# The role of arithmetic in fast parallel matrix inversion

Bruno Codenotti* Mauro Leoncini†
Franco P. Preparata‡

**Abstract**

In the last two decades several NC algorithms for solving basic linear algebraic problems have appeared in the literature. This interest was clearly motivated by the emergence of a parallel computing technology and by the wide applicability of matrix computations. The traditionally adopted computation model, however, ignores the arithmetic aspects of the applications, and no analysis is currently available demonstrating the concrete feasibility of many of the known fast methods. In this paper we give strong evidence to the contrary, on the sole basis of the issue of robustness, indicating that some theoretically brilliant solutions fail the severe test of the "Engineering of Algorithms". We perform a comparative analysis of several well known numerical matrix inversion algorithms under both fixed- and variable-precision models of arithmetic. We show that, for most methods investigated, a typical input leads to poor numerical performance, and that in the exact-arithmetic setting no benefit derives from conditions usually deemed favorable in standard scientific computing. Under these circumstances, the only algorithm admitting sufficiently accurate NC implementations is Newton's iterative method, and the word size required to guarantee worst-case correctness appears to be the critical

*Istituto di Matematica Computazionale, Consiglio Nazionale delle Ricerche, Via S. Maria 46, 56126 Pisa, Italy.

†Facoltà di Economia di Foggia, Università di Bari, Via IV Novembre 1, 71100 Foggia, Italy, and IMC-CNR, 56126 Pisa, Italy.

‡Dept. of Computer Science, Brown University, Providence, Rhode Island 02912. Supported in part by NSF Grant CCR94 -00232.

complexity measure. Our analysis also accounts for the observed instability of the considered superfast methods when implemented with the same floating-point arithmetic that is perfectly adequate for the fixed-precision approach.

# 1 Introduction

In the last two decades several fast parallel methods for computing the inverse of a matrix, and solving related linear algebraic problems, have appeared in the literature. This interest was clearly motivated by the emergence of a parallel computing technology and by the importance of the problems. Indeed, it is well known that, besides the applications to scientific computing, the efficient solution of many combinatorial problems "boils down to linear algebra" [20].

The conception of NC (polylogarithmic time) parallelizations of such applications under the requirement of work (processor) efficiency is an issue hardly worth any justification. As is typical in algorithmic design, the first conceptual step is the definition of the model of computation, as a compromise between complexity and fidelity to the real-world situation. The resulting simplifications, however, may ignore some critical feature of the problem, as we shall illustrate below. While a conveniently simplified model is a powerful scientific tool, because it enables the elucidation of the interdependence of the various operations, the translation of an algorithm into an executable program is an engineering task, where all relevant features must be carefully scrutinized. Besides the legitimate concerns caused by the high operation count, a most serious concern is the impact of arithmetic considerations on the feasibility of the algorithms, an issue structurally concealed by the nature of the models. Such considerations are relevant to a number of areas not purely of a combinatorial nature. A major example is Computational Geometry, where recently the recognition of the crucial role of arithmetic has prompted a major re-examination of standard algorithms to ensure their robustness. In Numerical Linear Algebra the corresponding question of the quality of the results can be investigated from two complementary viewpoints. Consider the matrix inversion problem, which is important in its own right and, more significantly, for its relevance to the solution of linear systems of equations. Starting from an exactly representable matrix, i.e., an integer matrix, one can evaluate the number of digits required to com-

pute its exact *rational* inverse, to be presented as the ratio of adjoint and determinant, using exact arithmetic over the integers or over their modular representations. (Recourse to modular arithmetic, with Chinese remaindering, may alleviate the word-size requirements, but not in a significant way – see Section 3.) The alternative is to resort to an arithmetic on fixed-length representations and to develop upper bounds to the roundoff error affecting the computed approximate inverse. The latter approach is typical of numerical analysis investigations, and indeed numerical analysts have developed sophisticated techniques to bound the roundoff error. In this approach numbers are typically represented as floating-point *reals* (approximated with a fixed-size representation).

In this paper we consider some of the most significant and best known fast parallel matrix inversion algorithms, and develop a comparative analysis of their numerical behavior in both models mentioned above (exact arithmetic, or rationals, and fixed-precision arithmetic, or approximate reals). As the title of the paper indicates, we restrict ourselves to numerical applications. This investigation is motivated by the lack of precise information available on the numerical aspects of some algorithms running in polylogarithmic parallel time. In particular, our analysis considers the now classical Csanky's algorithm [4], a recursive factorization (RF) algorithm independently proposed by Reif [29] and by Bini and Pan [1](page 357), Gaussian elimination with no pivoting [2], and Newton's iterative method [24, 25, 27]. This selection covers a wide spectrum of features, and is (in some sense) a complete one[1]. Csanky's algorithm, the easiest to describe and to program, is representative of a class of methods which resort to the computation of the characteristic polynomial or matrix powers (see, e.g., [1, 11, 22] and the many references contained therein)[2]. The RF method is the first fast parallel algorithm which avoids the computation of the characteristic polynomial – which is known to be numerically unstable. Gaussian elimination (or GE for short) has been analyzed from both the variable and fixed-precision viewpoints. In sequential applications, GE is most frequently applied with the partial pivoting fea-

[1]We exclude here the group of nonnumerical algorithms devised for computations over abstract fields [1, 10, 15, 16, 8].

[2]The published research on the various aspects of matrix inversion in the context of parallel computation is almost monumental, and no pretense of adequate referencing is claimed in this paper. The interested reader, wishing to delve into the mathematical and computational issues of the problem, should refer to the archival work of researchers such as Pan, von zur Gathen, Bini, cited in the Bibliography.

ture, which provides good numerical behavior, but there is widespread belief [33, 17] that this accurate version of GE does not admit NC implementation. On the other hand, GE with no pivoting admits an NC-parallelization [2], but the ensuing computational penalty may doom its practicability. Finally, Newton's method, which can be implemented as an NC computation when the input is a matrix with $n^{O(1)}$ condition number, covers the class of iterative solvers, and has been proposed by Pan and Reif [21, 24] for fast parallel matrix inversion, both as a stand-alone technique and as a building block of other methods.

One peculiarity of linear algebraic problems, of which matrix inversion is paradigmatic, is the large amount of data involved and, correspondingly, the large number of arithmetic and boolean operations to be executed. Our analysis shows that, even for well conditioned matrices (the case traditionally favored by computational scientists), such large amount of data may make the variable-precision arithmetic approach (which is designed to yield exact rational output) practically infeasible. Indeed, the word size $w$ required to guarantee worst-case correctness may be the critical complexity measure, with reference both to storage requirements and to the execution time of the arithmetic operations. In particular, we give evidence to the fact that in the exact-arithmetic methods no benefit derives from conditions usually deemed favorable in standard scientific computing. This state-of-affairs is clearly brought to the fore by the analysis of GE. The word size requirement of the exact-arithmetic implementation of GE depends on the magnitude of the determinants of certain submatrices of the input matrix, and this is usually very large even for the *simplest* cases. On the other hand, the roundoff analysis of the fixed-precision version of GE is much more informative, revealing that the potential instabilities for the exact-arithmetic approaches do not necessarily affect the global numerical behavior. Our analysis also accounts for the observed instability of the considered superfast methods (typically, Csanky's algorithm [5]) when implemented with the same floating-point arithmetic that is perfectly adequate for the fixed-precision approach.

The results of our analysis are summarized in Table 1. The first three rows in the table refer to exact-arithmetic methods (rational output), and the last three rows refer to approximate methods (approximate real output). In the latter cases, integer $d$, the adopted *precision*, denotes the number of bits of the significand. The *ideal bound* is obtained by a sensitivity analysis of the problem of computing the inverse matrix (see Section 2). As such, it gives a minimum word size requirement which applies to all matrix in-

version algorithms (see Section 2 for the details). The quantities $\kappa(\cdot)$ and $\rho$ denote the matrix condition number[3] and the growth factor, respectively. While the condition number, which measures the sensitivity of the solution to perturbations in the matrix coefficients, is an algorithm-independent notion, $\rho$ is a quantity which measures how large the numbers generated by the computation become with respect to the magnitude of the elements of the input matrix [12]. Note that the result on approximate GE is reported for comparison purposes, with the *caveat* that no NC implementations are known for this approach with the stated numerical properties. Altogether, Table 1 reveals that the only known algorithm admitting sufficiently accurate NC implementations is Newton's method. It must be emphasized that our analysis is limited to the impact of the arithmetic requirements and does not refer to any other aspect of parallel algorithm design, such as the focal problems of work and processor efficiency and complexity of interprocessor communication.

| | |
|---|---|
| Csanky | $n \log n$ |
| RF | $\geq n \log n$ |
| Exact GE | $\geq n$ |
| Ideal | $d + \log \kappa(\cdot)$ |
| Approx. GE | $\leq d + \log n + \log \rho + \log \kappa(\cdot)$ |
| Newton | $\leq d + \log n + \log \kappa(\cdot)$ |

Table 1: Word size requirements of the algorithms considered. $n$ is the order of the input matrix, whose elements are assumed to be representable with $d$ bits(with $n \geq d$).

