12. Willem M. Minimax theorems / M. Willem. — Boston : Birkhduser, 1996. — 162 p.

The article deals with infinite systems of differential equations that describe infinite system of nonlinear coupled nonlinear oscillators on 2D-lattice. It is obtained result on existence of supersonic periodic travelling waves.

Key words: *nonlinear oscillators, 2D-lattice, supersonic periodic travelling waves, critical points, mountain pass theorem.*

Отримано: 16.04.2015

УДК 519.6

А. Ю. Баранов, аспірант

Інститут кібернетики ім. В. М. Глушкова НАН України, м. Київ

АЛГОРИТМ ФАКТОРИЗАЦІЇ СТРІЧКОВИХ НЕСИМЕТРИЧНИХ МАТРИЦЬ НА КОМП'ЮТЕРАХ З ГРАФІЧНИМИ ПРИСКОРЮВАЧАМИ

Розроблено і досліджено алгоритм факторизації стрічкових несиметричних матриць на гібридних комп'ютерах. Розглянуто питання програмної реалізації алгоритму на комп'ютерах з графічними процесорами.

Ключові слова: *паралельні обчислення, CUDA, гібридний алгоритм, СЛАР, стрічкові матриці, метод Гауса.*

Вступ. При чисельному розв'язанні задач в багатьох випадках виникає необхідність розв'язувати систему лінійних алгебраїчних рівнянь (СЛАР). Наприклад, задачі лінійної алгебри виникають при дискретизації крайових задач чи задач на власні значення проекційно-різницевим методом (скінченних елементів, скінченних різниць). Також, при використанні ітераційних методів розв'язання нелінійних задач часто на кожній ітерації розв'язується лінеаризована задача — СЛАР.

Важливою особливістю задач лінійної алгебри, які виникають при дискретизації, являється те, що кількість ненульових елементів матриць таких задач складає kn, де $k << n$, а n — порядок матриці, тобто матриці є розрідженими [1]. Структура розрідженої матриці визначається нумерацією невідомих задачі і часто є стрічковою, блочно-діагональною з обрамленням, профільною і тому подібне. В даній статті розглядуються несиметричні матриці стрічкової структури.

Іншою важливою особливістю є великий порядок матриць СЛАР — до десятків мільйонів. Це зумовлюється бажанням використовувати більш точні дискретні моделі, що дає можливість отримувати наближені розв'язки більш близькі до розв'язків вихідних задач, враховувати локальні особливості даного процесу чи явища.

На даний час відомі паралельні алгоритми факторизації стрічкових матриць на багатоядерних комп'ютерах[8; 9]. В даній статті розглядається новий алгоритм факторизації стрічкових несиметричних матриць на гібридних системах — комп'ютерах з багатоядерними процесорами та графічними прискорювачами.

Постановка задачі. Розглянемо систему лінійних алгебраїчних рівнянь:

$$Ax = b, \tag{1}$$

де матриця A — стрічкова несиметрична, n — порядок матриці A, k_1 — кількість нижніх діагоналей, k_2 — кількість верхніх діагоналей.

Найбільш ефективним прямим методом розв'язання такої задачі є, як відомо, метод Гауса [2]. Розв'язання системи (1) полягає в розв'язанні підзадач: трикутне розвинення матриці системи (2), розв'язання двох СЛАР з трикутними матрицями (3) та (4):

$$A = PLU. \tag{2}$$

$$Ly = b. \tag{3}$$

$$Ux = y. \tag{4}$$

Обчислювальна складність розв'язання задачі (1) визначається підзадачою (2). Тому надалі будемо розглядати задачу трикутного розвинення матриці СЛАР (2).

Послідовний алгоритм факторизації. У роботі [3] показано, що кількість верхніх діагоналей матриці U дорівнює $k1+k2$. Тому, для зручності опису алгоритму, будемо вважати, що матриця A задана з $k1+k2$ верхніми діагоналями. Нехай порядок стрічкової несиметричної матриці A дорівнює $n = pb$, $k1 = lb, k1+k2 = ub$, де b це деяке наперед задане натуральне число. Розіб'ємо матрицю A на квадратні блоки порядку b:

$$\begin{pmatrix}
A_{11} & A_{12} & A_{13} & \cdots & A_{1(1+u)} & 0 & 0 & \cdots & 0 \\
A_{21} & A_{22} & A_{23} & \cdots & A_{2(1+u)} & A_{2(2+u)} & 0 & \cdots & 0 \\
A_{31} & A_{32} & \ddots & & & & & & \\
\vdots & \vdots & & & & & & & \\
A_{(l+1)1} & A_{(l+1)2} & & & & & & & \\
0 & A_{(l+2)2} & & & & & & & \\
0 & 0 & & & & & \vdots & \vdots & \\
\vdots & \vdots & & & & & \cdots & A_{(p-1)(p-1)} & A_{(p-1)p} \\
0 & 0 & & & & & \cdots & A_{p(p-1)} & A_{pp}
\end{pmatrix}.$$

