Abstract

LEE, WEN-SHIN. Early Termination Strategies in Sparse Interpolation Algorithms. (Under the direction of Professor Erich Kaltofen.)

A black box polynomial is an object that takes as input a value for each variable and evaluates the polynomial at the given input. The process of determining the coefficients and terms of a black box polynomial is the problem of black box polynomial interpolation. Two major approaches have been addressing such purpose: the dense algorithms whose computational complexities are sensitive to the degree of the target polynomial, and the sparse algorithms that take advantage of the situation when the number of non-zero terms in a designate basis is small. In this dissertation we cover power, Chebyshev, and Pochhammer term bases. However, a sparse algorithm is less efficient when the target polynomial is dense, and both approaches require as input an upper bound on either the degree or the number of non-zero terms. By introducing randomization into existing algorithms, we demonstrate and develop a probabilistic approach which we call "early termination." In particular we prove that with high probability of correctness the early termination strategy makes different polynomial interpolation algorithms "smart" by adapting to the degree or to the number of non-zero terms during the process when either is not supplied as an input. Based on the early termination strategy, we describe new efficient univariate algorithms that race a dense against a sparse interpolation algorithm in order to exploit the superiority of one of them. We apply these racing algorithms as the univariate interpolation procedure needed in Zippel's multivariate sparse interpolation method. We enhance the early termination approach with thresholds, and present insights to other such heuristic improvements. Some potential of the early termination strategy is observed for computing a sparse shift, where a polynomial becomes sparse through shifting the variables by a constant.

KEYWORDS: Sparse polynomial, black box polynomial, interpolation, sparse interpolation, randomized algorithm, Chebyshev basis, Pochhammer basis, early termination, racing two algorithms, sparse shift, Zippel's algorithm, Ben-Or's and Tiwari's algorithm.

EARLY TERMINATION STRATEGIES IN SPARSE INTERPOLATION ALGORITHMS

BY

WEN-SHIN LEE

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF
NORTH CAROLINA STATE UNIVERSITY
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

APPLIED MATHEMATICS
COMPUTATIONAL MATHEMATICS CONCENTRATION

RALEIGH

2001

APPROVED BY:

ERICH KALTOFEN CHAIR OF ADVISORY COMMITTEE	HOON HONG
CARLA SAVAGE	MICHAEL SINGER

Dedicated to my parents

Po-chun Lee and Ping-i Tsai

謹獻給我的父母

李伯諄和蔡秉義

Biography

Wen-shin Lee was born on March 18, 1971, in Taipei city, Taiwan. Following her graduation from Taipei Municipal First Girls' Senior High School in 1989, she attended National Taiwan University in Taipei city, Taiwan, where she received her B.S. degree in mathematics in 1994. In 1999, she obtained her M.S. degree in applied mathematics, and in 2001 she earned her Ph.D. in applied mathematics, computational mathematics concentration, with a minor in computer science, both from North Carolina State University, Raleigh, North Carolina.

Acknowledgements

My gratitude goes to my advisor, Professor Erich Kaltofen, for his academic guidance, friendship, and many valuable lessons that will benefit me beyond my graduate study. I also want to express my appreciation to my committee professors, Professor Hoon Hong, Professor Carla Savage, and Professor Michael Singer, for their precious comments and support.

I thank current and past members of Symbolic Solutions Group, particularly, William Turner, John May, Jack Perry, Peter Berman, Manfred Minimair, Markus Hitz, Angel Díaz, Austin Lobo, and Lakshman Y. N., for their encouragement and help. Special recognition goes to April Jackson and Brenda Currin at mathematics department for their kind assistance.

I appreciate the support provided by the mathematics department and the graduate school at North Carolina State University as well as National Science Foundation.

For all the blessings I have been given, I thank God.

Table of Contents

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

1.1 Representations and interpolations of polynomials 2
1.2 Early termination strategy and racing algorithms 5
1.3 Hybrids of Zippel and other improvements 7
1.4 Maple implementation and further developments 8

2 Early Termination in Dense Univariate Interpolations **10**

2.1 Dense univariate interpolations 11
2.2 Early termination in dense univariate interpolations 12

3 Early Termination in Sparse Interpolations with respect to the Power Basis **16**

3.1 The Berlekamp/Massey algorithm 17
3.2 The Ben-Or/Tiwari interpolation algorithm 19
3.3 Early termination in the Ben-Or/Tiwari interpolation algorithm . . 26

4 Early Termination in Sparse Interpolations with respect to Non-standard Bases **34**

4.1 Univariate sparse interpolations in the Pochhammer basis 35
4.2 Early termination of sparse interpolations in the Pochhammer basis 37
4.3 Univariate sparse interpolations in the Chebyshev basis 43
4.4 Early termination of sparse interpolations in the Chebyshev basis . 45

5 Early Termination in Racing Algorithms **58**

5.1 Race dense against sparse interpolations in early termination 59
5.2 Racing algorithm 68

6 Hybrids of Zippel Algorithm and Other Improvements **76**
6.1 The Zippel algorithm 77
6.2 Prunings and hybrids of Zippel algorithm 80
6.3 Modular techniques in the univariate Ben-Or/Tiwari algorithm . . . 84

7 Maple Implementation **86**
7.1 Black box probes 86
7.2 Thresholds 88
7.3 The heuristic on small moduli 91

8 Conclusions and Further Developments **98**
8.1 Conclusions 98
8.2 Future developments 100

Literature Citations **104**

List of Tables

7.1 Polynomials in *ProtoBox* performance tests. 87
7.2 Black box probes needed for different embedded univariate interpolations. 88
7.3 Sets of different combinations of thresholds used in the tests. 90
7.4 Interpolations on very small moduli without homogenization modification after 100 runs. 91

8.1 Early termination in sparse shifts and the Ben-Or/Tiwari algorithm. 101

List of Figures

1.1 A black box polynomial $f(x_1, \ldots, x_n)$ and its interpolation. 3

3.1 Subresultant coefficient 29

5.1 A possible scenario of the racing algorithm. 70

6.1 Interpolate $f(x, y) = 3x^5y^3 + 2x^5 + y^2 + 5$ via Zippel algorithm. . . 80

7.1 The performance of interpolating f_1 under different threshold combinations and moduli after 100 runs. 92

7.2 The performance of interpolating f_2 under different threshold combinations and moduli after 100 runs. 93

7.3 The performance of interpolating f_3 under different threshold combinations and moduli after 100 runs. 94

7.4 The performance of interpolating f_4 under different threshold combinations and moduli after 100 runs. 95

7.5 The order of elements in modulo 11 and the least order required in interpolating different polynomials through Ben-Or/Tiwari. 96

7.6 The order of elements in modulo 13 and the least order required in interpolating different polynomials through Ben-Or/Tiwari. 97

Chapter 1

Introduction

Interpolation has long been an extensively studied topic, for both its role as a basic tool in approximating given functions and its wide applications. Our research focuses on the problem of efficiently interpolating polynomials that have sparse representations, or with respect to their sparse representations.

By introducing randomization into existing algorithms, we demonstrate and develop a probabilistic approach, *early termination*. With high probability of correctness, the early termination strategy makes different polynomial interpolation algorithms "smart" by adapting to the degree or the number of non-zero terms during the process when neither is supplied as an input.

Derived from the early termination strategy, we describe new efficient interpolation algorithms and present insights to other heuristic improvements as well as certain related topics. Further potentials of the early termination strategy are also observed based on its success in polynomial interpolations.

1.1 Representations and interpolations of polynomials

As the most commonly used representation, a polynomial representation represents a polynomial $f(x_1, \ldots, x_n)$ as a sum of terms with their coefficients (in the power basis)

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{e_{j,1}} \cdots x_n^{e_{j,n}}. \tag{1.1}$$

In general, when a polynomial is represented as in (1.1) with total degree $d = \deg(f)$, the number of its non-zero terms could be as many as

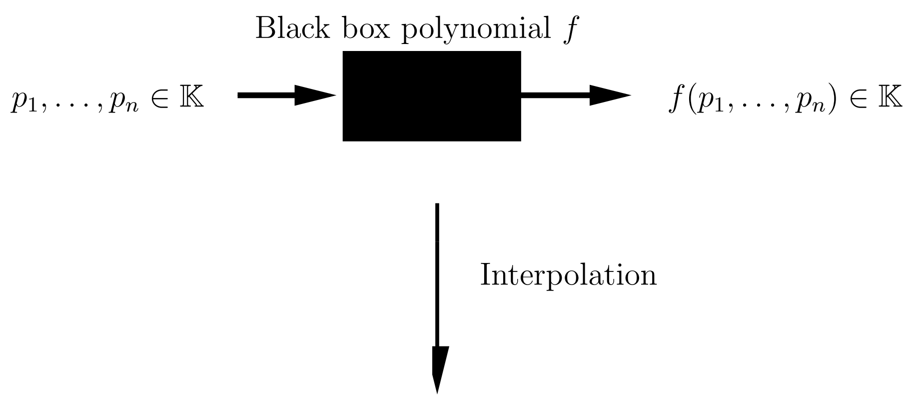
$$\binom{n+d}{d},$$

which grows exponentially with both n and d.

For the polynomials that have much fewer non-zero coefficient terms, or sparse polynomials, not only their polynomial representations are more concise, thus occupy less space in representations, but also there are efficient algorithms that take advantage of the sparsity in the polynomial representations.

Though many large scale problems tend to be sparse, it is also natural to ask whether we can determine a representation basis such that it can represent a given polynomial in a sparse manner. Notice that for any given polynomial, there are different polynomial representations based on different bases, and as a result, the sparsity of a polynomial depends on the choice of the bases.

Another method to represent a polynomial is the black box representation, which is the object that takes as input a value for each variable and evaluates the polynomial being represented at the input point. The black box representations can be manipulated efficiently [16], and it is useful to convert black box representations to polynomial representations. The operation of recovering the coefficients and terms of a black box polynomial from its values at given points is the black box polynomial interpolation (see Figure 1.1.)



$$f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{e_{j,1}} \cdots x_n^{e_{j,n}} \in \mathbb{K}[x_1, \ldots, x_n], \mathbb{K} \text{ is a field.}$$

Figure 1.1: A black box polynomial $f(x_1, \ldots, x_n)$ and its interpolation.

When interpolating a polynomial from its evaluations, Newton interpolation and Lagrange interpolation have been widely implemented. As univariate interpolation algorithms, both need $\deg(f(x)) + 1$ evaluations, or black box probes, at distinct points in order to recover a polynomial representation of the target polynomial $f(x)$. In the case the target polynomial is multivariate, both algorithms can be implemented in a variable by variable manner and recover one variable at a time until the target polynomial is fully interpolated. Such algorithms are called "dense" because they proceed regardless of the number of terms with zero coefficient, and do not take advantage of the sparsity in the target polynomial. It can be easily observed the drawback as the inefficiency of a dense algorithm when the target polynomial has much fewer non-zero monomials.

Interpolation algorithms whose computational complexities are sensitive to the sparsity of their target polynomials are one of the major contributions of computer algebra to computer science and mathematics. The first such result was obtained by Richard Zippel in 1979 [24]. Zippel's algorithm is efficient in the multivariate case when the target polynomial is sparse than a variable by variable Lagrange or Newton interpolation algorithm. Zippel's algorithm requires randomization, and in its variable by variable interpolation method each variable is still interpolated densely.

In 1988 Michael Ben-Or and Prasoon Tiwari gave a different algorithm [1] that is based on the Berlekamp/Massey algorithm from coding theory. This algorithm interpolates all the variables at once and works equally well for sparse univariate polynomials. In its original form, it does not require randomization, but for its correctness it must be given an upper bound for the number of terms in the target polynomial.

Since then, both approaches have been generalized and improved. The Vandermonde techniques of Ben-Or and Tiwari can be applied to Zippel's algorithm [25, 14]. Lakshman Y. N., B. David Saunders, and Dima Yu. Grigoriev extended the Ben-Or/Tiwari approach to sparsity with respect to non-standard polynomial bases, such as the Pochhammer, Chebyshev, and shifted power bases [17, 11, 18]. For polynomials over small finite fields both Zippel's and Ben-Or's and

Tiwari's algorithms require modification [10, 23, and the references given there]. And both Zippel's and Ben-Or's and Tiwari's interpolation algorithms have been implemented by several authors on both single and multi-processor computers, among them [13, 20, 5].

Although a sparse algorithm takes into account the sparsity of the target polynomial, it is less efficient than a dense algorithm when the target polynomial is dense. Therefore, when the information of the sparsity in the target polynomial is not provided, we encounter the problem of choosing a more efficient algorithm.

1.2 Early termination strategy and racing algorithms

So far both the dense and sparse algorithms mentioned in Section 1.1 require a bound as an input: a degree bound for the dense algorithms, and a bound on the number of non-zero terms for the sparse algorithms. One question is easily raised: can a polynomial f still be interpolated when no such bound is supplied? Or, in other word, can we determine when to accurately finish proceeding with these algorithms even though the required bound is not given?

We can guess an input bound, compute a candidate polynomial g with respect to the guessed bound, and compare g and f at an additional random point. Whenever g cannot be successfully computed or the values of f and g are different at the additional random point, we double the guess for the bound. With this probabilistic approach, we face the problem of guessing a bound that is "efficient."

A better probabilistic approach, *early termination strategy*, is presented. It is observed that in the interpolation process, an already fully interpolated polynomial does not change at more interpolation points. Based on the observation, this strategy claims that after the interpolant stops changing at a random point, the target polynomial is interpolated with high probability. Therefore, without a bound supplied as an input, with high probability of correctness and within one single interpolation run, the early termination provides a method determining when to accurately finish proceeding with these algorithms. Austin Lobo suggested the early termination phenomenon in the setting of the Wiedemann algorithm [22].

Chapter 2 addresses the early termination strategy in dense interpolations. Chapter 3 and Chapter 4 present the early termination in sparse interpolation algorithms with respect to the power basis (in Chapter 3) and certain non-standard bases (the Pochhammer and Chebyshev bases in Chapter 4.)

Our early termination versions of algorithms are all randomized in the Monte Carlo sense, that is, their results are correct with high probability. In our implementation, we adopt another strategy of putting additional partial verification computations into some of our procedures: the early termination is only triggered after encountering a series of zero discrepancies, and passing a series of checks, in a row. The length of the series is a threshold given as an optional argument to the procedures. We prove the early termination strategy is correct with high probability 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. The further analysis is complicated and our early termination algorithms then become heuristics that can interpolate polynomials of a size at the very edge of what current software and hardware can reach.

Without a bound for the target polynomial as an input, the early termination enables us to terminate the interpolation procedure when necessary evaluations required by the algorithm have been performed. However, knowing that the sparse algorithms are less efficient when the target polynomial is dense, we are still in a predicament of choosing an efficient algorithm for a target polynomial whose sparsity is unknown.

To solve the above predicament, in the univariate case we propose *racing algorithms* that are efficient to both dense and sparse polynomials. Based upon the early termination, Chapter 5 presents how a racing algorithm "races" Newton interpolation against a sparse algorithm on a same set of evaluation points. The overall racing algorithm requires no additional evaluations, and terminates as either of the racer algorithms first terminates. In terms of black box probes, or polynomial evaluations, the overall racing algorithm is in average the more efficient algorithm with respect to the varying sparsity in different polynomials. Moreover, when the target polynomial is sparse, the probability of correctness can be further improved by utilizing the information obtained from both racer algorithms via cross-checking a sparse answer for degree consistency with the partial Newton interpolant.

1.3 Hybrids of Zippel and other improvements

Zippel's algorithm has a shortcoming over Ben-Or's and Tiwari's in that it proceeds one variable at a time, and each variable is still interpolated densely. On the other hand, when the Ben-Or/Tiwari algorithm is implemented in a modular fashion [13], in the multivariate case to our knowledge the modulus must be large enough for the recovery of all non-zero terms evaluated at prime numbers. Yet in the univariate case, there are special tricks that can reduce the size of the modulus. Also, we note that the implementation of Ben-Or's and Tiwari's algorithm with rational number arithmetic causes extreme intermediate expression swell, while in Zippel's algorithm the modulus only needs to capture the coefficients, and large enough for randomization.

In Chapter 6, we propose the hybrids of Zippel's algorithm as the following: when interpolating a multivariate polynomial, under Zippel's variable by variable scheme, each variable is interpolated through a racing algorithm that is based on early termination. Therefore, we ameliorate the inefficiency of the dense univariate interpolations embedded in the original Zippel's algorithm, and in the case of the multivariate power basis reduce the large size of the modulus required by the Ben-Or/Tiwari sparse algorithm.

Refining the idea of prunings via homogenization as described in [5], *permanent prunings* and *temporary prunings* are also presented and discussed in Chapter 6.

1.4 Maple implementation and further developments

Many of our ideas are implemented in a Maple package, *ProtoBox*, as a black box polynomial sparse interpolation over the integers modulo a prime number. Chapter 7 presents the performance of *ProtoBox* on a series of bench mark polynomials and some heuristic examples. Clearly, there is a trade-off between the additional arithmetic operations introduced by concurrently performing two univariate interpolation algorithms within the racing algorithm and the savings in the probes of the target polynomial. We intend our algorithms for polynomials produced by the calculus of black box polynomials [16, 5].

In Chapter 8, we conclude our contributions and comment on our prospective further developments.

Chapter 2

Early Termination in Dense Univariate Interpolations

Many dense interpolation algorithms have been implemented for a long time. Two widely known examples are Newton's interpolation and Lagrange's interpolation. In order to proceed, these algorithms require as input the degree of the target polynomial, or an upper bound on the degree. With high probability, early termination strategy enables such algorithms to interpolate the target polynomial without any knowledge on the degree.

Our research has been focusing on the algorithms which interpolate the target polynomial merely from its evaluations, and that no other information about the polynomial is assumed. Namely, those are the algorithms that can convert a black box representation into a polynomial representation, or perform black box polynomial interpolations. Therefore, for example, we do not consider algorithms such as the Hermite polynomial interpolation for it also requires a set of derivative values of the target polynomial.

2.1 Dense univariate interpolations

In this section, we describe the attribute of a dense univariate interpolation algorithm. When a univariate black box polynomial $f(x)$ is being interpolated from its evaluations on a sequence of distinct values $p_0, p_1, p_2, \ldots$, a dense interpolation procedure updates the i-th interpolant $f^{[i]}(x)$, $i \geq 0$, where $f^{[i]}(x)$ is a polynomial satisfies both $\deg f^{[i]}(x) \leq i$ and $f^{[i]}(p_j) = f(p_j)$ for $0 \leq j \leq i$. When i reaches the degree of the target polynomial d, the updated interpolant represents the target polynomial and the interpolation is complete.

It is observed that regardless the coefficient is zero or not, an i-th order term in $f(x)$ is being interpolated in order to update the i-th interpolant $f^{[i]}(x)$. As the target polynomial being constructed in a possibly most dense form, a dense algorithm interpolates terms of every order until the supplied degree or degree bound is reached. Two widely known dense algorithms are Newton's interpolation and Lagrange's interpolation.

In the case of Newton's interpolation, the i-th interpolant is updated from the previous interpolant:

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

where $f^{[0]}(x) = f(p_0)$, $i > 0$, and c_i the i-th divided difference.

In Newton's interpolation, the target polynomial can be viewed as being interpolated with respect to a mixed power basis: $1, (x - p_0), (x - p_0)(x - p_1), (x - p_0)(x - p_1)(x - p_2), \ldots$. Notice that once the polynomial is interpolated, that is, whenever $f^{[d]}(x) = f(x)$ for some d, the interpolant $f^{[d]}(x)$ does not change even if we keep interpolating $f(x)$ at more distinct values $p_{d+1}, p_{d+2}, p_{d+3}, \ldots$. In other word, $f^{[d+j]}(x) = f^{[d]}(x) = f(x)$ for all $j \geq 0$ and $d = \deg f(x)$.

2.2 Early termination in dense univariate interpolations

Based on the observation that an already interpolated polynomial does not changed at more interpolation points, we implement the strategy of *early termination* in a dense interpolation as the following: a positive integer η is given as a threshold, the sequence $p_0, p_1, p_2, \ldots$ are formed as random values, and the interpolant $f^{[i]}(x)$ is updated as i being increased. Whenever the interpolant $f^{[i]}(x)$ stops changing as many as η times in a row, $f(x)$ is interpolated as $f^{[i]}(x)$ with high probability. Theorem 2.1 states and proves the early termination strategy in dense interpolation algorithms.

