Beyond the Guruswami-Sudan (and Parvaresh–Vardy) Radii: Folded Reed-Solomon, Multiplicity and Derivative Codes

Neophytos Charalambides

December 10, 2019

Abstract

The classical family of Reed-Solomon codes consist of evaluations of polynomials over the finite field $\mathbb{F}_q$ of degree less than $k$, at $n$ distinct field elements. These are arguably the most widely used and studied codes, as they have both erasure and error-correction capabilities, among many others nice properties. In this survey we study closely related codes, folded Reed-Solomon codes, which are the first constructive codes to achieve the list decoding capacity. We then study two more codes which also have this feature, multiplicity codes and derivative codes. Our focus for the most part are the list decoding algorithms of these codes, though we also look into the local decodability of multiplicity codes.

1 Introduction

Communicating information is ubiquitous in modern technologies and every day interactions between people and corporations. Communication is achieved by encoding a message of length $k$ to a codeword of length $n$ over alphabets, which is sent though a channel. The codeword may be corrupted in a subset of up to $\delta = pn$ symbols, for some $p \in (0, 1)$. The purpose of encoding the original message is to have reliable communication over the channel, which means that the fraction of corrupted symbols may be restored; in order to retrieve the original message. A code $C$ over an alphabet $\Sigma$ is a structured subset of $\Sigma^n$ for which such recoveries are possible, as long as there are no more than $pn$ corrupted symbols. The rate of $C$ is defined as $R = \frac{\log |C|}{n \log |\Sigma|} = \frac{k}{n}$.

A basic trade-off in this setting, is the one between rate $R$ and error fraction $p$; or equivalently between $R$ and the relative distance $\delta$. Clearly, $R \leq 1 - p$. If we relax the decoding we require from unique, to listing a set of codewords which contain the correct codeword, this rate is asymptotically met. That is, there exist codes of rate $R = 1 - p - o(1)$ which are $p$-list-decodable. We refer to $1 - R$ as the list decoding capacity, which coincides with the fraction of errors we can correct, and is the optimal limit. Surprisingly, this is twice the fraction of errors that one could decode when requiring unique decoding [G+07]! Though the above argument is non-constructive, folded Reed-Solomon codes achieve list decoding from an error rate approaching $1 - R$, with a polynomial time decoding algorithm [GR08]. We present these codes in §2, along with the original ideas and results from [GR08]. We then describe two more completely different list decoding procedures for these codes, a linear-algebraic approach in §3, and one based on Hensel-lifting §4.

In §5 we shift our focus to study another recent family of codes, multiplicity codes [KSY10]. Multiplicity error-correcting codes are locally decodable codes which have efficient local decoding
algorithms, with rate approaching 1 and a low number of queries. They are based on evaluating multivariate polynomials and their derivatives. Finally, in §6 we delve into a closely related family of codes, derivative codes. These are simpler and more natural codes which relate to folded Reed-Solomon codes, putting together a lot of the ideas we will see throughout this survey.

## 2 Folded Reed-Solomon Codes

Recall that a Reed-Solomon code $RS_q[n,k]$ (RS) over $\mathbb{F}_q$, is the encoding of polynomials of degree at most $k-1$ which represents our message, over the defining set of points $A = \{\alpha_1, \cdots, \alpha_n\} \subset \mathbb{F}_q$,

$$RS_q[n,k] = \{ \{f(\alpha_1), f(\alpha_2), \cdots, f(\alpha_n)\} \mid f(X) \in \mathbb{F}_q[X] \text{ of degree } \leq k-1 \}$$

Typically $n = |\mathbb{F}_q| - 1 = q - 1$ and $\alpha_i = \alpha^i$ for all $i \in \mathbb{N}_n := \{1, 2, \cdots, n\}$, where $\alpha$ is some primitive element in $\mathbb{F}_q$. The encoding of the message $m = [m_0, \cdots, m_{k-1}] \mapsto m(X) = \sum_{i=0}^{k-1} m_i X^i \in \mathbb{F}_q[X]$ is defined by the evaluation mapping

$$\text{Enc}(m) = [m(\alpha_1), m(\alpha_2), \cdots, m(\alpha_n)] \in \mathbb{F}_q.$$

One difficulty with RS codes is that we need to be able to correct any pattern of $\lfloor \frac{n-k+1}{2} \rfloor$. Guruswami and Rudra [GR08] address this problem, by “bundling” parts of the codewords together, which considerably decreases the error pattern we have to handle. What they define as folded Reed-Solomon codes (FRS), are in fact exactly RS codes, but viewed as a code over a larger alphabet by careful bundling of codeword symbols. Informally, an $m$-FRS code is a RS code over $\mathbb{F}_q$, where $m$ consecutive positions in the RS code are identified with an element in $\mathbb{F}_q^m$. That is, the columns of the encoded matrix (2.1) may each be considered as an element in $\mathbb{F}_q^m$ — folding a vector in $\mathbb{F}_q^m$ to an element in $\mathbb{F}_q$. We point out that the term folded Reed-Solomon was first introduced in [Kra03] to correct burst errors, though the folding operation is slightly different to what we are considering.

**Definition 2.1.** Consider $\Sigma = \mathbb{F}_q$, its nonzero elements $\{1, \gamma, \cdots, \gamma^{n-1}\}$, for $n = q - 1$, and $\gamma$ a primitive element. Let $m$ be a positive factor of $n$; i.e. $n = N \cdot m$, and degree parameter $k \in \mathbb{N}_n$. The $m$-folded Reed-Solomon code $FRS_q(m)[k]$, is a code over $\mathbb{F}_q^m \cong \mathbb{F}_q$, that encodes a polynomial $f(X) \in \mathbb{F}_q[X]$ of degree $k - 1$,

$$\text{Enc}(f(X)) = \begin{bmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{m-1}) \end{bmatrix}, \begin{bmatrix} f(\gamma^m) \\ f(\gamma^{m+1}) \\ \vdots \\ f(\gamma^{2m-1}) \end{bmatrix}, \cdots, \begin{bmatrix} f(\gamma^{n-m}) \\ f(\gamma^{n-m+1}) \\ \vdots \\ f(\gamma^{n-1}) \end{bmatrix} \cong \begin{bmatrix} f(1) & \cdots & f(n-m) \\ f(\gamma) & \cdots & f(\gamma^{n-m+1}) \\ \vdots & \ddots & \vdots \\ f(\gamma^{m-1}) & \cdots & f(\gamma^{n-1}) \end{bmatrix}. \quad (2.1)$$

**Proposition 2.2.** The FRS (nonlinear) code over $\mathbb{F}_q^m$ defined above, has block length $N$, rate $R = \frac{k}{n} = \frac{k}{Nm}$, and minimum distance $d_{\text{min}} = N - \lfloor k/m \rfloor + 1 \simeq (1 - R)N$.

Suppose a FRS codeword was transmitted, and a received (potentially corrupted) string

$$y = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mN} \end{pmatrix} \in \mathbb{F}_q^{m \times N} \quad (2.2)$$
$y \in (\mathbb{F}_q^n)^N$ was received, which we view as a matrix in $\mathbb{F}_q^{n \times N}$. The goal is to recover a list of all polynomials in $\mathbb{F}_q[X]$ of degree at most $k - 1$, whose encoding (2.1) agrees with $y$ in at least $t$ columns, for some agreement parameter $t$. Ideally, we would like $t$ to be as small as possible, as this corresponds to list decoding up to $n - t$ errors. We know how to list decode a RS code up to $1 - \sqrt{R}$ [GS98] in poly$(n,q)$ time, so by simply unfolding and treating it as a regular RS codes, we can solve this for $t \geq \sqrt{R}N$. A crucial difference in FRS is its additional structure: when a column is correct, we know the correct values of all $m$ values in the column.

Decoding these codes is similar in spirit to list decoding of RS. The gain comes from interpolating in more than two dimensions. That is, we seek a higher dimensional analog of the identity $Q(X, f(X)) = 0$ from the RS case, a low-degree nonzero polynomial which is interpolated through the data. The essence is to argue that this identity suffices to retrieve a small list of possibilities efficiently. The main steps in this higher dimensional version of the Berlekamp-Welch algorithm [BW86], are interpolation and root-finding. We now present a fundamental result of [GR08].

**Theorem 2.3** ([Gur11]). For every integer $s, 1 \leq s \leq m$ and any constant $\delta > 0$, there is a list decoding algorithm for $\text{FRS}_q^{(m)}[n, k]$ that list decodes from up to $e$ errors, as long as

$$e \leq N - (1 + \delta) \frac{(k^s N (m - s + 1))^{1/(s+1)}}{m - s + 1}$$

where $N = \frac{n}{m}$ is the code block length. The algorithm runs in $(O_\delta(q))^{O(s)}$ time, and outputs a list of size at most $q^{s-1}$.

The fraction of errors corrected by this algorithm as a function of the rate $R$ is

$$1 - (1 + \delta) \left( \frac{mR}{m - s + 1} \right)^{s/(s+1)} \simeq 1 - (1 - \varepsilon) \left( \frac{R}{1 - \varepsilon + \varepsilon^2} \right)^{1/(1+\varepsilon)} \geq 1 - R - \varepsilon$$

(2.3)

where in $\simeq$ we pick $\delta \simeq \varepsilon$, $s \simeq 1/\varepsilon$ and $m \simeq s^2$. The decoding complexity and list-size are $\simeq q^{O(1/\varepsilon)}$.

Another important thing to note here is that we consider a fraction $\simeq 1 - \left( \frac{mR}{m - s + 1} \right)^{s/(s+1)}$ of errors, where for large enough $m$ we get $\simeq 1 - \sqrt{s} \sqrt{R}$. The second term is precisely the geometric mean of $1, R, \cdots, R$. This is analogous to the improvement achieved in RS codes ($s = 1$), where the agreement required between the received string and codeword was reduced from $\frac{1+R}{2}$ (arithmetic mean) to $\sqrt{R}$ (geometric mean). The corresponding radius was consequently improved from $\frac{1-R}{2}$ to $1 - \sqrt{R}$, which is always better by the AM-GM inequality.

The main gain of the folding operation on RS codes, is that we can construct list decodable FRS codes up to radius roughly $1 - \sqrt{s} \sqrt{R}$, for any $s \in \mathbb{Z}_+$. By selecting $s$ large enough, we can get within any desired $\varepsilon$ from capacity, attaining list decodability up to fraction $1 - R - \varepsilon$ of errors. Moreover, list decoding capacity was achieved over large alphabets [GR08], as $\lim_{s \to \infty} \left\{ 1 - \sqrt{s} \sqrt{R^s} \right\} = 1 - R$, though there was room for improvement with respect to some of the parameters.

