I do not pretend to start with precise questions. I do not think you can start with anything precise. You have to achieve such precision as you can, as you go along.

—Bertrand Russell, *The Philosophy of Logical Atomism* (1918)

Chapter 7

Sparse interpolation

The next two chapters examine methods to determine the sparse representation of an unknown polynomial, given only a way to evaluate the polynomial at chosen points. These methods are generally termed black-box sparse interpolation.

First, we examine a new approach to the standard sparse interpolation problem over large finite fields and approximate complex numbers. This builds on recent work by Garg and Schost (2009), improving the complexity of the fastest existing algorithm over large finite fields, and the numerical stability of the state of the art for approximate sparse interpolation. The primary new tool that we introduce is a randomised method to distinguish terms in the unknown polynomial based on their coefficients, which we term diversification. Using this technique, our new algorithms gain practical and theoretical efficiency over previous methods by avoiding the need to use factorization algorithms as a subroutine.

We gratefully acknowledge the helpful and useful comments by Éric Schost as well as the Symbolic Computation Group members at the University of Waterloo on preliminary versions of the work in this chapter.

7.1 Background

Polynomial interpolation is a long-studied and important problem in computer algebra and symbolic computation. Given a way to evaluate an unknown polynomial at any chosen point, and an upper bound on the degree, the interpolation problem is to determine a representation for the polynomial. In *sparse* interpolation, we are also given an upper bound on the number of nonzero terms in the unknown polynomial, and the output is returned in the sparse representation. Generally speaking, we seek algorithms whose cost is polynomially-bounded by the size of the output, i.e., the sparse representation size of the unknown polynomial.

Sparse interpolation has numerous applications in computer algebra and engineering. Numerous mathematical computations suffer from the problem of *intermediate expression swell*, whereby the size of polynomials encountered in the middle of a computation is much larger than the size of the input or the output. In such cases where the output is known to be sparse, sparse interpolation can eliminate the need to store large intermediate expressions explicitly, thus greatly reducing not only the intermediate storage requirement but also the computational cost. Important examples of such problems are multivariate polynomial factorization and system solving (Canny, Kaltofen, and Yagati, 1989; Kaltofen and Trager, 1990; Díaz and Kaltofen, 1995, 1998; Javadi and Monagan, 2007, 2009).

Solving non-linear systems of multivariate polynomials with approximate coefficients has numerous practical applications, especially in engineering (see for example Sommese and Wampler, 2005). Homotopy methods are a popular way of solving such systems and related problems by following the paths of single solution points from an initial, easy system, to the target system of interest. Once enough points are known in the target system, sparse interpolation methods are used to recover a polynomial expression for the actual solution. These techniques generally fall under the category of hybrid symbolic-numeric computing, and in particular have been applied to solving non-linear systems (e.g., Sommese, Verschelde, and Wampler, 2001, 2004; Stetter, 2004) and factoring approximate multivariate polynomials (e.g., Kaltofen, May, Yang, and Zhi, 2008).

Sparse interpolation is also of interest in theoretical computer science. It is a non-trivial generalisation of the important problem of *polynomial identity testing*, or PIT for short. Generally speaking, the PIT problem is to identify whether the polynomial represented by a given algebraic circuit is identically zero. Surprisingly, although this question is easily answered using randomisation and the classical results of Demillo and Lipton (1978); Zippel (1979); Schwartz (1980), no *deterministic* polynomial-time algorithm is known. In fact, even derandomising the problem for circuits of depth four would have important consequences (Kabanets and Impagliazzo, 2004). We will not discuss this problem further, but the point the reader to the excellent recent surveys of Saxena (2009) and Shpilka and Yehudayoff (2010).

7.1.1 Problem definition

Let $f \in \mathsf{F}[x_1,\ldots,x_n]$ have degree less than d. A *black box* for f is a function which takes as input a vector $(a_1,\ldots,a_n) \in \mathsf{F}^n$ and produces $f(a_1,\ldots,a_n) \in \mathsf{F}$. The cost of the black box is the number of operations in F required to evaluate it at a given input.

Clausen, Dress, Grabmeier, and Karpinski (1991) showed that, if only evaluations over the ground field F are allowed, then for some instances at least $\Omega(n^{\log t})$ black box probes are required. Hence if we seek polynomial-time algorithms, we must extend the capabilities of the black box. To this end, Díaz and Kaltofen (1998) introduced the idea of an *extended domain black box* which is capable of evaluating $f(b_1,\ldots,b_n) \in \mathsf{E}$ for any $(b_1,\ldots,b_n) \in \mathsf{E}^n$ where E is any extension field of F. That is, we can change every operation in the black box to work over an extension field, usually paying an extra cost per evaluation proportional to the size of the extension.

Motivated by the case of black boxes that are division-free algebraic circuits, we will use the following model which we believe to be fair and cover all previous relevant results. Again we use the notation of $\mathsf{M}(m)$ for the number of field operations required to multiply two dense univariate polynomials with degrees less than m, and $\tilde{O}(m)$ to represent any function bounded by $m(\log m)^{O(1)}$.

Definition 7.1. *Let $f \in \mathsf{F}[x_1,\ldots,x_n]$ and $\ell > 0$. A* remainder black box *for f with size ℓ is a procedure which, given any monic square-free polynomial $g \in \mathsf{F}[y]$ with $\deg g = m$, and any $h_1,\ldots,h_n \in \mathsf{F}[y]$ with each $\deg h_i < m$, produces $f(h_1,\ldots,h_n) \operatorname{rem} g$ using at most $\ell \cdot \mathsf{M}(m)$ operations in F.*

This definition is general enough to cover the algorithms we know of over finite fields, and we submit that the cost model is fair to the standard black box, extended domain black box, and algebraic circuit settings. The model also makes sense over complex numbers, as we will see.

7.1.2 Interpolation over finite fields

We first summarize previously known univariate interpolation algorithms when F is a finite field with q elements and identify our new contributions here. For now, let $f \in \mathbb{F}_q[x]$ have degree less than d and sparsity t. We will assume we have a remainder black box for f with size ℓ. Since field elements can be represented with $O(\log q)$ bits, a polynomial-time algorithm will have cost polynomial in ℓ, t, $\log d$, and $\log q$.

For the dense output representation, one can use the classical method of Newton, Waring, and Lagrange to interpolate in $\tilde{O}(\ell d)$ time (von zur Gathen and Gerhard, 2003, §10.2).

The algorithm of Ben-Or and Tiwari (1988) for sparse polynomial interpolation, and in particular the version developed by Kaltofen and Yagati (1989), can be adapted to arbitrary finite fields. Unfortunately, these algorithms require t discrete logarithm computations in $\mathbb{F}_q^*$, whose cost is small if the field size q is chosen carefully (as in Kaltofen (2010)), but not in general. For arbitrary (and potentially large) q, we can take advantage of the fact that each discrete logarithm that needs to be computed falls in the range $[0,1,\ldots,d-1]$. The "kangaroo method" of Pollard (1978, 2000) can, with high probability, compute such a discrete log with $O(\sqrt{d})$ field operations. Using this algorithm makes brings the total worst-case cost of Ben-Or and Tiwari's algorithm to $O(t\ell + t^2 + t\sqrt{d})$.

This chapter builds most directly on the work of (Garg and Schost, 2009), who gave the first polynomial-time algorithm for sparse interpolation over an arbitrary finite field. Their algorithm works roughly as follows. For very small primes p, use the black box to compute f modulo $x^p - 1$. A prime p is a "good prime" if and only if all the terms of f are still distinct modulo $x^p - 1$. If we do this for all p in the range of roughly $O(t^2 \log d)$, then there will be sufficient good primes to recover the unique symmetric polynomial over $\mathbb{Z}[y]$ whose roots are the exponents of nonzero terms in f. We then factor this polynomial to find those exponents, and correlate with any good prime image to determine the coefficients. The total cost is $\tilde{O}(\ell t^4 \log^2 d)$ field operations. Using randomization, it is easy to reduce this to $\tilde{O}(\ell t^3 \log^2 d)$.

