PROBABILISTICALLY STABLE NUMERICAL SPARSE POLYNOMIAL INTERPOLATION

MARK GIESBRECHT, GEORGE LABAHN, AND WEN-SHIN LEE

Abstract. We consider the problem of sparse interpolation of a 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 values are represented in standard, fixed-precision, floating-point arithmetic. By interpolating the black box evaluated at *random* primitive roots of unity, we give an efficient and numerically robust solution with high probability. We outline the numerical stability of our algorithm, as well as the expected conditioning achieved through randomization. Finally, we demonstrate the effectiveness of our techniques through numerical experiments.

1. Introduction

When computing with multivariate polynomials and polynomial systems, it is often effective and even necessary to work with an implicit representation. Computationally, a black box for a polynomial is a procedure that, for any given input, outputs the polynomial evaluated at that input. Black boxes may also represent approximate polynomials, where the coefficients may have errors or noise. In such cases the evaluations of the black box are expected to have errors as well.

In this paper we demonstrate a probabilistically robust numerical algorithm for the sparse interpolation of 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,

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

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

$$\alpha_1 = f(\nu_1),\ \alpha_2 = f(\nu_2),\ \ldots, \alpha_\kappa = f(\nu_\kappa),$$

at our chosen points $\nu_1, \nu_2, \ldots \nu_\kappa \in \mathbb{C}^n$, where $\kappa = O(t)$, we seek to determine coefficients $c_1, \ldots, c_t \in \mathbb{C}$ and 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 our algorithm to be numerically robust: if evaluations $\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_\kappa$ are relatively close to their true values, we want the coefficients $\widetilde{c}_1, \ldots, \widetilde{c}_t \in \mathbb{C}$ also be relatively close to their values.

2000 *Mathematics Subject Classification.* Primary 68W30, 65D10.

Key words and phrases. Symbolic-numeric computing, multivariate interpolation.

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

Dagstuhl Seminar Proceedings 06271
Challenges in Symbolic Computation Software
http://drops.dagstuhl.de/opus/volltexte/2006/775

The best known interpolation methods that are sensitive to the sparsity of the target polynomial are the algorithms of Ben-Or/Tiwari [3] and of Zippel [19]. Although both approaches have been generalized and improved (see [20, 11, 10, 18]), they all depend upon exact arithmetic. With recent advances in approximate polynomial computation, we are led to investigate sparse interpolation in an approximate setting.

The problem of multivariate polynomial interpolation is not new, with early work going back at least to Kronecker [13, 6]. More recently there has been much activity on the topic, of both an algorithmic and mathematical nature [14]. To our knowledge, none of the previous numerical work has considered the problems of identifying the (sparse) support and sparse multivariate interpolation. On the other hand, sparsity is considered in a different, bit-complexity model, using arbitrary precision arithmetic by Mansour [15], 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).

In Section 2, we describe Prony's algorithm [16] for interpolating a sum of exponential functions, which is very similar to the sparse polynomial interpolation of Ben-Or and Tiwari [3]. Then we adapt the Ben-Or/Tiwari method to floating-point arithmetic and identify the numerical difficulties.

In Section 3, we outline the numerical behaviour of our algorithm and sensitivity of the underlying problems. We show that the stability of our algorithm is governed by $\|V^{-1}\|^2/\min|c_j|$, where V is a Vandermonde matrix of the non-zero terms in the polynomial evaluated at the sample points. The coefficients $c_1,\ldots,c_t$ are intrinsic to the problem, and 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 is the use of evaluation points at *random* roots of unity, which allows reconstructing the multivariate exponents by the Chinese remainder algorithm and adds considerable stability by avoiding large variations in magnitude when evaluating polynomials of high degree. In particular, the associated Vandermonde matrix V will have entries which are roots of unity. Still, difficulties can arise when different term values in the polynomial are clustered, and a naive floating point implementation of Ben-Or/Tiwari may be unstable. It is the choice of a *random* primitive root of unity which removes this clustering with high probability.

In Section 4, we experiment with the effects of varying noise and term clustering and the potential numerical instability it can cause. We demonstrate the effectiveness of randomization at increasing stability dramatically, with high probability, in such circumstances.

