Acceleration of Euclidean Algorithm and Extensions *

Victor Y. Pan
Dept. of Mathematics and Computer Science
Lehman College of CUNY
Bronx, NY 10468, USA
vpan@lehman.cuny.edu

Xinmao Wang
Ph.D. Program in Mathematics
Graduate School of CUNY
New York, NY 10036, USA
xwang2@gc.cuny.edu

ABSTRACT

We accelerate the extended Euclidean algorithm for integers, the rational number reconstruction, and consequently, the stage of the recovery of the solution of a nonsingular integer system of linear equations via Hensel's lifting. The acceleration is by the order of magnitude and yields nearly optimal randomized algorithms. In the highly important case of Toeplitz, Hankel, and Toeplitz/Hankel-like linear systems, the accleration is potentially practical.

Keywords

extended Euclidean algorithm, linear system of equations, Toeplitz and Hankel matrices, Smith invariant factors, p-adic lifting, randomized algorithms, bit operation complexity, rational number reconstruction

2000 Mathematics Subject Classification:

15A36, 15A06, 68Q25, 68W40, 12Y05

1. INTRODUCTION

A customary approach in computer algebra is to perform computations with rational numbers modulo a large integer q (a prime, prime power, or product of several selected primes) and then to reconstruct the rational output from its value modulo q [9]. In particular, the *rational number reconstruction* is the final stage of the solution of a nonsingular linear system of n equations by means of p-adic lifting [12], [7], and we pay special attention to this application in the present paper (see [9] on other important applications).

Each component of the solution is reconstructed as a pair of integers η and δ from three integers q, k, and r such that

$$|\eta| < k < q, \quad 1 \le \delta \le q/k, \quad r = (\eta/\delta) \bmod q. \tag{1.1}$$

The hardest stage of computing η and δ is the application of the extended Euclidean algorithm to q and r. The algorithm recursively produces triples (r_j, s_j, t_j), $j = 1, \dots, l$; in our case we only need the triple where $|r_j| < k$ and j is the smallest. Theorem 5.26 in [9] supplies all details for the transition from this triple to the integers η and δ satisfying (1.1). The known algorithms compute the triple by using

$$\rho(q) = O(h^2), \text{ for } h = \log_2 q, \tag{1.2}$$

bit operations.

Now, given a nonsingular linear system of equations $A\mathbf{x} = \mathbf{b}$, we first compute $A^{-1}\mathbf{b} \bmod p$ for a fixed prime

$$p = n^{O(1)} \log |A| \tag{1.3}$$

where

$$|A| = \max_j \sum_i |a_{i,j}| \tag{1.4}$$

is the ∞-norm of a matrix $A = (a_{i,j})_{i,j}$ and the matrix $A \bmod p$ is still nonsingular. Then we apply Hensel's lifting [12], [7] to compute $A^{-1}\mathbf{b} \bmod q$ for

$$q = p^g, \quad g = \lfloor \log(2|A|^{2n-1}|\mathbf{b}|) \rfloor + 1, \tag{1.5}$$

so $q > 2|\eta|\delta$ for every compoment η/δ of the vector $A^{-1}\mathbf{b}$. Finally, we choose $k = |A|^{n-1}|\mathbf{b}| + 1$ and recover the n components of the solution at the bit cost $n\rho(q)$ for $\rho(q)$ in (1.2). We assume realistically that

$$\log |\mathbf{b}| = O(n \log |A|). \tag{1.6}$$

The bit cost of the recovery of $\mathbf{x} = A^{-1}\mathbf{b}$ from $\mathbf{x} \bmod q$ is $O(n^3 \log^2 |A|)$. This is close to the bit cost of computing $\mathbf{x} \bmod q$ for a general linear system $A\mathbf{x} = \mathbf{b}$. The latter cost, however, decreases by roughly the factor of n where A has structure of Toeplitz or Hankel type. So, in this highly important case [18], the bit cost of the recovery stage strongly dominates the overall cost of the solution of a linear system. The slow performance at the recovery stage devaluates the p-adic lifting approach to solving structured linear systems, making it inferior to other methods.

Our present paper fixes this mishap. We accelerate the rational reconstruction stage by roughly the factor of n by using two techniques. One of them exploits a simple trick already used in [16], [4], [5], [13], although in a different context. The other technique is quite advanced: it solves the classical problem of accelerating the extended Euclidean algorithm for integers. In the result of this progress, the overall cost of the solution of structured linear systems via Hensel's lifting is now dominated at the stage of computing

*Supported by NSF Grant CCR 9732206 and PSC CUNY Award 61393-0030, 62435-0031 and 66383-0032

the solution modulo q. Versus the other known algorithms for structured linear system [18], the overall acceleration is only by the factor of $\log n$ but is practically significant because the problem is highly important and because the bit operation cost of the solution is now placed within the factor of $(m(n)/n)\mu(\log p)/\log p$ from the information lower bound (see Corollary 3.4 and Theorem 4.1). Here and hereafter we assume that $m(n)$ field operations are sufficient to multiply two polynomials modulo x^n, and $\mu(h)$ bit operations are sufficient to multiply two integers modulo 2^h+1, so

$$n \le m(n) = O((n\log n)\log\log n), \tag{1.7}$$

$$h \le \mu(h) = O((h\log h)\log\log h) \tag{1.8}$$

(see [6] and [22], respectively). The same refinement of the rational number reconstruction stage applies to the general linear systems, but the relative improvement of the overall bit cost of system's solution is less significant.

