Записки научных
семинаров ПОМИ
Том 405, 2012 г.

Р. Р. Ахунов, С. П. Куксенко, В. К. Салов, Т. Р. Газизов

УСОВЕРШЕНСТВОВАНИЕ АЛГОРИТМА ILU(0)-РАЗЛОЖЕНИЯ, ИСПОЛЬЗУЮЩЕГО РАЗРЕЖЕННЫЙ СТРОЧНЫЙ ФОРМАТ

§1. Введение

При решении СЛАУ с плотной матрицей минимальные вычислительные затраты можно получить за счет использования итерационного метода с предобусловливанием [1]. Однако для предобусловливания необходимо хранить элементы дополнительной матрицы, что приводит к увеличению требуемой памяти компьютера. Между тем, разреженность данной матрицы позволяет экономить требуемую память компьютера и, кроме того, получать ускорение решения СЛАУ [2], если для хранения предобусловливателя используется специализированный разреженный формат. Так, при использовании разреженного строчного формата для хранения элементов матрицы предобусловливания на основе ILU(0)-разложения показано уменьшение памяти до 9 раз, а времени – до 2,5 раз [3].

В сущности, возможность этого появилась благодаря системному эффекту от использования ILU(0)-разложения совместно со сжатием разреженной матрицы. Примечательно, что в практических задачах часто важнее уменьшение не памяти, а времени. При этом, по сути, возникает ситуация, когда побочный эффект (уменьшение времени) становится более значимым, чем основной (уменьшение памяти). Поэтому усиление этого побочного эффекта весьма актуально. Первые шаги в этом направлении сделаны в работе [3], однако все ресурсы ещё не исчерпаны.

Ключевые слова: система линейных алгебраических уравнений, разреженная матрица, итерационные методы, предобусловливание, предфильтрация.

Работа выполнена в порядке реализации постановления No. 218 Правительства РФ от 09.04.2010 г. “О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства” и договора No. 13.G25.31.0017 от 07.09.2010 между ОАО “ИСС” им. акад. М. Ф. Решетнева” и Минобрнауки РФ.

Цель данной работы – усовершенствование алгоритма ILU(0)-разложения [3] для дополнительного ускорения его работы.

Статья построена следующим образом. Сначала для полноты изложения приведено краткое описание предобусловливания и алгоритма ILU(0)-разложения, основанного на ikj-версии LU-разложения, а также описание разреженного строчного формата, выбранного для хранения матрицы предобусловливания. Затем приведены описание и анализ предложенных усовершенствований алгоритма ILU(0)-разложения, а также полученные алгоритмы. Приведены результаты вычислительного эксперимента на примере вычисления электрической ёмкости двух полосок с помощью метода моментов. В заключении сделаны выводы и обозначены направления дальнейшей работы.

Итерационный метод BiCGStab [4], используемый совместно с неявным предобусловливанием, хорошо зарекомендовал себя при решении СЛАУ с плотной матрицей [5]. При неявном предобусловливании исходную СЛАУ можно представить в виде $\mathbf{MAx} = \mathbf{Mb}$. Наиболее работоспособные методы построения неявного предобусловливания основаны на LU-разложении [5]. Однако при его использовании происходит неконтролируемое появление новых ненулевых элементов, что при использовании разреженных форматов недопустимо, так как данные форматы могут стать неэффективными. Поэтому был выбран метод ILU(0)-разложения, исключающий добавление новых элементов в матрицу [6]. Далее приведен алгоритм данного метода (алгоритм 1), основанный на ikj-версии LU-разложения.

