробно описана, например, в работе [23].

Поскольку $V_{ij}=0$ и $a_{ij}=0$ для *i* и *j,* соответствующих узлам разделителей первого уровня, находящимся в разных подобластях, то, используя метод 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 с использованием алгоритма 3. Напомним, что перед началом вычисления матрицы *U* всем d_i было присвоено значение 0.

Алгоритм 3
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*-й строки *A* и *i*-й строки *V*:

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

$v_j := a_{ij} + V_{ij}$

end for

3. Сделать поправку к вектору *v*:

for $t=\bar{i}_1(s),...,i\text{-}1$

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 3 $\bar{i}_1(s)$, $\bar{i}_2(s)$ – 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 2. Кроме того, нужно произвести суммирование по всем процессорам вычисленных там значений Δd_j (изменений значений d_j, полученных на этапе 6 алгоритма 3) 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 3 добавляется еще одно слагаемое: $\overline{V}_{ij}$, где $\overline{V}_{ij}$ – элементы матрицы $\overline{V}$. Затем вычисляются элементы вспомогательной матрицы $\widetilde{V}$, 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, полученной для матрицы A, структура разреженности которой изображена на рис. 1.

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

Обращение матрицы предобусловливания PIC1(τ) осуществляется так же, как обращение матрицы предобусловливания IC(0) в работе [19]. Обращение матрицы предобусловливания состоит из двух этапов: $\hat{w}^k = U^{-T} r^{k-1}$ и $w^k = U^{-1}\hat{w}^k$, где k – номер итерации в алгоритме предобусловленного метода сопряженных градиентов (см., например, [24]). Перед выполнением первого и после выполнения второго этапов необходимо произвести переупорядочение элементов векторов r^{k-1} и, соответственно, w^k.

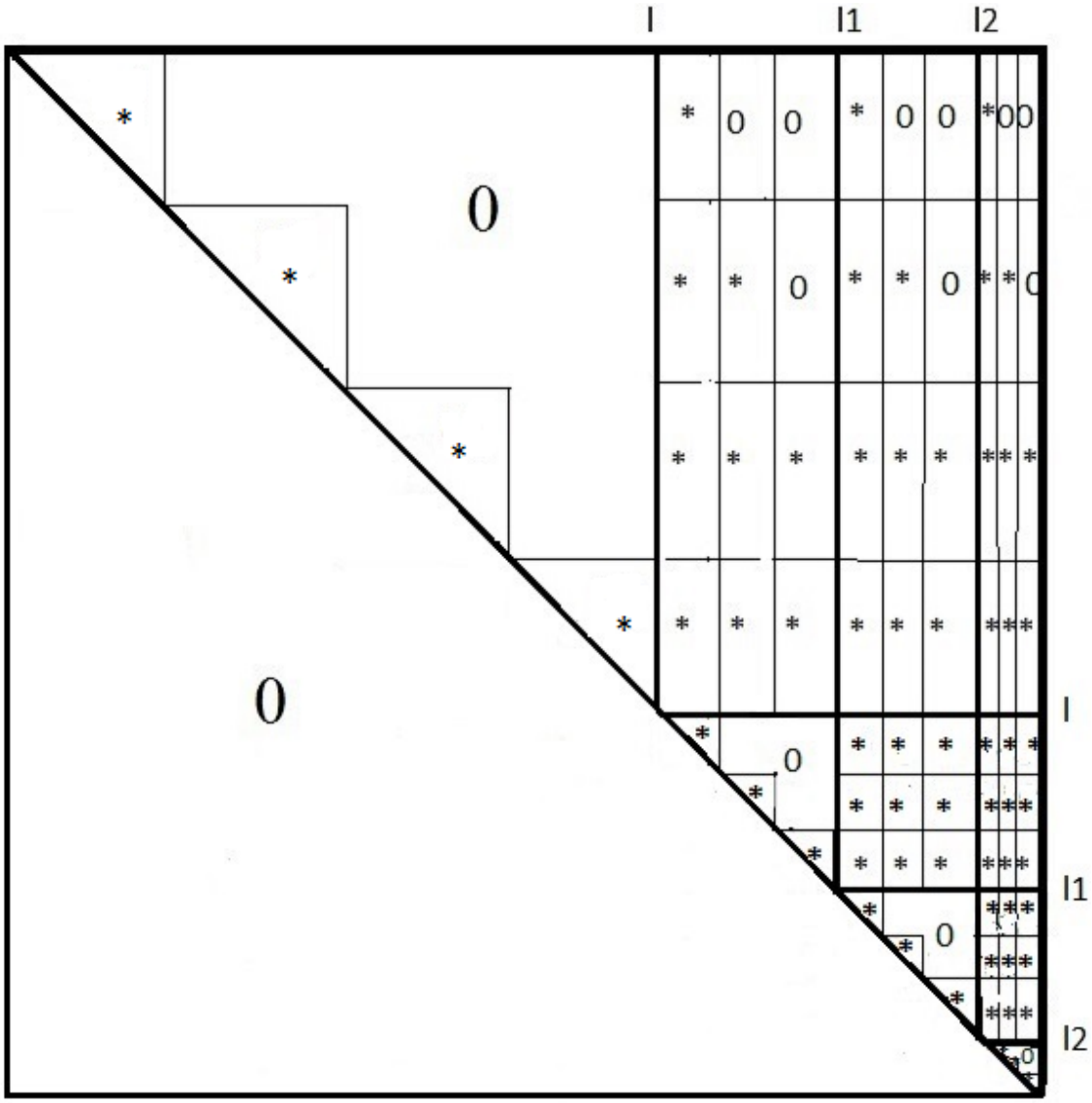

Рис. 2. Пример структуры разреженности матрицы U

При вычислении $\hat{w}^k = U^{-T} r^{k-1}$ будем использовать алгоритм обращения транспонированной матрицы [11], подробно описанный в работах [19, 21]. На каждом этапе обращения матрицы предобусловливания осуществляются необходимые пересылки [19, 21].

