Randomness Efficient Identity Testing of Multivariate Polynomials

Adam R. Klivans*
Laboratory for Computer Science
MIT
Cambridge, MA 02139
`klivans@math.mit.edu`

Daniel A. Spielman†
Department of Mathematics
MIT
Cambridge, MA 02139
`spielman@math.mit.edu`

Abstract

We present a randomized polynomial time algorithm to determine if a multivariate polynomial is zero using $O(\log mn\delta)$ random bits where n is the number of variables, m is the number of monomials, and δ is the total degree of the unknown polynomial. All other known randomized identity tests (see for example [7, 12, 1]) use $\Omega(n)$ random bits even when the polynomial is sparse and has low total degree. In such cases our algorithm has an exponential savings in randomness. In addition, we obtain the first polynomial time algorithm for interpolating sparse polynomials over finite fields of large characteristic. Our approach uses an error correcting code combined with the randomness optimal isolation lemma of [8] and yields a generalized isolation lemma which works with respect to a set of linear forms over a base set.

*Supported in part by NSF grant CCR: 9701304

†Partially supported by an Alfred P. Sloan Foundation Fellowship, and NSF CAREER award CCR: 9701304

1 Introduction

Many well known problems in algorithms and complexity reduce to the polynomial identity testing problem: given a multivariate polynomial $p(x_1, \ldots, x_n)$ over a field F, determine if the polynomial is identically zero. Algorithms for testing primality [1] or if a graph has a perfect matching [14], for example, involve testing if a particular polynomial is equal to zero. In addition, fundamental structural results in complexity theory such as IP=PSPACE [17] or the PCP theorem [4, 2] make heavy use of identity testing.

The complexity of the problem is highly representation dependent. If the input polynomial is specified by a list of monomials then the problem is trivial. On the other hand, there are many representations of polynomials such as arithmetic circuits or straight line programs which can succintly encode polynomials with exponentially many monomials. Still, almost all representations used in computer science have the property that the input polynomial can be quickly *evaluated* at certain points.

Schwartz [16] and Zippel [20] observed that by evaluating a polynomial at randomly chosen points from a sufficiently large domain, we can determine if the polynomial is nonzero with high probability. The correctness of their algorithm follows from the simple observation that any polynomial of total degree d cannot have many roots (relative to the field size) over a field whose size is much larger than d.

Along these lines, new tests have been discovered by Chen and Kao [7] and Lewin and Vadhan [12] which choose random evaluation points in a more sophisticated manner (either as irrational points in the former or square roots of irreducible polynomials in the latter). Further work by Agrawal and Biswas [1] departs from this evaluation paradigm and gives a new identity test via chinese remaindering polynomials. Their algorithm relies on the fact that any nonzero polynomial cannot be divisible by many low-degree relatively prime polynomials.

All of the above approaches were motivated by a longstanding problem from derandomization: how can we deterministically decide if a polynomial of total degree d and representation size s is identically zero in time polynomial in d and s? A solution to this problem would imply, among other things, an NC algorithm for matching and, if the solution runs in time polynomial in s and $\log(d)$, a polynomial time algorithm for primality [1]. The above algorithms make important contributions towards solving this problem as they show how to take advantage of a polynomial's structure and reduce the amount of randomness required by earlier Schwartz-Zippel schemes. Still, all of the above algorithms require $\Omega(n)$ random bits irrespective of the total degree of the input polynomial. For a polynomial in n variables with total degree δ, the best known lower bound on the number of random bits required to test if it is zero is $\Omega(\delta \log n)$ (when the polynomial is specified as a black box). More generally, for a polyomial with m monomials, one can prove a lower bound of $\Omega(\log m)$ on the number of random bits needed to perform an identity test. If the input polynomial is known to be sparse or have low total degree (which is frequently the case), the above algorithms use exponentially more randomness than known lower bounds indicate is necessary.

In this paper we show how to take advantage of the sparseness of our input polynomial. We develop a randomized polynomial time algorithm which uses only $O(\log(mn\delta))$ random bits where n is the number of variables, m is the number of monomials, and δ is the total degree. Thus, in any case where a polynomial has total degree less than n our algorithm achieves a significant savings in randomness. In any case where the total degree is polynomial in n, our algorithm uses $O(\log mn)$ random bits which is a significant savings over previous algorithms when m is subexponential and is essentially optimal (within an additive $O(\log n)$ term) when the polynomial is input as a black box. Our algorithms run in time polynomial in n, $\log(\delta)$, $\log(m)$ and $\log(\epsilon^{-1})$, where ϵ is an error parameter.

Our approach also yields the first polynomial time algorithm for interpolating sparse polynomials over finite fields of large characteristic, improving on a long line of research that includes the works of Ben-Or and Tiwari [5], Grigoriev, Karpinski, and Singer [10], and Karpinski and Shparlinski [11]. Further, we obtain a randomized polynomial time algorithm for retrieving a single coefficient of a monomial of an unknown polynomial regardless of its sparsity.

1.1 Our Approach

Our approach differs from other identity testing algorithms in that we give a randomized reduction from nonzero multivariate polynomials to nonzero univariate polynomials which uses very little randomness and incurs only a small blow up in the degree of the univariate polynomial. We can view monomials of a multivariate polynomial as linear forms over the input variables $x_1, \ldots, x_n$. For example, $x_1^3 x_2^2 x_3$ corresponds to the linear form $3y_1 + 2y_2 + y_3$. Then for a vector $\vec{r} = (r_1, \ldots, r_n)$ where each $r_i \in \{0, \ldots, p\}$ for some prime p, the substitution $x_i = y^{r_i}$ will produce a univariate polynomial whose terms are obtained by evaluating each monomial's linear form at the point $\vec{r}$. Viewed this way, we want a set of evaluation points which maps distinct monomials to distinct powers of y. Instead we will construct a small sample space of sets of evaluation points such that a randomly chosen set of points has the desired property with high probability. Our sample space is constructed via an error correcting code with good distance properties. To reduce the size of our evaluation points we will appeal to a proposition from the work on randomness optimal isolation due to [8]. In fact, we can use our error correcting code approach to extend the results in [8] and give a new isolation scheme which works with respect to set systems of linear forms over a base set.