Алгоритм 1 ILU(0)-разложение

1 Для $i = 2, \ldots, N$
2 Для $k = 1,\ldots, i-1$
3 Если $a^S_{ik} \neq 0$
4 $a^S_{ik} = a^S_{ik} \,/\, a^S_{kk}$
5 Для $j = k+1, \ldots, N$
6 Если $a^S_{ij} \neq 0$
7 $a^S_{ij} = a^S_{ij} - a^S_{ik} \times a^S_{kj}$
8 Увеличить j
9 Увеличить k
10 Увеличить i

В алгоритме 1 a^S – элемент дополнительной матрицы $\mathbf{A}_S$, которая получается из матрицы $\mathbf{A}$ после предфильтрации, т.е. обнуления малозначимых элементов матрицы. При предфильтрации использовался подход, основанный на нормировке всей матрицы с помощью максимального элемента [7]. В результате выполнения алгоритма 1 получается матрица предобусловливания $\mathbf{M}$, хранящаяся на месте матрицы $\mathbf{A}_S$.

Разреженный строчный формат предъявляет минимальные требования к памяти и, в то же время, оказывается очень удобным для выполнения нескольких важных операций над разреженными матрицами [2]: сложения, умножения, перестановок строк и столбцов, транспонирования, решения СЛАУ с разреженными матрицами [6]. Значения ненулевых элементов матрицы и соответствующие индексы столбцов хранятся в этом формате по строкам в двух массивах. Используется также массив указателей на ненулевые элементы, с которых начинается очередная строка.

§2. Усовершенствование алгоритма ILU(0)-разложения

Алгоритм 1 был переработан с целью его использования в сочетании с разреженным строчным форматом [3]. Также с целью ускорения работы алгоритма был добавлен дополнительный массив, в котором хранятся указатели на диагональные элементы матрицы. В результате был получен алгоритм 2 [3].

В алгоритме 2 **aelem** – вектор значений ненулевых элементов; **iptr** – вектор номеров столбцов ненулевых элементов; **jptr** – вектор указателей на первые элементы в строках; **diag** – вектор указателей на диагональные элементы в матрице.

Как показывает анализ, алгоритм 2 обладает недостатком: тратится время на поиски ненулевых элементов матрицы в строках 3, 7. Данные поиски необходимы, так как при использовании разреженного строчного формата нет прямой ссылки на первый ненулевой элемент в цикле j (строка 5 в алгоритме 1). Таким образом, если уйти от упомянутого поиска, то можно сэкономить затрачиваемое машинное время.

Если обратиться к исходному алгоритму 1 и расположить элементы матрицы, используемые в алгоритме, на рисунке, то на примере матрицы порядка N=4 получится рис. 1. Изображения матрицы поделены по шагам таким образом, что один шаг соответствует одному

Алгоритм 2 ILU(0)-разложение с использованием разреженного строчного формата

```
1 Для i = 2, ..., N
2   Для k = 1, ..., i − 1
3     Найти s − номер элемента a^S_ik в векторе aelem
4     Если aelem(s) ≠ 0
5       aelem(s) = aelem(s) / aelem(diag(k))
6     y1 = s + 1
7     Найти t2 − номер элемента a^S_kj в векторе aelem
8     y2 = t2
9     Пока pr = Истина Продолжать
10      Если jptr(y1) = jptr(y2)
11        aelem(y1) = aelem(y1) − aelem(t1) × aelem(y2)
12        Увеличить y1 и y2
13      Если jptr(y1) > jptr(y2)
14        Увеличить y2
15      Если jptr(y1) < jptr(y2)
16        Увеличить y1
17      Если iptr(k + 1) < y2 или iptr(i+1) ¡ y2
18        pr = Ложь
19  Увеличить k
20 Увеличить i
```

значению переменных i и k. Видно, что элементы a_{ik} участвуют в цикле, который всегда начинается с первого элемента строки, и перемещаются по строкам до последнего элемента в нижнетреугольной матрице. Причем там, где данный элемент равен нулю, соответствующий шаг ik пропускается. Данную цикличность можно использовать в алгоритме с разреженным строчным форматом, поскольку движение элемента происходит по строкам. Причем в этом случае нет необходимости искать элемент a_{ik} в каждом цикле (строка 3 в алгоритме 2).

