FAST AND EFFICIENT PARALLEL SOLUTION OF DENSE LINEAR SYSTEMS

V. PAN[1]† and J. REIF[2]‡

[1]Department of Mathematics, Lehman College, CUNY, Bronx, NY 10468, U.S.A. and Computer Science Department, SUNY Albany, Albany, NY 12222, U.S.A.

[2]Computer Science Department, Duke University, Durham, NC 27706, U.S.A.

Abstract—The most efficient previously known parallel algorithms for the inversion of a nonsingular $n \times n$ matrix A or solving a linear system $A\mathbf{x} = \mathbf{b}$ over the rational numbers require $O(\log^2 n)$ time and $M(n)\sqrt{n}$ processors [provided that $M(n)$ processors suffice in order to multiply two $n \times n$ rational matrices in time $O(\log n)$]. Furthermore, the known polylog arithmetic time algorithms for those problems are numerically unstable. In this paper we apply Newton's iteration and initially choose an approximate inverse matrix by following Ben-Israel. This quadratically convergent and numerically stable iterative method takes $O(\log^2 n)$ parallel time using $M(n)$ processors to compute the inverse (within the relative precision 2^{-n^c} for a positive constant c) of an $n \times n$ rational matrix A with the condition number at most n^d for a constant d. This is the optimum processor bound and by a factor of $\sqrt{n}$ improvement of the previously known processor bounds for polylogarithmic time matrix inversion. The algorithm does not require to precompute the condition number of the input matrix, but it just converges slower for ill-conditioned input matrices.

1. INTRODUCTION

Recently, it has become feasible to construct computer architectures with a large number of processors. We assume the customary model of parallel arithmetic computations with shared memory (which can be thought of as PRAM or arithmetic circuits [1]), where, in each step, each processor can perform a single addition, subtraction, multiplication, or division over rational numbers. In the present paper we are concerned about the efficient use of this parallelism for solving some fundamental problems of numerical linear algebra such as

(1) INVERT: given an $n \times n$ rational matrix $A = (a_{ij})$, then output A^{-1} within a prescribed accuracy ϵ if A is well-conditioned, else output: ILL-CONDITIONED.
(2) LINEAR-SOLVE: given a well-conditioned $n \times n$ matrix A and a column vector $\mathbf{b}$ of length n, find the vector $\mathbf{x} = A^{-1}\mathbf{b}$ within a prescribed accuracy ϵ.

With respect to a fixed matrix operator norm (see Definition A1 in the Appendix) we say that a matrix A is *well-conditioned* if cond $A < C$ for a certain parameter C, and we say that we have *computed the matrix* A^{-1} (*or the vector* $\mathbf{x}$) *within the accuracy* ϵ if the norm of the error matrix (or of the error vector) divided by the norm of A^{-1} (or of $\mathbf{x}$, respectively) does not exceed ϵ.

We are interested in parallel numerically stable algorithms that support polylogarithmic time bounds and small processor bounds for INVERT and LINEAR-SOLVE. Certainly, LINEAR-SOLVE can be immediately reduced to INVERT by the multiplication of A^{-1} by $\mathbf{b}$; such a reduction is appropriate if several linear systems $A\mathbf{x} = \mathbf{b}$ wth the same matrix A and different vectors $\mathbf{b}$ must be solved.

We will present algorithms for INVERT and for LINEAR-SOLVE that *do not require to precompute* C; they use a procedure that converges for any nonsingular input matrix A and for any choice of ϵ. On the other hand, the complexity estimates of our algorithms depend on the value of C and on the choice of ϵ. In this paper we will be primarily concerned about the complexity estimates in the case where $C = n^c$, $\epsilon = 2^{N(n)}$, $N(n) = n^d$ for some positive constants c and d, say, $c = 100$, $d = 10$. This covers all the instances of practical interest. For the sake of completeness, we will also supply the estimates in the case of arbitrary C and ϵ. To simplify the estimates,

†Supported by NSF Grants MCS-8203232 and DCR-8507573.
‡This work was supported by NSF Grant DCR 8503151 and Office of Naval Research Contract N00014-80-C-0647.

Preprint of paper appearing in Computers and Mathematics with Applications, Vol. 17, No. 11, 1989, pp. 1481-1491.

we will assume that the arithmetic operations are performed with infinite precision. In addition, we will present the error estimates in the case of finite precision computation; this will demonstrate that our iterative algorithms are stable; moreover, two of them are self-correcting.

The sequential time of our parallel algorithms exceeds the sequential time of the Gaussian elimination by a factor of $\log n$. For that reason and also taking into account the complexity of processor communication, whose influence diminishes the gain of our parallel acceleration, the rather straightforward parallelization of Gaussian elimination (that requires parallel time n and $O(n^2)$ processors) is still preferred in practice to our dense matrix inversion algorithms. On the other hand, the processor efficient polylogarithmic time matrix inversion is important theoretically itself and due to its extensions and applications to several computations in linear algebra, linear programming, and combinatorial analysis [2–10]. The extension to the computations with dense structured matrices [10] has potential practical value.

1.1. Previous work on parallel matrix inversion

Parallel algorithms with simultaneous polylog time and polynomial processor bounds are known to exist for the INVERT and LINEAR-SOLVE problems, but their practical utility is limited due to their large processor bounds and numerical stability problems.

