

УДК 519.612

doi 10.26089/NumMet.v16r109

ПАРАЛЛЕЛЬНОЕ ФОРМИРОВАНИЕ ПРЕДОБУСЛОВЛИВАТЕЛЯ, ОСНОВАННОГО НА АППРОКСИМАЦИИ ОБРАЩЕНИЯ ШЕРМАНА–МОРРИСОНА

Н. С. Недожогин¹, С. П. Копысов², А. К. Новиков³

Исследуются возможности ускорения предобусловленных методов бисопряженных градиентов (BiCGStab, Bi-Conjugate Gradient Stabilized) с предобусловлителем на основе аппроксимации обращения матрицы по формуле Шермана–Моррисона. Рассмотрена новая форма параллельного алгоритма, использующая матрично-векторные произведения при формировании матриц предобусловливателя. Показана эффективность распараллеливания наиболее ресурсоемких операций этого предобусловливателя на графических процессорах.

Ключевые слова: линейные системы уравнений, явное предобусловливание, формула Шермана–Моррисона, параллельные вычисления, графические ускорители.

1. Введение. Построение эффективных методов решения больших систем линейных алгебраических уравнений на основе предобусловленных итерационных методов, особенно в контексте параллельных вычислений, является достаточно трудной задачей. Матрица-предобусловливатель не только должна быть в определенном смысле близка к обратной матрице коэффициентов системы, но и должна допускать эффективно распараллеливаемый алгоритм ее формирования и умножения на вектор.

К широко используемым предобусловливателям сегодня можно отнести методы, ориентированные на разреженные матрицы и основанные на неполном разложении на треугольные составляющие, такие как метод неполного LU-разложения [1]. Несмотря на высокую эффективность и популярность, данные алгоритмы сталкиваются с известными проблемами при их параллельной реализации на гибридных вычислительных системах [2].

Высокий потенциал распараллеливания имеют предобусловливатели на основе аппроксимации обратной матрицы: полиномиальные (TNS, Truncated Neumann Series [1]), разреженные аппроксимации обратной матрицы (AINV, Approximate INVerse [3]), аппроксимации обратной матрицы в факторизованной форме (такие как FSAI, Factorized Sparse Approximate Inverse [4], SPAI, Sparse Approximate Inverse [5] и др.), а также метод AISM (Approximate Inverse based on the Sherman–Morrison formula) [6], основанный на явном предобусловливателе, использующем малоранговую модификацию Шермана–Моррисона [7, 8].

В настоящей статье ставятся две задачи: разработка методов параллельного построения предобусловливателей AISM, а также реализация и практическая оценка параллельной эффективности предложенных методов для гибридных вычислительных систем.

2. Предобусловливатель AISM. В этом разделе мы рассмотрим технику предобусловливания при решении систем линейных алгебраических уравнений $Ax = b$, предложенную в [6].

За основу построения обратной матрицы A^{-1} возьмем матрицу B того же порядка, что и A , но с известной обратной матрицей.

Теорема [7]. Пусть B – невырожденная матрица и векторы u и v , такие, что $r = 1 + v^T B^{-1} u \neq 0$. Тогда матрица $A = B + uv^T$ является обратимой и

$$A^{-1} = B^{-1} - r^{-1} B^{-1} uv^T B^{-1}. \quad (1)$$

Пусть A_0 – невырожденная матрица, обращение которой легко вычисляется, например диагональная или единичная. Тогда $A_k = A_0 + \sum_{i=1}^k u_i v_i^T$, где $k = 1, \dots, n$ и $A = A_n$. Если A_k, u_k, v_k удовлетворяют

¹ Институт механики Уральского отделения РАН (ИМ УрО РАН), ул. Т. Барамзиной, д. 34, 426067, Ижевск; мл. науч. сотр., e-mail: Negozhogin@inbox.ru

² Институт механики Уральского отделения РАН (ИМ УрО РАН), ул. Т. Барамзиной, д. 34, 426067, Ижевск; зав. лабораторией, e-mail: s.kopysov@gmail.com

³ Институт механики Уральского отделения РАН (ИМ УрО РАН), ул. Т. Барамзиной, д. 34, 426067, Ижевск; ст. науч. сотр., e-mail: sc_work@mail.ru

