Early Termination in Sparse Interpolation Algorithms

Erich Kaltofen [a], Wen-shin Lee [b]

[a] *Department of Mathematics, North Carolina State University*
Raleigh, North Carolina 27695-8205, U.S.A.

[b] *School of Computer Science, University of Waterloo*
Waterloo, Ontario N2L 3G1, Canada

Abstract

A probabilistic strategy, *early termination,* enables different interpolation algorithms to adapt to the degree or the number of terms in the target polynomial when neither is supplied in the input. In addition to dense algorithms, we implement this strategy in sparse interpolation algorithms. Based on early termination, *racing algorithms* execute simultaneously a dense and a sparse algorithm. The racing algorithms can be embedded as the univariate interpolation substep within Zippel's multivariate method. In addition, we experimentally verify some heuristics of early termination, which make use of thresholds and post-verification.

Key words: Early termination, sparse polynomial, black box polynomial, interpolation, sparse interpolation, randomized algorithm, Chebyshev basis, Pochhammer basis, racing two algorithms, Zippel's algorithm, Ben-Or's and Tiwari's algorithm.

Email addresses: `kaltofen@math.ncsu.edu` (Erich Kaltofen), `ws2lee@scg.uwaterloo.ca` (Wen-shin Lee).

URLs: `www.kaltofen.us` (Erich Kaltofen), `www.wen-shin.com` (Wen-shin Lee).

Appears in J. Symbolic Comput., vol. 36, nr. 3–4, pp. 365–400 (2003).

1 Introduction

1.1 *Polynomial representations and interpolations*

A polynomial $f(x_1, \dots, x_n)$ is represented as

$$f(x_1, \dots, x_n) = \sum_{j=1}^{t} c_j x_1^{e_{j,1}} \cdots x_n^{e_{j,n}} \text{ (in the standard power basis).} \qquad (1)$$

The black box representation of a polynomial is an object that takes as input a value for each variable and evaluates the polynomial at the given input. To determine the coefficients and terms of a black box polynomial is the problem of black box interpolation.

In general, in a polynomial in (1) with total degree $d = \deg f$, the number of non-zero terms could be as many as $\binom{n+d}{d}$. When there are much fewer non-zero terms, there are efficient interpolation algorithms that take advantage of such situation: for polynomials that are sparse in the multivariate case, Zippel's probabilistic algorithm (Zippel 1979a) is more efficient than a variable by variable Lagrange or Newton interpolation; based on the Berlekamp/Massey algorithm (Massey 1969) from coding theory, Ben-Or and Tiwari (1988) gave an algorithm that interpolates all variables at once. Both approaches have been generalized and improved: the Vandermonde techniques of Ben-Or/Tiwari can be applied to Zippel's algorithm (Zippel 1990; Kaltofen and Lakshman Yagati 1988); the Ben-Or/Tiwari approach have been extended to some non-standard polynomial bases (Lakshman Y. N. and Saunders 1995). For polynomials over small finite fields both require modification (Grigoriev et al. 1990; Zilic and Radecka 1999, and the references given there).

1.2 *Early termination strategy and racing algorithms*

The algorithms described so far require a bound in the input: a degree bound for dense algorithms; and a bound on the number of terms for sparse algorithms. When no such bound is supplied, an efficient probabilistic approach, *early termination*, can be employed. This is based on the fact that an already interpolated polynomial does not change as more interpolation points are added.

Our early termination algorithms are randomized in the Monte Carlo sense: their results are correct with high probability. In our implementation, we further adopt another strategy of putting partial verifications into our procedures, and the early termination is only triggered after encountering a series of zero discrepancies in a row. The length of the series is a *threshold* given as an optional argument. For dense interpolations, we show how a higher threshold can improve the lower bound of the probability of correctness (Lemma 2). For sparse interpolations, we prove the early termination is correct for threshold one, and note that higher thresholds weed out bad random choices from sets that are much smaller than the early termination theorem would require. Further analysis is complicated and our early termination algorithms with higher thresholds thus become heuristics that can interpolate polynomials of a size at the very edge of what current software and hardware can reach.

Sparse algorithms are less efficient when the target polynomial is dense. Based on early termination, we propose *racing algorithms* that run a dense against a sparse algorithm on a same set of evaluation points. The racing algorithm is superior since it terminates as soon as either of the racer algorithms terminates while requiring no additional evaluations in comparison to a single algorithm.

The early termination strategy seems to belong to the “folklore” of computer algebra. We have used early termination in the mid-1980s (Freeman et al. 1988; Kaltofen and Trager 1990) for purpose of determining the degree of a straight-line and black box polynomial. The algorithms perform Newton interpolation at non-random points and test whether the interpolant agrees with the input polynomial at a random point (“post testing”, see section 6), thus allowing for preconditioning in the interpolation process while guaranteeing a given probability of success. Chinese remaindering with early termination is applied to exact computations in geometry by Emiris (1998). Austin Lobo observed early termination phenomenon in the setting of the Wiedemann (1986) algorithm. We also note the vanishing of Wronskians as a criterion of t-sparsity in (Grigoriev et al. 1991, 1994).

1.3 Hybrids of Zippel algorithm and other improvements

Zippel’s algorithm has a shortcoming over Ben-Or’s and Tiwari’s: it interpolates one variable at a time, and that each variable is interpolated densely. On the other hand, when the Ben-Or/Tiwari algorithm is implemented in a modular fashion (Kaltofen et al. 1990), in the multivariate case the modulus must be large enough to recover all non-zero terms. We also notice that the Ben-Or/Tiwari algorithm with rational number arithmetic causes extreme intermediate expression swell, while in Zippel’s algorithm the modulus only needs to capture the coefficients, and be large enough for randomization.

For multivariate polynomial interpolations, we propose the hybrids of Zippel’s algorithm: under Zippel’s variable by variable method, each variable is interpolated through a racing algorithm. Thus we can ameliorate the inefficiency of dense univariate interpolations in the original Zippel's algorithm, and reduce the large modulus required by the Ben-Or/Tiwari in the multivariate case.

Refining the idea of prunings via homogenization (Díaz and Kaltofen 1998), we present and discuss *permanent prunings* and *temporary prunings*.

1.4 Maple implementation and further developments

Some of our ideas are implemented in a Maple package, `ProtoBox`. Clearly, there is a trade-off between the operations introduced by concurrently performing two interpolation algorithms in a racing algorithm and the savings of polynomial evaluations. We intend our algorithms for polynomials produced by the calculus of black box polynomials (Kaltofen and Trager 1990; Díaz and Kaltofen 1998).

1.5 Related work

Some of the results here have been reported in preliminary form in (Kaltofen et al. 2000) and are part of Lee's Ph.D. thesis (Lee 2001). In (Giesbrecht et al. 2003, 2002) we have used our early termination approach to extend our sparse interpolation algorithms to the problem of computing sparsest shifts, that is, computing elements $a_1, \ldots, a_n$ in the coefficient field or an algebraic extension such that $f(y_1 + a_1, \ldots, y_n + a_n)$ (see (1)) has a minimum number of terms in the y_i. Again we consider power, Chebyshev and Pochhammer bases.

2 Early Termination in the Standard Basis

2.1 Early termination with thresholds in dense interpolations

To interpolate a univariate polynomial $f(x)$ from its evaluations at distinct points $p_0, p_1, \ldots$, a dense algorithm updates an i-th interpolant $f^{[i]}(x)$ for every $i \geq 0$, where $f^{[i]}(x)$ is a polynomial interpolating $f(p_0), \ldots, f(p_i)$ and $\deg f^{[i]}(x) \leq i$. Since at least one i-th order term is constructed in every $f^{[i]}(x)$, the target polynomial is recovered as a possible dense polynomial up to the degree bound.

In Newton's interpolation, $f^{[0]}(x) = f(p_0)$, and for $i > 0$, c_i the i-th divided difference, $f^{[i]}(x)$ is updated as:

$$f^{[i]}(x) = f^{[i-1]}(x) + c_i(x-p_0)(x-p_1)\cdots(x-p_{i-1}).$$

Note that the target polynomial can be viewed as being interpolated in a mixed power basis: $1, (x-p_0), (x-p_0)(x-p_1), (x-p_0)(x-p_1)(x-p_2), \ldots$.

Once the target polynomial is interpolated, the interpolant does not change even if we keep interpolating $f(x)$ at more distinct points, namely, $f^{[d+j]}(x) = f(x)$ for $d = \deg f$ and $j \geq 0$. Based on the observation, the early termination with thresholds is applied as the following: an integer $\eta > 0$ is given as a threshold, the sequence $p_0, p_1, \ldots$ are random values, and $f^{[i]}(x)$ is updated for every $i \geq 0$. Whenever $f^{[i]}(x)$ stops changing η times in a row, $f(x) = f^{[i]}(x)$ with high probability.

Theorem 1 (Early termination with threshold in dense univariate interpolations) *Given are a black box univariate polynomial $f(x)$ over a field and an integer $\eta > 0$ as the threshold. Let $p_0, p_1, \ldots$ be chosen randomly and uniformly from a subset S of the domain, and $f^{[i]}(x)$ the i-th interpolant that interpolates $f(p_0), \ldots, f(p_i)$. Note that p_i are not necessarily all distinct. If d is the smallest non-negative integer such that*

$$f^{[d]}(x) = f^{[d+1]}(x) = \cdots = f^{[d+\eta]}(x), \tag{2}$$

then $f^{[d]}(x)$ correctly interpolates $f(x)$ with probability no less than

$$1 - \eta \cdot \deg f(x) \cdot \left(\frac{\deg f(x)}{\#(S)}\right)^{\eta}. \tag{3}$$

PROOF. If d is the smallest integer that satisfies (2) and $f^{[d]}(x) \neq f(x)$, then both of the following happen:

1. either $d = 0$, or p_d is not a root of $f(x) - f^{[d-1]}(x)$;

2. $p_{d+1}, \ldots, p_{d+\eta}$ are all roots of $f(x) - f^{[d]}(x)$.

If $f^{[d]}(x) \neq f(x)$, by the nature of a dense algorithm, $\deg f^{[d]}(x) < \deg f(x)$ and $\deg(f(x) - f^{[d]}(x)) = \deg f(x)$. There are at most $\deg f(x)$ distinct roots in $f(x) - f^{[d]}(x)$. The probability of randomly hitting a root of $f(x) - f^{[d]}(x)$ in S is no more than $\deg f(x)/\#(S)$. We define a probability function $P(i)$ as the following: when $i = 0$, $P(i)$ is the probability that $f^{[0]}(x) \neq f(x)$ but $f^{[0]}(x) = f^{[1]}(x) = \cdots = f^{[\eta]}(x)$, that is, $p_1, \ldots, p_\eta$ are all roots of $f(x) - f^{[0]}(x)$; when $i \geq 1$, $P(i)$ is the probability that $f^{[i]}(x) \neq f(x)$ and i is the smallest integer such that $f^{[i]}(x) \neq f^{[i-1]}(x)$ and $f^{[i]}(x) = f^{[i+1]}(x) = \cdots = f^{[i+\eta]}(x)$, in other word, $p_{i+1}, \ldots, p_{i+\eta}$ are all roots of $f(x) - f^{[i]}(x)$. For every $i \geq 0$, $P(i) \leq (\deg f(x)/\#(S))^\eta$ because we need to hit a root of $f(x) - f^{[i]}(x)$ for η times.

If $f(x)$ is interpolated correctly, at most $\eta \cdot \deg f(x)$ values can be interpolated before the target polynomial is obtained, which only happens when each interpolant stops changing for exactly $\eta - 1$ many times. Therefore, $\sum_{i=0}^{\eta\cdot\deg f(x)-1} P(i)$ covers all the possibilities of $f(x)$ being falsely interpolated, and $f^{[d]}(x)$ correctly interpolates $f(x)$ with probability no less than

$$1 - \sum_{i=0}^{\eta\cdot\deg f(x)-1} P(i) \geq 1 - \eta \cdot \deg f(x) \cdot \left(\frac{\deg f(x)}{\#(S)}\right)^{\eta}. \quad \square$$

In Theorem 1, we estimate the probability loosely: whenever $f^{[i]}(x) \neq f^{[i-1]}(x)$, $f(x)$ cannot be falsely interpolated at any of $f^{[i+1]}(x), \ldots, f^{[i+\eta-1]}(x)$.

Based on (3), when $\#(S)$ is large enough, a higher threshold can improve the lower bound of the probability of correctness.

Lemma 2 *In Theorem 1, the lower bound of the probability of correctness in* (3) *can be improved when the threshold η is increased to $\eta + \Delta\eta$ if*

$$\left(\frac{\deg f(x)}{\#(S)}\right)^{\Delta\eta} < \frac{\eta}{\eta + \Delta\eta}$$

and $\Delta\eta$ is a positive integer.

The lower bound discussed in Theorem 1 and Lemma 2 does not reflect the real performance improved by higher thresholds, which are evident for small $\#(S)$ (see Section 6 for test results). The points $p_0, p_1, \ldots$ are not necessarily distinct in Theorem 1, in our implementation, instead of $f^{[i]}(x)$, we update $f^{\{k\}}(x)$ at a non-repeated point so that $f^{\{k\}}(x)$ interpolates the first $k+1$ *distinct* ones (also see the algorithm steps in Subsection 4.2). This modification avoids the false early terminations due to interpolating at repeated points. To ensure a successful interpolation, the size of S needs to cover enough distinct points required by the early termination, that is, no less than $\deg f(x) + 1 + \eta$.

