Численные методы

УДК 519.688

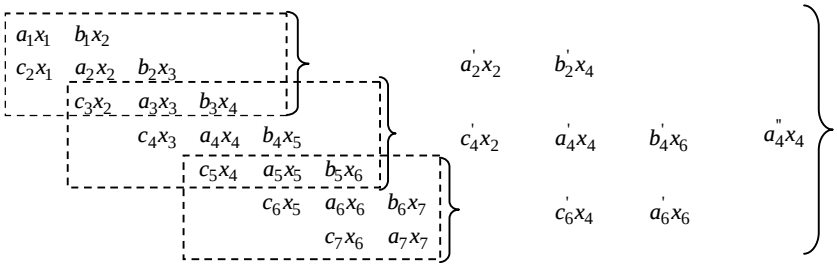
С.В. Ярмушкин, Д.Л. Головашкин

ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ РЕШЕНИЯ ТРЕХДИАГОНАЛЬНЫХ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Исследованы два параллельных алгоритма решения трехдиагональных систем линейных алгебраических уравнений. Произведены аналитические и численные исследования характеристик данных алгоритмов. Показана высокая эффективность применения параллельных алгоритмов. Проведен сравнительный анализ алгоритмов. Приведены результаты экспериментов по исследованию ускорения и масштабируемости алгоритмов.

При решении различных задач возникает необходимость решения трехдиагональных систем линейных алгебраических уравнений (СЛАУ). Такие задачи возникают и в физике (оптика, теория теплопроводности, газовая динамика и др.), и в математике (теория разностных схем, проекционные и вариационные методы) [1]. При этом число уравнений в системах может быть велико и однопроцессорный компьютер либо не может их решить из-за нехватки памяти, либо их решение занимает очень много времени, поэтому возникает необходимость разработки параллельных алгоритмов [2] решения трехдиагональных систем. Рассмотренные в данной работе параллельные алгоритмы циклической редукции и декомпозиции области известны из литературы [3], однако они не описаны подробно и даны без сравнительного анализа.

Алгоритм циклической редукции. Суть алгоритма заключается в исключение неизвестных с нечетными индексами из перекрывающихся троек уравнений. На рис. 1 изображена схема исключения для 7 уравнений.



Р и с. 1. Схема циклической редукции

Считая N (размерность системы) нечетным числом, мы получаем в конце процесса модифицированную систему, содержащую только четные переменные. Данная система будет также трехдиагональной и можно повторять процесс до тех пор, пока дальнейшая редукция будет невозможна. В частности, при $N=2^q-1$ редукция закончится единственным финальным уравнением. Решив его, начнем обратную подстановку. Если $N \neq 2^q-1$, то редукцию можно закончить системой с небольшим количеством переменных. Решив ее, начнем обратную подстановку. Альтернативный вариант предусматривает добавление к системе некоторого количества фиктивных уравнений вида $x_i=1$, чтобы общее число переменных стало равно 2^q-1 для некоторого q. Можно заметить, что операции, ведущие к новым уравнениям для $x_2, x_4 \ldots x_N$ независимы и могут выполняться одновременно. На этом и основан параллельный алгоритм. Его схема приведена на примере для 7 уравнений и 3 задач (рис. 2).

Ускорение оценим величиной

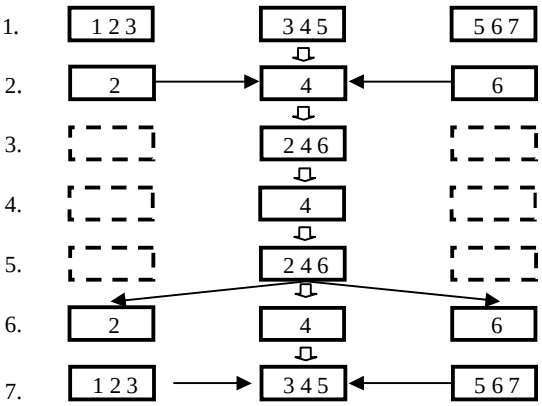
$$S = \frac{t(2^q - 1)}{t(\frac{2^q - 1}{np}) + 2qt_i},$$

где $t(N)$ – время решения системы из N уравнений; t_i – время пересылки коэффициентов.

