SIAM J. COMPUT.
Vol. 5, No. 4, December 1976

FAST PARALLEL MATRIX INVERSION ALGORITHMS*

L. CSANKY†

Abstract. The parallel arithmetic complexities of matrix inversion, solving systems of linear equations, computing determinants and computing the characteristic polynomial of a matrix are shown to have the same growth rate. Algorithms are given that compute these problems in $O(\log^2 n)$ steps using a number of processors polynomial in n. (n is the order of the matrix of the problem.)

Key words. complexity of parallel computation, parallel algorithms, functions of matrices

1. Introduction. The parallel arithmetic complexity of solving systems of linear equations has been an open question for several years. The gap between the complexity of the best algorithms ($2n + O(1)$, where n is the number of unknowns/equations) and the only proved lower bound (2log n (all logarithms in this paper are of base two)) was huge. Csanky [1] exhibited an algorithm for solving systems of linear equations in $2n - O(\log^2 n)$ steps. This also proved that Gaussian elimination, Jordan elimination, Jacobi's method and Strassen's method are not the fastest for parallel computation.

2. The model of parallel computation and some basic definitions. The *model* has an arbitrary number of identical processors with independent control and an arbitrarily large memory with unrestricted access. Each processor is capable of taking its operands from the memory, performing any one of the binary operations $+$, $-$, $*$, $/$ and storing the result in the memory in unit time. This unit time is called a *step*. (The bookkeeping overhead is ignored.) Before starting the computation, the input data is stored in the memory. The *parallel arithmetic complexity of the computation* is the least number of steps necessary to produce the result in the memory.

If C is a computational problem of size n, then the *parallel arithmetic computational complexity* $C(n)$ of C is the least number of steps necessary to compute C for any possible input.

Let A be an order n matrix. Let det (A), adj (A), tr (A) denote the determinant of A, the adjoint of A and the trace of A, respectively. Unless specified otherwise, capital letters denote matrices, lower case letters denote scalars.

Let $I(n), E(n), D(n), P(n)$ denote the parallel arithmetic complexity 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.

3. Results.

LEMMA 1. $2 \log n \leqq I(n), E(n), D(n), P(n)$.

Proof. The proof is direct from the fact that in each case, at least one partial result is a nontrivial function of at least n^2 variables and fan in argument. Q.E.D.

* Received by the editors June 10, 1975, and in revised form September 3, 1975.

† Concord, California. This work was supported in part by National Science Foundation under Grant DCR72-03725-A02.

THEOREM 2. $I(n) = O(f(n)) \Leftrightarrow E(n) = O(f(n)) \Leftrightarrow D(n) = O(f(n)) \Leftrightarrow P(n) = O(f(n))$.

Proof. (i) $D(n) \leqq E(n) + \log n + O(1)$. Let D_t denote an order t determinant and let D_n be the determinant to be computed. Define $x_k = D_{n-k}/D_{n-k+1}$, where $1 \leqq k \leqq n-1$ and D_{n-k} is a properly chosen minor of D_{n-k+1}. Since $\prod_{(k)} x_k = D_1/D_n$, $D_n = D_1/\prod_{(k)} x_k$. Thus to compute D_n, compute x_k for all k in $E(n-k+1)$ parallel steps by solving the corresponding systems of equations, then in $\leqq \log n + O(1)$ additional steps, compute D_n.

(ii) $E(n) \leqq I(n) + 2$. Bring $Ax = b$ to the form $A'x = (I)_{*1}$ in two steps by row operations, then invert A' in $I(n)$ steps. $x = (A'^{-1})_{*1}$. ($(A)_{*j}$ denotes the jth column of A.)

(iii) $I(n) \leqq P(n) + 1$. Invert A by the classical inverse formula using that $g(0) = \det(B)$, where g is the characteristic polynomial of B.

(iv) $P(n) \leqq D(n) + \log n + O(1)$. To compute the characteristic polynomial g of A, first compute $g(w^i)$ for all distinct w^i parallel, where w is a primitive $(n+1)$st root of unity, by using the algorithm for computing determinants. This computation takes $D(n) + 1$ steps. Then compute the coefficients of g by fast fourier transform. This takes $\log n + O(1)$ steps.

From Lemma 1 and the four inequalities above, the theorem follows. Q.E.D.