### 2.1 Interpolation step

The list decoding algorithm ([Gur10],[V+12],[GRS19] — modified version of the original algorithm) first interpolates a linear polynomial $Q(X,Y) = Q(X,Y_1,\cdots,Y_s)$ of degree 1 in the $Y_i$'s through
certain \((s + 1)\)-tuples, where \(Y\) denotes the formal variables \(Y_1, \cdots, Y_s\). Given a \(y \in \mathbb{F}_q^{m \times N}\), we interpolate a nonzero polynomial

\[
Q(X, Y) = A_0(X) + A_1(X)Y_1 + \cdots + A_s(X)Y_s = A_0(X) + \sum_{i=1}^{s} A_i(X)Y_i \tag{2.4}
\]

where \(\deg(A_i) \leq d\) for all \(i \in \mathbb{N}_s\) and \(\deg(A_0) \leq d + k - 1\), for a suitable degree parameter. The total number of monomials which appear in \(Q\) with these restrictions is

\[(d + 1)s + d + k = (d + 1)(s + 1) + k - 1 \geq N(m - s + 1) + s + 1 > N(m - s + 1)\]

for \(d\) chosen to be

\[d = \left\lceil \frac{N(m - s + 1) - k + 1}{s + 1} \right\rceil \tag{2.5}\]

and \(Q \in \mathbb{F}_q[X][Y] \cong \mathbb{F}_q[X, Y] = \mathbb{F}_q[X, Y_1, \cdots, Y_s]\) must satisfy the interpolation step

\[Q \left( \gamma^{im+j}, y_{im+j}, y_{im+(j+1)}, \cdots, y_{im+(j+s-1)} \right) = 0 \tag{2.6}\]

for all \(i = 0, 1, \cdots, N - 1, \ j = 0, \cdots, m - s\), which may be viewed as \(N(m + s - 1)\) constraints. Since we have more monomials than constraints, such a nonzero polynomial \(Q\) exists, which can be found by solving a homogeneous linear system. This explains our choice of \(d\). The following lemma gives a necessary algebraic condition which message polynomials \(f(X)\) in our desired list must satisfy.

**Lemma 2.4.** If \(f(X) \in \mathbb{F}_q[X]\) is a polynomial of degree at most \(k - 1\) whose FRS encoding (2.1) agrees with \(y\) in at least \(t\) columns for \(t \geq d + k\), and \(s = m\), then

\[Q(X, f(X), f(\gamma X), \cdots, f(\gamma^{s-1} X)) = 0. \tag{2.7}\]

We refer to the polynomials \(f(X) \in \mathbb{F}_q[X]\) satisfying (2.7) as \(Y\)-root of \(Q\).

**Proof.** Define \(R(X) := Q(X, f(X), f(\gamma X), \cdots, f(\gamma^{s-1} X))\), for which \(\deg(R) \leq d + k - 1\). If \(\text{Enc}(f(X))\) agrees with \(y\) in the \(i\)th column; then \(f(\gamma^{im+i}) = y_{im+1+i}\), for \(i \in \{0, 1, \cdots, m - 1\}\) and \(i\)’s as defined for (2.6). Together with condition (2.6) the assumption that \(m = s\), this implies that

\[R(\gamma^i) = Q(\gamma^i, f(\gamma^i), f(\gamma^{i+1}), \cdots, f(\gamma^{i+s-1})) = 0.\]

It follows that \(R(X)\) has at least \(t \geq d + k\) zeros. Since \(\deg(R) \leq d + k - 1\), by the fundamental theorem of algebra \(R(X) \equiv 0\). \(\square\)

All in all, lemma 2.4 provides the correctness of the procedure we are describing.

### 2.2 Root-finding step

The second step of our decoding algorithm is an \((s + 1)\)-variate “root-type” problem:

Given \(Q(X, Y) \neq 0\) with coefficients in \(\mathbb{F}_q\), \(\gamma \in \mathbb{F}_q\) a primitive element, and parameter \(k < n = q - 1\), find the list of all polynomials \(f(X)\) of degree at most \(k - 1\) such that

\[Q(X, f(X), f(\gamma X), \cdots, f(\gamma^{s-1} X)) = 0.\]
The following algebraic lemma is an important step to solving this problem.

**Lemma 2.5.** For $\gamma \in \mathbb{F}_q$ a primitive element, we have:

1. The polynomial $E(X) := X^{q-1} - \gamma$ is irreducible over $\mathbb{F}_q$.
2. If $\deg(f) < q - 1$, then $f(\gamma X) \equiv f(X)^q \mod (X^{q-1} - \gamma)$.

We now present how to list the polynomials $f(X) \in \mathbb{F}_q[X]$ of degree $\leq k - 1$ for the trivariate case ($s = 2$), to satisfy the condition $Q(X, f(X), f(\gamma X)) \equiv 0$. We then discuss how this is can be generalized to $s \geq 3$.

**Theorem 2.6.** Consider the finite field $\mathbb{F}_q$ with a primitive element $\gamma$, and $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$ nonzero with $\deg_{Y_1}(Q) \leq q - 1$, along with an integer parameter $k < q$. There is a deterministic algorithm with runtime $\text{poly}(q)$, which outputs the list of all $f(X)$ of degree at most $k - 1$, satisfying $Q(X, f(X), f(\gamma X)) \equiv 0$.

**Proof.** We know by lemma 2.5 part 1 that $E(X)$ is irreducible. For $b \in \mathbb{N}_0$ such that $E(X)^b \parallel Q(X, Y_1, Y_2)$, i.e. $E(X)^{b+1} \nmid Q(X, Y_1, Y_2)$ while $E(X)^b \mid Q(X, Y_1, Y_2)$, factor out $E(X)^b$ to obtain $Q_0(X, Y_1, Y_2) = E(X)^{-b} \cdot Q(X, Y_1, Y_2)$. It is clear that $E(X) \nmid Q_0(X, Y_1, Y_2)$, and that if $Q(X, f(X), f(\gamma X)) = 0$, then $Q_0(X, f(X), f(\gamma X)) = 0$.

We may therefore focus on $Q_0$ instead, which we view as a polynomial $T_0(Y_1, Y_2) \in \mathbb{F}_q[Y_1, Y_2]$, for which we reduce the coefficients of modulo $E(X)$ to get $T(Y_1, Y_2) \in \mathbb{F}[Y_1, Y_2]$, for $\mathbb{F} := \mathbb{F}_q[X]/(E(X)) \cong \mathbb{F}_{q^b}$. That is, the bivariate polynomial $T(Y_1, Y_2)$ is over the extension field $\mathbb{F}$. Further note that $T(Y_1, Y_2) \not\equiv 0$, since $E(X) \nmid Q_0(X, Y_1, Y_2)$.

From the second part of our lemma, it suffices to find all polynomials $f(X)$ of degree $\leq k - 1$ satisfying $Q_0(X, f(X), f(X)^q) \equiv 0 \mod E(X)$, i.e. $E(X) \mid Q_0(X, f(X), f(X)^q)$. This reduces to finding the elements $\Gamma \in \mathbb{F}$ satisfying $T(\Gamma, \Gamma^q) = 0$. For the univariate polynomial $R_2(Y_1) := T(Y_1, Y_1^q)$, this corresponds to finding its roots in $\mathbb{F} - R_2(\Gamma) = 0$. To recap, $R_2(\Gamma) = 0$ for $\Gamma \in \mathbb{F}$ has a correspondence with the coefficients of $f(x) \in \mathbb{F}_q[X]$ of $T_0(Y_1, Y_2)$, for which $Q_0(X, f(X), x(\gamma X)) = 0$.

Note that $R_2(Y_1) = 0$ if and only if $(Y_2 - Y_1^q) \mid T(Y_1, Y_2)$, which cannot happen as $\deg_{Y_1}(T) < q$ (char($\mathbb{F}$) $\leq q$). Furthermore, $\deg(R_2) \leq dq$ for $d$ the total degree of $Q(X, Y_1, Y_2)$. Since char($\mathbb{F}$) $\leq q$ and $[\mathbb{F} : \mathbb{F}_q] = [\mathbb{F} : \mathbb{F}_p] \cdot [\mathbb{F}_p : \mathbb{F}_q] = (q - 1) \log q$, we have $[\mathbb{F} : \mathbb{F}_q] \leq q \log q$. Using Berlekamp’s deterministic factorization algorithm, we can find all roots of $R_2(Y_1)$ in time $\text{poly}(d, q)$ [Ber70],[Ker90]. Each such root is retrieved as an element in $\mathbb{F}$, which corresponds to a polynomial $f(X) \in \mathbb{F}_q[X]$ of degree less than $q - 1$. Once we have this list, we reduce it by only outputting the polynomials $f(X)$ of degree at most $k - 1$ satisfying $Q_0(X, f(X), f(\gamma X)) = 0$. \[\Box\]

In the case where $s \geq 3$ the ideas in the above proof still apply, where now we want to list all degree $k - 1$ polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying (2.7), i.e. the $Y$-roots of $Q$. By dividing $Q(X, Y) \in \mathbb{F}_q[X][Y]$ by $E(X)$ enough times, we can assume that not all coefficients in $\mathbb{F}_q[X]$ are divisible by $E(X)$. We then quotient out $E(X)$ to get a nonzero polynomial $T(Y)$ over $\mathbb{F}$. By lemma 2.5 part 2, $f(\gamma^j X) \equiv f(X)^q \mod E(X)$ for all $j \in \mathbb{Z}_+$. Our root-finding task is now reduced to finding all roots $\Gamma \in \mathbb{F}$ of $R_s(Y_1) := T(Y_1, Y_1^q, Y_1^{q^2}, \ldots, Y_1^{q^{s-1}})$. We further need to make the assumption that the total degree of $T$ is less than $q$, to ensure that $R_s(Y_1) \not\equiv 0$. The degree of $R_s(Y_1)$ is at most $q^s$, which means all its roots can be found in $q^{O(s)}$ time.

With this approach, we retrieve a list of at most $q^s$ polynomials in poly($q$) time. With rate $R$ we achieve polynomial time list decoding up to a fraction $1 - R - \varepsilon$ of errors for every $R$ and
arbitrary \( \varepsilon > 0 \), where the alphabet size is \( n^{O(1/\varepsilon)} \) [GR08]. The optimal trade-off between rate and error-correction capability is therefore attained algorithmically.

