Vol. 6 No. 4 JOURNAL OF COMPUTATIONAL MATHEMATICS October 1988

FAST PARALLEL ALGORITHMS FOR COMPUTING GENERALIZED INVERSES A^+ AND A^+_{MN}

WANG GUO-RONG(王国荣) LU SEN-QUAN(陆森泉)
(Shanghai Teachers' University, Shanghai, China)

Abstract

The parallel arithmetic complexities for computing generalized inverse A^+, computing the minimum-norm least-squares solution of $Ax=b$, computing order $m+n-r$ determinants and finding the characteristic polynomials of order $m+n-r$ matrices are shown to have the same grawth rate. Algorithms are given that compute A^+ and A^+_{MN} in $O(\log r\cdot\log n+\log m)$ and $O(\log^2 n+\log m)$ steps using a number of processors which is a ploynomial in m, n and r ($A\in R_r^{m\times n}$, $r=\text{rank } A$).

§ 1. Introduction

Let $I(n)$, $E(n)$, $D(n)$, $P(n)$ denote the parallel arithmetic complexities of inverting order n matrices, solving a system of n linear equations in n unknowns, computing order n determinants and finding the characteristic polynomials of order n matrices respectively. Then Csanky gave an important theoretical result [1]:

Theorem 1. $I(n)=O(f(n))\Leftrightarrow E(n)=O(f(n))\Leftrightarrow D(n)=O(f(n))\Leftrightarrow P(n)=O(f(n))$.

He also gives algorithms that compute these problems in $O(\log^2 n)$ steps using a number of processors which is a polynomial in n (n is the order of the matrix of the problem).

Let $A\in R_r^{m\times n}$, $r=\text{rank } A$. In this paper, we give two parallel algorithms for computing A^+ and A^+_{MN} respectively. The one for A^+ is based on Decell's method in [2], and the one for A^+_{MN} is a generalization of Decell's method in [3].

The parallel arithmetic complexities for computing the generalized inverse A^+, computing the minimum-norm least-squares solution of $Ax=b$, computing order $m+n-r$ determinants and finding the characteristic polynomials of order $m+n-r$ matrices are shown to have the same growth rate.

§ 2. The Parallel Algorithm for Computing A^+

Let $A\in R_r^{m\times n}$. Then there is a unique matrix $X\in R_r^{n\times m}$ satisfying

$$AXA=A,\ XAX=X,\ (AX)^T=AX,\ (XA)^T=XA.$$

This X is called the M-P inverse of A and is denoted by $X=A^+$.

In [2], Decell gave a finite algorithm for computing A^+. We rewrite it as follows:

* Received March 21, 1987.

Algorithm 1. (1) Parallelly compute $B=A^TA$.

(2) Parallelly compute $B^k=(b_{ij}^{(k)})$, $k=1, 2, \cdots, r$.

(3) Let $\lambda_1, \lambda_2, \cdots, \lambda_n$ denote the roots of the characteristic polynomial $f(\lambda)$ of B. Let

$$s_k=\sum_{i=1}^{n}\lambda_i^k,\quad k=1, 2, \cdots, r.$$

Parallelly compute

$$s_k=\mathrm{tr}(B^k)=\sum_{i=1}^{n}b_{ii}^{(k)},\quad k=1, 2, \cdots, r.$$

(4) Let the characteristic polynomial $f(\lambda)$ of B be

$$f(\lambda)=\det(\lambda I-B)=\lambda^n+c_1\lambda^{n-1}+\cdots+c_n.$$

From the Newton formula

$$s_k+c_1s_{k-1}+c_2s_{k-2}+\cdots+c_{k-1}s_1+kc_k=0,\quad k\leqslant n$$

we have

$$\begin{bmatrix} 1 & & & & \\ s_1 & 2 & & & \\ s_2 & s_1 & 3 & & \\ \vdots & \vdots & & \ddots & \\ s_{r-1} & s_{r-2} & \cdots & s_1 & r \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_r \end{bmatrix}=-\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_r \end{bmatrix}.$$

Parallelly compute the solution of the above triangular system.

