MATHEMATICS OF COMPUTATION
Volume 70, Number 235, Pages 1169–1182
S 0025-5718(00)01235-7
Article electronically published on March 24, 2000

INVERSION OF CIRCULANT MATRICES OVER $\mathbf{Z}_m$

DARIO BINI, GIANNA M. DEL CORSO, GIOVANNI MANZINI, AND LUCIANO MARGARA

ABSTRACT. In this paper we consider the problem of inverting an $n \times n$ circulant matrix with entries over $\mathbf{Z}_m$. We show that the algorithm for inverting circulants, based on the reduction to diagonal form by means of FFT, has some drawbacks when working over $\mathbf{Z}_m$. We present three different algorithms which do not use this approach. Our algorithms require different degrees of knowledge of m and n, and their costs range, roughly, from $n \log n \log\log n$ to $n \log^2 n \log\log n \log m$ operations over $\mathbf{Z}_m$. Moreover, for each algorithm we give the cost in terms of bit operations. We also present an algorithm for the inversion of finitely generated bi-infinite Toeplitz matrices. The problems considered in this paper have applications to the theory of linear cellular automata.

1. INTRODUCTION

In this paper we consider the problem of inverting circulant and finitely generated bi-infinite Toeplitz matrices with entries over the ring $\mathbf{Z}_m$. In addition to their own interest in linear algebra, these problems play an important role in the theory of linear Cellular Automata.

The standard algorithm for inverting circulant matrices with real or complex entries is based on the fact that any $n \times n$ circulant is diagonalizable by means of the Fourier matrix F (defined by $F_{ij} = \omega^{(i-1)(j-1)}$, where ω is a primitive nth root of unity). Hence, we can compute the eigenvalues of the matrix with a single FFT. Then, to compute the inverse of the matrix, it suffices to invert the eigenvalues and perform FFT twice. The total cost of inverting an $n \times n$ circulant is therefore $O(n \log n)$ arithmetic operations.

Unfortunately this method does not generalize, not even to circulant matrices over the field $\mathbf{Z}_p$. The reason is that if $\gcd(p, n) > 1$ no extension field of $\mathbf{Z}_p$ contains a primitive nth root of unity, and one can easily verify that $n \times n$ circulant matrices are not diagonalizable. If $\gcd(p, n) = 1$ we are guaranteed that a primitive nth root of unity exists in a suitable extension field of $\mathbf{Z}_p$. However, the approach based on the FFT still poses some problems. In fact, working in an extension of $\mathbf{Z}_p$

Received by the editor December 17, 1998 and, in revised form, June 28, 1999.
2000 *Mathematics Subject Classification.* Primary 15A33, 65F05.
Key words and phrases. Circulant matrices, bi-infinite Toeplitz matrices, inversion over rings, Laurent series.
The first author was supported by Progetto Coordinato GNIM-CNR Toeplitz-like matrices: structures algorithms and applications.
A preliminary version of this paper has been presented to the 25th International Colloquium on Automata, Languages and Programming (ICALP '98).

requires that we find a suitable irreducible polynomial $q(x)$ and every operation in the field involves manipulation of polynomials of degree up to $\deg(q(x)) - 1$.

In this paper we describe three algorithms for inverting an $n \times n$ circulant matrix over $\mathbf{Z}_m$ which are not based on the reduction to diagonal form by means of FFT. Instead, we transform the original problem into an equivalent problem over the ring $\mathbf{Z}_m[x]$. In our analysis we are mainly concerned with asymptotic bounds on the bit cost of the algorithms. These bounds can be derived from the bounds on the arithmetic cost just by multiplying the latter by the bit cost of a single arithmetic operation.

Our first algorithm assumes the factorization of m is known and requires $n \log^2 n + n \log m$ multiplications and $n \log^2 n \log\log n$ additions over $\mathbf{Z}_m$. This corresponds to the bit complexity bound

$$O\big((n \log^2 n + n \log m)\mu(\log m) + n \log^2 n \log\log n \log m\big),$$

where $\mu(d)$ denotes the bit complexity of multiplying d-bit integers. Our second algorithm does not require the factorization of m and its cost is greater, by a factor $\log m$, than in the previous case. The third algorithm needs no assumption about m but works only for $n = 2^k$; it is the fastest algorithm and has the same asymptotic cost of a single multiplication between degree n polynomials in $\mathbf{Z}_m[x]$, that is

$$O((n \log n + \log\log m)\mu(\log m) + n \log n \log\log n \log m).$$

Finally, we show that the latter algorithm can be used to build a fast procedure for the inversion of finitely generated bi-infinite Toeplitz matrices. The relevance of these algorithms for the theory of linear cellular automata is described in the Appendix.

The problem of inverting a circulant matrix with entries over an arbitrary commutative ring R has been addressed in [5]. There, the author shows how to compute the determinant and the adjoint of an $n \times n$ circulant matrix of the form $I + \sum_{i=1}^{l} \beta_i U^i$, (where $U_{ij} = 1$ for $i - j \equiv 1 \pmod n$, and 0 otherwise). A naive implementation of the proposed method takes $O\big(nl + 2^l\big)$ operations over R. Although the same computation can be done in $O(nl + M(l) \log n)$ operations, where $M(l)$ is the cost of $l \times l$ matrix multiplication (hence, $M(l) = l^\theta$, with $2 \le \theta < 2.376$), this algorithm remains competitive only for very small values of the "band" l. At the end of Section 5 we present a faster algorithm for the inversion of circulant matrices with small "bandwidth" when $n = 2^k$.

2. Definitions, notation and auxiliary results

Circulant matrices. Let U denote the $n \times n$ cyclic shift matrix whose entries are $U_{ij} = 1$ if $j - i \equiv 1 \pmod n$, and 0 otherwise. An $n \times n$ circulant matrix over $\mathbf{Z}_m$ can be written as $A = \sum_{i=0}^{n-1} a_i U^i$, where $a_i \in \mathbf{Z}_m$. Assuming A is invertible over $\mathbf{Z}_m$, we consider the problem of computing a circulant matrix $B = \sum_{i=0}^{n-1} b_i U^i$, such that $AB = I$ (it is well known that the inverse of a circulant matrix is still circulant).

