Sparse Polynomial Interpolation and Berlekamp/Massey Algorithms That Correct Outlier Errors in Input Values*

Matthew T. Comer
Dept. of Mathematics, NCSU
Raleigh, NC 27695, USA
mcomer@math.ncsu.edu

Erich L. Kaltofen
Dept. of Mathematics, NCSU
Raleigh, NC 27695, USA
kaltofen@math.ncsu.edu
www4.ncsu.edu/~kaltofen

Clément Pernet
INRIA-MOAIS, LIG, Université Joseph Fourier, Grenoble, France
clement.pernet@imag.fr
membres-liglab.imag.fr/pernet/

ABSTRACT

We propose algorithms performing sparse interpolation with errors, based on Prony's–Ben-Or's & Tiwari's algorithm, using a Berlekamp/Massey algorithm with early termination. First, we present an algorithm that can recover a t-sparse polynomial f from a sequence of values, where some of the values are wrong, spoiled by either random or misleading errors. Our algorithm requires bounds $T \geq t$ and $E \geq e$, where e is the number of evaluation errors. It interpolates $f(\omega^i)$ for $i = 1, \ldots, 2T(E+1)$, where ω is a field element at which each non-zero term evaluates distinctly.

We also investigate the problem of recovering the minimal linear generator from a sequence of field elements that are linearly generated, but where again $e \leq E$ elements are erroneous. We show that there exist sequences of $< 2t(2e+1)$ elements, such that two distinct generators of length t satisfy the linear recurrence up to e faults, at least if the field has characteristic $\neq 2$. Uniqueness can be proven (for any field characteristic) for length $\geq 2t(2e+1)$ of the sequence with e errors. Finally, we present the Majority Rule Berlekamp/Massey algorithm, which can recover the unique minimal linear generator of degree t when given bounds $T \geq t$ and $E \geq e$ and the initial sequence segment of $2T(2E+1)$ elements. Our algorithm also corrects the sequence segment. The Majority Rule algorithm yields a unique sparse interpolant for the first problem.

The algorithms are applied to sparse interpolation algorithms with numeric noise, into which we now can bring outlier errors in the values.

*This material is based on work supported in part by the National Science Foundation under Grants CCF-0830347 and CCF-1115772 (Comer and Kaltofen) and the French ANR under Grant ANR-11-BS02-013 (Pernet).

Categories and Subject Descriptors: I.1.2 [Symbolic and Algebraic Manipulation]: Algorithms; G.1.1 [Numerical Analysis]: Interpolation—smoothing

General Terms: Algorithms, Experimentation

Keywords: linearly generated sequences, polynomial recovery, fault tolerance

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 [10, 1, 13, 14], namely the "GLL"-algorithm [7], 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.

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 [1] 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 [20]. The exact sparse problem is also investigated in [28], 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.

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 [9], 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.

Numeric Sparse Interpolation. Linear recurrence-based sparse interpolation goes back to the French revolution [25]. 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. [4]). 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 [14] together with analysis of the distribution of the corresponding condition numbers [7, 18] 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.

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 [7, 18] 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 6.

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 [1] *Let $b_j = \omega^{e_j}$, where ω 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 Λ for $(a_0, a_1, \ldots)$, using the Berlekamp/Massey algorithm.
2. Compute the roots b_j of Λ, using univariate polynomial factorization.
3. Recover the exponents e_j of f, by repeatedly dividing b_j by ω.
4. Recover the coefficients c_j of f, by solving the transposed $t \times t$ Vandermonde system

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

By Blahut's theorem [2, 22], 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 Λ. In the presence of errors in some of the evaluations, this fails. Our work shows how to overcome that obstruction.

2. LENGTH BOUNDS FOR COMPUTING LINEAR GENERATORS FROM SEQUENCES WITH ERROR

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 α be an e-th root of unity, for e even, and consider the following sequence of $2(2e+1) - 1$ entries:

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

If the odd powers of $-\alpha$ are changed to odd powers of α, then $-\alpha + \lambda$ is the minimal linear generator. However, if the odd powers of α 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:

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

When e is odd, we can take the example for $e+1$, remove the last $4t$ entries, then change the last α^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 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 $m = \max\{l, k\}$, Λ and Ξ are both linear generators for $(a_m, a_{m+1}, \ldots)$, but Λ is minimal by Lemma 3, so $\Lambda \mid \Xi$.

