6.897 Algorithmic Introduction to Coding Theory	September 26, 2001
Lecture 6	
Lecturer: Madhu Sudan	*Scribe: Nicole Immorlica*

Today we will talk about:

- Wozencraft construction continued
- Building codes from other codes
  - Parity check bit
  - Puncturing
  - Restriction
  - Direct Product
  - Concatenation
- Forney codes
- Justesen codes

1 Wozencraft construction (continued)

The Wozencraft construction gives a $2^{O(n)}$ time algorithm for constructing $[n,k,d]_2$ codes. We pick up where we left off in the last lecture. Recall our goal is to construct a family of sets $S_1, S_2, \ldots, S_t \subseteq \{0,1\}^n - \mathbf{0}$ such that

1. The sets are pairwise disjoint.
2. $\forall i$, $S_i \cup \{\mathbf{0}\}$ is a linear subspace of $\{0,1\}^n$.
3. $t \geq \text{Vol}(d,n)$.
4. $\forall i,j : |S_i| = 2^k - 1$.

We saw last lecture that if we can construct such a family of sets, one of these sets will yield a $[n,k,d]_2$ code. Today we will see Wozencraft's construction of such a family of sets. We will show the construction only $n = 2k$. It is fairly simple to generalize it to a construction for $n = ck$ for any integer c.

We will use the correspondence between fields and vector spaces that preserves addition (see Lecture Notes on Algebra, Section 6). In particular we will view $\mathbb{F}_2^k$ as $\mathbb{F}_{2^k}$ and $\mathbb{F}_2^n$ as $\mathbb{F}_{2^k}^2$. The sets we will construct will be indexed by $\alpha \in \mathbb{F}_{2^k}$, with S_α defined as follows: $S_\alpha = \{(x, \alpha x) \mid x \in F_{2^k} - \{0\}\}$. We now verify that the S_α's satisfy the above conditions for $t = 2^k$ and d such that $\text{Vol}(d,n) \leq t$.

1. S_α's are pairwise disjoint: In particular, For every $(x,y) \in F_{2^k}^2$, there is at most one α such that $(x,y) \in S_\alpha$, namely $\alpha = xy^{-1}$ provided y is non-zero and $\alpha = 0$ if $y = 0$. (If $x = 0$ then $(x,y) \notin S_\alpha$ for any α.)
2. $S_\alpha \cup \{0\}$ is linear: Clearly each S_α is a linear subspace of $F_{2^k}^2$ and is generated by the matrix $[1\alpha]$. Since the correspondence between $\mathbb{F}_2^k$ and $\mathbb{F}_{2^k}$ respects addition, it follows that $S_\alpha \cup \{0\}$ are linear over $\mathbb{F}_2$ as well.
3. There are clearly $t = 2^k$ of the S_α's. The condition $t \geq \text{Vol}(d,n)$ follows from the definition of d.

4. It is also obvious that $|S_\alpha| = 2^k - 1$.

Taking the ratios k/n and d/n we note that the codes S_α always have a rate of $\frac{1}{2}$. Further if we fix any $\epsilon > 0$, and set $d = (H^{-1}(\frac{1}{2}) - \epsilon)n$ then for all sufficiently large n we have $\mathrm{Vol}(d, n) \leq 2^{n/2}$ and thus the family above gives a code of rate $\frac{1}{2}$ and relative distance approaching $H^{-1}(\frac{1}{2})$.

By a slightly more careful argument we can actually verify that most codes in the family achieve the Gilbert-Varshamov bound. Specifically, we can prove:

Theorem 1 *For every $\epsilon > 0$ and for all sufficiently large even numbers n, Wozencraft's construction with parameter n gives a family of $2^{n/2}$ codes with all but ϵ fraction of which are $[n, \frac{1}{2}n, (H^{-1}(\frac{1}{2}) - \epsilon)n]_2$-codes.*

Remarks:

1. Furthermore, for all such n, given an index i of a code from the family with parameter n, any specific entry of the generator matrix of the ith code can be computed in time polynomial in n.
2. If n is of the form $4 \cdot 3^t$, then the computation can be carried out in $O(\log n)$ space. This part follows from the fact that the irreducible polynomial for such $\mathbb{F}_{2^k}$ where $k = n/2$ is known explicitly and this polynomial is sparse. (Thanks to Dieter van Melkebeek (dieter@ias.edu) for pointing out this use of sparsity.)

