Reconstructing Curves in Three (and Higher) Dimensional Space from Noisy Data

Don Coppersmith*

Madhu Sudan†

ABSTRACT

We consider the task of reconstructing a curve in constant dimensional space from noisy data. We consider curves of the form $\mathcal{C} = \{(x, y_1, \ldots, y_c) \mid y_j = p_j(x)\}$, where the p_j's are polynomials of low degree. Given n points in $(c+1)$-dimensional space, such that t of these lie on some such unknown curve $\mathcal{C}$ while the other $n - t$ are chosen *randomly and independently*, we give an efficient algorithm to recover the curve $\mathcal{C}$ and the identity of the good points. The success of our algorithm depends on the relation between n, t, c and the degree of the curve $\mathcal{C}$, requiring $t = \Omega(n \deg(\mathcal{C}))^{1/(c+1)}$. This generalizes, in the restricted setting of random errors, the work of Sudan (J. Complexity, 1997) and of Guruswami and Sudan (IEEE Trans. Inf. Th. 1999) that considered the case $c = 1$.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computation on Polynomials*; E.4 [**Coding and Information Theory**]: Error control codes

General Terms

Algorithms, Reliabiity, Theory

Keywords

Error-correction, Maximum Distance Separable Codes, Random Errors, Decoding Algorithm, Algebraic Codes

1. INTRODUCTION

Inspired by a cryptosystem proposed by Kiayias and Yung [5] we consider a variant of the standard polynomial reconstruction problem. In the standard problem, one is given a sequence of distinct pairs $\langle(\alpha_i, \beta_i)\rangle_{i=1}^n$, where $\alpha_i, \beta_i \in \mathbb{F}$ for some field $\mathbb{F}$, and a degree parameter k and an agreement parameter t; and one wishes to find a degree k polynomial $p \in \mathbb{F}[x]$ such that $p(\alpha_i) = \beta_i$ for at least t values of $i \in [n]$. (Here and later, $[n]$ denotes the set of integers $\{1, \ldots, n\}$.) This problem is well-known to be equivalent to the decoding problem for Reed-Solomon codes and has been studied extensively in the past. The "list-decoding" algorithms of Sudan [6] and Guruswami and Sudan [4] solve the above problem provided $t > \sqrt{kn}$, and in fact produce a list of *all* polynomials that have the desired agreement with the given sequence of points.

The variant of the problem we consider is the following:

Curve Reconstruction Problem:

Given: A sequence $\langle(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\rangle_{i=1}^n$ with $\alpha_i, \beta_{i,j} \in \mathbb{F}$, α_i's being distinct; and degree parameters $k_1, \ldots, k_c$ and agreement parameter t.

Goal: Find polynomials $p_1, \ldots, p_c$ with $\deg(p_j) \le k_j$ for all $j \in [c]$, such that for every $j \in [c]$ there exist at least t values of $i \in [n]$ such that $p_j(\alpha_i) = \beta_{i,j}$.

Thus in the variant we are considering we are looking to reconstruct a curve in somewhat higher (though constant) dimensional space from erroneous evaluations. The only kind of curves we consider here are those formed by the intersection of surfaces of the form $\langle y_j - p_j(x)\rangle_{j\in[c]}$. Our goal is to be able to do so with far less agreement than required by the results of [4], as c grows large. To get such improvements, without actually improving the state of affairs vis-a-vis Reed-Solomon decoding, we need to make some additional assumptions and we do so below.

Restricting Assumptions:

Synchronized error locations: There exists a set $I \subseteq [n]$ with $|I| \ge t$ and polynomials $p_1, \ldots, p_c$ with $\deg(p_j) \le k_j$, such that for every $i \in I$ and $j \in [c]$, $p_j(\alpha_i) = \beta_{i,j}$.

Random errors: For every $i \notin I$ and $j \in [c]$, the value $\beta_{i,j}$ is distributed uniformly at random in $\mathbb{F}$ (independent of all other values in the sequence).

We use the term Restricted Curve Reconstruction Problem to describe the Curve Reconstruction Problem under the restricting assumptions above. In order to justify the two assumptions, note first that without the random error assumption, the standard Reed-Solomon decoding problem trivially reduces to the problem above: Given $\langle(\alpha_i, \beta_i)\rangle_{i=1}^n$, we set $\beta_{i,1} = \beta_i$ and $\beta_{i,2}, \ldots, \beta_{i,c} = 0$. A solution to the Curve

*IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598. email: dcopper@us.ibm.com.

†Massachusetts Institute of Technology, Laboratory for Computer Science, 200 Technology Square, Cambridge, MA 02139. email: madhu@mit.edu.

Supported in part by NSF Awards CCR 0205390 and MIT NTT Award 2001-04.

STOC'03, June 9–11, 2003, San Diego, California, USA.

Reconstruction Problem, now gives a solution to the Reed-Solomon decoding problem. Next, we note that without the synchronized error assumption, the problem above decomposes into c independent Reed-Solomon decoding problems with random error. In particular, the case $c = 1$ reduces to the case of larger c. While it is possible that this problem is easier than the list-decoding problem (with adversarial error), no better algorithms are known for this case either.

The main result of this paper is the following:

THEOREM 1. *For every fixed constant c, the Restricted Curve Reconstruction Problem can be solved in polynomial time, with probability at least $1 - O(n^{O(c)}/|\mathbb{F}|)$ over the random choice of errors, provided $t > \left(n \prod_{j=1}^{c} k_j\right)^{1/(c+1)} + \max_{j \in [c]}\{k_j\} + 1$,*

Note that the state of affairs does not improve for the standard case ($c = 1$) even if the errors are random. However as c grows, the requirement on the agreements above reduces. This is best seen by setting all k_j's equal to some k and noticing that requirement above simplifies to $t >\approx k \cdot (n/k)^{1/(c+1)}$ which gets closer to $O(k)$ as $c \to \infty$.

1.1 Coding theoretic interpretation

As mentioned earlier, our investigation was inspired by a cryptographic scheme proposed by Kiayias and Yung [5]. Our results show that their system breaks for certain choices of the parameters. Independent of our work, Bleichenbacher, Kiayias, and Yung [1] also give a solution to the Restricted Curve Reconstruction Problem. Their solution is weaker than ours (requires more agreement), and is incomparable with known results for list-decoding.

Our results also have a coding theoretic interpretation (analogous to those of [1]). Let $\Sigma = \mathbb{F} \times \mathbb{F}$ and let $Q = |\Sigma|$. Let $\alpha_1, \ldots, \alpha_n$ be distinct elements of $\mathbb{F}$ (and so $\sqrt{Q} \geq n$). Consider an error-correcting code whose messages are pairs of polynomials (p_1, p_2) of degree at most k (interpreted as elements of Σ^{k+1}), and whose encoding is the sequence $\langle (p_1(\alpha_i), p_2(\alpha_i)) \rangle_{i=1}^{n}$ (viewed as elements of Σ^n). This encoding scheme gives a code of minimum distance $n - k$ making it a Maximum Distance Separable (MDS) code. Our reconstruction algorithm shows how to recover the codeword from random errors over a Q-ary symmetric channel with error probability $1 - O((k/n)^{c/(c+1)})$.