THEOREM 2.1. **(Early Termination in Dense Univariate Interpolations)** *Given are a black box univariate polynomial $f(x)$ over a field and a positive integer η as the threshold. Let $p_0, p_1, p_2, \ldots$ be chosen randomly and uniformly from a subset S of the domain of values, and let $f^{[i]}(x)$ denote the i-th interpolant that interpolates $f(p_0), \ldots, f(p_i)$. Note that the p_i are not necessarily all distinct. If there is a non-negative integer d such that*

$$f^{[d]}(x) = f^{[d+1]}(x) = \cdots = f^{[d+\eta]}(x). \tag{2.1}$$

Then with probability no less than

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

$f^{[d]}(x)$ *correctly interpolates* $f(x)$.

Proof:

Suppose d is the smallest integer that satisfies (2.1) and if $f^{[d]}(x)$ does not correctly interpolate $f(x)$, which implies $f(x) - f^{[d-1]}(x) \neq 0$, 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 $f(x) - f^{[d]}(x)$.

By the nature of a dense interpolation, we have $\deg f^{[d]}(x) < \deg f(x)$ whenever $f^{[d]}(x) \neq f(x)$. Therefore $\deg(f(x) - f^{[d]}(x)) = \deg f(x)$, and there are at most $\deg f(x)$ many distinct roots for $f(x) - f^{[d]}(x)$. The probability of randomly generating an element from a set S that happens to be a root of $f(x) - f^{[d]}(x)$ is no more than $\deg f(x)/\#(S)$.

We define the probability function $P(i)$ as follows.

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 number 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}, p_{i+2}, …, $p_{i+\eta}$ are all roots of $f(x) - f^{[i]}(x)$.

For each $i \geq 0$, we have $P(i) \leq (\deg f(x)/\#(S))^\eta$. Because we need to at least hit a root of $f(x) - f^{[i]}(x)$ as many as η times so that p_{i+1}, …, $p_{i+\eta}$ are all roots of $f(x) - f^{[i]}(x)$.

When $f(x)$ is interpolated correctly, at most $\eta \cdot \deg f(x)$ many values can be interpolated before the target polynomial is obtained. Such a case happens only when each newly updated interpolant stops changing for exactly $\eta - 1$ many times. The sum $\sum_{i=0}^{\eta \cdot \deg f(x)-1} P(i)$ covers all the possibilities of $f(x)$ being falsely interpolated.

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

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

In Theorem 2.1, we are permissive toward the worst case in estimating the probability. Notice that whenever $f^{[i]}(x) \neq f^{[i-1]}(x)$, $f(x)$ cannot be falsely interpolated as any of the interpolants from $f^{[i+1]}(x)$, …, $f^{[i+\eta-1]}(x)$.

The size of S, which is the subset where p_i are picked, affects the probability of a correct interpolation. The following lemma shows that when $\#(S)$ is large enough, the increasing of threshold η can improve the lower bound of the probability of correctness.

LEMMA 2.1. *In theorem 2.1, the lower bound of the probability of correctness can be improved when the threshold η is increased to $\eta + \Delta\eta$ if*

$$\left(\frac{\deg f(x)}{\#(S)} \right)^{\Delta\eta} < \frac{\eta}{\eta + \Delta\eta} \tag{2.2}$$

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

Proof:

From inequality (2.2) we have

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

Which implies

$$1 - \eta \cdot \deg f(x) \cdot \left(\frac{\deg f(x)}{\#(S)}\right)^{\eta} < 1 - (\eta + \Delta\eta) \cdot \deg f(x) \cdot \left(\frac{\deg f(x)}{\#(S)}\right)^{\eta+\Delta\eta}. \quad \boxtimes$$

Notice the lower bound discussed in Lemma 2.1 does not exactly reflect how the performance being improved by a higher threshold, whose effects are evident when the size of S is relatively small (see Chapter 7 for test results.)

In Theorem 2.1, the elements in the random value sequence $p_0, p_1, \ldots$ are not necessarily all distinct. In our implementation, instead of computing $f^{[i]}(x)$ that interpolates all the first $i+1$ elements from the sequence $p_0, p_1, \ldots$, we compute the interpolant $f^{\{k\}}(x)$ that interpolates the first $k+1$ *distinct* elements from the same sequence. The interpolant $f^{\{k\}}(x)$ is only updated when a non-repeated point is introduced in the evaluation and we still have $\deg f^{\{k\}}(x) \leq k$. This modification avoids unnecessary false early terminations due to interpolating on repeated points and further improves the probability of correctness.

Without the degree or a degree bound as an input, the early termination allows a dense algorithm to proceed with interpolation merely from polynomial evaluations. In order to interpolate the target polynomial at enough distinct points, the size of the interpolation domain S needs to be large enough to construct the highest order term in the target polynomial, that is, at least $\deg(f(x)) + 1$. In the case of early termination, for passing the threshold test, $\#(S)$ needs to be at least $\deg f(x) + 1 + \eta$.

Chapter 3

Early Termination in Sparse Interpolations with respect to the Power Basis

There are interpolation algorithms whose computational complexities are sensitive to the sparsity of the target polynomials. Instead of constructing a term for each order, the *sparse interpolation algorithms* focus on the recovery of all the monomials with non-zero coefficients.

In this chapter, we describe the Ben-Or/Tiwari interpolation algorithm that is based on the Berlekamp/Massey algorithm from coding theory. This sparse algorithm recovers all the non-zero terms at once and works in the multivariate case as well as in the univariate case.

We notice the sparsity of a polynomial depends on the basis in its polynomial presentation. While the sparsity of a polynomial might change under different bases, the degree remains the same. The Ben-Or/Tiwari algorithm is a sparse interpolation algorithm with respect to a multivariate power basis. In addition to describing the Ben-Or/Tiwari algorithm, we present the early termination version of this algorithm. Later in Chapter 4, we will demonstrate the early termination strategy in sparse interpolations with respect to some non-standard bases, which can be viewed as generalizations of the Ben-Or/Tiwari algorithm in the univariate case.

3.1 The Berlekamp/Massey algorithm

Consider a vector space V over a field $\mathbb{K}$ and a sequence $\{a_i\}_{i=0}^{\infty}$ in V. $\{a_i\}_{i=0}^{\infty}$ is *linearly generated* if and only if there exist $c_0, \ldots, c_m \in \mathbb{K}$ with $c_m \neq 0$ such that for all $j \geq 0$,

$$c_0 a_j + c_1 a_{j+1} + \cdots + c_m a_{j+m} = 0.$$

Therefore, for all $j \geq 0$ we have

$$a_{j+m} = -\frac{c_{m-1}}{c_m} \cdot a_{j+m-1} - \cdots - \frac{c_0}{c_m} \cdot a_j.$$

Let z be an indeterminate, we call $c_m z^m + c_{m-1} z^{m-1} + \cdots + c_0$ the *generating polynomial* for $\{a_i\}_{i=0}^{\infty}$, and the polynomial

$$z^m + \frac{c_{m-1}}{c_m} \cdot z^{m-1} + \frac{c_{m-2}}{c_m} \cdot z^{m-2} + \cdots + \frac{c_0}{c_m}$$

its *monic associate.*

When $\{a_i\}_{i=0}^{\infty}$ is a linearly generated sequence, the set of all its generating polynomials, together with zero, form an ideal in $\mathbb{K}[z]$. Since $\mathbb{K}[z]$ is a principal ideal domain, the unique monic Λ in $\mathbb{K}[z]$ generates this domain is the *minimal polynomial* of $\{a_i\}_{i=0}^{\infty}$.

The Berlekamp/Massey algorithm [19] 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 $\Lambda(z) = z^t + \lambda_{t-1}z^{t-1} + \cdots + \lambda_0$ after processing $2t$ elements from the stream.

The stream can be unbounded, however, and the algorithm can update the current guess for the minimal polynomial appropriately whenever the next stream element a_i does not fit the current linear recursion, which occurs when a non-zero discrepancy, $\Delta \neq 0$, is detected. When the linear generator, or the minimal polynomial, is updated, either the generator jumps in degree (Step 3 below,) or the coefficients of the lower degree terms in the generator get adjusted (Step 4 below.) Notice that the algorithm computes the reverse of the minimal generating polynomial $\Lambda(z)$, that is, $\Lambda^{(rev)}(z)$.

The Berlekamp/Massey algorithm

Input:

$a_0, a_1, \ldots \in \mathbb{K}$

1. (Initialization.)

   $\Lambda_0^{(rev)} \leftarrow 1;$

   $B_0 \leftarrow 0;$

   $L_0 \leftarrow 0;$

   $\Delta \leftarrow 1;$

   For $i = 1, 2, \ldots$ ***Do***

2. (Compute the *discrepancy* Δ_i. At this point, $\Lambda_{i-1}^{(rev)}(z) = \lambda_0 z^s + \lambda_1 z^{s-1} + \cdots + \lambda_s$, where $s = \deg(\Lambda_{i-1}^{(rev)})$ and $\lambda_0, \ldots, \lambda_s \in \mathbb{K}$ with $\lambda_0 \neq 0$. Note that we always

have $\lambda_s = 1$.)

$\Delta_i \leftarrow \lambda_s a_{i-1} + \lambda_{s-1} a_{i-2} + \cdots + \lambda_0 a_{i-s-1}$;

If $\Delta_i = 0$ *then*

$\Lambda_i^{(rev)} \leftarrow \Lambda_{i-1}^{(rev)}$;

$B_i \leftarrow z \cdot B_{i-1}$;

$L_i \leftarrow L_{i-1}$;

3. *If* $\Delta_i \neq 0$ *and* $2L_{i-1} < i$ *then*

   $B_i \leftarrow \Lambda_{i-1}^{(rev)}$;

   $\Lambda_i^{(rev)} \leftarrow \Lambda_{i-1}^{(rev)} - (\Delta_i/\Delta) \cdot z \cdot B_{i-1}$;

   $L_i \leftarrow i - L_{i-1}$;

   $\Delta \leftarrow \Delta_i$;

4. *If* $\Delta_i \neq 0$ *and* $2L_{i-1} \geq i$ *then*

   $\Lambda_i^{(rev)} \leftarrow \Lambda_{i-1}^{(rev)} - (\Delta_i/\Delta) \cdot z \cdot B_{i-1}$;

   $B_i \leftarrow z \cdot B_{i-1}$;

   $L_i \leftarrow L_{i-1}$;

End For;

End.

3.2 The Ben-Or/Tiwari interpolation algorithm

Let f be a multivariate polynomial over a field, t the number of its terms with non-zero coefficients. Suppose $c_j \neq 0$ are the coefficients of the non-zero terms

$\beta_j(x_1, \ldots, x_n) = x_1^{e_{j,1}} x_2^{e_{j,2}} \cdots x_n^{e_{j,n}}$, then

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

Suppose $p_1, p_2, \ldots, p_n$ are distinct primes from the corresponding domains, and b_j the evaluations of β_j at $(p_1, p_2, \ldots, p_n)$, namely, $b_j = \beta_j(p_1, p_2, \ldots, p_n) = p_1^{e_{j,1}} \cdots p_n^{e_{j,n}}$. We have $b_i \neq b_j$ whenever $i \neq j$. The evaluations of β_j at the powers of $(p_1, p_2, \ldots, p_n)$ can be viewed as the powers of b_j:

$$\beta_j(p_1^i, p_2^i, \ldots, p_n^i) = (p_1^i)^{e_{j,1}} \cdots (p_n^i)^{e_{j,n}} = (p_1^{e_{j,1}} \cdots p_n^{e_{j,n}})^i = b_j^i.$$

Evaluate f at a sequence of powers of $(p_1, p_2, \ldots, p_n)$ and define $a_i = f(p_1^i, \ldots, p_n^i)$, we have

$$a_i = f(p_1^i, \ldots, p_n^i) = \sum_{j=1}^{t} c_j b_j^i.$$

Consider an auxiliary polynomial $\Lambda(z)$ defined as the following:

$$\Lambda(z) = \prod_{j=1}^{t} (z - b_j) = z^t + \lambda_{t-1} z^{t-1} + \cdots + \lambda_0. \tag{3.1}$$

By definition, the roots of $\Lambda(z)$ are b_j, which are the non-zero terms in f evaluated at $(p_1, p_2, \ldots, p_n)$. When f is a polynomial and $a_i = f(p_1^i, \ldots, p_n^i)$, Theorem 3.1 shows that the sequence $\{a_i\}_{i \geq 0}$ is linearly generated and the auxiliary polynomial $\Lambda(z)$ is its minimal generating polynomial.

THEOREM 3.1. *Let* $f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{e_{j,1}} \cdots x_n^{e_{j,n}}$, *where* $c_j \neq 0$, *and* $a_i = f(p_1^i, \ldots, p_n^i)$ *for* $i \geq 0$, p_i *distinct primes. The sequence of* $\{a_i\}_{i \geq 0}$ *is linearly generated by the polynomial* $\Lambda(z)$. *Furthermore,* $\Lambda(z)$ *is the minimal polynomial* *of* $\{a_i\}_{i\geq 0}$.

Proof:

As defined in (3.1), $\Lambda(z) = \prod_{j=1}^{t}(z-b_j)$, we have $\Lambda(b_j) = 0$ for $j = 1 \ldots t$. And for any integer $i \geq 0$ we have

$$\begin{aligned}
0 &= \sum_{j=1}^{t} c_j b_j^i \Lambda(b_j) = \sum_{j=1}^{t} c_j b_j^i (b_j^t + \lambda_{t-1} b_j^{t-1} + \cdots + \lambda_0) \\
&= (c_1 b_1^{i+t} + \cdots + c_t b_t^{i+t}) + \sum_{k=0}^{t-1} \lambda_k (c_1 b_1^{i+k} + \cdots + c_t b_t^{i+k}) \\
&= a_{t+i} + \sum_{k=0}^{t-1} \lambda_k a_{k+i}.
\end{aligned}$$

Thus, $\Lambda(z)$ generates $\{a_i\}_{i\geq 0}$.

Consider the Hankel matrix:

$$A_t = \begin{bmatrix} a_0 & a_1 & \ldots & a_{t-1} \\ a_1 & a_2 & \ldots & a_t \\ \vdots & \vdots & \ddots & \vdots \\ a_{t-1} & a_{t-2} & \ldots & a_{2t-2} \end{bmatrix}.$$

If $\Lambda(z)$ is not the minimal polynomial of $\{a_i\}_{i\geq 0}$, then there exists at least one column that is a linear combination of other columns within matrix A_t and $\det(A_t) = 0$. In order to prove $\Lambda(z)$ is the minimal polynomial, we show $\det(A_t)$

$\neq 0$ by factoring A_t into non-singular matrices,

$$A_t = B_t \underbrace{\begin{bmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_t \end{bmatrix}}_{C_t} B_t^{\mathrm{Tr}},$$

where

$$B_t = \begin{bmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \dots & b_t^{t-1} \end{bmatrix}.$$

Since whenever $i \neq j$ we have $b_i \neq b_j$, the transposed Vandermonde matrix B_t is non-singular. The matrix C_t is also non-singular due to all the nonzero diagonal c_i. ⊠

In Section 3.1, we give the Berlekamp/Massey algorithm that can compute $\Lambda(z)$ from $\{a_i\}_{i\geq 0}$. As a consequence of the definition in (3.1), b_j can be obtained by way of finding the roots of $\Lambda(z)$. If p_i are different primes and the evaluations are done over a coefficient field of characteristic zero, then each term $\beta_j = x_1^{e_{j,1}} \cdots x_n^{e_{j,n}}$ can be recovered through repeatedly dividing b_j by $p_1, \dots, p_n$. The coefficients c_j can be determined 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:

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_t \\ b_1^2 & b_2^2 & \dots & b_t^2 \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \dots & b_t^{t-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{t-1} \end{bmatrix}. \tag{3.2}$$

Efficient algorithms for solving transposed Vandermonde systems can be found in [14, 25].

The Ben-Or/Tiwari interpolation algorithm

Input:

f**:** *a multivariate black box polynomial.*

τ**:** $\tau \geq t$*,* t *is the number of the terms with non-zero coefficients in* f*.*

Output:

c_j *and* β_j: $f = \sum_{j=1}^{t} c_j \beta_j$ *and* $c_j \neq 0$*.*

1. (The Berlekamp/Massey algorithm.)

   $a_i = f(p_1^i, \dots, p_n^i), 0 \leq i \leq 2\tau - 1$, *where* p_i *are distinct primes.*

   Compute $\Lambda(z)$ *from* $\{a_i\}_{2\tau-1 \geq i \geq 0}$*.*

2. (Determine β_j.)

   Find all t *distinct roots of* $\Lambda(z)$*, which are* b_j*.*

   Determine each β_j *through repeatedly dividing every* b_j *by* $p_1, \dots, p_n$*.*

3. (Compute the coefficients c_j.)

   Solve a transposed Vandermonde system as in (3.2).

End.

We use an example to elucidate the steps of the Ben-Or/Tiwari algorithm.

Example 3.1. *Suppose $p_1 = 2$, $p_2 = 3$, and we interpolate the polynomial $f = x^{12} + 5x^3y + y^4 - 3$.*

The sequence $\{a_i\}_{i\geq 0}$, where $a_i = f(2^i, 3^i)$, is formed as the following sequence of values: {4, 4294, 16786654, 68720077294, 281475021416254, 1152921508133444494, 4722366483153030265054, 19342813113856966520110894, ...}. *Since there are four non-zero terms in f, after performing the Berlekamp/ Massey algorithm on the first eight values of this sequence, the auxiliary polynomial $\Lambda(z) = z^4 - 4202z^3 + 436225z^2 - 8394648z + 7962624$ is obtained.*

All the terms in f are determined from the roots of $\Lambda(z)$:

$$\Lambda(z) = (z-1)(z-24)(z-81)(z-4096),$$

where $1 = 2^0 \cdot 3^0 \to x^0y^0$, $24 = 2^3 \cdot 3 \to x^3y$, $81 = 3^4 \to y^4$, and $4096 = 2^{12} \to x^{12}$.

The polynomial f is fully interpolated after the corresponding coefficients c_i are computed by solving the transposed Vandermonde system,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 24 & 81 & 4096 \\ 1^2 & 24^2 & 81^2 & 4096^2 \\ 1^3 & 24^3 & 81^3 & 4096^3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} a_0 = 4 \\ a_1 = 4294 \\ a_2 = 16786654 \\ a_3 = 68720077294 \end{bmatrix},$$

with solution

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} -3 \\ 5 \\ 1 \\ 1 \end{bmatrix}. \quad \boxtimes$$

When the Ben-Or/Tiwari algorithm is implemented in a modular fashion, ignoring the size of the coefficients, the modulus must be large enough for the recovery of all the terms, that is, no less than $p_1^{d_1} p_2^{d_2} \cdots p_n^{d_n}$, where n is the number of variables, d_i the degrees in different variables such that $d_1 \geq d_2 \geq \cdots \geq d_n$, and p_i the i-th prime. Such an exponentially increasing figure can be perceived in Example 3.1.

The modulus can be reduced if we only interpolate few variables at a time through the Ben-Or/Tiwari algorithm. Then we keep interpolating the partially interpolated result with respect to a subset from the yet-interpolated variables until the target polynomial is fully interpolated. Certainly, there is a trade-off between the reduced modulus and the additional interpolations for every term that consists of variables from the different subsets. Since such a term needs to be interpolated with respect to all the subsets that contain a variable in the term.

In the case of interpolating one variable at a time, in order to recover all the terms, instead of a modulus larger than 2^{d_1}, where d_1 is the maximum among the degrees in different variables, there are tricks reducing the modulus to a prime only larger than d_1 (see Section 6.3.) Similarly, the trade-off is between the reduced modulus and the additional interpolations for every multivariate term. Because a multivariate term needs to be interpolated with respect to each variable involved.