представлению (1), то обращение матрицы A может быть вычислено следующим образом:

$$A^{-1} = A_0^{-1} - \sum_{k=1}^n r_k^{-1} A_{k-1}^{-1} u_k v_k^T A_{k-1}^{-1}. \quad (2)$$

Представим (2) в матричной форме:

$$A_0^{-1} - A^{-1} = \Phi \Omega^{-1} \Psi^T. \quad (3)$$

Здесь $\Phi = [A_0^{-1} u_1, A_1^{-1} u_2, \dots, A_{n-1}^{-1} u_n]$, $\Psi = [v_1^T A_0^{-1}, v_2^T A_1^{-1}, \dots, v_n^T A_{n-1}^{-1}]$ и $\Omega^{-1} = \text{diag} [r_1^{-1}, r_2^{-1}, \dots, r_n^{-1}]$. Факторизация (3) записывается без явного вычисления A_k^{-1} через векторы u_k, v_k в виде

$$s_k = u_k - \sum_{i=0}^{k-1} \frac{t_i^T A_0^{-1} u_k}{r_i} s_i, \quad t_k = v_k - \sum_{i=0}^{k-1} \frac{v_k^T A_0^{-1} s_i}{r_i} t_i, \quad k = 1, \dots, n. \quad (4)$$

Тогда выполняются соотношения

$$A_{k-1}^{-1} u_k = A_0^{-1} s_k, \quad u_k^T A_{k-1}^{-1} = t_k^T A_0^{-1}, \quad (5)$$

$$r_k = 1 + v_k^T A_0^{-1} s_k = 1 + t_k^T A_0^{-1} u_k. \quad (6)$$

С учетом (5) соотношение (3) запишем в форме $A_0^{-1} - A^{-1} = A_0^{-1} S \Omega^{-1} T^T A_0^{-1}$, где $S = [s_1, s_2, \dots, s_n]$ и $T = [t_1, t_2, \dots, t_n]$ — матрицы, столбцы которых вычисляются по u_k, v_k .

Определим выбор A_0, u_k, v_k [6]: $A_0 = gI_n$, $u_k = e_k$, $v_k = (a^k - a_0^k)^T$, $k = 1, \dots, n$, где I_n и e_k — единичная матрица и ее k -й столбец, векторы a^k и a_0^k — k -е строки матриц A и A_0 . Тогда аппроксимация обратной матрицы и предобусловливатель примут вид $P_1 = A^{-1} = gI_n - g^{-2} U \Omega^{-1} V^T$.

Рассмотрим еще один вариант разложения обрабатываемой матрицы $A = W - Z$, где W — обратимая матрица, $Z = UV^T = \sum_{k=1}^n u_k v_k^T$, а векторы v_k, u_k такие, что $d_k = 1 - v_k^T W_{k-1}^{-1} u_k \neq 0$, $W_k = W_0 - \sum_{i=1}^k u_i v_i^T$.

Задавая выбор матриц $W = \beta \text{diag}(A)$, $\beta > 0$, $U = I$, $V = Z^T$ и следуя соотношениям (4)–(6), получим выражения для вычисления столбцов матриц S и T : $s_k = u_k - \sum_{i=1}^{k-1} \frac{t_i^T W^{-1} u_k}{d_i} s_i$ и $t_k = v_k - \sum_{i=1}^{k-1} \frac{v_k^T W^{-1} s_i}{d_i} t_i$.

Выражение для обратной матрицы имеет вид

$$A^{-1} = W^{-1} - W^{-1} S D^{-1} T^T W^{-1}. \quad (7)$$

Применялась следующая стратегия фильтрации: при вычислении матриц S и T оставляем элементы, значения которых по модулю больше некоторой величины τ (полученные матрицы обозначим \tilde{S} и \tilde{T}). Основываясь на (7), выпишем предобусловливатель, аппроксимирующий обратную матрицу, в виде

$$P = W^{-1} - W^{-1} \tilde{S} D^{-1} \tilde{T}^T W^{-1}. \quad (8)$$

Последовательный процесс формирования рассматриваемого предобусловливателя представлен в виде алгоритма 1.

