Interpolation Decoding
(A Chapter in the Concise Encyclopedia of Coding Theory)

Swastik Kopparty

1 Introduction

In this chapter we will see some beautiful algorithmic ideas based on *polynomial interpolation* for decoding algebraic codes. Here we will focus only on (Generalized) Reed-Solomon codes, but these ideas extend very naturally to the important class of algebraic-geometric codes.

Let us quickly recall the "polynomial-evaluation" based definition of (narrow-sense) Reed-Solomon codes from Chapter **??**. Following the notation from that chapter, let q be a prime power, let $\alpha \in \mathbb{F}_q$ be a generator of the multiplicative group of $\mathbb{F}_q$, and define:[1]

$$\mathcal{P}_{k,q} = \{p(x) \in \mathbb{F}_q[x] \mid \deg(p) < k\}$$

$$\mathcal{RS}[q,k] = \{(p(\alpha^0), p(\alpha^1), p(\alpha^2), \ldots, p(\alpha^{q-2}) \mid p(x) \in \mathcal{P}_{k,q}\} \subseteq \mathbb{F}_q^{q-1}.$$

Informally, the codewords of $\mathcal{RS}[q,k]$ are the vectors of all evaluations of a polynomial of degree $< k$ at all the nonzero points of $\mathbb{F}_q$.

More generally, we can take an arbitrary $S \subseteq \mathbb{F}_q$ with $|S| = n < q$ and consider evaluations of polynomials of degree $< k$ at all the points of S. This code is known as the **Generalized Reed-Solomon** code[2]:

$$\mathcal{GRS}[q,k,S] = \{(p(\beta))_{\beta \in S} \mid p(x) \in \mathcal{P}_{k,q}\}.$$

Abusing notation, we will sometimes refer to the polynomial $p(x)$ of degree $< k$ as codewords.

Working with this more general family of codes greatly clarifies and even motivates the important ideas in this chapter. We begin with the most basic and useful fact about polynomials.

Lemma 1.1. *Let $p(x) \in \mathbb{F}_q[x]$ be a nonzero polynomial of degree at most d. Then the number of $u \in \mathbb{F}_q$ such that $p(u) = 0$ is at most d.*

This lemma lets us compute the minimum distance of $\mathcal{GRS}[q,k,S]$.

Lemma 1.2. *Let $S \subseteq \mathbb{F}_q$ and let $n = |S|$. Suppose $1 \leq k \leq n$. Then $\mathcal{GRS}[q,k,S]$ is an $[n, k, n-k+1]$ code.*

Indeed, by Lemma 1.1 every nonzero codeword of $\mathcal{GRS}[q,k,S]$ has at most $k-1$ coordinates equal to 0, and thus at least $n-k+1$ nonzero coordinates. This implies that the minimum distance is at least $n-k+1$. Finally, we can check that this bound is achieved by the codeword coming from the polynomial $p(x) = \prod_{i=1}^{k-1}(x - u_i)$, where $u_1, \ldots, u_{k-1}$ are arbitrary distinct elements of S.

For generalized Reed-Solomon codes, the question of decoding up to half the minimum distance takes on the following pleasant form. Let $e = \lfloor (n-k)/2 \rfloor$. We are given a function $r : S \to \mathbb{F}_q$ representing the received vector. We would like to find the unique polynomial $p(x) \in \mathbb{F}_q[x]$ of degree $< k$ such that $p(u) = r(u)$ for all but at most e values of $u \in S$.

[1] We will follow the convention that the degree of the 0 polynomial is $-\infty$.

[2] Sometimes the word generalized is omitted.

In this formulation, the problem looks like one of "error-tolerant" polynomial interpolation. Classical polynomial interpolation is the problem of finding a low degree polynomial taking desired values at certain points. Here we need to do interpolation despite some of the values being wrong.

There is a naive brute force search algorithm for this problem, namely to try all $\binom{n}{\leq e}$ possible subsets of S as candidates for the set of error locations E, and to do standard polynomial interpolation through the remaining points. This takes time exponential in e.

Remarkably, there are polynomial time algorithms for this problem! In time $(n \log q)^{O(1)}$ one can find the polynomial p. The first such algorithm was found by Peterson [Pet60]. Below we will see the ingenious algorithm of Berlekamp and Welch [BW] for this problem. Later we will see powerful generalizations and extensions of this algorithm by Sudan [Sud97] and Guruswami-Sudan [GS99].

2 The Berlekamp-Welch Algorithm

Let $p(x) \in \mathbb{F}_q[x]$ be the polynomial that we are trying to find, namely the polynomial with degree $< k$ such that $p(u) = r(u)$ for all but at most e values of u. Let $E = \{u \in S \mid p(u) \neq r(u)\}$ be the set of error locations. Let $Z(x) \in \mathbb{F}_q[x]$ be the polynomial given by:

$$Z(x) = \prod_{u \in E} (x - u).$$

$Z(x)$ is called the **error-locating polynomial**.

We do not know $Z(x)$ (indeed, finding $Z(x)$ is as hard as finding $p(x)$). Nevertheless thinking about $Z(x)$ will motivate our algorithm.

The crux of the Berlekamp-Welch algorithm is the following identity. For each $u \in S$, we have:

$$Z(u) \cdot r(u) = Z(u) \cdot p(u) \tag{1}$$

Indeed, if $u \notin E$, then $r(u) = p(u)$ and so the above identity holds. Otherwise $u \in E$ and $Z(u) = 0$, and so the above identity holds.

