SIAM J. COMPUT.
Vol. 24, No. 2, pp. 387–397, April 1995

SPARSE POLYNOMIAL INTERPOLATION IN NONSTANDARD BASES[*]

Y. N. LAKSHMAN[†] AND B. DAVID SAUNDERS[‡]

Abstract. In this paper, we consider the problem of interpolating univariate polynomials over a field of characteristic zeros that are sparse in (a) the Pochhammer basis, or (b) the Chebyshev basis. The polynomials are assumed to be given by black boxes, i.e., one can obtain the value of a polynomial at any point by querying its black box. We describe efficient new algorithms for these problems. Our algorithms may be regarded as generalizations of Ben-Or and Tiwari's (1988) algorithm (based on the BCH decoding algorithm) for interpolating polynomials that are sparse in the standard basis. The arithmetic complexity of the algorithms is $O(t^2 + t\log d)$, which is also the complexity of the univariate version of the Ben-Or and Tiwari algorithm. That algorithm and those presented here also share the requirement of $2t$ evaluation points.

Key words. polynomial interpolation, sparsity, Chebyshev polynomial, Pochhamer basis, BCH codes

AMS subject classifications. 68Q40, 12Y05, 41A05

Introduction. In this paper, we consider the problem of efficiently interpolating polynomials given by black boxes that have sparse representations in various bases. Usually, one considers a polynomial $f(x) = \sum_{i=0}^{n} a_i x^i$ as t-sparse if at most t of the coefficients a_i are non-zero. The problem of interpolating a sparse polynomial from a list of values at specific points, or from values given by a black box that evaluates the polynomial is of great importance in computational algebra. Sparse interpolation has received significant attention in several recent papers. (Ben-Or and Tiwari (1988), Clausen, Dress, Grabmeier, and Karpinski (1988), Kaltofen and Lakshman (1988), Grigoriev and Karpinski (1987), Grigoriev, Karpinski, and Singer (1990), (1991a), (1994), Borodin and Tiwari (1990), Zippel (1990).) In particular, the problem of interpolating a t-sparse univariate polynomial given a black box for evaluating the polynomial can be efficiently solved using Ben-Or and Tiwari's adaptation of the BCH decoding algorithm.

Sparse interpolation is well recognized as a very useful tool for controlling intermediate expression swell in computer algebra (Zippel (1990), Kaltofen and Trager (1990)). Sparse polynomials and rational functions can be evaluated quickly and that makes them attractive to several applications and an interesting line of research is to try to infer properties of sparse polynomials (such as divisibility, existence of nontrivial greatest common divisor, the existence of real roots, etc.) from their values at a small number points (see Grigoriev, Karpinski, and Odlyzko (1992)). Traditionally, "sparse polynomial (or rational function)" is taken to mean a polynomial (or rational function) with a "few terms" where the "terms" are power products of the variables involved. One can reasonably ask for sparse representations for polynomials in other bases such as the Chebyshev polynomials or the shifted power basis $1, x-\alpha, (x-\alpha)^2, \ldots$. In this paper, we consider two classes of polynomials—those which are sparse in the Chebyshev basis and those which are sparse in the Pochhammer basis. A polynomial $f(x) = \sum_{i=0}^{n} a_i T_i(x)$ is t-sparse in the Chebyshev basis if at most t of the coefficients a_i are nonzero, where $T_i(x)$ is the ith Chebyshev polynomial. Sparse Pochhammer polynomials are defined similarly. We provide algorithms for interpolating such polynomials efficiently, given a black box that evaluates the polynomial. The algorithms require as input an upper bound t on the number of nonzero terms in the interpolating polynomial. The arithmetic complexity of the algorithms is $O(t^2 + t\log d)$, which is also the complexity of the

[*]Received by the editors October 1, 1992; accepted for publication (in revised form) November 11, 1993.

[†]Department of Mathematics and Computer Science, Drexel University, Philadelphia, Pennsylvania 19104 (lakshman@mcs.drexel.edu). The work of this author was supported in part by NSF grant CCR–9203062.

[‡]Department of Computer and Information Sciences, University of Delaware, Newark, Delaware 19716 (saunders@udel.edu). The work of this author was supported in part by NSF grant CCR-9123666.

univariate version of the Ben-Or and Tiwari algorithm. That algorithm and those presented here also share the requirement of $2t$ evaluation points. However, when one considers the bit complexity (when the ground field is $\mathbf{Q}$), the Ben-Or and Tiwari algorithm and our sparse Chebyshev algorithm involve intermediate values of bit size $O(td)$, whereas in the sparse Pochhammer algorithm intermediate values grow to a bit length of $O(d\log(t))$.

Several recent results concerning sparse interpolation are similar in spirit to the BCH-decoding algorithm. In Dress and Grabmeier (1991), it is shown that many of the sparse interpolation algorithms can be formulated and proven correct in the context of t-sparse sums of abelian monoids. A somewhat different framework in Grigoriev, Karpinski, and Singer (1991a) in terms of t-sparse sums of eigenfunctions of operators is shown to provide several generalizations of the sparse interpolation algorithm of Ben-Or and Tiwari. In fact, an efficient algorithm for interpolating polynomials sparse in the Pochhammer basis follows from an observation in Grigoriev, Karpinski, and Singer (1991a) which points out that t-sparse sums of eigenfunctions of the operator $x\Delta(f) = x(f(x) - f(x-1))$ correspond to polynomials that are t-sparse in the Pochhammer basis. We use the Pochhammer case as a motivating example for our discussion of the Chebyshev case.