Exercise: Extend the argument above to construct for every integer c, every $\epsilon > 0$, and all sufficiently large k, an ensemble of $2^{(c-1)k}$ codes such that all but an ϵ-fraction of the ensemble are $[ck, k, (H^{-1}(1 - \frac{1}{c}) - \epsilon)(ck)]_2$-codes. Your construction should take time $2^{O(ck)}$.

References: The Wozencraft ensemble of codes do not appear in any paper by Wozencraft. They are alluded to in a monograph by Massey [3, Section 2.5]. The actual family as described above is from Justesen's paper [2]. The extension asked for in the exercise is from the paper of Weldon [4].

2 Building codes from other codes

In the previous section we saw that asymptotically good codes exist. However, we had no explicit construction for them. The second holy grail of coding theory is to construct in polynomial time binary codes that meet the GV-bound. No one knows how to do this yet. One approach to this problem is to create new codes from existing ones. We look at five ways of getting new codes from old codes. Four of them don't improve the asymptotics of the code. The fifth leads to constructions of families of asymptotically good codes. (However, they do not meet the GV-bound.)

2.1 Parity check bit

We recall a construction of Hamming (see notes for Lecture 3). Given a code $C = [n, k, d]_2$, create a new code $C' = [n + 1, k, d']_2$ as follows. First encode the message using C to get a codeword $\mathbf{c}$ of length n. Then, add an extra bit which is the parity of the bits of c. This new codeword, $\mathbf{c}'$ has length $n + 1$. Furthermore, as argued in Lecture 3, if n is odd, the new distance $d' = d + 1$. Otherwise the distance may remain d.

The parity check bit operation does improve relative distance for codes of odd length but not for codes of even length. Furthermore, the rate suffers. So we can not repeat this method to obtain really great codes.

2.2 Puncturing

Given a code $C = [n, k, d]_q$, create a new code $C' = [n - t, k, d']_q$ by simply deleting t coordinates. The new distance d' will be $d - t \leq d' \leq d$. For $t = 1$ we can think of the puncturing operation as achieving the effect of the inverse of the parity check bit operation (in a very loose sense).

This operation has the benefit of decreasing the encoding length thereby improving the rate. But at the same time it sacrifices the minimum distance of the code and thus decreases the relative distance.

While this operation does not yield a generic construction method for good codes, it turns out to be very useful in special cases. Often the best known code for a specific choice of, say n and k, might be a code obtained from puncturing a well-known code of longer block length. In such cases, special features of the code are often used to show that the distance is larger than the proven bound. Note further that all linear codes are punctured Hadamard codes! So obviously puncturing can lead to good codes. The question remains: When does it work? and what part of the codes should be punctured?

2.3 Restriction

Given a code $C = (n, k, d)_q$ over an alphabet Σ, create a new code $C' = (n-1, k', d)_q$ by choosing $\alpha \in \Sigma$ and $i \in [n]$ and retaining only those codewords $\mathbf{c}$ in which the ith coordinate of the codeword is α. The code C' is then obtained by deleting the ith coordinate from all remaining codewords.

The resulting code has block length n. If we pick α so that it is the most common letter in the ith coordinate (among codewords of C) then at least q^k/q messages will remain in C'. Since codewords differed in d positions to start with, and the only codewords that remain agreed in the deleted coordinate, the new codewords are still at Hamming distance at least d.

Restriction does improve the relative distance, but not necessarily the rate.