Способ применения MPI+OpenMP технологии при построении и обращении предобусловливателя PIC1(τ) является аналогичным способу 1 применения MPI+OpenMP технологии при построении и обращении предобусловливателя IC(0) [19] и подробно описан в работах [19, 21]. Сначала производится разбиение всей области расчета на p подобластей и переупорядочение всех узлов сетки типа DDO для крупнозернистого распараллеливания. При использовании MPI+OpenMP технологии в настоящей работе (и в работах [19, 21]) производится дополнительное переупорядочение только множества узлов сетки, являющихся внутренними при DDO упорядочении для использования MPI. При этом осуществляется разбиение

множества внутренних узлов каждой подобласти с номером *s=1, 2, …,p,* содержащее *il0(s*) узлов, на *m* частей, где *m* – предполагаемое число потоков, в порядке следования внутренних узлов подобласти на примерно одинаковые части. Затем применяется упорядочение типа DDO [8] множества внутренних узлов в каждой подобласти. При упорядочении для мелкозернистого распараллеливания в настоящей работе сначала идут первые $\bar{l}_S$ «внутренних» узлов из каждой «внутренней» подобласти, где $\bar{l}_S$ – минимальное число «внутренних» узлов в подобластях, полученных при разбиении множества внутренних узлов в подобласти с номером *s* (предполагается, что $\bar{l}_S \neq 0$). Затем, начиная с номера $\bar{l}_S m+1,$ идут оставшиеся «внутренние» узлы «внутренних» подобластей. Далее идут узлы «разделителей» первого, второго и третьего уровней. Здесь кавычки перед и после слов внутренние и разделители означает, что речь идет о внутренних узлах и узлах разделителей при упорядочении для мелкозернистого распараллеливания. После этого идут узлы разделителей первого, второго и третьего уровней в установленном ранее порядке.

Применение OpenMP технологии в каждом процессоре с номером *s* осуществляется при построении первых $M1 = m\bar{l}_S$ строк матрицы U_S, где U_S – часть матрицы *U*, хранящаяся в процессоре с номером *s* (см. работу [21]). Если число узлов во всех «внутренних» подобластях достаточно велико, то для подавляющего большинства строк матрицы *U* вычисление ее элементов происходит с использованием OpenMP технологии. Заметим, что при использовании OpenMP технологии не происходит дополнительного отсечения по позициям.

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. При этом OpenMP технологии не используется.

На первом этапе обращения матрицы предобусловливания сначала вычисляются элементы вектора $\hat{w}^k$, соответствующие внутренним узлам сетки, с помощью обращения нижнетреугольных матриц $L1_S$. При вычислении остальных элементов вектора $\hat{w}^k$ используется алгоритм обращения транспонированной матрицы. OpenMP технологии применяются при вычислении первых $M1$ элементов вектора $\hat{w}^k$, а также при вычислении последних $M1$ элементов вектора $w^k = U^{-1}\hat{w}^k$ (см. работу [21]).

4. Теоретическое исследование безотказности метода PIC1(τ)-CG

В настоящей работе под безотказностью метода (алгоритма) будем понимать корректность его выполнения, т.е. обеспечение выполнения неравенства $d_i > 0$ для всех значений i=1,2,…,n при вычислении предобусловливателя. Алгоритмы методов PIC1(τ)-CG, IC1(τ)-CG в общем случае нельзя назвать безотказными. Исследуем их безотказность в случае, если элементы матрицы $A = A^T > 0$ удовлетворяют условиям

$$a_{ii} > 0,\ a_{ij} \le 0, j \ne i,\ a_{ii} \ge \sum_{j \ne i} \left| a_{ij} \right|, 1 \le i \le n, 1 \le j \le n, \exists i = i_0 : a_{i_0 i_0} > \sum_{j \ne i_0} \left| a_{i_0 j} \right|. \quad (4.1)$$

Заметим, что при выполнении условий (4.1) неприводимая матрица A является М-матрицей [25]. Обоснование возможности построения точного треугольного разложения для М-матриц сделано в работе [17].

Для матриц $A = A^T > 0$ существует точное треугольное разложение $A = U_0^T U_0$. При выполнении условий (4.1) в случае точного треугольного разложения диагональные элементы верхнетреугольной матрицы U_0 положительны, а ее внедиагональные элементы неположительны.