Перейдемо до опису алгоритму. Для кожного $i = \overline{1, p}$ виконати наступні кроки:

1. Виконати факторизацію прямокутної матриці, що складається з блоків $A_{ii}, A_{(i+1)i}, A_{(i+2)i}, \ldots, A_{qi}$, де $q = \min(i + l, p)$, використовуючи алгоритм Гауса для щільних матриць:
$$\begin{pmatrix} A_{ii} \\ \vdots \\ A_{qi} \end{pmatrix} = P_i \begin{pmatrix} L_{ii} \\ \vdots \\ L_{qi} \end{pmatrix} U_{ii}. \quad (5)$$
2. Виконати перестановку рядків використовуючи матрицю P_i.
3. Знайти матриці $U_{i(i+1)}, U_{i(i+2)}, \ldots, U_{ir}$, де $r = \min(i + u, p)$ за формулою:
$$\begin{pmatrix} U_{i(i+1)} & \cdots & U_{ir} \end{pmatrix} = L_{ii}^{-1} \begin{pmatrix} A_{i(i+1)} & \cdots & A_{ir} \end{pmatrix}. \quad (6)$$
4. Оновити матриці A_{jt}, де $j = \overline{i+1, q}$ та $t = \overline{i+1, r}$ за формулою:
$$A_{jt} \leftarrow A_{jt} - L_{ji} U_{it}. \quad (7)$$

Після виконання всіх кроків алгоритму отримуємо трикутне розвинення вихідної матриці.

Гібридний алгоритм факторизації. Розглянемо алгоритм факторизації несиметричної стрічкової матриці для гібридних комп'ютерів з одним графічним прискорювачем. Такий алгоритм потребує копіювання даних з пам'яті CPU до пам'яті GPU та навпаки. Сучасні GPU мають можливість одночасно виконувати копіювання даних та операції обчислення. Також, об'єм глобальної пам'яті GPU є меншим за пам'ять CPU. Тому, доцільним є створення гібридних алгоритмів, які будуть оптимально використовувати пам'ять GPU, використовуватимуть можливість одночасно виконувати обчислення та копіювання даних.

Зрозуміло, що на i-му кроці послідовного алгоритму факторизації нам потрібні тільки блоки, які знаходяться в i-му стовпчику, та в u стовпчиках, що слідують за ним. При цьому, в кожному стовпчику знаходиться щонайбільше $u + l + 1$ блоків. Таким чином, на кожному кроці гібридного алгоритму, достатньо зберігати в GPU тільки ці блоки. Перейдемо до опису гібридного алгоритму факторизації. Блоки матриці A, що знаходяться на GPU, будемо позначати використовуючи верхній індекс d. Тоді гібридний алгоритм факторизації складається з таких кроків:

1. Копіювання блоків матриці A, що знаходяться в перших $u + 1$ стовпчиках, в пам'ять GPU.

2. Для кожного $i = \overline{1, p}$ виконати наступні кроки:
   - якщо $i > 1$, виконати копіювання блоків A_{ii}, $A_{(i+1)i}$, $A_{(i+2)i}$, …, $A_{q,i}$ в пам'ять CPU;
   - трикутне розвинення матриці методом Гауса, утвореної блоками $A_{ii}, A_{(i+1)i}, A_{(i+2)i}, \ldots, A_{q,i}$, за формулою (5);
   - копіювання блоків $L_{ii}, \ldots, L_{qi}$, U_{ii}, P_i в пам'ять GPU;
   - виконати перестановку рядків використовуючи матрицю P_i^d;
   - виконати обчислення блоків $U^d_{i(i+1)}, U^d_{i(i+2)}, \ldots, U^d_{ir}$, де $r = \min(i+u, p)$, за формулою:

$$\left(U^d_{i(i+1)} \quad \cdots \quad U^d_{ir}\right) = \left(L^d_{ii}\right)^{-1} \left(A^d_{i(i+1)} \quad \cdots \quad A^d_{ir}\right) \tag{8}$$

   - модифікувати блоки A^d_{jt}, де $j = \overline{i+1, q}$ та $t = \overline{i+1, r}$ за формулою:

$$A^d_{jt} \leftarrow A^d_{jt} - L^d_{ji} U^d_{it} \tag{9}$$

   - копіювання блоків $U^d_{i(i+1)}, U^d_{i(i+2)}, \ldots, U^d_{ir}$ в пам'ять CPU;
   - якщо $i + u + 2 < p$, виконати копіювання блоків матриці в стовбчику $i + u + 2$ в пам'ять GPU.

