An Interpolation Algorithm for Sparse Polynomials over $\mathbb{Z}_m$

Kai Werther*

August 9, 1994

Abstract

We give a quasi-polynomial time algorithm for the problem of interpolating sparse polynomials over integer residue class rings $\mathbb{Z}_m$ from their values given by a black box. This is a further development of [10].

1 Introduction and Preliminaries

The problem of reconstructing algebraic functions from some given data reaches back into the 17th century. Newton was among the first to discover an interpolation formula for univariate polynomials of degree d over the integers. He also proved that at least $d+1$ points are necessary for this purpose. It is easy to see that this generalizes to $(d+1)^n$ points for any interpolation formula for the class of integer polynomials in n variables with $\deg_{x_i} \leq d$. This number equals the number of undetermined coefficients of some polynomial f in this class. As this number is exponential in n such formulae can be considered to be impractical for large values of n.

Often the number t of nonzero coefficients of the polynomial f, hereafter denoted as the sparsity of f, is known to be small. The following question appears: Given a black box for an unknown n-variate polynomial f of degree d in each variable that is t-sparse, i.e., the sparsity of f is bounded by t, is there an algorithm for reconstructing f in time polynomial in n, t and d?

For the case of fields of characteristic zero, Zippel [12] answered the question in the affirmative, giving the first probabilistic algorithm for this problem. Using ideas of Grigoriev and Karpinski [5], Ben-Or and Tiwari [1] came up with a deterministic algorithm. In their seminal paper, Clausen et al. [2] showed that the same is not true for polynomials over finite fields. Although there exists efficient interpolation algorithms using field extension ([2, 6, 11, 13]), interpolation over the ground field requires at least a superpolynomial number of $V(n, \lfloor \log 2t \rfloor)$ evaluations, where

$$V(n,r) := \sum_{i=0}^{r} \binom{n}{i}.$$

Over the field with two elements $\mathbb{F}_2$ this constitutes also an upper bound. Roth and Benedek [8] and Dür and Grabmeier [3] gave algorithms for the sparse polynomial interpolation problem over $\mathbb{F}_2$ that use this optimal number of evaluations. For arbitrary finite fields, Werther [9, 10] has developed an algorithm using evaluation points in a set that is only a factor polynomial in t larger than a minimal set.

Let $\mathbb{Z}_m = \{0, \ldots, m-1\}$ denote the integer residue class ring $\mathbb{Z}/m\mathbb{Z}$. We consider the problem of interpolating sparse multivariate polynomials over $\mathbb{Z}_m$. The case of m being prime ($\mathbb{Z}_m$ being a field) has been studied in [9, 10]. In this paper we extend this result to arbitrary m.

To understand the difficulties that arise in transforming the interpolation algorithm from prime fields to $\mathbb{Z}_m$, we take a closer look at the ring $\mathbb{Z}_m[x_1, \ldots, x_n]$ of multivariate polynomials over $\mathbb{Z}_m$ and its relation to the

*Institut für Informatik V, Römerstr. 164, Universität Bonn, 53117 Bonn, Germany

ring $P[x_1, \ldots, x_n]$ of polynomial mappings, i.e, of all mappings $\varphi : \mathbb{Z}_m^n \to \mathbb{Z}_m$ that can be expressed by a polynomial, $\varphi = \varphi_f : a \mapsto f(a)$ for some polynomial $f \in \mathbb{Z}_m[x_1, \ldots, x_n]$. It is sufficient to consider the case of m being a prime power.

Section 2 deals with this problem where the notion of a degree-reduced polynomial is developed. We show that for all $\varphi \in P[x_1, \ldots, x_n]$ there is a unique degree-reduced polynomial f with $\varphi = \varphi_f$. Section 3 estimates the sparsity of a degree-reduced polynomial f in terms of the sparsity of any polynomial g satisfying $\varphi_f = \varphi_g$. In Section 4, we design an interpolation algorithm for t-sparse degree-reduced polynomials. This algorithm is also correct for arbitrary t-sparse polynomials by adjusting the parameter t using the results of Section 3 (Theorem 3.2). Finally, Section 5 genearlizes the algorithm to arbitrary values of m. A Mathematica Listing of an implementation can be found in the appendix.

2 Degree-Reduced Polynomials

A polynomial $f \in \mathbb{Z}_m[x_1, \ldots, x_n]$ induces a mapping

$$\varphi_f : \mathbb{Z}_m^n \to \mathbb{Z}_m, \quad (a_1, \ldots, a_n) \mapsto f(a_1, \ldots, a_n).$$

Let $P[x_1, \ldots, x_n]$ denote the subset of all polynomial mappings $\varphi_f : \mathbb{Z}_m^n \to \mathbb{Z}_m$. Obviously, $P[x_1, \ldots, x_n]$ is a ring and the map

$$\begin{array}{rcl} \Psi_m : \mathbb{Z}_m[x_1, \ldots, x_n] & \to & P[x_1, \ldots, x_n] \\ f & \mapsto & \varphi_f \end{array}$$

is a ring homomorphism. The kernel of Ψ_m is an ideal $I \subset \mathbb{Z}_m[x_1, \ldots, x_n]$ and the rings $\mathbb{Z}_m[x_1, \ldots, x_n]/I$ and $P[x_1, \ldots, x_n]$ are isomorphic. In a well-defined (black-box) interpolation problem one therefore seeks for an element of $\mathbb{Z}_m[x_1, \ldots, x_n]/I$.

In the following we will determine I and define for all cosets of $\mathbb{Z}_m[x_1, \ldots, x_n]/I$ an unique representative in $\mathbb{Z}_m[x_1, \ldots, x_n]$ that is minimal in some sense. We call these polynomials *degree-reduced polynomials*.

Let us consider the univariate case first. Let I_m be the kernel of $\Psi_m : \mathbb{Z}_m[x] \to P[x]$. I_m is a finite ideal and therefore finitely generated. The first lemma tells us that it is sufficient to determine I_m for m being a prime power.

Lemma 2.1 *Let $m = m_1 m_2$ where m_1 and m_2 are coprime. Then*

$$I_m = \{m_2 f_1 + m_1 f_2 \mid f_1 \in I_{m_1}, f_2 \in I_{m_2}\}.$$

Proof. Obviously, the right hand side is contained in I_m. Let $f \in \mathbb{Z}[x]$. For all integers m, f can be considered as a polynomial over $\mathbb{Z}_m$ by taking coefficients modulo m. If $f \in I_m$, then also $f \in I_{m_1}$ and $f \in I_{m_2}$. Since m_1 and m_2 are coprime there exists l_1 and l_2 with

$$m_1 l_2 + m_2 l_1 = 1.$$

Let $f_1 = l_1 f \in I_{m_1}$ and $f_2 = l_2 f \in I_{m_2}$. Then

$$m_2 f_1 + m_1 f_2 = (m_2 l_1 + m_1 l_2) f = f.$$

The assertion follows. □

In the following let m be a prime power. Lemma 2.2 reduces the determination of I_m to finding a monic polynomial of smallest degree in I_m.