3 Linear-Algebraic List Decoding of Folded Reed-Solomon Codes

In [Gur11] a linear-algebra based analysis of a variant of the above algorithm was given, which avoids the computationally expensive root-finding step over \( \mathbb{F} \). The main idea is to solve one linear system in place of the interpolation step, and another one to find a “small” subspace of candidate solutions. There is again the step of “pruning” the list of candidate solutions (in this case a subspace), but other than this, the linear-algebraic algorithm can be implemented in quadratic time. Further, if \( |A| \neq 0 \) the solutions to the algebraic equations we wish to solve was not crucial in finding the roots. However, this identity seems to be the only known way to bound the list-size when higher degrees are used in the interpolation.

For our new list decoding algorithm, we need to find all polynomials \( f(X) \in \mathbb{F}_q[X] \) of degree at most \( k - 1 \) that satisfy the system of linear equations

\[
\Lambda(X) := A_0(X) + A_1(X)f(X) + A_2(X)f(\gamma X) + \cdots + A_s(X)f(\gamma^{s-1}X) = 0
\]  

in the coefficients of \( f(X) = \sum_{i=0}^{k-1} f_i X^i \in \mathbb{F}_q[X] \). Fact 3.1 gives an efficient algorithm to find a compact representation of all the solutions of (3.1). Additionally, the proof of lemma 3.2 exposes the simple structure of (3.1), which can be used to find the basis of solutions in quadratic time.

**Fact 3.1.** The solutions \( (f_0, f_1, \cdots, f_{k-1}) \) of (3.1) form an affine subspace of \( \mathbb{F}_q^k \).

**Lemma 3.2.** If \( \text{ord}(\gamma) \geq k \) (which is met for \( \gamma \) primitive and \( k \leq q - 1 \)), the affine subspace of solutions to (3.1) has dimension \( d \leq s - 1 \). Further, one can compute using \( O((N m^3)^2) = O(n^2) \) operations over \( \mathbb{F}_q \) a matrix \( M \in \mathbb{F}^{k \times d} \) (for some \( d \leq s - 1 \)) and a vector \( z \in \mathbb{F}_q^k \), such that the solutions are contained in the affine space \( Mx + z \) for \( x \in \mathbb{F}_q^d \). Also, \( M \) can be assumed to have the identity matrix \( I_q \) as a submatrix (without any extra computation).

**Proof.** By factoring out the common powers of \( X \) that divide \( \{A_i(X)\}_{i=0}^s \) from (2.4), we can assume that \( X \nmid A_i(X) \) for at least one \( i \in \{0, 1, \cdots, s \} \) — more specifically, has a nonzero constant term. Further, if \( X \nmid A_i(X) \) for all \( i \in \mathbb{N}_s \); then \( X \nmid A_0(X) \), and we can take \( i \in \mathbb{N}_s \).

For \( 0 \leq i \leq s \) denote \( A_i(X) = \sum_{j=0}^{d+k-1} a_{i,j} X^j \). By the degree constraints on \( \{A_i(X)\}_{i=1}^s \) we have \( a_{i,j} = 0 \) for all pairs \( (i, i) \in \mathbb{N}_s \times \mathbb{Z}_{>d} \), but we still introduce these coefficients for notational convenience. Define

\[
B(X) := a_{1,0} + a_{2,0}X + a_{3,0}X^2 + \cdots + a_{s,0}X^{s-1} = \sum_{i=0}^{s} a_{i,0}X^{i-1}
\]
which corresponds to $A_0(X)$, for which $\deg(B) \leq s - 1$. Since $a_{i,0} \neq 0$, it follows that $B(X) \neq 0$.

It is clear that the constant term of $A(X)$ equals \( a_{0,0} + f_0 \sum_{i=0}^{s} a_{i,0} \). Thus if $B(1) \neq 0$, the coefficient $f_0$ is uniquely determined as $-a_{0,0}/B(1)$. If $B(1) = 0$; then $a_{0,0} = 0$, or there will be no solutions to (3.1). In that case, we assign an arbitrary value in $\mathbb{F}_q$ to $f_0$.

The coefficient of $X^r$ of $A(X)$ equals

\[
\lambda_r = a_{0,r} + \left( \sum_{j=0}^{r} f_{r-j} \left( \sum_{i=1}^{s} a_{i,j} \gamma^{(i-1)(r-j)} \right) \right) = B(\gamma^r) f_r + \left( \sum_{l=0}^{r-1} b_l^{(r)} f_l \right) + a_{0,r} \tag{3.2}
\]

for some coefficients $b_l^{(r)} \in \mathbb{F}_q$, and (3.2) must equal zero. Furthermore, if $B(\gamma^r) \neq 0; f_r$ is an affine combination of $\{f_j\}_{j=0}^{r-1}$. In particular, $f_r$ is uniquely determined given the values of $\{f_j\}_{j=0}^{r-1}$.

The dimension of the space of solutions of (3.1) is therefore at most $r$; $0 \leq r < k$, for which $B(\gamma^r) = 0$. By our assumption that $\ord(\gamma) \geq k$, it follows that $\{\gamma^r\}_{r=0}^{k-1}$ are all distinct. Since $B(X)$ is a nonzero polynomial and $\deg(B) \leq s - 1$, we know that $B(\gamma^r) = 0$ for at most $s - 1$ values of $r$. This concludes the proof that the solution space is of dimension at most $s - 1$. The claim regarding quadratic complexity and the structure of $M$, follows from the fact that (3.2) resembles a “lower-triangular” form, which can be solved in $O(n^2)$ with the back-substitution method. \( \square \)

We close off this section with some comments on the rest of the results from [Gur11]. The algorithm we saw gives a quadratic runtime for the list decoder; except for the final step of pruning the subspace, which could take $O(q^k)$ time. The formal statement may be found in [Gur11] theorem 7, which also relates to the discussion we had following theorem 2.3. Lastly, the author discusses a possible approach to improving the possible worst case list-size bound, by restricting the message coefficients $(f_0, \ldots, f_{k-1})$ to belong to a special subset $V \subseteq \mathbb{F}_q^k$ which satisfy two conflicting demands; largeness and (what he coined as) subspace-evasive [DL12],[BAS14].

4 Hensel-Lifting for Folded Reed-Solomon Codes

In this section we present a third approach to the root-finding problem. This is quite different to the approaches discussed in §2.2, §3, and uses ideas developed in number theory; namely Hensel’s (lifting) lemma 4.1. We give a brief discussion on the importance of this in appendix A. By theorem 2.3 we already have a polynomial time algorithm which predates the algorithm based on Hensel-lifting [Bra10],[BB09], though the fact that it works over the exponentially large finite field $\mathbb{F} \cong \mathbb{F}_{q^{q-1}}$, makes any practical implementations difficult and even more numerically unstable. This newer decoder, is also faster experimentally.

**Lemma 4.1** (Simplest version, [NZM91]). Suppose that $f(x) \in \mathbb{Z}[x]$. If $f(a) \equiv 0 \mod p^j$ and $f'(a) \not\equiv 0 \mod p$, then there exists a unique $t \mod p$ such that $f(a + tp^j) \equiv 0 \mod p^{j+1}$.

**Definition 4.2.** The polynomial $g(X)$ is a partial $Y$-root of precision $b$ in $Q(X,Y)$, if $g(X) \equiv f(X)(\mod X^b)$, for some $Y$-root $f(X) \in \mathbb{F}_q[X]$ of $Q(X,Y)$.

We note that the degree of a partial $Y$-root of precision $b$, is at most $b - 1$, and Hensel-lifting is a general procedure for computing such roots. The key is to recursively lift a partial root of degree $b$ to a new one of precision $b + 1$, as in the simplest case of Hensel’s lemma 4.1. Let $Q$ satisfy (2.6) (in
The above discussion can be summarized in the recursive expression
\[ Q(X, f(X), f(\gamma X), \cdots, f(\gamma^{s-1} X)) \equiv Q(0, f_0, \cdots, f_0) \pmod{X} \equiv 0 \pmod{X}. \] (4.1)

Since \(X \nmid Q(0, f_0, \cdots, f_0)\), it follows that \(Q(0, f_0, \cdots, f_0) = 0\). A partial root \(f_0\) of precision \(b = 1\) must therefore satisfy this condition.

It is clear that if \(X^r \mid Q(X, Y)\) for some \(r \in \mathbb{Z}_+\), then any \(Y\)-root of \(Q\) will also be a root of \(X^{-r}Q\) — this idea resembles the constructive proof of lemma 4.1. We may therefore assume that \(X \nmid Q\), or equivalently that \(Q(0, Y) \neq 0\). There is a subtlety here though. If \(Q(0, Y) \in \mathcal{I}\), for \(\mathcal{I}\) the proper ideal \(\mathcal{I} = (Y_1, Y_2, Y_3, \cdots, Y_{s-1} - Y_s) < F_q[Y]\); then \(Q(0, Y, \cdots, Y) = 0\), and (4.1) reveals nothing about \(f_0\). In such a case, we have \(q\) partial roots of precision \(b = 1\). Furthermore if \(Q(0, Y, \cdots, Y) \neq 0\), by (4.1) \(f_0\) must be among the roots of this polynomial. Since \(\text{deg}(Q(X, f(X), \cdots, f(\gamma^{s-1} X))) \leq \text{deg}(Q(0, Y))\), one can constrain the number of possible partial roots of precision \(b = 1\) (at most \(\ell = \left\lceil \frac{\Delta}{k} \right\rceil\), for \(\Delta\) defined in [Bra10] corollary 5.6).

For lifting partial roots, under the assumption that \(f(X) = f_0 + X\tilde{f}(X)\) is a \(Y\)-root of \(Q\), i.e. \(Q(X, f(X), f(\gamma X), \cdots, f(\gamma^{s-1} X)) = Q(X, f_0 + X\tilde{f}(X), f_0 + \gamma X\tilde{f}(\gamma X), \cdots, f_0 + \gamma^{s-1} X\tilde{f}(\gamma^{s-1} X)) = 0\) and \(f_0\) is known, it follows that \(\tilde{f}(X)\) is a \(Y\)-root of
\[ \tilde{Q}(X, Y) := Q(X, f_0 + X \cdot Y_1, f_0 + \gamma X \cdot Y_2, \cdots, f_0 + \gamma^{s-1} X \cdot Y_s) \]
where now \(\gamma^{s-1} X\) from the above identity “replace” the corresponding variable \(Y_i\), for all \(i \in \mathbb{N}_s\).

**Lemma 4.3.** Let the polynomial \(Q(X, Y) \in F_q[X, Y]\) be nonzero, and \(b \in F_q\). Then the polynomial \(Q(X, b + X \cdot Y_1, \cdots, b + \gamma^{s-1} X \cdot Y_s)\) is also nonzero.