Let $W(x) \in \mathbb{F}_q[x]$ be the polynomial $Z(x) \cdot p(x)$. Observe that $\deg(Z) \leq e$ and $\deg(W) < k + e$. This motivates the following idea: let us try to search for the polynomials Z and W by solving the linear equations

$$Z(u) \cdot r(u) = W(u)$$

for each $u \in S$. Hopefully we will then recover $p(x)$ as $W(x)/Z(x)$. This algorithm actually works, but its analysis is more subtle than the naive arguments given above suggest.

We formally present the algorithm below.

Algorithm RSDecode
Input: $r : S \to \mathbb{F}_q$.

1. **Interpolation:** Let $a_0, a_1, \ldots, a_e$ and $b_0, b_1, \ldots, b_{k+e-1}$ be indeterminates. Consider the following system of homogeneous linear equations in these indeterminates, one equation for each $u \in S$:

$$\left(\sum_{i=0}^{e} a_i u^i \right) \cdot r(u) = \sum_{j=0}^{k+e-1} b_j u^j. \tag{2}$$

   Solve this system to get a *nonzero* solution $(a_0, \ldots, a_e, b_0, \ldots, b_{k+e-1}) \in \mathbb{F}_q^{k+2e+1}$. If there is no nonzero solution, FAIL.

2. **Polynomial Algebra:** For the solution found in the previous step, define $A(x), B(x) \in \mathbb{F}_q[x]$ by:

$$A(x) = \sum_{i=0}^{e} a_i x^i,$$

$$B(x) = \sum_{j=0}^{k+e-1} b_j x^j.$$

If $A(x)$ divides $B(x)$, then return $B(x)/A(x)$ (otherwise FAIL).

We will show the following theorem.

Theorem 2.1. *Suppose r is within Hamming distance e of some codeword of $\mathcal{GRS}(q, k, S)$. Then Algorithm RSDecode will output that codeword.*

Before analyzing correctness of the algorithm, we make some observations about its running time. The main operations involved are solving systems of linear equations and polynomial division. By well-known algorithms, both of these can be solved in polynomial time (in fact, time $O(n^3 \log^2 q)$). With more advanced algebraic algorithms, the running time can even be made $O(n(\log q \cdot \log n)^3)$. This is nearly-linear time, and almost as fast as noiseless polynomial interpolation!

2.1 Correctness of the Algorithm RSDecode

Suppose $p(x)$ is the desired polynomial whose distance from r is at most e. We need to show that the algorithm outputs $p(x)$. We do this through two claims.

- **Claim 1:** In step 1 there does indeed exist a nonzero solution $(a_0, \ldots, a_e, b_0, \ldots, b_{k+e-1}) \in \mathbb{F}_q^{k+2e+1}$.

  Let $Z(x)$ be the error locating polynomial, and let $W(x) = Z(x) \cdot p(x)$. Note that $Z(x)$ is a nonzero polynomial.

  Taking $a_0, \ldots, a_e$ to be the coefficients of $Z(X)$, and taking $b_0, \ldots, b_{k+e-1}$ to be the coefficients of $W(x)$, identity (1) immediately implies that this $(a_0, \ldots, a_e, b_0, \ldots, b_{k+e-1})$ is a nonzero solution to the system of equations (3).

- **Claim 2:** In step 2 the polynomial $A(x)$ does divide $B(x)$, and $B(x)/A(x) = p(x)$.

  Take $A(x), B(x)$ and consider the polynomial $H(x) = B(x) - p(x)A(x) \in \mathbb{F}_q[x]$. We know that for every $u \notin E$, we have

  $$H(u) = B(u) - p(u)A(u) = B(u) - r(u)A(u) = 0,$$

  where the last equality follows from the fact that $(a_0, \ldots, a_e, b_0, \ldots, b_{k+e-1})$ satisfies Equation (3).

  Observe that $H(x)$ is a polynomial of degree at most $k + e - 1$. By the above, we have that $H(u)$ vanishes for at least $n - e > k + e - 1$ values of $u \in S$. By Lemma 1.1, this implies that $H(x)$ is the identically zero polynomial. Thus $B(x) = p(x)A(x)$ and the claim follows.

  It is worth noting that there may be multiple nonzero solutions to the system of equations (3). Claim 2 is about every such nonzero solution!

This completes the analysis of the Berlekamp-Welch algorithm.

Remark 2.2. *Another way of viewing the Berlekamp-Welch algorithm is through the lens of* **rational function interpolation**.

Roughly, we start off trying to find a rational function $B(x)/A(x)$ such that $\deg(A), \deg(B)$ are both small, and $B(u)/A(u) = r(u)$ for all *$u \in S$. We have to be careful about what we mean by division when the denominator is 0.*

Definition 2.3 (Rational function interpolation)**.** *Let $S \subseteq \mathbb{F}_q$ and let $r : S \to \mathbb{F}_q$ be a function. We say that the rational function $B(x)/A(x) \in \mathbb{F}_q(x)$ interpolates r if for every $u \in S$, either:*

- *$A(u) \neq 0$ and $B(u)/A(u) = r(u)$, or*
- *$A(u) = 0$ and $B(u) = 0$.*

The Berlekamp-Welch algorithm is essentially based on the fact that rational interpolation can be efficiently solved using linear algebra, and the analysis basically shows any low degree rational interpolation of r is (after division) the nearby polynomial $p(x)$.

