Introduction to Coding Theory **CMU: Spring 2010**

Notes 7: Justesen codes, Reed-Solomon and concatenated codes decoding

March 2010

Lecturer: V. Guruswami *Scribe: Venkat Guruswami & Balakrishnan Narayanaswamy*

1 Review - Concatenated codes and Zyablov's tradeoff

In the last class we saw (Theorem 14) that it is possible to *efficiently* construct an (asymptotically good) concatenated code of rate R with distance meeting the Zyablov trade-off between relative distance δ and rate R:

$$\delta_{\text{Zyablov}}(R) = \max_{R \le r \le 1} (1 - \frac{R}{r}) h^{-1}(1-r) \tag{1}$$

Given a specified rate target R, the construction involved a brute force search for the inner code C_{in} of rate r that met the Gilbert-Varshamov bound and had relative distance $h^{-1}(1-r)$. Once such an inner code was found, it was used to encode each symbol of the outer Reed-Solomon code.

This use of a search step left open the question of constructing fully explicit codes, with no brute-force search for a smaller code, with similar trade-offs.

2 Justesen's code

Justesen [9] provided an explicit concatenated code construction that achieved the Zyablov tradeoff over a range of rates (approximately rates $R \ge .31$) and was aymptotically good for any desired rate $R \in (0,1)$. The construction was based on the following insights:

1. A inner codes C_{in}^i do not have to be the same
2. It is sufficient if most (a fraction $1 - o(1)$) of the inner codes meet the Gilbert-Varshamov bound.

How can we exploit these insights ? We fix the outer code C_{out} as the $[n = 2^m - 1, k, n-k+1]_{2^m}$ Reed Solomon code $\text{RS}_{\mathbb{F},\mathbb{F}^*}[n,k]$ over the field $\mathbb{F} = \mathbb{F}_{2^m}$. Each symbol in the resulting code can be mapped (bijectively) into a binary sequence of length m using an $\mathbb{F}_2$-linear map $\sigma : \mathbb{F}_{2^m} \to \mathbb{F}_2^m$. We code each sequence i (for $i = 1, \ldots, n$) with a different binary rate half inner code C_{in}^i. That is each C_{in}^i maps the binary sequence of length m to a binary sequence of length $2m$. Suppose that most (at least $(1 - o(1))n$) of the inner codes C_{in}^i have a relative distance $\delta_i \ge \delta_g$.

This concatenated code has rate $\frac{R}{2} = \frac{k}{2n}$ and relative distance at least $(1 - 2R - o(1))\delta_g$. If most of the inner codes (almost) meet the GV bound for rate 1/2 codes, so that

$$\delta_g = \delta_{GV}(1/2) - o(1) = h^{-1}(1/2) - o(1) \ , \tag{2}$$

then the overall code has relative distance given by

$$\delta(R) = (1-2R)h^{-1}(1/2) - o(1) \ . \tag{3}$$

In the Exercise 1 below, we will construct a family of codes such that all but a small fraction of them asymptotically meet the GV bound.

Exercise 1 *Show that there is a family $\mathcal{F}$ of $[2m, m]_2$ binary linear codes such that the following hold*

1. $|\mathcal{F}| = 2^m - 1$
2. *Most of the codes $C \in \mathcal{F}$ have relative distance at least $h^{-1}(1/2) - o(1)$*

*(***Hint***: Consider the code family from HW 1, Question 5.b. For $\alpha \in \mathbb{F}_{2^m}$, $\alpha \neq 0$, consider the map $L_\alpha : \mathbb{F}_2^m \to F_2^{2m}$ defined as*

$$L_\alpha(\mathbf{x}) = (\mathbf{x}; \sigma(\alpha \bullet \sigma^{-1}(\mathbf{x}))) \tag{4}$$

and the family of codes $\mathcal{F} = \{L_\alpha : \alpha \neq 0\}$. We showed that there exist codes in this family that asymptotically meet the GV bound. The same argument can actually show that this is true for most codes in the family.)

Using these codes as the different inner codes C_{in}^i's with outer code $\mathrm{RS}_{\mathbb{F},\mathbb{F}^*}[n,k]$ gives us the following result:

Lemma 1 *There are explicit binary linear codes of rate $R < 1/2$ and relative distance*

$$(1-2R)h^{-1}(\frac{1}{2}) - o(1) \ . \tag{5}$$

Remark 2 *In the above concatenated code, the message is a polynomial $f \in \mathbb{F}_{2^m}[X]$ of degree at most $k-1$ and it is encoded by the evaluations of both $f(X)$ and $Xf(X)$ at the nonzero elements of the field, which are then expressed as elements of $\mathbb{F}_2^m$ as per some basis.*

We see that the above only gives codes of overall rate less than $1/2$ (since the inner code itself has rate only $1/2$). For larger rates we modify the construction in Exercise 1 to construct a small family of codes of any desired rate $r \in (1/2, 1)$ that meet the GV bound.

Exercise 2 *For any $1 \leq s \leq m$, show that there is a family $\mathcal{F}$ of $[m+s, m]_2$ binary linear codes such that the following hold*

1. $|\mathcal{F}| = 2^m - 1$
2. *Most of the codes $C \in \mathcal{F}$ have relative distance at least $h^{-1}(s/(m+s)) - o(1)$*

(**Hint**: *Modify the code family from Exercise 1 as follows. Define* $\sigma' : \mathbb{F}_{2^m} \to F_2^s$ *such that* $\sigma'(\mathbf{x}) =$ *first s bits of* $\sigma(\mathbf{x})$. *For* $\alpha \in \mathbb{F}_{2^m}$, $\alpha \neq 0$, *consider the map* $L'_\alpha : \mathbb{F}_2^m \to F_2^{m+s}$ *defined as*

$$L'_\alpha(\mathbf{x}) = (\mathbf{x}; \sigma'(\alpha \bullet \sigma^{-1}(\mathbf{x}))) \tag{6}$$