Lemma 2.2 *Let $m = p^k$ be a prime power. For $1 \le l \le k$, let $g^{(l)}$ be a monic polynomial of smallest degree in I_{p^l}. Then*

$$I_m = \langle p^{k-l} g^{(l)} \rangle_{l=1}^{k}.$$

Proof. Again, the right hand side is contained in I_m. The other direction is proved by an induction on the degree d of a polynomial in I_m. □

If m is a prime, it is well known that $x^m - x = \prod_{a \in \mathbb{Z}_m}(x-a) \in \mathbb{Z}_m[x]$ is a monic polynomial of smallest degree in I_m. How does this generatlize to prime powers $m = p^k$ for $k > 1$?

The polynomial $\prod_{a \in Z_m}(x-a) \in \mathbb{Z}_m[x]$ is surely contained in I_m. Since $\lambda_p(x) = \prod_{a \in \mathbb{Z}_p}(x-a) \in \mathbb{Z}_p$, the polynomial $\lambda_p^k \in I_{p^k} = I_m$ is a monic polynomial of degree pk in I_m. However, there are monic polynomials of smaller degree than λ_p^k for the general case. For any ν, consider the polynomial

$$\lambda_\nu(x) := \prod_{i=0}^{\nu-1}(x-i) \in \mathbb{Z}_{p^k}[x].$$

Let a be in $\mathbb{Z}_{p^k}$ and consider the factors $l_i(a) = (a-i)$ of $\lambda_\nu(a)$. The factors $l_i(a)$ with $a \equiv i \pmod p$ are multiples of p. For each $a \in \mathbb{Z}_{p^k}$ there are at least $\lfloor \nu/p \rfloor$ factors that are multiples of p, at least $\lfloor \nu/p^2 \rfloor$ of these are multiples of p^2 and so on. In general, at least $\lfloor \nu/p^j \rfloor$ factors are multiples of p^j. Thus the product of these factors is divisible by the $\sum_{j \geq 1}\lfloor \nu p^{-j} \rfloor$-th power of p. Let us determine the smallest value of ν such that $\sum_{j \geq 1}\lfloor \nu p^{-j} \rfloor \geq k$. Obviously, $\nu = \nu(p,k)$ must be a multiple of p. With the auxilary functions $\sigma(p,r) := \sum_{j=0}^{\infty}\lfloor rp^{-j} \rfloor$, and its quasi-inverse $\tau(p,k) := \min\{r \mid \sigma(p,r) \geq k\}$, we have $\nu = p\tau(p,k)$.

Observe that σ is strongly monotonically increasing in its second argument, that

$$\sigma(p, \tau(p,k)-1) < k \leq \sigma(p, \tau(p,k)), \tag{1}$$

$$\tau(p, \sigma(p,k)) = k, \tag{2}$$

and that

$$\sigma(p, \sum_{j \geq 0} a_j p^j) = \sum_{i \geq 0}\left\lfloor \sum_{j \geq 0} a_j p^{j-i} \right\rfloor = \sum_{i \geq 0}\sum_{j \geq i} a_j p^{j-i} = \sum_{j \geq 0} a_j \sigma(p, p^j). \tag{3}$$

In the sequel, we will use the abbreviations $\tau = \tau(p,k)$, $\nu = \nu(p,k)$, and $\sigma(l) = \sigma(p,p^l)$ whenever p,k, and l are understood.

We prove that the degree ν is optimal.

Lemma 2.3 *There is no monic polynomial of degree less than ν in I_{p^k}.*

Proof. Let g be a monic polynomial in I_{p^k}. Consider any factorization of g (in general, this factorization is not unique):

$$g(x) = \prod_{a \in A}(x-a).$$

Let $d = |A|$ be the degree of g. By the pidgeon hole principle there is some $b \in \mathbb{Z}_{p^k}$, such that for all $l \geq 1$, A contains at most $\lfloor |A| p^{-l} \rfloor$ elements that are equivalent to b modulo p^l. Hence

$$g(b) \not\equiv 0 \pmod{p^{1+\sum_{l \geq 1}\lfloor |A| p^{-l} \rfloor}}.$$

If $|A| < \nu(p,k)$ then the left side of (1) implies

$$\sum_{i \geq 1}\lfloor |A| p^{-i} \rfloor = \sigma(p, \lfloor |A|/p \rfloor) < k.$$

Hence $g(b) \not\equiv 0 \pmod{p^k}$. The assertion follows. □

Applying the above lemma in conjunction with Lemma 2.2 we get

Corollary 2.4

$$I_{p^k} = \langle p^{k-l} \prod_{i=0}^{\nu(p,l)-1} (x-i) \rangle_{l=1}^{k}. \tag{4}$$

Example.

$$\begin{aligned} I_{81} = \; & \langle \; \prod_{i=0}^{8}(x-i), \quad 3\prod_{i=0}^{8}(x-i), \quad 9\prod_{i=0}^{5}(x-i), \quad 27\prod_{i=0}^{2}(x-i) \; \rangle \\ = \; & \langle \; x^9 + 45\,x^8 + 60\,x^7 + 12\,x^5 + 27\,x^4 + 26\,x^3 + 9\,x^2 + 63\,x, \\ & 9\,x^6 + 27\,x^5 + 36\,x^4 + 36\,x^2 + 54\,x, \quad 27\,x^3 + 53\,x \; \rangle. \end{aligned}$$

For every coset $[f]$ of f we define a unique representative in $[f]$.

Definition 2.5 A polynomial $f = \sum c_i x^i \in \mathbb{Z}_{p^k}[x]$ is called *degree-reduced* iff

$$\forall\, 0 \le j < k \quad \forall\, i \ge \nu(p, k-j): \quad c_i < p^j$$

or equivalently

$$\forall\, i: \quad c_i < p^{k - \sigma(p, \lfloor i/p \rfloor)}. \tag{5}$$

As can readily be seen from Corollary 2.4, each coset contains exactly one degree-reduced polynomial.

We extend the notion of degree-reduced polynomials to the multivariate case:

Definition 2.6 A polynomial $f = \sum c_\alpha \mathbf{x}^\alpha \in \mathbb{Z}_{p^k}[x_1, \ldots, x_n]$ is called *degree-reduced* if

$$\forall\, \alpha: \quad c_\alpha < p^{k - \sum_{i=1}^{n} \sigma(p, \lfloor \alpha_i/p \rfloor)}.$$

Otherwise f is called *degree-reducible*.