Reinterpreting the above, for any $\alpha > 1$ we can get codes of rate $\Omega(\epsilon^{\alpha})$ that can correct $1-\epsilon$ fraction of random errors, with alphabet size that is polynomially large in the length of the message. The best previous results we are aware of, due to Guruswami and Indyk [2, 3], had rate growing quadratically in the parameter ϵ (though their results are stronger in that they can tolerate adversarial error, require only constant sized alphabets, and their decoding algorithm are (nearly) linear time).

1.2 Techniques

Our algorithm(s) have some similarities in their setup with algorithms of [6, 4]. But some of the ingredients in the algorithm, as well as their analysis are conceptually quite different. Here we stress some of the novelties.

First, let us describe the basic setup of the algorithms of [6, 4]. For the standard polynomial reconstruction problem with inputs $\langle (\alpha_i, \beta_i) \rangle_{i=1}^{n}$ their algorithm first finds a non-zero polynomial $Q(x, y)$ such that $Q(\alpha_i, \beta_i) = 0$ for every $i \in [n]$ (and possibly some additional conditions). Then they factor this polynomial Q, and show that factors of the form $y - p(x)$ for degree k polynomials p reveal all polynomials p with significant agreement with the input pairs.

Our approach starts off similarly. Note that the first step in the algorithms of [6, 4] amounts to solving a homogenous linear system. Say, the linear system they create defines a matrix $\mathbf{A}$ where they are looking for a vector in the left kernel (i.e., a vector $\mathbf{c}$ such that $\mathbf{c} \cdot \mathbf{A} = \mathbf{0}$). In our algorithm also we create a matrix, which turns out to be the same matrix $\mathbf{A}$ when $c = 1$. However we don't look for a vector in the left kernel of this matrix, but rather in its right kernel! We note that the coordinates of the vector in the right kernel correspond to the n given points; and that the non-zero coordinates of this vector are very likely to be the error-free locations. We this use this information to reduce the decoding problem to one of decoding with erasures - which is quite simple for any linear code. This makes our approach similar in spirit to the classical decoding algorithms that worked using "error-locator" polynomials.

The distinction between working with vectors in the left kernel versus vectors in the right kernel forces us to alter our analysis almost completely. For instance, where previously the task of proving that there is a vector in the kernel was a mere counting argument, in our case this becomes non-trivial and requires large number of agreements between the given point and some degree k polynomial. On the other hand we don't/can't handle non-unique solutions. In fact, the hard steps in our analyses show that for generic points the right kernel does not contain any non-zero vectors. Thus the only reason the right kernel has any non-zero vectors is because of the large agreement with some low-degree polynomial; and this manifests itself by forcing the null vector to have its support entirely on the "non-error" points.

We feel the new analysis, in addition to providing much stronger results in our setting, sheds new light on the previous decoding algorithms. Furthermore, the fact that we do not rely on a heavy duty tool such as factoring of multivariate polynomials, leads to hope that our algorithm may be applicable in settings where the previous approaches did not work.

Furthermore, while we only treat the case of special families of curves in $\mathbb{F}^{c+1}$ in this paper, our approach seems quite suitable to handle the case of general algebraic varieties in constant dimensional space. Indeed the main parameters of the problem space that the algorithm rely on seem intimately related to the algebraic notions of dimensions and degrees of varieties, suggesting that the algorithm could work in essentially the same way in the general setting.

1.3 Organization of this paper

In Section 2 we give some intuition into our approach and give a basic algorithm which corrects $n - \Omega\left(\left(n \prod_{j=1}^{c} k_j\right)^{1/(c+1)}\right)$ errors. (See Theorem 9.) This algorithm is analogous to the one of [6]. In Section 3, we augment our algorithm to correct up to $n - \left(n \prod_{j=1}^{c} k_j\right)^{1/(c+1)} - \max_{j \in [c]}\{k_j\}$ errors, thus yielding a proof of Theorem 1. This step is analogous to that of [4].

2. THE BASIC ALGORITHM(S)

Recall the problem we wish to solve. We are given a sequence $\langle(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c})\rangle_{i=1}^n$ and parameters $k_1, \ldots, k_c$ and t. We wish to find polynomials $p_1, \ldots, p_c$ with $\deg(p_j) \leq k_j$, and a set $I \subseteq [n]$ such that $|I| \geq t$ and $p_j(\alpha_i) = \beta_{i,j}$ for every $i \in I$ and $j \in [c]$.

In this section we'll give an algorithm to solve this problem with high probability in restricted case (correlated error locations and random errors), provided $t = \Omega((n \prod_{j=1}^c k_j)^{1/(c+1)})$.

2.1 Intuition and Motivation

We start by motivating our algorithm for the classical — "error less than half the minimum distance" — case with $c = 1$. So we wish to recover a single polynomial of degree at most k. We will skip the second subscript on $\beta_{i,j}$'s since it is always "1".

For every data point (α_i, β_i) we build a (column) vector $\mathbf{f}_{\alpha_i,\beta_i}$ as follows: Let ℓ be a parameter. Then $\mathbf{f}_{\alpha_i,\beta_i} \in \mathbb{F}^{2(\ell+1)-k}$ has its coordinates indexed by monomials of the form x^m, $0 \leq m \leq \ell$ or $x^m \cdot y$, $0 \leq m \leq \ell - k$. The coordinate corresponding to $x^m y^j$ in $\mathbf{f}_{\alpha_i,\beta_i}$ is $\alpha_i^m \beta_i^j$. Now given a collection of points $\langle(\alpha_i, \beta_i)\rangle_{i=n}$, we build a matrix $\mathbf{A} \in \mathbb{F}^{(2(\ell+1)-k)\times n}$ whose columns are indexed by points $i \in [n]$ and whose ith column is $\mathbf{f}_{\alpha_i,\beta_i}$.

The interest in the vectors $\mathbf{f}_{\alpha_i,\beta_i}$ and the matrix $\mathbf{A}$ arises from the following simple fact: If $\beta_i = p(\alpha_i)$ for some degree k polynomial p and every $i \in [n]$, then the matrix $\mathbf{A}$ has rank at most $\ell + 1$, independent of n. (To see this, note that the rows of $\mathbf{A}$ correspond to monomials of the form $x^m y^j$. Furthermore, since $y = p(x)$, rows corresponding to monomials of the form $y \cdot x^m$ are really just degree $m + k \leq \ell$ polynomials in x and thus can be expressed as linear combinations of the first $\ell+1$ rows of $\mathbf{A}$.) Now if we consider the case when $k \ll \ell \ll n$, then this upper bound on the rank of $\mathbf{A}$ is nearly a factor of two saving over the obvious rank of $2(\ell+1) - k \approx 2\ell$. In fact, it can be shown (and we'll do something like this later) that if the β_i's were generic or random, then indeed the rank $\mathbf{A}$ would be full.