2.2 The Ben-Or/Tiwari sparse interpolation algorithm

The Berlekamp/Massey algorithm (Massey 1969) processes a stream of elements $a_0, a_1, \ldots$ from a field $\mathbb{K}$. If the sequence is linearly generated, the algorithm can determine its minimal polynomial $\Lambda(z) = z^t + \lambda_{t-1}z^{t-1} + \cdots + \lambda_0$ such that

$$a_{t+j} = -\lambda_{t-1}a_{t-1+j} - \cdots - \lambda_0 a_j \text{ for all } j \geq 0 \tag{4}$$

after processing exactly $2t$ elements. A linear generator (4) is a column relation in an infinite Hankel matrix. The Berlekamp/Massey algorithm updates that relation as depicted in Figure 1 below. A generator $\Lambda^{[L]}$ of degree L is valid as

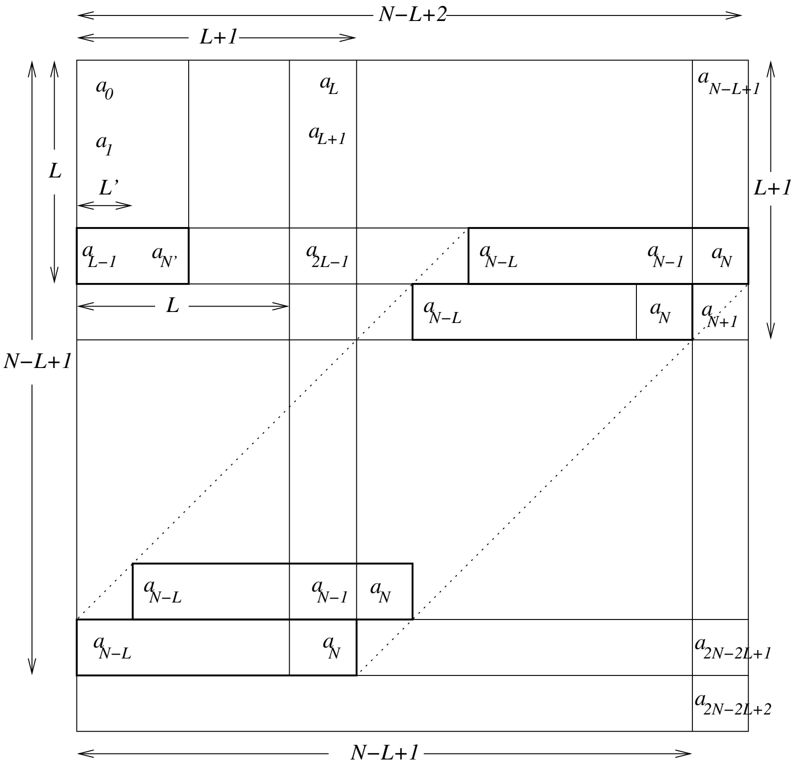

Fig. 1. Berlekamp/Massey algorithm.

far as a_{N-1}, but fails to generate a_N. By induction hypothesis we can assume that $\Lambda^{[L]}$ is minimal for a_{2L-1}, and more concretely that the leading principal $L \times L$ submatrix in Figure 1 was non-singular. The new generator is of degree $N - L + 1$ (Massey 1969, cf. Theorem 1): the last column of the $(L + 1) \times (N - L + 1)$ submatrix cannot be generated by preceding columns, because the corresponding row in the transposed matrix has increased the rank to $L+1$. First, one captures a_N by a linear combination of the previous generator $\Lambda^{[L']}$ and the shifted $\Lambda^{[L]}$. Both generators leave a non-zero discrepancies, the former for $a_{N'}$ and the later for a_N, so a linear combination can generate a_N. That linear combination works for $a_{N-1}, a_{N-2}, \ldots, a_{N-L+1}$ because both $\Lambda^{[L']}$ and the shifted $\Lambda^{[L]}$ have zero discrepancies in those rows. The elements $a_{N+1}, a_{N+2}, \ldots, a_{2N-2L+1}$ are generated by modifying the newly constructed linear combination further through the discrepancies of the step-wise shifted $\Lambda^{[L]}$ in each new row. Finally, the leading principal $(N-L+1) \times (N-L+1)$ submatrix is non-singular, because any column relation could be shifted up and right to give one that generates the last column of the $(L+1) \times (N-L+1)$ submatrix.

The Berlekamp/Massey algorithm implements both kinds of updates, namely jumping the degree and completing to a square submatrix in a single loop with a conditional to test which case one is processing. We note that the generator $\Lambda^{[N-L+1]}$ has a non-zero discrepancy for $a_{2N-2L+2}$ if and only if the leading principal $(2N - 2L + 2) \times (2N - 2L + 2)$ submatrix is non-singular.

Now consider a multivariate polynomial f over a field of characteristic zero:

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{e_{j,1}} \cdots x_n^{e_{j,n}} = \sum_{j=1}^{t} c_j \beta_j(x_1, \ldots, x_n), \quad c_j \neq 0. \tag{5}$$

Let $p_1, \ldots, p_n$ be distinct primes, $b_j = \beta_j(p_1, \ldots, p_n) = p_1^{e_{j,1}} \cdots p_n^{e_{j,n}}$, and $a_i = f(p_1^i, \ldots, p_n^i) = \sum_{j=1}^{t} c_j b_j^i$. Define an auxiliary polynomial $\Lambda(z)$ as:

$$\Lambda(z) = \prod_{j=1}^{t} (z - b_j) = z^t + \lambda_{t-1} z^{t-1} + \cdots + \lambda_0.$$

Theorem 3 *For a polynomial f in* (5)*, and $a_i = f(p_1^i, \ldots, p_n^i)$ with distinct primes $p_1, \ldots, p_n$, the sequence $\{a_i\}_{i \geq 0}$ is linearly generated by $\Lambda(z)$. Furthermore, $\Lambda(z)$ is the minimal polynomial of $\{a_i\}_{i \geq 0}$* (Ben-Or and Tiwari 1988)[1].

The Berlekamp/Massey algorithm can determine $\Lambda(z)$ from $\{a_i\}_{i \geq 0}$; by finding the roots of $\Lambda(z)$, b_j can be obtained. Then each term $\beta_j = x_1^{e_{j,1}} \cdots x_n^{e_{j,n}}$ are recovered through repeatedly dividing b_j by $p_1, \ldots, p_n$. Finally, the coefficients c_j are computed via solving the linear system $a_i = \sum_{j=1}^{t} c_j b_j^i$ with $0 \leq i \leq t-1$, which turns out to be a $t \times t$ transposed Vandermonde system:

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \ldots & b_t^{t-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{bmatrix}. \tag{6}$$

Efficient algorithms for solving transposed Vandermonde systems can be found in (Kaltofen and Lakshman Yagati 1988; Zippel 1990).

Algorithm: `Ben-Or/Tiwari`

```
Input:  ▸ $f(x_1, \ldots, x_n)$: a multivariate black box polynomial.
        ▸ $\tau$: $\tau \geq t$, where $t$ is the number of the terms with non-zero coefficients in $f$.
Output: ▸ $c_j$ and $\beta_j$: $f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j \beta_j$ and $c_j \neq 0$.
(1) [The Berlekamp/Massey algorithm.]
    $a_i = f(p_1^i, \ldots, p_n^i), 0 \leq i \leq 2\tau - 1$, where $p_1, \ldots, p_n$ are relatively prime.
    Compute $\Lambda(z)$ from $\{a_i\}_{2\tau-1 \geq i \geq 0}$.
(2) [Determine $\beta_j$.]
    Find all $t$ distinct roots of $\Lambda(z)$, which are $b_j$.
    Determine each $\beta_j$ through repeatedly dividing every $b_j$ by $p_1, \ldots, p_n$.
```

[1] George Labahn has pointed out to us a similarity of the Ben-Or and Tiwari algorithm to Prony's method (Prony III (1795)) in signal processing.

(3) [Compute the coefficients c_j.]
Solve a transposed Vandermonde system.

2.3 Early termination in the Ben-Or/Tiwari interpolation algorithm

Both algorithms of Ben-Or and Tiwari (1988) and Kaltofen et al. (1990) need to know the number of terms t, or an upper bound $\tau \geq t$. Otherwise, we can guess τ, within τ compute a candidate polynomial g for f, and then compare g and f at an additional random point. If the values are different, or it fails in computing g, we double our guess for τ.

The early termination version of the Ben-Or/Tiwari algorithm requires a single interpolation run. Here is the basic idea: pick a random point $p = (p_1, \ldots, p_n)$ for the evaluations $f(p_1^i, \ldots, p_n^i)$ in the Ben-Or/Tiwari algorithm, and show that with high probability the embedded Berlekamp/Massey algorithm does not encounter a singular $L \times L$ principal submatrix (see Figure 1) until $L = t + 1$.

However, this is not generally true: for any $f(x) = \sum_{j=1}^{t} c_j x^{e_j}$ that satisfies $f(p^0) = a_0 = c_1 + \cdots + c_t = 0$, the first discrepancy is zero. We have two ways to fix this problem: either pick another random $p_c \neq 0$ and proceed the interpolation with $f + p_c$ (see Subsection 3.4); or, as shown in this section, shift the sequence by one element.

We want to show that for symbolic values $x_1, \ldots, x_n$, the first singular leading principal submatrix appears at $L = t + 1$. Let $\beta_j = x_1^{e_{j,1}} \cdots x_n^{e_{j,n}}$ be the j-th non-zero term in f, and $\alpha_i = f(x_1^i, \ldots, x_n^i)$ the symbolic evaluations of f at powers, we have

$$\mathcal{A}_i = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_i \\ \alpha_2 & \alpha_3 & \ldots & \alpha_{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_i & \alpha_{i+1} & \ldots & \alpha_{2i-1} \end{bmatrix} = \mathcal{B}_i C_t \bar{\mathcal{B}}_i^{\mathrm{Tr}}, \tag{7}$$

where

$$\mathcal{B}_i = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \beta_1 & \beta_2 & \ldots & \beta_t \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1^{i-1} & \beta_2^{i-1} & \ldots & \beta_t^{i-1} \end{bmatrix}, \quad C_t = \begin{bmatrix} c_1 & 0 & \ldots & 0 \\ 0 & c_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & c_t \end{bmatrix},$$

and

$$\bar{\mathcal{B}}_i = \begin{bmatrix} \beta_1 & \beta_2 & \dots & \beta_t \\ \beta_1^2 & \beta_2^2 & \dots & \beta_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1^i & \beta_2^i & \dots & \beta_t^i \end{bmatrix}.$$

Theorem 4 *The determinant of $\mathcal{A}_i$ is non-zero for $i = 1, \dots, t$.*

PROOF. Let $M_{J,K}$ be the determinant of the submatrix of M consisting of rows in J and columns in K. By the Binet-Cauchy formula (Gantmacher 1977),

$$(AB)_{J,L} = \sum_{1 \le k_1 < k_2 < \cdots < k_i \le n} A_{J,\{k_1,\dots,k_i\}} B_{\{k_1,\dots,k_i\},L}, \tag{8}$$

where n is the number of columns in A and $\#(J) = \#(L) = i$.

Applying (8) to (7) with $I = \{1, \dots, i\}$ for $1 \le i \le t$, we have

$$\begin{aligned} \det \mathcal{A}_i = (\mathcal{B}_i C_t \bar{\mathcal{B}}_i^{\mathrm{Tr}})_{I,I} &= \sum_J \sum_K (\mathcal{B}_i)_{I,J} (C_t)_{J,K} (\bar{\mathcal{B}}_i^{\mathrm{Tr}})_{K,I} = \sum_J (\mathcal{B}_i)_{I,J} (C_t)_{J,J} (\bar{\mathcal{B}}_i^{\mathrm{Tr}})_{J,I} \\ &= \sum_{J=\{j_1,\dots,j_i\}} c_{j_1} \cdots c_{j_i} \, \beta_{j_1} \beta_{j_2} \cdots \beta_{j_i} \cdot \det \left(\begin{bmatrix} 1 & 1 & \dots & 1 \\ \beta_{j_1} & \beta_{j_2} & \cdots & \beta_{j_i} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{j_1}^{i-1} & \beta_{j_2}^{i-1} & \dots & \beta_{j_i}^{i-1} \end{bmatrix} \right)^2 \\ &= \sum_{J=\{j_1,\dots,j_i\}} c_{j_1} \cdots c_{j_i} \, \beta_{j_1} \beta_{j_2} \cdots \beta_{j_i} \cdot \prod_{1 \le v < u \le i} (\beta_{j_u} - \beta_{j_v})^2. \end{aligned} \tag{9}$$

Now let the terms $\beta_1 \succ \beta_2 \succ \cdots \succ \beta_t$ be ordered lexicographically. The summand

$$c_1 \cdots c_i \, \beta_1 \beta_2 \cdots \beta_i \prod_{1 \le v < u \le i} (\beta_v - \beta_u)^2$$

has the term $\beta_1^{2i-1} \beta_2^{2i-3} \cdots \beta_i$ which occurs nowhere else,[2] and $\det \mathcal{A}_i$ does not vanish symbolically. □

We make the transition from symbolic $x_1, \dots, x_n$ to random field elements $p_1, \dots, p_n$ in the customary fashion via the the Schwartz-Zippel lemma (Zippel 1979a; Schwartz 1980; DeMillo and Lipton 1978).

[2] In this argument we make use of the shift by 1 element. We do not know if shifting is needed if one were to exclude the first discrepancy from the termination test.

Theorem 5 *If $p_1, \ldots, p_n$ are chosen randomly and uniformly from a subset S of the domain, which is assumed to be an integral domain, then for the sequence $\{a_i\}_{i\geq 1}$, where $a_i = f(p_1^i, \ldots, p_n^i)$, the Berlekamp/Massey algorithm encounters a singular Hankel matrix (and the corresponding zero discrepancy) the first time at $N = 2t + 1$ with probability no less than*

$$1 - \frac{t(t+1)(2t+1)\deg(f)}{6 \cdot \#(S)},$$

where $\#(S)$ is the number of elements in S.

PROOF. By (9), $\deg(\det \mathcal{A}_i) \leq i^2 \deg(f)$. We have to avoid all possible zeroes in $\prod_{i=1}^{t} \det \mathcal{A}_i$, whose degree is no more than $t(t+1)(2t+1)\deg(f)/6$. The estimate of the probability follows from Lemma 1 in (Schwartz 1980). □

The estimate in Theorem 5 is, like the Zippel-Schwartz estimate, somewhat pessimistic. Consider the following argument. Over a finite field of q elements we may choose the set S to be the entire field, that is, $q = \#(S)$. If we assume that $a_i = f(p_1^i, \ldots, p_n^i)$ are randomly uniformly distributed, the probability that

$$0 \neq (\det(\mathcal{A}_1) \cdots \det(\mathcal{A}_t))_{\alpha_1 \leftarrow a_1, \ldots, \alpha_{2t-1} \leftarrow a_{2t-1}}$$

is exactly $(1 - 1/q)^t \geq 1 - t/q$; cf. (Kaltofen and Lobo 1996); the proof is by induction on i, viewing $\det \mathcal{A}_{i+1}$ as a linear polynomial in α_{2i+1} whose coefficient is $\det \mathcal{A}_i$. Even then, the probability of premature false termination can become unacceptably high, especially when q is small. In our implementation enhanced with thresholds, the user can supply an integer $\zeta \geq 1$, and the early termination is triggered after a singular Hankel matrix occurs ζ times in a row. The precise analysis is complicated and governed by the conditional probabilities $P(\det(A_{i+1}) = 0 \mid \det(A_i) = 0)$ for $i \geq 1$.

Algorithm: `Early Termination Ben-Or/Tiwari`

```
Input:   ▸ $f(x_1, \ldots, x_n)$: a multivariate black box polynomial.
         ▸ $\zeta$: a positive integer, the threshold for early termination.
Output:  ▸ $c_j$ and $\beta_j$: $f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j \beta_j$ with high probability.
         ▸ Or an error message: if the procedure fails to complete.
(1)  [The early termination within the Berlekamp/Massey algorithm.]
     Pick random elements: $p_1, \ldots, p_n \notin \{0, 1\}$.
     For $i = 1, 2, \ldots$
         Perform the Berlekamp/Massey algorithm on $\{f(p_1^i, \ldots, p_n^i)\}_{i\geq 1}$.
         If Hankel matrix singularity happens $\zeta$ many times in a row, then
             break out of the loop;
(2)  [Determine $\beta_j$.]
```

```
    Compute all the roots $b_j$ of $\Lambda(z)$ in the domain of $p_1, \ldots, p_n$.
    If $\Lambda(z)$ does not completely factor, or not all the roots are distinct, then
        the early termination was false.
    Otherwise, determine $\beta_j$: repeatedly divide the roots $b_j$ by $p_1, \ldots, p_n$.
        Again, this might fail for unlucky $p_i$.
(3) [Determine $c_j$.]
    Solve a transposed Vandermonde system.
```

Remark: If the coefficient field is a subfield of real numbers and $c_i > 0$ for all i, no randomization is necessary. The following argument is standard for the least squares problem with a weighted inner product:

$$B_i C_t B_i^{\mathrm{Tr}} y = 0 \Longrightarrow y^{\mathrm{Tr}} B_i C_t B_i^{\mathrm{Tr}} y = 0 \Longrightarrow (B_i^{\mathrm{Tr}} y)^{\mathrm{Tr}} C_t (B_i^{\mathrm{Tr}} y) = 0 \Longrightarrow B_i^{\mathrm{Tr}} y = 0,$$

because $0 = z^{\mathrm{Tr}} C_t z = \sum c_j z_j^2 \Longrightarrow z = 0$. Therefore $y = 0$, and $B_i C_t B_i^{\mathrm{Tr}}$ is non-singular.

