Symbolic-numeric Sparse Interpolation of Multivariate Polynomials

[Extended Abstract]

Mark Giesbrecht
School of Computer Science
University of Waterloo,
Canada
mwg@uwaterloo.ca

George Labahn
School of Computer Science
University of Waterloo,
Canada
glabahn@uwaterloo.ca

Wen-shin Lee
Dept. Wiskunde en Informatica
Universiteit Antwerpen,
Belgium
wenshin.lee@ua.ac.be

ABSTRACT

We consider the problem of sparse interpolation of an approximate multivariate black-box polynomial in floating-point arithmetic. That is, both the inputs and outputs of the black-box polynomial have some error, and all numbers are represented in standard, fixed-precision, floating point arithmetic. By interpolating the black box evaluated at *random* primitive roots of unity, we give efficient and numerically robust solutions. We note the similarity between the exact Ben-Or/Tiwari sparse interpolation algorithm and the classical Prony's method for interpolating a sum of exponential functions, and exploit the generalized eigenvalue reformulation of Prony's method. We analyze the numerical stability of our algorithms and the sensitivity of the solutions, as well as the expected conditioning achieved through randomization. Finally, we demonstrate the effectiveness of our techniques in practice through numerical experiments and applications.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algorithms

General Terms

Algorithms.

Keywords

Symbolic-numeric computing, multivariate interpolation

1. INTRODUCTION

In many computer algebra applications, multivariate polynomials are encountered in such a way that an implicit representation is the most cost-effective for computing. A black-box representation of a multivariate polynomial is a procedure that, for any given input, outputs the evaluation of the polynomial at that input. Black boxes may also represent approximate polynomials, where the coefficients may have errors or *noise*. In such cases one would expect the evaluations of the black box to have errors as well. In this paper we demonstrate effective numerical algorithms for the sparse interpolation problem for approximate black-box polynomials: how to reconstruct an accurate representation of the polynomial in the power basis. This representation is parameterized by the sparsity — the number of non-zero terms — and its cost will be proportional to this sparsity (instead of the dense representation size).

Suppose we have a black box for a multivariate polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$ which we know to be t-sparse, that is,

$$f = \sum_{1 \le j \le t} c_j x_1^{d_{j_1}} x_2^{d_{j_2}} \cdots x_n^{d_{j_n}} \in \mathbb{C}[x_1, \ldots, x_n], \qquad (1.1)$$

where $c_1, \ldots, c_t \in \mathbb{C}$, $(d_{j_1}, \ldots, d_{j_n}) \in \mathbb{Z}_{\ge 0}$ are distinct for $1 \le j \le t$, and t is "small." Evaluating

$$b_1 = f(\alpha_1),\ b_2 = f(\alpha_2),\ \ldots, b_\kappa = f(\alpha_\kappa),$$

at our own choice of points $\alpha_1, \alpha_2, \ldots \alpha_\kappa \in \mathbb{C}^n$, where $\kappa = O(t)$, we would like to determine the coefficients $c_1, \ldots, c_t \in \mathbb{C}$ and the exponents $d_{j_1}, \ldots, d_{j_n}$, for $1 \le j \le t$, of f. If the evaluation points are not exact, this may not possible, so we ask that our algorithms are numerically robust: if the evaluations $\widetilde{b}_1, \ldots, \widetilde{b}_\kappa$ are relatively close to their true values, we want the coefficients $\widetilde{c}_1, \ldots, \widetilde{c}_t \in \mathbb{C}$ we compute to also be relatively close to their values in the exact computation.

Black-box polynomials appear naturally in applications such as polynomial systems [5] and the manipulation of sparse polynomials (e.g., factoring polynomials [16, 6]). Sparsity with respect to the power (or other) basis is also playing an ever more important role in computer algebra. As problem sizes increase, we must be able to capitalize on the structure, and develop algorithms whose costs are proportional only to the size of the support for the algebraic structures with which we are computing. A primary example is that of (exact) sparse interpolation of f as in (1.1), reconstructing the exponents d_{j_k} and non-zero coefficients $c_1, \ldots, c_t$ from a small number of evaluations of f. The best known exact interpolation methods that are sensitive to the sparsity of the target polynomial are the algorithms of Ben-Or/Tiwari [3] and of Zippel [25]. Although both approaches have been generalized and improved (see [26, 15, 14, 24]), they all depend upon exact arithmetic.

With recent advances in approximate polynomial computation, we are led to investigate the problem of sparse interpolation in an approximate setting. Black-box polynomials can capture an implicit model of an object which can only be sampled approximately. Moreover, sheer size and complexity requires that we exploit sparsity and use efficient (i.e., IEEE floating point) arithmetic in a numerically sound manner. Polynomial interpolation is an important component in approximate multivariate factorization algorithms [7], and sparse interpolation can often be used to speed up the procedure when there are more than two variables.

The problem of multivariate polynomial interpolation is not new, with early work going back at least to Kronecker [17]. See [8] for a survey of early work in this area. More recently there has been much activity on the topic, of both an algorithmic and mathematical nature. See [18] for a good survey of the state of the art. To our knowledge, none of the previous numerical work has considered the problems of identifying the (sparse) support and sparse multivariate interpolation. Sparsity is considered in a different, bit-complexity model, using arbitrary precision arithmetic by Mansour [19], who presents a randomized algorithm for interpolating a sparse *integer* polynomial from (limited precision) interpolation points (wherein bits of guaranteed accuracy can be extracted at unit cost). The algorithm guarantees an answer with controllably high probability, though its cost is dependent on the absolute size L of the largest coefficient in f, as well as the sparsity t and degree. Moreover, it would also seem to be quite expensive, requiring about $O((\log L)^8 \cdot t \log \deg f)$ bit operations.

In Section 2, we present the algorithm of Gaspard Riche, Baron de Prony, from 1795 [21] (generally referred to as "Prony's algorithm") for interpolating sums of exponential functions. We show that it is very similar to the algorithm for sparse polynomial interpolation of Ben-Or and Tiwari [3]. In Section 3 we adapt Ben-Or and Tiwari's method to floating-point arithmetic and identify the numerical difficulties encountered. We also adapt a recent, and much more stable, variant of Prony's algorithm [20, 12] to the problem of polynomial interpolation. This algorithm, developed for the *shape from moments problem*, makes use of generalized eigenvalues for added numerical stability.