**Proof.** Define the bijection \(\phi_b : F_q[X][Y] \to F_q[X][Y]\) as
\[ \phi_b(P(X, Y_1, Y_2, \cdots, Y_s)) = P(X, b + X \cdot Y_1, b + \gamma X \cdot Y_2, \cdots, b + \gamma^{s-1} X \cdot Y_s) \]
for \(P(X, Y) \in F_q[X][Y]\). If \(P = 0\), then \(\phi_b(P) = 0\). Since \(F_q[X, Y] \cong F_q[X][Y]\), we may assume that \(P(X, Y) \in F_q[X, Y]\). By assuming that \(Q(X, Y) \neq 0\), it follows that \(\phi_b(Q) \neq 0\). \(\square\)

This lemma is precisely what we need for lifting the partial roots, and can be viewed as another case of lemma 4.1. Under the assumption that \(Q(X, Y) \neq 0\), it follows that \(\tilde{Q}(X, Y) \neq 0\). By lemma 4.3 it follows that there exists an integer \(r \geq 0\) for which \(X^r \mid \tilde{Q}(X, Y)\), and we define
\[ Q_{f_0}(X, Y) = X^{-r} \cdot \tilde{Q}(X, Y). \]
Recall that \(f_0\) was defined such that \(f(X) = f_0 + X\tilde{f}(X)\) is a \(Y\)-root of \(Q\), so by evaluating \(Q(X, Y)\) at \(X = 0\) we get \(Q(0, Y) = Q(0, f_0, \cdots, f_0) = 0\). This implies that \(X \mid Q(X, Y)\), hence \(r\) is positive. Since \(\tilde{f}(X)\) is a \(Y\)-root of \(Q\), it follows that it is also a \(Y\)-root of \(Q_{f_0}\).

The above discussion can be summarized in the recursive expression
\[ \Phi_k(Q) \subseteq \bigcup_{f_0 \in \Phi_1(Q)} \{f_0 + X \cdot \Phi_{k-1}(Q_{f_0})\} \] (4.2)
where $\Phi_k(Q)$ denotes the set of partial $Y$-roots of $Q$, of precision $k$. In the recursive expression, it is clear that partial roots of precision $b$, are also partial roots of precision $b + 1$. By definition, $\Phi_k(Q)$ contains the set of all $Y$-roots of $Q$ of degree at degree at most $k - 1$. Likewise, we define the list of polynomials

$$\Lambda_k(Q) = \bigcup_{f_0 \in \Lambda_1(Q)} (f_0 + X \cdot \Lambda_{k-1}(Q_{f_0})) \quad \text{where} \quad \Lambda_1(Q) = \{f_0 \in \mathbb{F}_q \mid Q(0, f_0, \cdots, f_0) = 0\}$$

and by the condition we showed that precision $b = 1$ partial roots $f_0$ must satisfy (4.1), we have

$$\Phi_k(Q) \subseteq \Lambda_k(Q)$$

for all $k \in \mathbb{Z}_+$. We can attain $\Lambda_1(Q)$ by enumerating all $f_0 \in \mathbb{F}_q$ satisfying (4.1), hence; we can recursively compute $\Lambda_k(Q)$ by algorithm 1.

**Algorithm 1: Enumeration of $\Lambda_k(Q)$**

**Input:** $k \in \mathbb{Z}_+$ and $Q(X, Y) \in \mathbb{F}_q[X, Y]$

**Output:** $\Lambda_k(Q)$ — a list containing all precision $k$ $Y$-roots of $Q$

if $k \leq 0$ then
  $\Lambda_k(Q) \leftarrow \{0\}$
else
  let $Q_{f_0}(X, Y) \leftarrow X^{-r} \cdot Q(X, Y)$, for $r$ s.t. $X^r \parallel Q(X, Y)$
  if $Q_{f_0}(0, Y, \cdots, Y) = 0$ then
    $B \leftarrow \mathbb{F}_q$
  else
    $B \leftarrow \{\beta \in \mathbb{F}_q : Q(0, \beta, \cdots, \beta) = 0\}$
  end
return $\Lambda_k(Q) \leftarrow \bigcup_{f_0 \in B} (f_0 + X \cdot \Lambda_{k-1}(Q_{f_0}))$  \  $\triangleright$ compute recursively

By our previous discussions, the set $\Lambda_k(Q)$ contains the list of polynomials we are looking for, from §2.2. This list can then be reduced to the set of $Y$-roots of $Q$ of degree at most $k - 1$, by retaining only the polynomials which satisfy (2.7). For further comparison of the root-methods, refer to [Bra10] sections 5.4.3.

## 5 Locally Decodable Multiplicity Codes

We now shift gears and turn our attention to *multiplicity codes* [KSY10], a type of locally decodable error-correcting codes (LDCs). Recall that the main parameters of LDCs are its length $n$ (or its rate $R = k/n$ for fixed $k$) and query complexity of local decoding. Ideally, we would like to have both of these parameters be small, though one cannot minimize them both simultaneously.

Most work prior to [KSY10] focused on studying codes in the low and constant query regimes, which have applications in cryptography and complexity theory. Multiplicity codes on the other hand, were introduced in order to study how the query complexity of large rate (approaching 1) LDCs can be minimized. Before the construction of these codes, it was unknown how to get any
nontrivial local decoding for codes of rate $R > 1/2$.

This relatively new family of codes has been named multiplicity codes, as they are based on evaluating multivariate polynomials and their derivatives, while also considering high-multiplicity zeroes. By the way they are defined, they inherit the local decodability of the classical multivariate polynomial codes based, while achieving better trade-offs and flexibility in the rate and minimum distance. In §6 we will see how variants of these codes relate to FRS codes.

Before we start, let us define a local self-correction property, as for our purposes we want to construct “locally self-correctable codes” (LSCCs) over “large alphabets” $\Sigma$. The code $\mathcal{C} \subseteq \Sigma^n$ of size $|\mathcal{C}| = |\Sigma|^k$ and large $R$ we want to construct, should satisfy the property: given access to a received string $r \in \Sigma^n$ which is close to some $c \in \mathcal{C}$, and given any coordinate index $i \in \mathbb{N}_n$, it is possible to make few queries to the coordinates of $r$, and with high probability retrieve $c_i$. We point out that this is different from the notion of local decodability, where to goal is to recover the coordinate of the original message $\vec{m} \in \Sigma^k$. We show in §5.3 though that for linear codes, LSCCs imply LDCs. Throughout §5, $q$ denotes a power of a prime $p$.

### 5.1 Bivariate Multiplicity Codes

In order to define the bivariate multiplicity codes, the simplest example of multiplicity codes, we first need to give several definitions. The bivariate multiplicity codes already have improvements in terms of rate for local self-correction over Reed-Muller codes (RM), while being locally self-correctable with only a constant factor more queries.

For a vector $\mathbf{i} = (i_1, \cdots, i_n) \in \mathbb{N}_0^n$, we denote its weight by $\text{wt}(\mathbf{i}) = \|\mathbf{i}\|_1 = \sum_{j=1}^n i_j$. As in §2.1, denote the formal variables $X_1, \cdots, X_n$ by $X$, thus $\mathbb{F}[X] = \mathbb{F}[X_1, \cdots, X_n]$. Lastly, for $\mathbf{i} \in \mathbb{N}_0^n$ let $X^\mathbf{i}$ denote the monomial $\prod_{j=1}^n X_j^{i_j} \in \mathbb{F}[X]$. It follows that (total) $\deg(X^\mathbf{i}) = \text{wt}(\mathbf{i})$.

**Definition 5.1.** For $P(X) \in \mathbb{F}[X]$ and $\mathbf{i} \in \mathbb{N}_0^n$, the $\mathbf{i}$th Hasse derivative of $P$, denoted $P^{(\mathbf{i})}(X) \in \mathbb{F}[X]$, is the coefficient of $X^\mathbf{i}$ in the polynomial $\tilde{P}(X, Z) := P(X + Z) \in \mathbb{F}[X, Z]$. Thus

$$P(X + Z) = \sum_{\mathbf{i}} P^{(\mathbf{i})}(X)Z^\mathbf{i}$$

and observe that for all $P, Q \in \mathbb{F}[X]$ and $\lambda \in \mathbb{F}$

$$(\lambda P)^{(\mathbf{i})}(X) = \lambda P^{(\mathbf{i})}(X) \quad \text{and} \quad P^{(\mathbf{i})}(X) + Q^{(\mathbf{i})}(X) = (P + Q)^{(\mathbf{i})}(X).$$

**Definition 5.2.** For $P(X) \in \mathbb{F}[X]$ and $a \in \mathbb{F}^n$, the multiplicity of $P$ at $a$, denoted by $\text{mult}(P, a)$, is the largest integer $M$ such that for every non-negative vector $\mathbf{i}$ with $\text{wt}(\mathbf{i}) < M$, we have $P^{(\mathbf{i})}(a) = 0$ (if $M$ is taken arbitrarily large, we set $\text{mult}(P, a) = \infty$). Note that $\text{mult}(P, a) \geq 0$ for every $a$.

**Definition 5.3.** The multiplicity code of order 2 evaluations of degree $d = 2(1 - \delta)q$ bivariate polynomials over $\mathbb{F}_q$ for $\delta > 0$, is the set of codeword vectors corresponding to the polynomials $P(X, Y) \in \mathbb{F}_q[X, Y]$

$$C(P) = \left\{ \left( P(a), \frac{\partial P}{\partial X}(a), \frac{\partial P}{\partial Y}(a) \right) \right\}_{a \in \mathbb{F}_q^2} \subseteq \mathbb{F}_q^{2q^2}$$

where $C(P)$ indicates the encoding of $P$. The coordinates are indexed by $\mathbb{F}_q^2$; thus $n = q^2$, and the codewords are indexed by the bivariate polynomials of degree at most $d$ over $\mathbb{F}_q$. 