В алгоритме 2 имеется еще одна особенность. Так, начальный элемент цикла kj всегда находится после соответствующего диагонального элемента. Этот факт позволяет исключить поиск элемента a_{kj} (строка 7 алгоритма 2).

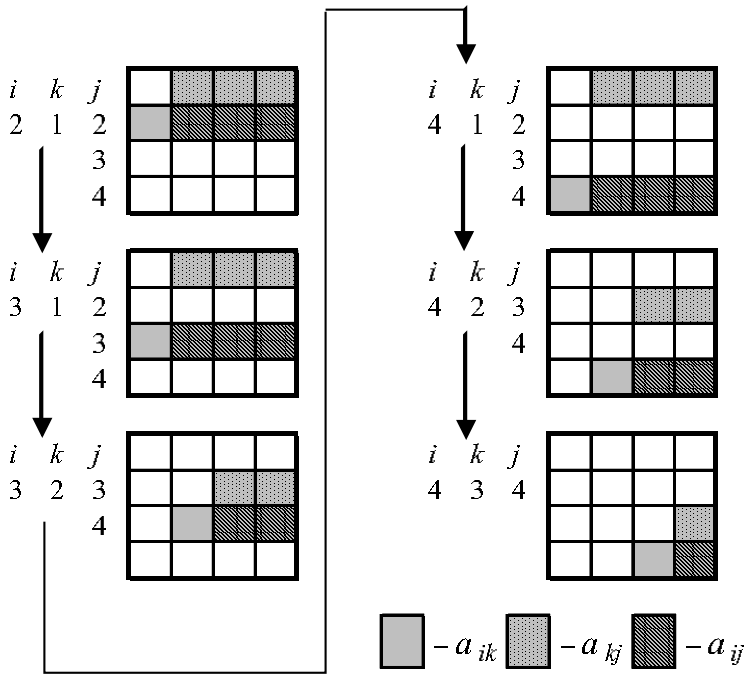

Рис. 1. Схематическое расположение элементов, используемых за все циклы ILU(0)-разложения

При использовании вышеперечисленных особенностей алгоритма 1 поиски ненулевых элементов в алгоритме 2 исключаются. После изменений получен алгоритм 3.

Анализ показывает, что в алгоритме 3 существует еще одна избыточность: большое количество сравнений в строках 19–27. В данных строках происходит сравнение индексов столбцов ненулевых элементов, находящихся в одной строке, с индексами ненулевых элементов другой строки, и если совпадение обнаружено, то происходит операция над этими элементами. Схематически данная операция изображена на рис. 2 (результат выполнения операций над элементами строк обозначен как Вектор-результат). Для выполнения этой операции приходится перебирать и сравнивать между собой все индексы столбцов ненулевых элементов двух векторов (строк), что неэффективно.

Алгоритм 3 ILU(0)-разложение с использованием разреженного строчного формата без поиска ненулевых элементов

```
1 Для i = 2, ..., N
2   s_1 = iptr(i) – номер начального элемента
3   pr_1 = Истина
4   Пока pr_1 = Истина Продолжать
5     k = jptr (s_1)
6     Если k ⩾ i
7       Прервать текущий цикл
8     aelem(s_1) = aelem(s_1) / aelem(diag(k))
9     s_2 = s_1
10    s_1 = s_1 + 1
11    y_1 = s_1
12    y_end1 = iptr(i+1)
13    y_2 = diag(k)+1
14    y_end2 = iptr(k+1)
15    Если y_end1 ⩽ y_1 или y_end2 ⩽ y_2 Тогда
16      Продолжить текущий цикл
17    pr_2 = Истина
18    Пока pr_2 = Истина Продолжать
19      Если jptr(y_1) = jptr(y_2)
20        aelem(y_1) = aelem(y_1) – aelem(t_1) × aelem(y_2)
21        Увеличить y_1 и y_2
22      Если jptr(y_1) > jptr(y_2)
23        Увеличить y_2
24      Если jptr(y_1) < jptr(y_2)
25        Увеличить y_1
26      Если iptr(k + 1) < y_2 или iptr(i + 1) < y_2
27        pr = Ложь
28 Увеличить i
```

Для этой операции можно воспользоваться временными векторами, так как в разреженном строчном формате хранятся не только сами элементы, но и адреса столбцов, в которых находятся эти элементы, что дает возможность построить циклы без дополнительных условий.