The entries of the table only exhibit the *structure* of the bounds and not their actual analytic form. In particular, constant factors are neglected, because they do not contribute to the conclusions that can be drawn.

Summarizing, we show that:

**(i)** all the algorithms require arithmetic capabilities in excess of the ideal bound;

[3]The condition number will always be defined with respect to the *spectral norm* (see the Appendix for the basic definitions).

**(ii)** however, while the algorithms of Csanky and Reif, as well as exact Gaussian elimination (without pivoting), always require word sizes superlinear in $n$, approximate Gaussian elimination and Newton's method exhibit comparable undesirable behavior only for exponential values of the condition number and/or the growth factor.

Our conclusion is that exact algorithms are unsuited to produce the approximate results that are adequate in the practice of scientific computations. Moreover, in the numerical model, the known NC implementations of matrix inversion algorithms are not reliable due to their instability, which occurs even in the most favorable case of well-conditioned input matrices; therefore, despite their high intellectual value, they are not expected to be practicable. The "Engineering of Algorithms" viewpoint appears to single out Newton's method as the only one to bear promise for practical NC implementations of the inversion of general dense matrices.

The rest of this paper is organized as follows. In Section 2 we show an example which highlights the major drawbacks of the exact-arithmetic approach. In Section 3 we present our analysis of exact NC inversion methods. In Section 4 we develop the error analysis of Newton's method under the floating-point model of arithmetic, which shows that sufficient accuracy can be obtained by using a reasonably small word size. Finally, we refer the reader to the Appendix, which illustrates the adopted notation as well as some background on number representations and matrix norms.

# 2 A motivating example

Consider the following tridiagonal matrix:

$$A_n = \begin{pmatrix} M & -1 & & & \\ -1 & M & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & M & -1 \\ & & & -1 & M \end{pmatrix}, \tag{1}$$

with $M$ an integer $> 2$. We show that, while the elements of an approximate inverse can be represented using a number of bits which depends on the desired output precision only, the exact representation of a selected entry of $A_n^{-1}$ needs $\Omega(n)$ bits in general. Let $B_n = (b_{ij})$ denote the adjoint of $A_n$, so

that $A_n^{-1} = \frac{1}{\det(A_n)} B_n$, and let $d_n = \det(A_n)$. We show that $b_{11}$ is large, $d_n$ is large, and $\gcd(b_{11}, d_n) = 1$, i.e., that, although $\frac{b_{11}}{d_n}$ might be very small, its exact computation requires the computation of two large quantities.

It is easy to see that $d_n$ satisfies the recurrence $d_n = M d_{n-1} - d_{n-2}$. Using this recurrence it is easy to prove that $d_n > (M-1)^n$. Since $b_{11} = d_{n-1}$, what remains to prove is that $\gcd(d_n, d_{n-1}) = 1$. The proof is by induction. For $n = 2$ we get $\gcd(M^2 - 1, M) = 1$. For $n > 2$ we have,

$$\begin{aligned} \gcd(d_n, d_{n-1}) &= \gcd(d_n \bmod d_{n-1}, d_{n-1}) \\ &= \gcd((M d_{n-1} - d_{n-2}) \bmod d_{n-1}, d_{n-1}) \\ &= \gcd((-d_{n-2}) \bmod d_{n-1}, d_{n-1}) \\ &= \gcd(d_{n-1} - d_{n-2}, d_{n-1}). \end{aligned}$$

By the inductive assumption, $\gcd(d_{n-1}, d_{n-2}) = 1$, hence if $h$ divides $d_{n-1}$ (with $hk = d_{n-1}$), then $\frac{d_{n-1} - d_{n-2}}{h} = k - \frac{d_{n-2}}{h}$ cannot be an integer, unless $h = 1$. This implies that $\gcd(d_{n-1} - d_{n-2}, d_{n-1}) = 1$. The fact that the fraction $\frac{d_n}{d_{n-1}}$ is irreducible implies that $b_{11}$ cannot be represented exactly with $o(n)$ bits.

On the other hand, the matrix $A_n$ has all the good properties which any numerical analyst could dream of. It is symmetric, positive definite (and strongly diagonally dominant) with all the eigenvalues in the open interval $(M-2, M+2)$. The latter property implies that the condition number, under the spectral norm, is bounded by $\frac{M+2}{M-2}$, which is independent of $n$. Also, $A_n^{-1}$ is positive with elements strictly bounded by 1. Finally, it is easy to check that GE (no pivoting) applied to $A_n$ is stable, with all the intermediate numbers generated bounded in magnitude by $M$. This fact can be expressed by saying that the growth factor $\rho$ is bounded by 1.

Why are the above properties very good from the numerical point of view? The reason is that the magnitude of the condition number and the growth factor (rather than the magnitude of the determinant) characterize the numerical behavior of GE under floating-point arithmetic. Actually, the condition number $\kappa(A)$ of a matrix $A$ affects any approximate method for computing $A^{-1}$, because it measures how sensitive $A^{-1}$ is to perturbations in $A$'s coefficients. Let us clarify these concepts.

For a given floating-point arithmetic $\mathcal{F}$ with working precision $\mu$, the quantity $\mu\kappa(A)$ can be regarded as an *unavoidable error*, i.e., one incurred by any methods for computing an approximate inverse $X$ of $A^{-1}$. In other

words, $\mu \leq \frac{\epsilon}{\kappa(A)}$ is deemed a necessary condition for the relative error $\frac{\|A^{-1}-X\|}{\|A^{-1}\|}$ to be bounded above by $\epsilon > 0$. We observe that a working precision $\mu$ implies that a number in $\mathcal{F}$ requires a significand of at least $\lceil \log \frac{1}{\mu} \rceil$ bits. Hence, the inequality $\mu \leq \frac{\epsilon}{\kappa(A)}$ requires that the word size $w$ satisfy

$$w = \Omega(\log \frac{1}{\epsilon} + \log \kappa(A)). \tag{2}$$

(2) is referred to as the "ideal" bound (see, e.g., [7]), against which the actual, usually larger, word size requirements of specific algorithms should be compared. In the case of our matrix $A_n$, the ideal bound is simply $\Omega(\log \frac{1}{\epsilon})$, which is also the output complexity.

Now, if we use GE to invert a generic nonsingular matrix $A$, standard error analysis yields the following bound (see Section 3.3 for the appropriate details)

$$w = O(\log \frac{1}{\epsilon} + \log n + \log \kappa(A) + \log \rho).$$

This bound reduces to $O(\log n + \log \frac{1}{\epsilon})$ for matrix $A_n$, which is essentially the best, given that $\Omega(\log \frac{1}{\epsilon})$ bits are required to represent the output, while $\Omega(\log n)$ bits are required for addressing the entries.

Suppose now that we wish to recover the exact inverse $A^{-1}$, i.e., the (rational) entries of $A^{-1}$ are expressed as ratio of integers. Obviously we must start from a very accurate approximation $X$, and we run again into the difficulties mentioned at the beginning of this section. More precisely, to recover the adjoint matrix, we must impose the condition

$$|\det(A)||X_{ij} - (A^{-1})_{ij}| < \frac{1}{2}, \tag{3}$$

so that straightforward rounding of $X_{ij} \det(A)$ yields the correct result. For any pair of indices $i, j$ such that $(A^{-1})_{ij} \neq 0$, we must then have

$$\frac{|X_{ij} - (A^{-1})_{ij}|}{|(A^{-1})_{ij}|} < \frac{1}{2|\det(A)||(A^{-1})_{ij}|} \tag{4}$$

In particular, (4) must hold for the indices $r, s$ such that $|(A^{-1})_{ij}|$ is maximum, so that we get

$$\frac{|X_{rs} - (A^{-1})_{rs}|}{|(A^{-1})_{rs}|} < \frac{\max_{ij} |A_{ij}|}{2|\det(A)|\kappa_M(A)}, \tag{5}$$

where $\kappa_M(A) = \max_{ij} |A_{ij}| \cdot \max_{ij} |(A^{-1})_{ij}|$ is the condition number according to the so-called *max* norm: $\|A\|_M = \max_{ij} |A_{ij}|$. It is well-known (cf. [13][page 122]) that $\|A\| \leq n\|A\|_M$ (and hence also $\kappa(A) \leq n^2\kappa_M(A)$). Together with the obvious bound $\max_{ij} |A_{ij}| \leq \|A\|$, this leads to the following inequality

$$\frac{\|X - (A^{-1})\|}{n\|A^{-1}\|} < \frac{|X_{rs} - (A^{-1})_{rs}|}{|(A^{-1})_{rs}|}. \tag{6}$$

(5) and (6) imply the upper bound $\epsilon = n\frac{\max_{ij} |A_{ij}|}{2|\det(A)|\kappa_M(A)}$ on the normwise relative error. Plugging $\epsilon$ into expression (2) for the ideal bound we obtain

$$w \geq 2\log\kappa(A) + \log|\det(A)| - \log\max_{ij}|A_{ij}| + 1 - 3\log n.$$

In the case of matrix $A_n$ we get $w = \Omega(n)$.