In Section 4, we give a detailed analysis of the numerical behaviour of our algorithms and sensitivity of the underlying problems. In particular, we show that the stability of our algorithms is governed by $\|V^{-1}\|^2/\min|c_j|$, where V is a (hidden) Vandermonde matrix of the support terms of the polynomial evaluated at the sample points. The coefficients $c_1,\ldots,c_t$ are intrinsic to the problem, and in some sense having one of them too small may indicate an incorrect choice of t. On the other hand, the condition of V (as indicated by $\|V^{-1}\|$) is really a property of the method, and we address this directly.

Our key innovation in this regard is the use of evaluation points at *random* roots of unity. The use of roots of unity allows us to reconstruct the exponents in our multivariate problem using the Chinese remainder algorithm. Numerically this also adds considerable stability to the interpolation problem by reducing large variations in magnitude implied by evaluating polynomials of high degree. In particular, the Vandermonde matrix V discussed above will have entries which are roots of unity. Still, difficulties can arise when the values of different terms in the target polynomial become clustered, and a naive floating point implementation of Ben-Or/Tiwari may still be unstable. It is the choice of a *random* primitive root of unity which avoids this clustering with high probability. We prove modest theoretical bounds to demonstrate this improvement by exhibiting a bound on $\|V^{-1}\|$ which is dependent only on the sparsity (and not on the degree or number of variables in f). Moreover, we show that in practice the improvement in stability is far more dramatic, and discuss why this might be so.

In Section 5, the numerical robustness of our algorithms is demonstrated. We show the effects of varying noise and term clustering and the potential numerical instability it can cause. We demonstrate the effectiveness of our randomization at increasing stability dramatically, with high probability, in such circumstances.

2. PRONY AND BEN-OR/TIWARI'S METHODS FOR EXACT INTERPOLATION

In this section we describe Prony's method for interpolating sums of exponentials and the Ben-Or/Tiwari algorithm for multivariate polynomials. We show that these two algorithms are closely related.

2.1 Prony's method

Prony's method seeks to interpolate a univariate function $F(x)$ as a sum of exponential functions. That is, it tries to determine $c_1,\ldots,c_t \in \mathbb{C}$ and $\mu_1,\ldots,\mu_t \in \mathbb{C}$ such that

$$F(x) = \sum_{j=1}^{t} c_j e^{\mu_j x} \text{ with } c_j \neq 0. \tag{2.1}$$

Since there are $2t$ unknowns, one would expect to need a system of at least the same number of equations in order to determine these unknowns. If $b_j = e^{\mu_j}$, by evaluating $F(0)$, $F(1)$, $\ldots$,$F(2t-1)$ we can obtain a non-linear system of $2t$ equations relating the $2t$ variables $\mu_1,\ldots,\mu_t$, $c_1,\ldots,c_t$. Prony's method solves this non-linear system by converting it into a problem of root finding for a single, univariate polynomial, and the solving of (structured) linear equations. Let $\Lambda(z)$ be the monic polynomial having the b_j's as zeros:

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

It is straightforward to derive that $\lambda_0,\ldots,\lambda_{t-1}$ satisfy

$$\begin{bmatrix} F(0) & F(1) & \ldots & F(t-1) \\ F(1) & F(2) & \ldots & F(t) \\ \vdots & \vdots & \ddots & \vdots \\ F(t-1) & F(t) & \ldots & F(2t-2) \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} F(t) \\ F(t+1) \\ \vdots \\ F(2t-1) \end{bmatrix}.$$

After solving the above system for coefficients $\lambda_0,\ldots,\lambda_{t-1}$ of $\Lambda(z)$, $b_1,\ldots,b_t$ (hence also $\mu_1,\ldots,\mu_t$) can be determined by finding the roots of $\Lambda(z)$. The remaining unknown coefficients $c_1,\ldots,c_t$ can then be computed by solving the transposed Vandermonde system:

$$\begin{bmatrix} 1 & \cdots & 1 \\ b_1 & \cdots & b_t \\ \vdots & \ddots & \vdots \\ b_1^{t-1} & \ldots & b_t^{t-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} F(0) \\ F(1) \\ \vdots \\ F(t-1) \end{bmatrix} \tag{2.2}$$

2.2 The Ben-Or/Tiwari method

For a given black-box polynomial f with n variables, in exact arithmetic the Ben-Or/Tiwari method finds coefficients c_j and integer exponents $(d_{j_1}, \ldots, d_{j_n})$ such that

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{d_{j1}} \cdots x_n^{d_{jn}}, \qquad (2.3)$$

for $1 \le j \le t$, with $c_1, \ldots, c_t \ne 0$. Let $\beta_j(x_1, \ldots, x_n) = x_1^{d_{j1}} \cdots x_n^{d_{jn}}$ be the j-th term in f, and

$$b_j = \beta_j(\omega_1, \ldots, \omega_n) = \omega_1^{d_{j1}} \cdots \omega_n^{d_{jn}}$$

with $\omega_1, \ldots, \omega_n \in \mathsf{D}$ pairwise relatively prime, where D is a unique factorization domain. Note that $b_j^k = \beta_j(\omega_1^k, \ldots, \omega_n^k)$ for any power k.

If we set $F(k) = f(\omega_1^k, \ldots, \omega_n^k)$, then the Ben-Or/Tiwari algorithm solves for the b_j and the c_j, much as is done in Prony's method. That is, it finds a generating polynomial $\Lambda(z)$, determines its roots, and then solves a Vandermonde system. In addition, once the individual terms b_j are found as the roots of $\Lambda(z) = 0$, the exponents $(d_{j_1}, \ldots, d_{j_n})$ are determined by looking at their unique factorizations: $b_j = \omega_1^{d_{j1}} \omega_2^{d_{j2}} \ldots, \omega_n^{d_{jn}}$, which can be easily achieved through repeated division of b_j by $\omega_1, \ldots, \omega_n$.