3 List-decoding of Reed-Solomon codes

We now come to the most spectacular application of interpolation ideas: to a more general decoding problem called list-decoding. Just as in the classical decoding problem for a code $\mathcal{C} \subseteq \Sigma^n$, we are given a received string $r \in \Sigma^n$. Now we are also given a radius parameter e, and we would like to find the list $\mathcal{L}$ of all codewords $c \in \mathcal{C}$ such that $\mathsf{dist}(r, c) < e$. In classical decoding, the radius parameter e is always at most half the minimum distance of $\mathcal{C}$ where we are guaranteed that $|\mathcal{L}| \leq 1$. In list-decoding, we can allow a larger radius e. Now the list size $|\mathcal{L}|$ may be larger than 1, but as long as it is not too big, it is reasonable to ask for a fast algorithm that finds $\mathcal{L}$.

In this section we will see how interpolation based ideas, vastly generalizing the ideas in the Berlekamp-Welch algorithm, lead to efficient list-decoding algorithms for Reed-Solomon codes to suprisingly large radii. This is known as the Sudan algorithm.

At the high level, this algorithm has two parts. The first step is interpolation. The goal of the interpolation step is to find a *bivariate polynomial* $Q(x, y)$ such that $Q(u, r(u)) = 0$ for *all* $u \in S$. Pictorially, this finds an algebraic curve (which looks like a union of irreducible algebraic curves) in the $\mathbb{F}_q \times \mathbb{F}_q$ plane that passes through all the points $(u, r(u))$.

For the second step, the key insight is to consider a polynomial $p(x) \in \mathbb{F}_q[x]$ that is close to r, and see how the graph of the relation $y = p(x)$ looks in relation to the above picture. We see that the two graphs $y = p(x)$ and $Q(x, y) = 0$ have many points of intersection. The classical Bezout Theorem implies that low degree curves $P(x, y) = 0$ and $Q(x, y) = 0$ cannot have too many points of intersection unless P and Q have a common factor. Thus[3] in our case, the polynomial $y - p(x)$, and $Q(x, y)$ must have a common factor, and the irreducibility of $y - p(x)$ implies that $y - p(x)$ must be a factor of $Q(x, y)$. The second step of the algorithm is to factor the polynomial $Q(x, y)$ and to thus find $p(x)$.

To gain some insight into this algorithm, we remark that it *is* a generalization of the Berlekamp-Welch algorithm. Indeed, the first step of the Berlekamp-Welch algorithm is to find a polynomial $Q(x, y)$ of the form $A(x)y - B(x)$ such that $Q(u, r(u)) = 0$ for all $u \in S$.

3.1 The Sudan Algorithm

We now give a formal description of (one version of) the Sudan algorithm.
Algorithm RSListDecodeV1
Input: $r : S \to \mathbb{F}_q$, decoding radius e.

1. **Interpolation:** Let $I = \lceil \sqrt{nk} \rceil$ and $J = \lceil \sqrt{\frac{n}{k}} \rceil$. For each i, j with $0 \leq i \leq I$ and $0 \leq j \leq J$, we let a_{ij} be an indeterminate.

   Consider the system of homogenous linear equations in these indeterminates, where for each $u \in S$, we have the equation:

   $$\sum_{i \leq I, j \leq J} a_{ij} u^i (r(u))^j = 0.$$

   Solve this system to find a *nonzero* solution $(a_{ij})_{i \leq I, j \leq J} \in \mathbb{F}_q^{(I+1)(J+1)}$. If there is no nonzero solution, FAIL.

2. **Polynomial Algebra:** Let $Q(x, y) \in \mathbb{F}_q[x, y]$ be the polynomial given by:

   $$Q(x, y) = \sum_{i \leq I, j \leq J} a_{ij} x^i y^j.$$

   Factor $Q(x, y)$ into its irreducible factors over $\mathbb{F}_q$. Let $\mathcal{L}$ be the set of all $p(x)$ for which $y - p(x)$ is an irreducible factor of $Q(x, y)$. These are our candidate codewords. Now output those codewords of $\mathcal{L}$ which are within distance e of the received word r.

[3] When we formally analyze the algorithm, everything will be elementary and we will not use the Bezout Theorem. We mention the Bezout Theorem only to provide motivation.

We will show the following theorem.

Theorem 3.1. *If $e < n - 2\sqrt{nk} - k$, then Algorithm RSListDecodeV1 outputs the list of all codewords in $\mathcal{GRS}(q, k, S)$ that are within Hamming distance e of r. Furthermore, this list has size at most $\lceil \sqrt{\frac{n}{k}} \rceil$.*

To understand how remarkable the above theorem is, consider the setting $k = (0.01)n$ and $e = (0.75)n$. Then the theorem says that Algorithm RSListDecodeV1 can find all codewords within distance $(0.75)n$ of any given received word r. Note that most entries of r may be wrong; yet we can find the corrected codeword!

Before analyzing the correctness of the algorithm, we discuss the running time. Factoring of bivariate polynomials of degree d over $\mathbb{F}_q$ can be done in time $(d \log q)^{O(1)}$ by a randomized algorithm (or $(dq)^{O(1)}$ by a deterministic algorithm). Thus the entire algorithm above can be made to run in polynomial time.

3.2 Correctness of Algorithm RSListDecodeV1

Suppose $p(x)$ is a codeword of $\mathcal{GRS}(q, k, S)$ whose distance from r is at most e. We want to show that $p(x)$ is one of the codewords output by the algorithm.

Let $E \subseteq S$ be the set of $u \in S$ such that $p(u) \neq r(u)$. This is the error set for p. By hypothesis, $|E| \leq e$.

First we show that the interpolation step of the algorithm does not FAIL. In this step, we have to solve a system of homogeneous linear equations to find a nonzero solution. There are n equations in $(I+1)(J+1)$ unknowns. It is well known, despite ample intuition to the contrary, that for general systems of linear equations we cannot deduce anything about the solvability based on the number of equations and the number of unknowns. But for *homogeneous* linear systems we can! If the number of unknowns is greater than the number of equations, there is a nonzero solution.