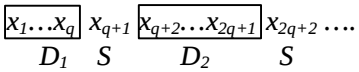
Алгоритм декомпозиции области. Рассмотрим алгоритм, основанный на декомпозиции области. Пусть имеется система уравнений вида $Ax=b$, где A – трехдиагональная матрица размера $N\times N$. Причем $N=pq+p-1$ для некоторых целых p и q $(p>q)$. Алгоритм заключается в разбиении множества неизвестных на непересекающиеся подмножества (рис. 3). Затем переменные переупорядочиваются путем присвоения последних индексов элементам множества S, что приводит к системе стреловидного вида:

$$\begin{bmatrix} A_1 & & & & B_1 \\ & A_2 & & & B_2 \\ & & \mathbf{O} & & \vdots \\ & & & A_p & B_p \\ C_1 & C_2 & \mathbf{L} & C_p & A_s \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \\ x_s \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \\ b_s \end{bmatrix}. \quad (1)$$

Вводя следующие обозначения A_l=diag$(A_1\ldots A_p)$, $B^T=(B_1\ldots B_p)$, $C=(C_1\ldots C_p)$, $x_l=(x_1\ldots x_p)$, $b_l=(b_1\ldots b_p)$, систему (1) можно представить в виде:



Р и с. 2. Схема параллельного алгоритма циклической редукции



Р и с. 3. Разбиение множества неизвестных

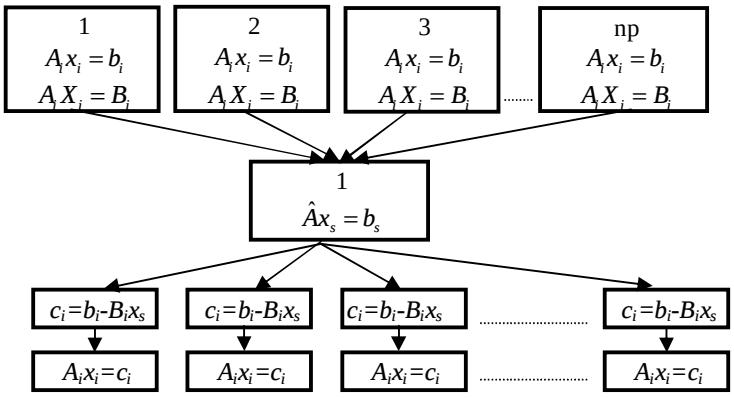
$$A_l x_l + B x_s = b_1 ; \quad (2)$$

$$C x_l + A_s x_s = b_s . \quad (3)$$

Предположив, что матрица A_l невырожденная, умножим (3) на CA_1^{-1} и вычтем полученное из (2):

$$\hat{A}x_s = \hat{b}, \quad \hat{A} = A_s - CA_l^{-1}B, \quad \hat{b} = b_s - CA_l^{-1}b . \quad (4)$$

Как только будет решена система (4) относительно x_s, остальные неизвестные можно найти из систем $A_l x_i = b_l - Bx_s$, $i=1...p$, которые можно решать независимо друг от друга. Кроме того, слагаемые $CA_l^{-1}B, CA_l^{-1}b$ также можно находить независимо. Схема алгоритма в терминах модели «задача-канал» [4] представлена на рис. 4.



Р и с. 4. Схема параллельного алгоритма декомпозиции области

Ускорение алгоритма будет оцениваться величиной

$$S = \frac{t(p)}{t(p_1) + t_{i}(q) + t_{n}(q(p-1)) + t_{n}(p-1)},$$

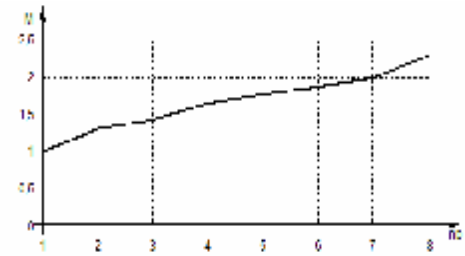
где $t(p)$ – время решения исходной системы; p_1 – число матриц на первой задаче; – время глобальной рассылки массива, состоящей из q элементов.

Результаты вычислительных экспериментов. Для исследований были составлены параллельные программы, реализующие данные алгоритмы. Все эксперименты проводились на кластере СНЦ РАН на ЭВМ с процессорами PENTIUM-3 с тактовой частотой 600 МГц и ОЗУ 512 Мб, работающими под управлением операционной системы Red Hat Linux. Программы на-

писаны на языке Си с использованием библиотеки MPI [5]. При проведении вычислительных экспериментов была исследована масштабируемость алгоритма декомпозиции области; ее график приведен на рис. 5. Масштабируемость — это величина, равная отношению объема данных V^N к объему данных V^1 последовательного процесса, при этом длительность работы должна совпадать [2]. Масштабируемость показывает, во сколько раз увеличится объем данных при реализации на N процессорах по сравнению с однопроцессорной ЭВМ, если время реализации останется постоянным:

