Sparse Polynomial Hermite Interpolation

Erich L. Kaltofen
Dept. of Mathematics, NCSU and Dept. of Computer Science, Duke University
Raleigh, Durham, North Carolina, USA

ABSTRACT
We present Hermite polynomial interpolation algorithms that for a sparse univariate polynomial \( f \) with coefficients from a field compute the polynomial from fewer points than the classical algorithms. If the interpolating polynomial \( f \) has \( t \) terms, our algorithms, require argument/value triples \( (\omega_i, f(\omega_i), f'(\omega_i)) \) for \( i = 0, \ldots, t + \lceil (t+1)/2 \rceil - 1 \), where \( \omega \) is randomly sampled and the probability of a correct output is determined from a degree bound for \( f \). With \( f' \) we denote the derivative of \( f \). Our algorithms generalize to multivariate polynomials, higher derivatives and sparsity with respect to Chebyshev polynomial bases. We have algorithms that can correct errors in the points by oversampling at a limited number of good values. If an upper bound \( B \geq t \) for the number of terms is given, our algorithms use a randomly selected \( \omega \) and, with high probability, \( \lfloor t/2 \rfloor + B \) triples, but then never return an incorrect output.

The algorithms are based on Prony’s sparse interpolation algorithm. While Prony’s algorithm and its variants use fewer values, namely, \( 2t + 1 \) and \( t + B \) values \( f(\omega_i) \), respectively, they need more arguments \( \omega_i \). The situation mirrors that in algebraic error correcting codes, where the Reed-Solomon code requires fewer values than the multiplicity code, which is based on Hermite interpolation, but the Reed-Solomon code requires more distinct arguments. Our sparse Hermite interpolation algorithms can interpolate polynomials over finite fields and over the complex numbers, and from our sparse Hermite interpolation algorithms can interpolate polynomials over finite fields and over the complex numbers, and from high probability, \( \lfloor t/2 \rfloor + B \) triples, but then never return an incorrect output.

The algorithms are based on Prony’s sparse interpolation algorithm. While Prony’s algorithm and its variants use fewer values, namely, \( 2t + 1 \) and \( t + B \) values \( f(\omega_i) \), respectively, they need more arguments \( \omega_i \). The situation mirrors that in algebraic error correcting codes, where the Reed-Solomon code requires fewer values than the multiplicity code, which is based on Hermite interpolation, but the Reed-Solomon code requires more distinct arguments. Our sparse Hermite interpolation algorithms can interpolate polynomials over finite fields and over the complex numbers, and from floating point data. Our Prony-based approach does not encounter the Birkhoff phenomenon of Hermite interpolation, when a gap in the derivative values causes multiple interpolants. We can interpolate from \( t + 1 \) values of \( f \) and \( 2t - 1 \) values of \( f' \).

1 INTRODUCTION
Let \( f(x) \) be a sparse Laurent polynomial in standard basis of powers of the variable over a field \( K \),

\[
f(x) = \sum_{j=0}^{t} c_j x^{n_j} \in K[x, \frac{1}{x}], \forall j: c_j \neq 0, e_j \in \mathbb{Z}, e_1 > e_2 > \cdots > e_t.
\]

We allow negative \( e_j \) from the start so that our algorithms can be transferred to orthogonal polynomial bases [2; 12]; see Section 5.2. The derivative of \( f(x) \) in \( x \) is defined \( f'(x) = \sum_{j=1}^{\lfloor t/2 \rfloor} c_j j x^{n_j-1} \), which is a polynomial of sparsity \( t \) or \( t - 1 \), the latter if there are \( e_j = 0 \), possibly taken modulo the positive characteristic of the coefficient field \( K \). The classical Hermite interpolation algorithm can recover \( f \) from \( \max(e_1, 0) - \min(0, e_t) + 1 = (e_1 + 1) + \cdots + (e_t + 1) \) values \( a_{i,j} = f^{(j)}(\xi_i) \) for \( 1 \leq i \leq n \) and \( 0 \leq j \leq t, \) where

\[
\deg(f) \leq \max(e_1, 0) - \min(0, e_t) \text{ and } f^{(j)}(x) = d^j f(x)/d x^j \text{ is the } j- \text{th derivative of } f.
\]

Here \( \xi_i \in K \) are distinct arguments. The classical Prony sparse interpolation algorithm can recover \( f \) from \( 2t \) values \( \xi_i = \omega_i \) for \( \omega_i \in K \) and \( i = 0, 1, \ldots, i_0 + 2t - 1 \), provided the term values \( \omega^{e_j} \) are distinct for \( 1 \leq j \leq t \) and \( e_j \) can be computed from \( \omega^{e_j} \). If \( e_1, e_t \) are not known before interpolation, bounds can be used; if the sparsity \( t \) is not known, either an upper bound \( B \) can be used and the number of values becomes \( t + B \), or \( t \) can be computed by a Monte-Carlo randomized algorithm from \( 2t + 1 \) values [16].

We shall transfer the Prony method to the Hermite interpolation problem. If there are at most first derivatives, that is \( e_t \leq 1 \), it is Prony’s idea that the term locator polynomial

\[
\Lambda(z) = z^{x_t} + \lambda_{t-1} z^{x_{t-1}} + \cdots + \lambda_0 (z - \omega^{e_t}) \cdots (z - \omega^{e_1})
\]

minimally, if squarefree, linearly generates \( a_{i,j} \equiv f(\omega^{e_t} + i) \) for an arbitrary integer starting index \( i_0 \) and \( 0 = i_0, \ldots, i_0 + 2t - 1 \). The polynomial \( \Lambda(z) \) also generates, not always minimally, \( \bar{a}_{i,j} \equiv f(\omega^{e_t} + i) \) for \( i = 0, 1, \ldots, 2t - 1 \), because \( f(x) \) has a termset \( \subseteq \{ x^{e_j} \}_{1 \leq j \leq t} \). Therefore, the vector sequence \( \left( \begin{array} \bar{a}_{i,j} \\
\end{array} \right)_{i \geq 0} \) is minimally generated by the term locator polynomial \( \Lambda(z) \), which can be computed by the matrix Berlekamp-Massey algorithm [22] using 1 candidate right generator and 2 candidate right auxiliary polynomials. One could also use a sequence of 2-dimensional row vectors and compute \( \Lambda(z) \) as the highest degree invariant factor of the \( 2 \times 2 \) right matrix generator polynomial, with 1 auxiliary right polynomial [22]. Here we pursue the goal adopted from Hermite interpolation with error correction (multiplicity code decoding) [18, 19] and determine the minimum number of values and derivative values required in order to always recover the term locator polynomial \( \Lambda(z) \).

If for all arguments \( \omega \) one has both \( a_t \) and \( \bar{a}_t \), experiments quickly show that \( t + \lfloor t/2 \rfloor \) arguments \( \omega^{e_t+1} \) are required. In general for \( r \) with \( 1 \leq r \leq t - 1 \), if \( f(\omega^{e_t+1}) \) is input for \( 0 \leq i \leq t + r - 1 \) and \( f'(\omega^{e_t+1}) \) is input for \( 0 \leq i \leq 2t - r - 1 \) one can always recover \( f \), for fields \( K \) of characteristic 0 or sufficiently large or randomly.
selected positive characteristic.

We now list the major differences to Prony sparse polynomial interpolation, for the case that highest order of derivatives is 1; higher derivatives are discussed later.

1. A total of 3t values and values of derivatives is minimally required; the Prony algorithm requires 2t. We assume for now that the sparsity is input. Such an increase in the required number of values is familiar from multiplicity code decoding, but for error correction the increase is a consequence of the Birkhoff problem in Hermite interpolation, when there are gaps in the inputs of the sequence of values \( f(\xi), f'(\xi), f''(\xi), \ldots \) at a single \( \xi \) (cf. [18, Example 1]). Here there can be gaps: in fact, for \( r = 1 \) one has \( t+1 \) values and \( 2t-1 \) values at derivatives and recovers \( f \).

2. In contrast to Prony’s algorithm, the minimal count of values of values requires randomization in the selection of \( \omega \). We have examples where for a given \( \omega \in \mathbb{K} = \mathbb{C} \) for which the term locator polynomial \( \Lambda(z) \) in (2) is squarefree, 3t values are insufficient to recover \( f \); see Example 3.3 below.

3. The number of arguments at which the algorithm computes values and/or values at derivatives can be less than the 2t of Prony’s algorithm, which is again familiar from the multiplicity code Hermite interpolation with error correction \([18, 19]\).

