Sparse Polynomial Interpolation by Variable Shift in the Presence of Noise and Outliers in the Evaluations*

Brice Boyer, Matthew T. Comer, and Erich L. Kaltofen

Key words: approximate function recovery, numeric and exact polynomial interpolation, outlier detection, sparse representations

Abstract We compute approximate sparse polynomial models of the form $\widetilde{f}(x) = \sum_{j=1}^{t} \widetilde{c}_j (x - \widetilde{s})^{e_j}$ to a function $f(x)$, of which an approximation of the evaluation $f(\zeta)$ at any complex argument value ζ can be obtained. We assume that several of the returned function evaluations $f(\zeta)$ are perturbed not just by approximation/noise errors but also by highly perturbed outliers. None of the $\widetilde{c}_j$, $\widetilde{s}$, e_j and the location of the outliers are known before-hand. We use a numerical version of an exact algorithm by [4] together with a numerical version of the Reed-Solomon error correcting coding algorithm. We also compare with a simpler approach based on root finding of derivatives, while restricted to characteristic 0. In this preliminary report, we discuss how some of the problems of numerical instability and ill-conditioning in the arising optimization problems can be overcome. By way of experiments, we show that our techniques can recover approximate sparse shifted polynomial models, provided that there are few terms t, few outliers and that the sparse shift is relatively small.

1 Introduction

Sparse polynomial interpolation algorithms, where the number of values required depends on the number of non-zero terms in a chosen representation base rather

Brice Boyer, Matthew T. Comer, Erich L. Kaltofen
Department of Mathematics, North Carolina State University, Raleigh, North Carolina 27695-8205, USA, e-mail: {bbboyer,mcomer,kaltofen}@ncsu.edu, webpage: http://www.kaltofen.us

* This material is based on work supported in part by the National Science Foundation under Grants CCF-0830347 and CCF-1115772.

than on the degree of the polynomial, originate from two sources. One is Prony's 1795 algorithm for reconstructing an exponential sum [18] (see also [2]) and another is Blahut's exact sparse polynomial interpolation algorithm in the decoding phase of the Reed-Solomon error correcting code. Both algorithms first determine the term structure via the generator ("error locator polynomial") of the linear recurrent sequence of the values $f(\omega^i)$, $i = 0,1,2,\ldots$, of the sparse function f. Blahut's algorithm has led to a rich collection of exact sparse multivariate polynomial interpolation algorithms, among them [1, 12, 20, 16, 13]. Prony's algorithm suffers from numerical instability unless randomization controls, with high probability and for functions of significant sparsity, the conditioning of intermediate Hankel matrices. The probabilistic spectral analysis in the GLL algorithm [5, 7] adapts the analysis of the exact early termination algorithm of [13]. The resulting numerical sparse interpolation algorithms have recently had a high impact on medical signal processing; see the web site `http://smartcare.be` of Wen-shin Lee and her collaborators. The GLL algorithm can be generalized to multivariate polynomial and rational function recovery via Zippel's variable-by-variable sparse interpolation [14].

Already in the beginning days of symbolic computation, the choice of polynomial basis was recognized: $(x-2)^{100}+1$ is a concise representation of a polynomial with 101 terms in power basis representation. The discrete-continuous optimization problem of computing the sparsest shift of an exact univariate polynomial surprisingly has a polynomial-time solution [9, 10, 4]. Our subject is the computation of an approximate interpolant that is sparsified through a shift. One can interpret our algorithm as a numerical version of the exact sparsest shift algorithms. As in least squares fitting, noise can be controlled by oversampling (cf. [8]). The main difficulty is that the shift is unknown. Our numerical algorithm adapts Algorithm `UniSparsestShifts⟨one proj, two seq⟩` in [4] to compute the shift: `UniSparsestShifts⟨one proj, two seq⟩` carries the shift as a symbolic variable z throughout the sparse interpolation algorithm. Since the coefficients of the polynomials in the shift variable z are spoiled by noise, the GCD step becomes an approximate polynomial GCD. A main question answered here is whether the arising non-linear optimization problems remain well-conditioned. Our answer is a conditional yes: an optimal approximate shift is found among the arguments of all local minima, but the number of local minima is high, preventing the application of standard approximate GCD algorithms. Instead, we perform global optimization, as a fallback, by computing all zeros of the gradient ideal. In addition, our algorithm requires high precision floating point arithmetic.