Let $\Xi = \Lambda\Gamma$. Then Γ is monic and has a non-zero constant term (because both Λ and Ξ 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 Γ and p are monomials in λ, 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 $\lambda^k \Lambda$. $\square$

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* Λ *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 Λ_{sr} (i.e., the scaled reciprocal polynomial of Λ) by Lemma 4. We can then call Algorithm 2 with Λ_{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 Λ 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 δ 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 δ, so the algorithm may stop; see Theorem 5 and Remark 1 in [16, 29]. 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 $T > k + t$. $\square$

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 [24] for a treatment of the subject in appropriate details.

Reed-Solomon as evaluation codes. The popular Reed-Solomon codes can be defined (as in their original presentation by [26]) as evaluation interpolation codes. Let K be the finite field $\mathbb{F}_q$, set $n = q - 1$ and let ξ 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 ξ: $c = \text{Ev}(f) = (f(\xi^0), f(\xi^1), \ldots, f(\xi^{n-1})) = V_\xi f$, 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 ε

affects the communication, so that one receives the message $c' = c + \varepsilon$, where e coefficients of ε 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 ε called the syndrome. Blahut's Theorem [3] 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 ξ^i where errors occurred.

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 ε 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 ε of weight t such that $c' = f + V_\xi \varepsilon$.* The Vandermonde matrix V_ξ 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 ξ^{-1}. Left-multiplying $c' = f + V_\xi \varepsilon$ by V_ξ^{-1} makes it a Reed-Solomon decoding problem (based on the primitive root ξ^{-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 [20] 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 guaranties a unique solution, but a with a longer sequence of evaluations, however optimal, with respect to the lower bound proven in Section 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 algorithm

```
Input: (a_0, ..., a_{n-1}) a sequence of field elements.
Result: Λ(λ) = Σ_{i=0}^{L_n} γ_i λ^i a monic polynomial of
        minimal degree L_n ≤ n such that
        Σ_{i=0}^{L_n} γ_i a_{i+j} = 0 for j = 0, ..., 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 Λ.*

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. Section 5 will address this issue with a stronger assumption on the length of the sequence: $n \geq 2T(2E+1)$.

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 $O(T^2 E)$ 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.

Algorithm 2: `SequenceCleanUp`

```
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: $k \leq n - 2t - 1$ initial position for clean-up
Output: $((c_0, \ldots, c_{n-1}), e)$ 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 $(c_k, \ldots, c_{k+2t-1}) = (a_k, \ldots, a_{k+2t-1})$ .
begin
    $(c_0, \ldots, c_{n-1}) \leftarrow (a_0, \ldots, a_{n-1})$
    $i \leftarrow k + 2t$; $e \leftarrow 0$
    while $i \leq n-1$ and $e \leq E$ do
1       if $\sum_{j=0}^{t} \gamma_j c_{j+i-t} \neq 0$ then
2           $c_i \leftarrow -\sum_{j=0}^{t-1} \gamma_j c_{i+j-t}$; $e \leftarrow e+1$
        $i \leftarrow i + 1$
    $i \leftarrow k - 1$
    while $i \geq 0$ and $e \leq E$ do
3       if $\sum_{j=0}^{t} \gamma_j c_{j+i} \neq 0$ then
            $c_i \leftarrow -\sum_{j=1}^{t} \frac{\gamma_j}{\gamma_0} c_{i+j}$; $e \leftarrow e+1$
        $i \leftarrow i - 1$
    return $(c_0, \ldots, c_{n-1}), e$
```

5. THE MAJORITY RULE BERLE–KAMP/MASSEY ALGORITHM

If one has e errors in a sequence of $2t(2e+1)$ linearly generated elements by a generator of degree t, $e+1$ "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 $E \geq e$.

Properties and correctness of the Majority Rule Berlekamp/Massey algorithm. For convenience, we first assume that $T = t$. By supplying Algorithm 4 with $2t(2E+1)$ sequence entries, we guarantee that during Step 1, at least $E+1$ of the Λ_i will be the correct ("intended") generator, Λ, by Lemma 3. If *exactly* $E+1$ of the Λ agree, then every block with Λ 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 Λ 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 Λ 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 Λ is the monic minimal linear generator for any $k > 1$ as well. □

Algorithm 3: `FTBM`: Fault Tolerant Berlekamp/Massey algorithm