Рассмотрим, как происходит точное треугольное разложение $A = U_0^T U_0$, где $A = A^T > 0$. Не ограничивая общности, предположим, что используется DDO. На каждом m-м шаге (m=2, …,n-1) имеем матрицу $A_m = \begin{bmatrix} \omega & a^T \\ a & \breve{A}_m \end{bmatrix}$, $A_1 = A$. Строка матрицы U_0 с номером m имеет вид $(0,0...,\sqrt{\omega}, a^T / \sqrt{\omega})$, где $\sqrt{\omega}$ стоит в позиции m, a – вектор длины n-m. Следующая активная подматрица $\hat{A}_{m+1}$ (дополнение Шура), которую следует представить в виде произведения нижнетреугольной и верхнетреугольной матриц, имеет вид

$$\hat{A}_{m+1} = \breve{A}_m - \frac{a\,a^T}{\omega}. \quad (4.2)$$

Можно доказать, что если для матрицы A выполнены условия (4.1), то для любого m элементы матрицы $\hat{A}_{m+1}$, определенной в (4.2), удовлетворяют условиям (4.1), и $\hat{A}_{m+1} = \hat{A}_{m+1}^T > 0$ [8].

Для облегчения понимания дальнейшего материала предположим (для простоты изложения), что элементы a^T при построении матрицы U

переупорядочены следующим образом: $a^T = (a_1^T, a_2^T)$, где $\left|(a_1^T)_j\right| \geq \tau\sqrt{\widetilde{\omega}}$, а $\left|(a_2^T)_j\right| < \tau\sqrt{\widetilde{\omega}}$. Тогда на каждом *m*-м шаге построения *U* имеем

$$A_m = \begin{bmatrix} \widetilde{\omega} & a_1^T & a_2^T \\ a_1 & \breve{A}_m & \\ a_2 & & \end{bmatrix}, \text{ где } \breve{A}_m = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

а следующая активная подматрица имеет вид

$$\hat{A}_{m+1} = \begin{bmatrix} A_{11} - a_1 a_1^T / \widetilde{\omega} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

Строка матрицы U с номером *m* имеет вид $(0,0...,\sqrt{\widetilde{\omega}}, a_1^T/\sqrt{\widetilde{\omega}}, 0,...,0)$, где $\sqrt{\widetilde{\omega}}$ стоит в позиции *m*, a_1 – вектор длины, не превосходящей *n-m*. Из алгоритма 1 видно, что при выполнении условий (4.1) $u_{ij} \leq 0, j > i$, причем по модулю они не могут превосходить соответствующие элементы матрицы U_0. Следовательно, окончательно посчитанные значения d_i при построении IC1(τ) разложении не могут быть меньше квадратов соответствующих диагональных элементов матрицы U_0, и, значит, они положительны.

Все диагональные элементы матрицы $\hat{A}_{m+1}$ (при исходной нумерации) не уменьшились по сравнению с диагональными элементами матрицы $\hat{A}_{m+1}$ при точном разложении, т.к. $(a_1)_j \leq 0$ и $a_j \leq 0$(при точном разложении) для любого *j*. Используя метод математической индукции и сравнение элементов матриц $\hat{A}_{m+1}$ при построении IC1(τ) и $\hat{A}_{m+1} = \breve{A}_m - \frac{a\,a^T}{\omega}$ при точном разложении, можно доказать, что для любого *m* для матрицы $\hat{A}_{m+1}$ при построении предобусловливателя IC1(τ) выполнены условия (4.1), $\hat{A}_{m+1} = \hat{A}_{m+1}^T > 0$, дальнейшее разложение возможно.

Итак, при вычислении предобусловливателя в методе IC1(τ)-CG во всех строках и при вычислении предобусловливателя в методе PIC1(τ)-CG в строках, соответствующих внутренним узлам подобластей, алгоритм будет безотказным, причем $u_{ii} > 0,\ d_i > 0, u_{ij} \leq 0, j \neq i$.

При вычислении предобусловливателя в методе PIC1(τ)-CG в строках, соответствующих узлам разделителей, для облегчения понимания представим

a^T следующим образом: $a^T = (\tilde{a}_1{}^T, \tilde{a}_2{}^T, \tilde{a}_3{}^T)$, где $(\tilde{a}_3{}^T)_j$ соответствуют тем элементам матрицы *U*, которые отбрасываются по позициям, а $\left|(\tilde{a}_1{}^T)_j\right| \ge \tau\sqrt{\tilde{\omega}}$, $\left|(\tilde{a}_2{}^T)_j\right| < \tau\sqrt{\tilde{\omega}}$. Тогда на каждом *m*-м шаге на разделителях первого уровня

$$A_m = \begin{bmatrix} \tilde{\omega} & \tilde{a}_1^T & \tilde{a}_2^T & \tilde{a}_3^T \\ \tilde{a}_1 & & & \\ \tilde{a}_2 & & \breve{A}_m & \\ \tilde{a}_3 & & & \end{bmatrix}, \text{ где } \breve{A}_m = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}, \qquad (4.3)$$

а следующая активная подматрица имеет вид

$$\hat{A}_{m+1} = \begin{bmatrix} A_{11} - \tilde{a}_1\tilde{a}_1^T / \tilde{\omega} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}. \qquad (4.4)$$