It is natural to associate with a circulant matrix $A = \sum_{i=0}^{n-1} a_i U^i$ the polynomial (over the ring $\mathbf{Z}_m[x]$) $f(x) = \sum_{i=0}^{n-1} a_i x^i$. Computing the inverse of A is clearly equivalent to finding a polynomial $g(x) = \sum_{i=0}^{r} b_i x^i$ in $\mathbf{Z}_m[x]$ such that

$$f(x)g(x) \equiv 1 \pmod{x^n - 1}. \tag{1}$$

The congruence modulo $x^n - 1$ follows from the equality $U^n = I$. Hence, the problem of inverting a circulant matrix is equivalent to inversion in the ring $\mathbf{Z}_m[x]/(x^n - 1)$.

Bi-infinite Toeplitz matrices. Let W, W^{-1}, W^0 denote the bi-infinite matrices defined by

$$W_{ij} = \begin{cases} 1, & \text{if } j - i = 1, \\ 0, & \text{otherwise;} \end{cases} \quad W^{-1}_{ij} = \begin{cases} 1, & \text{if } i - j = 1, \\ 0, & \text{otherwise;} \end{cases} \quad W^0_{ij} = \begin{cases} 1, & \text{if } j = i, \\ 0, & \text{otherwise;} \end{cases}$$

where both indices i, j range over $\mathbf{Z}$. If we extend in the obvious way the matrix product to the bi-infinite case, we have $WW^{-1} = W^{-1}W = W^0$. Hence, we can define the algebra of finitely generated bi-infinite Toeplitz matrices over $\mathbf{Z}_m$ as the set of all matrices of the form

$$T = \sum\nolimits_{i \in \mathbf{Z}} a_i W^i,$$

where $a_i \in \mathbf{Z}_m$ and only finitely many of them are nonzero. An equivalent representation of the elements of this algebra can be obtained using finite *Laurent series* (also called dipolynomials [13]) over $\mathbf{Z}_m$. For example, the matrix T above is represented by the finite Laurent series $h_T(x) = \sum_{i \in \mathbf{Z}} a_i x^i$. In the following we use $\mathbf{Z}_m\{x\}$ to denote the set of finite Laurent series over $\mathbf{Z}_m$. Instead of stating explicitly that only finitely many coefficients are nonzero, we write each element $f(x) \in \mathbf{Z}_m\{x\}$ as $f(x) = \sum_{i=-r}^{r} b_i x^i$ (where some of the b_i's can still be zero). Computing the inverse of a bi-infinite Toeplitz matrix T is clearly equivalent to finding $g(x) \in \mathbf{Z}_m\{x\}$ such that

$$h_T(x) g(x) \equiv 1 \pmod{m}. \tag{2}$$

Hence, inversion of finitely generated Toeplitz matrices is equivalent to inversion in the ring $\mathbf{Z}_m\{x\}$.

Conditions for invertibility over $\mathbf{Z}_m\{x\}$ and $\mathbf{Z}_m[x]/(x^n-1)$. A necessary and sufficient condition for the invertibility of an element in $\mathbf{Z}_m\{x\}$ has been given in [8] where the authors prove that a finite Laurent series $f(x) = \sum_{i=-r}^{r} a_i x^i$ is invertible if and only if for each prime factor p of m there exists a unique coefficient a_i such that $p \nmid a_i$. The following theorem (proved in [12]) provides an equivalent condition which does not require the knowledge of the factorization of the modulus m.

Theorem 2.1. *Let $f(x) = \sum_{i=-r}^{r} a_i x^i$ be a finite Laurent series over $\mathbf{Z}_m$, and let $k = \lfloor \log_2 m \rfloor$. For $i = -r, \ldots, r$, define*

$$z_i = [\gcd(a_i, m)]^k \qquad \text{and} \qquad q_i = \frac{m}{\gcd(m, z_i)}.$$

Then, $f(x)$ is invertible if and only if $q_{-r} \cdots q_r = m$.

The following theorem states a necessary and sufficient condition for the invertibility of a circulant matrix over $\mathbf{Z}_m$.

Theorem 2.2. *Let $m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$, denote the prime powers factorization of m and let $f(x)$ denote the polynomial over $\mathbf{Z}_m$ associated to a circulant matrix A. The matrix A is invertible if and only if, for $i = 1, \ldots, h$, we have*

$$\gcd(f(x), x^n - 1) = 1 \qquad \text{in } \mathbf{Z}_{p_i}[x].$$

Proof. If A is invertible, by (1) we have that there exists $t(x)$ such that for $i = 1, \ldots, h$

$$f(x)g(x) + t(x)(x^n - 1) = 1 \qquad \text{in } \mathbf{Z}_{p_i}[x].$$

Hence, $\gcd(f(x), x^n - 1) = 1$ in $\mathbf{Z}_{p_i}[x]$ as claimed. The proof that the above condition is sufficient for invertibility is constructive and will be given in Section 3 (Lemmas 3.1 and 3.2). □

Note that for m prime the above result coincides with the condition given in [6, Theorem 2.4] for the invertibility of circulant matrices over finite fields.

Review of bit complexity results. In the following we will give the cost of each algorithm in terms of number of *bit operations*. In our analysis we use the following well-known results (see for example [1] or [2]). Additions and subtractions in $\mathbf{Z}_m$ take $O(\log m)$ bit operations. We denote by $\mu(d) = O(d \log d \log\log d)$ the number of bit operations required by the Schönhage-Strassen algorithm [15] for multiplication of integers modulo $2^d + 1$. Hence, multiplication between elements of $\mathbf{Z}_m$ takes $\mu(\log m) = O(\log m \log\log m \log\log\log m)$ bit operations. Computing the inverse of an element $x \in \mathbf{Z}_m$ takes $\mu(\log m) \log\log m$ bit operations using a modified extended Euclidean algorithm (see [1, Theorem 8.20]). The same algorithm returns $\gcd(x, m)$ when x is not invertible.