4. If first derivatives (gradients) can be obtained at the arguments, \( f \) may be recovered; the Prony algorithm requires \( 2t \) values at \( t \) small points. In general, if first derivatives can be obtained, one requires even fewer points: \( t + \lceil t/f \rceil \) arguments \( \xi_j \) for \( t \)-th order derivatives, and a total number of values of \( (t+2)t/2 \); see Section 4.

4. For fields \( K \) of characteristic 0, and in certain situations for fields \( K \) of characteristic \( p > 0 \) (see Step 4 below), we can compute the term degrees \( e_j \) without taking logarithms of the roots of the term locator polynomial. In particular, we compute the term degrees even when evaluating at powers of a root of unity whose order is below the degree of \( f \), provided that the terms values remain distinct. The idea was used before for fast sparse interpolation algorithms of polynomial products [3] and of polynomials given by straight line programs [10, 11] where the asymptotic complexity was optimized, and the number of samples could be increased by a constant factor.

### 1.1 Relation to Previous Work

Sparse approximation by possibly multivariate functions of data that includes gradient information is a highly investigated subject, from local piecewise spline interpolation to global compressive sparsification by \( \ell^1 \)-norm optimization [1]. The Prony algorithm, which was originally designed for sums of exponentials [26] and used for sparse polynomials over finite fields to decode the 1959 BCH digital error correction code, is suitable for floating point data \([6, 7]\). Our algorithms for sparse polynomial Hermite interpolation compute an exact fit, that is, can be used for scalars from a finite field like the multiplicity error correction code \([19, 23]\). We note that multiplicity code decoding has already been shown to be suitable for floating point data \([14]\). As in Hermite interpolation with error correction, our algorithms necessarily require more data than the Prony algorithm, but at fewer points. The reduction in the number of measurement points remains effective for higher derivatives (Section 4), in the multivariate setting (Section 5.1), when interpolation sparse polynomials in orthogonal bases (Section 5.2) and when correcting errors (Section 5.3). A special algorithm interpolates \( f \) from \( f(1), f'(1), \ldots, f^{(2t-1)}(1) \) \([25]\).

### 2 SPARSE HERMITE INTERPOLATION

For the \( t \)-sparse Laurent polynomial \( f(x) \) and all integers \( i \), the sequences \( (a_i)_{i \geq 0} \) with \( a_i = f(\omega^i x) \) and \( (\tilde{a}_i)_{i \geq 0} \) with \( \tilde{a}_i = \omega^{i+1} f'(\omega^i x) \) for \( \omega \in K \) are linearly generated by \( \Lambda(z) \). Therefore, we have for all \( r \) with \( 1 \leq r \leq t-1 \)

\[
H_{t,r} = \begin{pmatrix}
\lambda_0 \\
\lambda_1 \\
\vdots \\
\lambda_{r-1}
\end{pmatrix}
= h_{t,r} \cdot
\begin{pmatrix}
0 & 0 & \ldots & 0 \\
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_{r-1}
\end{pmatrix}
\quad \text{where}
\begin{pmatrix}
-h_{t,r}^{-1} \\
0 \\
1 \\
\vdots \\
0
\end{pmatrix}
= \begin{pmatrix}
-\omega^{-r-1} \\
0 \\
1 \\
\vdots \\
0
\end{pmatrix}
\quad \text{for all } \omega \in K. \tag{3}
\]

The condition under which the term locator polynomial \( \Lambda \) is the minimal generator can be characterized for each sequence separately. For the sequence \((a_i)_{i \geq 0}\) a necessary and sufficient condition is that the \( \omega^i \) are distinct elements in \( K \) (see, for instance, \([2,\text{ Lemma 4.2}]\)). We will prove in Section 3 that for randomly sampled \( \omega \) the matrix \( H_{t,r} \) is non-singular with high probability, for all \( r \) and all fields \( K \) of characteristic 0; for fields \( K \) of characteristic \( p \) > 0, the statement remains valid if the prime \( p \) is sufficiently large (see Theorem 3.2) or if \( p \) can be randomly chosen and \( f \) projected modulo \( p \). However, \( \det(H_{t,r}) \) is 0 is possible for fields \( K \) of characteristic 0 even if all \( \omega^i \) are distinct: see Example 3.3 below.

Our sparse Hermite interpolation algorithm randomly selects \( \omega \), checks the matrix \( H_{t,r} \) for non-singularity, and then proceeds similar to Prony’s algorithm. Since the term exponents are computed differently, we explicitly state the algorithm. Because the algorithm uses \( f \) as a black box, it is useful to flag a false sparsity \( t \) or erroneous \( a_i \) or \( \tilde{a}_i \) if that can be diagnosed.

### 2.1 Algorithm for Explicit Sparsity Input

**Input:** Let \( f(x) = \sum_{j=1}^t c_j x^j \in \mathbb{K}[x, x^{-1}] \) with \( t \geq 2 \), \( c_j \in K, c_j \neq 0 \), \( e_j \in \mathbb{Z} \) for all \( j \), \( e_1 > \cdots > e_t \). The field of scalars \( K \) is of characteristic 0 or \( p > 0 \). The inputs are:

- The number \( t \) of terms in \( f \) and \( an \) \( r \) with \( 1 \leq r \leq t-1 \);
- An element \( \omega \in K, \omega \neq 0, \omega \neq 1 \), and \( \omega^i \neq 0 \in \mathbb{Z} \) and values \( a_i = f(\omega^i x) \) for \( 0 \leq i \leq t-r+1 \), \( \tilde{a}_i = \omega^{i+1} f'(\omega^i x) \) for \( 0 \leq i \leq 2t-r-1 \) such that \( \det(H_{t,r}) \neq 0 \) where \( H_{t,r} \) is defined in (3).

**Output:**

- Case 1(a): \( K \) has characteristic 0 or \( p > 0 \) and \( \omega^r \neq 1 \) for all \( v \geq 1 \): \( c_1, \ldots, c_r \) and \( e_1, \ldots, e_r \).
- Case 2: \( K \) has characteristic \( p > 0 \) and \( \exists \theta \geq 2: \omega^\theta = 1 \):

  - \( c_1, \ldots, c_r \) and \( e_1, \ldots, e_r \) such that for \( f(x) = \sum_{j=1}^t c_j x^{e_j} \) we have \( a_i(\omega^{i+1})x, 0 \leq i \leq t+r-1 \) and \( \tilde{a}_i(\omega^{i+1})x, 0 \leq i \leq 2t-r-1 \).

1. **Solve the \( t \times t \) linear system (3) in \( \tilde{a}_i \).** See \([22, \text{Table 1, Column 2}]\) for fast algorithms. Because \( H_{t,r} \) on input is non-singular, \( \Lambda(z) \) = \( z^r + \sum_{j=1}^t \lambda_j z^j = \prod_{j=1}^t (z - \omega^j) \), and the roots \( \omega^j \) are distinct.

2. **Factor \( \Lambda(z) \) = \( \prod_{j=1}^t (z - \omega^j) \in K[z] \).** If \( f(x) \) does not squarefree factor into linear factors, return “the \( a_i, \tilde{a}_i \) do not interpolate a \( t \)-sparse polynomial.”
3. Solve the two transposed Vandermonde linear systems

\[
\begin{align*}
\begin{bmatrix}
 b_1 & \cdots & b_t \\
 b_1^* & \cdots & b_t^*
\end{bmatrix}
\begin{bmatrix}
 \tilde{c}_1 \\
 \vdots \\
 \tilde{c}_t
\end{bmatrix}
&= \begin{bmatrix}
 \tilde{a}_1 \\
 \vdots \\
 \tilde{a}_t
\end{bmatrix}, \\
\begin{bmatrix}
 b_1 & \cdots & b_t \\
 b_1^{-1} & \cdots & b_t^{-1}
\end{bmatrix}
\begin{bmatrix}
 \tilde{c}_1 \\
 \vdots \\
 \tilde{c}_t
\end{bmatrix}
&= \begin{bmatrix}
 \tilde{a}_1 \bar{a}_1 \\
 \vdots \\
 \tilde{a}_t \bar{a}_t
\end{bmatrix}.
\end{align*}
\]

See [15, Section 5] for a fast algorithm.

4. Compute the term exponents \( \hat{t}_j = \tilde{t}_j / c_j \) for \( 1 \leq j \leq t \).

4(a) Case \( K \) has characteristic 0:

If there exists an index \( j \) with \( \hat{t}_j \notin \mathbb{Z} \) or \( \omega^{\hat{t}_j} \neq b_j \), return the \( \tilde{a}_i, \hat{a}_i \) do not interpolate a \( t \)-sparse polynomial.