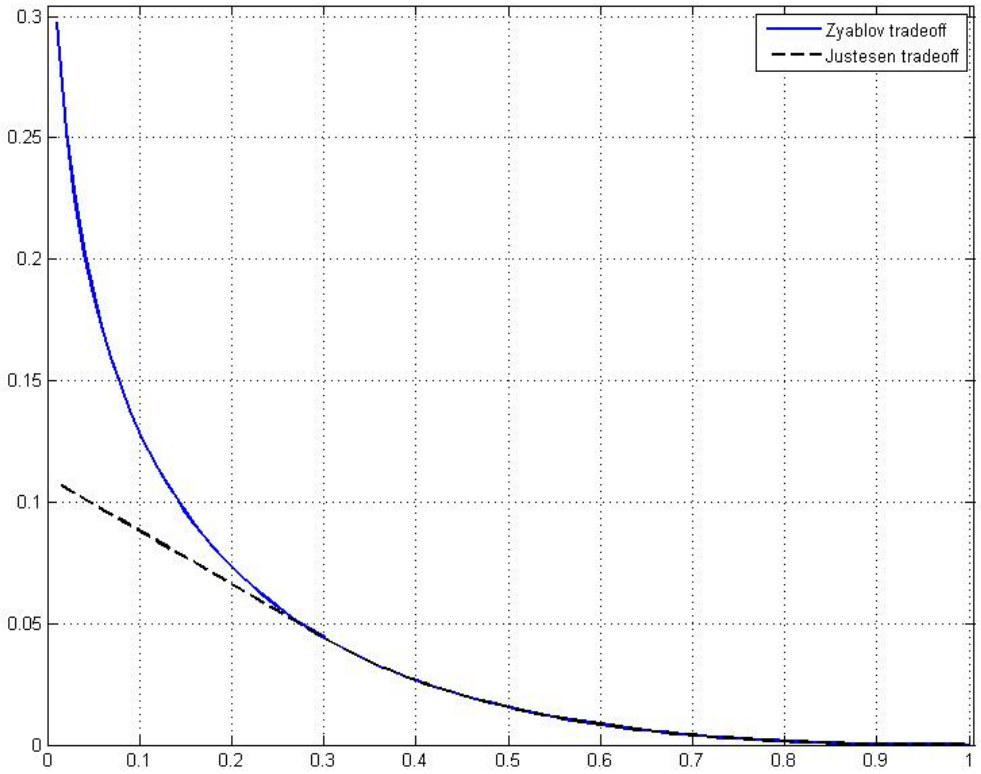
and the family of codes $\mathcal{F} = \left\{ L'_\alpha : \alpha \neq 0 \right\}$. *Use volume arguments to show that most codes in this family have relative distance* $\geq h^{-1}(\frac{s}{s+m}) - o(1)$.

By concatenating the outer Reed-Solomon code of length $2^m - 1$ with all the different codes in the above family, we get the following:

Theorem 3 *For any* $R \in (0, 1)$, *there are explicit binary lineat codes of rate at least* R *and relative distance at least*

$$\delta_{\text{Justesen}}(R) = \max_{r \geq \max(\frac{1}{2}, R)} (1 - \frac{R}{r}) h^{-1}(1 - r) - o(1) \tag{7}$$

We compare the two bounds constructive results: the Zyablov tradeoff (Eq. 1) and the explicit construction Justesen tradeoff (Eq. 7) in the figure below.



Exercise 3 *Show that the* $\delta_{Zyablov}(R) = \delta_{Justesen}(R)$ *for* $R \geq .31$.

Hint: *Differentiating the expression* $(1 - R/r)h^{-1}(1 - r)$ *w.r.t* r*, one finds that the* r *maximizing this expression (and thus leading to the Zyablov trade-off in (1)) satisfies*

$$R = \frac{r^2}{1 + \log_2 (1 - h^{-1}(1 - r))} \ .$$

For $R \geq 0.31$*, the solution* r *to the above equation lies in the range* $[1/2, 1]$*, and thus the Zyablov bound can be met by the Justesen construction.*

2.1 Meeting the Zyablov trade-off at lower rates

The Justesen construction in the previous section used a good ensemble of codes of rate $r = \frac{1}{2}$ for the inner code, with the size of the ensemble $|\mathcal{F}| < 2^m$. The reason that the same construction does not work for rates $R < .31$ is that we do not know small enough inner code ensembles with rate $r < \frac{1}{2}$ where most of the codes are meet the GV bound.

For example, for $R \approx 0.15$, the optimal choice of the inner rate r in the Zyablov bound is $\approx 1/3$. Consider the following map, similar to the ones used in Exercises 1 and 2,

$$L_{\alpha_1,\alpha_2}(\mathbf{x}) = (\mathbf{x}; \sigma(\alpha_1 \bullet \sigma^{-1}(\mathbf{x})); \sigma(\alpha_2 \bullet \sigma^{-1}(\mathbf{x}))) \tag{8}$$

and the family of codes defined by this map for all pairs of nonzero field elements α_1, α_2. One can show, via a similar counting argumemt, that this is a family of at most 2^{2m} codes where most of the codes meet the GV bound and have relative distance $\delta = h^{-1}(\frac{2}{3}) - o(1)$.

To use these rate $1/3$ codes in a Justesen-like construction, we need an outer code over alphabet $\mathbb{F}_{2^m}$ that has block length about 2^{2m}, and which nearly meets the Singleton bound. Reed-Solomon codes are limited to a block length of 2^m and thus are not long enough. The solution around this predicament is provided by algebraic-geometric (AG) codes. As we briefly mentioned earlier, AG codes over $\mathbb{F}_q$ are a generalization of RS curves based on evaluation of functions with few "poles" at the rational points of an appropriately chosen algebraic curve which has $\gg q$ points with coordinates in $\mathbb{F}_q$. Shen [12] proposed an explicit family of AG codes over $\mathbb{F}_q$ which can have block length at least q^c for any desired c, and whose relative distance as a function of the rate R is $1 - R - o_q(1)$. Using these codes as outer codes in place of RS codes and the appropriate extension of the above rate $1/3$ code ensemble in a Justesen-like construction, he was able to give an explicit construction achieving the Zyablov trade-off for the entire range of rates $R \in (0, 1)$. Discussing the details of these AG codes are beyond the scope of this course, and the interested reader can find the details in the original paper [12].

3 Decoding algorithms

We now turn to algorithmic aspects of error-correction. We first consider the (relatively benign) erasure channel and then move on to channels that arbitrarily corrupt a constant fraction of the transmitted codeword symbols. We will see that Reed-Solomon codes admit efficient decoding algorithms matching the combinatorial bounds possible by virtue of its minimum distance.