Our first (practical) improvement of the recovery stage relies on multiplying the rational output (computed modulo q) by the Smith leading invariant factor s_n of the matrix A. The product is an integer vector and thus is immediately reconstructed from its value modulo q. The known best algorithm for computing s_n is the algorithm **Largest Invariant Factor** in [8]. It is based on the approach proposed in [14, Appendix], [15], [1]. The approach reduces the problem to application of Hensel's lifting to a few linear systems $A\mathbf{x}^{(i)} = \mathbf{b}^{(i)}$ for random vectors $\mathbf{b}^{(i)}$ and to subsequent randomized recovery of s_n as the least common denominator of all components of all $\mathbf{x}^{(i)}$. This may seem to bring us to a vicious circle, but we now observe that for probabilistic computation of the Smith factor it is sufficient to replace the n denominators of the n rational components of $\mathbf{x}^{(i)}$ by a single denominator of the scalar $\mathbf{c}^T\mathbf{x}^{(i)} = \mathbf{c}^T A^{-1}\mathbf{b}^{(i)}$ for a random vector $\mathbf{c}$. This yields the desired randomized practical speedup.

Our further acceleration is technically much more involved. We speed up the computation of a selected entry in the extended Euclidean algorithm by the factor of almost h, that is, we decrease the bit cost bound (1.2) to the level

$$\rho(q) = O(\mu(h)\log h), \quad h = \log_2 q, \tag{1.9}$$

although the overhead constant hidden in the above "O" notation may be too large to allow the algorithm become practical.

A similar acceleration is known for the Euclidean algorithm applied to polynomials [11], [2], [3], but in the integer case a well known additional difficulty is due to the carries. Among the known methods only the Knuth-Schönhage algorithm [21] has settled the problem for integers but only in the case where $j = l$ and the triple (r_l, s_l, t_l) terminates the Euclidean algorithm, that is, where r_l is the gcd. In our work we were motivated by the following excerpt from [9, page 305] on the extended Euclidean algorithm for integers:

> *The method also works for integers, although there are some complications due to the carries,*

and by the recent comments of an expert Joachim von zur Gathen on the state of the art which he sent by email to one of the present authors:

> *Yes, I suppose rational number reconstruction can be done in time $O(m(n)\log n)$ for n-bit numbers and a given upper bound on the denominator. This is alluded to in [9], as you observed. But we do not give a proof, and I do not know any rigorous proof in the literature. I can imagine roughly what needs to be done, but it will be quite messy.*

In section 5 we clear the cited mess and come out with a desired algorithm (see Theorem 5.11). Otherwise we organize our paper as follows. After some preliminaries in section 2, we show randomized acceleration of computing the Smith leading factor s_n in section 3, and we extend this algorithm to solving linear systems in section 4.

2. PRELIMINARIES

Hereafter, we write log to replace $\log_2$ unless specified otherwise, and we use definitions (1.1)–(1.9).

Numbers, polynomials, vectors, matrices

Hereafter, $\mathbb{Q}$ is the field of rational numbers, $\mathbb{Z}$ is the ring of integers, $\mathbb{Z}_q$ is the ring of integers modulo q. We write $\delta(r) = \delta$, $\eta(r) = \eta$ for two coprime integers $\delta > 0$ and η and for the rational number $r = \eta/\delta$. For three integers h, p and m, we write $h = \mathrm{ord}_p\, m$ if p^h divides m but p^{h+1} does not divide m. $|u(x)| = \sum_i |u_i|$ for a polynomial $u(x) = \sum_i u_i x^i$. $\mathbf{v} = (v_i)_{i=1}^n$ is a vector with components $v_1, \dots, v_n$; $A = (a_{i,j})_{i,j=1}^n$ is an $n \times n$ matrix with the (i,j)-th entry $a_{i,j}$, $i, j = 1, \dots, n$; $\mathbf{v}^T$ and A^T are the transposes of $\mathbf{v}$ and A; I is the identity matrix. $\det(A)$ is the determinant of a matrix A.

Complexity estimates

Hereafter, we assume that v_W ring or field operations suffice to multiply a given $n \times n$ matrix W by a vector, $n \le v_W \le 2n^2 - n$, whereas i_W field operations suffice to invert W or to determine its singularity.

Smith invariant factors and ratios

DEFINITION 2.1. The greatest common divisor (gcd) $d_k = d_k(A)$ of all $k \times k$ minors (subdeterminants) of a matrix $A \in \mathbb{Z}^{n\times n}$ is called the k-th *determinantal divisor* of A, for $k = 1, \dots, n$. We write $s_0 = d_0 = 1$ and define the k-th *Smith invariant factor* of A as $s_k = s_k(A) = d_k/d_{k-1}$, and the k-th Smith ratio of A as $r_k = s_k/s_{k-1}$ for $k = 1, \dots, n$.

It is easily deduced that $s_1, \dots, s_n \in \mathbb{Z}$ and $|\det A| = s_1 \cdots s_n$, so (cf. (1.4))

$$s_n \le |\det(A)| \le |A|^n. \tag{2.1}$$

THEOREM 2.2. *For a nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ and its leading Smith factor s_n, we have $s_n A^{-1} \in \mathbb{Z}^{n\times n}$.*

Due to this theorem, reconstruction of A^{-1} from $A^{-1} \bmod q$ as well as $A^{-1}\mathbf{b}$ from $A^{-1}\mathbf{b} \bmod q$ (for larger q) is trivial if $s_n = s_n(A)$ is available.

3. COMPUTATION OF THE LEADING SMITH FACTOR

We follow [8] (cf. [14], [15], [1]) and reduce computing $s_n(A)$ to solving linear systems $A\mathbf{x}^{(k)} = \mathbf{b}^{(k)}$, for random vectors $\mathbf{b}^{(k)}$, but unlike [8], we recover $s_n(A)$ from the denominators of random linear combinations of the rationals $x_1^{(k)}, \dots, x_n^{(k)}$ rather than from the lcm's of the integer denominators, $\mathrm{lcm}(\delta(x_1^{(k)}), \dots, \delta(x_n^{(k)}))$. This saves us the factor of roughly n in the estimated bit cost of the recovery. The acceleration relies on a simple trick of obtaining the lcm as the denominator of a random linear combination of reciprocals already used in [16], [4], [5], and [13] although in a different context.