	Probes	Probe degree	Computation cost	Total cost
Dense	d	1	$\tilde{O}(d)$	$\tilde{O}(\ell d)$
Ben-Or & Tiwari	$O(t)$	1	$O(t^2 + t\sqrt{d})$	$\tilde{O}(\ell t + t^2 + t\sqrt{d})$
Garg & Schost	$\tilde{O}(t^2 \log d)$	$\tilde{O}(t^2 \log d)$	$\tilde{O}(t^4 \log^2 d)$	$\tilde{O}(\ell t^4 \log^2 d)$
Randomized G & S	$\tilde{O}(t \log d)$	$\tilde{O}(t^2 \log d)$	$\tilde{O}(t^3 \log^2 d)$	$\tilde{O}(\ell t^3 \log^2 d)$
Ours	$O(\log d)$	$\tilde{O}(t^2 \log d)$	$\tilde{O}(t^2 \log^2 d)$	$\tilde{O}(\ell t^2 \log^2 d)$

Table 7.1: Sparse univariate interpolation over large finite fields, with black box size ℓ, degree d, and t nonzero terms

Observe that the coefficients of the symmetric integer polynomial in Garg & Schost's algorithm are bounded by $O(d^t)$, which is much larger than the $O(d)$ size of the exponents ultimately recovered. Our primary contribution over finite fields of size at least $\Omega(t^2 d)$ is a new algorithm which avoids evaluating the symmetric polynomial and performing root finding over $\mathbb{Z}[y]$. As a result, we reduce the total number of required evaluations and develop a randomized algorithm with cost $\tilde{O}(\ell t^2 \log^2 d)$, which is roughly quadratic in the input and output sizes. Since this can be deterministically verified in the same time, our algorithm (as well as the randomized version of Garg & Schost) is of the Las Vegas type.

The relevant previous results mentioned above are summarized in Table 7.1, where we assume in all cases that the field size q is "large enough". In the table, the "probe degree" refers to the degree of g in each evaluation of the remainder black box as defined above.

7.1.3 Multivariate interpolation

Any of the univariate algorithms above can be used to generate a multivariate polynomial interpolation algorithm in at least two different ways. For what follows, write $\rho(d,t)$ for the number of remainder black box evaluations required by some univariate interpolation algorithm, $\Delta(d,t)$ for the degree of the remainder in each evaluation, and $\psi(d,t)$ for the number of other field operations required besides black box calls. Observe that these correspond to the first three columns in Table 7.1.

The first way to adapt a univariate interpolation algorithm to a multivariate one is Kronecker substitution: given a remainder black box for an unknown $f \in \mathsf{F}[x_1,\ldots,x_n]$, with each partial degree less than d, we can easily construct a remainder black box for the univariate polynomial $\hat{f} = f(x, x^d, x^{d^2}, \ldots, x^{d^{n-1}}) \in \mathsf{F}[x]$, whose terms correspond one-to-one with terms of f. This is the approach taken for instance in Kaltofen (2010, §2) for the interpolation of multivariate polynomials with rational coefficients. The cost is simply the cost of the chosen underlying univariate algorithm, with the degree increased to d^n.

The other method for constructing a multivariate interpolation algorithm is due to Zippel (1990). The technique is inherently probabilistic and works variable-by-variable, at each step solving a number of $\hat{t} \times \hat{t}$ transposed Vandermonde systems, for some $\hat{t} \le t$. Specifically, each system is of the form $Ax = b$, where A is a $\hat{t} \times \hat{t}$ matrix of scalars from the coefficient field $\mathbb{F}_q$. The vector v consists of the output of $\hat{t}$ remainder black box evaluations, and so its

	Kronecker	Zippel
Dense	$\tilde{O}(\ell d^n)$	$\tilde{O}(\ell n t d)$
Ben-Or & Tiwari	$\tilde{O}(\ell t + t^2 + t d^{n/2})$	$\tilde{O}(n t^3 + n t^2 \sqrt{d} + \ell n t^2)$
Garg & Schost	$\tilde{O}(\ell n^2 t^4 \log^2 d)$	$\tilde{O}(\ell n t^5 \log^2 d)$
Randomized G & S	$\tilde{O}(\ell n^2 t^3 \log^2 d)$	$\tilde{O}(\ell n t^4 \log^2 d)$
Ours	$\tilde{O}(\ell n^2 t^2 \log^2 d)$	$\tilde{O}(\ell n t^3 \log^2 d)$

Table 7.2: Sparse multivariate interpolation over large finite fields, with black box size ℓ, n variables, degree d, and t nonzero terms

elements are in $\mathbb{F}_q[y]$, and the system must be solved modulo some $g \in \mathbb{F}_q[y]$, as specified by the underlying univariate algorithm. Observe however that since A does not contain polynomials, computing $x = A^{-1}b$ requires no modular polynomial arithmetic. In fact, using the same techniques as Kaltofen and Yagati (1989, §5), employing fast dense bivariate polynomial arithmetic, each system can be solved using

$$O\Big(\mathsf{M}\big(t \cdot \Delta(d,t)\big) \cdot \log\big(t \cdot \Delta(d,t)\big)\Big)$$

field operations.

Each transposed Vandermonde system gives the remainder black box evaluation of each of $\hat{t}$ univariate polynomials that we are interpolating in that step. The number of such systems that must be solved is therefore $\rho(d,t)$, as determined by the underlying univariate algorithm. Finally, each of the $\hat{t}$ univariate interpolations proceeds with the given evaluations. The total cost, over all iterations, is

$$\tilde{O}\big(\ell n t \cdot \Delta(d,t) \cdot \rho(d,t)\big)$$

field operations for the remainder black box evaluations, plus

$$\tilde{O}\big(n t \psi(d,t) + \ell n t \cdot \Delta(d,t)\big)$$

field operations for additional computation. Zippel (1990) used the dense algorithm for univariate interpolation; using Ben-Or and Tiwari's algorithm instead was studied by Kaltofen and Lee (2003).

Table 7.2 summarizes the cost of the univariate algorithms mentioned above applied to sparse multivariate interpolation over a sufficiently large finite field, using Kronecker's and Zippel's methods.

For completeness, we mention a few more results on closely related problems that do not have a direct bearing on the current study. Grigoriev, Karpinski, and Singer (1990) give a parallel algorithm with small depth but which is not competitive in our model due to the large number of processors required. A practical parallel version of Ben-Or and Tiwari's algorithm has been developed by (Javadi and Monagan, 2010). (Kaltofen, Lakshman, and Wiley, 1990) and (Avendaño, Krick, and Pacetti, 2006) present modular algorithms for interpolating polynomials with rational and integer coefficients. However, their methods do not seem to apply to finite fields.

7.1.4 Approximate Polynomial Interpolation

In Section 7.4 we consider the case of approximate sparse interpolation. Our goal is to provide both a numerically more robust practical algorithm, but also the first algorithm which is provably numerically stable, with no heuristics or conjectures. We define an "ϵ-approximate black box" as one which evaluates an unknown t-sparse target polynomial $f \in \mathbb{C}[x]$, of degree d, with relative error at most $\epsilon > 0$. Our goal is to build a t-sparse polynomial g such that $\|f - g\| \leq \epsilon \|f\|$. A bound on the degree and sparsity of the target polynomial, as well as ϵ, must also be provided. In Section 7.4 we formally define the above problem, and demonstrate that the problem of sparse interpolation is well-posed. We then adapt our variant of the (Garg and Schost, 2009) algorithm for the approximate case, prove it is numerically accurate in terms of the relative error of the output, and analyze its cost. We also present a full implementation in Section 7.5 and validating experiments.

Recently, a number of numerically-focussed sparse interpolation algorithms have been presented. The algorithm of (Giesbrecht, Labahn, and Lee, 2009) is a numerical adaptation of (Ben-Or and Tiwari, 1988), which samples f at $O(t)$ randomly chosen roots of unity $\omega \in \mathbb{C}$ on the unit circle. In particular, ω is chosen to have (high) order at least the degree, and a randomization scheme is used to avoid clustering of nodes which will cause dramatic ill-conditioning. A relatively weak theoretical bound is proven there on the randomized conditioning scheme, though experimental and heuristic evidence suggests it is much better in practice. (Cuyt and Lee, 2008) adapt Rutishauser's qd algorithm to alleviate the need for bounds on the partial degrees and the sparsity, but still evaluate at high-order roots of unity. Approximate sparse rational function interpolation is considered by (Kaltofen and Yang, 2007) and (Kaltofen, Yang, and Zhi, 2007), using the Structured Total Least Norm (STLN) method and, in the latter, randomization to improve conditioning. Approximate sparse interpolation is also considered for integer polynomials by (Mansour, 1995), where a polynomial-time algorithm is presented in quite a different model from ours. In particular the evaluation error is absolute (not relative) and the complexity is sensitive to the bit length of the integer coefficients.

