# Abstract

COMER, MATTHEW T. Error Correction for Symbolic and Hybrid Symbolic-Numeric Sparse Interpolation Algorithms. (Under the direction of Erich L. Kaltofen.)

We introduce error correction to two problems of sparse polynomial interpolation. In the first problem, we recover a $t$-sparse polynomial $f(x)$ (in the power basis) from values $f(\omega^{i-1})$, $i = 1, 2, \ldots, 2t(2E+1)$, where $\omega$ is a field element of our choice, and at most $E$ values contain random/misleading errors. Our algorithm is adapted from the linear recurrence-based algorithm of [Prony III (1795), *J. de l'École Polytechnique*, 1:24-76]. In the exact setting, error correction is implemented directly in the computation of the minimal generator of the linearly recurrent sequence $\{f(\omega^{i-1})\}$. In the numeric setting, where every evaluation is perturbed by a small relative "noise" error and at most $E$ evaluations are perturbed by large "outlier" errors, we appeal to the results of [Kaltofen and Lee 2003, *J. Symbolic Comput.*, 36(3-4):365-400] and [Kaltofen, Lee, and Yang 2011, *Proc. 2011 Internat. Workshop on Symbolic-Numeric Comput.*, pages 130-136] for computation of the approximate minimal generator.

In the second problem, we compute a shift $s$ that will yield the sparsest representation of a polynomial $f(x)$ in the shifted power basis $(x-s)^k$. To this end, two algorithms are presented. The first algorithm computes the representation from polynomials $f(\omega^i + z) \in \mathbb{F}[z]$. The polynomials are interpolated densely, using Blahut's decoding method for Reed-Solomon error-correction, then the shift $s$ and sparsity $t$ are computed from the error-free "values" $f(\omega^{i-1} + z)$, where now $i = 1, 2, \ldots, 2t+1$. The second step employs the results of [Giesbrecht, Kaltofen, and Lee 2003, *J. Symbolic Comput.*, 36(3-4):401-424]. The second algorithm takes advantage of the representation $f(x) = \sum_{i=1}^{t} c_i (x-s)^{e_i}$ being the Taylor expansion of $f(x)$ at $x = s$. Erroneous evaluations are detected by first interpolating $f(x)$ densely, then comparing all evaluations to their corresponding values of the interpolant. Both algorithms have adaptations for the numeric setting; the first algorithm relies on results of [Hutton, Kaltofen, and Zhi 2010, *Proc. 2010 Internat. Symp. Symbolic Algebraic Comput.*, pages 227-234] to determine the shift and sparsity.

We also present an algorithm for counting square singular Hankel matrices with entries from a finite field, where some anti-diagonals on or above the main diagonal (equivalently, on or below the main anti-diagonal) may be fixed to prescribed values. The count follows one of three formulas, depending on which anti-diagonals are fixed and to what values. The result is obtained by extending a result of [Kaltofen and Yuhasz 2013a, *ACM Trans. Algorithms*, to appear], where the Berlekamp/Massey algorithm is executed on the sequence of anti-diagonal values of the matrix.

Error Correction for Symbolic and Hybrid Symbolic-Numeric Sparse Interpolation Algorithms

by

Matthew T. Comer

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

Raleigh, North Carolina

2013

APPROVED BY:

Erich L. Kaltofen
North Carolina State University
Chair of Advisory Committee

Irina A. Kogan
North Carolina State University

Wen-shin Lee
University of Antwerp

Michael F. Singer
North Carolina State University

Ralph C. Smith
North Carolina State University

# Dedication

To Nana, who always supported me from afar, even through the end

# Biography

Matthew T. Comer earned a Bachelor of Science degree in Mathematics from the University of Connecticut in 2007, graduating magna cum laude from the Honors Scholar Program. He then earned a Master of Science degree in Mathematics from North Carolina State University in 2009, continuing to be accepted by Erich L. Kaltofen as a Ph.D. student in the research area of Computer Algebra. Matthew successfully defended this dissertation in April 2013.

# Acknowledgements

First, I thank my advisor for his mentoring and support during productive times, his patience and understanding during the challenging times, and for sharing his original ideas, for which I will always be grateful. In addition, I thank him for his advice and guidance well after the defense, both in time and scope. Thank you, Erich.

I thank my Ph.D. committee for their time and consideration, and for their comments and suggestions during the preliminary and final oral exams. Thank you, Irina Kogan, Wen-shin Lee, Jon Rust, Michael Singer, Charles Smith, and Ralph Smith.

I thank our collaborators and friends for the opportunities to share the research process with them and to learn from them. Thank you, Brice Boyer, Feng Guo, and Clément Pernet.

I thank everyone who supported me in each facet of my research experiences, which have taken me not only around the USA, but also China and France. From travel to mentoring to conversation to advice, thank you, all.

I thank the professors of my courses at North Carolina State University, who taught me not only the content and skills appropriate for each course, but also general skills and strategies that have helped me throughout all of the mathematics that I have encountered in my career thus far. Thank you, all.

I thank my fellow students at North Carolina State University for forming a support group beyond measure. I will always fondly reminisce our experiences together, both curricular and extracurricular. Thank you, friends.

I thank professors Keith Conrad and Joe McKenna at the University of Connecticut for their guidance and advice about upper-level math courses, undergraduate research, and graduate school. Thank you, both.

I thank all of my friends and professors at the University of Connecticut for inspiring my passion for mathematics, and further inspiring me to apply to graduate school. Thank you, all.

I thank my family for their love and support throughout all of my education, especially the graduate school years. Thank you, all.

I thank Heather Cannon for her constant support through all of the opportunities and accomplishments, as well as tolerance through the frustration and stress. Thank you, Heather.

# Table of Contents

# List of Tables

# List of Figures

Chapter One

# Introduction

Error-correcting decoding is ubiquitous in modern computing. With polynomial interpolation being used as a key step for error-correcting decoding, it is a natural extension to account for errors in the process of polynomial interpolation itself. Without errors, the process of recovering a polynomial from a sampling of its points has been studied in various contexts. For example, to interpolate a polynomial $f(x)$ of degree less than $d$, one could obtain $d$ evaluations of $f(x)$ (at distinct $x$-values) and then solve a Vandermonde system to produce a representation of $f(x)$. It is important to note that given *any* set of $d$ points (again, for distinct $x$-values), a Vandermonde system will yield a polynomial of degree less than $d$ that passes through each point: one cannot simply inspect the proposed interpolant to determine if some evaluations were faulty.

As with many schemes of error-correcting (de)coding, the introduction of redundancy enables the problem of error-correcting interpolation to become more tractable. Specifically, if we wish to interpolate a polynomial of degree less than $d$, but we have a bound $E$ on the maximum number of erroneous evaluations in a given set, then obtaining $d + 2E$ evaluations guarantees that there can be only one *$E$-error interpolant* (i.e., a polynomial that passes through all but $k$ points, where $k \leq E$). For suppose there were two $E$-error interpolants, $f(x)$ and $g(x)$, each of degree less than $d$. Then it must be that $f$ and $g$ agree for at least $d$ points, which implies $f = g$ (because the polynomial $f - g$, also of degree less than $d$, has at least $d$ roots). As a counter-example for $d = E = 1$, consider the $d + 2E - 1 = 2$ points $(x_1, y_1)$ and $(x_2, y_2)$, where $x_1 \neq x_2$ and $y_1 \neq y_2$. Then $f(x) = y_1$ and $g(x) = y_2$ are each a 1-error interpolant for the set of two points. One can generalize this example for arbitrary $d$ and $E$ (starting with $d - 1$ points common to $f$ and $g$, then including $E$ points unique to each of $f$ and $g$) to show that $d + 2E$ points are necessary and sufficient to guarantee uniqueness of the interpolant.

Even when we can certify that all evaluations of a polynomial are correct, there exists the

optimization problem of minimizing the number of evaluations needed to recover the unique interpolant. Knowing only that a polynomial is of degree less than $d$, one generally requires $d$ points for interpolation. However, if we know that the polynomial has few terms compared to its degree – that is, the polynomial is *sparse* – then the minimum number of evaluations may be decreased, due to a phenomenon discovered by Prony during the French Revolution. In [Prony III (1795)], it is shown that one can interpolate a sum of $t$ exponential functions, say, $f(x) = \sum_{i=1}^{t} c_i b_i^x$, from only $2t$ evaluations. Here, Prony used not an arbitrary set of evaluations, but rather a sequence of evaluations at consecutive multiples of a single $x$-value: $0, \alpha, 2\alpha, \ldots, (2t-1)\alpha$.

Prony showed that this sequence of evaluations satisfies a particular *recurrence relation*: if we denote the evaluations by $a_i = f(i\alpha)$, then

$$\gamma_0 a_{0+i} + \gamma_1 a_{1+i} + \ldots + \gamma_{t-1} a_{t-1+i} + a_{t+i} = 0, \quad i = 0, 1, \ldots, t-1,$$

for some constants $\gamma_i$. (More accurately, the sequence satisfies a *homogeneous linear recurrence relation with constant coefficients*, which will continue to be called simply a recurrence relation, and such a sequence will be called simply a *linearly recurrent sequence*.) By examining the ratios $a_1/a_0, a_2/a_0, \ldots, a_{2t-1}/a_0$, Prony derived explicit formulas for the $\gamma_i$ and showed that the polynomial $\Lambda(\lambda) = \left(\sum_{i=0}^{t-1} \gamma_i \lambda^i\right) + \lambda^t$ factors as $\Lambda(\lambda) = \prod_{i=1}^{t} (\lambda - b_i^\alpha)$. Thus, from only the sequence of evaluations, one can determine (through formulas and polynomial factorization) all of the bases $b_i$. Equivalently, if we re-write $f(x) = \sum_{i=1}^{t} c_i b^{e_i x}$, where $e_i = \log_b(b_i)$, then we have $\Lambda(\lambda) = \prod_{i=1}^{t} (\lambda - b^{e_1 \alpha})$, so that we wish to recover the exponents $e_i$.

A special case is $t = 1$: the recurrence relation is given by $\gamma_0 a_i = a_{i+1}$, so that the sequence is geometric. Indeed,

$$\frac{f\big((n+1)\alpha\big)}{f(n\alpha)} = \frac{c_1 b^{(n+1)\alpha e_1}}{c_1 b^{n\alpha e_1}} = \frac{(b^{\alpha e_1})^{n+1}}{(b^{\alpha e_1})^n} = b^{\alpha e_1},$$

from which one can recover $e_1$ (knowing $b$ and $\alpha$). Then the only unknown quantity in $f(x) = c_1 b^{e_1 x}$ is $c_1$, which can be recovered immediately from $a_0 = c_1$. For general $t$, one computes the exponents $e_1, e_2, \ldots, e_t$ from the $2t$ evaluations, then one can compute the coefficients $c_1, c_2, \ldots, c_t$ from any subset of $t$ evaluations (e.g., by solving a $t \times t$ Vandermonde system). To relate this result to polynomials, we note that if we substitute $\alpha = \log_b(\omega)$, then this becomes

$$\frac{f\big(\log_b(\omega^{n+1})\big)}{f\big(\log_b(\omega^n)\big)} = b^{\log_b(\omega) e_1} = \omega^{e_1} = \frac{g(\omega^{n+1})}{g(\omega^n)}, \qquad \text{for } g(x) = c_1 x^{e_1}.$$

Thus, we can interpolate a polynomial with $t$ terms (called a *$t$-sparse polynomial*) from evaluations at consecutive powers of a single $x$-value: $1, \omega, \omega^2, \ldots, \omega^{2t-1}$. Whether we interpolate

sparse functions in an exponential or polynomial basis, the focus is the computation of the recurrence relation among $a_0, a_1, \ldots, a_{2t-1}$. Furthermore, if some of the evaluations are erroneous, then we must have some method of accounting for errors in the recovery of the recurrence relation. By introducing redundancy as in the general/dense polynomial example above, we are able to detect and correct errors, but we appeal to a more modern algorithm for computing a recurrence relation.

Seventeen decades after Prony's discovery, Berlekamp introduced a decoding method for multiple-error-correcting binary Bose-Chaudhuri-Hocquenghem (BCH) codes; Algorithm 7.4 of [Berlekamp 1968] is presented for determining the shortest feedback-shift-register (i.e., minimal recurrence relation) of a sequence of digits representing the (possibly erroneous) transmission of a codeword. (Such a minimal recurrence relation exists due to the module structure of the space of all sequences being acted-on by the univariate polynomial ring; see, e.g., Section 12 of [von zur Gathen and Gerhard 1999].) In Berlekamp's formulation, the recurrence relation gives the coefficients of the "error locator polynomial" (also called the *generator* of the sequence), whose roots are the reciprocals of Galois field elements that represent the locations where errors occurred.

One immediate advantage over Prony's method is that Berlekamp's algorithm iterates through the sequence, processing one entry at a time: for each $n = 1, 2, \ldots$, the minimal generator is found for the first $n$ entries of the sequence, using the computations that produced the generator for the first $n-1$ entries (where the generator for $n = 0$ is the polynomial 1 by convention). Berlekamp proved that the degree of the minimal generator (equivalently, length of the minimal recurrence relation) could be computed at the beginning of each iteration by testing if the generator for $n$ entries continues to generate the $(n+1)$-st entry.

Whereas Berlekamp's algorithm was intended for sequences that arise in BCH decoding, Massey generalized Berlekamp's algorithm to the realm of arbitrary linearly generated sequences, granting the name "Berlekamp Iterative Algorithm" (see Section III of [Massey 1969]), which later adopted the name Berlekamp/Massey Algorithm. Through direct manipulation of recurrence relations, Massey showed that the degree of the minimal generator follows a concise rule: if we denote by $L_n(\mathbf{a})$ the degree of the minimal generator for the first $n$ entries of the sequence $\mathbf{a}$, then $L_{n+1}(\mathbf{a}) = \max\{L_n(\mathbf{a}), n + 1 - L_n(\mathbf{a})\}$. This property allows the iterative step of the Berlekamp/Massey Algorithm to depend on a simple switch: whether or not there is a *length change* (i.e., $L_{n+1} > L_n$). It is the nature of this length change test that allows us to introduce error correction to the minimal generator computation: eventually in a linearly generated sequence, a single error will cause a large increase in the length of the generator.

In Chapter 2, we formally define the problem of error correction for linearly generated sequences: suppose that a sequence $a_0, a_1, \ldots$ of elements from a field, $\mathbb{F}$, has a minimal generator $\Lambda$ of degree $t$. Given a sequence $b_0, b_1, \ldots, b_{n-1}$ where $b_i \neq a_i$ for at most $E$ values of $i$, we wish to

recover $\Lambda$ (the *intended* generator) and correct the errors in the sequence. (Counter-)examples 1 and 2 show that one must have $n \geq 2t(2E+1)$ for the problem to have a unique solution, at least when $\mathrm{char}(\mathbb{F}) \neq 2$. Given this lower bound on $n$, the determination of the intended generator amounts to a "majority vote" of generator candidates obtained by executing the Berlekamp/ Massey algorithm on each contiguous subsequence of $2t$ entries; this follows from Lemma 3.

After recovering the intended generator, detection and correction of errors is not as simple as applying the generator to the entire sequence $\{b_i\}$, changing entries as necessary to fit the recurrence. Rather, it may be that an erroneous "block" of $2t$ entries yields the intended generator. For example, consider the two sequences

$$\begin{matrix} \bar{0},\ 0,\ 1,\ \bar{0},\ 0,\ 2 \\ \bar{0},\ 1,\ 1,\ \bar{0},\ 2,\ 2 \end{matrix}, \quad \bar{0} = \overbrace{0, \ldots, 0}^{t-2 \text{ times}},$$

both of which yield $-2+\lambda^t$ as minimal generator. Suppose the entire intended sequence $\{a_i\}$ were a continuation of the first sequence (using $-2+\lambda^t$ to generate further entries), but the erroneous sequence $\{b_i\}$ were formed by substituting the first block as above. Testing the generator with the first block as initial input, the recurrence would fail at locations that agree with the intended sequence. We call such a block as above a "deceptive block"; Theorem 4 proves that a deceptive block will cause more than $E$ such failure locations, so that one will know to use a different block as initial input to the generator test. It is also proven that the method works when only a bound $T \geq t$ is known.

Thus having an error-correction strategy for linearly generated sequences, Prony's method becomes part of an error-correcting algorithm for sparse interpolation. We describe the full algorithm as an adaptation of Ben-Or's & Tiwari's algorithm for error-free sparse interpolation (see [Ben-Or and Tiwari 1988]), which applies Prony's method to the multivariate case, substituting a distinct prime for each variable.

We also explore a numeric/approximate adaptation of the univariate case, where we choose $\omega$ to be an appropriate complex root of unity (for numerical stability); Section 2.6 describes the algorithm and presents some experiments where every evaluation is perturbed by a (relatively) small "noise" error, while only a few evaluations are perturbed by a (relatively) large "outlier" error. It should be noted that when the evaluations are only approximate, the Berlekamp/ Massey algorithm cannot be directly implemented, as it includes an explicit test for zero (when determining whether a recurrence relation holds); thus, we appeal to the results of [Kaltofen and Lee 2003] and [Kaltofen, Lee, and Yang 2011] to compute the approximate minimal generator. Also, it may be that the majority vote is inconclusive, so we appeal to oversampling: several sequences of evaluations are obtained, substituting different values for $\omega$.

In the exact case where the field $\mathbb{F}$ is a finite field, choosing $\omega$ to be a primitive root of

unity causes the sparse interpolation with errors problem to be dual to that of decoding Reed-Solomon codes, as shown in Section 2.3. With a result of [Blahut 1983], we present the sparse interpolation with errors problem from the perspective of error-correcting coding.

Whereas Chapter 2 addresses the problem of sparse polynomial interpolation with errors in the power basis $(1, x, x^2, \ldots)$, Chapter 3 addresses a similar problem, but in the *shifted* power basis: $1, (x-s), (x-s)^2, \ldots$ for some constant value $s$, which is a priori unknown. This results in an optimization problem: find the value of $s$ that yields the fewest terms in the shifted representation $f(x) = \sum_{i=1}^{t} c_i (x-s)^{e_i}$.

For a first interpolation algorithm, we appeal to the results of [Kaltofen and Lee 2003]. Consider that $g(x) = f(x+s)$ is sparse in the (un-shifted) power basis, so the sequence of evaluations $a_0 = g(1), a_1 = g(\omega), a_2 = g(\omega^2), \ldots$ is linearly generated by $\Lambda(\lambda) = \prod_{i=1}^{t} (\lambda - \omega^{e_i})$. Equivalently, the Hankel matrix $H_m = [a_{i+j-2}]_{i,j=1}^{m}$ is singular for $m = t+1$, with a column relation given by the coefficients of $\Lambda$. Furthermore, Theorem 4 of [Kaltofen and Lee 2003] states that, with high probability, $H_m$ is non-singular for $1 \leq m \leq t$ when $\omega$ is randomly sampled from a sufficiently large subset of $\mathbb{F}$.

In [Giesbrecht, Kaltofen, and Lee 2003], by defining two sequences $a_i = f(\omega_1^i + z)$ and $b_i = f(\omega_2^i + z)$ with $z$ an indeterminate for $s$ and $\omega_1, \omega_2$ each randomly sampled, it is shown that (again, with high probability) if one computes the GCD of the determinants of the $m \times m$ Hankel matrices formed by the two sequences, then $z + s$ will be the first non-trivial GCD, exactly when $m = t+1$. To introduce error correction in the function evaluations, we pre-condition the evaluations by performing dense interpolation with errors (based on Blahut's method for decoding Reed-Solomon codes) on $f(\omega_i + z)$, as a dense polynomial in $z$.

If we view the representation $f(x) = \sum_{i=1}^{t} c_i (x-s)^{e_i}$ as the Taylor expansion of $f$ at $x = s$, then we see that $s$ must be a root of all derivatives of $f$ whose order is not some $e_i$. Thus, we search for the root where the most derivatives vanish, which immediately gives all values $e_i$ as well. If $f$ has no more than half the possible number of terms, then Theorem 1 of [Lakshman Y. N. and Saunders 1996] guarantees that the value of $s$ is unique and rational. We show that one need compute only the $2t$ highest-order derivatives (the highest being the $(\deg(f)-1)$-st derivative in this context) to find $s$, with the possibility of early termination. To compute these derivatives without having knowledge of $s$, one would need methods of dense interpolation with errors to compute the $2t$ highest-order terms of $f$. Due to the necessary division in Taylor coefficients, we restrict this algorithm to fields of characteristic zero.

We give numeric adaptations of both algorithms, where again we assume that every evaluation is perturbed by a (relatively) small "noise" error, while only a few evaluations are perturbed by a (relatively) large "outlier" error. In addition, we focus on the error-locating stage of Blahut's method for decoding Reed-Solomon codes. As described in Section 3.3, locating outliers in a sequence of dense polynomial evaluations (in the exact setting) relies on a sparse

polynomial interpolation, where the exponents of the sparse polynomial correspond to the error locations in the sequence of dense polynomial evaluations. In particular for the numeric setting, we explore the relative difference between the magnitudes of noise and outlier in the 1-outlier case, giving a sufficient condition that will guarantee recovery of the error location. We show some experiments of what can happen when the magnitude of an outlier approaches that of general noise.

Chapter 4 addresses the problem of enumerating square singular Hankel matrices with entries from a finite field $\mathbb{F}_q$, specifically when some of the anti-diagonals may be fixed to particular values. Though not directly related to sparse interpolation, the solution is intimately connected to the Berlekamp/Massey algorithm, as a Hankel matrix $H = [a_{i+j-2}]_{i,j=1}^{n}$ can be represented by the sequence $a_0, a_1, \ldots, a_{2n-2}$. By executing the Berlekamp/Massey algorithm on the sequence, one can determine which leading principal submatrices of $H$ are non-singular, due to Lemma 2 of [Kaltofen and Yuhasz 2013a]: there will be a length change, $L_m > L_{m-1}$, exactly when the $L_m \times L_m$ leading principal submatrix is non-singular.

We use this property to determine, for any singular Hankel matrix, a unique anti-diagonal that, if changed to any of the other $q-1$ values in $\mathbb{F}_q$, will cause the matrix to be non-singular. This induces a map from the set of singular Hankel matrices to the power set of non-singular Hankel matrices (see Section 4.3). By showing that the images of the map will partition the set of non-singular matrices, we can prove the already-established property that a fraction of $1/q$ of all Hankel matrices are singular. In addition, because the singularity-controlling anti-diagonal is found on or below the main anti-diagonal, we can determine singularity when any of the anti-diagonals on or above the main anti-diagonal are fixed (while the other anti-diagonals range over $\mathbb{F}_q$). Theorem 5 shows the three cases for counting square singular Hankel matrices, which depend on the choice of fixed anti-diagonals and the values of those anti-diagonals. The result is analogous when some anti-diagonals are fixed on or below the main anti-diagonal, but not when anti-diagonals above and below the main anti-diagonal are fixed.