4(b) Case \( K \) has characteristic \( p > 0 \): for proper inputs, we have \( \hat{t}_j \equiv e_j \pmod{p} \). Therefore, if \( \hat{\omega}^{\hat{t}_j} \neq b_j \) or \( \omega^{\hat{t}_j} - p \equiv b_j \), then \( \hat{t}_j \) can be computed from \( \tilde{t}_j \).

For \( j = 1, \ldots, t \) do verify \( \hat{t}_j \) or \( \tilde{t}_j - p \) by checking \( \omega^{\hat{t}_j} = b_j \) or \( \hat{\omega}^{\hat{t}_j} = b_j \); if success in the latter, \( \tilde{t}_j \leftarrow \hat{t}_j - p \). Note that \( \omega^{\hat{t}_j} = \hat{\omega}^{\hat{t}_j} \), because \( \omega^p = 1 \).

If all checks succeed, return \( \tilde{c}_1, \ldots, \tilde{c}_t \) and \( \hat{c}_1, \ldots, \hat{c}_t \), sorted in descending term degrees.

Note that even if bounds for \( e_1, e_2 \) are known, the checks verify the \( a_1, \hat{a}_1 \) by verifying the Prony generator \( \Lambda \).

At this point, for all \( j \in J \subseteq \{1, \ldots, t\}, J \neq \emptyset \), \( e_j \) is too large to be directly determined from \( t_j \pmod{p} \). For those \( j \) we require an integer logarithm algorithm, as is required in the original Prony algorithm. We have two subcases.

4(b)i. Subcase \( \hat{\omega}^{\hat{t}_j} = 1 \) for \( \theta \geq 2 \): here \( \mathbb{Z}_p(\hat{\omega}) \) is a finite subfield of \( K \), the latter of which could be an infinite function field. Furthermore, \( \theta \) divides \( [\mathbb{Z}_p(\hat{\omega})] = p^\nu - 1 \), where \( \nu \geq 1 \). For all \( j \in J \) compute \( \hat{\tilde{t}}_j \in \mathbb{Z}_p(\hat{\omega}) \) such that \( \hat{\omega}^{\hat{t}_j} = b_j \) and \( \hat{t}_j \equiv \hat{t}_j \pmod{p} \). One can Chinese remainder the index (discrete logarithm) of \( b_j \) with base \( \omega \) and \( \hat{t}_j \) using the relatively prime moduli \( \theta \) and \( p \). If \( \hat{\tilde{t}}_j \) exist for all \( j \in J \), return \( \tilde{c}_1, \ldots, \tilde{c}_t \) and \( \hat{c}_1, \ldots, \hat{c}_t \), sorted in descending term degrees.

4(b)ii. Subcase \( \omega \) has infinite multiplicative order. For all \( j \in J \) compute \( \hat{\tilde{t}}_j \in \mathbb{Z} \) such that \( \omega^{\hat{t}_j} = b_j \) and verify that \( \hat{t}_j \equiv \hat{t}_j \pmod{p} \). If \( \hat{\tilde{t}}_j \) are computed for all \( j \in J \), return \( \tilde{c}_1, \ldots, \tilde{c}_t \) and \( \hat{c}_1, \ldots, \hat{c}_t \), sorted in descending term degrees. For example, if \( K = \mathbb{Z}_p(\omega) \) is a rational function field, \( \hat{t}_j \) can be determined from the degrees of \( b_j(\omega) \) and \( \omega(\omega) \).

4(b) concluded. At this point, either Step 4(b)i or Step 4(b)ii failed to compute a term degree for a \( b_j \). Return "the \( \tilde{a}_i, \hat{a}_i \) do not interpolate a \( t \)-sparse polynomial."
cannot be linearly generated by the previous columns, implying that the degree of the generator would have to be \( \geq M \geq B + 1 \). The cases where \( t \equiv 1 \pmod{2} \) and/or when the discrepancy is at \( \hat{a} \leq k \) are done similarly.

We briefly consider when \( H_{t}[2/3] \) is singular, that is, the randomly selected \( \omega \) is "unlucky," for instance, an \( \omega \) in Example 3.3 or an \( \omega \) with \( \omega^{p} = \omega^{v} \) for \( p \neq v \). For a \( k \leq t \) it has computed a linear combination \( \lambda \) of the first \( k \) columns that gives the \( (k + 1) \)'st column at the first \( 2B - k \) (even) or \( 2B - k - 1 \) (odd) rows of \( H_{t}[2/3] \), using \( 2B + k \) values \( a_{i}, \hat{a}_{i} \). If the top left \( k \times k \) matrix is singular, one has verified that \( \omega \) as unlucky, but one still may recover \( f \) if \( B \geq t \) for the column relation \( \lambda \), Algorithm 2.1, Steps 2-4 are executed. If the algorithm fails to compute an interpolant, one has verified \( k < t \). An example is \( B = t \) and \( \omega \) as in Example 3.3: \( k \leq t - 1 \) and the \( (k + 1) \)'s column in \( H_{t}[2/3] \) is linearly dependent on the first \( k \) columns, but the \( 2B + k \leq 3t - 1 \) values do not determine the term locator polynomial. However, Algorithm 2.1 may compute a \( k \)-sparse polynomial \( h(x) \) that fits the \( a_{i}, \hat{a}_{i} \) for \( i \leq [k/2] + B \), for example, if \( \omega^{p} = \omega^{v} = \omega^{x persistence} \) for distinct \( p, v, e \) (cf. [2, Lemma 4.2]) and \( f(x) = c_{p}x^{p} + c_{v}x^{v} + c_{e}x^{e}, h(x) = (c_{p} + c_{v} + c_{e})x^{p} \) with \( c_{p}c_{v}c_{e} \neq 0. \) The computer interpolant \( h(x) \) fits the \( a_{i} = f(x_{i}^{(kx+1)}) = h(x_{i}^{(kx+1)}), \hat{a}_{i} = h(x_{i}^{(kx+1)}) \) all \( i \geq k/2 + B \) to \( k \leq t \) and because there was a discrepancy with the values of \( f, f' \), the next linear generator would have degree \( B \) by a matrix factorization similar to (9) after permuting rows so that the top left \( k \times k \) matrix is non-singular, and could not be the term locator polynomial for \( f \). Therefore, unlike the early termination strategy of Item 1 (cf. [12, example after the proof of Theorem 4.3]), from a bound \( B \geq t \) no interpolant is returned that does not fit the infinite sequence of evaluations of \( f \).

3 PROBABILISTIC ANALYSIS

We prove that for random \( \omega \) the matrix \( H_{B,r} \) in (3) is non-singular with high probability. The exact statement is given in Theorem 3.2 below. The method follows that of the early termination proof for the Prony algorithm in [16]. We shall first prove that for the value \( \omega = x \) and evaluations \( a_{i}(x) = f(x^{(kx+1)}) \) and \( \hat{a}_{i}(x) = x^{(kx+1)}f'(x) \) for \( i = 0, 1, \ldots \) the corresponding symbolic matrix \( H_{B,r}(x) \) is \( K[1, \frac{1}{2}] \) is non-singular, at least for a field \( K \) of characteristic \( 0 \). We have the following matrix factorization:

\[
H_{B,r} \overset{\text{def}}{=} \begin{bmatrix}
\alpha_{0} & \alpha_{1} & \ldots & \alpha_{t-1} \\
\alpha_{t-1} & \alpha_{t} & \ldots & \alpha_{t+1} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_{t-1} & \alpha_{t} & \ldots & \alpha_{t-n}
\end{bmatrix}
\]

where \( W_{r} \) is defined in (11) below for \( k = t \) and \( \beta_{j} = x^{ej} \) are the Laurent terms in \( f \). The last 2 matrix factors in (10) are non-singular because \( \beta_{0} = x^{f_{0}} \neq \beta_{i} = x^{f_{i}} \) for \( 1 \leq i < \mu \leq t \) and the last factor is a transposed Vandermonde matrix. We prove that the first factor, \( W_{k,r} \) is non-singular by virtue that the highest degree term has a non-zero coefficient. The proof is by induction on the dimension.