```
Input: $(c_0, \ldots, c_{n-1})$ a sequence of field elements.
Input: $T$ an upper bound for $t$, the degree of the
       monic minimal generator.
Input: $E$ an upper 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 $\Lambda(0) \neq 0$.
begin
    $L \leftarrow [\,]$
    for $i \leftarrow 0, \ldots, \lfloor \frac{n}{2T} \rfloor - 1$ do
        $\Lambda \leftarrow$ BM($[a_{2Ti}, \ldots, a_{2Ti+2T-1}]$)
        /* $\Lambda(\lambda) = \gamma_0 + \gamma_1\lambda + \cdots + \gamma_s\lambda^s$, $\gamma_s = 1$ */
        if $\gamma_0 \neq 0$ then
            $((c_0, \ldots, c_{n-1}), e) \leftarrow$
            SequenceCleanUp($\Lambda, E, (a_0, \ldots, a_{n-1}), 2Ti$)
            if $e \leq E$ then
                $L \leftarrow L$.push($((c_0, \ldots, c_{n-1}), \Lambda)$)
    return $L$
```

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 Λ (see, e.g.,* [12], 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 Λ as monic minimal linear generator.*

Proof. First, we consider the case of the last error no later than entry t. Suppose that Λ 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 $(a_{k-1} - b_{k-1})\gamma_0 \neq 0$, con-

Algorithm 4: Majority Rule Berlekamp/Massey