The sum of two polynomials in $\mathbf{Z}_m[x]$ of degree at most n can be trivially computed in $O(n \log m)$ bit operations. The product of two such polynomials can be computed in $O(n \log n)$ multiplications and $O(n \log n \log\log n)$ additions/subtractions in $\mathbf{Z}_m$ (see [2, Theorem 1.7.1]). Therefore, the asymptotic cost of polynomial multiplication is $O(\Pi(m, n))$ bit operations,[1] where

$$\Pi(m, n) = n \log n \mu(\log m) + n \log n \log\log n \log m. \tag{3}$$

Given two polynomials $a(x), b(x) \in \mathbf{Z}_p[x]$ (p prime) of degree at most n, we can compute $d(x) = \gcd(a(x), b(x))$ in $O(\Gamma(p, n))$ bit operations, where

$$\Gamma(p, n) = \Pi(p, n) \log n + n\mu(\log p) \log\log p. \tag{4}$$

The same algorithm also returns $s(x)$ and $t(x)$ such that $a(x)s(x) + b(x)t(x) = d(x)$. The bound (4) follows by a straightforward modification of the polynomial gcd algorithm described in [1, Section 8.9] (the term $n\mu(\log p) \log\log p$ comes from the fact that we must compute the inverse of $O(n)$ elements of $\mathbf{Z}_p$).

3. Inversion in $\mathbf{Z}_m[x]/(x^n - 1)$. Factorization of m known

In this section we consider the problem of computing the inverse of a circulant matrix over $\mathbf{Z}_m$ when the factorization $m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$ of the modulus m is known. We consider the equivalent problem of inverting a polynomial $f(x)$ over $\mathbf{Z}_m[x]/(x^n - 1)$, and we show that we can compute the inverse by combining known techniques (Chinese remaindering, the extended Euclidean algorithm, and Newton-Hensel lifting). We start by showing that it suffices to find the inverse of $f(x)$ modulo the prime powers $p_i^{k_i}$.

[1] This bound can be reduced when m is prime and we are allowed to do some preprocessing, see [9].

Lemma 3.1. *Let $m = p_1^{k_1} p_2^{k_2} \cdots p_h^{k_h}$, and let $f(x)$ be a polynomial in $\mathbf{Z}_m[x]$. Given $g_1(x), \ldots, g_h(x)$ such that*

$$f(x)g_i(x) \equiv 1 \pmod{x^n - 1} \quad \text{in } \mathbf{Z}_{p_i^{k_i}}[x] \quad i = 1, 2, \ldots h,$$

we can find $g(x) \in \mathbf{Z}_m[x]$ which satisfies (1) at the cost of

$$O(nh\mu(\log m) + \mu(\log m) \log\log m)$$

bit operations.

Proof. The proof is constructive. Since $f(x)g_i(x) \equiv 1 \pmod{x^n - 1}$ in $\mathbf{Z}_{p_i^{k_i}}[x]$, we have

$$f(x)g_i(x) \equiv 1 + \lambda_i(x)(x^n - 1) \pmod{p_i^{k_i}}.$$

Let $\alpha_i = m/p_i^{k_i}$. Clearly, for $j \neq i$, $\alpha_i \equiv 0 \pmod{p_j^{k_j}}$. Since $\gcd(\alpha_i, p_i^{k_i}) = 1$, we can find β_i such that $\alpha_i\beta_i \equiv 1 \pmod{p_i^{k_i}}$. Let

$$g(x) = \sum_{i=1}^{h} \alpha_i\beta_i g_i(x), \qquad \lambda(x) = \sum_{i=1}^{h} \alpha_i\beta_i\lambda_i(x).$$

By construction, for $i = 1, 2, \ldots, h$, we have $g(x) \equiv g_i(x) \pmod{p_i^{k_i}}$ and $\lambda(x) \equiv \lambda_i(x) \pmod{p_i^{k_i}}$. Hence, for $i = 1, 2, \ldots, h$, we have

$$\begin{aligned} f(x)g(x) &= \sum_{j=1}^{h} \alpha_j\beta_j f(x)g_j(x) \\ &\equiv f(x)g_i(x) \pmod{p_i^{k_i}} \\ &\equiv 1 + \lambda_i(x)(x^n - 1) \pmod{p_i^{k_i}} \\ &\equiv 1 + \lambda(x)(x^n - 1) \pmod{p_i^{k_i}}. \end{aligned}$$

We conclude that

$$f(x)g(x) \equiv 1 + \lambda(x)(x^n - 1) \pmod{m},$$

or, equivalently,

$$f(x)g(x) \equiv 1 \pmod{x^n - 1} \quad \text{in } \mathbf{Z}_m[x].$$

The computation of $g(x)$ consists in n (one for each coefficient) applications of Chinese remaindering. Obviously, the computation of α_i, β_i, $i = 1, \ldots, h$, should be done only once. Since integer division has the same asymptotic cost as multiplication, we can compute $\alpha_1, \ldots, \alpha_h$ in $O(h\mu(\log m))$ bit operations. Since each β_i is obtained through an inversion in $\mathbf{Z}_{p_i{}^{k_i}}$, computing the $\beta_1, \ldots, \beta_h$ takes

$$O\left(\sum_{j=1}^{h} \mu(\log p_j^{k_j}) \log\log p_j^{k_j}\right)$$

bit operations . Finally, given $\alpha_1, \ldots, \alpha_h$, $\beta_1, \ldots, \beta_h$, $g_1(x), \ldots, g_h(x)$ we can compute $g(x)$ in $O(nh\mu(\log m))$ bit operations. The thesis follows using the inequality

$$\mu(\log a) \log\log a + \mu(\log b) \log\log b \leq \mu(\log(ab)) \log\log(ab).$$

□

```
Inverse1(f(x), m, n) → g(x)
{ Computes the inverse g(x) of the polynomial f(x) in Z_m[x]/(x^n − 1)}
1.   let m = p_1^{k_1} p_2^{k_2} ... p_h^{k_h};
2.   for j = 1, 2, ··· , h do
3.      if gcd(f(x), x^n − 1) = 1 in Z_{p_j}[x] then
4.         computeg_j(x) such that f(x)g_j(x) ≡ 1 (mod x^n − 1)in Z_{p_j^{k_j}}[x]
5.         using Newton-Hensel lifting (Lemma 3.2);
6.       else
7.          return "f(x) is not invertible";
8.       endif
9.    endfor
10.  compute g(x) using Chinese remaindering (Lemma 3.1).
```

ALGORITHM 1. Inversion in $\mathbf{Z}_m[x]/(x^n-1)$. Factorization of m known.