3 Early Termination in Non-Standard Bases

As generalizations of the Ben-Or/Tiwari algorithm in the univariate case, Lakshman Y. N. and Saunders (1995) gave sparse algorithms in the Pochhammer and Chebyshev bases. We present the early termination versions of these algorithms.

3.1 Univariate sparse interpolations in the Pochhammer basis

The Pochhammer symbol, $x^{\overline{n}} = x(x+1)\cdots(x+n-1)$, is defined for any integer $n \geq 0$; a polynomial $f(x)$ is represented in the Pochhammer basis as

$$f(x) = \sum_{j=1}^{t} c_j x^{\overline{e_j}}, 0 \leq e_1 < e_2 < \cdots < e_t \text{ and } c_j \neq 0 \text{ for } 1 \leq j \leq t.$$

Let $f^{(k)}(x) = \sum_{j=1}^{t} e_j^k c_j x^{\overline{e_j}}$ for $k \geq 0$ and define the finite difference operator $\Delta(f(x)) = f(x+1) - f(x)$. Then $\Delta(x^{\overline{k}}) = (x+1)^{\overline{k}} - x^{\overline{k}} = k(x+1)^{\overline{k-1}}$ and

$$x \cdot \Delta(f^{(k)}(x)) = f^{(k+1)}(x). \tag{10}$$

For $0 \leq k \leq 2t-1$, $f^{(k)}(p)$ can be computed by applying the recurrence in (10) to the subsequent evaluations $f(p+k)$. Lemma 1 in (Lakshman Y.

N. and Saunders 1995) shows the finite sequence $\{f^{(k)}(p)\}_{2t-1\geq k\geq 0}$ is linearly generated by

$$\Lambda(z) = \prod_{j=1}^{t}(z - e_j) = \lambda_t z^t + \lambda_{t-1}z^{t-1} + \cdots + \lambda_0 \text{ and } \lambda_t = 1. \tag{11}$$

Theorem 1 in (Dress and Grabmeier 1991) shows that for any $p > 0$,

$$\begin{bmatrix} f^{(0)}(p) & f^{(1)}(p) & \dots & f^{(t-1)}(p) \\ f^{(1)}(p) & f^{(2)}(p) & \dots & f^{(t)}(p) \\ \vdots & \vdots & \ddots & \vdots \\ f^{(t-1)}(p) & f^{(t)}(p) & \dots & f^{(2t-2)}(p) \end{bmatrix} \text{is non-singular.}$$

Algorithm: `Sparse Interpolation <Pochhammer Basis>` (Lakshman Y. N. and Saunders 1995)

Input: ▸ $f(x)$: a univariate black box polynomial.
▸ t: the number of non-zero terms of f in the Pochhammer basis.
Output:▸ c_j and e_j: $f(x) = \sum_{j=1}^{t} c_j x^{\overline{e}_j}$.
(1) [The Berlekamp/Massey algorithm.]
Compute $f^{(k)}(p)$ from $f(p+k)$, where $0 \leq k \leq 2t-1$ and $p > 0$.
Determine $\Lambda(z)$ from $\{f^{(k)}(p)\}_{0\leq k\leq 2t-1}$.
(2) [Determine e_j.]
Find all t distinct roots e_j of $\Lambda(z)$.
(3) [Compute the coefficients c_j.]
Solve a transposed Vandermonde system to obtain $c_j p^{\overline{e}_j}$.
Compute c_j from $c_j p^{\overline{e}_j}$ since both p and e_j are known.

3.2 Early termination of sparse interpolations in the Pochhammer basis

The sparse algorithm in the Pochhammer basis (Lakshman Y. N. and Saunders 1995) requires an input t as the number of Pochhammer terms in $f(x)$. To apply the early termination, we need to show that in addition to $\{f^{(k)}(p)\}_{2t-1\geq k\geq 0}$, $\Lambda(z)$ in (11) generates the entire $\{f^{(k)}(p)\}_{k\geq 0}$.

Theorem 6 *For any $p > 0$, $\Lambda(z)$ generates $\{f^{(k)}(p)\}_{k\geq 0}$.*

PROOF. From Lemma 1 in (Lakshman Y. N. and Saunders 1995), we have

$$\sum_{j=0}^{t} \lambda_j f^{(j+k)}(p) = 0 \quad \text{and} \quad \sum_{j=0}^{t} \lambda_j f^{(j+k)}(p+1) = 0 \text{ for } k = 0, \dots, t-1.$$

By (10) and the induction on k,

$$
\begin{aligned}
&p\cdot\Big(\sum_{j=0}^{t}\lambda_j f^{(j+k)}(p+1) - \sum_{j=0}^{t}\lambda_j f^{(j+k)}(p)\Big)\\
&\quad= \sum_{j=0}^{t}\lambda_j\cdot p\cdot\Big(f^{(j+k)}(p+1) - f^{(j+k)}(p)\Big) = \sum_{j=0}^{t}\lambda_j f^{(j+k+1)}(p) = 0. \quad \square
\end{aligned}
$$

To show that $\Lambda(z)$ is the minimal polynomial of $\{f^{(k)}(p)\}_{k\geq 0}$, we need to consider the following $k\times k$ Hankel matrix in variable x:

$$
\mathcal{A}_k = \begin{bmatrix}
f^{(0)}(x) & f^{(1)}(x) & \ldots & f^{(k-1)}(x)\\
f^{(1)}(x) & f^{(2)}(x) & \ldots & f^{(k)}(x)\\
\vdots & \vdots & \ddots & \vdots\\
f^{(k-1)}(x) & f^{(k)}(x) & \ldots & f^{(2k-2)}(x)
\end{bmatrix}.
$$

Theorem 7 *The determinant of $\mathcal{A}_k$ is nonzero for $k = 1,\ldots,t$.*

PROOF. The following factorization can be verified by matrix multiplications:

$$
\begin{aligned}
\mathcal{A}_k &= \begin{bmatrix}
1 & 1 & \ldots & 1\\
e_1^1 & e_2^1 & \ldots & e_t^1\\
\vdots & \vdots & \ddots & \vdots\\
e_1^{k-1} & e_2^{k-1} & \ldots & e_t^{k-1}
\end{bmatrix}
\begin{bmatrix}
c_1 x^{\overline{e}_1} & 0 & \ldots & 0\\
0 & c_2 x^{\overline{e}_2} & \ldots & 0\\
\vdots & \vdots & \ddots & \vdots\\
0 & 0 & \ldots & c_t x^{\overline{e}_t}
\end{bmatrix}
\begin{bmatrix}
1 & e_1^1 & \ldots & e_1^{k-1}\\
1 & e_2^1 & \ldots & e_2^{k-1}\\
\vdots & \vdots & \ddots & \vdots\\
1 & e_t^1 & \ldots & e_t^{k-1}
\end{bmatrix}\\
&= \mathcal{B}_k C_t \mathcal{B}_k^{\mathrm{Tr}}.
\end{aligned}
$$

Apply the Binet-Cauchy formula (Gantmacher 1977) in (8) with $K = \{1,\ldots,k\}$:

$$
\begin{aligned}
\det\mathcal{A}_k &= \sum_J\sum_L (\mathcal{B}_k)_{K,J}(C_t)_{J,L}(\mathcal{B}_k^{\mathrm{Tr}})_{L,K} = \sum_J (\mathcal{B}_k)_{K,J}(C_t)_{J,J}(\mathcal{B}_k^{\mathrm{Tr}})_{J,K}\\
&= \sum_{J=\{j_1,\ldots,j_k\}} c_{j_1}\cdots c_{j_k}\, x^{\overline{e}_{j_1}} x^{\overline{e}_{j_2}}\cdots x^{\overline{e}_{j_k}}\cdot\det\left(\begin{bmatrix}
1 & 1 & \ldots & 1\\
e_{j_1}^1 & e_{j_2}^1 & \ldots & e_{j_k}^1\\
\vdots & \vdots & \ddots & \vdots\\
e_{j_1}^{k-1} & e_{j_2}^{k-1} & \ldots & e_{j_k}^{k-1}
\end{bmatrix}\right)^2\\
&= \sum_{J=\{j_1,\ldots,j_k\}} c_{j_1}\cdots c_{j_k}\, x^{\overline{e}_{j_1}} x^{\overline{e}_{j_2}}\cdots x^{\overline{e}_{j_k}}\cdot\prod_{1\leq v<u\leq k}(e_{j_u} - e_{j_v})^2. \qquad (12)
\end{aligned}
$$

The highest order term $c_t \cdots c_{t-k+1}\ x^{\overline{e_t}} \cdots x^{\overline{e}_{t-k+1}} \prod_{1 \le v < u \le k}(e_u - e_v)^2$ in (12) appears only once, and $\det \mathcal{A}_k$ does not vanish for $1 \le k \le t$. □

Lemma 8 $\det \mathcal{A}_k = 0$ *if and only if* $k > t$.

PROOF. For $k = 1, \ldots, t$, $\det \mathcal{A}_k \neq 0$, therefore $\det \mathcal{A}_k = 0$ implies $k > t$. Because $\Lambda(z)$ generates $\{f^{(k)}(x)\}_{k \ge 0}$, when $k > t$, the k-th row of $\mathcal{A}_k$ is a linear combination of $(k-t)$-th through $(k-1)$-th rows, and $\det \mathcal{A}_k = 0$. □

We now conclude for a random $p > 0$, with high probability the Berlekamp/ Massey algorithm on $\{f^{(k)}(p)\}_{k \ge 0}$ encounters the first singular Hankel matrix when $N = 2t + 1$.

Theorem 9 *Let S be a subset of the domain, which is assumed to be an integral domain, and that all elements of S are positive. Consider $f^{(k)}(x) = \sum_{j=1}^{t} e_j^k c_j x^{\overline{e}_j}$ for $f(x) = \sum_{j=1}^{t} c_j x^{\overline{e}_j}$. If p is chosen randomly and uniformly from S, then for $\{f^{(k)}(p)\}_{k \ge 0}$, the Berlekamp/Massey algorithm encounters a singular Hankel matrix the first time at $N = 2t + 1$ with probability no less than*

$$1 - \frac{t(t+1)(3 \deg f + 1 - t)}{6 \cdot \#(S)},$$

where $\#(S)$ is the number of elements in S.

PROOF. From (12), we have $\deg(\det \mathcal{A}_k) \le \sum_{j=0}^{k-1} \Big(\deg f - j\Big) = k \deg f + k/2 - k^2/2$. We need to avoid all possible zeroes in $\prod_{k=1}^{t} \det \mathcal{A}_k$, whose degree is no more than $\sum_{k=1}^{t}(k \deg f + k/2 - k^2/2)$. □

A higher threshold $\zeta > 1$ can also be introduced as the early termination is triggered after Hankel matrix singularity occurs ζ times in a row. The analysis of probability with higher thresholds requires further investigations on $P(\det(A_{k+1}) = 0 \mid \det(A_k) = 0)$, where A_k are $\mathcal{A}_k$ evaluated at $x = p$.

Algorithm: `Early Termination Sparse Interpolation <Pochhammer Basis>`

```
Input:  ▸ $f(x)$: a univariate black box polynomial.
        ▸ $\zeta$: a positive integer, the threshold for early termination.
Output: ▸ $c_j$ and $e_j$: $f(x) = \sum_{j=1}^{t} c_j x^{\overline{e}_j}$ with high probability.
        ▸ Or an error message: if the procedure fails to complete.
(1) [The early termination within the Berlekamp/Massey algorithm.]
    Pick a random positive value $p > 0$.
    For $i = 1, 2, \ldots$
```

```
          Perform the Berlekamp/Massey algorithm on $\{f^{(i)}(p)\}_{i\geq 0}$;
          $f^{(i)}(p)$ are computed from $f(p+k)$ for $k = 0, \dots, i-1$.
          If Hankel matrix singularity happens $\zeta$ times in a row,
              then break out of the loop;
(2) [Determine $e_j$.]
      Compute all the roots of $\Lambda(z)$.
      If not that all roots are distinct non-negative integers, then
          the early termination was false;
      else, the roots are $e_j$, the Pochhammer exponents in $f(x)$.
(3) [Compute the coefficients $c_j$.]
      Solve a transposed Vandermonde system to obtain $c_j p^{\overline{e}_j}$.
      Compute $c_j$ from $c_j p^{\overline{e}_j}$ since both $p$ and $e_j$ are known.
```

3.3 Univariate sparse interpolations in the Chebyshev basis

Let $T_i(x)$ denote the i-th Chebyshev polynomial of the first kind: $T_0(x) = 1$, $T_1(x) = x$, $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$ for $i \geq 2$. A polynomial $f(x)$ over a field $\mathbb{K}$ is represented in the Chebyshev basis if $c_j \neq 0$ and

$$f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x), 0 \leq \delta_1 < \delta_2 < \cdots < \delta_t.$$

Let $a_k = f(T_k(p))$ for some $p > 1$ and $0 \leq k \leq 2t-1$, consider:

$$\Lambda(z) = \prod_{j=1}^{t}(z - T_{\delta_i}(p)) = \lambda_t T_t(z) + \lambda_{t-1}T_{t-1}(z) + \cdots + \lambda_0 T_0(z) \text{ with } \lambda_t = 1.$$

Lakshman Y. N. and Saunders (1995) showed that for $i \geq 0$:

$$\sum_{j=0}^{t-1} \lambda_j (a_{j+i} + a_{|j-i|}) = -(a_{t+i} + a_{|t-i|}). \tag{13}$$

And the linear relations in (13) form the following system:

$$\begin{bmatrix} 2a_0 & 2a_1 & \dots & 2a_{t-1} \\ 2a_1 & a_2 + a_0 & \dots & a_t + a_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{t-1} & a_t + a_{t-2} & \dots & a_{2t-2} + a_0 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} 2a_t \\ a_{t+1} + a_{t-1} \\ \vdots \\ a_{2t-1} + a_1 \end{bmatrix}. \tag{14}$$

Consider the $t \times t$ symmetric Hankel-plus-Toeplitz matrix A_t in (14):

$$A_t = \begin{bmatrix} 2a_0 & 2a_1 & \dots & 2a_{t-1} \\ 2a_1 & a_2 + a_0 & \dots & a_t + a_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{t-1} & a_t + a_{t-2} & \dots & a_{2t-2} + a_0 \end{bmatrix}. \qquad (15)$$

The sparse interpolation in the Chebyshev basis follows from the fact that A_t is non-singular (Lakshman Y. N. and Saunders 1995, see Lemma 6).

```
Algorithm: Sparse Interpolation <Chebyshev Basis> (Lakshman Y. N. and Saunders 1995)
Input:  ▸ $f(x)$: a univariate black box polynomial.
        ▸ $t$: the number of non-zero terms of $f$ in the Chebyshev basis.
Output: ▸ $c_j$ and $\delta_j$: $f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x)$.
(1) [Solve the symmetric Hankel-plus-Toeplitz system in (14).]
    $p > 1$, $a_k = f(T_k(p))$ for $k = 0, 1, \dots, 2t-1$.
    Determine $\Lambda(z)$: $\lambda_t = 1$, for $0 \le j \le t-1$, $\lambda_j$ are obtained by solving (14).
(2) [Determine $\delta_j$.]
    Find all $t$ distinct roots of $\Lambda(z)$, which are $T_{\delta_j}(p)$.
    Determine the Chebyshev exponents $\delta_j$ from $T_{\delta_j}(p)$.
(3) [Compute the coefficients $c_j$.]
    Solve a transposed Vandermonde-like system to obtain $c_j$ (Lakshman Y. N. and Saunders 1995, the discussion on pp. 394–396).
```

Remark: By showing A_t non-singular, Lakshman Y. N. and Saunders (1995) assured the solution to (14). Unfortunately, in general a non-singular A_t alone does not guarantee that a solution can be computed in $O(t^2)$ deterministic field operations.