(5) Parallelly compute

$$A^+=-((A^TA)^{r-1}+c_1(A^TA)^{r-2}+\cdots+c_{r-1}I)A^T/c_r. \tag{2.1}$$

Theorem 2. *Let $A\in R_r^{m\times n}$, and $GI(m, n)$ denote the parallel arithmetic complexity for computing the* M-P *inverse A^+. Then*

$$GI(m, n)=\log r(\log n+7/2)+(1/2)\log^2 r+2\log n+\log m+4=0(f(m, n, r))$$

and the number of processors used in the algorithm is

$$cp=\begin{cases} n^3r/2, & m<nr/2, \\ mn^2, & m\geqslant nr/2. \end{cases}$$

Proof. (1) Parallel computation of $B=A^TA$ takes $T_1=\log m+1$ steps and $cp_1=mn^2$ processors.

(2) Parallel computation of B^k ($k=1, 2, \cdots, r$) takes $T_2=\log r(\log n+1)$ steps and $cp_2=n^3r/2$ processors.

(3) Parallel computation of s_k ($k=1, 2, \cdots, r$) takes $T_3=\log n$ steps and $cp_3=rn/2$ processors.

(4) Parallel computation of $c_k (k=1, 2, \cdots, r)$ takes $T_4=(1/2)\log^2 r+(3/2)\log r$ steps and $cp_4=O(r^3)$ processors.

(5) Since $B^2, \cdots, B^{r-1}$ are already available, parallel computation of A^+ takes $T_5=\log r+\log n+3$ steps and $cp_5=n^2m$ processors.

Thus

$$cp=\max_{1\leqslant i\leqslant 5} cp_i=\begin{cases} n^3r/2, & m<nr/2, \\ mn^2. & m\geqslant nr/2. \end{cases}$$

$$GI(m,\ n)=\sum_{i=1}^{5} T_i=\log r(\log n+7/2)+(1/2)\log^2 r+2\log n+\log m+4.$$

§ 3. The Parallel Algorithm for Computing A^+_{MN}

Let $A\in R_r^{m\times n}$, and M and N be positive definite matrices of order m and n respectively. Then there is a unique matrix $X\in R_r^{n\times m}$ satisfying

$$AXA=A,\ XAX=X,\ (MAX)^T=MAX,\ (NXA)^T=NXA.$$

This X is called the weighted M-P inverse of A, and is denoted by $X=A^+_{MN}$.

In [3], Wang gave a finite algorithm for computing A^+_{MN}. We rewrite it as follows:

Let $\tilde{A}=M^{1/2}AN^{-1/2}$, and the characteristic polynomial of $\tilde{A}^T\tilde{A}$ be

$$h(\lambda)=\det(\lambda I-\tilde{A}^T\tilde{A})=\lambda^n+a_1\lambda^{n-1}+\cdots+a_n.$$

From section 2,

$$(M^{1/2}AN^{-1/2})^+=\tilde{A}^+=-((\tilde{A}^T\tilde{A})^{r-1}+a_1(\tilde{A}^T\tilde{A})^{r-2}+\cdots+a_{r-1}I)\tilde{A}^T/a_r.$$

Hence, from [4] we have

$$\begin{aligned}A^+_{MN}&=N^{-1/2}(M^{1/2}AN^{-1/2})^+M^{1/2}\\&=-((N^{-1}A^TMA)^{r-1}+a_1(N^{-1}A^TMA)^{r-2}+\cdots+a_{r-1}I)N^{-1}A^TM/a_r.\end{aligned}$$

Let $A^*=N^{-1}A^TM$. Then

$$A^+_{MN}=-((A^*A)^{r-1}+a_1(A^*A)^{r-2}+\cdots+a_{r-1}I)A^*/a_r. \tag{3.1}$$

Algorithm 2. (1) Parallelly compute N^{-1}.

(2) Parallelly compute $A^*=N^{-1}A^TM$ and $B=A^*A=(N^{-1}A^T)(MA)$.

(3) Parallelly compute $B^k=(b_{ij}^{(k)})$, $k=1,\ 2,\ \cdots,\ r$.

(4) Parallelly compute $s_k=\mathrm{tr}(B^k)=\sum_{i=1}^{n} b_{ii}^{(k)}$, $k=1,\ 2,\ \cdots,\ r$.

