Improved Sparse Multivariate Polynomial Interpolation Algorithms*

Erich Kaltofen and Lakshman Yagati

Department of Computer Science
Rensselaer Polytechnic Institute
Troy, New York, 12180-3590
Inter-net: kaltofen@cs.rpi.edu, yagatil@cs.rpi.edu

ABSTRACT. We consider the problem of interpolating sparse multivariate polynomials from their values. We discuss two algorithms for sparse interpolation, one due to Ben-Or and Tiwari (1988) and the other due to Zippel (1988). We present efficient algorithms for finding the rank of certain special Toeplitz systems arising in the Ben-Or and Tiwari algorithm and for solving transposed Vandermonde systems of equations, the use of which greatly improves the time complexities of the two interpolation algorithms.

1. Introduction

We consider the problem of interpolating a multivariate polynomial over a field of characteristic zero from its values at several points. While techniques for interpolating dense polynomials have been known for a long time (e.g., Lagrangian interpolation formula for univariate polynomials), and probabilistic algorithms for interpolating sparse multivariate polynomials have existed since 1979 (Zippel 1979, 1988), until recently no algorithm was known to interpolate sparse multivariate polynomials deterministically in polynomial-time.

Ben-Or and Tiwari (1988) have recently proposed a deterministic sparse interpolation algorithm based on decoding BCH codes and certain special substitutions used in (Grigoryev and Karpinski 1987). In this paper, we discuss the algorithm of Ben-Or and Tiwari and also a probabilistic algorithm due to Zippel (1988). The Ben-Or and Tiwari algorithm has time complexity $O(nd\tau^3 \log(n))$ (where n is the number of variables, d is the total degree and τ is an upper bound on the number of terms in the polynomial) on an algebraic RAM (i.e., the operations $+, -, \times$ and $\div$ are considered as unit step operations). It evaluates the polynomial to be interpolated at 2τ distinct points. Zippel's algorithm performs $O(ndt^2)$ arithmetic operations to interpolate the sparse polynomial. Here t is the actual number of terms and not an upper bound. The algorithm performs $O(ndt)$ distinct evaluations of the polynomial.

We present new efficient algorithms to

1) find the rank of and solve a special Toeplitz system of equations arising in the Ben-Or and Tiwari Interpolation algorithm by making use of results by Brent, Gustavson and Yun (1980) in time quasi-linear in τ (i.e., $\tau \times$ poly-log(τ)), and

*This material is based on work supported by the National Science Foundation under Grant No. CCR-87-05363 and by an IBM faculty development award. Appears in Symbolic Algebraic Comput. Internat. Symp. ISSAC '88 Proc., P. Gianni, editor, Springer Lecture Notes in Computer Science, vol. 358, pp. 467–474 (1988).

2) solve a transposed Vandermonde system in time quasi-linear in t.

The use of our algorithms improves the running time of the Ben-Or and Tiwari algorithm to $O(dn\log(n)\,\tau \mathrm{M}(\tau)\log(\tau))$, where $\mathrm{M}(\tau)$ denotes the complexity of multiplying two univariate polynomials of degree τ. Our fast algorithm for solving transposed Vandermonde systems also improves the running time of Zippel's interpolation algorithm from quadratic in the number of terms t to quasi-linear in t.

The rest of the paper is organized as follows. We first set up some notation and then give brief descriptions of the Ben-Or and Tiwari algorithm and Zippel's algorithm. We then describe our algorithms for finding the rank of special Toeplitz systems and solving transposed Vandermonde systems. In §6, we discuss some difficulties in a root finding step in the Ben-Or and Tiwari algorithm and in conclusion, we mention an application of the interpolation algorithms.

Notation:

Let $P(x_1,\ldots,x_n) = c_1m_1 + c_2m_2 + \ldots + c_tm_t$ be the polynomial to be interpolated. The $m_i = x_1^{e_{i,1}}\ldots x_n^{e_{i,n}}$ are distinct monomials and the c_i are the corresponding non-zero coefficients; t is the number of terms in P. Let $v_i = p_1^{e_{i,1}}\ldots p_n^{e_{i,n}}$ denote the value of the monomial m_i at $(p_1,\ldots,p_n)$ where p_i is the i-th prime number. Clearly, different monomials evaluate to different values under this evaluation. Let $a_0, a_1, \ldots, a_{2\tau+1}$ denote the value of P at the $2(\tau+1)$ points $(p_1^0,\ldots,p_n^0)$, $\ldots$, $(p_1^{2(\tau+1)-1},\ldots,p_n^{2(\tau+1)-1})$. We have $a_i = \sum_{j=1}^{\tau+1} c_j v_j^i$.