3.1 Erasure decoding of linear codes

Consider a $[n, k, d]_q$ code with message $x \in \mathbb{F}_q^k$ and corresponding codeword $y = Gx \in \mathbb{F}_q^n$. Suppose a subset $S \subseteq \{1, \ldots, n\}$ is received (uncorrupted) and the rest of the positions are erased. The location of the erasures are known at the receiving end. Decoding the message involves solving the linear system

$$G_S x = y_S \tag{9}$$

We reorder the indices so that the first $|S|$ entries in y are the uncorrupted entries, resulting in the following matrix system,

$$\left[\begin{array}{c} G_S \\ \hline G_{\overline{S}} \end{array}\right] \left[\begin{array}{c} \\ x \\ \\ \end{array}\right] = \left[\begin{array}{c} y_S \\ \hline y_{\overline{S}} \end{array}\right] \tag{10}$$

As long as the rank of G_S is k, the solution to $G_S x = y_S$ is uniquely determined. When the number of erasures is less than d, i.e., $|S| > n - d$, the distance property of the code implies that G_S has full rank. Thus one can correct any pattern of $d - 1$ or fewer erasures by solving the linear system $G_S x = y_S$ in $O(n^3)$ time. However, specific structured G matrices can allow much faster solution of the linear system (even linear time in certain cases as we will see later on when discussing expander codes), leading to faster erasure recovery algorithms.

3.2 Erasure decoding of Reed Solomon Codes

We recall the interpretation of Reed Solomon codes from the previous lecture,

Definition 4 *Reed-Solomon codes*
For integers $1 < k < n$, field $\mathbb{F}$ of size $|\mathbb{F}| > n$, and a set $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}$, we define the Reed-Solomon code

$$RS_{\mathbb{F},S}[n, k] \quad = \quad \{(p(\alpha_1), \ldots, p(\alpha_n)) \in \mathbb{F}^n | p \in \mathbb{F}[X] \text{ a polynomial of degree} \leq k - 1\} \tag{11}$$

To encode a message $m = (m_0, \ldots, m_{k-1}) \in \mathbb{F}^k$, we interpret the polynomial as

$$p(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-1} \in \mathbb{F}[X] \tag{12}$$

Suppose a codeword from a Reed Solomon code is transmitted over an erasure channel and all but t symbols are erased. Then the decoder must reconstruct the message m from t pairs of values $\{(\alpha_1, f(\alpha_1)), \ldots, (\alpha_t, f(\alpha_t))\}$. Since the polynomial $p(X)$ is a degree $k-1$ polynomial it is uniquely determined by its value at any $t \geq k$ points. This can be done using FFTs in $n \log^{O(1)} n$ time or using polynomial interpolation in $O(n^2)$ time. Suppose $t = k$ and nonerased locations correspond to $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. Note that if we define the polynomials p_j for $1 \leq j \leq k$

$$p_j(X) = \prod_{i=1, i \neq j}^{k} \frac{X - \alpha_i}{\alpha_j - \alpha_i} \ , \tag{13}$$

the interpolated polynomial is then

$$f(X) = \sum_{j=1}^{k} f(\alpha_j) p_j(X) \ . \tag{14}$$

The number $n-k$ of erasures corrected by RS codes is optimal, since to have any hope of recovering the k message symbols, one must receive at least k symbols at the receiving end.

3.3 Decoding Reed-Solomon codes from errors

We now turn to the more challenging problem of decoding Reed-Solomon codes from worst-case errors. Specifically, we would like to decode the RS code RS$[n, k]$ up to $\tau = \lfloor \frac{n-k}{2} \rfloor$ errors. (Recall that the code has distance $n - k + 1$, so the correct codeword is uniquely determined as the closest codeword to the received word if up to τ errors occur.)

Suppose a polynomial $f \in \mathbb{F}_q[X]$ of degree $k-1$ is encoded as the RS codeword $(f(\alpha_1), \ldots, f(\alpha_n))$ and transmitted, but it is received as the noisy word $y = (y_1, \ldots, y_n)$ satisfying $y_i \neq f(\alpha_i)$ for at most τ values of i. The goal is to recover the polynomial $f(X)$ from y.

We now discuss an algorithm due to Welch and Berlekamp [15] for solving this problem. (The streamlined and simplified presentation discussed here is due to Gemmell and Sudan [7].) Note that if we knew the location of the errors, i.e., the set $E = \{i \mid y_i \neq f(\alpha_i)\}$, then the decoding is easy, as we can erase the erroneous and interpolate the polynomial on the rest of the locations.

To this end, let us define the error locator polynomial (which is unknown to the decoder):

$$E(X) = \prod_{f(\alpha_i) \neq y_i} (X - \alpha_i) \tag{15}$$

The degree of $E(X)$ is $\leq \tau$. Clearly $E(X)$ has the property that for $1 \leq i \leq n$, $E(\alpha_i) y_i = E(\alpha_i) f(\alpha_i)$. Define the polynomial

$$N(X) = E(X) F(X) \ , \tag{16}$$

which has degree at most $\tau + k - 1$. Now the bivariate polynomial

$$P(X, Y) = E(X) Y - N(X) \tag{17}$$

satisfies $P(\alpha_i, y_i) = 0, \forall i$. We will use the existence of such a P to find a similar bivariate polynomial from which we can find $f(X)$.

Formally, the algorithms proceeds in two steps.

Step 1 : Find a non-zero polynomial $Q(X, Y)$ such that,

1. $Q(X, Y) = E_1(X) Y - N_1(X)$
2. $deg\ E_1 \leq \tau$ and $deg\ N_1 \leq \tau + k - 1$
3. $Q(\alpha_i, y_i) = 0, \forall i$

Step 2 : Output $\frac{N_1(X)}{E_1(X)}$ as $f(X)$

Proposition 5 *A non-zero solution Q to Step 1 exists.*

PROOF: Take $E_1 = E$, $N_1 = N$. $\square$

Proposition 6 *Any solution (E_1, N_1) must satisfy $\frac{N_1}{E_1} = f$.*

PROOF: Define the polynomial

$$R(X) = E_1(X)f(X) - N_1(X) \tag{18}$$

Fact 1: *deg* $R \leq \tau + k - 1$. This follows immediately from the conditions imposed on the degree of E_1 and N_1.
Fact 2: R has at least $n - \tau$ roots. Indeed, for each locations i that is not in error, i.e., $f(\alpha_i) = y_i$, we have $R(\alpha_i) = Q(\alpha_i, y_i) = 0$.

Using the above two facts, we can conclude that if $n - \tau > \tau + k - 1$, then R is identically 0, which means that $f(X) = N_(X)/E_1(X)$. Since $\tau = \lfloor \frac{n-k}{2} \rfloor$, this condition on τ is met, and we conclude that the algorithm successfully recovers $f(X)$. $\square$

We now argue that the above algorithm can be implemented in polynomial time. Clearly the second step is easy. For the first interpolation step, note that it can be solved by finding a non-zero solution to a homogeneous linear system with unknowns being the coefficients of the polynomials N_1, E_1, and n linear constraints $Q(\alpha_i, y_i) = 0$. Since we guaranteed the existence of a nonzero solution, one can find some nonzero solution by Gaussian elimination in $O(n^3)$ field operations.

