CS250/EE387 - LECTURE 4 - A few more bounds... and REED-SOLOMON CODES!!!

(1/18/2017)

These are my favorite things.

TINY GASTROPOD FACT.

The world's smallest snail, Angustopila dominikae, (found in limestone in China) are < 1mm long. 10 of them can fit through the eye of a needle at once!

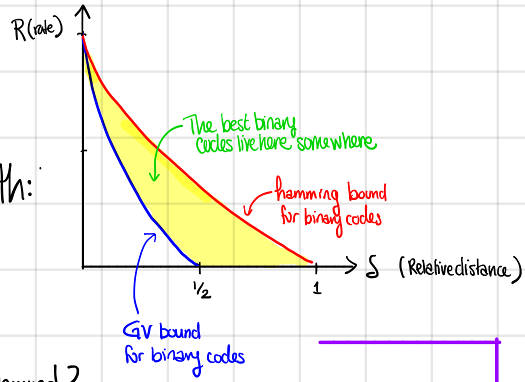
AGENDA

1. Plotkin + Singleton bounds
2. Reed Solomon Codes!
3. Dual view of RS Codes

+ more algebra! ☺

0. Recall from last time:

We took some limits, let n get big, and ended up with:



SOME QUESTIONS.

QUESTION Are there families of codes that beat the GV bound?

ANSWER 1: Yes. For $q \geq 49$, "Algebraic Geometry Codes" beat the GV bound.

ANSWER 2: ???

For binary codes, we don't know. OPEN PROBLEM!

QUESTION Can we find explicit constructions of families of codes that meet the GV bound?

ANSWER 1. For large alphabets, yes. (We'll see soon)

ANSWER 2. ???

For binary codes, recent work of [Ta-Shma 2017] gives something close in a very particular parameter regime... but in general, OPEN PROBLEM!

① Singleton & Plotkin bounds

Let's try to narrow down that  region a little bit.

THM. [Singleton Bound] If $\mathcal{C}$ is an $(n,k,d)_q$ code, then $k \le n-d+1$.

Proof. For $c \in \mathcal{C}$, consider throwing out the last $d-1$ coordinates:

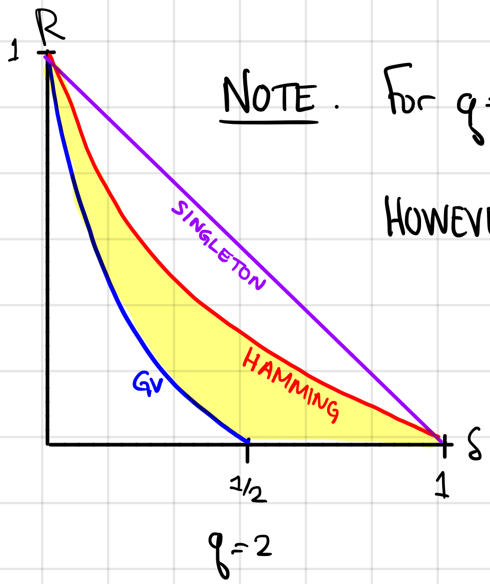
$$c = (\underbrace{x_1, x_2, \dots x_{n-d+1}}_{\text{call this } \varphi(c) \in \Sigma^{n-d+1}}, \underbrace{x_{n-d+2}, \dots, x_n}_{\text{get rid of these}})$$

Consider $\check{\mathcal{C}} = \{ \varphi(c) : c \in \mathcal{C} \}$, so $\check{\mathcal{C}} \subseteq \Sigma^{n-d+1}$

CLAIM 1: $|\mathcal{C}| = |\check{\mathcal{C}}|$ $\rightarrow$ If not, then $\exists\, c, c'$ s.t. $\varphi(c) = \varphi(c')$. But then $\Delta(c, c') \le d-1$ ↯

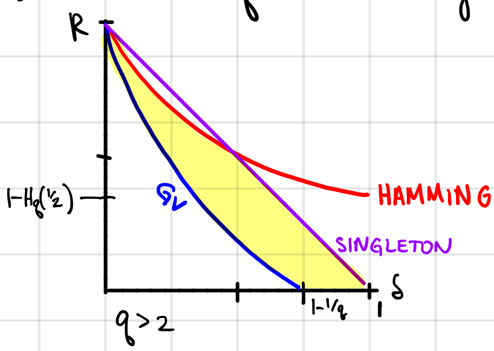
CLAIM 2: $|\check{\mathcal{C}}| \le q^{n-d+1}$ $\rightarrow$ Since $\check{\mathcal{C}} \subseteq \Sigma^{n-d+1}$

Thus, $|\mathcal{C}| \le q^{n-d+1} \Rightarrow q^k \le q^{n-d+1} \Rightarrow k \le n-d+1$.

