FAST PARALLEL EXACT COMPUTATION OF THE GENERALIZED INVERSE AND ETC(U)
JUL 81 E V KRISHNAMURTHY
AFOSR-77-3271
UNCLASSIFIED TR-1079 AFOSR-TR-81-0659
END
FAST PARALLEL EXACT COMPUTATION OF THE GENERALIZED INVERSE AND RANK OF A MATRIX

Based on the fast parallel matrix multiplication scheme of Krishnamurthy and Klette, $O(\log m)$ step algorithms using $m$ matrix processors are described for the exact determination of the Moore-Penrose generalized inverse and the rank of an $(m \times m)$ matrix with integer entries.
ABSTRACT

Based on the fast parallel matrix multiplication scheme of Krishnamurthy and Klette, $O(\log m)$ step algorithms using $m$ matrix processors are described for the exact determination of the Moore-Penrose generalized inverse and the rank of an $(m \times m)$ matrix with integer entries.

The support of the U.S. Air Force Office of Scientific Research under Grant AFOSR-77-3271 is gratefully acknowledged, as is the help of Janet Salzman in preparing this paper.

*Permanent address: Indian Institute of Science, Bangalore, India
1. **Introduction**

In a recent paper, Krishnamurthy and Klette [1] have shown that the exact product of two \((m \times m)\) matrices having integer elements with \(e\)-bit precision can be obtained in the MIMD mode with complexity \(\mu = O(\log r \log e + (\log em)(\max_i \log e_i))\) using prime moduli arithmetic with \(r\) primes, each of precision \(e_i\) bits. (All logarithms are taken to base 2.)

Based on this parallel scheme for multiplication, we describe here a parallel method with \(m\) such matrix processors to determine exactly, in \(O(\log m)\) steps, the rank and the generalized inverse of a rectangular \((m \times n)\) matrix with integral entries. This can be extended to matrices with complex number entries. However, this procedure is in general invalid for the determination of the rank of a matrix over a finite field.

Since the rank of a matrix is very sensitive to errors in computation (especially with matrices which are ill-conditioned), throughout this paper our discussion will be confined to exact computational procedures using residue arithmetic. For the principles and practice of these procedures readers are referred to the papers by Krishnamurthy, Rao and Subramanian [2] and Krishnamurthy [3].
2. Principle

The computation of the $g$-inverse and rank of a rectangular matrix is based on the following theorem [2,4]:

Theorem. Let $A$ be any $m \times n$ matrix with real entries. Let $B(\lambda) = (\lambda^m + a_1\lambda^{m-1} + \ldots + a_m)$ be the characteristic polynomial of $B = AA^t$ ($A^t$ is the transpose of $A$). If $k \neq 0$ is the largest integer such that $a_k \neq 0$ then the Moore-Penrose inverse of $A$ is

$$ A^+ = -a_k^{-1} A^t [(AA^t)^{k-1} + \ldots + a_{k-1}I] $$ (1)

If $k=0$ is the largest integer such that $a_k \neq 0$, then $A^+ = 0$.

Also, the rank of $A$ is $k$.

Based on this theorem an algorithm is described in [2] for computing the exact generalized inverse of $A$ and its rank. In this paper we describe a parallel version of this algorithm using the processor model described in [1].

Computation of the rank and $g$-inverse proceeds through the following steps:

1. Computing $B = AA^t$

2. Finding the characteristic equation of $B$ and computing $A^+$ using (1). This can be done by computing the coefficients $a_k$ of $B(\lambda)$ using Leverrier's method [5], as shown below:

Let $\lambda_1, \lambda_2, \ldots, \lambda_m$ be the characteristic roots of $B$ and let

$$ S_k = \sum_{i=1}^{m} \lambda_i^k \quad 1 \leq k \leq m $$

then $S_k = \text{trace}(B^k)$ for $1 \leq k \leq m$
and

\[
\begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
S_1 & 2 & 0 & \ldots & 0 \\
S_2 & S_1 & 3 & \ldots & 0 \\
S_3 & S_2 & S_1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
S_{m-1} & S_{m-2} & \ldots & S_1 & m
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
\vdots \\
a_m
\end{bmatrix}
=
\begin{bmatrix}
S_1 \\
S_2 \\
\vdots \\
S_m
\end{bmatrix}
\]  
(2)

which in matrix form can be expressed as

\[ Mc = S \]

Equation (2) can be proved by using the well known Newton's identities [5].