In our case, the number of unknowns is $(I+1)(J+1)$, which is greater than the number n of equations because:

$$(I+1)(J+1) > \sqrt{nk} \cdot \sqrt{\frac{n}{k}} = n.$$

Thus the first step of the algorithm succeeds in finding a nonzero solution.

Now we examine the second step. By construction, the polynomial $Q(x, y)$ has the property that $Q(u, r(u)) = 0$ for all $u \in S$.

Consider the polynomial $H(x) = Q(x, p(x))$. For any point $u \in S \setminus E$, we have $H(u) = Q(u, p(u)) = Q(u, r(u)) = 0$. Furthermore, the degree of H is at most $I + (k-1)J$.

Thus $H(x)$ is a polynomial of degree most:

$$I + (k-1)J < (\sqrt{nk} + 1) + (k-1)\left(\sqrt{\frac{n}{k}} + 1\right) < 2\sqrt{nk} + k,$$

whose number of roots is at least:

$$|S| - |E| \geq n - e > n - \left(n - 2\sqrt{nk} - k\right) = 2\sqrt{nk} + k.$$

By Lemma 1.1, we conclude that $H(x)$ must be the zero polynomial.

Finally, we use the fact that if $Q(x, y)$ is such that $Q(x, p(x)) = 0$, it means that the bivariate polynomial $y - p(x)$ divides $Q(x, y)$. This is a form[4] of the "factor theorem".

So $y - p(x)$ will appear in the list of factors of Q, and thus $p(x)$ will appear in the output of Algorithm RSListDecodeV1, as desired. The number of such factors is at most the y-degree of Q, which is at most $J = \lceil \sqrt{\frac{n}{k}} \rceil$.

We make some remarks about the argument that we just saw.

1. The precise algebraic problem that we need to solve in the second step of the algorithm is "root finding" rather than factoring. There are faster and simpler algorithms for root finding than for general factoring.

[4] In more detail: write $Q(x, z + p(x))$ as a $h_0(x) + zh_1(x) + z^2h_2(x) + \cdots$. Then $Q(x, p(x)) = 0$ means that $h_0(x) = 0$. Thus z divides $Q(x, z + p(x))$. Setting $y = z + p(x)$, we get the claim.

2. A slightly cleverer choice of monomials[5] used in the polynomial $Q(x, y)$ leads to an improvement of the decoding radius to $n - \sqrt{2nk}$. This is the error-correction performance of the original Sudan algorithm. This is larger than half the minimum distance for all $k < n/3$.

3. The significance of the Sudan algorithm is that it showed for the first time that it is possible to efficiently list decode positive rate codes beyond what is possible for unique decoding. For a code of rate R, the classical Singleton bound implies that it is not possible to unique-decode from more than $((1-R)/2)$-fraction errors. We just saw that Reed-Solomon codes of rate R can be efficiently decoded from $(1 - 2\sqrt{R} - R)$-fraction errors, which for small R is larger than $(1-R)/2$.

4 List-Decoding of Reed-Solomon codes using multiplicities

Now we come to a powerful new tool that greatly strengthens the reach of the interpolation method: multiplicities. We will see the algorithm of Guruswami and Sudan for list-decoding Reed-Solomon codes from $n - \sqrt{nk}$ errors. This is larger than half the minimum distance for *all* k. It correspondingly shows that it is possible to have codes of rate R for which efficient list-decoding from $(1 - \sqrt{R})$-fraction errors is possible – this is larger than the unique-decoding limit of $(1-R)/2$ for all R.

We begin with a definition of *multiplicity of vanishing*.

Definition 4.1. *Over a field $\mathbb{F}$, let $Q(x_1, \ldots, x_m) \in \mathbb{F}[x_1, \ldots, x_m]$ be a polynomial. Let $\mathbf{u} = (u_1, \ldots, u_m) \in \mathbb{F}^m$ be a point. We define the* **multiplicity of vanishing** *of Q at $\mathbf{u}$, denoted $\mathsf{mult}(Q, \mathbf{u})$, to be the smallest integer M such that the polynomial $Q(\mathbf{u} + \mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_m]$ has no monomials $\prod_{j=1}^{m} x_j^{b_j}$ of degree $< M$.*

If Q is the zero polynomial, we define $\mathsf{mult}(Q, \mathbf{u}) = -\infty$ by convention.

Classically, over fields of characteristic 0 multiplicity is defined using derivatives. Over finite fields one has to be careful. Everything works well if we use the notion of Hasse derivative. See [HKT08] for more on this.

Example 4.2. *The following multiplicity calculations are easy to check.*

- $\mathsf{mult}((x-1)^2(3+2x), 1) = 2$.
- $\mathsf{mult}(y^2 - x^3 - x^4, (0,0)) = 2$.

The Guruswami-Sudan algorithm for list-decoding Reed-Solomon codes is based on interpolating a bivariate polynomial $Q(x, y)$ that vanishes at each point $(u, r(u))$ for $u \in S$ *with high multiplicity*. Asking that a polynomial vanishes somewhere with high multiplicity imposes more linear constraints on the coefficients of Q than simply asking that Q vanishes there. To accomodate this, we need to enlarge the space where we search for Q, and thus we end up with such a Q of higher degree than before.

We then benefit from the increased vanishing multiplicity to deduce that for any p that is near the received word r, the univariate polynomial $H(x) = Q(x, p(x))$ vanishes *with high multiplicity* at many points, and thus must be identically 0. This lets us recover $p(x)$ by factoring Q, as before. A-priori it is not clear that there will be an improvement in the decoding radius: the larger degree of Q is traded off against the increased vanishing multiplicity. Nevertheless there is an improvement. We discuss the philosphical reason to expect an improvement later in this section.