Note that all these works evaluate the polynomial only on the unit circle. This is necessary because we allow and expect f to have very large degree, which would cause a catastrophic loss of precision at data points of non-unit magnitude. Similarly, we assume that the complex argument of evaluation points is exactly specified, which is again necessary because any error in the argument would be exponentially magnified by the degree.

The primary contribution of our work in Section 7.4 below is to provide an algorithm with both rigorously provable relative error and good practical performance. Our algorithm typically requires $\tilde{O}(t^2 \log^2 d)$ evaluations at primitive roots of unity of order $\tilde{O}(t^2 \log d)$ (as opposed to order d in previous approaches). We guarantee that it finds a t-sparse polynomial g such that $\|g - f\| \leq 2\epsilon \|f\|$. An experimental demonstration of the numerical robustness is given in Section 7.5.

7.2 Sparse interpolation for generic fields

For the remainder, we assume the unknown polynomial f is always univariate. This is without loss of generality, as we can use the Kronecker substitution as discussed previously. The exponential increase in the univariate degree only corresponds to a factor of n increase in $\log\deg f$, and since our algorithms will ultimately have cost polynomial in $\log\deg f$, polynomial time is preserved.

Assume a fixed, unknown, t-sparse univariate polynomial $f \in \mathsf{F}[x]$ with degree at most d. We will use a remainder black box for f to evaluate $f \operatorname{rem}(x^p - 1)$ for small primes p. We say p is a "good prime" if the sparsity of $f \operatorname{rem}(x^p - 1)$ is the same as that of f itself — that is, none of the exponents are equivalent modulo p.

The minimal bit length required so that a randomly chosen prime is with high probability a good prime is shown in the following lemma.

Lemma 7.2. *Let $f \in \mathsf{F}[x]$ be a t-sparse polynomial with degree d, and let*

$$\lambda = \max\left(21, \left\lceil \frac{5}{3} t(t-1)\ln d \right\rceil\right).$$

A prime chosen at random in the range $\lambda, \ldots, 2\lambda$ is a good prime for f with probability at least $1/2$.

Proof. Write $e_1, \ldots, e_t$ for the exponents of nonzero terms in f. If p is a bad prime, then p divides $(e_j - e_i)$ for some $i < j$. Each $e_j - e_i \le d$, so there can be at most $\log_\lambda d = \ln d / \ln \lambda$ primes that divide each $e_j - e_i$. There are exactly $\binom{t}{2}$ such pairs of exponents, so the total number of bad primes is at most $(t(t-1)\ln d)/(2\ln\lambda)$.

From Rosser and Schoenfeld (1962, Corollary 3 to Theorem 2), the total number of primes in the range $\lambda, \ldots, 2\lambda$ is at least $3\lambda/(5\ln\lambda)$ when $\lambda \ge 21$, which is at least $t(t-1)\ln d/\ln\lambda$, at least twice the number of bad primes. □

Now observe an easy case for the sparse interpolation problem. If a polynomial $f \in \mathsf{F}[x]$, has all coefficients distinct; that is, $f = \sum_{1 \le i \le t} c_i x^{e_i}$ and $c_i = c_j \Rightarrow i = j$, then we say f is *diverse*. To interpolate a diverse polynomial $f \in \mathsf{F}[x]$, we first follow the method of Garg and Schost (2009) by computing $f \operatorname{rem}(x^{p_i} - 1)$ for "good primes" p_i such that the sparsity of $f \operatorname{rem}(x^{p_i} - 1)$ is the same as that of f. Since f is diverse, $f \operatorname{rem}(x^{p_i} - 1)$ is also diverse and in fact each modular image has the same set of coefficients. Using this fact, we avoid the need to construct and subsequently factor the symmetric polynomial in the exponents. Instead, we correlate like terms based on the (unique) coefficients in each modular image, then use simple Chinese remaindering to construct each exponent e_i from its image modulo each p_i. This requires only $O(\log d)$ remainder black box evaluations at good primes, gaining a factor of t improvement over the randomized version of Garg and Schost (2009) for diverse polynomials.

In the following sections, we will show how to choose an $\alpha \in \mathsf{F}$ so that $f(\alpha x)$ — which we can easily construct a remainder black box for — is diverse. With such a procedure, Algorithm 7.1 gives a Monte Carlo algorithm for interpolation over a general field.

Algorithm 7.1: Generic interpolation

```
Input: $\mu \in \mathbb{R}_{>0}$, $T, D, q \in \mathbb{N}$, and a remainder black box for unknown $T$-sparse $f \in \mathsf{F}[x]$
        with $\deg f < D$
Output: $t \in \mathbb{N}$, $e_1, \ldots, e_t \in \mathbb{N}$, and $c_1, \ldots, c_t \in \mathsf{F}$ such that $f = \sum_{1 \le i \le t} c_i x^{e_i}$
1  $t \leftarrow 0$
2  $\lambda \leftarrow \max\left(21, \left\lceil \frac{5}{3} T(T-1) \ln D \right\rceil\right)$
3  for $\lceil \log_2(3/\mu) \rceil$ primes $p \in \{\lambda, \ldots, 2\lambda\}$ do
4      Use black box to compute $f_p = f(x) \operatorname{rem} (x^p - 1)$
5      if $f_p$ has more than $t$ terms then
6          $t \leftarrow$ sparsity of $f_p$
7          $\varrho \leftarrow p$
8  $\alpha \leftarrow$ element of $\mathsf{F}$ such that $\Pr[f(\alpha x)$ is not diverse$] < \mu/3$
9  $g_\varrho \leftarrow f(\alpha x) \operatorname{rem} (x^\varrho - 1)$
10 $c_1, \ldots, c_t \leftarrow$ nonzero coefficients of $g_\varrho$
11 $e_1, \ldots, e_t \leftarrow 0$
12 for $\lceil 2 \ln(3/\mu) + 4(\ln D)/(\ln \lambda) \rceil$ primes $p \in \{\lambda, \ldots, 2\lambda\}$ do
13     Use black box to compute $g_p = f(\alpha x) \operatorname{rem} (x^p - 1)$
14     if $g_p$ has exactly $t$ nonzero terms then
15         for $i = 1, \ldots, t$ do Update $e_i$ with exponent of $c_i$ in $g_p$ modulo $p$ via Chinese
           remaindering
16 for $i = 1, \ldots, t$ do $c_i \leftarrow c_i \alpha^{-e_i}$
17 return $f(x) = \sum_{1 \le i \le t} c_i x^{e_i}$
```

Theorem 7.3. *With inputs as specified, Algorithm 7.1 correctly computes the unknown polynomial f with probability at least $1-\mu$. The total cost in field operations (except for step 8) is*

$$O\left(\ell\cdot\left(\frac{\log D}{\log T+\log\log D}+\log\frac{1}{\mu}\right)\cdot \mathsf{M}\left(T^2\log D\right)\right).$$

Proof. The for loop on line 3 searches for the true sparsity t and a single good prime ϱ. Since each prime p in the given range is good with probability at least $1/2$ by Lemma 7.2, the probability of failure at this stage is at most $\mu/3$.

The for loop on line 12 searches for and uses sufficiently many good primes to recover the exponents of f. The product of all the good primes must be at least D, and since each prime is at least λ, at least $(\ln D)/(\ln\lambda)$ good primes are required.

Let $n=\lceil 2\ln(3/\mu)+4(\ln D)/(\ln\lambda)\rceil$ be the number of primes sampled in this loop, and $k=\lceil(\ln D)/(\ln\lambda)\rceil$ the number of good primes required. We can derive that $(n/2-k)^2\geq(\ln(3/\mu)+k)^2>(n/2)\ln(3/\mu)$, and therefore $\exp(-2(\frac{n}{2}-k)^2/n)<\mu/3$. Using Hoeffding's Inequality (Hoeffding, 1963), this means the probability of encountering fewer than k good primes is less than $\mu/3$.