We note that, as an alternative which we employ in the our algorithms in the next section, we could also choose $\omega_1, \ldots, \omega_n$ to be roots of unity of relatively prime order (i.e., $\omega_i^{p_i} = 1$, $\omega_i^j \ne 1$ for $1 \le j < p_i$, and $p_i > \deg_{x_i} f$, $\gcd(p_i, p_j) = 1$ whenever $i \ne j$). Then, given b_j, we can again uniquely determine $(d_{j_1}, \ldots, d_{j_n})$.

3. NUMERICAL METHODS FOR SPARSE INTERPOLATION

In this section we present two methods for black-box interpolation of sparse multivariate polynomials in floating-point arithmetic. One is a straightforward modification of the Ben-Or/Tiwari algorithm, while the other method makes use of a reformulation of Prony's method using generalized eigenvalues [12].

3.1 A Modified Numeric Ben-Or/Tiwari Algorithm

If the steps of the Ben-Or/Tiwari algorithm are directly implemented in floating-point arithmetic, then difficulties arise at various stages of the computation. The first difficulty is that the subroutines employed for linear system solving and root finding in the Ben-Or/Tiwari algorithm need to use floating-point arithmetic. Hence, they may encounter significant numerical errors. The second difficulty is that we can no longer employ exact divisions to recover the exponents of each variable in a multivariate term.

While it is well-known that Hankel and Vandermonde matrices can often be ill-conditioned, this is particularly true when the input is *real*, as it is in the Ben-Or/Tiwari algorithm. For example, when all the coefficients of f are positive, the Hankel matrix in Prony's algorithm is positive definite, and its condition number may grow exponentially with the dimension [1].

Instead, our modified numeric Ben-Or/Tiwari algorithm uses evaluation points at appropriate primitive (complex) roots of unity. This turns out to reduce our conditioning problems with the encountered Hankel and Vandermonde systems (see Subsection 4.1), and has the added advantage that it allows us to recover the exponent of each variable in a multivariate term. We also assume that we have an upper bound on the degree of each variable in f. Let f be as in (2.3). Choose $p_1, \ldots, p_n \in \mathbb{Z}_{>0}$ pairwise relatively prime such that $p_k > \deg_{x_k} f$ for $1 \le k \le n$. The complex root of unity $\omega_k = \exp(2\pi i/p_k)$ has order p_k, which is relatively prime to the product of other p_j's. Now consider the following sequence for interpolation:

$$\alpha_s = f(\omega_1^s, \omega_2^s, \ldots, \omega_n^s) \quad \text{for } 0 \le s \le 2t-1, \qquad (3.1)$$

with $\omega_k = \exp(2\pi i/p_k)$. Setting $m = p_1 \cdots p_n$ and $\omega = \exp(2\pi i/m)$, we see $\omega_k = \omega^{m/p_k}$ for $1 \le k \le n$.

Each term $\beta_j(x_1, \ldots, x_n)$ in f is evaluated as $\beta_j(\omega_1, \ \ldots, \ \omega_n) = \omega^{d_j}$, and each d_j can be computed by rounding $\log_\omega(\omega^{d_j}) = \log_\omega(\beta_j(\omega_1, \ \ldots, \ \omega_n))$ to the nearest integer. Note that this logarithm is defined modulo $m = p_1 \cdots p_n$. Because the p_k's are relatively prime, the exponent for each variable $(d_{j_1}, \ldots, d_{j_n}) \in \mathbb{Z}_{>0}^n$ can be uniquely determined by the reverse steps of the Chinese remainder algorithm (see, e.g., [11]). That is, we have $d_j \equiv d_{j_k} \bmod p_k$ for $1 \le k \le n$ and

$$d_j = d_{j_1} \cdot \left(\frac{m}{p_1}\right) + \cdots + d_{j_n} \cdot \left(\frac{m}{p_n}\right). \qquad (3.2)$$

We present our modified Ben-Or/Tiwari algorithm.

Algorithm: `ModBOTInterp`

Input: ▸ a floating-point black box f: the target polynomial;

▸ t, the number of terms in f;

▸ $D_1, \ldots, D_n$: $D_k \ge \deg(f_{x_k})$.

Output: ▸ c_j and $(d_{j_1}, \ldots, d_{j_n})$ for $1 \le j \le t$ such that $\sum_{j=1}^{t} c_j x_1^{d_{j1}} \cdots x_n^{d_{jn}}$ approximately interpolates f.

(1) [*Evaluate f at roots of unity.*]

(1.1) Choose $p_1, \ldots, p_n$ pairwise relatively prime and $p_j > D_j$. Let $m = p_1 \cdots p_n$, $\omega = \exp(2\pi i/m)$, and $\omega_k = \exp(2\pi i/p_k) = \omega^{m/p_k}$.

(1.2) Evaluate $\alpha_s = f(\omega_1^s, \omega_2^s, \ldots, \omega_n^s)$, $0 \le s \le 2t-1$.

(2) [*Recover $(d_{j_1}, \ldots, d_{j_n})$.*]

(2.1) Solve the associated Hankel system

$$\underbrace{\begin{bmatrix} \alpha_0 & \ldots & \alpha_{t-1} \\ \alpha_1 & \ldots & \alpha_t \\ \vdots & \ddots & \vdots \\ \alpha_{t-1} & \ldots & \alpha_{2t-2} \end{bmatrix}}_{H_0} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} \alpha_t \\ \alpha_{t+1} \\ \vdots \\ \lambda_{2t-1} \end{bmatrix}. \qquad (3.3)$$

(2.2) Find roots $b_1, \ldots, b_t$ for $\Lambda(z) = z^t + \lambda_{t-1} z^{t-1} + \cdots + \lambda_0 = 0$.

(2.3) Recover $(d_{j_1}, \ldots, d_{j_n})$ from $d_j = \text{round}(\log_\omega b_j)$ via (3.2) by the reverse Chinese remainder algorithm.