4.1 Preparations

We state below some simple properties of multiplicity. We omit the (easy) proofs.

Lemma 4.3. *Suppose $H(x) \in \mathbb{F}[x]$ and $u \in \mathbb{F}$ are such that $\mathsf{mult}(H, u) \geq M$. Then*

$$(x-u)^M \mid H(x).$$

[5] We will see this cleverer choice when we discuss the Guruswami-Sudan algorithm.

As an immediate consequence of the previous lemma, we get the multiplicity analogue of Lemma 1.1.

Lemma 4.4. *Let $H(x) \in \mathbb{F}[x]$ be a nonzero polynomial of degree at most d. Then $\sum_{u \in \mathbb{F}} \mathsf{mult}(H, u) \leq d$.*

Lemma 4.5. *Suppose $Q(x, y) \in \mathbb{F}[x, y]$ and $p(x) \in \mathbb{F}[x]$. Suppose $u \in \mathbb{F}$ is such that $\mathsf{mult}(Q, (u, p(u))) \geq M$. Then, letting $H(x) = Q(x, p(x))$, we have $\mathsf{mult}(H, u) \geq M$.*

We will be dealing with certain weighted degrees of bivariate polynomials. For a monomial $x^i y^j$, its (α, β)-**weighted degree** is defined to be $\alpha i + \beta j$.

4.2 The Guruswami-Sudan Algorithm

With these lemmas in hand we can now formally describe the Guruswami-Sudan algorithm.
Algorithm RSListDecodeV2
Input: $r : S \to \mathbb{F}_q$, decoding radius e.

1. **Interpolation:** Let $M = k$. Let $D = M\sqrt{nk}$.

   We let $\mathcal{L}_D$ be the set of all monomials $x^i y^j$ for which the $(1, k-1)$-weighted degree is at most D. Thus:

   $$\mathcal{L}_D = \{x^i y^j \mid i + (k-1)j \leq D\}.$$

   We also consider the corresponding sets of exponents.

   $$T_D = \{(i, j) \mid i, j \geq 0, i + (k-1)j \leq D\}.$$

   For each $(i, j) \in T_D$, we let a_{ij} be an indeterminate.

   Let $Q(x, y) \in \mathbb{F}_q[x, y]$ be the polynomial of $(1, k-1)$-weighted degree at most D whose coefficients are a_{ij}:

   $$Q(x, y) = \sum_{(i,j) \in T_D} a_{ij} x^i y^j.$$

   Consider the system of homogeneous linear equations on the a_{ij} by imposing the condtion, for each $u \in S$:

   $$\mathsf{mult}(Q, (u, r(u))) \geq M.$$

   Find a nonzero solution $(a_{ij}) \in \mathbb{F}_q^{T_D}$. If no nonzero solution exists, FAIL.

2. **Polynomial Algebra:** Factor $Q(x, y)$ into its irreducible factors over $\mathbb{F}_q$. Let $\mathcal{L}$ be the set of all $p(x)$ for which $y - p(x)$ is an irreducible factor of $Q(x, y)$. These are our candidate codewords. Now output those codewords of $\mathcal{L}$ which are within distance e of the received word r.

We will show the following theorem.

Theorem 4.6. *If $e < n - \sqrt{nk}$, then Algorithm RSListDecodeV2 outputs the list of all codewords in $\mathcal{GRS}(q, k, S)$ that are within Hamming distance e of r. Furthermore, this list has size at most $2\sqrt{nk}$.*

4.3 Correctness of the Algorithm RSListDecodeV2

Suppose $p(x)$ is a codeword of $\mathcal{GRS}(q, k, S)$ whose distance from r is at most e. We want to show that $p(x)$ is one of the codewords output by the algorithm.

Let $E \subseteq S$ be the set of $u \in S$ such that $p(u) \neq r(u)$. This is the error set for p. By hypothesis, $|E| \leq e$.

First we look at the interpolation step. Since it is a system of *homogeneous* linear equations, we can show existence of a nonzero solution by counting equations and unknowns. The total number of unknowns is $|T_D|$, which can be lower bounded by:

$$|T_D| = \sum_{j \leq D/(k-1)} ((D+1) - (k-1)j) \geq \frac{D^2}{2(k-1)}.$$

How many equations are there? Asking that Q vanishes at a point $(u, r(u))$ with multiplicity at least M is the same as asking that $\binom{M+1}{2}$ coefficients of the polynomial $Q(x+u, y+r(u))$ vanish. Thus the total number of equations in the system is $n \cdot \binom{M+1}{2}$.

We now check that the number of unknowns is larger than the number of constraints holds.

$$\begin{aligned}\frac{D^2}{2(k-1)} &= \frac{nkM^2}{2(k-1)} \\ &= n \cdot \frac{M^2}{2} \cdot \frac{k}{k-1} \\ &\geq n \cdot \frac{M(M+1)}{2},\end{aligned}$$

where the last inequality uses the fact that $\frac{M+1}{M} < \frac{k}{k-1}$, since $M = k$.

Thus the system of equations has a nonzero solution, and the first step of the algorithm succeeds.

Now we consider the second step. By construction, the polynomial $Q(x, y)$ has the property that $\mathsf{mult}(Q, (u, r(u))) \geq M$ for all $u \in S$.

Consider the polynomial $H(x) = Q(x, p(x))$. For any point $u \in S \setminus E$, we have that $p(u) = r(u)$, and thus by Lemma 4.5, $\mathsf{mult}(H, u) \geq M$. Furthermore, the degree of H is at most D.