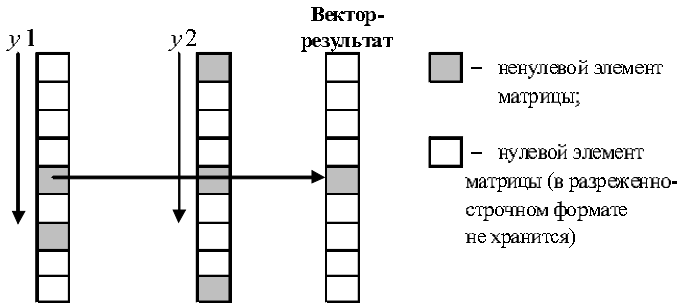

Рис. 2. Схематическое изображение операции сравнения над ненулевыми элементами в алгоритме 3

Таким образом, суть предлагаемого усовершенствования заключается в том, что операция поиска одинаковых индексов столбцов двух векторов (строк) разделена на два этапа. На первом этапе создаются векторы **tmpvec** и **tmpjptr**: в первый записываются статусы наличия ненулевых элементов в анализируемом векторе (строке), а во втором хранятся соответствующие адреса столбцов этих ненулевых элементов. На втором этапе сравниваются элементы второго вектора (строки) с элементами временного вектора (**tmpvec**), и если найдены ненулевые элементы с одинаковыми индексами столбцов, то выполняется соответствующая операция над элементами. Схематически этапы изображены на рис. 3.

В алгоритме 1 выгодно записывать во временные векторы целиком строку i, поскольку она используется на протяжении всего шага ik (см. рис. 1), т.е. временный вектор будет создаваться один раз и использоваться на протяжении всего цикла.

В результате использования временных векторов получен алгоритм 4.

В используемом в алгоритме 4 векторе **tmpvec** хранится значение Истина, если элемент ненулевой, а в векторе **tmpjptr** хранится адрес этого ненулевого элемента. Из приведенного алгоритма видно, что для дополнительных векторов понадобится дополнительная память компьютера для хранения N булевых элементов вектора **tmpvec** и N целочисленных значений их адресов в векторе **tmpjptr**. Такое увеличение машинной памяти по сравнению с памятью для хранения основной матрицы является незначительным и не учитывается.

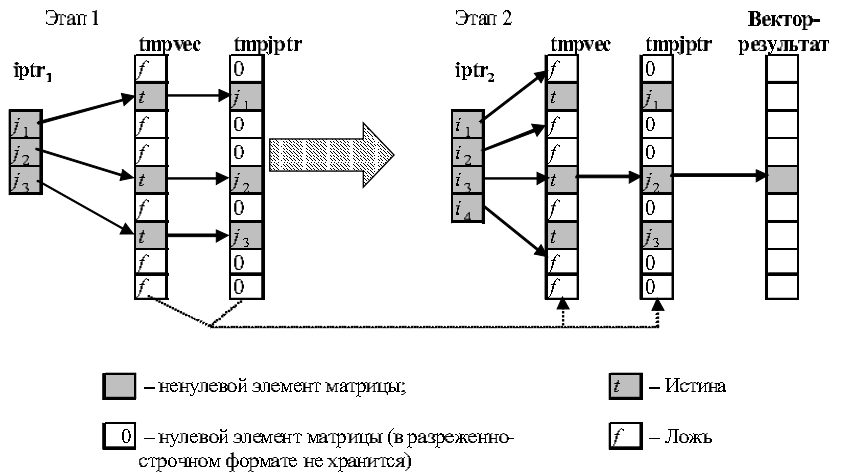

Рис. 3. Схематическое изображение этапов операций над ненулевыми элементами в алгоритме ILU(0)-разложения с использованием временных векторов