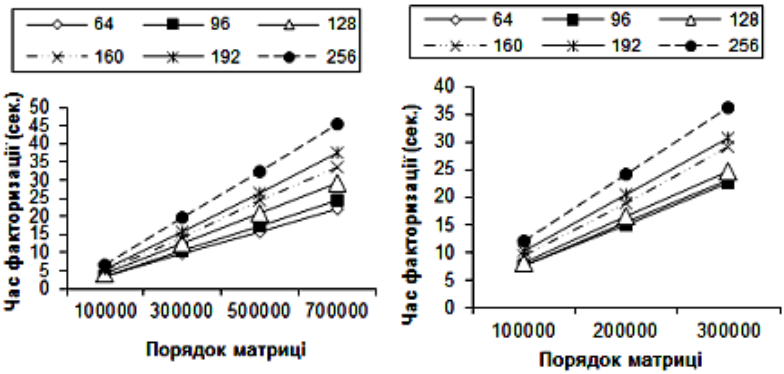
Деякі аспекти реалізації гібридного алгоритму. При реалізації гібридного алгоритму, доцільно використовувати функції з бібліотеки cuBLAS [6]. Для обрахунків за формулою (8) доцільно використовувати функцію cublasDtrsm, а для формули (9) використовуємо cublasDgemm. Для факторизації щільної прямокутної матриці на CPU за формулою (5) доцільно використовувати високопродуктивну функцію з бібліотеки Intel MKL [5].

На i-му кроці, оновлення блоків зі стовпчика $i+1$, за формулою (9), слід виконувати в спеціально виділеному потоці cudaStream_t. Такий підхід надає можливість на $i+1$-му кроці робити копіювання блоків в пам'ять CPU одночасно з модифікацією блоків за формулою (9) на i-му кроці. Таким чином, копіювання блоків на CPU, їх факторизації та копіювання в пам'ять GPU буде виконуватись одночасно з оновленням блоків матриці.

Чисельні експерименти. Запропонований паралельний алгоритм реалізовано на комп'ютерах з гібридною (MIMD, SIMD) архітектурою: комп'ютері з графічними прискорювачами Інпарком (Tesla

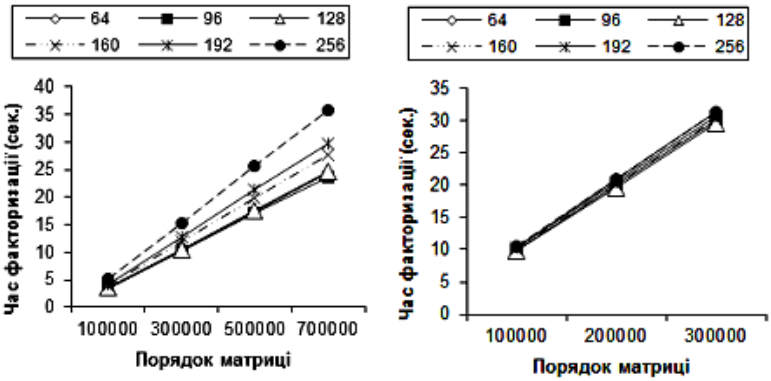
M2090, 2 CPU Intel(R) Xeon(R) E5606 @ 2.13GHz) [4], та СКІТ-4 (Tesla M2050, 4 CPU Intel(R) Xeon(R) X5675 @ 3.07GHz) [7].

На рис. 1 а) та рис. 1 б) подано залежність часу факторизації матриці з різною напівшириною стрічки (1000, 2000) від порядку блоку для гібридного алгоритму на комп'ютері Інпарком. На рис. 2 а) та рис. 2 б) подано залежність часу факторизації матриці з різною напівшириною стрічки (1000, 2000) від порядку блоку для гібридного алгоритму на комп'ютері СКІТ-4. З графіків видно, що оптимальний розмір ширини блоку відрізняється на різних комп'ютерах, а також залежить від напівширини стрічки.