We also provide counts for the number of block-Hankel matrices with *block generic rank profile*, i.e., matrices $H$ of rank $mr$ (where blocks are size $m \times m$) such that $\text{rank}(H_k) = mk$ for $k = 1, 2, \ldots, r$, where $H_k$ is the $k \times k$ block leading principal submatrix of $H$. We compare this result to that of [Kaltofen and Lobo 1996], where Toeplitz matrices of generic rank profile (i.e., block size 1) are counted.

Chapter Two

# Sparse Polynomial Interpolation with Noise and Outliers (SPINO)

This chapter is [Comer, Kaltofen, and Pernet 2012], except for the modifications listed in Note 2.

## 2.1 Introduction

The problem of reconstructing a sparse polynomial from its values stands at the nexus of symbolic and numeric computing. The astounding success of the numerical versions of the exact symbolic sparse interpolation algorithms [Grigoriev and Karpinski 1987; Ben-Or and Tiwari 1988; Kaltofen, Lakshman Y. N., and Wiley 1990; Kaltofen and Lee 2003], namely the "GLL"-algorithm [Giesbrecht, Labahn, and Lee 2009], in medical signal processing by Annie Cuyt, Wen-shin Lee, and others (see, e.g., http://smartcare.be) leads to a statistical problem variant: allow for outlier measurement errors in addition to noise.

### 2.1.1 Exact Sparse Interpolation With Errors

We begin by investigating the exact, symbolic problem formulation: we give algorithms that reconstruct a sparse polynomial from its values even if some values are incorrect. More precisely, we give algorithms that from $2T(E+1)$ values at points of our choice (evaluations at consecutive powers of an element, e.g., roots of unity) can compute a $t$-sparse interpolant, where the input includes upper bounds $T \geq t$ and $E \geq e$, where $e$ evaluations are incorrect. The output can be a list of at most $E$ interpolants, but necessarily containing the correct one. We can produce a unique $t$-sparse interpolant from $2T(2E+1)$ evaluations at points of our choice. Our method

is based on Ben-Or's and Tiwari's (1988) algorithm, which computes the sparse support (the set of non-zero terms) via a linear recurrence. We compute the linear generators for multiple segments of the sequence to expose the faults in some of the segments.

We also study on its own the Berlekamp/Massey algorithm on input sequences with faulty elements, and show that uniqueness of a linear generator of degree $t$ cannot be guaranteed from a sequence segment of $< 2t(2e+1)$ consecutive elements with $e$ errors. For sequence segments of $\geq 2T(2E+1)$ consecutive elements with $e \leq E$ errors, we not only can easily recover the unique linear generator of degree $t \leq T$ where the bounds $T, E$ are input, but we can also locate and correct the errors. For sequences arising in sparse interpolation, fewer elements may suffice for a unique interpolant, as we do not have the corresponding counterexamples.

The exact dense interpolation problem with errors is studied in [Khonji, Pernet, Roch, Roche, and Stalinsky 2010]. The exact sparse problem is also investigated in [Saraf and Yekhanin 2011], where it is assumed, as we do not, that the support of the sparse polynomial is known. However, Saraf's CCC '11 talk has motivated our investigations.

### 2.1.2 The Nature of Noise

In numerical data, noise is a result of imprecise measurement or of floating point arithmetic. It is assumed that there is some accuracy in the data. In digital data, noise spoils several bits and accuracy is measured as having a small Hamming distance. When approximating a transcendental function by a Taylor series or Padé fraction, noise is the model error of the inexact representation. Hybrid symbolic-numeric algorithms have traditionally assumed that the input scalars have with some relative error been deformed. For instance, if one allows substantial oversampling, a sparse polynomial can be recovered from numerical noisy values, where each value has a relative error of 0.5 [Giesbrecht and Roche 2011], perhaps even 0.99, with still more oversampling. Here we consider errors in the Hamming distance sense, i.e., several incorrect values without assumption on any accuracy. But we can combine our model with that of numerical noise in all other values, hence we borrow the statistical term *outlier* for the nature of these errors.

One could also presume, as Annie Cuyt does, that the black box mechanism that produces the values for our interpolant, with possibly both inaccuracies and outlier errors, represents an unknown, possibly irrational, function that the sparse polynomial model approximates. However, the model fitting algorithm cannot distinguish errors in the model from noise and outliers in the values, at least not if the produced errors are independent on the locality of the input probe.

### 2.1.3 Numeric Sparse Interpolation

Linear recurrence-based sparse interpolation goes back to the French revolution [Prony III (1795)]. In fact, Prony's algorithm is Ben-Or's/Tiwari's if one replaces the polynomial variables by exponential functions $x = e^y$. For imaginary replacements $x = e^{iy}$ one obtains periodic sparse sinusoid (finite Fourier) signals (cf. [Cadzow 1988]). Prony's algorithm disappeared from numerical analysis books for reason of conditioning [Wen-shin Lee, private communication]. The probabilistic analysis of the randomized early termination algorithm [Kaltofen and Lee 2003] together with analysis of the distribution of the corresponding condition numbers [Giesbrecht, Labahn, and Lee 2009; Kaltofen, Lee, and Yang 2011] now justifies its use, even numerically. Ill-conditioning arises precisely at the point of termination, and condition number estimation or stochastic input sensitivity analysis can be used to identify this point. Already Prony's paper shows that the sparse terms can have negative, and possibly rational, exponents.

### 2.1.4 Numeric Outliers

Important variants of our algorithms can deal with numerical noise, from floating point or empirical input values, and outliers. We demonstrate that the Prony-GLL style algorithms [Giesbrecht, Labahn, and Lee 2009; Kaltofen, Lee, and Yang 2011] are applicable to our majority voting approach by describing our early-terminated numeric version of the Ben-Or/Tiwari Algorithm; some particular examples are given in Section 2.6.

### 2.1.5 The Ben-Or/Tiwari Algorithm

We briefly review Ben-Or's/Tiwari's algorithm in the setting of univariate sparse polynomial interpolation. Let $f$ be a univariate polynomial, $m_j$ its distinct terms, $t$ the number of terms, and $c_j$ the corresponding non-zero coefficients:

$$f(x) = \sum_{j=1}^{t} c_j x^{e_j} = \sum_{j=1}^{t} c_j m_j \neq 0, e_j \in \mathbb{Z}.$$

**Theorem 1.** [Ben-Or and Tiwari 1988] *Let $b_j = \omega^{e_j}$, where $\omega$ is a value from the coefficient domain to be specified later, let $a_i = f(\omega^i) = \sum_{j=1}^{t} c_j b_j^i$, and let $\Lambda(\lambda) = \prod_{j=1}^{t}(\lambda - b_j) = \lambda^t + \gamma_{t-1}\lambda^{t-1} + \cdots + \gamma_0$. The sequence $(a_0, a_1, \ldots)$ is linearly generated by the minimal polynomial $\Lambda(z)$.*

The Ben-Or/Tiwari algorithm then proceeds in the four following steps:

1. Find the minimal-degree generating polynomial $\Lambda$ for $(a_0, a_1, \ldots)$, using the Berlekamp/ Massey algorithm.

2. Compute the roots $b_j$ of $\Lambda$, using univariate polynomial factorization.

3. Recover the exponents $e_j$ of $f$, by repeatedly dividing $b_j$ by $\omega$.

4. Recover the coefficients $c_j$ of $f$, by solving the transposed $t \times t$ Vandermonde system

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

By Blahut's theorem [Blahut 1979; Massey and Schaub 1988], the sequence $(a_i)_{i \geq 0}$ has linear complexity $t$, hence only $2t$ coefficients suffice for the Berlekamp/Massey algorithm to recover the minimal polynomial $\Lambda$. In the presence of errors in some of the evaluations, this fails. Our work shows how to overcome that obstruction.

## 2.2 Length Bounds for Computing Linear Generators From Sequences with Error

### 2.2.1 Necessary Length of the Input Sequence

**Example 1.** Suppose the base field is $\mathbb{F}$, where $\mathrm{char}(\mathbb{F}) \neq 2$. Given $e, t \geq 1$, let $\bar{0}$ denote the zero vector of length $t-1$. Consider the following sequence of length $4t$:

$$(\bar{0}, 1, \bar{0}, \underbrace{1}_{-1}, \bar{0}, 1, \bar{0}, \overbrace{-1}^{1}) \quad \begin{matrix} \Rightarrow -1 + \lambda^t \\ \Rightarrow 1 + \lambda^t \end{matrix}.$$

Here, the underbrace and overbrace represent distinct "corrections" that would yield two different minimal linear generators ($1 + \lambda^t$ and $-1 + \lambda^t$, respectively). If we concatenate this sequence with itself $e-1$ times, then append the sequence $(\bar{0}, 1, \bar{0})$, the result is a sequence of length $4et + 2t - 1 = 2t(2e+1) - 1$, where two sets of "corrections" would each yield a different minimal linear generator. Thus, at least $2t(2e+1)$ sequence entries are necessary to guarantee a unique solution.

**Example 2.** Suppose again that the base field is $\mathbb{F}$, where $\mathrm{char}(\mathbb{F}) \neq 2$. Let $\alpha$ be an $e$-th root of unity, for $e$ even, and consider the following sequence of $2(2e+1) - 1$ entries:

$$(1, \alpha, \dots, \alpha^{e-1}, 1, -\alpha, \dots, (-\alpha)^{e-1}, 1, \alpha, \dots, \alpha^{e-1}, 1, -\alpha, \dots, (-\alpha)^{e-1}, 1).$$

If the odd powers of $-\alpha$ are changed to odd powers of $\alpha$, then $-\alpha + \lambda$ is the minimal linear generator. However, if the odd powers of $\alpha$ are changed to odd powers of $-\alpha$, then $\alpha + \lambda$ is the minimal linear generator. Note that there are $e$ changes made to the sequence in either case. Thus, at least $2(2e+1)$ entries are required to guarantee a unique solution when $t = 1$.

For the case $t > 1$, we place the zero-vector of length $t-1$ in between each entry above, and on each end of the sequence. Again, we see that at least $2t(2e+1)$ entries are required to guarantee a unique solution.

As an explicit example, let $e = 4$, $t = 1$, and consider the following sequence:

$$(1, \underbrace{\boldsymbol{i}}_{-\boldsymbol{i}}, -1, \underbrace{-\boldsymbol{i}}_{\boldsymbol{i}}, 1, \overbrace{-\boldsymbol{i}}^{\boldsymbol{i}}, -1, \overbrace{\boldsymbol{i}}^{-\boldsymbol{i}}, 1, \underbrace{\boldsymbol{i}}_{-\boldsymbol{i}}, -1, \underbrace{-\boldsymbol{i}}_{\boldsymbol{i}}, 1, \overbrace{-\boldsymbol{i}}^{\boldsymbol{i}}, -1, \overbrace{\boldsymbol{i}}^{-\boldsymbol{i}}, 1).$$

When $e$ is odd, we can take the example for $e+1$, remove the last $4t$ entries, then change the last $\alpha^e$ to $-\alpha^e$. The resulting sequence has $2t(2(e+1)+1) - 1 - 4t = 2t(2e+1) - 1$ entries, as in the case where $e$ is even. As an explicit example, let $t = 1$ and $e = 3$. We start with the example for $e = 4$:

$$(1, \underbrace{\alpha}_{-\alpha}, \alpha^2, \underbrace{\alpha^3}_{-\alpha^3}, 1, \overbrace{-\alpha}^{\alpha}, \alpha^2, \overbrace{-\alpha^3}^{\alpha^3}, 1, \underbrace{\alpha}_{-\alpha}, \alpha^2, \underbrace{\alpha^3}_{-\alpha^3}, 1, \overbrace{-\alpha}^{\alpha}, \alpha^2, \overbrace{-\alpha^3}^{\alpha^3}, 1)$$

then modify it to

$$(1, \underbrace{\alpha}_{-\alpha}, \alpha^2, \underbrace{\alpha^3}_{-\alpha^3}, 1, \overbrace{-\alpha}^{\alpha}, \alpha^2, \overbrace{-\alpha^3}^{\alpha^3}, 1, \underbrace{\alpha}_{-\alpha}, \alpha^2, \overbrace{-\alpha^3}^{\alpha^3}, 1).$$

### 2.2.2 Bounds on Generator Degree and Location of Last Error

**Lemma 1.** *Suppose the infinite sequence $(a_0, a_1, \ldots)$ has monic minimal linear generator $\Lambda(\lambda)$ with $\Lambda(0) \neq 0$. If we introduce errors into the sequence, with the last error at entry $k$, then $\lambda^k \Lambda(\lambda)$ is the monic minimal linear generator of the infinite sequence $(b_0, b_1, \ldots, b_{k-1}, a_k, a_{k+1}, \ldots)$, where $b_{k-1} \neq a_{k-1}$.*

*Proof.* Write the monic minimal linear generator of the erroneous sequence as $\lambda^l \Xi(\lambda)$, where $\Xi(0) \neq 0$. The polynomial $\lambda^k \Lambda$ will generate the erroneous sequence, which implies $\lambda^l \Xi \mid \lambda^k \Lambda$. Note that for $m = \max\{l, k\}$, $\Lambda$ and $\Xi$ are both linear generators for $(a_m, a_{m+1}, \ldots)$, but $\Lambda$ is minimal by Lemma 3, so $\Lambda \mid \Xi$.

Let $\Xi = \Lambda\Gamma$. Then $\Gamma$ is monic and has a non-zero constant term (because both $\Lambda$ and $\Xi$ do), so then $\lambda^l \Lambda\Gamma \mid \lambda^k \Lambda$, say $\lambda^l \Lambda\Gamma p = \lambda^k \Lambda$. Then we have $\lambda^l \Lambda(\Gamma p - \lambda^{k-l}) = 0$, which implies

$\Gamma p = \lambda^{k-l}$ because $\Lambda \neq 0$, where $l \leq k$ due to the divisibility statements above. Thus, both $\Gamma$ and $p$ are monomials in $\lambda$, but this forces $\Gamma = 1$, so $\Xi = \Lambda$.

To finish the proof, note that if $l < k$, then $\lambda^l \Xi = \lambda^l \Lambda$ would fail to generate $(b_0, b_1, \ldots, b_{k-1}, a_k, a_{k+1}, \ldots, a_{k+t-1})$, so we must have $l = k$ and the minimal linear generator of the erroneous sequence is $\lambda^k \Lambda$. □

**Theorem 2.** *Suppose the infinite sequence $(a_0, a_1, \ldots)$ has monic minimal linear generator $\Lambda(\lambda)$ with $\Lambda(0) \neq 0$ and $\deg(\Lambda) = t$, and suppose that errors are introduced into the sequence, with the last error occurring at entry $k$. If only bounds for $t$ and $k$ are known, say, $t \leq T$ and $k \leq K$, then Algorithm 1 and Algorithm 2 can be used to recover $\Lambda$ and the intended sequence, if given $\min\{K + 2T, k + t + K + T\}$ entries of the erroneous sequence.*

*Proof.* If given $K + 2T$ entries, then calling Algorithm 1 on the sequence $(a_{K+2T-1}, a_{K+2T-2}, \ldots, a_K)$, which contains no error, will return $\Lambda_{\text{sr}}$ (i.e., the scaled reciprocal polynomial of $\Lambda$) by Lemma 4. We can then call Algorithm 2 with $\Lambda_{\text{sr}}$, $K$, and $(a_{K+T-1}, a_{K+T-2}, \ldots, a_0)$ as input.

However, if $T \gg t$, then $K + 2T$ entries may be more than necessary, as we may be able to find $\Lambda$ from an early-terminated Algorithm 1 running in the forward direction. Specifically, if a sequence has a monic minimal linear generator of unknown degree $d$ and $d \leq \delta$, where $\delta$ is known, then Algorithm 1 will compute the (reciprocal polynomial of the) generator after processing $2d$ entries. After processing $d + \delta$ entries, any future discrepancy would cause the degree of the generator to exceed $\delta$, so the algorithm may stop; see Theorem 5 of [Kaltofen and Yuhasz 2013a]. Here, we have $d = k + t$ (by Lemma 1) and $\delta = K + T$, so Algorithm 1 may stop after processing $k + t + K + T$ entries, which will be fewer than $K + 2T$ entries when $T > k + t$. □

## 2.3 Reed-Solomon decoding

We show in this section that the problem of sparse interpolation with errors is dual with that of Reed-Solomon decoding. This rapid overview on error correcting codes is not comprehensive. The reader can refer to [Moon 2005] for a treatment of the subject in appropriate details.

### 2.3.1 Reed-Solomon as Evaluation Codes

The popular Reed-Solomon codes can be defined (as in their original presentation by [Reed and Solomon 1960]) as evaluation interpolation codes. Let $K$ be the finite field $\mathbb{F}_q$, set $n = q - 1$ and let $\xi$ be a primitive $n$-th root of unity in $K$.

A message is a vector of $k < n$ symbols from $\mathbb{F}_q$, forming a polynomial $f = a_0 + a_1 X + \cdots + a_{k-1} X^{k-1}$ of degree at most $k - 1$. The encoding of the message is the vector of the $n$

evaluations of $f$ in the consecutive powers of $\xi$:

$$c = \mathrm{Ev}(f) = (f(\xi^0), f(\xi^1), \ldots, f(\xi^{n-1})) = V_\xi f, \qquad \text{where } V_\xi = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & \xi & \ldots & \xi^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi^{n-1} & \ldots & \xi^{(n-1)^2} \end{bmatrix}$$

is the Vandermonde matrix of the evaluation points $1, \xi, \xi^2, \ldots \xi^{n-1}$. For simplicity, we equate a polynomial with the vector of its coefficients. This procedure defines the $(n, k)$-Reed-Solomon code as the set $\mathcal{C}$ of evaluations of all polynomials of degree at most $k-1$:

$$\mathcal{C} = \{(f(\xi^0), f(\xi^1), \ldots, f(\xi^{n-1})) \mid f \in \mathbb{F}_q[X], \deg(f) \leq k\}.$$

Decoding works by a simple interpolation. Suppose that a weight $e$ error $\varepsilon$ affects the communication, so that one receives the message $c' = c + \varepsilon$, where $e$ coefficients of $\varepsilon$ are non zero. A consequence of the BCH theorem is that if $E = \lfloor \frac{n-k}{2} \rfloor$ and $e \leq E$, then $c$ is the unique codeword at distance no more than $E$ to $c'$. This makes it possible to correct and decode a codeword affected by up to $E$ errors.

Let $I$ be the interpolation function associated to the points $\xi, \xi^2, \ldots, \xi^n$. As $I$ is linear, it satisfies $I(c') = I(c) + I(\varepsilon) = f + I(\varepsilon)$. In particular, the last $n-k$ monomials of $I(c')$ are those of $I(\varepsilon)$, which form a contiguous subsequence of the Discrete Fourier Transform of $\varepsilon$ called the syndrome. Blahut's Theorem [Blahut 1983] states that the D.F.T. of a vector of weight $t$ is linearly generated by a polynomial of degree less than $t$. Hence applying the Berlekamp/Massey algorithm on these $n - k \geq 2t$ coefficients will recover this generating polynomial, called error locator: it vanishes at the $\xi^i$ where errors occurred.

### 2.3.2 Relation with Sparse Interpolation with Errors

To summarize, the Reed-Solomon decoding problem is the following:

*Given $c' \in \mathbb{F}_q^n$, find $f$ of degree less than $k$ and $\varepsilon$ of weight $t$ such that $c' = V_\xi f + \varepsilon$.*

This problem has a unique solution provided $t \leq \frac{n-k}{2}$. In comparison, the sparse interpolation problem can be written as

*Given $c' \in \mathbb{F}_q^n$, find an error $f$ and a sparse polynomial $\varepsilon$ of weight $t$ such that $c' = f + V_\xi \varepsilon$.*

The Vandermonde matrix $V_\xi$ satisfies $V_\xi^{-1} = V_{\xi^{-1}}/n$, hence its inverse is a (scaled) Vandermonde matrix and corresponds, up to sign, to the evaluation function in the powers of $\xi^{-1}$. Left-multiplying $c' = f + V_\xi \varepsilon$ by $V_\xi^{-1}$ makes it a Reed-Solomon decoding problem (based on the

primitive root $\xi^{-1}$). This problem has a unique solution provided that the $2t$ trailing coefficients of the error vector $f$ are zeros. Hence, a $t$-sparse polynomial can be recovered from $n$ evaluations provided that the last $2t$ of them are not erroneous.

Now as the location of a consecutive sub-sequence of $2t$ non-faulty evaluations is a priori unknown, one needs to inspect several segments of length $2t$. In the Reed-Solomon decoding viewpoint, one tries to decode that same sequence with several Reed-Solomon codes (of varying length, dimension and set of evaluation points). This adaptive decoding is similar to the one proposed in [Khonji, Pernet, Roch, Roche, and Stalinsky 2010] for decoding CRT codes. In this context, the uniqueness of the solution is no longer guaranteed. In the following sections, we will propose two decoding algorithms: the first decodes with the shortest sequence of evaluation points, but can return a list a several candidates, and the second guarantees a unique solution, but a with a longer sequence of evaluations, however optimal, with respect to the lower bound proven in Section 2.2.

## 2.4 A Fault-Tolerant Berlekamp/Massey algorithm

We address the problem of recovering the minimal-degree monic polynomial that generates a sequence of $n$ elements, where at most $E$ entries have been modified by errors; we address this problem first by a heuristic: it returns a list of at most $E$ candidates, but that necessarily contains the correct one.

We recall in Algorithm 1 the specification of the well-known Berlekamp/Massey algorithm to find the monic minimal generating polynomial of a sequence; this algorithm can recover the monic generating polynomial of least degree $t$ from $2t$ consecutive sequence entries.

**Algorithm 1:** Berlekamp/Massey

| | |
|---|---|
| **Input:** | $(a_0, a_1, \ldots, a_{n-1})$, a sequence of field elements |
| **Output:** | $\Lambda(\lambda) = \sum_{i=0}^{L_n} \gamma_i \lambda^i$, a monic polynomial of minimal degree $L_n \leq n$ such that $\sum_{i=0}^{L_n} \gamma_i a_{i+j} = 0$ for $j = 0, 1, \ldots, n - L_n - 1$ |

**Lemma 2.** *Let $S = (a_0, a_1, \ldots)$ be an infinite sequence generated by a minimal linear relation of degree $t$, so that $a_{i+t} = \sum_{j=0}^{t-1} \gamma_j a_{i+j}$ for all $i \geq 0$ with $\gamma_0 \neq 0$. Let $\Lambda(\lambda) = \lambda^t - \sum_{j=0}^{t} \gamma_j \lambda^j$. Then calling Algorithm 1 on any subsequence $(a_i, \ldots, a_{i+2t-1})$ will return $\Lambda$.*

In a sequence of $2T(E+1)$ elements affected by at most $E$ errors, there has to be at least one clean subsequence of length $2T$. However, some unlikely combination of errors may lead to a block of $2T$ elements for which the Berlekamp/Massey algorithm will produce a degree $t \leq T$

generating polynomial that is not the original generating polynomial. We can discriminate against some of these false positive cases by checking the generating polynomial against the remaining elements in the sequence. This is done by Algorithm 2, which can then be used in Algorithm 3 to select only the polynomials of degree $t$ that generate a sequence of Hamming distance less than $E$ to the input sequence. Unfortunately, some false positive may still generate the sequence with less than $E$ errors, as shown in Section 2.2. Section 2.5 will address this issue with a stronger assumption on the length of the sequence: $n \geq 2T(2E+1)$.