Reference [11] gave the first proof that INVERT, over the fields of characteristic 0, can be done in polylog time. The result was at that time surprising, and the proof is quite elegant. Csanky used the Cayley–Hamilton theorem to reduce the problem of matrix inversion essentially to the problem of computing $O(n)$ products of $n \times n$ matrices. Let $M(n)$ processors suffice in order to multiply two $n \times n$ matrices in $O(\log n)$ time. Here and hereafter the numbers of processors are defined within a constant factor (for we may decrease the number of processors from P to $O(P/s)$ using s times more parallel steps for any natural $s \leqslant P$). By the upper bounds of [12], $M(n) \leqslant n^{2.81}$. That result can be extended as follows. If $k \times k$ matrices can be multiplied involving $0.5\,k^{\beta}$ arithmetic operations for some k and β, then $M(n) \leqslant n^{\omega}$ for some $\omega < \beta$ and for all n [4]. The current best upper bound on β and ω is 2.375... [13]; however, for matrices of smaller sizes, we should rather count on $M(n) = n^3/\log n$, due to the considerable overhead of the known asymptotically fast algorithms for matrix multiplication [14]. Csanky's algorithm simultaneously takes time $O(\log^2 n)$ and uses $nM(n)$ processors. The later works [15, 16] reduced the processor bounds at first to $\sqrt{n}M(n)$ and then, furthermore, to $\sqrt{n}M(n)/n^{\delta}$ for a small positive $\delta = \delta(\omega)$. If nonexact arithmetic is used, as occurs in practice, all of these methods suffer from numerical instability, due to their use of the Cayley–Hamilton theorem and to involving the evaluation of the characteristic polynomial of the input matrix, which is an unstable computation (see the analysis in [17, 18]).

Two methods for parallel matrix inversion over arbitrary fields of constants are known. In [1], the authors observed that the results of [19] and [20] combined can be used to parallelize the sequential Gaussian elimination algorithm for matrix inversion. This yielded an algorithm for INVERT over any field that simultaneously used $O(\log^2 n)$ parallel time and a very large but polynomial in n processor bound. Reference [21] presented another such matrix inversion algorithm over arbitrary fields. Neither of these two polylog time algorithms can be viewed as practical in the case of real, complex, or rational inputs.

1.2. Our results for INVERT

We will indicate a parallel iterative solution for INVERT that in the case of well-conditioned matrices runs in polylog time and significantly decreases the known processor bounds.

The iterative methods that we describe in Sections 2 and 4 are known ones, namely, Newton's iteration and its extensions. The first use of Newton's iteration in order to invert a matrix is attributed by Householder to [22] (compare also [23–30] for more recent descriptions and study of such methods).

To make Newton's method work for matrix inversion, an initial approximate inverse B of a given matrix A is required such that $\| I - BA \|$ is substantially less than 1 where I is the identity matrix. Now we arrive at the problem of efficiently (i.e. without inverting A) finding such an approximate inverse of a well-conditioned matrix A. For a strictly diagonally dominant matrix A an approximate inverse B of A is an easily computable diagonal matrix. Similarly, a good approximate inverse B can be easily computed for many other special input matrices A (see Section 5 of this paper), but

such recipes do not apply to the case of general dense matrices A, which are encountered, say, in numerical solution of integral equations.

We observe that it suffices to define an approximate inverse B of a matrix A such that $\|I - BA\| \leqslant 1 - (1/n^{O(1)})$. (Here and hereafter $n^{O(1)}$ denotes the values bounded by a polynomial in n as $n \to \infty$.) Then, since Newton's method is quadratically convergent, its $O(\log n)$ iteration steps are sufficient for numerical computation of A^{-1} (up to accuracy 2^{-n^c} for any fixed positive c). Each iteration takes only $O(\log n)$ time using $M(n)$ processors. In that situation we begin Newton's iteration by computing (by a method of [31]) an approximate inverse B of any given well-conditioned matrix A such that $\|I - BA\| < 1 - (1/n^{O(1)})$ (see Lemma 2.3 in Section 2 and its substantiation in Sections 3 and 4). The evaluation of such a matrix B involves only $3n^2 - 2n + 1$ arithmetic operations and $2n - 2$ comparisons that can be implemented using $O(\log n)$ parallel steps and n^2 processors (see Lemma 2.3 in Section 2). The resulting algorithm computes A^{-1} using only relatively few $[O(\log n)]$ matrix multiplications, that is, using $O(\log^2 n)$ time and $M(n)$ processors. This makes the algorithm efficient for both sequential and parallel computations. Furthermore, matrix multiplication can be reduced to matrix inversion [32, p. 51; 14], so the above processor bound, as well as the parallel and sequential time bounds, are optimal or nearly optimal (up to within a factor of $O(\log^2 n)$). We present the asymptotic complexity estimates, but the reader can see from our analysis that Newton's algorithm *has small overhead.* To simplify our presentation, we work with the quadratically convergent algorithm and describe the kth order convergent variations in Section 6 (see [28, p. 83] on the advantage of using cubically convergent methods).

In [4, 33] the above results have been extended (using several different techniques) to the *exact* evaluation of the inverse of A, of the determinant of A, and of all the coefficients of the characteristic polynomial of A in $O(\log^2 n)$ steps using $M(n)$ processors in the case where A is an arbitrary integer matrix such that $\log \|A\| = n^{O(1)}$.

1.3. Contents

We present the iterative parallel algorithm for INVERT and its complexity estimates in Section 2, and prove the main lemma of Section 2 in Section 3. Section 4 contains an alternative proof of that lemma, leading to some extensions of the results. In Section 5 we consider the simplified methods for computing approximate inverses of some special matrices. In Section 6 we consider an alternative to the iterative algorithms of Section 2. In the Appendix we recall some basic definitions from the matrix theory.

2. ITERATIVE METHODS FOR PARALLEL MATRIX INVERSION

In this section $A = (a_{ij})$ denotes a fixed $n \times n$ nonsingular complex matrix (so there exists the inverse matrix A^{-1}), $R(X)$ denotes $I - XA$. A matrix B is called an *approximate inverse* of A if

$$R(B) = I - BA, \quad \|R(B)\| = q < 1. \tag{1}$$

(Actually the relations (1) imply that the matrices A and B are nonsingular [34, p. 465].) Note that $\|(R(B))^i\| \leqslant q^i \to 0$ as $i \to \infty$ if the relations (1) hold.