2. The Ben-Or and Tiwari Interpolation Algorithm

The algorithm needs as input an upper bound $\tau+1 \geq t$ on the number of terms in P. The algorithm proceeds in two stages. The monomial values v_i are determined first by the use of an auxiliary polynomial $\zeta(z)$. Once the v_i are known, the coefficients c_i can be obtained easily.

The polynomial $\zeta(z)$ is defined as follows. Let

$$\zeta(z) = \prod_{i=1}^{t}(z - v_i) = z^t + \zeta_{t-1}z^{t-1} + \ldots + \zeta_1 z + \zeta_0.$$

Consider the sum

$$\sum_{i=1}^{t} c_i v_i^j \zeta(v_i) = \sum_{k=0}^{t-1} \zeta_k(c_1v_1^{k+j} + c_2v_2^{k+j} + \ldots + c_tv_t^{k+j}) + (c_1v_1^{t+j} + c_2v_2^{t+j} + \ldots + c_tv_t^{t+j})$$

for all j, $0 \leq j \leq t-1$. Since $\zeta(v_i) = 0$, we have

$$a_j\zeta_0 + a_{j+1}\zeta_1 + \ldots + a_{j+t-1}\zeta_{t-1} + a_{j+t} = 0, \qquad 0 \leq j \leq t-1.$$

We now have the Toeplitz system $T_{t-1,t-1}\hat{\zeta}_{t-1} = \hat{t}_{2t-1,t-1}$ where

$$T_{u,v} = \begin{pmatrix} a_u & a_{u+1} & \ldots & a_{u+v} \\ a_{u-1} & a_u & \ldots & a_{u+v-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{u-v} & a_{u-v+1} & \ldots & a_u \end{pmatrix}, \qquad \hat{\zeta}_v = \begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \vdots \\ \zeta_v \end{pmatrix}, \qquad \hat{t}_{u,v} = -\begin{pmatrix} a_u \\ a_{u-1} \\ \vdots \\ a_{u-v} \end{pmatrix}.$$

This system is non-singular as can be seen from the factorization

$$T_{t-1,t-1} = \begin{pmatrix} v_1^{t-1} & v_2^{t-1} & \dots & v_t^{t-1} \\ v_1^{t-2} & v_2^{t-2} & \dots & v_t^{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix} \begin{pmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_t \end{pmatrix} \begin{pmatrix} 1 & v_1 & \dots & v_1^{t-1} \\ 1 & v_2 & \dots & v_2^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & v_t & \dots & v_t^{t-1} \end{pmatrix}.$$

Since the v_i are distinct, the two Vandermonde matrices are nonsingular and as no c_i is zero, the diagonal matrix is nonsingular, too. If the input value of the upper bound $\tau + 1$ is greater than t, then the coefficients c_k, for $k > t$, can be regarded as zero and the resulting system $T_{\tau,\tau}$ would be singular.

Lemma 1. *If t is the exact number of terms in $P(x_1, \dots, x_n)$, then*
a) $T_{i,t-1}$ is non-singular for all $i \geq t-1$.
b) $T_{i,t+j}$ is singular for all $i \geq t-1, j \geq 0$.

Proof. Every $T_{u,v}$, $u, v \geq t-1$, can be factored like $T_{t-1,t-1}$ above. The statements a) and b) are obvious from such a factorization of $T_{u,v}$. ⊠

The roots of the polynomial $\zeta(z)$ give the v_i and by choosing the first t evaluations of P, we get the following transposed Vandermonde system of equations $A\hat{c} = \hat{a}$ for the coefficients of P, where

$$A = \begin{pmatrix} 1 & 1 & \dots & 1 \\ v_1 & v_2 & \dots & v_t \\ \vdots & \vdots & \ddots & \vdots \\ v_1^{t-1} & v_2^{t-1} & \dots & v_t^{t-1} \end{pmatrix}, \qquad \hat{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{pmatrix}, \qquad \hat{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} \qquad (1).$$

We now give a more precise description of the interpolation algorithm.