Using this reduction, identity testing for n variable polynomials reduces to identity testing for low degree univariate polynomials which can be solved quickly and with little randomness. In addition, we can use our reduction to simplify the multivariate interpolation problem. Again, we reduce the multivariate interpolation problem to the univariate interpolation problem. Since our techniques are essentially independent of the characteristic of the underlying field, we obtain the first deterministic polynomial time algorithms for sparse interpolation over finite fields with large characteristic, thereby improving and simplyifing the results of Ben-Or and Tiwari [5], Karpinski and Shparlinski [11], and others.

1.2 Comparison with other results

The tests of Chen/Kao [7] and Lewin/Vadhan [12] use $\sum_{i=1}^{n} \lceil \log(d_i+1) \rceil$ random bits to test if a polynomial is zero where d_i is the degree of the ith variable in p. The Agrawal/Biswas test uses $\lceil \sum_{i=1}^{n} \log(d_i + 1) \rceil$ random bits. One nice property of these tests is that the number of random bits is independent of the error parameter (they use more time to obtain high accuracy). Still, given any bound on the total degree or number of monomials of a polynomial, we must assume d_i is at least 1 for all i. Thus each of the above tests use $\Omega(n)$ random bits.

We give a lengthy explanation of the relationship between our work and other work on deterministic sparse multivariate polynomial interpolaton in Section 6.

1.3 Organization

In Section 3 we show how to reduce a multivariate polynomial to a univariate polynomial which will have an unsatisfactory blow up in degree. In Section 4, we show how to reduce the blow up signficantly. From this we obtain our main result. In Section 5, we show how to implement the identity test using small bit lengths. In Section 6 we show how to obtain new interpolation algorithms.

2 Preliminaries

2.1 Notation

For any polynomial p, n will denote the number of variables in p. We use δ to denote the total degree of p. Note that the maximum degree over all individual variables x_i in p is bounded by δ. We let m denote the number of monomials of an unknown polynomial p. Recall that if a polynomial p in n variables has total degree δ then it has at most $O(n^\delta)$ monomials. For two vectors $\vec{d}$ and $\vec{a}$ we let $\left\langle \vec{d} | \vec{a} \right\rangle$ denote the inner product of $\vec{d}$ and $\vec{a}$. For an n element vector $\vec{d}$, we let $x^{\vec{d}}$ denote the monomial in variables $x_1, \dots, x_n$ where x_i is raised to the power d_i.

We will work with polynomials in the *black box* setting: our algorithms get only input/output behavior for any unknown polynomial we wish to test. If we say an algorithm or reduction works in polynomial time then it runs in time polynomial in $n, \log(\delta)$, $\log(m)$, and $\log(\epsilon^{-1})$, an error parameter.

2.2 Lower Bounds

One can show that any randomized scheme for identity testing in the black box model which succeeds with probability $1/2$ must use at least $(1 - o(1)) \log m$ random bits where m is the number of monomials of the unknown polynomial. We defer a proof and discussion of this to the Appendix (See Section A).

2.3 Generating a Prime Efficiently

We will need to generate primes of bit length m in polynomial time using only $O(m)$ random bits. This can be accomplished, and we defer the proof to the Appendix (See Section B).

2.4 Black Box Polynomials over Finite Fields

Given black box access to a polynomial over a finite field, one can debate whether or not it is safe to assume that one can ask for values of the polynomial at elements of an extension of that field. We believe this is a reasonable assumption. See Section C for a discussion.

3 Reduction to one variable

The first step of our zero-test will be to take a degree δ multivariate polynomial, such as

$$\sum_{j=1}^{m} c_j \vec{x}^{\vec{d}^{(j)}},$$

and map it into a low-degree non-zero univariate polynomial. A naive approach to this reduction would be to set $x_i = y^i$, for all i. This polynomial would have degree at most δn, but it could be zero (consider $x_1 x_2 - x_3$). On the other hand, one could guarantee that the polynomial will be non-zero by setting $x_i = y^{\delta 2^i}$; but, this yields unacceptably high degree.

We will construct a collection of vectors $\vec{a}^{(1)}, \ldots \vec{a}^{(t)}$ such that, for most k, the substitution $x_i = y^{\vec{a}_i^{(k)}}$ produces a non-zero polynomial. To guarantee that a $1-\epsilon$ fraction of the $\vec{a}^{(k)}$s yield non-zero polynomials, we will need $t = \lceil mn/\epsilon \rceil$ vectors with elements between 1 and $2mn/\epsilon$. Thus, the univariate polynomial we produce will have degree at most $2mn\delta/\epsilon$. In Section 4 we will reduce the degree to $(n\delta)^{O(1)}$

Under the substitution $x_i = y^{\vec{a}_i^{(k)}}$, $\vec{x}^{\vec{d}^{(j)}}$ maps to $y^{\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle}$. The idea behind our reduction is to arrange that each vector of the form

$$\left(\left\langle \vec{d}^{(j)} | \vec{a}^{(1)} \right\rangle, \left\langle \vec{d}^{(j)} | \vec{a}^{(2)} \right\rangle, \ldots, \left\langle \vec{d}^{(j)} | \vec{a}^{(t)} \right\rangle \right)$$

is roughly a codeword in a Reed-Solomon code. To this end, we define

$$\vec{a}_i^{(k)} = k^{i-1} \mod p,$$

where p is a prime slightly greater than t.