The interpolation step is really rational function interpolation and near-linear time algorithms are known for it. Also one can do fast polynomial division in $n \log^{O(1)} n$ field operations. Thus overall the algorithm can also be implemented in near-linear time.

We conclude by recording the main result concerning decoding RS codes up to half the distance.

Theorem 7 *There is a polynomial time decoding algorithm for an $[n, k]_q$ Reed-Solomon code that can correct up to $\lceil \frac{n-k}{2} \rceil$ worst-case errors.*

Thus, for a given rate R we can correct a $\frac{1-R}{2}$ fraction of errors (using RS codes and the above algorithm) which is the best possible by the Singleton bound. Later in the course we will look at list decoding algorithms that can improve on these parameters by allowing the decoder to output a (small) list of candidate codewords.

The primary disadvantage of RS codes are that they are defined over very large alphabets (of size at least the codeword length). We will soon see efficiently decodable binary codes constructed via code concatenation. But next we see a different algorithm for decoding RS codes up to half the distance, which was historically the first such algorithm.

4 Peterson Algorithm for decoding RS codes

The Peterson Algorithm from 1960 [11] is another algorithm to decode Reed Solomon codes up to half the minimum distance. One interesting feature of its discovery is that it is a non-trivial polynomial time algorithm for a non-trivial problem proposed *before* polynomial time was formalized as the theoretical standard for efficient computation!

The Peterson Algorithm works with the parity check view of RS codes which we recall here

Definition 8 (Parity-check characterization) *For integers $1 \leq k < n$, a field $\mathbb{F}$ of size $|\mathbb{F}| = q = n+1$, a primitive element $\alpha \in \mathbb{F}_q^*$, and the set $S = \{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, the $[n, k, n-k+1]_q$ Reed-Solomon code over $\mathbb{F}$ with evaluation set S is given by*

$$\begin{aligned} \mathrm{RS}_{\mathbb{F},S}[n,k] &= \{(c_0, c_1, ..., c_{n-1}) \in \mathbb{F}^n | c(X) = c_0 + c_1 X + \ldots + c_{n-1} X_{n-1} \\ &\quad satisfies\ c(\alpha) = c(\alpha^2) = \ldots = c(\alpha^{n-k}) = 0\} \end{aligned}$$

Suppose a codeword $c \in \mathrm{RS}_{\mathbb{F},S}[n,k]$ is transmitted and an error vector $e \in \mathbb{F}^n$ of Hamming weight at most

$$\tau = \left\lfloor \frac{n-k}{2} \right\rfloor$$

is added to c, so that it is received as $y = c + e = (y_0, y_1, \ldots, y_{n-1})$. The goal is to efficiently recover c (or the polynomial $f \in \mathbb{F}[X]$ of degree $< k$ that it encodes).

For $l \in \{1, \ldots, n-k\}$, we can compute the syndrome

$$S_l = \sum_{j=0}^{n-1} y_j \alpha^{lj} = c(\alpha^l) + \sum_{j=0}^{n-1} e_j \alpha^{lj} = \sum_{j=0}^{n-1} e_j \alpha^{lj} \tag{19}$$

since $c(\alpha^l) = 0$ for $1 \leq l \leq n-k$. Let us define the *syndrome polynomial* $S(X)$

$$S(X) = \sum_{l=1}^{n-k} S_l X^{l-1} \ . \tag{20}$$

This polynomial and its properties play a key role in the development and analysis of the decoding algorithm. Note that the syndrome polynomial can be efficiently computed from the received word.

We define $T \subseteq \{0, 1, \ldots, n-1\}$ be the set of error locations, i.e., $T = \{i \mid e_i \neq 0\}$. Define the Error Locator Polynomial as follows (note that T and the error locator polynomial are *not* known at the decoder, and computing them is at the heart of the decoding task).

$$E(X) = \prod_{j \in T} (1 - \alpha^j X) \tag{21}$$

This polynomial is defined so that it has roots exactly at α^{-j} for those indices j where the received vector y is in error. The degree of E is $|T| \leq \tau$.

We now simplify the expression for the syndrome polynomial:

$$\begin{aligned} S(X) &= \sum_{l=1}^{n-k} S_l X^{l-1} \\ &= \sum_{l=1}^{n-k} X^{l-1} \sum_{j \in T} e_j \alpha^{lj} \\ &= \sum_{j \in T} e_j \alpha^j \sum_{l=1}^{n-k} X^{l-1} \alpha^{j(l-1)} \\ &= \sum_{j \in T} e_j \alpha^j \left(\frac{1 - (\alpha^j X)^{n-k}}{1 - \alpha^j X} \right) \end{aligned}$$

Hence

$$E(X)S(X) = \Big(\prod_{j \in T} (1 - \alpha^j X) \Big) S(X) = \sum_{j \in T} e_j \alpha^j \big(1 - (\alpha^j X)^{n-k} \big) \prod_{i \in T, i \neq j} (1 - \alpha^i X) \ .$$

Defining

$$\Gamma(X) = \sum_{j \in T} e_j \alpha^j \prod_{i \in T, i \neq j} (1 - \alpha^i X)$$

leads us to the equation (often called the *Key Equation*)

$$E(X)S(X) \equiv \Gamma(X) \pmod{X^{n-k}} \tag{22}$$

We would like to use the above equation to solve for $E(X)$. Once we do that we can find the roots of E to determine the error locations, and then find the message polynomial by interpolation using the non-erroneous locations. Note that in the Key Equation, we know $S(X)$ but do not know either $E(X)$ or $\Gamma(X)$. However, we observe the following property of $\Gamma(X)$: it has degree at most $\tau - 1$. Therefore the Key Equation implies that the coefficients of the X^j for $\tau \leq j \leq n-k-1$ in $E(X)S(X)$ all equal 0. The decoder will use this to find the coefficients of $E(X)$ by solving a homegeneous linear system. Specifically, the algorithm will solve for unknowns a_i where $E(X) = 1 + \sum_{i=1}^{\tau} a_i X^i$.

The decoding algorithm proceeds as follows. (Since the system solved by the decoder could have multiple solutions, we will denote the candidate error locator polynomial found by the decoder as E_1 and later prove that E_1 must be divisible by E, which suffices to locate all the erroneous positions.)

Step 1: Compute the syndrome polynomial $S(X)$.