THEOREM 3.1. *For a nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ and a positive $\epsilon < 1$, independent of n and A, it is sufficient to generate a sufficiently large random prime $p = n^{O(1)} \log |A|$ and $K = O(\log(1/\epsilon))$ random vectors $\mathbf{b}^{(k)}, \mathbf{c}^{(k)} \in \mathbb{Z}_M^n$ for $k = 1, 2, \dots, K$, $K = O(\log(1/\epsilon))$, and $M = \max\{\lceil \sqrt{n \log |A|} \rceil, 4000\}$, that is, to generate a total of $O((n \log(n|A|)) \log(1/\epsilon))$ random bits, and in addition to perform $i_A \mu(\log p) + O(((v_A + v_{A^{-1}})(\mu(\log p)/\log p) n \log |A| + \rho(|A|^n)) \log(1/\epsilon))$ bit operations (for $\mu(h)$ in (1.8), $\rho(q) = O(\mu(\log q) \log\log q)$ in (1.9)) in order to compute a positive s_n^* dividing $s_n = s_n(A)$ and such that*

$$\text{Probability}(s_n^* = s_n) \geq 1 - \epsilon. \tag{3.1}$$

PROOF. To support Theorem 3.1 for $\epsilon = 1/2$, let us modify the algorithm **Largest Invariant Factor** in [8, Section 2] by changing its parameters M and $t_n^{(k)}$. As in [8], the extension to any fixed ϵ, $0 < \epsilon < 1$, is by increasing the parameter K by the factor of $\log(1/\epsilon)$.

ALGORITHM 3.2 (LEADING SMITH FACTOR).

INPUT: *A nonsingular matrix $A \in \mathbb{Z}^{n\times n}$.*

OUTPUT: *A positive integer $s_n^* \mid s_n$.*

INITIALIZATION: *M, p, $\mathbf{b}^{(k)}$ and $\mathbf{c}^{(k)}$ are as in Theorem 3.1 for $K = 2$, and $h = \lceil 2 \log_p(|A|^{2n-1} M) \rceil$ such that $q = p^h \geq |A|^{2n-1} M$.*

COMPUTATION: *For $k = 1, 2$, compute*

1. $\mathbf{x}^{(k)} = (x_i^{(k)})_{i=1}^n = A^{-1}\mathbf{b}^{(k)} \in \mathbb{Z}_q^n$,
2. $y^{(k)} = \mathbf{c}^{(k)T}\mathbf{x}^{(k)} = \sum_{i=1}^n c_i^{(k)} x_i^{(k)} \in \mathbb{Z}_q$,
3. $t_n^{(k)} = \delta(y^{(k)})$ *such that* $0 \leq t_n^{(k)} \leq |A|^n$,
4. *output* $s_n^* = \mathrm{lcm}(t_n^{(1)}, t_n^{(2)})$.

Clearly, $s_n^* \mid s_n$. Let us prove that $s_n^* = s_n$ with a probability of at least $1/2$. It is sufficient to repeat the proof of Theorem 2 in [8] complemented by the next lemma, which for every k validates using the denominator $t_n^{(k)}$ of linear combinations of $x_1^{(k)}, \dots, x_n^{(k)}$ instead of the lcm of all denominators $\delta(x_1^{(k)}), \dots, \delta(x_n^{(k)})$.

LEMMA 3.3. *Write $\delta^{(k)} = \mathrm{lcm}(\delta(x_1^{(k)}), \dots, \delta(x_n^{(k)}))$, so $t_n^{(k)} \mid \delta^{(k)}$. Then for any prime $\bar{p}$, we have*
a) Probability $\left(\mathrm{ord}_{\bar{p}}(s_n) \neq \mathrm{ord}_{\bar{p}}(\delta^{(k)})\right) \leq \max\{1/M, 1/\bar{p}\}$;
b) Probability $\left(\mathrm{ord}_{\bar{p}}(t_n^{(k)}) \neq \mathrm{ord}_{\bar{p}}(\delta^{(k)})\right) \leq \max\{1/M, 1/\bar{p}\}$.

PROOF OF LEMMA 3.3. Part a) follows from Theorem 2 in [1]. To prove part b), write $y^{(k)} = \eta^{(k)}/\delta^{(k)}$, so $\eta^{(k)} = \sum_{i=1}^n c_i^{(k)} \eta(x_i^{(k)}) \delta_i^{(k)}$, for $\delta_i^{(k)} = \delta^{(k)}/\delta(x_i^{(k)})$. By the definition of $\delta^{(k)}$, we have $\gcd(\bar{p}, \eta(x_i^{(k)})\delta_i^{(k)}) = 1$ for at least one i, $1 \leq i \leq n$ (excluding the trivial case where $\mathbf{x}^{(k)} = \mathbf{0} \bmod \bar{p}$ for all k). Let this hold, say for $i = 1$. Write $\sum_{i=2}^n c_i^{(k)} \eta(x_i^{(k)}) \delta_i^{(k)} = q\bar{p} + r$, $q, r \in \mathbb{Z}$, $0 \leq r < \bar{p}$. Then $\bar{p} \mid \eta^{(k)}$ only where $\bar{p} \mid (r + c_1^{(1)} \eta(x_1^{(k)}) \delta_i^{(k)})$, that is, for at most $M \max\{1/M, 1/\bar{p}\}$ choices of $c_1^{(1)}$ in $\mathbb{Z}_M$. This proves part b). □