Основные затраты последовательного алгоритма составляют два вложенных цикла, в которых определяются столбцы матриц \tilde{S} и \tilde{T} . Отметим, что матрицы формируются последовательно так, что существует зависимость по данным. При вычислении k -го столбца матрицы \tilde{S} выполняется скалярное произведение векторов $(t_i^T W^{-1}, u_k)$, здесь t_i — i -й столбец матрицы \tilde{T} и $(t_i)_k$ — k -й элемент данного столбца. Для формирования k -го столбца матрицы \tilde{T} , наоборот, требуются значения i -х столбцов матрицы \tilde{S} с вычислением скалярного произведения вида $(v_k^T W^{-1}, s_i)$.

Вторая значимая операция — построение матрицы явного предобусловливателя P на основе полученных матриц \tilde{S} и \tilde{T} (см. строку 29 алгоритма 1).

3. Параллельное построение предобусловливателя. Рассмотрим процесс построения предобусловливателя в модели параллелизма по данным, которая присуща алгоритму 1 и связана с вычислением скалярных, матрично-векторных и матричных произведений.

Вариант распараллеливания, включающий в себя вычисление скалярных произведений (строки 8, 12, 25 в алгоритме 1) на графических процессорах (GPU, Graphics Processing Unit), показал, что достижение ускорения существенно ограничено лимитирующим фактором — доступом к памяти.

Дальнейшее эффективное распараллеливание алгоритма связано с возможностью использования операций линейной алгебры, обладающих большим потенциалом распараллеливания, таких как матрично-векторные и матричные произведения, а также с сокращением перемещения данных по уровням иерархии памяти GPU. В этой связи предложен параллельный алгоритм 2 формирования предобусловливателя на GPU, в котором скалярные произведения в строках 8 и 12 алгоритма 1 заменены матрично-векторными произведениями. Для этого введены матрицы $S_k = \{s_1, \dots, s_{k-1}\}$ и $T_k = \{t_1, \dots, t_{k-1}\}$, состоящие из столбцов, вычисленных на $k - 1$ шаге. В этом случае $k - 1$ скалярное произведение выполняется в виде матрично-векторных произведений (строки 7 и 13 алгоритма 2); полученные векторы обозначены через x , а через x_i — компоненты этих векторов.

Алгоритм 1. Построение предобусловливателя AISM

1. $A = W - Z$	15. end if
2. $W = \beta \text{diag}(A); \quad Z = W - A$	16. end for
3. $U = I; \quad V = Z^T$	17. for $j = 1$ to n do
4. for $k = 1$ to n do	18. if $ (s_k)_j < \tau_u$ then
5. $s_k = u_k$	19. $(s_k)_j = 0$
6. $t_k = v_k$	20. end if
7. for $i = 1$ to $k - 1$ do	21. if $ (t_k)_j < \tau_v$ then
8. $\delta = (t_i^T W^{-1}, u_k)$	22. $(t_k)_j = 0$
9. if $\left \frac{\delta}{d_i} \right > \tau_u$ then	23. end if
10. $s_k = u_k - \frac{\delta}{d_i} s_i$	24. end for
11. end if	25. $d_k = 1 - (t_k^T W^{-1}, u_k)$
12. $\delta = (v_k^T W^{-1}, s_i)$	26. end for
13. if $\left \frac{\delta}{d_i} \right > \tau_v$ then	27. $\tilde{S} = \{s_1, s_2, \dots, s_n\}, \quad \tilde{T} = \{t_1, t_2, \dots, t_n\}$
14. $t_k = v_k - \frac{\delta}{d_i} s_i$	28. $D = \{d_1, d_2, \dots, d_n\}$
	29. $P = W^{-1} - W^{-1} \tilde{S} D^{-1} \tilde{T}^T W^{-1}$