In [3], we have introduced outlier values to the sparse interpolation problem. There, outlier removal requires high oversampling, as the worst case of k-error linear complexity is $2t(2k+1)$, where t is the generator degree. However, ours is only an upper bound for sparse interpolation. The situation is different for Algorithm `UniSparsestShifts⟨one proj, two seq⟩`. Outliers can be removed at the construction stage of the values containing the shift variable z, by a numeric version of Blahut's decoding algorithm for interpolation with errors. The algorithm, numerical interpolation with outliers, is interesting in its own right. As we will show in Section 3, the analysis in [7, 3] does not directly apply, as random-

ization can only be applied with a limited choice of random evaluation points. We have successfully tested it as a subroutine of our numerical sparsest shift algorithm. Note that a few outliers per interpolation lead to a very small sparse interpolation problem for error location, which can be handled successfully by sparse interpolation with noisy values.

For the sake of comparison with this algorithm, we restrict to characteristic 0 and compare a sparse shift representation to a Taylor expansion expressed at a point that will make the representation sparse. This leads to finding a root common to many derivatives. Combined with a weighted least squares fit for removing outliers and tolerating noise, we manage to compare favorably to the main algorithm.

In Section 4, we present the preliminary experimental results that our algorithms can recover sparse models even in the presence of substantial noise and outliers. See Section 4.3 for our conclusions.

2 Computing Sparse Shifts

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

2.1 Main Algorithm with Early Termination.

The Early Termination Theorem in [13] is at the heart of computing a sparsest shift. Let

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

be a t-sparse polynomial with coefficients in an integral domain D. Furthermore, let

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

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

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

one must have $\det(\mathscr{H}_{t+1}) = 0$. Theorem 4 in [13] simply states that $\det(\mathscr{H}_i) \neq 0$ for all $1 \leq i \leq t$. One can replace the indeterminate y by a randomly sampled coefficient domain element to have $\det(\mathscr{H}_i) \neq 0$ for all $1 \leq i \leq t$ with high probability (w.h.p.).

We seek an s in any extension of the field of K such that for a given $f(x) \in \mathsf{K}[x]$ the polynomial $f(x+s) = h(x)$ is t-sparse for a minimal t. Now consider $g(x) = f(x+z) \in \mathsf{D}[x]$ with $\mathsf{D} = \mathsf{K}[z]$. We have $\Delta_i(y,z) = \det(\mathscr{H}_i) \in (\mathsf{K}[z])[y]$; note that $\alpha_i(y,z) = g(y^i) = f(y^i + z)$. By the above Theorem 4, the sparsest shift is an s with $\Delta_{t+1}(y,s) = 0$ for the smallest t. Algorithm `UniSparsestShifts⟨one proj, two seq⟩` computes s as

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

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

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

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

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

We now give the outline of our Algorithm `ApproxUniSparseShift⟨one proj, sev seq⟩`. Note that because of the approximate nature of the shifted sparse interpolant, there is a trade-off between backward error and sparsity. Hence we call our algorithm a "sparse shift" algorithm. As in Algorithm `UniSparsestShifts⟨one proj, two seq⟩`, for L complex values $y = \omega^{[1]}, \omega^{[2]}, \ldots, \omega^{[L]}$ we compute $\widetilde{\delta}_i^{[\ell]}(z) = \widetilde{\Delta}_i(\omega^{[\ell]}, z)$ from $\widetilde{\alpha}_i(\omega^{[\ell]}, z)$, $\ell = 1, 2, \ldots, L$. Here the tilde accent mark $\widetilde{\ }$ indicates that the values have noise in their scalars. As in [7], we choose the $\omega^{[\ell]}$ to be different random roots of unity of prime order. Our algorithm consists of the four following tasks:

Step 1: For $\ell = 1, 2, \ldots, L$, compute the numeric complex polynomials $\widetilde{\alpha}_i(\omega^{[\ell]}, z)$ via a numeric version of the Blahut decoding algorithm; see Section 3. Step 1 is assumed to have removed all outliers.