Again, each coset has a unique degree-reduced polynomial as its representative.

Lemma 2.7 *Each coset contains exactly one degree-reduced polynomial.*

Proof. To prove the lemma, it suffices to show that $p^j \mathbf{x}^\alpha$ is degree-reducible iff $j \ge k - \sum_{i=1}^{n} \sigma(p, \alpha_i/p)$. By Definition 2.5 the univariate monomial $x_i^{\alpha_i}$ is degree-reducible in $\mathbb{Z}_{p^l}[x_i]$ iff $\sigma(p, \lfloor \alpha_i/p \rfloor) \ge l$. Since the variables are independent, $\mathbf{x}^\alpha$ is degree-reducible in $\mathbb{Z}_{p^{k-j}}[x_1, \ldots, x_n]$ iff $\sum_{i=1}^{n} \sigma(p, \lfloor \alpha_i/p \rfloor) \ge k - j$. The assertion follows. □

3 Sparsity

In this section, we investigate the relationship between the number of terms of a polynomial f and the corresponding degree-reduced polynomial g in the coset $[f]$.

We define a family $\{h_i\}$ of sparse polynomials generating I_m and use these polynomials to estimate the sparsity of the degree-reduced polynomial of $[f]$ in terms of the sparsity of f.

Define $h_0(x) := x^p - x$ and, recursively,

$$h_{l+1}(x) := (h_l(x))^p - p^{(p^{l+1}-1)} h_l(x).$$

The degree d of h_l is p^{l+1}. Furthermore, the expanded form of h_l is

$$x^d + c_{d-(p-1)} x^{d-(p-1)} + c_{d-2(p-1)} x^{d-2(p-1)} + \ldots, \tag{6}$$

as can be seen by induction on l. The main property of h_l is characterized by the following lemma.

Lemma 3.1 *For all l and all $a \in \mathbb{Z}$, $h_l(a)$ is divisible by $p^{\sigma(l)}$.*

By Lemma 2.3, $\sigma(l)$ is the largest power of p for which the above statement holds.

Proof. The proof is by induction on l. For $l = 0$ the statement follows from the well known fact that $x^p - x \equiv 0 \pmod p$. Using the induction hypothesis for l we can write $h_l(x) = p^{\sigma(l)} g(x)$ for some function $g : \mathbb{Z} \to \mathbb{Z}$. Thus, using the definition of h_{l+1}, we have

$$\begin{aligned} h_{l+1}(x) &= (g(x)p^{\sigma(l)})^p - p^{p^{l+1}-1} g(x) p^{\sigma(l)} \\ &= g^p(x) p^{p\sigma(l)} - g(x) p^{p^{l+1}-1+\sigma(l)}. \end{aligned} \tag{7}$$

Furthermore,

$$p\sigma(l) = p \sum_{j=0}^{l} p^j = \sigma(l+1) - 1 = p^{l+1} + \sigma(l) - 1$$

and, for some $\tilde{g} : \mathbb{Z} \to \mathbb{Z}$, $g(x)^p - g(x) = p\tilde{g}(x)$. Therefore, (7) simplifies to

$$h_{l+1}(x) = p^{\sigma(l+1)-1} (g(x)^p - g(x)) = p^{\sigma(l+1)} \tilde{g}(x).$$

The statement follows. □

Let $r = \sum_i a_i p^i$ be in p-adic expansion. Then using the additivity (3) of σ we conclude that $p^{\sigma(p,r)}$ divides

$$g_r = h_0^{a_0} h_1^{a_1} \cdots.$$

Consequently, p^k divides $g_{\tau(p,k)}$. Also, observe that $g_{\tau(p,k)}$ is of the form (6) and its degree is $\nu(p,k)$. Therefore we have

$$I_{p^k} = \langle p^{k-l} g_{\tau(p,l)} \rangle_{l=1}^{k}.$$

The degree-reduced polynomial of a coset $[f]$ can be computed by adding an appropriate element from I_m. If f is a monomial x^L, then subtracting a polynomial $x^d g_\tau$ of degree L from x^L will only introduce terms x^l, such that $L - l$ is a multiple of $p - 1$. Hence the degree-reduced polynomial g of $[x^L]$ is of the form (6) with $d < \nu(p,k)$. Furthermore, g never contains a constant term, except for $L = 0$.

The following theorem bounds the sparsity of the degree-reduced representative of the coset of some multivariate monomial.

Theorem 3.2 *Let $\mathbf{x}^\alpha \in \mathbb{Z}_{p^k}[x_1, \ldots, x_n]$ be a multivariate monomial. Then the degree-reduced representative f of $[\mathbf{x}^\alpha]$ consists of at most*

$$2^{k-1} \binom{n-2+k}{k-1}$$

terms.

If $n = 1$ the above discussion shows that this number can also be bounded by $\nu(p,k)/(p-1)$.

Proof. Given a monomial $\mathbf{x}^\alpha$, we count the number of degree-reduced monomials $\mathbf{x}^\beta$ satisfying

$$\forall\, i: \quad \alpha_i \equiv \beta_i \pmod{p-1}. \tag{8}$$

Monomials that do not satisfy (8) can not be introduced during the calculation of the degree-reduced polynomial in $[\mathbf{x}^\alpha]$. A degree-reduced monomial $\mathbf{x}^\beta$ satisfies

$$\sum_{i=1}^{n} \lfloor \sigma(p, \lfloor \beta_i / p \rfloor) \rfloor < k.$$

Since $l \leq \sigma(p,l)$, any degree-reduced polynomial also satisfies

$$\sum_{i=1}^{n} \rho(\beta_i) \leq k-1, \tag{9}$$

where $\rho(\beta_i) = \lfloor \beta_i/p \rfloor$. Therefore, the sparsity of the degree-reduced polynomial in $[\mathbf{x}^\alpha]$ is bounded from above by the number of monomials satisfying (8) and (9). Since two different values of β_i differ by at least $p-1$ there are at most two values of β_i with the same value of $\rho(\beta_i)$. Since for $\alpha_i \neq 0$ also $\beta_i \neq 0$, there is only one possible value for β_i in $\{1,\ldots,p-1\}$. This implies that the value of β_i with $\rho(\beta_i) = 0$ is unique. Thus there are at most 2^{k-1} different β such that for all i the value of $\rho(\beta_i)$ is the same.