# 3 Exact-arithmetic (direct) methods

In analyzing the exact-arithmetic methods we will assume that the input matrix $A$ has $d$ bit integer entries and that the exact inverse sought has to be presented as the pair $\langle \mathrm{adj}(A), \det(A)\rangle$. Here $\mathrm{adj}(A)$ denotes the adjoint matrix of $A$, and it is clearly over the integers provided that so is $A$. For each method we will determine the word size sufficient to compute the inverse, using direct inspection of the magnitude of the integers generated during the computation. Although the algorithms of [4, 28, 10, 15, 16, 8] and several algorithms of [1] are applicable over abstract fields (unlike, for example, the cited algorithms of [21, 22, 1, 29, 30, 23]), due to the stated focus of our analysis, we shall restrict ourselves to the domain of the integers.

## 3.1 Csanky's algorithm

In this section we analyze the word-size requirements of Csanky's algorithm [4]. We recall that this algorithm computes $A^{-1}$ by means of Leverrier's inversion formula (22), which is based on Cayley-Hamilton's Theorem and therefore requires the explicit determination of the characteristic polynomial of $A$. The coefficients of the latter are obtained by solving a triangular system based on Newton's inequalities (see, e.g., [9]). After Csanky's original

paper, this algorithm has been widely studied from the important viewpoint of processor efficiency [28, 10, 14, 15, 16, 8]. Our present objective, however, is the impact of the arithmetic on the viability of the algorithm.

Csanky's algorithm is designed to compute the exact inverse, represented as the pair $< \mathrm{adj}(A), \det(A) >$. We assume that $A$ is of size $n$ with integer entries of maximum absolute value $M$. We shall prove that $\Theta(n \log n)$ bits are both necessary and sufficient to carry out the calculations.

Since Leverrier's method only involves addition and multiplication over the base ring in order to compute the output in rational form, it is appropriate to resort to calculations in modular arithmetic with Chinese Remaindering, the modulus being chosen not smaller than the maximum possible value of any output entry. This choice assures us of the correctness of the output integers. This maximum value is readily supplied by Hadamard's inequality $|\det(A)| \leq (M\sqrt{n})^n$ (see Appendix). Thus $(n/2)\log n + \log M$ bits are sufficient to execute the calculations correctly. A detailed analysis of the steps of Csanky's algorithm, omitted here, would reveal that execution in ordinary integer arithmetic can be carried out with comparable word-size resources.

Unfortunately, we also observe that a word size of the same order is also necessary, so that resort to modular arithmetic would not significantly affect the word size (as alluded to in the Introduction). Indeed, a constructive sequence of matrices whose determinants closely approach Hadamard's bound has been known for some time [3]. Referring, for simplicity, to matrices whose entries are all $+1$ or $-1$, a matrix of order $n$ can be constructed whose determinant has the absolute value

$$D_n = 2^{\sum_{i=0}^{n-1} w(i)} \tag{7}$$

where $w(i)$ is the number of 1's in the binary representation of the integer $i$. It has also been shown in [3] that

$$\sum_{i=0}^{n-1} w(i) > \frac{n}{2}\log n - \frac{n}{2}\log_2 \frac{4}{3}.$$

As a curiosity, we now present an alternative elementary proof of (7). It is simple to show that the matrices of [3] coincide with leading order-$n$ submatrices of standard order- $2^d$ Hadamard matrices(which are also known as Sylvester matrices). Let $H_n^*$ be the $n \times n$ leading matrix of $H_{2^d}$, $d = \lceil \log n \rceil$. The determinant of $H_n^*$ can be computed by expansion with respect to the $n_1 = n - 2^{d-1}$ bottom rows. This determinant is the sum of the (signed)

products of the determinant of each order-$n_1$ minor multiplied by its cofactor in $H_n^*$. These cofactors have value 0 (because of identical columns), except those corresponding to minors such that their $i$-th column is either the $i$-th or the $(i + 2^{d-1})$-th column of $K$. It can be verified that the signed values of these contributions are all positive. It follows that

$$D_n = 2^{n-2^{d-1}} \det(H_{2^{d-1}}) D_{n_1}.$$

Assuming inductively that $D_{n_1} = \sum_{i=0}^{n_1-1} w(i)$, since $\det(H_{2^{d-1}}) = (d-1)2^{d-2} = \sum_{i=0}^{2^{d-1}-1} w(i)$, the conclusion follows.

The above result confirms that about $(n/2)\log n + n \log M$ bits of word size are also necessary to correctly execute the algorithm over the integers in all cases.

A natural question is whether Csanky's algorithm may have better numerical behavior if we relax the requirement of exact rational results, accepting instead as results numbers that are approximate reals with a given precision. In other words, we ask whether we can profitably execute Csanky's algorithm with floating-point arithmetic. To our knowledge, no analysis of the algorithm under such model is available in the literature, and its full development is beyond our present objectives. We shall content ourselves, instead, with the circumstantial evidence that even simple matrices pose unacceptable word-size requirements if the target approximation is to be met. The following very coarse observations, coupled with significant experimentation by Demmel, expose the inherent inability of the method to capitalize on favorable numerical properties (such as small condition number).

Consider the particularly simple matrix, $A = kI$, with $k$ a small integer exceeding 2, and let $s$ be the minimum integer such that $k^s > 2^b$, where $b$ is the length of the significand. For $n < s$, the power $k^n$ is exactly representable. For larger values of $n$, the relative error affecting the computed power $k^j$, $j = s, \ldots, n$ is approximately $(j/s)2^{-b}$, provided that $k^j$ is computed as $(k^s)^q \cdot k^h$, where $j = s \cdot q + h$, $h = j \bmod s$. Now, the entries of the triangular linear system $Tc = b$ to be solved in order to recover the coefficients of the characteristic polynomial are the traces of $A^j = (kI)^j$, i.e., $nk^j$ for $j = 1, \ldots, n-1$, so that the actual entries along the $j$-th diagonal $(j \geq s)$ are affected by relative errors $n$ times as large as those affecting the powers $k^j$.

To get a fixed number $b'$ of correct bits in the solution $\hat{c}$ to the actual system $\hat{T}c = b$ the relative error $\|\hat{c} - c\|/\|c\|$ must not exceed $2^{-b'}$. The lack

of a tight estimate of $\|\hat{c} - c\|/\|c\|$ forces one to resort to an upper bound to it. According to standard perturbation analysis[4], we have, to first order,

$$\frac{\|\hat{c} - c\|}{\|c\|} < \kappa(T) \frac{\|\hat{T} - T\|}{\|T\|}. \tag{8}$$

Since $\kappa(T) = \|T\|.\|T^{-1}\|$ and $\|T^{-1}\| \geq 1$, the above bound may be replaced by the stricter requirement $\|\hat{T} - T\|$. Using the standard inequality $\|A\| \leq \sqrt{n}\|A\|_\infty$, and the observation that $\|\hat{T} - T\|_\infty$ is given by the sum of its last row terms, we obtain

$$\|\hat{T} - T\| \approx \sqrt{n}.n \sum_{j=s}^{n-1} \frac{j}{s} k^{j-s} = n^{3/2} 2^{-b} \sum_{j=s}^{n-1} \frac{j}{s} k^j \approx n^{5/2} 2^{-b} k^n.$$

Thus the latter quantity must be bounded above by $2^{-b'}$, indicating that it is sufficient to adopt a word length $b$ as large as the order of the matrix.

The outcome of the above analysis is confirmed by some experiments carried using double-precision (i.e., $b = 53$) floating-point arithmetic (IEEE Standard 754). Demmel reports that, for $k = 3$, Csanky's algorithm loses all precision for $n$ as small as 60 [5].

## 3.2 A Recursive Factorization (RF) algorithm

In [29] Reif and in [1] Bini and Pan present $O(\log^2 n)$-time processor-efficient parallel algorithms to compute the exact $LU$ factorization of dense matrices with integer entries. These algorithms combine the techniques of variable diagonal and $p-$adic lifting, various apects of which are discussed in [21, 22]. The key feature of such algorithms is that they avoid the computation of the characteristic polynomial, a process that is known to be generally unstable. By means of a simple analysis, however, we shall conclude that Reif's algorithm still requires a very large word size $w = \Omega(n \log n)$.