Now if we introduce e errors, (i.e, for e values of $i \in [n]$, $\beta_i \neq p(\alpha_i)$), we still have that the matrix $\mathbf{A}$ has rank at most $\ell + e + 1$ which is still non-trivially small provided $\ell + e + 1 < \min\{n, 2(\ell + 1) - k\}$. Setting $\ell \approx (n + k)/2$ maximizes e under these constraints and gives $e \approx (n-k)/2$.

Thus the matrix $\mathbf{A}$ distinguishes a random sequence of β_i's from a sequence generated from a sequence obtained from the values of p with few (e) corruptions. To make constructive use of the above distinguishing property, we look into the set of columns that are linearly dependent. We show that if the field size is sufficiently large, then no such dependence could involve a random β_i. Thus we conclude that the linear dependence "identifies" a large collection of good columns and this can be used to find the message polynomial p.

To correct errors beyond half the minimum distance, we now adopt a completely natural idea. We'll throw in more monomials of the form $x^m y^j$ provided we don't increase the rank of the matrix $\mathbf{A}$ that we develop this way (when all data points come from polynomial evaluations). It is clear that the rank remains bounded from above by ℓ provided $m + kj \leq \ell$. So we increase the dimension of the vectors $\mathbf{f}_{\alpha,\beta}$ by throwing in all such monomials. The resulting algorithm (analyzed later) corrects at least $n - \sqrt{2kn}$ random errors (matching the bound of [6]). Indeed the matrix we work with is the same as the one used by [6], however the reasoning leading to it is quite different. Finally with little modification, we enhance our algorithms so as to handle the cases $c = 2, 3, \ldots$ and get significantly better results as c increases.

2.2 Notations and Definitions

We'll use the letters x, y, z, sometimes with subscripts, to denote indeterminates. We'll use α, β, γ, again possibly with subscripts, to denote field elements.

For degree parameters $k_1, \ldots, k_c$ and a monomial $M = x^m y_1^{j_1} \cdots y_c^{j_c}$, its $(k_1, \ldots, k_c)$-weighted degree is $\sum_{i=1}^c k_i j_i$. If $k_1, \ldots, k_c$ are clear from context (i.e., whenever we wish to) we'll drop the parameters $k_1, \ldots, k_c$ and just refer to the weighted degree of a monomial. For degree parameters $k_1, \ldots, k_c$ and a bound ℓ, let $\mathcal{M}_{k_1,\ldots,k_c,\ell}$ be the set of monomials in $x, y_1, \ldots, y_c$ of weighted degree at most ℓ. For a set of monomials $\mathcal{M}$ in $x, y_1, \ldots, y_c$, and $\alpha, \beta_1, \ldots, \beta_c \in \mathbb{F}$, we'll let $\mathbf{f}_{\alpha,\beta_1,\ldots,\beta_c;\mathcal{M}}$ be the vector in $\mathbb{F}^{|\mathcal{M}|}$ whose coordinates are indexed by monomials M in $\mathcal{M}$ and whose Mth coordinate is $\alpha^m \beta_1^{j_1} \cdots \beta_c^{j_c}$ if $M = x^m y_1^{j_1} \cdots y_c^{j_c}$. Similarly $\mathbf{f}_{x,y_1,\ldots,y_c;\mathcal{M}}$ will simply be the vector whose Mth coordinate is the (formal) monomial M. Typically $\mathcal{M}$ will just be $\mathcal{M}_\ell$ for some ℓ that will be fixed once the input parameters to the algorithm are fixed. So, we'll omit ℓ and $\mathcal{M}$ in the subscripts.

For a positive integer r, and implicit positive integer c, set $\mathcal{S}_r = \{(d, e_1, \ldots, e_c) | d, e_i \geq 0; d + \sum e_i < r\}$, and $S_r = |\mathcal{S}_r| = \binom{c+r}{c+1}$. Given $(d, e_1, \ldots, e_c) \in \mathcal{S}_r$, let $\mathbf{f}^{[d,e_1,\ldots,e_c]}_{\alpha,\beta_1,\ldots,\beta_c;\mathcal{M}}$ be the vector in $\mathbb{F}^{|\mathcal{M}|}$ whose coordinates are indexed by monomials M in $\mathcal{M}$ and whose Mth coordinate is the coefficient of $u^d v_1^{e_1} \cdots v_c^{e_c}$ in $(\alpha + u)^m (\beta_1 + v_1)^{j_1} \cdots (\beta_c + v_c)^{j_c}$ if $M = x^m y_1^{j_1} \cdots y_c^{j_c}$, namely $\binom{m}{d}\binom{j_1}{e_1} \cdots \binom{j_c}{e_c} \alpha^{m-d} \beta_1^{j_1-e_1} \cdots \beta_c^{j_c-e_c}$. By $\mathbf{f}^{[r*]}_{\alpha,\beta_1,\ldots,\beta_c;\mathcal{M}}$ we denote the collection of all S_r such columns indexed by $\mathcal{S}_r$.

Similarly $\mathbf{f}^{[d,e_1,\ldots,e_c]}_{x,y_1,\ldots,y_c;\mathcal{M}}$ is the vector whose Mth coordinate is the formal monomial $\binom{m}{d}\binom{j_1}{e_1} \cdots \binom{j_c}{e_c} x^{m-d} y_1^{j_1-e_1} \cdots y_c^{j_c-e_c}$.

2.3 Reconstructing a single polynomial

Again we start with the case of $c = 1$ to describe our algorithm and analysis.

2.3.1 Algorithm

Given the outline and notation from earlier sections, our algorithm is easy to describe. The only issue is the choice of ℓ. We'll pick ℓ so that $|\mathcal{M}_\ell| \geq n$.

Given $\langle(\alpha_i, \beta_i)\rangle_{i=1}^n$ and degree parameter k, let $\ell = \sqrt{2kn}$ and $\mathcal{M} = \mathcal{M}_\ell$. Let $m = |\mathcal{M}|$.

Step 1 Let $\mathbf{A} \in \mathbb{F}^{m\times n}$ be the matrix whose ith column is $\mathbf{f}_{\alpha_i,\beta_i}$. Find a non-zero vector $\mathbf{b} \in \mathbb{F}^n$ such that $\mathbf{A} \cdot \mathbf{b} = \mathbf{0}$, if one exists.