Algorithm *Polynomial Interpolation (by Ben-Or and Tiwari)*

Input: A black box for evaluating a multivariate polynomial $P(x_1, \dots, x_n)$ and an upper bound $\tau + 1$ on the number of terms in P.

Output: The polynomial $P = \sum_{i=1}^{t} c_i m_i$ where t, c_i, m_i all denote the same things as before.

Step 1: (Evaluation.) *Evaluate P at the $2(\tau+1)$ points $(p_1^0, \dots, p_n^0), \dots, (p_1^{2(\tau+1)-1}, \dots, p_n^{2(\tau+1)-1})$.* Let $a_0, \dots, a_{2\tau+1}$ be the corresponding values.

Step 2: (Computing the auxiliary polynomial $\zeta(z)$.) *Solve the Toeplitz system $T_{\tau,\tau}\hat{\zeta}_\tau = \hat{t}_{2\tau+1,\tau}$ (or the largest non-singular subsystem $T_{j,2\tau-j} = \hat{t}_{2\tau+1,2\tau-j}$ of $T_{0,\tau}$, if it is singular) to obtain the polynomial $\zeta(z) = \sum_{i=0}^{t} \zeta_i z^i$.*

Step 3: (Finding integer roots of $\zeta(z)$.) *Find the integer roots of $\zeta(z)$ to get the v_i. Compute the monomial m_i from v_i by repeatedly dividing v_i by $p_1, \dots, p_n$.*

Step 4: (Computing the coefficients.) *Find the coefficients c_i by solving the transposed Vandermonde system $A\hat{c} = \hat{a}$ described earlier.* □

Step 2 can be performed in $O(\tau^2)$ arithmetic operations using the Berlekamp-Massey algorithm (Blahut 1983). Step 3 can be performed in $O(t^3 \log(B))$ operations ($B = 2^{O(dn \log(n))}$ is an upper bound on the values of the roots, d being the total degree of the polynomial to be interpolated)

using the p-adic root finder in (Loos 1983). Step 4 can be performed in $O(t^2)$ arithmetic operations using Zippel's transposed Vandermonde inversion algorithm (Zippel 1988). The complexity of the root finding step clearly dominates that of the other steps and the overall complexity of *Polynomial Interpolation* is $O(\tau^2 + t^3 \log(B))$. The polynomial is evaluated at $2(\tau + 1)$ distinct points and the bit size of the evaluation points is $O(\tau n \log(n \log n))$.

3. Zippel's Interpolation Algorithm

We now describe briefly Zippel's probabilistic sparse interpolation algorithm. As before, $P = c_1 m_1 + c_2 m_2 + \ldots + c_t m_t$ where $m_i = x_1^{e_{i,1}} \ldots x_n^{e_{i,n}}$ are distinct monomials. Let $m_{i,k} = x_1^{e_{i,1}} \ldots x_k^{e_{i,k}}$ denote m_i restricted to the first k variables. Let $v_{i,k} = p_1^{e_{i,1}} \ldots p_k^{e_{i,k}}$ where p_i are distinct primes. Let

$$P_k(x_1, \ldots, x_k, b_{k+1}, \ldots, b_n) = c_{1,k} m_{i,k} + \ldots + c_{t',k} m_{t',k}$$

where $b_{k+1}, \ldots, b_n$ are randomly chosen from $\mathbb{Z}$, the integers, and $t' \leq t$. (It can happen that $m_{i,k} = m_{j,k}$ for $i \neq j$.) P_k is called the k-th skeleton of P. The algorithm proceeds in stages. In the k-th stage, P_{k+1} is obtained from P_k. The algorithm needs as input an upper bound d on the degree of P in each variable. Now,

$$P_{k+1}(x_1, \ldots, x_{k+1}, b_{k+2}, \ldots, b_n) = \hat{P}_1(x_{k+1}) m_{i,k} + \ldots + \hat{P}_{t'}(x_{k+1}) m_{t',k}$$