In view of Lemma 3.1 we can restrict ourselves to the problem of inverting a polynomial over $\mathbf{Z}_m[x]/(x^n-1)$ when $m=p^k$ is a prime power. Next lemma shows how to solve this particular problem.

Lemma 3.2. *Let $f(x)$ be a polynomial in $\mathbf{Z}_{p^k}[x]$. If $\gcd(f(x),x^n-1)=1$ in $\mathbf{Z}_p[x]$, then $f(x)$ is invertible in $\mathbf{Z}_{p^k}[x]/(x^n-1)$. In this case, the inverse of $f(x)$ can be computed in $O\big(\Gamma(p,n)+\Pi(p^k,n)\big)$ bit operations, where $\Gamma(p,n)$ and $\Pi(p^k,n)$ are defined by* (4) *and* (3).

Proof. If $\gcd(f(x),x^n-1)=1$ in $\mathbf{Z}_p[x]$, by Bezout's lemma there exist $s(x),t(x)$ such that

$$f(x)s(x)+(x^n-1)t(x)\equiv 1 \pmod p.$$

Next we consider the sequence

$$g_0(x)=s(x), \qquad g_i(x)=2g_{i-1}(x)-[g_{i-1}(x)]^2 f(x) \mod (x^n-1),$$

known as Newton-Hensel lifting. It is straightforward to verify by induction that $g_i(x)f(x)\equiv 1+p^{2^i}\lambda_i(x) \pmod{x^n-1}$. Hence, the inverse of $f(x)$ in $\mathbf{Z}_{p^k}[x]/(x^n-1)$ is $g_{\lceil \log k\rceil}(x)$.

The computation of $s(x)$ takes $O(\Gamma(p,n))$ bit operations. For computing $g_1(x)$, $g_2(x),\dots,g_{\lceil\log k\rceil}(x)$ we observe that it suffices to compute each g_i modulo p^{2^i}. Hence, the cost of obtaining the whole sequence is

$$O\Big(\Pi(p^2,n)+\Pi(p^4,n)+\cdots+\Pi(p^{2^{\lceil\log k\rceil}},n)\Big)=O\big(\Pi(p^k,n)\big)$$

bit operations. □

Note that from Lemmas 3.1 and 3.2, we find that the condition given in Theorem 2.2 is indeed a sufficient condition for invertibility of a circulant matrix. Combining the above lemmas we obtain Algorithm 1 for the inversion of a polynomial $f(x)$ over $\mathbf{Z}_m[x]/(x^n-1)$. The cost of the algorithm is

$$T(m,n)=O\left(nh\mu(\log m)+\mu(\log m)\log\log m+\sum_{j=1}^{h}\Gamma(p_j,n)+\Pi(p_j^{k_j},n)\right)$$

bit operations. In order to get a more manageable expression, we bound h with $\log m$ and p_j with $p_j^{k_j}$. In addition, we use the inequalities $\Pi(a,n)+\Pi(b,n) \leq \Pi(ab,n)$ and $\Gamma(a,n)+\Gamma(b,n) \leq \Gamma(ab,n)$. We obtain

$$\begin{aligned} T(m,n) &= O(n \log m \mu(\log m) + \mu(\log m) \log\log m + \Gamma(m,n) + \Pi(m,n)) \\ &= O(n \log m \mu(\log m) + \Pi(m,n) \log n)\,. \end{aligned}$$

Note that if $m = O(n)$ the dominant term is $\Pi(m,n)\log n$. That is, the cost of inverting $f(x)$ is asymptotically bounded by the cost of executing $\log n$ multiplications in $\mathbf{Z}_m[x]$.

4. A general inversion algorithm in $\mathbf{Z}_m[x]/(x^n-1)$

The algorithm described in Section 3 relies on the fact that the factorization of the modulus m is known. If this is not the case and the factorization must be computed beforehand, the increase in the running time may be significant since the fastest known factorization algorithms require time exponential in $\log m$ (see for example [10]). In this section we show how to compute the inverse of $f(x)$ without knowing the factorization of the modulus. The number of bit operations of the new algorithm is only a factor $O(\log m)$ greater than in the previous case.

Our idea consists in trying to compute $\gcd(f(x), x^n-1)$ in $\mathbf{Z}_m[x]$ using the gcd algorithm for $\mathbf{Z}_p[x]$. Such algorithm requires the inversion of some scalars, which is not a problem in $\mathbf{Z}_p[x]$, but it is not always possible if m is not prime. Therefore, the computation of $\gcd(f(x), x^n-1)$ may fail. However, if the gcd algorithm terminates we have solved the problem. In fact, together with the alleged[2] gcd $a(x)$ the algorithm also returns $s(x), t(x)$ such that $f(x)s(x)+(x^n-1)t(x) = a(x)$ in $\mathbf{Z}_m[x]$. If $a(x)=1$, then $s(x)$ is the inverse of $f(x)$. If $\deg(a(x)) \neq 0$, one can easily prove that $f(x)$ is not invertible in $\mathbf{Z}_m[x]/(x^n-1)$. Note that we must force the gcd algorithm to return a monic polynomial.