Алгоритм 2. Построение предобусловливателя AISM на GPU

1. $A = W - Z$	17. end if
2. $W = \beta \text{diag}(A); \quad Z = W - A$	18. end for
3. $U = I; \quad V = Z^T$	19. for $j = 1$ to n do
4. for $k = 1$ to n do	20. if $ (s_k)_j < \tau_u$ then
5. $s_k = u_k$	21. $(s_k)_j = 0$
6. $t_k = v_k$	22. end if
7. $x = u_k T_k^T W^{-1}$	23. if $ (t_k)_j < \tau_v$ then
8. for $i = 1$ to $k - 1$ do	24. $(t_k)_j = 0$
9. if $\left \frac{x_i}{d_i} \right > \tau_u$ then	25. end if
10. $s_k = u_k - \frac{\delta}{d_i} s_i$	26. end for
11. end if	27. $d_k = 1 - (t_k^T W^{-1}, u_k)$
12. end for	28. end for
13. $x = v_k^T S_k W^{-1}$	29. $\tilde{S} = \{s_1, s_2, \dots, s_n\}, \quad \tilde{T} = \{t_1, t_2, \dots, t_n\}$
14. for $i = 1$ to $k - 1$ do	30. $D = \{d_1, d_2, \dots, d_n\}$
15. if $\left \frac{x_i}{d_i} \right > \tau_v$ then	31. $P = W^{-1} - W^{-1} \tilde{S} D^{-1} \tilde{T}^T W^{-1}$
16. $t_k = v_k - \frac{x_i}{d_i} s_i$	

Для вычислений на графических ускорителях скалярных произведений векторов использовалась функция `sublasSdot` из библиотеки `cuBLAS` (CUDA Basic Linear Algebra Subroutine library). Другие векторные операции были реализованы в виде ядер (kernel) в рамках технологии CUDA (Compute Unified

Device Architecture) собственной разработки.

Последний шаг алгоритма 2, содержащий матричные операции (умножение и сложение), был также распараллелен в рамках технологии CUDA. Было разработано ядро CUDA, в котором произведение матриц вычисляется в виде последовательности матрично-векторных произведений. Эффективность распараллеливания данного этапа очень высокая и затраты существенно меньше, чем на предыдущих шагах вычислений.

Алгоритм 3. Преобразование из заполненного формата хранения в CSR

```

1.  for i = 0 to n - 1 do
2.    for j = 0 to n - 1 do
3.      if aij ≠ 0 then
4.        ANL [i + 1] = ANL [i + 1] + 1
5.      end if
6.    end for
7.  end for
8.  k = 0, l = 0
9.  for i = 1 to n do
10.   l = ANL [i]
11.   ANL [i] = k
12.   k = k + l
13. end for
14. for i = 0 to n - 1 do
15.   for j = 0 to n - 1 do
16.     if aij ≠ 0 then
17.       row = ANL [i + 1]
18.       ANC [row] = j
19.       AV [row] = aij
20.       ANL [i + 1] = ANL [i + 1] + 1
21.     end if
22.   end for
23. end for
    
```

При построении предобусловливателя P_{AISM} разреженность матрицы неизвестна. Промежуточные вычисления на этапе формирования выполнялись над векторами s_k, t_k , которые являются столбцами матриц \tilde{S}, \tilde{T} , хранящихся по строкам. Расходы по памяти увеличивались, но сокращалось время обращения к элементам векторов. Для преобразования из сжатого формата хранения матриц CSR (Compressed Sparse Row) в формат хранения полных строк (этап построения предобусловливателя) и обратного преобразования (матрично-векторное произведение при решении линейных систем) были разработаны эффективные параллельные алгоритмы, позволяющие пренебречь затратами на преобразование матриц.

Для представления матрицы A размера $n \times n$ в формате CSR создаются три массива: AV — массив ненулевых элементов матрицы A размера nnz ; ANC — массив соответствующих столбцовых индексов, размера nnz ; ANL — массив размера $n + 1$, в котором i -й элемент массива показывает, с какого элемента в массиве AV начинается i -я строка матрицы A .

Преобразование из общего формата хранения в CSR выполняется в три этапа (алгоритм 3):

- 1) параллельно считаем число ненулевых элементов в каждой строке; для i -й строки полученное значение записываем в ANL [$i + 1$];
- 2) суммируем элементы массива ANL (строки 8–13);
- 3) пробегаем по элементам матрицы A и записываем ненулевые элементы и столбцовые индексы в соответствующие позиции массивов AV и ANC; при этом начальная позиция следующей строки используется как номер текущей позиции (см. строки 17, 20).

Каждая строка матрицы обрабатывается независимо, и каждый из этапов преобразования форматов реализован в виде kernel-функции CUDA.