Lemma 2.1

Let (1) hold. Then

$$A^{-1} = \sum_{i=0}^{\infty} (R(B))^i B.$$

Proof.

$$A^{-1} = (A^{-1}B^{-1})B = (BA)^{-1}B = (I - R(B))^{-1}B = \sum_{i=0}^{\infty} (R(B))^i B.$$

Q.E.D.

Lemma 2.1 immediately implies

Lemma 2.2

Let

$$X_k = \left(\prod_{h=0}^{k-1} (I + (R(B))^{2^h})\right) B.$$

Then

$$A^{-1} - X_k = \sum_{g=2^k}^{\infty} (R(B))^g B.$$

The relations (1) and Lemma 2.2 imply that

$$\| A^{-1} - X_k \| \leqslant \sum_{g=2^k}^{\infty} q^g \| B \| = q^{2^k} \| B \| / (1-q). \tag{2}$$

This suggests an algorithm for INVERT that successively computes $(R(B))^{2^h}$ and X_{h+1} for $h = 0, 1, \ldots, k-1$. Recall that $\| (R(B))^g \| \leqslant q^g$ by the virtue of the relations (1), so the computation by that algorithm is numerically stable in a certain weak sense, but much stronger numerical stability is ensured in the following algorithm:

$$Y_{h+1} = (2I - Y_h A) Y_h = (I + R(Y_h)) Y_h, \quad h = 0, 1, \ldots \tag{3}$$

This second algorithm is exactly Newton's iteration, applied to the matrix equation $R(Y) = O$ (as is described, for example, in [28]). The Newton iteration (3) is very strongly stable, that is, is self-correcting and computes a good approximation to A^{-1}, even if the computation is performed with finite precision of $d = O(\log n)$ binary bits. In Remark 5.1 in Section 5 we will suggest a modification of the algorithm (3). In Section 6 we will recall yet another algorithm (having higher order of convergence) that is also fast, efficient, and strongly stable in the above sense.

It is readily verified that $R(Y_h) = (R(Y_{h-1}))^2$ for $h = 1, 2, \ldots$, so

$$I - Y_k A = R(Y_k) = (R(Y_0))^{2^k}, \quad \| I - Y_k A \| \leqslant q^{2^k} \text{ for all } k. \tag{4}$$

We postmultiply the equations of (4) by A^{-1}, note that the relations (1) and Lemma 2.1 imply that

$$\| A^{-1} \| \leqslant \| B \| / (1-q),$$

and arrive at the bound

$$\| A^{-1} - Y_k \| \leqslant q^{2^k} \| A^{-1} \| \leqslant q^{2^k} \| B \| / (1-q), \tag{5}$$

which extends the relations (2) to the case of the Newton's algorithm. The inequalities (5) imply the next theorem.

Theorem 2.1

Let A and B be two fixed $n \times n$ matrices such that

$$\| I - BA \| = \| R(B) \| = q = 1 - 1/n^{O(1)} \quad \text{as } n \to \infty. \tag{6}$$

Let c be a constant. Then $O(\log n)$ iterations of the algorithm (3), that is, $O(\log^2 n)$ time and simultaneously $M(n)$ processors, suffice in order to compute a matrix $\tilde{A}^{-1}$ such that

$$\| A^{-1} - \tilde{A}^{-1} \| \leqslant 2^{-n^c} \| A^{-1} \| \leqslant 2^{-n^c} \| B \| / (1-q). \tag{7}$$

The latter bound (7) suffices for all practical purposes and cannot be reduced further if $|(\log \| B \|)| = n^{O(1)}$ and if the entries of $\tilde{A}^{-1}$ are to be represented by binary numbers of the form $a2^{-k}$ where a and k are integers, $|a| < 2^m$, $0 \leqslant k < m$, $m = n^{O(1)}$.

It remains to choose B satisfying the equations (6). Following [31] let

$$B = tA^H, \; t = 1/(\| A \|_1 \| A \|_\infty) = 1 \Big/ \left(\max_i \sum_j |a_{ij}| \max_j \sum_i |a_{ij}| \right). \tag{8}$$

We immediately verify the next estimate.

Lemma 2.3

Computing matrix B by means of the equations (8) only requires $3n^2-2n+1$ arithmetic operations and $2n-2$ comparisons, which can be performed in $O(\log n)$ parallel time using n^2 processors.

We will prove the next lemma in Sections 3 and 4 (compare also Remark 4.1 in Section 4).

Lemma 2.4

Let $A=(a_{ij})$ be a nonsingular matrix, let the matrix B be defined by the equations (8), and let the matrix $R(B)$ be defined by the equation (1). Then

$$\|B\|_2 \leqslant 1/\|A\|_2 \leqslant 1\Big/\max_{i,j}|a_{ij}|, \quad \|R(B)\|_2 \leqslant 1-1/((\operatorname{cond}A)_2^2 n). \tag{9}$$

Combining Theorem 2.1, Lemmas 2.3 and 2.4, and the inequality of (A2) of the Appendix we arrive at

Corollary 2.1

Let $A=(a_{ij})$, c be arbitrary constant, $(\operatorname{cond}A)_2 = n^{O(1)}$. Then $O(\log^2 n)$ time, $M(n)$ processors suffice in order to compute a matrix $\tilde{A}^{-1}$ such that

$$\|A^{-1}-\tilde{A}^{-1}\|_2 \leqslant 2^{-n^c}/\|A\|_2 \leqslant \|A^{-1}\|_2 2^{-n^c}/(\operatorname{cond}A)_2 \leqslant \|A^{-1}\|_2 2^{-n^c}.$$

Due to the inequalities (A3) of the Appendix, we may replace the 2-norm of matrices and $(\operatorname{cond}A)_2$ in Corollary 2.1 by the s-norm and by $(\operatorname{cond}A)_s$ for $s=1$ and for $s=\infty$.