Thus $H(x)$ is a polynomial of degree at most D whose total multiplicity of vanishing at all points in S is at least:

$$(|S| - |E|) \cdot M \geq (n - e) \cdot M > (n - (n - \sqrt{nk})) \cdot M = \sqrt{nk} \cdot M = D.$$

By Lemma 4.4, we conclude that $H(x)$ must be the zero polynomial.

Again, since $Q(x, p(x)) = H(x) = 0$, we get that $y - p(x)$ divides $Q(x, y)$. So $y - p(x)$ will appear in the list of factors of Q, and thus $p(x)$ will appear in the output of Algorithm RSListDecodeV2, as desired. The number of such factors is at most the y-degree of Q, which is at most $\frac{D}{k-1} \leq \frac{k\sqrt{nk}}{k-1} \leq 2\sqrt{nk}$.

4.4 Why do multiplicities help?

The role of multiplicities in the previous argument seems mysterious. Why did multiplicities improve the decoding radius of the Sudan algorithm?

In the univariate case, if we have a set $T \subseteq \mathbb{F}_q$, then we have the simple fact that any polynomial $Q(x)$ that vanishes at all points of T must be a multiple of $Z_T(x) = \prod_{u \in T}(x - u)$. Analogously, any polynomial $Q(x)$ that vanishes with multiplicity M at each point of T must be a multiple of $Z_T(x)^M$. Informally, this means that in the univariate case, we don't get more information by asking Q to vanish with high multiplicity at the points of a set.

In the multivariate case, there is no similar tight connection between simply vanishing at all points of a set $T \subseteq \mathbb{F}_q^m$ and vanishing with high multiplicity at all points of T.

The following facts shed further light on this phenomenon. If we take a "typical" polynomial $Q(x, y) \in \mathbb{F}_q[x, y]$ of low degree, it will vanish at approximately q points in $\mathbb{F}_q \times \mathbb{F}_q$. On the other hand, a typical Q will vanish with multiplicity ≥ 2 at only $O(1)$ points of $\mathbb{F}_q \times \mathbb{F}_q$. This means that a polynomial that vanishes with high multiplicity at all the points of a set of interest is a very special polynomial, and presumably its fate is more strongly tied to that of the set.

There have been many successful uses of the idea of interpolation in several areas of mathematics, and taking multiplicities into account is often useful [Sch09, Sch06, DKSS09].

5 Decoding of interleaved Reed-Solomon codes under random error

We give one final example of the power of interpolation ideas for decoding algebraic codes. This example concerns *interleaved Reed-Solomon codes*, which we now define.

Let q be a prime power, $S \subseteq \mathbb{F}_q$, and let $k \geq 0$ and $s \geq 1$ be integers. We define the **interleaved Reed-Solomon code** $\mathcal{IRS}[q, k, S, s]$ as follows. The alphabet Σ of this code equals $\mathbb{F}_q^s$, and the coordinates of each codeword are indexed by S (thus the blocklength equals $|S|$). For each s-tuple of polynomials $(p_1(x), \ldots, p_s(x)) \in (\mathbb{F}_q[x])^s$ with $\deg(p_i) < k$ for each i, there is a codeword:

$$((p_1(\alpha), p_2(\alpha), \ldots, p_s(\alpha)) \mid \alpha \in S) \in \Sigma^S.$$

Another way to view this is as follows. To get codewords of the interleaved Reed-Solomon code $\mathcal{IRS}[q, k, S, s]$, we take a $s \times |S|$ matrix whose rows are codewords of $\mathcal{GRS}[q, k, S, s]$, and view each column of the matrix as a single symbol in $\Sigma = \mathbb{F}_q^s$. The strings in $\Sigma^{|S|}$ so obtained are the codwords of $\mathcal{IRS}[q, k, S, s]$.

Clearly, interleaved Reed-Solomon codes are very closely related to Reed-Solomon codes. It is easy to see that the rate of $\mathcal{IRS}[q, k, S, s]$ equals $k/|S|$ and the minimum distance of $\mathcal{IRS}[q, k, S, s]$ equals $|S| - k + 1$ (exactly as in the case of Reed-Solomon codes).

We now turn our attention to decoding algorithms. The problem of decoding interleaved Reed-Solomon codes from e errors is the following natural-looking question. Let $r : S \to \mathbb{F}_q^s$ be the "received word". We will sometimes think of r as a tuple of s functions $r^{(1)}, r^{(2)}, \ldots, r^{(s)}$, where $r^{(i)} : S \to \mathbb{F}_q$ is the ith output coordinate of r. Our goal is to find one/many/all tuples of polynomials $p = (p_1, \ldots, p_s) \in (\mathbb{F}_q[x])^s$ with $\deg(p_i) < k$ such that $r(u) = (p_1(u), \ldots, p_s(u))$ for all but at most e values of $u \in S$ (in this case we write $\mathsf{dist}(p, r) \leq e$.