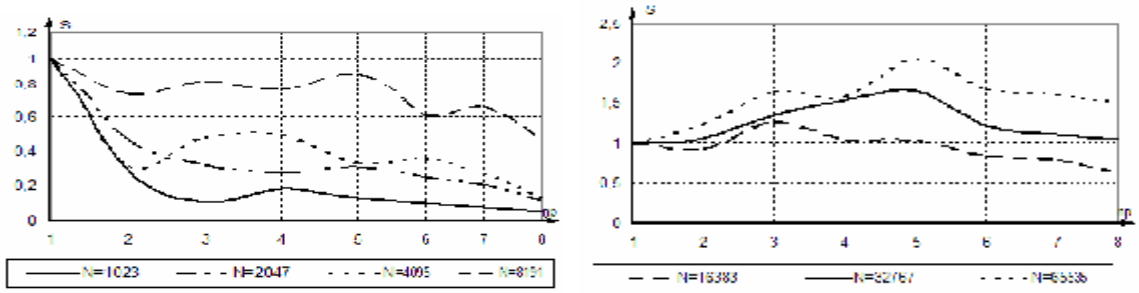
$$M = \frac{V^N}{V^1} \text{ при } T^1(V^1) = T^N(V^N),$$

где $T^1(V^1)$ – время работы последовательного алгоритма на объеме данных V^1, а $T^N(V^N)$ – время работы параллельного алгоритма на объеме данных V^N. Как видно из графика (см. рис. 5) метод декомпозиции области обладает свойством масштабируемости, кроме того график является линейным. Это говорит о том, что при увеличении числа задач масштабируемость будет также увеличиваться. Для метода циклической редукции масштабируемость не рассчитывалась из-за особенности алгоритма.



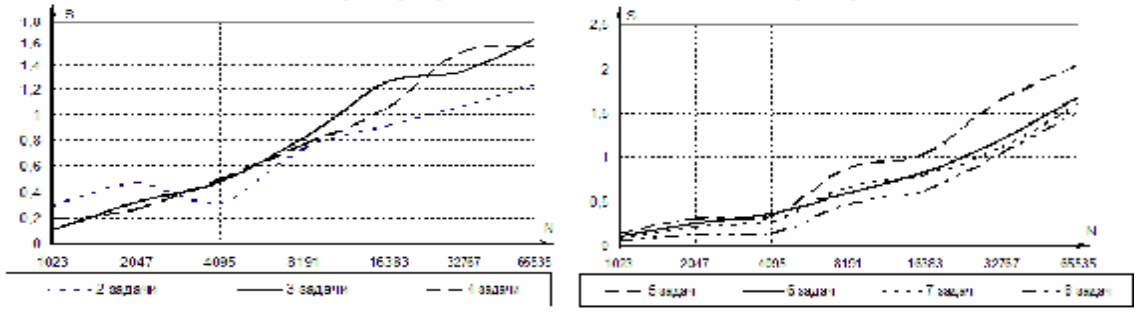
Р и с. 5. График зависимости масштабируемости от числа процессов

Результаты исследования зависимости ускорения алгоритма циклической редукции от числа задач и числа уравнений представлены на рис. 6 и рис. 7.



а *б*

Р и с. 6. График зависимости ускорения (S) от числа задач (np):
а -для малых размеров решаемой системы; *б* – для больших размеров системы



а *б*

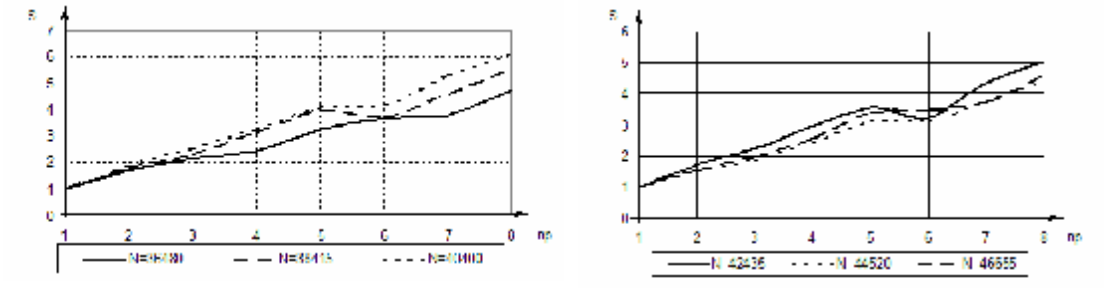
Р и с. 7. Графики зависимости ускорения (S) от числа уравнений (N):
а — для 2, 3, 4 задач; *б* — для 5, 6, 7, 8 задач