(3) [*Compute the coefficients c_j.*]

Solve an associated Vandermonde system: (now $\beta_j = x_1^{d_{j1}} \cdots x_n^{d_{jn}}$ are recovered, $\tilde{b}_j$ can be either b_j or

$\beta_j(\omega_1, \ldots, \omega_n)$)

$$\begin{bmatrix} 1 & \cdots & 1 \\ \tilde{b}_1 & \cdots & \tilde{b}_t \\ \vdots & \ddots & \vdots \\ \tilde{b}_1^{t-1} & \cdots & \tilde{b}_t^{t-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{t-1} \end{bmatrix}. \quad (3.4)$$

3.2 Interpolation via Generalized Eigenvalues

We now give another algorithm which avoids the solving for a Hankel system and the subsequent root finding. This is done by using a reformulation of Prony's method as a generalized eigenvalue problem, following [12]. As before, consider f as in (2.3) evaluated at primitive roots of unity as in (3.1). Define Hankel systems

$$H_0 = \begin{bmatrix} \alpha_0 & \cdots & \alpha_{t-1} \\ \vdots & \ddots & \vdots \\ \alpha_{t-1} & \cdots & \alpha_{2t-2} \end{bmatrix}, \text{ and } H_1 = \begin{bmatrix} \alpha_1 & \ldots & \alpha_t \\ \vdots & \ddots & \vdots \\ \alpha_t & \ldots & \alpha_{2t-1} \end{bmatrix}.$$

Let $b_j = \beta_j(\omega_1, \ldots, \omega_n)$. If we set $Y = \operatorname{diag}(b_1, \ldots, b_t)$, $D = \operatorname{diag}(c_1, \ldots, c_t)$, and

$$V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \ldots & b_t^{t-1} \end{bmatrix}, \quad (3.5)$$

then $H_0 = VDV^{\mathrm{Tr}}$, $H_1 = VDYV^{\mathrm{Tr}}$. The solutions for $z \in \mathbb{C}$ in the generalized eigenvalue problem

$$(H_1 - zH_0)v = 0, \quad (3.6)$$

for a generalized eigenvector $v \in \mathbb{C}^{t\times 1}$, are $b_j = \beta_j(\omega_1, \ldots \omega_n)$ for $1 \le j \le t$. If $\omega_1, \ldots, \omega_n$ are chosen as described in the previous subsection, we can also recover the multivariate terms $\beta_j(x_1, \ldots, x_n)$ through the same method. To complete the interpolation, we need to compute the coefficients, which requires the solving of a transposed Vandermonde system over a numerical domain. The cost of the entire procedure is bounded by the cost of solving the generalized eigenvalue problem, which can be accomplished in a numerically stable manner with $O(t^3)$ operations using the QZ algorithm (see, e.g., [13]).

The algorithm for sparse interpolation using generalized eigenvalues is the same as `ModBOTInterp` with the exception of step (2), which we present here.

Algorithm: `GEVInterp` (Step 2)

```
(2) [Recover $(d_{j_1}, \ldots, d_{j_n})$.]
    (2.1) Find solutions $b_1, \ldots, b_t$ for $z$ in the generalized
          eigenvalue problem $H_1 v = zH_0 v$.
    (2.2) Recover $(d_{j_1}, \ldots, d_{j_n})$ from $d_j = \operatorname{round}(\log_\omega b_j)$
          via (3.2) by the reverse Chinese remainder algorithm.
```

4. SENSITIVITY ANALYSIS AND RANDOMIZED CONDITIONING

In this section we focus on the numerical accuracy of the sparse interpolation algorithms presented in the previous section. We also introduce a new randomized technique which will dramatically improve the expected numerical stability of our algorithms.

Both the Ben-Or/Tiwari algorithm and the generalized eigenvalue method first recover the polynomial support. That is, they determine which terms are non-zero in the target polynomial. We look at the numerical sensitivity of both techniques, and link it directly to the choice of sparsity t and the condition of the associated Vandermonde system V. After the non-zero terms are determined, both methods need to separate the exponents for different variables and recover the corresponding coefficients, again via the Vandermonde system V. Finally, we show how randomization of the choice of evaluation points can substantially improve the conditioning of V, and hence improve the stability of the entire interpolation process.

4.1 Conditioning of associated Hankel system

Consider the modified numeric Ben-Or/Tiwari algorithm described in Subsection 3.1. In order to determine coefficients for the polynomial $\Lambda(z) = z^t + \lambda_{t-1}z^{t-1} + \cdots + \lambda_0$, we need to solve a Hankel system as in (3.3). In general, if the polynomial f is evaluated at powers of real values, the difference between the sizes of varying powers will contribute detrimentally to the conditioning of the Hankel system. This problem of scaling is avoided in our method, since our H_0 is formed from the evaluations on the unit circle. The following proposition links the condition of H_0 directly to the condition of V and to the size of the reciprocals $1/|c_j|$ of the coefficients c_j in the target polynomial (for $1 \le j \le t$).

PROPOSITION 4.1.

- $\|H_0^{-1}\| \ge \dfrac{1}{t} \max\limits_j \dfrac{1}{|c_j|}$, *and* $\|H_0^{-1}\| \ge \dfrac{\|V^{-1}\|^2}{\sum_{1\le j\le t}|c_j|}$.
- $\|H_0^{-1}\| \le \|V^{-1}\|^2 \cdot \max\limits_j \dfrac{1}{|c_j|}$.

Thus, bounds for $\|H_0^{-1}\|$ involve both the (inverses of) the coefficients of the interpolated polynomial $c_1, \ldots, c_t$ and the condition of the Vandermonde system V. In some sense the coefficients $c_1, \ldots, c_t$ are intrinsic to a problem instance, and having them very small (and hence with large reciprocals) means that we have chosen t too large. The Vandermonde matrix V, on the other hand, is intrinsic to our algorithm, and we will address its conditioning, and methods to improve this conditioning, in the following sections.

4.2 Root finding on the generating polynomial

In our modified numeric Ben-Or/Tiwari algorithm, for recovering non-zero terms in f, we need to find the roots of $\Lambda(z) = 0$. In general, root finding can be very ill-conditioned with respect to perturbations in the coefficients [23].