The number $R(n,k)$ of n-tuples $(\rho_1,\ldots,\rho_n)$ of non negative integers with $\sum_{i=1}^{n} \rho_i \leq k-1$ is the number of β that differ in at least one component of $(\rho(\beta_1),\ldots,\rho(\beta_n))$ and satisfy (9). This number can be estimated by the following recursion formula:

$$R(n,k) \leq R(n-1,k) + R(n-1,k-1) + \ldots + R(n-1,1)$$

with the basis $R(n,1) = 1$. One can easily check that

$$R(n,k) \leq \binom{n-2+k}{k-1},$$

using the parallel summation formula for binomial coefficients (c.f. page 174 of [4]).

It follows that the total number of degree-reduced monomials that can be derived from a monomial is bounded by $2^{k-1}\binom{n-2+k}{k-1}$. □

Corollary 3.3 *Let $f \in \mathbb{Z}_{p^k}[x_1,\ldots,x_n]$ be t-sparse. Then the degree-reduced polynomial g in $[f]$ consists of at most*

$$2^{k-1}\binom{n-2+k}{k-1}t$$

monomials.

4 Interpolation modulo prime powers

In this section, we develop an interpolation algorithm for degree-reduced t-sparse multivariate polynomials over $\mathbb{Z}_{p^k}$. Our algorithm is derived from an interpolation algorithm for t-spare multivariate polynomials over finite fields ([9],[10]).

In order to avoid a clumpsy notation, let $r = \lceil \log(t+1) \rceil$. Then $t \leq 2^r - 1$.

The following simple lemma reveals a recursive definition of multivariate polynomials and is crucial for the algorithm. We call a polynomial $h \in \mathbb{Z}_{p^k}[x_1,\ldots,x_n]$ *degree-bounded*, if $\deg_{x_i} h < \nu := \nu(p,k)$ for all $1 \leq i \leq n$.

Lemma 4.1 *Let $f \in \mathbb{Z}_{p^k}[x_1,\ldots,x_n]$ be a (2^r-1)-sparse degree-bounded polynomial. Rewrite f as*

$$f(x_1,\ldots,x_n) = \sum_{i=0}^{\nu} x_n^i f_i(x_1,\ldots,x_{n-1}). \tag{10}$$

Let

$$f_+ = f_0 + \ldots + f_{\nu-1}. \tag{11}$$

Then all but at most one of the coefficient polynomials, say f_j, are $(2^{r-1}-1)$-sparse, and f_+ and f_j are (2^r-1)-sparse. Furthermore, if f is degree-reduced, the subpolynomials f_i are degree-reduced polynomials in $\mathbb{Z}_{p^{k'}}[x_1,\ldots,x_{n-1}]$ for $k' = k - \sigma(p,\lfloor i/p \rfloor)$.

Proof. The assertion on the sparsity of the polynomials is obvious from the pigeon hole principle. For the second part of the lemma consider any subterm $cx_1^{\alpha_1}x_2^{\alpha_2}\cdots x_n^{\alpha_n}$ of f. If f is degree-reduced we have

$$c < p^k - \textstyle\sum_{i=1}^{n} \sigma(p, \lfloor \alpha_i/p \rfloor)$$

by Definition 2.6. The above term is absorbed in f_{α_n} and $cx_1^{\alpha_1}x_2^{\alpha_2}\cdots x_{n-1}^{\alpha_{n-1}}$ is the corresponding subterm of f_{α_n}. Note that

$$c < p^{k'} - \textstyle\sum_{i=1}^{n-1} \sigma(p, \lfloor \alpha_i/p \rfloor)$$

for $k' = k - \sigma(p, \lfloor \alpha_n/p \rfloor)$ (furthermore, this is the smallest possible k' in general). Thus f_{α_n} can be thought of as a degree-reduced polynomial over $\mathbb{Z}_{p^{k'}}$. □

Following [10], we proceed using a recursive scheme. Given the values of f at certain points, we compute values for the subpolynomials. Then, the subpolynomials are reconstructed from these values, and finally, the subpolynomials determine f.

It is important to construct a set of appropriate points at which f is evaluated. First, the points in this set S must contain enough information theoretic content to uniquely determine f. Second, for the recursive approach the set of points at which the subpolynomials are evaluated must have the same structure as S.

The following set satisfies both requirements:

$$S_\nu(n, r) := \{(a_1, \ldots, a_n) \in \{0, \ldots, \nu - 1\}^n \mid \#\{i \mid a_i \neq 1\} \leq \lfloor \log r \rfloor\}.$$

The values of f at certain points are related to those of the subpolynomials f_i via a linear system of equation, a so called Vandermonde system. More precisely, the following lemma holds.

Lemma 4.2 *Let f be a degree-reduced polynomial in $\mathbb{Z}_{p^k}[x_1, \ldots, x_n]$ and let f_i the subpolynomials of f defined by Equation (10). Then for all $a \in (\mathbb{Z}_{p^k})^{n-1}$*

$$\begin{pmatrix} f(a,0) \\ f(a,1) \\ \vdots \\ f(a,\nu-1) \end{pmatrix} = \begin{pmatrix} 0^0 & 0^1 & \cdots & 0^{\nu-1} \\ 1^0 & 1^1 & \cdots & 1^{\nu-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \nu-1 & \cdots & (\nu-1)^{\nu-1} \end{pmatrix} \cdot \begin{pmatrix} f_0(a) \\ f_1(a) \\ \vdots \\ f_{\nu-1}(a) \end{pmatrix} \tag{12}$$

Furthermore, if $(c_0, \ldots, c_{\nu-1})$ and $(d_0, \ldots, d_{\nu-1})$ are two solutions of (12), then

$$c_i \equiv d_i \pmod{p^{k-\sigma(p,\lfloor i/p \rfloor)}} \tag{13}$$

Proof. The first statement is elementary, the second follows from Corollary 2.4. □

We will call a solution $(c_0, \ldots, c_{\nu-1})$ *degree-reduced*, if the corresponding polynomial $f = \sum_{i=0}^{\nu} c_i x^i$ is degree-reduced. Now we can prove the following important property of $S_{\nu(p,k)}(n, r)$.

Lemma 4.3 *Let $f \in \mathbb{Z}_{p^k}[x_1, \ldots, x_n]$ be a nonvanishing degree-bounded $(2^{r+1} - 1)$-sparse polynomial. Then there is an $a \in S_{\nu(p,k)}(n, r)$ with $f(a) \neq 0$.*

Proof. We proceed by induction on n and r. If $n = 1$, f is a univariate polynomial. Since f is nonvanishing, the degree-reduced polynomial $\tilde{f}$ in $[f]$ is nonzero. Lemma 4.2 implies that at least one of the values $f(0), \ldots, f(\nu - 1)$ is nonzero. If $r = 0$, f consists of at most one monomial. Hence if f is nonzero, then also $f(1, \ldots, 1) \neq 0$. As can easily be verified, the corresponding sets $S_\nu(1, r)$ and $S_\nu(n, 0)$ contain the necessary test elements.