На графике, изображенном на рис. 6, *а* видно, что при малом числе уравнений (до 8191 включительно) ускорение меньше единицы, т.е. параллельная программа работает медленнее, чем последовательная. Это связано с тем, что затраты времени на пересылку данных превышают выигрыш в скорости выполнения арифметических операций; также необходимо учитывать тот факт, что часть процессоров по мере выполнения прямого хода алгоритма простаивают. На больших же размерах системы алгоритм эффективен (рис. 6, *б*), так как ускорение больше единицы. Рис. 7 показывает, что ускорение увеличивается с ростом числа уравнений.

Необходимо отметить также тот факт, что при числе задач больше 5 ускорение уменьшается: это связано с тем, что при увеличении числа задач, возрастает общий объем пересылаемых данных, увеличивается сложность синхронизации задач друг с другом (так как некоторые зада-

чи заканчивают работу раньше других), а также увеличивается время простоев. Исходя из этого, можно сделать вывод, что для данного метода и числа уравнений пять задач является оптимальным числом задач. Для большего количества уравнений необходимы дополнительные исследования.

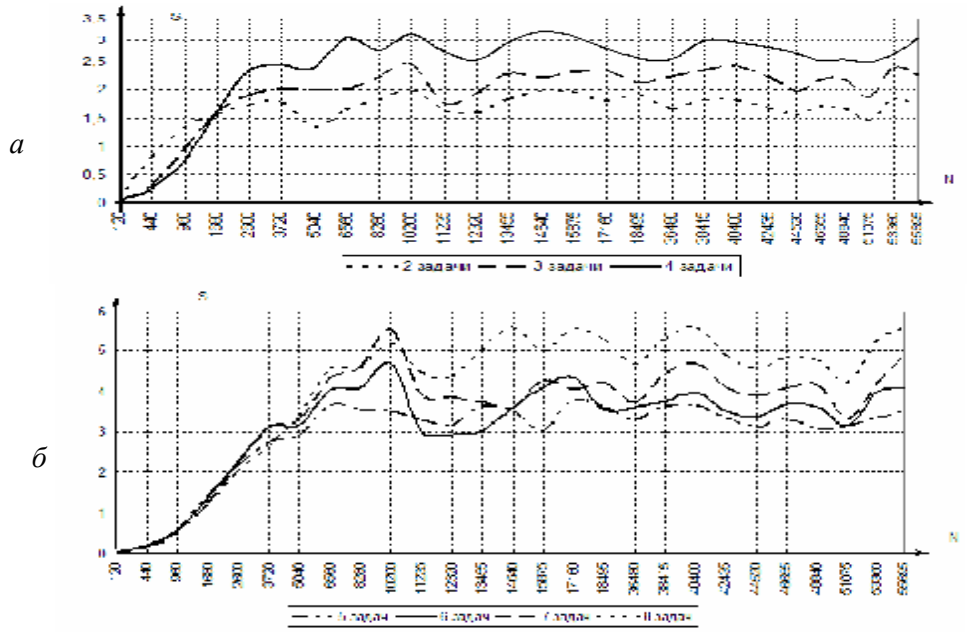
Для метода декомпозиции области были получены результаты, представленные на рис. 8 и 9.



а *б*

Р и с. 8. Графики зависимости ускорения (S) от числа задач (пр)
а — для числа уравнений от 36480 до 40400; *б* —для числа уравнений от 42435 до 46655

По графикам, изображенным на рис. 8, можно сказать, что ускорение линейно увеличивается с ростом числа задач. Это позволяет уменьшать время работы программы, увеличивая число задач.



а

б

Р и с. 9. Графики зависимости ускорения (*S*) метода декомпозиции области от размерности решаемой системы (*N*): *а* — для 2, 3, 4 задач; *б* — 5, 6, 7, 8 задач

Как видно из графиков, ускорение больше 1 (при числе уравнений больше 2000). Это говорит о том, что параллельный вариант работает быстрее, чем последовательный. Графики на рис. 9 подчиняются закону Амдала [2].