Therefore the total probability of failure is at most μ. For the cost analysis, the dominating cost will be the modular black box evaluations in the last for loop. The number of evaluations in this loop is $O(\log(1/\mu)+(\log D)/(\log\lambda))$, and each evaluation has cost $O(\ell\cdot\mathsf{M}(\lambda))$. Since the size of each prime is $\Theta((\log D)/(\log T+\log\log D))$, the complexity bound is correct as stated. □

In case the bound T on the number of nonzero terms is very bad, we could choose a smaller value of λ based on the true sparsity t before line 8, improving the cost of the remainder of the algorithm.

In addition, as our bound on possible number of "bad primes" seems to be quite loose. A more efficient approach in practice would be to replace the for loop on line 12 with one that starts with a prime much smaller than λ and incrementally searches for larger primes, until the product of all good primes is at least D. We could choose the lower bound to start searching from based on lower bounds on the birthday problem. That is, assuming (falsely) that the exponents are randomly distributed modulo p, start with the least p that will have no exponents collide modulo p with high probability. This would yield an algorithm more sensitive to the true bound on bad primes, but unfortunately gives a worse formal cost analysis.

7.3 Sparse interpolation over finite fields

We now examine the case that the ground field F is the finite field with q elements, which we denote $\mathbb{F}_q$. First we show how to effectively diversify the unknown polynomial f in order to complete Algorithm 7.1 for the case of large finite fields. Then we show how to extend this to a Las Vegas algorithm with the same complexity.

7.3.1 Diversification

For an unknown $f \in \mathbb{F}_q[x]$ given by a remainder black box, we must find an α so that $f(\alpha x)$ is diverse. A surprisingly simple trick works: evaluating $f(\alpha x)$ for a randomly chosen nonzero $\alpha \in \mathbb{F}_q$.

Theorem 7.4. *For $q \ge T(T-1)D+1$ and any T-sparse polynomial $f \in \mathbb{F}_q[x]$ with $\deg f < D$, if α is chosen uniformly at random from $\mathbb{F}_q^*$, the probability that $f(\alpha x)$ is diverse is at least $1/2$.*

Proof. Let $t \le T$ be the exact number of nonzero terms in f, and write $f = \sum_{1\le i\le t} c_i x^{e_i}$, with nonzero coefficients $c_i \in \mathbb{F}_q^*$ and $e_1 < e_2 < \cdots < e_t$. So the ith coefficient of $f(\alpha x)$ is $c_i \alpha^{e_i}$.

If $f(\alpha x)$ is *not* diverse, then we must have $c_i\alpha^{e_i} = c_j\alpha^{e_j}$ for some $i \ne j$. Therefore consider the polynomial $A \in \mathbb{F}_q[y]$ defined by

$$A = \prod_{1\le i<j\le t} \left(c_i y^{e_i} - c_j y^{e_j}\right).$$

We see that $f(\alpha x)$ is diverse if and only if $A(\alpha) \ne 0$, hence the number of roots of A over $\mathbb{F}_q$ is exactly the number of unlucky choices for α.

The polynomial A is the product of exactly $\binom{t}{2}$ binomials, each of which has degree less than D. Therefore

$$\deg A < \frac{T(T-1)D}{2},$$

and this also gives an upper bound on the number of roots of A. Hence $q - 1 \ge 2\deg A$, and at least half of the elements of $\mathbb{F}_q^*$ are not roots of A, yielding the stated result. □

Using this result, given a black box for f and the exact sparsity t of f, we can find an $\alpha \in \mathbb{F}_q$ such that $f(\alpha x)$ is diverse by sampling random values $\alpha \in \mathbb{F}_q$, evaluating $f(\alpha x) \operatorname{rem} x^p - 1$ for a single good prime p, and checking whether the polynomial is diverse. With probability at least $1-\mu$, this will succeed in finding a diversifying α after at most $\lceil \log_2(1/\mu)\rceil$ iterations. Therefore we can use this approach in Algorithm 7.1 with no effect on the asymptotic complexity.

7.3.2 Verification

So far, Algorithm 7.1 over a finite field is probabilistic of the Monte Carlo type; that is, it may give the wrong answer with some controllably-small probability. To provide a more robust Las Vegas probabilistic algorithm, we require only a fast way to check that a candidate answer is in fact correct. To do this, observe that given a modular black box for an unknown T-sparse $f \in \mathbb{F}_q[x]$ and an explicit T-sparse polynomial $g \in \mathbb{F}_q[x]$, we can construct a modular black box for the $2T$-sparse polynomial $f - g$ of their difference. Verifying that $f = g$ thus reduces to the well-studied problem of deterministic polynomial identity testing.

The following algorithm is due to (Bläser, Hardt, Lipton, and Vishnoi, 2009) and provides this check in essentially the same time as the interpolation algorithm; we restate it in Algorithm 7.2 for completeness and to use our notation.

Algorithm 7.2: Verification over finite fields

```
Input: $T, D, q \in \mathbb{N}$ and remainder black box for unknown $T$-sparse $f \in \mathbb{F}_q[x]$ with
       $\deg f \leq D$
Output: ZERO iff $f$ is identically zero
1 for the least $(T-1)\log_2 D$ primes $p$ do
2     Use black box to compute $f_p = f \operatorname{rem}(x^p - 1)$
3     if $f_p \neq 0$ then return NONZERO
4 return ZERO
```

Theorem 7.5. *Algorithm 7.2 works correctly as stated and uses at most*

$$O\left(\ell T \log D \cdot \mathsf{M}\left(T \log D \cdot (\log T + \log\log D)\right)\right)$$

field operations.

Proof. For correctness, notice that the requirements for a "good prime" for identity testing are much weaker than for interpolation. Here, we only require that a single nonzero term not collide with any other nonzero term. That is, every bad prime p will divide $e_j - e_1$ for some $2 \leq j \leq T$. There can be $\log_2 D$ distinct prime divisors of each $e_j - e_1$, and there are $T-1$ such differences. Therefore testing that the polynomial is zero modulo $x^p - 1$ for the first $(T-1)\log_2 D$ primes is sufficient to guarantee at least one nonzero evaluation of a nonzero T-sparse polynomial.

For the cost analysis, the prime number theorem (Bach and Shallit, 1996, Theorem 8.8.4), tells us that the first $(T-1)\log_2 D$ primes are each bounded by $O(T \cdot \log D \cdot (\log T + \log\log D))$. The stated bound follows directly. □

This provides all that we need to prove the main result of this section:

Theorem 7.6. *Given $q \geq T(T-1)D + 1$, any $T, D \in \mathbb{N}$, and a modular black box for unknown T-sparse $f \in \mathbb{F}_q[x]$ with $\deg f \leq D$, there is an algorithm that always produces the correct polynomial f and with high probability uses only $\tilde{O}\left(\ell T^2 \log^2 D\right)$ field operations.*

Proof. Use Algorithms 7.1 and 7.2 with $\mu = 1/2$, looping as necessary until the verification step succeeds. With high probability, only a constant number of iterations will be necessary, and so the cost is as stated. □

For the small field case, when $q \in O(T^2 D)$, the obvious approach would be to work in an extension E of size $O(\log T + \log D)$ over $\mathbb{F}_q$. Unfortunately, this would presumably increase the cost of each evaluation by a factor of $\log D$, potentially dominating our factor of T savings compared to the randomized version of (Garg and Schost, 2009) when the unknown polynomial has very few terms and extremely high degree.

In practice, it seems that a much smaller extension than this is sufficient in any case to make each $\gcd(e_j - e_i, q-1)$ small compared to $q-1$, but we do not yet know how to prove any tighter bound in the worst case.

7.4 Approximate sparse interpolation algorithms

In this section we consider the problem of interpolating an approximate sparse polynomial $f \in \mathbb{C}[x]$ from evaluations on the unit circle. We will generally assume that f is t-sparse:

$$f = \sum_{1 \le i \le t} c_i x^{e_i} \text{ for } c_i \in \mathbb{C} \text{ and } e_1 < \cdots < e_t = d. \tag{7.1}$$

We require a notion of size for such polynomials, and define the coefficient 2-norm of $f = \sum_{0 \le i \le d} f_i x^i$ as

$$\|f\| = \sqrt{\sum_{0 \le i \le d} |f_i|^2}.$$

The following identity relates the norm of evaluations on the unit circle and the norm of the coefficients. As in Section 7.2, for $f \in \mathbb{C}[x]$ is as in (7.1), we say that a prime p is a *good prime* for f if $p \nmid (e_i - e_j)$ for all $i \neq j$.