2.4 Direct Product

Given a codes $C_1 = [n_1, k_1, d_1]_q$ and $C_2 = [n_2, k_2, d_2]_q$, the direct product of C_1 and C_2, denoted $C_1 \otimes C_2$, is an $[n_1 n_2, k_1 k_2, d_1 d_2]_q$ constructed as follows. View a message of $C_1 \otimes C_2$ as a k_2 by k_1 matrix $\mathbf{M}$. Encode each row of $\mathbf{M}$ by the code C_1 to obtain an k_2 by n_1 intermediary matrix. Encode each column of this intermediary matrix with the C_2 code to get an n_2 by n_1 matrix representing the codeword encoding $\mathbf{M}$. This process works generally - for linear as well as non-linear codes C_1 and C_2. We first show that the resulting code has distance at least $d_1 d_2$ in either case. Then we show that if C_1 and C_2 are linear, then the resulting code is also linear, and furthermore is the same as the code that would be obtained by encoding the columns with C_2 first and then encoding the rows with C_1.

We prove this new code has distance at least $d_1 d_2$. Consider two distinct message matrices $\mathbf{M}_1$ and $\mathbf{M}_2$. Let $\mathbf{N}_1$ and $\mathbf{N}_2$ be the intermediate matrices obtained after the first step of the encoding process. Let $\mathbf{C}_1$ and $\mathbf{C}_2$ be the final codewords obtained from these matrices. Suppose $\mathbf{M}_1$ and $\mathbf{M}_2$ differ on the ith row. Then $\mathbf{N}_1$ and $\mathbf{N}_2$ must differ on at least d_1 coordinates on the ith row. In particular they differ on at least d_1 columns. Say $j_1, \ldots, j_{d_1}$ are indices of d_1 such columns where $\mathbf{N}_1$ and $\mathbf{N}_2$ differ. Then the column-by-column encoding results in codewords $\mathbf{C}_1$ and $\mathbf{C}_2$ which differ on at least d_2 coordinates on each of these d_1 columns. Thus $\mathbf{C}_1$ and $\mathbf{C}_2$ differ on at least $d_1 d_2$ entries.

Next we show that $C_1 \otimes C_2$ is linear if C_1 and C_2 are linear, and the encoding functions used are linear functions.

Claim 2 *Let $\mathbf{R}_1 \in \mathbb{F}_q^{k_1 \times n_1}$ generate the code C_1 and let $\mathbf{R}_2 \in \mathbb{F}_q^{k_2 \times n_2}$ generate the code C_2. Then the* *direct product code* $C_1 \otimes C_2$ *is a linear code that has as its codewords* $\{\mathbf{R_2}^T \mathbf{M} \mathbf{R}_1 \mid \mathbf{M} \in \mathbb{F}_q^{k_2 \times k_1}\}$.

Remark: As a consequence, we note that it does not matter if we encode the rows first and then the columns as above or vice versa.

Proof The proof follows easily from the fact that the intermediate matrix equals $\mathbf{MR}_1$ and thus the final matrix equals $\mathbf{R_2}^T(\mathbf{MR}_1)$. The interchangeability follows from associativity of matrix multiplication. The linear follows from the fact that the matrix $\mathbf{R_2}^T\mathbf{M}_1\mathbf{R}_1 + \mathbf{R_2}^T\mathbf{M}_2\mathbf{R}_1$ is just the encoding of $\mathbf{M}_1 + \mathbf{M}_2$ and the matrix $\alpha\mathbf{R_2}^T\mathbf{M}_1\mathbf{R}_1$ is the encoding of $\alpha\mathbf{M}_1$, where $\alpha \in \mathbb{F}_q$. ■

Exercise: In general the direct product of two codes depends on the choice of the encoding function. Prove that this is not the case for linear codes. Specifically, prove that if $\mathbf{R}_1$ and $\mathbf{R}'_1$ generate C_1 and $\mathbf{R}_2$ and $\mathbf{R}'_2$ generate C_2, then $\{\mathbf{R_2}^T\mathbf{MR}_1 \mid \mathbf{M}\} = \{\mathbf{R}'^T_2\mathbf{MR}'_1 \mid \mathbf{M}\}$.

Again, the direct product does not help in the construction of asymptotically good codes. E.g. if we started with codes C_1 and C_2 of rate and relative distance $\frac{1}{10}$, then the resulting code is weaker and has rate and relative distance of only $\frac{1}{100}$.