For the induction step let $n > 1$ and assume the correctness of the lemma for all values smaller than n. Analogously to (10), f has the decomposition

$$f(x_1, \ldots, x_n) = x_n^0 f_0(x_1, \ldots, x_{n-1}) + \ldots + x_n^{\nu-1} f_{\nu-1}(x_1, \ldots, x_{n-1}).$$

Consider the following two cases:

- $f_+ = f_0 + \ldots + f_{\nu-1}$ is nonzero. Since f_+ is a $(2^{r+1}-1)$-sparse $(n-1)$-variate polynomial, by the induction hypothesis for $n-1$, f_+ can be separated from zero by some element $a \in S_\nu(n-1,r)$. Thus $(a,1) \in S_\nu(n,r)$ separates f from zero.
- f_+ is zero. Then at least one of the f_i is a nonzero (2^r-1)-sparse $(n-1)$-variate polynomial. By the induction hypothesis there is some $a \in S_\nu(n-1,r-1)$ with $f_i(a) \neq 0$. Plugging a into the first $n-1$ variables makes f a nonzero univariate polynomial. Using the induction hypothesis for $n=1$, we have at least one nonzero value among $f(a,0), f(a,1), \ldots, f(a,\nu-1)$. By the given structure of the set, $S_\nu(n,r)$ contains the points $(a,0),\ldots,(a,\nu-1)$. $\square$.

This lemma implies that evaluating at the points in $S_\nu(n,r)$ is sufficient for reconstructing any (2^r-1)-sparse degree-reduced polynomial.

Corollary 4.4 *Let h be a (2^r-1)-sparse degree-bounded polynomial in $\mathbb{Z}_{p^k}[x_1,\ldots,x_n]$. Then h is uniquely determined by its values at $S_{\nu(p,k)}(n,r)$.*

Proof. Suppose h_1 and h_2 are degree-bounded (2^r-1)-sparse polynomials and take the same values at $S_\nu(n,r)$. Then $f = h_1 - h_2$ is a degree-bounded $(2^{r+1}-1)$-sparse nonzero polynomial that evaluates to zero at all points of $S_\nu(n,r)$. This contradicts Lemma 4.3. $\square$

In the following we assume all polynomials in question to be degree-bounded.

Instead of the set $S_\nu(n,r)$ the algorithm uses a list that is adopted to the recursive structure of the algorithm. This list, hereafter also denoted by $S_\nu(n,r)$, consists of ν copies of the list $S_\nu(n-1,r-1)$ (for the subpolynomials f_i) and a list $S_\nu(n-1,r)$ (for the polynomial f_+). It is easy to see that the size of the list is bounded by

$$|S_\nu(n,r)| \leq \nu^{r+1} V(n,r). \tag{14}$$

But there are difficulties in transforming Corollary 4.4 into an efficient algorithm. First of all, given the values of f at the points in $S_{\nu(p,k)}(n,r)$ the algorithm should output a uniquely determined polynomial from $[f]$. If we choose this polynomial to be the unique degree-reduced polynomial in $[f]$, then the sparsity might increase (Theorem 3.2). Therefore we restrict ourselves to degree-reduced polynomials. Compare this to the case of finit fields, where the notions of degree-bounded polynomials and degree-reduced polynomials are identical (c.f. [10]).

There are two more problems. The first also appears in [10]: given a (2^r-1)-sparse f, we are not guaranteed that all subpolynomials f_i of the decomposition (10) are $(2^{r-1}-1)$-sparse.

The following lemma is the key for solving this problem.

Lemma 4.5 *Let f, f_+, and f_i, $0 \leq i < \nu$ be specified as in Lemma 4.1. Let $0 \leq j < \nu$ and g_j be such that $\#g_j \leq 2^{r-1}-1$ and $\#(f_j - g_j) > 2^r$. Then*

$$\#g_j + \#(f_+ - g_j) > 2^r.$$

Proof. Using the triangle inequatlity $\#g_j > 2^r - \#f_j$ we obtain

$$\begin{aligned}
\#g_j + \#(f_+ - g_j) &> 2^r - \#f_j + \#(\sum_{i\neq j} f_i + (f_j - g_j)) \\
&> 2^r - \#f_j + 2^r - \#(\sum_{i\neq j} f_i) && \text{(triangle inequality)} \\
&= 2\cdot 2^r - \#f_j - \#(\sum_{i\neq j} f_i) \geq 2^r. && \square
\end{aligned}$$

Lemma 4.5 implies a method to check the correctness of the subpolynomials returned by recursive calls. Having detected the possibly incorrect polynomial, the correct polynomial can easily computed using (11).

The second problem is new: the Vandermonde system (12) is singular, i.e., the solution of the system is not unique. This singularity is directly coupled with the fact that there might exist many degree-bounded polynomials that lie in the same coset. This in turn does not mean that the subpolynomials also lie in the same cosets. Merely, the different solutions of the Vandermonde system corresponds to different degree-bounded subpolynomials. We solve this problem by reconstructing the *unique* degree-reduced polynomial in the coset.

We compute the subpolynomials $f_0, \ldots, f_{\nu-1}$ in groups $f_{jp}, \ldots, f_{jp+p-1}$ of p polynomials starting from $f_{\nu-p}, \ldots, f_{\nu-1}$ (thus j runs from $\tau - 1$ down to 0). These $(n-1)$-variate polynomials can be considered as being polynomials over $\mathbb{Z}_{p^{k'}}$ for $k' = k - \sigma(p,j)$ (Corollary 2.4). Using the Vandermonde sytem (12) we compute the values of these polynomials modulo $p^{k'}$. The recursion step with appropriate parameters will return the polynomials f_i for $jp \le i < (j+1)p$. Since these polynomials are known we can use the values of the modified polynomial $\tilde{f} = f - \sum_{jp \le i < (j+1)p} f_i$ for the next smaller value of j.

But there is still a catch. Again, we are not guaranteed that the f_i are reconstructed correctly. To compute the group $f_{jp}, \ldots, f_{jp+p-1}$ correctly, we do not only use the recursion step for these polynomials, but also for the polynomials f_i, $i < jp$ considering these also as polynomials over $\mathbb{Z}_{p^{k'}}$ (this does not increase the sparsity of f_i). Now we use the test implied by Lemma 4.5. Finally, this gives the correct polynomials $f_{jp}, \ldots, f_{jp+p-1}$.

After the description of the main ideas we sketch the algorithm.

Input: the number of variables n, an integer r, such that the sparsity $t \le 2^r - 1$, a prime p, a nonnegative exponent k, the set $S = S_{\nu(p,k)}(n,r)$ together with values $y(a)$ for all $a \in S$.