The problem of efficiently interpolating polynomials that are sparse in the Chebyshev basis was stated as an open problem in Borodin and Tiwari (1990). Our algorithm provides a solution and it is another generalization of the algorithm of Ben-Or and Tiwari. It uses properties shared by the standard power basis and the Chebyshev polynomials and appears to be different from the generalizations presented in Dress and Grabmeier (1991), Grigoriev, Karpinski, and Singer (1991a). While the general structure of our algorithm appears similar to the algorithms falling into the Dress–Grabmeier framework, the details are quite different. In particular, the centerpiece of the algorithms in the Dress–Grabmeier framework is a certain Toeplitz matrix where, as in our algorithm, a similar role is played by a matrix that is *the sum of a Toeplitz matrix and a Hankel matrix.* It would be interesting to reconcile our algorithm with the Dress–Grabmeier or the Grigoriev–Karpinski–Singer framework.

The rest of the paper is organized as follows. In §1, we motivate our discussion by briefly stating the algorithm for interpolating polynomials that are sparse in the Pochhammer basis. In §2, we describe our algorithm for interpolating polynomials sparse in the Chebyshev basis. Section 3 provides an analysis of the complexity of the algorithm, which is followed by some additional remarks in the last section.

1. Sparse interpolation in the Pochhammer basis.

The Pochhammer symbol $x^{\overline{n}}$ denotes the rising factorial power

$$x(x+1)\dots(x+n-1)$$

for any integer $n \geq 0$. It is easily shown that $x^{\overline{i}},\ i = 0, 1, 2, \dots$ is a $\mathbf{Q}$ basis for the polynomial ring $\mathbf{Q}[x]$. A polynomial $f(x)$ is t-sparse in the Pochhammer basis (of rising powers) if and only if $\exists f_i \in \mathbf{Q},\ e_i \in \mathcal{Z}_+,\ i = 1, \dots, t$, such that

$$f(x) = \sum_{k=1}^{t} f_k x^{\overline{e_k}}.$$

We assume that $e_1 > e_2 > \cdots > e_t$. Suppose we are given a black box that returns the value of $f(x)$ at any $x \in \mathbf{Q}$ and a bound t on the number of nonzero terms in the representation of $f(x)$ in the Pochhammer basis. Then, we can interpolate $f(x)$ from its values at $x = 1, 2, \dots, 2t$. The algorithm follows from an observation in Grigoriev, Karpinski, and Singer (1991a) which places this problem in the framework of Dress and Grabmeier. It is an elementary fact that the finite difference operator behaves like the derivative on the Pochhammer symbol. More precisely, let

$$\Delta(f(x)) = f(x+1) - f(x).$$

We have

$$\begin{aligned}\Delta(x^{\overline{n}}) &= (x+1)^{\overline{n}} - x^{\overline{n}}\\ &= n(x+1)^{\overline{n-1}}.\end{aligned}$$

Let

$$f^{(i)}(x) = \sum_{k=1}^{t} e_k^i f_k x^{\overline{e_k}} \text{ for } i = 0, 1, \ldots.$$

Note that

$$x\Delta(f^{(i)}(x)) = f^{(i+1)}(x).$$

We can compute $f^{(i)}(a)$ for $i = 0, 1, \ldots, 2t-1$ by repeated applications of the above recurrence from the values $f(a+i)$, $i = 0, 1, \ldots, 2t-1$.

Consider the polynomial $\Psi(z)$ of degree t whose roots are the e_i, $i = 1, \ldots, t$, i.e.,

$$\Psi(z) = \prod_{i=1}^{t}(z - e_i).$$

Let

$$\Psi(z) = z^t + \psi_{t-1}z^{t-1} + \cdots + \psi_1 z + \psi_0.$$

The $f^{(i)}$ and ψ_i satisfy the following linear relations. Assume that $\psi_t = 1$.

LEMMA 1. *For* $j = 0, \ldots, t-1$, $\sum_{i=0}^{t} \psi_i f^{(i+j)}(a) = 0$.

Proof. This is a special case of Theorem 1 in Dress and Grabmeier (1991). (See page 62 in that paper.) □

Furthermore, Theorem 1 in Dress and Grabmeier (1991) shows that if t is the number of nonzero terms in the representation of $f(x)$ in the Pochhammer basis, then the $t \times t$ matrix $\mathcal{F}$ with $\mathcal{F}_{i,j} = f^{(i+j-2)}(a)$ is nonsingular for any $a > 0$. By using this, one can compute the polynomial $\Psi(z)$ from $f(a), f(a+1), \ldots, f(a+2t-1)$. The roots of $\Psi(z)$ are the "Pochhammer exponents" of $f(x)$. Knowing these one can easily compute the coefficients f_k. We omit details to avoid repetition but state the algorithm for the sake of completeness.

Algorithm Sparse-Pochhammer-Interpolation

Input: A black box for evaluating a polynomial $f(x)$. $f(x)$ is known to be t-sparse in the Pochhammer basis.

Output: A list of pairs (f_i, e_i), $i = 1, \ldots, t$ such that $f(x) = \sum_{i=1}^{t} f_i x^{\overline{e_i}}$.

1. *Query the black box to obtain the values of* $f(x)$ *at* $x = a, a+1, \ldots, a+2t-1$ *for some* $a \neq 0$. *Compute* $f^{(i)}(a)$ *for* $i = 0, 1, \ldots, 2t-1$ *from* $f(a+i)$, $i = 0, 1, \ldots, 2t-1$.
2. *Solve the system of linear equations* $\mathcal{F}\vec{\psi} = -\vec{\mathsf{f}}$. *If*
$$\vec{\psi} = (\ \psi_0 \quad \psi_1 \quad \cdots \quad \psi_{t-1}\ )^T,$$
then the auxiliary polynomial $\Psi(z)$ *is given by*
$$\Psi(z) = z^t + \psi_{t-1}z^{t-1} + \cdots + \psi_1 z + \phi_0.$$
3. *Find the integer roots of* $\Psi(z)$. *The roots are the exponents of* $f(x)$, *i.e.,* $e_1, e_2, \ldots, e_t$.
4. *Solve the linear system of equations* $\mathcal{V}\vec{\mathsf{f}} = \vec{\mathsf{F}}$ *to obtain* $f_1 a^{\overline{e_1}}, \ldots, f_t a^{\overline{e_t}}$. *Since we know the* e_i *and* a, *the coefficients* f_i *are easily determined. Output the list of pairs* $[(f_1, e_1), \ldots, (f_t, e_t)]$.