Затраты по памяти при хранении матриц \tilde{S}, \tilde{T} и P составляют $18n^2$ байт для варианта с двойной точностью и $12n^2$ — с одинарной, что накладывает ограничение на максимальный размер рассматриваемых

Таблица 1
Ускорение при формировании предобусловливателя P_{AISM}

Матрица	$A(n/nnz)$	Cond (A)	P_{AISM}	
			OpenMP	GPU
Симметричные				
nasa2910	2910 / 174296	9.53×10^{64}	1.67	38.3
bcsstk15	3948 / 117816	6.64×10^9	1.81	36.2
Kuu	7102 / 340200	15.75×10^3	1.61	32.1
msc10848	10848 / 1229778	9.97×10^7	1.88	32.1
vibrobox	12328 / 301700	1.04×10^{19}	1.7	22.9
Несимметричные				
cdde5	961 / 4681	1.64×10^4	1.64	19.9
ex37	3565 / 67591	1.79×10^2	1.7	30.1
rajat03	7602 / 32653	1.26×10^7	1.67	23.8
flowmeter5	9669 / 67391	7.1×10^6	1.7	23.6
ex19	12005 / 259577	2.15×10^{12}	1.7	21.9
sme3Da	12504 / 874887	5.22×10^7	3.5	40.4
poisson3Da	13514 / 352762	1.12×10^3	1.65	20.9

мых систем для решения на GPU ($n \sim 13000$). Хотя возможны варианты сокращения затрат памяти при хранении матриц S и T в одном из разреженных форматов.

Для сравнения все операции над векторами (инициализация, умножение вектора на скаляр, сложение векторов, скалярное произведение векторов) были реализованы тоже в модели общей памяти OpenMP с помощью директивы распараллеливания циклов. Некоторые результаты, демонстрирующие эффективность распараллеливания при построении предобусловливателя, и характеристики матриц представлены в табл. 1. Ускорение оценивалось по сравнению с однопоточным вариантом, выполняемым на центральном процессоре.

4. Результаты численных экспериментов. Предобусловливание проводилось для параллельных версий методов сопряженных градиентов и бисопряженных стабилизированных градиентов, выполняемых на графическом ускорителе. Параллельные алгоритмы тестировались на GPU-ускорителе GeForce GTX 780 (графическая память 3 ГБ) и на восьми ядрах CPU (два четырехъядерных процессора Intel Xeon E5-2609, 2.4 ГГц; 64 ГБ оперативной памяти).

В численных экспериментах были использованы матрицы из коллекции The University of Florida Sparse Matrix Collection. Решались системы уравнений $Ay = f$ с известным точным решением $y = [1, 1, \dots, 1]$, матрицы которых хранились в сжатом строчном формате (CSR). В качестве начального приближения выбиралось $y_0 = [0, 0, \dots, 0]$, а критерий сходимости — $\|r_i\| \leq 10^{-6}\|r_0\|$, где $r_i = f - Ay_i$.

В расчетах рассматривался один из вариантов представления матрицы $W = \beta \text{diag}(A)$, хотя возможны и другие, например $W = \beta I$. Выбор стратегии фильтрации по значениям элементов и предельных значений τ основывался на рекомендациях, приведенных в работе [6]. Отметим, что затраты на формирование возрастают не существенно при уменьшении порогового значения. Выбор оптимального значения параметра τ для рассматриваемых матриц заключался в минимизации числа итераций и времени решения. Точность фильтрации для матриц \tilde{S} и \tilde{T} выбиралась $\tau_u = \tau_v = 0.01; 0.0001, \beta = 100$.