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

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

too small. It might happen that an accurate shift is diagnosed too early, but then the constructed model produces a worse backward error.
The 2-norm distance to the nearest polynomial system with a common root $\widetilde{s}$ is given by formula (see [11] and the literature cited there):

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

where $d = \max_\ell\{\deg(\widetilde{\delta}_i^{[\ell]}(z))\}$ and in all polynomials, any term coefficients of z^m, where $m \in \{0,1,\ldots,d\}$, can be deformed.
In our experiments in Section 4, we have only considered real shifts $\widetilde{s} \in \mathbb{R}$. The optimization problem is then

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

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

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

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

Step 4: With the approximate sparsest shift $\widetilde{s}$, complete the sparse polynomial reconstruction, as in [6] and [3, Section 6].
One reuses the evaluations from previous steps, having removed those that were declared outliers in Step 1.

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

2.2 Using Taylor Expansions

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

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

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

Remark 1. It is stated in Theorem 1 in [17] that if $t \leq (d+1)/2$ then the shift s is unique and rational. Moreover, the proof gives the stronger statement: for any other shift $\hat{s}$, with a sparsity $\hat{t}$, one has $\hat{t} > d+1-t$.

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

Lemma 1. *Let S_{2t} be the list of the last $2t$ elements in S. The root that zeros the maximum number of polynomials in S_{2t} is the sparsest shift.*

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

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

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

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

Step 1: Compute the exact interpolant using $D+1$ calls to the black box, where $D \geq d$.
Step 2: Try early termination on S_τ, for a small τ, and return if successful.
Step 3: Compute all remaining derivatives in S_θ, for $\theta = \min(2T, D)$.
Step 4: The sparsest shift is the rational root s that zeros the most derivatives in S_θ.
Step 5: The "back-shifted" polynomial $f(x+s)$ gives the support for the sparse polynomial.

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

Step 1: Compute a degree-D weighted least squares fit with $O(D+E)$ calls to the black box.
Step 2: Remove outliers by comparing relative errors, then update the fit.
Step 3: Compute the θ derivatives in S_θ (possibly terminate early and proceed to Step 6).
Step 4: The approximate root s that zeros most derivatives is the sparsest shift.
Step 5: The polynomial $f(x+s)$ gives the support for the sparse polynomial.
Step 6: A Newton iteration can be conducted on the result of Step 5 to increase accuracy.

2.3 Discussion on the Numeric Algorithm

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

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

3 Numeric Interpolation with Outliers

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

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

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

Fourier (Vandermonde) matrix $V(\omega) = [\omega^{i\cdot m}]_{0\le i,m\le n-1}$ satisfies

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

hence

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

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

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

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

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

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

Not knowing k, the probabilistic analysis of early termination as in [13] and Section 2 does not directly apply, as the choice of ω is restricted to a primitive n'th root of unity. Furthermore, the locations e_j of the outlier errors a'_{e_j} may depend on the evaluation points ω^i. Blahut's decoding algorithm processes all $2E$ values c'_l.

If one has ε_i in each evaluation, namely $\widetilde{b}_i = a_i + a'_i + \varepsilon_i$, where $|a_{e_j} - a'_{e_j}|/|a_{e_j}| \gg 0$ (one may assume that $\varepsilon_{e_j} = 0$), then