The RF algorithm is based on the computation of the so called *Recursive Factorization Tree* (or simply *RF* tree) of a matrix as a vehicle for obtaining the $LU$ factorization. We now briefly review the approach for the reader's convenience. Let $A$ be a matrix of order $n = 2^d$ with integer entries of maximum magnitude $M$. Its $RF$ tree (if it exists) is a full binary tree of depth $d$ whose nodes are indexed by a pair $(k, \alpha)$, where $k \in \{0, \ldots, d\}$ is

[4]For simplicity, we ignore the errors affecting the right-hand side vector $b$.

the level and $\alpha$ is a binary string of length $k$ indexing, as a binary integer, the nodes at level $k$ from left to right. Node $(k, \alpha)$ is associated with an $(n/2^k) \times (n/2^k)$ matrix $A_{k,\alpha}$. Clearly, the root of the tree is labeled by $(0, \lambda)$, where $\lambda$ is the empty string, and is associated with the matrix $A_{0,\lambda} = A$. For a given pair $(k, \alpha)$, let

$$A_{k,\alpha} = \begin{pmatrix} W_{k,\alpha} & X_{k,\alpha} \\ Y_{k,\alpha} & Z_{k,\alpha} \end{pmatrix}.$$

The matrices of the nodes of the $RF$ tree are recursively defined as follows:

$$A_{k+1,\alpha 0} = W_{k,\alpha} \qquad \text{and} \qquad A_{k+1,\alpha 1} = Z_{k,\alpha} - Y_{k,\alpha} W_{k,\alpha}^{-1} X_{k,\alpha}.$$

In other words, the left and right children of $A_{k,\alpha}$ are respectively associated with its leading principal submatrix $W_{k,\alpha}$ of order $\frac{n}{2^{k+1}}$, and the Schur's complement of $W_{k,\alpha}$ in $A_{k,\alpha}$.

RF algorithm consists of two distinct phases. In the first phase, matrix $A$ is replaced by a related matrix $\bar{A}$, better suited for the subsequent computations, and the $RF$ tree of $\bar{A}$ is computed; this phase also provides the inverse $\bar{A}^{-1}$ of $\bar{A}$. In the second phase $A^{-1}$ is recovered from $\bar{A}^{-1}$ as a pair $< \mathrm{adj}(A), \det(A) >$.

**Algorithm RF**
(**First phase**)

1. Set $\bar{A} = A + cI$.

   *Comment*: Here $c = c(n, M)$ must be chosen sufficiently large to ensure the so-called extreme diagonal dominance, i.e., for any $i = 1, \ldots, n$,

   $$\epsilon |\bar{a}_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |\bar{a}_{ij}|,$$

   for some positive small $\epsilon$. Diagonal dominance is needed to ensure fast convergence of the Newton's iterations of step 2. The constant $c$ is also chosen as a multiple of a large prime $p$ with the property that, if $\det(A) \neq 0$, then $\det(A) \not\equiv 0 \bmod p$. To this end, Reif suggests $p > (n\|A\|)^{2n}$. Note that $\bar{A} - A$ is an integer multiple of $pI$, which implies that $\bar{A}^{-1}$ and $A^{-1}$ (provided it exists) are congruent modulo $p$.

2. Compute an approximate $RF$ of $\bar{A}$ by proceeding down the tree in $\log n$ stages. Each left node supplies an approximate inverse of its matrix to its right sibling.

   *Comment*: The inverse of $\bar{A}_{k,\alpha}$ is approximate because this step is based on Newton's iterations to accuracy $\delta$ (a parameter to be selected later)[5].

3. For each node $(k, \alpha)$ of the $RF$ compute the exact value of $\bar{A}_{k,\alpha}$.

   *Comment*: One must compute exactly as integer entries the pair $<$ adj$(\bar{A}_{k,\alpha})$, det$(\bar{A}_{k,\alpha})$ $>$. This result is obtained in the following two steps:

   (a) Compute det$(\bar{A}_{k,\alpha})$ by rounding the approximate value of det$(\bar{A}_{k,\alpha})$ obtained by the Newton iteration to the nearest integer.

   *Comment*: To determine the accuracy required by Newton's iterations, define $\epsilon_{ij}$ as the absolute error affecting the $(i,j)$th entry of $\bar{A}_{k,\alpha}$. A standard first order analysis then shows that the perturbation error $\epsilon_{\text{abs}}$ affecting the (approximate) determinant is

$$\epsilon_{\text{abs}} = \sum_{ij} (-1)^{i+j} \det \bar{A}_{ij} \epsilon_{ij},$$

   where, for simplicity of notation, we have omitted the reference to $k$ and $\alpha$. Using the bound (21)(see Appendix), we see that $|\epsilon_{\text{abs}}|$ can be as large as $n^2 (M\sqrt{n-1})^{n-1} \max_{ij} |\epsilon_{ij}|$. To be able to round the approximate determinant to the exact integer value, we must require that $|\epsilon_{\text{abs}}| < \frac{1}{2}$ and this leads to

$$\delta = \max_{ij} |\epsilon_{ij}| < \frac{1}{2n^2 (M\sqrt{n-1})^{n-1}}.$$

   (b) Compute adj$(\bar{A}_{k,\alpha})$ (a matrix of integers) exactly. This is done by multiplying each entry of the approximate $\bar{A}_{k,\alpha}^{-1}$ by the exact value of det$(\bar{A}_{k,\alpha})$ and rounding to the nearest integer.

[5] As described here, this step is easily seen to require $O(\log^3 n)$ time on $O(P(n))$ processors, where $P(n)$ processors suffice to perform matrix multiplication in time $O(\log n)$. By "pipelining" the tree traversal along the line of [26], Reif in [29] improves the bound to $O(\log^2 n)$. However the simpler description is adequate for our purpose, which is the analysis of the arithmetic requirements.

(**Second phase**)

4. For each node $(k, \alpha)$ of the $RF$ reduce $\bar{A}^{-1}_{k,\alpha}$ mod $p$, which coincides with $A^{-1}_{k,\alpha}$ mod $p$.

   *Comment:* Note that $A^{-1}_{k,\alpha}$ mod $p$ as obtained is a matrix of integers. To obtain $A^{-1}_{k,\alpha}$ as a pair $\langle \mathrm{adj}(A_{k,\alpha}), \det(A_{k,\alpha})\rangle$, we must compute $A^{-1}_{k,\alpha}$ mod $p^u$ where $p^u$ is a power of the prime $p$ not smaller than the maximum of the absolute value of $\det(A)$, i.e., $(M\sqrt{n})^n$. This is accomplished by Newton-Hensel lifting [19] in the next step.

5. Execute $t$ iterations of the Newton-Hensel lifting starting from $A^{-1}$ mod $p$ to obtain $A^{-1}$mod $p^{2^t}$, for $t = c \log n$.

   *Comment:* The matrix $A^{-1}$mod $p^{2^t}$, which is of course a matrix of integers, contains all information necessary to readily derive $A^{-1}$. In fact, since $A^{-1} = (\mathrm{adj}A/\det(A))$, in order to obtain the $(i, j)$-entry of $\mathrm{adj}(A)$ we must multiply the the $(i, j)$-entry of $A^{-1}$mod $p^{2^t}$ by the value of $\det(A)$ computed above.

We now analyze the word requirements of the just outlined algorithm. First of all we observe that, differently from Csanky's algorithm, recourse to modular arithmetic is infeasible, for the integer results are recovered from non-integer approximation supplied by Newton's iteration. Next, we recall that, when recovering the exact inverse from an approximation of it, the word size is bounded below by $\log \kappa()$. To guarantee that $\kappa(\bar{A})$ be small, $c$ must be very large ($c > p(n\|A\|_\infty)^{2n}$, where $\|A\|_\infty$ can be as large as $nM$). Thus the word size requirement of the algorithm is at least $w \geq 20n \log n + 10n \log M$, which even exceeds the requirement of the previously analyzed Csanky's algorithm. In his paper, Reif claims that the word size requirement can be lowered, by choosing a much smaller quantity $c$, and a nontrivial modification due to Pan [23] achieves an upper bound $w \approx n \log n$. Not much better result was to be expected due to the lower bound on the size of entries derived in the preceding section.

The conclusion of this analysis is that, although the RF algorithm avoids the computation of the characteristic polynomial (which is conventionally viewed as a source of numerical instability), the recourse to extreme diagonal dominance, as a device to achieve stability, imposes word size requirements analogous to those for which Csanky's algorithm can be executed exactly.

## 3.3 Gaussian Elimination

The classical Gaussian elimination algorithm computes the $LU$ decomposition of $A$, i.e., it determines two matrices $L$ and $U$ such that:

1. $L$ is lower triangular with 1's on the main diagonal;
2. $U$ is upper triangular;
3. $A = LU$.

The $LU$ decomposition exists (and is unique) provided that the leading principal minors of $A$ of order $i$ are nonsingular, for $i = 1, \ldots, n-1$.

Once the $LU$ decomposition of $A$ is available, it is well known that solution of a linear system $Ax = b$, as well as computation of $A^{-1}$ and of $\det(A)$ become straighforward operations.