(5) Parallelly compute a_k $(k=1,\ 2,\ \cdots,\ r)$, from the following triangular system

$$\begin{bmatrix}1 & & & & \\ s_1 & 2 & & & \\ s_2 & s_1 & 3 & & \\ \vdots & \vdots & & \ddots & \\ s_{r-1} & s_{r-2} & \cdots & s_1 & r\end{bmatrix}\begin{bmatrix}a_1\\a_2\\ \vdots\\ a_r\end{bmatrix}=-\begin{bmatrix}s_1\\s_2\\ \vdots\\ s_r\end{bmatrix}.$$

(6) Parallelly compute A^+_{MN} from (3.1).

Theorem 3. *Let $A\in R_r^{m\times n}$, and M and N be positive definite matrices of order m and n respectively. $WGI(m,\ n)$ denotes the parallel arithmetic complexity for computing the weighted* M–P *inverse A^+_{MN}. Then*

$$WGI(m,\ n)=GI(m,\ n)+\log m+(3/2)\log^2 n+(11/2)\log n+4$$

and the number of processors used in the algorithm is

$$cp=\begin{cases}n^4/2, & m\leqslant(n/2)(\sqrt{1+2n}-1),\\ m^2n+mn^2, & m>(n/2)(\sqrt{1+2n}-1).\end{cases}$$

Proof. (1) From [1], parallel computation of N^{-1} takes $T_1=(3/2)\log^2 n+(11/2)\log n+3$ steps and $cp_1=n^4/2$ processors.

(2) Parallel computation of $A^\#$ and B. First, parallel computation of $N^{-1}A^T$ and MA takes $1+\log m$ steps and m^2n+mn^2 processors; then parallelly computing $A^\#=(N^{-1}A^T)M$ and $B=(N^{-1}A^T)(MA)$. Thus it takes $T_2=2(1+\log m)$ steps and $cp_2=m^2n+mn^2$ processors.

(3) Parallel computation of $B^k(k=1, 2, \cdots, r)$ takes $T_3=\log r(1+\log n)$ steps and $cp_3=n^3r/2$ processors.

(4) Parallel computation of s_k $(k=1, 2, \cdots, r)$ takes $T_4=\log n$ steps and $cp_4=rn/2$ processors.

(5) Parallel computation of $a_k(k=1, 2, \cdots, r)$ takes $T_5=(1/2)\log^2 r+(3/2)\log r$ steps and $cp_5=O(r^3)$ processors.

(6) Parallel computation of A^+_{MN} takes $T_6=\log r+\log n+3$ steps and $cp_6=n^2m$ processors.

Thus

$$cp=\max_{1\le i\le 6} cp_i=\begin{cases} n^4/2, & m\le n(\sqrt{1+2n}-1)/2, \\ m^2n+mn^2, & m>n(\sqrt{1+2n}-1)/2 \end{cases}$$

and

$$WGI(m, n)=\sum_{i=1}^{6} T_i=GI(m, n)+(3/2)\log^2 n+(11/2)\log n+\log m+4.$$

§ 4. Equivalence Theorem

First of all, we give some preliminaries.

Lemma 1. *Let $A\in R_r^{m\times n}$, $U\in R_{m-r}^{m\times(m-r)}$ and $V\in R_{n-r}^{n\times(n-r)}$ be matrices whose columns form bases for $N(A^*)$ and $N(A)$ respectively. Then*

$$\mathscr{A}=\begin{pmatrix} A & U \\ V^* & O \end{pmatrix}$$

is nonsingular and

$$\mathscr{A}^{-1}=\begin{pmatrix} A^+ & V^{*+} \\ U^+ & O \end{pmatrix}. \tag{4.1}$$

$\mathscr{A}$ is called a generalized matrix (but not unique) of A. If A is nonsingular, we adopt the convention

$$\mathscr{A}=A.$$

Let adj $\mathscr{A}$ be the common adjoint matrix of $\mathscr{A}$. An $n\times m$ submatrix that lies in the upper left-hand corner of adj $\mathscr{A}$ is called a generalized adjoint matrix of A, and is denoted by Adj $\mathscr{A}$. If A is nonsingular, we adopt the convention

$$\mathrm{Adj}\,\mathscr{A}=\mathrm{adj}\,A.$$

Corollary 1.

$$A^+=\mathrm{Adj}\,\mathscr{A}/\det\mathscr{A}. \tag{4.2}$$

In [5], Noble gave a method of computing bases for $N(A^*)$ and $N(A)$.