Step 2 Let $J \subseteq [n]$ be the set of coordinates where $\mathbf{b}$ is non-zero. If there exists exactly one polynomial p' of degree at most k such that $p'(\alpha_i) = \beta_i$ for every $i \in J$, then output p', else output FAIL.

2.3.2 Analysis

The algorithm obviously runs in polynomial time with the dominating cost being that of solving the homogenous linear system in Step 1. Here we analyze the correctness of the algorithm.

For the analysis assume that there exists a polynomial p and a set $I \subseteq [n]$ with $|I| > \ell$, such that $\beta_i = p(\alpha_i)$ for every $i \in I$. Further, we'll assume that the sequence $\langle \beta_i \rangle_{i \in [n]-I}$ are uniform and independent random variables from $\mathbb{F}$. Our goal is to show that the output p' of the algorithm is the polynomial p, with high probability.

Our analysis goes via two simple claims and a harder lemma. The claims show that the matrix $\mathbf{A}$ does have a non-trivial right kernel and that the support of the vectors in the kernel is large enough to determine the polynomial p' uniquely in Step 2. The lemma shows that the support is contained entirely among the non-error points, with high probability. Together this shows that the reconstruction finds the right set of points with high probability.

CLAIM 2. *If $|I| > \ell$, then Step 1 is successful. I.e., there exists a non-zero vector $\mathbf{b}$ such that $\mathbf{Ab} = \mathbf{0}$.*

PROOF. Permute the columns of $\mathbf{A}$ and write it as $[\mathbf{B}|\mathbf{C}]$ where $\mathbf{B}$ corresponds to the columns in I and $\mathbf{C}$ to the rest. The rank of $\mathbf{C}$ is at most the number of columns and thus at most $n - |I|$. On the other hand, as noted in Section 2.1, the rank of $\mathbf{B}$ is at most ℓ since the rows corresponding to the monomials $x^0, \ldots, x^{\ell-1}$ generate it. Thus the rank of $\mathbf{A}$ is at most $n - |I| + \ell < n$. Thus it has less than full column rank and thus there exists a non-zero vector $\mathbf{b}$ such that $\mathbf{Ab} = \mathbf{0}$. □

CLAIM 3. *In Step 2, the J produced by the algorithm satisfies $|J| > \ell > k$.*

PROOF. This is simple since restricting the matrix $\mathbf{A}$ to the rows corresponding to the monomials x^m makes it a Vandermonde matrix with ℓ rows. Thus every set of ℓ columns of this matrix are linearly independent. □

LEMMA 4. *If $m \geq n$, then with probability at least $1 - n\ell/(kq)$, we have $J \subseteq I$.*

PROOF. Let $t = |I|$ be the number of non-errors. Assume, without loss of generality, that $I = [t]$. Let $\mathbf{A}_i$ denote the $m \times i$ matrix obtained by restricting $\mathbf{A}$ to its first i columns. Let $\mathbf{B}_i$ be the $m \times i$ matrix obtained by replacing the ith column of $\mathbf{A}_i$ by the vector $\mathbf{f}_{\alpha_i, y}$. (Thus the ith column of $\mathbf{B}_i$ has as entries, formal polynomials in y.) We'll show below that: (1) For every $i \in [n]$, the matrix $\mathbf{B}_i$ has rank strictly larger than the rank of the matrix $\mathbf{A}_{i-1}$ (assuming $m \geq i$). (2) For any fixed $i \in [n] - [t]$, the probability that the rank of $\mathbf{A}_i$ is less than that of $\mathbf{B}_i$ is at most $\ell/(kq)$. Thus with probability $1 - (n-t) \cdot \ell/(kq)$ none of the last $n - t$ columns are involved in any linear dependency and this implies the lemma.

(1) Assume $\mathbf{B}_i$ has the same rank as the matrix $\mathbf{A}_{i-1}$. This implies that for every vector $\mathbf{c}$ that satisfies $\mathbf{c} \cdot \mathbf{A}_{i-1} = \mathbf{0}$, it is also the case that $\mathbf{c} \cdot \mathbf{B}_i = \mathbf{0}$. Recalling that the rows of $\mathbf{A}$ correspond to monomials of weighted degree at most ℓ and columns correspond to points, this is equivalent to the following assertion: Every non-zero polynomial $Q(x, y)$ of weighted degree at most ℓ that satisfies $Q(\alpha_1, \beta_1) = \cdots = Q(\alpha_{i-1}, \beta_{i-1}) = 0$ also satisfies $Q(\alpha_i, y) = 0$. But now consider the smallest degree non-zero polynomial Q that satisfies $Q(\alpha_1, \beta_1) = \cdots = Q(\alpha_{i-1}, \beta_{i-1}) = 0$. (Such a polynomial does exist, since $m \geq n > \text{rank}(\mathbf{A}_i)$.) Since $Q(\alpha_i, y) = 0$, we must have $(x - \alpha_i)$ divides $Q(x, y)$. But then the polynomial $Q'(x, y) = Q(x, y)/(x - \alpha_i)$ also satisfies $Q'(\alpha_1, \beta_1) = \cdots = Q'(\alpha_{i-1}, \beta_{i-1}) = 0$ and has smaller degree than Q, contradicting the minimality of Q. Thus we conclude that $\mathbf{B}_i$ has rank greater than the rank of $\mathbf{A}_{i-1}$.

(2) Now we need to show that $\mathbf{A}_i$ is unlikely to have smaller rank than $\mathbf{B}_i$. For this consider a full rank square submatrix $\mathbf{C}$ in $\mathbf{B}_i$. Let $g(y)$ denote the determinant of $\mathbf{C}$. (By part (1), we know that this matrix includes the ith column, and so the determinant will be a formal polynomial in y.) Since $\mathbf{C}$ is non-singular, $g(y) \neq 0$. Furthermore, the degree of g is at most ℓ/k, since it is a linear polynomial in the entries of the ith column which are, in turn, of degree at most ℓ/k in y. Now, the matrix $\mathbf{A}_i$ is obtained from $\mathbf{B}_i$ by picking β_i at random from $\mathbb{F}$ and setting $y = \beta_i$. Thus the square submatrix corresponding to $\mathbf{C}$ in $\mathbf{A}_i$ has determinant $g(\beta_i)$. The probability that this is zero is at most (ℓ/kq) where $q = |\mathbb{F}|$.