Lemma 7.7. *Let $f \in \mathbb{C}[x]$, p a good prime for f, and $\omega \in \mathbb{C}$ a pth primitive root of unity. Then*

$$\|f\|^2 = \frac{1}{p} \sum_{0 \le i < p} \left| f(\omega^i) \right|^2.$$

Proof. See Theorem 6.9 in the previous chapter. □

We can now formally define the approximate sparse univariate interpolation problem.

Definition 7.8. *Let $\epsilon > 0$ and assume there exists an unknown t-sparse $f \in \mathbb{C}[x]$ of degree at most D. An ϵ-*approximate black box *for f takes an input $\xi \in \mathbb{C}$ and produces a $\gamma \in \mathbb{C}$ such that $|\gamma - f(\xi)| \le \epsilon |f(\xi)|$.*

That is, the relative error of any evaluation is at most ϵ. As noted in the introduction, we will specify our input points exactly, at (relatively low order) roots of unity. The *approximate sparse univariate interpolation problem* is then as follows: given $D, T \in \mathbb{N}$ and $\delta \ge \epsilon > 0$, and an ϵ-approximate black box for an unknown T-sparse polynomial $f \in \mathbb{C}[x]$ of degree at most D, find a T-sparse polynomial $g \in \mathbb{C}[x]$ such that $\|f - g\| \le \delta \|g\|$.

The following theorem shows that t-sparse polynomials are well-defined by good evaluations on the unit circle.

Theorem 7.9. *Let $\epsilon > 0$ and $f \in \mathbb{C}[x]$ be a t-sparse polynomial. Suppose there exists a t-sparse polynomial $g \in \mathbb{C}[x]$ such that for a prime p which is good for both f and $f - g$, and pth primitive root of unity $\omega \in \mathbb{C}$, we have*

$$|f(\omega^i) - g(\omega^i)| \le \epsilon |f(\omega^i)| \quad \text{for } 0 \le i < p.$$

Then $\|f - g\| \le \epsilon \|f\|$. Moreover, if $g_0 \in \mathbb{C}[x]$ is formed from g by deleting all the terms not in the support of f, then $\|f - g_0\| \le 2\epsilon \|f\|$.

Proof. Summing over powers of ω we have

$$\sum_{0\le i<p} |f(\omega^i)-g(\omega^i)|^2 \le \epsilon^2 \sum_{0\le i<p} |f(\omega^i)|^2.$$

Thus, since p is a good prime for both $f-g$ and f, using Lemma 7.7, $p\cdot\left\|f-g\right\|^2 \le \epsilon^2\cdot p\cdot\left\|f\right\|^2$ and $\left\|f-g\right\| \le \epsilon\left\|f\right\|$.

Since $g-g_0$ has no support in common with f

$$\left\|g-g_0\right\| \le \left\|f-g\right\| \le \epsilon\left\|f\right\|.$$

Thus

$$\begin{aligned}\left\|f-g_0\right\| &= \left\|f-g+(g-g_0)\right\|\\ &\le \left\|f-g\right\|+\left\|g-g_0\right\| \le 2\epsilon\left\|f\right\|. \quad \square\end{aligned}$$

$\square$

In other words, any t-sparse polynomial whose values are very close to f must have the same support except possibly for some terms with very small coefficients.

7.4.1 Computing the norm of an approximate sparse polynomial

As a warm-up exercise, consider the problem of computing an approximation for the 2-norm of an unknown polynomial given by a black box. Of course we this is an easier problem than actually interpolating the polynomial, but the technique bears some similarity. In practice, we might also use an approximation for the norm to normalize the black-box polynomial, simplifying certain computations and bounds calculations.

Let $0<\epsilon<1/2$ and assume we are given an ϵ-approximate black box for some t-sparse polynomial $f\in\mathbb{C}[x]$. We first consider the problem of computing $\left\|f\right\|$.

Algorithm 7.3: Approximate norm

```
Input: $T, D \in \mathbb{N}$ and $\epsilon$-approximate black box for unknown $T$-sparse $f \in \mathbb{C}[x]$ with
       $\deg f \le D$
Output: $\sigma \in \mathbb{R}$, an approximation to $\left\|f\right\|$
1 $\lambda \leftarrow \max\left(21, \left\lceil \frac{5}{3} t(t-1)\ln d \right\rceil\right)$
2 Choose a prime $p$ randomly from $\{\lambda,\dots,2\lambda\}$
3 $\omega \leftarrow \exp(2\pi i/p)$
4 $w \leftarrow (f(\omega^0),\dots,f(\omega^{p-1})) \in \mathbb{C}^p$ computed using the black box
5 return $(1/\sqrt{p})\cdot\|w\|$
```

Theorem 7.10. *Algorithm 7.3 works as stated. On any invocation, with probability at least* $1/2$, *it returns a value* $\sigma\in\mathbb{R}_{\ge 0}$ *such that*

$$(1-2\epsilon)\left\|f\right\| < \sigma < (1+\epsilon)\left\|f\right\|.$$

Proof. Let $v = (f(\omega^0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ be the vector of exact evaluations of f. Then by the properties of our ϵ-approximate black box we have $w = v + \epsilon\Delta$, where $|\Delta_i| < |f(\omega^i)|$ for $0 \le i < p$, and hence $\|\Delta\| < \|v\|$. By the triangle inequality $\|w\| \le \|v\| + \epsilon\|\Delta\| < (1+\epsilon)\|v\|$. By Lemmas 7.2 and 7.7, $\|v\| = \sqrt{p}\left\|f\right\|$ with probability at least 1/2, so $(1/\sqrt{p}) \cdot \|w\| < (1+\epsilon)\left\|f\right\|$ with this same probability.

To establish a lower bound on the output, note that we can make error in the evaluation relative to the output magnitude: because $\epsilon < 1/2$, $|f(\omega^i) - w_i| < 2\epsilon|w_i|$ for $0 \le i < p$. We can write $v = w + 2\epsilon\nabla$, where $\|\nabla\| < \|w\|$. Then $\|v\| \le (1+2\epsilon)\|w\|$, and $(1-2\epsilon)\left\|f\right\| < (1/\sqrt{p}) \cdot \|w\|$. □

7.4.2 Constructing an ϵ-approximate remainder black box

Assume that we have chosen a good prime p for a t-sparse $f \in \mathsf{F}[x]$. Our goal in this subsection is a simple algorithm and numerical analysis to accurately compute $f \operatorname{rem} x^p - 1$.

Assume that $f \operatorname{rem} x^p - 1 = \sum_{0 \le i < p} b_i x^i$ exactly. For a primitive pth root of unity $\omega \in \mathbb{C}$, let $V(\omega) \in \mathbb{C}^{p \times p}$ be the Vandermonde matrix built from the points $1, \omega, \ldots, \omega^{p-1}$. Recall that $V(\omega) \cdot (b_0, \ldots, b_{p-1})^T = (f(\omega^0), \ldots, f(\omega^{p-1}))^T$ and $V(\omega^{-1}) = p \cdot V(\omega)^{-1}$. Matrix vector product by such Vandermonde matrices is computed very quickly and in a numerically stable manner by the Fast Fourier Transform (FFT).

Algorithm 7.4: Approximate Remainder

```
  Input: An $\epsilon$-approximate black box for the unknown $t$-sparse $f \in \mathbb{C}[x]$, and $p \in \mathbb{N}$, a
         good prime for $f$
  Output: $h \in \mathbb{C}[x]$ such that $\left\|(f \operatorname{rem} x^p - 1) - h\right\| \le \epsilon \left\|f\right\|$.
1 $w \leftarrow (f(\omega^0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ computed using the $\epsilon$-approximate black box for $f$
2 $u \leftarrow (1/p) \cdot V(\omega^{-1}) w \in \mathbb{C}^p$ using the FFT algorithm
3 return $h = \sum_{0 \le i < p} u_i x^i \in \mathbb{C}[x]$
```

Theorem 7.11. *Algorithm 7.4 works as stated, and*

$$\left\|(f \operatorname{rem} x^p - 1) - h\right\| \le \epsilon \left\|f\right\|.$$

It requires $O(p \log p)$ floating point operations and p evaluations of the black box.