To prove Theorem 3.1, it remains to estimate from above the number of bit operations used in Algorithm 3.2. We have the following upper bounds: $i_A \mu(\log p)$ for computing A^{-1} at stage 1 (once for all k); $O(v_A + v_{A^{-1}})\mu(\log p)h$, where $h = O(n \log_p A)$, in Hensel's p-adic lifting applied for each fixed k to compute $A^{-1}\mathbf{b}^{(k)} \bmod p^h$ [12], [7], and $O(\mu(h) \log h) = O(\mu(n \log |A|) \log(n \log |A|))$ for each k at stage 3 (see (1.9)). The cost of lifting dominates the cost at stages 2 and 4 (recall our assumption that $v_A \geq n$). Summarizing, we complete the proof of Theorem 3.1. □

Let us next specify the bit operation bounds of Theorem 3.1 for general and Toeplitz matrices A. For a general $n \times n$ non-singular matrix A, we have $i_A = O(n^3)$, $v_A \leq 2n^2 - n$, $v_{A^{-1}} \leq 2n^2 - n$. If A is a Toeplitz matrix, we have $i_A = O(m(n) \log n)$, $v_A = O(m(n))$, $v_{A^{-1}} = O(m(n))$ (cf. (1.7)), provided the inverse A^{-1} is represented by its first and last columns or by a pair of its other products by a fixed pair of vectors [18]. The same cost bounds hold for Hankel matrices A and for a larger and highly important class of nonsingular structured matrices A having structure of Toeplitz/Hankel type and represented (as well as A^{-1}) with their displacement generators of length $O(1)$ [18]. Substituting these bounds, we specialize Theorem 3.1 as follows (see more details in [20]).

COROLLARY 3.4. *Let $A \in \mathbb{Z}^{n\times n}$, $\det A \neq 0$, $|A| \geq n$, and $\epsilon > 0$. Then a divisor s_n^* of the leading Smith factor s_n such that $s_n^* = s_n$ with a probability of at least $1 - \epsilon$ can be computed by generating $O((n \log |A|) \log(1/\epsilon))$ random bits and performing $O(((n^3 \mu(\log p)/\log p) \log |A| + \rho(|A|^n)) \log(1/\epsilon))$ bit operations for $p = O(n \log |A|)$, $\mu(h)$ in (1.8), and $\rho(q)$ in (1.9). If the matrix A has Toeplitz/Hankel-like structure, then the bit operation cost bound decreases to $O(((nm(n)\mu(\log p)/\log p) \log |A| + \rho(|A|^n)) \log(1/\epsilon))$ for $m(n)$ in (1.7). If $\log |A| = O(\log n)$, then the above*

bounds turn into $O(n^3\mu(\log n)\log(1/\epsilon))$ *for general matrices* A *and into* $O((nm(n)\mu(\log n)/\log n)\log(1/\epsilon))$ *for Toeplitz/Hankel-like matrices* A. *By ignoring the factors* $\log(1/\epsilon)$ *(assuming* $1/\epsilon = O(1)$*) and* $(\log\log n)^{O(1)}$, *we obtain the bounds* $O(n^3 \log n)$ *and* $O(n^2 \log n)$, *respectively.*

4. SOLVING A LINEAR SYSTEM OF EQUATIONS

Given a nonsingular matrix $A \in \mathbb{Z}^{n\times n}$ and a vector $\mathbf{b} \in \mathbb{Z}^n$, we first apply Theorem 3.1 and Algorithm 3.2 to compute probabilistically $s_n = s_n(A)$, then compute the integer vector $\mathbf{y} = s_n A^{-1}\mathbf{b}$ (see Theorem 2.2), and finally compute $\mathbf{x} = \mathbf{y}/s_n$.

The entries of $\mathbf{y}$ are integers not exceeding $s_n|A|^{n-1}|\mathbf{b}|$, so we may immediately recover each of them from $\mathbf{y} \bmod p^h$ for $h = \lfloor \log_p(2s_n|A|^{n-1}|\mathbf{b}|)\rfloor + 1$ at the bit operation cost $O(p^h)$ (see (2.1)). The asymptotic bit operation cost of computing $\mathbf{y} \bmod p^h$ and recovering $\mathbf{y}$ is dominated by the respective estimates in Theorem 3.1 and Corollary 3.4.

The well-known techniques enable extension to computing a basis for the null space of A and solving a linear system $A\mathbf{x} = \mathbf{b}$ where A is singular [10], [18, Chapter 5]. The next theorem summarizes the cost estimates (see [20] for further details).

THEOREM 4.1. *Let* $\log|\mathbf{b}| = O(n\log|A|)$. *Then all bit cost estimates of Theorem 3.1 and Corollary 3.4 apply to the problem of the solution of a nonsingular linear system* $A\mathbf{x} = \mathbf{b}$. *If* A *is a singular Toeplitz/Hankel-like matrix, the same estimates apply to the solution of a consistent system* $A\mathbf{x} = \mathbf{b}$ *and to computing a short displacement generator for a matrix whose columns form a basis for the null space of* A. *For a general matrix* A, *computing a basis for its null space requires additional* $O((n-\beta)\beta^3\mu(\log p)/\log p)$ *bit operations for* $p = O(\log(|A|^\beta|\mathbf{b}|))$ *and* $\beta = \mathrm{rank}(A)$. *The bounds cover the bit cost of verification of correctness of the solution where it is represented by a single vector or by* $O(1)$ *vectors. If the solution is represented by* k *vectors,* $k \le n-\beta$, *then additional* $O(knm(n)\mu(\log n)/\log n)$ *bit operations are required to verify their correctness.*

REMARK 4.2. The known bit operation cost bounds for computing $s_n = s_n(A)$ and solving linear systems $A\mathbf{x} = \mathbf{b}$ are improved in Theorems 3.1 and 4.1 and Corollary 3.4 by roughly the factor of $\log|A|$, for general matrices A. For matrices A with Toeplitz/Hankel structure such that $n\log|A| = O(2^{n^{0.5-\delta}})$ and for a fixed positive constant δ, the improvement is by the factor of $\log n$, and the new bounds are within the factor of $(m(n)/n)\mu(\log n)/\log n$, from the information lower bound of the order of $n^2\log|A|$.