If the computation of $\gcd(f(x), x^n-1)$ fails, we use recursion. In fact, the gcd algorithm fails if it cannot invert an element $y \in \mathbf{Z}_m$. Inversion is done by using the *integer* gcd algorithm. If y is not invertible, the integer gcd algorithm returns $d = \gcd(m,y)$, with $d > 1$. Hence, d is a nontrivial factor of m. We use d to compute either a pair m_1, m_2 such that $\gcd(m_1,m_2) = 1$ and $m_1 m_2 = m$, or a single factor m_1 such that $m_1 | m$ and $m | (m_1)^2$. In the first case we invert $f(x)$ in $\mathbf{Z}_{m_1}[x]/(x^n-1)$ and $\mathbf{Z}_{m_2}[x](x^n-1)$, and we use Chinese remaindering to get the desired result. In the second case, we invert $f(x)$ in $\mathbf{Z}_{m_1}[x]/(x^n-1)$ and we use one step of Newton-Hensel lifting to get the inverse in $\mathbf{Z}_m[x]/(x^n-1)$. The computation of the factors m_1, m_2 is done by procedure `GetFactors` (see Algorithm 2 below) whose correctness is proven by Lemmas 4.1 and 4.2. Combining these procedures together we obtain Algorithm 2. Note that the idea of working "as if" m were a prime is a known technique in algorithmic number theory (see for example [4, Section 2.3.4]).

Lemma 4.1. *Let α, $\alpha > 1$, be a divisor of m and let $\alpha' = \gcd(m, \alpha^{\lfloor \log m \rfloor})$. Then, α' is a divisor of m and $\gcd(\alpha', m/\alpha') = 1$.*

Proof. Let $m = p_1^{k_1} \cdots p_h^{k_h}$ denote the prime factorization of m. Clearly, $\alpha^{\lfloor \log m \rfloor}$ contains every prime p_i which is in α, with an exponent at least k_i

[2]The correctness of the gcd algorithm has been proven only for polynomials over fields, so we do not claim any property for the output of the algorithm when working in $\mathbf{Z}_m[x]$.

```
Inverse2(f(x), m) → g(x)
{Computes the inverse g(x) of the polynomial f(x) in Z_m[x]/(x^n − 1)}
1.    if gcd(f(x), x^n − 1) = 1  then
2.      let s(x), t(x) such that f(x)s(x) + (x^n − 1)t(x) = 1  in  Z_m[x];
3.       return s(x);
4.    else if gcd(f(x), x^n − 1) = a(x), deg(a(x)) > 0  then
5.       return "f(x) is not invertible";
6.    else if gcd(f(x), x^n − 1)  fails let d be such that d|m;
7.      let (m1, m2) ← GetFactors(m, d);
8.       if m2 ≠ 1,  then
9.          g1(x) ← Inverse2(f(x), m1);
10.         g2(x) ← Inverse2(f(x), m2);
11.          compute g(x) using Chinese remaindering (Lemma 3.1);
12.      else
13.         g1(x) ← Inverse2(f(x), m1);
14.          compute g(x) using Newton-Hensel lifting (Lemma 3.2);
15.      endif
16.      return g(x);
17.   endif

GetFactors(m, d) → (m1, m2)
18.   let m1 ← gcd(m, d^⌊log m⌋);
19.   if (m/m1) ≠ 1  then
20.      return (m1, m/m1);
21.   endif
22.   let e ← m/d;
23.   let m1 ← gcd(m, e^⌊log m⌋);
24.   if (m/m1) ≠ 1  then
25.      return (m1, m/m1);
26.   endif
27.   let m1 ← lcm(d, e);
28.   return (m1, 1);
```

ALGORITHM 2. Inversion in $\mathbf{Z}_m[x]/(x^n - 1)$. Factorization of m unknown.

(since $k_i \leq \lfloor \log m \rfloor$). Hence, α' contains each prime p_i which is in α with exponent exactly k_i. In addition, m/α' contains each prime p_j which is not in α with exponent exactly k_j; hence $\gcd(\alpha', m/\alpha') = 1$ as claimed. □

Lemma 4.2. *Let α, β be such that $\alpha\beta = m$ and $\gcd(m, \alpha^{\lfloor \log m \rfloor}) = \gcd(m, \beta^{\lfloor \log m \rfloor}) = m$. Then $\gamma = \operatorname{lcm}(\alpha, \beta) = m/\gcd(\alpha, \beta)$ is such that $\gamma | m$ and $m | \gamma^2$.*

Proof. Let $m = p_1^{k_1} \cdots p_h^{k_h}$ denote the prime factorization of m. By Lemma 4.1 we know that both α and β contain every prime p_i, $i = 1, \ldots, h$. Since $\alpha\beta = m$, each prime p_i appears in γ with exponent at least $\lfloor k_i/2 \rfloor$. Hence m divides γ^2 as claimed. □

Theorem 4.3. *If $f(x)$ is invertible in $\mathbf{Z}_m[x]/(x^n - 1)$, Algorithm 2 returns the inverse $g(x)$ in $O(\Gamma(m, n) \log m)$ bit operations.*

Proof. One can easily prove the correctness of the algorithm by induction on m, the base on the induction being the case in which m is prime where the inverse is computed by the gcd algorithm.

To prove the bound on the number of bit operations we first consider the cost of the single steps. By (4) we know that computing $\gcd(f(x), x^n - 1)$ takes

$$O(\Gamma(m,n)) = O(\Pi(m,n)\log n + n\mu(\log m)\log\log m)$$

bit operations. By Lemma 3.1 we know that Chinese remaindering at Step 11 takes

$$O(n\mu(\log m) + \mu(\log m)\log\log m)$$

bit operations. By Lemma 3.2 we know that Newton-Hensel lifting at Step 14 takes $O(\Pi(m,n))$ bit operations. Finally, it is straightforward to verify that `GetFactors` computes (m_1, m_2) in $O(\mu(\log m)\log\log m)$ bit operations. We conclude that, apart from the recursive calls, the cost of the algorithm is dominated by the cost of the gcd computation no matter which is the output of the gcd algorithm. Hence, there exists a constant c such that the total number of bit operations satisfies the recurrence

$$T(m,n) \le c\,\Gamma(m,n) + T(m_1,n) + T(m_2,n),$$

where we assume $T(m_2,n) = 0$ if $m_2 = 1$. Let $m = p_1^{k_1}\cdots p_h^{k_h}$ denote the prime factorization of m. Define $l(m) = k_1 + k_2 + \cdots + k_h$. We now show that $T(m,n) \le cl(m)\Gamma(m,n)$. Since $l(m) \le \log m$ this will prove the theorem. We prove the result by induction on $l(m)$. If $l(m) = 1$, then m is prime and the inequality holds since the computation is done without any recursive call. Let $l(m) > 1$. By induction we have

$$T(m_1,n) \le c\,l(m_1)\Gamma(m_1,n), \qquad T(m_2,n) \le c\,l(m_2)\Gamma(m_2,n).$$