§3. Вычислительный эксперимент

Для вычислительного эксперимента использовался персональный компьютер (при вычислениях не использовалось распараллеливание, т.е. работало одно ядро процессора) с параметрами: процессор – Intel(R) Core(TM) i7 CPU 970; частота процессора – 3,20 ГГц; объем ОЗУ – 16 Гбайт; число ядер – 6; операционная система – Linux Kubuntu 11.10 x64.

Оценка временных затрат сделана на нескольких СЛАУ (полученных последовательным учащением сегментации границ проводник-диэлектрик и диэлектрик-диэлектрик из задачи вычисления электрической ёмкости двух полосок с помощью метода моментов), решаемых методом BiCGStab с использованием предобусловливания по алгоритмам 2, 3 и 4.

В экспериментах изменялся допуск обнуления, меняющий плотность матрицы $\mathbf{A}_S$. Итерации продолжались, пока относительная норма вектора невязки была больше 10^{-6}. Для оценки ускорения решения СЛАУ использовано отношение времени решения СЛАУ по усовершенствованным алгоритмам 3, 4 (Ts и Tr) ко времени исходного алгоритма 2

Алгоритм 4 ILU(0)-разложение с использованием разреженного строчного формата с двумя дополнительными векторами

```
1 Для i = 2, ..., N
2   s_1 = iptr(i) – номер начального элемента
3   pr_1 = Истина
4   Для j = s_1, ..., iptr(i+1)
5     tmpvec(jptr (j))=Истина
6     tmpjptr(jptr (j))=j
7   Увеличить j
8   Пока pr_1 = Истина Продолжать
9     k = jptr (s_1)
10    Если k ⩾ i
11      Прервать текущий цикл
12    aelem(s_1) = aelem(s_1) / aelem(diag(k))
13    s_2 = s_1
14    s_1 = s_1 + 1
15    y_1 = s_1
16    y_end1 = iptr(i+1)
17    y_2 = diag(k)+1
18    y_end2 = iptr(k+1)
19    Если y_end1 ⩽ y_1 или y_end2 ⩽ y_2 Тогда
20      Продолжить текущий цикл
21    Для j = y_2, ..., y_end2
22      Если tmpvec(jptr(j))=Истина
23        aelem(tmpjptr(jptr(j)))
            = aelem(tmpjptr(jptr(j))) – aelem(s_2) × aelem(j)
24    Увеличить j
25 Увеличить i
```

(T). При вычислениях использовалась система TALGAT [8]. Вид конфигурации 1, состоящей из двух проводников на слое диэлектрика над идеально проводящей плоскостью, приведен на рис. 4*a*, а конфигурации 2, состоящей из двух проводников (без диэлектрика) над идеально

проводящей плоскостью, – на рис. 4*б*. За счёт изменения числа сегментов для каждой конфигурации сформированы матрицы разных порядков. Полученные зависимости ускорения решения СЛАУ от плотности матрицы $\mathbf{A}_S$ приведены на рис. 5.

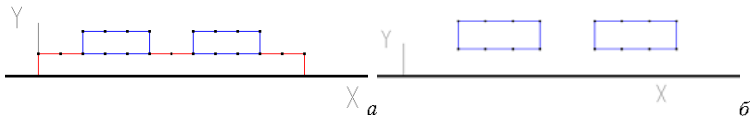

Рис. 4. Исследуемые конфигурации: а – два проводника с диэлектриком, б – два проводника без диэлектрика

Из рис. 5 следует, что усовершенствованный алгоритм 3 работает быстрее алгоритма 2, а алгоритм 4 еще быстрее. Максимальные ускорения получены при $N = 4800$: в 1,6 раза для алгоритма 3, и в 2,5 раза для алгоритма 4. При $N = 8000$ максимальное ускорение составило 1,2 раза для алгоритма 3 и 1,7 раза для алгоритма 4.