3.4 Early termination of sparse interpolations in the Chebyshev basis

Gohberg and Koltracht (1989) gave an $O(t^2)$ algorithm for solving a $t \times t$ symmetric Hankel-plus-Toeplitz system with all principal leading submatrices non-singular. In general this is not true for A_t in (14): for any $f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x)$ with $c_1 + \dots + c_t = 0$, $A_1 = \begin{bmatrix} 2a_0 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$ for all $p > 1$. We fix this problem through randomization: whenever $f(T_0(p)) = 0$ for any $p > 1$, pick a suitable $p_c \neq 0$ and interpolate $\tilde{f}(x) = f(x) + p_c$ instead. As a result, we always start with a 1×1 non-singular leading submatrix. After $\tilde{f}(x)$ is interpolated, $f(x)$

can be recovered by removing p_c from $\tilde{f}(x)$.

This random p_c further provides all principal leading submatrices of A_t non-singular with high probability. Suppose $\tilde{f}(x) = \sum_{j=1}^{\tilde{t}} \tilde{c}_j T_{\delta_j}(x)$ with $0 \le \delta_1 < \delta_2 < \cdots < \delta_{\tilde{t}}$, whose constant has already been "randomized." Let y represent the random component in the constant, namely, $\sum_{j=1}^{t} c_j + y = \sum_{j=1}^{\tilde{t}} \tilde{c}_j$. Consider $\alpha_k = \tilde{f}(T_k(x))$ and the $i \times i$ symmetric Hankel-plus-Toeplitz systems:

$$\mathcal{A}_i = \begin{bmatrix} 2\alpha_0 & 2\alpha_1 & \dots & 2\alpha_{i-1} \\ 2\alpha_1 & \alpha_2 + \alpha_0 & \dots & \alpha_i + \alpha_{i-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2\alpha_{i-1} & \alpha_i + \alpha_{i-2} & \dots & \alpha_{2i-2} + \alpha_0 \end{bmatrix}. \tag{16}$$

The entries of $\mathcal{A}_i$ are polynomials in x and y. Our purpose is to prove that $\mathcal{A}_i$ is non-singular for $1 \le i \le \tilde{t}$ in x and y symbolically, and that $\mathcal{A}_{\tilde{t}+1}$ is singular. The singularity of $\mathcal{A}_i$ for $i > \tilde{t}+1$ is concluded from Lemma 5 in (Lakshman Y. N. and Saunders 1995).

Based on the proof of Lemma 6 in (Lakshman Y. N. and Saunders 1995), $\mathcal{A}_i = \mathcal{V}_i C \mathcal{V}_i^{\mathrm{Tr}}$ for $1 \le i \le \tilde{t}$, where

$$\mathcal{V}_i = \begin{bmatrix} T_{\delta_1}(T_0(x)) & T_{\delta_2}(T_0(x)) & \dots & T_{\delta_{\tilde{t}}}(T_0(x)) \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1}(T_{i-1}(x)) & T_{\delta_2}(T_{i-1}(x)) & \dots & T_{\delta_{\tilde{t}}}(T_{i-1}(x)) \end{bmatrix} \text{ and}$$

$$C = \begin{bmatrix} 2\tilde{c}_1 & 0 & \dots & 0 \\ 0 & 2\tilde{c}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 2\tilde{c}_{\tilde{t}} \end{bmatrix}.$$

The Chebyshev polynomials commute with respect to composition: for $m, n \ge 0$, $T_n(T_m(x)) = T_{mn}(x) = T_m(T_n(x))$, and

$$\mathcal{V}_i = \begin{bmatrix} T_{\delta_1 0}(x) & T_{\delta_2 0}(x) & \dots & T_{\delta_{\tilde{t}} 0}(x) \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1 (i-1)}(x) & T_{\delta_2 (i-1)}(x) & \dots & T_{\delta_{\tilde{t}} (i-1)}(x) \end{bmatrix}. \tag{17}$$

Lemma 10 *For $n \ge 1$, $T_{n\delta}(x) = \sum_{i=0}^{n} \gamma_{n,i} T_\delta(x)^i$ and $\gamma_{n,n} = 2^{n-1}$.*

PROOF. When $n = 1, 2$, the above statement is true.

Suppose the statement is true for all $n \leq k$, by induction consider $n = k + 1$:

$$\begin{aligned} T_{(k+1)\delta}(x) &= 2T_\delta(x)T_{k\delta}(x) - T_{(k-1)\delta}(x) \\ &= 2T_\delta(x)\Big(2^{k-1}T_\delta(x)^k + \sum_{i=0}^{k-1} \gamma_{k,i}T_\delta(x)^i\Big) - \sum_{i=0}^{k-1} \gamma_{k-1,i}T_\delta(x)^i \\ &= 2^k T_\delta(x)^{k+1} + \sum_{i=0}^{k} \gamma_{k+1,i}T_\delta(x)^i = \sum_{i=0}^{k+1} \gamma_{k+1,i}T_\delta(x)^i. \quad \square \end{aligned}$$

By Lemma 10, $\mathcal{V}_i$ in (17) can be factorized as:

$$\mathcal{V}_i = \begin{bmatrix} 1\ 0\ 0 \ldots & 0 \\ 0\ 1\ 0 \ldots & \vdots \\ *\ *\ 2 \ldots & \vdots \\ \vdots\ \vdots\ \vdots\ \ddots & \vdots \\ *\ *\ * \ldots & 2^{i-2} \end{bmatrix} \begin{bmatrix} T_{\delta_1}(x)^0 & T_{\delta_2}(x)^0 & \ldots & T_{\delta_{\tilde{t}}}(x)^0 \\ T_{\delta_1}(x)^1 & T_{\delta_2}(x)^1 & \ldots & T_{\delta_{\tilde{t}}}(x)^1 \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1}(x)^{i-1} & T_{\delta_2}(x)^{i-1} & \ldots & T_{\delta_{\tilde{t}}}(x)^{i-1} \end{bmatrix} = \mathcal{L}_i\mathcal{B}_i.$$

For $1 \leq i \leq \tilde{t}$,

$$\mathcal{A}_i = \mathcal{V}_i C \mathcal{V}_i^{\mathrm{Tr}} = \mathcal{L}_i\mathcal{B}_i C(\mathcal{L}_i\mathcal{B}_i)^{\mathrm{Tr}} = \mathcal{L}_i(\mathcal{B}_i C\mathcal{B}_i^{\mathrm{Tr}})\mathcal{L}_i^{\mathrm{Tr}}. \tag{18}$$

Theorem 11 *The determinant of $\mathcal{A}_i$ is non-zero for $1 \leq i \leq \tilde{t}$.*

PROOF. When $i = \tilde{t}$, this is Lemma 6 in (Lakshman Y. N. and Saunders 1995). Now consider $1 \leq i < \tilde{t}$, we have $\det \mathcal{A}_i = \det \mathcal{L}_i \det(\mathcal{B}_i C\mathcal{B}_i^{\mathrm{Tr}}) \det \mathcal{L}_i^{\mathrm{Tr}}$. Assume $\det(\mathcal{B}_i C\mathcal{B}_i^{\mathrm{Tr}}) = 0$ for some i, which implies all terms in $\det(\mathcal{B}_i C\mathcal{B}_i^{\mathrm{Tr}})$ are zero. In other word, for every ordered list I from $\{1, 2, \ldots, \tilde{t}\}$ such that $\#(I) = i - 1$ and $j_k \in I$ with index $k \in \{1, \ldots, i - 1\}$, the coefficient of term $\prod_{j_k \in I} T_{\delta_{j_k}}(x)^{2k}$ is zero, that is, $2^i \prod_{j_k \in I} \tilde{c}_{j_k} \sum_{j \in \{1,2,\ldots,\tilde{t}\}-I} \tilde{c}_j = 0$. Knowing that both 2^i and $\prod_{j_k \in I} \tilde{c}_{j_k}$ are non-zero, it must be $\sum_{j \in \{1,2,\ldots,\tilde{t}\}-I} \tilde{c}_j = 0$. Adding up all such sums, $\sum_{\#(I)=i-1} \sum_{j \in \{1,2,\ldots,\tilde{t}\}-I} \tilde{c}_j = (\tilde{t}-1)(\tilde{t}-2)\cdots(\tilde{t}-i+1) \sum_{j=1}^{\tilde{t}} \tilde{c}_j = 0$ implies $\sum_{j=1}^{t} \tilde{c}_j = y + \sum_{j=1}^{t} c_j = 0$, which is a contradiction since y cannot be cancelled in $y + \sum_{j=1}^{t} c_j$ symbolically. Hence, $\det(\mathcal{B}_i C\mathcal{B}_i^{\mathrm{Tr}}) \neq 0$. From the non-zero diagonals, $\det \mathcal{L}_i^{\mathrm{Tr}} = \det \mathcal{L}_i \neq 0$, and $\det \mathcal{A}_i \neq 0$ is concluded. $\square$

The original algorithm of Gohberg and Koltracht (1989) solves $Ax = c$ when A and c are given. In our application, we solve for (13), which is the solution $\underline{\lambda} = [\lambda_0, \ldots, \lambda_{\tilde{t}}]^{\mathrm{Tr}}$ to the first singular system such that $\lambda_{\tilde{t}} = 1$.

We denote the entry at i-th row and j-th column in $A_{\tilde{t}+1}$ as $\tilde{a}_{i,j}$, and underline the vector variables to distinguish them from their indexed components, for example $\underline{\gamma} = [\gamma_1, \ldots, \gamma_i]^{\mathrm{Tr}}$. The $i \times i$ identity matrix is I_i, and L_i is the $i \times i$ matrix defined as:

$$L_i = \begin{bmatrix} 0 & \dots & \dots & 0 \\ 1 & 0 & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix}.$$

Algorithm: `Modified Gohberg/Koltracht`

Input: ▸ h_k and t_k, $k \in \mathbb{Z}_{\geq 0}$: $\tilde{a}_{i,j} = h_{i+j-2} + t_{|i-j|}$ define the entries in the given symmetric Hankel-plus-Toeplitz system and that $\tilde{a}_{1,1} \neq 0$.

Output: ▸ $\underline{\lambda} = [\lambda_0, \ldots, \lambda_t]^{\mathrm{Tr}}$: $A_{t+1}\underline{\lambda} = 0$ with $\lambda_t = 1$, where $t \geq 1$ is the smallest integer such that the symmetric Hankel-plus-Toeplitz system A_{t+1} is singular.

```
(1) [With $\tilde{a}_{1,1} \neq 0$, $\Delta$ reflects the singularity of $A_2$. If $\Delta = 0$, return $\underline{\lambda}$ such
    that $A_2\underline{\lambda} = 0$ and $\lambda_1 = 1$; otherwise, proceed with the initialization.]
    $\Delta \leftarrow \tilde{a}_{1,1}\tilde{a}_{2,2} - \tilde{a}_{1,2}\tilde{a}_{2,1}$;
    If $\Delta = 0$ then
        Return $\underline{\lambda} = [-\tilde{a}_{1,2}/\tilde{a}_{1,1}, 1]^{\mathrm{Tr}}$;
    Else
        $i \leftarrow 1$; $\underline{\gamma} \leftarrow [1/\tilde{a}_{1,1}]$; $\underline{\psi} \leftarrow [1/\tilde{a}_{1,1}]$; $\underline{\phi} \leftarrow [t_1/\tilde{a}_{1,1}]$; $\alpha \leftarrow \tilde{a}_{2,1}/\tilde{a}_{1,1}$;
        $\underline{\gamma}^{\mathrm{new}} \leftarrow (1/\Delta)[-\tilde{a}_{1,2}, \tilde{a}_{1,1}]^{\mathrm{Tr}}$;
(2) [Increase $i$, if $\Delta \neq 0$, follow (Gohberg and Koltracht 1989) to update $\underline{\gamma}^{\mathrm{new}}$.
    If $\Delta = 0$, then $A_{i+1}$ is singular (see Theorem 13). We assign $\lambda_t = 1$ and
    update the rest of $\underline{\lambda}$ so that $A_{i+1}\underline{\lambda} = 0$.]
    While $\Delta \neq 0$ do
        $i \leftarrow i + 1$; $\kappa \leftarrow (t_i + h_{i-2}) - \sum_{j=1}^{i-1} \tilde{a}_{i,j}\phi_j$; $\mu \leftarrow -\sum_{j=1}^{i-1} \tilde{a}_{i,j}\psi_j$;
        $\underline{\phi}^{\mathrm{new}} \leftarrow [\underline{\phi}^{\mathrm{Tr}}, 0]^{\mathrm{Tr}} + \kappa\underline{\gamma}^{\mathrm{new}}$; $\underline{\psi}^{\mathrm{new}} \leftarrow [\underline{\psi}^{\mathrm{Tr}}, 0]^{\mathrm{Tr}} + \mu\underline{\gamma}^{\mathrm{new}}$;
        $\alpha^{\mathrm{new}} \leftarrow \sum_{j=1}^{i} \tilde{a}_{i+1,j}\gamma_j^{\mathrm{new}}$;
        $\underline{b} \leftarrow \Big((\alpha - \alpha^{\mathrm{new}})I_i + L_i + L_i^{\mathrm{Tr}}\Big)\underline{\gamma}^{\mathrm{new}} - [\underline{\gamma}^{\mathrm{Tr}}, 0]^{\mathrm{Tr}} + \psi_i^{\mathrm{new}}\underline{\phi}^{\mathrm{new}} - \phi_i^{\mathrm{new}}\underline{\psi}^{\mathrm{new}}$;
        $\nu \leftarrow \Big(\gamma_i^{\mathrm{new}}\Big)^{-1} \sum_{j=1}^{i} \tilde{a}_{i+1,j}b_j$; $\Delta \leftarrow \nu + \tilde{a}_{i+1,i+1}$;
        If $\Delta = 0$ then
            $\lambda_i \leftarrow 1$; For $j = 0 \ldots i-1$ do $\lambda_j \leftarrow b_{j+1}/\gamma_i^{\mathrm{new}}$;
            Return $\underline{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_i]^{\mathrm{Tr}}$;
        Else [At the end of Step (2), update variables for next $i$.]
            $\underline{\gamma} \leftarrow \underline{\gamma}^{\mathrm{new}}$; $\gamma_{i+1}^{\mathrm{new}} \leftarrow 1/\Delta$; $\underline{\phi} \leftarrow \underline{\phi}^{\mathrm{new}}$; $\underline{\psi} \leftarrow \underline{\psi}^{\mathrm{new}}$; $\alpha \leftarrow \alpha^{\mathrm{new}}$;
            For $j = 1, \ldots, i$ do
                $\gamma_j^{\mathrm{new}} \leftarrow \Big(\gamma_{i+1}^{\mathrm{new}}/\gamma_i\Big)b_j$;
```