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

Both the Ben-Or/Tiwari [1] and the Kaltofen *et al.* [13] algorithms need to know the number of non-zero terms t, or an upper bound τ on t, $\tau \geq t$. Otherwise, we can guess t, if there is no failure in finding the roots of Λ, compute a sparse candidate polynomial g for f, and compare g and f at an additional random point. If the values are different, or it fails in finding the roots of Λ, we can double the guess for t. This scheme is randomized in the Monte Carlo sense.

Based on the early termination strategy, we present a more efficient probabilistic approach, which requires only one single interpolation run. The idea is simple: pick a random point coordinates p for the Ben-Or/Tiwari algorithm and show that with high probability the embedded Berlekamp/Massey algorithm of Section 3.1 does not encounter a zero discrepancy Δ at $i > 2L$[1] until $i > 2t$ (in Step 2 for the case that $i > 2L$, i.e., by which would be divided in Step 3 if the discrepancy were non-zero.)

However, this is not generally true: for any polynomial $f(x) = \sum_{j=1}^{t} c_j x^{e_j}$ satisfying $f(p^0) = a_0 = c_1 + \cdots + c_t = 0$, we always have the first discrepancy $\Delta_1 = 0$. We suggest two schemes to fix this problem: either pick an additional random value p_c, so that we can claim $f(p^0) + p_c$ is non-zero with high probability and then proceed the interpolation with $f + p_c$ (see Section 4.4 for the application in the sparse interpolation under the Chebyshev basis;) or, through shifting the sequence by 1 element, the early termination property can be proved as the following.

[1] Clearly, the shift register length can be stored in a single integer variable without a subscript.

We first show that for symbolic values, or variables $x_1, \ldots, x_n$, the first zero discrepancy for $i > L$ appears at $i = 2t + 1$. Let $\beta_j = x_1^{e_{j,1}} \cdots x_n^{e_{j,n}}$ be the j-th non-zero term in f, and $\alpha_i = f(x_1^i, \ldots, x_n^i)$, we have

$$\mathcal{A}_i = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_i \\ \alpha_2 & \alpha_3 & \ldots & \alpha_{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_i & \alpha_{i+1} & \ldots & \alpha_{2i-1} \end{bmatrix} = \mathcal{B}_i C_t \bar{\mathcal{B}}_i^{\mathrm{Tr}}, \tag{3.3}$$

where

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

and

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

The singularity of the Hankel matrix $\mathcal{A}_i$ in (3.3) is directly related to the vanishing of discrepancies when computing a linear generator of $\alpha_1, \alpha_2, \ldots$ through the Berlekamp/Massey algorithm (Section 3.1.) The argument makes use of the interpretation of the Berlekamp/Massey algorithm as the extended Euclidean algorithm on the polynomials $F_{-1} = X^N$ and $F_0 = \alpha_1 X^{N-1} + \alpha_2 X^{N-2} + \cdots$ [6] combined with the fundamental theorem on subresultants [2]. Here N is the number of elements that are considered for determining the linear generator. Dornstetter shows that Δ_i in Step 3 of the Berlekamp/Massey algorithm of Section 3.1 is the leading coefficient in a remainder, F_i, in a polynomial remainder sequence (PRS) of F_{-1} and F_0. The discrepancies in Step 4 are the trailing coefficients in F_i. Step 2 processes both trailing coefficients that are zero and the search for the non-zero leading coefficient of the next remainder. The former can be diagnosed by the shift-register length $2L \geq i$. The remainder polynomials F_i in the PRS, which yields the polynomials B_i and Λ_i at $2L = i$ as the reverse polynomials of the consecutive Bezout coefficients T_{i-1} and T_i in the extended Euclidean scheme $F_i = S_i F_{-1} + T_i F_0 \equiv T_i F_0 \pmod{X^N}$, are adjusted by non-zero scalar multipliers, namely $-\Delta_i/\Delta$ of Step 3. Dornstetter's analysis yields the following fact.

FACT 1. *If the PRS is normal, i.e.,* $\deg(F_i) = \deg(F_{i-1}) - 1 = N - i - 1$, *where* $i \geq 0$, *then* $\Delta_i \neq 0$ *whenever* $2L < i$ *and* $L < t$, *where* t *is the degree of the linear generator.*

Appealing now to the fundamental theorem of subresultants, the PRS is normal if and only if the leading coefficient of the $N-i-1$'st subresultant of F_{-1} and F_0 does not vanish. By definition in [2], this is the determinant of a $(2i+1)\times(2i+1)$ matrix shown in Figure 3.1.

From the proof of Theorem 3.1, we always have $\det(A_i) = 0$ for all $i > t$. The early termination strategy is correct if the determinant of $\mathcal{A}_i$ is non-zero for $i = 1 \ldots t$ in symbolic evaluations. This is our next theorem.

$$
\det \begin{bmatrix}
1 & 0 & \cdots & & 0 & & & 0 \\
& \ddots & & & & \ddots & & \vdots \\
& & 1 & & & & & 0 \\
& & & \ddots & & & & \vdots \\
0 & & \cdots & & 1 & 0 & \cdots & 0 \\
\alpha_1 & & & \cdots & \alpha_i & \alpha_{i+1} & & \alpha_{2i+1} \\
& \ddots & & & & & & \\
& & \alpha_1 & & & \vdots & \ddots & \vdots \\
& & & \ddots & & & & \\
& & & & \alpha_1 & & & \\
0 & & & & 0 & \alpha_1 & \cdots & \alpha_{i+1}
\end{bmatrix} = \pm \det(\mathcal{A}_{i+1}).
$$

Figure 3.1: Subresultant coefficient

THEOREM 3.2. *The determinant of $\mathcal{A}_i$ is non-zero for $i = 1, \ldots, t$.*

Proof:

Let $M_{J,K}$ be the determinant of the submatrix of M consisting of the rows listed in the set J and the columns listed in the set K. The Binet-Cauchy formula [8] states for the determinant of a matrix product AB that

$$
(AB)_{J,L} = \sum_{1 \le k_1 < k_2 < \cdots < k_i \le n} A_{J,\{k_1,\ldots,k_i\}} B_{\{k_1,\ldots,k_i\},L}, \tag{3.4}
$$

where n is the number of columns of A, and J and L are sets of row and column indices with i elements each.

Applying (3.4) to (3.3) with $I = \{1, \ldots, i\}$, the determinant of $\mathcal{A}_i$ is a polynomial in β_j as (3.5). The rest of our proof shows $\mathcal{A}_i$ is non-singular for every $i = 1, \ldots, t$ since every such polynomial is a non-zero polynomial.

$$
\begin{aligned}
\det(\mathcal{A}_i) &= (\mathcal{B}_i C_t \bar{\mathcal{B}}_i^{\mathrm{Tr}})_{I,I} \\
&= \sum_J \sum_K (\mathcal{B}_i)_{I,J} (C_t)_{J,K} (\bar{\mathcal{B}}_i^{\mathrm{Tr}})_{K,I} \\
&= \sum_J (\mathcal{B}_i)_{I,J} (C_t)_{J,J} (\bar{\mathcal{B}}_i^{\mathrm{Tr}})_{J,I} \\
&= \sum_{J=\{j_1,\ldots,j_i\}} c_{j_1} \cdots c_{j_i}\, \beta_{j_1}\beta_{j_2}\cdots\beta_{j_i} \cdot \det\left(\begin{bmatrix} 1 & 1 & \ldots & 1 \\ \beta_{j_1} & \beta_{j_2} & \cdots & \beta_{j_i} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{j_1}^{i-1} & \beta_{j_2}^{i-1} & \ldots & \beta_{j_i}^{i-1} \end{bmatrix} \right)^2 \\
&= \sum_{J=\{j_1,\ldots,j_i\}} c_{j_1} \cdots c_{j_i}\, \beta_{j_1}\beta_{j_2}\cdots\beta_{j_i} \cdot \prod_{1\le v<u\le i} (\beta_{j_u} - \beta_{j_v})^2. \qquad (3.5)
\end{aligned}
$$

Now let the terms $\beta_1 \succ \beta_2 \succ \cdots \succ \beta_t$ be ordered lexicographically. Then 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] hence $\det(\mathcal{A}_i)$ does not vanish. $\boxtimes$

We make the transition from symbolic point coordinates $x_1, \ldots, x_n$ to random field elements $p_1, \ldots, p_n$ in the customary fashion via the the Schwartz-Zippel lemma [24, 21], also see [4].

Theorem 3.3. *If $p_1, \ldots, p_n$ are chosen randomly and uniformly from a subset*

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

S of the domain of values, 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 algorithms encounters $\Delta = 0$ and $i > 2L$ the first time at $i = 2t + 1$ 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 (3.5) we obtain $\deg(\det \mathcal{A}_i) \leq i^2 \deg(f)$. We have to avoid all the possible zeroes of the product $\prod_{i=1}^{t} \det \mathcal{A}_i$, whose degree is no more than $t(t+1)(2t+1)\deg(f)/6$. The estimate of the probability follows from Lemma 1 in [21]. $\boxtimes$

The estimate in Theorem 3.3 is, like the Zippel-Schwartz estimate, somewhat pessimistic. The following argument attempts to shed further light on the situation. Over a finite field of q elements we may choose the set S to be the entire field, that is, $q = \#(S)$. If we make the assumption 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. [15]; 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, we therefore make a further modification: the user can supply a threshold $\zeta \geq 1$. Then early termination strategy requires ζ many times in a row the zero discrepancies with $i > 2L$. Clearly, for a random a_i then there are more acceptable A_t, where A_i are matrices $\mathcal{A}_i$ evaluated at $\alpha_j = a_j$. The precise analysis on the effects of a threshold $\zeta > 1$ depends on the conditional probabilities $P(\det(A_{i+1}) = 0 | \det(A_i) = 0)$ for $i \geq 1$. Nevertheless, in Chapter 7, we test and demonstrate heuristic examples for some small moduli with thresholds larger than 1.

The early termination version of the Ben-Or/Tiwari algorithm does not need an upper bound on the numbers of non-zero terms as an input.

The Ben-Or/Tiwari algorithm with early termination

Input:

f**:** *a multivariate black box polynomial.*

ζ**:** *a positive integer, the threshold for early termination.*

Output:

c_j *and* β_j*:* $f = \sum_{j=1}^{t} c_j\beta_j$ *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: $p_1, \ldots, p_n$*, and* $p_j \notin \{0, 1\}$.

   For $i = 1, 2, \ldots$

   Perform the Berlekamp/Massey algorithm on $\{f(p_1^j, \ldots, p_n^j)\}_{1 \leq j \leq i}$.

   If $\Delta_i = 0$ *and* $i > 2L$ *happens* ζ *many times in a row, then*

   break out of the loop;

   set $\Lambda(z)$ *to the reverse of* $\Lambda_r^{(rev)}(z)$ *that was computed inside the algorithm;*

2. (Determine β_j.)

   Compute all the roots of $\Lambda(z)$ *in the domain of* p_i.

   If $\Lambda(z)$ *does not completely factor, or not all the roots are distinct, then* *the early termination was false.*

Otherwise, determine the terms β_j *from the roots* b_j*:*

repeatedly divide b_j *by* $p_1, \ldots, p_n$*. Again, the term recovery might fail for unlucky* p_i*.*

3. (Determine c_j.)

Recover the coefficients c_j *of the corresponding terms* β_j:

solve a transposed Vandermonde system as in (3.2).

End.

Remark: If the coefficient field is a subfield of the 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:

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

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.

Chapter 4

Early Termination in Sparse Interpolations with respect to Non-standard Bases

As we have noticed, while the order of a polynomial remains the same, the bases in the polynomial representations define the sparsity of a polynomial. Therefore, a sparse interpolation algorithm depends on the construction of the designate basis in which the target polynomial is being interpolated.

As generalizations of the Ben-Or/Tiwari algorithm in the univariate case, Lakshman and Saunders [17] proposed sparse interpolation algorithms in the Pochhammer and Chebyshev bases. In this chapter, we will show the early termination versions of these algorithms.

4.1 Univariate sparse interpolations in the Pochhammer basis

For any integer $n \geq 0$, the Pochhammer symbol $x^{\overline{n}}$ denotes the rising factorial power

$$x(x+1)\dots(x+n-1).$$

A univariate polynomial $f(x)$ over a field $\mathbb{K}$ is t-sparse in the Pochhammer basis if and only if for $j = 1, \dots, t$, $c_j \neq 0$ and $c_j \in \mathbb{K}$, $e_j \in \mathbb{Z}_{\geq 0}$, such that

$$f(x) = \sum_{j=1}^{t} c_j x^{\overline{e}_j}, e_1 < e_2 < \cdots < e_t.$$

For $k = 0, 1, \dots$, define $f^{(k)}(x)$ as the following:

$$f^{(k)}(x) = \sum_{j=1}^{t} e_j^k c_j x^{\overline{e}_j}. \tag{4.1}$$

The finite difference operator $\Delta(f(x)) = f(x+1) - f(x)$ behaves like the derivative on the Pochhammer symbol: that is, $\Delta(x^{\overline{k}}) = (x+1)^{\overline{k}} - x^{\overline{k}} = k(x+1)^{\overline{k-1}}$ and

$$x \cdot \Delta(f^{(k)}(x)) = f^{(k+1)}(x). \tag{4.2}$$

Therefore, $f^{(k)}(p)$ with $k = 0, 1, \dots, 2t-1$ can be computed by repeatedly applying the recurrence from the subsequent polynomial evaluations $f(p+k)$, $k = 0, 1, \dots, 2t-1$.

Lemma 1 in [17] shows that the sequence $\{f^{(k)}(p)\}_{2t-1 \geq k \geq 0}$ is linearly generated by an auxiliary polynomial $\Lambda(z)$ defined as:

$$\Lambda(z) = \prod_{j=1}^{t}(z - e_j) = \lambda_t z^t + \lambda_{t-1} z^{t-1} + \cdots + \lambda_1 z + \lambda_0. \tag{4.3}$$

Which means that for $t-1 \geq k \geq 0$, we have $\sum_{j=0}^{t} \lambda_j f^{(j+k)}(p) = 0$. Furthermore, Theorem 1 in [7] shows the following $t \times t$ matrix is non-singular for any $p > 0$:

$$\begin{bmatrix} f^{(0)}(p) & f^{(1)}(p) & \dots & f^{(t-1)}(p) \\ f^{(1)}(p) & f^{(2)}(p) & \dots & f^{(t)}(p) \\ \vdots & \vdots & \ddots & \vdots \\ f^{(t-1)}(p) & f^{(t)}(p) & \dots & f^{(2t-2)}(p) \end{bmatrix}. \tag{4.4}$$

Lakshman and Saunders gave the following univariate sparse interpolation in the Pochhammar basis [17].

The sparse interpolation in the Pochhammer basis

Input:

$f(x)$**:** *a univariate black box polynomial.*

t**:** *the number of non-zero terms in* 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 the evaluations* $f(p+k)$*, 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 $\Lambda(z)$, which are the Pochhammer term exponents in $f(x)$.

3. (Compute the coefficients c_j.)

   Solve a transposed Vandermonde system:

   $c_j p^{\bar{e}_j}$ *can be obtained from solving a transposed Vandermonde system.*

   Compute c_j from $c_j p^{\bar{e}_j}$ since p and e_j are known.

End.

4.2 Early termination of sparse interpolations in the Pochhammer basis

The sparse interpolation in the Pochhammer basis [17] needs an input t as the number of non-zero Pochhammer terms in the target polynomial $f(x)$. Then in order to form the sequence $\{f^{(k)}(p)\}_{0\leq k\leq 2t-1}$, $f(x)$ is queried as many as $2t$ times at $p+k$, where $k = 0 \ldots 2t-1$ and $p > 0$.

Notice that during the evaluation step: rather than a sequence of powers of a value p as in the Ben-Or/Tiwari algorithm, $f(x)$ is evaluated at a sequence of subsequent numbers of a positive value p, which are p, $p+1$, ... The difference in the sequences of evaluation points reflects the construction of the varying bases.

Recall that $\Lambda(z)$ in (4.3) generates $\{f^{(k)}(p)\}_{2t-1\geq k\geq 0}$. In order to implement early termination in the sparse interpolation under the Pochhammer basis, we need to show $\Lambda(z)$ generates $\{f^{(k)}(p)\}_{k\geq 0}$ as well.

THEOREM 4.1. *For $p > 0$, $\Lambda(z)$ generates $\{f^{(k)}(p)\}_{k\geq 0}$.*

Proof:

From Lemma 1 in [17], we have

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

By (4.2) and the induction on k,

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

Now, if $\Lambda(z)$ is the minimal polynomial of $\{f^{(k)}(p)\}_{k\geq 0}$, $\Lambda(z)$ can be obtained through the Berlekamp/Massey algorithm. Similar to the discussion in Section 3.3, the occurrence of a zero discrepancy at $i > 2L$ on sequence $\{f^{(k)}(x)\}_{k\geq 0}$ is directly related to the singularity of the following Hankel matrix:

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

Theorem 4.2. *The determinant of $\mathcal{A}_k$ is nonzero for $k = 1, \dots, t$.*

Proof:

Recall the definition of $f^{(k)}(x)$ in (4.1) for $2t - 2 \geq k \geq 0$. The following factorizations of $\mathcal{A}_k$ can be verified via matrix multiplications:

$$\begin{aligned}
\mathcal{A}_k &= \begin{bmatrix} f^{(0)}(x) & f^{(1)}(x) & \dots & f^{(k-1)}(x) \\ f^{(1)}(x) & f^{(2)}(x) & \dots & f^{(k)}(x) \\ \vdots & \vdots & \ddots & \vdots \\ f^{(k-1)}(x) & f^{(k)}(x) & \dots & f^{(2k-2)}(x) \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & \dots & 1 \\ e_1^1 & e_2^1 & \dots & e_t^1 \\ \vdots & \vdots & \ddots & \vdots \\ e_1^{k-1} & e_2^{k-1} & \dots & e_t^{k-1} \end{bmatrix} \begin{bmatrix} c_1 x^{\bar{e}_1} & 0 & \dots & 0 \\ 0 & c_2 x^{\bar{e}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_t x^{\bar{e}_t} \end{bmatrix} \begin{bmatrix} 1 & e_1^1 & \dots & e_1^{k-1} \\ 1 & e_2^1 & \dots & e_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e_t^1 & \dots & e_t^{k-1} \end{bmatrix} \\
&= \mathcal{B}_k C_t \mathcal{B}_k^{\mathrm{Tr}}. \qquad (4.5)
\end{aligned}$$

Apply the Binet-Cauchy formula [8] in (3.4) to (4.5) with $K = \{1,\ldots,k\}$ we have

$$\begin{aligned}
\det(\mathcal{A}_k) &= (\mathcal{B}_k C_t \mathcal{B}_k^{\mathrm{Tr}})_{K,K} \\
&= \sum_J \sum_L (\mathcal{B}_k)_{K,J} (C_t)_{J,L} (\mathcal{B}_k^{\mathrm{Tr}})_{L,K} \\
&= \sum_J (\mathcal{B}_k)_{K,J} (C_t)_{J,J} (\mathcal{B}_k^{\mathrm{Tr}})_{J,K} \\
&= \sum_{J=\{j_1,\ldots,j_k\}} c_{j_1} \cdots c_{j_k}\, x^{\bar{e}_{j_1}} x^{\bar{e}_{j_2}} \cdots x^{\bar{e}_{j_k}} \cdot \det\left(\begin{bmatrix} 1 & 1 & \dots & 1 \\ e_{j_1}^1 & e_{j_2}^1 & \dots & e_{j_k}^1 \\ \vdots & \vdots & \ddots & \vdots \\ e_{j_1}^{k-1} & e_{j_2}^{k-1} & \dots & e_{j_k}^{k-1} \end{bmatrix} \right)^2 \\
&= \sum_{J=\{j_1,\ldots,j_k\}} c_{j_1} \cdots c_{j_k}\, x^{\bar{e}_{j_1}} x^{\bar{e}_{j_2}} \cdots x^{\bar{e}_{j_k}} \cdot \prod_{1 \le v < u \le k} (e_{j_u} - e_{j_v})^2. \qquad (4.6)
\end{aligned}$$

In polynomial (4.6), the highest order term $c_t \cdots c_{t-k+1}\, x^{\overline{e_t}} \cdots x^{\overline{e}_{t-k+1}} \prod_{1 \le v < u \le k}(e_u - e_v)^2$ appears nowhere else, hence the polynomial does not vanish and $\det(\mathcal{A}_k)$ is nonzero for $t \ge k \ge 1$. $\boxtimes$

From Theorem 4.2 and the following lemma, we conclude that with high probability when performing the Berlekamp/Massey algorithm on $\{f^{(k)}(p)\}_{k\ge 0}$ a zero discrepancy at $i > 2L$ occurs the first time when $i = 2t+1$ (that is, at $f^{(2t)}(p)$.)