10
In simpler words, the \( a \) coordinate consists of the evaluations of \( P \) and its two partial derivatives at \( a \). By [DKSS13] lemma 8 (strengthening of the Schwartz-Zippel lemma), it follows that two distinct polynomials of degree at most \( d \) can agree with multiplicity 2 on at most \( d/2q \)-fraction of the points in \( \mathbb{F}_q^2 \); hence this code has relative distance \( \delta = 1 - d/2q \). Since now \( |\Sigma| = q^3 \), the message length \( k \) equals the number of \( q^3 \)-ary symbols required to specify a polynomial of degree at most \( d \). Since we have \( d + 1 \) monomials, 2 variables and are “grouping” the elements in pairs of three (going from \( q \)-ary to \( q^3 \)-ary), we get \( k = \binom{d+1}{2}/3 \). The rate of the bivariate multiplicity code is therefore

\[
R = \frac{k}{n} = \binom{d+1}{2}/3 \approx \frac{d^2}{q^2} \cdot \frac{2(1-\delta)^2}{3} = \frac{2(1-\delta)^2}{3} \implies \lim_{\delta \to 0} \{R\} = \frac{2}{3}
\]

an improvement to the rate of the corresponding RM code; which was less than \( \frac{1}{2} \), while having the same distance. The bivariate RM code is instead defined by \( C(P) = \langle P(a) \rangle_{a \in \mathbb{F}_q^2} \in \mathbb{F}_q^{q^2} \), and has parameters \( \delta = 1 - d/q, k = \binom{d+1}{2}, n = q^2, \) thus \( R = \binom{d+1}{2}/q^2 \approx \frac{(1-\delta)^2}{2} < \frac{1}{2} \).

### 5.2 Local Self-Correction of Bivariate Multiplicity Codes

We now see how local self-correction is achieved. Given a received word \( r \in \langle \mathbb{F}_q^3 \rangle q^2 \) close to the codeword \( C(P) \) in terms of Hamming distance \( \Delta_H(\cdot, \cdot) \), we want to recover the “correct” symbol at coordinate \( a \) of a given point \( a \in \mathbb{F}_q^2 \), namely \( \langle P(a), \frac{\partial P}{\partial X}(a), \frac{\partial P}{\partial Y}(a) \rangle \). The approach is similar to local self-correction of RM codes, where we pick a random direction \( b \in \mathbb{F}_q^2 \) and look at the restriction of \( r \) to coordinates in the line \( L = \{a + bt \mid t \in \mathbb{F}_q\} \). With high probability over the choice of \( b \), \( r|_L \) and \( C(P)|_L \) agree in many locations; i.e. \( \Delta_H (r|_L, C(P)|_L) \) is small. The next step is to recover \( Q(T) = P(a + bT) \) for which deg\((Q)\) \( \leq 2(1-\delta)q \), in order to compute the 3-tuple defining \( C(P) \).

It is important to notice that for every \( t \in \mathbb{F}_q \), the \( a + bt \in L \) coordinate of \( C(P) \) completely determines both the value and the 1st derivative of \( Q(T) \) at point \( t \), as by the chain rule we have

\[
\left( Q(t), \frac{\partial Q}{\partial T}(t) \right) = \left( P(a + bt), b_1 \frac{\partial P}{\partial X}(a + bt) + b_2 \frac{\partial P}{\partial Y}(a + bt) \right) \in \mathbb{F}_q^2.
\]

Our knowledge of \( r|_L \) therefore gives us access to \( q \) “noisy” evaluations of \( Q(T) \) (one for each \( t \in \mathbb{F}_q \)), and its derivative \( \frac{\partial Q}{\partial T}(T) \), which is enough for recovering \( Q(T) \). Clearly \( Q(0) = P(a) \), and \( \frac{\partial Q}{\partial T}(0) = b_1 \frac{\partial P}{\partial X}(a) + b_2 \frac{\partial P}{\partial Y}(a) \) is the directional derivative of \( P \) at \( a \) in direction \( b \).

We repeat the above for a different direction \( b \in \mathbb{F}_q^2 \) and \( \dot{Q}(T) = P(a + b\dot{t}) \), to recover the directional derivative \( \frac{\partial \dot{Q}}{\partial T}(0) = b_1 \frac{\partial P}{\partial X}(a) + b_2 \frac{\partial P}{\partial Y}(a) \) of \( P \) at \( a \) in direction \( b \). Together, the two directional derivatives \( \frac{\partial Q}{\partial T}(0) \) and \( \frac{\partial \dot{Q}}{\partial T}(0) \) suffice to recover \( \frac{\partial P}{\partial X}(a) \) and \( \frac{\partial P}{\partial Y}(a) \), as we have a linear system of two equations; with two unknowns which we want to recover. All in all, this approach makes \( 2q = O(\sqrt{k}) \) queries; needed for the “noisy” evaluations of \( Q(T) \) and \( \dot{Q}(T) \). This sublinear query complexity was something not known before, for local decoding in the regime of \( R > 1/2 \).

### 5.3 Multiplicity Codes and Local Self-Correction

In order to get multiplicity codes of rate approaching 1, we also consider evaluations of all derivatives of the multivariate polynomial \( P \) up to an even higher order. To locally recover the evaluations
of the higher order at a point \(a\), we pick many random lines passing through \(a\), try to recover the restriction of \(P\) to those lines (correspond to univariate polynomials), which we combine in a certain way. The procedure is formally explained in algorithm 2. By simultaneously increasing the maximum order of derivative taken and the number of variables, we attain multiplicity codes with the desired rate and local decodability. To state the results on the existence of LDCs with rate approaching 1, we need the following definitions.

**Definition 5.4.** The relative Hamming distance of two strings \(c, c' \in \Sigma^n\), is the fraction of coordinates in which they differ: \(\delta_H(c, c') = \frac{\Delta_H(c, c')}{n} = \Pr_{i \in \mathbb{N}_n}[c_i \neq c'_i].\)

**Definition 5.5 (Locally Self-Correctable Code).** A code \(C \subseteq \Sigma^n\) is said to be locally self-correctable from \(\delta'\)-fraction errors with \(t\) queries, if there is a randomized algorithm \(A\) such that:

- **Self-Correction:** Whenever \(\delta_H(r, c) < \delta'\) for \(c \in C\) and \(r \in \Sigma^n\), then for each \(i \in \mathbb{N}_n\)

\[
\Pr[A^r(i) = c_i] \geq \frac{2}{3}
\]

- **Query Complexity** \(t\): \(A^r(i)\) always makes at most \(t\) queries to \(r\)

where \(A^r\) represents the situation where \(A\) is given query access to \(r\).

**Definition 5.6 (Locally Decodable Code).** Let \(C \subseteq \Sigma^n\) be a code with \(|C| = |\Sigma|^k\), and \(E : \Sigma^k \rightarrow C\) a bijection; which is \(C\)’s encoding map. We say that \((C, E)\) is locally decodable from \(\delta'\)-fraction errors with \(t\) queries, if there is a randomized algorithm \(A\) such that:

- **Decoding:** Whenever \(\tilde{m} \in \Sigma^k\) and \(r \in \Sigma^n\) are such that \(\delta_H(r, E(\tilde{m})) < \delta'\), then for each \(i \in \mathbb{N}_k\)

\[
\Pr[A^r(i) = m_i] \geq \frac{2}{3}
\]

- **Query Complexity** \(t\): \(A^r(i)\) always makes at most \(t\) queries to \(r\)

where \(A^r\) represents the situation where \(A\) is given query access to \(r\).

Recall that any linear code has a systematic encoding, which means there is an encoding \(E\) such that for each \(\tilde{m} \in \Sigma^k\) and \(i \in \mathbb{N}_k\), there is a \(j \in \mathbb{N}_n\) such that \(E(\tilde{m})_j = m_i\). This gives us the implication that if \(C\) is an LSCC, then \((C, E)\) is a LDC, with the same fraction of errors \(\delta'\) and query complexity \(t\). We can view this implication as a reduction, which allows us to focus on constructing linear LSCCs. There is a caveat here, the fact that multiplicity codes are not linear codes. However, it is possibly to achieve linear LSCCs by concatenating multiplicity codes with suitable “good” linear codes over the small alphabet \(\Sigma = F_q\). The resulting LDCs have similar parameters. Furthermore, multiplicity codes themselves can also be locally decoded with a factor \(\exp(m + s)\)-increase in the query complexity, for a suitable encoding \(E\). Though obvious, it is also important to point out that local decoding is a function of the encoding \(E\).

We now define multiplicity codes, state and prove their rate and distance, and then show how their local self-correction is achieved. The compelling part about the relationship between rate and distance, is that if we keep \(\delta\) fixed and let the multiplicity parameter \(s\) grow the rate improves, as it approaches \((1 - \delta)^m\). For our constructions, we assume that \(\Sigma = F_q\).

**Definition 5.7.** Let \(s, d, m \in \mathbb{N}_0\) and \(\Sigma = F_q^{(m+s-1)} = F_q^{\{1^{|\text{wt}(i)|<s}\}}\). For \(P(X) \in F_q[X]\) where \(X = (X_1, \cdots, X_m)\), and \(a \in F_q^m\), the order \(s\) evaluation of \(P\) at \(a\), denoted \(P^{(s)}(a)\), is the vector \(\langle P^{(s)}(a) \rangle_{\text{wt}(i)<s} \in \Sigma\). The multiplicity code of order \(s\) evaluations of degree \(d\)
polynomials in $m$ variables over $\mathbb{F}_q$, is the code over $\Sigma$ of length $n = q^m$ (where the coordinates are indexed by the elements $a \in \mathbb{F}_q^m$). For each $P(X) \in \mathbb{F}_q[X]$ with $\deg(P) \leq d$, there is a codeword in $C$ given by the encoding:

$$\text{Enc}_{s,d,m,q}(P) := \left\langle P^{(s)}(a) \right\rangle_{a \in \mathbb{F}_q^m} \in (\Sigma)^q^m.$$ 

**Lemma 5.8** (Rate and distance of multiplicity codes). Let $C$ be a multiplicity code of order $s$ evaluations of degree $d$ polynomials in $m$ variables over $\mathbb{F}_q$. Then $C$ has $\delta = 1 - \frac{d}{sq}$ and $R = \left(\frac{d+m}{s+m}\right)_{q^m}$, for which $R \geq \left(\frac{s}{m+s}\right)^m \cdot \left(\frac{1}{m-s}\right)^m \geq \left(1 - \frac{m^2}{s}\right)^m$.

**Proof.** Consider two codewords $P_1 = \text{Enc}_{s,d,m,q}(P_1)$ and $P_2 = \text{Enc}_{s,d,m,q}(P_2)$ where $P_1 \neq P_2$. For the coordinates $a \in \mathbb{F}_q^m$ where $c_1, c_2$ agree; i.e. $(c_1)_a = (c_2)_a$, we have $P_1^{(s)}(a) = P_2^{(s)}(a)$. Consequently, for any such $a$ we have $(P_1 - P_2)^{(s)}(a) = 0$ for each $i \in \{i : \deg(i) < s\}$, thus mult($P_1 - P_2, a$) $\geq s$. From [KSY10], [DKSS13] lemmas 7 and 8 respectively, mult($P_1 - P_2, a$) $\geq s$ can occur on a fraction of at most $\frac{d}{sq}$ points $a \in \mathbb{F}_q^m$. The minimum relative distance $\delta$ of $C$ is therefore at least $\delta \geq 1 - \frac{d}{sq}$.