При вычислении первой строки разделителя первого уровня в каждой подобласти все диагональные элементы матрицы $\hat{A}_{m+1}$, определенной в (4.4), не уменьшились по сравнению с диагональными элементами соответствующей активной подматрицы при точном разложении $\hat{A}_{m+1} = \breve{A}_m - \dfrac{a\,a^T}{\omega}$ (и при IC1(τ) разложении), т.к. $(\tilde{a}_1)_j \le 0$ и $a_j \le 0$ (при точном разложении). Следовательно, в первой строке разделителя первого уровня в каждой подобласти $u_{ii} > 0$, $d_i > 0$. Все внедиагональные элементы A_m, определенной в (4.3), не положительны, так как до этого применялось IC1(τ) разложение. Следовательно, все внедиагональные элементы матрицы $\hat{A}_{m+1}$, определенной в (4.4), не положительны, причем по модулю они не превосходят соответствующие элементы при точном разложении, а диагональные элементы положительные и не меньше соответствующих элементов при точном разложении. Следовательно, для элементов матрицы $\hat{A}_{m+1}$, определенной в (4.4), выполнены условия (4.1), в том числе выполнено условие диагонального преобладания, поэтому $\hat{A}_{m+1} = \hat{A}_{m+1}^T > 0$ и дальнейшее разложение возможно. Кроме того, $u_{ii} > 0, d_i > 0$, $u_{ij} \le 0, j \ne i$. Это верно для первой строки разделителя в каждой подобласти.

Используя метод математической индукции, можно доказать, что в процессе вычисления на разделителях первого уровня в каждой подобласти для

любого m для матрицы $\hat{A}_{m+1}$, определенной в (4.4), выполнены условия (4.1), $\hat{A}_{m+1} = \hat{A}^T_{m+1} > 0$, для каждого m диагональные элементы матрицы $\hat{A}_{m+1}$ не убывают по сравнению с диагональными элементами активной подматрицы на соответствующем шаге при точном разложении и при IC1(τ) разложении; $u_{ii} > 0, d_i > 0$, $u_{ij} \le 0, j \ne i$ для каждой строки i, соответствующей узлу сетки из множества узлов разделителей первого уровня.

Аналогично доказывается, что при расчете на разделителях второго и третьего уровня для любого m $\hat{A}_{m+1} = \hat{A}^T_{m+1} > 0$, неравенства $u_{ii} > 0, d_i > 0$, $u_{ij} \le 0, j \ne i$ верны для всех строк разделителей второго и третьего уровня, для любого m дальнейшее разложение возможно. Итак, доказана

Теорема 1. Пусть элементы матрицы $A = A^T > 0$ удовлетворяют условиям (4.1). Тогда методы IC1(τ)-CG и PIC1(τ)-CG являются безотказными при любом τ>0.

Замечание 1. Можно доказать, что если элементы неприводимой матрицы $A = A^T > 0$ удовлетворяют условиям (4.1), то методы BJIC1((τ)-CG, BIIC(*p,q*)-IC1(τ)-CG являются безотказными при любом τ>0.

Заметим, что в теореме 1 приведены достаточные условия безотказности методов IC1(τ)-CG и PIC1(τ)-CG.

5. Результаты расчетов

Программы, реализующие применение метода PIC1(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. В составе кластера К60 находятся 86 вычислительных узлов, содержащих 2408 ядер (по 28 ядер на узел). Каждый узел представляет собой двухпроцессорный сервер с процессорами Intel Xeon E5-2690 v4. Суммарная пиковая производительность К-60 81,9 терафлопс. Тестирование и сравнение методов производилось с помощью решения модельной задачи – разностной задачи Дирихле для уравнения Пуассона в единичном квадрате на равномерной ортогональной сетке при различных значениях n. Использовалась стандартная 5-точечная аппроксимация лапласиана, при n=1048576 имя матрицы **5_1048576**. Для тестирования предложенного параллельного метода на небольшом числе процессоров использовались также некоторые матрицы из коллекции разреженных матриц SuiteSparse [22]. Перечислим имена используемых тестовых матриц и укажем источник их происхождения:

apache2 – трехмерная конечно-разностная схема;

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 [26], 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	$nz_{\min}$	$nz_{\max}$	$Cond(A_0)$
apache2	715176	4817870	2	0	4	8	0.12 E+7
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 [20], и масштабирования исходной матрицы, с единичной правой частью ($b_i \equiv 1$), начальное приближение $x_0 \equiv 0$. Счет продолжался до выполнения условия (1.2), где $\varepsilon = 10^{-8}$.

В настоящей работе, в отличие от работы [21], расчеты с использованием MPI и MPI+OpenMP технологии проводились при участии одного процессора в каждом вычислительном узле, что позволило немного ускорить вычисления. Заметим, что в работе [11] показано, что оптимальное ускорение счета при решении СЛАУ (1.1) методом BIIC(p,q)-IC2S(τ)-CG с использованием крупнозернистого распараллеливания наблюдалось при участии в расчетах одного вычислительного ядра на узел МВС-100К.

В таблицах 2–5 приведены числа итераций и время счета при решении методом PIC1(τ)-CG тестовых задач 1–4 с соответствующими именами матриц при применении MPI и MPI+OpenMP технологии. Под временем счета в таблицах 2–5 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 2–5 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, приведены курсивом коэффициенты μ ускорения счета задач благодаря использованию OpenMP технологии.