Lemma 4.1. $\det(\mathcal{A}_k) = 0$ *if and only if* $k > t$.

Proof:

For $k = 1, \ldots, t$, $\det(\mathcal{A}_k) \ne 0$, therefore $\det(\mathcal{A}_k) = 0$ implies $k > t$.

Since $\Lambda(z)$ generates $\{f^{(k)}(x)\}_{k \ge 0}$, when $k > t$, the k-th row (or column) of $\mathcal{A}_k$ is a linear combination of $(k-t)$-th through $(k-1)$-th rows (or columns.) Therefore, $\det(\mathcal{A}_k) = 0$ when $k > t$. $\boxtimes$

Again, following the transition from a symbolic point x to a random value p via the the Schwartz-Zippel lemma [24, 21] (also see [4],) the early termination strategy is implemented in the univariate sparse interpolation under the Pochhammer basis.

Theorem 4.3. *Let S be a subset of the domain of values, which is assumed to be an integral domain, and that all the elements in S are positive. If p is chosen randomly and uniformly from S, suppose*

$$f^{(k)}(x) = \sum_{j=1}^{t} e_j^k c_j x^{\overline{e}_j},$$

where $f(x) = \sum_{j=1}^{t} c_j x^{\overline{e}_j}$ and $x^{\overline{e}_j}$ the Pochhammer terms. Then on the sequence of $\{f^{(k)}(p)\}_{k\ge 0}$, the Berlekamp/Massey algorithm encounters $\Delta = 0$ and $i > 2L$ the *first time at $i = 2t+1$ 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:

Through (4.6), we have $\deg(\det \mathcal{A}_k) \leq \sum_{j=0}^{k-1} \big(\deg(f) - j\big) = k\deg(f) + k/2 - k^2/2$. We need to avoid all the possible zeroes in the product $\prod_{k=1}^{t} \det \mathcal{A}_k$, whose degree is no more than

$$\sum_{k=1}^{t} \left(k\deg(f) + \frac{k}{2} - \frac{k^2}{2}\right) = \frac{t(t+1)(3\deg(f)+1-t)}{6 \cdot \#(S)}.$$

The estimate of the probability follows from Lemma 1 in [21]. $\boxtimes$

As a similar modification to the early termination of the Ben-Or/Tiwari algorithm in Section 3.3, to improve the probability of correctness, we introduce threshold ζ to our early termination version of the sparse interpolation in the Pochhammer basis. The early termination is only triggered after a zero discrepancy with $i > 2L$ occurs ζ many times in a row. Again, the analysis of the probability with higher thresholds requires further investigations on the conditional probabilities $P(\det(A_{k+1}) = 0 | \det(A_k) = 0)$, where A_k are $\mathcal{A}_k$ evaluated at $x = p$.

The sparse interpolation with early termination in the Pochhammer basis

Input:

$f(x)$**:** *a univariate black box polynomial.*

ζ**:** *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 the sequence $\{f^{(k)}(p)\}_{i-1 \geq k \geq 0}$, where $f^{(k)}(p)$ are computed from the evaluations $f(p+k)$ for $k = 0, \ldots, i-1$.

   If both $\Delta_i = 0$ and $i > 2L$ happen ζ many times in a row, then

   break out of the loop;

   set $\Lambda(z)$ to the reverse of $\Lambda_r^{(rev)}(z)$ that was computed inside the algorithm.

2. (Determine e_j.)

   Compute all the roots of $\Lambda(z)$.

   If $\Lambda(z)$ does not completely factor, or not all the roots are distinct non-negative integers, then

   the early termination was false;

   else,

   the roots are e_j, the exponents of Pochhammer terms in $f(x)$.

3. (Compute the coefficients c_j.)

   Solve a transposed Vandermonde system:

   $c_j p^{\overline{e}_j}$ can be obtained from solving a transposed Vandermonde system.

   Compute c_j from $c_j p^{\overline{e}_j}$ since p and e_j are known.

End.

4.3 Univariate sparse interpolations in the Chebyshev basis

Let $T_i(x)$ denote the i-th Chebyshev polynomial of the first kind:

$$T_0(x) = 1, T_1(x) = x, T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x) \text{ for } i \geq 2.$$

A polynomial $f(x)$ over a field $\mathbb{K}$ is t-sparse in the Chebyshev basis if and only if for $j = 1, \ldots, t$, $c_j \neq 0$ and $c_j \in \mathbb{K}$, $\delta_j \in \mathbb{Z}_{\geq 0}$, such that

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

Consider for some $p > 1$, $a_k = f(T_k(p))$ for $k = 0, 1, \ldots, 2t-1$, and define an auxiliary polynomial $\Lambda(z)$ of degree t as the following:

$$\begin{aligned}\Lambda(z) &= \prod_{j=1}^{t}(z - T_{\delta_i}(p)) = \lambda_t T_t(z) + \lambda_{t-1}T_{t-1}(z) + \cdots + \lambda_0 T_0(z) \\ &= T_t(z) + \lambda_{t-1}T_{t-1}(z) + \cdots + \lambda_0 T_0(z).\end{aligned} \tag{4.7}$$

Lakshman and Saunders showed that for $i \geq 0$, we have the following linear relations (see Lemma 5 in [17]:)

$$\sum_{j=0}^{t-1} \lambda_j (a_{j+i} + a_{|j-i|}) = -(a_{t+i} + a_{|t-i|}). \tag{4.8}$$

The linear relations (4.8) form the following system of equations:

$$\begin{bmatrix} 2a_0 & 2a_1 & \dots & 2a_{t-1} \\ 2a_1 & a_2 + a_0 & \dots & a_t + a_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{t-1} & a_t + a_{t-2} & \dots & a_{2t-2} + a_0 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} 2a_t \\ a_{t+1} + a_{t-1} \\ \vdots \\ a_{2t-1} + a_1 \end{bmatrix}. \tag{4.9}$$

Consider the $t \times t$ symmetric Hankel-plus-Toeplitz matrix A_t in (4.9):

$$A_t = \begin{bmatrix} 2a_0 & 2a_1 & \dots & 2a_{t-1} \\ 2a_1 & a_2 + a_0 & \dots & a_t + a_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{t-1} & a_t + a_{t-2} & \dots & a_{2t-2} + a_0 \end{bmatrix}. \tag{4.10}$$

By showing that A_t is non-singular (Lemma 6 in [17],) Lakshman and Saunders proposed the following univariate sparse interpolation in the Chebyshev basis (see [17] for more details.)

The sparse interpolation in the Chebyshev basis

Input:

$f(x)$**:** *a univariate black box polynomial.*

t**:** *the number of non-zero terms in* f*, in the Chebyshev bases.*

Output:

c_j *and* δ_j*:* $f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x)$.

1. (Solve the symmetric Hankel-plus-Toeplitz system in (4.9).)

   $p > 1$, $a_k = f(T_k(p))$ *for* $k = 0, 1, \dots, 2t-1$.

Determine $\Lambda(z)$*:*

$\lambda_t = 1$*, and for* $0 \le j \le t-1$*, the coefficients* λ_j *are obtained by solving the symmetric Hankel-plus-Toeplitz system in* (4.9)*.*

2. (Determine δ_j.)

   Find all t *distinct roots of* $\Lambda(z)$*, which are* $T_{\delta_j}(p)$*.*

   Determine the Chebyshev term exponents δ_j *from* $T_{\delta_j}(p)$*.*

3. (Compute the coefficients c_j.)

   Recover the coefficients c_j *of the Chebyshev terms* $T_{\delta_j}(p)$:

   solve a transposed Vandermode-like system (see the discussion in [17].)

End.

Remark: By proving A_t is non-singular, Lakshman and Saunders assured the solution to system (4.9) in Step 1. Nevertheless, in general it does not promise the system being solved in $O(t^2)$ field operations. Gohberg and Koltracht presented an algorithm that can solve a symmetric Hankel-plus-Toeplitz matrix in $O(t^2)$, yet it requires all principal leading submatrices being non-singular. More details are carried out in Section 4.4.

4.4 Early termination of sparse interpolations in the Chebyshev basis

Gohberg and Koltracht [9] presented an efficient $O(n^2)$ algorithm for solving an $n \times n$ symmetric Hankel-plus-Toeplitz matrix with non-singular principal leading submatrices.

In order to apply the efficient $O(n^2)$ algorithm to our symmetric Hankel-plus-Toeplitz matrix, we need to assure that for a random value $p > 1$, with high probability, the corresponding matrix A_t defined as in (4.10) and all its principal leading submatrices are non-singular. Yet, this is not true in general: for any polynomial $f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x)$ with $c_j \neq 0$ and $c_1 + c_2 + \cdots + c_t = 0$, we have $a_0 = f(T_0(p)) = f(1) = \sum_{j=1}^{t} c_j = 0$ and the first leading submatrix

$$A_1 = \begin{bmatrix} 2a_0 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

singular.

We fix this problem by adding an additional random value to the constant term of $f(x)$: pick a random value p_c, instead of $f(x)$, we proceed with the interpolation on $\tilde{f}(x) = f(x) + p_c$. With the additional random value included, the sum of all coefficients in $\tilde{f}(x)$ is non-zero with high probability. (In case that $\tilde{f}(T_0(p)) = 0$ is encountered, we can pick another non-zero random value $\tilde{p}_c$ and reassign p_c as p_c + $\tilde{p}_c$. Thus, we always start with a 1×1 non-singular leading submatrix.) After the interpolation is done, the original target polynomial $f(x)$ can be recovered by simply removing the additional random value from $\tilde{f}(x)$. In Theorem 4.4, this additional random number also provides all other principal leading submatrices non-singular with high probability.

Consider $\tilde{f}(x) = \sum_{j=1}^{\tilde{t}} \tilde{c}_j T_{\delta_j}(x)$ with $\delta_1 < \delta_2 < \cdots < \delta_{\tilde{t}}$ whose constant term has already been "randomized," and let the symbol y represent the random component in the constant term of $\tilde{f}(x)$. Therefore, we have $\sum_{j=1}^{t} c_j + y = \sum_{j=1}^{\tilde{t}} \tilde{c}_j$. Our purpose is to prove that in symbolic values, or variables x and y, the symmetric

Hankel-plus-Toeplitz matrices $\mathcal{A}_i$ are non-singular for $i = 1, \ldots, \tilde{t}$ and that $\mathcal{A}_{\tilde{t}+1}$ is singular.

Let $\alpha_k = \tilde{f}(T_k(x))$ for $k = 0, 1, \ldots, 2\tilde{t} - 1$, and consider the $i \times i$ symmetric Hankel-plus-Toeplitz matrix $\mathcal{A}_i$ for $i \geq 1$ whose entries are polynomials in variables x and y:

$$\mathcal{A}_i = \begin{bmatrix} 2\alpha_0 & 2\alpha_1 & \ldots & 2\alpha_{i-1} \\ 2\alpha_1 & \alpha_2 + \alpha_0 & \ldots & \alpha_i + \alpha_{i-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2\alpha_{i-1} & \alpha_i + \alpha_{i-2} & \ldots & \alpha_{2i-2} + \alpha_0 \end{bmatrix}. \tag{4.11}$$

The singularity of $\mathcal{A}_i$ for $i \geq \tilde{t} + 1$ can be concluded through Lemma 5 in [17], which states any $p > 1$ satisfies the corresponding linear relations $\sum_{j=0}^{\tilde{t}} \lambda_j (a_{j+i} + a_{|j-i|}) = 0$ for $i = 0, 1, \ldots$.

As for $1 \leq i \leq \tilde{t}$, via matrix multiplications (also see Lemma 6 in [17],) the factorization $\mathcal{A}_i = \mathcal{V}_i C \mathcal{V}_i^{\mathrm{Tr}}$ can be verified, where

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

$$C = \begin{bmatrix} 2c_1 & 0 & \ldots & 0 \\ 0 & 2c_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 2c_{\tilde{t}} \end{bmatrix}.$$

The Chebyshev polynomials commute with respect to composition: that is, for

$m, n \geq 0$, $T_n(T_m(x)) = T_{mn}(x) = T_m(T_n(x))$, and

$$\mathcal{V}_i = \begin{bmatrix} T_{\delta_1 \cdot 0}(x) & T_{\delta_2 \cdot 0}(x) & \ldots & T_{\delta_{\tilde{t}} \cdot 0}(x) \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1 \cdot (i-1)}(x) & T_{\delta_2 \cdot (i-1)}(x) & \ldots & T_{\delta_{\tilde{t}} \cdot (i-1)}(x) \end{bmatrix}. \tag{4.12}$$

LEMMA 4.2. *For $n \geq 1$, $T_{n \cdot \delta}(x) = \sum_{i=0}^{n} \gamma_{n,i} T_\delta(x)^i$ and $\gamma_{n,n} = 2^{n-1}$.*

Proof:

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

Assume for some k the statement is true for all $n \leq k$, and consider $n = k + 1$,

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

By mathematical induction, $T_{(k+1)\cdot\delta}(x) = \sum_{i=0}^{k+1} \gamma_{k+1,i} T_\delta(x)^i$ with $\gamma_{k+1,k+1} = 2^k$, and that Lemma 4.2 is proved. ⊠

Through Lemma 4.2, we can factorize $\mathcal{V}_i$ in (4.12) as a product of an $i \times i$ lower triangular matrix $\mathcal{L}_i$ and an $i \times \tilde{t}$ rectangular Vandermonde matrix $\mathcal{B}_i$ as the following:

$$\mathcal{V}_i = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & \vdots \\ * & * & 2 & 0 & \dots & \vdots \\ * & * & * & 4 & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & * & \dots & 2^{i-2} \end{bmatrix} \begin{bmatrix} T_{\delta_1}(x)^0 & T_{\delta_2}(x)^0 & \dots & T_{\delta_{\tilde{t}}}(x)^0 \\ T_{\delta_1}(x)^1 & T_{\delta_2}(x)^1 & \dots & T_{\delta_{\tilde{t}}}(x)^1 \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1}(x)^{i-1} & T_{\delta_2}(x)^{i-1} & \dots & T_{\delta_{\tilde{t}}}(x)^{i-1} \end{bmatrix}$$

$$= \mathcal{L}_i \mathcal{B}_i. \tag{4.13}$$

Therefore, for $1 \leq i \leq \tilde{t}$, $\mathcal{A}_i$ can be factorized as

$$\mathcal{A}_i = \mathcal{V}_i C \mathcal{V}_i^{\mathrm{Tr}} = \mathcal{L}_i \mathcal{B}_i C (\mathcal{L}_i \mathcal{B}_i)^{\mathrm{Tr}} = \mathcal{L}_i (\mathcal{B}_i C \mathcal{B}_i^{\mathrm{Tr}}) \mathcal{L}_i^{\mathrm{Tr}}. \tag{4.14}$$

In [17], Lakshman and Saunders showed $\mathcal{A}_{\tilde{t}}$ is non-singular (see Lemma 6.) Our next theorem shows all its principal leading submatrices are also non-singular.

THEOREM 4.4. *The determinant of $\mathcal{A}_i$ is non-zero for $i = 1, \dots, \tilde{t} - 1$.*

Proof:

For i with $1 \leq i < \tilde{t}$, from $\mathcal{A}_i = \mathcal{L}_i (\mathcal{B}_i C \mathcal{B}_i^{\mathrm{Tr}}) \mathcal{L}_i^{\mathrm{Tr}}$ in (4.14) we have

$$\det(\mathcal{A}_i) = \det(\mathcal{L}_i) \cdot \det(\mathcal{B}_i C \mathcal{B}_i^{\mathrm{Tr}}) \cdot \det(\mathcal{L}_i^{\mathrm{Tr}}).$$

Assume that $\det(\mathcal{B}_i C \mathcal{B}_i^{\mathrm{Tr}}) = 0$ for some $1 \leq i < \tilde{t}$, which implies all the terms in $\det(\mathcal{B}_i C \mathcal{B}_i^{\mathrm{Tr}})$ are zero. In other word, for every ordered list I from $\{1, 2, \dots, \tilde{t}\}$ such that $\#(I) = i - 1$ and $j_k \in I$ with index $k = 1, \dots, i - 1$, the coefficient of term $\prod_{j_k\in I}\left(T_{\delta_j}(x)\right)^{2k}$ is zero, that is,

$$2^i\cdot\prod_{j_k\in I}\tilde{c}_{j_k}\cdot\sum_{j\in\{1,2,\ldots,\tilde{t}\}-I}\tilde{c}_j=0.$$

Knowing that both 2^i and $\prod_{j_k\in I}\tilde{c}_{j_k}$ are non-zero, it has to be that

$$\sum_{j\in\{1,2,\ldots,\tilde{t}\}-I}\tilde{c}_j=0. \tag{4.15}$$

Adding all the possible sums in (4.15), we have

$$\sum_{\#(I)=i-1}\Big(\sum_{j\in\{1,2,\ldots,\tilde{t}\}-I}\tilde{c}_j\Big)=(\tilde{t}-1)\cdot(\tilde{t}-2)\cdots(\tilde{t}-i+1)\cdot\sum_{j=1}^{\tilde{t}}\tilde{c}_j=0. \tag{4.16}$$

The sum in (4.16) implies that $\sum_{j=1}^{t}\tilde{c}_j=y+\sum_{j=1}^{t}c_j=0$, which is a contradiction since the variable y cannot be cancelled in $y+\sum_{j=1}^{t}c_j$. Hence, $\det(\mathcal{B}_iC\mathcal{B}_i^{\mathrm{Tr}})\neq 0$ for all $1\leq i<\tilde{t}$.

We conclude $\det(\mathcal{L}_i^{\mathrm{Tr}})=\det(\mathcal{L}_i)\neq 0$ from their non-zero diagonals. Therefore, $\det\mathcal{A}_i\neq 0$ for $1\leq i<\tilde{t}$. ⊠

We have $\mathcal{A}_i$ being non-singular in symbolic values x and y for $i=1,\ldots,\tilde{t}$ and singular for $i\geq\tilde{t}+1$. Now make a transition from symbolic values x and y to random numbers p and p_c, and evaluate $\mathcal{A}_i$ in (4.11) at $x=p$ and $y=p_c$ as matrix values A_i. With high probability, when i is being increased, matrix A_i is singular the first time at $i=\tilde{t}+1$.

The original Gohberg/Koltracht algorithm [9] solves for $Ax=c$ when A and c are given and that A is provided as a non-singular symmetric Hankel-plus-Toeplitz matrix with all its principal leading submatrices non-singular. In our implementation, which is presented as the following modified Gohberg/Koltracht algorithm, we look for λ_j, $0 \le j \le \tilde{t}$, that satisfy the relations as in (4.8). In other word, with the first non-singular 1×1 matrix provided, we find the solution to the first singular system $A_{\tilde{t}+1}\underline{\lambda} = 0$ such that $\underline{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_{\tilde{t}}]^{\mathrm{Tr}}$ and $\lambda_{\tilde{t}} = 1$.