5. ACCELERATED EXTENDED EUCLIDEAN ALGORITHM

DEFINITION 5.1. $\lfloor x\rfloor$ and $\lceil x\rceil$ are two integers closest to a real number x such that $\lfloor x\rfloor \le x \le \lceil x\rceil$. $\{x\} = x - \lfloor x\rfloor$. $\|A\| = \max_{i,j}|a_{i,j}|$ for any real matrix $A = (a_{i,j})_{i,j}$. $m \bmod n$ is defined to be $m - n\lfloor m/n\rfloor$ for $m, n \in \mathbb{Z}$ and $n > 0$. (Here we slightly abuse the notation: in this section n is not related to the matrix dimension n in the preceding sections.)

ALGORITHM 5.2 (EUCLIDEAN ALGORITHM).

INPUT: *A pair of natural numbers* $m \ge n$.

OUTPUT: $\gcd(m, n)$.

COMPUTATION: *Write* $x_0 = m$, $x_1 = n$. *Compute*

$$x_{i+1} = x_{i-1} \bmod x_i$$

for $i = 1, 2, \ldots, k$, *until* $x_{k+1} = 0$. *Output* x_k.

DEFINITION 5.3.

$$Q_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad Q_{k+1} = \begin{pmatrix} \infty & \infty \\ \infty & \infty \end{pmatrix},$$

$$Q_i = Q_{i-1}\begin{pmatrix} \lfloor x_{i-1}/x_i\rfloor & 1 \\ 1 & 0 \end{pmatrix} \text{ for } 1 \le i \le k.$$

REMARK 5.4.

(i) $x_i > x_{i+1} > 0$, $x_i \ge x_{i+1} + x_{i+2}$ for $i = 0, 1, \ldots, k-1$.

(ii) $\det(Q_i) = (-1)^i$ and $\begin{pmatrix} m \\ n \end{pmatrix} = Q_i \begin{pmatrix} x_i \\ x_{i+1} \end{pmatrix}$ for $i = 0, 1, \ldots, k$.

(iii) Write $Q_i = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}$, then $b_i = a_{i-1}$, $d_i = c_{i-1}$ for $i = 1, 2, \ldots, k$. Since $a_i = a_{i-1}\lfloor x_{i-1}/x_i\rfloor + a_{i-2}$, $c_i = c_{i-1}\lfloor x_{i-1}/x_i\rfloor + c_{i-2}$, we have $a_i > a_{i-1}$, $c_i > c_{i-1}$ for $i = 2, 3, \ldots, k$. Furthermore, since $a_0 > c_0$, $a_1 \ge c_1$, we have $a_i > c_i$ for $i = 2, 3, \ldots, k$. Therefore,

(iv) $\|Q_i\| = a_i$ for $i = 0, 1, \ldots, k$ and $\|Q_i\| \ge \|Q_{i-1}\| + \|Q_{i-2}\|$ for $i = 2, 3, \ldots, k$.

(v) Q_{i-1} can be computed from Q_i by $a_{i-2} = a_i \bmod a_{i-1}$, $c_{i-2} = c_i \bmod c_{i-1}$ for $i = 3, 4, \ldots, k$.

(vi) Combining (i) and (iv), we have $m/2 < x_i\|Q_i\| \le m$ for $i = 0, 1, \ldots, k$.

For the input of another pair of natural numbers $m^* \ge n^*$, the Euclidean Algorithm computes the sequences $\{x_i^*\}_{i=1}^{k^*}$ and $\{Q_i^*\}_{i=1}^{k^*}$. When $m/n = m^*/n^*$, it is obviously that the two sequences $\{Q_i\}_{i=1}^{k}$ and $\{Q_i^*\}_{i=1}^{k^*}$ are identical. When m/n and m^*/n^* are very close to each other, we believe that the first several terms of $\{Q_i\}_{i=1}^{k}$ and $\{Q_i^*\}_{i=1}^{k^*}$ are the same. The next theorems show us some sufficient conditions under which $Q_i = Q_i^*$ for a given i. Therefore, we may accelerate our computation of Q_i by using some smaller integers $m^* \le m$ and $n^* \le n$.

THEOREM 5.5. *Given two pairs of natural numbers* $m \ge n$ *and* $m^* \ge n^*$, *then*

$$\min(x_i x_{i+1}^*, x_{i+1}x_i^*) \ge |mn^* - nm^*| \implies Q_i = Q_i^*.$$

PROOF. By Remark 5.4 (i), $\min(x_j x^*_{j+1}, x_{j+1} x^*_j) \geq |mn^* - nm^*|$ for $j = 0, 1, \ldots, i$. We prove Theorem 5.5 by induction. $Q_0 = Q^*_0$ is trivial. Suppose $Q_{j-1} = Q^*_{j-1}$, then

$$\begin{pmatrix} m & m^* \\ n & n^* \end{pmatrix} = Q_{j-1} \begin{pmatrix} x_{j-1} & x^*_{j-1} \\ x_j & x^*_j \end{pmatrix},$$

$$\left| \frac{x_{j-1}}{x_j} - \frac{x^*_{j-1}}{x^*_j} \right| = \frac{|mn^* - nm^*|}{x_j x^*_j} \leq \frac{\min(x_j x^*_{j+1}, x_{j+1} x^*_j)}{x_j x^*_j}$$
$$= \min\left(\frac{x_{j+1}}{x_j}, \frac{x^*_{j+1}}{x^*_j} \right) = \min\left(\{\frac{x_{j-1}}{x_j}\}, \{\frac{x^*_{j-1}}{x^*_j}\} \right).$$