а) *б)*

Рис. 1. *Залежність часу факторизації матриці від порядку блоку на комп'ютері Інпарком*



а) *б)*

Рис. 2. *Залежність часу факторизації матриці від порядку блоку на комп'ютері СКІТ-4*

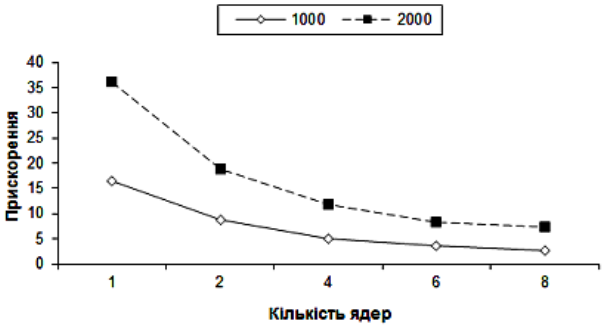


Рис. 3. *Порівняння часу факторизації матриці з використанням MKL та гібридного алгоритму на комп'ютері Інпарком*

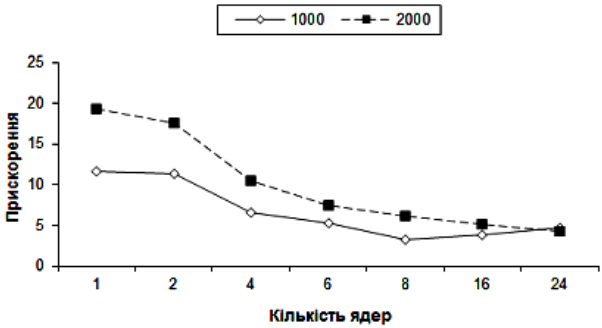


Рис. 4. *Порівняння часу факторизації матриці з використання MKL та гібридного алгоритму на комп'ютері СКІТ-4*

На рис. 3 подано прискорення часу факторизації матриці для різної напівширини стрічки (1000, 2000), використовуючи Intel MKL[5] та гібридний алгоритм на комп'ютері Інпарком. На рис. 4 подано порівняння часу факторизації матриці для різної напівширини стрічки (1000, 2000), використовуючи Intel MKL [5] та гібридний алгоритм на комп'ютері СКІТ-4. З графіків видно, що гібридний алгоритм дає значне прискорення в порівнянні з паралельним алгоритмом реалізованим в Intel MKL, який використовує лише CPU.

Висновки. Запропоновано алгоритм факторизації стрічкових несиметричних матриць, який забезпечує високу ефективність розпаралелювання на GPU, враховує структуру стрічкової матриці, оптимізує використання пам'яті GPU. Експериментально проведено дослідження вибору оптимального порядку блоків, на які розбивається вихідна матриця.

Подальші дослідження доцільно направити на розробку алгоритмів з використанням декількох CPU і декількох GPU як на системах з спільною так і з розподіленою пам'яттю.

Список використаних джерел:

1. Химич А. Н. Алгоритмы параллельных вычислений для задач линейной алгебры с матрицами нерегулярной структуры / А. Н. Химич, А. В. Попов, В. В. Полянко // Кибернетика и систем. анализ — 2011. — Вып. 47, № 6. — С. 159–174.
2. Уилкинсон Дж. Х. Справочник алгоритмов на языке Алгол. Линейная алгебра / Дж. Х. Уилкинсон, К. Райнш. — М. : Машиностроение, 1976. — 389 с.
3. Golub G. H. Van Loan. Matrix Computations / G. H. Golub, F. Charles. — (3rd ed.). — Baltimore, MD, USA : Johns Hopkins University Press, 1996.
4. Интеллектуальный персональный компьютер гибридной архитектуры / И. Н. Молчанов, А. Н. Химич, В. И. Мова, А. А. Николайчук // Искусственный интеллект. — 2012. — № 3. — С. 73–78.
5. Режим доступу: http://software.intel.com/en-us/intel-mkl.
6. Режим доступу: https://developer.nvidia.com/cuBLAS
7. Режим доступу: http://icybcluster.org.ua/
8. Dongarra J. J. Solving Banded Systems on a Parallel Processor / J. J. Dongarra, L. Johnsson // Parallel Computing, — 1987. — № 5. — P. 219–246.
9. Попов А. В. Про паралельні алгоритми факторизації розріджених матриць / А. В. Попов // Компьютерная математика : сб. науч. трудов. — 2013. — Вып. 2. — С. 115–124.

Algorithm for factorization of band nonsymmetric matrices using hybrid computers is developed and investigated. Implementation of algorithm on computers with graphics processing units is considered.

Key words: *parallel computation, CUDA, hybrid algorithm, SLE, band matrix, Gaussian elimination.*

Отримано: 22.04.2015