Analysis of the Sparse-Pochhammer-Interpolation algorithm. While the sparse Pochhammer interpolation algorithm and the algorithm of Ben-Or and Tiwari for interpolating polynomials sparse in the standard power basis both illustrate the Dress–Grabmeier framework, the details are quite different. The sparse Pochhammer algorithm differs in a fundamental way from the algorithm of Ben-Or and Tiwari. The counterpart of the auxiliary polynomial $\Psi(z)$ in Ben-Or and Tiwari's algorithm has as roots a^{e_i}, whereas here the auxiliary polynomial $\Psi(z)$ has as roots the *degrees* e_i of the nonzero terms in the interpolating polynomial. This has implications on how we actually compute the roots of the auxiliary polynomial. Also, this algorithm uses a completely different set of evaluation points (importantly, the evaluation points are smaller).

The analysis will be brief, pointing out the effect of the differences just mentioned. We assume unit cost for a black box query. The $f^{(i)}(a)$ for $i = 0, 1, \dots, 2t-1$ can be computed from $f(a+i),\ i = 0, 1, \dots, 2t-1$ by using $O(t^2)$ field operations. The Toeplitz system of equations $\mathcal{F}\vec{\psi} = -\vec{\mathrm{f}}$ can be solved using $O(t^2)$ field operations using the Berlekamp–Massey algorithm (Kaltofen and Lakshman (1988)) to find the coefficients of the auxiliary polynomial $\Psi(z)$. We can use the Cantor–Zassenhaus algorithm to factor $\Psi(z)$ modulo a suitable prime p and the Hensel lifting to recover the integer roots. The cost is $O(t^2 + t\log d)$, where d is the degree of the interpolating polynomial (see, for instance, Loos (1983)). The final step of solving a transposed Vandermonde system of equations can be done using the algorithm of Zippel (1990) in $O(t^2)$ field operations. We can now add up the costs of the four steps and we have the following theorem.

THEOREM 2. *The sparse Pochhammer interpolation algorithm performs $O(t^2 + t\log d)$ field operations to interpolate a t-Pochhammer-sparse polynomial given by a black box. Here, d is the degree of the polynomial being interpolated (not part of the input). The algorithm makes $2t$ queries to the black box evaluating the polynomial. The numerator and denominator of the rational numbers that arise during the algorithm have $O(d\log(t))$ bits.* □

2. Sparse interpolation in the Chebyshev basis.

In this section, we describe an efficient algorithm for interpolating polynomials in $\mathbf{Q}[x]$ that are sparse in the Chebyshev basis, where $\mathbf{Q}$ is the field of rational numbers. The algorithm is entirely rational, so it applies over any field of characteristic zero. The analysis in §3 in terms of arithmetic operations would apply in the general characteristic zero case as well, provided the evaluation point a is rational and $a \geq 2$. The bit complexity analysis, of course, is specific to the base field $\mathbf{Q}$. Let $T_n(x)$ denote the nth Chebyshev polynomial of the first kind:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \text{ for } n \geq 2.$$

It is well known that $T_i(x),\ i = 0, 1, 2, \dots$ is a $\mathbf{Q}$-basis for the polynomial ring $\mathbf{Q}[x]$.

Let $f(x) = \sum_{i=0}^{m} f_i T_i(x)$ be a polynomial in the ring $\mathbf{Q}[x]$. As usual, we say that $f(x)$ is t-sparse (in the Chebyshev basis) if and only if at most t of the f_i's are nonzero.

Suppose we are given a black box that returns the value of $f(a)$ for any $a \in \mathbf{Q}$. We present an algorithm that interpolates $f(x)$ from its values at specially chosen $2t$ points. Our algorithm may be regarded as a generalization of the Ben-Or and Tiwari algorithm (Ben-Or and Tiwari (1988)). We identify two crucial properties that are required for the generalization to work. In the context of Chebyshev polynomials, they are the following items:

- For $m, n \geq 0$, $T_n(T_m(x)) = T_{mn}(x) = T_m(T_n(x))$, i.e., Chebyshev polynomials commute with respect to composition.
- For $m, n \geq 0$, $T_m(x)T_n(x) = 1/2(T_{m+n}(x) + T_{|m-n|}(x))$, i.e., the product of two Chebyshev polynomials is a *fixed* linear form in two others.

These properties are easy consequences of well-known properties of the Chebyshev polynomials. Before describing our interpolation algorithm, we need a statement of uniqueness of a sparse interpolating polynomial.