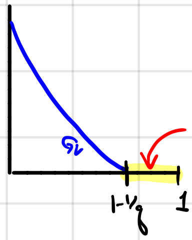


NOTE. For $q=2$, the Singleton bound is WORSE than the Hamming bound! ⍨

HOWEVER (a) it's simpler, and (b) as $q \to \infty$ we'll get something better.



The GV bound only works up to $d/n \le 1 - 1/q$.
Is this necessary? Turns out, YES, at least asymptotically.



can we get anything here??

THM [PLOTKIN BOUND]
Let C be a $(n, k, d)_q$ code.
(a) If $d = (1 - 1/q) \cdot n$, then $|C| \le 2 \cdot q \cdot n$

(b) If $d > (1 - 1/q) \cdot n$, then $|C| \le \dfrac{d}{d - (1 - 1/q) \cdot n}$

Notice that either (a) or (b) imply $R \to 0$ as $n \to \infty$.
Thus, in order to have a constant-rate code, we should have $d < (1 - 1/q) \cdot n$.

We'll omit the proof of the Plotkin bound in class — Check out ESSENTIAL CODING THEORY §4.4 for a proof.

COR. Let C be a family of codes of rate R and distance δ. Then

$$R \le 1 - \left(\frac{q}{q-1}\right) \cdot \delta + o(1)$$

Proof. (Assuming the Plotkin bound)

n' is the largest n' s.t. $n' \le (1 - 1/q) \cdot d$

SOMETHING IS FISHY HERE!!!

Choose $n' = \left\lfloor \frac{dq}{q-1} \right\rfloor - 1$. For all $x \in \Sigma^{n-n'}$, define

$$C_x = \{ (c_{n-n'+1}, \ldots, c_n) \mid c \in C \text{ with } (c_1, \ldots, c_{n-n'}) = x \}$$
= the set of ENDS of codewords that BEGIN with x.

Now C_x has distance $\ge d$, block length $n' \le (1 - 1/q) \cdot d$.
Applying the Plotkin bound, $|C_x| \le \frac{d}{d - (1 - 1/q) n'} \le d$

ctd...

proof ctd.