We now compute the code’s rate $R = \frac{\log |C|}{n \log |\Sigma|}$. By definition 5.7 our alphabet size is $q^{m+s-1}$ and block-length is $n = q^m$, so it remains to calculate $|C|$. A codeword is specified by giving coefficients to each of the monomials of degree at most $d$, thus $|C| = q^{\frac{(d+m)}{m}}$. The rate is therefore

$$R = \frac{(d+m)}{(s+m-1)}q^m = \frac{\prod_{j=0}^{m-1} (d+m-j)}{\prod_{j=1}^{m} (s+m-j)q} \geq \left(1 - \frac{m^2}{s}\right)^m \geq \left(1 - \frac{m^2}{s}\right)^m.$$

Using the parameters of definition 5.7, let $r : \mathbb{F}_q^m \to \Sigma$ be a received word for $\Sigma$ our code’s alphabet. Suppose $P(X) \in \mathbb{F}_q[X]$ has $\deg(P) \leq d$ such that $\delta_H(r, \text{Enc}_{s,d,m,q}(P))$ is small, and let $a \in \mathbb{F}_q^m$. Before showing how to locally recover $P^{(s)}(a)$ (algorithm 2) when given oracle access to $r$, we establish two relationships between the derivatives of the restriction of $P$ to a line to the derivatives of $P$ itself. Fix $a, b \in \mathbb{F}_q^m$ for $b \neq 0$, and consider the polynomial $Q(T) = P(a + bT)$.

- Relationship of $Q(T)$ with the derivatives of $P$ at $a$: By 5.1: $Q(T) = \sum_i P^{(i)}(a)b^iT^{\deg(i)}$.

  By grouping terms: $\left(\sum_{i:|\deg(i)|=e} P^{(i)}(a)b^i\right) = \text{coefficient of the monomial } T^e \text{ in } Q(T)$.

- Relationship of derivatives of $Q(T)$ at $t$ with the derivatives of $P$ at $a + bt$: By 5.1: $P(a + b(t + R)) = Q(t + R) = \sum_j Q^{(j)}(t)R^j$ and $P(a + b(t + R)) = \sum_i P^{(i)}(a + bt)(bR)^i$.

  for $t \in \mathbb{F}_q$. Thus: $Q^{(j)}(t) = \left(\sum_{i:|\deg(i)|=j} P^{(i)}(a + bt)b^i\right)$. More precisely, $Q^{(j)}(t)$ is a linear combination of various $P^{(i)}(a + bt)$ with coefficients $b^i$, over different $i$ of weight $j$. 

13
Recall that we want to recover $P^{(<s)}(a)$, where $P(X)$ is such that $\text{Enc}_{s,d,m,q}(P)$ is close to $r$, i.e. $\delta_H (r, \text{Enc}_{s,d,m,q}(P))$ is small. We denote the i coordinate of $r(a)$ by $r^{(i)}(a)$. This is done by algorithm 2.

**Algorithm 2: Simplified Local Self-Correction of Multiplicity Codes**

**Input:** received word $r : \mathbb{F}_q^m \rightarrow \Sigma$, and point $a \in \mathbb{F}_q^m$

**Output:** Vector $\langle u_i \rangle_{\text{wt}(i)<s}$

1. Pick a set of directions $B$: Choose $B \subseteq \mathbb{F}_q^m \setminus \{0\}$ uniformly at random, of size $(m+s-1)$.

2. Recover $P(a + bT)$ for $b \in B$: For each $b \in B$ consider $\ell_b : \mathbb{F}_q \rightarrow \mathbb{F}_q^s$ given by

$$\left( \ell_b(t) \right)_j = \sum_{i | \text{wt}(i) = j} r^{(i)}(a + bt)b^i.$$

Find $Q_b(T) \in \mathbb{F}_q[T]$ with $\deg(Q_b) \leq d$ (if any), s.t. $\delta_H(\text{Enc}_{s,d,1,q}(Q_b), \ell_b) < \delta/2$.

3. Solve a linear system to recover $P^{(<s)}(a)$: For each $c \in \{0, 1, \ldots, s - 1\}$ consider the system of equations in the variables $\langle u_i \rangle_{\text{wt}(i) = e}$ (with one equation for each $b \in B$):

$$\left( \sum_{i | \text{wt}(i) = e} b^iu_i \right) = \text{coefficient of } T^e \text{ in } Q_b(T). \quad (5.1)$$

Find all $\langle u_i \rangle_{\text{wt}(i) = e}$ satisfying the system (5.1).

4. Existence and uniqueness: If the solution does not exist or is not unique, output FAIL.

**Claim 5.9.** Algorithm 2 is a local-self corrector from a $\frac{\delta}{100w}$-fraction of errors, for $w = \binom{m+s-1}{m}$. Overall, it outputs $P^{(i)}(a)$ with probability at least $\left( \frac{9}{10} \right)^2 \approx 0.8$.

We validate the above claim, by analyzing the three steps of the algorithm. Fix a received word $r : \mathbb{F}_q^m \rightarrow \Sigma$ and $a \in \mathbb{F}_q^m$, and let $P(X)$ be a polynomial such that $\delta_H(\text{Enc}_{s,d,m,q}(P), r) < \frac{\delta}{100w}$. We call the points where $r$ and $\text{Enc}_{s,d,m,q}(P)$ differ the “errors”.

**Step 1 – All $b \in B$ are “good”:** For a fixed $b \in \mathbb{F}_q^m \setminus \{0\}$, we are interested in the fraction of errors on the line $L_b = \{a + bt \mid t \in \mathbb{F}_q^s\}$ through $a$ in direction $b$. Considering the space $\mathbb{F}_q^m$, the lines defined from the points in $B$ cover $\mathbb{F}_q^m \setminus \{a\}$ uniformly. By this, at most $\frac{1}{50w}$ of the lines containing $a$ have more than a $\frac{\delta}{2}$-fraction error on them. Therefore

$$\Pr_{B \leftarrow \mathcal{P}_w(\mathbb{F}_q^m \setminus \{0\})} \left[ \text{all } b \in B \text{ will be s.t. } L_b \text{ through } a \text{ has } \frac{\delta}{2} \text{ errors on it} \right] \geq \frac{9}{10},$$

where $\mathcal{P}_w$ denotes the subsets of cardinality $w$.

**Step 2 – $Q_b(T) = P(a + bT)$ for each $b \in B$:** In the case where $B$ satisfies the above event, by the third equation of the relationship relating the derivatives of $Q$ and $P$ (identity for $Q^{(i)}(t)$), for each $b$, the corresponding $\ell_b$ will satisfy $\delta_H(\text{Enc}_{s,d,1,q}(P(a + bT)), \ell_b) < \delta/2$. Thus, for each $b \in B$, the algorithm will find $Q_b(T) = P(a + bT)$.

**Step 3 – $u_i = P^{(i)}(a)$ for each $i$:** Since $Q_b(T) = P(a + bT)$ for each $b \in B$, by the second
The list decoding of derivative codes involves an interpolation step, and a second step of retrieving the list of polynomials satisfying a certain algebraic condition, similar to what we saw for FRS codes. The first step consists of fitting a polynomial of the form \( f(X) = \sum_{i=0}^{s} a_i X^i \) to the evaluations of the message polynomial \( f \) at each point of the defining set of points \( \mathcal{A} \), which holds as long as \( \deg(f) = s \). Without further ado, let us define derivative codes.

A central result of [Kop13] (theorem 10) states that for \( q \geq \max\{10m, \frac{d+6}{3}, 5(s+1)\} \) and \( \delta = 1 - \frac{d}{q^2} \), \( \mathcal{C} \) is locally self-correctable from \( \frac{d}{9} \)-fraction errors with \( q \cdot O(s)^m \) queries. The proof of the theorem in which this statement appears uses a slightly different algorithm for local self-correction of multiplicity codes. For more details and further results on multiplicity codes, please refer to [Kop13], [Kop15], and [Kop15]; in which another explicit capacity-achieving list decodable code was developed.

6 Derivative Codes

We present one last family of codes, derivative codes [GW11],[GW13], which we relate to the others presented thus far. This gives an alternate construction to FRS codes, for achieving the optimal trade-off between rate and list decoding error-correction radius. Informally, rather than bundling evaluations of the message polynomial \( f(X) \) at consecutive powers of \( \gamma \) as in (2.1), in an order-\( m \) derivative code, we bundle the evaluations of \( f(X) \) along with its first \( (m-1) \) derivatives at each point of the defining set of points \( \mathcal{A} = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q \). This resemblance makes this construction arguably just as natural as that of FRS codes. An interesting artifact of this construction is that the rate does not decrease, as one can pick higher degree polynomials; while still maintaining the distance. The reason is that two distinct polynomials of degree \( \ell \) and their first \( (m-1) \) derivatives, can agree in at most \( \ell/m \) points.

The list decoding of derivative codes involves an interpolation step, and a second step of retrieving the list of polynomials satisfying a certain algebraic condition, similar to what we saw for FRS codes. The first step consists of fitting a polynomial of the form (2.4). The second step which is new to us, consists of solving a “differential equation”. This was also considered in [Kop15], where the power series expansion of the potential solution was used to solve the same differential equation. Without further ado, let us define derivative codes.
Consider the received corrupted codeword from \( \text{Der}_q[m,n,k] \) or \( \text{Der}_q[n,k] \). The \( m \times n \) order derivative code \( \text{Der}_q(n,k) \) over the alphabet \( \mathbb{F}_q^m \cong \mathbb{F}_q^m \), encodes the polynomial \( f(X) \in \mathbb{F}_q[X] \) with degree \( f = k - 1 \) by

\[
\begin{pmatrix}
    f(\alpha_1) \\
    f'(\alpha_1) \\
    \vdots \\
    f^{(m-1)}(\alpha_1)
\end{pmatrix}, \quad \begin{pmatrix}
    f(\alpha_2) \\
    f'(\alpha_2) \\
    \vdots \\
    f^{(m-1)}(\alpha_2)
\end{pmatrix}, \quad \ldots, \quad \begin{pmatrix}
    f(\alpha_n) \\
    f'(\alpha_n) \\
    \vdots \\
    f^{(m-1)}(\alpha_n)
\end{pmatrix}
\end{equation}

where \( f'(X) \) denotes the formal derivative of \( f(X) \), and \( f^{(i)}(X) \) its \( i \)th formal derivative. This codes has length \( n \), rate \( R = \frac{k}{nm} \) and minimum distance \( d_{\text{min}} = n - \left\lfloor \frac{k-1}{m} \right\rfloor \approx (1 - R)n \). Furthermore, for \( m = 1 \) we get a \( \text{RS}_q[n,k] \).