Таблица 2

Вычислительные затраты предобусловливателей при решении систем с несимметричными матрицами, $t_p/t_{its}(its)$

Матрица A	P_{DIAG} GPU/GPU	$P_{\text{ILU}(0)}$ CPU/GPU	$P_{\text{ILU}(1)}$ CPU/GPU	P_{AISM} GPU/GPU	
				$\tau = 0.01$	$\tau = 0.0001$
cdde5	$2 \times 10^{-4}/0.38(737)$	0.0002/0.13 (140)	0.002/0.12 (106)	0.56/0.13 (179)	0.57/0.13(167)
ex37	$2 \times 10^{-4}/0.008(13)$	0.003/0.03 (3)	1.4/0.03 (2)	15.66/0.004 (5)	15.44/0.004 (4)
rajat03	—	—	—	169/0.07 (84)	169/0.11 (135)
flowmeter5	$3 \times 10^{-4}/0.25 (450)$	0.002/0.36 (63)	0.04/0.34 (32)	357.6/0.11 (120)	357.9/0.15 (112)
ex19	$4 \times 10^{-4}/0.28 (404)$	0.04/—	699/0.5 (279)	700/0.35 (132)	703/0.39 (128)
sme3Da	—	0.15/13.23(1700)	495/16.61(158)	814/11.83(2032)	843/13.5(1338)
poisson3Da	$4 \times 10^{-4}/0.062(87)$	0.04/0.13 (24)	56.3/0.58 (11)	1049/0.05 (28)	1066/0.24 (30)

Сравним сначала результаты, полученные при решении несимметричных систем. В таблицах приведены затраты на построение предобусловливателей неполного разложения с контролем заполнения $\text{ILU}(p)$ (рассматривались варианты $p = 0$ и $p = 1$) и на основе явного вычисления приближенной обратной матрицы FSAI, AINV, TNS в реализации пакета PARALUTION (<http://www.paralution.com/>) на центральных процессорах и графических ускорителях.

При проведении численных экспериментов матрицы, свойства которых приведены в табл. 1, хранились в сжатом формате CSR. Предобусловливатели $\text{ILU}(p)$ формировались на центральном процессоре, а итерационный процесс BiCGStab — на графическом ускорителе вычислений. В табл. 2 приведены времена формирования предобусловливателя (t_p) и итерационного процесса (t_{its}) и число итераций (its), необходимых для решения систем линейных уравнений с использованием неполных LU-разложений и рассматриваемого алгоритма AISM.

При рассмотрении плохо обусловленных матриц большего размера (ex19, sme3Da) затраты на формирование предобусловливателя P_{AISM} сравнимы с вариантом $P_{\text{ILU}(1)}$. Как видно из табл. 2, в ряде случаев не только диагональный предобусловливатель, но и предобусловливатели на основе неполного разложения не обеспечили сходимость решения. Так, использование $\text{ILU}(0)$ в случае матрицы ex19 не привело к решению системы, а при решении системы с матрицей rajat03 из приведенных предобусловливателей только алгоритм AISM обеспечил сходимость к точному решению.

По времени работы алгоритмов BiCGStab решения систем видны преимущества предобусловливателя P_{AISM} . Затраты на одну итерацию в этом случае существенно ниже, чем при $ILU(p)$. Достигнутое ускорение при формировании явного предобусловливателя P_{AISM} все-таки требует достаточно больших вычислительных затрат и дальнейших исследований.

В рамках одной итерации цикла алгоритма 2 при вычислении столбцов матриц s_k и t_k не возникает ситуации блокировки памяти, что позволяет выполнять операции матрично-векторного и скалярных произведений (строки 7, 13) независимо в параллельных нитях. Такой подход возможен при реализации вычислений на нескольких GPU. В этом случае каждая последующая итерация цикла зависит от данных, полученных на предыдущем шаге, и требуется выполнение обмена векторами s_k и t_k между памятью различных GPU.

Таблица 3

Вычислительные затраты явных предобусловливателей при решении систем с симметричными матрицами, t_p/t_{its} (its)

Матрица A	P_{DIAG} GPU/GPU	$P_{ILU(0)}$ CPU/GPU	$P_{ILU(1)}$ CPU/GPU	P_{TNS} GPU/GPU	$P_{AISM} (\tau = 0.0001)$ GPU/GPU
nasa2910	$10^{-4}/0.556$ (1039)	2.5/0.65 (314)	19.5/0.04 (129)	0.002/0.32 (962)	8.78/0.62 (387)
bcsstk15	$10^{-4}/0.095$ (166)	0.15/0.6 (293)	1.47/0.03 (109)	0.002/0.08 (259)	20.37/0.17 (81)
Kuu	$10^{-4}/0.16$ (263)	4.03/0.16 (75)	11.7/0.02 (45)	0.004/0.1 (241)	142.03/0.18 (103)
mcs10848	$10^{-4}/1.19$ (1693)	1.48/2.68 (1190)	1168/0.02 (35)	0.006/14.7 (21871)	505.5/5.12 (846)
vibrobox	$10^{-4}/0.084$ (121)	1.29/1.42 (683)	85/0.05 (92)	0.003/1.98 (4682)	814/0.81 (52)