Сравнение алгоритмов. Рассмотрим достоинства и недостатки обоих алгоритмов. Сравним их сначала численно. Для примера используем следующие данные: число уравнений 16383. Тогда для первого алгоритма q =14, для второго — q=126 и p=128; число задач возьмем равным 5. Для первого алгоритма характеристики будем считать для центральной задачи, а для второго – для первой, так как эти задачи работают либо без простоев (в случае декомпозиции области), либо с минимальными простоями (для циклической редукции), выполняют больше действий и занимают большую память. Результаты сведем в таблицу, где использованы следующие обозначения: $t_{àð}$ – время выполнения одной арифметической операции (считаем, что все они выполняются одинаковое количество времени), $t_{ï}$ – время передачи сообщения.

Сравнение характеристик алгоритмов

	Алгоритм циклической редукции	Алгоритм декомпозиции области
Используемая память (число элементов массивов)	$\sim 2{,}5 \cdot 10^5$	$\sim 10^6$
Число операций	$2 \cdot 10^5$	$\sim 10^8$
Время простоя	$\sim 16383 t_{àä} + 2 t_{ï}$	0

Как видно из приведенной таблицы, во втором алгоритме число арифметических операций больше на три порядка, что подтверждают полученные результаты вычислений времени работы программ, реализующие эти алгоритмы.

Сравним алгоритмы по типу коммуникаций: в первом алгоритме – локальные коммуникации, во втором – глобальные. Локальные коммуникации предпочтительнее, так как в случае глобальных коммуникаций принимающая задача принимает данные последовательно, кроме того, некоторые задачи не могут передать данные непосредственно принимающей задаче. В случае локальных коммуникаций соседние задачи общаются друг с другом напрямую.

Отличаются алгоритмы и по декомпозиции данных: в алгоритме циклической редукции используются перекрывающиеся тройки уравнений, а в методе декомпозиции области неизвестные разбиваются на непересекающиеся подмножества.

Как видно из рис. 6 – 9, алгоритм декомпозиции области имеет большее ускорение по сравнению с циклической редукцией. Это связано с тем, что второй метод изначально разработан как параллельный и в нем не надо синхронизировать задачи друг с другом на каждом шаге. Здесь необходимы только две массовые пересылки, причем одну из них можно совместить с операцией сложения. Однако по времени данный алгоритм работает медленнее, так как здесь присутствует такая операция как перемножение матриц, а это, как известно, довольно медленный процесс. Возможно, используя другую архитектуру вычислительных систем (например, векторную) или другие алгоритмы перемножения матриц (например, параллельные), можно достичь лучших результатов.

Второй алгоритм имеет меньшее время простоев, что также отражается на ускорении. Простоев в первом алгоритме не избежать, так как число уравнений в системе становится меньше с каждым шагом, и нет смысла дублировать вычисления на нескольких задачах. Второй алгоритм гораздо легче в практической реализации, здесь не надо отслеживать, какие из задач закончили свою работу. Кроме того, в первом алгоритме необходимо отмечать шаг, на котором они это сделали, чтобы учесть этот момент при обратной подстановке.

Одним из главных недостатков алгоритма циклической редукции является привязка к степени двойки, если, конечно, не брать в расчет модификации этого алгоритма. Подобное можно сказать и об алгоритме декомпозиции области (напомним, число его уравнений зависит от двух параметров: $N=pq+p-1$), однако в этом случае возможностей изменять число уравнений больше.

Заключение. Общим недостатком обоих алгоритмов является то, что их нельзя использовать для любого числа уравнений. Среди достоинств можно отметить тот факт, что оба параллельных алгоритма, при соблюдении некоторых условий (одно из которых — большое число уравнений) работают быстрее своих последовательных вариантов. Первый алгоритм является оптимальным по времени работы, объему используемой памяти, количеству арифметических операций, однако он проигрывает второму по ускорению, по времени простоев. Кроме того, алгоритм декомпозиции области более гибок в вопросе выбора числа уравнений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Самарский А.А.* Теория разностных схем. М.: Наука, 1989. 614с.
2. *Головашкин Д.Л., Головашкина С.П.* Методы параллельных вычислений. Часть 2: Учеб. пособие. Самара: Самар. гос. аэрокосм. ун-т. 2003. 103 с.
3. *Ортега Д.* Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991, 364с.
4. *Brucel P.L.* Parallel programming an introduction. Prentice Hall International (UK) Limited, 1993. 270 с.
5. Введение в программирование для параллельных ЭВМ и кластеров: Учеб. пособие/Кравчук В.В., Попов С.Б., Привалов А.Ю., Фурсов В.А., Шустов В.А.; Под ред. В.А. Фурсова. Самара: Самар. научный центр РАН, Самар. гос. аэрокосм. ун-т. 2000. 87 с.

Поступила 11.03.2004 г.