THEOREM 3. $I(n) \leqq O(\log^2 n)$, *and the number of processors used in the algorithm is a polynomial in n.*

Proof. Let $\lambda_1, \cdots, \lambda_n$ denote the roots of the characteristic polynomial $f(\lambda)$ of A. Let

$$s_k = \sum_{i=1}^{n} \lambda_i^k \quad \text{for } 1 \leqq k \leqq n.$$

Then

$$s_k = \operatorname{tr}(A^k) \quad \text{for } 1 \leqq k \leqq n$$

and

$$\begin{bmatrix} 1 & & & & & & \\ s_1 & 2 & & & & & \\ s_2 & s_1 & 3 & & & & \\ s_3 & s_2 & s_1 & 4 & & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \\ s_{n-1} & s_{n-2} & s_{n-3} & s_{n-4} & \cdots & s_1 & n \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ \vdots \\ \vdots \\ c_n \end{bmatrix} = - \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ \vdots \\ s_n \end{bmatrix},$$

or in a more compact form,

$$S\mathbf{c} = -\mathbf{s}.$$

(These formulas constitute Leverrier's method (Faddeev–Faddeeva [2]) for finding the coefficients of the characteristic polynomial of a matrix.)

To invert A, first compute s_k for $1 \leqq k \leqq n$; this takes $\log^2 n + O(\log n)$ steps and $\frac{1}{2}n^4$ processors. Then invert triangular matrix S in $\log^2 n + O(\log n)$ steps using $O(n^3)$ processors (Heller [4] and Csanky [1] have independently developed such algorithms). Compute c_i for $1 \leqq i \leqq n$ in $\log n + O(1)$ steps from $\mathbf{c} = -S^{-1}\mathbf{s}$ using $O(n^2)$ processors. A is invertible $\Leftrightarrow c_n \neq 0$ and

$$A^{-1} = -\frac{A^{n-1} + c_1 A^{n-2} + \cdots + c_{n-1} I}{c_n}.$$

This formula, since $A^2, \cdots, A^{n-1}$ are already available, can be evaluated in $\log n + O(1)$ steps using $O(n^3)$ processors. Thus

$$I(n) \leqq 2\log^2 n + O(\log n)$$

and

$$p \leqq \tfrac{1}{2}n^4. \qquad \text{Q.E.D.}$$

Alternate proof. Let A be the order n matrix to be inverted. Let the characteristic polynomial $f(\lambda)$ of A be

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

Let

$$\operatorname{adj}(\lambda I - A) = I\lambda^{n-1} + B_2\lambda^{n-2} + \cdots + B_{n-1}\lambda + B_n.$$

Using the idea of one of the simplest proofs of the Cayley–Hamilton theorem (Marcus–Ming [5]), the following formulas were obtained:

(1) $$B_1 = I,$$

(2) $$B_k = AB_{k-1} - \frac{I}{k-1}\operatorname{tr}(AB_{k-1}),$$

(3) $$c_i = -\frac{1}{i}\operatorname{tr}(AB_i)$$

and

(4) $$A^{-1} = n\frac{B_n}{\operatorname{tr}(AB_n)}.$$

(A number of people derived essentially the same formulas in recent years. Frame [3] seems to have been the first.)

Define operator T as

$$TN = \operatorname{tr}(N),$$

$$(I + MT)N = N + MTN = N + M\operatorname{tr}(N),$$

where M, N are order n matrices. Then

$$B_k = \left(I - \frac{I}{k-1}T\right)AB_{k-1}$$

and

$$B_n = \left(I - \frac{I}{n-1}T\right)\left\{A\left[\left(I - \frac{I}{n-2}T\right)\{A[\cdots(I - IT)\{A[I]\}\cdots]\}\right]\right\}. \tag{5}$$

In this formula, A can be thought of as a second operator, its action being multiplication on the left.

Observe that

$$T(AT) = (TA)T, \tag{6}$$

that is, operators T and A associate. This implies that the factors in the expression for B_n associate, i.e.,

$$B_n = \left(A - \frac{I}{n-1}TA\right)\left(A - \frac{I}{n-2}TA\right)\cdots\left(A - \frac{I}{2}TA\right)(A - ITA). \tag{7}$$

Thus B_n can be computed in roughly $\log n$ stages by the well-known binary tree method.

Stage 1. For all $i + 1 = 2t$, compute from $(A - [I/(i+1)]TA)(A - (I/i)TA)$ the form

$$A^2 - M^{(1)}_{1\,2t}TA^2 - M^{(2)}_{1\,2t}TA.$$

As a consequence of (6), matrices $M^{(h)}_{1\,2t}$ can be computed independently. The powers of A can also be computed independently.

⋮

Stage j. For all $2^j t$, compute from

$$(A^{2^{j-1}} - M^{(1)}_{j-1\;2^j t}TA^{2^{j-1}} - \cdots - M^{(2^{j-1})}_{j-1\;2^j t}TA)$$
$$\cdot(A^{2^{j-1}} - M^{(1)}_{j-1\;2^j(t-1)}TA^{2^{j-1}} - \cdots - M^{(2^{j-1})}_{j-1\;2^j(t-1)}TA)$$

the form

$$A^{2^j} - M^{(1)}_{j\,2^j t}TA^{2^j} - M^{(2)}_{j\,2^j t}TA^{2^{j-1}} - \cdots - M^{(2^j)}_{j\,2^j t}TA.$$

As a consequence of (6), matrices $M^{(h)}_{j\,2^j t}$ can be computed independently. The powers of A can also be computed independently.