Definition. *A matrix $H \in R^{n\times n}$ is said to be in Hermite echelon form if its elements h_{ij} satisfy the following conditions:*

(1) $h_{ij}=0,\ i>j$.

(2) h_{ii} *is either* 0 *or* 1.

(3) *If* $h_{ii}=0$, *then* $h_{ik}=0$ *for every* k, $1\leqslant k\leqslant n$.

(4) *If* $h_{ii}=1$, *then* $h_{ki}=0$ *for every* $k\neq i$.

For a given matrix $A\in R^{n\times n}$, the Hermite form H_A obtained by row reducing A is unique; $N(A)=N(H_A)=R(I-H_A)$ and a basis for $N(A)$ is the set of non-zero columns of $I-H_A$.

Algorithm 3. Let $A\in R_r^{n\times n}$, this algorithm computes a generalized matrix of A.

(1) Row reduce A to its Hermite form H_A.

(2) Form $I-H_A$, and select the non-zero columns $v_1, v_2, \cdots, v_{n-r}$ from this matrix, $V=(v_1, v_2, \cdots, v_{n-r})$.

(3) Row reduce A^* to its Hermite form H_{A^*}.

(4) Form $I-H_{A^*}$, and select the non-zero columns $u_1, u_2, \cdots, u_{n-r}$ from this matrix, $U=(u_1, u_2, \cdots, u_{n-r})$.

(5) Form nonsingular matrix

$$\mathscr{A}=\begin{bmatrix} A & U \\ V^* & 0\end{bmatrix}.$$

Although Algorithm 3 is stated for square matrices, it is easy to use it for non-square ones. Add zero rows or zero columns to construct a square matrix and use the fact that

$$[A \,\vdots\, 0]^+=\begin{bmatrix} A^+ \\ \cdots \\ 0^*\end{bmatrix} \text{ and } \begin{bmatrix} A \\ \cdots \\ 0\end{bmatrix}^+=[A^+ \,\vdots\, 0^*].$$

Let $F(U, V)$ denote the parallel arithmetic complexity for computing the submatrices U and V in generalized matrix $\mathscr{A}$ of A.

Lemma 2. *Let the number of processors used in Algorithm* 3 *be*

$$(n-1)(2n-j_1-j_1^*).$$

Then

$$4r\leqslant F(U, V)\leqslant 2(n+r),$$

where j_1 and j_1^* are the numbers of first nonzero column of A and A^* respectively, n is the order of A, and $r=\operatorname{rank} A$.

Let $GI(m, n)$ denote the parallel arithmetic complexity for computing A^+. From (4.2), there are $n\cdot m$ order $m+n-r-1$ determinants and one order $m+n-r$ determinant to be computed. They can be computed in parallel; hence we have

Corollary 2. $GI(m, n)=D(m+n-r)+F(U, V)+O(1)$.

Let $A(j\to y)$ denote the matrix obtained by replacing the j-th column of A by the vector y.

Lemma 3. *Let $A\in R_r^{m\times n}$, $b\in R^m$ and $b\overline{\in} R(A)$. Then the component x_j of the minimum-norm least-squares solution of inconsistent linear equations $Ax=b$ are*

$$x_j=\det \mathscr{A}(j\to \tilde{b})/\det \mathscr{A},\ j=1, 2, \cdots, n \tag{4.3}$$

where $\mathscr{A}$ *is a generalized matrix of* A, *and* $\tilde{b}=\begin{pmatrix} b \\ O \end{pmatrix}\in R^{m+n-r}$.