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

so

$$\widetilde{c}'_l = (V(\omega^{-1})\tilde{\mathbf{b}})_l = g(\omega^{-l}) + (V(\omega^{-1})\varepsilon)_l = g(\omega^{-l}) + \overline{\varepsilon}_l \text{ for } d \le l \le d+2E-1,$$

where $|\overline{\varepsilon}_l| \le |\varepsilon_1| + \cdots + |\varepsilon_n|$. Again, there is an immediate trade-off between noise and outliers: at what magnitude does noise ε_i become an outlier a'_i? For now we assume that the relative error in noise is small, say 10^{-10}, while the relative error in outliers is big, say 10^5. The recovery of an approximate interpolant $\widetilde{g}(x) = \sum_{j=1}^{k}\widetilde{a}'_{e_j}x^{e_j}$ for the evaluations $\widetilde{c}'_l$ hinges on the condition number of the $k \times k$ Hankel matrix:

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

If the matrix is well-conditioned, the error locations e_j can be determined from the approximate linear generator $\widetilde{\Lambda}$ as in the GLL algorithm [7, 3]. As is shown there, the conditioning is bounded by $1/|\omega^{e_u} - \omega^{e_v}|$. Large values there are prevented by randomizing ω, as the term exponents e_j are fixed for any evaluation. Using an ω^r instead of ω here, where $\mathrm{GCD}(r,n) = 1$, allows redistributing of the ω^{e_j}, but the e_j may then become different.

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

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

which, by our assumption on a large outlier a'_{e_1} and small noise, is a well-conditioned matrix. This is the case we tested in Section 4.

Remark 3. When the relative difference between the magnitudes of the outlier a'_{e_1} and noise $\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{d+2E-1}$ is not so pronounced, erroneous recovery of the exponent e_1 can occur: we have $(\widetilde{c}'_d, \widetilde{c}'_{d+1}, \ldots, \widetilde{c}'_{d+2E-1}) = (\widetilde{c}'_d, \widetilde{c}'_{d+1})$, so the linear generator $\widetilde{\Lambda}(\lambda) = \lambda - \omega^{-e_1}$ can be approximated by computing

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

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

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

By the distribution of complex roots of unity (of order n) on the unit circle, we have that $|\omega^{s+1} - \omega^s| = |\omega - 1| = 2\sin(\pi/n)$ for any integer s. Thus, $|\widetilde{\omega} - \omega^{-e_1}| < \sin(\pi/n)$ will guarantee $|\widetilde{\omega} - \omega^{-e_1}| < |\widetilde{\omega} - \omega^s|$ for any $s \not\equiv -e_1 \pmod{n}$.

Combining this fact with (6) above, we arrive at the sufficient condition

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

Table 1 shows some experiments of decreasing $\Theta^{[\text{abs}]}_{\text{outlier}}$ for a fixed $\varepsilon^{[\text{abs}]}_{\text{noise}}$. Throughout the experiment, we have $f(x) = 87x^{11} - 56x^{10} - 62x^8 + 97x^7 - 73x^4 - 4x^3 - 83x - 10$ and $d - 1 = 11$, evaluating at powers of the order $n = d + 2E = 14$ complex root of unity $\omega = \exp(2\pi\boldsymbol{i}/14)$. We add to each evaluation noise, which is implemented as a complex number with polar modulus uniformly chosen at random in the range $[0, \varepsilon^{[\text{abs}]}_{\text{noise}}]$ and polar argument uniformly chosen at random in the range $[0, 2\pi]$. An absolute outlier value is chosen the same way, but the modulus

is in the range $[\Theta^{[\mathrm{abs}]}_{\mathrm{outlier}}, 2\Theta^{[\mathrm{abs}]}_{\mathrm{outlier}}]$; the exponent e_1 is also chosen uniformly at random from $\{0,1,\ldots,d+2E-1=13\}$. Each row of the table corresponds to 1000 realizations of the random variable that generates noise and outliers, re-seeding the random number generator with each run. All computations were performed with 15 floating point digits of precision. In the table, $C_n = \sin(\pi/n)/\big(2+\sin(\pi/n)\big)$.