На рис. 5 видно, что при малых плотностях матрицы $\mathbf{A}_S$ ускорение падает. Этот факт объясняется тем, что в данной работе оптимизировался только этап ILU(0)-разложения, т.е. одна из трех составляющих общего времени: предфильтрации (Tp), ILU(0)-разложения (Tlu) и итерационного процесса (Ti). На рис. 6 приведено ускорение ILU(0)-разложения при использовании алгоритмов 3 ($Tlus$) и 4 ($Tlur$) относительно алгоритма 2 (Tlu). Видно, что для $N = 4800$ максимальное ускорение для алгоритма 3 составляет 2 раза, а для алгоритма 4 – почти 4 раза. На всех графиках ускорение увеличивается с уменьшением плотности матрицы. Нет отсутствия ускорения при минимальной плотности матрицы, как это было на рис. 5.

Показательно сравнение ускорения на разных этапах решения при использовании алгоритмов 2 и 4. Для наглядности зависимость соотношения времени этапов решения СЛАУ по алгоритму 4 от плотности матрицы $\mathbf{A}_S$ для $N = 4800$ изображена на рис 7.

Из рис. 7 ясно видно, что при уменьшении плотности матрицы часть времени, затрачиваемая на итерационный процесс, увеличивается, а на ILU(0)-разложение (за счет которого получено ускорение) – уменьшается. Следовательно, ускорение общего решения СЛАУ также

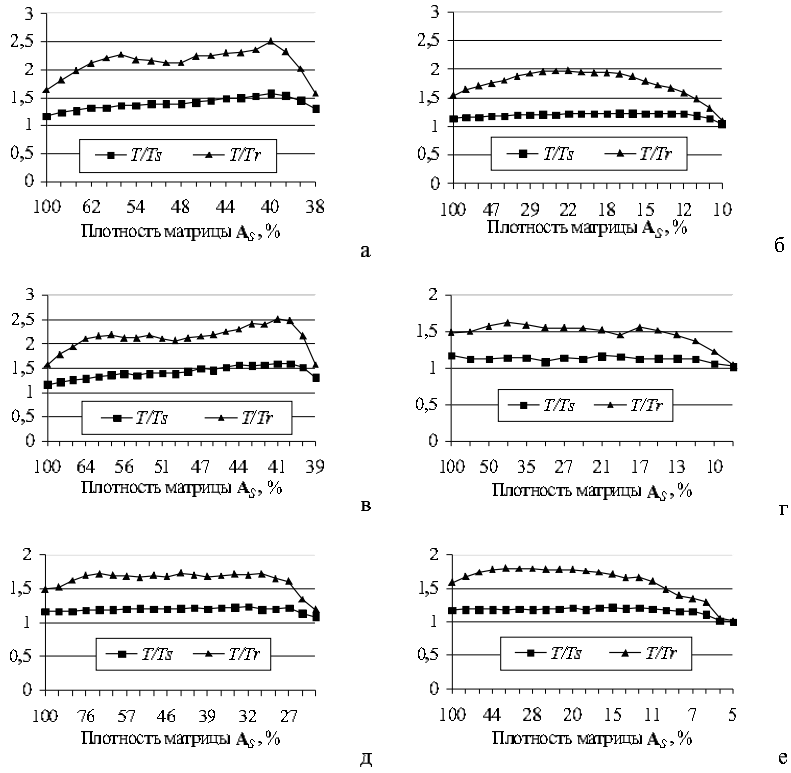

Рис. 5. Отношение времени решения СЛАУ методом BiCGStab при использовании алгоритма 2 (T) ко времени алгоритма 3 (Ts) и алгоритма 4 (Tr) для конфигурации 1 при N= 4800 (а), 6000 (в), 8000 (д) и конфигурации 2 при N=2240 (б), 6800 (г), 7999 (е)

уменьшается. Между тем, уменьшение ускорения при малой плотности матрицы не уменьшает значимости усовершенствования алгоритма. Действительно, на практике требуется минимизовать время решения СЛАУ за счет выбора оптимального значения допуска обнуления. При этом значении время ILU(0)-разложения примерно равно времени итерационного процесса [1]. Поэтому уменьшение времени вычисления ILU(0)-разложения весьма важно.