The NC algorithm given in [2] is based on classical GE, and computes the determinant of an integer matrix. It is highly inefficient in terms of the overall work, as it performs $O(n^{15})$ operations, but again we are not concerned with its inefficiency. Rather, we wish to highlight the large word size demand of the algorithm when run under the exact-arithmetic model, and to contrast this result with the fact that (usually) good accuracy can be obtained by using finite-precision floating-point implementations of GE.

We now briefly review the algorithm of [2]. The idea is to develop a sequential, division-free version of GE for the computation of the determinant of the following matrix

$$\begin{pmatrix} 1-x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & 1-x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ x_{n1} & x_{n2} & \ldots & 1-x_{nn} \end{pmatrix}, \tag{9}$$

in which the $x_{ij}$ are integer variables. The sequential program can then be parallelized using the general technique of Valiant et al. [32]. Note that, by setting $x_{ii} = 1 - a_{ii}$ and $x_{ij} = a_{ij}$, the algorithm computes $\det(A)$.

An important property of the approach is that it is division-free, since all divisions performed by GE on the input matrix $X$ are of the form $\frac{w}{1-z}$, where $w$ and $z$ are polynomials in the $x_{ij}$'s, and $z$ has constant term 0. Under these circumstances, and knowing that the result is a degree-$n$ polynomial in the $x_{ij}$'s, the division by $1 - z$ can be replaced by the multiplication

by $1 + z + \ldots + z^n$. This property is crucial for applying to this case the considerations developed in connection with Csanky's algorithm, i.e., the ability to resort to modular arithmetic with reference to the maximum entry size provided by Hadamard's inequality. Thus a word size of $\Theta(n \log n)$ bits is in order, even though the ratio of huge numerators and denominators could be represented to high accuracy using relatively few bits.

The above situation is in fact in sharp contrast with the known results about the stability of Gaussian elimination under the floating-point model of arithmetic, which we now briefly recall. The results are taken essentially from [6], to which we address the reader interested in the details. A more comprehensive study can be found in the excellent text [13].

The analysis in [6] applies to the block version of Gaussian elimination, which generalizes the classical, scalar GE. The input matrix $A$ is viewed as a $2 \times 2$ block-partitioned matrix and the same partitioning applies to the (block) $L$ and $U$ factors. With obvious meaning of the indices, we have the following:

**Algorithm BLU**[6]

1. Set $U_{11} = A_{11}$ and $U_{12} = A_{12}$.
2. Compute $L_{21} = A_{21}A_{11}^{-1}$.
3. Set $S = A_{22} - L_{21}A_{12}$ ($S$ is a Schur's complement of of $A_{11}$ in $A$).
4. Recursively compute the block $LU$ factorization of $S$.

We now turn our attention to the numerical accuracy of GE, to which the following result applies:

**Theorem 1** *[6] Suppose that the inverse matrix needed at step 2 of* **BLU** *is computed exactly, and let $\hat{L}\hat{U}$ be the actual factorization returned by the algorithm. Then $\hat{L}\hat{U} = A + H$ with $H = (h_{ij})$ such that*

$$\max_{1 \le i,j \le n} |h_{ij}| \le c_n \mu M^3 \rho^4 M'^2 + O(\mu^2).$$

*where $M'$ denotes the largest element of $A^{-1}$ and $c_n$ is a quantity bounded by a small polynomial in $n$.*

[6] We note that the block algorithm offers potential speed-ups, by using an NC algorithm to compute $A_{11}^{-1}$. In light of the preceding analyses, this could be numerically feasible provided that the size of the block is sufficiently small.

Note that the quantity $MM'$ gives a way to measure the condition number $\kappa(A)$ of the matrix $A$[7]. Hence, the result of Theorem 1 can be restated as follows: if the input matrix is well conditioned *and* the growth factor $\rho$ is sufficiently small, then algorithm BLU computes the exact factorization of a perturbed matrix $A+H$ which is sufficiently close to $A$. Results like the one of Theorem 1 are known as *backward error bounds*.

We only mention the fact that the subsequent computation of the inverse, performed with any of the block triangular solvers available, does not worsen the quality of the bound given by Theorem 1. We can say in conclusion that the computation of the inverse through Gaussian elimination determines a matrix $X$ which is the exact inverse of $A+E$, with

$$\|E\| \leq d_n \|A\| \kappa(A)^\alpha \rho^\beta \mu + O(\mu^2), \tag{10}$$

where $d_n$ is a slowly growing polynomial in $n$ and $\alpha$ and $\beta$ are constants.

Admittedly, the bound (10) (obtained by backward error analysis [13, 34, 35]) is not accurate. However, this is perfectly in line with the philosophy of roundoff analysis (see [35][p. 567] or [12][p. 64]), whose main objective is to "expose the potential instabilities, if any, of an algorithm". The precise bound is usually not very interesting because it is unavoidably weak (e.g., due to the repeated application of inequalities like $\|AB\| \leq \|A\|\|B\|$, which are almost always very pessimistic). In our case, the occurrence of an exponential condition number and/or exponential growth factor represent the potential instabilities mentioned above. However, unlike the determinant, which is commonly very large, the condition number and the growth factor are commonly small.

The backward bound (10) can be turned into a bound on the arithmetic by using standard perturbation analysis techniques [12]. In particular, if $\delta$ is such that $\|E\| \leq \delta \|A\|$, then it can be proved that

$$\frac{\|A^{-1} - X\|}{\|A^{-1}\|} \leq \delta \kappa(A) + O(\delta^2).$$

Reasoning as in Section 2, we see that $\delta \leq \frac{\epsilon}{\kappa(A)}$ is required to keep the error $\frac{\|A^{-1}-X\|}{\|A^{-1}\|}$ smaller than $\epsilon$. Combining this with (10) (i.e., setting $\delta = d_n \kappa(A)^\alpha \rho^\beta \mu$), we see that we must have

$$\frac{1}{\mu} \leq \epsilon^{-1} d_n \kappa(A)^{\alpha+1} \rho^\beta \mu + O(\mu^2),$$

[7] More precisely, we have $\frac{1}{n^2}\kappa(A) \leq MM' \leq \kappa(A)$.

and hence

$$w = \Omega(\log n + \log \frac{1}{\epsilon} + \log \kappa(A) + \log \rho).$$

# 4 Newton's method

Let $A$ be a nonsingular matrix and consider the matrix equation

$$R(X) = I - XA = O.$$

Given any approximate inverse of $A$, i.e., a matrix $B$ satisfying $\|R(B)\| = q < 1$, Newton's iteration

$$\begin{cases} X_0 = B, \\ X_k = (2I - X_{k-1}A)X_{k-1}, \qquad k = 1, 2, \ldots \end{cases} \tag{11}$$

converges quadratically to $A^{-1}$ under real arithmetic (i.e., if $R(X_0) = q$ then $R(X_k) \leq q^{2^k}$). For the benefit of the reader, in the following we redo a complete analysis of the behavior of Newton's method. Except for the roundoff error investigation, distinct from ones in [27, 31], most of the material here can be found in [24, 25, 27, 31].

First of all there is the question of the initial approximation $B$. Following [24], one may choose

$$B = tA^T, \quad t = \frac{1}{\|A\|_1 \|A\|_\infty} = \frac{1}{\max_i \sum_j |a_{ij}| \max_j \sum_i |a_{ij}|}$$

(see [27] for some other possible choices). With this choice, since the matrix $I - tA^TA$ is symmetric, we have (see the Appendix)

$$q = \|I - tA^TA\| = \varrho(I - tA^TA).$$

Using the properties of symmetric matrices mentioned in the Appendix (in particular, $\|A^TA\| = \|A\|^2$) and inequality (23), we see that $\varrho(tA^TA) \leq 1$ so that $\varrho(I - tA^TA) = 1 - \lambda_{\min}(tA^TA)$. Finally, using (24) and the properties of the condition number of symmetric matrices (see again the end of the Appendix), we get

$$\begin{aligned} q &= 1 - \lambda_{\min}(tA^TA) \\ &= 1 - \frac{1}{\|A\|_1 \|A\|_\infty} \lambda_{\min}(A^TA) \\ &= 1 - \frac{1}{\|A\|_1 \|A\|_\infty} \frac{\lambda_{\max}(A^TA)}{\kappa(A^TA)} \\ &\leq 1 - \frac{1}{n\|A\|^2} \frac{\|A\|^2}{\kappa^2(A)} \\ &= 1 - \frac{1}{n\kappa^2(A)}. \end{aligned} \tag{12}$$

We now wish to determine the minimum word size sufficient for the *total* relative error to be bounded by $2^{-d}$ after $k$ steps, so that:

$$\frac{\|\mathrm{fl}(X)_k - A^{-1}\|}{\|A^{-1}\|} \leq 2^{-d}. \tag{13}$$

Observe that

$$\|\mathrm{fl}(X_k) - A^{-1}\| \leq \|\mathrm{fl}(X_k) - X_k\| + \|X_k - A^{-1}\| = E_R(k) + E_A(k),$$

where $E_R(k) = \|\mathrm{fl}(X_k) - X_k)\|$ is the absolute *roundoff* (or *algorithmic*) error at step $k$ and $E_A(k) = \|X_k - A^{-1}\|$ is the absolute *truncation* (or *analytic*) error at step $k$. Goal (13) can be achieved if we ensure that

$$E_A(k) \leq 2^{-(d+1)}\|A^{-1}\| \tag{14}$$

and

$$E_R(k) \leq 2^{-(d+1)}\|A^{-1}\|. \tag{15}$$

Our approach is the following. We analyze the number $k$ of steps sufficient to achieve (14) under real arithmetic, and then determine the word size sufficient for (15) to hold after $k$ steps as well.