The calculation of \( a_k \) from (2) and determination of \( A^+ \) proceeds as follows:

i) Computation of \( S_k \), \( 1 \leq k \leq m \):

This requires computation of the powers of \( B \), viz., \( B^2, B^3, \ldots, B^k, \ldots, B^m \), and the computation of the trace of each \( B^k \) \( (1 \leq k \leq m) \).

ii) Computation of \( M^{-1} \):

The determination of \( a_k \) and hence the rank \( k \) requires the computation of the inverse of the non-singular triangular matrix \( M \).

iii) Determination of \( M^{-1}S \)

iv) Determination of \( A^+ \) using (1).
3. **Complexity**

We now proceed to compute the complexity. For this purpose we choose as the unit of measurement, the matrix multiplication time $(\mu)$ and matrix addition time $(\alpha)$ each with complexity as defined in [1]. We also assume that by a processor we mean a single matrix processor which can multiply two matrices in time $\mu$ or add in time $\alpha$. The total complexity can therefore be computed in terms of the basic operations using these and the definitions in [1].

The computation of powers $B^k$ requires $\log m$ parallel steps with $m/2$ matrix processors each performing a matrix multiplication in time $\mu$. The computation of the trace requires $\log m$ steps of addition time for $m$ numbers in each $B^k$; this can be done with $m$ matrix processors in parallel.

For example, if $m=16$, with 8 processors the number of required steps is $\log_2 16 = 4$, and at each step the calculations are organized as follows:

<table>
<thead>
<tr>
<th>Time Step</th>
<th>Processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$B^2$</td>
</tr>
<tr>
<td>2</td>
<td>$B^4$ $B^3$</td>
</tr>
<tr>
<td>3</td>
<td>$B^8$ $B^7$ $B^6$ $B^5$</td>
</tr>
<tr>
<td>4</td>
<td>$B^{16}$ $B^{15}$ $B^{14}$ $B^{13}$ $B^{12}$ $B^{11}$ $B^{10}$ $B^9$</td>
</tr>
</tbody>
</table>

The computation of the inverse of the non-singular triangular matrix $M$ is carried out using a formula similar to (1). Since $M$ is a triangular matrix, its eigenvalues are simply the diagonal elements $1, 2, \ldots, m$; therefore, the coefficients of the characteristic polynomial of $M$, namely $M(\lambda) = \lambda^m + b_1 \lambda^{m-1} + b_2 \lambda^{m-2} + \ldots + b_m = 0$ (3)
can be precomputed once and for all and stored. Using (3) and the Cayley-Hamilton theorem, we can write

\[ M^{-1} = \frac{M^{m-1} + b_1 M^{m-2} + \ldots + b_{m-1} I}{b_m} \]

This can be computed in \(2 + \log m\) steps with \(m\) matrix processors each performing matrix multiplication in time \(\mu\), and \(\log m\) steps of addition in time \(\alpha\). We then need to compute \(M^{-1}S\). Thus the rank can be computed in

\[ 2(\mu + \alpha) \log m + 3\mu = O(\log m) \]

matrix multiplication steps.

The determination of \(A^+\) can be carried out from the stored values of \(B^k\) and accumulation of these multiplied by the coefficients as in (1); this is then multiplied by \(A^t\) and divided by \(a_k\). This requires \(\log m\) matrix addition steps and 3 matrix multiplication steps, using \(m\) matrix processors.

Thus the computation of \(A^+\) takes \((2\mu + 3\alpha) \log m + 6\mu\) time complexity using \(m\) matrix processors.
4. Concluding remarks

(i) The above algorithm fails for a matrix over a finite field, since the rank of $AA^t$ is not in general equal to the rank of $A$. Also if we want to use a characteristic equation method by directly computing the polynomial of $A$, we cannot, in general, say that the rank of the matrix is equal to the number of non-zero characteristic roots. Further, even in the special case where rank $AA^t$ equals rank $A$, equation (2) is not solvable over the finite field. For instance, over GF(2) the diagonal elements of $M$ are alternatively 1 and 0 and hence the coefficients $a_k$ (except $a_1$) cannot be determined. Even for this special case, over GF(p), this procedure can determine at most the rank of a $(p-1)x(p-1)$ matrix. Hence this algorithm cannot be used for matrices over a finite field which occur in graph theory, coding theory and other areas of computer science.

(ii) The algorithm can be used to compute the inverses of polynomial matrices [3].
References

1. E. V. Krishnamurthy and R. Klette, Fast parallel realization of matrix multiplication, University of Maryland, Computer Science Center, TR-834, November 1979; to appear in EIK 17, 1981.