Lemma 1 *Let $\vec{d}^{(1)}, \ldots, \vec{d}^{(m)}$ be distinct vectors with entries in $\{0, 1, 2, \ldots, \delta\}$, and let p be a prime greater than t and δ. Then, for all j,*

$$\mathbf{Pr}_{1 \le k \le t} \left[\forall j' \neq j, \left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle \neq \left\langle \vec{d}^{(j')} | \vec{a}^{(k)} \right\rangle \right] \geq 1 - mn/t.$$

Proof First note that $\delta < p$ implies that the vectors $\vec{d}^{(1)}, \ldots, \vec{d}^{(m)}$ are all distinct modulo p. To each vector $\vec{d}^{(j)}$, we associate a polynomial p_j over $GF(p)$ given by

$$p_j(z) = \sum_{i=1}^{n} \vec{d}_i^{(j)} z^{i-1} \mod p.$$

By the construction of the vectors $\vec{a}^{(k)}$, we find

$$\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle \mod p = p_j(k).$$

As these polynomials have degree $n-1$, and two distinct degree $n-1$ polynomials can agree in at most n places, we find that for every j and any j', there are at most n values of k for which $p_j(k) = p_{j'}(k)$. As there are at most $m-1$ choices for j', we conclude that for every j, there are at most $(m-1)n$ values of k for which there exists a j' for which $p_j(k) = p_{j'}(k)$.

To conclude the proof, we note that if

$$\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle \mod p \neq \left\langle \vec{d}^{(j')} | \vec{a}^{(k)} \right\rangle \mod p,$$

then

$$\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle \neq \left\langle \vec{d}^{(j')} | \vec{a}^{(k)} \right\rangle.$$

■

Lemma 2 *Let*

$$P(\vec{x}) = \sum_{j=1}^{m} c_j \vec{x}^{\vec{d}^{(j)}},$$

be a non-zero polynomial of degree at most δ. Then, for k chosen at random between 1 and t, the probability that the univariate polynomial

$$P'(y) = P(y^{\vec{a}_1^{(k)}}, y^{\vec{a}_2^{(k)}}, \ldots y^{\vec{a}_n^{(k)}})$$

is zero is at most $1 - mn/t$. Moreover, the degree of $P'(y)$ is always at most $p\delta$, where p is the prime greater than t and δ used to define the vectors $\vec{a}^{(k)}$.

Proof Assume without loss of generality that $c_j \neq 0$. By Lemma 1, for at least $t - mn$ choices of k, the jth monomial is the only monomial that gets degree $\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle$ in y. That is, with probability $1 - mn/t$ over the choice of k between 1 and t, the coefficient of $y^{\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle}$ in $P'(y)$ is c_j, so $P'(y)$ is non-zero.

Finally, the total degree of P is at most δ and each entry of $\vec{a}^{(k)}$ is at most $p-1$; so, the total degree of $P'(y)$ is at most δp ■

4 Reducing the degree

From Lemma 2 we know there exists a randomized reduction from non-zero multivariate polynomials to non-zero univariate polynomials which succeeds with probability $1 - \epsilon$, uses $\log(mn/\epsilon)$ random bits and outputs a polynomial of degree at most $2mn\delta/\epsilon$. In this section we will show how to modify our reduction so that the output polynomial has considerably smaller degree while paying a small price in randomness. The idea behind our degree reduction is to view a multivariate polynomial as a set of linear forms, one for each monomial, over a base set of variables $\{x_1, \ldots, x_n\}$. In Section 3 we showed how to evaluate these linear forms so that with high probability one form evaluates to a unique value. In this section we show how to evaluate the forms so that with high probability *every* value obtained is unique and then apply a modified proposition from [8] to reduce the size of our evaluation points (which will return us to having only one unique value).

The first step of this process will be to apply the following straightforward modification of Lemma 1:

Lemma 3 *Let $\vec{d}^{(1)}, \ldots, \vec{d}^{(m)}$ be distinct vectors with entries in $\{0, 1, 2, \ldots, \delta\}$, and let p be a prime greater than t and δ. Then*

$$\mathbf{Pr}_{1 \leq k \leq t}\left[\left\langle \vec{d}^{(j)} | \vec{a}^{(k)} \right\rangle \text{ are dist. for } 1 \leq j \leq m \right] \geq 1 - m^2 n/t.$$

We will use a slight improvement of an isolation lemma from [8], which is based on a lemma from [14]. Our proof closely follows those of [8, 14].

Lemma 4 *Let C be any collection of distinct linear forms in variables $z_1, \ldots, z_\ell$ with coefficients in the range $\{0, \ldots, K\}$. If $z_1, \ldots, z_\ell$ are independently chosen uniformly at random in $\{0, \ldots, K\ell/\epsilon\}$, then, with probability greater than $1-\epsilon$, there is a unique form of minimal value at $z_1, \ldots, z_\ell$.*

Proof Following previous proofs of such statements, we call an index i *singular* under the assignment if there exist two forms in C that have different coefficients of z_i and that both achieve the minimum value. As all the forms are distinct, if more than one form achieves the minimal value then some index will be singular. To prove the lemma, we will show that for any given index, the probability that it is singular is at most ϵ/ℓ. Thus, the probability that there exists a singular index is at most ϵ.