LEMMA 3. *Let $f(x)$ be a t-sparse polynomial (in the Chebyshev basis). If for some $a > 1$, $f(T_i(a)) = 0$ for $i = 0, 1, 2, \ldots, t-1$, then $f(x)$ is identically zero.*

Proof. Since $f(x)$ is t-sparse, there exist $f_i \in \mathbf{Q}$ and $\delta_i \in \mathcal{Z}_+$, for $i = 1, \ldots, t$ such that

$$f(x) = f_1 T_{\delta_1}(x) + f_2 T_{\delta_2}(x) + \cdots + f_t T_{\delta_t}(x).$$

Now, we can rewrite the fact that $f(T_i(a)) = 0$ for $i = 0, 1, 2, \ldots, t-1$ as $\mathcal{V}\vec{\mathsf{f}} = \vec{\mathbf{0}}$, where

$$\mathcal{V} = \begin{pmatrix} T_{\delta_1}(T_0(a)) & T_{\delta_2}(T_0(a)) & \ldots & T_{\delta_t}(T_0(a)) \\ T_{\delta_1}(T_1(a)) & T_{\delta_2}(T_1(a)) & \ldots & T_{\delta_t}(T_1(a)) \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1}(T_{t-1}(a)) & T_{\delta_2}(T_{t-1}(a)) & \ldots & T_{\delta_t}(T_{t-1}(a)) \end{pmatrix}, \quad \vec{\mathsf{f}} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_t \end{pmatrix}.$$

If f is not identically zero, then $\mathcal{V}$ is singular. Let $\vec{c} = (c_0, c_1, \ldots, c_{t-1})^T$ be such that $\vec{c}^T \mathcal{V} = 0$. Consider the polynomial

$$C(x) = c_0 T_0(x) + c_1 T_1(x) + \cdots + c_{t-1} T_{t-1}(x)$$

of degree at most $t-1$. Use the fact that $T_{\delta_j}(T_i(a)) = T_i(T_{\delta_j}(a))$ and rewrite $\mathcal{V}$ as

$$\mathcal{V} = \begin{pmatrix} T_0(T_{\delta_1}(a)) & T_0(T_{\delta_2}(a)) & \ldots & T_0(T_{\delta_t}(a)) \\ T_1(T_{\delta_1}(a)) & T_1(T_{\delta_2}(a)) & \ldots & T_1(T_{\delta_t}(a)) \\ \vdots & \vdots & \ddots & \vdots \\ T_{t-1}(T_{\delta_1}(a)) & T_{t-1}(T_{\delta_2}(a)) & \ldots & T_{t-1}(T_{\delta_t}(a)) \end{pmatrix}.$$

Now, $\vec{c}^T \mathcal{V} = 0$ implies that $T_{\delta_1}(a), T_{\delta_2}(a), \ldots, T_{\delta_t}(a)$ are all roots of $C(x)$. These t values are distinct. (In fact, the values $T_i(a)$ are strictly monotonic increasing if $a > 1$, which is straightforward to check from the defining recurrence.) But $C(x)$ is of degree $t-1$ and cannot have t roots. Hence, $\mathcal{V}$ cannot be singular, and f must be identically zero. □

As a consequence, we have the next corollary.

COROLLARY 4. *If $f(x)$ and $g(x)$ are two distinct t-sparse polynomials, then for each $a > 1$ there is an i, $0 \le i \le 2t-1$ such that $f(T_i(a)) \ne g(T_i(a))$.*

Proof. Consider $h(x) = f(x) - g(x)$. The polynomial $h(x)$ is $2t$-sparse and if the corollary is not true, then $h(T_i(a)) = 0$ for $i = 0, \ldots, 2t-1$. Apply the previous lemma. □

We can now describe the sparse interpolation algorithm. As before, $f(x) = \sum_{i=1}^{t} f_i T_{\delta_i}(x)$. The algorithm determines the δ_i first. Once the δ_i are known, it is a simple task to compute the coefficients f_i. Let $a_i = f(T_i(a)), i = 0, 1, \ldots, 2t-1$ for some $a > 1$. Consider the polynomial of degree t whose roots are $T_{\delta_i}(a)$ represented in the Chebyshev basis, normalized to have a leading coefficient of 1, i.e.,

$$\Phi(z) = T_t(z) + \phi_{t-1} T_{t-1}(z) + \cdots + \phi_0 T_0(z)$$

and

$$\Phi(T_{\delta_i}(a)) = 0 \ \text{ for } \ i = 1, \ldots, t.$$

Our strategy is to compute $\Phi(z)$ and then find its integer roots. First, we show that the coefficients of $\Phi(z)$ satisfy the following linear relations:

LEMMA 5.

$$\sum_{j=0}^{t-1} \phi_j \times (a_{i+j} + a_{|j-i|}) = -(a_{i+t} + a_{|t-i|})$$

for $i = 0, 1, 2, \ldots$.

Proof. Assume $\phi_t = 1$.

$$\begin{aligned}
\left(\sum_{j=0}^{t-1} \phi_j a_{j+i}\right) + a_{t+i} &= \sum_{j=0}^{t} \phi_j \left(\sum_{l=1}^{t} f_l T_{\delta_l}(T_{j+i}(a))\right)\\
&= \sum_{l=1}^{t} f_l \left(\sum_{j=0}^{t} \phi_j T_{\delta_l}(T_{j+i}(a))\right) = \sum_{l=1}^{t} f_l \left(\sum_{j=0}^{t} \phi_j T_{j+i}(T_{\delta_l}(a))\right)\\
&= \sum_{l=1}^{t} f_l \left(\sum_{j=0}^{t} \phi_j (2T_j(T_{\delta_l}(a))T_i(T_{\delta_l}(a)) - T_{|j-i|}(T_{\delta_l}(a)))\right)\\
&= \sum_{l=1}^{t} f_l \left[\Phi(T_{\delta_l}(a)) \times 2T_i(T_{\delta_l}(a)) - \left(\sum_{j=0}^{t} \phi_j T_{|j-i|}(T_{\delta_l}(a))\right)\right]\\
&= -\sum_{l=1}^{t} f_l \left(\sum_{j=0}^{t} \phi_j T_{\delta_l}(T_{|j-i|}(a))\right)\\
&= -\phi_j \left(\sum_{l=1}^{t} f_l T_{\delta_l}(T_{|j-i|}(a))\right) = -\sum_{j=0}^{t} \phi_j a_{|j-i|}\\
&= -\left(\sum_{j=0}^{t-1} \phi_j a_{|j-i|}\right) - a_{|t-i|}.
\end{aligned}$$