**Theorem 3.1.** Let \( k \geq 1, 0 \leq r \leq k \) and let

\[
W_{k,r} \overset{\text{def}}{=} \begin{bmatrix}
1 & \beta_{1} & \ldots & \beta_{t-1} \\
\beta_{1}^{r} & \beta_{1}^{r} & \ldots & \beta_{t-1}^{r} \\
\vdots & \vdots & \ddots & \vdots \\
\beta_{n-1}^{r} & \beta_{n-1}^{r} & \ldots & \beta_{n-1}^{r}
\end{bmatrix}
\]  

\( r \text{ rows} \) \( k \text{ rows} \)

where \( \beta_{j} = x^{ej} \) are Laurent terms with \( e_{j} \in \mathbb{Z} \) for all \( 1 \leq j \leq k \) with \( e_{1} < e_{2} \cdots \leq e_{k} \) and \( e_{j} \neq 0 \) for \( r < k/2 \) and \( 1 \leq j \leq k - 2r \). Then the leading monomial of \( \det(W_{k,r}) \) in the variable \( x \) is:

\[
\sigma(k, r) = \begin{cases}
\max(m(k-r-1), (e_{k-2r-1} - e_{k-2r})) & \text{for } r \leq \frac{k}{2} \\
\min(m(r-k), (e_{k-2r-1} - e_{k-2r})) & \text{for } \frac{k}{2} < r \leq k,
\end{cases}
\]

where the exponents are

\[
\eta(k, r, j) = \max(\frac{k - j - 1}{2}, \max(r, k - r) - j). 
\]

We have \( \eta(k, r, j) \geq \eta(k, r, j+1) \) for all \( j \) with \( 1 \leq j \leq k - 1 \). Specifically, for \( r = [k/2] \) the leading monomial is:

\[
(-1)^{[k/4]} e_{1} - e_{2} e_{3} - e_{4} \cdots \left( e_{k-1} - e_{k} \right) \times \left( \beta_{1}^{2} \beta_{2}^{2} \cdots \beta_{k-3}^{2} \beta_{k-2} \right)
\]

\( k \text{ is even,} \)

\[
(-1)^{[k/4]+1} e_{1} - e_{2} e_{3} - e_{4} \cdots \left( e_{k-1} - e_{k} \right) \times \left( \beta_{1}^{4} \beta_{2}^{4} \cdots \beta_{k-3}^{4} \beta_{k-2} \right)
\]

\( k \text{ is odd.} \)

For \( k \equiv 1 \pmod{2} \) and \( r = [k/2] \) the leading monomial is:

\[
(-1)^{[k/4]-1} e_{1} - e_{2} e_{3} - e_{4} \cdots \left( e_{k-1} - e_{k} \right) \times \left( \beta_{1}^{2k-1} \beta_{2}^{2k-2} \cdots \beta_{k-3}^{2k-2} \beta_{k-2} \right)
\]

**Proof.** The formulas are proven by induction with minor expansion along the first column of \( W_{k,r} \) in (11), as if arising by themselves.2 See Section 7. \( \square \)

The condition that \( e_{j} \neq 0 \) for \( r < k/2 \) and \( j \leq k - 2r \) in Theorem 3.1 is required so that the factor \( \prod_{r=1}^{[k/2]} r \neq 0 \) in (12). If there is an index \( j' < k \) with \( e_{j'} = 0 \), which over \( \mathbb{Z} \) implies \( e_{k} = 0 \), that is, there are terms with negative degree, the corresponding \( j' \)th column in \( W_{k,r} \) in (11) contains zeros, and the leading monomial changes. Note that for \( e_{k} = 0 \) the leading monomial (12) is valid for \( r \geq 1 \) because the factor \( e_{k} \) only appears for \( r = 0 \), in which case \( \det(W_{k,0}) = 0 \). The case in which an \( e_{j} = 0 \) for \( j' < k \) can be reduced to Theorem 3.1 as follows:

\[\text{See the proof of Theorem 4.1 for a full explanation.}\]
Corollary 3.1. Let \( k \geq 1 \) and

\[
W_{k,r}^{[j]} \equiv \begin{bmatrix}
\beta_1 & \ldots & \beta_{r-1} & 1 & \beta_{r+1} & \ldots & \beta_k \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\beta_{r-1}^{k-1} & \ldots & \beta_{r-1} & 1 & \beta_{r+1}^{k-1} & \ldots & \beta_k^{k-1} \\
\varepsilon_1 & \ldots & \varepsilon_{r-1} & 0 & \varepsilon_{r+1} & \ldots & \varepsilon_k \\
\varepsilon_1 \beta_1 & \ldots & \varepsilon_{r-1} \beta_{r-1} & 0 & \varepsilon_{r+1} \beta_{r+1} & \ldots & \varepsilon_k \beta_k \\
\vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
\varepsilon_1 \beta_1^{k-1} & \ldots & \varepsilon_{r-1} \beta_{r-1}^{k-1} & 0 & \varepsilon_{r+1} \beta_{r+1}^{k-1} & \ldots & \varepsilon_k \beta_k^{k-1}
\end{bmatrix}
\]

where \( \beta_j = x^{e_j} \) are Laurent terms with \( e_j \in \mathbb{Z} \) for all \( 1 \leq j \leq k \) with \( e_1 > \cdots > e_j > \cdots > e_k \); note that \( \beta_j = 1 \).

Let \( \tau(k, r, j, s) \) be the permutation on \( \{1, \ldots, k\} \) with

\[
\tau(k, r, j, s) = \begin{cases} 
 j & \text{for } j \leq j^* - 1 \text{ or } j \geq k - 2r + 2, \\
 j^* & \text{for } j = k - 2r + 1, \\
 j + 1 & \text{for } j^* < j \leq k - 2r.
\end{cases}
\]

Then the leading monomial of \( \det(W_{k,r}^{[j]}) \) in the variable \( x \) is:

\[
(-1)^{k-2r-j^*+1} \sigma(k, r) \left( \prod_{\nu=1}^{k-2r-j} t_{\tau(k,r,j^*,\nu)} \right) \left( \prod_{\rho=0}^{\min(r,k-r)-1} \right)
\]

\[
e_{\tau(k,r,j^*,k-2\nu-1)} - e_{\tau(k,r,j^*,2k-\nu)} \prod_{j=1}^{k} \beta_j^{k-1} e_j \tag{20}
\]

where \( \beta_j = 1 \) and \( e_j = 0 \), and where \( \sigma \) is defined in (13) and \( \eta \) in (14).

Proof. All terms in \( \det(W_{k,r}) \) when keeping \( e_j \), \( j \), \( \beta_j \) as variables factor like the lead term in (12) using a different variable ordering for the \( \beta_j \). Substituting the variable \( \beta_j = x^{e_j} \) (k,r) and the variable \( e_j \) with the numeric value \( e_j(k,r,j) \) all terms corresponding to a variable orderings for the \( \beta_j \) which place \( \beta_j \) before the \( (k-2r+1) \)-st position and which have would on evaluation a higher degree in \( x \) have a zero coefficient because of the factor \( \prod_{\nu=0}^{k-2r-j} e_j \) in (12).

Note that \( W_{k,r} \) or \( W_{k,r}^{[j]} \) can not be non-singular even if there is a pair \( e_p = e_q \), for instance in (16) if \( e_1 = e_2 \). The matrix \( H_{k,r} \) is then singular by virtue that the Vandermonde factor in (10) is singular.

Theorem 3.2. Let \( f, t, r, i_0 \) be as in the input specifications of Algorithm 2.1. Let \( K \) be a field of characteristic 0 or \( p \geq \deg(f) + 1 \) and \( \max(e_1, 0) = \min(e_0, 0) + 1 \). Suppose the element \( \omega \in \mathbb{K} \) and uniformly selected from a finite set \( S \subset \mathbb{K} \) with \( 0 \notin S \) and \( 1 \notin S \) of cardinality \( |S| \). Then the probability that \( H_{k,r} \) in (3) and Step 1 is non-singular is \( \geq 1 - (t-1)^2(e_1 - e_1)/|S| \).