This concludes the proof of the lemma. □

We thus conclude with the following theorem:

THEOREM 5. *The algorithm of Section 2.3.1 solves the restricted curve reconstruction problem for $c = 1$, with probability at least $1 - O(n^{1.5}/q)$, provided the number of errors is less than $n - \sqrt{2kn}$.*

PROOF. By Claims 2 and 3 it follows Step 1 will find a non-zero vector $\mathbf{b}$ whose support is at least $k+1$. Thus there will be at most one polynomial satisfying the condition of Step 2. Furthermore, by Lemma 4, we get that the support of $\mathbf{b}$ is completely over the non-errors with high probability and in such case, the polynomial output by the algorithm will be p. □

2.4 Recovering curves in higher dimensions

We now move to the case of larger, but constant, c. The algorithm as well as the analysis are easy to generalize. Again we need to pick ℓ so that $|\mathcal{M}_\ell| \geq n$. Using the approximation, $|\mathcal{M}_\ell| \approx \frac{\ell^{c+1}}{(c+1)! \prod_{j=1}^{c} k_j}$, we find we need to set $\ell = \left((c+1)! n \prod_j k_j\right)^{1/(c+1)}$.

2.4.1 Algorithm

Given $\langle (\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c}) \rangle_{i=1}^n$ and degree parameters $k_1, \ldots, k_c$, let $\ell = \left((c+1)! n \prod_j k_j\right)^{1/(c+1)}$ and $\mathcal{M} = \mathcal{M}_\ell$. Let $m = |\mathcal{M}|$.

Step 1 Let $\mathbf{A} \in \mathbb{F}^{m \times n}$ be the matrix whose ith column is $\mathbf{f}_{\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c}}$. Find a non-zero vector $\mathbf{b} \in \mathbb{F}^n$ such that $\mathbf{A} \cdot \mathbf{b} = \mathbf{0}$, if one exists.

Step 2 Let $J \subseteq [n]$ be the set of coordinates where $\mathbf{b}$ is non-zero. For $j \in [c]$ do: If there exists exactly one polynomial p'_j of degree at most k_j such that $p'_j(\alpha_i) = \beta_{i,j}$ for every $i \in J$, then output p'_j as the jth polynomial; else output FAIL.

2.4.2 Analysis

Once again, for the analysis we assume that there exists a sequence of polynomials $p_1, \ldots, p_c$ and a set $I \subseteq [n]$ such that $\beta_{i,j} = p_j(\alpha_i)$ for every $i \in I$ and $j \in [c]$. We'll also assume that the errors ($\beta i, j$ for $i \notin I$) are random. The claims below follow in essentially the same manner as earlier, and we skip their proofs.

CLAIM 6. *If $|I| > \ell$, then Step 1 is successful. I.e., there exists a non-zero vector $\mathbf{b}$ such that $\mathbf{Ab} = \mathbf{0}$.*

CLAIM 7. *In Step 2, the J produced by the algorithm satisfies $|J| > \ell > k$.*

We now move to the main lemma and its proof.

LEMMA 8. *If $m \geq n$, then with probability at least $1 - n\ell/(kq)$, we have $J \subseteq I$.*

PROOF. Again we let $t = |I|$ and assume w.l.o.g. that $I = [t]$. Let $\mathbf{A}_i$ denote the $m \times i$ matrix obtained by restricting $\mathbf{A}$ to its first i columns. Let $\mathbf{B}_i$ be the $m \times i$ matrix obtained by replacing the ith column of $\mathbf{A}_i$ by the vector $\mathbf{f}_{\alpha_i, y_1, \ldots, y_c}$.

First we show that $\mathrm{rank}(\mathbf{B}_i) > \mathrm{rank}(\mathbf{A}_{i-1})$: Assume otherwise, then this implies that every non-zero polynomial $Q(x, y_1, \ldots, y_c)$ of weighted degree at most ℓ that satisfies $Q(\alpha_{i'}, \beta_{i',j}) = 0$ for every $i' \in [i-1]$ and $j \in [c]$, also satisfies $Q(\alpha_i, y_1, \ldots, y_c) = 0$. But now consider the smallest degree polynomial Q that satisfies $Q(\alpha_1, \beta_{1,1}, \ldots, \beta_{1,c}) = \cdots = Q(\alpha_{i-1}, \beta_{i-1,1}, \ldots, \beta_{i-1,c}) = 0$. (Again, the fact that $m > \mathrm{rank}(\mathbf{A}_{i-1})$ implies such a polynomial Q must exist.) Since $Q(\alpha_i, y_1, \ldots, y_c) = 0$, we now have $(x - \alpha_i)$ divides $Q(x, y_1, \ldots, y_c)$. But then the polynomial $Q'(x, y_1, \ldots, y_c) = Q(x, y_1, \ldots, y_c)/(x - \alpha_i)$ also satisfies $Q'(\alpha_1, \beta_{1,1}, \ldots, \beta_{1,c}) = \cdots = Q'(\alpha_{i-1}, \beta_{i-1,1}, \ldots, \beta_{i-1,c}) = 0$ and has smaller degree than Q, contradicting the minimality of Q. Thus we conclude that $\mathbf{B}_i$ has rank greater than the rank of $\mathbf{A}_{i-1}$.

Next we note, as in the proof of Lemma 4, that the probability that the rank of $\mathbf{A}_i$ less than the rank of $\mathbf{B}_i$ is at most $\ell/(kq)$, where $k = \min_{i=1}^{c}\{k_j\}$. Thus with probability at least $1 - (n-t) \cdot \ell/(kq)$ none of the last $n - t$ columns are involved in any linear dependency and this implies the lemma. $\square$

We thus conclude with the following theorem:

THEOREM 9. *The algorithm of Section 2.4.1 solves the restricted curve reconstruction problem, with probability at least $1 - O(n^{(c+2)/(c+1)}/q)$, provided the number of errors is less than $n - \left((c+1)! n \prod_{j=1}^{c} k_j\right)^{1/(c+1)}$.*

PROOF. Follows immediately from Claims 6 and 7 and Lemma 8. $\square$

3. AN IMPROVED ALGORITHM

In this section we improve the algorithms of the previous section to correct roughly $n - \left(n \prod_{j=1}^{c} k_j\right)^{\frac{1}{c+1}}$ errors. We use technique analogous to those of [4], who improve the $n - \sqrt{2kn}$ bound to $n - \sqrt{kn}$. They achieve their improvement by first finding a polynomial $Q(x, y)$ that vanishes with multiplicity r, for some suitably large r, at each input point (α_i, β_i). They then factor this polynomial Q, showing that all solution polynomials $p(x)$ are included as factors of the form $y - p(x)$ of $Q(x, y)$.

We will follow a similar strategy. We define a similar homogenous linear system (same if $c = 1$) as they do to find the polynomial $Q(x, y)$. Now we look for a vector in the right kernel of the associated matrix (while the Q of [4] corresponds to a vector in the left kernel). We then interpret the non-zero coordinates of the vector in the right kernel as flagging "correct" data points (as opposed to the errors) and interpolate through these points to (hopefully) find the polynomials $p_1, \ldots, p_c$. We describe the algorithm without further motivation.