Hence, $\sum_{j=0}^{t-1} \phi_j (a_{i+j} + a_{|j-i|}) = -(a_{i+t} + a_{|t-i|})$. □

The above lemma says that the coefficients of $\Phi(z)$ satisfy the system of equations

$$\mathcal{A}\vec{\phi} = -\vec{\alpha},$$

where

$$\mathcal{A} = \begin{pmatrix} 2a_0 & 2a_1 & \dots & 2a_{t-1} \\ 2a_1 & a_2 + a_0 & \dots & a_t + a_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{t-1} & a_t + a_{t-2} & \dots & a_{2t-2} + a_0 \end{pmatrix}, \quad \vec{\phi} = \begin{pmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{t-1} \end{pmatrix},$$

$$\vec{\alpha} = \begin{pmatrix} 2a_t \\ a_{t+1} + a_{t-1} \\ \vdots \\ a_{2t-1} + a_1 \end{pmatrix}.$$

Notice that $\mathcal{A}_{i,j} = a_{i+j} + a_{|i-j|}$ is the sum of a Toeplitz matrix and a symmetric Hankel matrix. Next, we show that $\mathcal{A}$ is nonsingular.

LEMMA 6. *$\mathcal{A}$ is nonsingular.*

Proof. Consider the (transposed) *Vandermonde-like* matrix $\mathcal{V}$ and diagonal matrix $\mathcal{B}$.

$$\mathcal{V} = \begin{pmatrix} T_{\delta_1}(T_0(a)) & T_{\delta_2}(T_0(a)) & \dots & T_{\delta_t}(T_0(a)) \\ T_{\delta_1}(T_1(a)) & T_{\delta_2}(T_1(a)) & \dots & T_{\delta_t}(T_1(a)) \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1}(T_{t-1}(a)) & T_{\delta_2}(T_{t-1}(a)) & \dots & T_{\delta_t}(T_{t-1}(a)) \end{pmatrix},$$

$$\mathcal{B} = \begin{pmatrix} 2f_1 & 0 & \dots & 0 \\ 0 & 2f_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 2f_t \end{pmatrix}.$$

According to arguments similar to those in Lemma 1, $\mathcal{V}$ is nonsingular. Since f has exactly t terms, no f_i is zero, hence $\mathcal{B}$ is nonsingular. We claim that $\mathcal{A} = \mathcal{V}\mathcal{B}\mathcal{V}^T$, thereby proving that $\mathcal{A}$ is nonsingular. The (i, j)th element of the product $\mathcal{V}\mathcal{B}\mathcal{V}^{tr}$ is

$$\sum_{l=1}^{t} 2 f_l T_{\delta_l}(T_i(a)) T_{\delta_l}(T_j(a)) = \sum_{l=1}^{t} 2 f_l T_i(T_{\delta_l}(a)) T_j(T_{\delta_l}(a))$$

$$= \sum_{l=1}^{t} f_l (T_{i+j}(T_{\delta_l}(a)) + T_{|i-j|}(T_{\delta_l}(a))) = \sum_{l=1}^{t} f_l (T_{\delta_l}(T_{i+j}(a)) + T_{\delta_l}(T_{|i-j|}(a)))$$

$$= a_{i+j} + a_{|i-j|} = \mathcal{A}_{i,j}. \qquad \square$$

From the above lemmas we see that we can compute the coefficients of $\Phi(z)$ by solving the linear system of equations given by $\mathcal{A}\vec{\phi} = -\vec{\alpha}$. The roots of $\Phi(z)$ are integers with special properties and they can be determined without using numerical root finding algorithms. The roots of $\Phi(z)$ are $T_{\delta_i(a)}$ and we can determine the "Chebyshev exponents" δ_i from $T_{\delta_i}(a)$. The coefficients f_i of f can be determined by solving the linear system of equations given by

$$\mathcal{V}\vec{\mathsf{f}} = \vec{\mathsf{a}},$$

where $\mathcal{V}$ is the matrix in the previous lemma and

$$\vec{\mathsf{f}} = (\ f_1 \quad f_2 \quad \dots \quad f_t\ )^T, \qquad \vec{\mathsf{a}} = (\ a_0 \quad a_1 \quad \dots \quad a_{t-1}\ )^T.$$

To summarize, we now collect all the steps of our interpolation algorithm.

Algorithm Sparse-Chebyshev-Interpolation
Input: A black box for evaluating a polynomial $f(x)$ known to be t-sparse in the Chebyshev basis.
Output: A list of pairs (f_i, δ_i), $i = 1, \dots, t$ such that $f(x) = \sum_{i=1}^{t} f_i T_{\delta_i}(x)$.