2. Prony and Ben-Or/Tiwari's methods for interpolation

We describe Prony's method for interpolating a sum of exponentials and the related Ben-Or/Tiwari algorithm for multivariate polynomials. Then we present our modification of the Ben-Or/Tiwari algorithm in floating-point arithmetic.

2.1. **Prony's method.**

Prony's method interpolates a univariate function as a sum of exponentials: it determines $c_1,\ldots,c_t \in \mathbb{C}$ and $\mu_1,\ldots,\mu_t \in \mathbb{C}$ such that

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

If $b_j = e^{\mu_j}$, by evaluating $F(0)$, $F(1)$, ...,$F(2t-1)$ we 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 the root finding of a single, univariate polynomial, and the solving of linear equations. Let $\Lambda(z)$ be the monic polynomial having the b_j's as zeros:

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

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

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

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 for $1 \le j \le t$,

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{t} c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}, \tag{2.3}$$

with $c_1, \ldots, c_t \neq 0$. Let $\beta_j(x_1, \ldots, x_n) = x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}$ be the jth term in f, and

$$b_j = \beta_j(\omega_1, \ldots, \omega_n) = \omega_1^{d_{j_1}} \cdots \omega_n^{d_{j_n}}$$

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 $F(k) = f(\omega_1^k, \ldots, \omega_n^k)$, the Ben-Or/Tiwari algorithm solves for b_j and c_j similar to the Prony's method. That is, it finds a generating polynomial $\Lambda(z)$, determines its roots, and solves a Vandermonde system. 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: $b_j = \omega_1^{d_{j_1}} \omega_2^{d_{j_2}} \ldots, \omega_n^{d_{j_n}}$.

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

2.3. **A modified numeric Ben-Or/Tiwari algorithm.** If the steps of the Ben-Or/Tiwari algorithm are directly implemented in floating-point arithmetic, difficulties arise at various stages of the computation. The first difficulty is that the

subroutines for linear system solving and root finding may encounter significant numerical errors. The second difficulty is that we can no longer employ exact divisions to recover the exponents in a multivariate term.

While it is well-known that Hankel and Vandermonde matrices can often be ill-conditioned [1], this is particularly true when the input is *real*. Therefore, we modify the Ben-Or/Tiwari algorithm by evaluating at primitive roots of unity. This turns out to improve the conditioning problems of the encountered Hankel and Vandermonde systems (see Subsection 3.1), and has the added advantage that it allows for the recovering of the exponent of each variable in a multivariate term. We now assume 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 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 with $\omega_k = \exp(2\pi i/p_k)$:

$$\alpha_s = f(\omega_1^s, \omega_2^s, \ldots, \omega_n^s) \text{ for } 0 \le s \le 2t-1. \tag{2.4}$$

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 in modulo $m = p_1 \cdots p_n$. Because 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., [9]). That is, we have $d_j \equiv d_{j_k} \bmod p_k$ for $1 \le k \le n$ and

$$d_j = d_{j_1} \cdot \left(\frac{m}{p_1}\right) + \cdots + d_{j_n} \cdot \left(\frac{m}{p_n}\right). \tag{2.5}$$

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 $\sum_{j=1}^{t} c_j x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}$ 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

$$\underbrace{\begin{bmatrix} \alpha_0 & \ldots & \alpha_{t-1} \\ \alpha_1 & \ldots & \alpha_t \\ \vdots & \ddots & \vdots \\ \alpha_{t-1} & \ldots & \alpha_{2t-2} \end{bmatrix}}_{H_0} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} \alpha_t \\ \alpha_{t+1} \\ \vdots \\ \alpha_{2t-1} \end{bmatrix}. \tag{2.6}$$

(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 (2.5) by the reverse Chinese remainder algorithm.

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

Solve an associated Vandermonde system: (now $\beta_j = x_1^{d_{j_1}} \cdots x_n^{d_{j_n}}$ are recovered, $\tilde{b}_j$ can be either b_j or $\beta_j(\omega_1, \ldots, \omega_n)$)

$$\begin{bmatrix} 1 & \cdots & 1 \\ \tilde{b}_1 & \cdots & \tilde{b}_t \\ \vdots & \ddots & \vdots \\ \tilde{b}_1^{t-1} & \cdots & \tilde{b}_t^{t-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{t-1} \end{bmatrix}. \tag{2.7}$$

3. Sensitivity analysis and randomized conditioning

Now we focus on the numerical accuracy of the interpolation algorithm presented in the previous section. We introduce a randomized technique which will dramatically improve the expected numerical stability of our algorithm.