However, all the roots $b_j = \beta_j(\omega_1, \ldots, \omega_n)$ as (2.3) are on the unit circle by our choice of evaluation points. Using Wilkinson's argument for points on the unit circle, the following theorem shows that the condition can be improved, and related to the separation of the roots $b_1, \ldots, b_t$.

THEOREM 4.1. *For a given polynomial $f(x_1, \ldots, x_n) = \sum_{j=1}^t c_j\beta_j(x_1, \ldots, x_n)$ interpolated on the unit circle, let b_k be a zero of $\Lambda(z)$ and $\tilde{b}_k$ a zero of $\Lambda(z) + \epsilon\Gamma(z)$, then*

$$|b_k - \tilde{b}_k| < \frac{\epsilon \cdot \|\Gamma(z)\|_1}{|\prod_{j\ne k}(b_k - b_j)|} + K\epsilon^2$$

Note that $\epsilon \cdot \|\Gamma(z)\|_1$ is an upper bound for the perturbation of the polynomial $\Lambda(z)$ evaluated on the unit circle, which is also a measure of the size of a perturbation in the solution of the Hankel system (3.3). The value of $|\prod_{j\neq k}(b_k - b_j)|$ is directly related to the condition of the Vandermonde system (3.5), and depends on the distribution of b_j's on the unit circle (see Subsection 4.6).

4.3 Error bounds for generalized eigenvalues

We can further analyze the generalized eigenvalue approach described in Subsection 3.2. In particular, we once again link the sensitivity directly to the condition of V, that is, to $\|V^{-1}\|$, and to the magnitude of the smallest coefficient. Along similar lines to [12], we can prove the following:

THEOREM 4.2. *Assume the generalized eigenvalue problem in* (3.6) *has generalized eigenvalues* $b_1, \ldots, b_t \in \mathbb{C}$ *and corresponding eigenvectors* $v_1, \ldots, v_t \in \mathbb{C}^{\times 1}$. *Consider the perturbed problem*

$$\Big((H_1 + \epsilon\,\widehat{H}_1) - z(H_0 + \epsilon\,\widehat{H}_0)\Big)\, v = 0 \qquad (4.1)$$

for $\epsilon > 0$ *and normalized perturbations* $\widehat{H}_0, \widehat{H}_1 \in \mathbb{C}^{t\times t}$ *with* $\|\widehat{H}_0\| = \|H_0\|$ *and* $\|\widehat{H}_1\| = \|H_1\|$. *Then* (4.1) *has solutions (generalized eigenvalues)* $\widetilde{b}_1, \ldots, \widetilde{b}_t \in \mathbb{C}$, *with*

$$|\widetilde{b}_j - b_j| < \epsilon \cdot \frac{2t^2 \cdot \|(c_1, \ldots, c_t)\|_\infty \cdot \|V^{-1}\|^2}{|c_j|}$$

for $1 \le j \le t$.

4.4 Separation of powers

After computing approximations $\widetilde{b}_1, \ldots, \widetilde{b}_t$ for the term values $b_1, \ldots, b_t$, we still need to consider the precision required for correctly recovering the integer exponents (with respect to $\omega = \exp(2\pi i/m)$) by taking the logarithms of $b_j = \omega^{d_j}$ (with respect to ω), for $1 \le j \le t$, as in (3.2). Since each b_j lies on the unit circle, we really need only consider the argument of $\widetilde{b}_j$ in determining its logarithm with respect to ω (i.e., we normalize $\widetilde{b}_j := \widetilde{b}_j/|\widetilde{b}_j|$).

Two consecutive mth roots of unity on the unit circle are separated by an angle of radian $\frac{2\pi}{m}$, and the distance between these two points is bounded below by twice the sine of half the angle between them. Thus, in order to separate any two such points by rounding one must have the computed values $\widetilde{b}_1, \ldots, \widetilde{b}_t$ of $b_1, \ldots, b_t$ correct to

$$|b_j - \widetilde{b}_j| \le \frac{1}{2}|2\sin(\frac{\pi}{m})| < \frac{\pi}{m}, \qquad \text{and} \qquad m = p_1 \cdots p_n,$$

for $1 \le j \le t$, where $p_k > \deg f_{x_k}$ for $1 \le k \le n$.

We note that π/m is not a particularly demanding bound, and is easily achieved (for fixed precision floating point numbers) when H is well-conditioned, for reasonably size m. In particular, we need only $O(\log m)$ bits correct to effectively identify the non-zero terms in our target sparse polynomial.

4.5 Recovering the coefficients

Once the values of $b_1, \ldots, b_t$, and hence exponents of the non-zero terms, have been determined, it still remains to compute their coefficients $c_1, \ldots, c_t$, which can be done in a number of ways. Most straightforwardly, we can solve the Vandermonde system V in equation (3.4) (Step 3 in algorithm `ModBOTInterp`) to determine the coefficients $c_1, \ldots, c_t$. The main issue in this case is the condition of V, which is not obviously good. We examine this in Subsection 4.6. If the term are determined as general eigenvalues in (3.6) by the QZ algorithm, the computed eigenvectors $v_1, \ldots, v_t$ can be used to reconstruct the coefficients. See [12].

4.6 Condition of the Vandermonde System

While Vandermonde matrices can be poorly conditioned, particularly for real number data [10, 1], our problem will be better behaved. First, all our *nodes* $(b_1, \ldots, b_t)$ lie on the unit circle. For example, in the case of $t \times t$ Vandermonde matrices as in (3.5), the 2-norm condition number has the optimal value of 1 when the nodes are all the mth roots of unity [9, example 6.4]. A slightly less uniform sequence of nodes is studied in [4], where the nodes are chosen according to a Van der Corput sequence, to achieve a 2-norm condition number of $\sqrt{2t}$ of a $t \times t$ Vandermonde matrix (for any t). Both results suggest the possibility of well-conditioning of complex Vandermonde matrices, especially when the spacing of the nodes is relatively regular.