1. *Query the black box to obtain the values of $f(x)$ at the $2t$ points $x = T_i(a)$, $i = 0, 1, \dots, 2t - 1$ for some $a > 1$. Let $a_i = f(T_i(a))$.*
2. *Solve the system of linear equations $\mathcal{A}\vec{\phi} = -\vec{\alpha}$. If*

$$\vec{\phi} = (\ \phi_0 \quad \phi_1 \quad \dots \quad \phi_{t-1}\ )^T,$$

then the auxiliary polynomial $\Phi(z)$ is given by

$$\Phi(z) = z^t + \phi_{t-1} z^{t-1} + \dots + \phi_1 z + \phi_0.$$

3. *Find the integer roots of $\Phi(z)$. The roots are $T_{\delta_1}(a), T_{\delta_2}(a), \dots, T_{\delta_t}(a)$. Find the "Chebyshev powers" $\delta_1, \delta_2, \dots, \delta_t$ from $T_{\delta_1}(a), T_{\delta_2}(a), \dots, T_{\delta_t}(a)$.*
4. *Solve the linear system of equations $\mathcal{V}\vec{\mathsf{f}} = \vec{\mathsf{a}}$ to obtain the coefficients $f_1, \dots, f_t$ of f. Output the list of pairs $[(f_1, \delta_1), \dots, (f_t, \delta_t)]$.*

3. Analysis of the Chebyshev Interpolation Algorithm. In our analysis, we first count the number of field operations ($+, -, \times, /$ involving rational numbers) performed by the algorithm in the worst case. We then bound the bit sizes of the rational numbers that might arise during the computation, thus giving upper bounds on the number of bit operations performed by the algorithm. We also assume that querying the black box for the value of $f(x)$ is a unit cost operation.

Solving $\mathcal{A}\vec{\phi} = -\vec{\alpha}$. The matrix $\mathcal{A}$ is the sum of a Hankel and a Toeplitz matrix of size $t \times t$. Such systems can be solved by well-known techniques using $O(t^2)$ field operations (see, for example, Merchant and Parks (1982)).

Finding the integer roots of $\Phi(z)$. The special nature of the roots of $\Phi(z)$ (all are integers of the type $T_{\delta_j}(a)$) allows us to compute them without using regular root finders. Since $\Phi(z)$ is expressed in the Chebyshev basis with leading coefficient 1, ϕ_{t-1}, the coefficient of $T_{t-1}(z)$, is given by

$$\phi_{t-1} = \begin{cases} -T_{\delta_1}(a) & \text{if } t = 1, \\ -2\sum_{j=1}^{t} T_{\delta_j}(a) & \text{if } t > 1. \end{cases}$$

Let $s = -\phi_{t-1}/2$. It is easily shown from the defining recurrence that $T_n(a) = 1/2(\beta^n + \bar{\beta}^n)$, where $\beta = a + \sqrt{a^2 - 1}$ and $\bar{\beta} = a - \sqrt{a^2 - 1}$ for $n = 0, 1, 2, \ldots$. Suppose that for convenience we choose $a = 2$. Then $\beta = 2 + \sqrt{3}$ and $\bar{\beta} = 2 - \sqrt{3}$. Without any loss of generality, assume that $\delta_1 > \delta_2 > \cdots > \delta_t$. Clearly, δ_1 is the unique integer k such that

$$1/2(\beta^k + \bar{\beta}^k) \le s < 1/2(\beta^{k+1} + \bar{\beta}^{k+1}).$$

We can find such a k by an algorithm analogous to the binary powering algorithm as follows:

```
Choose $D = 2\lceil \log_2(s) \rceil$; $(1/2(\beta^D + \bar{\beta}^D) > s)$
Compute integer arrays $p, q$ such that
    $p_i + q_i\sqrt{3} = \beta^{2^i}$ for $i = 0, 1, \ldots$ to at most $D$.
$i := 0$; $p_0 := 1$; $q_0 := 0$;
loop
    increment($i$);
    $p_i := p_{i-1}^2 + 3q_{i-1}^2$; $q_i := 2p_{i-1}q_{i-1}$;
until $p_i > s$;
decrement($i$);
$k := 2^i$; $j := i - 1$; $\widehat{p} := p_i$; $\widehat{q} := q_i$;
loop
    $tmp := \widehat{p} \times p_i + 3\widehat{q} \times q_i$;
    if ($tmp \le s$) then
        $\widehat{q} := p_i \times \widehat{q} + q_i \times \widehat{p}$; $\widehat{p} := tmp$; $k := k + 2^j$; fi;
    decrement($j$);
until $j = 0$;
return $k, \widehat{p}$; $\{\widehat{p} = T_k(2)\}$
```

Once we know k, i.e., δ_1, we can set $s := s - T_k(2)$ and repeat the process to get δ_2 and so on. The number of field operations performed by the above algorithm is $O(\log_2 D)$ and we need to run it t times to compute all the δ_i. Therefore, the Chebyshev exponents can be computed by performing $O(t \log_2 D)$ field operations. D is chosen to be $2\lceil \log_2(s) \rceil$ and

$$s = -\phi_{t-1}/2 = \sum_{j=1}^{t} T_{\delta_j}(a) < 1/2(\beta^{\delta_1+1} + \bar{\beta}^{\delta_1+1}) < 4^{\delta_1}.$$

Therefore, the number of bits in the integers involved in the root finding step is no more than $2\delta_1$.

Solving $\mathcal{V}\vec{\mathsf{f}} = \vec{\mathsf{a}}$. $\mathcal{V}$ is the transpose of a Vandermonde-like matrix. An $n \times n$ nonsingular Vandermonde matrix can be inverted by using $O(n^2)$ field operations and $O(n)$ space (Zippel (1990)). More efficient algorithms ($O(n \log^2 n \log(\log n))$ time) that use fast polynomial multiplication are also known (Canny, Kaltofen, and Lakshman (1989)). In the following discussion, we adapt Zippel's algorithm for solving the system of equations $\mathcal{V}\vec{\mathsf{f}} = \vec{\mathsf{a}}$. Consider the product

$$\mathcal{V}^T\mathcal{A} = \begin{pmatrix} T_0(T_{\delta_1}(a)) & T_1(T_{\delta_1}(a)) & \dots & T_{t-1}(T_{\delta_1}(a)) \\ T_0(T_{\delta_2}(a)) & T_1(T_{\delta_2}(a)) & \dots & T_{t-1}(T_{\delta_2}(a)) \\ \vdots & \vdots & \ddots & \vdots \\ T_0(T_{\delta_t}(a)) & T_1(T_{\delta_t}(a)) & \dots & T_{t-1}(T_{\delta_t}(a)) \end{pmatrix} \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1t} \\ c_{21} & c_{22} & \dots & c_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ c_{t1} & c_{t2} & \dots & c_{tt} \end{pmatrix}.$$