The quadratic rate of convergence of Newton's iterations means that

$$E_A(k) = \|X_k - A^{-1}\| \leq q^{2^k}\|A^{-1}\|. \tag{16}$$

Hence, under real arithmetic, $k = O(\log d - \log\log \frac{1}{q})$ steps suffice to reduce the relative truncation to no more than $2^{-(d+1)}$.

In order to analyze the roundoff error, letting $R_k \stackrel{\triangle}{=} R(X_k)$, we begin by observing that:

$$\begin{aligned} X_k &= (2I - X_{k-1}A)X_{k-1} \\ &= (I + R_{k-1})X_{k-1} \\ &= X_{k-1} + R_{k-1}X_{k-1}, \end{aligned}$$

and

$$\begin{aligned} R_k &= I - X_k A \\ &= I - (X_{k-1} + R_{k-1}X_{k-1})A \\ &= I - X_{k-1}A - R_{k-1}X_{k-1}A \\ &= R_{k-1}(I - X_{k-1}A) \\ &= R_{k-1}^2. \end{aligned}$$

Hence we may refer to the following implementation of Newton's iteration:

$$\begin{cases} X_k := X_{k-1} + R_{k-1}X_{k-1} \\ \\ R_k := R_{k-1}^2. \end{cases} \qquad k = 1, 2, \ldots \tag{17}$$

In what follows, $\mathrm{fl}(x)$ will denote the value of $x$ computed by floating-point arithmetic with roundoff unit $\mu$. We first recall the following lemma, whose proof can be found in a number of textbooks, notably [34].

**Lemma 1** *Let $A$ and $B$ be matrices of order $n$, and let $n\mu < 0.1$. Then*

$$\|\mathrm{fl}(AB) - AB\| \leq n^2\mu\|A\|\,\|B\| + O(\mu^2),$$

*and*

$$\|\mathrm{fl}(A+B) - (A+B)\| \leq \sqrt{n}\mu\|A+B\| + O(\mu^2).$$

**Theorem 2** *Let $\Delta R_k = \mathrm{fl}(R_k) - R_k$ be the absolute error affecting the value $\mathrm{fl}(R_k)$ computed according to (17). Then*

$$\|\Delta R_k\| \leq n^2\mu(2^k - 1)q^{2^k}.$$

Proof: The proof is an easy induction. We clearly have $\Delta R_0 = O$. Now, executing $R_{k+1} := R_k^2$ in floating-point we have:

$$\mathrm{fl}(R_{k+1}) = \mathrm{fl}(()\mathrm{fl}(()R_k)^2) = \mathrm{fl}(R_k)^2 + M_k,$$

where, by Lemma 1, $\|M_k\| \leq n^2\mu\|fl(R_k)\|^2 \approx n^2\mu\|R_k\|^2 \leq n^2\mu q^{2^{k+1}}$. Hence, using (17), we obtain for $k > 0$:

$$\begin{aligned} \Delta R_{k+1} &= \mathrm{fl}(R_{k+1}) - R_{k+1} \\ &= \mathrm{fl}(()r_k)^2 + M_k - R_{k+1} \\ &= (R_k + \Delta R_k)^2 + M_k - R_{k+1} \\ &= R_k\Delta R_k + \Delta R_k R_k + (\Delta R_k)^2 + M_k, \end{aligned}$$

Taking norms we get

$$\|\Delta R_{k+1}\| \leq 2\|R_k\|\|\Delta R_k\| + \|\Delta R_k\|^2 + \|M_k\|.$$

Finally, neglecting $\|\Delta R_k\|^2 = \Theta(\mu^2)$ and using the inductive hypothesis we obtain

$$\begin{aligned} \|\Delta R_{k+1}\| &\leq 2.q^{2^k}.(n^2\mu(2^k-1)q^{2^k}) + n^2\mu q^{2^{k+1}} \\ &= n^2\mu(2(2^k-1)q^{2^{k+1}} + q^{2^{k+1}}) \\ &= n^2\mu(2^{k+1}-1)q^{2^{k+1}}. \end{aligned}$$

$\diamond$

**Theorem 3** *Let $\Delta X_k = \mathrm{fl}(X_k) - X_k$ be the absolute error affecting the value $\mathrm{fl}(X_k)$ computed according to (17). Then*

$$E_R(k+1) < \mu\|A^{-1}\|2^k(kn^2 + 2\sqrt{n}).$$

Proof: We have

$$\mathrm{fl}(X_{k+1}) = \mathrm{fl}(R_k)\mathrm{fl}(X_k) + \mathrm{fl}(X_k) + M'_k + A'_k, \tag{18}$$

where $M'_k$ and $A'_k$ are the error terms due to matrix multiplication and addition, respectively. By Lemma 1,we have

$$\begin{aligned} \|M'_k\| &\leq n^2\mu\|\mathrm{fl}(R_k)\|\|\mathrm{fl}(X_k)\| \\ &\leq n^2\mu\|R_k\|\,\|X_k\| + O(\mu^2) \\ &\leq n^2\mu q^{2^k}\|A^{-1}\| + O(\mu^2), \end{aligned}$$

and

$$\|A'_k\| \leq \sqrt{n}\mu(\|X_{k+1}\| + O(\mu)) \leq \sqrt{n}\mu\|A^{-1}\| + O(\mu^2).$$

Using (18) we get

$$X_{k+1}+\Delta X_{k+1} = R_kX_k+R_k\Delta X_k+\Delta R_kX_k+\Delta R_k\Delta X_k+X_k+\Delta X_k+M_k+A_k.$$

Hence, using (17) and neglecting the terms $O(\mu^2)$, we may write

$$\begin{aligned} \Delta X_1 &\approx M'_0 + A'_0, \\ \Delta X_{k+1} &\approx (I+R_k)\Delta X_k + \Delta R_k X_k + M'_k + A'_k, \quad k > 0, \end{aligned}$$

Hence, for $k > 0$, we see that the following approximate equality holds

$$\Delta X_{k+1} \approx \sum_{i=1}^{k}\left(\prod_{j=i+1}^{k}(I+R_j)\right)(\Delta R_i X_i + M'_i + A'_i).$$

Taking norms we have (since $\Delta(X_{k+1}) = E_r(k+1)$ )

$$E_R(k+1) \leq \sum_{i=0}^{k} \| \prod_{j=i+1}^{k} (I + R_j)\| \, (\|\Delta R_i\|\|X_i\| + \|M_i'\| + \|A_i\|).$$

Since the residuals $R_j$ are symmetric matrices, we have $\|I+R_j\| = 1+\varrho(R_j) = 1 + \|R_j\| \leq 1 + q^{2^j} < 2$, and then

$$\| \prod_{j=i+1}^{k} (I + R_j)\| < 2^{k-i}.$$

Using the inequality $\|X_k\| \leq \|A^{-1}\|$, we thus get

$$E_R(k+1) \leq \mu \|A^{-1}\| 2^k (kn^2 + 2\sqrt{n}).$$

$\diamond$

On the basis of Theorem 3, to achieve condition (15) it is sufficient to choose $\mu$ in such a way that

$$\mu 2^k (kn^2 + 2\sqrt{n}) < 2^{-(d+1)},$$

or, equivalently,

$$\frac{1}{\mu} > 2^{d+1} 2^k (kn^2 + 2\sqrt{n}).$$

Hence, approximately $w = \lceil \log \frac{1}{\mu} \rceil$ bits suffice, with

$$w = O(d + k + \log n).$$

We will consider two significant cases:

1. There exists a constant $r$ such that $q \leq r < 1$, i.e., $\|R(B)\|$ can be bounded by a constant strictly less than 1, i.e., $\log \frac{1}{q} = O(1)$. In this case $k = O(\log d)$ and $w = O(d + \log n)$. Note that the bound on the arithmetic is determined by the output precision and the precision required by the inner product computations used in matrix multiplication (Lemma 1).

2. Bound (12) is the best can be proved of $q$. Then $\log \frac{1}{q} \geq \log \frac{1}{1 - \frac{1}{n\kappa^2(A)}} \approx \frac{1}{n\kappa^2(A)}$, and hence $-\log\log \frac{1}{q} = O(\log n \log \kappa(A))$. Here we have $k = O(\log d + \log n + \log \kappa(A))$ and $w = O(d + \log n + \log \kappa(A))$. With respect to the previous case, we observe a (potential) increase of the asymptotic number of steps. On the other hand, the arithmetic demand will increase only in case $A$ is ill-conditioned (say, for $\kappa(A) = \Omega(2^n)$).