Step 2: Solve for $\{a_i\}_{i=1}^{\tau}$ to find $E_1(X) = 1 + \sum_{i=1}^{\tau} a_i X^i$ such that the coefficients of X^j in $E_1(X)S(X)$ all equal 0 for $j = \tau, \tau+1, \ldots, n-k-1$. (This is a system of $n-k-\tau$ homogeneous linear equations in the a_i's.)

Step 3: Find all roots of $E_1(X)$ using brute force search (we also later describe an optimization called Chien search which is faster for hardware implementations.) Suppose the roots are $\alpha^{-i_1}, \ldots, \alpha^{-i_l}$ for some $l \leq \tau$.

Step 4: Erase the locations $\{i_1, i_2, \ldots, i_l\}$ in the received word y and then interpolate a degree $< k$ polynomial $f(X)$ through the unerased positions. If this is not possible, declare decoding failure. Otherwise, return $f(X)$ as the message polynomial.

We now prove that assuming at most τ errors occurred, the polynomial $E_1(X)$ found in Step 2 will be divisible by $E(X)$ and hence all the error locations will be roots of $E_1(X)$ as well.

Proposition 9 *The error locator polynomial $E(X)$ divides the polynomial $E_1(X)$ found by the algorithm.*

PROOF: Recall that $E(X) = \prod_{j \in T}(1 - \alpha^j X)$. In general polynomials may not have a multiplicative inverse modulo X^{n-k} (for example the polynomial X has no inverse $\mod X^{n-k}$). However, $E(X)$ has an inverse modulo X^{n-k}, namely

$$E^{-1}(X) = \prod_{j \in T}(1 + \alpha^j X + \ldots + (\alpha^j X)^{n-k-1}).$$

Therefore we can solve for the syndrome (modulo X^{n-k}) as

$$S(X) \equiv \Gamma(X) E^{-1}(X) \pmod{X^{n-k}} . \tag{23}$$

Let $\Gamma_(X)$ be the polynomial of degree at most $\tau - 1$ such that

$$E_1(X) S(X) \equiv \Gamma_1(X) \pmod{X^{n-k}} . \tag{24}$$

Combining (23) and (24), we get

$$E_1(X)\Gamma(X) \equiv E(X)\Gamma_1(X) \pmod{X^{n-k}} .$$

Both the polynomials $E_1(X)\Gamma(X)$ and $E(X)\Gamma_1(X)$ have degree at most $\tau + (\tau - 1) = 2\tau - 1 \leq n - k - 1$, therefore we can conclude that they are in fact equal as polynomials:

$$E_1(X)\Gamma(X) = E(X)\Gamma_1(X) . \tag{25}$$

We now note that $\gcd(E(X), \Gamma(X)) = 1$. This follows from the following two observations: (i) the elements α^{-j} for $j \in T$ are *all* the roots of $E(X)$, and (ii) by definition of $\Gamma(X)$, it follows that $\Gamma(\alpha^{-j}) = e_j \prod_{i \in T; i \neq j}(\alpha^j - \alpha^i) \neq 0$. By (25) we know that $E(X)$ divides $E_1(X)\Gamma(X)$, and since $\gcd(E(X), \Gamma(X)) = 1$, we conclude that $E(X)$ must divide $E_1(X)$. □ Since both E and E_1 have constant term equal to 1, it follows from Proposition 9 that if instead of finding a degree τ polynomial $E_1(X)$, we try to find a polynomial of degree $1, 2, \ldots, \tau$ successively till the algorithms succeeds, then in fact we will recover the error locator polynomial $E(X)$ as $E_1(X)$. This is an alternate way to implement the algorithm, which will in fact be faster when the number of errors is much smaller than τ.

Corollary 10 *If we find $E_1(X)$ of the smallest degree that satisfies the Key Equation E_1 (22) then, $E_1(X) = E(X)$.*

Remark 11 *We can find the roots of $E_1(X)$ in Step 3 of the Peterson Algorithm by brute force search over all elements of the field checking if $E_1(\alpha^j) = 0$ for $j = 0, 1, 2, \ldots, n-1$. An optimization called "Chien search" leads to more practical hardware implementations. Chien search [4] is based on the following observation (our description is from [16])*

> *If $E_1(X)$ has degree τ, then $E_1(\alpha^{i+1})$ can be computed from $E_1(\alpha^i)$ by τ multiplications of variables with **constants** as opposed to τ multiplications of variables with **variables** needed in the brute force search.*

This is because of the following relationship between the evaluations $E_1(\alpha^i)$ and $E_1(\alpha^{i+1})$. If

$$\begin{aligned} E_1(\alpha^i) &= e_0 + e_1\alpha^i + \ldots + e_\tau\alpha^{i\tau} = \gamma_{0,i} + \gamma_{1,i} + \ldots + \gamma_{\tau,i} \ , \ \textit{then} \\ E_1(\alpha^{i+1}) &= e_0 + e_1\alpha^{i+1} + \ldots + e_\tau\alpha^{(i+1)\tau} = \gamma_{0,i} + \gamma_{1,i}\alpha + \ldots + \gamma_{\tau,i}\alpha^\tau \ . \end{aligned}$$

Exercise 4 (Forney's formula) *Once the error locations T are computed, show that the error values are given by the following formula. For every $j \in T$,*

$$e_j = \frac{-\alpha^j \Gamma(\alpha^{-j})}{E'(\alpha^{-j})} \tag{26}$$

where $E'(X)$ is the derivative of $E(X)$. This can be used in place of the interpolation step for recovering the codeword.

Remark 12 *For binary BCH codes, once we know the exact error locations (say by finding a solution $E_1(X)$ with smallest degree), we can just flip those locations to get the true codeword, and no separate step is needed to compute the error values.*

Remark 13 (Complexity the algorithm) *The naive implementation of the algorithm takes cubic time, with the dominant step being solving the linear system to find $E_1(X)$. Sugiyama, Kasahara, Hirasawa and Namekawa [14] gave a quadratic time implementation using Euclid's algorithm for solving the Key equation.*

Berlekamp [2] gave an iterative quadratic time implementation of the solution to the key equation, which was later generalized by Massey [10] to give an algorithm for finding the shortest linear feedback shift register (LFSR) that generates a given sequence. This method is now referred to as the Berlekamp-Massey algorithm and is widely used in practice for decoding Reed-Solomon codes. Blahut [3] gave a method to accelerate the Berlekamp-Massey algorithm through recursion, resulting in a near-linear ($n \log^{O(1)} n$) time algorithm for decoding RS codes.