where each $\hat{P}_i(x_{k+1})$ is of degree at most d in x_{k+1}. The values of each $\hat{P}_i(x_{k+1})$ are obtained for $d+1$ distinct values $u_1, \ldots, u_{d+1}$ of x_{k+1} as follows:

$$\begin{aligned}
P_{k+1}(1, \ldots, 1, u_j, b_{k+2}, \ldots, b_n) &= \hat{P}_1(u_j) + \ldots + \hat{P}_{t'}(u_j) \\
P_{k+1}(p_1, \ldots, p_k, u_j, b_{k+2}, \ldots, b_n) &= \hat{P}_1(u_j) v_{1,k} + \ldots + \hat{P}_{t'}(u_j) v_{t',k} \\
P_{k+1}(p_1^2, \ldots, p_k^2, u_j, b_{k+2}, \ldots, b_n) &= \hat{P}_1(u_j) v_{1,k}^2 + \ldots + \hat{P}_{t'}(u_j) v_{t',k}^2 \\
&\vdots \\
P_{k+1}(p_1^{t'}, \ldots, p_k^{t'}, u_j, b_{k+2}, \ldots, b_n) &= \hat{P}_1(u_j) v_{1,k}^{t'} + \ldots + \hat{P}_{t'}(u_j) v_{t',k}^{t'}
\end{aligned}$$

This is a transposed Vandermonde system of equations and can be solved for the $\hat{P}_i(u_j)$ in $O(t^2)$ arithmetic operations. From the values at $v_1, \ldots, v_{d+1}$, each $\hat{P}_i(x_{k+1})$ can now be recovered by univariate interpolation. This step needs $O(d^2 t)$ arithmetic operations as there are at most t polynomial coefficients $\hat{P}_i(x_{k+1})$ to be recovered. The algorithm starts with $P(b_1, \ldots, b_n)$ for randomly chosen $b_1, \ldots, b_n$ and applies stages n times to get P. There is a small chance that the answer produced by this algorithm is wrong – this can happen if we choose a bad initial evaluation point (or the "anchor") $b_1, \ldots, b_n$. Zippel proves that if the b_i are chosen uniformly randomly from a set of size nd^2t^2/ϵ, then the probability of error is less than ϵ (for details, refer to (Zippel 1988).) Since there are n stages and each stage performs $O(dt^2)$ arithmetic operations, the interpolation algorithm performs $O(ndt^2)$ arithmetic operations in all. In each stage, the polynomial P is evaluated atmost $(d+1)t$ times. Therefore, the total number of evaluations performed is $O(ndt)$. The size of the evaluation points is $O(tn \log(n \log n))$.

4. Solving Toeplitz Systems

We have to solve the Toeplitz system $T_{\tau,\tau}\zeta_\tau = \hat{t}_{2\tau+1,\tau}$ in step 2 of *Polynomial Interpolation* to determine the auxiliary polynomial $\zeta(z)$. Efficient algorithms, such as the one due to (Brent et al 1980) are known for solving a non-singular Toeplitz system of equations. However, $T_{\tau,\tau}$ may be singular in which case we have to invert a largest non-singular block submatrix (i.e., made of contiguous rows and columns) of $T_{\tau,\tau}$. Equivalently, we want to find the smallest $j\,(\tau \le j \le 2\tau)$, such that $T_{j,2\tau-j}$ is non-singular. For the sake of clarity, $T_{j,2\tau-j}$ and $\hat{t}_{2\tau+1,2\tau-j}$ are displayed again:

$$T_{j,2\tau-j} = \begin{pmatrix} a_j & \dots & a_{2\tau} \\ a_{j-1} & \dots & a_{2\tau-1} \\ \vdots & \ddots & \vdots \\ a_{2j-2\tau} & \dots & a_j \end{pmatrix}_{(2\tau-j+1)\times(2\tau-j+1)} \quad \text{and} \quad \hat{t}_{2\tau+1,2\tau-j} = -\begin{pmatrix} a_{2\tau+1} \\ a_{2\tau} \\ \vdots \\ a_{j+1} \end{pmatrix}_{(2\tau-j+1)\times 1}.$$

We now extend the method of (Brent et al 1980) to find the rank of $T_{\tau,\tau}$, if it is singular. The algorithm uses a polynomial remainder sequence and the fundamental theorem of subresultants (Brown and Traub 1971).