When $b_1, \ldots, b_t$ are all mth roots of unity (for $m \ge t$) we have the following bounds for $\|V^{-1}\|$ from [9]:

$$\max_{1\le k\le t} \frac{1/\sqrt{t}}{\prod_{j\neq k}|b_j - b_k|} < \|V^{-1}\| \le \max_{1\le k\le t} \frac{2^{t-1}\sqrt{t}}{\prod_{j\neq k}|b_j - b_k|}. \qquad (4.2)$$

These bounds may still be dependent exponentially on t and m, particularly if $b_1, \ldots, b_t$ are clustered. In the worst case, we find

$$\|V^{-1}\| > \frac{1}{\sqrt{t}} \cdot \left(\frac{m}{2\pi(t-1)}\right)^{t-1}.$$

For a more general discussion, see [2].

This indicates that as m, as well as t, gets larger, the condition of V can get dramatically worse, particularly if m is large. As an example, if $m = 1000$ (which might occur with a tri-variate polynomial of degree 10 in each variable) with 10 terms, V could have condition number greater than 10^{16}. This is quite worrisome, as m is proportional to the number of possible terms in the *dense* representation, and in particular is exponential in the number of variables n. Moreover, the bound seems surprising bad, as one might hope for better conditioning as m gets larger, when there is greater "opportunity" for node distribution. This is addressed in the next subsection.

4.7 Randomized reconditioning

We now demonstrate how a small amount of randomization ameliorates the problem of potential ill-conditioning in the Vandermonde matrix dramatically.

Suppose $p_1, \ldots, p_n$ are distinct primes, $p_k > \deg_{x_k} f$, and $\omega = \exp(2\pi i/m)$ for $m = p_1 \cdots p_n$. If the target polynomial f is evaluated at powers of $(\omega_1, \ldots, \omega_n)$ for $\omega_k = \omega^{m/p_k}$ (cf. Subsection 3.1), the distribution of term values on the unit circle is fixed because the polynomial terms are fixed. We may well end up in a situation where the Vandermonde matrix is ill-conditioned as discussed above. To eliminate this possibility with high probability, we will introduce a randomization as follows. Instead of using $\omega_k = \omega^{m/p_k} = \exp(2\pi i/p_k)$, the principle p_kth primitive root of unity, we choose a random p_kth primitive root of unity, $\omega_k = \exp(2\pi i r_k/p_k)$, for some $1 \le r_k < p_k$. Equivalently, we choose a single r with $r \equiv r_k \bmod p_k$, $1 \le r < m$, so that $\omega_k = \omega^{mr/p_k}$ (see (3.2)).

To analyze the distribution of term values, instead of the multivariate $f = \sum_{j=1}^{t} c_j x_1^{d_{j1}} \cdots x_n^{d_{jn}}$, we equivalently consider the univariate $\widetilde{f}(x) = \sum_{j=1}^{t} c_j x^{d_j}$ where $d_j = d_{j1}(m/p_1) + \cdots + d_{jn}(m/p_n)$ (cf. Subsection 3.1). The term values are $\omega^{d_1}, \ldots, \omega^{d_t}$, and the stability of recovering the d_js depends upon the condition of the Vandermonde matrix V on nodes $\omega^{d_1}, \ldots, \omega^{d_t}$. This is inversely related to the product of differences $|\omega^{d_j} - \omega^{d_k}|$ for $1 \le j < k \le t$ as described in (4.2).

For each interpolation attempt, we pick an r uniformly and randomly from $1 \ldots m-1$. The condition number of the new Vandermonde matrix $\widetilde{V}$, with nodes $b_j = \omega^{rd_j}$ for $1 \le j \le t$ is now inversely related to the differences $|rd_j - rd_k| = r|d_j - d_k| \bmod m$. In some sense we are multiplying each difference by (the same) random number r, hopefully minimizing the chance that there are many small differences. Once the Hankel matrix H_0 is constructed, we can check the conditioning, and if it is poor, we can choose another random r and repeat the process. The next theorem, and especially the following discussion, gives us some assurance that we never have to do this very often.

THEOREM 4.3. *Let $p_1, \ldots, p_n > t^2/2$ be distinct primes as above, with $m = p_1 \ldots p_t$ and $\omega = \exp(2\pi i/m)$. Let $0 \le d_1, \ldots, d_t \le m-1$ be distinct. Suppose r is chosen uniformly and randomly from $1, \ldots, m-1$ and let $\widetilde{V}$ be the Vandermonde matrix on nodes $b_i = \omega^{rd_i}$. Then, with probability at least $1/2$,*

$$\|\widetilde{V}^{-1}\| \le \sqrt{t}\left(\frac{2t^2}{\pi}\right)^{t-1}.$$

PROOF. For $1 \le j < k \le t$, let $\Delta_{jk} = |d_j - d_j| \bmod m$. There are at most $\binom{t}{2} \le t^2/2$ distinct values of Δ_{jk}. Fix $\ell := m/t^2$, and let $c \in \{1, \ldots, \ell\}$. For each Δ_{jk} there is at most one $r \in \mathbb{Z}_m$ such that $r\Delta_{jk} \equiv c \bmod m$. Thus, there are at most $t^2/2 \cdot \ell = m/2$ values of r such that for any Δ_{jk} and any $c \in \{1, \ldots, \ell\}$ we have $r\Delta_{jk} \not\equiv c \bmod m$.

Assume that the chosen r is such that $r\Delta_{jk} \not\equiv 1, \ldots, \ell$. Then for all $1 \le j < k \le t$ we have

$$|\omega^{rd_j} - \omega^{rd_k}| = |\omega^{r(d_j - d_k)} - 1| \ge |\omega^{m/t^2} - 1| = |e^{2\pi i/t^2} - 1|$$
$$= 2\sin(\pi/t^2) \ge 2\left(\frac{\pi}{t^2} - \frac{\pi^3}{6t^6}\right) \ge \frac{\pi}{t^2}.$$

Using (4.2) this yields

$$\|\widetilde{V}^{-1}\| \le \sqrt{t} \cdot \max_{1 \le k \le t} \frac{2^{t-1}}{\prod_{j \ne k} |\omega^{d_j} - \omega^{d_k}|} \le \sqrt{t}\left(\frac{2t^2}{\pi}\right)^{t-1}. \quad \square$$

This eliminates any dependence upon m, and hence any dependence upon the size of the dense representation of the polynomial. However, we believe this is probably still far from optimal. Considerable cancellation might be expected in the sizes of the entries of V^{-1}, though bounding these formally seems difficult.