Lemma 12 *In the modified Gohberg/Koltracht algorithm, if we encounter $\Delta = 0$ for some $i \geq 1$, then at the end of Step (2), we have*

$$A_{i+1}\underline{\lambda} = [0, \cdots, 0]^{Tr}.$$

PROOF. If $\Delta = 0$, $\lambda_i = 1$ and $\lambda_j = b_j/\gamma_{i+1}^{\text{new}}$ for $0 \leq j \leq i-1$. The matrix multiplication of the $(i+1)$-th row in A_{i+1} and $\underline{\lambda}$ is

$$[\tilde{a}_{i+1,1}, \cdots, \tilde{a}_{i+1,i+1}]\underline{\lambda} = \tilde{a}_{i+1,i+1} + \underbrace{\sum_{j=1}^{i} \tilde{a}_{i+1,j}\left(\frac{b_j}{\gamma_i^{\text{new}}}\right)}_{\nu} = \Delta = 0.$$

When $1 \leq j \leq i$, the matrix multiplications of the j-th row and $\underline{\lambda}$ are all zero due to the definition of $\underline{\lambda}$ (Gohberg and Koltracht 1989, pp. 139–140). □

Theorem 13 *In the modified Gohberg/Koltracht algorithm, for any $i \geq 1$, if $\det(A_j) \neq 0$ for all $1 \leq j \leq i$, then $\Delta = 0$ if and only if $\det(A_{i+1}) = 0$.*

PROOF. If $\det(A_j) \neq 0$ for all $1 \leq j \leq i$ and $\Delta \neq 0$, A_{i+1} can be inverted by the Gohberg/Koltracht algorithm and $\det(A_{i+1}) \neq 0$. To prove another direction: if $\Delta = 0$, from Lemma 12, $\underline{\lambda} \neq 0$ and $A_{i+1}\underline{\lambda} = 0$. As a result, $\det(A_{i+1}) = 0$. □

Now back to the sparse interpolation in the Chebyshev basis. Without an input as $\tilde{t}$, using the discrepancy $\Delta = 0$ as the termination test, the early termination can be implemented in interpolating $\tilde{f}(x)$. Notice that $\deg f = \deg \tilde{f}$.

Theorem 14 *If p is chosen randomly and uniformly from a subset S of the domain, which is assumed to be an integral domain, and that all the elements in S are larger than 1, then for $a_k = \tilde{f}(T_k(p))$ with $k \geq 0$ the following matrix*

$$A_i = \begin{bmatrix} 2a_0 & 2a_1 & \ldots & 2a_{i-1} \\ 2a_1 & a_2 + a_0 & \ldots & a_i + a_{i-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{i-1} & a_i + a_{i-2} & \ldots & a_{2i-2} + a_0 \end{bmatrix}$$

becomes singular the first time at $i = \tilde{t} + 1$ with probability no less than

$$1 - \frac{(\tilde{t}-1)(2\tilde{t}^2 + 5\tilde{t} + 6)\deg \tilde{f}}{6 \cdot \#(S)}.$$

PROOF. From (18) we have $\deg(\det \mathcal{A}_i) \leq i^2 \deg \tilde{f}$. If $\det A_i \neq 0$ until $i = \tilde{t}+1$ with $\det A_1 \neq 0$ provided, we need to avoid hitting a zero of $\prod_{i=2}^{\tilde{t}} \det A_i$, whose degree is no more than $(\tilde{t}-1)(2\tilde{t}^2+5\tilde{t}+6)\deg \tilde{f}/6$. □

Because the modified Gohberg/Koltracht algorithm requires all principal leading matrices non-singular, we cannot directly apply the higher thresholds for early termination here. To exploit the threshold implementation, we refer to the approach of (Delsarte et al. 1985). We can also further check λ_j at additional $k = i, i+1, \ldots$ for $\sum_{j=0}^{i-1} \lambda_j (a_{j+k} + a_{|j-k|}) = -(a_{i+k} + a_{|i-k|})$.

Algorithm: `Early Termination Sparse Interpolation <Chebyshev Basis>`

```
Input:  ▸ $f(x)$: a univariate black box polynomial.
Output: ▸ $c_j$ and $\delta_j$: $f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x)$ with high probability.
        ▸ Or an error message: if the procedure fails to complete.
(1) [The first leading principal submatrix is non-singular.]
    Pick a random element $p > 1$.
    If $a_0 = f(T_0(p)) = 0$ then
        pick a random $\tilde{p}_c \neq 0$; $a_0 \leftarrow \tilde{p}_c$; $f(x) \leftarrow f(x) + p_c$;
    else $p_c = 0$; $f(x) \leftarrow f(x) + p_c$;
(2) [The early termination in the modified Gohberg/Koltracht algorithm.]
    For $i = 1, 2, \ldots$
        Perform the modified Gohberg/Koltracht algorithm on the $i \times i$ matrix
        $[\tilde{a}_{k,l}]$ with $\tilde{a}_{k,l} = a_{k+l-2} + a_{|k-l|}$ and $a_i = f(T_i(p))$.
        If $\Delta = 0$, then returns $\lambda_j$ that define $\Lambda(z)$; break out of the loop.
(3) [Determine $\delta_j$.]
    Compute all the roots of $\Lambda(z)$ in the domain of $p$.
    If $\Lambda(z)$ does not completely factor, or not all the roots are distinct, then
        the early termination was false.
    else determine $\delta_j$ from $T_{\delta_j}(p)$, $T_{\delta_j}(p)$ are the roots of $\Lambda(z)$:
        again, the recovery of $\delta_j$ might fail.
(4) [Compute the coefficients $c_j$.]
    Solve a transposed Vandermonde-like system (Lakshman Y. N. and Saunders 1995, the discussion on pp. 394–396).
    Recover the input $f(x)$ by removing $p_c$ from the result.
```

Remarks: Adding $p_c \neq 0$ to f might introduce one more term to f (the constant), which causes extra overhead in the sparse interpolation algorithm. Nevertheless, we consider such overhead to be minor.

Georg Heinig has pointed out to us that the relation of the entries in the Toeplitz summand with the Hankel part in (15) may allow the use of algorithms for discrete trigonometric transforms, which could yield a speedup over the general Gohberg/Koltracht algorithm.

4 Racing Algorithms and Early Termination

4.1 Early termination in racing algorithms

A dense algorithm evaluates the target polynomial at sufficiently many distinct points; performing a sparse algorithm does not prevent us from simultaneously interpolating the same points through a dense one. Therefore, we propose racing algorithms: for every black box probe, we apply both algorithms; whenever either racer algorithm terminates via early termination, the overall algorithm terminates. Yet in Theorem 1, an early termination dense algorithm interpolates on a sequence $p_0, p_1, \ldots$, where each p_i is randomly generated. While in the case of sparse algorithms the sequence is constructed by a random p: $p, p^2, p^3, \ldots$ in the standard basis; $p, p+1, p+2, \ldots$ in the Pochhammer basis; and $T_0(p), T_1(p), T_2(p), \ldots$ in the Chebyshev basis. We need to show the early termination for a dense algorithm is also true when the evaluation points are constructed by a random p from a sparse algorithm.

Let $\mathbb{Z}_{\geq 0}$ denote the set of non-negative integers, and $\mathbb{K}[p]$ a polynomial ring. Consider a generic basis b_i for $\mathbb{K}[p]$ with $i \in \mathbb{Z}_{\geq 0}$ such that $\deg(b_i) = i$ and $\deg(b_i b_j) = i + j$, and a generic rising factorial power in x for every $n \in \mathbb{Z}_{\geq 0}$: $x^{\{n\}} = (x - b_0)(x - b_1) \cdots (x - b_{n-1})$. By Newton interpolation, a polynomial $f(x)$ with degree n interpolated at $b_0, \ldots, b_n$, a generic basis constructed by p, is represented as

$$f(x) = \sum_{i=0}^{n} \overline{a}_i x^{\{i\}} \quad \text{where } \overline{a}_i \in \mathbb{K}[p]. \tag{19}$$

Our purpose is to show that all $\overline{a}_i$ are non-zero polynomials in $\mathbb{K}[p]$ for $0 \leq i \leq n$ (Theorem 18). Therefore, when p is random, with high probability the first $\overline{a}_i = 0$ occurs at $i = n + 1$ and $f(x)$ is interpolated as (19). Now compare $f(x)$ in the standard basis and a generic rising factorial basis:

$$f(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} \overline{a}_i x^{\{i\}}.$$

The coefficients $c_i^{(n)}$, that depend on b_j, define the transformation from a generic factorial basis to the standard basis:

$$x^n = \sum_{i=0}^{n} c_i^{(n)} x^{\{i\}}.$$

Lemma 15 *For any integer $k > 1$ and every $1 \leq j \leq k-1$,*

$$x \sum_{s=0}^{j} c_s^{(k-1)} x^{\{s\}} = c_j^{(k-1)} x^{\{j+1\}} + b_j c_j^{(k-1)} x^{\{j\}} + x \sum_{s=0}^{j-1} c_s^{(k-1)} x^{\{s\}}.$$

PROOF. Replace x by $(x - b_j + b_j)$ in $x \sum_{s=0}^{j} c_s^{(k-1)} x^{\{s\}}$:

$$\begin{aligned}(x-b_j + b_j) \sum_{s=0}^{j} c_s^{(k-1)} x^{\{s\}} &= (x - b_j) \sum_{s=0}^{j} c_s^{(k-1)} x^{\{s\}} + b_j \sum_{s=0}^{j} c_s^{(k-1)} x^{\{s\}} \\ =& c_j^{(k-1)} x^{\{j\}} (x - b_j) + x \sum_{s=0}^{j-1} c_s^{(k-1)} x^{\{s\}} + b_j \Big(- \sum_{s=0}^{j-1} c_s^{(k-1)} x^{\{s\}} + \sum_{s=0}^{j} c_s^{(k-1)} x^{\{s\}} \Big) \\ =& c_j^{(k-1)} x^{\{j+1\}} + b_j c_j^{(k-1)} x^{\{j\}} + x \sum_{s=0}^{j-1} c_s^{(k-1)} x^{\{s\}}. \quad \square\end{aligned}$$

Theorem 16 *For $n \geq 1$, $c_n^{(n)} = 1$, $c_0^{(n)} = b_0 c_0^{(n-1)}$, and for $n > s > 0$,*

$$c_s^{(n)} = b_s c_s^{(n-1)} + c_{s-1}^{(n-1)}.$$

PROOF. Repeatedly apply Lemma 15 for j from $n-1$ to 1:

$$\begin{aligned}x \cdot x^{n-1} = x \sum_{s=0}^{n-1} c_s^{(n-1)} x^{\{s\}} &= c_{n-1}^{(n-1)} x^{\{n\}} + b_{n-1} c_{n-1}^{(n-1)} x^{\{n-1\}} + x \sum_{s=0}^{n-2} c_s^{(n-1)} x^{\{s\}} \\ &= c_{n-1}^{(n-1)} x^{\{n\}} + \Big(b_{n-1} c_{n-1}^{(n-1)} + c_{n-2}^{(n-1)} \Big) x^{\{n-1\}} + \cdots \\ &\qquad + \Big(b_s c_s^{(n-1)} + c_{s-1}^{(n-1)} \Big) x^{\{s\}} \\ &+ \cdots + \Big(b_1 c_1^{(n-1)} + c_0^{(n-1)} \Big) x^{\{1\}} + b_0 c_0^{(n-1)} x^{\{0\}} = \sum_{s=0}^{n} c_s^{(n)} x^{\{s\}}.\end{aligned}$$

Through comparison, $c_0^{(n)} = b_0 c_0^{(n-1)}$ and $c_s^{(n)} = b_s c_s^{(n-1)} + c_{s-1}^{(n-1)}$ for $0 < s < n$. To prove $1 = c_n^{(n)}$ is easy because $c_n^{(n)} = c_{n-1}^{(n-1)}$ and $c_0^{(0)} = 1$. $\square$

Now we define $c_s^{(n)} = 0$ for all $s > n$ and consider $c_s^{(n)}$, b_j as polynomials in p.

Theorem 17 *For any integer $n > 0$, and any integer s such that $n > s > 0$,*

$$\deg \big(c_s^{(n)}(p)\big) = s \cdot (n - s).$$

PROOF. Repeatedly apply Theorem 16:

$$\begin{aligned}
c_s^{(n)} &= b_s c_s^{(n-1)} + c_{s-1}^{(n-1)} = b_s\Big(b_s c_s^{(n-2)} + c_{s-1}^{(n-2)}\Big) + \Big(b_{s-1} c_{s-1}^{(n-2)} + c_{s-2}^{(n-2)}\Big) \\
&= \Big(b_s\Big)^2 \Big(b_s c_s^{(n-3)} + c_{s-1}^{(n-3)}\Big) + b_s\Big(b_{s-1} c_{s-1}^{(n-3)} + c_{s-2}^{(n-3)}\Big) + \cdots \\
&= \Big(b_s\Big)^{n-s} c_s^{(s)} + \text{lower degree terms in } p.
\end{aligned}$$

Since $\deg(b_i b_j) = i + j$ and $c_s^{(s)} = c_{s-1}^{(s-1)} = 1$, $\deg\big(c_s^{(n)}(p)\big) = s \cdot (n-s)$. □

From Theorem 17, $\deg c_s^{(n+1)} > \deg c_s^{(n)}$ for any integer $n > 0$ and $n > s > 0$.

Theorem 18 *Let $f(x) = \sum_{i=0}^n a_i x^i = \sum_{i=0}^n \overline{a}_i x^{\{i\}}$. If $a_n \neq 0$, $\overline{a}_i$ is a non-zero polynomial in p for $0 < i \leq n$. Moreover, $\overline{a}_n = a_n$.*

PROOF. Expand $f(x) = \sum_{i=0}^n a_i x^i$ and collect the terms with respect to $x^{\{i\}}$:

$$\begin{aligned}
\sum_{i=0}^{n} \Big(a_i \sum_{j=0}^{i} c_j^{(i)} x^{\{j\}}\Big) &= a_n\Big(\sum_{j=0}^{n} c_j^{(n)} x^{\{j\}}\Big) + \cdots + a_1\Big(\sum_{j=0}^{1} c_j^{(1)} x^{\{j\}}\Big) + a_0 x^{\{0\}} \\
=& a_n c_n^{(n)} x^{\{n\}} + \cdots + \Big(a_n c_0^{(n)} + \cdots + a_0 c_0^{(0)}\Big) x^{\{0\}} = \sum_{i=0}^{n}\Big(\sum_{j=0}^{n-i} a_{n-j} c_i^{(n-j)}\Big) x^{\{i\}}.
\end{aligned}$$

Comparing the coefficients, we have $\overline{a}_i = \sum_{j=0}^{n-i} a_{n-j} c_i^{(n-j)}$ for $0 < i \leq n$, and $\overline{a}_n = a_n c_n^{(n)} = a_n$ with $c_n^{(n)} = 1$. The highest degree term in $c_i^{(n)}$ occurs only once in $\overline{a}_i = a_n c_i^{(n)} + \cdots + a_i c_i^{(i)}$ (see Theorem 17) and $a_n \neq 0$, so $\overline{a}_i$ is a non-zero polynomial in p for $0 < i \leq n$. □

Now that $\overline{a}_i$ are non-zero polynomials for $0 < i \leq n = \deg f$ and $\overline{a}_i = 0$ for $i > n$, $\overline{a}_i(p) = 0$ at a random p the first time when $i = n + 1$ with high probability.