```
Input: $(a_0, \dots, a_{2T(2E+1)-1}) + \vec{\varepsilon}$ where $(a_i)$ is a
       linearly recurrent sequence (of degree $t \le T$) of
       field elements, and $\vec{\varepsilon}$ is a vector of Hamming
       weight $e \le E$. 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: $(a_0, \dots, a_{2T(2E+1)-1})$ the intended sequence
Output: $\Lambda$ the monic minimal linear generator of the
        intended sequence
begin
1   for $i \leftarrow 0, \dots, 2E$ do
      $\Lambda_i \leftarrow$ BM($(b_{0+2Ti}, \dots, b_{2T-1+2Ti})$)
2   $L \leftarrow [0, 1, \dots, 2E]$; $m \leftarrow 0$
3   for $i \in L$ do
4       $L_i \leftarrow [\,]$
5       for $j \in L$ do
6           if $\Lambda_i = \Lambda_j$ then $L_i \leftarrow L_i \cup \{j\}$;
            $L \leftarrow L \setminus \{j\}$
7       if $Card(L_i) > Card(L_m)$ then $m \leftarrow i$
8   $\Lambda \leftarrow \Lambda_m$
9   for $i \in L_m$ do
10      $((c_0, \dots, c_{2T(2E+1)-1}), e) \leftarrow$ Sequence-
        CleanUp($\Lambda, E, (b_0, \dots, b_{2T(2E+1)-1}), 2Ti$)
11      if $e \le E$ then BREAK
12  return $(c_0, \dots, c_{2T(2E+1)-1}), \Lambda$
```

tradiction. 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 $(a_{2t-1}, \dots, a_0)$ and $(b_{2t-1}, b_{2t-2}, \dots, b_{k-1}, a_{k-2}, a_{k-3}, \dots, a_{t-1}, \dots, a_0)$, noting by Lemma 4 that the former sequence has Λ_{sr} as monic minimal linear generator. The original erroneous sequence has Λ as monic minimal linear generator if and only if the reversed erroneous sequence has Λ_{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 Λ_{sr} as monic minimal linear generator. Therefore, the original erroneous sequence cannot have Λ as its monic minimal linear generator. □

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

As stated earlier, if exactly $E+1$ of the Λ_i agree in Step 1, 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 10-11 in parallel, correcting each erroneous block with the nearest clean block.

If *more than* $E+1$ of the Λ_i agree in Step 1, 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 10 with a deceptive block.

Theorem 4 *If Algorithm 2 is called in Step 10 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., $(b_0, \dots, b_{2t-1})$. Suppose the last error in the deceptive block is $b_{k-1} \ne a_{k-1}$, where $t+1 \le k \le 2t$. If $(b_{2t}, b_{2t+1}, \dots, b_{k+t-1}) = (a_{2t}, a_{2t+1}, \dots, a_{k+t-1})$, then there must be a (non-zero) discrepancy in Step 1 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 $\Lambda(0) \ne 0$. In this case, an “erroneous correction” will occur (i.e., Algorithm 2 will change b_{k+t-1} to $b'_{k+t-1} \ne a_{k+t-1}$).

If there is at least one error in entries $b_{2t}, b_{2t+1}, \dots, b_{k+t-1}$, 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 $(c_{2t}, c_{2t+1}, \dots, c_{k+t-1})$ the output of Algorithm 2 when run on entries $b_{2t}, b_{2t+1}, \dots, b_{k+t-1}$ (whether or not there is an error). Let c_s be the last entry such that $c_s \ne a_s$; note that this must exist because $b_{k-1} \ne a_{k-1}$, so that Algorithm 2 cannot return $(c_{2t}, c_{2t+1}, \dots, c_{k+t-1}) = (a_{2t}, a_{2t+1}, \dots, a_{k+t-1})$. Considering entries $b_{s+1}, b_{s+2}, \dots, b_{s+t}$, 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 1 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 10 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 9-11 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 Λ is clean, then the intended generator Λ will be found in Step 8 of Algorithm 4 and all errors will be detected and corrected thereafter. If the first block of $2T$ entries that yields Λ 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 [23]. This contradicts the deceptive $2T$-block yielding Λ.

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 [23], so in fact $L_{2T} > t$, hence the deceptive block of $2T$ entries cannot yield Λ, 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 Λ.

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

6. NUMERIC NOISE AND OUTLIERS

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 [1], 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 [18]: 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 [27], 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 ω 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_{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.

As noted in Section 1 of [18], 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 Λ by obtaining a least squares solution to a well-conditioned $t' \times t'$ Hankel system. (In the exact case, the coefficients of Λ form the solution to the non-singular Hankel system.)

After finding the roots b_j of Λ, we take advantage of the fact that ω is a prime-order root of unity, by comparing the arguments of ω 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 [18], 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 below.

Root	Sparsity Estimates			Majority Vote
ω_1	5	5	7	5
ω_2	?	5	?	?
ω_3	4	6	5	?
ω_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}.

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 [5] and [17, Section 2.1], computing the sparsest shift [11, 6] and the supersparsest shift [8], or the more difficult exact and numeric sparse and supersparse rational function recovery [15, 19]. Our methods immediately apply to algorithms that are based on computing a linear recurrence, such as the supersparse interpolation algorithms in [17] and [5]. 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 [9]. The sparsest shift algorithms in [6] 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 [7, 18], which in turn could be incorporated into the Zippel [30] variable-by-variable multivariate ZNIPR-algorithm for polynomials [15]. 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 [17, Section 2.2], which potentially has better error detection/noise reduction properties.