We use $\tilde{a}_{i,j}$ to represent the entry at i-th row and j-th column in the given symmetric Hankel-plus-Toeplitz matrix, and underline vector variables to distinguish them from their indexed components, for example $\underline{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_i]^{\mathrm{Tr}}$. The matrix I_i is the $i \times i$ identity matrix, and L_i is the $i \times i$ matrix defined as:

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

The modified Gohberg/Koltracht algorithm

Input:

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

Output:

$\underline{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_t]^{\mathbf{Tr}}$**:** $A_{t+1}\underline{\lambda} = 0$ *with* $\lambda_t = 1$, *where* $t \ge 1$ *is the smallest integer such that the given symmetric Hankel-plus-Toeplitz system* A_{t+1} *is singular.*

1. (We have $\tilde{a}_{1,1} \neq 0$, the discrepancy Δ checks whether A_2 is singular. 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 $\underline{\lambda} = [-\tilde{a}_{1,2}/\tilde{a}_{1,1}, 1]^{Tr}$*;*

Else

$\underline{\gamma} \leftarrow [1/\tilde{a}_{1,1}]$*;*

$i \leftarrow 1$*;*

$\underline{\psi} \leftarrow [1/\tilde{a}_{1,1}]$*;*

$\underline{\phi} \leftarrow [t_1/\tilde{a}_{1,1}]$*;*

$\alpha \leftarrow \tilde{a}_{2,1}/\tilde{a}_{1,1}$*;*

$\underline{\gamma}^{new} \leftarrow (1/\Delta) \cdot [-\tilde{a}_{1,2}, \tilde{a}_{1,1}]^{Tr}$*;*

2. (As i being increased, if $\Delta \neq 0$, follow the Gohberg/Koltracht algorithm [9] and update $\underline{\gamma}^{\text{new}}$ such that $A_{i+1}\underline{\gamma}^{\text{new}} = [0, \ldots, 0, 1]^{\text{Tr}}$. If $\Delta = 0$, than A_{i+1} is singular (see Theorem 4.5,) 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*

$i \leftarrow i + 1$*;*

$\kappa \leftarrow (t_i + h_{i-2}) - \sum_{j=1}^{i-1} \tilde{a}_{i,j} \cdot \phi_j$*;*

$\mu \leftarrow -\sum_{j=1}^{i-1} \tilde{a}_{i,j} \cdot \psi_j$*;*

$\underline{\phi}^{new} \leftarrow [\underline{\phi}^{Tr}, 0]^{Tr} + \kappa \cdot \underline{\gamma}^{new}$*;*

$\underline{\psi}^{new} \leftarrow [\underline{\psi}^{Tr}, 0]^{Tr} + \mu \cdot \underline{\gamma}^{new}$*;*

$\alpha^{new} \leftarrow \sum_{j=1}^{i} \tilde{a}_{i+1,j} \cdot \gamma_j^{new}$*;*

$\underline{b} \leftarrow \left((\alpha - \alpha^{new})I_i + L_i + L_i^{Tr}\right) \cdot \underline{\gamma}^{new} - [\underline{\gamma}^{Tr}, 0]^{Tr} + \psi_i^{new} \cdot \underline{\phi}^{new} - \phi_i^{new} \cdot \underline{\psi}^{new}$*;*

$\nu \leftarrow \left(\gamma_i^{new}\right)^{-1} \cdot \sum_{j=1}^{i} \tilde{a}_{i+1,j} \cdot b_j$*;*

$\Delta \leftarrow \nu + \tilde{a}_{i+1,i+1}$*;*

If $\Delta = 0$ *then*

$\lambda_i \leftarrow 1;$

For $j = 0 \ldots i-1$ *do*

$\lambda_j \leftarrow b_{j+1}/\gamma_i^{new};$

Return $\underline{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_i]^{Tr};$

Else

$\underline{\gamma} \leftarrow \underline{\gamma}^{new};$

$\gamma_{i+1}^{new} \leftarrow 1/\Delta;$

For $j = 1, \ldots, i$ *do*

$\gamma_j^{new} \leftarrow \left(\gamma_{i+1}^{new}/\gamma_i\right) \cdot b_j;$

(At the end of Step 2, update variables for next i.)

$\underline{\phi} \leftarrow \underline{\phi}^{new};$

$\underline{\psi} \leftarrow \underline{\psi}^{new};$

$\alpha \leftarrow \alpha^{new};$

End.

Notice that in our modified Gohberg/Koltracht algorithm, instead of checking the value of a determinant, we check the discrepancy Δ. Theorem 4.5 shows the discrepancy Δ in the modified Gohberg/Koltracht algorithm reflects the singularity a corresponding matrix.

We need the following lemma for Theorem 4.5.

Lemma 4.3. *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 $\Delta = 0$, we have $\lambda_i = 1$ 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 λ is the following:

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

The products of the matrix multiplication between the j-th row and λ for $1 \leq j \leq i$ are all zero by the definitions of λ_j for $0 \leq j \leq i-1$ (see page 139–140 in [9].) $\boxtimes$

THEOREM 4.5. *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 the discrepancy Δ is zero if and only if $\det(A_{i+1}) = 0$.*

Proof:

If $\det(A_j) \neq 0$ for all $1 \leq j \leq i$ and Δ is non-zero, A_{i+1} can be inverted through Gohberg/Koltracht algorithm. Therefore, $\det(A_{i+1}) \neq 0$.

To prove another direction, we assume the discrepancy Δ to be zero. From Lemma 4.3, we have a non-zero solution to $A_{i+1}\underline{\lambda} = 0$ and $\det(A_{i+1}) = 0$. $\boxtimes$

In Theorem 4.5, we show the vanishing of discrepancies indicates the singularities of A_{i+1}. Now back to our sparse interpolation in the Chebyshev basis, without the input $\tilde{t}$, the number of non-zero terms in the Chebyshev basis, the coefficients λ_j in $\Lambda(z)$ can be determined at the zero discrepancy $\Delta = 0$ with high probability. Using the discrepancy $\Delta = 0$ as the termination test, the early termination strategy can be implemented to the sparse interpolation under the Chebyshev basis.

It is questionable whether we can directly implement a threshold $\zeta > 1$ in a similar manner as in the early termination versions of the Ben-Or/Tiwari algorithm and the sparse interpolation in the Pochhammer basis. Since the embedded modified Gohberg/Koltracht algorithm requires $\Delta \neq 0$ for all the principal leading matrices in order to proceed with the next i. To exploit the threshold implementation, we refer to the approach of [3]. We also mention another strategy to further check λ_j: when λ_j are determined from A_{i+1}, check whether $\sum_{j=0}^{i-1} \lambda_j(a_{j+k} + a_{|j-k|})$ $= -(a_{i+k} + a_{|i-k|})$ at additional $k = i, i+1, \ldots$.

Theorem 4.6. *If p is chosen randomly and uniformly from a subset S of the domain of values, which is assumed to be an integral domain, and that all the elements in S are larger than 1, then for the sequence $a_k = \tilde{f}(T_k(p))$ with $k \geq 0$ the determinant of the matrix*

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

encounters 0 the first time at $i = \tilde{t} + 1$ with probability no less than

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

where $\tilde{f}(x)$ is a polynomial whose constant has been "randomized" and $\tilde{t}$ the number of non-zero terms in $\tilde{f}(x)$ with respect to the Chebyshev basis.

Proof:

Through (4.13) and (4.14), we obtain $\deg(\det \mathcal{A}_i) \leq i^2 \cdot \deg(\tilde{f}(x))$. If $\det A_i$ does not encounter a zero until $i = \tilde{t} + 1$ and that $\det A_1 \neq 0$ is provided, we need to avoid hitting a root of the product $\prod_{i=2}^{\tilde{t}} \det(A_i)$, whose degree is no more than $(\tilde{t}-1)(2\tilde{t}^2+5\tilde{t}+6)\deg(\tilde{f})/6$. The estimate of the probability follows from Lemma 1 in [21]. $\boxtimes$

Since $\deg(f)=\deg(\tilde{f})$, we can use $\deg(f)$ for the probability estimate in Theorem 4.6. Now we are ready to present the early termination version of the sparse interpolation in the Chebyshev basis.

The sparse interpolation with early termination in the Chebyshev basis

Input:

$f(x)$**:** *a univariate black box polynomial.*

Output:

c_j *and* δ_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 always non-singular.)

   Pick a random element p *with* $p>1$ *and another random element* p_c*.*

   If $a_0=f(T_0(p))+p_c=0$ *then*

   pick a random $\tilde{p}_c \neq 0$*;*

   $a_0 \leftarrow \tilde{p}_c$*;*

   $p_c \leftarrow p_c+\tilde{p}_c$*;*

   $f(x) \leftarrow f(x)+p_c$*;*

   else

   $f(x) \leftarrow f(x)+p_c$*;*

2. (The early termination in the modified Gohberg/Koltracht algorithm.)

   For $i=1,2,\ldots$

Perform the modified Gohberg/Koltracht algorithm on the matrix whose entries are defined as $\tilde{a}_{i,j} = a_{i+j-2} + a_{|i-j|}$, where $1 \leq j \leq i$ and $a_i = f(T_i(p))$.

If $\Delta = 0$, then

the modified Gohberg/Koltracht algorithm returns λ_j, which define $\Lambda(z)$ through its coefficients;

break out of the loop.

3. (Determine δ_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 δ_j from $T_{\delta_j}(p)$, $T_{\delta_j}(p)$ are the roots of $\Lambda(z)$:

   again, the recovery of δ_j might fail.

4. (Compute the coefficients c_j.)

   Recover the coefficients c_j of the Chebyshev terms $T_{\delta_j}(p)$:

   solve a transposed Vandermode-like system (see the discussion in [17].)

   Recover the original input $f(x)$ by removing p_c from the result.

End.

Remark: By adding a random value p_c or $p_c + \tilde{p}_c$ to the constant term of $f(x)$, we might introduce one more term (the constant term) to $f(x)$. Namely, $\tilde{t} = t$ when there is a non-zero constant in $f(x)$, and $\tilde{t} = t + 1$ when the constant in $f(x)$ is zero, in which case certainly the overhead interpolations are introduced in a sparse algorithm. Nevertheless, we consider such added overhead limited.

Chapter 5

Early Termination in Racing Algorithms

We have implemented the early termination strategy to different existing interpolation algorithms. Now suppose a given univariate black box polynomial $f(x)$ has t as the number of its non-zero terms in a designate basis, η and ζ the given thresholds, without an input as a bound on either $\deg(f)$ or t, with high probability the early termination strategy enables a dense algorithm to interpolate f in $\deg(f) + \eta + 1$ black box queries, and a sparse algorithm to interpolate f in $2t + \zeta$ queries.

Although a dense algorithm does not take advantage of the sparsity in the target polynomial, it is more efficient than a sparse algorithm in interpolating a dense polynomial. Therefore, even though without a bound as an input we can interpolate the target polynomial through early termination, we still face the predicament of selecting a more efficient algorithm which is derived from the sparsity of the target polynomial.

Based on the early termination, we propose the racing interpolation algorithms. A racing algorithm races an early termination sparse algorithm against an early termination dense interpolation on a same set of evaluation points. Without introducing more polynomial evaluations, it performs two interpolation algorithms in parallel and terminates when either of the racers finishes interpolation first. Therefore, the overall racing algorithm is superior than either of the racers because it takes advantage of both algorithms and compensates for the disadvantage of either one. We prove the early termination property also exists in the overall racing algorithm.

5.1 Race dense against sparse interpolations in early termination

A dense interpolation algorithm requires the evaluations of the target polynomial on a set of sufficiently many distinct points. Performing a sparse interpolation algorithm does not prevent us from interpolating on the same set of points via a dense one in parallel. As a result, we propose an algorithm that races the early termination versions of a dense interpolation against a sparse interpolation as the following: on the same set of evaluation points, for every black box probe, we implement and keep track of both algorithms; whenever either one of the racer algorithms first terminates via early termination, the overall algorithm terminates. This is our racing algorithm.

When there is not much, or even without, information on the sparsity of the target polynomial, the overall racing algorithm is superior in average: the sparse racer provides the more efficient early termination whenever it is possible; while at the same time the dense racer guarantees the overall algorithm being terminated with respect to the degree of the target polynomial.

However, instead of a sequence of random numbers, the early termination sparse algorithms need to interpolate on a sequence of numbers constructed by a random number p. And such construction depends on the designate bases in the sparse algorithms: $p, p^2, p^3, \ldots$ for the power basis; $p, p+1, p+2, \ldots$ for the Pochhammer basis; and $T_0(p), T_1(p), T_2(p), \ldots$ for the Chebyshev basis.

The values in a sequence constructed by a random number p might be distinct. Yet, in Theorem 2.1, the early termination of dense interpolations requires polynomial evaluations on a sequence of random numbers, $p_0, p_1, p_2, \ldots$, and that each p_i is randomly generated. Therefore, in order to show the overall racing algorithm is correct with high probability, we need to show the early termination property for dense interpolations is true when the sequences of evaluation points are constructed by a random number p.

Suppose $i \in \mathbb{Z}_{\geq 0}$, let b_i form a generic basis for a polynomial ring $\mathbb{K}[p]$ such that $\deg(b_i) = i$ and $\deg(b_i \cdot b_j) = i + j$. We use $x^{\{n\}}$, $n \in \mathbb{Z}_{\geq 0}$, to denote a generic rising factorial power in x:

$$x^{\{n\}} = (x - b_0)(x - b_1) \cdots (x - b_{n-1}).$$

A polynomial $f(x)$ with degree n interpolated through Newton on a sequence of elements constructed by p can be viewed as $f(x)$ interpolated at $b_0, b_1, \ldots$, and $f(x)$ represented as:

$$f(x) = \sum_{i=0}^{n} \overline{a}_i x^{\{i\}}, \tag{5.1}$$

where $\overline{a}_i$ depend on p and are polynomials in $\mathbb{K}[p]$.

Our purpose is to show $\overline{a}_i$ are nonzero polynomials in $\mathbb{K}[p]$ for $0 \leq i \leq n$ (Theorem 5.4,) so that we can claim when p is a random value, we encounter $\overline{a}_i = 0$ first time at $i = n + 1$ and that $f(x)$ is interpolated as (5.1) with high probability.

When $\deg(f) = n$, comparing the representations of $f(x)$ in the power basis and in the generic rising factorial power basis, we have

$$f(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} \overline{a}_i x^{\{i\}}, \tag{5.2}$$

with both a_n and $\overline{a}_n$ nonzero.

Now consider the transformations from the generic factorial power basis to the power basis. For all n, the coefficients $c_i^{(n)}$, which depend on b_j, define the transformation:

$$x^n = \sum_{i=0}^{n} c_i^{(n)} x^{\{i\}}. \tag{5.3}$$

We need the following lemma for proving Theorem 5.1.

LEMMA 5.1. *For any integer $k > 1$ and every $k - 1 \geq j \geq 1$,*

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

Proof:

Replace x as $(x - b_j + b_j)$:

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

The following theorem shows how to update $c_i^{(n)}$ from $c_j^{(n-1)}$ and b_j.

THEOREM 5.1. *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 5.1 for j from $n-1$ to 1 to

$$x \cdot \sum_{s=0}^{n-1} c_s^{(n-1)} x^{\{s\}} :$$

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

Compare the coefficients in $c_s^{(n)}$ and $c_s^{(n-1)}$, we have $c_0^{(n)} = b_0 c_0^{(n-1)}$ and $c_s^{(n)} = b_s c_s^{(n-1)} + c_{s-1}^{(n-1)}$ for $0 < s < n$.

We still need to prove $1 = c_n^{(n)}$. This is easy because $c_n^{(n)} = c_{n-1}^{(n-1)}$ and $c_0^{(0)} = 1$.

⊠

Now we define $c_s^{(n)} = 0$ for all $s > n$ and consider $c_s^{(n)}$ in terms of b_j and p, where b_j are univariate polynomials in p. Next theorem is on the degree of $c_s^{(n)}$ in the variable p.

THEOREM 5.2. *For any integer $n > 0$, and any integer s such that $n > s > 0$,*

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

Proof:

Repeatedly apply Theorem 5.1,

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

Since $\deg(b_i \cdot b_j) = i + j$ and $c_s^{(s)} = c_{s-1}^{(s-1)} = 1$, when $n > s > 0$,

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

Based on Theorem 5.2, next theorem compares the degrees between $c_s^{(n+1)}(p)$ and $c_s^{(n)}(p)$ in the variable p.

THEOREM 5.3. *For any integer $n > 0$, and any integer s such that $n > s > 0$,*

$$\deg\big(c_s^{(n+1)}(p)\big) > \deg\big(c_s^{(n)}(p)\big).$$

Proof:

When $n > s > 0$, through Theorem 5.2,

$$\deg\big(c_s^{(n+1)}(p)\big) = s \cdot (n + 1 - s) > s \cdot (n - s) = \deg\big(c_s^{(n)}(p)\big). \quad \boxtimes$$

For a polynomial $f(x)$ represented in the power basis and the generic factorial power basis in (5.2), $\deg(f) = n$ implies $a_n \neq 0$. Next theorem shows that for every $0 < i \leq n$, $\overline{a}_i$ is a nonzero polynomial in variable p.

THEOREM 5.4. *Let* $f(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} \overline{a}_i x^{\{i\}}$. *If* $a_n \neq 0$, *for every* $0 < i \leq n$, $\overline{a}_i$ *is a non-zero polynomial in* p. *Moreover,* $\overline{a}_n = a_n$.

Proof:

Expand $f(x)$ and collect all the terms with respect to the generic bases $x^{\{i\}}$:

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

Compare the coefficients, for $n \geq i > 0$, we have

$$\overline{a}_i = \sum_{j=0}^{n-i} a_{n-j} c_i^{(n-j)}$$

and $\overline{a}_n = a_n c_n^{(n)} = a_n$ since $c_n^{(n)} = 1$.

From Theorem 5.3, the highest order term in $c_i^{(n)}(p)$ occurs nowhere else in $\overline{a}_i(p) = a_n c_i^{(n)} + a_{n-1} c_i^{(n-1)} + \cdots + a_i c_i^{(i)}$. Therefore, if a_n is non-zero, $\overline{a}_i(p)$ is not a zero polynomial in p for every $0 < i \leq n$. ⊠

Theorem 5.4 states that $\overline{a}_i(p)$ are non-zero polynomials for $0 < i \leq n$ when $n = \deg(f)$. Therefore, if p is a random number, then with high probability $\overline{a}_i(p) = 0$ occurs the first time at $i = n + 1$. Our racing algorithms require the early termination of Newton interpolation to be true on sequences constructed by a random number p in the sparse algorithms. By showing the designate bases as special cases of the generic basis in Theorem 5.4, following theorems provide the early termination for different racing algorithms.

Theorem 5.5. *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 number p_c, we can 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 5.4. ⊠

Theorem 5.6. *If p is randomly picked, the early termination of Newton interpolation is true if it interpolates on the sequence $T_0(p)$, $T_1(p)$, …, $T_k(p)$, ….*

Proof:

Let $T_i(p) = b_i$ and $\deg(b_i) = \deg(T_i(p)) = i$, then apply Theorem 5.4. ⊠

Remark: In our implementation, we first race both algorithms on a sequence, $b_0(p_1)$, $b_1(p_1)$, …, constructed by a random number p_1. It is possible that the sparse racer terminates first, yet unsuccessfully, while the dense racer has not finished the interpolation process. Whenever such scenario happens, we pick another random value p_2 and use p_2 to construct a new sequence $b_0(p_2), b_1(p_2), \ldots$. Then on this new sequence, we restart the sparse racer but keep updating the existing dense interpolant (see Section 5.2 for more details.) To show the early termination in Newton exists for the "restarting" of the race, both Theorems 5.5 and 5.6 can be modified by assigning p_i in a lexicographic order as $p_1 \prec p_2 \prec \cdots \prec p_i \prec \cdots$.

When Newton is racing against the sparse algorithm in the Pochhammer basis, it interpolates on the subsequent values $p, p+1, p+2, \ldots$. Since $\deg(p) =$

$\deg(p+1) = \deg(p+2) = \cdots$, in this case Theorem 5.4 cannot be applied to the early termination of Newton interpolation. The next theorem provides the early termination for the racing algorithm that races Newton against the sparse algorithm in the Pochhammer basis.

THEOREM 5.7. *If p is randomly picked, the early termination of Newton interpolation is true if it interpolates on the sequence $p, p+1, p+2, \ldots$.*

Proof:

Suppose $\deg(f) = n$ and $f^{[i]}$ the i-th Newton interpolant which interpolates p, $p+1$, $p+2$, ..., $p+i$, we have

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

If f is not a zero polynomial, $c_0 = f(p)$ is not a zero polynomial in variable p.

Let $0 \le i < n$, suppose for every $0 \le k \le i$, c_k is a non-zero polynomial in p and $\deg(f^{[i]}(x)) = i$. Consider c_{i+1} in the $i+1$-th interpolant:

$$f^{[i+1]}(x) = f^{[i]}(x) + c_{i+1}(x-p)(x-p-1)\cdots(x-p-i). \tag{5.4}$$

When $i = n-1$, c_{i+1} must be a non-zero polynomial in p, since if not, $\deg(f)$ $= n = \deg(f^{[n-1]}) = \deg(f^{[i]})$ and $\deg(f^{[i]}) = i = n-1$, a contradiction.