**Algorithm 2:** Sequence Clean-up (SCU)

**Input:** $\Lambda(\lambda) = \sum_{i=0}^{t} \gamma_i \lambda^i$, a monic polynomial of degree $t$, such that $\Lambda(0) \neq 0$
**Input:** $E$, the maximum number of changes to the sequence allowed
**Input:** $(a_0, \ldots, a_{n-1})$ a sequence of field elements where $n \geq 2t+1$.
**Input:** $k \leq n - 2t - 1$, the initial position for clean-up
**Output:** $((c_0, \ldots, c_{n-1}), e)$, a sequence of field elements, such that either $e > E$ or $(c_0, \ldots, c_{n-1})$ is a linearly recurrent sequence of field elements, generated by $\Lambda$, of Hamming distance at most $E$ to $(a_0, \ldots, a_{n-1})$, such that $(c_k, \ldots, c_{k+2t-1}) = (a_k, \ldots, a_{k+2t-1})$

SCU1 Initialize $(c_0, \ldots, c_{n-1}) \leftarrow (a_0, \ldots, a_{n-1})$.
*If $(c_0, \ldots, c_{n-1})$ is already linearly recurrent of length/degree $t$ or less, then the output will be $((a_0, \ldots, a_{n-1}, 0)$.*

SCU2 Initialize $i \leftarrow k + 2t$ and $e \leftarrow 0$.

SCU3 While $i \leq n-1$ and $e \leq E$, perform steps SCU4-SCU5.
*We re-write sequence entries to force a linear recurrence of length/degree $t$ or less.*

SCU4 If $\sum_{j=0}^{t} \gamma_j c_{j+i-t} \neq 0$, then set $c_i \leftarrow -\sum_{j=0}^{t-1} \gamma_j c_{i+j-t}$ and $e \leftarrow e+1$.

SCU5 Increment $i \leftarrow i+1$.

SCU6 Set $i \leftarrow k-1$.

SCU7 While $i \geq 0$ and $e \leq E$, perform steps SCU8-SCU9.
*These steps are similar to steps SCU4-SCU5, but process the sequence in the reverse direction.*

SCU8 If $\sum_{j=0}^{t} \gamma_j c_{j+i} \neq 0$, then set $c_i \leftarrow -\sum_{j=1}^{t} \frac{\gamma_j}{\gamma_0} c_{i+j}$ and $e \leftarrow e+1$.

SCU9 Decrement $i \leftarrow i-1$.

SCU10 Return $(c_0, \ldots, c_{n-1}), e$.

**Algorithm 3:** Fault Tolerant Berlekamp/Massey (FTBM)

**Input:** $(a_0, \ldots, a_{n-1})$, a sequence of field elements
**Input:** $T$, an upper bound for $t$, the degree of the monic minimal generator
**Input:** $E$, an uppper bound on the number of errors
**Output:** $L$, a list of pairs $((c_0, \ldots, c_{n-1}), \Lambda)$ formed by a sequence of distance less than $E$ to $(a_0, \ldots, a_{n-1})$ and its minimal degree monic generating polynomial. We require $\Lambda(0) \neq 0$.

FTBM1 Initialize $L \leftarrow [\,]$ (the empty list) and $i \leftarrow 0$.

FTBM2 While $i \leq \left\lfloor \frac{n}{2T} \right\rfloor - 1$, perform Steps FTBM3-FTBM6.

FTBM3 Call Algorithm 1 on $(a_{2Ti}, \ldots, a_{2Ti+2T-1})$; store the output as $\Lambda$.

FTBM4 If $\Lambda(0) \neq 0$, then call Algorithm 2 on $(\Lambda, E, (a_0, \ldots, a_{n-1}), 2Ti)$; store the output as $((c_0, \ldots, c_{n-1}), e)$.

FTBM5 If $e \leq E$, then append $((c_0, \ldots, c_{n-1}), \Lambda)$ to the list $L$.

FTBM6 Increment $i \leftarrow i + 1$.

FTBM7 Return $L$.

**Theorem 3.** *If $n \geq 2T(E+1)$, Algorithm 3 run on a sequence altered by at most $E$ errors returns a list of less than $E$ polynomials containing the generating polynomial of the initial clean sequence. It runs in $O(T^2E)$ arithmetic operations.*

*Proof.* As $n \geq 2T(E+1)$, there has to be an iteration where the segment $(a_{2Ti}, \ldots, a_{2Ti+2T-1})$ has no error, and for which the Berlekamp/Massey algorithm will return the correct polynomial and the call to Algorithm 2 will fix the sequence with less than $E$ corrections. The complexity is that of $E$ applications of the Berlekamp-Massey algorithm on $2T$ elements. □

Note that the condition on the length of the sequence $n \geq 2T(E+1)$ is the tightest possible in order to apply a syndrome decoding. Indeed if $n < 2T(E+1)$ some errors of weight $E$, e.g. $e = \sum_{i=1}^{\lfloor \frac{n}{2T} \rfloor} e_{2Ti}$ where $e_i$ denotes the $i$-th canonical vector, are such that no length $2T$ consecutive sub-sequence of evaluations is error-free, and Ben-Or/Tiwari's algorithm can not be applied on any part of such a sequence.

On the other hand, this algorithm can also return a list of several candidates. Each reconstructed sparse polynomial can be tested on a few more evaluations until only one remains.

## 2.5 The Majority Rule Berlekamp/Massey algorithm

If one has $e$ errors in a sequence of $2t(2e+1)$ linearly generated elements by a generator of degree $t$, then $e+1$ "blocks" of $2t$ elements must have the same correct generator. However, it is not so clear that with the correct generator one can locate and correct the errors, because an erroneous block could still have the same correct generator. Here, we show that location and correction of errors are always possible, and that one only needs upper bounds $T \geq t$ and $E \geq e$.

**Algorithm 4:** Majority Rule Berlekamp/Massey (MRBM)

**Input:** $(a_0, \ldots, a_{2T(2E+1)-1}) + \vec{\varepsilon}$, where $(a_i)$ is a linearly recurrent sequence (of degree $t \leq T$) of field elements, and $\vec{\varepsilon}$ is a vector of Hamming weight $e \leq E$. Denote this sequence as $(b_i)$.
**Input:** $T$, an upper bound for $t$, the degree of the monic minimal linear generator
**Input:** $E$, an upper bound for the number of errors in the above sequence
**Output:** $(a_0, \ldots, a_{2T(2E+1)-1})$, the intended sequence
**Output:** $\Lambda$, the monic minimal linear generator of the intended sequence

MRBM1 For $i = 0, \ldots, 2E$, call Algorithm 1 on $(b_{0+2Ti}, \ldots, b_{2T-1+2Ti})$; store the outputs as $\Lambda_i$, respectively.

MRBM2 Initialize $L \leftarrow [0, 1, \ldots, 2E]$ and $m \leftarrow 0$.

MRBM3 For $i \in L$, perform Steps MRBM4-MRBM7.
*We perform a majority vote on the candidates* $\Lambda_0, \ldots, \Lambda_{2E}$.

MRBM4 Initialize $L_i \leftarrow [\,]$ (the empty list).

MRBM5 For $j \in L$, perform Step MRBM6.

MRBM6 If $\Lambda_i = \Lambda_j$, then set $L_i \leftarrow L_i \cup \{j\}$ and $L \leftarrow L \setminus \{j\}$.

MRBM7 If $\mathrm{card}(L_i) > \mathrm{card}(L_m)$, then set $m \leftarrow i$.

MRBM8 Set $\Lambda \leftarrow \Lambda_m$.

MRBM9 For $i \in L_m$, perform Steps MRBM10-MRBM11.

MRBM10 Call Algorithm 2 on $(\Lambda, E, (b_0, \ldots, b_{2T(2E+1)-1}), 2Ti)$; store the output as $((c_0, \ldots, c_{2T(2E+1)-1}), e)$.

MRBM11 If $e \leq E$, then BREAK (i.e., proceed immediately to Step MRBM12).

MRBM12 Return $(c_0, \ldots, c_{2T(2E+1)-1}), \Lambda$.

### 2.5.1 Properties and Correctness

For convenience, we first assume that $T = t$. By supplying Algorithm 4 with $2t(2E+1)$ sequence entries, we guarantee that during Step MRBM1, at least $E + 1$ of the $\Lambda_i$ will be the correct ("intended") generator, $\Lambda$, by Lemma 3. If *exactly* $E + 1$ of the $\Lambda$ agree, then every block with $\Lambda$ as a generator is "clean", while every other block contains exactly one error.

**Lemma 3.** *Suppose the infinite sequence* $(a_0, a_1, \ldots)$ *has monic minimal linear generator* $\Lambda(\lambda)$ *with* $\Lambda(0) \neq 0$ *and* $\deg(\Lambda) = t$. *Then* $\Lambda$ *is also the minimal linear generator of* $(a_k, a_{k+1}, \ldots)$, *for any integer* $k \geq 0$.

*Proof.* First, consider $k = 1$. Let $\Gamma(\lambda)$ be the minimal linear generator for the sequence $(a_1, a_2, \ldots)$. This sequence is generated by $\Lambda$ as well, so we have $\Gamma \mid \Lambda$, which implies $\deg(\Gamma) \leq \deg(\Lambda)$. Suppose that $\deg(\Gamma) < \deg(\Lambda)$. We have that $\lambda\Gamma$ generates the sequence $(a_0, a_1, \ldots)$, so we must have $\Lambda \mid \lambda\Gamma$ as well, so $\deg(\Lambda) \leq \deg(\lambda\Gamma)$. But $\deg(\lambda\Gamma) \leq \deg(\Lambda)$ because $\deg(\Gamma) < \deg(\Lambda)$, thus we have $\deg(\lambda\Gamma) = \deg(\Lambda)$. Both of these polynomials are monic, so $\Lambda \mid \lambda\Gamma$ implies that $\Lambda = \lambda\Gamma$, which implies $\Lambda(0) = 0$, contradiction. Thus, $\deg(\Gamma) = \deg(\Lambda)$, so that $\Gamma \mid \Lambda$ implies $\Gamma = \Lambda$ (again because both polynomials are monic). We can inductively repeat this argument to show that $\Lambda$ is the monic minimal linear generator for any $k > 1$ as well. □

**Lemma 4.** *Suppose the sequence* $(a_0, a_1, \ldots, a_{2t-1})$ *has monic minimal linear generator* $\Lambda(\lambda)$ *with* $\Lambda(0) \neq 0$ *and* $\deg(\Lambda) = t$, *where* $\Lambda = -\gamma_0 - \gamma_1\lambda - \cdots - \gamma_{t-1}\lambda^{t-1} + \lambda^t$. *Then the sequence* $(a_{2t-1}, a_{2t-2}, \ldots, a_0)$ *has monic minimal linear generator* $\Lambda_{\text{sr}} = (1/\gamma_0)(-1+\gamma_{t-1}\lambda+\cdots+\gamma_1\lambda^{t-1}+\gamma_0\lambda^t)$, *called the (scaled) reciprocal polynomial of* $\Lambda$ (*see, e.g.,* [Imamura and Yoshida 1987, Section V]).

**Lemma 5.** *Suppose the sequence* $(a_0, a_1, \ldots, a_{2t-1})$ *has monic minimal linear generator* $\Lambda(\lambda)$ *with* $\Lambda(0) \neq 0$ *and* $\deg(\Lambda) = t$, *where* $\Lambda = -\gamma_0 - \gamma_1\lambda - \cdots - \gamma_{t-1}\lambda^{t-1} + \lambda^t$. *If we introduce at most* $t$ *errors into the sequence, with the last error no later than entry* $t$, *or equivalently the first error no earlier than entry* $t+1$, *then the erroneous sequence cannot also have* $\Lambda$ *as monic minimal linear generator.*

*Proof.* First, we consider the case of the last error no later than entry $t$. Suppose that $\Lambda$ also generated the erroneous sequence. Let the last error be located at entry $k$ and denote the erroneous sequence by

$$(b_0, b_1, \ldots, b_{k-1}, a_k, a_{k+1}, \ldots, a_t, \ldots, a_{2t-1}),$$

where $b_{k-1} \neq a_{k-1}$. Denote by $H$ and $H'$ the $t \times t$ Hankel matrices generated by the first

$2t-1$ entries of the original and erroneous sequences, respectively, and denote by $\vec{x}$ the vector $(\gamma_0, \gamma_1, \ldots, \gamma_{t-1})^T$. Then $H\vec{x} = H'\vec{x} = (a_t, a_{t+1}, \ldots, a_{2t-1})^T$, so that $(H - H')\vec{x} = \vec{0}$.

Note that row $k$ of $H - H'$ is $(a_k - b_k, 0, 0, \ldots, 0)$, so that entry $k$ of $(H - H')\vec{x}$ is $(a_{k-1} - b_{k-1})\gamma_0 \neq 0$, contradiction. Thus, the erroneous sequence cannot have $f$ as monic minimal linear generator.

For the case of the first error no earlier than entry $t+1$, let the first error be located at entry $k$ and consider the reversed sequences $(a_{2t-1}, \ldots, a_0)$ and

$$(b_{2t-1}, b_{2t-2}, \ldots, b_{k-1}, a_{k-2}, a_{k-3}, \ldots, a_{t-1}, \ldots, a_0),$$

noting by Lemma 4 that the former sequence has $\Lambda_{\text{sr}}$ as monic minimal linear generator. The original erroneous sequence has $\Lambda$ as monic minimal linear generator if and only if the reversed erroneous sequence has $\Lambda_{\text{sr}}$ as monic minimal linear generator, again by Lemma 4. We then repeat the argument above to show that the reversed erroneous sequence cannot have $\Lambda_{\text{sr}}$ as monic minimal linear generator. Therefore, the original erroneous sequence cannot have $\Lambda$ as its monic minimal linear generator. □

**Corollary 1.** *Suppose the sequence $(a_0, a_1, \ldots, a_{2t-1})$ has monic minimal linear generator $\Lambda(\lambda)$ with $\Lambda(0) \neq 0$ and $\deg(\Lambda) = t$. Then the one-error sequence $(a_0, a_1, \ldots, a_{k-2}, b_{k-1}, a_k, \ldots, a_{2t-1})$, $b_{k-1} \neq a_{k-1}$ cannot also have monic minimal linear generator $\Lambda$.*

As stated earlier, if exactly $E+1$ of the $\Lambda_i$ agree in Step MRBM1, then each of the $E$ remaining blocks of $2t$ entries must contain exactly one error. In this case, we can run Algorithm 2 during Steps MRBM10-MRBM11 in parallel, correcting each erroneous block with the nearest clean block.

If *more than* $E+1$ of the $\Lambda_i$ agree in Step MRBM1, then there may be an erroneous block of of $2t$ entries that falsely yields the correct generator; call this a "deceptive block". This block must contain at least one error in both its first $t$ entries and its last $t$ entries, by Lemma 5. In this case, we show that Algorithm 2 must return $e > E$ if it is seeded in Step MRBM10 with a deceptive block.

**Theorem 4.** *If Algorithm 2 is called in Step MRBM10 of Algorithm 4 with a deceptive block, then Algorithm 2 will return $e > E$.*

*Proof.* For convenience, assume the deceptive block is first, i.e., $(b_0, \ldots, b_{2t-1})$. Suppose the last error in the deceptive block is $b_{k-1} \neq a_{k-1}$, where $t+1 \leq k \leq 2t$. If $(b_{2t}, b_{2t+1}, \ldots, b_{k+t-1}) = (a_{2t}, a_{2t+1}, \ldots, a_{k+t-1})$, then there must be a (non-zero) discrepancy in Step SCU4 in Algorithm 2 for $i = k+t-1$ because $b_{k-1}$ is the only erroneous value in the test of the linear recurrence, and is multiplied by $\Lambda(0) \neq 0$. In this case, an "erroneous correction" will occur (i.e., Algorithm 2 will change $b_{k+t-1}$ to $b'_{k+t-1} \neq a_{k+t-1}$).

If there is at least one error in entries $b_{2t}, b_{2t+1}, \ldots, b_{k+t-1}$, then it is possible for Algorithm 2 to make no change before entry $b_{k+t-1}$, in which case there was at least one "deceptive error" in the second block. If Algorithm 2 does make a correction before entry $b_{k+t-1}$, then it is not necessarily erroneous.

Denote by $(c_{2t}, c_{2t+1}, \ldots, c_{k+t-1})$ the output of Algorithm 2 when run on entries $b_{2t}, b_{2t+1}$, $\ldots, b_{k+t-1}$ (whether or not there is an error). Let $c_s$ be the last entry such that $c_s \neq a_s$; note that this must exist because $b_{k-1} \neq a_{k-1}$, so that Algorithm 2 cannot return $(c_{2t}, c_{2t+1}, \ldots,$ $c_{k+t-1}) = (a_{2t}, a_{2t+1}, \ldots, a_{k+t-1})$. Considering entries $b_{s+1}, b_{s+2}, \ldots, b_{s+t}$, we can repeat the above arguments to show that there will be at least one correction (erroneous or not) or deceptive error during the execution of Algorithm 2.

Continuing this process, we see that after entry $k$, there will be at least one correction or deceptive error in every block of length $t$. At least two errors occurred in the first block of length $2t$, so there may be at most $E-2$ deceptive errors. At the $(E+1)$-st correction, the first block of length $2t$ must have been deceptive.

To prove why we will see the $(E+1)$-st correction, note that in Step MRBM1 of Algorithm 4, there must be a clean block no later than the $(E+1)$-st block. At this point, we relax the assumption that the deceptive block is first. The deceptive block must occur before the $(E+1)$-st block in order to be used for seeding in Step MRBM10 of Algorithm 4. To see the $(E+1)$-st correction in Algorithm 2, we need at most $2t + t(E-2) + t(E+1) = t(2E+1)$ consecutive entries (including the seeded $2t$ block), but we are guaranteed at least $2t(E+2)$ consecutive entries. Thus, Algorithm 2 will return $e > E$. □

It follows immediately from Theorem 4 that deceptive blocks will be exposed until the first clean block is found (Steps MRBM9-MRBM11 of Algorithm 4), at which point all remaining errors will be found and corrected.

Now suppose that $T > t$. If the first block of $2T$ entries that yields $\Lambda$ is clean, then the intended generator $\Lambda$ will be found in Step MRBM8 of Algorithm 4 and all errors will be detected and corrected thereafter. If the first block of $2T$ entries that yields $\Lambda$ is deceptive, then we look to the properties of the corresponding block of $2t$ entries.

If entries $1, 2, \ldots, 2t$ are clean, then there must be an error before entry $2T$, but the first non-zero discrepancy after entry $2t$, say at entry $r > 2T$, will cause

$$L_r = \max\{L_{r-1}, r - L_{r-1}\} = \max\{t, r-t\} > t$$

by [Massey 1969]. This contradicts the deceptive $2T$-block yielding $\Lambda$.

If the block of entries $1, 2, \ldots, 2t$ is itself deceptive, then we repeat the argument of Theorem 4, as we will see at least one correction or deceptive error in every block of length $t$, following entry $2t$.

If the block of entries $1, 2, \ldots, 2t$ is itself erroneous, i.e., yields generator $\Gamma \neq \Lambda$, then we must have $\deg(\Gamma) \leq t$, else $L_{2T} > t$, contradiction. If $\deg(\Gamma) = t' < t$, i.e., if $L_{2t} = t' < t$, then there must be a degree jump before entry $2T$, say at entry $r > 2t$; at this jump, we will have

$$L_r = \max\{L_{r-1}, r - L_{r-1}\} = r - L_{r-1} = r - t' > 2t - t' > t,$$

again by [Massey 1969], so in fact $L_{2T} > t$, hence the deceptive block of $2T$ entries cannot yield $\Lambda$, contradiction.

If $\deg(\Gamma) = t$, then there must be a non-zero discrepancy before entry $2T$, say at entry $r > 2t$, else the deceptive block yields $\Gamma \neq \Lambda$, contradiction. But then

$$L_r = \max\{L_{r-1}, r - L_{r-1}\} = \max\{t, r - t\} > t,$$

which again contradicts the deceptive $2T$-block yielding $\Lambda$.

Therefore, even in the case of $T > t$, Algorithm 4 will return the intended sequence and generator.

## 2.6 Implementation and Experiments of SPINO

We now adopt the strategy of Majority Rule to account for outlier errors in an early-terminated numeric version of the sparse polynomial interpolation algorithm in [Ben-Or and Tiwari 1988], which is recalled in the algorithm that follows Theorem 1. Here, we assume only upper bounds $T \geq t$ and $D \geq d = \deg(f)$; early termination follows from the algorithm detailed in [Kaltofen, Lee, and Yang 2011]: instead of executing the Berlekamp/Massey algorithm to determine $t$, we compute the 2-norm relative condition number of the leading principal submatrices of the Hankel matrix $H = [u_{i+j-2}]_{i,j=1}^{2T}$, where $u_k = f(\omega^{k+1})$. Note that by [Rump 2003], for any Hankel matrix $\bar{H}$, the reciprocal of $\|\bar{H}^{-1}\|_2$ is the distance from $\bar{H}$ to the nearest singular Hankel matrix.

In the numeric setting, we choose for $\omega$ a complex root of unity, with prime order $p > D$. The algorithm also works for interpolation of sparse Laurent polynomials, in which case we choose $p > D - D_{\text{low}}$, where $D_{\text{low}}$ is a lower bound on the low degree of $f$. We implement noise as a (randomly positive or negative) scaling factor on a random floating-point number between 1 and 5 times $\|f\|_2$, which is added to each evaluation. Outliers simply multiply a random position by 5. All computations are done in double floating point precision.

**Note 1.** Wen-shin Lee has told us that the algorithm will work even when $f$ is a polynomial and $D_{\text{low}} > 0$: the condition $p > D - D_{\text{low}}$ is still sufficient when determining a value for $p$. In particular, the algorithm will work when $D_{\text{low}} = D = d$, in which case $f(x) = cx^d$ and $p \geq 2$.

Substituting the (complex) 2-nd root of unity $\omega = -1$ allows recovery of $c$ in the exact case by Majority Rule.