So $\lfloor x_{j-1}/x_j \rfloor = \lfloor x^*_{j-1}/x^*_j \rfloor$, $Q_j = Q^*_j$ for $j = 1, 2, \ldots, i$. □

COROLLARY 5.6. *Given a pair of natural numbers $m \geq n$, let $m^* = \lfloor m/\lambda \rfloor$, $n^* = \lfloor n/\lambda \rfloor$ for a natural number λ. Then*

$$m^* \geq 4 \, \|Q_{i+1}\| \cdot \|Q^*_{i+1}\| \quad \Longrightarrow \quad Q_i = Q^*_i.$$

PROOF. $|mn^* - nm^*| = |(m - m^*\lambda)n^* - (n - n^*\lambda)m^*| < m^*\lambda \leq m$, $x_i > x_{i+1} > \frac{m}{2\|Q_{i+1}\|}$, $x^*_i > x^*_{i+1} > \frac{m^*}{2\|Q^*_{i+1}\|}$. By Theorem 5.5, $Q_i = Q^*_i$. □

THEOREM 5.7. *Given a pair of natural numbers $m \geq n$, let $m^* = \lfloor m/\lambda \rfloor$, $n^* = \lfloor n/\lambda \rfloor$ for a natural number λ. Then*

$$x^*_{i+2} \geq \|Q^*_{i+1}\| \quad \Longrightarrow \quad Q_i = Q^*_i.$$

PROOF. Write $\begin{pmatrix} u_j \\ v_j \end{pmatrix} = {Q^*_j}^{-1} \begin{pmatrix} m \\ n \end{pmatrix}$ for $j = 0, 1, \ldots, i+1$.

(i) $\begin{pmatrix} u_{j+1} \\ v_{j+1} \end{pmatrix} = \begin{pmatrix} 1 & \\ 1 & -\lfloor x^*_j / x^*_{j+1} \rfloor \end{pmatrix} \begin{pmatrix} u_j \\ v_j \end{pmatrix} \Longrightarrow u_{j+1} = v_j$ for $j = 0, 1, \ldots, i$.

(ii) $\begin{pmatrix} u_j \\ v_j \end{pmatrix} = \begin{pmatrix} x^*_j \\ x^*_{j+1} \end{pmatrix} \lambda + {Q^*_j}^{-1} \begin{pmatrix} m - m^*\lambda \\ n - n^*\lambda \end{pmatrix}$.

Note that the first row of ${Q^*_j}^{-1}$ is two integers (one is ≥ 0, another is ≤ 0) bounded by $\|Q^*_{j-1}\|$ and the second row of ${Q^*_j}^{-1}$ is two integers (one is ≤ 0, another is ≥ 0) bounded by $\|Q^*_j\|$. Therefore $v_j > (x^*_{j+1} - \|Q^*_j\|)\lambda \geq 0$ and $u_j - v_j > (x^*_j - \|Q^*_{j-1}\|)\lambda - (x^*_{j+1} + \|Q^*_j\|)\lambda \geq (x^*_{j+2} - \|Q^*_{j+1}\|)\lambda \geq 0$ for $j = 1, 2, \ldots, i$. Now we have $u_0 = m$, $u_1 = n$, $u_{j+1} = u_{j-1} \bmod u_j$ for $j = 1, 2, \ldots, i$. So $u_j = x_j$, $Q_j = Q^*_j$ for $j = 0, 1, \ldots, i$. □

THEOREM 5.8. *Given a pair of natural numbers $m \geq n$, let $m^* = \lfloor m/\lambda \rfloor$, $n^* = \lfloor n/\lambda \rfloor$ for a natural number λ. Suppose $m^* \geq 2k^2$ and $\|Q^*_i\| \leq k < \|Q^*_{i+1}\|$ for a natural number k, then $\|Q_j\| \leq k < \|Q_{j+1}\|$ for some j such that $i - 2 \leq j \leq i + 3$.*

PROOF. Since $x^*_i > \frac{m^*}{2\|Q^*_i\|} \geq k \geq \|Q^*_i\|$, by Theorem 5.7, $Q_{i-2} = Q^*_{i-2}$. If $\|Q_i\| > k/2$ then $\|Q_{i+2}\| \geq 2\|Q_i\| > k$, $\|Q_j\| \leq k < \|Q_{j+1}\|$ for some j such that $i - 2 \leq j \leq i + 1$.

From now on we assume $\|Q_i\| \leq k/2$, therefore, $4 \, \|Q_i\| \cdot \|Q^*_i\| \leq 2k^2 \leq m^*$. By Theorem 5.5 and Corollary 5.6, we have $Q_{i-1} = Q^*_{i-1}$ and

$$\left| \frac{x_{i-1}}{x_i} - \frac{x^*_{i-1}}{x^*_i} \right| = \frac{|mn^* - nm^*|}{x_i x^*_i} < \frac{m}{x_i x^*_i} < \frac{2\|Q_i\|}{x^*_i} \leq 1.$$