In our proof that some index i is unlikely to be singular, we will only make use of the variable z_i. Accordingly, let all the other variables be set arbitrarily. Once we fix these variables, each form becomes a linear polynomial in z_i in which the constant term is determined by the setting of the other variables. We group these polynomials by their coefficient of z_i. As the coefficients are in the range $\{0, \ldots, K\}$, we get at most $K+1$ classes. In each class, it is clear that only the polynomial with the smallest constant term can achieve the minimal value. We let this polynomial be the representative of its class.

Index i is made singular by a setting of z_i if and only if the representatives of two different classes have the same value at z_i and this value is the minimum among the representatives. We now show that there are at most K possible settings of z_i that can cause this to happen. Thus, if z_i is chosen at random in $\{1, \ldots, K\ell/\epsilon\}$, the probability that i is singular is at most ϵ/ℓ.

To see why there are at most K possible settings of z_i that can cause two of the $K+1$ representative polynomials to both achieve the minimal value, consider the open polygon defined as the set of points in the plane below the $K+1$ lines which are the graphs of the representative polynomials (recall that these polynomials are linear). Each vertex of this polygon is a point at which two of the representative polynomials achieve the same minimal value (see Figure 1). There are at most K such points because an open polygon defined by $K+1$ ines can have at most K vertices. ■

The difference between our Lemma 4 and that proved in [8, 14] is that their proofs considered all points at which two of the lines intersected; so, they chose values for the

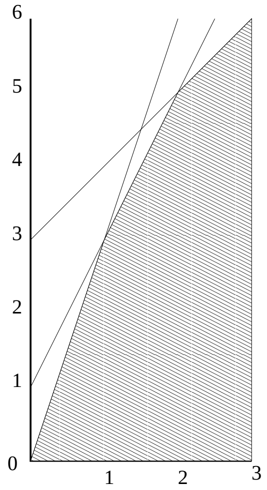

Figure 1: The polygon under the lines $x+3$, $2x+1$, and $3x$.

variables in the range $\left\{0, \ldots, K^2\ell/\epsilon\right\}$.

With this in hand we obtain the following:

Theorem 5 *There exists a randomized polynomial time algorithm which maps a nonzero multivariate polynomial p in n variables of total degree at most δ with m monomials to a nonzero univariate polynomial of degree $O(n^6\delta^4)$. The algorithm succeeds with probability $2/3$ and uses $O(\log mn)$ random bits.*

Proof Assume that we have randomly selected an $\vec{a}^{(k)}$ satisfying the conclusion of Lemma 3 (with, say $\epsilon = 1/6$). As in [8], write the ith element of $\vec{a}^{(k)}$ as a q bit number where $q = \min\{n, \log(6m^2n)\}$. Divide these bits into ℓ blocks each having q/ℓ bits. Let $b_{i,j}$ be the number in $\{0, \ldots, 2^{q/l}\}$ represented by the bits in block j. For a vector $\vec{d} \in \{\vec{d}^{(1)}, \ldots, \vec{d}^{(m)}\}$ let $w_i(\vec{d}) = \sum_{r=0}^{\ell-1} d_i b_{i,r} y_r$ (where the y_r's are new variables).

Lemma 6 *[8] The linear forms $w(\vec{d}) = \sum_{i=1}^{n} w_i(\vec{d})$ are distinct for all $\vec{d}$.*

Proof Each linear form $w(\vec{d})$ evaluated at $y_k = 2^{kq/l}$ for $0 \le k \le \ell - 1$ equals $\left\langle \vec{d} | \vec{a}^{(k)} \right\rangle$ which is a distinct value for every $\vec{d} \in \{\vec{d}^{(1)}, \ldots, \vec{d}^{(m)}\}$. ■

Now we define a vector $\vec{z}$. Let the ith component of vector $\vec{z}$ equal $\sum_{j=0}^{\ell-1} b_{i,j} y_j$ where y_j is chosen uniformly at random from $\{1, \ldots, (6\ell\delta n 2^{q/l})\}$. Then by Lemma 4 with probability $5/6$ there exists j such that $\left\langle \vec{d}^{(j)} | \vec{z} \right\rangle < \left\langle \vec{d}^{(j')} | \vec{z} \right\rangle$ for all $j' \neq j$. In other words, there exists a linear form $w(\vec{d})$ which, with respect to the evaluation points $y_1, \ldots, y_\ell$, takes on a unique minimum value.

Now consider

$$P'(h) = P(h^{\vec{z}_1}, h^{\vec{z}_2}, \ldots h^{\vec{z}_n})$$

The degree of each monomial in $P'(h)$ is equal to $\left\langle \vec{d}^{(j)} | \vec{z} \right\rangle$ for each monomial corresponding to $\vec{d}^{(j)}$. Thus, we obtain a univariate polynomial P' in h where precisely one monomial in our original polynomial has mapped to the smallest degree monomial in P'. Since our first transformation into distinct linear forms works with probability 5/6, both transformations succeed with probability 2/3. Setting $\ell = q/\log(6\delta n)$ tells us that our linear forms take on values bounded by $O(n^6\delta^4)$. This implies that our polynomial P' has degree at most $O(n^6\delta^4)$. The number of random bits required is $O(q) = O(\log mn)$. We defer the proof of these bounds to the appendix (see Section D.1). ■

The above theorem can be strengthened to work with probability $1 - \epsilon$. This will increase the degree of P' by a factor of ϵ^{-1} and require $O(\log(\epsilon^{-1}))$ additional random bits.