Clearly, if $\mathsf{dist}(p, r) \leq e$, then we have that for all i, $\mathsf{dist}(p_i, r^{(i)}) \leq e$. Note that the converse does not hold: the sets of coordinates $u \in S$ where p_i and $r^{(i)}$ agree can look very different for different i. When the e is at most $(|S| - k)/2$ (half the minimum distance of the code), then the above observation gives an easy algorithm to decode interleaved Reed-Solomon codes from adversarial errors: for each i, decode $r^{(i)}$ (using a standard Reed-Solomon decoder) to find the unique polynomial p_i of degree $< k$ that is within distance e of it. Taking these p_i together to form a tuple p, we get a candidate codeword p of $\mathcal{IRS}$, and then we check that $\mathsf{dist}(p, r) \leq e$. [6]

We will now describe an algorithm that can decode from a signficantly larger number of *random* errors. Specifically, the model for generating r is as follows. There is some unknown tuple of polynomials $p = (p_1, \ldots, p_s) \in (\mathbb{F}_q[x])^s$ where $\deg(p_i) < k$. There is an unknown set $J \subseteq S$ (the set of error locations) of size at most e. Based on these unknowns, the received word $r : S \to \mathbb{F}_q^s$ is generated by setting:

$$r(u) = \begin{cases} (p_1(u), p_2(u), \ldots, p_s(u)) & u \notin J \\ \text{a uniformly random element of } \mathbb{F}_q^s & u \in J \end{cases}.$$

Note that the number of errors (i.e., the number of $u \in S$ for which $p(u) \neq r(u)$) is always at most $|J|$, and is with high probability equal to $|J|$, but it could be smaller.

In this setting, we would like to design a decoding algorithm that takes r as input and recovers p, the underlying codeword. Below we give a natural (given the previous sections) algorithm for this due to Bleichenbacher, Kiayias and Yung. A clever and nontrivial analysis shows that this algorithm succeeds with high probability even when the number of errors e which is as large as $\frac{s}{s+1} \cdot (n - k)$, which for large s is close to the minimum distance of the code (*twice the worst-case unique decoding radius*!). This result is very surprising.

The key idea is to run a Berlekamp-Welch type decoding algorithm for each of the $r^{(i)}$, while taking into account the fact that *the error locations, and hence the error-locating polynomials, are the same.*

Algorithm IRSDecode
Input: $r : S \to \mathbb{F}_q^s$.

1. **Interpolation:** Let $a_0, a_1, \ldots, a_e$ be indeterminates and for each $\ell \in \{1, \ldots, s\}$, let $b_{\ell,0}, b_{\ell,1}, \ldots, b_{\ell,k+e-1}$ be indeterminates. Consider the following system of homogeneous linear equations in these indetermi-

[6] A more sophisticated variation using the Guruswami-Sudan list-decoding algorithm can extend the above algorithm to decode interleaved Reed-Solomon codes from any $e \leq n - \sqrt{nk}$ errors. Further improvements seem difficult: decoding from an even larger number of errors in the worst case would need a breakthrough on decoding of standard Reed-Solomon codes.

nates, one equation for each $(u, \ell) \in S \times \{1, \ldots, s\}$:

$$\left(\sum_{i=0}^{e} a_i u^i\right) \cdot r^{(\ell)}(u) = \sum_{j=0}^{k+e-1} b_{\ell,j} u^j. \tag{3}$$

Solve the system to find any *nonzero* solution $(a_0, \ldots, a_e, b_{1,0}, \ldots, b_{s,k+e-1}) \in \mathbb{F}_q^{e+s(k+e)}$. If there is no nonzer solution, then FAIL.

2. **Polynomial Algebra:** For the solution found in the previous step, define $A(x), B_1(x), B_2(x), \ldots, B_s(x) \in \mathbb{F}_q[x]$ by:

$$A(x) = \sum_{i=0}^{e} a_i x^i,$$

$$B_\ell(x) = \sum_{j=0}^{k+e-1} b_{\ell,j} x^j.$$

If $A(x)$ divides $B_\ell(x)$ for each ℓ, then return the tuple

$$(B_1(x)/A(x), B_2(x)/A(x), \ldots, B_s(x)/A(x))$$

(otherwise FAIL).

The correctness guarantee of the above algorithm is given by the following theorem.

Theorem 5.1. *Suppose $e < \frac{s}{s+1} \cdot (n-k)$, and let $J \subseteq S$ be of size at most e. Suppose $p = (p_1(x), \ldots, p_s(x)) \in (\mathbb{F}_q[x])^s$ is such that $\deg(p_i) < k$. Finally, let r be generated based on p and J by the random process described above.*

Then with probability at least $1 - \frac{n}{q}$, Algorithm IRSDecode on input r will return p.

A proof of this theorem is a bit too involved to give here. We instead just give a high-level overview of what we expect will happen.

Our hope is that $A(x)$ ends up equalling the error-locating polynomial for this setting, namely:

$$Z(x) = \prod_{\alpha \in J} (x - \alpha).$$

This error-locating polynomial has degree at most e. Consider also the polynomial $W_\ell(x) = Z(x) \cdot p_\ell(x)$, which has degree $< k + e$. We clearly have:

$$Z(x) r^{(i)}(x) = W_i(x)$$

for each $x \in S$. Thus $(Z(x), W_1(x), W_2(x), \ldots, W_s(x))$ is a valid solution to the system of linear equations above, and then the correct solution $p_i(x) = W_i(x)/Z(x)$ does get returned.

There may be other valid solutions to the system of linear equations. For example, if the error set J has size $< e$, then $A(x) = Z(x) \cdot (X + 1)$ and $B_i(x) = W_i(x) \cdot (X + 1)$ is also a valid solution, but the final output of the algorithm is still the same.

The actual proof of Bleichenbacher, Kiayias and Yung is very interesting, and shows that these are the only possibilities with high probability. It is is based on treating the random variables $(r^{(i)}(u))_{u \in J}$ as formal variables, and careful studying the multivariate polynomials that arise as subdeterminants of the matrix underlying the system of linear equations. Most crucially, one needs to identify appropriate Vandermonde matrices inside this matrix, and then apply the Schwartz-Zippel lemma (which says that nonzero multivariate polynomials evaluate to nonzero at a random point with high probability).