Proof. Because f and $f \operatorname{rem} x^p - 1$ have exactly the same coefficients (p is a good prime for f), they have exactly the same norm. The FFT in Step 2 is accomplished in $O(p \log p)$ floating point operations. This algorithm is numerically stable since $(1/\sqrt{p}) \cdot V(\omega^{-1})$ is unitary. That is, assume $v = (f(\omega_0), \ldots, f(\omega^{p-1})) \in \mathbb{C}^p$ is the vector of *exact* evaluations of f, so $\|v - w\| \le \epsilon\|v\|$ by the black box specification. Then, using the fact that $\|v\| = \sqrt{p}\left\|f\right\|$,

$$\left\|(f \operatorname{rem} x^{p-1}) - h\right\| = \left\|\frac{1}{p} V(\omega^{-1}) v - \frac{1}{p} V(\omega^{-1}) w\right\|$$

$$= \frac{1}{\sqrt{p}} \left\|\frac{1}{\sqrt{p}} V(\omega^{-1}) \cdot (v - w)\right\| = \frac{1}{\sqrt{p}} \|v - w\| \le \frac{\epsilon}{\sqrt{p}} \|v\| = \epsilon \left\|f\right\|.$$ □

7.4.3 Creating ϵ-diversity

First, we extend the notion of polynomial diversity to the approximate case.

Definition 7.12. *Let $f \in \mathbb{C}[x]$ be a t-sparse polynomial as in* (7.1) *and $\delta \geq \epsilon > 0$ such that $|c_i| \geq \delta \left\|f\right\|$ for $1 \leq i \leq t$. The polynomial f is said to be ϵ-*diverse *if and only if every pair of distinct coefficients is at least $\epsilon \left\|f\right\|$ apart. That is, for every $1 \leq i < j \leq t$, $|c_i - c_j| \geq \epsilon \left\|f\right\|$.*

Intuitively, if $(\epsilon/2)$ corresponds to the machine precision, this means that an algorithm can reliably distinguish the coefficients of a ϵ-diverse polynomial. We now show how to choose a random α to guarantee ϵ-diversity.

Theorem 7.13. *Let $\delta \geq \epsilon > 0$ and $f \in \mathbb{C}[x]$ a t-sparse polynomial whose non-zero coefficients are of magnitude at least $\delta \left\|f\right\|$. If s is a prime satisfying $s > 12$ and*

$$t(t-1) \leq s \leq 3.1\frac{\delta}{\epsilon},$$

then for $\zeta = \mathbf{e}^{2\pi\mathbf{i}/s}$ an s-PRU and $k \in \mathbb{N}$ chosen uniformly at random from $\{0,1,\ldots,s-1\}$, $f(\zeta^k x)$ is ϵ-diverse with probability at least $\frac{1}{2}$.

Proof. For each $1 \leq i \leq t$, write the coefficient c_i in polar notation to base ζ as $c_i = r_i\zeta^{\theta_i}$, where each r_i and θ_i are nonnegative real numbers and $r_i \geq \delta \left\|f\right\|$.

Suppose $f(\zeta^k x)$ is *not* ϵ-diverse. Then there exist indices $1 \leq i < j \leq t$ such that

$$\left|r_i\zeta^{\theta_i}\zeta^{ke_i} - r_j\zeta^{\theta_j}\zeta^{ke_j}\right| \leq \epsilon \left\|f\right\|.$$

Because $\min(r_i, r_j) \geq \delta \left\|f\right\|$, the value of the left hand side is at least $\delta \left\|f\right\| \cdot \left|\zeta^{\theta_i+ke_i} - \zeta^{\theta_j+ke_j}\right|$. Dividing out $\zeta^{\theta_j+ke_i}$, we get

$$\left|\zeta^{\theta_i-\theta_j} - \zeta^{k(e_j-e_i)}\right| \leq \frac{\epsilon}{\delta}.$$

By way of contradiction, assume there exist distinct choices of k that satisfy the above inequality, say $k_1, k_2 \in \{0,\ldots,s-1\}$. Since $\zeta^{\theta_i-\theta_j}$ and $\zeta^{e_j-e_i}$ are a fixed powers of ζ not depending on the choice of k, this means

$$\left|\zeta^{k_1(e_j-e_i)} - \zeta^{k_2(e_j-e_i)}\right| \leq 2\frac{\epsilon}{\delta}.$$

Because s is prime, $e_i \neq e_j$, and we assumed $k_1 \neq k_2$, the left hand side is at least $|\zeta - 1|$. Observe that $2\pi/s$, the distance on the unit circle from 1 to ζ, is a good approximation for this Euclidean distance when s is large. In particular, since $s > 12$,

$$\frac{|\zeta-1|}{2\pi/s} > \frac{\sqrt{2}\left(\sqrt{3}-1\right)/2}{\pi/6},$$

and therefore $|\zeta - 1| > 6\sqrt{2}(\sqrt{3}-1)/s > 6.2/s$, which from the statement of the theorem is at least $2\epsilon/\delta$. This is a contradiction, and therefore the assumption was false; namely, there is at most one choice of k such that the i'th and j'th coefficients collide.

Then, since there are exactly $\binom{t}{2}$ distinct pairs of coefficients, and $s \geq t(t-1) = 2\binom{t}{2}$, $f(\zeta^k x)$ is diverse for at least half of the choices for k. □

Algorithm 7.5: Adaptive diversification

```
Input: $\epsilon$-approximate black box for $f$, known good prime $p$, known sparsity $t$
Output: $\zeta, k$ such that $f(\zeta^k x)$ is $\epsilon$-diverse, or FAIL
1 $s \leftarrow 1, \quad \delta \leftarrow \infty, \quad f_p \leftarrow 0$
2 while $s \leq t^2$ and $\#\{$coeffs $c$ of $f_s$ s.t. $|c| \geq \delta\} < t$ do
3     $s \leftarrow$ least prime $\geq 2s$
4     $\zeta \leftarrow \exp(2\pi\mathbf{i}/s)$
5     $k \leftarrow$ random integer in $\{0, 1, \ldots, s-1\}$
6     Compute $f_s = f(\zeta^k x) \operatorname{rem} x^p - 1$
7     $\delta \leftarrow$ least number s.t. all coefficients of $f_s$ at least $\delta$ in absolute value are pairwise
      $\epsilon$-distinct
8 if $\delta > 2\epsilon$ then return FAIL
9 else return $\zeta^k$
```

We note that the diversification which maps $f(x)$ to $f(\zeta^k x)$ and back is numerically stable since ζ is on the unit circle.

In practice, the previous theorem will be far too pessimistic. We therefore propose the method of Algorithm 7.5 to adaptively choose s, δ, and ζ^k simultaneously, given a good prime p.

Suppose there exists a threshold $S \in \mathbb{N}$ such that for all primes $s > S$, a random sth primitive root of unity ζ^k makes $f(\zeta^k x)$ ϵ-diverse with high probability. Then Algorithm 7.5 will return a root of unity whose order is within a constant factor of S, with high probability. From the previous theorem, if such an S exists it must be $O(t^2)$, and hence the number of iterations required is $O(\log t)$.

Otherwise, if no such S exists, then we cannot diversify the polynomial. Roughly speaking, this corresponds to the situation that f has too many coefficients with absolute value close to the machine precision. However, the diversification is not actually necessary in the approximate case to guarantee numerical stability. At the cost of more evaluations, as well as the need to factor an integer polynomial, the algorithm of (Garg and Schost, 2009) for finite fields can be adapted quite nicely for the approximate case. This is essentially the approach we outline below, and even if the diversification step is omitted, the stated numerical properties of the computation will still hold true.

7.4.4 Approximate interpolation algorithm

We now plug our ϵ-approximate remainder black box, and method for making f ϵ-diverse, into our generic Algorithm 7.1 to complete our algorithm for approximate interpolation.