Theorem 19 (Newton v.s. Ben-Or/Tiwari) *If p is randomly picked and $p \notin \{0, 1\}$, the early termination of Newton interpolation is true if it interpolates on the sequence $p^1, p^2, \ldots, p^k, \ldots$.*

PROOF. For any non-zero value p_c, let $b_i = p_c \cdot p^i$ and $\deg(b_i) = i$. Now that p is a non-zero random number, assign p_c as p and apply Theorem 18. □

Theorem 20 (Newton v.s. sparse Chebyshev basis interpolation) *If $p > 1$ is randomly picked, the early termination of Newton interpolation is true if it interpolates on the sequence $T_0(p)$, $T_1(p)$, $\ldots$, $T_k(p)$, $\ldots$.*

PROOF. Let $T_i(p) = b_i$ and apply Theorem 18. $\square$

Remark: Our racing algorithms match Newton interpolation against a sparse algorithm on sequence $b_0(p_1)$, $b_1(p_1)$, $\ldots$ constructed by a random p_1. Whenever the sparse racer terminates first, but unsuccessfully, and the Newton interpolation has yet finished, we pick another random p_2 and construct a new sequence $b_0(p_2), b_1(p_2), \ldots$. On this new sequence, we restart the sparse racer but keep updating the existing Newton interpolant (see Subsection 4.2). The early termination of Newton interpolation in the "restarting" phase can be prove by modifying Theorems 19 and 20 by assigning p_i in a lexicographic order: $p_1 \prec p_2 \prec \cdots$.

The sparse interpolation in the Pochhammer basis evaluates a subsequent values $p, p+1, \ldots$. Since $\deg(p) = \deg(p+1) = \cdots$ in $\mathbb{K}[p]$, Theorem 18 cannot be applied in this case.

Theorem 21 (Newton v.s. sparse Pochhammer basis interpolation) *If $p > 0$ is randomly picked, the early termination of Newton interpolation is true if it interpolates on the sequence $p, p+1, p+2, \ldots$.*

PROOF. We want to show the coefficient c_i in the i-th Newton interpolant $f^{[i]}(x) = f^{[i-1]}(x) + c_i(x-p)(x-p-1)\cdots(x-p-i+1)$ is a non-zero polynomial in p for $0 \le i \le \deg f = n$.

If f is a non-zero polynomial, then $c_0 = f(p)$ is a non-zero polynomial in p. Now consider $0 \le i < n$, if for every $0 \le k \le i$, c_k is a non-zero polynomial in p and c_{i+1} is a zero polynomial, then we claim that $f^{[i]} = f^{[i+1]} = \cdots = f^{[n]} = f$. For otherwise, suppose $f^{[j]}$ is the first interpolant being updated since $f^{[i+1]}$, that is, $c_{j-1} = 0$ for all p and

$$f^{[j]} = f^{[i]} + c_j(x-p)(x-p-1)\cdots(x-p-j+1) \tag{20}$$

with $c_j \ne 0$ and $i+1 < j \le n$. We expand the newly updated term in (20) with respect to p shifted by 1 as the following:

$$\begin{aligned} c_j(x-p)\cdots(x-p-j+1) &= c_j(x-p-j+j)(x-p-1)\cdots(x-p-j+1) \\ &= c_j(x-p-1)\cdots(x-p-j) + \underbrace{c_j \cdot j}_{c_{j-1}\ne 0} \cdot (x-p-1)\cdots(x-p-j+1). \end{aligned}$$

Therefore, if f is interpolated at $p-1$, p, $p+1$, $\ldots$, we have $c_{j-1} \ne 0$, which contradicts to the claim that $c_{j-1} = 0$ for all p.

On the other hand, because $\deg f(x) = n > i = \deg f^{[i]}(x)$, c_n cannot be a zero polynomial and we have concluded $c_i \ne 0$ for every $0 \le i \le n$. $\square$

Remark: The argument for early termination of Newton interpolation in the "restarting" phase when racing against the sparse Pochhammer basis interpolation is such that each p_i is randomly generated when a new sequence $p_i, p_i + 1, \dots$ is started (cf. Theorem 1).

4.2 Racing algorithms

We now present our racing algorithms. For a univariate black box polynomial f, pick a random p_1 and construct $b_0(p_1)$, $b_1(p_1)$, $\dots$ as required by the sparse racer algorithm. On this sequence, interpolate f by the early termination versions of both Newton interpolation and the sparse algorithm. Whenever the sparse algorithm successfully terminates earlier, the overall racing algorithm terminates. If the sparse racer fails while the Newton interpolation is yet unfinished, pick another random p_2 and restart the sparse racer on $b_0(p_2), b_1(p_2), \dots$ while continuing the Newton interpolation with the new sequence. Such restarts can be repeated until either of the algorithms terminates.

Although our racing algorithms do not require a bound on either the degree or the number of terms, in our implementation an upper bound δ is requested to confine the overall interpolation efforts (and guard the racing algorithm from an infinite loop, e.g., when the function values do not correspond to a polynomial). To prevent Newton interpolation from aborting too early, δ should be no less than $\deg f(x) + 1 + \eta$, where η is the threshold for Newton interpolation. If we have enough distinct values, in a sparse case the sparse algorithm might terminate earlier, yet it might fail due to the unlucky numbers and not finish at all; Newton interpolation might cost more black box probes, but it always finishes. The overall racing algorithm is superior: it can terminate earlier whenever it is possible while the termination is guaranteed. Also, the probability of correctness can be further improved by cross checking the information acquired from two different algorithms: for example, the sparse result cannot be correct when its degree is smaller than the most updated Newton interpolant.

Algorithm: `Racing <Newton v.s. Sparse>`

Input:
- ▸ $f(x)$: a univariate black box polynomial.
- ▸ δ: a bound for confining the overall interpolation efforts.
- ▸ η: the threshold in Newton interpolation.
- ▸ ζ: the threshold in the sparse racer algorithm.

Output:
- ▸ $\tilde{f}(x)$: with high probability, $\tilde{f}(x) = f(x)$.
- ▸ Or an error message: if the procedure fails.

```
(1) $0 \neq p$ random[3], $a_0 \leftarrow b_0(p)$; $\tilde{a}_0 \leftarrow a_0$; $\text{new}_{\text{[race]}} \leftarrow$ false; $j \leftarrow 0$; $k \leftarrow 0$;
    Initialize Newton interpolant $f_N^{[0]}$ at $a_0$: $f_N^{\{0\}} \leftarrow f_N^{[0]}$;
    Initialize the sparse racer algorithm at $\tilde{a}_0$;
(2) [Interpolate at one more point.]
    For $i = 1, \ldots, \delta$ Do
        If $\text{new}_{\text{[race]}} =$ false then
            $j \leftarrow j + 1$; $a_i \leftarrow b_{j+1}(p)$; $\tilde{a}_j \leftarrow a_i$;
            Update Newton interpolant $f_N^{[i]}$ on $a_0$, $a_1$, ..., $a_i$;
            If $a_i \notin \{a_0, \ldots, a_{i-1}\}$ then $k \leftarrow k + 1$; $f_N^{\{k\}} \leftarrow f_N^{[i]}$;
            Update the sparse racer algorithm on $\tilde{a}_0$, $\tilde{a}_1$, ..., $\tilde{a}_j$;
        Else
            $j \leftarrow 0$; randomly generate a non-zero $p$ from $S$;
            $\text{new}_{\text{[race]}} \leftarrow$ false; $a_i \leftarrow b_0(p)$; $\tilde{a}_0 \leftarrow a_i$;
            Update Newton interpolation $f_N^{[i]}$ on $a_0, a_1, \ldots, a_i$;
            If $a_i \notin \{a_0, \ldots, a_{i-1}\}$ then $k \leftarrow k + 1$; $f_N^{\{k\}} \leftarrow f_N^{[i]}$;
            Initialize the sparse racer algorithm at $\tilde{a}_0$;
(3)     [Check whether any racer finishes.]
        If $f_N^{\{k\}} = f_N^{\{k-1\}} = \cdots = f_N^{\{k-\eta\}}$ then break; Return $\tilde{f} \leftarrow f^{\{k\}}$;
        Else if the early termination criteria is met $\zeta$ times in a row for the
        sparse racer, then
(4)         Complete the sparse racer algorithm;
            If fail to complete, then $\text{new}_{\text{[race]}} \leftarrow$ true;
    End For;
    If $\tilde{f}$ is not defined then Fail;
```

5 Hybrids of Zippel Algorithm and Other Improvements

5.1 Prunings and hybrids of Zippel algorithm

Consider a black box polynomial f represented as

$$f(x_1, \ldots, x_n) = \sum_{(e_1, \ldots, e_n) \in J} c_{e_1, \ldots, e_n} x_1^{e_1} \cdots x_n^{e_n}, \tag{21}$$

where $0 \neq c_{e_1, \ldots, e_n} \in \mathbb{K}$, $J \subseteq (\mathbb{Z}_{\geq 0})^n$. Here $\#(J)$ is the number of non-zero terms in f. The Zippel algorithm (Zippel 1979a) is based on the following idea: if the representation in (21) is sparse, than during the variable by variable

[3] Depending on the sparse racer algorithm, there could be other restrictions on p.

interpolation, a zero coefficient is the image of a zero polynomial with high probability.

```
Algorithm: Zippel  (Zippel 1979a, 1990)
Input:   ▸ $f(x_1, \ldots, x_n)$: a multivariate black box polynomial over $\mathbb{K}$.
         ▸ $(x_1, \ldots, x_n)$: an ordered list of variables in $f$.
         ▸ $\delta$: an upper bound of $\deg(f)$.
Output:  ▸ $\sum_{(e_1,\ldots,e_n)\in J} c_{e_1,\ldots,e_n} x_1^{e_1} \cdots x_n^{e_n}$: which equals $f$ with high probability.
         ▸ Or an error message: if the procedure fails.
(1) [Initialize the anchor points.]
    Randomly pick $a_2, \ldots, a_n$ from a finite subset $S \subseteq \mathbb{K}$;
(2) [Interpolate one more variable: with high probability, we now have
    $f(x_1, \ldots, x_{i-1}, a_i, \ldots, a_n) = \sum_{(e_1,\ldots,e_{i-1})\in J_{i-1}} c_{e_1,\ldots,e_{i-1}} x_1^{e_1} \cdots x_{i-1}^{e_{i-1}}$,
    where $0 \neq c_{e_1,\ldots,e_{i-1}} \in \mathbb{K}, J_{i-1} \subset \mathbb{Z}_{\geq 0}^{i-1}$.]
    For $i = 1, \ldots, n$ Do
        [Update the degree upper bound for monomials in $x_i$.]
        $\delta_i = \max\{\delta - e_1 - \cdots - e_{i-1} \mid (e_1, \ldots, e_{i-1}) \in J_{i-1}\}$;
        [Update the number of monomials in $x_1, \ldots, x_{i-1}$.]
        $j_{i-1} \leftarrow \#(J_{i-1})$;
(3)     [Interpolate the coefficients of terms in $x_1, \ldots, x_{i-1}$ within $f$. These
        coefficients are polynomials in $\mathbb{K}[x_i]$ with degrees bounded by $\delta_i$.]
        For $k = 0, \ldots, \delta_i$ Do
            Randomly pick $b_k$ from a subset of $\mathbb{K}$;
(4)     [Locate the value of every such coefficient polynomial at $x_i = b_k$.]
        Set up a $j_{i-1}$ by $j_{i-1}$ transposed Vandermonde system:
        For $j = 0, \ldots, j_{i-1} - 1$ Do

            $\sum_{(e_1,\ldots,e_{i-1})\in J_{i-1}} \gamma_{e_1,\ldots,e_{i-1},k} (\tilde{a}_1^j)^{e_1} \cdots (\tilde{a}_{i-1}^j)^{e_{i-1}}$
                $= f(\tilde{a}_1^j, \ldots, \tilde{a}_{i-1}^j, b_k, a_{i+1}, \ldots, a_n);$        (22)

        If the system is singular then report "Failure;"
        Else solve for all $\gamma_{e_1,\ldots,e_{i-1},k}$; (Kaltofen and Lakshman Yagati 1988)
(5)     [Interpolate $j_{i-1}$ many coefficient polynomials in $x_i$ from their evaluations
        at $b_k$, $\gamma_{e_1,\ldots,e_{i-1},k}$, for $0 \leq k \leq \delta_i$.]
        For every $(e_1, \ldots, e_{i-1}) \in J_{i-1}$ Do
            Perform Newton interpolation so that
            $c^{[k]}_{i,(e_1,\ldots,e_{i-1})}(x_i) \in \mathbb{K}[x_i]$ and $c^{[k]}_{i,(e_1,\ldots,e_{i-1})}(b_s) = \gamma_{e_1,\ldots,e_{i-1},s}$, $0 \leq s \leq k$;
            $c^{[k]}_{i,(e_1,\ldots,e_{i-1})}(x_i) \leftarrow \sum_{s=0}^{k} c_{i,(e_1,\ldots,e_{i-1}),s} x_i^s$;
(6)     [Prune all the monomials with zero coefficient and update $J_i$.]
        $J_i = \emptyset$;
```

```
For every $(e_1, \ldots, e_{i-1}) \in J_{i-1}$ and $s = 0, \ldots, \delta_i$ Do
    If $c^{[\delta_i]}_{i,(e_1,\ldots,e_{i-1}),s} \neq 0$ then
        $c_{e_1,\ldots,e_{i-1},s} \leftarrow c^{[\delta_i]}_{i,(e_1,\ldots,e_{i-1}),s}$; $J_i \leftarrow J_i \cup \{(e_1, \ldots, e_{i-1}, s)\}$;
    Randomly pick $\tilde{a}_i$ from a subset of $\mathbb{K}$;
```

Introduce the *homogenizing variable* x_0 (Díaz and Kaltofen 1998) into f in (21), and define $\tilde{f} = f(x_0x_1, \ldots, x_0x_n) = \sum_{(e_1,\ldots,e_n)\in J} c_{e_1,\ldots,e_n} x_1^{e_1} \cdots x_n^{e_n} x_0^{e_1+\cdots+e_n}$. By interpolating $\tilde{f}(x_0,\, a_1,\, \ldots,\, a_n)$ in x_0 via Zippel's algorithm, we can prune the support structure of f in $f_0(x_0) \in \mathbb{K}[x_0]$.

Let $f_0(x_0) = \sum_{k=0}^{d} \gamma_{0,k} x_0^k$. Then $c_k(a_1, \ldots, a_n) = \gamma_{0,k}$, and

$$c_k(x_1, \ldots, x_n) = \sum_{e_1+\cdots+e_n=k,(e_1,\ldots,e_n)\in J} c_{e_1,\ldots,e_n} x_1^{e_1} \cdots x_n^{e_n}.$$

Every term in c_k is of degree k in $\mathbb{K}[x_1, \ldots, x_n]$ and $\deg f_0$ in $\mathbb{K}[x_0]$ evaluates $\deg f$ in $\mathbb{K}[x_1, \ldots, x_n]$. The degree of every non-zero term in $f_0(x_0)$ provides an upper bound for the degrees of all the intermediate terms of its coefficient polynomial.