Remark: This approach can be used to generalize the randomness optimal isolation lemma proved in [8] which shows that for a set system $\mathcal{F}$ over a base set S there exists a random weighting of the elements of S such that with high probability there is a unique minimal weight set $F \in \mathcal{F}$. Their scheme uses only $O(\log(|\mathcal{F}|) + \log(|\mathcal{S}|))$ random bits and uses integer weights in the range $\{0, \ldots, |S|^{O(1)}\}$. Their construction works by first assigning weights to the base set and then invoking a theorem from number theory to conclude that with high probability every element of the set system $\mathcal{F}$ has a unique weight. We can replace this step by using the error correcting code approach from Section 3 which has the nice property that it maps sets of *linear forms* over subsets of variables from the base set to distinct values. Combining this idea with Lemma 4 we obtain the following generalized isolation lemma:

Theorem 7 *Let S be a set of variables and let $\mathcal{F}$ be a set of linear forms over subsets of variables from S with coefficients bounded by δ. Then there is a randomized scheme assigning integer weights to the elements of S such that with probability greater than $1 - \epsilon$ there is a unique minimum weight form in $\mathcal{F}$. The scheme uses $O(\log(|S|\delta) + \log(|\mathcal{F}|/\epsilon))$ random bits and assigns weights to the variables in the range $\{0, \ldots, (\epsilon^{-1}|S|\delta)^{O(1)}\}$.*

Proof Let $\mathcal{F} = \{\vec{d}^{(1)}, \ldots, \vec{d}^{(j)}\}$ and apply Lemma 3 to obtain a set of evaluation points such that every linear form evaluates to a distinct value. Then apply Lemma 4 to reduce the size of the evaluation points. The linear forms will no longer evaluate to distinct values, but there will exist an evaluated linear form of unique minimum value. Our bounds on randomness and the size of the integer weights follow as in the proof of Theorem 5. ■

5 Performing the test

At this point, we have obtained a collection of vectors $\vec{z}^{(1)}, \ldots, \vec{z}^{(t)}$, where each vector corresponds to an evaluation point and for every multilinear polynomial with at most m

monomials and degree δ, the linear polynomial obtained by the substitutions $x_i = y^{\vec{z}_i^{(k)}}$ is probably non-zero for a randomly chosen $\vec{z}^{(k)}$. If M is greater than any element of a vector $\vec{z}^{(k)}$, then this univariate polynomial has degree at most $M\delta$. Thus, if we choose y at random among a set of $M\delta/\epsilon$ elements, the chance that the polynomial is zero at this point is at most ϵ.

Of course, we don't feed the black-box a value for y—we feed it values for $x_1, \ldots, x_n$. If we are not working over a finite field, this can result in feeding the black box integers of unnecessarily long bit-length. If we choose y at random from the set $\{1, \ldots, M\delta/\epsilon\}$, then by applying $x_i = y^{\vec{z}_i^{(k)}}$ we obtain an upper bound on the x_i values of $(M\delta/\epsilon)^{M\delta}$. While the bit-length of this number is not prohibitively large, it could be much smaller. A natural idea for decreasing the bit-length would be to set

$$x_i = y^{\vec{z}_i^{(k)}} \mod p_2,$$

where p_2 is a prime greater than $(M\delta/\epsilon)$. The bit-length of these x_is is then $\log p_2$, which need be no larger than $\log(M\delta/\epsilon) + 1$. It is clear that this idea works over $GF(p_2)$, but it might not be immediately clear that it works over the integers. To show that it does, let

$$P(\vec{x}) = \sum_{j=1}^{m} c_j \vec{x}^{\vec{d}^{(j)}},$$

be a multivariate polynomial in which, without loss of generality, each $c_j \neq 0$, and let $\vec{z}^{(k)}$ be a vector for which there exists a j such that $\vec{x}^{\vec{d}^{(j)}}$ maps to a unique monomial in y under the mapping defined by $\vec{z}^{(k)}$. Consider the matrix M in which the entry in the jth row and lth column is the evaluation of the jth monomial at l modulo p_2:

$$\prod_{i=1}^{n} l^{\vec{z}_i^{(k)} \vec{d}_i^{(j)}} \mod p_2.$$

Since no non-zero degree $M\delta$ polynomial can have more than $M\delta$ roots, we can conclude that, for every non-zero m-vector v over $GF(p_2)$, the product vM will have at most $M\delta$ zero entries. As a sub-matrix of M is singular over $GF(p_2)$ if it is singular over the Reals, if we multiply M by a non-zero vector v over the Reals, the product vM will have at most $M\delta$ zero entries. Finally, if v is the vector of coefficients $(c_j)_j$, then the lth entry of vM is the result of evaluating $P(\vec{x})$ at the point given by

$$x_i = l^{\vec{b}_i^{(k)}} \mod p_2.$$

Combining this test with the reduction of the previous two sections we obtain:

Theorem 8 *Let p be a polynomial over the Reals with n variables, m monomials, and total degree δ. Our randomized algorithm will test if the polynomial is zero and succeed with probability $1-\epsilon$ using $O(\log(mn/\epsilon) + \log(n\delta/\epsilon))$ random bits and will use time polynomial in n, $\log(\delta)$, and $\log(\epsilon^{-1})$. Moreover, our algorithm queries the polynomial at points with bit length $O(\log(n\delta/\epsilon))$.*

For finite fields we need to assume the field has enough elements or that we can construct elements from a small extension field (see Section 2.4 for a discussion):