Таблица 2

Числа итераций и время счета методом PIC1(τ)-CG задачи с матрицей **5_1048576** на p процессорах без применения и с применением OpenMP технологии

p	3	4	5	6
MPI	310, 4.95	314, 3.74	329, 3.7	327, 2.67
MPI+OpenMP	336, 2.20(6)	345, 1.79(6)	357, 2.36(6)	351, 1.55(6)
μ	*2.25*	*2.09*	*1.56*	*1.72*

Таблица 3

Числа итераций и время счета методом PIC1(τ)-CG задачи с матрицей **ecology2** на p процессорах без применения и с применением OpenMP технологии

p	3	4	5	6
MPI	565, 9.19	577, 7.30	583, 5.11	595, 5.61
MPI+OpenMP	609, 3.63(8)	603, 2.72(6)	630, 2.50(6)	624, 2.84(4)
μ	*2.53*	*2.68*	*2.04*	*1.97*

Таблица 4

Числа итераций и время счета методом PIC1(τ)-CG задачи с матрицей **apache2** на p процессорах без применения и с применением OpenMP технологии

p	2	3	4	5	6
MPI	348, 5.11	387, 3.60	362, 3.09	366, 3.56	387, 3.75
MPI+OpenMP	369, 2.51(8)	394, 1.78(8)	374, 1.40(10)	370, 2.17(4)	394, 2.60(3)
μ	*2.03*	*2.02*	*2.21*	*1.64*	*1.44*

Таблица 5

Числа итераций и время счета методом PIC1(τ)-CG задачи с матрицей **tmt_sym** на p процессорах без применения и с применением OpenMP технологии

p	2	3	4	5	6
MPI	548, 9.53	544, 6.31	547, 5.12	559, 5.43	551, 4.37
MPI+OpenMP	587,3.36(12)	582, 2.24(8)	578, 2.19(6)	592, 2.52(8)	583, 3.20(4)
μ	*2.83*	*2.81*	*2.34*	*2.15*	*1.36*

Заметим, что применение OpenMP технологии при использовании MPI+OpenMP технологии не позволило ускорить вычисление матрицы

предобусловливания в методе PIC1(τ)-CG при решении тестовых задач 1–4, так же как и в методе IC(0)-CG. Ускорение решения СЛАУ происходило исключительно благодаря ускорению счета итерационного процесса, причем время счета итерационного процесса составляло основную часть времени решения СЛАУ с этими матрицами.

На рисунках 3, 4 приведены графики зависимости времени счета от числа процессоров p (в логарифмическом масштабе) для двух тестовых задач методом PIC1(τ)-CG с использованием MPI (сплошные линии) и с использованием MPI+OpenMP технологии (штриховые линии). Графики зависимости времени счета от числа процессоров для тестовых задач с матрицами **ecology2** и **tmt_sym** при использовании MPI и MPI+OpenMP технологии имеют аналогичный вид. Как видно из таблиц 2–5 и рис. 3, 4, применение MPI+OpenMP технологии при решении тестовых задач 1–4 позволило значительно ускорить их решение по сравнению с применением только MPI на небольшом числе процессоров.

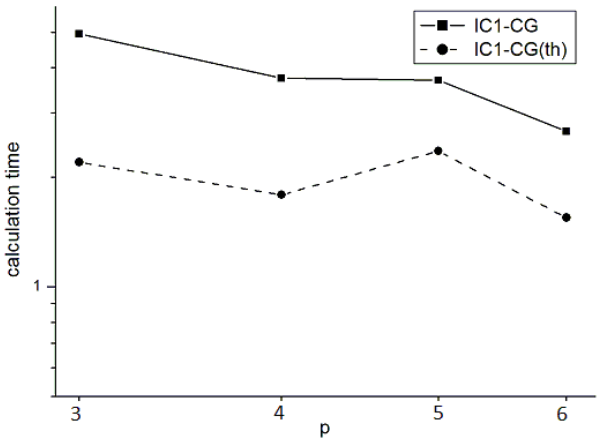

Рис. 3. Время счета задачи с матрицей **5_1048576** методом PIC1(τ)-CG

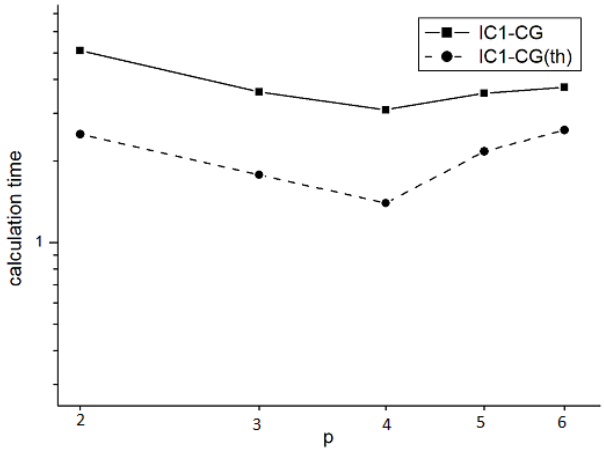

Рис. 4. Время счета задачи с матрицей **apache2** методом PIC1(τ)-CG

На рисунках 5, 6 приведены графики зависимости ускорения счета от числа процессоров при решении четырех тестовых задач с использованием только MPI и MPI+OpenMP технологии по сравнению со временем решением этих задач на 3 (рис. 5) или 2 (рис. 6) 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 методом [20] и разбиения множеств внутренних узлов подобластей при использовании OpenMP технологии.