Output: the unique degree-reduced (2^r-1)-sparse polynomial $f \in \mathbb{Z}_{p^k}[x_1, \ldots, x_n]$ with $f(a) = y(a)$ for all $a \in S$, if it exists, "failure" otherwise.

Step 1: Treat the cases $n = 0$, $r = 0$, and $\forall\, a \in S : y(a) = 0$.

Step 2: Compute for all $0 \le i < \nu(p,k) =: \nu$ the values of f_i at $S_\nu(n-1, r-1)$ selecting the degree-reduced solution from the Vandermonde system (12). Set $\tau = \tau(p,k)$. The values of $f_+(a)$ for $a \in S_\nu(n-1,r)$ are copied from those of $y(a,1)$.

Step 3: **if** $n = 1$ **then** construct f **else** perform Step 4.

Step 4: **for** $j = \tau - 1$ **down to** 0 **do**

1. Compute $k' = k - \sigma(p,j)$, $\nu' = \nu(p,k')$.
2. For $0 \le i < (j+1)p$ compute the values of f_i at points in $S_{\nu'}(n-1, r-1)$ modulo $p^{k'}$.
3. Compute the values of f_+ at points in $S_{\nu'}(n-1,r)$ modulo $p^{k'}$.
4. For $0 \le i < (j+1)p$ call the procedure recursively for the polynomials f_i with parameter $n-1$, $r-1$, p, k' and the set $S_{\nu'}(n-1, r-1)$ together with the corresponding values.
5. Call the procedure recursively for the polynomial f_+ with parameter $n-1$, r, p, k' and the set $S_{\nu'}(n-1,r)$ together with the corresponding values.
6. **if** $f_+ =$ "failed" of two $f_i =$ "failed" **then return** "failed".
7. **if** one $f_i =$ "failed" **then** $l := i$ **else** use Lemma 4.5 to find out the index l of the possibly incorrect polynomial.
8. **if** $jp \le l < (j+1)p$ **then** correct the polynomial using (11).
9. Compare the values of f_i, $jp \le i < (j+1)p$ modulo p^k with those obtained in Step 2 and compute a solution of the Vandermonde system that is consistent with these values.

Step 5: Construct f via (10). Return f, if it is (2^r-1)-sparse, otherwise return "failure".

It remains to show the correctness of the algorithm and to give a running time analysis.

Theorem 4.6 *The above algorithm is correct, i.e., it computes the output data from the input data.*

Proof. The proof is by induction on n and r. The basis of induction $n = 0$ or $r = 0$ is trivial. The case $n = 1$ is handled in Step 3. The coefficients c_i of f are just the values f_i. Now consider the case $n > 1$. The decomposition of f into degree-reduced subpolynomial as in (10) is unique.

We prove by induction on j, $j = \tau - 1, \ldots, 0$, that the input of the recursive calls is computed correctly. Let $0 \le j \le \tau - 1$ and suppose that the polynomials f_i, $i \ge (j+1)p$ are known. If $j < \tau - 1$ then the polynomials f_i, $i \ge (j+1)p$ were obtained in the previous iteration and proved to be correct by induction hypothesis, otherwise the set $\{f_i \mid i \ge (j+1)p\}$ is empty. In the first case, Step 4.9 of the previous iteration gives a solution of (12) that is consistent with f_i, $i \ge (j+1)p$. By Lemma 4.2, the polynomials $f_{jp}, \ldots, f_{jp+p-1}$ can be considered as polynomials over $\mathbb{Z}_{p^{k'}}$ for $k' = k - \sigma(p, j)$. Furthermore, the polynomials f_i, $i < jp$ can be considered over $\mathbb{Z}_{p^{k'}}$ and by Lemma 4.2 any solution to (12) will give the same values modulo $p^{k'}$. Especially, Steps 4.2 and 4.3 are well defined. Furthermore, the sparsity of the polynomials f_i, $i < jp$, when considered as polynomials over $\mathbb{Z}_{p^{k'}}$, does not increase.

The main induction hypothesis on n and r implies that for all i the recursive call returns either the polynomial f_i, some polynomial g_i that coincide with f_i on the set $S_{\nu(p,k')}(n-1, r-1)$, or the value "failed". The polynomial f_i is returned, if the values $y(a)$ attached to the points in $a \in S_{\nu(p,k')}(n-1, r-1)$ correspond to a $(2^{r-1}-1)$-sparse polynomial. A similar statement holds for f_+. By Lemma 4.1, the following may happen:

- The value of the recursive call for f_+ is "failed" or the value of at least two recursive calls for f_i are "failed". Then there is no $(2^r - 1)$-sparse polynomial satisfying the requirements of the algorithm and "failed" is returned.
- The value of one recursive call for f_i is "failed". Then this polynomial can be corrected by (11) and the polynomial f can be constructed.
- All recursive calls return a polynomial but polynomial f_i is detected to be incorrect by means of Lemma 4.5. Then this polynomial can be corrected by (11) and the polynomial f can be constructed.
- All recursive calls return a polynomial and all are proved to be correct. Then f can be constructed from these polynomials.

These cases are handled in Steps 4.6 - 4.8. Now the polynomials $f_{jp}, \ldots, f_{jp+p-1}$ are computed correctly. Step 4.9 computes a consistent solution of (12) settling the induction hypothesis for the next smaller value of j. Finally, Step 5 outputs the specified data. □

Theorem 4.7 *The algorithm with parameter n, r, p, and k uses*

$$\mathcal{O}(n^k p^k (pk)^{r+1} V(n, r))$$

arithmetic operation with integers bounded by p^k.

Proof. We simplify the analysis of the algorithm using the estimation $\nu(p, k) \le pk$. Fix p and let $T(n, r, k)$ be the number of arithmetic operation performed by the algorithm with parameter n, r and k. The amount of time used by the algorithm is majorized by Step 2 and the recursion step (Step 4). Step 2 can be completed in

$$(pk)^2 |S_{pk}(n-1, r-1)| + |S_{pk}(n-1, r)| \le pk S_{pk}(n, r)$$

arithmetic steps. The amount of operations of the recursion step can be estimated by

$$\sum_{j=1}^{k} T(n-1, r, j) + \sum_{j=1}^{k} (k-j+1) T(n-1, r-1, j) + c' pk |S_{pk}(n, r)|$$

for some constant c'. Since the work is dominated by Steps 2 and 4, there is a c such that the following recurrence relation holds

$$T(n, r, k) \le \sum_{j=1}^{k} [T(n-1, r, j) + kT(n-1, r-1, j)] + cpk |S_{pk}(n, t)| \tag{15}$$

Using the fact that $|S_\nu(n,r)| = |S_\nu(n-1,r)| + \nu|S_\nu(n-1,r-1)|$ it is easy to verify that

$$T(n,r,k) := cn^k p^k |S_{pk}(n,r)|$$

is a solution of (15). Since $|S_{\nu(p,k)}| \leq \nu^{(r+1)}V(n,r)$ by (14), the assertion follows. □

5 Interpolation in general rings

In this section we extend the results of the previous section to general integer residue class rings $\mathbb{Z}_m$. The main tool is the Chinese Remainder Theorem. We will use it in the following form (c.f. [7]).