Proof. By the matrix factorization (10) and by Theorem 3.1 and Corollary 3.1 the det\((H_{k,r}) \neq 0 \) both in characteristic 0 and \( p \geq \deg(f) + 1 \), the latter because the coefficients of the leading monomials of \( W_{k,r} \) (12) and \( W_{k,r}^{[j]} \) (20) are non-zero modulo \( p \). It remains to estimate the degree of \( x \) in the leading and trailing monomial of both det\((W_{k,r}) \) and the determinant of the transposed Vandermonde factor (10). Note that the diagonal factor in (10) is non-singular for all \( \omega \in S \). The degree of the leading monomial in det\((W_{k,r}) \) is by (12,20)

\[
\leq \sum_{j=1}^{t-2} e_j \eta(t, r, j, r) \leq (t-2)(e_1 - j + 1)(t - j - 1)
\]

\[
= \frac{1}{6}(t-1)(t-2)(3e_1 + t - 3). \tag{21}
\]

By reversing the monomial order \( \beta_1 \leq \cdots \leq \beta_k \) we have \( (t-1)^{x} \) the degree of trailing monomial in det\((W_{k,r}) \) to be bounded from below by \( (t-1)(t-2)(3e_1 - t + 2)/6 \). For the determinant of the right transposed Vandermonde factor the corresponding bounds are \( \pm(t-1)(3e_j - t + 2)/6 \) for \( j = 1, \ldots, t - 2 \). Summing the two differences of the upper and lower bounds yields a degree bound \( (t-1)^2(e_1 - e_t) \). Sharper bounds can be derived for a given \( r \) from the first estimate in (21).

Example 3.3. When computing the determinant of \( W_{k,r} \) in a polynomial in \( x \) one obtains values \( \omega \in \mathbb{K} \) for which \( \omega^{e_0} \neq \omega^{e_0} \) for all \( 1 \leq \mu < \nu \leq t \) but for which det\((H_{k,r}) = 0 \) (where \( H_{k,r} \) is defined in (3)). For example for \( t = 3, r = 1 \) and \( e_2 = e_1 - 1, e_3 = e_1 - 2 \) we have

\[
det(W_{3,1}) = det\left( \begin{bmatrix}
1 & e_1 & e_1 - 1 \\
e_1 e_1 & e_1 e_1 - 1 & e_1 e_1 - 1 \\
e_1 e_1 - 1 & e_1 e_1 - 1 & e_1 e_1 - 1
\end{bmatrix}\right)
\]

\[
= -x^{e_1 - 1}(x - 1)(x - e_1 - 1) \tag{22}
\]

Therefore, for \( \omega = 1 - 1/2, 1 \) we have det\((H_{k,r}) = 0 \) and \( f_2(x) = c_1x^{e_1} + c_1x^{e_1-3} + c_1x^{e_1-6} \) cannot be interpolated from \( a_i = f_2\left(1 - \frac{2}{e_i}\right) \) for \( i = 1, \ldots, 4 \). Here \( \omega \neq 1 \) because one needs \( \omega^{e_1} \neq \omega^{e_1} \).

Remark 3.4. Corollary 3.1 considers \( W_{k,r}^{[j]} \) (18) as a polynomial matrix with integer coefficients. When taking its determinant modulo \( p \) the characteristic \( \rho \geq 2 \) of the coefficient field \( K \) of \( f \) in (1), the coefficient of the term (20) can map to zero. However, a lower degree term will survive if there exists a permutation \( \tau \) on \( \{1, \ldots, k\} \) such that \( \left( \prod_{\nu=0}^{k-2r} e_\tau(\nu) / \prod_{\mu=0}^{\min(r,k-r)-1} e_\tau(\nu-1) - e_\tau(2k-2\nu) \right) \neq 0 \) (mod \( p \)). Note that it is possible that det\((W_{k,r}^{[j]}) \equiv 0 \) (mod \( p \)) for distinct term values \( \omega^{e_0} \neq \omega^{e_0} \) for all \( 1 \leq \mu < \nu \leq t \), for instance, if \( e_1 \equiv e_2 \equiv 0 \) (mod \( p \)) and \( r = 1 \).

The next theorem is used for estimating the probability of success when computing the sparsity \( r \) by the randomized early termination strategy.

Theorem 3.5. Let \( K \) be a field of characteristic 0 or \( p \geq \deg(f) + 1 \) \( = \max(e_1, 0) - \min(e_0, 0) + 1 \). Then \( H_{k,\lfloor k/2 \rfloor} \in K[x]^k \) (10) is
non-singular for \( i_0 \neq 0 \) and for all \( k = 1, 2, \ldots, t \).

**Proof.** The proof follows directly from [16]. The highest order term in \( \mathcal{H}_{k,[k]} \) is for even \( k \), for example, by the Cauchy-Binet argument of [16, Proof of Theorem 4] and by (15)
\[
\begin{align*}
\beta_1^{i_0+k-1} k-1 &- 1 k+2 k-2 - 1 k+2 k-2 - 1 k+4 k/2 - 2 \\
\beta_2^{i_0+3+1} k+4 k+2 &- 1 k+2 k-1 \\
\beta_3^{i_0+4} k+4 k+2 &- 1 k+2 k-1 \\
\beta_4^{i_0+5} k+4 k+2 &- 1 k+2 k-1 \\
\cdots &
\end{align*}
\]
(23)
We note that in [8, Theorem 4.1] the condition \( i_0 \neq 0 \) is removed for the Prony algorithm for fields \( K \) of characteristic \( \neq 2 \), but we do not know if the argument can be generalized to \( \mathcal{H}_{k,[k]} \) for \( k \geq 2 \).

We now give a very rough estimate on the probability that for a random \( \omega \), the matrices \( \mathcal{H}_{k,[k]} \) are non-singular for all \( 1 \leq k \leq t \).

**Corollary 3.2.** Let \( f, t, r, i_0 \) be as in the input specifications of Algorithm 2.1. Let \( K \) be a field of characteristic 0 or \( \rho \geq \deg(f)+1 = \max(e_1, 0) - \min(e_2, 0) + 1 \). Suppose the element \( \omega \) is randomly and uniformly selected from a finite set \( S \subseteq K \) with \( 0 \notin S \) and \( 1 \notin S \) of cardinality \([S]\). Then the probability that \( \mathcal{H}_{k,[k]} \) in (3) is non-singular for all \( k \) with \( 1 \leq k \leq t \) is \( 1 - \frac{1}{(13^3 + t^2)}(1 - 1)(2t - 1) + \frac{1}{(12^3 + t^2)}(t + 1) = \frac{1}{13^3 + t^2} \). Note that the bound is not sharp.

4 HIGHER GENERATIVES

The algorithm generalizes to higher derivatives. We only discuss the case when the highest derivative values are second derivatives, and the derivative values are evenly spaced. We suppose that second derivatives values \( \tilde{a}_i = \omega^{2((i+1)/3)}((i+2)/3) \) for \( i = 0, 1, \ldots \) are also available for interpolation. The linear system (3) in Step 1 now is
\[
H_{t,[t/3]+p_0,[t/3]+p_1} \mathbf{x} = H_{t,[t/3]+p_0,[t/3]+p_1} \mathbf{x}
\]
with \( p_0 = p_1 = 0 \) for \( t \equiv 0 \pmod{3} \), \( p_0 = 1, p_1 = 0 \) for \( t \equiv 1 \pmod{3} \), and \( p_0 = 0, p_1 = 1 \) for \( t \equiv 2 \pmod{3} \), and when the augmented coefficient matrix \( [H_{t,r,n_1} \mid H_{t,n_1}] \) of (24) is (25):
\[
\begin{bmatrix}
a_0 & a_1 & \ldots & a_t-1 \\
\vdots & \vdots & \ddots & \vdots \\
a_r & a_1 & \ldots & a_{r+1} \\
\vdots & \vdots & \ddots & \vdots \\
a_r & a_1 & \ldots & a_{r+1} \\
\vdots & \vdots & \ddots & \vdots \\
a_r & a_1 & \ldots & a_{r+1} \\
\vdots & \vdots & \ddots & \vdots \\
a_{r-n_1} & a_{r-n_1-1} & \ldots & a_{t-n_1-1} \\
\end{bmatrix}
\]
(25)
In total, we need \( 4t \) values at \( t = [t/3] + 1 \) arguments \( \mathbf{i} = \omega^{i/3} \). The coefficient matrix \( H_{t,r,n_1} \) in (25) is with high probability non-singular by Theorem 4.1 below, which corresponds to (15,16) in Theorem 3.1 above.