The number of steps stage j takes is roughly

$$\log n + 1 + \log n + 1 + \log 2^j = \log n + \log n + 2 + j,$$

where the first two terms are the number of steps the multiplication on the right by $A^{2^{j-1}}$ and matrices $M^{(h)}$ takes, the third term is the number of steps computing the traces takes, multiplication by the trace takes 1 step and addition of the matrices takes j steps. Thus the total number of steps for all stages is roughly

$$(\log n)(2\log n + 2) + 1 + 2 + 3 + \cdots + \log n = 2.5\log^2 n + 2.5\log n.$$

Since $nB_n/(TAB_n)$ ($TAB_n \neq 0 \Leftrightarrow A$ is invertible) can be computed in $O(\log n)$ additional steps, we have

$$I(n) \leqq 2.5\log^2 n + O(\log n).$$

A very crude upper bound on the number of processors can be obtained by observing that in any stage, the multiplication phase takes the maximum number of processors.

In stage j, the number of matrix multiplications is roughly

$$2^{j-1} + \frac{n}{2^j} 2^{2j-2}.$$

Thus, for the number of processors p, we obtain

$$p \leqq \max_{(j)} \left\{ n^3 2^{j-1} + \frac{n^4}{2^j} 2^{2j-2} \right\} \approx \frac{1}{4} n^5 + \frac{1}{2} n^3. \qquad \text{Q.E.D.}$$

COROLLARY 4. *$E(n)$, $D(n)$, $P(n) \leqq O(\log^2 n)$, and the number of processors used in the algorithms is a polynomial in n.*

Proof. Solve $Ax = b$, where b is a column vector by inverting A; then $x = A^{-1}b$. Compute $\det(A)$ from $\det(A) = -c_n = (TAB_n)/n$. Compute $f(\lambda)$ from $c_i = -(TAB_i)/i$. Q.E.D.

Let PWR(n) denote the parallel arithmetic complexity of computing A^n. Then we have the next theorem.

THEOREM 5. $I(n) \leqq 2\text{PWR}(n) + O(\log n)$.

Proof. To invert A, first compute the triangular matrix S in PWR $(n) + \log n + O(1)$ steps. Then compute $S^2, S^3, \cdots, S^n$ in PWR (n) additional steps.

Since $\log n \leqq \text{PWR}(n)$ and one can compute the coefficients d_i of the characteristic polynomial $g(\lambda)$ of S from the roots of $g(\lambda)$ (these roots are $1, 2, \cdots, n$) in $2 \log n + O(1)$ steps (Csanky [1]), by the time the powers of S are computed, the coefficients of $g(\lambda)$ are also computed. Then

$$S^{-1} = -\frac{S^{n-1} + d_1 S^{n-2} + \cdots + d_{n-1} I}{d_n},$$

and S^{-1} can be computed in $\log n + O(1)$ additional steps. Compute c_i for $1 \leqq i \leqq n$ from $\mathbf{c} = S^{-1}\mathbf{s}$ in $\log n + O(1)$ steps.

$$A^{-1} = -\frac{A^{n-1} + c_1 A^{n-2} + \cdots + c_{n-1} I}{c_n}$$

can be computed in $\log n + O(1)$ additional steps. Q.E.D.

4. Conclusions. This work has decreased the gap between the lower and upper bounds on the parallel arithmetic complexity of problems I, E, D, P. Indeed, we have

$$O(\log n) \leqq I(n), E(n), D(n), P(n) \leqq O(\text{PWR}(n)) \leqq O(\log^2 n).$$

It is our belief that if $I(n) = O(\log n)$, then the algorithm which establishes this will have to use some new, surprising and fundamental result.

Acknowledgments. I am indebted to Professor Phil Spira for his friendship and constructive criticism during the initial stage of this research. I wish to thank Professor Manuel Blum for his encouragement and interest in this work.

REFERENCES

[1] L. CSANKY, *On the parallel complexity of some computational problems*, Ph.D. dissertation, Computer Sci. Div., Univ. of Calif., Berkeley, 1974.

[2] D. K. FADDEEV AND V. N. FADDEEVA, *Computational Methods of Linear Algebra*, W. H. Freeman, San Francisco, 1963.

[3] J. S. FRAME, *A simple recurrent formula for inverting a matrix*, Bull. Amer. Math. Soc., 55 (1949), p. 1045.

[4] D. HELLER, *A determinant theorem with applications to parallel algorithms*, Dept. of Computer Sci., Carnegie-Mellon Univ., Pittsburgh, 1973.

[5] M. MARCUS AND H. MING, *A Survey of Matrix Theory and Matrix Inequalities*, Allyn and Bacon, Boston, 1964.