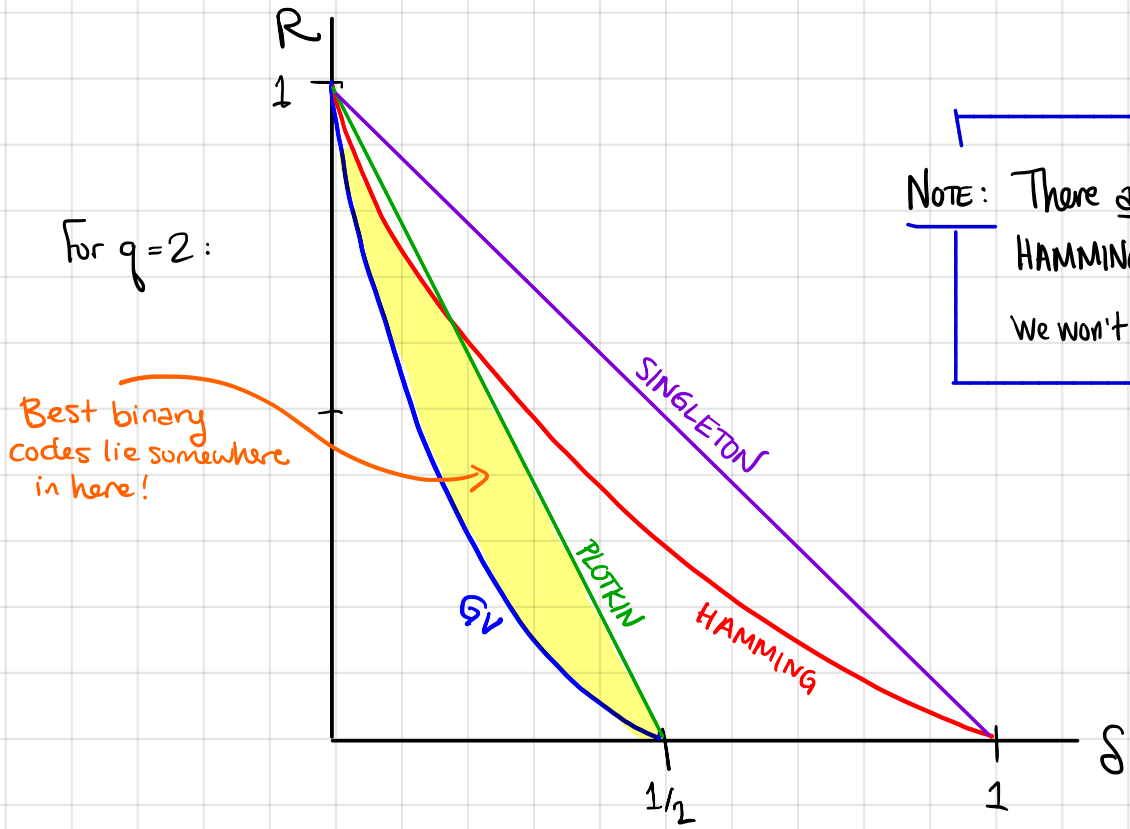
But then

$$|C| = \sum_{x \in \Sigma^{n-n'}} |C_x| \le q^{n-n'} \cdot d$$
$$= q^{\left(n - \lfloor \frac{qd}{q-1} \rfloor + 1\right)} \cdot d$$
$$= \exp_q\left(n - \frac{qd}{q-1} + o(n)\right)$$
$$= \exp_q\left(n\left(1 - \delta\left(\tfrac{q}{q-1}\right) + o(1)\right)\right),$$

So $R \le 1 - \left(\frac{q}{q-1}\right)\delta + o(1)$, as desired. ∎

Did we make progress? Yes! We narrowed down the yellow region a bit.

For $q=2$:

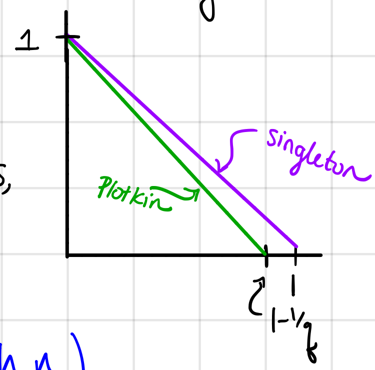


NOTE: There are bounds better than HAMMING ∧ PLOTKIN, but we won't cover them in this course.

FUN EXERCISE: What happens to this picture as $q \to \infty$?

② REED-SOLOMON CODES.

Notice that for any fixed q, the Plotkin bound is strictly better than the Singleton bound.



AND YET, today we are going to see Reed-Solomon Codes, which EXACTLY ACHIEVE the SINGLETON BOUND.

(The trick: the alphabet size will be growing with n)

We can define polynomials over finite fields, just like we can over $\mathbb{R}$.

$$f(X) = a_0 + a_1 \cdot X + a_2 \cdot X^2 + \cdots + a_d \cdot X^d$$

$a_i \in \mathbb{F}_q$

X is a variable that we think of as taking values in $\mathbb{F}_q$

$a_d \neq 0$

d is the DEGREE of the polynomial.

> Note: depending on your background, it's totally normal to use capital X as a variable or it's totally weird. If it's the latter, get over it.

The set of all univariate polynomials w/ coeffs in $\mathbb{F}_q$ is denoted $\mathbb{F}_q[X]$.

FACT. A polynomial f of degree d over $\mathbb{F}_q$ has at most d roots.

"pf". (Sketch). If $f(\beta) = 0$, then $(X-\beta) \mid f$. So if $\beta_1, \ldots, \beta_{d+1}$ are roots of f, then $\underbrace{(X-\beta_1)(X-\beta_2)\cdots(X-\beta_{d+1})}_{\text{degree } d+1} \mid \underbrace{f}_{\text{degree} \leq d}$, a contradiction.