Theorem 9 *Let p be a polynomial with n variables, m monomials, and total degree δ over a field F. Further assume that F has at least $(n\delta/\epsilon)^6$ elements or that we can construct elements from an extension field of F having at least $(n\delta/\epsilon)^6$ elements. Then our randomized algorithm will test if the polynomial is zero and succeed with probability $1-\epsilon$ using $O(\log(mn/\epsilon)+\log(n\delta/\epsilon))$ random bits and will use time polynomial in n, $\log(\delta)$, and $\log(\epsilon^{-1})$. Moreover, our algorithm queries the polynomial at points with bit length $O(\log(n\delta/\epsilon))$.*

6 New Determinstic Algorithms for Sparse Identity Testing and Interpolation

There have been many papers written about deterministic interpolation and zero-testing of sparse multivariate polynomials over finite fields and the Reals. We begin by surveying the techniques used in these results and explain how they relate to our setting (some of these results can be used to obtain randomized identity tests similar to ours for the special case in which m is polynomial in n). We then explain how our techniques can be used to obtain new results for the problems considered in those papers. They key difference between our work and other results in this area is that our reduction from a multivariate polynomial to a univariate polynomial works irrespective of the underlying field. Therefore our results do not have a bad dependence on the field size or its characteristic.

6.1 Background

Work by Grigoriev and Karpinski [9] and Ben-Or and Tiwari [5] provides a technique for zero testing and interpolating sparse multivariate polynomials over the Reals. They evaluate the polynomial at points

$$(p_1^k, \dots p_n^k)$$

where k ranges between 1 and some constant times m, and $p_1, \dots p_n$ are distinct primes. Using these points, they construct a set of $O(m/\epsilon)$ points such that any polynomial with at most m monomials can be zero on at most an ϵ fraction of these points. We refer to such a set as a *test set* with error parameter ϵ. Our results immediately imply the existence of ϵ test sets of size $O((mn\delta/\epsilon)^{O(1)})$. While the test set of size $O(m/\epsilon)$ mentioned above is smaller than that which we construct, it has a significant disadvantage: the points in the set have bit-length $\Omega(m \log n)$. Thus, for super-polynomial m, a polynomial-time algorithm does not have time to output a random test vector from this set, so it is not useful in our setting. In contrast, the points in our test sets have bit-length $O(\log(n\delta/\epsilon))$. Alon and Mansour [3] produce test sets that have small bit-length, but the size of the test sets has a bad dependency on the size of the coefficients of the polynomial we wish to test.

A sequence of papers has extended these results to finite fields [11, 6, 19, 10]. These papers focus on polynomials defined over fixed finite fields with a corresponding fixed degree in each variable. To test whether these polynomials are zero, they work over these finite fields or their extensions. The test sets used in these papers are derived from a primitive element in this extension field. While one can efficiently find a polynomial size set that contains a primitive element of a field of small characteristic or over a field of prime or prime squared order assuming the extended Riemann hypothesis [18], in general it takes time polynomial in the *order* of a field to find a primitive element in that field or a small extension. At the moment, it is not known how to construct in polynomial time the test sets that these papers would use to zero-test polynomials over fields of characteristic superpolynomial in n. Assuming that we can construct the needed primitive element, we find that the technique of Grigoriev Karpinski and Singer [10] can be used to construct a test set for which any sparse polynomial will be non-zero at most points. The papers of Clausen et al. [6] and Werther [19] focus on how the size of a minimal test set varies with the degree of the extension field in which one works. Their results imply that if one can only evaluate in a small field, then it is not possible to construct test sets from which random points are good. The techniques of [10, 6, 19] all require time exponential in the ground field. Karpinski and Shparlinski [11] produce test sets in time polynomial in the characteristic of the ground field, which is an improvement for fields of small characteristic, but still takes superpolynomial time for fields with large characteristic.

6.2 Deterministic Sparse Zero Testing

For m and δ polynomial in n, our test sets have size polynomial in n. Thus, one can use our test sets to determinstically test whether a sparse multivariate polynomial is zero. The only restriction that our test places upon the ground field is that it have at least $(n\delta)^{O(1)}$ elements, or that we be able to evaluate the polynomial over an extension field with at least this many elements. By inspecting our results, one can see that the characteristic of the ground field has no impact on the efficacy of our test. On the other hand, previous techniques for deterministic zero testing run in time polynomial in the characteristic of the ground field. Thus, over fields of superpolynomial characteristic, we reduce the time required to zero-test multivariate polynomials with a polynomial number of monomials from exponential to polynomial time:

Theorem 10 *In time polynomial in n, m, δ, and $\log(|\mathcal{F}|)$, we can output a test set of vectors such that every non-zero degree δ n-variate polynomial over $\mathcal{F}$ with at most m monomials must be non-zero at one of the vectors in the set. If $\mathcal{F}$ is the Reals, then each element of each vector in the set has bit-length at most $O(\log(n\delta))$. If $\mathcal{F}$ is a finite field with less than $(n\delta)^6$ elements, then the elements of the vectors lie in the smallest extension of $\mathcal{F}$ with at least $(n\delta)^6$ elements; otherwise, the vectors contain just elements of $\mathcal{F}$.*

6.3 Deterministic Sparse Interpolation

Our techniques also provide a very simple solution to the interpolation problems considered in [5, 11, 6, 19, 10]. Consider a vector $\vec{a}^{()}$ produced by Lemma 3 under which the image of each monomial in $x_1, \ldots x_n$ maps to a distinct monomial in y. The univariate polynomial we obtain has degree at most $2m^2n\delta$. If we evaluate the polynomial at $2m^2n\delta$ elements of the base field, then we can interpolate the coefficients of the univariate polynomial using standard univariate interpolation. We thereby obtain the set of coefficients of the original multivariate polynomial; but, we don't know to which monomials these coefficients belong. That is, we have learned the c_js but not the $\vec{d}^{(j)}$s.