We have conducted intensive numerical experiments which suggest that the bound (in terms of t) on the condition number is *much* lower. For the experiments, we assume the worst case before the randomization, with nodes clustered as $\omega, \omega^2, \ldots, \omega^t$. We assume that we are in the univariate case, where m is prime. Neither of these assumptions have any substantial effect on the results of the experiments. We ran the experiment 100 times for each value of m and sparsity percentage t, and report the median condition number.

n \%t	0.1	1	2	5	10
101	2.21	2.19	3.64	9.91	26.9
211	2.25	3.69	6.95	25.2	69.4
401	2.42	6.63	15.5	75.3	137
1009	2.25	23.1	64.3	205	1.53e3
10007	22.6	1.10e3	1.85e3	2.79e3	3.36e4

Figure 2. Median condition number of V; t a percentage of m.

The actual condition number appears to be remarkably small, and a (perhaps naive) conjecture might be that it is linear in t. In any case, the condition number is low, and in practice this makes for a very stable recovery process from V. This will be fully validated in the upcoming Section 5.

Finally, we note that the techniques in Theorem 4.3 are easily extended to show that *all* leading minors of H_0 are similarly well-conditioned. This leads us to a possible way to identify the sparsity t of f by simply computing $\alpha_0, \alpha_1, \ldots$ (at a random root of unity) until H_0 becomes ill-conditioned. With high probability we should identified t. Again, numerical evidence suggests much better expected conditioning of the leading minors of H_0, and hence quite a strong criteria for identifying the sparsity of f.

5. EXPERIMENTS

For our experiments we have tested both the modified Ben-Or/Tiwari and the generalized eigenvalue methods. Our computational environment is the computer algebra system Maple 10 using hardware arithmetic (IEEE floating point).

Our algorithms interpolate multivariate polynomials. However, during the computation, a multivariate polynomial is regarded as a univariate polynomial on the unit circle through the (reverse) steps of the Chinese remainder algorithm (essentially variable substitution; see Subsection 3.1). Therefore, we concentrate our tests on sparse univariate examples. Since the stability of our algorithms is directly dependent upon the condition of the underlying Vandermonde system, we arrange our tests by the condition of this system. We look at the case when it is well conditioned, and when it starts off poorly conditioned, and examine how randomness generally avoids the poorly conditioned case.

Term values evenly distributed on the unit circle

This is the best and "easiest" case, wherein the Vandermonde system is well-conditioned. We randomly generated 100 univariate polynomials, with the number of terms between 10 and 50, and roughly evenly distributed the term degrees between 0 and 1000. When the non-zero coefficients are randomly distributed between -1 and 1, the following table reveals the performance of both interpolation algorithms. Robustness is evaluated as the 2-norm distance between the interpolation result and the target polynomial. For this we list both the mean and median for the performance of the interpolation of these 100 random polynomials.

Random noise		Ben-Or/Tiwari
0	Mean	$.120505981901393e-11$
	Median	$.133841077792715e-11$
$\pm 10^{-12} \sim 10^{-9}$	Mean	$.581398079681344e-9$
	Median	$.582075115365304e-9$
$\pm 10^{-9} \sim 10^{-6}$	Mean	$.570763804647327e-6$
	Median	$.569467774610552e-6$
$\pm 10^{-6} \sim 10^{-3}$	Mean	$.577975930552999e-3$
	Median	$.583391747553225e-3$

Random noise		Generalized Eigenvalue
0	Mean	$.120594593261080e-11$
	Median	$.133636116920920e-11$
$\pm 10^{-12} \sim 10^{-9}$	Mean	$.581398474087412e-9$
	Median	$.582077799081834e-9$
$\pm 10^{-9} \sim 10^{-6}$	Mean	$.570763804248465e-6$
	Median	$.569467779291746e-6$
$\pm 10^{-6} \sim 10^{-3}$	Mean	$.577975930554979e-3$
	Median	$.583391747541653e-3$

As the above table illustrates, well-conditioned Vandermonde systems give excellent interpolation results, and the amount of the input noise is proportional to the error in the output. We also note that there is little gain in using the generalized eigenvalue algorithm in this case (and indeed, it is considerably slower). This should not be particularly surprising given Proposition 4.1.

Clustered term values

For a second experiment, we interpolate polynomials with terms x^0, x^3, x^6, $x^{\lfloor \frac{994}{t-2} \rfloor + 6}$, $x^{\lfloor \frac{2\cdot 994}{t-2} \rfloor + 6}$, $\ldots$, $x^{\lfloor \frac{(t-3)\cdot 994}{t-2} \rfloor + 6}$ at powers of $\omega = \exp(2\pi/1000)$, in which terms x^0, x^3, and x^6 are close to each other.

In our test, we encounter a (numerically) singular system when the (random) noise is in the range of $\pm 10^{-9} \sim 10^{-6}$. We list the mean and median of all the non-singular results. We also note that 11 of the 99 non-singular results are of distance less or around .0001 from the target polynomial.

Random noise		Ben-Or/Tiwari
0	Mean	$.136907950785253e-9$
	Median	$.101038098751213e-9$
$\pm 10^{-12} \sim 10^{-9}$	Mean	$.118191438770386e-6$
	Median	$.700404450937545e-7$
$\pm 10^{-9} \sim 10^{-6}$	Mean	$.713728504313218$
	Median	$.641238385320081$
$\pm 10^{-6} \sim 10^{-3}$	Mean	$.843675339146120$
	Median	$.754345867272459$

Random noise		Generalized Eigenvalue
0	Mean	$.137847635557337e-9$
	Median	$.105150252450990e-9$
$\pm 10^{-12} \sim 10^{-9}$	Mean	$.118192220230628e-6$
	Median	$.700455264514340e-7$
$\pm 10^{-9} \sim 10^{-6}$	Mean	$.710891838764534$
	Median	$.641238385320072$
$\pm 10^{-6} \sim 10^{-3}$	Mean	$.843662476563188$
	Median	$.754345867272456$

In this experiment, good interpolation results may still be obtained for Vandermonde systems with a few nodes clustered on the unit circle. However, such results tend to be very sensitive to noise.