Acknowledgments: We thank Wen-shin Lee and the referees for their helpful comments.

8. REFERENCES

[1] Ben-Or, M., and Tiwari, P. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proc. 20th Annual ACM STOC* (1988), pp. 301–309.

[2] Blahut, R. E. Transform techniques for error control codes. *IBM J. Res. Dev. 23* (May 1979), 299–315.

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

[4] Cadzow, J. A. Signal enhancement—a composite property mapping approach. *IEEE Trans. Acoust., Speech, Signal Process.* ASSP-*36* (1988), 49–62.

[5] Garg, S., and Schost, Éric. Interpolation of polynomials given by straight-line programs. *Theoretical Comput. Sci. 410*, 27-29 (2009), 2659 – 2662.

[6] Giesbrecht, M., Kaltofen, E., and Lee, W. Algorithms for computing sparsest shifts of polynomials in power, Chebychev, and Pochhammer bases. *J. Symbolic Comput. 36*, 3–4 (2003), 401–424. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/03/GKL03.pdf.

[7] Giesbrecht, M., Labahn, G., and Lee, W. Symbolic-numeric sparse interpolation of multivariate polynomials. *J. Symbolic Comput. 44* (2009), 943–959.

[8] Giesbrecht, M., and Roche, D. S. Interpolation of shifted-lacunary polynomials. *Computational Complexity 19*, 3 (Sept. 2010), 333–354.

[9] Giesbrecht, M., and Roche, D. S. Diversification improves interpolation. In Leykin [21], pp. 123–130.

[10] Grigoriev, D. Y., and Karpinski, M. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In *Proc. 28th Annual Symp. FOCS* (1987), IEEE, pp. 166–172.

[11] Grigoriev, D. Y., and Karpinski, M. A zero-test and an interpolation algorithm for the shifted sparse polynomials. In *Proc. AAECC-10* (1993), vol. 673 of *Lect. Notes Comput. Sci.*, Springer Verlag, pp. 162–169.

[12] Imamura, K., and Yoshida, W. A simple derivation of the Berlekamp-Massey algorithm and some applications. *IEEE Trans. Inf. Theory* IT-*33* (1987), 146–150.

[13] Kaltofen, E., Lakshman Y. N., and Wiley, J. M. Modular rational sparse multivariate polynomial interpolation. In *Proc. 1990 ISSAC* (1990), S. Watanabe and M. Nagata, Eds., ACM Press, pp. 135–139. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/90/KLW90.pdf.

[14] Kaltofen, E., and Lee, W. Early termination in sparse interpolation algorithms. *J. Symbolic Comput. 36*, 3–4 (2003), 365–400. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/03/KL03.pdf.

[15] Kaltofen, E., Yang, Z., and Zhi, L. On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In *Proc. 2007 SNC* (2007), J. Verschelde and S. M. Watt, Eds., ACM, pp. 11–17. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/07/KYZ07.pdf.

[16] Kaltofen, E., and Yuhasz, G. On the matrix Berlekamp-Massey algorithm, Dec. 2006. Manuscript, 29 pages. Submitted.

[17] Kaltofen, E. L. Fifteen years after DSC and WLSS2 What parallel computations I do today. In *Proc. 2010 PASCO* (July 2010), M. Moreno Maza and J.-L. Roch, Eds., ACM, pp. 10–17. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/10/Ka10_pasco.pdf.

[18] Kaltofen, E. L., Lee, W., and Yang, Z. Fast estimates of Hankel matrix condition numbers and numeric sparse interpolation. In *Proc. 2011 SNC* (June 2011), M. Moreno Maza, Ed., ACM, pp. 130–136. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/11/KLY11.pdf.

[19] Kaltofen, E. L., and Nehring, M. Supersparse black box rational function interpolation. In Leykin [21], pp. 177–185. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/11/KaNe11.pdf.

[20] Khonji, M., Pernet, C., Roch, J.-L., Roche, T., and Stalinsky, T. Output-sensitive decoding for redundant residue systems. In *Proc. 2010 ISSAC* (July 2010), pp. 265–272.

[21] Leykin, A., Ed. *Proc. 2011 ISSAC* (2011), ACM.

[22] Massey, J., and Schaub, T. Linear complexity in coding theory. In *Coding Theory and Applications*, G. Cohen and P. Godlewski, Eds., vol. 311 of *Lecture Notes in Computer Science*. Springer Verlag, 1988, pp. 19–32.

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

[24] Moon, T. K. *Error correction coding: mathematical methods and algorithms.* Wiley-Interscience, 2005.

[25] Prony, R. 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* (Floréal et Prairial III (1795)), 24–76.

[26] Reed, I. S., and Solomon, G. S. Polynomial codes over certain finite fields. *J. SIAM 8*, 2 (June 1960), 300–304.

[27] Rump, S. M. Structured perturbations part I: Normwise distances. *SIAM J. Matr. Anal. Appl. 25*, 1 (2003), 1–30.

[28] Saraf, S., and Yekhanin, S. Noisy interpolation of sparse polynomials, and applications. In *IEEE Conference on Computational Complexity* (2011), IEEE Computer Society, pp. 86–92.

[29] Yuhasz, G. *Berlekamp/Massey Algorithms for Linearly Generated Matrix Sequences.* PhD thesis, North Carolina State Univ., Raleigh, North Carolina, May 2009.

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

Note added on August 29, 2012:
Some typographical errors have been corrected:

1. Section 1, first paragraph (changed "interpolations algorithms" to "interpolation algorithms")
2. Section 1, seventh paragraph (changed "Ben-Or/Tiwari's" to "Ben-Or's/Tiwari's")
3. Section 4, second-to-last paragraph (changed "are error-free" to "is error-free")
4. Section 5, first sentence (changed "in a t of" to "in a sequence of")
5. Proof of Lemma 5, first paragraph (changed "with bands given" to "generated")
6. Section 7, first paragraph (changed "polynomials residues" to "polynomial residues")