Corollary 2.1 covers most of the instances A of practical interest; the inequalities (5) and (9) imply the following immediate generalization of that corollary.

Corollary 2.2

For any number k and for any nonsingular $n \times n$ matrix A, it is sufficient to use $O(k)$ parallel steps and $M(n)$ processors in order to compute a matrix $\tilde{A}^{-1}$ such that

$$\|\tilde{A}^{-1}-A^{-1}\|_2 \leqslant (1-1/(n(\operatorname{cond}A)_2^2))^{2^k}/\|A\|_2 \leqslant \|A^{-1}\|_2(1-1/(n(\operatorname{cond}A)_2^2))^{2^k},$$

so the output error bound $\|A\|_2\|\tilde{A}^{-1}-A^{-1}\|_2 < \epsilon$ can be ensured using $O(\log(n(\operatorname{cond}A)_2^2\log(1/\epsilon)))$ parallel steps and $M(n)$ processors, for an arbitrary positive $\epsilon < 1$.

3. PROOF OF LEMMA 2.4

In this section we will prove Lemma 2.4 following [2]. Our Section 4 contains an alternative proof (leading to some further extensions of the results of Section 2 [26, 35, 36]).

We will use the three following well-known results [34, pp. 416–431; 18; 37, p. 15].

Lemma 3.1

$$\|W\|_2 = \|W^{\mathrm{H}}\|_2 \text{ for all } W; \quad \|W\|_2 = \rho(W) \text{ if } W = W^{\mathrm{H}}.$$

Lemma 3.2

A Hermitian positive semidefinite matrix has only nonnegative eigenvalues.

Lemma 3.3

Let $W=(w_{ij})$. Then

$$\|W^{\mathrm{H}}W\|_2 = \|W\|_2^2 \leqslant \|W^{\mathrm{H}}W\|_1 \leqslant \max_i \sum_j |w_{ij}| \max_j \sum_i |w_{ij}| \leqslant n\|W^{\mathrm{H}}W\|_2.$$

Applying Lemma 3.3 to $W=A^{\mathrm{H}}$, we obtain that $\|A^{\mathrm{H}}\|_2^2 \leqslant 1/t$, where t is defined by the equations (8). Therefore, by the virtue of Lemma 3.1, $\|A^{\mathrm{H}}\|_2 \leqslant 1/(t\|A\|_2)$. Due to the equation (A.1) of the Appendix,

$$\|A\|_2 \geqslant \max_{i,j}|a_{ij}|,$$

so we arrive at the first two inequalities of (9). It remains to prove the last inequality of (9).

Lemma 3.4

Let λ be an eigenvalue of A^HA and A be nonsingular. Then $1/\|A^{-1}\|_2^2 \leqslant \lambda \leqslant \|A\|_2^2$.

Proof. $\lambda \leqslant \rho(A^HA) = \|A\|_2^2$ (see Definition A4 in the Appendix and Lemma 3.3). On the other hand, $(A^HA)^{-1} = A^{-1}(A^{-1})^H$, so $(A^HA)^{-1}$ is a Hermitian positive definite matrix. Recall that $1/\lambda$ is an eigenvalue of $(A^HA)^{-1}$, so $1/\lambda \leqslant \rho((A^HA)^{-1}) = \rho(A^{-1}(A^{-1})^H) = \|A^{-1}\|_2^2$. Q.E.D.

Lemma 3.5

Let B and t be defined by the equations (8). Let μ be an eigenvalue of $R(B) = I - BA$. Then $0 \leqslant \mu \leqslant 1 - 1/((\text{cond}\, A)_2^2 n)$.

Proof. Let $R(B)\mathbf{v} = \mu\mathbf{v}$ for $\mathbf{v} \neq 0$. Then $(I - tA^HA)\mathbf{v} = \mathbf{v} - tA^HA\mathbf{v} = \mu\mathbf{v}$. Therefore, $A^HA\mathbf{v} = \lambda\mathbf{v}$ for $\lambda = (1-\mu)/t$, so λ is an eigenvalue of A^HA. Lemma 3.4 implies that $1/\|A^{-1}\|_2^2 \leqslant \lambda = (1-\mu)/t \leqslant \|A\|_2^2$. It immediately follows that

$$1 - t\|A\|_2^2 \leqslant \mu \leqslant 1 - t/\|A^{-1}\|_2^2.$$

It remains to recall the equations (8) and to apply the inequalities of Lemma 3.3. Q.E.D.

Since μ is an arbitrary eigenvalue of $R(B)$, $\rho(R(B)) \leqslant 1 - 1/((\text{cond}\, A)_2^2 n)$. On the other hand, $\|R(B))\|_2 = \rho(R(B))$ since $R(B) = I - tA^HA$ is Hermitian. This completes the proof of Lemma 2.4. Q.E.D.

4. AN ALTERNATIVE PROOF OF LEMMA 2.4 AND SOME EXTENSIONS OF THE RESULTS OF SECTION 2

We will only prove the last inequality of Lemma 2.4. We will use the following well-known results [37, pp. 16–17]:

Lemma 4.1

For an arbitrary $n \times n$ matrix W there exist its singular value decomposition (SVD), that is, there exists $n \times n$ matrices $U = U(W)$, $V = V(W)$ and $\Sigma = \Sigma(W)$ such that

$$W = U\sum V^H,$$
$$U^HU = V^HV = I,$$
$$\sum = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n),$$
$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_n \geqslant 0; \tag{10}$$

furthermore,

$$\sigma_1 = \sigma_1(W) = \|W\|_2 \tag{11}$$

(σ_i are called the *singular values* of W; $\lambda_i = \sigma_i^2$ are the eigenvalues of W^HW for $i = 1, 2, \ldots, n$).