Theorem 7.14. *Let $\delta > 0$, $f \in \mathbb{C}[x]$ with degree at most D and sparsity at most T, and suppose all nonzero coefficients c of f satisfy $|c| > \delta \left\|f\right\|$. Suppose also that $\epsilon < 1.5\delta/(T(T-1))$, and we are given an ϵ-approximate black box for f. Then, for any $\mu < 1/2$ we have an algorithm to produce a $g \in \mathbb{C}[x]$ satisfying the conditions of Theorem 7.9. The algorithm succeeds with* *probability at least* $1-\mu$ *and uses* $\tilde{O}(T^2 \cdot \log(1/\mu) \cdot \log^2 D)$ *black box evaluations and floating point operations.*

Proof. Construct an approximate remainder black box for f using Algorithm 7.4. Then run Algorithm 7.1 using this black box as input. On step 8 of Algorithm 7.1, run Algorithm 7.5, iterating steps 5–7 $\lceil \log_2(3/\mu) \rceil$ times on each iteration through the while loop to choose a diversifying $\alpha = \zeta^k$ with probability at least $1-\mu/3$.

The cost comes from Theorems 7.3 and 7.11 along with the previous discussion and Theorem 7.13. □

Observe that the resulting algorithm is Monte Carlo, but could be made Las Vegas by combining the finite fields zero testing algorithm discussed in Section 7.3.2 with the guarantees of Theorem 7.9.

7.5 Implementation results

We implemented our algorithms in the MVP library, using GMP (Granlund et al., 2010) and NTL (Shoup, 2009) for the exponent arithmetic. For comparison with the algorithm of Garg and Schost (2009), we also used NTL's squarefree polynomial factorization routines. We note that, in our experiments, the cost of integer polynomial factorization (for Garg & Schost) and Chinese remaindering were always negligible.

In our timing results, "G&S Determ" refers to the deterministic algorithm as stated in Garg and Schost (2009) and "Alg 7.1" is the algorithm we have presented here over finite fields, without the verification step. We also developed and implemented a more adaptive, Monte Carlo version of these algorithms, as briefly described at the end of Section 7.2. The basic idea is to sample modulo $x^p - 1$ for just one prime $p \in \Theta(t^2 \log d)$ that is good with high probability, then to search for much smaller good primes. This good prime search starts at a lower bound of order $\Theta(t^2)$ based on the birthday problem, and finds consecutively larger primes until enough primes have been found to recover the symmetric polynomial in the exponents (for Garg & Schost) or just the exponents (for our method). The corresponding improved algorithms are referred to as "G&S MC" and "Alg 7.1++" in the table, respectively.

Table 7.3 summarizes some timings for these four algorithms on our benchmarking machine. Note that the numbers listed reflect the *base-2 logarithm* of the degree bound and the sparsity bound for the randomly-generated test cases. The tests were all performed over the finite field $\mathbb{Z}/65521\mathbb{Z}$. This modulus was chosen for convenience of implementation, although other methods such as the Ben-Or and Tiwari algorithm might be more efficient in this particlar field since discrete logarithms could be computed quickly. However, observe that our algorithms (and those from Garg and Schost) have only poly-logarithmic dependence on the field size, and so will eventually dominate.

The timings are mostly as expected based on our complexity estimates, and also confirm our suspicion that primes of size $O(t^2)$ are sufficient to avoid exponent collisions. It is satisfying but not particularly surprising to see that our "Alg 7.1++" is the fastest on all inputs, as

$\log_2 D$	T	G&S Determ	G&S MC	Alg 7.1	Alg 7.1++
12	10	3.77	0.03	0.03	0.01
16	10	46.82	0.11	0.11	0.08
20	10	—	0.38	0.52	0.33
24	10	—	0.68	0.85	0.38
28	10	—	1.12	2.35	0.53
32	10	—	1.58	2.11	0.66
12	20	37.32	0.15	0.02	0.02
16	20	—	0.91	0.52	0.28
20	20	—	3.5	3.37	1.94
24	20	—	6.59	5.94	2.99
28	20	—	10.91	10.22	3.71
32	20	—	14.83	16.22	4.24
12	30	—	0.31	0.01	0.01
16	30	—	3.66	1.06	0.65
20	30	—	10.95	6.7	3.56
24	30	—	25.04	12.42	9.32
28	30	—	38.86	19.36	13.8
32	30	—	62.53	68.1	14.66
12	40	—	0.58	0.01	0.02
16	40	—	8.98	3.7	1.54
20	40	—	30.1	12.9	8.42
24	40	—	67.97	38.34	16.57
28	40	—	—	73.69	36.24
32	40	—	—	—	40.79

Table 7.3: Finite Fields Algorithm Timings

Noise	Mean Error	Median Error	Max Error
0	4.440 e−16	4.402 e−16	8.003 e−16
$\pm 10^{-12}$	1.113 e−14	1.119 e−14	1.179 e−14
$\pm 10^{-9}$	1.149 e−11	1.191 e−11	1.248 e−11
$\pm 10^{-6}$	1.145 e−8	1.149 e−8	1.281 e−8

Table 7.4: Approximate Algorithm Stability

all the algorithms have a similar basic structure. Had we compared to the Ben-Or and Tiwari or Zippel's method, they would probably be more efficient for small sizes, but would be easily beaten for large degree and arbitrary finite fields as their costs are super-polynomial.

The implementation of the approximate algorithm uses machine `double` precision (based on the IEEE standard), the built-in C++ `complex<double>` type, and the popular Fastest Fourier Transform in the West (FFTW) package for computing FFTs (Frigo and Johnson, 2005). Our stability results are summarized in Table 7.4. Each test case was randomly generated with degree at most 2^{20} and at most 50 nonzero terms. We varied the precision as specified in the table and ran 10 tests in each range. Observe that the error in our results was often *less* than the ϵ error on the evaluations themselves.

7.6 Conclusions

We have presented two new algorithms, using the basic idea of Garg and Schost (2009), plus our new technique of diversification, to gain improvements for sparse interpolation over large finite fields and approximate complex numbers.

There is much room for improvement in both of these methods. For the case of finite fields, the limitation to *large* finite fields of size at least $\Omega(t^2 d^n)$ is in some sense the interesting case, since for very small finite fields it will be faster to use Ben-Or and Tiwari's algorithm and compute the discrete logs as required. However, the restriction on field size for our algorithms does seem rather harsh, and we would like to reduce it, perhaps to a bound of size only $t^{O(1)}$. This would have a tremendous advantage because working in an extension field to guarantee that size would only add a logarithmic factor to the overall complexity. By contrast, working an extension large enough to satisfy the conditions of our algorithm could add a factor of $\tilde{O}(\log d)$ to the complexity, which is significant for lacunary algorithms.

Over approximate complex numbers, our algorithm provides the first provably numerically stable method for sparse interpolation. However, this improved numerical stability is at the expense of extra computational cost and (more significantly) extra black-box probes. For many applications, the cost of probing the unknown function easily dominates the cost of the interpolation algorithm, and therefore reducing the number of probes as much as possible is highly desirable. Ideally, we would like to have an approximate interpolation algorithm with the numerical stability of ours, but using only $O(t)$ probes, as in Ben-Or and Tiwari's algorithm.

Another area for potential improvement is in the bounds used for the number of possible bad primes, in both algorithms. From our experiments and intuition, it seems that using primes of size roughly $O(t^2 + t\log d)$ should be sufficient, rather than the $O(t^2 \log d)$ size our current bounds require. However, proving this seems quite challenging. These and other related bounds will be discussed at greater length in the next chapter.

Bibliography

Martín Avendaño, Teresa Krick, and Ariel Pacetti. Newton-Hensel interpolation lifting. *Found. Comput. Math.*, 6(1):81–120, 2006. ISSN 1615-3375.
doi: 10.1007/s10208-005-0172-3. Referenced on page 121.

Eric Bach and Jeffrey Shallit. *Algorithmic number theory. Vol. 1*. Foundations of Computing Series. MIT Press, Cambridge, MA, 1996. ISBN 0-262-02405-5. Referenced on page 127.

Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 301–309, New York, NY, USA, 1988. ACM. ISBN 0-89791-264-0.
doi: 10.1145/62212.62241. Referenced on pages 119 and 122.

Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Information Processing Letters*, 109(3):187 – 192, 2009. ISSN 0020-0190.
doi: 10.1016/j.ipl.2008.09.029. Referenced on page 126.

John F. Canny, Erich Kaltofen, and Lakshman Yagati. Solving systems of nonlinear polynomial equations faster. In *Proceedings of the ACM-SIGSAM 1989 international symposium on Symbolic and algebraic computation*, ISSAC '89, pages 121–128, New York, NY, USA, 1989. ACM. ISBN 0-89791-325-6.
doi: 10.1145/74540.74556. Referenced on page 118.