[This proof implicitly uses:

"Thm:" Arithmetic over $\mathbb{F}[X]$ behaves like you think it should.

That Theorem is true.]

EXAMPLES Over $\mathbb{F}_3$,

$f(X) = X^2 - 1$ has two roots. [$f(2) = f(1) = 0$]
$f(X) = X^2 + 2X + 1$ has one root. [$f(2) = 2^2 + 2\cdot 2 + 1 =$ "9" $= 0$]
$f(X) = X^2 + 1$ has zero roots. [$f(0) = 1$, $f(1) = 2$, $f(2) =$ "5" $= 2$]

Notice that X^2+1 DOES have a root over $\mathbb{F}_2$, so the field matters.

DEF. A VANDERMONDE MATRIX has the form

$$V = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^m \\ 1 & \alpha_2 & \alpha_2^2 & \cdots & \alpha_2^m \\ 1 & \alpha_3 & & & \\ \vdots & & & & \\ 1 & \alpha_n & \alpha_n^2 & \cdots & \alpha_n^m \end{bmatrix}$$

for some $\alpha_1, \dots, \alpha_n \in \mathbb{F}_q$. Aka, $V_{ij} = \alpha_i^{j-1}$.

distinct

[Note: I also use "Vandermonde" to refer to the transpose of a matrix of this form.]

FACT A square Vandermonde matrix is invertible.

proof 1. $V \cdot \vec{a} = \begin{pmatrix} \sum_i a_i \cdot \alpha_1^i \\ \sum_i a_2 \cdot \alpha_2^i \\ \vdots \\ \sum_i a_n \cdot \alpha_n^i \end{pmatrix} = \begin{pmatrix} f(\alpha_1) \\ f(\alpha_2) \\ \vdots \\ f(\alpha_n) \end{pmatrix}$ if $f(X) = a_0 + a_1 X + \cdots + a_{n-1} X^{n-1}$.

Since f is a nonzero polynomial of degree $\le n-1$, it doesn't have n roots, so $V \cdot \vec{a} \neq 0$ for all nonzero $\vec{a} \in \mathbb{F}_q^n$. Hence, $\mathrm{Ker}(V) = \emptyset$, so V is invertible.

proof 2. $\det(V) = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^n \alpha_i^{\sigma(i)-1} = \prod_{1 \le i < j \le n} (\alpha_j - \alpha_i)$

[The LHS is alternating, meaning that if you switch α_i and α_j, the sign flips. So $(\alpha_j - \alpha_i)$ divides it for all $i \neq j$, and then counting degrees says that has to be everything.]

Since $\alpha_i \neq \alpha_j$ $\forall i \neq j$, the RHS has no zero factors and so is nonzero. [this uses the fact from your HW that $\alpha \cdot \beta \neq 0$ if $\alpha, \beta \neq 0$].

COR. Any square submatrix of a Vandermonde matrix is invertible.

proof. A square submatrix looks like

$$\begin{bmatrix} \alpha_i^{j} & \alpha_i^{j+1} & \alpha_i^{j+2} & \cdots & \alpha_i^{j+r} \\ \alpha_{i+1}^{j} & & & & \alpha_{i+1}^{j+r} \\ \vdots & & & & \\ \alpha_{i+r}^{j} & & \cdots & & \alpha_{i+r}^{j+r} \end{bmatrix} = \underbrace{D}_{\mathrm{diag}(\alpha_i, \ldots, \alpha_{i+r})} \cdot \underbrace{V}_{\text{a square Vandermonde matrix.}}$$

These facts about Vandermonde matrices will be useful.
First, they imply:

THEOREM. "Polynomial interpolation works over $\mathbb{F}_q$."
Formally, given $(\alpha_i, y_i) \in \mathbb{F}_q \times \mathbb{F}_q$ for $i = 1, \ldots, d+1$, there is a unique degree polynomial f so that $f(\alpha_i) = y_i \;\forall i$.

proof. If $f(X) = a_0 + a_1 X + \cdots + a_d X^d$, then the requirements that $f(\alpha_i) = y_i \;\forall i$ are precisely

$$\boxed{V}\,\vec{a} = \vec{y}$$

for a square Vandermonde matrix V.

Hence, $a = V^{-1} y$ is the unique solution. (Because linear algebra "works" over $\mathbb{F}_q$.)