First, recall some notation, already introduced in Section 2.2. Let $\mathcal{M}_{k_1, \ldots, k_c, \ell}$ denote the set of monomials in $x, y_1, \ldots, y_c$ of $(k_1, \ldots, k_c)$-weighted degree at most ℓ. For a collection of monomials $\mathcal{M}$ in $x, y_1, \ldots, y_c$, $\alpha, \beta_1, \ldots, \beta_c \in \mathbb{F}$, and integers $d, e_1, \ldots, e_c$, we let $\mathbf{f}^{[d, e_1, \ldots, e_c]}_{\alpha, \beta_1, \ldots, \beta_c; \mathcal{M}}$ be the vector in $\mathbb{F}^{|\mathcal{M}|}$ whose coordinates are indexed by monomials in $\mathcal{M}$ whose value on the coordinate corresponding to $M = x^i y_1^{j_1} \cdots y_c^{j_c}$ is the coefficient of $x^d y_1^{e_1} \cdots y_c^{e_c}$ in $(x+\alpha)^i (y_1 + \beta_1)^{j_1} \cdots (y_c + \beta_c)^{j_c}$. Similarly for indeterminates $u, v_1, \ldots, v_c$, we'll let $\mathbf{f}^{[d, e_1, \ldots, e_c]}_{u, v_1, \ldots, v_c; \mathcal{M}}$ be the vectors whose coordinates are the corresponding formal polynomials in $u, v_1, \ldots, v_c$. Notice that the old notation $\mathbf{f}_{\alpha, \beta_1, \ldots, \beta_c; \mathcal{M}}$ corresponds, in the new notation, to $\mathbf{f}^{[0, \ldots, 0]}_{\alpha, \beta_1, \ldots, \beta_c; \mathcal{M}}$.

For positive integer r, let $\mathcal{S}_r = \{(d, e_1, \ldots, e_c) | d + \sum_i e_i < r\}$ and let $S_r = |\mathcal{S}_r|$. Note $S_r = \binom{c+r}{c+1}$. Let $\mathbf{f}^{[r*]}_{\alpha, \beta_1, \ldots, \beta_c; \mathcal{M}}$ denote the collection of all (column) vectors $\mathbf{f}^{[d, e_1, \ldots, e_c]}_{\alpha, \beta_1, \ldots, \beta_c; \mathcal{M}}$ for $(d, e_1, \ldots, e_c) \in \mathcal{S}_r$. This collection of vectors will be central to our algorithm of this section.

3.1 The Algorithm

Given as input, n tuples $\{(\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c}) | i \in [n]\}$, we proceed as follows:

Parameters Let r be a sufficiently large integer, and ℓ such that $|\mathcal{M}_\ell| \geq n \cdot S_r$. Let $m = |\mathcal{M}_\ell|$ and $N = n \cdot S_r$.

Step 1: Let $\mathbf{A} \in \mathbb{F}^{m \times N}$ be the matrix whose columns are indexed by pairs $(i, (d, e_1, \ldots, e_c))$ with $i \in [n]$ and $(d, e_1, \ldots, e_c) \in \mathcal{S}_r$ where the $(i, (d, e_1, \ldots, e_c))$th column is $\mathbf{f}^{[d, e_1, \ldots, e_c]}_{\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c}; \mathcal{M}_\ell}$. Let $\mathbf{b} \in \mathbb{F}^N$ be a non-zero vector such that $\mathbf{A} \cdot \mathbf{b} = \mathbf{0}$.

Step 2: Let J be the set of all indices $i \in [n]$ such that there exists a tuple $(d, e_1, \ldots, e_c) \in \mathcal{S}_r$ for which the $(i, (d, e_1, \ldots, e_c))$th coordinate of $\mathbf{b}$ is non-zero. If for every $j \in [c]$ there exists exactly one polynomial p'_j such that $p'_j(\alpha_i) = \beta_{i,j}$ for every $i \in J$, then output the tuple $(p'_1, \ldots, p'_j)$, else output FAIL.

Notice that the algorithm above specializes to the algorithm of Section 2.4.1 if we set $r = 1$. We show below that if r is sufficiently large, and so is $|\mathbb{F}|$, then the algorithm above terminates successfully.

3.2 Analysis

Let I denote the set of non-errors with $t = |I|$, and let $p_1, \ldots, p_c$ be the correct polynomials. We wish to show that, with high probability over the random choice of the error values, the algorithm does not output FAIL, and that the outputs $p'_1, \ldots, p'_c$ satisfy $p'_j = p_j$. We show this in a sequence of three claims: (1) The matrix $\mathbf{A}$ does have rank less than N and thus a vector $\mathbf{b}$ as required in Step 1 does exist. (2) With high probability, the subset J found in Step 2 is a subset of I. More specifically, we claim that with high probability, the columns of $\mathbf{A}$ do not have any linear dependencies involving any of the block of columns corresponding to $\mathbf{f}^{[r*]}_{\alpha_i, \beta_{i,1}, \ldots, \beta_{i,c}; \mathcal{M}_\ell}$ for $i \notin I$. (3) For any vector in the right kernel of $\mathbf{A}$ the associated set J (as found in Step 2) must be of size at least $\max_j\{k_j + 1\}$ and so there is at most one polynomial p'_j, for every $j \in [c]$, in Step 2. Combining the three claims with the observation that p_j's are polynomials satisfying the condition in Step 2, if $J \subseteq I$, we get that the outputs satisfy $p'_j = p_j$ with high probability.

We start by showing that the matrix $\mathbf{A}$ does show a column dependency. Let $\mathbf{B}$ be the $(\binom{c+r}{c+1} \cdot t) \times m$ matrix consisting of those columns of $\mathbf{A}$ corresponding to $i \in I$. We show in the claim below that the rank of $\mathbf{B}$ is less than $\binom{c+r}{c+1} \cdot t$, while the number of columns equals $\binom{c+r}{c+1} \cdot t$. Thus, we get that the columns of $\mathbf{B}$, and hence some columns of $\mathbf{A}$, are linearly dependent.

CLAIM 10. *If $t > \ell/r$, then the matrix $\mathbf{B}$ has a column dependency.*

PROOF. Let $T = \binom{c+r}{c+1} \cdot t$. Notice that $\mathbf{B}$ has T columns. We'll show that there exist strictly fewer than T vectors of dimension T that generate all the rows of $\mathbf{B}$, and thus its rank is smaller than the number of columns. Note that the rows of $\mathbf{B}$ correspond to monomials in $x, y_1, \dots, y_c$. Let $\mathbf{b}_M$ be the row corresponding to the monomial M. Let $\mathcal{M}_\ell^{[r,t]}$ be the set of monomials $x^d y_1^{e_1} \cdots y_c^{e_c}$ such that $(d/t) + e_1 + \cdots + e_c < r$ and $d + e_i k_i < \ell$. We'll show specifically that the set of vectors $\{\mathbf{b}_M | M \in \mathcal{M}_\ell^{[r,t]}\}$ generate all the rows of $\mathbf{B}$. Note that $|\mathcal{M}_\ell^{[r,t]}| < T$ provided $t > \ell/r$.