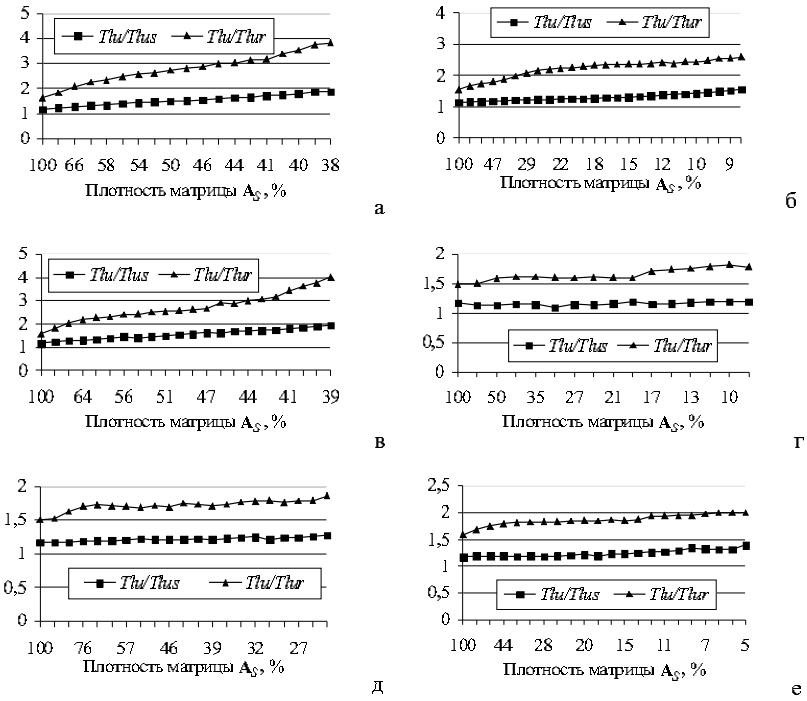

Рис. 6. Отношение времени ILU(0)-разложения при использовании алгоритма 2 (Tlu) ко времени алгоритма 3 ($Tlus$) и алгоритма 4 ($Tlur$) для конфигурации 1 при N=4800 (а), 6000 (в), 8000 (д) и конфигурации 2 при N=2240 (б), 6800 (г), 7999 (е)

§4. Заключение

В работе предложены усовершенствованные варианты алгоритма ILU(0)-разложения при использовании разреженного строчного формата хранения. Устранена избыточность алгоритма (поиск стартовых элементов), а также предложено использовать дополнительные временные векторы для получения ускорения. При решении задачи вычисления электрической ёмкости двух полосок данные усовершенствования позволили ускорить вычисление ILU(0)-разложения в 4 раза, а