So far all the operations on codes have been ineffective in getting to asymptotically good codes. In retrospect one may say that this is because all these operations fixed the alphabet and tried to play around with the other three parameters. A simply but brilliant idea, due to Forney [1], showed how to extend the game to include the alphabet size in the parameters altered/expoited by the operations on codes. This operation is that of "concatenating codes". This method turns out to have profound impact on our ability to construct asymptotically good binary codes. We describe this method an its consequences in the next section.

3 Concatenation of codes

To motivate the notion of concatenation, let us recall the example using Reed-Solomon codes on CD players. Reed-Solomon codes were defined on large alphabets, while CD players work with the binary alphabet. However, given an $[n,k,d]_{2^r}$ Reed-Solomon code, we interpreted this code as an $[nr,kr,d]_2$ *binary* code by naively representing the alphabet of the RS code, elements of $\mathbb{F}_{2^r}$, as binary strings of length r. The main idea of concatenation is to focus on this "naive interpretation" step and to generalize it so that elements of $\mathbb{F}_{2^r}$ can be represented by binary strings of length larger than r. Note that the main loss in performance is due to the fact that in going from strings of length n (over $\mathbb{F}_{2^r}$) to binary strings of length nr, we did not increase the minimum distance of the code, and so lost in terms of the relative distance. A careful choice of the encoding in the second step ought to be able to moderate this loss, and this is exactly what the method of concatenation addresses.

As in the case of direct product codes, it is best to explain concatenation of codes in terms of the encoding functions. First we define the l-fold concatenation of a single encoding function.

Definition 3 *For positive integer l, linearity preserving bijective map $\pi : \mathbb{F}_{q^k} \to \mathbb{F}_q^k$ and encoding function $E : \mathbb{F}_q^k \to \mathbb{F}_q^n$ the l-*fold concatenation *of E is the function $\diamond_l^\pi E : \mathbb{F}_{q^k}^l \to \mathbb{F}_q^{nl}$ given by $\langle \mathbf{x}_1, \ldots, \mathbf{x}_l \rangle \mapsto \langle E(\pi(\mathbf{x}_1)), \ldots, E(\pi(\mathbf{x}_l)) \rangle$, where $\mathbf{x}_i \in \mathbb{F}_{q^k}$ for $i \in [l]$.*

Typically the exact map $\pi : \mathbb{F}_{q^k} \to \mathbb{F}_q^k$ is irrelevant so we will simply ignore it. Further if l is clear from context, we will ignore it and simply refer to the map $\diamond E$. We now define the concatenation of two codes.

Definition 4 *For encoding functions* $E_1 : \mathbb{F}_{q^{k_2}}^{k_1} \to \mathbb{F}_{q_2^k}^{n_1}$ *and* $E_2 : \mathbb{F}_q^{k_2} \to \mathbb{F}_q^{n_2}$ *(and some implicit bijection* $\pi : \mathbb{F}_{q^{k_2}} \to \mathbb{F}_q^{k_2}$*), the concatenation of* E_1 *and* E_2 *is the function* $E_1 \diamond E_2 : \mathbb{F}_q^{k_1 k_2} \to \mathbb{F}_q^{n_1 n_2}$ *given by*

$$\mathbb{F}_q^{k_1 k_2} \xrightarrow{\pi^{-1}} \mathbb{F}_{q^{k_2}}^{k_1} \xrightarrow{E_1} \mathbb{F}_{q^{k_2}}^{n_1} \xrightarrow{\pi} \left(\mathbb{F}_q^{k_2}\right)^{n_1} \xrightarrow{\diamond E_2} \left(F_q^{n_2}\right)^{n_1} \longrightarrow F_q^{n_1 n_2}.$$

In the message $\langle \mathbf{x}_1, \ldots, \mathbf{x}_{k_1} \rangle$ *is mapped to the vector* $\diamond_{n_1}^{\pi} E_2(E_1(\langle \pi^{-1}(\mathbf{x}_1), \ldots, \pi^{-1}(\mathbf{x}_{k_1}) \rangle))$.