Effective randomization to ameliorate term value accumulation

In our third set of tests we consider the effect of randomization to improve the numerical conditioning of the interpolation problems. Here we consider polynomial interpolation associated with a Vandermonde system with 3 terms clustered. That is, the 100 random univariate polynomials, with the number of terms between 10 and 50, all have terms x^0, x, and x^2. All other remaining term are roughly evenly distributed the term degrees between 3 and 1000.

We interpolate the polynomial at powers of $\exp(2\pi i/1009)$. As the following table shows, the clustering greatly affects the effectiveness of both interpolation algorithms.

Random noise		Ben-Or/Tiwari
0	Mean	92.8019727202980
	Median	73.4823536193264

Random noise		Generalized Eigenvalue
0	Mean	92.8019727200298
	Median	73.4823536202312

However, after randomization, that is, instead of interpolating at powers of $\omega = \exp(2\pi i/1000)$, we interpolate at powers of $\omega = \exp(2r\pi i/1009)$ for a random $r \in \{1, \ldots, 1008\}$, for the same set of random polynomials, we have the following results.

Random noise		Ben-Or/Tiwari
0	Mean	27.9983307662379
	Median	$.242793778266858e-7$
$\pm 10^{-12} \sim 10^{-9}$	Mean	$.869652877288326$
	Median	$.170781612648532e-6$

Random noise		Generalized Eigenvalue
0	Mean	30.6022221605261
	Median	$.242734723141759e-7$
$\pm 10^{-12} \sim 10^{-9}$	Mean	$.863424321492980$
	Median	$.170790199598136e-6$

Notice that, although we do not obtain good interpolation results each time, the error at the median is generally quite good (a terribly conditioned randomization can affect the mean dramatically). In practice, upon obtaining an ill-conditioned result, we would simply re-randomize and repeat the computation. Theorem 4.3 provides assurances that we should never have to restart this many times before achieving a well-conditioned Vandermonde matrix, and hence obtain reliable results.

The full Maple code along with a broader range of experiments (including the examples mentioned in [22], can be found at the web site:
`http://www.scg.uwaterloo.ca/~ws2lee/issac06-interp`.

Acknowledgments

We thank Erich Kaltofen for his encouragement and comments, Bernhard Beckermann for his help (particularly on Section 4.1), and Bernard Mourrain. We would also like to thank Annie Cuyt and Brigitte Verdonk for pointing us to recent related works.

The authors would like to thank NSERC and MITACS Canada for their support of this work. Wen-shin Lee would also like to thank the University of Antwerp (visiting postdoc grant 1015) and the FWO (research grant on *Rational Modeling*).

6. REFERENCES

[1] B. Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numeriche Mathematik*, 85:553–577, 2000.

[2] B. Beckermann, G. Golub, and G. Labahn. On the numerical condition of a generalized Hankel eigenvalue problem. *submitted to Numerische Matematik*, 2005.

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

[4] A. Córdova, W. Gautschi, and S. Ruscheweyh. Vandermonde matrices on the circle: spectral properties and conditioning. *Numerische Mathematik*, 57:577–591, 1990.

[5] R.M. Corless, M. Giesbrecht, I. Kotsireas, and S.M. Watt. Numerical implicitization of parametric hypersurfaces with linear algebra. In E. Roanes-Lozano, editor, *Artificial Intelligence and Symbolic Computation: International Conference AISC 2000*, pages 174–183, Heidelberg, Germany, 2001. Springer Verlag.

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

[7] S. Gao, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. In *ISSAC 2004 Proc. 2004 Internat. Symp. Symbolic Algebraic Comput.*, pages 167–174, 2004.

[8] M. Gasca and T. Sauer. On the history of multivariate polynomial interpolation. *J. Computational and Applied Mathematics*, 122:23–35, 2000.

[9] W. Gautschi. Norm estimates for inverses of Vandermonde matrices. *Numerische Mathematik*, 23:337–347, 1975.

[10] W. Gautschi and G. Inglese. Lower bounds for the condition numbers of Vandermonde matrices. *Numerische Mathematik*, 52:241–250, 1988.

[11] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publ., Boston, Massachusetts, USA, 1992.

[12] G. H. Golub, P. Milanfar, and J. Varah. A stable numerical method for inverting shape from moments. *SIAM J. Sci. Comput.*, 21(4):1222–1243, 1999.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

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

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

[16] E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symbolic Comput.*, 9(3):301–320, 1990.

[17] L. Kronecker. *Über einige Interpolationsformeln für ganze Funktionen mehrerer Variabeln, Lecture at the academy of sciences, December 21, 1865*, volume H. Hensel (Ed.), L. Kroneckers Werke, Vol. I. Teubner, Stuttgart, 1895. reprinted by Chelsea, New York, 1968.

[18] R. Lorentz. Multivariate Hermite interpolation by algebaic polynomials: a survey. *J. Computational and Applied Mathematics*, 122:167–201, 2000.

[19] Y. Mansour. Randomized approximation and interpolation of sparse polynomials. *SIAM Journal on Computing*, 24(2):357–368, 1995.

[20] P. Milanfar, G. C. Verghese, W. C. Karl, and A. S. Wilsky. Reconstructing polygons from moments with connections to array processing. *IEEE Trans. Signal Processing*, 43(2):432–443, 1995.

[21] Baron de Prony, Gaspard-Clair-François-Marie Riche. Essai expérimental et analytique sur les lois de la Dilatabilité des fluides élastique et sur celles de la Force expansive de la vapeur de l'eau et de la vapeur de l'alkool, à différentes températures. *J. de l'École Polytechnique*, 1:24–76, 1795.

[22] A. Sommese, J. Verschelde, and C. Wampler. Numerical factorization of multivariate complex polynomials. *Theoretical Computer Science*, 315(2–3):651–669, 2004.

[23] J. H. Wilkinson. *Rounding errors in algebraic processes*. Prentice-Hall, Englewood Cliffs, N.J., 1963.

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

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

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