Consider the received corrupted codeword from \( \text{Der}_q[n,k] \) as a string \( y \in (\mathbb{F}_q^m)^n \cong \mathbb{F}_q^{m \times n} \), which we realize as a \( m \times n \) matrix over \( \mathbb{F}_q \); as we did for (2.2). Just like in §2.1, the goal is to recover all polynomials \( f(X) \) of degree \( k - 1 \) whose encoding (6.1) agrees with \( y \) in at least \( t \) columns. This corresponds to decoding from \( n - t \) symbol errors for \( \text{Der}_q[m,n,k] \). The algorithm we present, as the one in §2, may be viewed as a higher dimensional analog of the Berlekamp-Welch algorithm.

### 6.1 Interpolation step

The interpolation step is similar in spirit to the one presented in §2.1. Using the same notation, let

\[
W = \{B_0(X) + B_1(X)Y_1 + \cdots + B_m(X)Y_m \mid B_i \in \mathbb{F}_q[X]\}
\]

which is a \( \mathbb{F}_q \)-linear subspace \( \mathbb{F}_q[X,Y] \). For \( p(X) \in \mathbb{F}_q[X] \) and \( i \in \mathbb{N}_m \), we define the \( \mathbb{F}_q \)-linear map

\[
D : p(X) \mapsto p'(X) = \left(B_0(X) + \sum_{i=1}^{m} B'_i(Y_i)\right)
\]

and

\[
D : p(Y_i) \mapsto \left(p'(X)Y_i + p(X)Y_{i+1}\right)
\]

from \( \mathbb{F}_q[X] \) to \( W \), where we take \( Y_{m+1} = Y_1 \).

For \( s \in \mathbb{N}_m \), we define the nonzero polynomial \( Q(X,Y) \) as in (2.4), satisfying the conditions

\[
Q(\alpha_i, y_{i1}, \cdots, y_{is}) = 0 \quad \text{and} \quad (D^k Q)(\alpha_i, y_{i1}, \cdots, y_{im}) = 0 \quad (6.2)
\]

for all \( i \in \mathbb{N}_n \), where \( k \in \mathbb{N}_{m-s} \) and \( D^k \) denotes the \( k \)th fold composition of \( D \) (apply it \( k \) times). Note that conditions (6.2) resemble (2.6). Furthermore, note that for each \( i \) the conditions (6.2) comprises a collection of \( (m - s + 1) \) homogeneous linear constraints on the coefficients of \( Q \).

The next two lemmas show why the conditions suffice, and that \( Q \) exists and can be found efficiently. The proofs are relatively simple. For the first substitutions \( Y_i = f^{(j-1)}(X) \) take place, and for the second it suffices to solve a homogeneous linear system imposed on the coefficients of \( Q \) with at most \( nm \) constraints. The details can be found in [GW11]. Once again, there is a resemblance between (2.7) an (6.3).

**Lemma 6.2.** Suppose \( Q \) of the form (2.4) satisfies (6.2). If the received word \( y \) agrees with the encoding (6.1) at location \( i \), i.e. \( f^{(j)}(\alpha_i) = y_{j+1} \) for \( 0 \leq j < m \) (row \( j+1 \) of \( y \)), then the polynomial

\[
\hat{Q}(X) := Q(X, f(X), f'(X), \cdots, f^{(s-1)}(X)) \text{ satisfies } \hat{Q}(\alpha_i) = 0 \text{ and } \hat{Q}^{(k)}(\alpha_i) = 0 \quad (6.3)
\]

for all \( k \in \mathbb{N}_{m-s} \), where \( \hat{Q}^{(k)}(X) \) is the \( k \)th derivative of \( \hat{Q} \).
Lemma 6.3. Let \( d \) be as in (2.5), except that \( N \) is replaced with our current block length \( n \). Then, a nonzero \( Q \) of the form (2.4) satisfying (6.2), with \( \deg(A_0) \leq d + k - 1 \) and \( \deg(A_i) \leq d \) for \( j \in \mathbb{N} \), exists, and can be found in \( O\left((nm)^3\right) \) field operations over \( \mathbb{F}_q \).

6.2 Retrieve candidate polynomials

Now that we know how to find a polynomial \( Q(X, Y) \) satisfying (6.2), it remains to list the polynomials \( f(X) \) which agree in sufficiently many locations with the received word \( y \). The following lemma gives an identity which should be satisfied by these candidate polynomials.

Lemma 6.4. If \( f(X) \in \mathbb{F}_q[X] \) has degree at most \( k - 1 \) and an encoding (6.1) agreeing with the received word \( y \) in at least \( t > \frac{d + k - 1}{m - s + 1} \) columns, then

\[
\hat{Q}(X) = Q(X, f(X), f'(X), \ldots, f^{(s-1)}(X)) = 0.
\]  

Lemma 6.4 is identical to lemma 2.4, with the only difference that we substitute \( f(\gamma^{i-1}X) \) with \( f^{(i-1)}(X) \) for all \( i \in \mathbb{N} \). With our choice of \( d \), it follows that any \( f(X) \) which agrees with \( y \) on

\[
t > \frac{d + k - 1}{m - s + 1} \geq \frac{n(m-s+1)-k+1}{m-s+1} + k - 1
\]

columns satisfies (6.4). Similarly to §2.2, our second step now is to find all polynomials \( f(X) \) of degree at most \( k - 1 \), such that

\[
\Xi(X) := A_0(X) + A_1(X)f(X) + A_2(X)f'(X) + \cdots + A_s(X)f^{(s-1)}(X) = 0
\]  

(6.5)

where \( A_i(X) = \sum_{j=0}^{\deg(A_i)} a_{ij}X^j \) for each \( i \). We view (6.5) as a system of linear equations over \( \mathbb{F}_q \) in the coefficients of \( f(X) = \sum_{i=0}^{k-1} f_iX^i \in \mathbb{F}_q[X] \), for which we note the following fact.

Fact 6.5. The solutions \((f_0, f_1, \cdots, f_{k-1})\) of (6.5) form an affine subspace of \( \mathbb{F}_q^k \).

The goal is almost identical to the one in §3. That is, we want to bound the dimension of the affine subspace of solutions of (6.5) by exposing its structure, and then use this to efficiently find an explicit basis. For this, it suffices to give an algorithm in the case that the constant term \( a_{s0} \) of \( A_s(X) \) is nonzero ([GW11] lemma 5), and we can then use lemma 6.6.

Lemma 6.6. If \( a_{s0} \neq 0 \), the affine solution space of (6.5) has dimension at most \( s - 1 \).

Proof. The proof idea is parallel to that of lemma 3.2. The coefficients of \( X^r \) of \( \Xi(X) \) equals

\[
\xi_r = a_{0r} + \left( a_{10} \cdot f_r + a_{11} \cdot f_{r-1} + \cdots + a_{1r} \cdot f_0 \right) + \left( a_{20} \cdot (r + 1) \cdot f_{r+1} + a_{21} \cdot r \cdot f_r + \cdots + a_{2r} \cdot f_1 \right) + \cdots + \left( a_{s0} \cdot \frac{(r + s - 1)!}{r!} \cdot f_{r+s-1} + \cdots + a_{s1} \cdot (s - 1)! \cdot f_{s-1} \right)
\]

\[
= a_{0r} + \sum_{j=1}^{s} \sum_{k=0}^{r} \frac{(k + j - 1)!}{k!} \cdot a_{j(r-k)} \cdot f_{k+j-1}.
\]
If \((f_0, \ldots, f_{k-1})\) is a solution to \(\Xi(X)\), then \(\xi_r = 0\) for every \(r\). For each \(r\); \(\xi_r\) depends only on \(f_j\) for \(j < r + s\), and the coefficient of \(f_{r+s-1}\) is
\[
a_{s0} \cdot (r + s - 1) \cdot (r + s - 2) \cdot \ldots \cdot (r + 1) = a_{s0} \cdot \frac{(r + s - 1)!}{r!}.
\]
By the assumption that \({\text{char}}(\mathbb{F}_q) > k\), it follows that this coefficient is nonzero when \(r + s \leq k\). Hence, if we fix \(\{f_i\}_{i=0}^{s-2}\), the rest of the coefficients \(\{f_i\}_{i=s-1}^{k-1}\) are uniquely determined. This implies that the dimension of the solution space is at most \(s - 1\).

What we showed implies the main result of [GW11] (its theorem 6 and corollary 7), which for parameters \(s \simeq 1/\varepsilon, m \simeq s^2\) suggests that \(\text{Der}_q^{(m)}[n,k]\) of rate at least \(R\) for \(R \in (0,1)\), can be list decoded from a fraction of \(1 - R - \varepsilon\) of errors, with a list-size of \(q^{O(1/\varepsilon)}\). Lastly, there is potential for improving the large list-size of derivative codes, by drawing codewords from subspace-evasive sets.

## 7 Concluding Remarks

In this survey we first saw how the gap was closed for the optimal trade-off between rate and error-correction capability for list decoding algorithmically, through folded Reed-Solomon codes. We then showed several ways in which this can be achieved, using very different approaches, but at their core same ideas, and attaining same results. This is not just impressive, but also important; as different point of views may clear any ambiguity and make things easier to interpret and understand. We also looked into local self-correction and local decodability of multiplicity codes.

Two list decoding algorithms which were not discussed, are the list decoding of Parvaresh-Vardy (PV) codes [PV05] which has decoding radius \(1 - \frac{M}{M^2 + 1} \sqrt{M^M R M^M}\) for an arbitrary parameter \(M \in \mathbb{Z}_+\), and the list decoding of multiplicity codes [Kop15] which achieves the list decoding capacity \(1 - R\). Chronologically, the first major breakthrough in this area was presented in [Sud97] which had radius \(1 - 2\sqrt{R}\), followed by the Guruswami-Sudan radius \(1 - \sqrt{R}\) [GS98], and then PV was a stepping stone between towards folded Reed-Solomon codes. The main ideas in all these achievements come from the seminal paper of Sudan, and a lot of what we presented relates to the construction of PV codes; in which certain powers of the evaluations of the polynomial \(f(X)\) are being bundled together. These codes have also been used in other applications, e.g. randomness extractors.

There are a lot more articles in this (general) area which were not discussed. We only bring to your attention two such articles. In [Gur09], the folding operation was extended to certain algebraic-geometry codes, which contain FRS is a special case. These codes are referred to as \textit{folded cyclotomic codes}. The second is [HN09], which uses a similar approach to what was discussed in §4, for folded versions of algebraic-geometric codes.