This product is the matrix

$$\begin{pmatrix} P_1(T_{\delta_1}) & P_2(T_{\delta_1}) & \dots & P_t(T_{\delta_1}) \\ P_1(T_{\delta_2}) & P_2(T_{\delta_2}) & \dots & P_t(T_{\delta_2}) \\ \vdots & \vdots & \ddots & \vdots \\ P_1(T_{\delta_t}) & P_2(T_{\delta_t}) & \dots & P_t(T_{\delta_t}) \end{pmatrix},$$

where $P_j(z)$ is the polynomial

$$P_j(z) = c_{1j}T_0(z) + c_{2j}T_1(z) + c_{3j}T_2(z) + \cdots + c_{tj}T_{t-1}(z).$$

Notice that by choosing

$$P_j(z) = \prod_{i \neq j} \frac{z - T_{\delta_i}(a)}{T_{\delta_j}(a) - T_{\delta_i}(a)},$$

the product becomes the identity matrix. In other words, the jth column of the inverse of $\mathcal{V}^T$ is made up of the coefficients of $P_j(z)$ *expressed in the Chebyshev basis.* Let $P_{j,i}$ denote the coefficient of $T_i(z)$ in $P_j(z)$. Since the inverse of the transpose is the transpose of the inverse, the system of equations $\mathcal{V}\vec{\mathsf{f}} = \vec{\mathsf{a}}$, i.e.,

$$\begin{pmatrix} T_{\delta_1}(T_0(a)) & T_{\delta_2}(T_0(a)) & \dots & T_{\delta_t}(T_0(a)) \\ T_{\delta_1}(T_1(a)) & T_{\delta_2}(T_1(a)) & \dots & T_{\delta_t}(T_1(a)) \\ \vdots & \vdots & \ddots & \vdots \\ T_{\delta_1}(T_{t-1}(a)) & T_{\delta_2}(T_{t-1}(a)) & \dots & T_{\delta_t}(T_{t-1}(a)) \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_t \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix},$$

has the solution

$$f_j = a_0 P_{j,0} + a_1 P_{j,1} + \cdots + a_{t-1} P_{j,t-1} \quad \text{for} \quad j = 1, \dots, t.$$

The coefficients $P_{j,i}$ can be computed as follows. Let

$$P(z) = \prod_{k=1}^{t} (z - T_{\delta_k}(a)).$$

($P(z)$ is actually $\Phi(z)/2^{t-1}$, $\Phi(z)$ being the auxiliary polynomial computed in step 2 of the interpolation algorithm.) Let

$$\widehat{P}_j(z) = P(z)/(z - T_{\delta_j}(a)),$$

be expressed in the Chebyshev basis. We can compute $\widehat{P}_j(z)$ from $P(z)$ by "synthetic Chebyshev division"—use the identity

$$T_k(z) = (z - T_{\delta_j}(a)) \times 2T_{k-1}(z) + 2T_{\delta_j}(a)T_{k-1}(z) + T_{k-2}(z) \quad \text{for} \quad k > 1$$

$t - 1$ times on $P(z)$. This division can be performed by using $O(t)$ field operations. Let $D_j = \widehat{P}_j(T_{\delta_j}(a))$. Clearly,

$$P_j(z) = \widehat{P}_j(z)/D_j.$$

D_j can be computed by using $O(t)$ field operations (evaluate $\widehat{P}_j(z)$ at $T_{\delta_j}(a)$). Therefore, we can compute all the Chebyshev coefficients of $P_j(z)$ in $O(t)$ field operations, and from them we can compute f_j by using $O(t)$ field operations. Since each f_j needs $O(t)$ field operations in this scheme, we need $O(t^2)$ field operations to compute all the f_j, $j = 1, \dots, t$. Notice that we need $O(t)$ space to hold the coefficients of each $P_j(z)$. However, we can reuse the same space for all the $P_j(z)$. Hence, we only need $O(t)$ storage units. Also, notice that the rational numbers encountered in this scheme have numerators and denominators with $O(t \log(T_{\delta_1}(a)))$ bits. Notice that $\log(T_{\delta_1}(a) = O(d))$, where d is the degree of the interpolating polynomial. We can now add up the costs of the four steps and we have the next theorem.

THEOREM 7. *The sparse Chebyshev interpolation algorithm performs $O(t^2 + t \log d)$ field operations to interpolate a t-Chebyshev-sparse polynomial given by a black box. Here d is the degree of the polynomial being interpolated (not part of the input). The algorithm makes $2t$ queries to the black box evaluating the polynomial. The numerator and denominator of the rational numbers that arise during the algorithm have $O(td)$ bits.* □

Remark 1. We may not know the exact number of terms in the polynomial, just only an upper bound $\tau \geq t$. In such a case, we can still use the algorithm, except that the matrix $\mathcal{A}$ will be singular (for $\tau > t$). The solution is to use the largest nonsingular leading principle minor of $\mathcal{A}$ in place of $\mathcal{A}$. Its rank gives the exact number of terms in the interpolating polynomial. The details are worked out for the Ben-Or and Tiwari algorithm in Kaltofen and Lakshman (1988) and are essentially the same for this case.