**Theorem 4.1.** Let \( k \geq 1 \) and let where \( \beta_j = x^{j} \) are Laurent terms with \( e_j \in \mathbb{Z} \) for all \( 1 \leq j \leq k \) with \( e_j > e_j > \cdots > e_k \). Then the leading monomial in the variable \( x \) of \( \det(W_{k,r,n_1}) \in \mathbb{K}_{k,(k/3)+k_0}, \) where \( W_{k,r,n_1} \subseteq \mathbb{Z}[x, \frac{1}{x}]^{k 	imes k} \), \( 0 \leq r \leq k, 0 \leq r \leq r_1 \leq k - r_0 = 0 \) is (26),
\[
\begin{bmatrix}
\beta_1 & \beta_2 & \ldots & \beta_k \\
\vdots & \vdots & \ddots & \vdots \\
\beta_k & \beta_k & \ldots & \beta_k \\
\end{bmatrix}
\]
(26)
is the following if \( k \equiv 0 \pmod{3} \) and \( k_0 = k_1 = 0 ) :
\[
(-1)^{k/3} \left( \prod_{\mu=1}^{k-2} (\epsilon_\mu - \epsilon_{\mu+1}) \left( \prod_{\nu=2}^{k-1} (\epsilon_{t_{\nu+1}} - \epsilon_{t_{\nu+1}}) \right) \right) \times 
\]
(27)
and if \( k \equiv 1 \pmod{3} \) and \( k_0 = 1 \) and \( k_1 = 1 ) :
\[
(+1)^{k-1} \left( \prod_{\mu=2}^{k-2} (\epsilon_\mu - \epsilon_{\mu+1}) \left( \prod_{\nu=3}^{k-1} (\epsilon_{t_{\nu+1}} - \epsilon_{t_{\nu+1}}) \right) \right) \times 
\]
(28)
and if \( k \equiv 2 \pmod{3} \) and \( k_0 = 1 \) and \( k_1 = 1 ) :
\[
(-1)^{k-2} \left( \prod_{\mu=3}^{k-2} (\epsilon_\mu - \epsilon_{\mu+1}) \left( \prod_{\nu=3}^{k-1} (\epsilon_{t_{\nu+1}} - \epsilon_{t_{\nu+1}}) \right) \right) \times 
\]
(29)

**Proof.** As for Theorem 3.1, the proof is by induction on \( k \). We write \( \text{FLM}(k, r_0, r_1; e_1, \ldots, e_k) \) for the formal leading monomial of \( \det(W_{k,r_0,r_1}) \), when the \( e_j \) coefficients are symbols. The terms \( \beta_j \) shall retain their numeric degrees. For \( k \equiv 0 \pmod{3} \) we obtain as the formal leading monomial:
\[
\text{FLM}(k, k/3, k/3; e_1, \ldots, e_k) = 
\]
(30)
Note that the sign in (30) is proper: it takes 3 row exchanges to move the rows \( k/3, 2k/3 \) and \( k \) to rows \( 1, 2, 3 \), and then \( k/3 - 4 \) exchanges to move the original row 1, which now is in row \( k/3 \), back to row 4, \( 2k/3 - 5 \) exchanges to move the original row 2, which is now in row \( 2k/3 \), back to row 5, and \( k - 6 \) exchanges to move the original row 3, which is now in row \( k \), back to row 6. The Vandermonde determinant in (30) is \( = (-1)^{k/3 - 1} k/3/3 - 1 k/3/3 - 1 (e_1 - e_2) (e_1 - e_3) (e_2 - e_3) \) which by (27) for \( k = 3 \) and exponents \( e_4, \ldots, e_k \)
proves (27) for $k$. The leading monomial formulas (28, 29) are proven similarly. □

The remainder of Section 2, performing early termination (Item 1) or having a bound $B \geq t$ (Item 2), carries over. Theorem 4.1 generalizes to derivative of order $\geq 3$ via its proof.

5 VARIANTS

The Prony algorithm for sparse polynomial interpolation has been used in many settings. One of the first is the multivariate version in [4]. In [24] the algorithm was deployed for Chebyshev and Pochhammer polynomial bases, which in [12] was generalized to Chebyshev bases of Second, Third and Fourth Kind. The numerical conditioning and stability has been studied extensively, as the algorithm builds exponential and harmonic sparse function approximations (see [6] and the references given). Digital error correction, that is, removing catastrophically false values, which is different from denoising the data, was introduced for the Prony algorithm in [5, 17], and for orthogonal bases in [2]. The state-of-the-art for error correction in sparse univariate interpolation today is [20]. All those adaptations can be transferred to sparse Hermite interpolation.

5.1 Multivariate Sparse Hermite Interpolation

The Prony algorithm is transferred to multivariate polynomials in [4]. We explain the transfer to sparse interpolation with values of partial derivatives, that on the case of bivariate polynomials:

$$f(x_1, x_2) = \sum_{j=1}^{t} c_j x_1^{e_{1j}} x_2^{e_{2j}} \in K[x_1, x_2], \quad \forall j: c_j \neq 0, (e_{1j}, e_{2j}) \in \mathbb{Z}^2.$$ 

The corresponding values are

$$a_i = f(\omega_1^{i_1+i_2}, \omega_2^{i_1+i_2}) \quad \text{and} \quad \tilde{a}_i = \omega_1^{i_1+i_2} \left( \frac{df(x_1, x_2)}{dx_1} \right)_{x_1=\omega_1^{i_1+i_2}, x_2=\omega_2^{i_1+i_2}}.$$ 

Again, the sequence \( \{ \tilde{a}_i \}_{i \geq 0} \) is linearly generated by the term locator polynomial \( \Lambda(z) = \prod_{i=1}^{2}(z - \omega_1^{i_1} \omega_2^{i_2}) \) and for randomly selected \( \omega_1, \omega_2 \), the matrix \( H_{f_1}(3) \) is non-singular with high probability, as indicated in the proof of Theorem 3.1. The technique in Step 4 in Algorithm 2.1 can possibly determine the \( e_{1j}, e_{2j} \) without a logarithm, but a logarithm computation is required for the \( e_{1j}, e_{2j} \)'s like in Steps 4(b)i and 4(b)ii. For fields \( K \) of characteristic \( p > 0 \), one may need an additional algorithm for computing \( e_{1j}, e_{2j} \) from \( b_j = \omega_1^{e_{1j}} \omega_2^{e_{2j}} \) in [4] \( \omega_1, \omega_2 \) are chosen different prime numbers for \( K = \mathbb{C} \) so that the \( b_j \)'s are distinct and the \( e_{1j}, e_{2j} \)'s are recoverable, but for fields \( K \) of characteristic 0 we can only recover \( e_{1j} \) directly from the derivative values and \( e_{2j} \) as a logarithm. In summary, the sparse multivariate Hermite interpolation algorithm recovers \( f(x_1, x_2) \) from \( t \) values, which include values at a partial derivative in one of the variables, at \( t + \lfloor t/2 \rfloor \) distinct arguments for \( f \).

Last, we note that the values at both partial derivatives may decrease the required number of arguments for certain exponents \( e_{1j}, e_{2j} \), but for the polynomial \( f(x, y) = \sum_{j=1}^{t} c_j x_j y_j \), additional values of \( df(x_j, y_j) / dx_1 \) seem not to help (cf. (26)). However, the algorithm in [27] transfers and partial derivatives in each variable can reduce the number of arguments to \( f \) in that algorithm, at the cost of more values. In general, the algorithm in [27] uses more values than the multivariate Prony algorithm (see also [9]).

5.2 Chebyshev Polynomial Basis

We explain the sparse recovery on Chebyshev polynomials of the First Kind; Second to Fourth kind follow from the substitutions in [12]: Let \( f(x) \) be a \( t \)-sparse polynomial in Chebyshev basis,

$$f(x) = \sum_{j=1}^{t} c_j T_{e_j}(x) \in K[x], \quad \forall j: c_j \neq 0, e_j \in \mathbb{Z}, e_1, \ldots, e_t \geq 0. \quad (31)$$

where \( (T_m)_{m \geq 0} \) are the \( m \)-degree Chebyshev Polynomials of the First Kind (we write “Chebyshev-1 Polynomials”):

$$T_m(x) = \begin{cases} 0 & m = 1 \\ (-1)^{\frac{m(m-1)}{2}} x^m & m \geq 2 \\ \end{cases}$$

\( \forall m \in \mathbb{Z}; T_m \left( \frac{1}{2} \left( y + \frac{1}{y} \right) \right) = \frac{1}{2} \left( y^m + \frac{1}{y^m} \right); \quad T_m \left( \cos(\theta) \right) = \cos(m\theta) \) with \( y = \exp(i\theta) \). \quad (32)

Note that for negative subscripts we have \( T_{-m}(x) = T_m(x) \). The recurrence (32) is non-trivial if the characteristic of \( K \) is \( \neq 2 \).