CASE I. $\lfloor x_{i-1}/x_i \rfloor = \lfloor x^*_{i-1}/x^*_i \rfloor$, then $Q_i = Q^*_i$. From

$$\begin{pmatrix} m & m^* \\ n & n^* \end{pmatrix} = Q^*_i \begin{pmatrix} x_i & x^*_i \\ x_{i+1} & x^*_{i+1} \end{pmatrix},$$

and

$$\begin{pmatrix} m & m^* \\ x_{i+1} & x^*_{i+1} \end{pmatrix} = \begin{pmatrix} \|Q^*_i\| & \|Q^*_{i-1}\| \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i & x^*_i \\ x_{i+1} & x^*_{i+1} \end{pmatrix},$$

we have

$$|mx^*_{i+1} - m^* x_{i+1}| = \|Q^*_i\| \cdot |mn^* - nm^*| < km.$$

Hence

$$x_{i+1} < \frac{m}{m^*}(2^h + x^*_{i+1}) < \frac{m}{2k} + \frac{m}{k} = \frac{3m}{2k},$$

$$\|Q_{i+4}\| > 3\|Q_{i+1}\| > \frac{3m}{2x_{i+1}} > k.$$

CASE II. $\lfloor x_{i-1}/x_i \rfloor = \lfloor x^*_{i-1}/x^*_i \rfloor - 1$, then $Q_i = Q^*_i \left(\begin{smallmatrix} 1 & \\ -1 & 1 \end{smallmatrix} \right)$. From

$$\begin{pmatrix} m & m^* \\ n & n^* \end{pmatrix} = Q^*_i \begin{pmatrix} x_i & x^*_i \\ x_{i+1} - x_i & x^*_{i+1} \end{pmatrix},$$

and

$$\begin{pmatrix} m & m^* \\ x_{i+1} - x_i & x^*_{i+1} \end{pmatrix} = \begin{pmatrix} \|Q^*_i\| & \|Q^*_{i-1}\| \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i & x^*_i \\ x_{i+1} - x_i & x^*_{i+1} \end{pmatrix},$$

we have

$$mx^*_{i+1} + m^*(x_i - x_{i+1}) = \|Q^*_i\| \cdot |mn^* - nm^*| < km.$$

Hence

$$x_{i+2} \leq x_i - x_{i+1} < \frac{km}{m^*} \leq \frac{m}{2k}, \quad \|Q_{i+2}\| > \frac{m}{2x_{i+2}} > k.$$

CASE III. $\lfloor x_{i-1}/x_i \rfloor = \lfloor x^*_{i-1}/x^*_i \rfloor + 1$, then $Q^*_i = Q_i \left(\begin{smallmatrix} 1 & \\ -1 & 1 \end{smallmatrix} \right)$. From

$$\begin{pmatrix} m & m^* \\ n & n^* \end{pmatrix} = Q_i \begin{pmatrix} x_i & x^*_i \\ x_{i+1} & x^*_{i+1} - x^*_i \end{pmatrix},$$

and

$$\begin{pmatrix} m & m^* \\ x_{i+1} & x^*_{i+1} - x^*_i \end{pmatrix} = \begin{pmatrix} \|Q_i\| & \|Q_{i-1}\| \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i & x^*_i \\ x_{i+1} & x^*_{i+1} - x^*_i \end{pmatrix},$$

we have

$$m(x^*_i - x^*_{i+1}) + m^* x_{i+1} = \|Q_i\| \cdot |mn^* - nm^*| < \|Q_i\| m.$$

Hence

$$2k^2 \leq m^* < \frac{\|Q_i\| m}{x_{i+1}} < 2\|Q_{i+1}\|^2, \quad \|Q_{i+1}\| > k.$$

In a word, we find $\|Q_j\| \leq k < \|Q_{j+1}\|$ for $i - 2 \leq j \leq i + 3$ in all the above cases. □

ALGORITHM 5.9 (SELECTED OUTPUT OF THE EEA).

INPUT: *A pair of natural numbers $m \geq n$, and an integer $h \geq 0$.*

OUTPUT: *The unique Q_k such that $\|Q_k\| \leq 2^h < \|Q_{k+1}\|$.*

COMPUTATION: *Let $d = \lfloor \log m \rfloor$.*

1. *When $h \leq \lfloor d/2 \rfloor - 1$, let $\lambda = 2^{d-2h-1}$, $m^* = \lfloor m/\lambda \rfloor$, $n^* = \lfloor n/\lambda \rfloor$, then $2^{2h+1} \leq m^* \leq m/2$. We first apply the algorithm for the input (m^*, n^*, h) and have the output Q_i^*. Theorem 5.8 tells us that $Q_{i-2} = Q_{i-2}^*$ and $\|Q_k\| \leq 2^h < \|Q_{k+1}\|$ for some $i-2 \leq k \leq i+3$. We may compute Q_{i-2} from Q_i^* and find Q_k in a few Euclidean steps.*

2. *When $\lfloor d/2 \rfloor \leq h \leq d-1$, we first apply the algorithm to find $\|Q_i\| \leq 2^{\lfloor h/2 \rfloor} < \|Q_{i+1}\|$. Next we apply the algorithm again for the input $(x_i, x_{i+1}, \lfloor h/2 \rfloor)$ and have the output $\tilde{Q}_j$. Now we have $Q_{i+j} = Q_i \tilde{Q}_j$, $\|Q_{i+j}\| < 2^{h+1}$, $\|Q_{i+j+2}\| > 2^{h-1}$. Then because $\|Q_k\| \leq 2^h < \|Q_{k+1}\|$ for some $i+j-2 \leq k \leq i+j+3$, we may find Q_k in a few Euclidean steps.*

3. *When $h \geq d$, we first apply the algorithm to find $\|Q_i\| \leq 2^{d-1} < \|Q_{i+1}\|$. Then because $\|Q_k\| \leq 2^h < \|Q_{k+1}\|$ for some $i \leq k \leq i+4$, we may find Q_k in a few Euclidean steps.*