To solve this problem, consider a new map in which we set $x_i = y^{a_i}$ if $i \geq 2$ and $x_1 = 2y^{a_1}$. If we now evaluate this polynomial at the same $2m^d nd$ points, each coefficient will change according to the exponent of x_1 in its monomial. That is, we learn $\vec{d}_1^{(j)}$ for each j. By repeating this experiment for each variable, we learn which monomial corresponds to which coefficient.

Over the Reals, one can do this experiment in one step instead of n by setting $x_i = p_i y^{a_i}$, where the p_is are distinct primes. From this we obtain:

Theorem 11 *In time polynomial in n, m, δ, and $\log(|\mathcal{F}|)$, we can output a test set of vectors such that, given the values of a degree δ n-variate polynomial over $\mathcal{F}$ with at most m monomials at every vector in the set, we can solve for the coefficients of the polynomial in polynomial time. If $\mathcal{F}$ is the Reals, then each element of each vector in the set has bit-length at most $O(\log(n\delta))$. If $\mathcal{F}$ is a finite field with less than $(n\delta)^6$ elements, then the elements of the vectors lie in the smallest extension of $\mathcal{F}$ with at least $(n\delta)^6$ elements; otherwise, the vectors contain just elements of $\mathcal{F}$.*

To contrast this technique with that of [5], we note that ours requires interpolation at vectors of exponentially smaller bit-length. Again we note that our results work in time logarithmic in the field size, whereas previous techniques require time polynomial in the characteristic. Moreover, we feel that this interpolation algorithim is conceptually much simpler. We will compare the complexities of these algorithms in the final version of the paper.

6.4 Isolating a coefficient

Finally note that even in the case that the number of monomials is very large, our technique provides a probabilistic algorithm that allows one to learn a polynomial number of the monomial coefficients in polynomial time. The idea is to apply the interpolation algorithm of the last section to the univariate polynomial produced by our main reduction in Section 4. This allows us to learn the monomial corresponding to the lowest degree univariate monomial. We can then subtract off this monomial and recurse. We state this as follows:

Theorem 12 *Given black-box access to a n-variate polynomial p over $\mathcal{F}$ of degree d with at most m monomials, in randomized time polynomal in n, $\log(\delta)$, $\log(\mathcal{F})$, and $\log(m)$, we can output a set of test vectors such that we can deduce at least a polynomial number of coefficients of p from its values at the test vectors.*

7 Open Problems: The Case of Arithmetic Circuits

Our algorithms take advantage of the sparsity of a multivarite polynomial to obtain randomness efficient tests. Still, it is unclear how to use the particular input representation of a polynomial (for example as the determinant of a matrix) to achieve better derandomizations. Here we briefly consider the case when the polynomial is input as an arithmetic circuit. If the input is a depth-2 arithmetic circuit then the problem is trivial as it is a list of monomials. We outline a randomness efficient algorithm for testing if an arithmetic circuit of depth 3 with fan in 2 at the root is identically zero. As with the other results in this paper, this algorithm only uses the polynomial only as a black box. Again, inspection of the actual arithmetic circuit is enough to determine if it is identically zero.

Let P be a depth 3 arithmetic circuit with fan in 2 at the root. Then P is the difference of two polynomials p and q. Let $p = \prod_{j=1}^{m} \sum_{i=1}^{n} \alpha_{j,i} x_i$ and $q = \prod_{j=1}^{m} \sum_{i=1}^{n} \beta_{j,i} x_i$. Consider the substitution $x_i = c^i y + z^i$ where y and z are new variables and c is some element of the field. Now $p = \prod_{j=1}^{m}((\sum_{i=1}^{n} c^i \alpha_{j,i}) y + \sum_{i=1}^{n} \alpha_{j,i} z^i)$. For the moment let us assume that $\sum_{i=1}^{n} c^i \alpha_{j,i}$ is non-zero for every j. Notice that p is now a product of irreducible polynomials as each term in the product is linear in y and similarly for q. Because we are working over a unique factorization domain and the coefficients of the z terms are in one to one correspondence with the coefficients $\alpha_{j,i}$ and $\beta_{j,i}$, p is equal to q if and only if p equals q after making the above substitution. But now p and q are polynomials in only two variables, and we can test if they are identically zero using only $O(\log n)$ random bits. Furthermore, if we pick c randomly in the range $\{1, \ldots, O(mn)\}$ then with high probability $\sum_{i=1}^{n} c^i \alpha_{j,i}$ will be non-zero for all j. Hence our identity test uses only $O(\log mn)$ random bits.

If the top gate has fan in 3 or higher, we are unaware of any efficient black-box derandomization for identity testing. In fact, we challenge the reader to derandomize the identity testing of such arithmetic circuits even when given the circuit as input.

8 Acknowledgements

Thanks to Rocco Servedio, Mike Sipser, and Salil Vadhan for useful conversations. Thanks to V. Guruswami for comments on this write-up.

References

[1] Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. In *Proceedings of 40th FOCS*, pages 202–209, New York, NY, 1999.

[2] Arora, Lund, Motwani, Sudan, and Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[3] Noga Alon and Yishay Mansour. ϵ-discrepancy sets and their application for interpolation of sparse polynomials. *Information Processing Letters*, 54(6):337–342, 23 June 1995.

[4] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998.

[5] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In Richard Cole, editor, *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing*, pages 301–309, Chicago, IL, May 1988. ACM Press.

[6] M. Clausen, A. Dress, J. Grabmeier, and M. Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151–164, July 1991.