Remark 2. We have used classical algorithms for solving the various subproblems in the interpolation algorithm. One can use asymptotically faster algorithms to improve the overall complexity of the interpolation algorithm to $O(t(\log^2 t \log(\log t) + \log d))$ field operations. See Kaltofen and Lakshman (1988) and Canny et al. (1989).

Remark 3. The algorithm that we have discussed can be viewed as a generalization of the Ben-Or and Tiwari algorithm which in turn is based on the BCH decoding algorithm. The generalization uses a crucial property of the Chebyshev polynomials, namely, that they commute with respect to composition, i.e., $T_n(T_m(z)) = T_m(T_n(z))$. It turns out that this property is rather special and the only *families* of polynomials that exhibit this property are the Chebyshev polynomials and the standard powers $\{1, z, z^2, \dots\}$. More precisely, it is known that if in a family of polynomials $f_0(z), f_1(z), f_2(z), \dots$ with $\deg(f_i(z)) = i$, it is true that $f_i(f_j(z)) = f_j(f_i(z))$ for any $i, j \geq 0$, then either $f_i(z) = \lambda((\lambda^{-1}(z))^i)$ for all i or $f_i(z) = \lambda(T_i(\lambda^{-1}(z)))$ for all i, where $\lambda(z) = az + b$ and $\lambda^{-1}(z) = (z - a)/b$ for some constants a, b. (See, for instance, Rivlin (1974).)[1] The algorithm can be adapted easily to interpolate polynomials that are sparse in such "generalized" power bases or Chebyshev bases.

One can consider several possible "natural" bases and polynomials that are sparse in those bases. For example, suppose we know that the polynomial given by a black box is t-sparse in the basis $1, x - \alpha, (x - \alpha)^2, (x - \alpha)^3, \dots$ for some unknown α. Two recent papers discuss algorithms for solving this problem (Grigoriev and Karpinski (1992) and Lakshman and Saunders (1994)).

Sparsity in the standard basis is related to the number of real roots of the polynomial (sparsity limits the number of variations in sign), which is not the case when we consider polynomials that are sparse in the Chebyshev or Pochhammer bases. What are the corresponding observations about sparsity in these bases?

Acknowledgments. We thank our colleagues Prof. Bobby Caviness, Prof. David Wood, and Prof. Jeremy Johnson for several helpful discussions. Our thanks also to Prof. Michael Singer of North Carolina State University for detailed comments on an earlier draft.

[1]We are grateful to Dr. Jeremy Johnson of Drexel University for bringing this to our attention.

REFERENCES

M. Ben Or and P. Tiwari, (1988), *A deterministic algorithm for sparse multivariate polynomial interpolation*, Proc. 20th Symp. Theory of Computing, ACM Press, New York, pp. 301–309.

A. Borodin and P. Tiwari, (1990), *On the decidability of sparse univariate polynomial interpolation*, Proc. 22nd Symp. Theory of Computing, ACM Press, New York, pp. 535–545.

J. F. Canny, E. Kaltofen, and Y. N. Lakshman, (1989), *Solving systems of non-linear equations faster*, Proc. ISSAC 1989, Portland, OR, ACM Press, New York, pp. 121–128.

M. Clausen, A. Dress, J. Grabmeier, and M. Karpinski, (1988), *On zero testing and interpolation of k-sparse multivariate polynomials over finite fields*, Tech. Rep. 88.06.006, IBM Germany, Heidelberg Scientific Center, June 1988.

A. Dress and J. Grabmeier, (1991), *The interpolation problem for k-sparse polynomials and Character sums*, Adv. in Appl. Math, 12, pp. 57–75.

D. Grigoriev and M. Karpinski, (1987), *The matching problem for bipartite graphs with polynomially bounded permanents is in* NC, Proc. 28th IEEE Symp. Foundations Comp. Sci., pp. 166–172.

D. Grigoriev, M. Karpinski, and M. Singer, (1990), *Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields*, SIAM J. Comput., 19, pp. 1059–1063.

———, (1991a), *The interpolation problem for k-sparse sums of eigenfunctions of operators*, Adv. in Appl. Math., 12, pp. 76–81.

———, (1994), *Computational complexity of sparse rational interpolation*, SIAM J. Comput., 23, pp. 1–11.

D. Grigoriev and M. Karpinski, (1992), *A zero-test and an interpolation algorithm for the shifted sparse polynomials*, Proc. AAECC 1993, Puerto-Rico, Lecture Notes in Comput. Sci. 673, Springer-Verlag, New York, pp. 162–169.

D. Grigoriev, M. Karpinski, and A. M. Odlyzko, (1992), *Existence of short proofs of non-divisibility of sparse polynomials under the extended Riemann hypothesis*, Proc. International Symposium on Symbolic and Algebraic Computation 92, ACM Press, New York, pp. 117–122.

E. Kaltofen and Y. N. Lakshman, (1988), *Improved sparse multivariate polynomial interpolation algorithms*, Proc. International Symposium on Symbolic and Algebraic Computation 1988, Rome, Italy, Lecture Notes in Comput. Sci. 358, Springer-Verlag, New York, pp. 467–474.

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

Y. N. Lakshman and B. D. Saunders, (1994), *On Computing sparse shifts for univariate polynomials*, Proc. International Symposium on Symbolic and Algebraic Computation 94, ACM Press, New York, pp. 108–113.

R. Loos, (1983), *Computing rational zeros of integral polynomials by p-adic expansion*, SIAM J. Comput., 12, pp. 286–293.

G. A. Merchant and T. M. Parks, (1982), *Efficient solution of a Toeplitz-plus-Hankel matrix system of equations*, IEEE Trans. Acoustics, Speech, and Signal Proc., 30, pp. 40–44.

T. Rivlin, (1974), *The Chebyshev polynomials*, John Wiley, New York.

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