We suppose that both \( f(x) \) and \( f'(x) \) can be evaluated as a black box. The degrees \( e_j \) and coefficients \( c_j \) are recovered by sparse polynomial Hermite interpolation of \( g(y) \) via the substitution

$$g(y) def \neq f \left( \frac{1}{2} \left( y + \frac{1}{y} \right) \right) = \sum_{j=1}^{t} c_j \left( f'(y') \right)_{y'=y} \times \frac{1}{2} \left( 1 - \frac{1}{y^2} \right), \quad (33)$$

The Laurent polynomial \( g(y) \) in power basis is a \( 2t \)-sparse or a \( (2t-1) \)-sparse polynomial, the latter if \( e_0 = 0 \), and algorithms of Section 2 apply. By the chain rule for the derivative we obtain

$$g'(y) = \frac{dg(y)}{dy} = \left( f'(x) \right)_{x=\frac{1}{2} \left( y + \frac{1}{y} \right)} \times \frac{1}{2} \left( 1 - \frac{1}{y^2} \right),$$

which is a polynomial identity over any field of characteristic \( \neq 2 \), and therefore we have \( g'''(y) \) is equal

$$\frac{1}{2} \left( f'(x) \right)_{x=\frac{1}{2} \left( y + \frac{1}{y} \right)} \left( y - \frac{1}{y} \right) = \sum_{j=1}^{t} \frac{c_j}{2} \left( e_j g'^{(j)} + (-e_j) \frac{1}{y^{e_j}} \right).$$

If, for example, the Laurent polynomial \( g(y) \) is \( 2t \)-sparse, the sparse polynomial Hermite interpolation algorithm requires \( 6t \) values at \( 3t \) arguments. But because \( g(y) = g(1/y) \) and \( g(y) = -g(1/y) \) one only computes \( 3t + O(1) \) values of \( f \) for example, we can set \( i_0 = -\lceil (3t-1)/2 \rceil \) in Algorithm 2.1 and use \( g(\omega^{i_0}) \) and \( g(\omega^{i_0+2}) \) for \( i = 0, 1, \ldots, 3t-1 \). Because the values for exponents \( i_0+i \geq 1 \) can be computed from those at exponents \( i_0-i \), one requires \( N = 2t(\lceil (3t-1)/2 \rceil + 1) + 3t + 2t + \) values of \( f' \) at \( n = \lceil (3t-1)/2 \rceil \) \( +1 + t + [1/2] + 1 \) arguments. Note that the additional \( "O(1)" \) evaluations may be reduced by the technique in [12, Section 2], which uses the fact that the term locator polynomial \( \Lambda \) here is a reciprocal polynomial with a symmetric coefficient vector.

5.3 Error Correction

The error correction variants remove erroneous \( \tilde{a}_k \neq f(\xi_k) = a_k \) and erroneous \( \tilde{a}_k \neq \xi_k f'(\xi_k) = \tilde{a}_k \) from the list of input values \( \tilde{a}_i \) for \( i = 0, \ldots, L-1 \), \( \tilde{a}_i \) for \( i = 0, \ldots, L-1 \), where \( B \) is an upper bound for the sparsity: \( t \leq B \). Note that by the hats "^\^" on the input values we indicate that some values can be erroneous. As in the multiplicity code one assumes that there are \( k \leq \hat{E} \) errors, where the bound \( \hat{E} \) is input and can be determined from the error rate.
in the values. Furthermore, the algorithms, which are restricted to polynomial-time running time, produce a list of valid sparse Hermite interpolants. The basic algorithm uses \( E + 1 \) redundant blocks of \( 3B \) values: we list the true polynomial values without errors:

\[
\begin{align*}
&f(x_0), \ldots, f(x_i + B/[B - 1]), f'(x_i), \ldots, f'(x_i + B/[B - 1]), \\
&\vdots & \\
&f(x_{E1}) + \ldots, f(x_{E1} + B/[B - 1]), f'(x_{E1}), \ldots, f'(x_{E1} + B/[B - 1]).
\end{align*}
\]

We enforce in (34) that all arguments are distinct: \( \omega, \omega_m \) for \( (v, i) \neq (u, m) \). If in the \( N = 3(E + 1)B \) input values \( \hat{a}_i, \hat{a}_j \) there are \( k \leq E \) errors, at least one block is error-free and Algorithm 2.1 recovers \( f \) in the list of Hermite interpolants with \( \leq E \) errors; there may be others. Here the sparsity is computed from the largest non-singular \( H_{E1/[2]}(3) \) of that block, and all remaining values and all values in all other blocks are checked to verify that there are \( \leq E \) errors in total. We pursue the goal in [17, 20] and reduce the number of values \( N < 3(E + 1)B \) while recovering \( f \) in a list of Hermite interpolants, all of which have sparsity \( \leq B \) and \( \leq E \) errors in the values. Note that there are \( N = (E + 1)(B + [B/2]) \) distinct arguments to \( f \) in (34), and our goal is to have error correction at fewer arguments than without derivatives [17, 20]. The techniques in [17] are different from those in [20], the latter producing a higher error tolerance. Both methods apply to the Hermite setting.

As in [17], we can sub-sample at all arithmetic sequences of exponents: \( \hat{a}_{3 + 8x} \) for \( i = 0, \ldots, B + [B/2] - 1, \hat{a}_{3 + 8x+i} \) for \( i = 0, \ldots, B + [B/2] - 1 \). For a list of \( N = 2n \) values \( \hat{a}_{3 + 8i}, \hat{a}_{3 + 8i+1}, i = 0, 1, \ldots, n - 1 \), we guarantee that there are no errors for one pair \((x, \psi)\). On that sub-sample the \( t \)-sparse polynomial \( f_{3 + 8}(x) \) is interpolated. We have \( \hat{a}_{x + 8i} = \sum_{j=1}^t c_j \omega^{e_j x + 8j}, \psi \) is interpolated.

\[
\omega^j f_{x + 8}(\psi) = \sum_{j=1}^t c_j \omega^{e_j x + 8j} \psi = \psi \hat{a}_{x + 8i}.
\]

Therefore, Algorithm 2.1 recovers \( f_{3 + 8}(x) \) from the good values \( \hat{a}_{3 + 8i} \) and \( \hat{a}_{3 + 8i+1} \). From \( (x, \psi) \) and \( f_{3 + 8}(x) \) we compute \( f \). Finally, we determine the minimum \( n_{B + [B/2]} \) such that in the sequence of 2-dimensional vectors \( \hat{a} \) for \( i = 0, \ldots, n_{B + [B/2]} - 1 \) one guarantees a sub-sample \( \hat{a}_{x + 8i} \) for \( i = 0, \ldots, B + [B/2] - 1 \) without any of the \( E \) errors. For example, we obtain from [17, Table 1], that when removing any arbitrary 11 elements from \( \{1, \ldots, 124\} \) one always has an arithmetic sub-sequence of length 12 in the remaining elements. Therefore, for \( B > 8, E = 11 \) and \( n_{12} = 124 \) we can always find a sub-sample of \( B + 2 \cdot 124 \) values from which one can interpolate the 8-sparse polynomial \( f \). Overall, we need \( N = 2n = 2 \cdot 124 = 248 \) values to correct \( \leq E = 11 \) errors; note that \( 3(E + 1)B = 3 \cdot 12 \cdot 8 = 288 \), which is the count (34) for redundant blocking. Without derivatives, one needs \( N = n = n_{12}, \) values. The exact entry is not computed in [17, Table 1], but we have \( n_{12} > 131, 156 \), which is \( > 124 \) arguments.

In [20] we raise the decodable error rate by a different redundant blocking technique. We demonstrate the technique on the example \( B = 3 \) (cf. [20, Example 1]), We first interpolate \( \hat{a}_i \) for \( i = 0, \ldots, 6 = 2B \) and \( \hat{a}_i \) for \( i = 0, \ldots, 4 = 2B - 2, \) a total of \( 4B = 12 \) values, and zero \( \leq E = 1 \) error. The augmented coefficient matrix that corresponds to the linear system (3) is

\[
\begin{bmatrix}
\hat{a}_0 & \hat{a}_1 & \hat{a}_2 & \hat{a}_3 \\
\hat{a}_4 & \hat{a}_5 & \hat{a}_6 & \hat{a}_7 \\
\end{bmatrix}.
\]

Again, with \( \leq 1 \) error, we recover \( f \) and possibly other interpolants that cause \( \leq 1 \) error. If the error is in the \( \hat{a}_i \), we compute the candidate interpolant from the first 4 rows by the Prony algorithm. If the error is in \( \hat{a}_j \neq f(\omega^k) \), say, we compute candidate interpolants from Rows \( 4 \leq 6 \). We have for \( b_1 = \omega^k, b_2 = \omega^{k+4} \), \( b_2 = \omega^k \):
ACKNOWLEDGMENTS

I thank the reviewers for their comments.