6 Further reading

Algorithms for decoding interleaved Reed-Solomon codes from random error were given by Bleichenbacher, Kiayas and Yung [BKY07] (the algorithm we saw here) and Coppersmith and Sudan [CS03] (a different but related algorithm).

There have been many advances on the list-decoding of error-correcting codes (and list-decoding of algebraic codes in particular) in recent years. The most important result is the construction and decoding of "capacity achieving" list-decodable codes by Guruswami and Rudra [GR08] (based on a breakthrough by Parvaresh and Vardy [PV05]). These codes, called Folded Reed-Solomon codes, are a variation on Reed-Solomon codes, and achieve the optimal tradeoff between rate and number of errors correctable by list-decoding with polynomially bounded list size.

Another family of algebraic codes called Multiplicity Codes (based on evaluating polynomials and their derivatives), were also shown to achieve list-decoding capacity by Guruswami and Wang [GW13] and Kopparty [Kop12].

All these list decoding algorith:ms use extensions of the interpolation decoding technique using high-variate interpolation. Very recently, Kopparty, Ron-Zewi, Saraf and Wootters [KRZSW18] showed that both Folded Reed-Solomon codes and Multiplicity codes achieve list-decoding capacity with constant list size.

The list-decodability of Reed-Solomon codes themselves is still open. For all we know, Reed-Solomon codes themselves may be list-decodable upto list decoding capacity: this corresponds to list-decoding from $O(k)$ agreements (instead of the $\sqrt{nk}$ agreements that the Guruswami-Sudan algorithm needs). The best negative result in this direction is by Ben-Sasson, Kopparty and Radhakrishnan [BSKR09], who show that the list size may become superpolynomial for Reed-Solomon codes of vanishing rate.

Interpolation methods have a long history in mathematics. Some striking classical applications include the Thue-Siegel-Roth theorems on diophantine approximation, the Gelfond-Schneider-Baker theorems on transcendental numbers [Bak90], and the Stepanov-Bombieri-Schmidt proofs of the Weil bounds [Sch06]. More recent applications include the results of Dvir, Lev, Kopparty, Saraf and Sudan [Dvi09, SS08, DKSS09, KLSS11] on the Kakeya problems over finite fields, and the results of Guth and Katz [GK10, GK15] in combinatorial geometry, and the results of Croot, Lev, Pach, Ellenberg and Giswijit on the cap set problem [CLP17, EG17].

References

[Bak90] Alan Baker. *Transcendental number theory*. Cambridge university press, 1990.

[BKY07] Daniel Bleichenbacher, Aggelos Kiayias, and Moti Yung. Decoding interleaved reed-solomon codes over noisy channels. *Theor. Comput. Sci*, 379(3):348–360, 2007.

[BSKR09] Eli Ben-Sasson, Swastik Kopparty, and Jaikumar Radhakrishnan. Subspace polynomials and limits to list decoding of reed–solomon codes. *IEEE Transactions on Information Theory*, 56(1):113–120, 2009.

[BW] E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470.

[CLP17] Ernie Croot, Vsevolod F Lev, and Péter Pál Pach. Progression-free sets in are exponentially small. *Annals of Mathematics*, pages 331–337, 2017.

[CS03] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional spaces from noisy data. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 136–142, 2003.

[DKSS09] Z. Dvir, S. Kopparty, S. Saraf, and M. Sudan. Extensions to the method of multiplicities, with applications to Kakeya sets and mergers. In *50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 181–190, 2009.

[Dvi09] Zeev Dvir. On the size of kakeya sets in finite fields. *Journal of the American Mathematical Society*, 22(4):1093–1097, 2009.

[EG17] Jordan S Ellenberg and Dion Gijswijt. On large subsets of with no three-term arithmetic progression. *Annals of Mathematics*, pages 339–343, 2017.

[GK10] L. Guth and N. H. Katz. Algebraic methods in discrete analogs of the kakeya problem. *Advances in Mathematics*, 225(5):2828–2839, 2010.

[GK15] Larry Guth and Nets Hawk Katz. On the erdős distinct distances problem in the plane. *Annals of mathematics*, pages 155–190, 2015.

[GR08] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of reed-solomon codes. *IEEE Trans. Information Theory*, 59(6):3257–3268, 2013.

[HKT08] J. W. P. Hirschfeld, G. Korchmaros, and F. Torres. *Algebraic Curves over a Finite Field (Princeton Series in Applied Mathematics)*. Princeton University Press, 2008.

[KLSS11] Swastik Kopparty, Vsevolod F Lev, Shubhangi Saraf, and Madhu Sudan. Kakeya-type sets in finite vector spaces. *Journal of Algebraic Combinatorics*, 34(3):337–355, 2011.

[Kop12] S. Kopparty. List-decoding multiplicity codes. In *Electronic Colloquium on Computational Complexity (ECCC)*, TR12-044, 2012.

[KRZSW18] Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded reed-solomon and multiplicity codes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 212–223. IEEE, 2018.

[Pet60] Wesley Peterson. Encoding and error-correction procedures for the bose-chaudhuri codes. *IRE Transactions on Information Theory*, 6(4):459–470, 1960.

[PV05] F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 285–294, 2005.

[Sch06] Wolfgang M Schmidt. *Equations over finite fields: an elementary approach*, volume 536. Springer, 2006.

[Sch09] WM Schmidt. *Diophantine Approximation*, volume 785. Springer, 2009.

[SS08] Shubhangi Saraf and Madhu Sudan. Improved lower bound on the size of Kakeya sets over finite fields. *Analysis and PDE*, 2008.

[Sud97] M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.