Как видно из рис. 5, 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 рис. 6, применение

MPI+OpenMP технологии позволяет решать задачу с матрицей **apache2** 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 $p<6$.

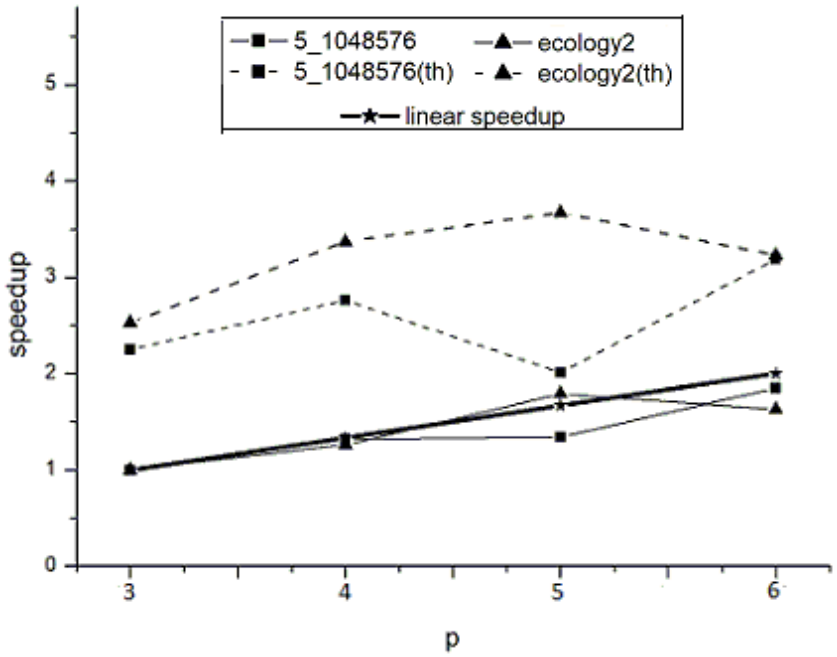

Рис. 5. Ускорение счета задач с матрицами **5_1048576**, **ecology2** при использовании MPI и MPI+OpenMP

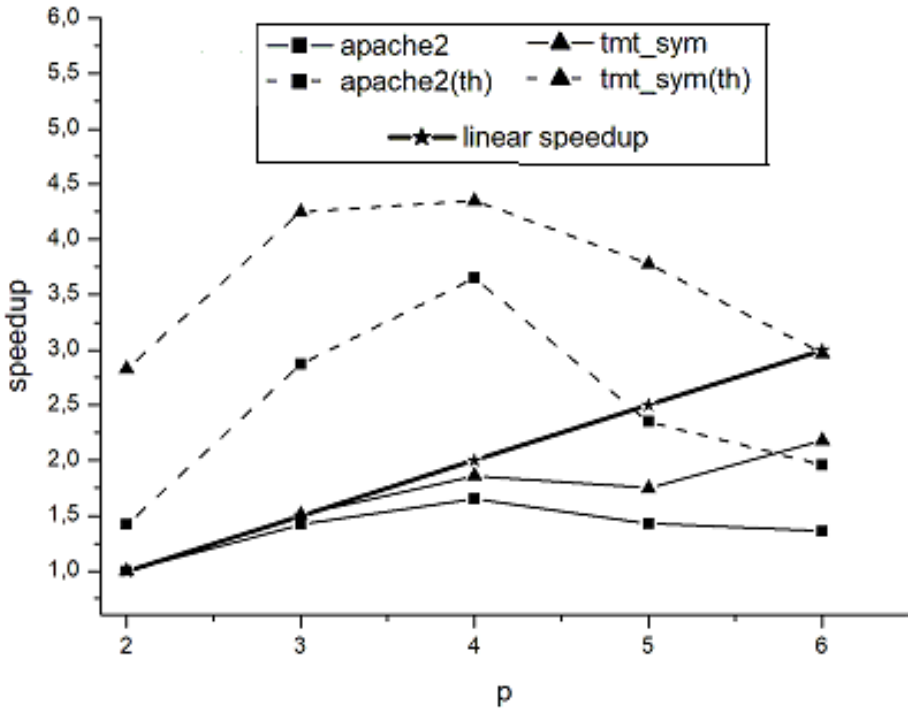

Рис. 6. Ускорение счета задач с матрицами **apache2**, **tmt_sym** при использовании MPI и MPI+OpenMP

Эффективность использования MPI+OpenMP технологии по сравнению с использованием только MPI падает с ростом числа процессоров, так как уменьшается число неизвестных в СЛАУ, которые после разбиения области расчета вычисляются в одном процессоре с применением OpenMP технологии. Начиная с некоторого значения $p>6$ использование OpenMP технологии при решении этих тестовых задач становится нецелесообразным.

Заметим, что в приведенных выше результатах для расчетов использовались матрицы относительно небольшого размера. В условиях высокой скорости расчета (увеличенной недавно) и относительно медленного обмена информацией между процессорами эффективность MPI распараллеливания счета в случае матриц небольшого размера не столь высока, как, например, в работах [15, 16, 18, 19]. Поэтому были проведены расчеты модельной задачи при возрастающих размерах разностной сетки. Результаты расчетов приведены в таблицах 6, 7, где первое число соответствует числу итераций, а второе – времени (в секундах) счета. Отсутствие заполнения одной клетки в таблице 7 объясняется недостатком памяти при решении соответствующей задачи на небольшом числе процессоров. Из таблиц 6, 7 видно, что ускорение счета благодаря использованию только MPI при n=2016400, 3062500, 3900625, 5062500, 6002500 происходило до 14 процессоров включительно, при n=7952400 до 16 процессоров. При этом время счета убывало не всегда монотонно.