Let $F_0 = x^{2\tau+1}$ and $F_1 = a_{2\tau}x^{2\tau} + ... + a_1x + a_0$. Let $S_j(F_0, F_1)$ denote the j-th subresultant of F_0 and F_1, i.e.

$$S_j(F_0,F_1) = \det\left(\begin{pmatrix} 1 & 0 & \dots & 0 & a_{2\tau} & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & a_{2\tau-1} & a_{2\tau} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & a_{j+1} & a_{j+2} & \dots & 0 \\ 0 & 0 & \dots & 0 & a_j & a_{j+1} & \dots & a_{2\tau} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_{2j-2\tau+1} & a_{2j-2\tau+2} & \dots & a_{j+1} \\ x^{2\tau-j-1}F_0 & x^{2\tau-j-2}F_0 & \dots & F_0 & x^{2\tau-j}F_1 & x^{2\tau-j-1}F_1 & \dots & F_1 \end{pmatrix}\right).$$

This is a polynomial in x of degree j and it is easily seen that its formal leading coefficient is $\det(T_{j,2\tau-j})$. Let F_{i-1} and F_i denote successive remainders in the polynomial remainder sequence of F_0 and F_1 such that degree of F_i is $\delta_i \le \tau$ and degree of F_{i-1} is $\delta_{i-1} > \tau$. In other words F_i is the first remainder in the remainder sequence of F_0 and F_1 whose degree is at most τ.

Theorem 1. *$T_{\tau,\tau}$ is non-singular iff $\delta_i = \tau$. If $\delta_i < \tau$ then $2\tau - \delta_{i-1} + 1$ is the size of the largest non-singular block submatrices of $T_{\tau,\tau}$.*

Proof. Let $\mathrm{ldcf}(f)$ denote the leading coefficient of the polynomial f. By the fundamental theorem of subresultants, if $\delta_i = \tau$ then $S_\tau(F_0, F_1) = bF_i$ where b is a non-vanishing scalar. Hence, $\mathrm{ldcf}(S_\tau(F_0, F_1)) = \mathrm{ldcf}(F_i)b$ i.e., $\det(T_{\tau,\tau}) = \mathrm{ldcf}(F_i)b$, a non-zero scalar, therefore $T_{\tau,\tau}$ is non-singular.

If $\delta_i < \tau$ then $S_\tau(F_0, F_1)$ is actually a polynomial of degree less than τ; hence its formal leading coefficient $\det(T_{\tau,\tau})$ has to vanish, i.e. $T_{\tau,\tau}$ is singular.

Now consider $S_{\delta_{i-1}}(F_0, F_1)$. It is an associate of F_{i-1}. Hence, its leading coefficient $\det(T_{\delta_{i-1},2\tau-\delta_{i-1}})$ is non-zero. By the fundamental theorem of subresultants, for $\delta_{i-1} - 1 \ge j \ge \delta_i$, every $S_j(F_0, F_1)$ vanishes identically. Hence their leading coefficients, $\det(T_{i,2\tau-i})$ vanish.

$S_{\delta_{i-1}-1}(F_0, F_1)$ is an associate of $S_{\delta_i}(F_0, F_1)$. But it is formally a polynomial of degree $\delta_{i-1} - 1$ and therefore, its leading coefficient $\det(T_{\delta_{i-1}-1})$ vanishes unless $\delta_{i-1} - 1 = \delta_i$ which would then be equal to τ. Therefore, for $\delta_{i-1} > j > \delta_i$, $\det(T_{j,2\tau-j}) = 0$.