Now consider $0 \le i < n-1$, if for every $0 \le k \le i$, c_k is a non-zero polynomial in p and $c_{i+1} = 0$ for every p in its domain, then c_{i+1} is a zero polynomial and $f^{[i]} = f^{[i+1]}$. This implies $f^{[i]} = f^{[i+1]} = \cdots = f^{[n]} = f$. Otherwise suppose $f^{[j]}$ is the first interpolant being updated since $f^{[i+1]}$, that is, $c_{j-1} = 0$ for all p and

$$f^{[j]} = f^{[i]} + c_j(x-p)(x-p-1)\cdots(x-p-j+1) \tag{5.5}$$

with $c_j \neq 0$ and $i+1 < j \leq n$. We expand the newly updated term in (5.5) with respect to p shifted by 1 as the following:

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

Therefore, $c_{j-1} \neq 0$ when the value of p is shifted by 1 and it is a contradiction to the claim that $c_{j-1} = 0$ for all p. Consequently, if $f^{[i]} = f^{[i+1]}$ for all p, it must be $f^{[i]} = f^{[i+1]} = \cdots = f^{[n]} = f$.

However, $f \neq f^{[i]}$ as $\deg(f) = n > i = \deg(f^{[i]})$, so c_{i+1} in (5.4) cannot be a zero polynomial for every $0 \leq i < n$.

When p is a random value and $f(x) \neq 0$, $c_i = 0$ for $0 \leq i \leq \deg(f)$ only when p hits a zero in c_i. As a result, with high probability $c_i = 0$ the first time at $i = \deg(f) + 1$. $\boxtimes$

5.2 Racing algorithm

In this section, we present the algorithm steps of a racing algorithm, which races a dense algorithm (Newton interpolation) against a sparse algorithm in interpolating a univariate black box polynomial $f(x)$.

In our implementation, we pick a random number p_1 and construct a sequence $b_0(p_1)$, $b_1(p_1)$, $b_2(p_1)$, ... derived from the designate basis in the sparse racer algorithm. Then we in parallel interpolate $f(x)$ via both the early termination versions of Newton interpolation (dense racer) and a sparse algorithm (sparse racer.) Whenever the sparse racer successfully terminates earlier than Newton interpolation, the overall racing algorithm terminates. Nevertheless, it is possible the early termination of the sparse racer fails for an unlucky p_1. In such case we pick another random number p_2 and restart the sparse racer on $b_0(p_2), b_1(p_2), b_2(p_2), \ldots$, while still keep on updating the existing Newton interpolant on this newly generated sequence. We can keep restarting such race until either one of the racers successfully terminates. Newton interpolation guarantees the termination of the overall algorithm if sufficiently many distinct points have been evaluated. Figure 5.1 shows an example of a possible scenario.

We give more details of the racing algorithms in the case of racing Newton interpolation against the Ben-Or/Tiwari algorithm, and provide the outlines of the racing algorithm for racing Newton against a sparse interpolation.

Recall that in Section 2.2, $f^{[i]}(x)$ is the interpolant from interpolating the first $i+1$ elements in the evaluation point sequence, and $f^{\{k\}}(x)$ the first $k+1$ distinct elements. To race Newton against Ben-Or/Tiwari, we pick a random number p_1 to form the sequence $p_1, p_1^2, \ldots$, and update the Newton interpolant $f_N^{[i]}(x) = f_N^{\{k\}}(x)$ as well as the error locator polynomial Λ_i embedded in the Berlekamp/Massey algorithm. When neither Λ_i nor $f_N^{[i]}(x)$ satisfies the corresponding early termination

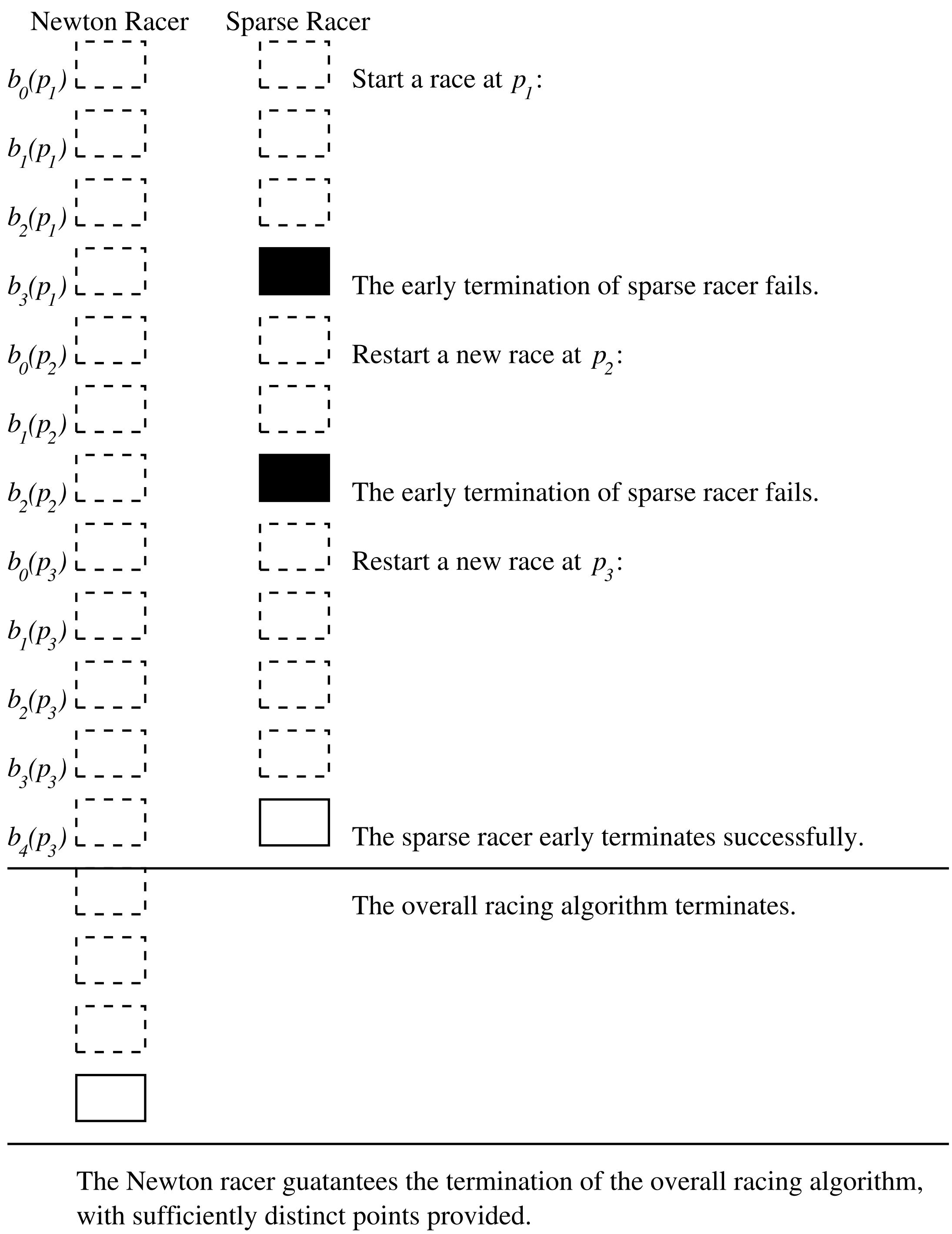


Figure 5.1: A possible scenario of the racing algorithm.

criteria, we proceed with the next i. If Λ_i satisfies the early termination, we continue with all the remaining steps of the Ben-Or/Tiwari algorithm. Suppose all the remaining steps are successfully finished, we have $f(x)$ interpolated through Ben-Or/Tiwari with high probability; otherwise, pick another random element p_2 and restart the Berlekamp/Massey algorithm on $p_2, p_2^2, \ldots$. In the meantime, $f^{[i+1]}(x)$ is updated as the interpolant which interpolates p_1, p_1^2, $\ldots$, p_1^i, p_1^{i+1}, p_2. Namely, we keep updating the Newton interpolant on the sequence of all the black box probes we have acquired so far. On the other hand, if the Newton interpolant $f_N^{[i]}(x) = f_N^{\{k\}}(x)$ satisfies the early termination conditions, then $f(x)$ is interpolated as $f_N^{[i]}(x)$ through Newton and the overall racing algorithm is terminated as well.

None of the early termination algorithms requires a bound on either the degree or the number of terms. In our application, we request δ as an upper bound on $\deg(f(x))$, not for the purpose of interpolations, but for guarding the overall racing algorithm from running into an infinite loop. In other word, the overall black box probes and the interpolation efforts are confined by the degree bound δ and the corresponding thresholds.

Under a domain that provides enough distinct numbers, the Ben-Or/Tiwari algorithm might terminate earlier in a sparse case, yet due to the unlucky numbers, it might not finish at all; Newton interpolation might cost more black box probes, it can always finish interpolating. Therefore, racing these two algorithms can take advantage of both algorithms. That is, whenever it is possible, it can terminate earlier while its termination is still guaranteed. In addition, by cross checking the polynomial information acquired from two different algorithms, we further improve the probability of correctness. For example, check whether $\deg(f(x))$ of $f(x)$ recovered from the Ben-Or/Tiwari algorithm is larger than or equal to the degree of the most recent Newton interpolant $\deg(f^{[i]})$. The result polynomial f recovered through Ben-Or/Tiwari cannot be correct if its degree is smaller than the degree of the most updated Newton interpolant $\deg(f^{[i]})$, because if it were, the target polynomial would have already been interpolated through Newton interpolation.

The racing algorithm that races Newton against a sparse algorithm[1] with early termination

Input:

$f(x)$**:** *a univariate black box polynomial over* $\mathbb{K}$.

δ**:** *an upper bound of* $\deg(f)$ *(for guarding against an infinite loop.)*

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. (Initialization.)

   $0 \neq p$ *random*[2], $p \in S \subseteq \mathbb{K}$;

   $a_0 \leftarrow b_0(p)$;

   $\tilde{a}_0 \leftarrow a_0$;

[1] Of course, the early termination of Newton is required to be proved for the interpolation points generated via the sparse racer algorithm. Section 5.1 provides proofs for the cases of sparse algorithms in the power, Pochhammer, and Chebyshev bases.

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

$new_{[race]} \leftarrow$ *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 $i = 1, \ldots, \delta + \eta$ ***Do***[3]

If $new_{[race]} =$ *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* $S \subseteq \mathbb{K}$*;*

[3]The upper bound $\delta + \eta$ is the default in our implementations. It can be set in other manners (see Section 7.2) but has to be no less than $\delta + \eta$ when δ is given as $\delta \geq \deg(f(x))$. This is to confine the overall black box probes and the interpolation efforts in order to avoid an infinite loop.

new$_{[race]} \leftarrow$ *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. (See whether the sparse racer algorithm finishes before the Newton interpolation.)

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 ζ many times in a row for the sparse racer, then

4. (Now we attempt to complete the sparse algorithm. If we fail to complete, we set new$_{\text{[race]}}$ as true in order to restart a race at a new random p.)

Complete the sparse racer algorithm;

If fail to complete, then

new$_{[race]} \leftarrow$ *true;*

End For;

If $\tilde{f}$ *is not defined then Fail;*

End.

Chapter 6

Hybrids of Zippel Algorithm and Other Improvements

In 1979, Richard Zippel gave an interpolation algorithm [24] that is efficient when the target polynomial is sparse in the multivariate case. It requires randomization, and interpolates one variable at a time through a dense univariate algorithm.

In this chapter, we present hybrids of Zippel algorithm by embedding our racing algorithms in Chapter 5 as the univariate interpolations into Zippel's scheme. As a result, the original Zippel algorithm is improved through adopting more efficient algorithms for its univariate interpolations.

As further refinements on the ideas of prunings in Zippel's approach through homogenization as described in [5], we implement *permanent prunings* and *temporary prunings* in our hybrids of Zippel algorithm. We also address some issues that arise from modular implementations.

6.1 The Zippel algorithm

A black box polynomial f can be represented as the following:

$$f(x_1, \ldots, x_n) = \sum_{(e_1,\ldots,e_n)\in J} c_{e_1,\ldots,e_n} x_1^{e_1} \cdots x_n^{e_n}, \tag{6.1}$$

where $0 \neq c_{e_1,\ldots,e_n} \in \mathbb{K}$, $J \subseteq (\mathbb{Z}_{\geq 0})^n$. Here $\mathbb{Z}_{\geq 0}$ is the set of non-negative integers. Note that $\#(J)$ is the number of non-zero terms in f.

The Zippel algorithm is based on the following idea: if the representation in (6.1) is sparse, than during the variable by variable interpolation, a zero coefficient is the image of a zero polynomial with high probability. We present the Zippel algorithm now:

The Zippel algorithm

Input:

f**:** *a multivariate black box polynomial over* $\mathbb{K}$.

$(x_1, \ldots, x_n)$**:** *an ordered list of variables in* f.

δ**:** *an upper bound of* $\deg(f)$.

Output:

$\tilde{f}(x_1, \ldots, x_n)$**:** *with high probability,* $\tilde{f} = f$.

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. (Interpolating one more variable: with high probability, we 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 $0 \neq$

$c_{e_1,\ldots,e_{i-1}} \in \mathbb{K}, J_{i-1} \subset \mathbb{Z}^{i-1}_{\geq 0}$.)

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 one more degree on the coefficient polynomials. We consider the coefficients of f in the variables $x_1, \ldots, x_{i-1}$ as polynomials in $\mathbb{K}[x_i]$ and interpolate those polynomials at $b_0, \ldots, b_{\delta_i}$ in the loop for k below. Since those polynomials are all of degrees no more than δ_i.)

   For $k = 0, \ldots, \delta_i$ ***Do***

   Randomly pick b_k *from a subset of* $\mathbb{K}$*;*

4. (For every b_k, by solving the following j_{i-1} by j_{i-1} transposed Vandermonde system, we can locate the value of every such coefficient polynomial evaluated 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***

$$\begin{aligned}\sum_{(e_1,\ldots,e_{i-1})\in J_{i-1}} &\gamma_{e_1,\ldots,e_{i-1},k}(\tilde{a}_1^j)^{e_i} \cdots (\tilde{a}_{i-1}^j)^{e_i} \\ &= f(\tilde{a}_1^j, \ldots, \tilde{a}_{i-1}^j, b_k, a_{i+1}, \ldots, a_n); \qquad (6.2)\end{aligned}$$

   End j ***For;***

   If the system is singular then report "Failure;"

Else solve the system for all $\gamma_{e_1,\ldots,e_{i-1},k}$ *(see [14];)*

5. (Next we interpolate j_{i-1} many univariate polynomials in x_i. Those polynomials are the coefficients, being viewed in $\mathbb{K}[x_i]$, of terms in the variables $x_1, \ldots, x_{i-1}$. We perform Newton algorithm to interpolate each polynomial through its values $\gamma_{e_1,\ldots,e_{i-1},k}$ of evaluations at b_k for $0 \le k \le \delta_i$.)

   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*

   $c^{[k]}_{i,(e_1,\ldots,e_{i-1})}(b_s) = \gamma_{e_1,\ldots,e_{i-1},s},\ 0 \le s \le k;$

   $$c^{[k]}_{i,(e_1,\ldots,e_{i-1})}(x_i) \leftarrow \sum_{s=0}^{k} c_{i,(e_1,\ldots,e_{i-1}),s} x_i^s;$$

   End $(e_1, \ldots, e_{i-1})$ ***For;***

   End k ***For;***

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 $c^{[\delta_i]}_{i,(e_1,\ldots,e_{i-1}),s} \ne 0$ *then*

   $c_{e_1,\ldots,e_{i-1},s} \leftarrow c^{[\delta_i]}_{i,(e_1,\ldots,e_{i-1}),s};$

   $J_i \leftarrow J_i \cup \{(e_1, \ldots, e_{i-1}, s)\};$

   End $(e_1, \ldots, e_{i-1}), s$ ***For;***

   Randomly pick $\tilde{a}_i$ *from a subset of* $\mathbb{K}$*;*

End i ***For;***

End.

The following example shows why Zippel algorithm can be efficient in a sparse case.

EXAMPLE 6.1. *Interpolate* $f(x,y) = 3x^5y^3 + 2x^5 + y^2 + 5 \in \mathbb{K}[x,y]$.

First fix y *at a random* $a \in S$ *and interpolate* $f(x,a)$ *with respect to* x. *After* $f(x,a)$ *is interpolated, we have*

$$f(x,a) = C_5(a)x^5 + C_0(a),$$

and through Zippel algorithm consider terms x, x^2, x^3, x^4 *all have a zero coefficient in* y. *We finish interpolating* f *by recovering* $C_5(y)$ *and* $C_0(y)$. *Figure 6.1 demonstrates this example in a diagram.*

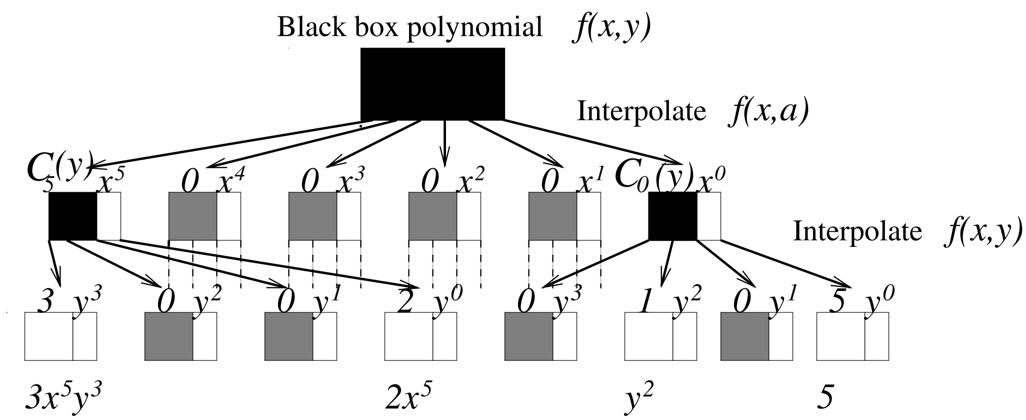


Figure 6.1*:* Interpolate $f(x,y) = 3x^5y^3 + 2x^5 + y^2 + 5$ via Zippel algorithm.

6.2 Prunings and hybrids of Zippel algorithm

In Zippel algorithm, each time after a new variable gets interpolated, the monomials with a zero coefficient are *pruned*; they are dropped and assumed to be zero polynomials in all other variables as well. If the anchor points are chosen at random, this is true with high probability.

Consider a new variable x_0, introduce the *homogenizing variable* into the representation of f in (6.1) as the following:

$$\begin{aligned}\tilde{f}(x_0, x_1, \ldots, x_n) &= f(x_0x_1, x_0x_2, \ldots, x_0x_n) \\ &= \sum_{(e_1,\ldots,e_n)\in J} (c_{e_1,\ldots,e_n} x_1^{e_1} \cdots x_n^{e_n}) x_0^{e_1+e_2+\cdots+e_n} \end{aligned} \tag{6.3}$$

Now instead of $f(x_1, \ldots, x_n)$, we interpolate $\tilde{f}(x_0,\ x_1,\ \ldots,\ x_n)$ and the anchor points a_1, …, a_n are randomly picked from a subset of $\mathbb{K}$. That is, we start interpolating $\tilde{f}(x_0,\ a_1,\ \ldots,\ a_n)$ with respect to x_0 and get a polynomial $f_0(x_0)$ in $\mathbb{K}[x_0]$. Through the Zippel algorithm, we can prune the support structure of f in $f_0(x_0)$.