Таблица 6

Числа итераций и время счета методом PIC1(τ)-CG модельной задачи при различных значениях n с использованием только MPI (часть 1)

$n\backslash p$	4	6	8	10	12	14
2016400	436, 11.76	448, 9.36	457, 6.34	465, 6.57	468, 7.62	451, 6.30
3062500	541, 20.53	555, 18.86	567, 10.44	567, 10.44	587, 10.85	579, 9.11
3900625	606, 31.42	620, 25.13	627, 17.06	629, 14.25	630, 12.52	648, 11.95

Таблица 7

Числа итераций и время счета методом PIC1(τ)-CG модельной задачи при различных значениях n с использованием только MPI (часть 2)

$n\backslash p$	6	8	10	12	14	15
5062500	723, 36.41	784, 26.56	718, 22.88	771, 20.94	779, 16.9	767, 19.44
6002500	838, 42.87	786, 34.74	782, 28.66	750, 26.99	798, 19.46	824, 26.36
7952400		979, 69.68	979, 45.39	928,39.58	918, 37.74	923, 36.15

На рисунке 7 приведены графики зависимости времени решения модельной задачи от числа процессоров при различных значениях n.

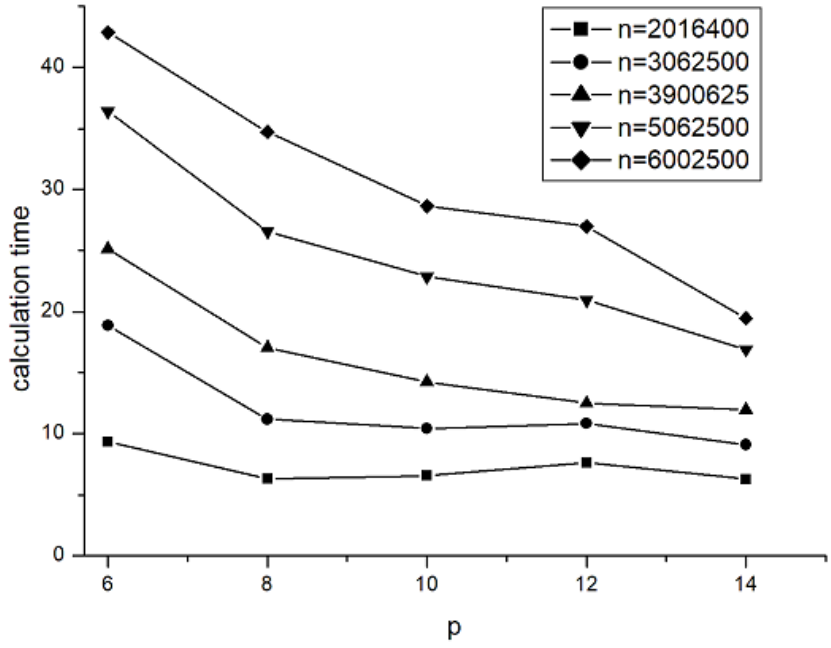

Рис. 7. Время решения модельной задачи при различных значениях n методом PIC1(τ)-CG с использованием только MPI

Проведем сравнение скорости сходимости различных методов. Числа итераций при решении тестовых задач 1–3 методом PIC1(τ)-CG при τ=0.01 с применением MPI и MPI+OpenMP технологии в несколько раз меньше, чем при их решении методом IC(0)-CG (см. работу [19]). Как показывают расчеты, числа итераций при решении этих задач методом PIC1(τ)-CG с применением MPI и MPI+OpenMP технологии при τ=0.01 были меньше чисел итераций при их решении методами BJIC1(τ)-CG, BJIC2S(τ)-CG, BIIC(p,1)-IC1(τ)-CG, BIIC(p,1)-IC2S(τ)-CG (см. работы [16, 18, 12]).

В таблице 8 в качестве примера приведены числа итераций, полученные при решении задачи с матрицей **5_1048576** методами PIC1(τ)-CG, IC(0)-CG, а также методами BJIC1(τ)-CG, BJIC2S(τ)-CG, BIIC(p,q)- IC1(τ)-CG, BIIC(p,q)-IC2S(τ)-CG при q=1,2, τ=0.01 с использованием только MPI. В скобках приведены значения коэффициента возрастания числа итераций по сравнению с числом итераций при счете на 4 процессорах. Из таблицы 8 видно, что числа итераций при решении этой задачи на небольшом и умеренном числе процессоров методом PIC1(τ)-CG в несколько раз меньше, чем при их решении методом IC(0)-CG; меньше, чем при ее решении методами BJIC1(τ)-CG, BJIC2(τ)-CG, BIIC(p,q)-IC1(τ)-CG (q=1, 2), BIIC(p,1)-IC2S(τ)-CG; сопоставимы с числами итераций при ее решении методом BIIC(p,2)-IC2S(τ)-CG, причем при p=32, 64 даже немного больше. Коэффициент возрастания числа итераций по сравнению с числом итераций на 4 процессорах при решении модельной задачи с матрицей **5_1048576** методом PIC1(τ)-CG меньше, чем при ее решении методами BJIC1(τ)-CG, BJIC2S(τ)-CG; сопоставим с ним при использовании BIIC(p,1)-IC1(τ)-CG и, как правило, при

использовании BIIC(p,1)-IC2S(τ)-CG; как правило, немного больше, чем при использовании BIIC(p,2)-IC1(τ)-CG, BIIC(p,2)-IC2S(τ)-CG.