If $\tau + 1 > T$ then $T_{\tau,\tau}$ is singular and it has just been proven that for $\delta_{i-1} > j > \delta_i$, $T_{j,2\tau-j}$ is singular. By Lemma 1 (§2), for $\delta_{i-1} > j > \delta_i$ and for all k, $T_{k,2\tau-j}$ is singular. Since $T_{\delta_{i-1},2\tau-\delta_{i-1}}$ is the largest block submatrix of $T_{\tau,\tau}$ that is non-singular, again by Lemma 1, $2\tau - \delta_{i-1} + 1$ is the exact number of terms in the polynomial $P(x_1, x_2, \ldots, x_k)$. ⊠

We can now use the scheme of Brent, Gustavson, and Yun to compute $T^{-1}_{\delta_{i-1},2\tau-\delta_{i-1}} \hat{t}_{\delta_{i-1}+1,2\tau-\delta_{i-1}}$. The remainders F_{i-1} and F_i can be computed in $O(\mathrm{M}(\tau)\log(\tau))$ arithmetic operations by the algorithm PRSDC in (Brent et al 1980). Here $\mathrm{M}(n)$ is the number of arithmetic operations needed to multiply two degree n polynomials. The algorithm of Brent et al to solve a non-singular Toeplitz system performs $O(\mathrm{M}(\tau)\log(\tau))$ arithmetic operations. Therefore the task of finding the rank of $T_{\tau,\tau}$ and computing the auxiliary polynomial $\zeta(z)$ can be performed in $O(\mathrm{M}(\tau)\log(\tau))$ arithmetic operations.

5. Solving Transposed Vandermonde Systems

We now describe an efficient algorithm to solve a transposed Vandermonde system of equations

$$A\hat{x} = \hat{a} \tag{2}$$

where

$$A = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ v_1 & v_2 & \ldots & v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_1^{n-1} & v_2^{n-1} & \ldots & v_n^{n-1} \end{pmatrix}, \qquad \hat{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \qquad \hat{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}.$$

This is needed in the final step of the algorithm *Polynomial Interpolation* and in every stage of Zippel's algorithm.

Let $B = A^{\mathrm{Tr}}$ (i.e., A-transposed). We have $\hat{x} = (B^{-1})^{\mathrm{Tr}}\hat{a}$. In (Zippel 1988), it is observed that if the j-th column $b_j = (b_{0,j}, b_{1,j}, \ldots, b_{(n-1),j})^{\mathrm{Tr}}$ of B^{-1} is regarded as the coefficients of $z^0, z^1, \ldots, z^{n-1}$ in the polynomial $B_j(z) = b_{0,j} + b_{1,j}z + \ldots + b_{(n-1),j}z^{n-1}$, then the (i,j)-th element of BB^{-1} is just

$$B_j(v_i) = \begin{cases} 1, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases}$$

Therefore,

$$B_j(z) = \prod_{\substack{1 \le k \le n \\ k \ne j}} \frac{z - v_k}{v_j - v_k}, \qquad 1 \le j \le n$$

and

$$\hat{x} = (B^{-1})^{\mathrm{Tr}}\hat{a} = \begin{pmatrix} \sum_{i=0}^{n-1} b_{i,1}\hat{a}_i \\ \vdots \\ \sum_{i=0}^{n-1} b_{i,n}\hat{a}_i \end{pmatrix}.$$

Notice that $x_i = \text{coefficient}(z^n)$ in $B_i(z)D(z)$ where

$$D(z) = a_0 z^n + a_1 z^{n-1} + \ldots + a_{n-2} z^2 + a_{n-1} z.$$

The x_i can be computed quickly without computing all the $B_i(z)D(z)$ by exploiting the fact that

$$B_i(z) = \frac{B(z)}{\alpha_i (z - v_i)} \qquad \text{where} \quad B(z) = \prod_{1 \le j \le n} (z - v_j), \qquad \alpha_i = \prod_{j \ne i} (v_i - v_j).$$

Let

$$B(z)D(z) = q_{2n} z^{2n} + q_{2n-1} z^{2n-1} + \ldots + q_1 z + q_0.$$

The coefficient of z^j in the quotient $B(z)D(z)/(z - w)$ is given by

$$Q_j(w) = q_{2n} w^{j-1} + q_{2n-1} w^{j-2} + \ldots + q_{j+2} w + q_{j+1}.$$

Clearly, $Q_n(v_i) = \alpha_i x_i$. $Q_n(w)$ is a degree $n - 1$ polynomial and has to be evaluated at n points $v_1, v_2, \ldots, v_n$.