In order to show this we use the following interpretation of a row dependency: Suppose $\sum_M c_M \mathbf{b}_M = \mathbf{0}$. Then the polynomial $Q(x, y_1, \dots, y_c) = \sum_M c_M \cdot M$ vanishes with multiplicity r at the points $(\alpha_i, \beta_{i,1}, \dots, \beta_{i,c})$ for every $i \in I$. In other words, if we let $\mathcal{I}$ be the ideal generated by the polynomials $y_1 - p_1(x), \dots, y_c - p_c(x)$ and $\prod_{i \in I}(x - \alpha_i)$; and let $\mathcal{I}^r$ denote the rth power of this ideal. Then $Q(x, y_1, \dots, y_c) \in \mathcal{I}^r$. Conversely if $Q = \sum_M c_M M$ is in $\mathcal{I}^r$, then $\sum_M c_m \mathbf{b}_M = \mathbf{0}$. Thus it suffices to show, for every monomial $M \in \mathcal{M}_\ell$ that there exists coefficients $c_{M'}$ such that the polynomial $Q = M + \sum_{M' \in \mathcal{M}_\ell^{[r,t]}} c_{M'} M'$ is in $\mathcal{I}^r$. Equivalently, it suffices to show that any monomial $M \in \mathcal{M}_\ell$ can be reduced, modulo polynomials in $\mathcal{I}^r$, to a polynomial whose support lies in $\mathcal{M}_\ell^{[r,t]}$.

Fix a lexicographic total ordering on all monomials with $y_c > \cdots > y_1 > x$. Extend the ordering to a partial ordering on all polynomials where $Q_1 > Q_2$ if the largest monomial of Q_1 is larger than the largest monomial of Q_2, else, if the two are the same, then if the second largest monomial of Q_1 is larger than the second largest of Q_2 and so on. (So two polynomials are incomparable iff they have the same monomials in their support.) For a polynomial $M \in \mathcal{M}_\ell$, let Q be a smallest polynomial equivalent to $M (\bmod \mathcal{I}^r)$ with all its support from $\mathcal{M}_\ell$. We claim that all monomials in Q must be from $\mathcal{M}_\ell^{[r,t]}$. Assume otherwise and let $M' = x^d y_1^{e_1} \cdots y_c^{e_c}$ be the largest monomial of Q that is not in $\mathcal{M}_\ell^{[r,t]}$. Let $c_{M'}$ be the coefficient of M'. Let $d' = \lfloor d/t \rfloor$. By hypothesis we have $d' + e_1 + \cdots e_c \geq r$. So the polynomial $P = (\prod_{i \in I}(x - \alpha_i))^{d'} \cdot \prod_{j=1}^c (y_j - p_j(x))^{e_j}$ belongs to $\mathcal{I}^r$ and thus $M' = M' - x^{d - td'} P (\bmod \mathcal{I}^r)$. Furthermore P has no larger weighted degree than M' and thus P has support in $\mathcal{M}_\ell$. Finally the leading term of P is strictly smaller than M' (in our ordering of monomials) and so the polynomial $Q' = Q - c_{M'}(M' - x^{d-td'} P)$ equals Q modulo $\mathcal{I}^r$, has all its support in $\mathcal{M}_\ell$ and is smaller than Q' (in our ordering on polynomials), thereby contradicting our hypothesis. $\square$

Next we show that there are no dependencies involving a column $(i, (d, e_1, \dots, e_c))$ where $i \notin I$.

CLAIM 11. *With probability at least $1 - (N \cdot \ell)/(q \cdot \min_j\{k_j\})$, the matrix $\mathbf{A}$ has no linear dependencies involving any of the columns indexed by $(i, (d, e_1, \dots, e_c))$ where $i \notin I$, provided $|\mathcal{M}_{\ell - (r-1) \cdot \max_j\{k_j\}}| > N$.*

PROOF. For this part we assume $I = [t]$ and that we insert the columns of $\mathbf{A}$ one block at a time with the ith block being the vectors of $f^{[r*]}_{\alpha_i, \beta_{i,1}, \dots, \beta_{i,c}; \mathcal{M}_\ell}$. Let $\mathbf{A}_i$ be the matrix obtained after including i blocks. For $i > t$, we show that with high probability the vectors $f^{[r*]}_{\alpha_i, \beta_{i,1}, \dots, \beta_{i,c}; \mathcal{M}_\ell}$ are involved in no linear dependency involving themselves and the vectors in $\mathbf{A}_{i-1}$. We do so in two steps: First we show that when we insert the block corresponding to the generic point $(\alpha_i, z_1, \dots, z_c)$, where $z_1, \dots, z_c$ are indeterminates, there are no linear dependencies involving the new columns. Next we show that when we perform the random substitutions, $z_1 = \beta_{i,1}, \dots, z_c = \beta_{i,c}$, with high probability, we don't introduce any dependencies.

Phase 1: Assume we insert the columns $f^{[d, e_1, \dots, e_c]}_{\alpha_i, z_1, \dots, z_c; \mathcal{M}_\ell}$ into the matrix $\mathbf{A}_{i-1}$ in order of non-decreasing $d + \sum_{j=1}^c e_j$. Let $f^{[d, e_1, \dots, e_c]}_{\alpha_i, z_1, \dots, z_c; \mathcal{M}_\ell}$ be the first column that is linearly dependent on lower numbered columns. This implies that every polynomial $Q(x, y_1, \dots, y_c)$ (with coefficients from $\mathbb{F}(z_1, \dots, z_c)$ and monomials from $\mathcal{M}_\ell$) that (1) vanishes with multiplicity r at the first $i - 1$ points, and (2) whose "partial derivatives" corresponding to $(d', e'_1, \dots, e'_c)$, for previously inserted columns are zero at $(\alpha_i, z_1, \dots, z_c)$, also (3) has a vanishing $(d, e_1, \dots, e_c)$th partial derivative at $(\alpha_1, z_1, \dots, z_c)$. We'll show a polynomial with properties (1) and (2), which does not satisfy property (3), giving the desired contradiction.