# 5 Conclusions

We have considered a number of well-known matrix inversion algorithms that can be implemented as NC computations. Heretofore, the widespread belief that all such methods were numerically inaccurate if implemented in floating-point arithmetic was only based either on experimental evidence (with very few cases reported in the literature) or on rather vague arguments, such as the presumed instability of characteristic polynomial computations.

We have shown that a wide spectrum of such methods, including Csanky's algorithm, Recursive Factorization algorithms, and Gaussian Elimination, are practically infeasible when the exact inverse of an integer matrix of a large size is sought (as rational output), as they require word sizes linear or superlinear in the order of the matrix. Our analysis agrees with the observed numerical instability of some NC methods when implemented in finite-precision floating-point arithmetic. The outstanding conclusion, based on roundoff analysis, is that Newton's iteration is the only viable candidate for NC implementations. Note, however, that the analysis is not affected by the recourse to modular arithmetic with Chinese remaindering, due to the worst-case lower bound on the size of the output integer entries. Once more, we stress that the key feature of Newton's method is that it correctly estimates the ratio of (potentially) extremely large integers without explicitly computing them.

Not surprisingly, the results of this paper point to a tradeoff between accuracy and speed for parallel computation of matrix inverse, which had been previously observed experimentally by other authors. Also, the various results on the presumably inherent sequentiality (in the order $n$ of the matrix) of known stable matrix inversion techniques (such as Gaussian Elimination with Partial or Complete Pivoting and orthogonal matrix transformations [33, 17, 18]) support this intuition from a different angle.

# Acknowledgments

We gratefully acknowledge the insightful observations of two referees who have helped us focus our objectives and sharpen our analysis.

# References


[1] Bini, D., and Pan, V., Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms, Birkhäuser, Boston, 1994.

[2] Borodin, A., J. von zur Gathen, and J. Hopcroft, Fast Parallel Matrix and GCD Computations, *Inform. and Control* **52** (1982) 241–256.

[3] Clements, G. F. and Lindstroem, B A Sequence of ($\pm 1$)-determinants with Large Values, *Proc. Amer. Math. Soc.* **16** (1965) 548-550.

[4] Csanky, L., Fast Parallel Matrix Inversion Algorithms, *SIAM J. Comput.* **5** (1976) 618–623.

[5] Demmel, J. W., Trading off Parallelism and Numerical Accuracy, Tech. Rep. CS-92-179, Univ. of Tennessee, June 1992 (Lapack Working Note 52).

[6] Demmel, J. W., Higham, N. J., and Schreiber, R. S., Block *LU* Factorization, *Lapack working note 40*, UT, CS-92-149, February 1992.

[7] Du Croz, J. J. and Higham, N. J., Stability of Methods for Matrix Inversion, *IMA Journal of Numerical Analysis*, **12** (1992), 1–19.

[8] Eberly, W., Processor-Efficient Parallel Matrix Conversion over abstract fields: two extensions, *Proc. Symp. on Parallel Algebraic and Symbolic Computations (Pasco '97)*, ACM Press, New York (1997), 38–45.

[9] Faddeev, D. K. and Faddeva, V.N., *Computational Methods of Linear Algebra*, W. H. Freeman and Co., San Francisco, Ca, 1963.

[10] Galil, Z. and Pan, V., Parallel Evaluation of the Determinant and of the Inverse of a Matrix, *Inform. Proc. Letters* **30** (1989), 41–45.

[11] Von zur Gathen, J., Parallel Linear Algebra, in Reif, J. (ed.), *Synthesis of Parallel Algorithms*, Morgan and Kaufmann Pub., San Mateo, CA, 1996, 573-617.

[12] Golub, G. H. and Van Loan, C. F., *Matrix Computations*, Third Edition, Johns Hopkins University Press, Baltimore, Maryland, 1996.

[13] Higham, N. J., *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[14] Huang, X. and Pan, V., Fast Rectangular Matrix Multiplication and Applications, *J. of Complexity*, **14** (1998), 257–299.

[15] Kaltofen, E. and Pan, V., Processor Efficient Parallel Solution of Linear Systems over an Abstract Field, *Proc. 3rd ACM Symp. on Parallel Algorithms and Architectures*, ACM Press, New York (1991),180–191.

[16] Kaltofen, E. and Pan, V., Processor Efficient Parallel Solution of Linear Systems II. The Positive Characteristic and Singular Cases, *Proc. 33rd Annual IEEE Symp. on Foundations of Computer Science*, IEEE Computer Society Press (1992), 714–723.

[17] Leoncini, M., On the Parallel Complexity of Gaussian Elimination with Pivoting, *Journal of Computer and System Sciences* **53** (1996), 380–394.

[18] Leoncini, M., Manzini, G., and Margara L., Parallel complexity of numerically accurate linear system solvers. *SIAM J. Computing*, to appear.

[19] Moenck, R. T. and Carter, J. H., Approximate algorithms to derive exact solutions to systems of linear equations, in: *Proc. EUROSAM*, Lecture Notes in Computer Science **206** (1985), 504–521.

[20] Lovasz L., Connectivity algorithms using rubber-bands, in: *Proc. Sixth Conference on Foundations of Software Technology and Theoretical Computer Science* (1986), Lecture Notes in Computer Science **241**, 394–412.

[21] Pan, V., Fast and Efficient Parallel Algorithms for the Exact Inversion of Integer Matrices, in: *Proc. Fifth Conference on Foundations of Software Technology and Theoretical Computer Science* (1985), Lecture Notes in Computer Science **206**, 504–521.

[22] Pan, V., Complexity of Parallel Matrix Computations, *Theoret. Comput. Sci.* **54** (1987), 65–85.

[23] Pan, V., Effective Parallel Computations with Toeplitz and Toeplitz-like Matrices Filled with Integers, in: Renegar, J., Shub, M., and Smale, S., *Lectures in Applied Mathematics* **32** (1996), 593–641.

[24] Pan, V. and Reif, J. H., Fast and Efficient Parallel Solution of Linear Systems, *Proc. 17-th ACM Symp. on Theory of Computing*, Providence, R. I. (1985), 143–152.

[25] Pan, V. and Reif, J. H., Fast and Efficient Parallel Solution of Dense Linear Systems, *Computers and Mathematics with Applications* **17** (1989), 1481–1491.

[26] Pan, V. and Reif, J. H., The Parallel Computation of the Minimum Cost Paths in Graphs by Stream Contraction, *Inform. Proc. Letters* **40** (1991), 79–83.

[27] Pan, V. and Schreiber R., An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM J. on Scientific and Statistical Computing* **12** (1991), 1109–1131.

[28] Preparata, F. P. and Sarwate D. V., An Improved Parallel Processor Bound in Fast Matrix Inversion, *Inform. Proc. Letters* **7** (1978), 148-149.

[29] Reif, J. H., $O(\log^2 n)$ Time Efficient Parallel Factorization of Dense, Sparse Separable, and Banded Matrices, *Proc. 6th ACM Symp. on Parallel Algorithms and Architectures*, ACM Press, New York (1994), 278–289.

[30] Reif, J. H., Work Efficient Parallel Solution of Toeplitz Systems and Polynomial GCD, *Proc. 27th Annual ACM Symp. on Theory of Computing*, ACM Press, New York (1995), 751–761.

[31] Söderström, T. and Stewart, G. W., An Improved Newton Iteration for the Generalized Inverse of a Matrix, with Applications, *SIAM J. Numer. Anal.***11** (1974), 61–74.

[32] Valiant, L. G., Skyum S., Berkowitz, S., and Rackoff, C., Fast parallel computation of polynomials using few processors, *SIAM J. Comput.* **12** (1983), 641–644.

[33] Vavasis, S.A., Gaussian Elimination with Pivoting is P-complete, *SIAM J. Disc. Math.* **2** (1989) 413–423.

[34] Wilkinson, J. H., *Rounding Errors in Algebraic Processes*, Prentice-Hall, 1963.

[35] Wilkinson, J. H., *Modern Error Analysis*, *SIAM Review*, **13** (1971), 548–568.

# A Appendix

We give some basic definitions and recall a number of facts about floating-point number representation and matrices.

**floating-point number representation**
A floating-point system $\mathcal{F}$ is characterized by four integers: the *base* $b$ of the representation (usually $b = 2$), the *precision* $t$, and the *range* of the exponent $[\ell, L]$. A number $f \in \mathcal{F}$ has the form

$$f = \pm .b_1 b_2 \dots b_t \times b^e,$$

where $b_1 \neq 0$, which is a normalization condition (of course, for $b = 2$ the digit $b_1 = 1$ need not be stored). The sequence $.b_1 b_2 \dots b_t$ is the *mantissa* (also called *significand*) while $e$ is the *exponent*.