This research was supported by the National Science Foundation under Grant CCF-1717100.


REFERENCES


7 APPENDIX: PROOF OF THEOREM 3.1 DETAILS

As stated, the formulas are proven by induction with minor expansion along the first column of $W_j$ in (11). The signs $\sigma$ and exponents $\eta$ satisfy the following recurrences:

$$\sigma(k,r) = \begin{cases} 1 & \text{for } k=1 \text{ and } (r_0 = 0 \text{ or } r_1 = 1), \\ -1 & \text{for } (k r) = 2, (1), \\ (-1)^{k\gamma} \sigma(k-1,r-1) & \text{for } \gamma < k, \\ (-1)^{k\gamma} \sigma(k-1,r) & \text{for } 0 \leq r < \gamma, \\ (-1)^{k\gamma} \sigma(k-2,\gamma-1) & \text{for } k > 4 \text{ and } \gamma \equiv 0 \text{ (mod } 2) \end{cases}$$

and $r = k/2$; (39)

$$\eta(k,r,j) = \begin{cases} k-j & \text{for } r = 0 \text{ or } r = k, \\ r-1 & \text{for } 2r > k \text{ and } j = 1, \\ \eta(k,r-1,j-1) & \text{for } 2r > k \text{ and } j > 1, \\ k-r-1 & \text{for } 2r < k \text{ and } j = 1, \\ \eta(k,r,j-1) & \text{for } 2r < k \text{ and } j > 1, \\ \left\lfloor \frac{k}{2} \right\rfloor & \text{for } k = 2r. \end{cases}$$

(40)

For $k = 1$ one has $\det(W_{1,j}) = \det([1]) = 1$ and $\det(W_{1,0}) = \det([1]) = 1$. Now let $k \geq 2$. We use the exponents $\epsilon_1, \ldots, \epsilon_k$ as arguments to $W_j(\epsilon_1, \ldots, \epsilon_k)$ in (11). Note that here $\beta_j = x^{\gamma^2}$: one could also use a multivariate term order with the variable order $\beta_1 \cdots > \beta_k$, as we need in Section 5.1. If $r = 1 < k = r$, then $\gamma^{r+1}$ is the highest degree term in the first column, and the
leading monomial of \( \det(W_{k,r}(e_1, \ldots, e_k)) \) is \((-1)^r \beta_1^{k-1} \times \) the leading monomial of \( \det(W_{k-1,r-1}(e_2, \ldots, e_k)) \). If \( r - 1 < k - r - 1 \), then \( \beta_1^{k-r} \) is the highest degree term in the first column, and the leading monomial of \( \det(W_{k,r}(e_1, \ldots, e_k)) \) is \((-1)^{k-r} \beta_1^{k-r-1} \times \) the leading monomial of \( \det(W_{k-1,r}(e_2, \ldots, e_k)) \). We obtain (12, 39, 40) for all \( k \neq 2r \).

Finally, we establish the formulas for \( k = 2r \). The leading monomial, denoted by LM, of \( \det(W_{k,k/2}(e_1, \ldots, e_k)) \) is

\[
\begin{align*}
&(-1)^{k-1} e_1 \beta_1^{k-1/2} \times \text{LM}(\det(W_{k-1,k/2}(e_2, \ldots, e_k)))+ \\
&(-1)^{k-1} \beta_1^{k-1/2} \times \text{LM}(\det(W_{k-1,k/2-1}(e_2, \ldots, e_k))).
\end{align*}
\]

The leading monomial in (41) is given by (16) and the leading monomial in (42) is given by (17), both for \( k - 1 \) replacing \( K \).

The induction hypothesis for \( k-1 \), one obtains as the leading monomial of \( W_k(e_1, \ldots, e_k) \), using (39):

\[
\begin{align*}
&((−1) × (k−1, k−2) e_{1} + (−1)^{k−2} \sigma(k−1, k−2) e_{2} ) \beta_{1}^{k−2} × \\
&((−1)^{k−2} \sigma(k−2, k−3) − (−1)^{k−2} \sigma(k−2, k−3)) \beta_{2}^{k−2} × \\
&(e_{1} − e_{2}) \cdots (e_{k−1} − e_{k}) \beta_{k}^{k−2} \cdots \beta_{k−3} \beta_{k−2}.
\end{align*}
\]

Because \( k \) is even, the first factor in (43) is \( \sigma(k, k/2)(e_1 - e_2) \) by (39), which concludes the inductive proof.

## 8 APPENDIX

This appendix is not included in the ISSAC Proceedings.

### Notation (in alphabetic order):

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>matrix of coefficients</td>
</tr>
<tr>
<td>( a_i )</td>
<td>coefficients of the polynomial</td>
</tr>
<tr>
<td>( \hat{a}_i )</td>
<td>coefficients of the polynomial</td>
</tr>
<tr>
<td>( b )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( B )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( \beta_j )</td>
<td>coefficients of the polynomial</td>
</tr>
<tr>
<td>( c_j )</td>
<td>coefficients of the polynomial</td>
</tr>
<tr>
<td>( D )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( E )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( \epsilon_j )</td>
<td>coefficients of the polynomial</td>
</tr>
<tr>
<td>( f(x) )</td>
<td>the transformed ( f((y + 1) / y) / 2 )</td>
</tr>
<tr>
<td>( g(y) )</td>
<td>the sparse interpolant</td>
</tr>
<tr>
<td>( \delta )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( \eta )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( \theta )</td>
<td>the characteristic of ( K ), if positive</td>
</tr>
<tr>
<td>( \tau(k, r, f^2) )</td>
<td>row permutation of ( W_{k,r} ) to move rows with ( e_j = 0 ) back</td>
</tr>
<tr>
<td>( \lambda_j, \Lambda(z) )</td>
<td>the term locator polynomial</td>
</tr>
<tr>
<td>( N )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( N )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( n )</td>
<td>the number of arguments</td>
</tr>
<tr>
<td>( \sigma(k, r) )</td>
<td>the symbolic coefficient matrix</td>
</tr>
<tr>
<td>( T_m(x) )</td>
<td>the degree Chebyshev polynomial of the First Kind</td>
</tr>
<tr>
<td>( t )</td>
<td>the number of terms in ( f )</td>
</tr>
<tr>
<td>( \tau(k, r, f^2, j) )</td>
<td>row permutation of ( W_{k,r} ) to move rows with ( e_j = 0 ) back</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the left matrix factor of ( \hat{H}_{k,r} )</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the right matrix factor of ( \hat{H}_{k,r} )</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the symbolic left matrix factor for 2 derivatives</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the symbolic left matrix factor for 2 derivatives</td>
</tr>
<tr>
<td>( \chi, \psi )</td>
<td>the linear arithmetic index subsequence</td>
</tr>
<tr>
<td>( \chi, \psi )</td>
<td>the linear arithmetic index subsequence</td>
</tr>
<tr>
<td>( \omega )</td>
<td>the variable in the term locator polynomial</td>
</tr>
<tr>
<td>( \omega )</td>
<td>the variable in the term locator polynomial</td>
</tr>
<tr>
<td>( \omega )</td>
<td>the variable in the term locator polynomial</td>
</tr>
</tbody>
</table>

### Notation continued (in alphabetic order):

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta )</td>
<td>the characteristic of ( K ), if positive</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the left matrix factor of ( \hat{H}_{k,r} )</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the right matrix factor of ( \hat{H}_{k,r} )</td>
</tr>
<tr>
<td>( \hat{W}_{k,r} )</td>
<td>the symbolic left matrix factor for 2 derivatives</td>
</tr>
<tr>
<td>( \chi, \psi )</td>
<td>the linear arithmetic index subsequence</td>
</tr>
<tr>
<td>( \chi, \psi )</td>
<td>the linear arithmetic index subsequence</td>
</tr>
<tr>
<td>( \omega )</td>
<td>the variable in the term locator polynomial</td>
</tr>
<tr>
<td>( \omega )</td>
<td>the variable in the term locator polynomial</td>
</tr>
<tr>
<td>( \omega )</td>
<td>the variable in the term locator polynomial</td>
</tr>
</tbody>
</table>