Lemma 2. *A univariate polynomial of degree n can be evaluated at $m \ge n$ distinct points in $O(m/n\, \mathrm{M}(n) \log(n))$ arithmetic operations.*

Proof. A degree n polynomial can be evaluated at n points in $O(\mathrm{M}(n)\, \log(n))$ arithmetic operations (Aho et al 1974). The lemma follows by dividing the set of m points into $\lceil (m/n) \rceil$ sets of n points each and evaluating the polynomial at all points of each of these sets. ⊠

By Lemma 2, $Q_n(w)$ can be evaluated at points $v_1, \ldots, v_n$ in $O(\mathrm{M}(n)\, \log(n))$ arithmetic operations.

The α_i still remain to be evaluated. Observe that $B'(z) = \mathrm{d}B(z)/\mathrm{d}z = \sum_i \prod_{j \ne i} (z - v_j)$. Therefore $B'(v_i) = \alpha_i$. All the α_i, $1 \le i \le n$, can be obtained by evaluating $B'(z)$ at points $v_1, \ldots, v_n$. This can be done efficiently using the algorithm that was used to evaluate $Q_n(w)$ earlier.

Following is a description of the algorithm to solve a transposed Vandermonde system of equations:

Algorithm *Invert*

Input: Entries $v_1, \ldots, v_n$ which make up the transposed Vandermonde matrix A; vector $\hat{a} = (a_1, \ldots, a_n)^{\mathrm{Tr}}$.

Output: Vector $\hat{x} = (x_1, \ldots, x_n)^{\mathrm{Tr}}$ where $\hat{x} = A^{-1}\hat{a}$.

Step 1: *Compute the polynomial $B(z) = \prod_{1 \le j \le n} (z - v_j)$ by the tree multiplication algorithm.*

Step 2: Let $D(z) = a_1 z^n + a_2 z^{n-1} + \ldots + a_{n-1} z^2 + a_n z$. *Compute $B(z)D(z)$ using fast polynomial multiplication.* Let $B(z)D(z) = q_{2n} z^{2n} + q_{2n-1} z^{2n-1} + \ldots + q_0$. *Read off the polynomial $Q_n(w) = q_{2n} w^{n-1} + q_{2n-1} w^{n-2} + \ldots + q_{n+2} w + q_{n+1}$ from $B(z)D(z)$.*

Step 3: *Evaluate $Q_n(w)$ at points $v_1, \ldots, v_n$ to obtain $\alpha_i x_i$.*(The α_i are the scalars described earlier.)

Step 4: *Compute the α_i by evaluating $B'(z)$ at points $v_1, \ldots, v_n$.*

Step 5: *Output $(\alpha_i x_i / \alpha_i)_{i=1,\ldots,n}$.* □

The polynomial $B(z)$ in step 1 can be computed in $O(\mathrm{M}(n)\ \log(n))$ arithmetic operations using the tree multiplication algorithm (Aho et al 1974). Steps 3 and 4 are multipoint evaluation steps and they can be accomplished in $O(\mathrm{M}(n)\ \log(n))$ arithmetic operations by lemma 2. Step 2 is a univariate polynomial multiplication step and can be performed using $O(\mathrm{M}(n))$ arithmetic operations. Step 5 needs n divisions. Therefore the total time complexity of algorithm *Invert* is $O(\mathrm{M}(n)\ \log(n))$. The algorithm uses $O(n)$ space.

Using *Invert* in each stage of Zippel's algorithm reduces the time complexity of Zippel's algorithm to $O(nd\,\mathrm{M}(t)\ \log(t))$.

6. Finding Integer Roots

The p-adic root finder in (Loos 1983) computes the roots of $\zeta(z) \bmod p$ where p is a prime of magnitude $O(t^2 \log B)$ and B is a bound on the values of the roots (for $\zeta(z)$, $B = O(2^{dn\ \log(n)})$ where d is the total degree of the polynomial to be interpolated). It can be shown that such a prime separates all the roots of $\zeta(z)$ with high probability. The (mod p) roots are determined by evaluating $\zeta(z)$ at the points $0, 1, \ldots, p-1$. A straight forward evaluation of a degree t polynomial at $O(t^2 \log B)$ points needs $O(t^3 \log B)$ steps. The (mod p) roots are then lifted upto the bound B. Each lifting step costs $O(t)$ steps (linear Hensel lifting) and the lifting has to be performed $O(\log B)$ times at most. Therefore, the total complexity of the root finding step in *Polynomial Interpolation* is $O(t^3 \log B)$.