As noted in Section 1 of [Kaltofen, Lee, and Yang 2011], the leading principal submatrices of $H$ are (with high probability) well-conditioned up to and including dimension $t$; the $(t+1)\times(t+1)$ leading principal submatrix will be ill-conditioned unless substantial noise (and/or an outlier) interferes. Thus, we set a threshold for ill-conditionedness, then obtain an estimate $t'$ for $t$. We then determine $\Lambda$ by obtaining a least squares solution to a well-conditioned $t' \times t'$ Hankel system. (In the exact case, the coefficients of $\Lambda$ form the solution to the non-singular Hankel system.)

After finding the roots $b_j$ of $\Lambda$, we take advantage of the fact that $\omega$ is a prime-order root of unity, by comparing the arguments of $\omega$ and $b_j$ to determine $e_j$ (modulo $p$, but there is a unique representative in the set $\{0, 1, \ldots, D\}$ because $p > D$). Finally, we determine the coefficients $c_i$ of $f$ by obtaining a least squares solution to a transposed Vandermonde system.

For Majority Rule to expose outliers, we require $2E+1$ segments of $2T+1$ evaluations, where again we assume there are $e \leq E$ outlier errors in the evaluations. As suggested in Section 3 of [Kaltofen, Lee, and Yang 2011], our implementation allows the use of several roots of unity as (initial) evaluation points in a single execution of the algorithm; in this case, we set $t'$ to the maximum of all sparsity estimates.

In contrast to the symbolic case, a majority is not guaranteed in the numeric case, because the numeric algorithm can under- and even overestimate $t$ due to unlucky randomization and noise, respectively. Similarly, wrong majorities may arise. In the cases where a majority does not exist, segments with outliers are indistinguishable from faulty noisy ones, so the algorithm returns `FAIL` if there is no majority for any root of unity. A hypothetical example of Majority Rule with four roots of unity and $E = 1$ (i.e., three segments per root) is shown in Table 1 below.

Table 1: Majority Rule example

| Root | Sparsity Estimates | | | Majority Vote |
|---|---|---|---|---|
| $\omega_1$ | 5 | 5 | 7 | 5 |
| $\omega_2$ | - | 5 | - | - |
| $\omega_3$ | 4 | 6 | 5 | - |
| $\omega_4$ | 4 | 4 | - | 4 |
| | Computed Sparsity | | | $\max\{5, 4\} = 5$ |

When substantial/unlucky noise does interfere, we still may be able to recover some accurate information. For example, given the 10-term Laurent polynomial

$$f = 48\,x^{32} + 24\,x^{25} - 53\,x^{22} + 67\,x^{-1} - 69\,x^{-7} - 5\,x^{-10} - 63\,x^{-16} - 37\,x^{-28} - 25\,x^{-35} + 16\,x^{-43}$$

with $D_{\text{low}} = -100$, $D = 100$, $T = 15$, a noise scaling factor of $10^{-7}$, ill-conditionedness threshold of $10^3$, and three random roots of unity, a particular randomization (of random floating point noise distribution and choice of roots of unity) yields an interpolant $f_9$ that has only nine terms and is a poor fit to the noisy evaluations (compared to $f$ itself). However, the nine exponents of $f_9$ are found in $f$, and $\|f_9\|_2$ has a relative error (with respect to $\|f\|_2$, ignoring the dropped monomial) of 0.036.

Another randomization yields an interpolant $f_{10}$ that has ten terms and is a slightly *better* fit to the noisy data than $f$ itself; in this case, $\|f_{10}\|_2$ has a relative error (with respect to $\|f\|_2$) on the order of $10^{-7}$. Yet another randomization yields an interpolant $f_{11}$ that has eleven terms and is also a slightly better fit to the noisy data than $f$ itself, though a worse fit than $f_{10}$. However, $\|f_{11}\|_2$ has a smaller relative error (with respect to $\|f\|_2$, ignoring the extra monomial) than $\|f_{10}\|$, which suggests that the behavior of the noise is partially encoded in the extra monomial of $f_{11}$, where the coefficient has absolute value on the order of $10^{-6}$.

## 2.7 Future Work

Our problem formulation, smoothing over incorrect values during the process of sparse reconstruction, applies to all such inverse problems, e.g., supersparse polynomial interpolation [Garg and Schost 2009] and [Kaltofen 2010, Section 2.1], computing the sparsest shift [Grigoriev and Karpinski 1993; Giesbrecht, Kaltofen, and Lee 2003] and the supersparsest shift [Giesbrecht and Roche 2010], or the more difficult exact and numeric sparse and supersparse rational function recovery [Kaltofen, Yang, and Zhi 2007; Kaltofen and Nehring 2011]. Our methods immediately apply to algorithms that are based on computing a linear recurrence, such as the supersparse interpolation algorithms in [Kaltofen 2010] and [Garg and Schost 2009]. The former needs no modification, and for the latter, one uses the majority rule algorithm for the sparse recovery with errors of the modular images $f(x) \bmod (x^p - 1)$, where $p$ is chosen sufficiently large (and random). The sparse interpolation with errors is at $\omega = (x^r \bmod (x^{p-1} + \cdots + 1))$, where $r$ is random for early termination. One may assume that some polynomial residues $f(\omega^i) \bmod (x^{p-1} + \cdots + 1)$ are faulty. The Chinese remaindering of the term exponents with several $p$ can be done by diversification [Giesbrecht and Roche 2011]. The sparsest shift algorithms in [Giesbrecht, Kaltofen, and Lee 2003] can be modified similarly. When computing symbolic polynomial values $f(y^i + z)$ with $y$ and $z$ variables one can use Reed-Solomon error correction.

Algorithms for the supersparsest shift and sparse and supersparse rational function recovery with outliers in the values are subjects of current research.

As stated in the introduction, an important variant is the hybrid symbolic-numeric algorithm for noisy sequences that also contain outlier errors. We have demonstrated that our approach can be combined with the Prony-GLL algorithms [Giesbrecht, Labahn, and Lee 2009; Kaltofen, Lee, and Yang 2011], which in turn could be incorporated into the Zippel (1990) variable-by-variable multivariate ZNIPR-algorithm for polynomials [Kaltofen, Yang, and Zhi 2007]. In addition, numeric Reed-Solomon decoding based on approximate polynomial GCD can be incorporated into ZNIPR. That and the numeric properties of the sparsest shift algorithm in the presence of noise and outliers are subjects of papers in preparation. A sparse interpolation algorithm using blocking and the matrix Berlekamp/Massey algorithm is described in [Kaltofen 2010, Section 2.2], which potentially has better error detection/noise reduction properties.

**Note 2.** The following modifications have been made from [Comer, Kaltofen, and Pernet 2012].

1. Page 7: "interpolations algorithms" has been changed to "interpolation algorithms".
2. Page 9: "is Ben-Or/Tiwari's" has been changed to "is Ben-Or's/Tiwari's".
3. Page 12: the citation in the proof of Theorem 2 has been updated.
4. Page 13: "Berlekamp-Massey" has been changed to "Berlekamp/Massey".
5. Page 14: "and the second guaranties" has been changed to "and the second guarantees".
6. Algorithms 1, 2, 3, and 4 have been reformatted.
7. The input $(c_0, \ldots, c_{n-1})$ of Algorithm 3 has been changed to $(a_0, \ldots, a_{n-1})$.
8. Page 16: "are error-free" has been changed to "is error-free".
9. Page 17: "errors in a t of " has been changed to "errors in a sequence of"
10. Page 17: "generator of degree $t$, $e+1$ 'blocks' " has been changed to "generator of degree $t$, then $e+1$ 'blocks' ".
11. The title of Section 2.5.1 has been changed.
12. Page 18: "with bands given by" has been changed to "generated by".
13. The title of Section 2.6 has been changed.
14. Note 1 has been added.
15. Page 23: "some polynomials residues" has been changed to "some polynomial residues".

Chapter Three

# Sparse Polynomial Interpolation by Variable Shift

This chapter is [Boyer, Comer, and Kaltofen 2012], except for the modifications listed in Note 3.

## 3.1 Introduction

Sparse polynomial interpolation algorithms, where the number of values required depends on the number of non-zero terms in a chosen representation base rather than on the degree of the polynomial, originate from two sources. One is Prony's 1795 algorithm for reconstructing an exponential sum [Prony III (1795)] (see also [Brezinski 2002]) and another is Blahut's exact sparse polynomial interpolation algorithm in the decoding phase of the Reed-Solomon error correcting code. Both algorithms first determine the term structure via the generator ("error locator polynomial") of the linear recurrent sequence of the values $f(\omega^i)$, $i = 0, 1, 2, \ldots$, of the sparse function $f$. Blahut's algorithm has led to a rich collection of exact sparse multivariate polynomial interpolation algorithms, among them [Ben-Or and Tiwari 1988; Kaltofen, Lakshman Y. N., and Wiley 1990; Zippel 1990; Lakshman Y. N. and Saunders 1995; Kaltofen and Lee 2003]. Prony's algorithm suffers from numerical instability unless randomization controls, with high probability and for functions of significant sparsity, the conditioning of intermediate Hankel matrices. The probabilistic spectral analysis in the GLL algorithm [Giesbrecht, Labahn, and Lee 2004, 2009] adapts the analysis of the exact early termination algorithm of [Kaltofen and Lee 2003]. The resulting numerical sparse interpolation algorithms have recently had a high impact on medical signal processing; see the web site http://smartcare.be of Wen-shin Lee and her collaborators. The GLL algorithm can be generalized to multivariate polynomial

and rational function recovery via Zippel's variable-by-variable sparse interpolation [Kaltofen, Yang, and Zhi 2007].

Already in the beginning days of symbolic computation, the choice of polynomial basis was recognized: $(x-2)^{100}+1$ is a concise representation of a polynomial with 101 terms in power basis representation. The discrete-continuous optimization problem of computing the sparsest shift of an exact univariate polynomial surprisingly has a polynomial-time solution [Grigoriev and Karpinski 1993; Grigoriev and Lakshman 2000; Giesbrecht, Kaltofen, and Lee 2003]. Our subject is the computation of an approximate interpolant that is sparsified through a shift. One can interpret our algorithm as a numerical version of the exact sparsest shift algorithms. As in least squares fitting, noise can be controlled by oversampling (cf. [Giesbrecht and Roche 2011]). The main difficulty is that the shift is unknown. Our numerical algorithm adapts Algorithm `UniSparsestShifts` ⟨`one proj, two seq`⟩ in [Giesbrecht, Kaltofen, and Lee 2003] to compute the shift: `UniSparsestShifts` ⟨`one proj, two seq`⟩ carries the shift as a symbolic variable $z$ throughout the sparse interpolation algorithm. Since the coefficients of the polynomials in the shift variable $z$ are spoiled by noise, the GCD step becomes an approximate polynomial GCD. A main question answered here is whether the arising non-linear optimization problems remain well-conditioned. Our answer is a conditional yes: an optimal approximate shift is found among the arguments of all local minima, but the number of local minima is high, preventing the application of standard approximate GCD algorithms. Instead, we perform global optimization, as a fallback, by computing all zeros of the gradient ideal. In addition, our algorithm requires high precision floating point arithmetic.

In [Comer, Kaltofen, and Pernet 2012], we have introduced outlier values to the sparse interpolation problem. There, outlier removal requires high oversampling, as the worst case of $k$-error linear complexity is $2t(2k+1)$, where $t$ is the generator degree. However, ours is only an upper bound for sparse interpolation. The situation is different for Algorithm `UniSparsestShifts` ⟨`one proj, two seq`⟩. Outliers can be removed at the construction stage of the values containing the shift variable $z$, by a numeric version of Blahut's decoding algorithm for interpolation with errors. The algorithm, numerical interpolation with outliers, is interesting in its own right. As we will show in Section 3.3, the analysis in [Giesbrecht, Labahn, and Lee 2009; Comer, Kaltofen, and Pernet 2012] does not directly apply, as randomization can only be applied with a limited choice of random evaluation points. We have successfully tested it as a subroutine of our numerical sparsest shift algorithm. Note that a few outliers per interpolation lead to a very small sparse interpolation problem for error location, which can be handled successfully by sparse interpolation with noisy values.

For the sake of comparison with this algorithm, we restrict to characteristic 0 and compare a sparse shift representation to a Taylor expansion expressed at a point that will make the representation sparse. This leads to finding a root common to many derivatives. Combined

with a weighted least squares fit for removing outliers and tolerating noise, we manage to compare favorably to the main algorithm.

In Section 3.4, we present the preliminary experimental results that our algorithms can recover sparse models even in the presence of substantial noise and outliers. See Section 3.4.3 for our conclusions.

## 3.2 Computing Sparse Shifts

We introduce in this subsection an algorithm to compute a shifted sparse interpolant in a numerical setting. The exact algorithm accepts outliers and uses early termination; we adapt it to a numerical setting, considering noisy and erroneous data. It is based on a numerical version of Blahut's decoding algorithm.

### 3.2.1 Main Algorithm with Early Termination.

The Early Termination Theorem in [Kaltofen and Lee 2003] is at the heart of computing a sparsest shift. Let

$$g(x) = \sum_{j=1}^{t} c_j x^{e_j}, \quad c_j \neq 0 \text{ for all } 1 \leq j \leq t,$$

be a $t$-sparse polynomial with coefficients in an integral domain $\mathsf{D}$. Furthermore, let

$$\alpha_i(y) = g(y^i) \in \mathsf{D}[y], \qquad \text{for } i = 1, 2, \ldots$$

be evaluations of $g$ at powers of an indeterminate $y$. Prony's/Blahut's theorem states that the sequence of the $\alpha_i$ is linearly generated by $\prod_{j=1}^{t}(\lambda - y^{e_j})$. Therefore, if one considers the Hankel matrices

$$\mathcal{H}_i(y) = \begin{bmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_i \\ \alpha_2 & \alpha_3 & \ldots & \alpha_{i+1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_i & \alpha_{i+1} & \ldots & \alpha_{2i-1} \end{bmatrix} \in \mathsf{D}[y]^{i \times i}, \quad \text{for } i = 1, 2, \ldots$$

one must have $\det(\mathcal{H}_{t+1}) = 0$. Theorem 4 in [Kaltofen and Lee 2003] simply states that $\det(\mathcal{H}_i) \neq 0$ for all $1 \leq i \leq t$. One can replace the indeterminate $y$ by a randomly sampled coefficient domain element to have $\det(\mathcal{H}_i) \neq 0$ for all $1 \leq i \leq t$ with high probability (w.h.p.).

We seek an $s$ in any extension of the field of $\mathsf{K}$ such that for a given $f(x) \in \mathsf{K}[x]$ the polynomial $f(x+s) = h(x)$ is $t$-sparse for a minimal $t$. Now consider $g(x) = f(x+z) \in \mathsf{D}[x]$ with $\mathsf{D} = \mathsf{K}[z]$. We have $\Delta_i(y, z) = \det(\mathcal{H}_i) \in (\mathsf{K}[z])[y]$; note that $\alpha_i(y, z) = g(y^i) = f(y^i + z)$.

By the above Theorem 4, the sparsest shift is an $s$ with $\Delta_{t+1}(y, s) = 0$ for the smallest $t$. Algorithm `UniSparsestShifts` ⟨`one proj, two seq`⟩ computes $s$ as

$$z + s \text{ divides (w.h.p.) } \mathrm{GCD}(\Delta_{t+1}(y_1, z), \Delta_{t+1}(y_2, z)), \text{ where } y_1, y_2 \text{ are random in } \mathsf{K};$$

note that the first $t$ with a nontrivial GCD is possibly smaller for the projection by $y = y_1$ and $y = y_2$, but with low probability.

For numeric sparse interpolation with a shift, we assume that for $f(x) \in \mathbb{C}[x]$ we can obtain

$$f(\zeta) + \textit{noise} + \textit{outlier error}, \quad \text{for any } \zeta \in \mathbb{C}.$$

Here only a fraction of the values contain an outlier error, and noise is a random perturbation of $f(\zeta)$ by a relative error of $10^{-10}$, say. Our algorithm returns a sparse interpolant $g(x)$ that at all probed values $\zeta$, save for a fraction that are removed as outliers, approximates the returned $f(\zeta) + \textit{noise}$. Note that probing $f$ at $\zeta$ twice may produce a different noise and possibly an outlier.

We now give the outline of our Algorithm `ApproxUniSparseShift` ⟨`one proj, sev seq`⟩. Note that because of the approximate nature of the shifted sparse interpolant, there is a trade-off between backward error and sparsity. Hence we call our algorithm a "sparse shift" algorithm. As in Algorithm `UniSparsestShifts` ⟨`one proj, two seq`⟩, for $L$ complex values $y = \omega^{[1]}, \omega^{[2]}, \ldots, \omega^{[L]}$ we compute $\tilde{\delta}_i^{[\ell]}(z) = \widetilde{\Delta}_i(\omega^{[\ell]}, z)$ from $\widetilde{\alpha}_i(\omega^{[\ell]}, z)$, $\ell = 1, 2, \ldots, L$. Here the tilde accent mark $\widetilde{\ }$ indicates that the values have noise in their scalars. As in [Giesbrecht, Labahn, and Lee 2009], we choose the $\omega^{[\ell]}$ to be different random roots of unity of prime order. Our algorithm consists of the four following tasks:

**Step 1:** For $\ell = 1, 2, \ldots, L$, compute the numeric complex polynomials $\widetilde{\alpha}_i(\omega^{[\ell]}, z)$ via a numeric version of the Blahut decoding algorithm; see Section 3.3. Step 1 is assumed to have removed all outliers.

**Step 2:** Compute the determinants $\tilde{\delta}_i^{[\ell]}(z)$ of numeric polynomial Hankel matrices $\widetilde{\mathcal{H}}_i^{[\ell]}(z)$ for all $\ell$, iterating Steps 3 and 4 on $i$. We perform the determinant computations with twice the floating point precision as we use for Steps 1, 3 and 4.

**Step 3:** Determine the sparsity and an approximate shift. Note that the approximate shift $\tilde{s}$ is an approximate root of the polynomials $\tilde{\delta}_i^{[1]}(z)$, $\tilde{\delta}_i^{[2]}(z)$, …, $\tilde{\delta}_i^{[L]}(z)$. Our method finds the smallest perturbation of the $\tilde{\delta}_i^{[\ell]}(z)$ that produces a common root, simultaneously for all $\ell$. If that distance is large, we assume that there is no common root and the dimension of the Hankel

matrix was too small. It might happen that an accurate shift is diagnosed too early, but then the constructed model produces a worse backward error.

The 2-norm distance to the nearest polynomial system with a common root $\tilde{s}$ is given by formula (see [Hutton, Kaltofen, and Zhi 2010] and the literature cited there):

$$\tilde{s} = \operatorname*{arginf}_{\zeta\in\mathbb{C}} \sum_{\ell=1}^{L} |\tilde{\delta}_i^{[\ell]}(\zeta)|^2 \Big/ \Big( \sum_{m=0}^{d} |\zeta^m|^2 \Big),$$

where $d = \max_\ell\{\deg(\tilde{\delta}_i^{[\ell]}(z))\}$ and in all polynomials, any term coefficients of $z^m$, where $m \in \{0, 1, \ldots, d\}$, can be deformed.

In our experiments in Section 3.4, we have only considered real shifts $\tilde{s} \in \mathbb{R}$. The optimization problem is then

$$\tilde{s}_{\text{real}} = \operatorname*{arginf}_{\xi\in\mathbb{R}} \sum_{\ell=1}^{L} \Big( (\Re\tilde{\delta}_i^{[\ell]})(\xi)^2 + (\Im\tilde{\delta}_i^{[\ell]})(\xi)^2 \Big) \Big/ \Big( \sum_{m=0}^{d} \xi^{2m} \Big), \tag{1}$$

where $\Re\tilde{\delta}_i^{[\ell]}$ and $\Im\tilde{\delta}_i^{[\ell]}$ are the real and imaginary parts of the polynomials $\delta_i^{[\ell]}$, respectively. We find $\tilde{s}_{\text{real}}$ among the real roots of the numerator of the derivative of the objective function in (1),

$$\frac{\partial \sum_l ((\Re\tilde{\delta}_i^{[\ell]})(z)^2 + (\Im\tilde{\delta}_i^{[\ell]})(z)^2)}{\partial z} \times \Big( \sum_{m=0}^{d} z^{2m} \Big) - \sum_{\ell=1}^{L} ((\Re\tilde{\delta}_i^{[\ell]})(z)^2 + (\Im\tilde{\delta}_i^{[\ell]})(z)^2) \times \Big( \sum_{m=0}^{d} (2m) z^{2m-1} \Big), \tag{2}$$

and choose the root that minimizes the objective function in (1).

We have observed that a larger number $L$ of separate $\omega^{[\ell]}$ can improve the accuracy of the optimal shift, at a cost of oversampling. We have also observed that the optimization problem (1) and (2) has numereous local optima, some near the optimal approximate shift, which prevents the use of any local approximate GCD algorithm.

**Step 4:** With the approximate sparsest shift $\tilde{s}$, complete the sparse polynomial reconstruction, as in [Giesbrecht, Labahn, and Lee 2006] and [Comer, Kaltofen, and Pernet 2012, Section 6]. One reuses the evaluations from previous steps, having removed those that were declared outliers in Step 1.

In the remainder of this section, we restrict ourselves to characteristic 0. We now describe a more naïve approach for the same problem. Some early termination can also be achieved here. Unlike the main algorithm, this one cannot recover errors in the exact setting.

### 3.2.2 Using Taylor Expansions

Let $f(x) = \sum_{i=1}^{t} c_i(x-s)^{e_i}$ be a $t$-sparse shifted polynomial of degree $d$. We can see this expression as a Taylor expansion of $f$ at $x = s$ :

$$f(x) = \sum_{i=0}^{\infty} \frac{\left(\partial^i f/\partial x^i\right)(s)}{i!} (x-s)^i .$$

A sparsest shift is then an $s$ that is a root of the maximum number of polynomials in the list $S = \left\{\left(\partial^i f/\partial x^i\right)(x) \mid i \in \{0, \ldots, d-1\}\right\}$.

**Remark 1.** It is stated in Theorem 1 in [Lakshman Y. N. and Saunders 1996] that if $t \leq (d+1)/2$ then the shift $s$ is unique and rational. Moreover, the proof gives the stronger statement: for any other shift $\hat{s}$, with a sparsity $\hat{t}$, one has $\hat{t} > d+1-t$.

This statement is not true in characteristic $p \neq 0$: for instance, consider the two shifts $-1$ and 0 in the polynomial $(x+1)^p = x^p + 1 \mod p$.

**Lemma 6.** *Let $S_{2t}$ be the list of the last $2t$ elements in $S$. The root that zeros the maximum number of polynomials in $S_{2t}$ is the sparsest shift.*

*Proof.* We prove this by contradiction. Assume a shift $s$ appears $r$ times in $S_{2t}$ and another shift $\hat{s}$ appears $\hat{r}$ times. We first notice that the number of elements in $S$ for which $\hat{s}$ is a root, and the number of elements for which it is not a root, sum to $d+1$. So we have the inequality $\hat{r} + \hat{t} \leq d+1$.