Michael Clausen, Andreas Dress, Johannes Grabmeier, and Marek Karpinski. On zero-testing and interpolation of k-sparse multivariate polynomials over finite fields. *Theoretical Computer Science*, 84(2):151–164, 1991. ISSN 0304-3975.
doi: 10.1016/0304-3975(91)90157-W. Referenced on page 118.

Annie Cuyt and Wen-shin Lee. A new algorithm for sparse interpolation of multivariate polynomials. *Theoretical Computer Science*, 409(2):180–185, 2008. ISSN 0304-3975.
doi: 10.1016/j.tcs.2008.09.002. Symbolic-Numerical Computations. Referenced on page 122.

Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. ISSN 0020-0190.
doi: 10.1016/0020-0190(78)90067-4. Referenced on page 118.

Angel Díaz and Erich Kaltofen. On computing greatest common divisors with polynomials given by black boxes for their evaluations. In *Proceedings of the 1995 international symposium on Symbolic and algebraic computation*, ISSAC '95, pages 232–239, New York, NY, USA, 1995. ACM. ISBN 0-89791-699-9.
doi: `10.1145/220346.220375`. Referenced on page 118.

Angel Díaz and Erich Kaltofen. FOXBOX: a system for manipulating symbolic objects in black box representation. In *Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, ISSAC '98, pages 30–37, New York, NY, USA, 1998. ACM. ISBN 1-58113-002-3.
doi: `10.1145/281508.281538`. Referenced on page 118.

Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation". Referenced on page 135.

Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theoretical Computer Science*, 410(27-29):2659–2662, 2009. ISSN 0304-3975.
doi: `10.1016/j.tcs.2009.03.030`. Referenced on pages 117, 119, 122, 123, 127, 132, 133 and 135.

Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, second edition, 2003. ISBN 0521826462. Referenced on page 119.

Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. *Journal of Symbolic Computation*, 44(8):943 – 959, 2009. ISSN 0747-7171.
doi: `10.1016/j.jsc.2008.11.003`. Referenced on page 122.

Torbjörn Granlund et al. *GNU Multiple Precision Arithmetic Library, The*. Free Software Foundation, Inc., 4.3.2 edition, January 2010.
URL `http://gmplib.org/`. Referenced on page 133.

Dima Yu. Grigoriev, Marek Karpinski, and Michael F. Singer. Fast parallel algorithms for sparse multivariate polynomial interpolation over finite fields. *SIAM Journal on Computing*, 19(6): 1059–1063, 1990.
doi: `10.1137/0219073`. Referenced on page 121.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, 58:13–30, 1963. ISSN 0162-1459.
URL `http://www.jstor.org/stable/2282952`. Referenced on page 125.

Seyed Mohammad Mahdi Javadi and Michael Monagan. A sparse modular GCD algorithm for polynomials over algebraic function fields. In *Proceedings of the 2007 international symposium on Symbolic and algebraic computation*, ISSAC '07, pages 187–194, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-743-8.
doi: `10.1145/1277548.1277575`. Referenced on page 118.

Seyed Mohammad Mahdi Javadi and Michael Monagan. On factorization of multivariate polynomials over algebraic number and function fields. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, ISSAC '09, pages 199–206, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-609-0.
doi: 10.1145/1576702.1576731. Referenced on page 118.

Seyed Mohammad Mahdi Javadi and Michael Monagan. Parallel sparse polynomial interpolation over finite fields. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, PASCO '10, pages 160–168, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0067-4.
doi: 10.1145/1837210.1837233. Referenced on page 121.

Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13:1–46, 2004. ISSN 1016-3328.
doi: 10.1007/s00037-004-0182-6. 10.1007/s00037-004-0182-6. Referenced on page 118.

Erich Kaltofen and Wen-shin Lee. Early termination in sparse interpolation algorithms. *Journal of Symbolic Computation*, 36(3-4):365–400, 2003. ISSN 0747-7171.
doi: 10.1016/S0747-7171(03)00088-9. ISSAC 2002. Referenced on page 121.

Erich Kaltofen and Barry M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990. ISSN 0747-7171.
doi: 10.1016/S0747-7171(08)80015-6. Computational algebraic complexity editorial. Referenced on page 118.

Erich Kaltofen and Lakshman Yagati. Improved sparse multivariate polynomial interpolation algorithms. In P. Gianni, editor, *Symbolic and Algebraic Computation*, volume 358 of *Lecture Notes in Computer Science*, pages 467–474. Springer Berlin / Heidelberg, 1989.
doi: 10.1007/3-540-51084-2_44. Referenced on pages 119 and 121.

Erich Kaltofen and Zhengfeng Yang. On exact and approximate interpolation of sparse rational functions. In *Proceedings of the 2007 international symposium on Symbolic and algebraic computation*, ISSAC '07, pages 203–210, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-743-8.
doi: 10.1145/1277548.1277577. Referenced on page 122.

Erich Kaltofen, Y. N. Lakshman, and John-Michael Wiley. Modular rational sparse multivariate polynomial interpolation. In *Proceedings of the international symposium on Symbolic and algebraic computation*, ISSAC '90, pages 135–139, New York, NY, USA, 1990. ACM. ISBN 0-201-54892-5.
doi: 10.1145/96877.96912. Referenced on page 121.

Erich Kaltofen, Zhengfeng Yang, and Lihong Zhi. On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms. In *Proceedings of the 2007 international workshop on Symbolic-numeric computation*, SNC '07, pages 11–17, New York, NY, USA, 2007. ACM.

ISBN 978-1-59593-744-5.
doi: 10.1145/1277500.1277503. Referenced on page 122.

Erich Kaltofen, John P. May, Zhengfeng Yang, and Lihong Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symbolic Computation*, 43(5):359–376, 2008. ISSN 0747-7171.
doi: 10.1016/j.jsc.2007.11.005. Referenced on page 118.

Erich L. Kaltofen. Fifteen years after DSC and WLSS2: What parallel computations I do today [invited lecture at PASCO 2010]. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*, PASCO '10, pages 10–17, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0067-4.
doi: 10.1145/1837210.1837213. Referenced on pages 119 and 120.

Yishay Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM Journal on Computing*, 24(2):357–368, 1995.
doi: 10.1137/S0097539792239291. Referenced on page 122.

J. M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 32(143): 918–924, 1978. ISSN 0025-5718.
doi: 10.1090/S0025-5718-1978-0491431-9. Referenced on page 119.

J. M. Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13:437–447, 2000. ISSN 0933-2790.
doi: 10.1007/s001450010010. Referenced on page 119.

J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.*, 6:64–94, 1962.
URL http://projecteuclid.org/euclid.ijm/1255631807. Referenced on page 123.

Nitin Saxena. Progress on polynomial identity testing. *Bull. EATCS*, 99:49–79, 2009. Referenced on page 118.

J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, October 1980. ISSN 0004-5411.
doi: 10.1145/322217.322225. Referenced on page 118.

Victor Shoup. NTL: A Library for doing Number Theory. Online, August 2009.
URL http://www.shop.net/ntl/. Version 5.5.2. Referenced on page 133.

Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. Referenced on page 118.

A. J. Sommese and C. W. Wampler. *Numerical solution of polynomial systems arising in engineering and science*. World Scientific, Singapore, 2005. Referenced on page 118.

Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Numerical decomposition of the solution sets of polynomial systems into irreducible components. *SIAM Journal on Numerical Analysis*, 38(6):2022–2046, 2001.
doi: 10.1137/S0036142900372549. Referenced on page 118.

Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler. Numerical factorization of multivariate complex polynomials. *Theoretical Computer Science*, 315(2-3):651–669, 2004. ISSN 0304-3975.
doi: 10.1016/j.tcs.2004.01.011. Referenced on page 118.

Hans J. Stetter. *Numerical Polynomial Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004. ISBN 0898715571. Referenced on page 118.

Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward Ng, editor, *Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer Berlin / Heidelberg, 1979.
doi: 10.1007/3-540-09519-5_73. Referenced on page 118.

Richard Zippel. Interpolating polynomials from their values. *Journal of Symbolic Computation*, 9(3):375–403, 1990. ISSN 0747-7171.
doi: 10.1016/S0747-7171(08)80018-1. Computational algebraic complexity editorial. Referenced on pages 120 and 121.