Исключительно для целей тестирования в табл. 3 приведены результаты для диагонального DIAG и явных предобусловливателей AINV, FSAI и TNS при решении симметричных систем уравнений методом сопряженных градиентов с формированием матрицы предобусловливателя на центральных процессорах и графических ускорителях. В этом случае симметричность матриц при формировании предобусловливателя P_{AISM} не учитывалась.

По скорости сходимости результаты многих тестов для различных предобусловливателей сопоставимы, а для матриц bcsstk15 и vibrobox применение P_{AISM} позволило получить меньшее число итераций. Затраты на решение систем уравнений с предобусловливателями P_{AISM} и P_{AINF} примерно одинаковы.

Потенциально сокращение затрат на формирование P_{AISM} для случая симметричных матриц представляется возможным и перспективным. Так, для достаточно большой и заполненной матрицы mcs10848 затраты на построение предобусловливателя P_{FSAI} в два раза превышают время формирования по рассматриваемому алгоритму.

Предобусловливатель P_{AISM} , основанный на рекурсивном обращении Шермана–Моррисона, имеет высокий потенциал сокращения арифметических операций и возможностей дальнейшего повышения параллельной эффективности его формирования. Прежде всего, это связано с блочным представлением алгоритма, с выделением крупноблочной декомпозиции матриц, построением предобусловливателя на нескольких ускорителях вычислений [9] и сокращением арифметических операций в случае симметричных матриц.

Рассмотренный подход особенно важен при наличии широких возможностей распараллеливания вычислений, в обычных же условиях затраты могут оказаться существенными и могут привести к нехватке ресурсов для формирования качественного предобусловливателя.

Программная реализация итерационных методов сопряженных и бисопряженных градиентов с параллельным предобусловливателем AISM, выполняемых на графических ускорителях, была успешно интегрирована в программные комплексы FEStudio [10] и OpenFOAM (<http://www.openfoam.com>).

Работа выполнена при финансовой поддержке РФФИ (проекты 14–01–00055-а и 14–01–31066 мол_а) и программы фундаментальных исследований УрО РАН (проект 15–7–1–11).

Статья рекомендована к публикации Программным комитетом Международной научной конференции “Параллельные вычислительные технологии” (ПаВТ-2015; <http://agora.guru.ru/pavt2015>).

СПИСОК ЛИТЕРАТУРЫ

1. Saad Y. Iterative methods for sparse linear systems. Philadelphia: SIAM Press, 2003.
2. Li R., Saad Y. GPU-accelerated preconditioned iterative linear solvers // Journal of Supercomputing. 2013. **63**, N 2. 443–466.

3. Benzi M. Preconditioning techniques for large linear systems: a survey // *Journal of Computational Physics*. 2002. **182**, N 2. 418–477.
4. Kolotilina L. Yu., Yerebin A. Yu. Factorized sparse approximate inverse preconditionings I: theory // *SIAM J. Matrix Anal. Appl.* 1993. **14**, N 1. 45–58.
5. Grote M.J., Huckle T. Parallel preconditioning with sparse approximate inverses // *SIAM J. Sci. Comput.* 1997. **18**, N 3. 838–853.
6. Bru R., Cerdán J., Marín J., Mas J. Preconditioning sparse nonsymmetric linear systems with the Sherman–Morrison formula // *SIAM J. Sci. Comput.* 2003. **25**, N 2. 701–715.
7. Sherman J., Morrison W.J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix // *Ann. Math. Statistics*. 1950. **21**, N 1. 124–127.
8. Недождогин Н.С., Сармакеева А.С., Копысов С.П. Высокопроизводительный алгоритм Шермана–Моррисона обращения матриц на GPU // *Вестник ЮУрГУ. Серия “Вычислительная математика и информатика”*. 2014. **3**, № 2. 101–108.
9. Kopysov S., Kuzmin I., Nedozhogin N., Novikov A., Sagdeeva Yu. Scalable hybrid implementation of the Schur complement method for multi-GPU systems // *Journal of Supercomputing*. 2014. **69**, Issue 1. 81–88.
10. Копысов С.П., Кузьмин И.М., Недождогин Н.С., Новиков А.К., Рычков В.Н., Сагдеева Ю.А., Тонков Л.Е. Параллельная реализация конечно-элементных алгоритмов на графических ускорителях в программном комплексе FESstudio // *Компьютерные исследования и моделирование*. 2014. **6**, № 1. 79–97.