Since $l(m_1) + l(m_2) = l(m)$, we have

$$T(m,n) \le c\Gamma(m,n) + c[l(m) - 1][\Gamma(m_1,n) + \Gamma(m_2,n)],$$

which implies the thesis since $\Gamma(m_1,n) + \Gamma(m_2,n) \le \Gamma(m,n)$. □

5. Inversion in $\mathbf{Z}_m[x]/(x^n - 1)$ when $n = 2^k$

In this section we describe an algorithm for computing the inverse of an $n \times n$ circulant matrix over $\mathbf{Z}_m$ when n is a power of 2. Our algorithm is inspired by a method first proposed by Dandelin, then by Lobachevsky [7], and rediscovered by Graeffe [14] for the approximation of polynomial zeros. The algorithm works by reducing the original problem to inversion of a circulant matrix of size $n/2$. This is possible because of the following lemma.

Lemma 5.1. *Let $f(x) \in \mathbf{Z}_m[x]$ and $n = 2^k$. If $f(x)$ is invertible over $\mathbf{Z}_m[x]/(x^n - 1)$, then $f(-x)$ is invertible as well. In addition, the product $f(x)f(-x)$ contains no odd power terms.*

Proof. Let $g(x)$ denote the inverse of $f(x)$. We have

$$f(-x)g(-x) \equiv 1 \pmod{(-x)^n - 1}.$$

Since n is even, $(-x)^n = x^n$ and the first thesis follows. To prove the second part of the lemma, let $f(x) = \sum_{i=0}^{n-1} a_i x^i$. The kth coefficient of the product $f(x)f(-x)$ is $\sum_{i+j=k} a_i a_j (-1)^j$. If k is odd, i and j must have opposite parity. Hence, the term $a_i a_j (-1)^j$ cancels with $a_j a_i (-1)^i$, and the sum is zero. □

```
Inverse3(f(x), m, n) → g(x)
{Computes the inverse g(x) of the polynomial f(x) in Z_m[x]/(x^n − 1), n = 2^k}
1.    if n = 1 then
2.       if gcd(f_0, m) = 1 return g_0 ← f_0^{-1};
3.       else return "f(x) is not invertible";
4.    else
5.      let F(x^2) ← f(x)f(−x) mod x^n − 1;
6.      let G(y) ← Inverse3(F(y), m, n/2);
7.      let S_e(x^2), S_o(x^2) be such that f(−x) = S_e(x^2) + xS_o(x^2);
8.      let T_e(y) ← G(y)S_e(y);
9.      let T_o(y) ← G(y)S_o(y);
10.      return g(x) ← T_e(x^2) + xT_o(x^2);
11.   endif
```

ALGORITHM 3. Inversion in $\mathbf{Z}_m[x]/(x^n-1)$. Requires $n=2^k$.

The above lemma suggests that we can halve the size of the original problem by splitting each polynomial in its even and odd powers. Let $F(x^2) = f(x)f(-x) \bmod x^n - 1$. By Lemma 5.1, if $f(x)$ is invertible the inverse $g(x)$ satisfies

$$F(x^2)g(x) \equiv f(-x) \pmod{x^n-1}. \tag{5}$$

Now we split $g(x)$ and $f(-x)$ in their odd and even powers. We obtain

$$g(x) = T_e(x^2) + xT_o(x^2), \qquad f(-x) = S_e(x^2) + xS_o(x^2).$$

From (5) we have

$$F(x^2)(T_e(x^2) + xT_o(x^2)) \equiv S_e(x^2) + xS_o(x^2) \pmod{x^n-1}.$$

If $f(x)$ is invertible over $\mathbf{Z}_m[x]/(x^n-1)$, $F(x^2)$ is invertible as well, its inverse being $g(x)g(-x)$. We can therefore retrieve $T_e(x^2)$ and $T_o(x^2)$ by solving the two subproblems

$$F(x^2)T_e(x^2) \equiv S_e(x^2) \pmod{x^n-1}, \quad F(x^2)T_o(x^2) \equiv S_o(x^2) \pmod{x^n-1}.$$

Hence, to find $g(x)$ it suffices to compute the inverse of $F(x^2)$ and to execute two multiplications between polynomials of degree $n/2$. By setting $y = x^2$, we reduce inverting $F(x^2)$ to an inversion modulo $x^{(n/2)} - 1$. Applying this approach recursively we obtain Algorithm 3 for the inversion over $\mathbf{Z}_m[x]/(x^n-1)$.

Theorem 5.2. *Algorithm 3 executes* $O(\Pi(m,n)+\mu(\log m)\log\log m)$ *bit operations.*

Proof. The thesis follows observing that the number of bit operations $T(m,n)$ satisfies the following recurrence:

$$T(m,n) = \begin{cases} \mu(\log m)\log\log m, & \text{if } n=1, \\ \Pi(m,n) + T(m,n/2) + 2\Pi(m,n/2), & \text{otherwise.} \end{cases}$$

□

Note that Algorithm 3 assumes nothing about m. When $m=2$, since $-x \equiv x \pmod 2$, $f(-x)$ is replaced with $f(x)$ in Algorithm 3, in fact $f(x)^2$ still contains only even powers.

```
Inverse4(f(x), m) → g(x)
{Computes the inverse g(x) of f(x) = Σ_{i=-r}^{r} a_i x^i}
1.    test if f(x) is invertible using Theorem 2.1;
2.    if  f(x) is invertible  then;
3.      let  M = 2^⌈2r log m⌉;
4.      let  h(x) ← Inverse3(x^r f(x), m, M);
5.       return  g(x) ← x^{r-M} h(x);
6.    endif
```

ALGORITHM 4. Inversion in $\mathbf{Z}_m\{x\}$.