If the encoding functions E_1, E_2 are linear maps giving linear codes C_1 and C_2 respectively, then $E_1 \diamond E_2$ is a linear map whose image is denoted by $C_1 \diamond C_2$. It may be verified that $C_1 \diamond C_2$ is a function of C_1 and C_2 alone and not dependent on E_1, E_2 or π. It is customary to call the code C_1 the *outer code* and the code C_2 the *inner code*, and $C_1 \diamond C_2$ is the concatenated code.

The next proposition verifies the distance properties of concatenated codes.

Proposition 5 *If* C_1 *is an* $[n_1, k_1, d_1]_{q^{k_2}}$*-code and* C_2 *is an* $[n_2, k_2, d_2]_q$*-code then* $C_1 \diamond C_2$ *is an* $[n_1 n_2, k_1 k_2, d_1 d_2]_q$*-code.*

Proof The only part that needs to be verified is the distance. To do so consider the encoding of a non-zero message. The encoding by E_1 leads to an intermediate word from $\mathbb{F}_{q^{k_2}}^{n_1}$ that in non-zero in d_1 coordinates. The n_1-fold concatenation of E_2 applied to the resulting codeword produces d_2 non-zero symbols in every block where the outer encoding produced a non-zero symbol. Thus we end up with at least $d_1 d_2$ non-zero symbols in the concatenated encoding. ■

If we ignore the non-trivial behavior with respect to the alphabet size, then the concatenation operator has essentially the same parameters as the direct product operator. However the concatenation operator allows the outer code to be over a larger alphabet and we have seen that it is easier to construct good codes over large alphabets. Thus the concatenation operator is strictly better than direct product. Below we show an example of non-trivial results it yields.

Example - RS $\diamond$ Hadamard: Suppose we concatenate an outer code that is an $[n, k, n-k]_n$-Reed-Solomon code with a $[n, \log n, \frac{n}{2}]_2$-Hadamard code. (Assume for this example that n is a power of 2.) Then the concatenated codes is an $[n^2, k \log n, \frac{n}{2}(n-k)]_2$-code. Depending on our choice of rate k/n of the outer code, we get a family of binary codes of constant relative distance and an inverse polynomial rate $R = \frac{k \log n}{n^2}$. This is a new range of parameters that we have not seen in the codes so far.

While it is possible to employ multiple levels of concatenation to improve the dependence of the block length n on the message length k making n closer and closer to being linear in n, we can never get an asymptotically good code this way. Informally, to get an asymptotically good family, we need both the inner code and outer code to be asymptotically good. In what follows, we will describe two approaches at getting constructions of asymptotically good codes using concatenation.

3.1 Forney codes/Zyablov bound

The first family of codes we describe are due to Forney [1], who described the basic idea of the codes, but did not stress the choice of parameters that would optimize the tradeoff between rate and relative distance. (Forney was after bigger fish, specifically an algorithmic version of Shannon's theorem. We will get to this when we get to algorithms.) The actual bounds were worked out by Zyablov [5] and are usually referred to as the Zyablov bounds.

The idea to get a polynomial time constructible family of asymptotically good codes is a simple one. As an outer code we will use a Reed-Solomon code over an n-ary alphabet, say an $[n, k, n-k]_n$-code.

For the inner code, we will search for the best linear code in, say, Wozencraft's ensemble of codes. This takes exponential time in the block length of the inner code, but the block length of the inner code only needs to be linear in the message length and the message length of the inner code is only $\log n$. Thus the time it takes to find the best code in Wozencraft's ensemble is only polynomial in n.