Поступила в редакцию
24.01.2015

Parallel Forming of Preconditioners Based on the Approximation of the Sherman–Morrison Inversion Formula

N. S. Nedozhogin¹, S. P. Kopysov², and A. K. Novikov³

¹ *Institute of Mechanics, Ural Branch of Russian Academy of Sciences; ulitsa Baramzinoi 34, Izhevsk, 426067, Russia; Junior Scientist, e-mail: Nedozhogin@inbox.ru*

² *Institute of Mechanics, Ural Branch of Russian Academy of Sciences; ulitsa Baramzinoi 34, Izhevsk, 426067, Russia; Dr. Sci, Professor, Head of Laboratory, e-mail: s.kopysov@gmail.com*

³ *Institute of Mechanics, Ural Branch of Russian Academy of Sciences; ulitsa Baramzinoi 34, Izhevsk, 426067, Russia; Ph.D., Associate Professor, e-mail: sc_work@mail.ru*

Received January 24, 2015

Abstract: Acceleration of preconditioned bi-conjugate gradient stabilized (BiCGStab) methods with preconditioners based on the matrix approximation by the Sherman–Morrison inversion formula is studied. A new form of the parallel algorithm using matrix-vector products to generate preconditioning matrices is proposed. A parallelization efficiency of the most resource-intensive operations of such preconditioners on multi-core central and graphics processing units (CPUs and GPUs) is shown.

Keywords: linear systems, explicit preconditioning, Sherman–Morrison formula, parallel computing, graphics accelerators.

References

1. Y. Saad, *Iterative Methods for Sparse Linear Systems* (SIAM, Philadelphia, 2003).
2. R. Li and Y. Saad, “GPU-Accelerated Preconditioned Iterative Linear Solvers,” *J. Supercomput.* **63** (2), 443–466 (2013).
3. M. Benzi, “Preconditioning Techniques for Large Linear Systems: A Survey,” *J. Comput. Phys.* **182** (2), 418–477 (2002).
4. L. Yu. Kolotilina and A. Yu. Yerebin, “Factorized Sparse Approximate Inverse Preconditionings I: Theory,” *SIAM J. Matrix Anal. Appl.* **14** (1), 45–58 (1993).
5. M. J. Grote and T. Huckle, “Parallel Preconditioning with Sparse Approximate Inverses,” *SIAM J. Sci. Comput.* **18** (3), 838–853 (1997).
6. R. Bru, J. Cerdán, J. Marín, and J. Mas, “Preconditioning Sparse Nonsymmetric Linear Systems with the Sherman–Morrison Formula,” *SIAM J. Sci. Comput.* **25** (2), 701–715 (2003).

7. J. Sherman and W. J. Morrison, "Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix," *Ann. Math. Stat.* **21** (1), 124–127 (1950).
8. N. S. Nedozhogin, A. S. Sarmakeeva, and S. P. Kopysov, "Sherman–Morrison High-Performance Algorithm for Matrix Inversion on GPU," *Vestn. South Ural Univ., Ser.: Vychisl. Mat. Inform.* **3** (2), 101–108 (2014).
9. S. Kopysov, I. Kuzmin, N. Nedozhogin, et al., "Scalable Hybrid Implementation of the Schur Complement Method for Multi-GPU Systems," *J. Supercomput.* **69** (1), 81–88 (2014).
10. S. P. Kopysov, I. M. Kuzmin, N. S. Nedozhogin, et al., "Parallel Implementation of a Finite-Element Algorithms on a Graphics Accelerator in the Software Package FEStudio," *Komp'yut. Issled. Model.* **6** (1), 79–97 (2014).