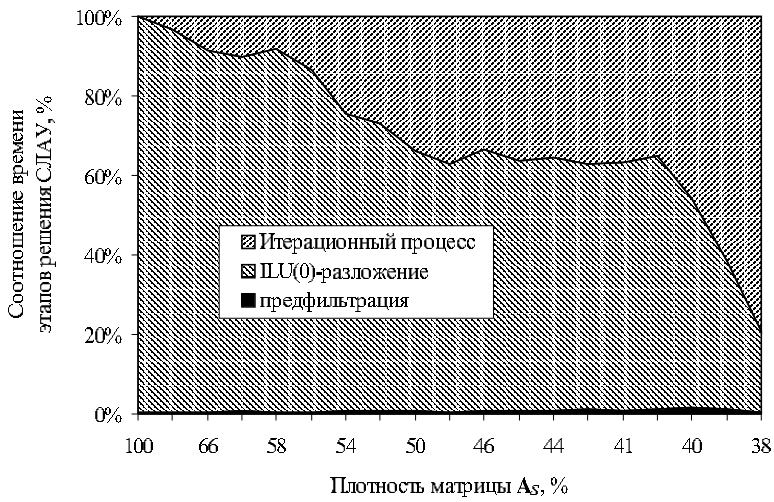

Рис. 7. Зависимость соотношения времени, затрачиваемого на этапы решения СЛАУ итерационным методом с предобусловливанием, от плотности матрицы $\mathbf{A}_S$ для N=4800

решение СЛАУ – в 2,5 раза. Таким образом, использование результатов работы позволяет уменьшить и время решения СЛАУ, и затраты памяти, что особенно актуально при решении задач большой размерности.

Предложенные в данной работе модификации алгоритма ILU(0)-разложения использовались только при одноядерных вычислениях, поэтому в дальнейшем необходимо рассмотреть возможность их применения в версиях, использующих многопоточность и кэш.

Литература

1. Т. Р. Газизов, С. П. Куксенко, *Оптимизация допуска обнуления при решении СЛАУ итерационными методами с предобусловливанием в задачах вычислительной электродинамики.* — Электромагн. волны электронн. сист. **8** (2004), 26–28.
2. С. Писсанецки, *Технология разреженных матриц.* Мир, М., 1988.
3. Р. Р. Ахунов, С. П. Куксенко, В. К. Салов, Т. Р. Газизов, *Форматы хранения разреженных матриц и ускорение решения СЛАУ с плотной матрицей*

итерационными методами. Зап. научн. семин. ПОМИ **405** (2012) (принято к печати).
4. H. van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of nonsymmetric linear systems.* — SIAM J. Sci. Stat. Comput. **13** (1992), 631–644.
5. С. П. Куксенко, Т. Р. Газизов, *Итерационные методы решения системы линейных алгебраических уравнений с плотной матрицей.* Томский государственный университет, Томск, 2007.
6. Y. Saad, *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.
7. С. П. Куксенко, Т. Р. Газизов, *Сравнение способов предфильтрации при решении СЛАУ с плотной матрицей итерационными методами с предобусловливанием.* — Инфокоммуникац. технол. **5**, No. 2 (2007), 14–18
8. *Свидетельство о государственной регистрации программы для ЭВМ No. 2012610712. TALGAT 2010.* Газизов Т.Р., Мелкозеров А. О., Газизов Т. Т., Куксенко С. П., Заболоцкий А. М., Аширбакиев Р. И., Вершинин Е. А., Салов В. К., Лежнин Е. В., Орлов П. Е., Бевзенко И. Г., Калимулин И. Ф. Заявка No. 2011617178. Дата поступления 26 сентября 2011 г. Зарегистрировано в Реестре программ для ЭВМ 13 января 2012 г.

Akhunov R. R., Kuksenko S. P., Salov V. K., Gazizov T. R. Optimization of the ILU(0) factorization algorithm with the use of compressed sparse row format.

Improvements to the ILU(0) factorization algorithm for preconditioning linear algebraic systems with dense matrices are suggested. The preconditioner is stored in compressed sparse row format. On the example of the problem of computing the electrical capacity of two stripes, it is demonstrated that the modifications proposed provide for a significant reduction of the time for computing the ILU(0) preconditioner (up to 4 times) and for solving the preconditioned linear system (up to 2.5 times).

Томский государственный университет
систем управления и радиоэлектроники,
кафедра телевидения и управления,
пр. Ленина, 40, г. Томск 634050, Россия

E-mail: arr@pop3.ru

E-mail: ksergp@sibmail.com

E-mail: catred@mail2000.ru

E-mail: talgat@tu.tusur.ru

Поступило 2 мая 2012 г.