The Ben-Or/Tiwari algorithm first recovers the polynomial support. That is, it determines which terms are non-zero in the target polynomial. We look at its numerical sensitivity, 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, we still need to separate the exponents of different variables and recover the corresponding coefficients, again via the Vandermonde system V. Finally, we show how randomization can substantially improve the conditioning of V, hence improve the stability of the entire interpolation process.

3.1. Conditioning of associated Hankel system.

Consider the modified numeric Ben-Or/Tiwari algorithm in Subsection 2.3. To determine polynomial $\Lambda(z) = z^t + \lambda_{t-1} z^{t-1} + \cdots + \lambda_0$, we need to solve a Hankel system as in (2.6). In general, if 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. Such scaling problem is avoided in our method, since our H_0 is formed from the evaluations on the unit circle.

Consider f in (2.3) at primitive roots of unity as in (2.4), $\alpha_s = f(\omega_1^s, \omega_2^s, \ldots, \omega_n^s)$ for $0 \le s \le 2t-1$ and $b_j = \beta_j(\omega_1, \ldots, \omega_n)$. Define $D = \text{diag}(c_1, \ldots, c_t)$,

$$H_0 = \begin{bmatrix} \alpha_0 & \cdots & \alpha_{t-1} \\ \vdots & \ddots & \vdots \\ \alpha_{t-1} & \cdots & \alpha_{2t-2} \end{bmatrix} \text{ and } V = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_t \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{t-1} & b_2^{t-1} & \ldots & b_t^{t-1} \end{bmatrix},$$

then

$$H_0 = V D V^{\text{Tr}}. \tag{3.1}$$

Following proposition links the condition of H_0 to the condition of V and to $1/|c_j|$ of coefficients c_j in the target polynomial (for $1 \le j \le t$).

Proposition 3.1.

(i) $\|H_0^{-1}\| \ge \frac{1}{t} \max_j \frac{1}{|c_j|}$, *and* $\|H_0^{-1}\| \ge \frac{\|V^{-1}\|^2}{\sum_{1 \le j \le t} |c_j|}$.

(ii) $\|H_0^{-1}\| \leq \|V^{-1}\|^2 \cdot \max_j \dfrac{1}{|c_j|}$.

Thus, bounds for $\|H_0^{-1}\|$ involve both the (inverses of) coefficients of the interpolated polynomial $c_1, \ldots, c_t$ and the condition of the Vandermonde system V. In some sense coefficients $c_1, \ldots, c_t$ are intrinsic to a problem instance, and having them very small (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.

3.2. **Root finding on the generating polynomial.** In our modified numeric Ben-Or/Tiwari algorithm, we also need to find roots for $\Lambda(z) = 0$. In general, root finding can be very ill-conditioned to the perturbations in the coefficients [17].

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

Theorem 3.1. *For 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*

$$|b_k - \tilde{b}_k| < \frac{\epsilon \cdot \|\Gamma(z)\|_1}{|\prod_{j \neq k}(b_k - b_j)|} + K\epsilon^2. \tag{3.2}$$

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 (2.6). The value of $|\prod_{j \neq k}(b_k - b_j)|$ is directly related to the condition of the Vandermonde system V, and depends on the distribution of b_j's on the unit circle (see Subsection 3.5).

3.3. **Separation of powers.** After computing approximations $\widetilde{b}_1, \ldots, \widetilde{b}_t$ for term values $b_1, \ldots, b_t$, we still need to consider the precision required for 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 $1 \leq j \leq t$, as in (2.5). Since each b_j lies on the unit circle, we only need to 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 is bounded below by twice the sine of half the angle between them. 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

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

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

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.

3.4. **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 $c_1, \ldots, c_t$. We can do this directly by solving the Vandermonde system V in equation (2.7) (Step 3 in algorithm `ModBOTInterp`). The main issue in this case is the condition of V. We examine this in Subsection 3.5.