The relations (10) immediately imply the following equations (where t is a constant):

$$W^H = V\Sigma U^H, \quad \|W^H\|_2 = \sigma_1 = \|W\|_2, \quad W^HW = V\Sigma^2V^H, \tag{12}$$

$$I - tW^HW = V\Omega V^H, \quad \Omega = I - t\Sigma^2 = \text{diag}(\omega_1, \ldots, \omega_n), \tag{13}$$

$$\omega_i = 1 - t\sigma_i^2, \quad i = 1, \ldots, n.$$

Furthermore, if W is nonsingular, then

$$W^{-1} = V\Sigma^{-1}U^H, \quad \Sigma^{-1} = \text{diag}(1/\sigma_1, \ldots, 1/\sigma_n). \tag{14}$$

The equations (13) and (14) define the SVDs of the matrices $I - tW^HW$ (for $t \leqslant 1/\sigma_1^2$) and W^{-1} (up to within reversing the order of the rows and/or columns of the matrices V, Ω, Σ^{-1}, and U^H); this observation combined with the equations (11) implies that

$$\|I - tW^HW\|_2 = \max_i |\omega_i| = \max\{|1 - t\sigma_1^2|, \quad |1 - t\sigma_n^2|\} = 1 - t\sigma_n^2 \tag{15}$$

if $t \leqslant 1/\sigma_1^2$,

$$\|W^{-1}\|_2 = 1/\sigma_n. \tag{16}$$

Applying the relations (11), (15), and (16) to the nonsingular matrix $W = A$ and to $R(B) = I - BA = I - tA^{H}A$, we obtain that

$$(\text{cond}\, A)_2 = \sigma_1/\sigma_n, \quad \sigma_n = \sigma_1/(\text{cond}\, A)_2,$$
$$\| R(B) \|_2 = 1 - t\sigma_n^2 = 1 - t\sigma_1^2/(\text{cond}\, A)_2^2, \tag{17}$$

as long as $t \leqslant 1/\sigma_1^2$.

We define t by the equation (8), deduce from the equations (11) for $W = A$ and from Lemma 3.3 that

$$1/(n\sigma_1^2) \leqslant t \leqslant 1/\sigma_1^2,$$

combine the latter lower bound on t with the equations (17), and arrive at the last inequality of (9). Q.E.D.

Remark 4.1

The latter proof of Lemma 2.4 and the results of Corollaries 2.1 and 2.2 can be immediately extended in several directions. We may, of course, choose $t = \| A^{H}A \|_1$ in the equations (8) (see Remark 5.1 below); moreover, any choice of t supports Corollaries 2.1 and 2.2 as long as

$$t\sigma_1^2 \leqslant 1 - 1/(n(\text{cond}\, A)_2^2),$$
$$t\sigma_n^2 \geqslant 1/(n(\text{cond}\, A)_2^2), \quad \sigma_i = \sigma_i(A), \quad i = 1, 2, \ldots, n.$$

Furthermore, even if A is singular, so that $\sigma_{r+1} = \sigma_{r+2} = \ldots = \sigma_n = 0$, $\sigma_r \neq 0$ for some $r < n$, the proofs and the results can still be extended if we just replace n by r, $(\text{cond}\, A)_2$ by $\kappa(A) = \sigma_1(A)/\sigma_r(A)$, and the inverse of $A = U\Sigma V^{T}$ by the *Moore–Penrose pseudo-inverse*,

$$A^{+} = V\Sigma^{+}U^{T},$$

where $\Sigma^{+} = \text{diag}(\sigma_1^{+}, \ldots, \sigma_n^{+})$, $\sigma_i^{+} = \sigma_i = 0$, if $\sigma_i = 0$, $\sigma_i^{+} = 1/\sigma_i$ otherwise (compare [37, pp. 139–140]). Thus Newton's iteration (3) always converges to A^{+} for any square matrix A and actually also for any rectangular matrix A (which can be formally made square by padding it with some zero rows or columns). This implies the usual applications to the linear least-squares computations [37] and to the evaluation of the rank $r(A)$ of a matrix A, $r(A) = \text{trace}(A^{+}A)$ [26]. Our complexity analysis is also extended to that case (we leave this to the reader), although Newton's algorithm (3) may no longer remain self-correcting if A is singular [36].

5. APPROXIMATE INVERSE OF A MATRIX IN SOME SPECIAL CASES

In this section we will improve the choice (8) of an approximate inverse of A for certain special classes of matrices A.