5 Decoding concatenated codes

5.1 A Naive algorithm

As we noted at the beginning of this class, RS codes require large alphabets and are therefore not suited for applications where we need, for example, binary codes. We also saw in previous

lectures the method of code concatenation which can be used to reduce the alphabet size to while preserving a good rate vs distance trade-off. We now consider efficient decoding algorithms for decoding concatenated codes with outer Reed-Solomon code. We start with a naive decoder, and later improve upon it.

Suppose that x is the message to be transmitted, coded with a Reed-Solomon outer code C_{out} to obtain symbols $c_1, \ldots, c_n$. Each of these is coded with an inner code C_{in} resulting in the transmitted codeword, whose i'th block is $C_{in}(c_i)$. As before assume that the outer code has distance D and inner codes have distance d. In theory, we should be able to correct up to $\frac{dD}{2}$ errors since the distance of the concatenated code is at least dD.

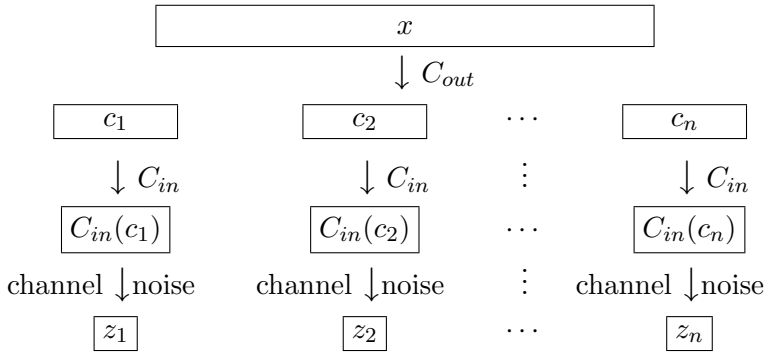


Figure 1: Decoding concatenated codes.

Suppose (due to errors) we receive z_i for $i = 1, \ldots, n$ instead of $C_{in}(c_i)$. Note that the total number of errors introduced equals $\sum_{i=1}^{n} \Delta(C_{in}(c_i), z_i)$. A simple decoder tries to reverse the encoding as follows.
Step 1 : Find a_i such that $\Delta(C_{in}(a_i), z_i)$ is minimized.
Step 2 : Decode $(a_1, \ldots, a_n)$ as per the decoder for the outer code C_{out} which can correct $< D/2$ errors.

Lemma 14 *The above algorithm recovers the correct codeword if the number of errors is less than* $\frac{dD}{4}$.

PROOF: Note that for each i for which $< \frac{d}{2}$ errors occur in i^{th} block, the inner decoding succeeds and we have $a_i = c_i$. If less than $\frac{dD}{4}$ errors occur in total, then the number of blocks with at least $d/2$ errors is less than $\frac{D}{2}$. Therefore the string $(a_1, \ldots, a_n)$ passed to the outer decoder differs from the true outer codeword $(c_1, c_2, \ldots, c_n)$ in $< D/2$ locations. The outer decoder then succeeds is recovering $(c_1, \ldots, c_n)$. □

5.2 Generalized Minimum Distance (GMD) Decoding of concatenated codes : Randomized version

Forney [6] gave a better algorithm for decoding that can correct a number of errors $< \frac{dD}{2}$ errors. This method is called Generalized Minimum Distance (GMD) decoding, and is based on using an errors-and-erasures decoder for the outer code. Recall that the Welch-Berlekamp algorithm that we discussed for Reed-Solomon can handle τ errors and s erasures as long as $2\tau + s < D$.

The naive algorithm discussed above is sub-optimal because all the guesses a_i from the inner decoder are treated on equal footing regardless of how far their inner encodings were from z_i. Intuitively, the outer decoder should place higher confidence in symbols whose inner encodings are close to z_i. This is achieved by assigning a measure of confidence to each a_i and erasing symbols whose confidence is below some threshold. Running such an algorithm for various choices of the threshold leads gives the GMD algorithm.

We first present a randomized version of GMD decoding that is easier and more intuitive to analyze, and then discuss how the algorithm can be derandomized to give a deterministic version. Below is the algorithm description.

Step 1: Decode z_i to a_i as before by finding the a_i that minimizes $\Delta(C_{in}(a_i), z_i)$

Step 2: Set $w_i = \min[\Delta(C^i_{in}(a_i), z_i), \frac{d}{2}]$.

Step 3: With probability $\frac{2w_i}{d}$ set a_i to be an erasure symbol '?'.

Step 4: Run the errors-and-erasures decoder (say Welch-Berlekamp in the case of Reed-Solomon codes) on the resulting sequence of a_is and ?s.

We assume that the total number of errors, $\sum_{i=1}^{n} \Delta(C_{in}(c_i), z_i) < dD/2$ and study the conditions under which this algorithm successfully decodes to the true codeword. Define τ as the number of errors and s as number of erasures in the outer codeword after the (randomized) inner decoding.

Lemma 15 *$\mathbb{E}[2\tau + s] < D$ where the expectation is taken over the random choices of erasures.*

PROOF: Let Z_i^{err} and $Z_i^{erasures}$ be indicator random variables for the occurrence of an error and declaration of an erasure respectively in the decoding of the i'th block z_i. We have

$$\tau = \sum_{i=1}^{n} Z_i^{err} \qquad s = \sum_{i=1}^{n} Z_i^{erasures} .$$

Claim 16 *$E[2Z_i^{err} + Z_i^{erasures}] < \frac{2e_i}{d}$, where $e_i = \Delta(z_i, C_{in}(a_i))$ is the number of errors introduced by the channel in the i^{th} block introduced by the channel.*

Note that once we prove the claim, by linearity of expectation

$$\mathbb{E}[2\tau + s] \leq \frac{2\sum_{i=1}^{n} e_i}{d} < D ,$$

so that Lemma 15 would be proved. □

PROOF: *(Of Claim 16)* We consider two cases.

Case 1 : $a_i = c_i$ (i'th block is correctly decoded)
In this case, trivially $E[Z_i^{err}] = 0$ and

$$\mathbb{E}[Z_i^{erasures}] = \Pr[Z_i^{erasures} = 1] = \frac{2w_i}{d} = \frac{2e_i}{d}$$

since $w_i = \min\{\Delta(C_{in}(a_i), z_i), \frac{d}{2}\} = \min\{e_i, \frac{d}{2}\} \leq e_i$. Thus $\mathbb{E}[2Z_i^{err} + Z_i^{erasures}] \leq \frac{2e_i}{d}$.