3.5. **Condition of the Vandermonde system.** While Vandermonde matrices can be poorly conditioned [8, 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 $m \times m$ Vandermonde matrices, the 2-norm condition number has the optimal value of 1 when the nodes are all the mth roots of unity [7, example 6.4]. A slightly less uniform sequence of nodes is studied in [5], 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.

When $b_1, \ldots, b_t$ are all mth roots of unity (for $m \geq t$) we have the following bounds for $\|V^{-1}\|$ from [7]:

$$\max_{1 \leq k \leq t} \frac{1/\sqrt{t}}{\prod_{j \neq k} |b_j - b_k|} < \|V^{-1}\| \leq \max_{1 \leq k \leq t} \frac{2^{t-1}\sqrt{t}}{\prod_{j \neq k} |b_j - b_k|}. \tag{3.3}$$

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. For 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 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.

3.6. **Randomized reconditioning.** We now demonstrate how randomization dramatically ameliorates the potential ill-conditioning in the Vandermonde matrix.

Let $p_1, \ldots, p_n$ be distinct primes, $p_k > \deg_{x_k} f$, and $\omega = \exp(2\pi i/m)$ for $m = p_1 \cdots p_n$. If polynomial f is evaluated at powers of $(\omega_1, \ldots, \omega_n)$ for $\omega_k = \omega^{m/p_k}$ (cf. Subsection 2.3), the distribution of term values on the unit circle is fixed because the polynomial terms are fixed. We may well end up in an ill-conditioned Vandermonde matrix as discussed above. To eliminate this situation with high probability, we introduce randomization as follows. Instead of using $\exp(2\pi i/p_k)$, 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 $1 \leq r_k < p_k$. Equivalently, we choose a single r with $r \equiv r_k \bmod p_k$, $1 \leq r < m$, so that $\omega_k = \omega^{mr/p_k}$ (see (2.5)).

To analyze the distribution of term values, we equivalently consider the univariate $\widetilde{f}(x) = \sum_{j=1}^{t} c_j x^{d_j}$ where $d_j = d_{j_1}(m/p_1) + \cdots + d_{j_n}(m/p_n)$ (cf. Subsection 2.3). The term values are $\omega^{d_1}, \ldots, \omega^{d_t}$, and the stability of recovering d_j depends

on the condition of the Vandermonde matrix V with nodes $\omega^{d_1}, \ldots, \omega^{d_t}$, which is inversely related to the product of $|\omega^{d_j} - \omega^{d_k}|$ for $1 \leq j < k \leq t$ described in (3.3).

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 $1 \leq j \leq t$ 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 a random r, hopefully minimizing the chance that there are many small differences. Once the Hankel matrix H_0 is constructed, we check the conditioning, and if it is poor, we can choose another random r and repeat the process. The next theorem gives us the assurance that we never have to do this very often.

Theorem 3.2. *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 $0 \leq d_1, \ldots, d_t \leq m-1$ 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 $1/2$,*

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

Thus we eliminate the dependence upon m, and hence the dependence upon the size of the dense representation of the polynomial. However, we believe this is probably still far from optimal. Considerable cancelation 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 (of H and V) is *much* lower.

		Sparsity(%)				
		0.1	1	2	5	10
Degree	101	2.2137	2.1942	3.6469	9.9189	26.974
	211	2.2551	3.6963	6.9576	25.279	69.442
	503	2.3136	9.4414	22.311	80.068	247.65
	701	2.2000	16.363	38.664	164.16	439.31
	1009	2.3247	29.810	72.378	481.44	765.84

Figure 4.1: *Median condition number of V with randomization.*

A difficult problem we have not addressed thus far is the determination of the sparsity t. While do not offer a complete solution, we note that randomization is of potential help.

Consider

$$H_1 = \begin{bmatrix} \alpha_1 & \ldots & \alpha_t \\ \vdots & \ddots & \vdots \\ \alpha_t & \ldots & \alpha_{2t-1} \end{bmatrix}.$$

The randomization appears to ensure that *all* leading minors of H_1 are well-conditioned with high probability. This leads to a possible way to identify the sparsity t of f by simply computing $\alpha_1, \alpha_2, \ldots$ (at a random root of unity) until the constructed H_1 becomes ill-conditioned. This can be achieved efficiently with the algorithm of [4], and with high probability should identify t.