Modification 5.1

(Compare [28, p. 84; 2].) For a matrix A, let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of $A^{H}A$ such that

$$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n > 0,$$

so $\sigma_i = \sqrt{\lambda_i}$ are the singular values of A,

$$\lambda_1 = \sigma_1^2(A) = \| A \|_2^2, \quad \lambda_n = \sigma_n^2(A) = 1/\| A^{-1} \|_2^2. \tag{18}$$

If for a given matrix A we may precompute λ_1 and λ_n then we may improve the choice (8) for the approximate inverse B of A by setting

$$B = t^{*}A^{H}, \quad t^{*} = 2/(\lambda_1 + \lambda_n). \tag{19}$$

The equations (18) and (19) imply that $\| R(B) \|_2 = (\lambda_1 - \lambda_n)/(\lambda_1 + \lambda_n)$, and this leads to the following equations, which improve the bounds of Lemma 2.4:

$$q = \| R(B) \|_2 = \frac{(\text{cond}\, A)_2^2 - 1}{(\text{cond}\, A)_2^2 + 1} = 1 - \frac{2}{(\text{cond}\, A)_2^2 + 1}. \tag{20}$$

The improvement over the estimate of Lemma 2.4 is limited, that is, $1 - \| R(B) \|_2$ increases less than $2n$ times, even if we precompute λ_1 and λ_n satisfying the equations (18) and then replace the equations (8) by (19). Reference [35] suggests saving about one half of iterations via the following acceleration procedure:

$$\hat{Y}_{h+1} = t_{h+1}(2I - \hat{Y}_h A)\hat{Y}_h,$$
$$t_{h+1} = 2/(1 + (2 - s_h)s_h), \quad s_{h+1} = t_{h+1}(2 - s_h)s_h), \quad h = 0, 1, \ldots,$$
$$s_0 = 2\lambda_1^2/(\lambda_1^2 + \lambda_n^2).$$

The approach can be extended to the case where some approximations to λ_1 and λ_n (rather than their exact values) are available.

Modification 5.2

If the matrix A is Hermitian positive definite, which is frequent in practical computations [37–39], then we may choose

$$B = t'I, \quad t' = 1 \Big/ \max_j \sum_i |a_{ij}| \leqslant 1/\| A \|_2 \leqslant 1 \Big/ \max_{i,j} |a_{ij}|.$$

Extending the proof of Lemma 2.4, we derive in this case that

$$\| R(B) \|_2 \leqslant 1 - t'/\| A^{-1} \|_2 \leqslant 1 - 1/(n^{1/2}(\operatorname{cond} A)_2). \tag{21}$$

Remark 5.1

Actually we may reduce the inversion of an arbitrary nonsingular matrix A to the inversion of the Hermitian positive definite matrix $A^H A$ since $A^{-1} = (A^H A)^{-1} A^H$. Then we may compute the 1-norm $\| A^H A \|_1$ of the matrix $A^H A$, choose $B = I/\| A^H A \|_1$, and arrive at the bound (21) where $A^H A$ replaces A. The resulting extension of the estimate (21) will be close to (but slightly better than) the estimate (9).

Modifications 5.1 and 5.2 can be combined together if A is a Hermitian positive definite matrix and if $\| A \|_2$ and $1/\| A^{-1} \|_2$ can be precomputed, in which case we could set $B = 2I/(\| A \|_2 + 1/\| A^{-1} \|_2)$ and yield

$$\| R(B) \|_2 = 1 - \frac{2}{(\operatorname{cond} A)_2 + 1}.$$

Modification 5.3. Case 1

Let $A = (a_{ij})$ be *strongly diagonally dominant*, that is, let for a constant c,

$$(2 - 1/n^c)\,|a_{ii}| > \sum_j |a_{ij}| \quad \text{for all } i \tag{22}$$

or

$$(2 - 1/n^c)\,|a_{ij}| > \sum_i |a_{ij}| \quad \text{for all } j. \tag{23}$$

The class of diagonally dominant matrices is very important in applications (see [40, 38]). The inequalities (22) imply that A is strongly row-diagonally dominant; the inequalities (23) imply that the matrix A is strongly column-diagonally dominant. In both cases we choose

$$B = \operatorname{diag}\{1/a_{11}, 1/a_{22}, \ldots, 1/a_{nn}\} \tag{24}$$

and immediately verify the following facts.

Lemma 5.1

The relations (22) and (24) imply that

$$\| R(B) \|_\infty \leqslant 1 - 1/n^c.$$

The relations (23) and (24) imply that

$$\| R(B) \|_1 \leqslant 1 - 1/n^c.$$

Thus, in the cases where A is strongly diagonally dominant, we choose the matrix B defined by the equation (24) and again arrive at the bounds of Corollary 2.1, provided that one of the two norms of Definition A2 of the Appendix substitutes for the 2-norm. The computation of the matrix B by the equation (24) involves only n arithmetic operations, which can be performed using one parallel step and n processors.

Modification 5.3. Case 2

Let $A = (a_{ij})$ be a triangular matrix. Then again define the matrix B by the equation (24) and apply the algorithm of Section 2 in order to compute X_k for $k = \lceil \log n \rceil$. Although the matrix B may not satisfy the inequality of (1) in this case, the method works, simply because $R(B)$ is an $n \times n$ triangular matrix whose diagonal is filled with zeros and therefore $R(B)^i = 0$ if $i \geqslant n$. Thus

$$A^{-1} = X_k = \sum_{i=0}^{n-1} (R(B))^i B,$$

so that the convergence of the algorithm to A^{-1} in $\lceil \log n \rceil$ steps immediately follows for any triangular nonsingular matrix A, even if $\operatorname{cond} A$ is exponentially large (compare [32, p. 146]). (Note that the inversion of a matrix A can be reduced to the inversion of triangular matrices if the QR-factors of A or LUP-factors of A are available.)

6. REFINEMENT OF THE APPROXIMATE INVERSE OF A MATRIX BY THE RESIDUAL CORRECTION METHOD

Next, we will consider yet another alternative to the two iterative algorithms for INVERT described in Section 2. As in the case of these two algorithms, we will assume that an approximate inverse B of matrix A has been precomputed such that the relations (1) hold.