Suppose now that $\hat{r} \geq r$. The sparsity of $f$ in the $s$-shifted basis being $t$, the number of elements in $S_{2t}$ that do not have $\hat{s}$ as a root is $2t - \hat{r} \leq t$, thus $\hat{r} \geq t$. On the other hand, $\hat{t} > d+1-t$. Summing these last two inequalities yields $\hat{r} + \hat{t} > d+1$, which is impossible. □

Early termination can be achieved; indeed, under certain circumstances, one need not compute all $2t$ derivatives. For example, suppose that the degree $(d-1)$ term of $f$ in the sparsest shifted basis is missing and try $\bar{s}$, the root of $\left(\partial^{d-1} f/\partial x^{d-1}\right)(x)$, as a shift; this is the Tschirnhaus transformation (originally introduced for solving cubic equations). If the "back-shifted" polynomial $f(x+\bar{s})$ has fewer than $(d+1)/2$ terms, then by Remark 1, $\bar{s}$ is the unique sparsest shift. We can extend this technique by trying all rational roots in the list $S_\tau$ for a small $\tau$.

Now we state the naïve algorithm based on the above, then we modify it for a numeric setting. Consider first the following exact algorithm:

**Step 1:** Compute the exact interpolant using $D+1$ calls to the black box, where $D \geq d$.

**Step 2:** Try early termination on $S_\tau$, for a small $\tau$, and return if successful.

**Step 3:** Compute all remaining derivatives in $S_\theta$, for $\theta = \min(2T, D)$.

**Step 4:** The sparsest shift is the rational root $s$ that zeros the most derivatives in $S_\theta$.

**Step 5:** The "back-shifted" polynomial $f(x+s)$ gives the support for the sparse polynomial.

This algorithm can be easily translated to a numerical one, based on least squares fitting:

**Step 1:** Compute a degree-$D$ weighted least squares fit with $\mathrm{O}(D+E)$ calls to the black box.

**Step 2:** Remove outliers by comparing relative errors, then update the fit.

**Step 3:** Compute the $\theta$ derivatives in $S_\theta$ (possibly terminate early and proceed to Step 6).

**Step 4:** The approximate root $s$ that zeros most derivatives is the sparsest shift.

**Step 5:** The polynomial $f(x+s)$ gives the support for the sparse polynomial.

**Step 6:** A Newton iteration can be conducted on the result of Step 5 to increase accuracy.

### 3.2.3 Discussion on the Numeric Algorithm

Step 4 is sensitive to noise and requires more sampling from Step 1. The approximate roots are determined to be equal up to a certain tolerance (for instance $10^{-2}$). In Steps 5 and 6, the coefficients near 0 may be forced to 0 (which would accelerate convergence in Step 6). Step 6 is conducted on the function $f(m'_1, \cdots, m'_k, s') = \sum_{i=1}^{k} m'_i (x-s')^{h_i}$, with initial condition from Step 5, random samples $x_j$ and noisy evaluations $f(x_j)$; the outliers are removed by checking relative errors. If the random samples $x_j$ are not only taken from data in Step 1, then oversampling will help "de-noising" the outputs.

**Remark 2.** It is unknown to us, in the exact algorithm, how to use a number of calls to the black box in Step 1 depending only on $T$, in order to compute the derivatives. However, it is reasonable to expose the following: we are only interested in the higher-degree terms of $f$. Consider the Euclidean division $f(x) = Q(x)x^q + R(x)$; then, with high numeric precision and big random $x_i$, we can recover an approximation of $Q$ by a least squares fit on samples $f(x_i)/x_i^q \approx Q(x_i)$.

## 3.3 Numeric Interpolation with Outliers

Blahut's decoding algorithm for Reed/Solomon codes is based on sparse interpolation. Suppose one has values of

$$f(x) = c_{d-1}x^{d-1} + \cdots + c_0 \in \mathsf{K}[x], \quad \deg(f) \le d-1$$

at powers $\omega^i$: $a_i = f(\omega^i)$, $i = 0, 1, 2, \ldots, n-1$, where $n = d+2E$. Furthermore suppose for $k \le E$, where the upper bound $E$ is known, those values are spoiled by $k$ outlier errors: $b_i = a_i + a'_i$, with $a'_{e_j} \neq 0$ exactly at the indices $0 \le e_1 < e_2 < \cdots < e_k \le d+2E-1$. If $\omega$ is an $n$'th $= (d+2E)$'th primitive root of unity, then the $n \times n$ Fourier (Vandermonde) matrix $V(\omega) = [\omega^{i \cdot m}]_{0 \le i,m \le n-1}$ satisfies

$$W = V(\omega)^{-1} = \frac{1}{n} V(\omega^{-1}) \text{ where } \omega^{-1} = \omega^{n-1}, \tag{3}$$

hence

$$W\vec{b} = W\vec{a} + W\vec{a'} = \begin{bmatrix} \vec{c} \\ 0 \end{bmatrix} + \frac{1}{n} V(\omega^{-1})\vec{a'}. \tag{4}$$

The last $2E$ entries in $W\vec{b}$ allow sparse interpolation of $g(x) = \sum_{j=1}^{k} a'_{e_j} x^{e_j}$:

$$c'_l = (V(\omega^{-1})\vec{b})_l = g(\omega^{-l}) \quad \text{for } d \le l \le d+2E-1.$$

Note that all vectors are indexed $0, 1, \ldots, n-1$, e.g.,

$$\vec{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_{n-1} \end{bmatrix} \quad \text{and} \quad \vec{b} = \begin{bmatrix} b_0 \\ \vdots \\ b_{n-1} \end{bmatrix}.$$

By our convention, primed $'$ quantities contain outlier information. Thus, as in Section 3.2, the sequence $c'_d, c'_{d+1}, \ldots$ is linearly generated by $\Lambda(\lambda) = \prod_{j=1}^{k}(\lambda - \omega^{-e_j})$ (called the "error locator polynomial"), which is a squarefree polynomial by virtue of the primitivity of $\omega$. One may also compute $\Lambda$ from the reverse sequence $c'_{d+2E-1}, c'_{d+2E-2}, \ldots$, which is linearly generated by the reciprocal polynomial $\prod_{j=1}^{k}(\lambda - \omega^{e_j})$.

Not knowing $k$, the probabilistic analysis of early termination as in [Kaltofen and Lee 2003] and Section 3.2 does not directly apply, as the choice of $\omega$ is restricted to a primitive $n$'th root of unity. Furthermore, the locations $e_j$ of the outlier errors $a'_{e_j}$ may depend on the evaluation points $\omega^i$. Blahut's decoding algorithm processes all $2E$ values $c'_l$.

If one has (in addition to outliers) numerical noise $\epsilon_i$ in each evaluation, namely $\tilde{b}_i =$

$a_i + a'_i + \epsilon_i$, where $|a_{e_j} - a'_{e_j}|/|a_{e_j}| \gg 0$ (one may assume that $\epsilon_{e_j} = 0$), then

$$W\vec{\tilde{b}} = \begin{bmatrix} \vec{c} \\ 0 \end{bmatrix} + \frac{1}{n} V(\omega^{-1})\vec{a'} + \frac{1}{n} V(\omega^{-1})\vec{\epsilon},$$

so

$$\tilde{c}'_l = (V(\omega^{-1})\vec{\tilde{b}})_l = g(\omega^{-l}) + (V(\omega^{-1})\vec{\epsilon})_l = g(\omega^{-l}) + \bar{\epsilon}_l \text{ for } d \le l \le d + 2E - 1,$$

where $|\bar{\epsilon}_l| \le |\epsilon_1| + \cdots + |\epsilon_n|$. Again, there is an immediate trade-off between noise and outliers: at what magnitude does noise $\epsilon_i$ become an outlier $a'_i$? For now we assume that the relative error in noise is small, say $10^{-10}$, while the relative error in outliers is big, say $10^5$. The recovery of an approximate interpolant $\tilde{g}(x) = \sum_{j=1}^{k} \tilde{a}'_{e_j} x^{e_j}$ for the evaluations $\tilde{c}'_l$ hinges on the condition number of the $k \times k$ Hankel matrix

$$\widetilde{\mathcal{H}}'_k = \begin{bmatrix} \tilde{c}'_d & \tilde{c}'_{d+1} & \cdots & \tilde{c}'_{d+k-1} \\ \tilde{c}'_{d+1} & \tilde{c}'_{d+2} & \cdots & \tilde{c}'_{d+k} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{c}'_{d+k-1} & \tilde{c}'_{d+k} & \cdots & \tilde{c}'_{d+2k-2} \end{bmatrix}.$$

If the matrix is well-conditioned, the error locations $e_j$ can be determined from the approximate linear generator $\widetilde{\Lambda}$ as in the GLL algorithm [Giesbrecht, Labahn, and Lee 2009; Comer, Kaltofen, and Pernet 2012]. As is shown there, the conditioning is bounded by $1/|\omega^{e_u} - \omega^{e_v}|$. Large values there are prevented by randomizing $\omega$, as the term exponents $e_j$ are fixed for any evaluation. Using an $\omega^r$ instead of $\omega$ here, where $\mathrm{GCD}(r, n) = 1$, allows redistributing of the $\omega^{e_j}$, but the $e_j$ may then become different.

A special case is $k = 1$: In that case

$$\widetilde{\mathcal{H}}'_1 = [\tilde{c}'_d] = [g(\omega^{-d}) + \bar{\epsilon}_d] = [a'_{e_1}\omega^{-de_1} + \bar{\epsilon}_d],$$

which, by our assumption on a large outlier $a'_{e_1}$ and small noise, is a well-conditioned matrix. This is the case we tested in Section 3.4.

**Remark 3.** When the relative difference between the magnitudes of the outlier $a'_{e_1}$ and noise $\epsilon_0, \epsilon_1, \ldots, \epsilon_{d+2E-1}$ is not so pronounced, erroneous recovery of the exponent $e_1$ can occur: we have $(\tilde{c}'_d, \tilde{c}'_{d+1}, \ldots, \tilde{c}'_{d+2E-1}) = (\tilde{c}'_d, \tilde{c}'_{d+1})$, so the linear generator $\widetilde{\Lambda}(\lambda) = \lambda - \omega^{-e_1}$ can be approximated by computing

$$\frac{\tilde{c}'_{d+1}}{\tilde{c}'_d} = \frac{a'_{e_1}\omega^{-(d+1)e_1} + \bar{\epsilon}_{d+1}}{a'_{e_1}\omega^{-de_1} + \bar{\epsilon}_d} = \omega^{-e_1} + \frac{\bar{\epsilon}_{d+1} - \omega^{-e_1}\bar{\epsilon}_d}{a'_{e_1}\omega^{-de_1} + \bar{\epsilon}_d} = \tilde{\omega}. \tag{5}$$

For this reason, we define a bound $\varepsilon_{\text{noise}} \geq \max_i |\epsilon_i|$ and assume $n\varepsilon_{\text{noise}} < |a'_{e_1}|$ so that

$$|\tilde{\omega} - \omega^{-e_1}| = \left| \frac{\bar{\epsilon}_{d+1} - \omega^{-e_1}\bar{\epsilon}_d}{a'_{e_1}\omega^{-de_1} + \bar{\epsilon}_d} \right| \leq \frac{|\bar{\epsilon}_{d+1}| + |\bar{\epsilon}_d|}{|a'_{e_1}| - |\bar{\epsilon}_d|} \leq \frac{2n\varepsilon_{\text{noise}}}{|a'_{e_1}| - n\varepsilon_{\text{noise}}}. \tag{6}$$

By the distribution of complex roots of unity (of order $n$) on the unit circle, we have that $|\omega^{s+1} - \omega^s| = |\omega - 1| = 2\sin(\pi/n)$ for any integer $s$. Thus, $|\tilde{\omega} - \omega^{-e_1}| < \sin(\pi/n)$ will guarantee $|\tilde{\omega} - \omega^{-e_1}| < |\tilde{\omega} - \omega^s|$ for any $s \not\equiv -e_1 \pmod{n}$. Combining this fact with (6) above, we arrive at the sufficient condition

$$|\tilde{\omega} - \omega^{-e_1}| \leq \frac{2n\varepsilon_{\text{noise}}}{|a'_{e_1}| - n\varepsilon_{\text{noise}}} < \sin(\pi/n) \quad \Leftrightarrow \quad n\varepsilon_{\text{noise}} < |a'_{e_1}| \cdot \frac{\sin(\pi/n)}{2 + \sin(\pi/n)}. \tag{7}$$

Table 2 shows some experiments of decreasing $\Theta^{[\text{abs}]}_{\text{outlier}}$ for a fixed $\varepsilon^{[\text{abs}]}_{\text{noise}}$. Throughout the experiment, we have $f(x) = 87\,x^{11} - 56\,x^{10} - 62\,x^8 + 97\,x^7 - 73\,x^4 - 4\,x^3 - 83\,x - 10$ and $d - 1 = 11$, evaluating at powers of the order $n = d + 2E = 14$ complex root of unity $\omega = \exp(2\pi\boldsymbol{i}/14)$. We add to each evaluation noise, which is implemented as a complex number with polar modulus uniformly chosen at random in the range $[0, \varepsilon^{[\text{abs}]}_{\text{noise}}]$ and polar argument uniformly chosen at random in the range $[0, 2\pi]$. An absolute outlier value is chosen the same way, but the modulus is in the range $[\Theta^{[\text{abs}]}_{\text{outlier}}, 2\Theta^{[\text{abs}]}_{\text{outlier}}]$; the exponent $e_1$ is also chosen uniformly at random from $\{0, 1, \ldots, d + 2E - 1 = 13\}$. Each row of the table corresponds to 1000 realizations of the random variable that generates noise and outliers, re-seeding the random number generator with each run. All computations were performed with 15 floating point digits of precision. In the table, $C_n = \sin(\pi/n)/\big(2 + \sin(\pi/n)\big)$.

The column "% Circle" shows the percentage of runs where $|\tilde{\omega} - \omega^{-e_1}| < \sin(\pi/n)$; "% Sector" shows the percentage of runs where $|\tilde{\omega} - \omega^{-e_1}| \geq \sin(\pi/n)$, but $|\tilde{\omega} - \omega^{-e_1}| < |\tilde{\omega} - \omega^s|$ for any $s \not\equiv -e_1 \pmod{n}$; "% Wrong" shows the percentage of the remainder of the runs. When the ratio $\varepsilon^{[\text{abs}]}_{\text{noise}}/\Theta^{[\text{abs}]}_{\text{outlier}}$ is either sufficiently large or small, one can see from (5) that the value of $\tilde{\omega}$ is determined mainly by the value of either $a'_{e_1}$ or $\bar{\epsilon}_{d+1}/\bar{\epsilon}_d$, respectively; this corresponds with the first and last rows of each section of Table 2, where $\bar{\epsilon}_{d+1}/\bar{\epsilon}_d$ is far from $\omega^{-e_1}$ in general.

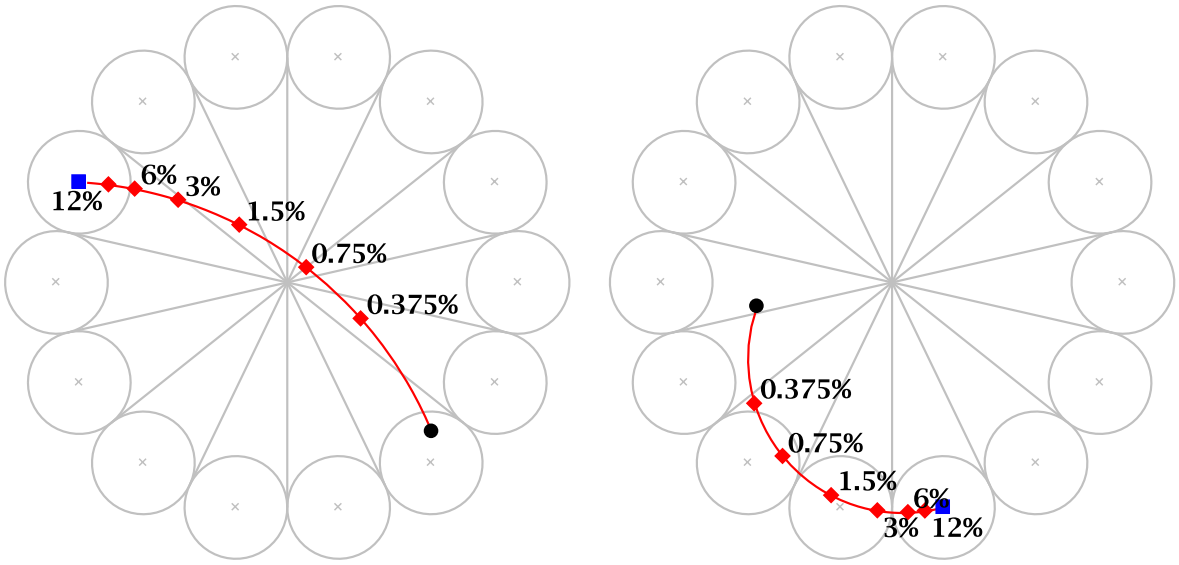
However, between the extreme values of $\tilde{\omega}$, more interesting behavior can occur. Figure 1 shows two individual algorithm runs of the table rows for $\varepsilon^{[\text{abs}]}_{\text{noise}} = 1$. Each power of $\omega$ is represented by a "×"; the sphere of radius $\sin(\pi/n)$ is drawn around each power of $\omega$, as well as the corresponding (interior) sector; the solid square denotes $\omega^{-e_1}$, while the solid circle denotes $\bar{\epsilon}_{d+1}/\bar{\epsilon}_d$; a complex outlier $a'_{e_1} = \xi$ is fixed, then the function $\tilde{\omega}(t\xi)$ (for $t \in [2^{-7}, 2^7]$) is plotted as a curve, with several points whose label is the relative error of $t\xi$ compared to $\omega^{-e_1}$. In Figure 1a, outliers of relative error less than 6% cause $\tilde{\omega}$ to approach 0, so that it becomes infeasible to compute a reliable guess for $e_1$; here, noise constitutes approximately a 0.38% relative error.

Table 2: Experiments of varying outlier error in the presence of noise

| $\varepsilon_{\text{noise}}^{[\text{abs}]}$ | $\Theta_{\text{outlier}}^{[\text{abs}]}$ | $\frac{n\varepsilon_{\text{noise}}^{[abs]}}{C_n\Theta_{\text{outlier}}^{[\text{abs}]}}$ | $\frac{n\varepsilon_{\text{noise}}^{[abs]}}{\Theta_{\text{outlier}}^{[\text{abs}]}}$ | % Circle | % Sector | % Wrong |
|---|---|---|---|---|---|---|
| 2.5e-01 | 8.0e+00 | 4.4e+00 | 4.4e-01 | 99.7 | 0.3 | 0.0 |
| 2.5e-01 | 4.0e+00 | 8.7e+00 | 8.8e-01 | 92.4 | 5.0 | 2.6 |
| 2.5e-01 | 2.0e+00 | 1.7e+01 | 1.8e+00 | 47.5 | 27.0 | 25.5 |
| 2.5e-01 | 1.0e+00 | 3.5e+01 | 3.5e+00 | 17.0 | 26.1 | 56.9 |
| 2.5e-01 | 5.0e-01 | 7.0e+01 | 7.0e+00 | 4.2 | 15.0 | 80.8 |
| 2.5e-01 | 2.5e-01 | 1.4e+02 | 1.4e+01 | 1.8 | 9.2 | 89.0 |
| 5.0e-01 | 1.6e+01 | 4.4e+00 | 4.4e-01 | 99.7 | 0.3 | 0.0 |
| 5.0e-01 | 8.0e+00 | 8.7e+00 | 8.8e-01 | 92.4 | 5.0 | 2.6 |
| 5.0e-01 | 4.0e+00 | 1.7e+01 | 1.8e+00 | 47.5 | 27.0 | 25.5 |
| 5.0e-01 | 2.0e+00 | 3.5e+01 | 3.5e+00 | 17.0 | 26.1 | 56.9 |
| 5.0e-01 | 1.0e+00 | 7.0e+01 | 7.0e+00 | 4.2 | 15.0 | 80.8 |
| 5.0e-01 | 5.0e-01 | 1.4e+02 | 1.4e+01 | 1.8 | 9.2 | 89.0 |
| 1.0e+00 | 3.2e+01 | 4.4e+00 | 4.4e-01 | 99.7 | 0.3 | 0.0 |
| 1.0e+00 | 1.6e+01 | 8.7e+00 | 8.8e-01 | 92.4 | 5.0 | 2.6 |
| 1.0e+00 | 8.0e+00 | 1.7e+01 | 1.8e+00 | 47.5 | 27.0 | 25.5 |
| 1.0e+00 | 4.0e+00 | 3.5e+01 | 3.5e+00 | 17.0 | 26.1 | 56.9 |
| 1.0e+00 | 2.0e+00 | 7.0e+01 | 7.0e+00 | 4.2 | 15.0 | 80.8 |
| 1.0e+00 | 1.0e+00 | 1.4e+02 | 1.4e+01 | 1.8 | 9.2 | 89.0 |

By contrast, Figure 1b shows an example where nearly any outlier relative error greater than 0.375% would result in $|\tilde{\omega} - \omega^s| < \sin(\pi/n)$ for one of three values of $s \pmod n$, so that the "nearest $\omega^s$ neighbor" criterion is no longer reliable; here, noise constitutes approximately a 0.40% relative error.

Decoding the interpolant $W\vec{b}$ can also be done via the extended Euclidean algorithm for any $\omega$ with $\omega^{e_u} \neq \omega^{e_v}$: the Berlekamp-Welch algorithm; see [Khonji, Pernet, Roch, Roche, and Stalinsky 2010]. We will study the numerical properties of variants based on approximate GCD techniques in follow-up work.




(a) (b)

Figure 1: Examples of varying outlier relative error (labeled as percentages). Noise relative error is approximately 0.40%.

## 3.4 Implementation and Experiments of `NumericSparsestShift`

### 3.4.1 Illustrative Examples for the Main Algorithm

We reversely engineer a noisy black box for

$$f_1(x) = 2\,(x-7)^3 + 3\,(x-7)^6 - 7\,(x-7)^{10} = \\ -7x^{10} + 490x^9 - 15435x^8 + 288120x^7 - 3529467x^6 + 29647422x^5 - 172941825x^4 \\ + 691755542x^3 - 1815804312x^2 + 2824450258x - 1976974482. \quad (8)$$

Our algorithm computes with a precision of 100 floating-point digits (except in Step 2, where the precision is doubled). To each evaluation, we add random noise causing a relative error of $1 \times 10^{-28}$. For each interpolation problem of a given degree $i$ in Step 1, we add one outlier error of relative error 5. We use $L = 3$ different 17'th roots of unity $\omega^{[\ell]}$.