It is shown in [12, Theorem 4] that all leading minors of H_1 are non-singular with high probability, which is clearly a necessary condition for the leading minors to be well-conditioned. The proof of [12, Theorem 4] makes use of the factorization

of the leading $k \times k$ minor $H_1^{(k)}$ of H_1,

$$H_1^{(k)} = V^{(k)} DY (V^{(k)})^{\mathrm{Tr}},$$

where matrix $V^{(k)} \in \mathbb{C}^{k \times t}$ consists of the first k rows of V and $Y = \mathrm{diag}(b_1, \ldots, b_t)$. Since Theorem 3.2 can easily be generalized to the $k \times t$ matrix $V^{(k)}$, a well-conditioned $V^{(k)}$ provides an explanation for a well-conditioned $H_1^{(k)}$.

4. Experiments

We have tested our modified Ben-Or/Tiwari method. Our computational environment is the computer algebra system Maple 10 using hardware arithmetic (IEEE floating point).

Our algorithm interpolates 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 (see Subsection 2.3). Therefore, we concentrate on univariate examples. Since the stability is directly dependent on the condition of the underlying Vandermonde system, we arrange our tests by the condition of this system.

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

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.

Clustered term values

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

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

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	Mean	Median
0	92.8019727202980	73.4823536193264

However, after randomization, that is, instead of interpolating at powers of $\omega = \exp(2\pi i/1009)$, 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	Mean	Median
0	27.9983307662379	$.242793778266858e-7$
$\pm 10^{-12} \sim 10^{-9}$	.869652877288326	$.170781612648532e-6$

In addition, when the random noise belongs to $\pm 10^{-9} \sim 10^{-6}$, a singular system is encountered in our test, and 22 among the 99 non-singular results are of distance less than 10^{-4} after randomization.

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 3.2 provides assurances that we should never have to restart this many times before achieving a well-conditioned Vandermonde matrix, and hence obtain reliable results.

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.

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] S. Cabay and R. Meleshko. A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices. *SIAM J. Matrix Anal. Appl.*, 14(3):735–765, 1993.
[5] A. Córdova, W. Gautschi, and S. Ruscheweyh. Vandermonde matrices on the circle: spectral properties and conditioning. *Numerische Mathematik*, 57:577–591, 1990.
[6] M. Gasca and T. Sauer. On the history of multivariate polynomial interpolation. *J. Computational and Applied Mathematics*, 122:23–35, 2000.
[7] W. Gautschi. Norm estimates for inverses of Vandermonde matrices. *Numerische Mathematik*, 23:337–347, 1975.
[8] W. Gautschi and G. Inglese. Lower bounds for the condition numbers of Vandermonde matrices. *Numerische Mathematik*, 52:241–250, 1988.
[9] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publ., Boston, Massachusetts, USA, 1992.
[10] 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.
[11] 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.
[12] Erich Kaltofen and Wen-shin Lee. Early termination in sparse interpolation algorithms. *J. Symbolic Comput.*, 36(3-4):365–400, 2003.
[13] 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.
[14] R. Lorentz. Multivariate Hermite interpolation by algebaic polynomials: a survey. *J. Computational and Applied Mathematics*, 122:167–201, 2000.
[15] Y. Mansour. Randomized approximation and interpolation of sparse polynomials. *SIAM Journal on Computing*, 24(2):357–368, 1995.
[16] 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.
[17] J. H. Wilkinson. *Rounding errors in algebraic processes*. Prentice-Hall, Englewood Cliffs, N.J., 1963.
[18] 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.
[19] 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.
[20] R. Zippel. Interpolating polynomials from their values. *J. Symbolic Comput.*, 9(3):375–403, 1990.

School of Computer Science, University of Waterloo, Canada
E-mail address: mwg@uwaterloo.ca

School of Computer Science, University of Waterloo, Canada
E-mail address: glabahn@uwaterloo.ca

Departement Wiskunde en Informatica, Universiteit Antwerpen, Belgium
E-mail address: wenshin.lee@ua.ac.be