The third algorithm relies on the well-known residual correction method (see [34, p. 469]). For a fixed $s \geqslant 2$, we first let $g = 0$, $Z_0 = B$ [see the equations (8)] and then successively compute

$$X_{h+1,g} = X_{h,g} + Z_g(I - AX_{h,g}), \quad X_{0,g} = Z_g \text{ for } h = 0, 1, \ldots, s-1. \tag{25}$$

Then we recursively perform the iteration sweep (25) for $g = 1, 2, \ldots$, choosing $Z_g = X_{s,g}$. Within each iteration sweep (25), $I - X_{h+1,g}A = (I - X_{h,g}A)(I - Z_gA)$, $h = 0, 1, \ldots, s-1$, so that $\| I - Z_{g+1}A \| \leqslant \| I - Z_gA \|^s$. Therefore, after k iteration sweeps (25), we arrive at an approximation Z_k to A^{-1} such that $\| I - Z_kA \| \leqslant q^{s^k}$. We postmultiply $I - Z_kA$ by A^{-1} and deduce that

$$\| A^{-1} - Z_k \| \leqslant q^{s^k} \| A^{-1} \| \leqslant q^{s^k} \| B \| / (1 - q). \tag{26}$$

Thus the algorithm (25) is an sth order method (compare [41]). For $s = 2$, the inequalities (26) are similar to the inequalities (2) and (5).

Unlike the first algorithm of Section 2, both algorithms based on the equations (3; Newton's) and (25) are self-correcting (for a nonsingular input matrix A); the errors of computing $\hat{Y}_k$ and $X_{h,g}$ for any k, h, g do not propagate, that is, they are automatically corrected in the next iteration step, provided, of course, that the computed approximations are not too much contaminated by the errors (in particular, we need to ensure that, in spite of the round-off errors, the computed matrices Y_k and $X_{h,g}$ remain approximate inverses of A).

REFERENCES

1. A. Borodin, J. von zur Gathen and J. Hopcroft, Fast parallel matrix and GCD computation. *Inform. Control* **52**, 241–256 (1982).
2. V. Pan and J. Reif, Efficient parallel solution of linear systems. *Proc. 17th Annual ACM Symp. Theory of Computing*, pp. 143–152 (1985).
3. V. Pan and J. Reif, Fast and efficient algorithms for linear programming and for the linear least squares problem. *Comput. Math. Applic.* **12A**, 1217–1227 (1986).
4. V. Pan, Complexity of parallel matrix computations. *Theor. Comput. Sci.* **54**, 65–85 (1987).

5. V. Pan and R. Schreiber, An improved Newton iteration for the generalized inverse of a matrix, with applications. TR 88-35, SUNY Albany, New York (1988).
6. L. Lovász, Connectivity algorithms using rubber-bands. *Proc. 6th Conf. Foundations of Software Technology and Theoretical Computer Science*, New Delhi, India. *Lecture Notes in Computer Science*, Vol. 241, pp. 394–412. Springer, Berlin (1986).
7. J. von zur Gathen, Parallel algorithms for algebraic problems. *SIAM J. Comput.* **13**, 802–824 (1984).
8. Z. Galil and V. Pan, Improved processor bounds for combinatorial problems in RNC. *Combinatorica* **8**, 193–204 (1988).
9. K. Mulmuley, U. Vazirani and V. Vazirani, Matching is as easy as matrix inversion. *Combinatorica* **7**, 105–114 (1987).
10. V. Pan, Fast and efficient parallel inversion of Toeplitz and block Toeplitz matrices. TR 88-28, SUNY Albany, New York (1988).
11. L. Csanky, Fast parallel matrix inversion algorithm. *SIAM J. Comput.* **5**, 618–623 (1976).
12. A. K. Chandra, Maximal parallelism in matrix multiplication. Report RC-6193, IBM T. J. Watson Research Center, Yorktown Heights, N.Y. (1976).
13. D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions. *J. symbolic Comput.* In press. Short version in *Proc. 19th Ann. ACM Symp. Theory of Computing*, pp. 1–6 (1987).
14. V. Pan, *How to Multiply Matrices Faster. Lecture Notes in Computer Science*, Vol. 179. Springer, Berlin (1984).
15. F. P. Preparata and D. V. Sarwate, An improved parallel processor bound in fast matrix inversion. *Inform. Process. Lett.* **7**, 148–149 (1978).
16. Z. Galil and V. Pan, Parallel evaluation of the determinant and of the inverse of a matrix. *Inform. Process. Lett.* In press.
17. J. H. Wilkinson, Error analysis of direct methods of matrix inversion. *J. ACM* **8**, 281–330 (1961).
18. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford (1965).
19. L. Valiant, S. Skyum, S. Berkowitz and C. Rackoff, Fast parallel computation of polynomials using few processors. *SIAM J. Comput.* **12**(4), 641–644 (1983).
20. V. Strassen, Vermeidung von Divisionen. *J. reine angew. Math.* **164**, 184–202 (1973).
21. S. Berkowitz, On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.* **18**, 147–150 (1984).
22. G. Schultz, Iterative Berechnung der reziproken Matrix. *Z. angew. Math. Mech.* **13**, 57–59 (1933).
23. H. Hotelling, Some new methods in matrix calculation. *Ann. Math. Statist.* **14**, 1–34 (1943).
24. H. Hotelling, Further points on matrix calculations and simultaneous equations. *Ann. Math. Statist.* **14**, 440–441 (1943).
25. E. Bodewig, *Matrix Calculus*, 2nd edn. Interscience, New York (1959).
26. A. Ben-Israel and D. Cohen, On iterative computation of generalized inverses and associated projections. *SIAM J. numer. Analysis*, 410–419 (1966).
27. A. S. Householder, *The Theory of Matrices in Numerical Analysis*. Blaisdell, New York (1964).
28. E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. Wiley, New York (1966).
29. M. Newman, Matrix computation. In *Survey of Numerical Analysis* (Ed. S. Todd), pp. 222–255. McGraw–Hill, New York (1982).
30. A. Bojańczyk, Complexity of solving linear systems in different models of computation. *SIAM J. numer. Analysis* **21**(3), 591–603 (1984).
31. A. Ben-Israel, A note on iterative method for generalized inversion of matrices. *Math. Comput.* **20**, 439–440 (1966).
32. A. Borodin and I. Munro, *The Computational Complexity of Algebraic and Numeric Problems*. Elsevier, New York (1975).
33. V. Pan, Fast and efficient parallel algorithms for the exact inversion of integer matrices. *Proc. 5th Conf. FST & TCS. Lecture Notes in Computer Science*, Vol. 206, pp. 504–521. Springer, New York (1985).
34. K. E. Atkinson, *An Introduction to Numerical Analysis*. Wiley, New York (1978).
35. R. Schreiber, Computing generalized inverses and eigenvalues of symmetric matrices using systolic arrays. In *Computing Methods in Applied Science and Engineering* (Ed. R. Glowinski and J.-L. Lions), pp. 285–295. Elsevier/North Holland, Amsterdam (1984).
36. T. Söderström and G. W. Stuart, On numerical properties of an iterative method for computing the Moore–Penrose generalized inverse. *SIAM J. numer. Analysis* **11**, 61–74 (1974).
37. G. H. Golub and C. F. van Loan, *Matrix Computations*. Johns Hopkins University Press, Baltimore, Md. (1983).
38. L. A. Hageman and D. M. Young, *Applied Iterative Methods*. Academic Press, New York (1981).
39. D. M. Young, *Iterative Solution of Large Linear Systems*. Academic Press, New York (1971).
40. R. S. Varga, *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, N.J. (1962).
41. J. M. Garnett, A. Ben-Israel and S. S. Yau, A hyperpower iterative method for computing matrix products involving the generalized inverse. *SIAM J. numer. Analysis* **8**(1), 104–109 (1971).