Getting a little more specific, to construct a code of relatve distance δ, we pick δ_1 and δ_2 so that $\delta_1\delta_2 = \delta$. For the outer code we pick an $[n, (1-\delta_1)n, \delta_1 n]_n$-RS-code. For the inner code we search Wozencraft's ensemble to obtain an $[n', (1-H(\delta_2))n', \delta_2 n']_2$-code with $(1-H(\delta_2))n' = \log n$. The resulting code has block length $nn' = O(n \log n)$, relative distance δ and rate $(1-\delta_1)(1-H(\delta_2))$. Thus we obtain the following theorem:

Theorem 6 *For every $\delta \in (0, \frac{1}{2})$, there exists an infinite family of polynomial time constructible codes $\mathcal{C}$ with rate R and relative distance δ satisfying*

$$R \geq \max_{\delta \leq \delta_2 < \frac{1}{2}} \left\{ (1 - H(\delta_2)) \cdot \left(1 - \frac{\delta}{\delta_2}\right) \right\}. \tag{1}$$

The bound (1) above is the Zyablov bound.

3.2 Explicit constructions

We take a brief digression to discuss what it means to construct a code explicitly. It is clear that this ought to be a complexity-theoretic definition, since a code is a finite set and one can obviously enumerate all finite sets to see if one of them gives, say, an (n, k, d)-code. The constructions of Gilbert took exponential time, while Varshamov's is a randomized polynomial time construction that possibly returns an erroneous solution (to the task of finding an $[n, k, d]$ code). We asserted that Forney's construction is somehow explicit, and yet this is not satisfactory to many mathematicians. Here we enumerate some criteria for explicit constructions for the case of codes (though similar criteria apply to constructions of all combinatorial objects).

Let $\{\mathcal{C}_{R,\delta}\}_{(R,\delta)}$ be a collection of families of codes, where the family $\mathcal{C}_{R,\delta}$ has rate R and relative distance δ. The following are possible notions of $\mathcal{C}$ being explicitly constructible:

Polytime For every $0 < R < 1$ and $0 < \delta < 1$, there exists a polynomial p such that generator matrix of the ith element of the family $\mathcal{C}_{R,\delta}$, with block length n_i, is constructible in time $p(n_i)$, if such a family exists.

Uniform polytime There exists a polynomial p such that for every $0 < R < 1$ and $0 < \delta < 1$, generator matrix of the ith element of the family $\mathcal{C}_{R,\delta}$, with block length n_i, is constructible in time $p(n_i)$, if such a family exists.

The difference between polytime constructibility and uniform polytime constructibility is relatively small. This distinction can be made in the remaining definitions too, but we will skip the extra quantifiers, and simply focus on what makes a code $\mathcal{C}$ constructible (leaving it to the reader to find a preference within uniform and nonuniform time bounds).

Logspace The generator matrix of the ith member of $\mathcal{C}$ is constructible in logarithmic space. (This implies that $\mathcal{C}$ is polynomial time constructible.)

Locally Polytime Constructible [1] Here we will require that a specific entry, say the j, lth entry, of the generator matrix of the ith member of the code $\mathcal{C}$ be computable in time polynomial in the

[1] Actually, this notion does not have a name and I had to generate one on the fly. Thanks to Anna Lysyanskaya for suggesting this name.

size of the binary representation of i, j, l. (Note this representation has size logarithmic in n and so this notion is much more explicit than earlier notions.)

Locally Logspace Constructible The j, lth entry of the generator matrix of the ith code is logspace constructible in the length of the binary representations of i, j and l.

As noted, the requirements get more stringent as we go down the list above. The notion of Locally Logspace Constructible is about as strong a requirement we can pose without getting involved with machine-dependent problems. (What operations are allowed? Why? etc.)

Forney's codes, as described above, are polytime constructible, but not uniform polytime or logspace constructible. The next family of codes we will describe are locally logspace constructible, making them as explicit as we could desire (define?).