If fast evaluation is used, the evaluation of $\zeta(z)$ at points $0, 1, \ldots, p-1$ can be performed in $O(t\ \log(B)\,\mathrm{M}(t)\ \log(t))$ steps (Lemma 2). The total complexity of the root finder can thus be brought down to $O(ndt\,\mathrm{M}(t)\ \log(t)\log(n))$.

The overall complexity of *Polynomial Interpolation* as a result of the proposed improvements is $O(t\ \log(B)\,\mathrm{M}(t)\ \log(t)+\mathrm{M}(\tau)\ \log(\tau))$. The best known upper bound for $\mathrm{M}(t)$ is $O(t\ \log(t)\ \log(\log t))$ (Schönhage 1977, Cantor and Kaltofen 1987). Therefore, an upper bound on the arithmetic complexity of *Polynomial Interpolation* is

$$O(t^2 \log^2(t) \log(\log\ t) \log(B) + \tau\ \log^2(\tau)\ \log(\log\ \tau)).$$

7. Discussion

We have presented two sparse interpolation algorithms, and new efficient algorithms for finding the rank of certain special Toeplitz systems arising in the Ben-Or and Tiwari algorithm and for solving transposed Vandermonde system of equations. In *Polynomial Interpolation*, the root finding step is clearly the most expensive one. We believe that this step can be drastically improved by working with several small primes (as opposed to a single prime of magnitude $O(t^2 \log B)$). However, at this time we do not have a theoretical justification for this claim.

Using *Invert* reduces the arithmetic complexity of Zippel's probabilistic interpolation algorithm to $O(dn\mathrm{M}(t) \log(t))$. A modified version of Zippel's algorithm for dense interpolation has been used in (Canny et al 1988) for evaluating the Macaulay determinants of a system of non-linear polynomial equations. The sparse interpolation algorithms are also very useful in polynomial factorization as has been demonstrated in (Kaltofen and Trager 1988).

References

Aho, A., Hopcroft, J., and Ullman, J., *The Design and Analysis of Algorithms*; Addison and Wesley, Reading, MA, 1974.

Ben-Or, M. and Tiwari, P., "A deterministic algorithm for sparse multivariate polynomial interpolation," *20th Annual ACM Symp. Theory Comp.*, pp. 301–309 (1988).

Blahut, R. E., *Theory and Practice of Error Control Codes*; Addison-Wesley, Reading, MA, 1983.

Brent, R. P., Gustavson, F. G., and Yun, D. Y. Y., "Fast solution of Toeplitz systems of equations and computation of Padé approximants," *J. Algorithms* **1**, pp. 259–295 (1980).

Brown, W. S. and Traub, J. F., "On Euclid's algorithm and the theory of subresultants," *J. ACM* **18**, pp. 505–514 (1971).

Canny, J., Kaltofen, E., and Lakshman, Yagati, "Solving systems of non-linear polynomial equations faster," *Manuscript*, 1988.

Cantor, D. G. and Kaltofen, E., "Fast multiplication of polynomials with coefficients from an arbitrary ring," *Manuscript*, March 1987.

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

Kaltofen, E. and Trager, B., "Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators," *Proc. 29th Annual Symp. Foundations of Comp. Sci.*, (1988 (to appear)).

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

Schönhage, A. and Strassen, V., "Schnelle Multiplikation grosser Zahlen," *Computing* **7**, pp. 281–292 (1971). (In German).

Zippel, R. E., "Probabilistic algorithms for sparse polynomials," *Proc. EUROSAM '79, Springer Lec. Notes Comp. Sci.* **72**, pp. 216–226 (1979).

Zippel, R. E., "Interpolating polynomials from their values," *Manuscript*, Symbolics Inc., January 1988.