Theorem 5.1 *Chinese Remainder Theorem*
Let m be an integer and $q_1, \ldots, q_l$ be pairwise relatively prime with $\prod_{i=1}^{l} q_i = m$. Let $0 \leq a_i < q_i$, $1 \leq i \leq l$ be integers. Then there exists a unique $0 \leq a \leq m$, such that for all $1 \leq i \leq l$, $a_i \equiv a \pmod{q_i}$.

We apply the Chinese Remainder Theorem in the following situation. Let m be an integer and a polynomial $f \in \mathbb{Z}_m[x_1, \ldots, x_n]$ be given. Let $q_1 = p_1^{k_1}, \ldots, q_l = p_l^{k_l}$ be the prime powers appearing in m. Then f can be considered as a polynomial $g_i \in \mathbb{Z}_{q_i}[x_1, \ldots, x_n]$. It is important to note that the sparsity of g_i is at most the sparsity $t = 2^r - 1$ of f.

Therefore, for all $1 \leq i \leq l$ we can evaluate the black box for f at points at $S^{(i)} = S_{\nu(p_i,k_i)}(n,r)$ and reduce these values modulo $p_i^{k_i}$. This gives the values of g_i at points in $S^{(i)}$. The interpolation algorithm of the previous section will compute g_i from these values. Using the Chinese Remainder Theorem in the opposite direction, the polynomials g_i give a polynomial from $[f]$ of sparsity at most tl. This can then be reduced to the t-sparse f without any extra cost

Observe, that $S_\nu(n,r) \subset S_{\nu'}(n,r)$ whenever $\nu < \nu'$. Therefore, we query the blackbox at $S_{\nu'}(n,r)$ for $\nu' = \max_{1 \leq i \leq l}\{\nu(p_i,k_i)\}$.

Altogether, we have the following main result.

Theorem 5.2 *Given a black box for a (2^r-1)-sparse degree-reduced polynomial $f \in \mathbb{Z}_m[x_1, \ldots, x_n]$. Let the prime decomposition of $m = p_1^{k_1} \cdots p_l^{k_l}$, $k' = \max\{k_i\}$ and $\nu' = \max\{\nu(p_i,k_i)\}$. Then there is an algorithm for reconstructing f using*

$$|S_{\nu'}(n,r)|$$

queries to the black box and

$$\mathcal{O}(lmn^{k'}|S_{\nu'}(n,r)|$$

arithmetic operation with integers bounded by m.

Using Theorem 5.2 in conjunction with Corollary 3.3 gives

Corollary 5.3 *Given a black box for a (2^r-1)-sparse polynomial $f \in \mathbb{Z}_m[x_1, \ldots, x_n]$. Let the prime decomposition of $m = p_1^{k_1} \cdots p_l^{k_l}$, $k' = \max\{k_i\}$, $r' = r + k' + \log\binom{n-2-k'}{k'-1} - 1$, and $\nu' = \max\{\nu(p_i,k_i)\}$. Then there is an algorithm for reconstructing f using*

$$|S_{\nu'}(n,r')|$$

queries to the black box and

$$\mathcal{O}(lmn^{k'}|S_{\nu'}(n,r')|$$

arithmetic operation with integers bounded by m.