Let $f_0(x_0) = \sum_{k=0}^{d} \gamma_{0,k} x_0^k$, by definition, $\gamma_{0,k}$ is the image of polynomial $c_k(x_1,$ $\ldots, x_n) \in \mathbb{K}[x_1, \ldots, x_n]$ evaluated at the anchor point, namely $c_k(a_1, \ldots, a_n) = \gamma_{0,k}$, and

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

Since every term in c_k is of degree k in $\mathbb{K}[x_1, \ldots, x_n]$, if f_0 is correctly interpolated, $\deg(f_0)$ in $\mathbb{K}[x_0]$ evaluates $\deg(f)$ in $\mathbb{K}[x_1, \ldots, x_n]$. And the degree of every non-zero monomial in $f_0(x_0)$ provides an upper bound for the degree of all the intermediate terms of its coefficient polynomial. Notice that in this section we have been using polynomial representations in the power basis, however, our discussion can be extended to coefficient polynomials in other bases as well.

Now, we refine the pruning idea in [5]. Comparing to Zippel's idea, we will do two more types of pruning so that we can possibly reduce the size of the transposed Vandermonde system in Step 4 of the Zippel algorithm in Section 6.1.

During the process of interpolating the homogenized polynomial in a variable by variable manner, as Step 2 in the Zippel algorithm of Section 6.1, with high probability we have

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

where $0 \neq c_{e_0, e_1, \ldots, e_{i-1}} \in \mathbb{K}$, $J_{i-1} \subset (\mathbb{Z}_{\geq 0})^i$.

For every term with $(e_0, \ldots, e_{i-1}) \in J_{i-1}$ such that $e_1 + \cdots + e_{i-1} = e_0$, the degree of the coefficient monomial in variables $x_1, \ldots, x_{i-1}$ has reached the total degree upper bound $e_0 = e_1 + \cdots + e_n$. That is, $c_{e_0, e_1, \ldots, e_{i-1}} x_1^{e_1} \cdots x_{i-1}^{e_{i-1}} x_0^{e_0}$ is an actual monomial in $\tilde{f}$. All such monomials have been fully interpolated and will not be changed in further interpolations with respect to other variables. We now let $g_{i-1}(x_0, x_1, \ldots, x_n)$ denote a polynomial summing up all such fully interpolated monomials in $x_0, \ldots, x_{i-1}$, and form the set 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 (6.2) in Step 4 of Zippel algorithm now becomes

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

Since $\#(J'_{i-1}) \leq \#(J_{i-1})$, by subtracting $g_{i-1}(\tilde{a}_0^j, \ldots, \tilde{a}_{i-1}^j, b_k, a_{i+1}, \ldots, a_n)$ from both sides of the transposed Vandermonde system (6.2), we can reduce the size of the system to be solved. All the monomials in g_{i-1} are called *permanently pruned* since they will not be interpolated in variables $x_i, \ldots, x_n$.

While we are interpolating the coefficients of terms in x_0, …, x_{i-1} of $\tilde{f}(x_0, \ldots,$ x_{i-1}, x_i, $a_{i+1}, \ldots$, $a_n)$, which are different polynomials in $\mathbb{K}[x_i]$, some of those coefficient polynomials might be interpolated via early termination before the degree bound is reached. Their values can be taken out of the loop in Step 3 before the rest of the coefficient polynomials are interpolated in x_i. As a result, the dimensions of the transposed Vandermonde system (6.2) can be further reduced. Those terms are *temporarily pruned* from all the remaining interpolations in variable x_i, nevertheless, they are not permanently pruned from all the remaining interpolations in other variables and still need to be interpolated with respect to x_{i+1}, …, x_n.

Without an input as the degree bound, the permanent prunings rely on the interpolation of homogenizing variable; while the temporary prunings can be carried out regardless whether the homogenizing variable is introduced

Our hybrids of Zippel algorithm use Zippel's technique as the outer loop. In addition, we introduce a homogenizing variable and perform both permanent and temporary term prunings accordingly. Yet, we allow the homogenizing variable modification to be "turned off" (see Section 7.3.) Within Zippel's scheme, each univariate interpolation implements a racing algorithm that races Newton against a sparse interpolation algorithm as in Section 5.2. As a result, our hybrid algorithms are more efficient due to the better univariate interpolations embedded.

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

Besides the efficiency in black box probes, our hybrid Zippel algorithm provides another possible advantage in modular implementations: when applying the univariate racing algorithm to race Newton against Ben-Or/Tiwari, we may greatly reduce the size of modulus required by the multivariate Ben-Or/Tiwari algorithm.

In order to control the size of the coefficients in the error locator polynomial of the embedded Berlekamp/Massey algorithm, Kaltofen *et al.* [13] 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 recovery of all the 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. Now consider a multivariate black box polynomial f and let $d = \deg(f)$; since 2 is the smallest prime, a sufficiently large modulo 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.

However, we notice that for the recovery of the terms in a sparse univariate polynomial $\tilde{f}(x)$ with degree $\tilde{d}$, a prime $\tilde{q}$ larger than $\tilde{d}$ can already provide a *sufficiently large* modulus. If 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 $\phi(\tilde{q}-1)$ primitive roots modulo $\tilde{q}$, with $u/\mathrm{loglog}u = O(\phi(u))$ for any integer u and ϕ denoting Euler's totient function [12, sec. 18.4], so a random residue has a fair chance of being a primitive root.

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 exponent computations highly likely lead to inconsistencies in later steps. An immediate inconsistency is to the degrees of the concurrent Newton interpolation. In that case, the algorithm recovers the univariate intermediate result by Newton interpolation or another restarted Ben-Or/Tiwari. If the false exponent is not caught then, with high likelihood the inconsistency shows up later, at the latest during the comparison of the final sparse interpolant with the black box input at an additional random point.

The trade-off between the size of the modulus and the number of black box probes in Ben-Or/Tiwari versus the univariate Ben-Or/Tiwari within Zippel, both with early termination, can now be quantified. Ignoring the size of the 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 (with early termination) may still outperform Zippel. Therefore, we add that at any stage in the variable by variable Zippel interpolation algorithm, we could complete the rest of the variables by a multivariate Ben-Or/Tiwari algorithm.

Also, we note that for a small finite coefficient field, say $\mathbb{Z}/2\mathbb{Z}$, one can switch to the coefficient domain $\mathbb{Z}/2\mathbb{Z}[x_n]$, where x_n is the last variable, and proceed modulo irreducible polynomials in $\mathbb{Z}/2\mathbb{Z}[x_n]$.

Chapter 7

Maple Implementation

The *ProtoBox* package is the Maple implementation of our new algorithms. We have implemented the early terminations with thresholds in Newton and Ben-Or/Tiwari algorithms, the racing algorithm that races Newton against the Ben-Or/Tiwari algorithm, its hybrid of Zippel interpolation with prunings, and the homogenizing modification.

We test the performance of *ProtoBox* in the polynomials listed in Table 7.1. Note that f_1 and f_2 are from [26, p. 100], along with f_3 and f_4 from [26, p. 102].

7.1 Black box probes

In our racing algorithm that races Newton against Ben-Or/Tiwari, we can "turn off" either racer algorithm by setting its threshold to ∞ and consequently forcing all the interpolations performed through the remaining active one.

Under Zippel's scheme, we compare the black box probes needed in interpolating different polynomials through different embedded univariate interpolations: Newton, Ben-Or/Tiwari, and the racing algorithm (all with threshold one for early

Table 7.1: Polynomials in *ProtoBox* performance tests.

$$
\begin{aligned}
f_1(x_1,\ldots,x_{10}) &= x_1^2x_3^3x_4x_6x_8x_9^2 + x_1x_2x_3x_4^2x_5^2x_8x_9 + x_2x_3x_4x_5^2x_8x_9 \\
&\quad + x_1x_3^3x_4^2x_5^2x_6^2x_7x_8^2 + x_2x_3x_4x_5^2x_6x_7x_8^2 \\
f_2(x_1,\ldots,x_{10}) &= x_1x_2^2x_4^2x_8x_9^2x_{10}^2 + x_2^2x_4x_5^2x_6x_7x_9x_{10}^2 + x_1^2x_2x_3x_5^2x_7^2x_9^2 \\
&\quad + x_1x_3^2x_4^2x_7^2x_9^2 + x_1^2x_3x_4x_7^2x_8^2 \\
f_3(x_1,\ldots,x_{10}) &= 9x_2^3x_3^3x_5^2x_6^2x_8^3x_9^3 + 9x_1^3x_2^2x_3^3x_5^2x_7^2x_8^2x_9^3 + x_1^4x_3^4x_4^2x_5^4x_6^4x_7x_8^5x_9 \\
&\quad + 10x_1^4x_2x_3^4x_4^4x_5^4x_7x_8^3x_9 + 12x_2^3x_4^3x_6^3x_7^2x_8^3 \\
f_4(x_1,\ldots,x_{10}) &= 9x_1^2x_3x_4x_6^3x_7^2x_8x_{10}^4 + 17x_1^3x_2x_5^2x_6^2x_7x_8^3x_9^4x_{10}^3 + 3x_1^3x_2^2x_6^3x_{10}^2 \\
&\quad + 17x_2^2x_3^4x_4^2x_7^4x_8^3x_9x_{10}^3 + 10x_1x_3x_5^2x_6^2x_7^4x_8^4 \\
f_5(x_1,\ldots,x_{50}) &= \textstyle\sum_{i=1}^{50} x_i^{50} \\
f_6(x_1,\ldots,x_5) &= \textstyle\sum_{i=1}^{5}(x_1+x_2+x_3+x_4+x_5)^i \\
f_7(x_1,x_2,x_3) &= x_1^{20} + 2x_2 + 2x_2^2 + 2x_2^3 + 2x_2^4 + 3x_3^{20}
\end{aligned}
$$

termination.) We choose relatively large moduli for the tests in order to reduce the possibility of hitting unlucky random numbers that might interfere with the behavior of each embedded algorithm. Our algorithms are all probabilistic, and we run each interpolation ten times to take the average of black box probes. In our test, for each interpolation we obtain the same count ten times.

Table 7.2 compares the black box probes needed in different embedded univariate interpolations. In average, the number of probes needed in a racing algorithm is no more than the minimum of either racer, which confirms that our racing algorithm takes advantage of both racer algorithms. Moreover, under Zippel's variable by variable approach, the racing algorithm always reflects the behavior of the winning algorithm. The winner might alternate between the two racers in different variables, therefore often the total black box count is less than the minimum of univariate interpolations completely through either Newton or Ben-Or/Tiwari.

Table 7.2: Black box probes needed for different embedded univariate interpolations.

	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

7.2 Thresholds

In addition to the thresholds for early terminations in Newton and Ben-Or/Tiwari, we introduce other thresholds in *ProtoBox* to further improve the probability of success and the throughput of the overall algorithm.

One way to further check the correctness of the interpolation result is to pick a few more random points and see whether the result agrees with the black box polynomial at those places. If they do not agree at any of them, we report an error message indicating the result cannot pass the post test. The number of random points generated for the post test is supplied as an optional argument "posttest_thresh" and is zero by default.

Within our probabilistic approach, random numbers are generated at different algorithm steps. Many times an unlucky random number in an intermediate step will abort the overall algorithm, which can be remedied by simply generating another random number. Nevertheless, in order to avoid a potential infinite loop, such efforts need to be bounded. In the case of Zippel algorithm, we may correctly interpolate all the monomials in the previous variables, yet due to a bad random point, two different terms may map to a same value yielding a singular transposed Vandermonde system in (6.2). In *ProtoBox*, the optional argument "mapmon_thresh," set by default to zero, defines the number of random points tried after encountering a singular system in (6.2). If the system (6.2) is non-singular at a random point during those tries, we proceed with the next step, otherwise report a failure after all the tries. Such threshold helps avoiding unnecessary failures and improving the throughput of the overall algorithm.

Considering the delay in our updating of Newton interpolant at repeated points, the optional argument "rndrep_thresh," default as zero, extends the upper bound for each univariate interpolation loop and lowers the failure rate due to the repeated points.

The chance to hit a bad random number in a large prime modulus is rare, thus the effects from the thresholds are less likely to be revealed when modulo a large prime. Consequently, we test *ProtoBox* on some relatively small moduli.

We totally have five arguments as different thresholds, and we test three combinations of varying thresholds listed in Table 7.3. In Table 7.3, Combination 1 corresponds to the default, while Combinations 2 and 3 are of higher thresholds.

Table 7.3: Sets of 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

We display the performance of the hybrid of Zippel algorithm, with racing Newton against Ben-Or/Tiwari algorithm embedded, on f_1, f_2, f_3, f_4 in Figures 7.1 through 7.4. As a probabilistic algorithm implementation, we test each interpolation 100 times for every modulus and threshold combination. The height of each rectangle, both solid and airy parts combined, reflects the number of accurate results in 100 runs. The three rectangles on each modulus correspond the results of three threshold combinations in Table 7.3, from left to right, Combinations 1 through 3. The accurate results are either the correct polynomial (number of times in 100 runs indicated as the solid part) or an error message (airy part of the rectangle.) When the result is not accurate, it is a polynomial that does not equal to the target polynomial.

For every polynomial and every modulus, when thresholds are increasing, often both the height of the rectangle and the solid part of the rectangles rise. However, not all the time the solid part rise at higher thresholds. We notice that higher thresholds require more evaluation points and checks, and that our record is just a random sample from 100 runs. We also observe the increasing size of modulus does not directly increase the overall throughput and probability of correctness. This is because it is the number of sufficiently large order elements, not the size of the modulus, that directly effects the success of the Ben-Or/Tiwari algorithm

(also see Section 7.3.)

7.3 The heuristic on small moduli

If a multivariate polynomial has degrees in each variable that are much smaller than the total degree, some very small moduli might suffice for interpolating such a polynomial provided we "turn off" the homogenizing variable modification. Because without the higher degree in the homogenizing variable interpolation, we only need to deal with lower degree interpolations in each variable.

We demonstrate the heuristic of polynomials being interpolated on some very small moduli and report the result after running 100 times in each listed case. In Table 7.4, 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 the result is the correct polynomial, under "!" the times an error message is returned, and "$\neq$" reports the times the result polynomial does not equal to the target polynomial. Note that when modulo 17, the terms with coefficient 17 in f_4 map to zero.

Table 7.4: Interpolations on very small moduli without homogenization modification after 100 runs.

	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 magnitude of modulus q provides the number of distinct points in the

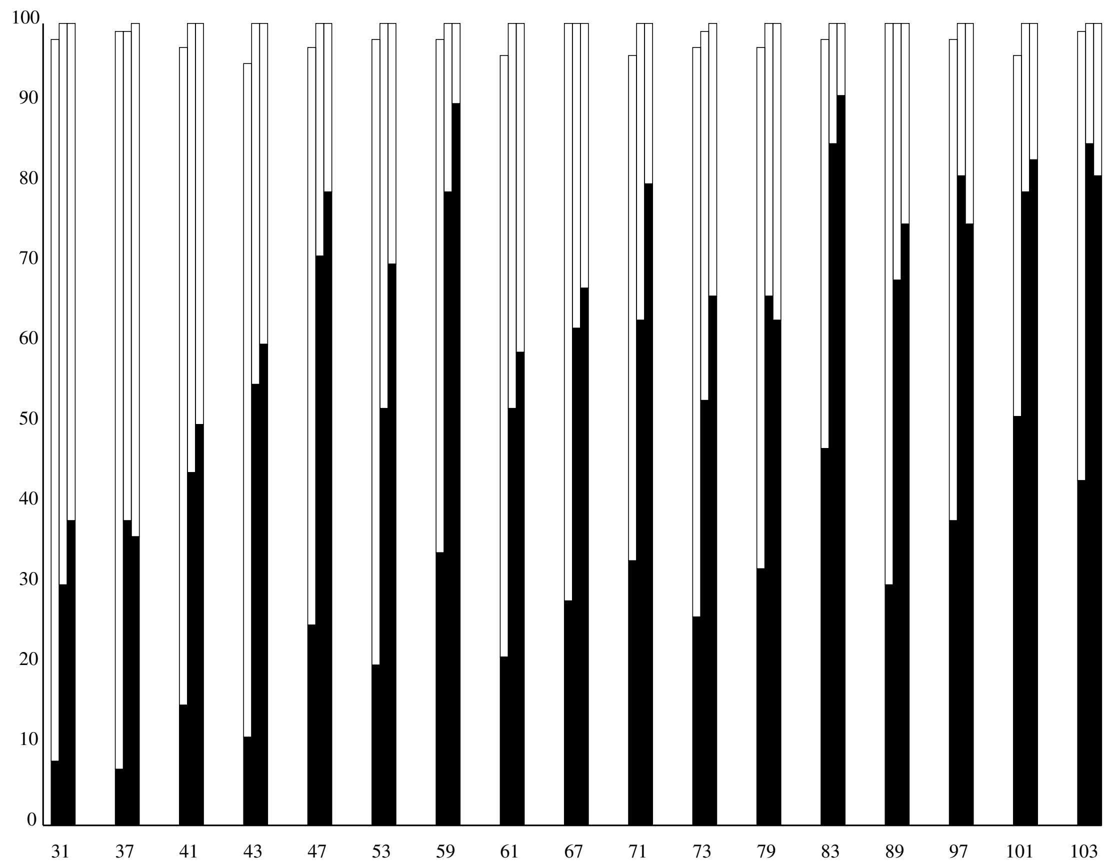


Figure 7.1: The performance of interpolating f_1 under different threshold combinations and moduli after 100 runs.

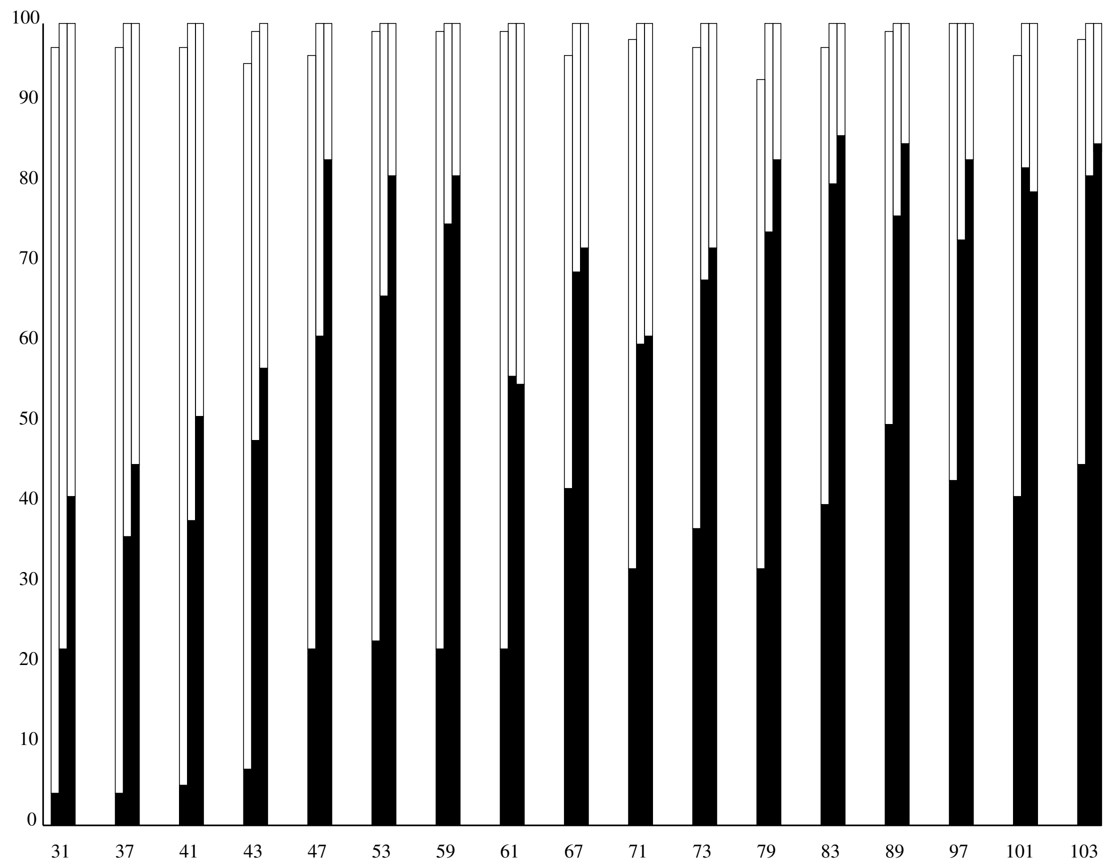


Figure 7.2: The performance of interpolating f_2 under different threshold combinations and moduli after 100 runs.