Step 1 correctly locates each of the outliers in its $21 = 3 \times 7$ interpolation calls. The relative 2-norm differences

$$\|\delta_4^{[\ell]}(z) - \tilde{\delta}_4^{[\ell]}(z)\|_2 / \|\delta_4^{[\ell]}(z)\|_2$$

of the coefficient vectors of the $4 \times 4$ matrix determinants after Step 2 are $2.126 \times 10^{-27}$, $2.681 \times 10^{-27}$, $6.596 \times 10^{-27}$ for $\ell = 1, 2, 3$ all within the added noise (after outlier removal).

The polynomial (2) in Step 3 has 4 real roots, and its minimum objective function value (1) is at $\tilde{s} = 6.9989162726$ with an objective function value of $2.028 \times 10^{-57}$, as opposed to the exact case (without noise) of $2.280 \times 10^{-71}$ at $s = 7$ (there is 1 more root with much larger objective value).

The sparse model recovered from $\tilde{s}$ produces the correct term exponents $e_1 = 3$, $e_2 = 6$, and $e_3 = 10$, and the least squares fit at the non-erroneous $252 = 273 - 21$ prior black box evaluations produces the approximate model for (8),

$$2.009369(x - \tilde{s})^3 + 2.998102(x - \tilde{s})^6 - 6.997705(x - \tilde{s})^{10}, \quad \tilde{s} = 6.9989162726.$$

The relative 2-norm backward error of the model (with respect to the noisy black box evaluations) is $1.596557 \times 10^{-3}$, while that of $f_1$ itself is $5.774667 \times 10^{-28}$. A similar model can be produced with 90 floating-point digits, but not with 80.

When doubling the noise to relative error $2 \times 10^{-28}$ with 100 floating-point digits, the computed model is

$$2.036489(x - \tilde{s})^3 + 2.992182(x - \tilde{s})^6 - 6.991277(x - \tilde{s})^{10}, \quad \tilde{s} = 6.9957389337,$$

with relative 2-norm backward error $6.222096 \times 10^{-3}$, compared to $1.154933 \times 10^{-27}$ for $f_1$. Even with relative noise of $4 \times 10^{-28}$, the computed model is

$$2.125876(x - \tilde{s})^3 + 2.967832(x - \tilde{s})^6 - 6.972579(x - \tilde{s})^{10}, \quad \tilde{s} = 6.9848087178,$$

with relative 2-norm backward error $2.151040 \times 10^{-2}$, compared to $2.309866 \times 10^{-27}$ for $f_1$. At relative noise of $8 \times 10^{-28}$, the algorithm fails to determine a sparse approximant, even when increasing the number of sequences to $L = 10$.

Such failure is deceptive. The lack of sparsity, namely 3 of a maximum of 11 terms, allows for denser models that provide fits. In addition, a shift of 7 produces large evaluations at roots of unity, as indicated in the power basis representation of (8). Making the shift smaller and the degree larger, and considering the polynomial

$$f_2(x) = 2\,(x - 1.55371)^3 + 3\,(x - 1.55371)^6 - 7\,(x - 1.55371)^{15},$$

we can recover from $L = 3$ sequences, with a relative noise in the evaluations of $1 \times 10^{-14}$, and

again 1 outlier per interpolation, the approximate model

$$1.999718(x-\tilde{s})^3 + 2.998609(x-\tilde{s})^6 - 7.000117(x-\tilde{s})^{15}, \quad \tilde{s} = 1.5537114392,$$

with relative 2-norm backward error $8.000329 \times 10^{-1}$, compared to $8 \times 10^{-1}$ (to 7 digits) for $f_1$ itself.

For this particular example, we see a case of the effect mentioned in [Comer, Kaltofen, and Pernet 2012], where the sparse model can fit the noisy evaluations nearly as well (and sometimes better) than the exact black box.

Increasing the noise still, another model with $\tilde{s} = 1.5537013193$ can be recovered with relative noise of $2 \times 10^{-13}$, where now the model and $f_1$ relative 2-norm backward errors are $5.180450 \times 10^{-2}$ and $1.108027 \times 10^{-11}$, respectively. In this case, a different choice of $L = 3$ different 17'th roots of unity was needed in order to compute a sparse model. Both computations used 357 black box evaluations.

### 3.4.2 Comparison with the Naïve Algorithm

For the examples given above, the naïve algorithm recovers the sparsest representation with noise such as $1 \times 10^{-10}$ and precision 20 floating point digits. The precision obtained is close to the level of noise ($1 \times 10^{-8}$ relative error for the shift and $2 \times 10^{-10}$ maximum relative error on the coefficients in the shifted basis). The number of calls to the black box is below 170.

For a more demanding example such as a degree 55 polynomial with sparsity 8 and a shift between 1 and 2, a level of relative noise $1 \times 10^{-28}$ is tolerable with precision 200 digits (as in an example above). However, the number of calls was above 600 to get a relative error less than $1 \times 10^{-20}$ on shift and coefficients. Due to the numerical optimization in Step 3, this is unattainable with the main algorithm, for the moment. The Tschirnhaus early termination was not used yet.

Besides, with more calls to the black box during the Newton iterations, we can further increase the precision on the shift and coefficients, this may however be considered as de-noising.

We can also run experiments on a black box of the type $P + Q_\epsilon$ where $P$ is a polynomial with a sparse shift representation and $Q_\epsilon$ is a dense polynomial of same degree with coefficients bounded by $\epsilon$ – this may be viewed as perturbation on the coefficients. The algorithms described perform well, however they do not remove outliers if they are introduced as an erroneous term.

### 3.4.3 Discussion

Our preliminary experiments lead to the following conclusions: Our correction of 1 outlier per interpolation with Blahut's numerical decoding is highly numerically reliable. The optimization

problem in Step 3 requires substantial precision for its real root finding, and is numerically sensitive when the shift is large and there is noise in the evaluations. Our main algorithm works well without noise and outliers, or in high precision with noise when the shift is small and the sparsity is high. We plan to work on a more thorough experimental analysis, including the case of two or more outliers per interpolation. The naïve algorithm gives motivation and potential for improvements to the main one. On the other hand, the number of calls to the black box in the former could be reduced.

**Note 3.** The following modifications have been made from [Boyer, Comer, and Kaltofen 2012].

1. Page 31: the footnote has been removed.
2. Page 31: "sparsity of $f$ n the" has been changed to "sparsity of $f$ in the".
3. Page 31: "by Section 2.2" has been changed to "by Remark 1".
4. Page 32: "possibly early terminate" has been changed to "possibly terminate early"
5. Remark 3 has been added.
6. The title of Section 3.4 has been changed.

Chapter Four

# Counting Singular Hankel Matrices over a Finite Field

This chapter is [Comer and Kaltofen 2012], except for the changes listed in Note 4.

## 4.1 Introduction

Throughout the discussion, entries will be taken from the field $\mathbb{F}_q$ of $q$ elements, and we will identify a square Hankel matrix

$$\begin{bmatrix} a_0 & a_1 & \dots & a_{n-2} & a_{n-1} \\ a_1 & a_2 & \dots & a_{n-1} & a_n \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ a_{n-2} & a_{n-1} & \dots & a_{2n-4} & a_{2n-3} \\ a_{n-1} & a_n & \dots & a_{2n-3} & a_{2n-2} \end{bmatrix} = [a_{i+j-2}]_{i,j=1}^{n}$$

with the list $[a_0, a_1, \dots, a_{2n-2}]$. A Toeplitz matrix is the mirror image $[a_{n+i-j-1}]_{i,j=1}^{n}$.

Our investigation was motivated by the question of Ramamohan Paturi who asked in October 2009 how many Toeplitz matrices over $\mathbb{F}_q$ with zeros on the main diagonal were non-singular. Paturi needed the estimate for the complexity of circuit satisfiability for lower bounds [Paturi and Pudlák 2010].

Daykin (1960) proved theorems regarding the number of Hankel matrices over a finite field with a specified rank or determinant. Kaltofen and Lobo (1996) established some of Daykin's counts using the extended Euclidean algorithm form of the Berlekamp/Massey algorithm for

polynomials [Sugiyama, Kasahara, Hirasawa, and Namekawa 1975]. Additionally, they gave the number of square Toeplitz matrices with *generic rank profile*. Generic rank profile means that for a matrix $A$ of rank $r$, the first $r$ leading principal submatrices $A_1, \ldots, A_r$ are non-singular. We prove analogous results here, albeit with a different approach, for the Hankel case. The counts for Toeplitz and Hankel matrices of generic rank profile are not the same.

The determination of singularity of a Hankel matrix has a natural connection with running the Berlekamp/Massey algorithm on the list $[a_0, a_1, \ldots, a_{2n-2}]$, and for this reason we count how many Hankel matrices have zeros along the anti-diagonal in order to answer the question regarding Toeplitz matrices. Kaltofen and Lee (2003) have observed that the Berlekamp/Massey algorithm [Massey 1969, cf. Theorem 1] detects the non-singular leading principal submatrices of a Hankel matrix from those non-zero discrepancies that increase the linear generator degrees, and that the corresponding sequence elements determine the singularity of the corresponding leading principal submatrices.

We will use this property to partition the space of Hankel matrices into unique correspondences of one singular Hankel matrix to $q-1$ non-singular Hankel matrices. This process generalizes when particular entries of the list $[a_0, \ldots, a_{2n-2}]$ are fixed to arbitrary values (such as the case of zeros along the anti-diagonal).

We then investigate the properties of block-Hankel matrices. We have no explicit formula for how many block-Hankel matrices are singular (with or without certain blocks fixed), but we present some brute-force counts that we have computed with Maple. Last, we follow in the theme of [Kaltofen and Lobo 1996] by counting block-Hankel matrices with *block generic rank profile*. A block matrix $A$ (of square submatrices of dimension $m$) of rank $mr$ has block generic rank profile if for $k = 1, 2, \ldots, r$, we have $\operatorname{rank}(A_k) = mk$, where $A_k$ is the $k \times k$ block leading principal submatrix of $A$.

The counts for unblocked rank $r$ Hankel matrices are given in [Garcìa-Armas, Ghorpade, and Ram 2011].

## 4.2 Connection with the Berlekamp/Massey Algorithm

Given a list of $n$ field elements, $[a_0, \ldots, a_{n-1}]$, the Berlekamp/Massey algorithm will produce for each $r \in \{1, 2, \ldots, n\}$ a monic polynomial

$$\Lambda_r = c_0 + c_1 z + \cdots + c_{L_r-1} z^{L_r-1} + z^{L_r}$$

of minimal degree $L_r \leq r-1$ such that

$$-c_0 a_i - c_1 a_{i+1} - \cdots - c_{L_r-1} a_{i+L_r-1} = a_{i+L_r}, \quad i = 0, 1, \ldots, r - L_r - 1. \tag{9}$$

Such a polynomial is called a *(minimal) generating polynomial*, and $L_r$ is called the *minimal length* required to generate the first $r$ elements of the sequence. Theorem 2 in [Massey 1969] states that if a polynomial (of minimal degree $L_n$) generates the list $[a_0, \ldots, a_{n-1}]$ but fails to generate $a_n$ (i.e., equation (9) does not hold for $i = n - L_r$), then the minimal generating polynomial of $[a_0, \ldots, a_n]$ will be of degree $L_{n+1} = n - L_n + 1$.

We visualize the generating polynomials in the following way: for each $r = 1, 2, \ldots, n$, define

$$H_r = \begin{bmatrix} a_0 & a_1 & \ldots & a_{L_r-2} & a_{L_r-1} \\ a_1 & a_2 & \ldots & a_{L_r-1} & a_{L_r} \\ \vdots & \vdots & \iddots & \vdots & \vdots \\ a_{r-L_r-1} & a_{r-L_r} & \ldots & a_{r-1} & a_{r-2} \\ a_{r-L_r} & a_{r-L_r+1} & \ldots & a_{r-2} & a_{r-1} \end{bmatrix}, \quad \lambda_r = \begin{bmatrix} -c_0 \\ -c_1 \\ \vdots \\ -c_{L_r-2} \\ -c_{L_r-1} \end{bmatrix}, \quad h_r = \begin{bmatrix} a_{L_r} \\ a_{L_r+1} \\ \vdots \\ a_{r-1} \\ a_r \end{bmatrix},$$

so that by definition of $\Lambda_r$ above, we have $H_r\lambda_r = h_r$ (for $r = 1, 2, \ldots, n-1$) except possibly in the last entry (which corresponds to equation (9) for $i = r - L_r$). The last entry of $H_r\lambda_r$ will not be $a_r$ if and only if $\Lambda_r$ generates $[a_0, \ldots, a_{r-1}]$ but not $a_r$.

Suppose that $\Lambda_{n'}$ of degree $L_{n'} \leq n'$ generates $[a_0, \ldots, a_{n-1}]$ but not $a_n$. Then the vector

$$\vec{\Lambda}_{n'} = [-\lambda_{n'} \quad 1]^T = [c_{L_{n'}} \quad c_{L_{n'}-1} \quad \ldots \quad c_1 \quad 1]^T$$

is a null-space vector of the leading $(n - L_{n'}) \times (L_{n'} + 1)$ submatrix of the matrix depicted in Figure 2; the first $L_{n'}$ columns of this submatrix form the first $n - L_{n'}$ rows of $H_{n-1}$.

As noted above, Theorem 2 in [Massey 1969] implies that the minimal length required to generate $[a_0, \ldots, a_n]$ is $L_{n+1} = n - L_{n'} + 1$. Thus, $H_{n+1}$ will have $n - L_{n'} + 1$ columns, and in fact, the entire $(n - L_{n'} + 1) \times (n - L_{n'} + 1)$ leading principal submatrix in Figure 2 will be non-singular. For completeness, we now give those details in the proof of Lemma 2 in [Kaltofen and Yuhasz 2013a], which justify that claim.

If we post-multiply the $(n - L_{n'} + 1) \times (n - L_{n'} + 1)$ leading principal submatrix

$$\begin{bmatrix} a_0 & \ldots & a_{L_{n'}-1} & \ldots & a_{n-L_{n'}} \\ \vdots & \iddots & \vdots & \iddots & \vdots \\ a_{L_{n'}-1} & \ldots & a_{2(L_{n'}-1)} & \ldots & a_{n-1} \\ \vdots & \iddots & \vdots & \iddots & \vdots \\ a_{n-L_{n'}} & \ldots & a_{n-1} & \ldots & a_{2(n-L_{n'})} \end{bmatrix}$$

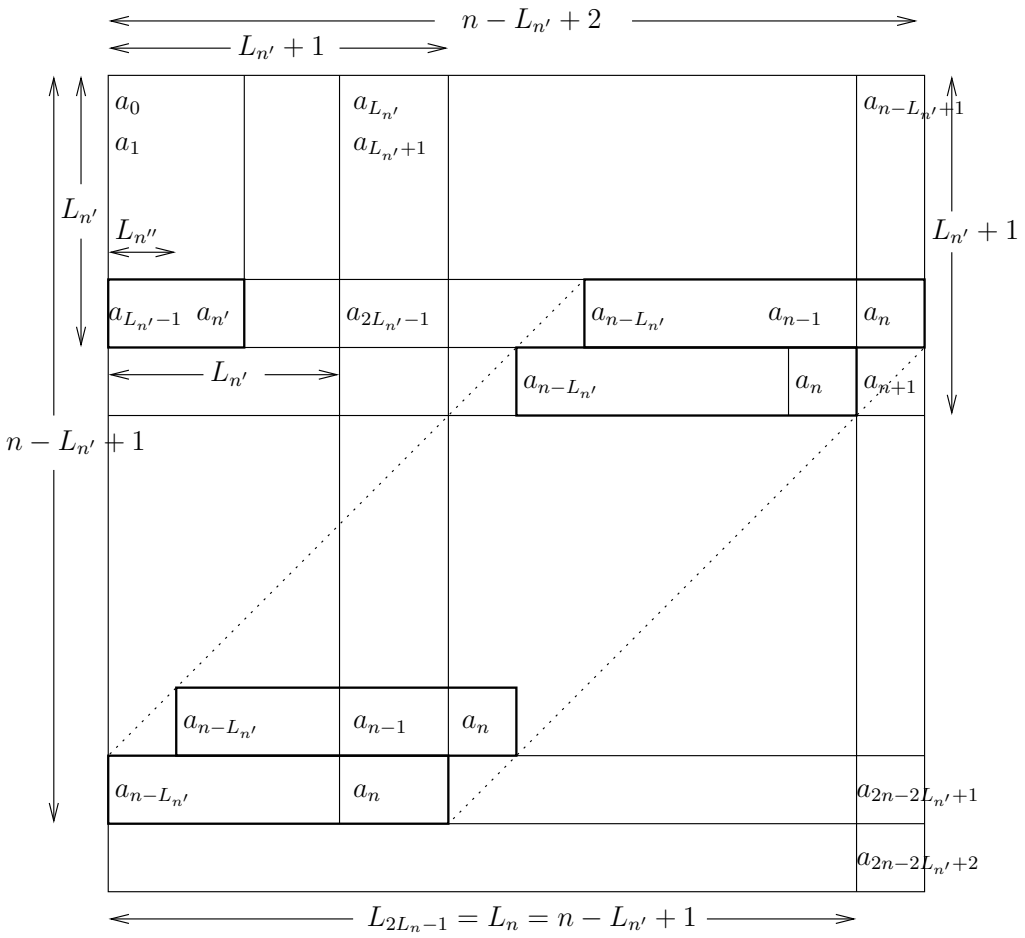



Figure 2: Berlekamp/Massey algorithm, as seen in [Kaltofen and Lee 2003]

by

$$\left[\begin{array}{c|ccccc} & c_{L_{n'}} & 0 & \cdots & 0 & 0 \\ & c_{L_{n'}-1} & c_{L_{n'}} & \ddots & \vdots & \vdots \\ I_{L_{n'}} & \vdots & c_{L_{n'}-1} & \ddots & 0 & \vdots \\ & c_1 & \vdots & \ddots & c_{L_{n'}} & 0 \\ \hline & 1 & c_1 & \ddots & c_{L_{n'}-1} & c_{L_{n'}} \\ 0^{m_1\times m_2} & 0 & 1 & \ddots & \vdots & c_{L_{n'}-1} \\ & \vdots & 0 & \ddots & c_1 & \vdots \\ & \vdots & \vdots & \ddots & 1 & c_1 \\ & 0 & 0 & \cdots & 0 & 1 \end{array}\right], \quad \begin{array}{l} m_1 = n - 2L_{n'} + 1 \\ m_2 = L_{n'} \end{array}, \tag{10}$$

the result is

$$\left[\begin{array}{c|ccccc} \bar{H} & 0^{m_1\times m_2} & & & & \\ \hline & 0 & \cdots & \cdots & 0 & \alpha \\ & \vdots & \iddots & \iddots & \alpha & * \\ * & \vdots & 0 & \iddots & * & \vdots \\ & 0 & \alpha & \iddots & \vdots & * \\ & \alpha & * & \cdots & * & * \end{array}\right], \quad \begin{array}{l} m_1 = L_{n'} \\ m_2 = n - 2L_{n'} + 1 \end{array}, \quad \alpha \neq 0, \tag{11}$$

where $\bar{H}$ is the $L_{n'} \times L_{n'}$ leading principal submatrix of $H$ and $\alpha \neq 0$ is the last entry of $H_n\lambda_n$. The $(n - L_{n'} + 1) \times (n - L_{n'} + 1)$ leading principal submatrix of $H$ will be non-singular if the above matrix is non-singular, which will happen if $\bar{H}$ is non-singular. This can be shown as a consequence of Lemmas 7 and 8 below.

**Lemma 7.** *Given a list* $[0, 0, \ldots, 0, \alpha]$ *with* $\alpha \neq 0$ *as the* $(k+1)$*-st entry, the first* $H_r$ *with* $L_r > 0$ *will be* $H_{k+1} = [0 \quad 0 \quad \cdots \quad 0 \quad \alpha]$.

*Proof.* The Berlekamp/Massey algorithm initializes $\Lambda_0 = 1$, which generates the zero sequence of any length. Thus we have $\Lambda_1 = \cdots = \Lambda_k = 1$ and $L_0 = \cdots = L_k = 0$, where $\Lambda_k$ generates $a_0, a_1, \ldots, a_{k-1}$ (the zero sequence) but not $a_k = \alpha$. We then have

$$L_{k+1} = \max\{L_k, k - L_k + 1\} = \max\{0, k - 0 + 1\} = k + 1,$$

and $\Lambda_{k+1} = z^{k+1}$.

So for $r \leq k$, $H_r$ is of size $(r - L_r + 1) \times L_r = (r + 1) \times 0$ (i.e., a matrix of $r + 1$ empty

rows), and $H_{k+1}$ is of size

$$(k+1-L_{k+1}+1)\times L_{k+1} = (k+1-(k+1)+1)\times(k+1) = 1\times(k+1),$$

and thus $H_{k+1}$ has the proposed form. □

With $H_{k+1}$ as in Lemma 7, the $H_r$ matrices will keep augmenting rows until a length change (in the minimal generator) occurs. We can show that a length change will not occur until the $H_r$ matrices "fill-up" the rest of the rows of the $L_{k+1}\times L_{k+1}$ leading principal submatrix of $H$.

**Lemma 8.** *Suppose $H_p$ is a leading submatrix of rows of the $L_p\times L_p$ leading principal submatrix of $H$, and suppose $L_p = L_{p+1} = \cdots = L_{p+q} < L_{p+q+1}$. Then $H_{p+q}$ will have at least $L_p+1$ rows, and $H_{p+q+1}$ will have exactly $L_p+1$ rows.*

*Proof.* We see that $H_p, H_{p+1}, \ldots, H_{p+q}$ all have the same number of columns, so by the definition of $H_r$ for arbitrary $r$, these matrices will be formed by augmenting by one row at a time. Also by the definition of $H_r$, we have

$$H_{p+q} = \begin{bmatrix} a_0 & \cdots & a_{L_{p+q}-1} \\ \vdots & \iddots & \vdots \\ a_{p+q-L_{p+q}} & \cdots & a_{p+q-1} \end{bmatrix} = \begin{bmatrix} a_0 & \cdots & a_{L_p-1} \\ \vdots & \iddots & \vdots \\ a_{p+q-L_p} & \cdots & a_{p+q-1} \end{bmatrix}.$$

Because of the length change between $L_{p+q}$ and $L_{p+q+1}$, we will have

$$\begin{aligned} L_{p+q+1} &= \max\{L_{p+q}, (p+q)-L_{p+q}+1\} \\ &= (p+q)-L_{p+q}+1 \\ &= p+q-L_p+1 \\ &> L_{p+q} = L_p, \end{aligned}$$

so that $H_{p+q}$ will have more than $L_p$ rows. Also, we will have

$$H_{p+q+1} = \begin{bmatrix} a_0 & \cdots & a_{L_{p+q+1}-1} \\ \vdots & \iddots & \vdots \\ a_{p+q+1-L_{p+q+1}} & \cdots & a_{p+q} \end{bmatrix} = \begin{bmatrix} a_0 & \cdots & a_{p+q-L_p} \\ \vdots & \iddots & \vdots \\ a_{L_p} & \cdots & a_{p+q} \end{bmatrix},$$

so that $H_{p+q+1}$ will have $L_p+1$ rows. □

**Corollary 2.** *If $L_{r+1} > L_r$, then $H_{r+1}$ is a leading submatrix of rows of the $L_{r+1}\times L_{r+1}$ leading principal submatrix of $H$.*

From Lemma 7, it is clear that the first $L_p \times L_p$ leading principal submatrix of $H$ is non-singular (for $L_p > 0$). From Lemma 8, we see that when the next length change occurs, say $L_{p+q+1} > L_p$, $H_{p+q}$ will have at least $L_p + 1$ rows, which corresponds to $\Lambda_p$ generating $[a_0, \ldots, a_{p+q-1}]$ but not $a_{p+q}$. We can post-multiply the $L_{p+q} \times L_{p+q}$ leading principal submatrix by an appropriate matrix like (10) to obtain a matrix product whose result is analogous to (11). Because $\bar{H}$ here is the $L_p \times L_p$ non-singular leading principal submatrix of $H$, we have that the $L_{p+q} \times L_{p+q}$ leading principal submatrix of $H$ is non-singular.