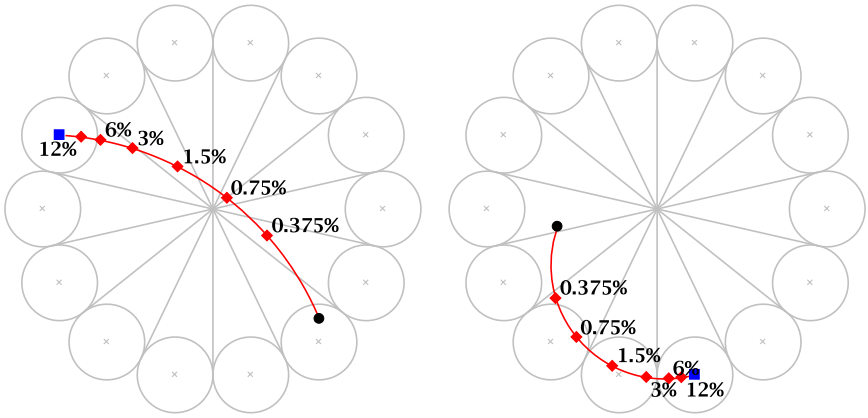
The column "% Circle" shows the percentage of runs where $|\widetilde{\omega}-\omega^{-e_1}| < \sin(\pi/n)$; "% Sector" shows the percentage of runs where $|\widetilde{\omega}-\omega^{-e_1}| \geq \sin(\pi/n)$, but $|\widetilde{\omega}-\omega^{-e_1}| < |\widetilde{\omega}-\omega^{s}|$ for any $s \not\equiv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

Table 1: Experiments of varying outlier error in the presence of noise

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

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

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



(a) (b)

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

Decoding the interpolant $W\mathbf{b}$ can also be done via the extended Euclidean algorithm for any ω with $\omega^{e_u} \neq \omega^{e_v}$: the Berlekamp-Welch algorithm; see [15]. We will study the numerical properties of variants based on approximate GCD techniques in follow-up work.

4 Implementation and Experiments of `NumericSparsestShift`

4.1 Illustrative Examples for the Main Algorithm

We reversely engineer a noisy black box for

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

we can recover from $L = 3$ sequences, with a relative noise in the evaluations of 1×10^{-14}, and again 1 outlier per interpolation, the approximate model

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

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

For this particular example, we see a case of the effect mentioned in [3], where the sparse model can fit the noisy evaluations nearly as well (and sometimes better) than the exact black box.

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

4.2 Comparison with the Naïve Algorithm

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

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

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

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

4.3 Discussion

Our preliminary experiments lead to the following conclusions: Our correction of 1 outlier per interpolation with Blahut's numerical decoding is highly numerically reliable. The optimization problem in Step 3 requires substantial precision for its real root finding, and is numerically sensitive when the shift is large and there is noise in the evaluations. Our main algorithm works well without noise and outliers, or in high precision with noise when the shift is small and the sparsity is high. We plan to work on a more thorough experimental analysis, including the case of two or more outliers per interpolation. The naïve algorithm gives motivation and potential for improvements to the main one. On the other hand, the number of calls to the black box in the former could be reduced.