Let $Q_1(x, y_1, \dots, y_c)$ be a polynomial satisfying property (1) of minimal weighted degree. By choice of ℓ we have such a polynomial Q_1 with weighted degree at most $\ell - (r-1) \cdot \max_j\{k_j\}$ exists. As in proof of Lemma 8, we have that $Q_1(\alpha_i, z_1, \dots, z_c) \neq 0$, or else $(x - \alpha_i)$ would divide Q_1 and then $Q_1/(x - \alpha_i)$ would be a lower degree polynomial satisfying property (1). Now let $Q(x, y_1, \dots, y_c) = Q_1 \cdot (x - \alpha_i)^d \cdot \prod_{j=1}^c (y_i - z_i)^{e_i}$. We claim that Q satisfies the required properties. First we have its weighted degree is at most ℓ as required. Next it satisfies property (1) since Q_1 satisfies this property. Next we observe, by inspection, that the only non-zero coefficient of total degree at most $d + \sum_{j=1}^c e_j$ in $Q(x + \alpha_i, y_1 + z_1, \dots, y_c + z_c)$ is the coefficient of $x^d y_1^{e_1} \cdots y_c^{e_c}$ which equals $Q_1(\alpha_1, z_1, \dots, z_c)$. Thus we have that Q satisfies property (2), but not (3), giving the desired contradiction.

Phase 2: Next we argue that the random substitution is unlikely to create any linear dependencies. Let $\mathbf{B}_i$ denote the matrix obtained above after inserting the columns corresponding to the generic point $(\alpha_i, z_1, \dots, z_c)$. Let $\mathbf{C}$ denote a full rank submatrix of $\mathbf{B}_i$ and let $p(z_1, \dots, z_c)$ be its determinant. By construction, we have p is a non-zero polynomial in $z_1, \dots, z_c$ of degree at most $\binom{c+r}{c+1} \cdot \ell/(\min_j\{k_j\})$. After the random substitution the value of determinant of the matrix corresponding to $\mathbf{C}$ is $p(\alpha_1, \dots, \alpha_c)$ which is non-zero with probability at least $1 - (\binom{c+r}{c+1} \cdot \ell)/(q \cdot \min_j\{k_j\})$.

Taking the union bound over $i = t+1, \dots, n$ yields the probability claimed in the claim. $\square$

By now we have argued that a dependent vector $\mathbf{b}$ can be found in Step 1, and with high probability the resulting set J will be contained in I. To conclude we show that the size of J is at least $\max_j\{k_j\}$. We do so next.

CLAIM 12. *There are no column dependencies in* $\mathbf{A}$ *involving fewer than* $\max_j\{k_j\}+1$ *blocks of columns, provided* $\ell \geq 2r \cdot \max_j\{k_j\}$.

PROOF. Assume there is a linear dependency involving the first i blocks of columns, where $i \leq \max_j\{k_j\}$. We will show that this yields a contradiction. Assume the dependency involves the ith block of columns. Insert the columns of this block one at a time, in non-decreasing order of the quantity $d+\sum_j e_j$. Suppose the first column to become dependent on previous blocks or previously inserted columns of this block is the one indexed $(i,(d,e_1,\ldots,e_c))$. Applying the interpretation of dependencies in terms of polynomials, we see that this implies that given any polynomial Q of weighted degree at most ℓ that (1) vanishes with multiplicity r at the first $i-1$ points, and further (2) has its $(d',e'_1,\ldots,e'_c)$th partial derivatives zero at the ith point if $d'+\sum_j e'_j \leq d+\sum_j e_j$ and $(d',e'_1,\ldots,e'_c) \neq (d,e_1,\ldots,e_c)$, it must be the case that (3) its $(d,e_1,\ldots,e_c)$th partial derivative at the ith point must also be zero.

To contradict the above, take the polynomial

$$Q = \Big(\prod_{i'=1}^{i-1}(x-\alpha_{i'})\Big)^r \cdot (x-\alpha_i)^d \cdot \prod_{j=1}^{c}(y_j-\beta_{i,j})^{e_j}.$$

Q satisfies conditions (1) and (2) above, but not (3). Furthermore, the weighted degree of Q is at most

$$r \cdot (i + \max_j\{k_j\}) \leq 2r \cdot (\max_j\{k_j\}) \leq \ell.$$

This yields the desired contradiction. $\square$

THEOREM 13. *The algorithm of Section 3.1 solves the restricted c-polynomial reconstruction problem, with probability at least* $1-O(n^{O(c)}/q)$, *provided the number of errors is less than* $n - \max\{\left(n\prod_{j=1}^{c}k_j\right)^{1/(c+1)} + k_{\max}+1, 2k_{\max}\}$, *where* $k_{\max} = \max_j\{k_j\}$.

PROOF. We set $r = O(cn)$ and ℓ to be

$$\max\left\{\begin{array}{c} 2rk_{\max}, \\ r\cdot\left(k_{\max}+c+(1+\frac{c}{r})\cdot\left(n\prod_{j=1}^{c}k_j\right)^{\frac{1}{c+1}}\right)\end{array}\right\}.$$

For this choice of ℓ we verify that the conditions of Claims 10-12 are satisfied. First, we have directly from the definition that $\ell \geq 2r \cdot k_{\max}$ as required for Claim 12. Next, we have

$$\begin{aligned} &\ell/r \\ &\leq \max\left\{2k_{\max}, k_{\max}+\frac{c}{r}+(1+\frac{c}{r})\cdot\left(n\prod_{j=1}^{c}k_j\right)^{\frac{1}{c+1}}\right\} \\ &\leq \max\left\{2k_{\max}, k_{\max}+\left(n\prod_{j=1}^{c}k_j\right)^{\frac{1}{c+1}}+1\right\} \\ &< t, \end{aligned}$$

as required for Claim 10. Finally, we also have

$$\begin{aligned} |\mathcal{M}_{\ell-rk_{\max}}| &\geq \frac{(\ell-rk_{\max}-c)^{c+1}}{(c+1)!\prod_{j=1}^{c}k_j} \\ &\geq \frac{(r+c)^{c+1}\cdot n\cdot\prod_{j=1}^{c}k_j}{(c+1)!\prod_{j=1}^{c}k_j} \\ &> \binom{r+c}{c+1}\cdot n \\ &= N \end{aligned}$$

So we conclude that Claim 11 holds with probability $1-O(N\cdot\ell/q) = 1-O(n^{O(c)}/q)$. We conclude that in Step 1, we do find a null vector $\mathbf{b}$ (from Claim 10), and that the corresponding set $J \subseteq I$ (from Claim 11) and that J is large enough to specify the polynomials p_j uniquely (Claim 12). So with probability $1-O(n^{O(c)}/q)$, we have that the algorithm correctly outputs $p_1,\ldots,p_c$. $\square$

4. REFERENCES

[1] D. Bleichenbacher, A. Kiayias, and M. Yung. Manuscript, 2002.

[2] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 658–667, Las Vegas, NV, October 2001.

[3] V. Guruswami and P. Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 812–821, Montreal, Quebec, 19-21 May 2002.

[4] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[5] A. Kiayias and M. Yung. Manuscript, 2002.

[6] M. Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.