We can use the technique described in this section to derive an efficient algorithm for the inversion of polynomials of degree $l \ll n$. This corresponds to the inversion of circulant matrices of small "bandwidth", that is, of the form $\sum_{i=0}^{l} \beta_i U^i$. The crucial observation is that if $f(x)$ has degree l and $2l < n$, then $F(y)$ has degree l as well. Hence, each recursive step halves the matrix size without changing the bandwidth. After $O(\log(n/l))$ steps we are left with a full matrix of size less than $2l$ which we invert using the standard algorithm. The running time of the complete procedure is $\Pi(m, l) \log n + \mu(\log m) \log\log m$ (where $O \log(n/l)\Pi(m, l)$ is for the first phase and $O(\Pi(m, l) + \mu(\log m) \log\log m)$ for the inversion of the dense matrix).

Remark: Inversion in $\mathbf{Z}_m\{x\}$. We now describe an algorithm for inverting a finite Laurent series $f(x) \in \mathbf{Z}_m\{x\}$. Our algorithm is based on the following observation which shows that we can compute the inverse of $f(x)$ inverting a polynomial over $\mathbf{Z}_m[x]/(x^n - 1)$ for a sufficiently large n.

Let $f(x) = \sum_{i=-r}^{r} a_i x^i$ denote an invertible finite Laurent series. By [12, Corollary 3.3] we know that the radius of the inverse is at most $R = (2 \log m - 1)r$. That is, the inverse $g(x)$ has the form $g(x) = \sum_{i=-R}^{R} b_i x^i$. Let M be such that $M > R + r = 2r \log m$. Since $f(x)g(x) = 1$ we have

$$[x^r f(x)][x^{M-r} g(x)] = x^M [f(x)g(x)] = x^M.$$

Hence, to compute the inverse of $f(x)$ it suffices to compute the inverse of $x^r f(x)$ over $\mathbf{Z}_m[x]/(x^M - 1)$. By choosing M as the smallest power of two greater than $2r \log m$, this inversion can be done using Algorithm 3 in

$$O(\Pi(m, 2r \log m) + \mu(\log m) \log\log m) = O(\Pi(m, 2r \log m))$$

bit operations. Verifying the invertibility of $f(x)$ using Theorem 2.1 takes $O(r\mu(\log m) \log\log m)$ bit operations, hence the cost of Algorithm 4 for inversion in $\mathbf{Z}_m\{x\}$ is $O(\Pi(m, 2r \log m))$ bit operations.

6. Conclusions and further works

We have described three algorithms for the inversion of an $n \times n$ circulant matrix with entries over the ring $\mathbf{Z}_m$. The three algorithms differ depending on the knowledge of m and n they require. The first algorithm assumes nothing about n but requires the factorization of m. The second algorithm requires nothing, while the third algorithm assumes nothing about m but works only for $n = 2^k$.

We believe it is possible to find new algorithms suited for different degrees of knowledge on m and n. A very promising approach is the following generalization of Algorithm 3. Suppose k is a factor of n and $\mathbf{Z}_m$ contains a primitive kth root of unity ω. Since $f(x)f(\omega x)\cdots f(\omega^{k-1}x) \bmod x^n - 1$ contains only powers which are multiples of k, reasoning as in Algorithm 3 we can reduce the original problem to a problem of size n/k. Since the ring $\mathbf{Z}_m$ contains a primitive pth root of unity for any prime divisor p of $\varphi(m)$, we can iterate this method to "remove" from n every factor which appears in $\gcd(n, \varphi(m))$. From that point the inversion procedure may continue using a different method (for example, Algorithm 1).

Given the efficiency of Algorithm 3 it may be worthwhile even to extend $\mathbf{Z}_m$ adding an appropriate root of unity in order to further reduce the degree of the polynomials involved in the computation. This has the same drawbacks we outlined for the FFT-based method. However, one should note that Algorithm 3 needs roots of smaller order with respect to the FFT method. As an example, for $n = 2^k$ Algorithm 3 only needs a primitive square root of unity, whereas the FFT method needs a primitive nth root of unity.

Acknowledgments

The authors wish to thank the anonymous referee whose useful comments greatly improved the paper.

A. Appendix

In this appendix we show that inversion of circulant and finitely generated bi-infinite Toeplitz matrices over $\mathbf{Z}_m$ arises naturally from the field of dynamical systems and in particular is related to the inversion of linear cellular automata over $\mathbf{Z}_m$.

Cellular Automata (CA) are dynamical systems consisting of a finite or infinite lattice of variables which can take a finite number of discrete values. The global state of the CA, specified by the values of all the variables at a given time, evolves in synchronous discrete time steps according to a given local rule which acts on the value of each single variable. In the following we restrict our attention to linear CA; that is, CA which are based on a linear local rule (they are sometimes called additive CA). Despite their apparent simplicity, linear CA may exhibit many complex behaviors (see for example [6, 8, 12, 11, 13, 16]). Linear CA have been used for pattern generation, design of error correcting codes and cipher systems, generation of hashing functions, etc. (see [3] for a survey of recent applications).

An infinite one-dimensional linear CA is defined as follows. For $m \geq 2$, let $\mathcal{C}_m$ denote the *space of configurations*

$$\mathcal{C}_m = \{c \mid \ c\colon \mathbf{Z} \to \mathbf{Z}_m\},$$

which consists of all functions from $\mathbf{Z}$ into $\mathbf{Z}_m$. Each element of $\mathcal{C}_m$ can be visualized as a bi-infinite array in which each cell contains an element of $\mathbf{Z}_m$. A local rule of *radius* r is defined by

$$f(x_{-r}, \ldots, x_r) = \sum_{i=-r}^{r} a_i x_i \mod m, \tag{6}$$

where the $2r+1$ coefficients $a_{-r},\dots,a_0,\dots,a_r$ belong to $\mathbf{Z}_m$. The global map $F\colon \mathcal{C}_m \to \mathcal{C}_m$ associated to the local rule f is given by

$$[F(c)](i) = \sum_{j=-r}^{r} a_j c(i+j) \mod m, \qquad \forall c \in \mathcal{C}_m,\ \forall i \in \mathbf{Z}.$$

In other words, the content of cell i in the configuration $F(c)$ is a function of the content of cells $i-r,\dots,i+r$ in the configuration c.