Case 2 : $a_i \neq c_i$ (i'th block is incorrectly decoded)
In this case

$$\mathbb{E}[Z_i^{erasures}] = \frac{2w_i}{d} \quad \text{and} \quad \mathbb{E}[Z_i^{err}] = 1 - \frac{2w_i}{d}$$

so that

$$\mathbb{E}[2Z_i^{err} + Z_i^{erasures}] = 2 - \frac{2w_i}{d} \ . \tag{27}$$

Since $a_i \neq c_i$, we have

$$d \leq \Delta(C_{in}(c_i), C_{in}(a_i)) \leq \Delta(C_{in}(c_i), z_i) + \Delta(z_i, C_{in}(a_i)) = e_i + \Delta(z_i, C_{in}(a_i)) \ .$$

Thus if $w_i = \Delta(z_i, C_{in}(a_i))$, then $w_i + e_i \geq d$. On the other hand if $w_i = d/2$, then $e_i \geq w_i \geq d/2$, so $w_i + e_i \geq d$ as well. Thus $w_i \geq d - e_i$. Plugging this into (27) we get $\mathbb{E}[2Z_i^{err} + Z_i^{erasures}] \geq 2e_i/d$ as desired.

The two cases together complete a proof of the claim. □

5.3 GMD Decoding: Deterministic version

We now see how to derandomize the previously presented GMD decoder. We first recall that the randomness was used when we declared a particular block to be an erasure with probability $\frac{2w_i}{d}$. The derandomization is based on the following observation.

Claim 17 *There exists some threshold Θ such that if we declare an erasure in the i'th location if $\Theta \leq \frac{2w_i}{d}$ for every i, then $2\bar{\tau} + \bar{s} < D$, where $\bar{\tau}$ and $\bar{s}$ as the number of errors and erasures when Θ is used as the threshold.*

PROOF: Suppose we pick $\Theta \in [0, 1]$ uniformly. For each i declare erasure for block if $\theta \leq \frac{2w_i}{d}$. Define $\bar{\tau}$ and $\bar{s}$ as the number of errors and erasures when Θ is used as the threshold. The previous argument shows that

$$E_\Theta[2\bar{\tau} + \bar{s}] < D \tag{28}$$

since all that was required for the argument was that at location i we declared an erasure with probability $2w_i/d$. Thus,there exists a Θ for which $2\bar{\tau} + \bar{s} < D$. □ Now the derandomization problem reduces to one of searching for an appropriate value for Θ. Since Θ takes value in the continuous range $[0, 1]$, we cannot try out all possibilities in the range. However, note that if we order the w_i in increasing order so that (with notational abuse) $0 \leq w_1 \leq \ldots \leq w_N$. All values of Θ in the range $[\frac{2w_i}{d}, \frac{2w_{i+1}}{d})$ lead to the same set of erasure decisions (the first i locations are erased, and the last $n - i$ are not). Thus, our search for the right threshold Θ only needs to be over the discrete set of values $\Theta \in \{0, 1\} \bigcup \{\frac{2w_1}{d}, \ldots, \frac{2w_n}{d}\}$. Noting that w_i is an integer in the range $[0, \frac{d}{2})$ (or $d/2$ itself), there are only $O(d)$ relevant values of Θ to search over. We have thus proved the following.

Theorem 18 *For a concatenated code with outer code of block length N, alphabet size Q, and distance D and an inner code of distance d and block length n, we can correct any pattern of*

$< dD/2$ errors using $O(d)$ calls to an errors-and-erasure decoder for the outer code that can correct any pattern of τ and s erasures provided $2\tau + s < D$. The runtime of the decoding algorithm is $O(NQn^{O(1)} + NT_{\text{out}})$ where T_{out} is the running time of the outer errors-and-erasures decoder.

Together with our construction of concatenated codes with outer Reed-Solomon code that meet the Zyablov bound, we can conclude the following.

Corollary 19 *For any $R \in (0,1)$ and $\gamma > 0$, there is a polynomial time constructible family of binary codes of rate R that can be decoded from up to a fraction*

$$\frac{1}{2} \max_{R \leq r \leq 1} (1 - R/r) h^{-1}(1 - r) - \gamma$$

of errors.

Remark 20 *The above result implies the following for the two extremes of low rate and high rate codes. For $\epsilon \to 0$, we can correct up to a fraction $(\frac{1}{4} - \epsilon)$ errors in polytime with explicit binary codes of rate $\Omega(\epsilon^3)$, and we can correct a fraction ϵ of errors with explicit binary codes of rate $1 - O(\sqrt{\epsilon \log(1/\epsilon)})$. (We leave it as an exercise to check these calculations.) Note that the non-constructive rate bounds guaranteed by the Gilbert-Varshamov bound for these regimes are $\Omega(\epsilon^2)$ and $1 - O(\epsilon \log(1/\epsilon))$ respectively.*

6 Capacity achieving codes for the BSC and BEC

We will now use concatenation with outer code that can correct a small fraction of worst-case errors to construct codes that achieve the Shannon capacity of the BEC and BSC, together with polynomial time encoding/decoding. First, let us recall the definition of the Binary erasure channel.

Definition 21 (The Binary Erasure Channel (BEC)) *is parameterized by a real α, $0 \leq \alpha \leq 1$, which is called the erasure probability, and is denoted BEC_α. Its input alphabet is $\mathcal{X} = \{0,1\}$ and output alphabet is $\mathcal{Y} = \{0,1,?\}$. Upon input $x \in X$, the channel outputs x with probability $1 - \alpha$, and outputs ? (corresponding to erasing the symbol) with probability α. (It never flips the value of a bit.)*

The capacity of BEC_α equals $1-\alpha$. Recall that, if we have a $[n,k,d]_q$ code with message $x \in \mathbb{F}_q^k$ and corresponding codeword $y = Gx \in \mathbb{F}_q^n$. Suppose a subset $S \subseteq \{1,\ldots,n\}$ is received (uncorrupted). Decoding the message involves solving the linear system

$$G x_S = y_S \tag{29}$$

We reorder the indices so that the first $|S|$ entries in y are the uncorrupted entries, resulting in the following matrix system,

$$\left[\begin{array}{c} G_S \\ \hline G_{\overline{S}} \end{array}\right] \left[\begin{array}{c} x \end{array}\right] = \left[\begin{array}{c} y_S \\ \hline y_{\overline{S}} \end{array}\right] \tag{30}$$

As long as $\text{rank}(S) \geq k$, then this linear system can be solved exactly in $O(n^3)$ time by Gaussian Elimination. As long as the matrix G_S is full column rank then the solution is unique. We have the following.