3.3 Justesen Codes

The principal barrier we seem to face in producing codes explicitly is that we know how to construct smaller and smaller ensembles of good codes, but we don't know how to get our hands on any particular good one. In fact in the ensembles we construct almost all codes are good. Is there any way to use this fact? Justesen's idea [2] is a brilliant one — one that "derandomizers" should take note of: On the one hand we can produce a small sample space of mostly good codes. On the other hand we need one good code that we wish to use repeatedly — n_1 times in the concatenation. Do we really need to use the same code n_1 times? Do they all have to be good? The answer, to both questions, is NO! And so, surprisingly enough, the ensemble of codes is exactly what suffices for the construction. Specifically, we take an $[n_1, k_1, d_1]_{q^{k_2}}$-outer code with encoding function E_1 and an ensemble consisting of n_1 inner codes with the ith member denoted $E_2^{(i)}$. We encode a message $\mathbf{m}$ by first applying the outer encoding function to get $E_1(\mathbf{m})$ and then applying the ith inner encoding function to the ith coordinate of $E_1(\mathbf{m})$, getting the vector $\langle E_2^{(1)}((E_1(\mathbf{m}))_1), \ldots, E_2^{(n_1)}((E_1(\mathbf{m}))_{n_1})\rangle$.

The above definition can be formalized to get a notion of concatenating an $[n_1, k_1, *]_{q^{k_2}}$-outer code with an ensemble containing n_1 $[n_2, k_2, *]_q$-inner codes ($*$ representing the fact that the distances are unknown, or possibly not all the same). Denoting the outer code by C_1, and the inner ensemble by $\overline{C_2}$, we extend the notation for concatenation and use $C_1 \diamond \overline{C_2}$ to denote such concatenations. The following proposition shows how the parameters of the concatenated codes relate to those of the outer code and inner ensemble.

Proposition 7 *Let C_1 be an $[n_1, k_1, d_1]_{q^{k_2}}$ code. Let $\overline{C_2}$ be an ensemble of n_1 $[n_2, k_2, *]_q$-codes of which all but ϵ-fraction have minimum distance d_2. Then the concatenated code $C_1 \diamond \overline{C_2}$ is an $[n_1 n_2, k_1 k_2, (d_1 - \epsilon n_1) d_2]_q$ code.*

Proof The proof follows from the fact that the first level encoding of a non-zero message leaves at least d_1 coordinates that are non-zero. At most ϵn_1 of the inner codes do not have minimum distance d_2. Thus at least $d_1 - \epsilon n_1$ coordinates, when encoded by $\overline{C_2}$ result in d_2 non-zero zymbols each. The distance follows. ■

Note that it is not entirely trivial to find an ensemble with just the right parameters: To use every element of the ensemble at least once, we need the inner ensemble size to be no larger than the outer block length. To use an RS code at the outer level, we need the outer block length to be no larger than the outer alphabet size. To use concatenation, we need the number of outer alphabet size to be no larger than the number of inner codewords. Putting it all together, we need an ensemble with no more members than codewords per member of the ensemble. Fortunately enough, this is exactly what is achieved by Wozencraft's ensemble, so we can use it. Consequenntly we get one fully explicit (locally logspace constructible) family of error-correcting codes on the Zyablov bound. In particular the code is asymptotically good.

Theorem 8 *For every $0 < \delta < H^{-1}(\frac{1}{2})$, there exists a locally logspace constructible infinite family of codes $\mathcal{C}$ that has relative distance δ and rate $\frac{1}{2}\left(1 - \frac{\delta}{H^{-1}(\frac{1}{2})}\right)$.*

The code above is obtained by concatenating a Reed-Solomon code of appropriate rate with the Wozencraft ensemble. We note that to get local logspace constructibility, we need the inner code length to be $4 \cdot 3^l$ for some integer l so that we can use the explicit construction of fields of size $2 \cdot 3^l$.

References

[1] G. David Forney. Generalized Minimum Distance decoding. *IEEE Transactions on Information Theory*, 12:125–131, 1966.

[2] Jørn Justesen. A class of constructive asymptotically good algebraic codes. *IEEE Transactions on Information Theory*, 18:652–656, 1972.

[3] James L. Massey. *Threshold decoding.* MIT Press, Cambridge, Massachusetts, USA, 1963.

[4] Edward J. Weldon, Jr. Justesen's construction — the low-rate case. *IEEE Transactions on Information Theory*, 19:711–713, 1973.

[5] Victor V. Zyablov. An estimate on the complexity of constructing binary linear cascade codes. *Problems of Information Transmission*, 7(1):3–10, 1971.