Now, we refine the pruning in (Díaz and Kaltofen 1998). Comparing to Zippel's idea, we do two more types of pruning so that we may further reduce the size of the transposed Vandermonde system in (22).

During the process of interpolating a homogenized polynomial via Zippel's algorithm, in Step (2) with high probability we have

$$\tilde{f}(x_0, x_1, \ldots, x_{i-1}, a_i, \ldots, a_n) = \sum_{(e_0,\ldots,e_{i-1})\in J_{i-1}} c_{e_0,\ldots,e_{i-1}} x_1^{e_1} \cdots x_{i-1}^{e_{i-1}} x_0^{e_0},$$

where $0 \neq c_{e_0,\ldots,e_{i-1}} \in \mathbb{K}$, $J_{i-1} \subset (\mathbb{Z}_{\geq 0})^i$. For every term with $(e_0, \ldots, e_{i-1}) \in J_{i-1}$ and $e_1 + \cdots + e_{i-1} = e_0$, the degree of the coefficient polynomial in variables $x_1, \ldots,\, x_{i-1}$ has already reached the total degree upper bound e_0. That is, $c_{e_0,\ldots,e_{i-1}} x_1^{e_1} \cdots x_{i-1}^{e_{i-1}} x_0^{e_0}$ is an actual term in $\tilde{f}$. We let $g_{i-1}(x_0, \ldots, x_n)$ denote a polynomial summing up all such fully interpolated terms, and form J'_{i-1} from J_{i-1} by removing all $(e_0, \ldots, e_{i-1})$'s such that $e_1 + \cdots + e_{i-1} = e_0$. The equation (22) in Step (4) of Zippel algorithm now becomes

$$\begin{aligned} f(\tilde{a}_0^j, \ldots, &\tilde{a}_{i-1}^j, b_k, a_{i+1}, \ldots, a_n) \\ &= \sum_{(e_0,\ldots,e_{i-1})\in J'_{i-1}} \gamma_{e_0,\ldots,e_{i-1},k} (\tilde{a}_1^j)^{e_1} \cdots (\tilde{a}_{i-1}^j)^{e_{i-1}} (\tilde{a}_0^j)^{e_0} \\ &\quad + g_{i-1}(\tilde{a}_0^j, \ldots, \tilde{a}_{i-1}^j, b_k, a_{i+1}, \ldots, a_n). \end{aligned}$$

Since $\#(J'_{i-1}) \leq \#(J_{i-1})$, by subtracting g_{i-1} from both sides of (22), we may reduce the size of the transposed Vandermonde system. All terms in g_{i-1} are *permanently pruned* since they have been fully interpolated and will not be further interpolated in $x_i, \ldots, x_n$.

On the other hand, when interpolating coefficients of the terms in $x_0,\ldots,x_{i-1}$ from $\tilde{f}(x_0, \ldots, x_{i-1}, x_i, a_{i+1}, \ldots, a_n)$, some coefficients might be interpolated via early termination before the degree bound is reached, and can be taken out of the loop in Step (3) before other coefficient polynomials are interpolated. As a result, the size of the system (22) may be further reduced. Those terms are *temporarily pruned* from the interpolation in x_i, and will be interpolated later in $x_{i+1}, \ldots, x_n$.

Without a total degree bound supplied as an input, the permanent prunings depend on the interpolation of homogenizing variable, while the temporary prunings can be carried out regardless the introduction of the homogenizing variable.

Our hybrids of Zippel algorithm use Zippel's variable by variable method as the outer loop, and introduce a homogenizing variable to perform permanent and temporary prunings. In our implementation, the homogenization modification can be "turned off" (Subsection 6.3), and a racing algorithm is employed in each univariate interpolation. Which takes advantage of better pruning techniques and more efficient embedded univariate interpolations.

5.2 Modular techniques in the univariate Ben-Or/Tiwari algorithm

In the modular implementation, the hybrids of Zippel algorithm may provide another advantage: when racing Newton against Ben-Or/Tiwari, the size of modulus required by the Ben-Or/Tiwari algorithm could be greatly reduced.

In order to control the size of the coefficients in the error locator polynomial of the Berlekamp/Massey algorithm that is embedded in the Ben-Or/Tiwari algorithm, Kaltofen et al. (1990) apply modular techniques for finding $\Lambda(z)$ and locating the roots of $\Lambda(z)$. However, to provide a modular image of $\Lambda(z)$ sufficient for the recovery of all terms β_i, the modulus q needs to be *sufficiently large*. They use a modulus p^k that is larger than each b_j, the value of term β_j evaluated a prime number. Consider a multivariate polynomial f and let $d = \deg f$. Since 2 is the smallest prime, a sufficiently large modulus p^k is at least 2^d. This means when $\deg f$ is relatively large, we need to perform all computations modulo an integer of length proportional to the degree, even though the coefficients could be of a much smaller size.

The modulus can be reduced by interpolating a subset of variables at a time. However, in the univariate case with $d = \deg f(x)$, a prime q larger than d (instead of 2^d) can already provide a *sufficiently large* modulus. Namely, we evaluate the single variable x at a primitive root ϱ and recover the term exponents as the discrete logarithms of b_j. There are $\phi(q-1)$ primitive roots modulo q, with $u/\text{loglog}u = O(\phi(u))$ for any integer u and ϕ denoting Euler's totient function (Hardy and Wright 1979, sec. 18.4). Even if x is evaluated at $\tilde{\varrho}$ that is not a primitive root, as long as the order of $\tilde{\varrho}$ is larger than d, we still can recover all the term exponents in f (also see Subsection 6.3).

Our algorithm picks a random residue ϱ for x and for each b_k tries the exponents $e_k = 0, 1, 2, \ldots$ until $\varrho^{e_k} \equiv b_k \pmod{q}$. The method produces an incorrect term exponent if for ϱ, ϱ^2, $\ldots$, $\varrho^{\lambda_q(\varrho)} \equiv 1 \pmod{q}$ we have $e_k \geq \lambda_q(\varrho)$. However, such false exponents highly likely lead to inconsistencies in later steps: a possible immediate inconsistency is to the degrees of the concurrent Newton interpolant. In that case, the univariate result is recovered by Newton or another restarted Ben-Or/Tiwari interpolation. If the false exponent is still not caught, with high likelihood the inconsistency shows up later, at the latest at the comparison of the final result with the black box evaluated at an additional random point (see Subsection 6.2).

We quantify the trade-off between the size of the modulus and the number of black box probes in the Ben-Or/Tiwari versus the univariate Ben-Or/Tiwari within Zippel. With early termination and ignoring the size of coefficients, in the former we have $q > 2^{\deg f}$ versus $q = O(\deg f)$, while the number of probes is $2t + \zeta$ versus $O(n(2t + \zeta))$. Therefore, if the degrees are small but there are many variables, the pure Ben-Or/Tiwari may still out-perform the hybrid of Zippel's algorithm. We add that at any stage in the variable by variable Zippel's method, the rest of the variables could be interpolated by a multivariate Ben-Or/Tiwari algorithm.

Note that for a small finite coefficient field, say $\mathbb{Z}_2$, we can switch to the coefficient domain $\mathbb{Z}_2[x_n]$, where x_n is the last variable, and proceed modulo irreducible polynomials in $\mathbb{Z}_2[x_n]$.

6 Maple Implementation

The ProtoBox package is the Maple implementation of some of our algorithms: the early terminations with thresholds in the Newton and Ben-Or/Tiwari algorithms, the racing algorithm that matches Newton against the Ben-Or/Tiwari algorithm, its hybrid of Zippel with pruning, and the homogenization modification.

We test ProtoBox to interpolate black box polynomials in Table 1. Note that f_1, f_2 are from (Zippel 1979b, p. 100), and f_3, f_4 from (Zippel 1979b, p. 102).

Table 1. Polynomials used in the tests.

$f_1(x_1, \ldots, x_{10})$	$=$	$x_1^2 x_3^3 x_4 x_6 x_8 x_9^2 + x_1 x_2 x_3 x_4^2 x_5^2 x_8 x_9 + x_2 x_3 x_4 x_5^2 x_8 x_9$ $+ x_1 x_3^3 x_4^2 x_5^2 x_6^2 x_7 x_8^2 + x_2 x_3 x_4 x_5^2 x_6 x_7 x_8^2$
$f_2(x_1, \ldots, x_{10})$	$=$	$x_1 x_2^2 x_4^2 x_8 x_9^2 x_{10}^2 + x_2^2 x_4 x_5^2 x_6 x_7 x_9 x_{10}^2 + x_1^2 x_2 x_3 x_5^2 x_7^2 x_9^2$ $+ x_1 x_3^2 x_4^2 x_7^2 x_9^2 + x_1^2 x_3 x_4 x_7^2 x_8^2$
$f_3(x_1, \ldots, x_{10})$	$=$	$9 x_2^3 x_3^3 x_5^2 x_6^2 x_8^3 x_9^3 + 9 x_1^3 x_2^2 x_3^3 x_5^2 x_7^2 x_8^2 x_9^3 + x_1^4 x_3^4 x_4^2 x_5^4 x_6^4 x_7 x_8^5 x_9$ $+ 10 x_1^4 x_2 x_3^4 x_4^4 x_5^4 x_7 x_8^3 x_9 + 12 x_2^3 x_4^3 x_6^3 x_7^2 x_8^3$
$f_4(x_1, \ldots, x_{10})$	$=$	$9 x_1^2 x_3 x_4 x_6^3 x_7^2 x_8 x_{10}^4 + 17 x_1^3 x_2 x_5^2 x_6^2 x_7 x_8^3 x_9^4 x_{10}^3 + 3 x_1^3 x_2^2 x_6^3 x_{10}^2$ $+ 17 x_2^2 x_3^4 x_4^2 x_7^4 x_8^3 x_9 x_{10}^3 + 10 x_1 x_3 x_5^2 x_6^2 x_7^4 x_8^4$
$f_5(x_1, \ldots, x_{50})$	$=$	$\sum_{i=1}^{50} x_i^{50}$
$f_6(x_1, \ldots, x_5)$	$=$	$\sum_{i=1}^{5} (x_1 + x_2 + x_3 + x_4 + x_5)^i$
$f_7(x_1, x_2, x_3)$	$=$	$x_1^{20} + 2x_2 + 2x_2^2 + 2x_2^3 + 2x_2^4 + 3x_3^{20}$

6.1 Black box probes

In the racing algorithm that runs Newton against Ben-Or/Tiwari, for purpose of comparison we have "turned off" either of the competing algorithms by setting the corresponding threshold to ∞ and thus forced all the interpolations through the remaining active one.

In the hybrids of Zippel, we compare the black box probes required in different embedded univariate interpolations: Newton, Ben-Or/Tiwari, and the racing of Newton against Ben-Or/Tiwari (all with threshold one). We use larger moduli to reduce the chance of hitting unlucky numbers that might interfere with the performance of each algorithm. We have run each algorithm ten times for each polynomial with different random numbers and taken the average of the number of black box probes needed. The results are listed in Table 2. Note that racing Ben-Or/Tiwari against Newton may yield a count less than the minimum of using either exclusively. This is because in Zippel's variable by variable approach, in each univariate interpolation subproblem the winners might alternate between Ben-Or/Tiwari and Newton.

6.2 Thresholds

In order to further improve or control the performance, we have implemented additional thresholds in `ProtoBox`, aside from the thresholds η and ζ for early termination.

Table 2. Black box probes needed for different hybrids of Zippel's algorithm.

	mod	Newton	Ben-Or/Tiwari	Racing
f_1	100003	147	137	126
f_2	100003	146	143	124
f_3	100003	209	143	133
f_4	100003	188	149	133
f_5	100000007	2652	251	251
f_6	100000007	965	1256	881
f_7	100003	94	46	41

One way to increase confidence in the interpolation result is to compare the value of the output with that of the black box polynomial at some random points. If a disagreement is discovered at one point, the result is declared invalid. The number of random points for the post test is an optional argument "`posttest_thresh`," which by default is zero.

Our probabilistic algorithms generate random scalars at different stages; sometimes an unlucky choice in an intermediate step will cause the overall algorithm to abort, although that could be remedied by simply trying another set of random elements in that step. Yet, to avoid a possibly infinite loop, as a principle, such retries shall be bounded in number. Such is the situation in Zippel's algorithm: it is possible that all the terms in previous variables are correctly interpolated, but in (22) two different terms map to a same value $\tilde{a}_1^{e_1} \cdots \tilde{a}_{i-1}^{e_{i-1}} = \tilde{a}_1^{\bar{e}_1} \cdots \tilde{a}_{i-1}^{\bar{e}_{i-1}}$ at unlucky $\tilde{a}_k$ and the Vandermonde system in (22) becomes singular. We experienced such failure for large results at the last variable, and all was lost. Therefore the optional argument "`mapmon_thresh`," zero by default, defines the number of retries with new $\tilde{a}_1, \ldots, \tilde{a}_{i-1}$.

We have to delay the updates of Newton interpolant when a point is repeated. In order to avoid incomplete interpolations due to repeated points, we extend the upper bound for each univariate interpolation loop by the optional argument "`rndrep_thresh`" that is zero by default.

We have tested our new thresholding techniques on small moduli, where the benefits are most apparent. There are now five different thresholds and we have tested the three settings listed in Table 3. Note that combination 1 is the default in `ProtoBox`.

Figures 2 through 5 display the results of the hybrid of Zippel's algorithm that races Newton against Ben-Or/Tiwari for interpolating f_1, f_2, f_3, and f_4. We have tested each threshold combination for each modulus for 100 random

Table 3. Different combinations of thresholds used in the tests.

Combination	η	ζ	`posttest_thresh`	`mapmon_thresh`	`rndrep_thresh`
1	1	1	0	0	0
2	2	2	1	2	2
3	3	3	2	4	4

seeds. The height of each rectangle at every modulus, solid and airy parts combined, reflects the number of non-false results in 100 runs, which are either correct results (indicated as the solid part) or error messages (airy part of the rectangle). The balance to full height 100 is the number of times that the algorithm returned an incorrect polynomial. The three slices on each modulus reflect the performance under the three threshold combinations in Table 3, with combinations 1 through 3 being listed from left to right.

We observe that higher thresholds might yield less success, e.g., in Figure 2 for $p = 97$ threshold combination 2 vs. 3. There is, of course, statistically variation possible, but we note that higher early termination thresholds may cause the algorithm to abort for lack of new test points while holding a correct partial result. In light of the Schwartz-Zippel lemma it appears paradoxical that modulo certain larger primes we see a significantly lower yield in our tables. We provide an explanation in the next section.