A The Mathematica Listing

```
RingInter[n_,r_,p_,k_, evalf_] :=
(* RingInter creates a polynomial over Z_(p^k) with at most
   2^r-1 terms in variables x[1],...,x[n], that satisfies the
   point-value pairs given by evalf                              *)

 Module[{Levalf,    (* list of evaluations of subpolynomials of f *)
             A,     (* Vandermonde Matrix for computing the values *)
            AI,     (* LUP decomposition of A *)
           vec,     (* vector used with A and argument of f *)
         i,j,l,     (* index variables *)
           tau,     (* parameter dependent on p and k *)
            nu,     (* parameter dependent on p and k *)
          kNew,     (* parameter for recursion *)
          mNew,     (* parameter for recursion *)
         nuNew,     (* parameter for recursion *)
         index,     (* parameter for recursion *)
     AuxLevalf,     (* auxilary list of evaluations *)
             f,     (* resulting polynomial *)
          fAux,     (* auxilary polynomial *)
            Lf,     (* list of subpolynomials *)
          flag,     (* auxilary boolean variable *)
             m = p^k,     (* size of the ring *)
     failindex = 0,       (* index for error recovery *)
             t = 2^r-1},  (* bound on the number of terms of f *)

   If[n <= 0,                     (* RingInter may never be called with *)
     Return[failed]               (* a nonpositive number of variables  *)
   ];

   If[Length[evalf] < 1,          (* RingInter may never be called with *)
     Return[failed]               (* an empty list                      *)
   ];

   If[r == 0,                     (* if r=0 then the polynomial must be *)
     If[evalf[[1,n+1]] == 0,      (* the constant 0                     *)
       Return[0],
       Return[failed]
     ]
   ];
                                  (* now n>=1 and r>=1                  *)

   For[i=1,i<=Length[evalf] && evalf[[i,n+1]] == 0, i++,];
                                  (* check whether all values of the    *)
                                  (* list are zero                      *)
   If[i>Length[evalf] && evalf[[i-1,n+1]] == 0, Return[0]];
                                  (* if this is the case return the     *)
                                  (* zero polynomial                    *)

   tau = Tau[p,k];
   nu = p tau;
   Levalf = ExtractList[evalf,n,nu];
                                  (* otherwise compute a list of the    *)
                                  (* evaluations of the subfunctions    *)
                                  (* and compute the right values       *)
   Lf = Table[0, {i,1,nu+1}];
```

```
  If[n==1,                        (* if n=1 the subpolynomials must     *)
    A = Vandermonde[nu,m];
    AI = LUPdecompose[A,nu,m];
    vec = Table[Levalf[[j,1,1]], {j,1,nu}];
    vec = InvProd[AI[[2]],AI[[1]],vec,nu,m];
    For[i=1,i<=nu,i++,            (* be constants                *)
      Lf[[i]] = vec[[i]]
    ],
(*Else*)                          (* otherwise we continue as follows   *)
    For[j=tau,j>=1,j--,           (* the fi split into tau groups each  *)
                                  (* of which we compute separately     *)
      failindex = 0;
      kNew = k-Sigma[p,j-1];
      mNew = p^(kNew);
      index = p*j;                (* set up new values for k,m and nu   *)
      nuNew = Nu[p,kNew];

      AuxLevalf = Table[{},{i,1,index+1}];
                                  (* and auxilary lists                 *)

      A = Vandermonde[index,mNew];
      AI = LUPdecompose[A,index,mNew];
      count = 0;
      For[i=1,i<=Length[Levalf[[1]]],i++,
        flag = True;
        For[l=1,l<n,l++,
          If[Levalf[[1,i,l]] >= nuNew,
              flag = False, Break[]
          ]
        ];
        If[flag,
          count += 1;
          vec = Table[Levalf[[l,i,n]], {l,1,index}];
          vec = InvProd[AI[[2]],AI[[1]],vec,index,mNew];
          For[l=1,l<=index,l++,
            AuxLevalf[[l]] = Append[AuxLevalf[[l]],Levalf[[1,i]]];
            AuxLevalf[[l,count,n]] = vec[[l]];
          ]
        ];
      ];

      For[i=1,i<= Length[Levalf[[nu+1]]],i++,
        vec = Levalf[[nu+1,i]];
        flag = True;
        For[l=1,l<n,l++,
          If[vec[[l]] >= mNew,
                                  (* that contain only evaluations at   *)
                                  (* small points                       *)
            flag = False; Break[]
          ]
        ];
        If[flag,
          vec[[n]] = Mod[vec[[n]],mNew];
                                  (* furthermore the value is reduced  *)
          AuxLevalf[[index+1]] = Append[AuxLevalf[[index+1]],vec]
        ]
      ];  (* end for i *)
      fAux = RingInter[n-1,r,p,kNew,AuxLevalf[[index+1]]];
```

```
                          (* the values for f+ are correct as   *)
                          (* they are copies from the input,    *)
                          (* thus we can interpolate f+         *)

If[fAux == failed,        (* if f+ can not be interpolated      *)
  Return[failed]           (* correctly that f can't             *)
];

Lf[[nu+1]] = fAux;        (* otherwise store f+                 *)

For[i=1,i<= Length[Levalf[[1]]],i++,
  vec = Levalf[[1,i]];
  flag = True;
  For[l=1,l<n,l++,
    If[vec[[l]] >= mNew,
                          (* that contain only evaluations at   *)
                          (* small points                       *)
      flag = False; Break[]
    ]
  ];
  If[flag,
    For[l=1,l<=index,l++,
      vec = Levalf[[l,i]];
      vec[[n]] = Mod[vec[[n]],mNew];
                          (* furthermore the value is reduced  *)
      AuxLevalf[[l]] = Append[AuxLevalf[[l]],vec]
    ]
  ]
];  (* end for i *)
                          (* using these auxilary lists we can *)
                          (* compute the polynomials f_(p j-p) *)
                          (* to f_(p j -1)                     *)
For[i=1,i<=index,i++,
  fAux = RingInter[n-1,r-1,p,kNew,AuxLevalf[[i]]];
  If[fAux == failed,
    If[failindex == 0, failindex = i,
     Return[failed]
    ]
  ];
  Lf[[i]] = fAux
];

If[failindex == 0,
  For[i=1,i<=index && failindex == 0,i++,
    fAux = PolynomialMod[Expand[Lf[[nu+1]] - Lf[[i]]],mNew];
    If[PolynomialLength[fAux] + PolynomialLength[Lf[[i]]] > t,
        failindex = i
    ]
  ]
];
If[failindex > p (j-1),
  Lf[[failindex]] = 0;
  Lf[[failindex]] = PolynomialMod[Expand[Lf[[nu+1]]
                       - Sum[Lf[[i]],{i,1,nu}]],mNew]
];


If[j>1,
```

```
      For[i=p*j, i>=p (j-1)+1,i--,
        For[l=1,l<=Length[Levalf[[i]]],l++,
          a =Levalf[[i,l,n]] -  EvalPoly[Lf[[i]],Delete[Levalf[[i,l]],n]];
          Levalf[[i-p+1,l,n]]  = PolynomialMod[Levalf[[i-p+1,l,n]] +
                    Levalf[[i,l,n]] -
                    EvalPoly[Lf[[i]],Delete[Levalf[[i,l]],n]],m];
        ]
       ]
     ];

  ]  (* end for j *)
];  (* end if n=1 *)

f = PolynomialMod[Expand[Sum[Lf[[i]] * x[n]^(i-1), {i,1,nu}]],m];
If[PolynomialLength[f] > t, Return[failed], Return[f]]
]
```

References

[1] Ben-Or, M., Tiwari, P., *A Deterministic Algorithm for Sparse Multivariate Polynomial Interpolation*, STOC 20, pp. 301–309, 1988.

[2] Clausen, M., Dress, A., Grabmeier, J., Karpinski, M., *On Zero-Testing and Interpolation of k-sparse Multivariate Polynomials over Finite Fields*, TCS **84**, pp. 151–164, 1991.

[3] Dür, A., Grabmeier, J., *Applying Coding Theory to Sparse Interpolation*, SIAM J. Comput. **22**, 695–704, 1993.

[4] Graham, R. L., Knuth, D. E., Patashnik, O., **Concrete Mathematics**, Addison Wesley, 1989.

[5] Grigoriev, D. Y., Karpinski, M., *The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents is in NC*, FOCS 28, pp. 166–172, 1987.

[6] Grigoriev, D. Y., Karpinski, M., Singer, M. F. *Fast Parallel Algorithms for Sparse Multivariate Polynomial Interpolation over Finite Fields*, SIAM J. Comput. **19**, pp. 1059–1063, 1990.

[7] Niven, I., Zuckerman, H. S., **An Introduction to the Theory of Numbers**, John Wiley & Sons, 1980.

[8] Roth, R. M., Benedek, G. M., *Interpolation and Approximation of Sparse Multivariate Polynomials over GF(2)*, SIAM J. Comput. **20**, pp. 291–314, 1991.

[9] Werther, K., *Interpolation und Approximation Boolescher Formeln*, Masters Thesis, University of Bonn, October 1991.

[10] Werther, K., *The Computational Complexity of Interpolating Sparse Multivariate Polynomials over Finite Fields*, Research Report. No 8577-CS, University of Bonn, 1992, to appear in AAECC.

[11] Werther, T., *A Survey of Efficient Parallel Algorithms for Sparse Interpolation over Arbitrary Fields*, Research Report No. 8529-CS, University of Bonn, 1988.

[12] Zippel, R., *Probabilistic Algorithms for Sparse Polynomials*, LNCS 72, pp. 216–226, 1979.

[13] Zippel, R., *Interpolating Polynomials from their Values*, J. Symb. Comp. **9**, 375–403, 1990.