Using induction on Lemma 8, we conclude that for all $n$, the $L_n \times L_n$ leading principal submatrix is non-singular (when $L_n > 0$).

**Lemma 9.** *Given an $n \times n$ Hankel matrix $H$, let $r$ be maximal such that $r - L_r + 1 \leq n$ and $L_r \leq n$ (i.e., $H_r$ is a submatrix of $H$ but $H_{r+1}$ is not). Then $H_{r+1}$ will have $n+1$ rows and at most $n$ columns.*

*Proof.* We make the convention that if $L_s = 0$ for some $s$, then $H_s$ has $s - L_s + 1 = s + 1$ empty rows, and is (trivially) a submatrix of $H$.

If $L_{r+1} = L_r \leq n$, then we will have

$$H_{r+1} = \left[\begin{array}{cccc} a_0 & a_1 & \ldots & a_{L_{r+1}-1} \\ a_1 & a_2 & \ldots & a_{L_{r+1}} \\ \vdots & \iddots & \iddots & \vdots \\ a_{r-L_{r+1}} & \ldots & \ldots & a_{r-1} \\ \hline a_{r+1-L_{r+1}} & \ldots & \ldots & a_r \end{array}\right] = \left[\begin{array}{cccc} a_0 & a_1 & \ldots & a_{L_r-1} \\ a_1 & a_2 & \ldots & a_{L_r} \\ \vdots & \iddots & \iddots & \vdots \\ a_{r-L_r} & \ldots & \ldots & a_{r-1} \\ \hline a_{r+1-L_r} & \ldots & \ldots & a_r \end{array}\right]$$
$$= \left[\begin{array}{c} H_r \\ \hline a_{r+1-L_r} \quad a_{r+1-(L_r-1)} \quad \ldots \quad a_r \end{array}\right],$$

so that $H_{r+1}$ has $L_{r+1} = L_r \leq n$ columns. By the maximality of $r$, we must have

$$n < (r+1) - L_{r+1} + 1 = (r - L_r + 1) + 1 \leq n + 1,$$

so $H_{r+1}$ has $n + 1$ rows.

If $L_{r+1} > L_r$, then by Theorem 2 in [Massey 1969] we have $L_{r+1} = r - L_r + 1 \leq n$, so $H_{r+1}$ has at most $n$ columns. Again by the maximality of $r$, we must have

$$n < (r+1) - L_{r+1} + 1 = (r+1) - (r - L_r + 1) + 1 = L_r + 1 \leq n + 1,$$

so again $H_{r+1}$ has $n + 1$ rows. □

Lemma 9 implies the following for any $n \times n$ Hankel matrix $H$: if we run the Berlekamp/Massey algorithm on the entries of $H$, then there will be an $r \in \{1, \ldots, 2n-2\}$ such that $H_r$ is a leading submatrix of entire columns of $H$, and $H_{r+1}$ is obtained by augmenting an appropriate row of entries to $H_r$, but $H_{r+1}$ is not a submatrix of $H$.

**Definition 1.** We will say that the Berlekamp/Massey algorithm *exits* an $n \times n$ Hankel matrix $H$ at $r$ if $H_r$ is a submatrix of $H$ but $H_{r+1}$ is not. We make the convention that if $L_1 = \cdots = L_n = 0$ (so that $H_{n-1}$ is $n \times 0$ and $H_n$ is $(n+1) \times 0$), then we say that the algorithm exits at $n-1$. Also, we use the terminology *exits at* $2n-1$ even though $a_{2n-1}$ is not defined in $H$.

**Lemma 10.** *Let $H$ be a square Hankel matrix. If $A$ is a non-singular leading principal submatrix of $H$, then $H_r = A$ for some $r \geq 1$, when running the Berlekamp/Massey algorithm on the entries of $H$.*

*Proof.* Let

$$A = \begin{bmatrix} a_0 & \ldots & a_{k-1} \\ \vdots & \iddots & \vdots \\ a_{k-1} & \ldots & a_{2k-2} \end{bmatrix}.$$

Then because $A$ is a square Hankel matrix, Lemma 9 implies that the Berlekamp/Massey algorithm will exit $A$ at one of $k-1, k, \ldots, 2k-1$. Let $m-1$ be that index and suppose $m-1 \leq 2k-2$. Then we may write

$$H_m = \begin{bmatrix} \tilde{A} \\ y^T \end{bmatrix}_{(m-L_m+1) \times L_m}, \quad h_m = \begin{bmatrix} \tilde{a} \\ \alpha \end{bmatrix}_{(m-L_m+1) \times 1},$$

where $\tilde{A}$ is an appropriate leading submatrix of columns of $A$, and $\tilde{a}$ is an appropriate column of $A$. Then we have

$$A = \begin{bmatrix} \tilde{A} & \tilde{a} & B \end{bmatrix},$$

so that

$$\begin{bmatrix} \tilde{A} & \tilde{a} & B \end{bmatrix} \cdot \begin{bmatrix} \lambda_m \\ -1 \\ 0 \end{bmatrix} = 0,$$

hence $A$ is singular, a contradiction.

Thus, we must have that the Berlekamp/Massey algorithm exits $A$ at $2k-1$, so that

$$H_{2k} = \begin{bmatrix} H_{2k-1} \\ y^T \end{bmatrix} = \begin{bmatrix} A \\ y^T \end{bmatrix},$$

hence $H_{2k-1} = A$. □

**Corollary 3.** *The Berlekamp/Massey algorithm will exit an $n \times n$ Hankel matrix $H$ at one of $n-1, n, \ldots, 2n-2$ if and only if the matrix is singular; the algorithm will exit at $2n-1$ if and only if the matrix is non-singular.*

*Proof.* By the proof of Lemma 10, the algorithm will exit at $2n-1$ if $H$ is non-singular.

Now suppose that the algorithm exits at $2n-1$. Then $H_{2n}$ will be formed by adding a row to $H_{2n-1}$, which will be $H$ itself, and we have

$$H_{2n} = \begin{bmatrix} H_{2n-1} \\ y^T \end{bmatrix} = \begin{bmatrix} H \\ y^T \end{bmatrix},$$

so that

$$H = H_{2n-1} = \begin{bmatrix} a_0 & \cdots & a_{L_{2n-1}-1} \\ \vdots & \iddots & \vdots \\ a_{2n-1-L_{2n-1}} & \cdots & a_{2n-2} \end{bmatrix}.$$

We see then that $L_{2n-1} = n$, hence $H$ is $L_{2n-1} \times L_{2n-1}$, and thus non-singular by the proof of Lemma 2 in [Kaltofen and Yuhasz 2013a]. □

## 4.3 Counting Singular Hankel Matrices

Let $\mathcal{H}^{n\times n}$ denote the set of all $n \times n$ Hankel matrices. We define maps

$$\begin{aligned} \varphi : \mathcal{H}^{n\times n}_{\text{non-singular}} &\to \mathcal{H}^{n\times n}_{\text{singular}} \\ [a_0, \ldots, a_k, \ldots, a_{2n-2}] &\mapsto [a_0, \ldots, a_{k-1}, a'_k, a_{k+1}, \ldots, a_{2n-2}] \end{aligned}$$

and

$$\begin{aligned} \psi : \mathcal{H}^{n\times n}_{\text{singular}} &\to \mathcal{P}(\mathcal{H}^{n\times n}_{\text{non-singular}}) \\ [a_0, \ldots, a_k, \ldots, a_{2n-2}] &\mapsto \left\{ [a_0, \ldots, a_{k-1}, a'_k, a_{k+1}, \ldots, a_{2n-2}] \;\middle|\; a'_k \in \mathbb{F}_q \setminus a_k \right\} \end{aligned}$$

in the following way: we run the Berlekamp/Massey algorithm on the list of entries associated with a Hankel matrix $H$, and we let $k$ be maximal such that $L_k < n$ and $k - L_k + 1 = n$ (i.e., $H_k$ is a *proper* submatrix of columns of $H$). Because $L_k < n$, $H_{k+1}$ will exit $H$ if and only if $H$ is singular; $H_{k+1}$ will have $L_{k+1} > L_k$ columns (and remain a submatrix of $H$) if and only if $H$ is non-singular.

We may block $H$ as

$$\begin{bmatrix} H_k & h_k & B \end{bmatrix} = \left[\begin{array}{cccc|c} a_0 & \cdots & a_{L_{k-1}} & a_{L_k} & \\ \vdots & \iddots & \vdots & \vdots & B \\ a_{k-L_k} & \cdots & a_{k-1} & a_k & \end{array}\right]$$
$$= \left[\begin{array}{cc|c} y_0^T & a_{L_k} & \\ \vdots & \vdots & B \\ y_{k-L_k}^T & a_k & \end{array}\right]$$

where $B$ may be empty, and $H_k\lambda_k - h_k = (0, 0, \ldots, 0, y_{k-L_k}^T\lambda_k - a_k)^T$.

We will have $H$ non-singular if and only if $y_{k-L_k}^T\lambda_k \neq a_k$, again by the proof of Lemma 2 in [Kaltofen and Yuhasz 2013a], so we define

$$\varphi(H) = [a_0, \ldots, a_{k-1}, y_{k-L_k}^T\lambda_k, a_{k+1}, \ldots, a_{2n-2}].$$

Similarly, if $H$ is singular (so that $y_{k-L_k}^T\lambda_k = a_k$), then changing $a_k$ to any other value will result in a non-singular matrix, so we define

$$\psi(H) = \left\{[a_0, \ldots, a_{k-1}, a'_k, a_{k+1}, \ldots, a_{2n-2}] \;\middle|\; a'_k \neq a_k\right\}.$$

**Lemma 11.** *Given $H \in \mathcal{H}^{n\times n}_{\text{singular}}$, $\varphi\big(\psi(H)\big) = H$.*

*Proof.* Given $H \in \mathcal{H}^{n\times n}_{\text{singular}}$, say $H = [a_0, \ldots, a_{2n-2}]$, we run the Berlekamp/Massey algorithm on the list of entries of $H$ to get

$$\psi(H) = \left\{[a_0, \ldots, a_{k-1}, a'_k, a_{k+1}, \ldots, a_{2n-2}] \;\middle|\; a'_k \neq a_k\right\}.$$

If we run the Berlekamp/Massey algorithm on $\psi(H)$, then $H_i$, $h_i$ and $\lambda_i$ will agree for $i = 1, 2, \ldots, k$, and we will have $y_{k-L_k}^T\lambda_k = a_k \neq a'_k$. By the discussion of $\varphi$ above, we will have

$$\begin{aligned} \varphi\big(\psi(H)\big) &= [a_0, \ldots, a_{k-1}, y_{k-L_k}^T\lambda_k, a_{k+1}, \ldots, a_{2n-2}] \\ &= [a_0, \ldots, a_{k-1}, a_k, a_{k+1}, \ldots, a_{2n-2}] \\ &= H \end{aligned}$$

as proposed. □

Lemma 11 immediately implies that if $H \neq \bar{H}$ in $\mathcal{H}^{n\times n}_{\text{singular}}$, then $\psi(H) \cap \psi(\bar{H}) = \{\emptyset\}$: if a matrix $\bar{\bar{H}}$ were in the intersection, then we would have $H = \varphi(\bar{\bar{H}}) = \bar{H}$, a contradiction.

We now use these maps to count singular $n \times n$ Hankel matrices, allowing a collection of entries to be fixed to prescribed values.

**Definition 2.** Given an $n \times n$ Hankel matrix $[a_0, \ldots, a_{2n-2}]$, we define

$$L = (L_{\text{ind}}, L_{\text{val}}) = ([i_1, \ldots, i_k], [\alpha_1, \ldots, \alpha_k])$$

(where $k \leq 2n-1$) to represent fixed entries in $H$, where $a_{i_j}$ is fixed to $\alpha_j$. When counting singular Hankel matrices over $\mathbb{F}_q$, we let $a_i$ vary over $\mathbb{F}_q$ if $i \notin L_{\text{ind}}$. We will let $\mathcal{H}_L^{n\times n}$ denote the set of Hankel matrices with entries fixed according to $L$. Note that $\text{card}(\mathcal{H}_L^{n\times n}) = q^{2n-1-k}$.

**Theorem 5** (General Count)**.** *The number of singular $n \times n$ Hankel matrices with entries fixed according to $L$ (as in Definition 2), where either $L_{\text{ind}} \subseteq \{0, \ldots, n-1\}$ or $L_{\text{ind}} \subseteq \{n-1, \ldots, 2n-2\}$, is equal to*

$$\sigma(n,q,L) = \begin{cases} q^{2n-2-k}, & \text{if } n-1 \notin L_{\text{ind}} \\ & \text{or if } n-1 \in L_{\text{ind}} \text{ with some} \\ & \quad \text{other } j \in L_{\text{ind}} \text{ and } \alpha_j \neq 0 \\ q^{2n-2-k} - q^{n-2}, & \text{if } n-1 \in L_{\text{ind}},\ \alpha_{n-1} \neq 0, \\ & \quad \text{and all other } \alpha_j = 0 \\ q^{2n-2-k} - q^{n-2} + q^{n-1}, & \text{if } n-1 \in L_{\text{ind}},\ \alpha_{n-1} = 0, \\ & \quad \text{and all other } \alpha_j = 0 \end{cases}.$$

*Proof.* We first prove the counts for $L_{\text{ind}} \subseteq \{0, \ldots, n-1\}$.

Suppose that $n-1 \notin L_{\text{ind}}$. Given a singular Hankel matrix $H \in \mathcal{H}_L^{n\times n}$, we run the Berlekamp/Massey algorithm on the entries of $H$; the algorithm will exit at one of the entries on the bottom row because $H$ is singular. Note that $n-1 \notin L_{\text{ind}}$ implies that $\varphi^{-1}(H)$ will yield a unique set of $q-1$ non-singular Hankel matrices for every singular $H \in \mathcal{H}_L^{n\times n}$ (because there is no restriction on any entry of the bottom row). It follows that a fraction of $1/q$ of the $q^{2n-1-k}$ matrices in $\mathcal{H}_L^{n\times n}$ will be singular.

Next, suppose that $n-1 \in L_{\text{ind}}$ and $\alpha_j \neq 0$ for some other $j \in L_{\text{ind}}$. We again run the Berlekamp/Massey algorithm on the entries of a singular $H \in \mathcal{H}_L^{n\times n}$, but now we have a restriction on an element of the bottom row (i.e., $a_{n-1}$), which poses a problem if the algorithm exits at $a_{n-1}$. However, the condition $\alpha_j \neq 0$ for some $j \in \{0, \ldots, n-2\}$ guarantees that the largest proper non-singular leading principal submatrix will have size at least $1 \times 1$, so the algorithm *cannot* exit at $a_{n-1}$. Thus, the map $\varphi$ is defined for each non-singular $H \in \mathcal{H}_L^{n\times n}$ and is surjective, so the count follows as in the case of $n-1 \notin L_{\text{ind}}$.

Now suppose that $n-1 \in L_{\text{ind}}$, $\alpha_{n-1} \neq 0$, and all other $\alpha_j \in L_{\text{val}}$ are zero. Consider the subset $N$ of $\mathcal{H}_L^{n\times n}$ where $a_0 = \cdots = a_{n-2} = 0$; there are $q^{n-1}$ such matrices, which are all non-singular because $\alpha_{n-1} \neq 0$. The map $\varphi$ is not defined on this set because the Berlekamp/ Massey algorithm would exit at $a_{n-1}$, and thus $\varphi$ would attempt to change $a_{n-1}$ to zero, which cannot be done. The matrices in $N$ do not contribute to the count, so we restrict the domain of $\varphi$ to $(\mathcal{H}_L^{n\times n} \cap \mathcal{H}_{\text{non-singular}}^{n\times n}) \setminus N$. As above, the condition $\alpha_j \neq 0$ for some $j \in \{0, \ldots, n-2\}$ guarantees that the map $\varphi$ is defined for each non-singular matrix in $\mathcal{H}_L^{n\times n} \setminus N$, and again is surjective. It follows that a fraction of $1/q$ of the $q^{2n-1-k} - q^{n-1}$ matrices in $\mathcal{H}_L^{n\times n} \setminus N$ is singular.

Last, suppose that $n-1 \in L_{\text{ind}}$, $\alpha_{n-1} = 0$, and all other $\alpha_j \in L_{\text{val}}$ are zero. We consider the set $N$ from above, but now all matrices in $N$ are singular because $\alpha_{n-1} = 0$. Restricting $\varphi$ to $(\mathcal{H}_L^{n\times n} \cap \mathcal{H}_{\text{non-singular}}^{n\times n}) \setminus N$ yields the same result, so we simply add the $q^{n-1}$ singular matrices in $N$ to the previous count.

To prove the result for $L_{\text{ind}} \subseteq \{n-1, \ldots, 2n-2\}$, let

$$J_n = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & & 1 & 0 \\ & \iddots & & \vdots \\ 1 & & 0 & 0 \end{bmatrix} \in \mathbb{F}_q^{n\times n}$$

be the "anti-identity" matrix. Consider the linear transformation

$$T_J : \mathcal{H}_L^{n\times n} \to \mathcal{H}_{L'}^{n\times n} \quad \text{via} \quad H \mapsto J_n H J_n^{-1},$$

which is a bijection onto its image, where $L'$ is the set obtained by mapping each $j \in L_{\text{ind}}$ to $2n-2-j$.

Note that in $\mathcal{H}_{L'}^{n\times n}$, all fixed entries are along or above the anti-diagonal, as they were in the case of $L_{\text{ind}} \subseteq \{0, \ldots, n-1\}$. We can therefore use the methods above to conclude the same counts in $\mathcal{H}_{L'}^{n\times n}$. Because $T_J$ is a bijection onto its image (and preserves singularity/non-singularity), we conclude the same counts for $\mathcal{H}_L^{n\times n}$. □

## 4.4 Counting Block-Hankel Matrices with Block Generic Rank Profile

**Definition 3.** We say that a block matrix $A$ (of square submatrices of dimension $m$) of rank $mr$ has *block generic rank profile* if $\text{rank}(A_k) = mk$ for $k = 1, 2, \ldots, r$, where $A_k$ is the $k \times k$ block leading principal submatrix of $A$.

**Lemma 12.** *The number of block-Hankel matrices ($m \times m$ submatrices arranged in $r \times r$ block-Hankel form) of rank $mr$ with block generic rank profile, denoted by $\mathcal{H}_{\text{bgrp}}^{mr\times mr}(r)$, is equal to $q^{m^2(r-1)}\big(\prod_{i=0}^{m-1}(q^m - q^i)\big)^r$.*

*Proof.* The proof follows by induction. For the case $r = 1$, $\mathcal{H}_{\text{blps}}^{m\times m}$ is simply the number of non-singular $m \times m$ matrices over $\mathbb{F}_q$, which is $\prod_{i=0}^{m-1}(q^m - q^i)$.

Now let $r > 1$ and suppose that the $(r-1)\times(r-1)$ block leading principal submatrix $A_{r-1}$ is non-singular:

$$H = \left[\begin{array}{cccc|c} M_0 & M_1 & \dots & M_{r-2} & M_{r-1} \\ M_1 & M_2 & \dots & M_{r-1} & M_r \\ \vdots & \iddots & \iddots & \vdots & \vdots \\ M_{r-2} & M_{r-1} & \dots & M_{2r-4} & M_{2r-3} \\ \hline M_{r-1} & M_r & \dots & M_{2r-3} & M_{2r-2} \end{array}\right] = \begin{bmatrix} A_{r-1} & B_{r-1} \\ C_{r-1} & M_{2r-2} \end{bmatrix}.$$

We derive conditions on $M_{2r-3}$ and $M_{2r-2}$ that make $H$ non-singular. It is clear that for any choice of $M_{2r-3}$, the system $A_{r-1}X = M_{2r-3}$ has a unique solution. We now determine conditions on the columns of $M_{2r-2}$.

Let the columns of $B_{r-1}$ be denoted $b_0, b_1, \dots, b_{m-1}$, and the columns of $M_{2r-2}$ denoted $v_0, v_1, \dots, v_{m-1}$, and consider the matrix

$$\begin{bmatrix} A_{r-1} & b_0 \\ C_{r-1} & v_0 \end{bmatrix}.$$

The system $A_{r-1}x = b_0$ will have a unique solution $x$ regardless of $b_0$, and correspondingly the block $2 \times 2$ matrix above will have full column rank if and only if $C_{r-1}x \neq v_0$. Thus, there are $(q^m - 1)$-many choices for $v_0$.

Next, suppose that we have chosen $v_0, \dots, v_{t-1}$ so that the matrix

$$\begin{bmatrix} A_{r-1} & b_0 & \dots & b_{t-1} \\ C_{r-1} & v_0 & \dots & v_{t-1} \end{bmatrix}$$

has full column rank. Then the matrix

$$\begin{bmatrix} A_{r-1} & b_0 & \dots & b_{t-1} & b_t \\ C_{r-1} & v_0 & \dots & v_{t-1} & v_t \end{bmatrix}$$

will have full column rank if and only if the vector $(b_t, v_t)^T$ is not in the span of the previous

columns. We see that if the system

$$\begin{bmatrix} A_{r-1} \\ C_{r-1} \end{bmatrix} x = \begin{bmatrix} b_t + \sum_{i=0}^{t-1} \alpha_i b_i \\ v_t + \sum_{i=0}^{t-1} \alpha_i v_i \end{bmatrix}$$

has a solution, then it will be unique by the non-singularity of $A_{r-1}$. Thus, for each choice of $(\alpha_0, \ldots, \alpha_{t-1}) \in \mathbb{F}_q^t$, there is one vector that $v_t$ must avoid, and so there are $(q^m - q^t)$-many choices for $v_t$. It follows that the number of suitable matrices $M_{2r-2}$ is $\prod_{i=0}^{m-1}(q^m - q^i)$.