## References

[BAS14] Avraham Ben-Aroya and Igor Shinkar. A note on subspace evasive sets. \textit{Chicago Journal of Theoretical Computer Science}, 2014.
[BB09] Peter Beelen and Kristian Brander. Decoding folded reed-solomon codes using hensel-lifting. In Gröbner Bases, Coding, and Cryptography, pages 389–394. Springer, 2009.

[Ber70] Elwyn R Berlekamp. Factoring polynomials over large finite fields. Mathematics of computation, 24(111):713–735, 1970.

[Bra10] Kristian Brander. Interpolation and list decoding of algebraic codes. PhD Thesis, Technical University of Denmark (DTU), 2010.

[BW86] Elwyn R. Berlekamp and Lloyd R. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470, 1986.

[Car17] Xavier Caruso. Computations with p-adic numbers. Hyper Articles en Ligne, 2017.

[Con] Keith Conrad. Hensel’s Lemma. https://kconrad.math.uconn.edu/blurbs/gradnumthy/hensel.pdf.

[CS95] A Robert Calderbank and Neil JA Sloane. Modular and p-adic cyclic codes. Designs, codes and Cryptography, 6(1):21–35, 1995.

[DHP06] Steven T. Dougherty and Young Ho Park. A note on subspace evasive sets. Designs, Codes and Cryptography, pages 65–80, 2006.

[DKSS13] Zeev Dvir, Swastik Kopparty, Shubhangi Saraf, and Madhu Sudan. Extensions to the method of multiplicities, with applications to kakeya sets and mergers. SIAM Journal on Computing, 42(6):2305–2328, 2013.

[DL12] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pages 351–358. ACM, 2012.

[DS08] Zeev Dvir and Amir Shpilka. Noisy interpolating sets for low degree polynomials. In 2008 23rd Annual IEEE Conference on Computational Complexity, pages 140–148. IEEE, 2008.

[G+07] Venkatesan Guruswami et al. Algorithmic results in list decoding. Foundations and Trends® in Theoretical Computer Science, 2(2):107–195, 2007.

[Gou06] Fernando Q. Gouvêa. Arithmetic of p-adic modular forms, volume 1304. Springer, 2006.

[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.

[GRS19] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory, 2019. https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf.

[GS98] Venkatesan Guruswami and Madhu Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280), pages 28–37. IEEE, 1998.

[Gur09] Venkatesan Guruswami. Artin automorphisms, cyclotomic function fields, and folded list-decodable codes. In Proceedings of the forty-first annual ACM symposium on Theory of computing, pages 23–32. ACM, 2009.

[Gur10] Venkatesan Guruswami. List decoding Folded Reed-Solomon codes, 2010. http://www.cs.cmu.edu/~venkatg/teaching/codingtheory/notes/notes11.pdf.
A Digression Into Number Theory — $p$-adic numbers

We digress from coding theory in this appendix to further discuss Hensel’s lemma from §4, and its connection to the $p$-adic numbers $\mathbb{Q}_p$. In 1897, Kurt Hensel himself introduced the field of $p$-adic
numbers \( \mathbb{Q}_p \), which have been thoroughly studied throughout the 20th century and are still active research area, though they were foreshadowed in Ernst Kummer’s work a few decades earlier. The first major breakthrough involving \( p \)-adic numbers is the Hasse–Minkowski theorem, which can be used to test efficiently whether a Quadratic form has a solution in \( \mathbb{Q} \). In the literature, there are also examples of codes over the \( p \)-adic integers and numbers; e.g. [CS95],[DHP06].

Vaguely speaking, they allow the use of analytic methods in the study of Diophantine equations, number theory, arithmetic geometry and more recently, numerical analysis [Car17]. After all, Hensel’s main motivation was the analogy between the unique factorization domain (UFD) \( \mathbb{Z} \) along with its field of fractions \( \mathbb{Q} \), and the UFD \( \mathbb{C}[X] \) along with its field of fractions \( \mathbb{C}(X) \). Essentially, \( p \in \mathbb{Z} \) are analogous to the (irreducible) polynomials \((X - a) \in \mathbb{C}[X]\) [Ogg14]. Here is a definition of the \( p \)-adic integers \( \mathbb{Z}_p \), and two definitions of the \( p \)-adic numbers \( \mathbb{Q}_p \); an algebraic A.2 and an analytic A.4 (which may be viewed as a theorem). By \( p \) we indicate a fixed prime.

**Definition A.1.** A \( p \)-adic integer is a formal sum \( \alpha = \sum_{i=0}^{\infty} a_i p^i \), for integers \( 0 \leq a_i < p \). The set of \( p \)-adic integers \( \mathbb{Z}_p \), forms a commutative ring. We can alternatively write \( \alpha = \cdots a_i \cdots a_2 a_1 a_0 \).

**Definition A.2.** The \( p \)-adic numbers are the series of the form
\[
a_{-n} \frac{1}{p^n} + a_{-n+1} \frac{1}{p^{n-1}} + \cdots + a_{-1} \frac{1}{p} + a_0 + a_1 p + \cdots
\]
which form the field we denoted by \( \mathbb{Q}_p \). Furthermore \( \mathbb{Q} \subseteq \mathbb{Q}_p \), and if \( \alpha \in \mathbb{Q}_p \), then \( \exists N \geq 0 \) such that \( p^N \alpha \in \mathbb{Z}_p \). In other words, \( \mathbb{Q} \) may be viewed as a subfield of \( \mathbb{Q}_p \).

**Definition A.3.** Let \( \alpha \in \mathbb{Q}^* \) where \( \alpha = p^k \frac{a}{h} \) for \( k \in \mathbb{Z} \), and \( p, g, h \) coprime. The \( p \)-adic valuation of \( \alpha \) is \( \text{ord}_p(\alpha) = k \) and its \( p \)-adic absolute value is \( \nu_p(\alpha) = p^{-k} \), which is a non-Archimedean metric; as \( \nu_p(\beta + \gamma) \leq \max\{\nu_p(\beta), \nu_p(\gamma)\} \) for \( \beta, \gamma \in \mathbb{Q} \). By convention \( \text{ord}_p(0) = \infty \) and \( \nu_p(0) = 0 \).

**Definition A.4.** The field of \( p \)-adic numbers \( \mathbb{Q}_p \) is the completion of \( \mathbb{Q} \) with respect to the metric induced by \( \nu_p(\cdot) \), i.e. every \( \text{Cauchy sequence converges} \). Moreover, \( \mathbb{Q} \) is dense in \( \mathbb{Q}_p \) (as is \( \mathbb{Z} \) in \( \mathbb{Z}_p \)).

By definition \( \mathbb{Q}_p = \mathbb{Z}_p / \mathbb{Z}_p^1 \), and it is the fraction field of \( \mathbb{Z}_p \). Another definition which resembles Hensel’s lemma, is defined through the ring homomorphism
\[
\pi_n : \mathbb{Z}_p \rightarrow \mathbb{Z}/p^n \mathbb{Z}
\]
\[
\sum_{i=0}^{\infty} a_i p^i \mapsto \left( \sum_{i=0}^{n-1} a_i p^i \right) \mod p^n
\]
for which \( \pi_{n+1}(\alpha) = \pi_n(\alpha) \mod p^n \). This definition uses the projective/inverse limit, and is not relevant to what we want to show. We want to demonstrate the resemblance with Hensel’s lemma.

**Lemma A.5** (Basic version [Con]). If \( f(X) \in \mathbb{Z}_p[X] \) and \( a \in \mathbb{Z}_p \) satisfies \( f(a) \equiv 0 \mod p \) and \( f'(a) \neq 0 \mod p \), then there exists a unique \( \alpha \in \mathbb{Z}_p \) such that \( f(\alpha) = 0 \) and \( \alpha \equiv a \mod p \).

**Theorem A.6** (Stronger version [Con]). Let \( f(X) \in \mathbb{Z}_p[X] \) and \( a \in \mathbb{Z}_p \) satisfy \( \nu_p(f(a)) < \nu_p(f'(a))^2 \). Then, there is a unique \( \alpha \in \mathbb{Z}_p \) such that \( f(\alpha) = 0 \) and \( \nu_p(\alpha - a) < \nu_p(f'(a)) \). Moreover:

1. \( \nu_p(\alpha - a) = \nu_p(f(a)/f'(a)) < \nu_p(f'(a)) \)
2. \( \nu_p(f'(\alpha)) = \nu_p(f'(a)) \).

One can restate the above theorem in a way which gives a construction of the \( \alpha \in \mathbb{Z}_p \) [Car17]. The striking part about this statement (and the construction of \( \alpha \)), is that the proof applies Newton’s method; establishing connections now to numerical analysis. This is a (approximate) root-finding
algorithm, which takes us back to §2.2. The remarkable thing about Newton's method is that it extends almost word for word to Hensel's lemma A.6, when $\mathbb{R}$ is replaced by $\mathbb{Q}_p$. More precisely, under the assumptions of lemma A.6 we construct the sequence $(x_i)_{i \in \mathbb{N}_0}$ by the recurrence $x_0 = a$; $x_{i+1} = x_i - f(x_i)/f'(x_i)$, which converges to $\alpha \in \mathbb{Z}_p$ with $f(\alpha) = 0$.

The $p$-adics are relatively hard to grasp and understand, though they have “simple” constructions (e.g., lemma 4.1). Part of the reason is that there are many ways to interpret them, as we have seen. We briefly discuss a final more visual representation of $\mathbb{Z}_p$, which is more meaningful and convenient geometrically. From definition A.1, it is clear that any $\alpha \in \mathbb{Z}_p$ can be decomposed in base $p$. We can then construct a tree with $p$ branches at each node (a full $p$-ary tree), with each branch corresponding to an integer coefficient $0 \leq a_i < p$, and nodes at the same height $h$ correspond to elements of the congruence class $\mod p^{h+1}$ (height here corresponds to the depth of the tree).

Where does Hensel's lemma come into play? We point out that definitions 4.1 and A.5 are in fact the same, with the latter stated in a more abstract way.

Name anything $p$-adic and most likely it has already been well-defined and studied extensively, from $p$-adic differential equations [Ked10], to $p$-adic modular forms [Gou06] and $p$-adic $\zeta$-functions [Kob12]. The most common use of $p$-adics though, is probably in the study of elliptic curves. This is where they appear in the solution of one of the most important problems in mathematics, Fermat’s last theorem (specifically, the proof of the modularity conjecture for semistable elliptic curves). As a humbled mathematician said twenty-six years ago, ‘I think I’ll stop here’.