Proof. Let U and V be matrices whose columns form bases for $N(A^*)$ and $N(A)$. Since $x=A^+b\in R(A^+)=R(A^*)=N(A)^\perp$, we have

$$V^*x=0. \tag{4.4}$$

From $A^*=(AA^+A)^*=A^*(AA^+)$, we have

$$A^*(b-Ax)=0, \quad b-Ax\in N(A^*).$$

Set

$$b-Ax=Ul, \quad l\in R^{n-r}. \tag{4.5}$$

From (4.4)—(4.5), the minimum-norm least-squares solution of $Ax=b$ satisfies

$$\begin{pmatrix} A & U \\ V^* & 0 \end{pmatrix}\begin{pmatrix} x \\ l \end{pmatrix}=\begin{pmatrix} b \\ 0 \end{pmatrix}. \tag{4.6}$$

From Lemma 1, $\begin{pmatrix} A & U \\ V^* & 0 \end{pmatrix}$ is nonsingular, and (4.3) follows from the common Cramer's rule.

Let $GE(m, n)$ denote the parallel arithmetic complexity for computing the minimum-norm least-squares solution of the inconsistent linear equation $Ax=b$. From Lemma 4, there are $n+1$ order $m+n-r$ determinants to be computed. They can be computed in parallel; hence we have

Corollary 3. $GE(m, n)=D(m+n-r)+F(U, V)+O(1)$.

Let $GP(m, n)$ and $GD(m, n)$ denote the parallel arithmetic complexities of finding the characteristic polynomials of order $m+n-r$ matrix $\mathscr{A}$ and computing the determinant of order $m+n-r$ matrix $\mathscr{A}$. Then the following results are obvious.

Corollary 4. $GD(m, n)=D(m+n-r)+F(U, V)$,
$GP(m, n)=P(m+n-r)+F(U, V)$.

From Corollaries 2—4 and Theorems 1—2, we obtain the following important result immediately.

Theorem 4. $GI(m, n)=O(f(m, n, r))\Leftrightarrow GE(m, n)=O(f(m, n, r))$
$\Leftrightarrow GD(m, n)=O(f(m, n, r))\Leftrightarrow GP(m, n)=O(f(m, n, r))$.

In [6], Wang showed that the matrix

$$\tilde{\mathscr{A}}=\begin{pmatrix} A & M^{-1}U \\ V^*N & 0 \end{pmatrix}$$

is nonsingular, and

$$\tilde{\mathscr{A}}^{-1}=\begin{pmatrix} A & M^{-1}U \\ V^*N & 0 \end{pmatrix}^{-1}=\begin{pmatrix} A^+_{MN} & V(V^*NV)^{-1} \\ (U^*M^{-1}U)^{-1}U^* & 0 \end{pmatrix}.$$

The component x_j of the minimum-norm (N) least-squares (M) solution x of $Ax=b$ satisfies

$$x_j=\det\tilde{\mathscr{A}}(j\to\tilde{b})/\det\tilde{\mathscr{A}}.$$

If we use $\widehat{\mathscr{A}}$ instead of $\mathscr{A}$, then results similar to Corollaries 2—4 may be obtained, and from Theorems 1 and 3, we have the following important result immediately.

Theorem 5. $WGI(m, n)=O(g(m, n, r))\Leftrightarrow WGE(m, n)=O(g(m, n, r))$
$\Leftrightarrow WGD(m, n)=O(g(m, n, r))\Leftrightarrow WGP(m, n)=O(g(m, n, r))$.

References

[1] Csanky, L.: Fast parallel matrix inversion algorithm, *SIAM. J, Comput.*, **5**: 618—623, (1976).
[2] Decell, H. P. Jr: An application of the Caylay-Hamilton theorem to generalized matrix inversion, *SIAM Review*, **7**: 4 526—528, (1965).
[3] Wang Guo-rong: A Finite algorithm for computing the weighted Moore-Penrose inverse A^{+}_{MN}, *Applied Mathematics and Computation*, **23**: 4(1987), 277—289.
[4] Ben-Israel, A.; Greville, T. N. E.: Generalized Inverses: Theory and Applications, Wiley, New York, 1974.
[5] Noble, B.: Applied Linear Algebra, Prentice-Hall, New Jersey, 1969.
[6] Wang Guo-rong: A Cramer rule for minimum-norm (T) least-squares (S) solution of inconsistent linear equations, *Linear Algebra Appl.*, **74** (1986), 213—218.