[7] Zhi-Zhong Chen and Ming-Yang Kao. Reducing randomness via irrational numbers. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 200–209, El Paso, Texas, 4–6 May 1997.

[8] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM Journal on Computing*, 24(5):1036–1050, October 1995.

[9] D. Y. Grigoriev and M. Karpinski. The matching problem for bipartite graphs with polynomially bounded permanents is in NC. In Ashok K. Chandra, editor, *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 166–172, Los Angeles, CA, October 1987. IEEE Computer Society Press.

[10] Dima Grigoriev, Marek Karpinski, and Michael F. Singer. Computational complexity of sparse rational interpolation. *SIAM Journal on Computing*, 23(1):1–11, February 1994.

[11] Marek Karpinski and Igor Shparlinski. On some approximation problems concerning sparse polynomials over finite fields. *Theoretical Computer Science*, 157(2):259–266, 5 May 1996.

[12] Daniel Lewin and Salil Vadhan. Checking polynomial identities over any field: Towards a derandomization. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)*, pages 438–447, New York, May 23–26 1998. ACM Press.

[13] G. L. Miller. Reimann's hypothesis and tests for primality. *J. Comput. System Sci*, 13:300–317, 1976.

[14] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

[15] M. O. Rabin. A probabilistic algorithm for testing primality. *Journal of Number Theory*, 12, 1980.

[16] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[17] A. Shamir. IP=PSPACE. *Journal of the ACM*, 39:869–877, 1992.

[18] Victor Shoup. Searching for primitive roots in finite fields. *Mathematics of Computation*, 58(197):369–380, January 1992.

[19] Kai Werther. The complexity of sparse polynomial interpolation over finite fields. *Applicable Algebra in Engineering, Communication, and Computing*, 5:91–103, 1994.

[20] R. Zippel. Probabilistic algorithms for sparse polynomials. In *ISSAC '79: Proc. Int'l. Symp. on Symbolic and Algebraic Computation,* Lecture Notes in Computer Science, Vol. 72. Springer-Verlag, 1979.

A Lower Bounds

In [12], Lewin and Vadhan give a lower bound on the number of random bits needed to perform an identity test in the black box model. They prove that any randomized polynomial time algorithm for identity testing which succeeds with probability 1/2 must use at least $(1-o(1))\sum_{i=1}^{n}\log(d_i+1)$ random bits where d_i is the degree of the ith variable. The idea behind the lower bound is that any *deterministic* scheme must make at least $\Pi_{i=1}^{n}(d_i+1)$ queries to the black box. Otherwise, there exists a non-zero polynomial with degree at most d_i in each variable which is zero at each query point via interpolation. Thus, any *randomized* scheme must use essentially $\log(\Pi_{i=1}^{n}(d_i+1))$ random bits. Their argument can be used to show that any randomized scheme for identity testing which succeeds with probability 1/2 must use at least $(1-o(1))\log m$ random bits where m is the number of monomials of the unknown polynomial.

B Generating a Prime Efficiently

To generate an m bit prime using $O(m)$ random bits, we first randomly generating m^2 pairwise independent numbers (each of bit length m) and then check if any number is prime using a randomized primality test such as the Miller-Rabin test [13] [15]. Generating the pairwise independent samples will require $O(m)$ random bits. We can decrease the error probability of the Miller-Rabin test to 2^{-m} using $O(m)$ random bits via a random walk on an expander. Since a $1/m$ fraction of m bit strings are prime by the Prime Number Theorem, we can apply the Chebyshev inequality and conclude that with high probability one of our m^2 samples is prime. In addition, the probability that Miller-Rabin test misclassifies one of our samples is at most $m^2 2^{-m}$. Thus, with high probability we can generate a prime of bit length m using $O(m)$ random bits.

C Black Box Polynomials over Finite Fields

For all of the realizations of black box polynomials of which we are aware, this assumption is reasonable if the algorithm calling the black box can construct the extension field. In our results, we merely need to be able to query the polynomial at $(n\delta)^{O(1)}$ different elements of a field containing the base field. If the base field does not have this many elements, then one can easily construct an extension of that base field that does. On the other hand, if one does not allow evaluation at extensions of the base field, then there are polynomials that are unlikely to be zero at random points in any test set [6].

D Deferred Proofs

D.1 Bounds on Randomness and Degree for Theorem 5

- Each $b_{ij} \in \{1, \ldots, 2^{q/l}\}$,
- Each linear form $w_i(\vec{d}) = \sum_{r=0}^{\ell-1} \vec{d}_i b_{i,r} y_r$ has coefficients in the range $\{1, \ldots, \delta 2^{q/l}\}$.
- Each linear form $w(\vec{d}) = \sum_{i=1}^{n} w_i(\vec{d})$ has coefficients in the range $\{1, \ldots, n\delta 2^{q/l}\}$.
- Each linear form $w(\vec{d})$ has ℓ variables. As dictated by Lemma 4, we choose a random setting for each variable in the range $\{1, \ldots, 6\ell\delta n 2^{q/l}\}$.
- Each linear form evaluates to a value in the range $\ell(6\ell\delta n 2^{q/l})(\delta n 2^{q/l}) \leq 6\ell^2\delta^2 n^2 2^{2q/l}$.
- The randomness required to choose ℓ evaluations points is $\ell(\log(6\ell\delta n) + q/\ell) \leq 2\ell \log(6\delta n) + q$ (recall $\ell \leq n$).

Setting $\ell = q/\log(6\delta n)$ tells us that our linear forms take on values bounded by $O(n^6\delta^4)$. This implies that our polynomial P' has degree at most $O(n^6\delta^4)$. The number of random bits required is $O(q) = O(\log mn)$.