BIT-COMPLEXITY OF THE SOLUTION: *Let $f(d,h)$ be the bit-cost of the algorithm for the input (m,n,h) where $d = \lfloor \log m \rfloor$, we have*

$$f(d,h) = \begin{cases} f(2h+1,h) + O(\mu(d)), & \text{if } h \leq \lfloor \frac{d}{2} \rfloor - 1; \\ f(d, \lfloor \frac{h}{2} \rfloor) + f(d - \lfloor \frac{h}{2} \rfloor, \lfloor \frac{h}{2} \rfloor) + O(\mu(d)), & \\ & \text{if } \lfloor \frac{d}{2} \rfloor \leq h \leq d-1; \\ f(d, d-1) + O(\mu(d)), & \text{if } h \geq d. \end{cases}$$

Especially, let $F(h) = f(2h+1,h)$ then

$$F(h) = 2F(\lfloor h/2 \rfloor) + O(\mu(2h)),$$

by which we have

$$F(h) = O(\mu(2h) \log h).$$

Therefore

$$\begin{aligned} f(d,h) &= \sum_{i=1}^{1+\lfloor \log h \rfloor} \Big(F(\lfloor h/2^i \rfloor) + O(\mu(d)) \Big) \\ &= O(\mu(d) \log h) \end{aligned} \tag{5.1}$$

REMARK 5.10. Given three natural numbers m, n, k, $m \geq n$, we may apply Algorithm 5.9 to find the matrix $Q_i = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ such that $\|Q_i\| \leq m/k < \|Q_{i+1}\|$. Write $y = (-1)^i x_{i+1}$. Since $\left(\begin{smallmatrix} x_i \\ x_{i+1} \end{smallmatrix}\right) = (-1)^i \left(\begin{smallmatrix} d & -b \\ -c & a \end{smallmatrix}\right) \left(\begin{smallmatrix} m \\ n \end{smallmatrix}\right)$, we have

$$|y| < \frac{m}{\|Q_{i+1}\|} \leq k, \quad 1 \leq a = \|Q_i\| \leq m/k, \quad n = y/a \bmod m.$$

Replacing m by q, n by r, y by η, and a by δ, turns the above equation into equation (1.1). Now we arrive at

THEOREM 5.11. *Algorithm 5.9 solves the problem (1.1) of rational number reconstruction by using $\rho(q)$ bit operations for $\rho(q)$ in (1.9).*

6. ACKNOWLEDGEMENTS

We are grateful to Joachim von zur Gathen for his expert advice on the state of the art of the bit complexity of rational number reconstruction and to him and the referees for helpful comments on the original draft of our paper.

7. REFERENCES

[1] J. Abbott, M. Bronstein, T. Mulders. Fast Deterministic Computations of the Determinants of Dense Matrices. *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'99)*, 197–204, ACM Press, New York, 1999.

[2] A. V. Aho, J. E. Hopcroft, J. D. Ullman. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Massachusetts, 1974.

[3] R. P. Brent, F. G. Gustavson, D. Y. Y. Yun. Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximations. *Journal of Algorithms*, 1:259–295, 1980.

[4] D. Bini, V. Y. Pan. *Polynomial and Matrix Computations, Vol.1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[5] G. Cooperman, S. Feisel, J. von zur Gathen, G. Havas. GCD of Many Integers. *Computing and Combinatorics, Lecture Notes in Computer Science*, 1627:310-317, Springer, Berlin, 1999.

[6] D. G. Cantor, E. Kaltofen. On Fast Multiplication of Polynomials over Arbitrary Rings. *Acta Informatica*, 28(7):697–701, 1991.

[7] J. D. Dixon. Exact Solution of Linear Equations Using p-adic Expansions. *Numerische Math.*, 40:137–141, 1982.

[8] W. Eberly, M. Giesbrecht, G. Villard. On Computing the Determinant and Smith Form of an Integer Matrix. *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS'2000)*, 675–685, IEEE Computer Society Press, Los Alamitos, California, 2000.

[9] J. von zur Gathen, J. Gerhard. *Modern Computer Algebra*, Cambridge University Press, Cambridge, UK, 1999.

[10] E. Kaltofen, V. Y. Pan. Processor Efficient Parallel Solution of Linear Systems over an Abstract Field. *Proceedings of 3rd Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'91)*, 180–191, ACM Press, New York, 1991.

[11] R. Moenck. Fast Computation of GCDs. *Proceedings of 5th ACM Annual Symposium on Theory of Computing*, 142–171, ACM Press, New York, 1973.

[12] R. T. Moenck, J. H. Carter. Approximate Algorithms to Derive Exact Solutions to Systems of Linear Equations. *Proceedings of EUROSAM, Lecture Notes in Computer Science*, 72:63–73, Springer, Berlin, 1979.

[13] T. Mulders, A. Storjohann. Certified Dense Linear System Solving, preprint, 2001.

[14] V. Y. Pan. Complexity of Parallel Matrix Computations. *Theoretical Computer Science*, 54:65–85, 1987.

[15] V. Y. Pan. Computing the Determinant and the Characteristic Polynomials of a Matrix via Solving Linear System of Equations. *Information Processing Letters*, 28:71–75, 1988.

[16] V. Y. Pan. Parametrization of Newton's Iteration for Computations with Structured Matrices and Applications. *Computers & Mathematics (with Applications)*, 24(3):61–75, 1992.

[17] V. Y. Pan. Parallel Complexity of Computations with General and Toeplitz-like Matrices Filled with Integers and Extensions. *SIAM Journal on Computing*, 30:1080–1125, 2000.

[18] V. Y. Pan. *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[19] V. Y. Pan. Randomized Acceleration of Fundamental matrix Computations. *Proc. of Symposium on Theoretical Aspects of Computer Science (STACS'2002), Lecture Notes in Computer Science*, Springer, Berlin, 2002.

[20] V. Y. Pan. Can We Optimize Computations with Structured Matrices? preprint, 2002.

[21] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971.

[22] A. Schönhage, V. Strassen. Schnelle Multiplikation grosse Zahlen. *Computing*, 7:281–292, 1971.