Proposition 22 *Using a binary matrix of size $n \times n(1-\alpha)$ with entries chosen i.i.d uniform from $\{0,1\}$ as the generator matrix G achieves the capacity of the BEC with high probability.*

The drawbacks with this solution are the cubic time and randomized nature of the construction. in addition, for a given choice of G it is hard to certify that (most) $(1+\alpha)k \times k$ sub-matrices of G have full (column) rank. We will use the idea of concatenation to make this constructive.

Let α be the erasure probability of the BEC and say our goal is to construct a code of rate $(1-\alpha-\epsilon)$ that enables reliable communication on BEC_α. Let C_1 be a linear time encodable/decodable binary code of rate $(1-\epsilon/2)$ that can correct a small constant fraction $\gamma = \gamma(\epsilon) > 0$ of *worst-case* erasures. Such codes were constructed in [13, 1]. For the concatenated coding, we do the following. For some parameter b, we block the codeword of C_1 into blocks of size b, and then encode each of these blocks by a suitable *inner* binary linear code C_2 of dimension b and rate $(1-\alpha-\epsilon/2)$. The inner code will be picked so that it achieves the capacity of the BEC_α, and specifically recovers the correct message with success probability at least $1-\gamma/2$. For $b = b(\epsilon,\gamma) = \Omega\left(\frac{\log(1/\gamma)}{\epsilon^2}\right)$, a random code meets this goal with high probability, so we can find one by brute-force search (that takes constant time depending only on ϵ).

The decoding proceeds as one would expect: first each of the inner blocks is decoded, by solving a linear system, returning either decoding failure or the correct value of the block. (There are no errors, so when successful, the decoder knows it is correct.) Since the inner blocks are chosen to be large enough, each inner decoding fails with probability at most $\gamma/2$. Since the noise on different blocks are independent, by a Chernoff bound, except with exponentially small probability, we have at most a fraction γ of erasures in the outer codeword. (For $R = 1-\alpha-\epsilon$, we have $Pr(decoder\ failure,\ error) \leq 2^{-\epsilon^2 n} < \gamma$). These are then handled by the linear-time erasure decoder for C_1. We thus have,

Theorem 23 *For the BEC_α, we can construct codes of rate $1-\alpha-\epsilon$, i.e., within ϵ of capacity, that can be encoded and decoded in $n/\epsilon^{O(1)}$ time.*

While this is pretty good, the brute-force search for the inner code is unsatisfying, and the BEC is simple enough that better runtimes (such as $O(n\log(1/\epsilon))$) are achieved by certain irregular LDPC codes.

A similar approach can be used for the BSC_p. We recall the definition of the BSC.

Definition 24 (The Binary Symmetric Channel (BSC)) *has input alphabet $\mathcal{X} = \{0,1\}$ and output alphabet $\mathcal{Y} = \{0,1\}$. The BSC is parameterized by a real number p, $0 \leq p \leq \frac{1}{2}$ called the crossover probability, and often denoted BSCp. The channel flips its input with probability p.*

For the BSC, the outer code C_1 must be picked so that it can correct a small fraction of worst-case *errors* — again, such codes of rate close to 1 with linear time encoding and decoding are known [13, 8]. Everything works as above, except that the decoding of the inner codes, where we

find the codeword of C_2 closest to the received block, requires a brute-force search and this takes $2^b = 2^{\Omega(1/\epsilon^2)}$ time. This can be improved to polynomial in $1/\epsilon$ by building a look-up table, but then the size of the look-up table, and hence the space complexity and time for precomputing the table, is exponential in $1/\epsilon$. Thus,

Theorem 25 *For the BSC_p, we can construct codes of rate $1 - H(p) - \epsilon$, i.e., within ϵ of capacity, that can be encoded in $n/\epsilon^{O(1)}$ time and which can be reliably decoded in $n2^{1/\epsilon^{O(1)}}$ time.*

It remains an important open question to obtain such a result with decoding complexity $n/\epsilon^{O(1)}$, or even poly(n/ϵ).[1]

We also want to point out that recently an alternate method using LP decoding has been used to obtain polynomial time decoding at rates arbitrarily close to capacity [5]. But this also suffers from a similar poor dependence on the gap ϵ to capacity.

References

[1] Noga Alon and Michael Luby. A linear time erasure-resilient code with nearly optimal recovery. *IEEE Transactions on Information Theory*, 42(6):1732–1736, 1996. 16

[2] E. R. Berlekamp. Nonbinary bch decoding. *International Symposium on Information Theory*, 1968. 11

[3] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley, 1983. 11

[4] R. T. Chien. Cyclic decoding procedure for the bose-chaudhuri-hocquenghem codes. *IEEE Transactions on Information Theory*, IT-10:357–363, 1964. 11

[5] Jon Feldman and Clifford Stein. LP decoding achieves capacity. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 460–469, 2005. 17

[6] G. D. Forney. Generalized minimum distance decoding. *IEEE Trans. Information Theory*, IT-12:125–131, 1966. 12

[7] Peter Gemmell and Madhu Sudan. Highly resilient correctors for multivariate polynomials. *Information Processing Letters*, 43(4):169–174, 1992. 6

[8] V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, October 2005. 16

[9] J. Justesen. Class of constructive asymptotically good algebraic codes. *Information Theory, IEEE Transactions on*, 18(5):652 – 656, sep 1972. 1

[1] We remark that asymptotically, with ϵ fixed and $n \to \infty$, the exponential dependence on $1/\epsilon$ can be absorbed into an additional factor with a slowly growing dependence on n. However, since in practice one is interested in moderate block length codes, say $n \leq 10^6$, a target runtime such as $O(n/\epsilon)$ seems like a clean way to pose the underlying theoretical question.

[10] J. L. Massey. Shift-register synthesis and bch decoding. *IEEE Trans. Information Theory*, IT-15:122–127, 1969. 11

[11] W. W. Peterson. Encoding and error-correction procedures for the bose-chaudhuri codes. *IRE Transactions on Information Theory*, IT-6:459–470, 1960. 8

[12] B.-Z. Shen. A justesen construction of binary concatenated codes that asymptotically meet the zyablov bound for low rate. *Information Theory, IEEE Transactions on*, 39(1):239 –242, jan 1993. 4

[13] D. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1732, 1996. 16

[14] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding goppa codes. *Information and Control*, 27:8799, 1975. 11

[15] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. *US Patent Number 4,633,470*, December 1986. 6

[16] Wikipedia. Chien search — Wikipedia, the free encyclopedia, 2004. [Online accessed 24-Mar-2010]. 11