Finite one-dimensional additive CA (of size n) are defined over the configuration space

$$\mathcal{C}^*_{n,m} = \{c \mid\ c\colon \{0,1,\dots,n-1\} \to \mathbf{Z}_m\},$$

which can be seen as the set of all possible n-tuples of elements of $\mathbf{Z}_m$. To the local rule (6) we associate the global map $G\colon \mathcal{C}^*_{n,m} \to \mathcal{C}^*_{n,m}$ defined by

$$[G(c)](i) = \sum_{j=-r}^{r} a_j c(i+j \bmod n) \mod m, \qquad \forall c \in \mathcal{C}^*_{n,m},\ \forall i \in \{0,1,\dots,n-1\}.$$

In other words, the new content of cell i depends on the content of cells $i-r,\dots,i+r$, and we wrap around the borders of the array when $i+r \geq n$ or $i-r<0$.

Linear CA are often studied using Laurent series. To each configuration $c \in \mathcal{C}_m$ we associate the (infinite) Laurent series

$$P_c(x) = \sum\nolimits_{i\in\mathbf{Z}} c(i)x^i.$$

The advantage of this representation is that the computation of a linear map is equivalent to power series multiplication. Let $F\colon \mathcal{C}_m \to \mathcal{C}_m$ be a linear map with local rule (6). We associate to f the finite Laurent series $A(x) \in \mathbf{Z}_m\{x\}$ given by $A(X) = \sum_{i=-r}^{r} a_i x^{-i}$. Then, for any $c \in \mathcal{C}_m$ we have

$$P_{F(c)}(x) = P_c(x)A(x) \mod m.$$

For finite additive CA we use a similar representation. To each configuration $c \in \mathcal{C}^*_{n,m}$ we associate the polynomial of degree $n-1$

$$P_c(X) = \sum\nolimits_{i=0}^{n-1} c(i)x^i.$$

Then, for any configuration $c \in \mathcal{C}^*_{n,m}$ we have

$$P_{G(c)}(x) = [P_c(x)A(x) \pmod{x^n-1}] \mod m.$$

The above results show that the inversion of the global maps F and G is equivalent to the inversion of $A(x)$ in $\mathbf{Z}_m\{x\}$ and $\mathbf{Z}_m[x]/(x^n-1)$, respectively. Therefore they are also equivalent to the inversion of bi-infinite Toeplitz and circulant matrices.

References

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Massachussets, 1974. MR **54:**1706
2. D. Bini and V. Y. Pan. *Polynomial and Matrix Computations, Volume 1: Fundamental Algorithms*. Birkhäuser, 1994. MR **95k:**65003
3. P. Chaudhuri, D. Chowdhury, S. Nandi, and S. Chattopadhyay. *Additive Cellular Automata Theory and Applications, Vol. 1*. IEEE Press, 1997.
4. H. Cohen. *A course in computational algebraic number theory*, Volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, 1993. MR **94i:**11105
5. P. Feinsilver. *Circulants, inversion of circulants, and some related matrix algebras*. Linear Algebra and Appl., **56** (1984), 29–43. MR **85e:**15005

6. P. Guan and Y. He. *Exact results for deterministic cellular automata with additive rules.* Jour. Stat. Physics, **43** (1986), 463–478. MR **87j:**68087
7. A. S. Householder. *The Numerical Treatment of a Single Nonlinear Equation*, McGraw-Hill, New York, 1970. MR **52:**9593
8. M. Ito, N. Osato, and M. Nasu. *Linear cellular automata over Z_m.* Journal of Computer and System Sciences, **27** (1983), 125–140. MR **85f:**68074
9. A. Lempel, G. Seroussi, and S. Winograd. *On the complexity of multiplication in finite fields.* Theoretical Computer Science, **22(3)** (1983), 285–296. MR **84c:**68031
10. A. K. Lenstra and H. W. Lenstra. *Algorithms in number theory*, In J. van Leeuwen, editor, Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity. The MIT Press/Elsevier, 1990. CMP 92:01
11. G. Manzini and L. Margara. *A complete and efficiently computable topological classification of D-dimensional linear cellular automata over Z_m.* Theoretical Computer Science, **221(2)** (1999), 157–177. CMP 99:16
12. G. Manzini and L. Margara. *Invertible linear cellular automata over Z_m: Algorithmic and dynamical aspects.* Journal of Computer and System Sciences, **56** (1998), 60–67. MR **99j:**68089
13. O. Martin, A. Odlyzko, and S. Wolfram. *Algebraic properties of cellular automata.* Comm. Math. Phys., **93** (1984), 219–258. MR **86a:**68073
14. A. M. Ostrowski. *Recherches sur la méthode de Graeffe et les zéros des polynomes et des series de Laurent.* Acta Math., **72** (1940), 99–257. MR **2:**342c
15. A. Schönhage and V. Strassen. *Schnelle Multiplikation grosser Zahlen.* Computing, **7** (1971), 281–292. MR **45:**1431
16. S. Takahashi. *Self-similarity of linear cellular automata.* Journal of Computer and System Sciences, **44(1)** (1992), 114–140. MR **94b:**58058

Dipartimento di Matematica, Università di Pisa, Pisa, Italy
E-mail address: bini@dm.unipi.it

Istituto di Matematica Computazionale-CNR, Pisa, Italy
E-mail address: delcorso@imc.pi.cnr.it

Dip. Scienze e Tecnologie Avanzate, Università del Piemonte Orientale, and IMC-CNR, Pisa, Italy
E-mail address: manzini@mfn.unipmn.it

Dipartimento di Informatica, Università di Bologna, Bologna, Italy
E-mail address: margara@cs.unibo.it