Combining this with the fact that $M_{2r-3}$ may be arbitrary, we see that the number of block-Hankel matrices (with $A_{r-1}$ as the $mr \times mr$ block leading principal submatrix) that have every block leading principal submatrix non-singular is $q^{m^2}\big(\prod_{i=0}^{m-1}(q^m - q^i)\big)$.

Overall, it follows that there are $q^{m^2(r-1)}\big(\prod_{i=0}^{m-1}(q^m - q^i)\big)^r$ block-Hankel matrices of rank $mr$ with block generic rank profile. □

**Theorem 6.** *The number of block-Hankel matrices ($m \times m$ submatrices arranged in $n \times n$ block-Hankel form) of rank $mr$ with block generic rank profile, denoted by $\mathcal{H}_{\text{bgrp}}^{mn \times mn}(r)$, is equal to*

$$\mathcal{H}_{\text{bgrp}}^{mn \times mn}(r) = \begin{cases} q^{m^2 r}\Big(\prod_{i=0}^{m-1}(q^m - q^i)\Big)^r, & r < n \\ q^{m^2(r-1)}\Big(\prod_{i=0}^{m-1}(q^m - q^i)\Big)^r, & r = n \end{cases}.$$

*Proof.* The case $r = n$ is proved in Lemma 12, so we assume $r < n$. Let $H$ be such a matrix. Then we can write

$$H = \left[\begin{array}{ccc|c|ccc} M_0 & \ldots & M_{r-1} & M_r & M_{r+1} & \ldots & M_{n-1} \\ \vdots & \iddots & \vdots & \vdots & \vdots & \iddots & \vdots \\ M_{r-1} & \ldots & M_{2r-2} & M_{2r-1} & M_{2r} & \ldots & M_{n+r-2} \\ \hline M_r & \ldots & M_{2r-1} & M_{2r} & M_{2r+1} & \ldots & M_{n+r-1} \\ \vdots & \iddots & \vdots & \vdots & \vdots & \iddots & \vdots \\ M_{n-1} & \ldots & M_{n+r-2} & M_{n+r-1} & M_{n+r} & \ldots & M_{2n-2} \end{array}\right]$$

$$= \left[\begin{array}{c|c|c} A_r & B_r & \\ \hline C_{r+1} & M_{2r} & \\ \vdots & \vdots & D \\ C_n & M_{n+r-1} & \end{array}\right],$$

where $A_r$ is non-singular. It is clear that for any choice of $M_{2r-1}$, the system $A_r X = B_r$ has a unique solution.

Let the columns of $B_r$ be denoted $b_0, b_1, \ldots, b_{m-1}$, and similarly let the columns of

$[M_{2r}^T \quad \dots \quad M_{n+r-1}^T]^T$ be denoted $v_0, v_1, \dots, v_{m-1}$. As in the proof of Lemma 12, consider the matrix

$$\begin{bmatrix} A_r & b_0 \\ C_{r+1} & v_0 \end{bmatrix}.$$

The system $A_r x = b_0$ will have a unique solution $x$ regardless of $b_0$, and correspondingly the block $2 \times 2$ matrix above will have rank $mr$ if and only if $C_{r+1}x = v_0$. The matrix $H$ must have rank $mr$, so we see that $v_0$ is predetermined.

Next, suppose that

$$\begin{bmatrix} A_r & b_0 & \dots & b_{t-1} \\ C_{r+1} & v_0 & \dots & v_{t-1} \end{bmatrix}$$

has rank $mr$. Then the matrix

$$\begin{bmatrix} A_r & b_0 & \dots & b_{t-1} & b_t \\ C_{r+1} & v_0 & \dots & v_{t-1} & v_t \end{bmatrix}$$

will have rank $mr$ if and only if the vector $(b_t, v_t)^T$ is in the span of the previous columns, which is equivalent to $(b_t, v_t)^T$ being in the span of the first $mr$ columns. Given any $b_t$, we see that the system

$$\begin{bmatrix} A_r \\ C_{r+1} \end{bmatrix} x = \begin{bmatrix} b_t \\ v_t \end{bmatrix}$$

will have at most one solution, by the non-singularity of $A_r$. There is a unique choice of $v_t$ that will make the system solvable: we set $v_t$ to $C_{r+1}x$, where $x$ is the unique solution to $A_r x = b_t$. Thus, $v_t$ is predetermined. It follows that $M_{2r}, \dots, M_{n+r-1}$ are predetermined.

Moreover, $M_{n+r}, \dots, M_{2n-2}$ are predetermined, a fact that was corrected by an anonymous referee. To see why, note that the first $mr$ columns of $H$ form a basis for the column space of $H$. If we denote the columns of $D$ by $d_0, d_1, \dots, d_{n-r-1}$, then the matrix

$$\left[\begin{array}{c|c|c} A_r & B_r & \\ \hline C_{r+1} & M_{2r} & \\ \vdots & \vdots & d_0 \\ C_n & M_{n+r-1} & \end{array}\right]$$

will have rank $mr$ if and only if $d_0$ is in the span of the first $m(r+1)$ columns of $H$, which equals the span of the first $mr$ columns of $H$. By the non-singularity of $A_r$, the resulting column relation would be unique, so the last $m$ entries of $d_0$ (i.e., the first column of $M_{n+r}$) are predetermined. The same argument follows inductively for every column of $D$, so that each of $M_{n+r}, \dots, M_{2n-2}$ is predetermined.

Because only $M_{2r-1}$ may be arbitrary, it follows that $H$ is one of

$$q^{m^2(r-1)}\left(\prod_{i=0}^{m-1}(q^m-q^i)\right)^r \cdot q^{m^2} = q^{m^2 r}\left(\prod_{i=0}^{m-1}(q^m-q^i)\right)^r$$

many matrices. □

For the case $m = 1$, Theorem 6 implies that the number of $n \times n$ Hankel matrices (with entries from $\mathbb{F}_q$) of rank $r$ with generic rank profile is

$$\mathcal{H}_{\text{bgrp}}^{n\times n}(r) = \begin{cases} q^r(q-1)^r, & r < n \\ q^{r-1}(q-1)^r, & r = n \end{cases}.$$

We can compare this to the result in [Kaltofen and Lobo 1996], which states that the number of $n \times n$ Toeplitz matrices (with entries from $\mathbb{F}_q$) of rank $r$ with generic rank profile is

$$N_r = \begin{cases} q^{2n-2}\left(1-\frac{1}{q}\right)^2\left(1-\frac{q-1}{q^2}\right)^{r-1}, & 0 < r < n \\ q^{2n-1}\left(1-\frac{1}{q}\right)\left(1-\frac{q-1}{q^2}\right)^{n-1}, & r = n \end{cases}.$$

We have investigated the properties of block-Hankel matrices, but we do not know explicit formulas for how many block-Hankel matrices are singular (with or without certain blocks fixed). Presented below are some brute-force counts for the number of singular block-Hankel matrices ($m \times m$ submatrices arranged in $n \times n$ block-form, with entries from $\mathbb{F}_q$).

Table 3: Fraction of singular block-Hankel matrices with entries from a finite field

| $m$ | $n$ | $q$ | Singular/Total |
|---|---|---|---|
| 2 | 2 | 2 | $\frac{2704}{4096}=\frac{2^4 \cdot 13^2}{2^{12}}$ |
| 2 | 2 | 3 | $\frac{226881}{531441}=\frac{3^4 \cdot 2801}{3^{12}}$ |
| 2 | 3 | 2 | $\frac{701440}{1048576}=\frac{2^{10} \cdot 5 \cdot 137}{2^{20}}$ |
| 3 | 2 | 2 | $\frac{93790208}{134217728}=\frac{2^{13} \cdot 107^2}{2^{27}}$ |
| 2 | 2 | 5 | $\frac{58080625}{244140625}=\frac{5^4 \cdot 19 \cdot 67 \cdot 73}{5^{12}}$ |
| 2 | 4 | 2 | $\frac{180158464}{268435456}=\frac{2^{16} \cdot 2749}{2^{28}}$ |

**Note 4.** The following modifications have been made from [Comer and Kaltofen 2012].

1. Page 44: The citation of Lemma 2 has been updated.
2. Page 47: "adding rows" has been changed to "augmenting rows".
3. Page 50: The citation of Lemma 2 has been updated.
4. Page 51: "We may factor $H$" has been changed to "We may block $H$".
5. Page 51: The citation of Lemma 2 has been updated.

Chapter Five

# Conclusion

## 5.1 Future Work

For counting square singular Hankel matrices, an explicit formula is still not known for the case of fixing anti-diagonals above and below the main anti-diagonal. Here, specific non-zero values of fixed anti-diagonals can determine a different number of singular matrices; by contrast, the General Count of Theorem 5 depends mostly on whether we fix anti-diagonals to zero. For example, consider the $2 \times 2$ Hankel matrix denoted by $H = [a_0, a_1, a_2]$. If we fix both $a_0$ and $a_2$ to $\alpha_0$ and $\alpha_2$, respectively, then the number of singular such matrices depends on solutions (in $a_1$) of the equation $a_1^2 = \alpha_0\alpha_2$. Over $\mathbb{F}_3$, there are 2 singular such matrices for $(\alpha_0, \alpha_2) = (1, 1)$, but no singular such matrix for $(\alpha_0, \alpha_2) = (1, 2)$.

Also still not known is an explicit formula for the number of singular block-Hankel matrices, even for the case when no block-anti-diagonal is fixed (see Table 3). It may be that the Matrix Berlekamp/Massey algorithm of [Kaltofen and Yuhasz 2013b] could provide some insight for the count in this case.

For linearly generated sequences with errors, Examples 1 and 2 prove the necessary number of sequence entries to guarantee a unique set of corrections and intended minimal generator. These examples do not directly generalize to show a necessary bound for the number of evaluations for sparse interpolation in general. It should be noted that the sequence entries in Prony's sparse interpolation algorithm have more structure than our arbitrary linearly generated sequences: Prony's sequence entries are evaluations of a sparse polynomial at consecutive powers of a field element; one should be able to take advantage of this additional structure in order to decrease the required number of evaluations. Also, with regard to the numeric algorithm of Section 2.6, it is not clear how to prevent numeric "deceptive blocks": a deception (or failed

majority vote) can occur due to misleading noise as well as an outlier error.

In the numeric versions of shifted-basis sparse interpolation algorithms (see Section 3.2), we implement outlier detection at a different time than the power basis algorithm: outlier errors are detected as an immediate pre-conditioning on the evaluations, then the error-free algorithms are executed on presumably "clean" input. It would be interesting to explore the inner steps of each algorithm to see if error-correction can be integrated into the existing steps of the error-free algorithms, as in the power basis case. Before exploring this, however, both algorithms of Section 3.2 should be analyzed for steps that can be numerically unstable.

## 5.2 Concluding Remarks

The Majority Rule Berlekamp/Massey algorithm of Section 2.5 addresses error correction for linearly generated sequences with exact arithmetic. For characteristic not equal to 2, we have shown optimality in the number of sequence entries required to guarantee a unique minimal generator and set of corrections; for characteristic 2, the algorithm still computes unique results. This error-correction algorithm can be integrated into Prony's sparse interpolation algorithm, where the only change to the algorithm inputs would be oversampling during the collection of evaluation points. It does require, of course, that the error model is such that one can place a bound $E$ on the number of errors obtained from a sufficiently large set of evaluations, as is described in Remark 1.1 of [Kaltofen and Yang 2013].

The high error rate of the error-correcting algorithm, requiring an additional $4t$ sequence entries for each error that may be introduced into the sequence, calls into question the practicality of implementing it as an error-correcting coding scheme: changing only a few entries in an already-linearly-generated sequence, without supplying additional sequence entries, causes the recovery of the intended sequence and minimal generator to become much more difficult (if possible at all, considering that deceptions may now occur). However, from a suggestion of one of the referees of [Comer, Kaltofen, and Pernet 2012], this high *$k$-error linear complexity* may prove useful in the realm of cryptography, where it is an advantage to have simultaneously an "easy encoding process" and "difficult decoding process" for private communications across a public channel.

The results of Section 3.3 show a floating-point-arithmetic implementation of Blahut's decoding algorithm for Reed-Solomon codes, which was originally implemented for finite fields. Currently, we have the algorithm analyzed for the case of $k = 1$ (i.e., one erroneous evaluation) in an interpolation of a dense polynomial; the bound in (7) shows a sufficient condition to guarantee proper recovery of the error location. The plotted curves in Figure 1, which show the trend when the outlier magnitude approaches that of general noise, show what can happen when no outlier occurred; that is, they show the case $k = 0$ as well. Thus, before determining

where an outlier occurred, it is a non-trivial step to determine whether we should search for an outlier at all. As seen in Figure 1a, one can obtain a "deception" during outlier detection if in fact no outlier occurred. While the case of $k = 1$ is conducive to obtaining conditions for outlier detection/deception, multiple outliers may interfere with each other to cause deceptions even when their individual magnitudes are large, so that the case $k \geq 2$ provides an interesting challenge.

# References


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.

Elwyn R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill Series in System Science. McGraw-Hill, 1968.

R. E. Blahut. Transform techniques for error control codes. *IBM J. Res. Dev.*, 23:299–315, May 1979. ISSN 0018-8646. doi: http://dx.doi.org/10.1147/rd.233.0299.

R. E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, 1983.

Brice Boyer, Matthew T. Comer, and Erich L. Kaltofen. Sparse polynomial interpolation by variable shift in the presence of noise and outliers in the evaluations. In *Electr. Proc. Tenth Asian Symposium on Computer Mathematics (ASCM 2012)*, 2012. URL: EKbib/12/BCK12.pdf.

Claude Brezinski. *Computational Aspects of Linear Control*. Springer Verlag, Heidelberg, Germany, 2002. ISBN 978-1402007118.

James A. Cadzow. Signal enhancement—a composite property mapping approach. *IEEE Trans. Acoust., Speech, Signal Process.*, ASSP-36:49–62, 1988.

Matthew T. Comer and Erich L. Kaltofen. On the Berlekamp/Massey algorithm and counting singular Hankel matrices over a finite field. *J. Symbolic Comput.*, 47(4):480–491, April 2012. URL: EKbib/10/CoKa10.pdf.

Matthew T. Comer, Erich L. Kaltofen, and Clément Pernet. Sparse polynomial interpolation and Berlekamp/Massey algorithms that correct outlier errors in input values. In Joris van der Hoeven and Mark van Hoeij, editors, *ISSAC 2012 Proc. 37th Internat. Symp. Symbolic Algebraic Comput.*, pages 138–145, New York, N. Y., July 2012. Association for Computing Machinery. ISBN 978-1-4503-1269. URL: EKbib/12/CKP12.pdf.

D. E. Daykin. Distribution of bordered persymmetric matrices in a finite field. *J. reine u. angew. Math.*, 203:47–54, 1960.

M. Garcìa-Armas, S. R. Ghorpade, and S. Ram. Relatively prime polynomials and nonsingular hankel matrices over finite fields. *J. Combin. Theory Ser. A*, 118:819–828, 2011.

Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Comput. Sci.*, 410(27-29):2659 – 2662, 2009. ISSN 0304-3975. doi: DOI:10.1016/j.tcs.2009.03.030. URL http://www.csd.uwo.ca/~eschost/publications/interp.pdf.

Joachim von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, New York, Melbourne, 1999. ISBN 0-521-64176-4. Second edition 2003.

Mark Giesbrecht and Daniel S. Roche. Interpolation of shifted-lacunary polynomials. *Computational Complexity*, 19(3):333–354, September 2010.

Mark Giesbrecht and Daniel S. Roche. Diversification improves interpolation. In [Leykin 2011], pages 123–130. ISBN 978-1-4503-0675-1.

Mark Giesbrecht, Erich Kaltofen, and Wen-shin Lee. Algorithms for computing sparsest shifts of polynomials in power, Chebychev, and Pochhammer bases. *J. Symbolic Comput.*, 36(3–4): 401–424, 2003. Special issue Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL: EKbib/03/GKL03.pdf.

Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic-numeric sparse interpolation of multivariate polynomials (extended abstract). In *Proc. Ninth Rhine Workshop on Computer Algebra (RWCA'04), University of Nijmegen, the Netherlands*, pages 127–139, 2004.

Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. In Jean-Guillaume Dumas, editor, *ISSAC MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput.*, pages 116–123, New York, N. Y., 2006. ACM Press. ISBN 1-59593-276-3. doi: http://doi.acm.org/10.1145/1145768.1145792.

Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. *J. Symbolic Comput.*, 44:943–959, 2009.

D. Yu. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Proc. 28th Annual Symp. Foundations of Comp. Sci.*, pages 166–172, Los Alamitos, California, 1987. IEEE Computer Society Press.

D. Yu. Grigoriev and M. Karpinski. A zero-test and an interpolation algorithm for the shifted sparse polynomials. In *Proc. AAECC-10*, volume 673 of *Lect. Notes Comput. Sci.*, pages 162–169, Heidelberg, Germany, 1993. Springer Verlag.

D. Yu. Grigoriev and Y. N. Lakshman. Algorithms for computing sparse shifts for multivariate polynomials. *Applic. Algebra Engin. Commun. Comput.*, 11(1):43–67, 2000.

Sharon E. Hutton, Erich L. Kaltofen, and Lihong Zhi. Computing the radius of positive semidefiniteness of a multivariate real polynomial via a dual of Seidenberg's method. In [Watt 2010], pages 227–234. ISBN 978-1-4503-0150-3. URL: EKbib/10/HKZ10.pdf.

Kyoki Imamura and Watura Yoshida. A simple derivation of the Berlekamp-Massey algorithm and some applications. *IEEE Trans. Inf. Theory*, IT-33:146–150, 1987.

E. Kaltofen and A. Lobo. On rank properties of Toeplitz matrices over finite fields. In Lakshman Y. N., editor, *Proc. 1996 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'96)*, pages 241–249, New York, N. Y., 1996. ACM Press. URL: EKbib/96/KaLo96_issac.pdf.

E. Kaltofen, Lakshman Y. N., and J. M. Wiley. Modular rational sparse multivariate polynomial interpolation. In S. Watanabe and M. Nagata, editors, *Proc. 1990 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'90)*, pages 135–139. ACM Press, 1990. URL: EKbib/90/KLW90.pdf.

Erich Kaltofen and Wen-shin Lee. Early termination in sparse interpolation algorithms. *J. Symbolic Comput.*, 36(3–4):365–400, 2003. Special issue Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL: EKbib/03/KL03.pdf.

Erich Kaltofen and Zhengfeng Yang. Sparse multivariate function recovery from values with noise and outlier errors. In Manuel Kauers, editor, *ISSAC 2013 Proc. 38th Internat. Symp. Symbolic Algebraic Comput.*, page to appear, New York, N. Y., 2013. Association for Computing Machinery.

Erich Kaltofen and George Yuhasz. On the matrix Berlekamp-Massey algorithm. *ACM Trans. Algorithms*, 2013a. To appear. URL: EKbib/06/KaYu06.pdf.

Erich Kaltofen and George Yuhasz. A fraction free matrix Berlekamp/Massey algorithm. *Linear Algebra and Applications*, 2013b. To appear. URL: EKbib/08/KaYu08.pdf.

Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi. On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In Jan Verschelde and Stephen M. Watt, editors, *SNC'07 Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput.*, pages 11–17, New York, N. Y., 2007. ACM Press. ISBN 978-1-59593-744-5. URL: EKbib/07/KYZ07.pdf.

Erich L. Kaltofen. Fifteen years after DSC and WLSS2 What parallel computations I do today [Invited lecture at PASCO 2010]. In M. Moreno Maza and Jean-Louis Roch, editors, *PASCO'10 Proc. 2010 Internat. Workshop on Parallel Symbolic Comput.*, pages 10–17, New York, N. Y., July 2010. ACM. ISBN 978-1-4503-0067-4. URL: EKbib/10/Ka10_pasco.pdf.

Erich L. Kaltofen and Michael Nehring. Supersparse black box rational function interpolation. In [Leykin 2011], pages 177–185. ISBN 978-1-4503-0675-1. URL: EKbib/11/KaNe11.pdf.

Erich L. Kaltofen, Wen-shin Lee, and Zhengfeng Yang. Fast estimates of Hankel matrix condition numbers and numeric sparse interpolation. In M. Moreno Maza, editor, *SNC'11 Proc. 2011 Internat. Workshop on Symbolic-Numeric Comput.*, pages 130–136, New York, N. Y., June 2011. Association for Computing Machinery. ISBN 978-1-4503-0515-0. URL: EKbib/11/KLY11.pdf.

Majid Khonji, Clément Pernet, Jean-Louis Roch, Thomas Roche, and Thomas Stalinsky. Output-sensitive decoding for redundant residue systems. In [Watt 2010], pages 265–272. ISBN 978-1-4503-0150-3.

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

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

Anton Leykin, editor. *Proc. 2011 Internat. Symp. Symbolic Algebraic Comput. ISSAC 2011*, New York, N. Y., 2011. Association for Computing Machinery. ISBN 978-1-4503-0675-1.

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

James Massey and Thomas Schaub. Linear complexity in coding theory. In G. Cohen and P. Godlewski, editors, *Coding Theory and Applications*, volume 311 of *Lecture Notes in Computer Science*, pages 19–32. Springer Verlag, 1988. ISBN 978-3-540-19368-5. URL http://dx.doi.org/10.1007/3-540-19368-5_2.

Todd K. Moon. *Error correction coding: mathematical methods and algorithms*. Wiley-Interscience, 2005. ISBN 0-471-64800-0 (cloth). URL ftp://uiarchive.cso.uiuc.edu/pub/etext/gutenberg/;http://www.loc.gov/catdir/toc/ecip055/2004031019.html.

Ramamohan Paturi and Pavel Pudlák. On the complexity of circuit satisfiability. In Leonard J. Schulman, editor, *STOC*, pages 241–250. ACM, 2010. ISBN 978-1-4503-0050-6.

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

Irving S. Reed and Glenn S. Solomon. Polynomial codes over certain finite fields. *J. SIAM*, 8 (2):300–304, June 1960. ISSN 0368-4245 (print), 1095-712X (electronic).

Siegfried M. Rump. Structured perturbations part I: Normwise distances. *SIAM J. Matrix Anal. Applic.*, 25(1):1–30, 2003.

Shubhangi Saraf and Sergey Yekhanin. Noisy interpolation of sparse polynomials, and applications. In *IEEE Conference on Computational Complexity*, pages 86–92. IEEE Computer Society, 2011.

Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding Goppa codes. *Information & Control*, 27:87–99, 1975.

Stephen M. Watt, editor. *Proc. 2010 Internat. Symp. Symbolic Algebraic Comput. ISSAC 2010*, New York, N. Y., 2010. Association for Computing Machinery. ISBN 978-1-4503-0150-3.

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