Таблица 8

Числа итераций при решении задачи с матрицей **5_1048576** разными методами на p процессорах с применением только MPI

p	4	8	16	32	64
IC1(τ)-CG	314	336(1.07)	344(1.1)	391(1.24)	397(1.26)
IC(0)-CG	1022	1034(1.01)	1122(1.1)	1138(1.11)	1146(1.12)
BJIC1(τ)-CG	446	490(1.1)	520(1.17)	578(1.3)	634(1.42)
BIIC(p,1)-IC1(τ)-CG	401	435(1.08)	444(1.1)	484(1.21)	507(1.26)
BIIC(p,2)-IC1(τ)-CG	373	389(1.04)	405(1.09)	434(1.16)	452(1.21)
BJIC2S(τ)-CG	376	428(1.14)	468(1.24)	522(1.39)	583(1.55)
BIIC(p,1)-IC2S(τ)-CG	343	375(1.09)	386(1.12)	428(1.25)	457(1.33)
BIIC(p,2)-IC2S(τ)-CG	321	347(1.08)	351(1.09)	381(1.19)	392(1.22)

Однако количество пересылок при параллельной реализации метода PIC1(τ)-CG больше, чем в методах BJIC1(τ)-CG, BJIC2S(τ)-CG, BIIC(p,1)-IC1(τ)-CG и BIIC(p,1)-IC2S(τ)-CG (прежде всего, при обращении матрицы предобусловливания), что замедляет решение задачи методом PIC1(τ)-CG.

6. Заключение

В работе рассмотрен новый метод предобусловливания PIC1(τ), сочетающий отсечение по значению, как в методе IC1(τ), и по позициям для ряда строк матрицы, соответствующих узлам разделителей при упорядочении типа DDO для крупнозернистого распараллеливания. Получены достаточные условия безотказности метода PIC1(τ)-CG. Предложены способы применения MPI и MPI+OpenMP технологии для построения и обращения предобусловливателя PIC1(τ) при решении СЛАУ (1.1) с произвольной симметричной положительно определенной матрицей. При этом OpenMP технологии применяются для большинства строк матрицы.

При решении тестовых задач методом PIC1(τ)-CG наблюдался небольшой рост числа итераций с ростом числа процессоров для их небольшого или умеренного количества. С помощью расчетов тестовых задач на небольшом числе процессоров показано, что использование OpenMP технологии позволяет значительно ускорить вычисления.

Числа итераций при решении методом PIC1(τ)-CG с применением MPI тестовых задач 1–3 на небольшом числе процессоров и тестовой задачи 1 (модельной задачи при n=1048576) на умеренном числе процессоров были значительно меньше чисел итераций при их решении методом IC(0)-CG,

меньше чисел итераций при их решении методами BJIC1(τ)-CG, BJIC2S(τ)-CG, BIIC(p,1)-IC1(τ)-CG, BIIC(p,1)-IC2S(τ)-CG при выбранном фиксированном значении τ. Числа итераций при решении модельной задачи при n=1048576 методом PIC1(τ)-CG с применением MPI на небольшом и умеренном числе процессоров были меньше чисел итераций при ее решении методом BIIC(p,2)-IC1(τ)-CG и сопоставимы с числами итераций при ее решении методом BIIC(p,2)-IC2S(τ)-CG при выбранном фиксированном значении 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. 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.
7. Milyukova O.Yu. Parallel approximate factorization method for solving discrete elliptic equations // Parallel computing. 2001. V. 27. P. 1365-1379.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. Чеботарь С.В. Параллельный предобусловливатель систем линейных алгебраических уравнений для решения нелинейного уравнения лучистой теплопроводности в методике «Корона» // ВАНТ. Серия Математическое моделирование физических процессов. 2023. Вып. 4. С. 34-43.
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. https://doi.org/10.20948/mm-2022-01-10.
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. мет. и програм. 2022. Т. 23. Вып. 3. С. 191-206. doi 10.26089/NumMet.v23r312.
19. Милюкова О.Ю. Некоторые способы параллельной реализации метода сопряженных градиентов с неявным факторизованным предобусловливателем // Математическое моделирование. 2024. Т. 36, № 2, С. 174-196. https://doi.org/10.20948/mm-2024-02-10.
20. 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.
21. 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.
22. Davis T., Hu Y.F. University of Florida sparse matrix collection // ACM Trans. On Math.~Software. 2011. V. 38. № 1. http://www.cise.ufl.edu/research/sparse/matrices.

23. 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.
24. Axelsson O. Iterative solution methods // New York: Cambridge Univ. Press. 1994.
25. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем // М.: Мир. 1991. 365 с.
26. 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