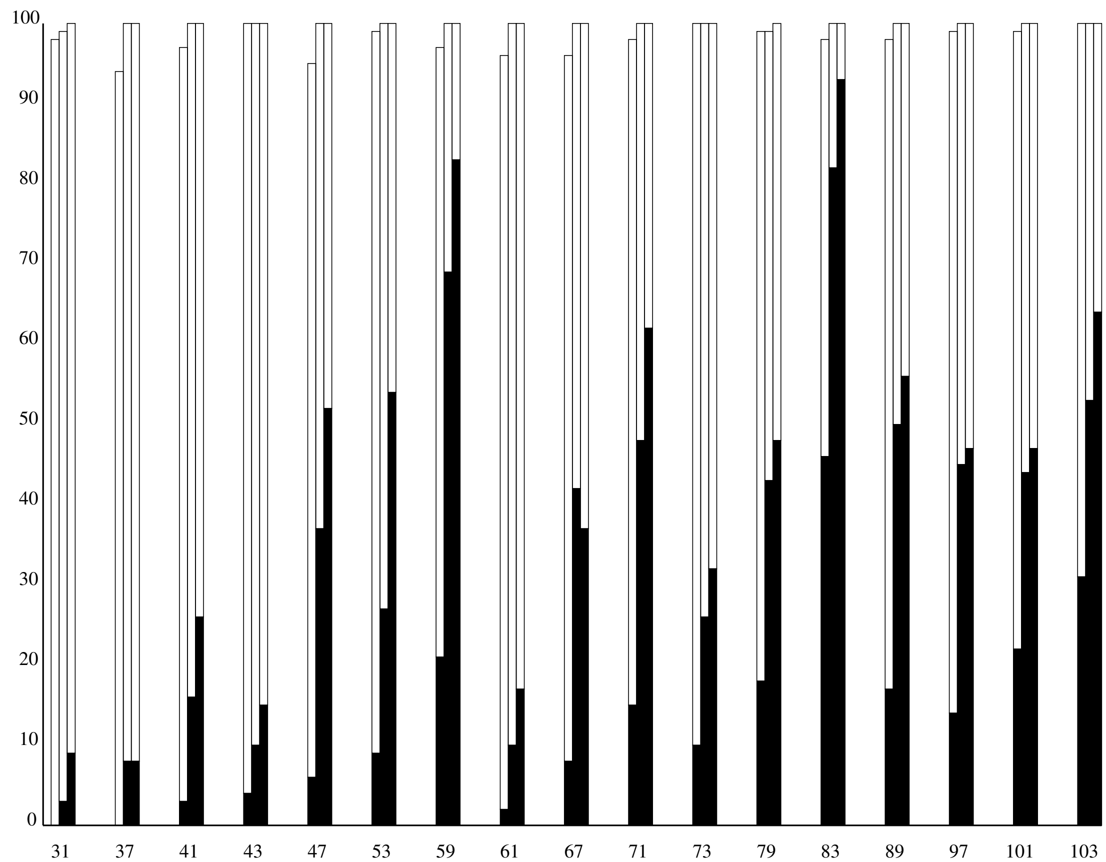


Figure 7.3: The performance of interpolating f_3 under different threshold combinations and moduli after 100 runs.

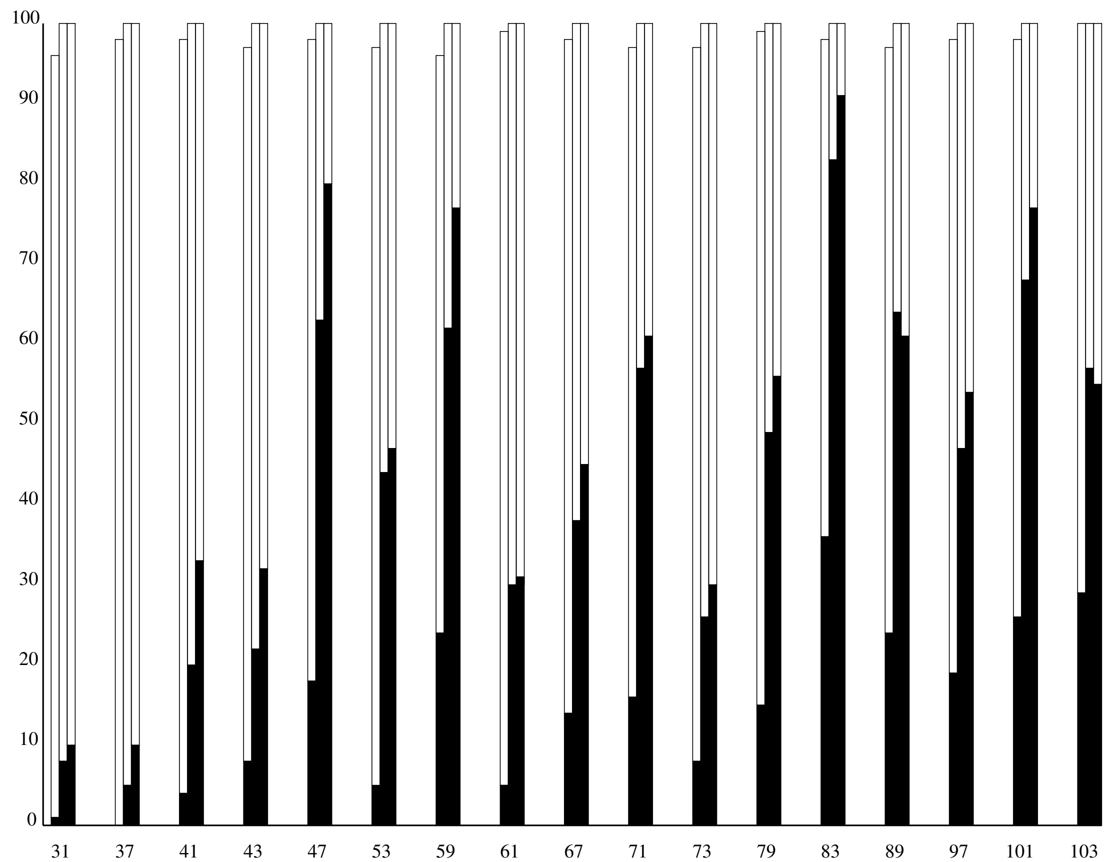


Figure 7.4: The performance of interpolating f_4 under different threshold combinations and moduli after 100 runs.

domain. Yet in the Ben-Or/Tiwari algorithm, in order to recover the term exponents, an element needs to generate enough distinct values from its powers. In other word, instead of the number of distinct elements, it is the amount of sufficient order elements that determines the success rate of the univariate Ben-Or/Tiwari algorithm.

Figures 7.5 and 7.6 display the order of each element in modulo 11 and 13. We use a dotted line to indicate the least order required for recovering every term exponent in a corresponding polynomial. For polynomials f_3 and f_4, there are less elements with sufficient order in modulo 13 than in modulo 11. As a result, the increase of modulus from 11 to 13 does not increase the success rate for these two polynomials.

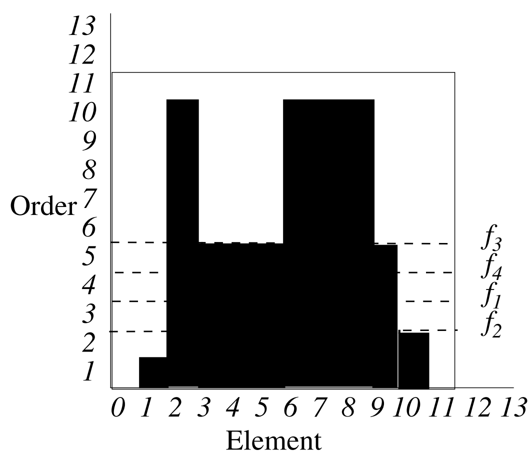


Figure 7.5: The order of elements in modulo 11 and the least order required in interpolating different polynomials through Ben-Or/Tiwari.

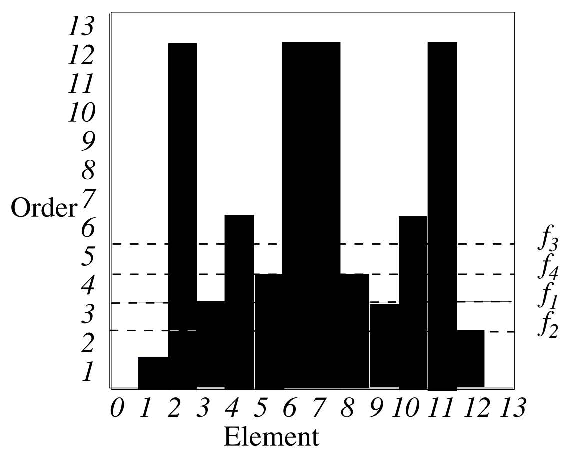


Figure 7.6: The order of elements in modulo 13 and the least order required in interpolating different polynomials through Ben-Or/Tiwari.

Chapter 8

Conclusions and Further Developments

We summarize our research efforts and contributions, then report our further developments.

8.1 Conclusions

In addition to the case of Newton interpolation, we developed the early termination strategy in the setting of sparse interpolation algorithms, in which the construction of different bases is taken into consideration. Among them, we have proved this strategy for the Ben-Or/Tiwari algorithm, sparse interpolations in the Pochhammer and Chebyshev bases. Within the sparse interpolation under the Chebyshev basis, we eliminated the complications in a symmetric Hankel-plus-Toeplitz matrix solver through randomization. Based on the early termination strategy, we proposed racing algorithms that race Newton interpolation against a sparse algorithm on the same set of evaluation points, and proved the early termination in the overall racing algorithms.

We introduced thresholds into the early termination and showed that higher thresholds can improve the lower bound of the probability of correctness in dense interpolations. The probability analysis of thresholds in sparse interpolations is more complicated and depends on certain conditional probabilities that require further investigations. Yet, we demonstrated some heuristic examples that higher thresholds weed out bad random choices.

We provided a new paradigm of racing algorithms: a better algorithm can be designed by racing two existing algorithms. Through racing two different algorithms, our racing algorithm performs as good as the winning algorithm in every case, and therefore provides a solution to the predicament of choosing between two algorithms who have different advantages. Moreover, the racing algorithms can cross check the results acquired from both racer algorithms and further improve the probability of correctness. By taking the advantage of the two racer algorithms, our racing algorithm is superior than either of them.

The original variable by variable Zippel algorithm interpolates each variable densely. We improved Zippel algorithm by embedding our racing algorithms as the univariate interpolations, and therefore enhanced the efficiency in univariate interpolations as well as the overall multivariate interpolations. We supplied an example in the paradigm of designing new algorithms from the hybrids of other existing algorithms.

Some of our algorithms are implemented in a Maple package, *ProtoBox*. The implementations include the racing algorithm that races Newton against Ben-Or/Tiwari with thresholds, its hybrid in Zippel, and the homogenization modification. In *ProtoBox*, we can “turn off” either of the racers for performance comparisons, and we verified from our tests the performance of our racing algorithm is better than either of the racer algorithms. In *ProtoBox*, besides the thresholds of early termination, we introduced other thresholds for the purpose of fault tolerance in order to prevent the failure of the overall process due to bad random choices.

We tested some bench mark polynomials under different combinations of thresholds, and presented heuristic examples of interpolating polynomials on very small moduli.

8.2 Future developments

Derived from the early termination of the Ben-Or/Tiwari algorithm, we present a heuristic on the early termination of sparse shifts.

Given a univariate[1] polynomial $f(x) = \sum_{i=1}^{t} c_i x^{e_i}$ with $c_i \neq 0$, we assume its most sparse shift occurs at α, that is,

$$f(x) = \sum_{i=1}^{\tilde{t}} \tilde{c}_i (x+\alpha)^{\tilde{e}_i}, \tilde{c}_i \neq 0. \tag{8.1}$$

Consider $y = x + \alpha$ and polynomial $g(\alpha, y) = f(y - \alpha)$ as the following:

$$g(\alpha, y) = f(y-\alpha) = \sum_{i=1}^{\tilde{t}} \tilde{c}_i (y - \alpha + \alpha)^{\tilde{e}_i} = \sum_{i=1}^{\tilde{t}} \tilde{c}_i y^{\tilde{e}_i}, \tag{8.2}$$

where $\tilde{c}_i$ are polynomials in α, that is, $\tilde{c}_i \in \mathbb{K}[\alpha]$.

Compare $g(\alpha, y)$ represented in (8.2) in variable y and the representation of f in its most sparse shift $(x + \alpha)$ in (8.1). If the most sparse shift α in (8.2) is known and p is chosen at random, performing the early termination Ben-Or/Tiwari

[1] As the Ben-Or/Tiwari algorithm is multivariate, we are optimistic of extending this approach to the multivariate case.

algorithm on sequence $g(\alpha, p)$, $g(\alpha, p^2)$, $g(\alpha, p^3)$, $\ldots$ will encounter the first zero discrepancy $\Delta_i = 0$ at $i = 2\tilde{t} + 1$ with high probability.

The value of α is unknown, and we turn to solve for the first α from the coefficient field such that $\Delta_i = 0$ at a random p. Notice now we perform the Berlekamp/Massey algorithm (Step 1 of the early termination Ben-Or/Tiwari algorithm in Section 3.3) on a sequence of polynomials $g(\alpha, p)$, $g(\alpha, p^2)$, $g(\alpha, p^3)$, $\ldots$, the discrepancies Δ_i become rational polynomials in α (see Table 8.1 for the comparisons of sparse shifts and the Ben-Or/Tiwari algorithm.) We need to prove that when p is random, the first α such that $\Delta_i(\alpha) = 0$ is the most sparse shift with high probability.

Table 8.1: Early termination in sparse shifts and the Ben-Or/Tiwari algorithm.

Early termination in sparse shifts	Early termination in Ben-Or/Tiwari
• At a random p, perform the Berlekamp/Massey algorithm on $g(\alpha, p)$, $g(\alpha, p^2)$, $g(\alpha, p^3)$, $\ldots$ $\Delta_i(\alpha) \in \mathbb{K}(\alpha)$ • Find the first $\alpha \in \mathbb{K}$ such that $\Delta_i(\alpha) = 0$ when $2L < i$	• At a random p, perform the Berlekamp/Massey algorithm on $f(p)$, $f(p^2)$, $f(p^3)$, $\ldots$ $\Delta_i \in \mathbb{K}$ • Early termination: the first $\Delta_i = 0$ when $2L < i$

If our claim can be proved, we have a polynomial time algorithm for early termination in sparse shifts. However, the efficiency of the algorithm needs to be addressed as well: either solving $\Delta_i(\alpha) = 0$ or factorizing the numerator of $\Delta_i(\alpha)$ is an excess in computing α.

Observe that once the most sparse shift is found, that is, $\Delta_{2\tilde{t}+1}(\alpha) = 0$, we have $\Delta_i(\alpha) = 0$ for all $i \geq 2\tilde{t} + 1$. In other word, once the linear factor $(z - \alpha)$

is introduced as a factor in the numerator of a discrepancy $\Delta_i(\alpha)$, it will stay in all the following discrepancies. As a result, we can deduce α from the polynomial GCD of the numerators of every two consecutive discrepancies, namely, $\gcd(\mathrm{numer}(\Delta_i(\alpha)), \mathrm{numer}(\Delta_{i+1}(\alpha)))$.

Recall the fact that when α is an integer and $(z-\alpha)$ a factor in the numerator of Δ_i, α is an integer factor of its constant term. Hence the most sparse integer shift α can be derived from finding the numeric GCD of the numerators of two consecutive discrepancies evaluated at 0, that is, $\gcd(\mathrm{numer}(\Delta_i(0)), \mathrm{numer}(\Delta_{i+1}(0)))$. (Yet, when $\alpha = 0$, this is the standard power basis. The correspondence may be investigated.)

A polynomial factor in the numerator of $\Delta_i(\alpha)$ and stays for all the following discrepancies is not necessarily a linear one. We may study the possible early termination in sparse decompositions as well as the sparse shifts.

For the wide applications and many useful properties, another possible research direction is the inquiry about early termination in the setting of the Bernstein bases. A polynomial $f(x)$ with $\deg(f) = n$ can be represented in the Bernstein bases as the following:

$$f(x) = \sum_{i=0}^{n} c_i B_{i,n}(x),$$

where

$$B_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i}.$$

Finally, we list some topics in our research that may be further studied: the interpolations on very small moduli in Section 7.3, the probability analysis of the algorithms with different thresholds and moduli, and as described in Section 6.3, the heuristic of switching from a small finite coefficient field, say $\mathbb{Z}/2\mathbb{Z}$, to a coefficient domain $\mathbb{Z}/2\mathbb{Z}[x_n]$, where x_n is the last variable, and proceeding modulo irreducible polynomials in $\mathbb{Z}/2\mathbb{Z}[x_n]$.

Literature Citations

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

[2] W. S. Brown and J. F. Traub. On Euclid's algorithm and the theory of subresultants. *J. ACM*, 18:505–514, 1971.

[3] P. Deldarte, Y. V. Genin, and Y. G. Kamp. A generalization of the levinson algorithm for hermitian toeplitz matrices with any rank profile. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-33 No. 4:946–971, 1985.

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

[5] A. Díaz and E. Kaltofen. FoxBox a system for manipulating symbolic objects in black box representation. In O. Gloor, editor, *ISSAC 98 Proc. 1998 Internat. Symp. Symbolic Algebraic Comput.*, pages 30–37, New York, N. Y., 1998. ACM Press.

[6] J. L. Dornstetter. On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inf. Theory*, IT-33(3):428–431, 1987.

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

[8] F. R. Gantmacher. *The theory of matrices*, volume 1. Chelsea publishing company, 1977.

[9] I. Gohberg and I. Koltracht. Efficient algorithm for toeplitz plus hankel matrices. *Integral Equations and Operator Theory*, 12:136–142, 1989.

[10] D. Yu. Grigoriev, M. Karpinski, and M. F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM J. Comput.*, 19(6):1059–1063, 1990.

[11] D. Yu. Grigoriev and Lakshman Y. N. Algorithms for computing sparse shifts for multivariate polynomials. In A. H. M. Levelt, editor, *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95*, pages 96–103, New York, N. Y., 1995. ACM Press.

[12] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Oxford Univ. Press, Oxford, 5 edition, 1979.

[13] E. Kaltofen, Lakshman Y. N., and J. M. Wiley. Modular rational sparse multivariate polynomial interpolation. In S. Watanabe and M. Nagata, editors, *ISSAC '90 Internat. Symp. Symbolic Algebraic Comput.*, pages 135–139. ACM Press, 1990.

[14] E. Kaltofen and Lakshman Yagati. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Symbolic Algebraic Comput. Internat. Symp. ISSAC '88 Proc.*, volume 358 of *Lect. Notes Comput. Sci.*, pages 467–474, Heidelberg, Germany, 1988. Springer Verlag.

[15] E. Kaltofen and A. Lobo. On rank properties of Toeplitz matrices over finite fields. In Lakshman Y. N., editor, *ISSAC 96 Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.*, pages 241–249, New York, N. Y., 1996. ACM Press.

[16] E. Kaltofen and B. Trager. 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, 1990.

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

[18] Lakshman Y. N. and B. D. Saunders. Sparse shifts for univariate polynomials. *Applic. Algebra Engin. Commun. Comput.*, 7(5):351–364, 1996.

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

[20] H. Murao and T. Fujise. Modular algorithm for sparse multivariate polynomial interpolation and its parallel implementation. In H. Hong, editor, *Proc. First Internat. Symp. Parallel Symbolic Comput. PASCO '94*, pages 304–315, Singapore, 1994. World Scientific Publishing Co.

[21] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, 1980.

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

[23] Z. Zilic and K. Radecka. On feasible multivariate polynomial interpolations over arbitrary fields. In S. Dooley, editor, *ISSAC 99 Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.*, pages 67–74, New York, N. Y., 1999. ACM Press.

[24] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. EUROSAM '79*, volume 72 of *Lect. Notes Comput. Sci.*, pages 216–226, Heidelberg, Germany, 1979. Springer Verlag.

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

[26] R. E. Zippel. *Probabilistic algorithms for sparse polynomials.* PhD thesis, Massachusetts Inst. of Technology, Cambridge, USA, September 1979.