References

1. Ben-Or, M., Tiwari, P.: A deterministic algorithm for sparse multivariate polynomial interpolation. In: Proc. Twentieth Annual ACM Symp. Theory Comput., pp. 301–309. ACM Press, New York, N.Y. (1988)
2. Brezinski, C.: Computational Aspects of Linear Control. Springer Verlag, Heidelberg, Germany (2002)
3. Comer, M.T., Kaltofen, E.L., Pernet, C.: Sparse polynomial interpolation and Berlekamp/Massey algorithms that correct outlier errors in input values. In: J. van der Hoeven, M. van Hoeij (eds.) ISSAC 2012 Proc. 37th Internat. Symp. Symbolic Algebraic Comput., pp. 138–145. Association for Computing Machinery, New York, N. Y. (2012). URL: EKbib/12/CKP12.pdf
4. Giesbrecht, M., Kaltofen, E., Lee, W.: Algorithms for computing sparsest shifts of polynomials in power, Chebychev, and Pochhammer bases. J. Symbolic Comput. **36**(3–4), 401–424 (2003). Special issue Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL: EKbib/03/GKL03.pdf
5. Giesbrecht, M., Labahn, G., Lee, W.: Symbolic-numeric sparse interpolation of multivariate polynomials (extended abstract). In: Proc. Ninth Rhine Workshop on Computer Algebra (RWCA'04), University of Nijmegen, the Netherlands, pp. 127–139 (2004)
6. Giesbrecht, M., Labahn, G., Lee, W.: Symbolic-numeric sparse interpolation of multivariate polynomials. In: J.G. Dumas (ed.) ISSAC MMVI Proc. 2006 Internat. Symp. Symbolic Algebraic Comput., pp. 116–123. ACM Press, New York, N. Y. (2006). DOI http://doi.acm.org/10.1145/1145768.1145792
7. Giesbrecht, M., Labahn, G., Lee, W.: Symbolic-numeric sparse interpolation of multivariate polynomials. J. Symbolic Comput. **44**, 943–959 (2009)
8. Giesbrecht, M., Roche, D.S.: Diversification improves interpolation. In: A. Leykin (ed.) Proc. 2011 Internat. Symp. Symbolic Algebraic Comput. ISSAC 2011, pp. 123–130. Association for Computing Machinery, New York, N. Y. (2011)
9. Grigoriev, D.Y., Karpinski, M.: A zero-test and an interpolation algorithm for the shifted sparse polynomials. In: Proc. AAECC-10, *Lect. Notes Comput. Sci.*, vol. 673, pp. 162–169. Springer Verlag, Heidelberg, Germany (1993)
10. Grigoriev, D.Y., Lakshman, Y.N.: Algorithms for computing sparse shifts for multivariate polynomials. Applic. Algebra Engin. Commun. Comput. **11**(1), 43–67 (2000)
11. Hutton, S.E., Kaltofen, E.L., Zhi, L.: Computing the radius of positive semidefiniteness of a multivariate real polynomial via a dual of Seidenberg's method. In: Watt [19], pp. 227–234. URL: EKbib/10/HKZ10.pdf

12. Kaltofen, E., Lakshman Y. N., Wiley, J.M.: Modular rational sparse multivariate polynomial interpolation. In: S. Watanabe, M. Nagata (eds.) Proc. 1990 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'90), pp. 135–139. ACM Press (1990). URL: EKbib/90/KLW90.pdf
13. Kaltofen, E., Lee, W.: Early termination in sparse interpolation algorithms. J. Symbolic Comput. **36**(3–4), 365–400 (2003). Special issue Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL: EKbib/03/KL03.pdf
14. Kaltofen, E., Yang, Z., Zhi, L.: On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In: J. Verschelde, S.M. Watt (eds.) SNC'07 Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput., pp. 11–17. ACM Press, New York, N. Y. (2007). URL: EKbib/07/KYZ07.pdf
15. Khonji, M., Pernet, C., Roch, J.L., Roche, T., Stalinsky, T.: Output-sensitive decoding for redundant residue systems. In: Watt [19], pp. 265–272
16. Lakshman Y. N., Saunders, B.D.: Sparse polynomial interpolation in non-standard bases. SIAM J. Comput. **24**(2), 387–397 (1995)
17. Lakshman Y. N., Saunders, B.D.: Sparse shifts for univariate polynomials. Applic. Algebra Engin. Commun. Comput. **7**(5), 351–364 (1996)
18. Prony, R.: Essai expérimental et analytique sur les lois de la Dilatabilité de fluides élastiques et sur celles de la Force expansive de la vapeur de l'eau et de la vapeur de l'alcool, à différentes températures. J. de l'École Polytechnique **1**, 24–76 (1795). R. Prony is Gaspard(-Clair-François-Marie) Riche, baron de Prony
19. Watt, S.M. (ed.): Proc. 2010 Internat. Symp. Symbolic Algebraic Comput. ISSAC 2010. Association for Computing Machinery, New York, N. Y. (2010)
20. Zippel, R.: Interpolating polynomials from their values. J. Symbolic Comput. **9**(3), 375–403 (1990)