APPENDIX

Some Definitions

In this Appendix we will recall some customary definitions from the matrix computation theory (compare [37]).

Definition A1

Let a vector norm be fixed. Then we extend it to the associated *operator norm of matrices*, such that for all matrices W

$$\| W \| = \max_{\mathbf{v} \neq 0} \| W\mathbf{v} \| / \| \mathbf{v} \| \tag{A.1}$$

where the maximum is over all nonzero vectors v. Furthermore, let

$$\text{cond } W = \| W \| * \| W^{-1} \| \quad \text{if } W \text{ is nonsingular,}$$

$$\text{cond } W = \infty \quad \text{otherwise.} \tag{A.2}$$

Note that $\text{cond } W \geqslant 1$ if $\| I \| = 1$.

We write $\| W \|$ in the relations that can be applied to *any* operator matrix norm. In this paper we refer to some specific operator norms of matrices, namely, to the three following norms. (Hereafter u^* denotes the *complex conjugate* of a complex number u.)

Definition A2

$$\| W \|_\infty = \max_i \sum_j |w_{ij}|, \quad \| W \|_1 = \max_j \sum_i |w_{ij}|, \quad \text{and} \quad \| W \|_2$$

are the operator norms of a matrix $W = |w_{ij}|$ associated with the *maximum norm*,

$$\| \mathbf{v} \|_\infty = \max_i |v_i|,$$

with the *1-norm*,

$$\| \mathbf{v} \|_1 = \sum_i |v_i|,$$

and with the *Euclidean* norm,

$$\| \mathbf{v} \|_2 = \left(\sum_i v_i^* v_i \right)^{1/2},$$

of a vector $\mathbf{v} = (v_i)$, respectively [compare (A1)]. We write $(\text{cond } W)_s = \| W \|_s \| W^{-1} \|_s$ if W is nonsingular; $(\text{cond } W)_s = \infty$ otherwise, for $s = \infty$, $s = 1$, $s = 2$.

It is known (see [37, p. 15]) that for an $n \times n$ matrix W,

$$\| W \|_s / \sqrt{n} \leqslant \| W \|_2 \leqslant \| W \|_s \sqrt{n} \quad \text{for } s = 1 \text{ and } s = \infty. \tag{A.3}$$

Definition A3

W^{H} designates the *Hermitian transpose of a matrix* $W = (w_{ij})$. $W^{\mathrm{H}} = (w_{ij}^*)$. W is called a *Hermitian matrix* if $W^{\mathrm{H}} = W$. If a matrix W is real, then W^{H} is the *transpose* of W (which we designate W^{T}), and Hermitian matrix W means *symmetric* matrix W, such that $w_{ij} = w_{ji}$ for all i, j.

Definition A4

A scalar value λ and a vector $\mathbf{v}$ form the *eigenvalue–eigenvector* pair for a matrix W if $\mathbf{v} \neq 0$ and $W\mathbf{v} = \lambda \mathbf{v}$. The absolute value $|\lambda|$ of the absolutely largest eigenvalue of W is called the *spectral radius* of W and is denoted $\rho(W)$.

Definition A5

A matrix is called *Hermitian positive semidefinite* (or *Hermitian nonnegative definite*) if it can be represented as $W^{\mathrm{H}}W$ for some matrix W, is called *Hermitian positive definite* if in addition it is nonsingular, and is called *symmetric positive definite* if in addition it is real. [$W^{\mathrm{H}}W$ is Hermitian, since $(W^{\mathrm{H}}W)^{\mathrm{H}} = W^{\mathrm{H}}(W^{\mathrm{H}})^{\mathrm{H}} = W^{\mathrm{H}}W$ (compare Definition A3).]

In the expressions

$$\sum_i, \quad \sum_j, \quad \max_i, \quad \max_j, \quad \max_{i,j}$$

the integer parameters i and j range from 1 to n.