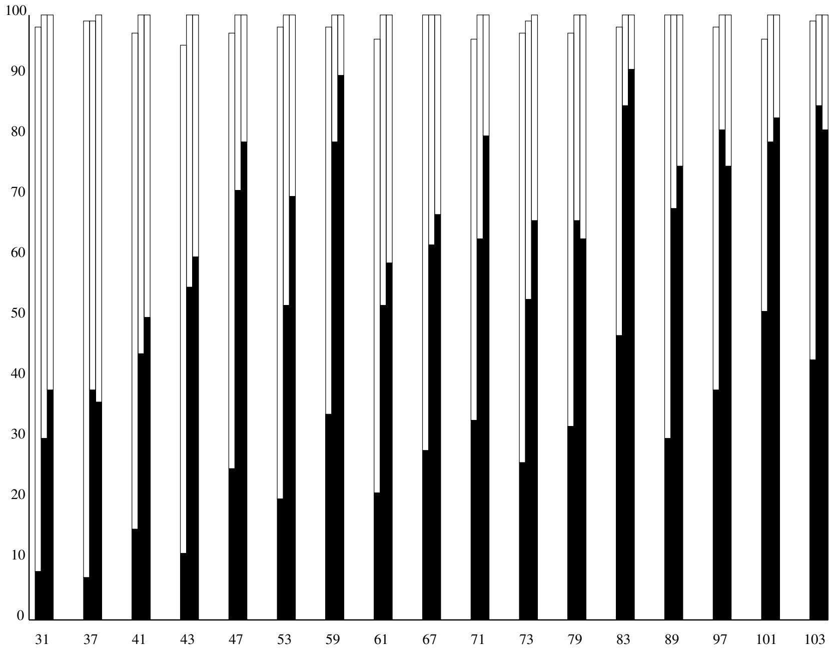

Fig. 2. Interpolations of f_1 in different threshold combinations and moduli after 100 runs.

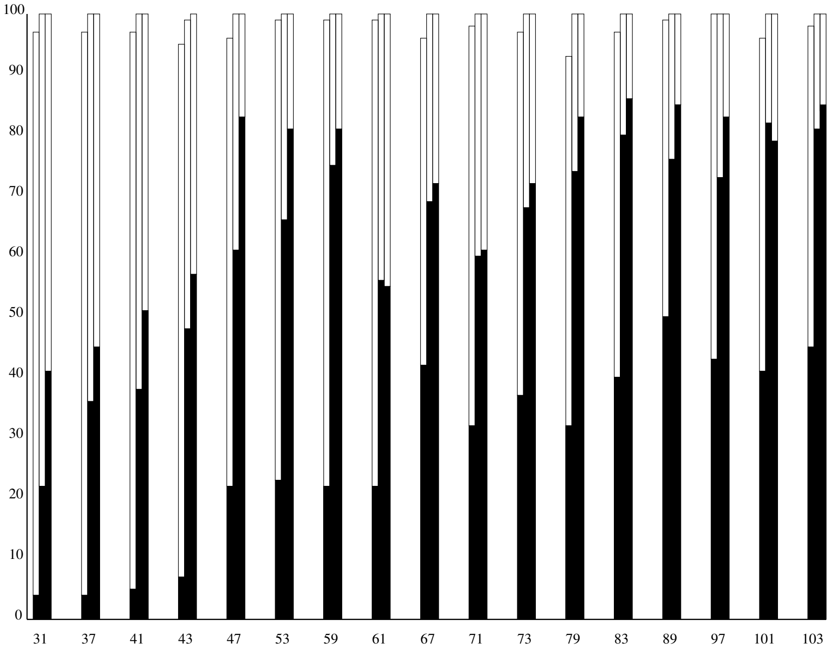

Fig. 3. Interpolations of f_2 in different threshold combinations and moduli after 100 runs.

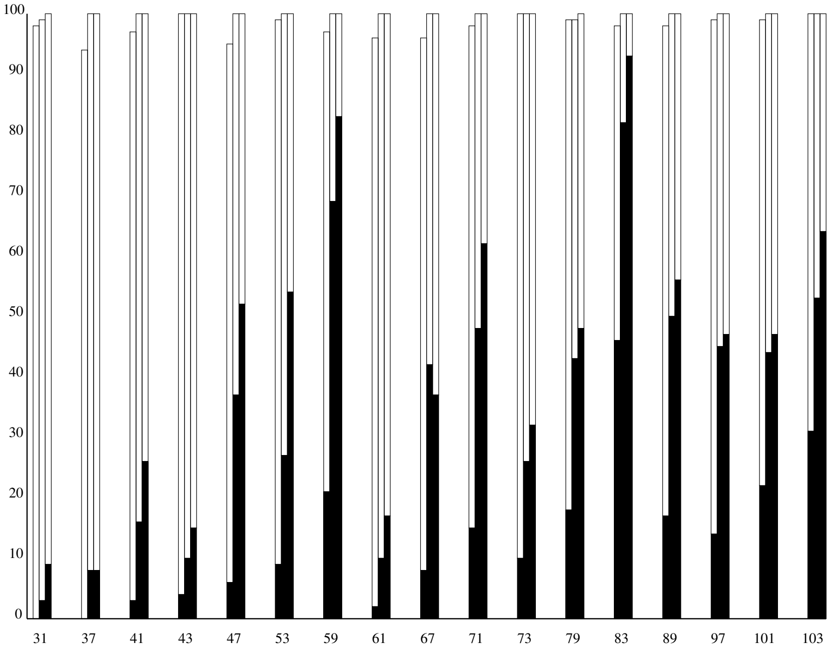

Fig. 4. Interpolations of f_3 in different threshold combinations and moduli after 100 runs.

6.3 The small modulus heuristic

For multivariate polynomials whose degree in each variable is much lower than the total degree, some very small moduli might suffice if we "turn off" the homogenization modification. Table 4 presents such heuristic of interpolations

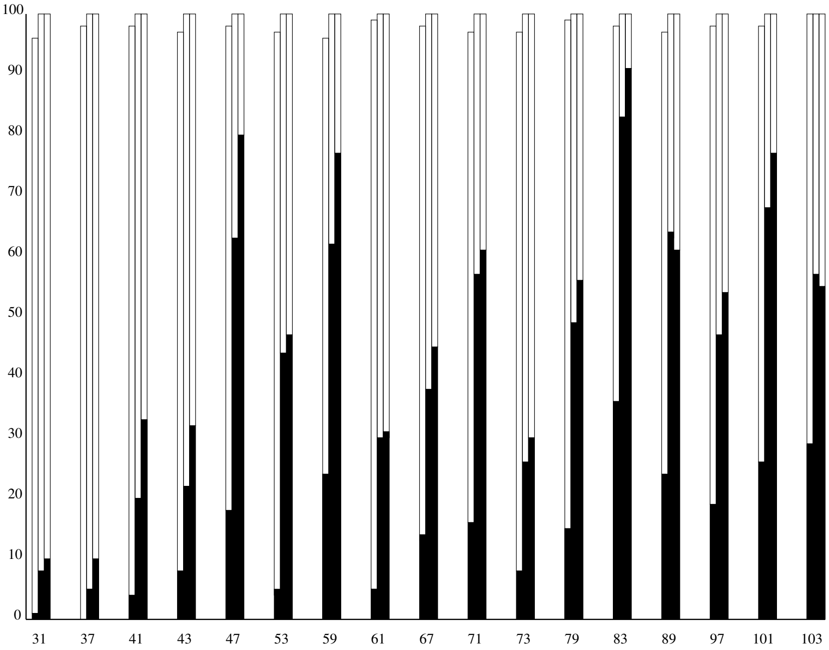

Fig. 5. Interpolations of f_4 in different threshold combinations and moduli after 100 runs.

on some very small moduli. For brevity, "`posttest_thresh`" is denoted as τ, "`mapmon_thresh`" as κ, and "`rndrep_thresh`" as γ. After 100 runs, the column under "=" records the times a correct result is returned, under "!" an error message, and under "$\neq$" a false result. Note that for modulo 17, two terms in f_4 are zero.

Table 4. Interpolations on small moduli without homogenization modification.

	Thresholds			mod 11			mod 13			mod 17			mod 19		
	η, ζ	τ	κ, γ	=	$\neq$	!	=	$\neq$	!	=	$\neq$	!	=	$\neq$	!
f_1	2	2	6	28	2	70	30	0	70	60	0	40	44	1	55
f_2	2	2	6	8	1	91	26	0	74	42	0	58	52	0	48
f_3	2	2	6	7	1	92	2	0	98	20	0	80	13	1	86
f_4	2	2	6	5	0	95	0	1	99	39	0	61	17	0	83

The Ben-Or/Tiwari algorithm requires a field element that has enough distinct values for its powers in order to recover different terms (see Subsection 5.2). Therefore, the total number of residues of high order directly affects the success rate of the univariate Ben-Or/Tiwari algorithm.[4] Figure 6 displays the order of each residue modulo 11 and modulo 13. The minimal orders required for different polynomials are indicated as dotted lines. For f_3 and f_4, there are

[4] We owe this observation to James H. Davenport.

fewer elements of sufficient order modulo 13 than modulo 11.

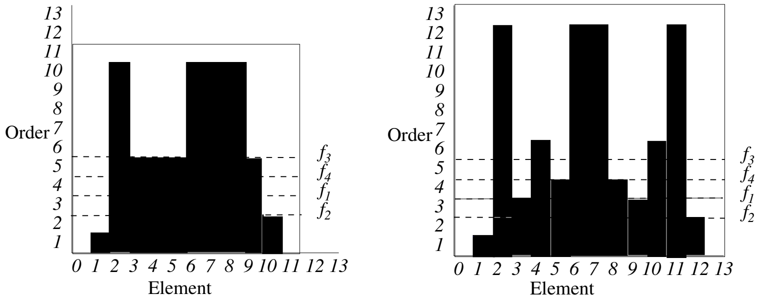

Fig. 6. The order of residues modulo 11 and 13, and the minimal orders required for interpolating different polynomials through the univariate Ben-Or/Tiwari algorithm.

Acknowledgment of Support

This material is based on work supported in part by the National Science Foundation under Grant Nos. DMS-9977392, CCR-9988177, and CCR-0113121 (Kaltofen) and by the Natural Sciences and Engineering Research Council of Canada, the Ontario Research & Development Challenge Fund, and Maplesoft (Lee).

References

Ben-Or, M., Tiwari, P., 1988. A deterministic algorithm for sparse multivariate polynomial interpolation. In: Proc. Twentieth Annual ACM Symp. Theory Comput. ACM Press, New York, N.Y., pp. 301–309.

Delsarte, P., Genin, Y. V., Kamp, Y. G., 1985. A generalization of the Levinson algorithm for Hermitian Toeplitz matrices with any rank profile. IEEE Trans. Acoustics, Speech, and Signal Process. ASSP-33 (4), 964–971.

DeMillo, R. A., Lipton, R. J., 1978. A probabilistic remark on algebraic program testing. Information Process. Letters 7 (4), 193–195.

Díaz, A., Kaltofen, E., 1998. FoxBox a system for manipulating symbolic objects in black box representation. In: Gloor, O. (Ed.), Proc. 1998 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'98). ACM Press, New York, N. Y., pp. 30–37.

Dress, A., Grabmeier, J., 1991. The interpolation problem for k-sparse polynomial and character sums. Adv. Appl. Math 12, 57–75.

Emiris, I. Z., 1998. A complete implementation for computing general dimensional convex hulls. Int. J. Comput. Geom. Appl. 8 (2), 223–254.

Freeman, T. S., Imirzian, G., Kaltofen, E., Lakshman Yagati, 1988. Dagwood: A system for manipulating polynomials given by straight-line programs. ACM Trans. Math. Software 14 (3), 218–240.

Gantmacher, F. R., 1977. The theory of matrices. Vol. 1. Chelsea publishing company.

Giesbrecht, M., Kaltofen, E., Lee, W.-s., 2002. Algorithms for computing the sparsest shifts for polynomials via the Berlekamp/Massey algorithm. In: Mora, T. (Ed.), Proc. 2002 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'02). ACM Press, New York, N. Y., pp. 101–108.

Giesbrecht, M., Kaltofen, E., Lee, W.-s., 2003. Algorithms for computing sparsest shifts of polynomials in power, Chebychev, and Pochhammer bases. J. Symbolic Comput. To appear, 27 pages. In the special issues on papers of the 2002 Internat. Symp. Symbolic Algebraic Comput.

Gohberg, I., Koltracht, I., 1989. Efficient algorithm for Toeplitz plus Hankel matrices. Integral Equations and Operator Theory 12, 136–142.

Grigoriev, D. Y., Karpinski, M., Singer, M. F., 1990. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. SIAM J. Comput. 19 (6), 1059–1063.

Grigoriev, D. Y., Karpinski, M., Singer, M. F., 1991. The interpolation problem for k-sparse sums of eigenfunctions of operators. Adv. Appl. Math. 12, 76–81.

Grigoriev, D. Y., Karpinski, M., Singer, M. F., 1994. Computational complexity of sparse rational function interpolation. SIAM J. Comput. 23, 1–11.

Hardy, G. H., Wright, E. M., 1979. An Introduction to the Theory of Numbers, 5th Edition. Oxford Univ. Press, Oxford.

Kaltofen, E., Lakshman Y. N., Wiley, J. M., 1990. Modular rational sparse multivariate polynomial interpolation. In: Watanabe, S., Nagata, M. (Eds.), Proc. 1990 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'90). ACM Press, pp. 135–139.

Kaltofen, E., Lakshman Yagati, 1988. Improved sparse multivariate polynomial interpolation algorithms. In: Gianni, P. (Ed.), Symbolic Algebraic Comput. Internat. Symp. ISSAC '88 Proc. Vol. 358 of Lect. Notes Comput. Sci. Springer Verlag, Heidelberg, Germany, pp. 467–474.

Kaltofen, E., Lee, W.-s., Lobo, A. A., 2000. Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm. In: Traverso, C. (Ed.), Proc. 2000 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'00). ACM Press, New York, N. Y., pp. 192–201.

Kaltofen, E., Lobo, A., 1996. On rank properties of Toeplitz matrices over finite fields. In: Lakshman Y. N. (Ed.), Proc. 1996 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'96). ACM Press, New York, N. Y., pp. 241–249.

Kaltofen, E., Trager, B., 1990. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. J. Symbolic Comput. 9 (3), 301–320.

Lakshman Y. N., Saunders, B. D., 1995. Sparse polynomial interpolation in non-standard bases. SIAM J. Comput. 24 (2), 387–397.

Lee, W.-s., Dec. 2001. Early termination strategies in sparse interpolation algorithms. Ph.D. thesis, North Carolina State Univ., Raleigh, North Carolina, 107 pages.

Massey, J. L., 1969. Shift-register synthesis and BCH decoding. IEEE Trans. Inf. Theory IT-15, 122–127.

Prony, R., Floréal et Prairial III (1795). Essai expérimental et analytique sur les lois de la Dilatabilité des fluides élastique et sur celles de la Force expansive de la vapeur de l'eau et de la vapeur de l'alkool, à différentes températures. J. de l'École Polytechnique 1, 24–76, r. Prony is Gaspard(-Clair-François-Marie) Riche, baron de Prony.

Schwartz, J. T., 1980. Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27, 701–717.

Wiedemann, D., 1986. Solving sparse linear equations over finite fields. IEEE Trans. Inf. Theory IT-32, 54–62.

Zilic, Z., Radecka, K., 1999. On feasible multivariate polynomial interpolations over arbitrary fields. In: Dooley, S. (Ed.), ISSAC 99 Proc. 1999 Internat. Symp. Symbolic Algebraic Comput. ACM Press, New York, N. Y., pp. 67–74.

Zippel, R., 1979a. Probabilistic algorithms for sparse polynomials. In: Proc. EUROSAM '79. Vol. 72 of Lect. Notes Comput. Sci. Springer Verlag, Heidelberg, Germany, pp. 216–226.

Zippel, R., 1990. Interpolating polynomials from their values. J. Symbolic Comput. 9 (3), 375–403.

Zippel, R. E., Sep. 1979b. Probabilistic algorithms for sparse polynomials. Ph.D. thesis, Massachusetts Inst. of Technology, Cambridge, USA.