If $f$ is a floating-point number, then $m \leq |f| \leq M$, where $m = b^{1-\ell}$ and $M = b^L(1 - b^{-t})$ are the smallest and largest positive floating-point numbers representable in $\mathcal{F}$, respectively. Now, if $x$ is a real number satisfying $|x| \in [m, M]$, we will define $\mathrm{fl}(x)$ as either the closest $c \in \mathcal{F}$ to $x$, or the closest $c \in \mathcal{F}$ satisfying $|c| \leq |x|$. The first case corresponds to *rounded* arithmetic, the second to *chopped* arithmetic. Note that, in case of rounded arithmetic, possible ties are resolved by rounding away from zero. It can be proved that, for $x$ satisfying the above conditions, $\mathrm{fl}(x) = x(1 + \epsilon)$, or equivalently, for $x \neq 0$, $(\mathrm{fl}(x) - x)/x = \epsilon$, where $|\epsilon| \leq \mu$ and

$$\mu = \begin{cases} \frac{1}{2} b^{1-t} & \text{for rounded arithmetic} \\ \\ b^{1-t} & \text{for chopped arithmetic.} \end{cases}$$

Parameter $\mu$ is the so called *machine precision* or *roundoff unit* and is used for roundoff error analysis. The quantity $(\mathrm{fl}(x) - x)/x$ is the relative *representation error* affecting $\mathrm{fl}(x)$. Also, if $x, y \in \mathcal{F}$, then

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1 + \epsilon), \tag{19}$$

where $|\epsilon| \leq \mu$ and op is one of the four arithmetic operations. (19) is known as the *standard model* of arithmetic. Again, if $x$ op $y \neq 0$, the standard

model implies that

$$\left| \frac{\mathrm{fl}(x \text{ op } y) - (x \text{ op } y)}{x \text{ op } y} \right| < \mu,$$

in which the quantity $(\mathrm{fl}(x \text{ op } y) - (x \text{ op } y))/(x \text{ op } y)$ is the relative roundoff error of the arithmetic operation op.

We stress that the errors (not only representation and roundoff errors) are measured in relative terms, which is in agreement with the philosophy of floating-point numbers. If the error were measured in absolute terms, it would be impossible to define a uniform bound $\mu$ depending on the parameters of the number system only. On the other hand, the absolute error is the right measure for fixed-point number representations. Finally, we observe that, if $(\mathrm{fl}(x) - x)/x = \epsilon$, $\log \frac{1}{|\epsilon|}$ can be interpreted as the number of correct digits of the mantissa of $\mathrm{fl}(x)$. Alternatively, if we require that the relative error affecting $\mathrm{fl}(x)$ be bounded by $\epsilon > 0$, then we must reserve at least $\lceil \log \frac{1}{\epsilon} \rceil$ bits for the mantissa. For this reason the length of the mantissa is also called the *precision* of the arithmetic.

Input integer $x = b_{d-1} \ldots b_1 b_0$ is represented in floating-point as $x = .b_{d-1} \ldots b_1 b_0 \cdot 2^d$, i.e., a number with a $d$-bit significand and a $\lceil \log d \rceil$-bit exponent. This is because the standard floating-point arithmetic is *normalized*, i.e., the absolute value $v$ of the significand satisfies $\frac{1}{2} \leq v < 1$. The same holds of any intermediate or final result: to ensure that a $k$-bit integer is represented exactly as a floating-point number, the floating-point system must allow for at least a $k$ bit significand and a $\lceil \log k \rceil$ bit exponent. Hence, with respect to the word size $w$ of the integer arithmetic, the floating-point representation requires additional $\log w$ bits.

### Matrices and matrix notations

We consider $n \times n$ matrices over the reals. We denote the generic element of a matrix $A$ by $a_{ij}$ ( the $(i,j)$th entry of $A$) and write $A = (a_{ij})$. The symbols $I$ and $O$ denote the identity matrix and the matrix of all zeros, respectively. A matrix $A = (a_{ij})$ is upper (lower) triangular if $a_{ij} = 0$ when $i > j$ $(i < j)$. The *transpose* of $A$ is denoted $A^T$.

A matrix $A$ is *nonsingular* if there is a unique matrix $B$, ccalled its *inverse* and denoted $A^{-1}$, such that $AB = BA = I$. The *determinant* of a $1 \times 1$ matrix $A = (a)$ is simply $\det(A) = a$. The determinant of an $n \times n$ matrix $A$ can be defined recursively in terms of determinants of matrices of order $n - 1$ as

follows:

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}), \tag{20}$$

where $i \in \{1, \ldots, n\}$ is arbitrary and $A_{ij}$ is the matrix of order $n-1$ obtained by deleting the $i$th row and $j$th column of $A$. As is well known, $A$ is nonsingular if and only if $\det(A) \neq 0$. We recall Hadamard's inequality

$$|\det(A)|^2 \leq \prod_{j=1}^{n} \sum_{i=1}^{n} a_{ij}^2,$$

which leads to the following useful bound

$$|\det(A)| \leq (M\sqrt{n})^n, \tag{21}$$

where $M = \max_{ij} |a_{ij}|$. Hadamard matrices meet the above bound with equality.

The *adjoint* of $A$, written $\mathrm{adj}(A)$, is the matrix whose $(i,j)$th entry is $(-1)^{i+j} \det(A_{ji})$. It can be easily seen, using simple properties of the determinant, that $A\,\mathrm{adj}(A) = \mathrm{adj}(A)A = \det(A)I$. Hence, if $A$ is nonsingular we have the inversion formula $A^{-1} = \frac{1}{\det(A)}\mathrm{adj}(A)$.

The characteristic polynomial of $A$ is the degree-$n$ polynomial $p(x) = \det(A - xI) = c_n x^n + \ldots + c_1 x + c_0$. The Cayley-Hamilton theorem states:

$$p(A) = c_n A^n + \ldots + c_1 A + c_0 I = O,$$

which, if $c_0 = \det(A) \neq 0$, leads immediately to Leverrier's formula for the inverse (see [9]):

$$A^{-1} = -\frac{1}{c_0}(c_n A^{n-1} + \ldots + c_2 A + c_1 I). \tag{22}$$

The *eigenvalues* of $A$ are the roots of the characteristic equation $\det(A - xI) = 0$. It follows that a matrix of order $n$ has exactly $n$ eigenvalues in the complex field (counting multiplicities), denoted $\lambda_i$, $i = 1, \ldots, n$. The *spectral radius* $\varrho(A)$ of $A$ is the maximum modulus among all the eigenvalues, i.e.,

$$\varrho(A) = \max_{1 \leq i \leq n} |\lambda_i|.$$

A matrix norm is a real-valued function $\|\cdot\|$ satisfying

1. $\|A\| \geq 0$, and $\|A\| = 0$ if and only if $A = O$;

2. $\|cA\| = |c|\|A\|$, for any complex number $c$;

3. $\|A+B\| \leq \|A\|+\|B\|$, for any pair of matrices $A$ and $B$ of same order.

In this paper we will use the following well-known matrix norms:

**1-norm:** $\|A\|_1 = \max_j \sum_i |a_{ij}|$.

**$\infty$-norm:** $\|A\|_\infty = \max_i \sum_j |a_{ij}|$.

**2-norm:** $\|A\|_2 = \sqrt{\varrho(A^T A)}$.

When we drop the subscript we assume the 2-norm, i.e., $\|\cdot\| = \|\cdot\|_2$, also called *spectral norm*. Note that, when $A$ is symmetric, $\|A\| = \varrho(A)$; in particular, $\|A^T A\| = \|A\|^2$.

In Section 4 we use the following inequalities:

$$\|A\| \leq \sqrt{\|A\|_1 \|A\|_\infty}, \tag{23}$$

and

$$\|A\|_1, \|A\|_\infty \leq \frac{1}{\sqrt{n}}\|A\|. \tag{24}$$

Norms are crucial in perturbation analysis for linear systems and other matrix problems. Normwise bounds on the perturbation error usually include the *condition number* of the matrix. This is the quantity $\kappa_x(A) = \|A\|_x\|A^{-1}\|_x$, where $x$ identifies the particular norm adopted ($x \in \{1, \infty, 2\}$ in this paper). Again, when we drop the subscript we assume that the condition number of $A$ is computed according to the spectral norm. The norms considered here are *submultiplicative*, i.e., they enjoy the additional property $\|AB\|_x \leq \|A\|_x\|B\|_x$, it follows that $\kappa_x(A) = \|A\|_x\|A^{-1}\|_x \geq \|I\| = 1$, for $x \in \{1, \infty, 2\}$.

Finally, when $A$ is symmetric and nonsingular, we have $\lambda_{\min}(A) = \frac{1}{\lambda_{\max}(A^{-1})}$. It follows that $\kappa(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ and, in addition, that $\kappa(A^T A) = \kappa(A)^2$.