Moreover, the proof implies that we can find f efficiently.

Actually, VERY efficiently. You can do an FFT-like thing to multiply by Vandermonde matrices real fast.

FACT. All functions $f: \mathbb{F}_q \to \mathbb{F}_q$ are polynomials of degree $\leq q-1$.

proof. There are only q pts in $\mathbb{F}_q$, so we can interpolate a (unique) degree $\leq q-1$ polynomial through any function.

[Second proof: there are q^q such functions and also q^q such polynomials]

EXAMPLE. $f(X) = X^q$ must have some representation as a degree $\le q-1$ poly over $\mathbb{F}_q$. What is it?

ANSWER: $X^q \equiv X$. This is because FACT: $\alpha^q = \alpha \; \forall \alpha \in \mathbb{F}_q$

Useful fact! Let's call it $(*)$. We won't prove it but we will use it a bunch. [It's not hard to prove: Check out the supplementary material on finite fields]

Now we are finally ready to define...

DEF. (REED-SOLOMON CODES)

Let $n \ge k$, $q \ge n$. The REED-SOLOMON CODE of dimension k over $\mathbb{F}_q$, with evaluation points $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$, is

$$RS_q(\vec{\alpha}, n, k) = \{ (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n)) : f \in \mathbb{F}_q[X], \; \deg(f) \le k-1 \}$$

Sometimes I'll just write $RS(n,k)$.

NOTE: This definition implies a natural encoding map for RS codes:

$$x = (x_0, \ldots, x_{k-1}) \longmapsto (f_x(\alpha_1), \ldots, f_x(\alpha_n)), \quad \text{where } f_x(X) = x_0 + x_1 \cdot X + \cdots + x_{k-1} X^{k-1}$$

[We've been 1-indexing but here it is convenient to zero-index].

This isn't the ONLY encoding map, but it's the one we will think about for most of the class.

PROP. $RS_q(\vec{\alpha}, n, k)$ is a linear code, and the generator matrix is the $n \times k$ Vandermonde matrix with rows corresponding to $\alpha_1, \alpha_2, \ldots, \alpha_n$.

proof. Staring. (If x has the coefficients of f, then $V \cdot f = \begin{pmatrix} f(\alpha_1) \\ \vdots \\ f(\alpha_n) \end{pmatrix}$.)

Notice: Since V has rank k, this implies that $\dim(RS(n,k)) = k$)

PROP The distance of $RS_q(n,k)$ is $d = n-k+1$.

Proof. Since $RS_q(n,k)$ is linear, $\mathrm{dist}(RS_q(n,k)) = \min_{c \in RS} \mathrm{wt}(c)$.

The minimum weight of any codeword is at least $n-k+1$, since any degree $k-1$ polynomial has at most $k-1$ roots.

Equivalent proof: the follows from the fact that every $k \times k$ minor of the generator matrix is full rank.

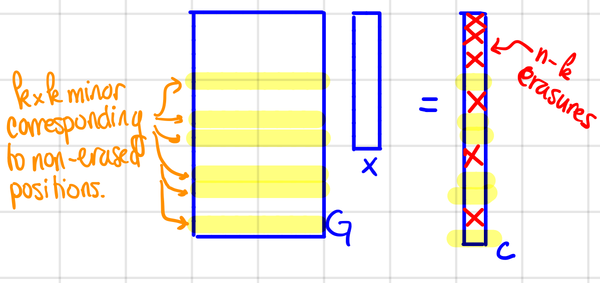
COR. RS codes exactly meet the Singleton Bound.

YAY! OPTIMALITY!! For any n and k we like!

DEF. A linear $(n,k,d)_q$ code with $n = k+d-1$ (aka, meeting the Singleton bd) is called MAXIMUM DISTANCE SEPARABLE. (MDS)

So, RS codes are MDS. Notice that MDS-ness is equivalent to the property: "every $k \times k$ minor of the generator matrix is full rank," which we just saw was true for RS codes.

In particular, if $\mathcal{C}$ is MDS, then any k positions of $c \in \mathcal{C}$ determine all of c.



Distance $n-k+1$ $\Leftrightarrow$ can correct any $n-k$ erasures

$\Leftrightarrow$ any $k \times k$ minor of G is invertible.

Notice that q must be growing in order to get an MDS code (by the Plotkin bound). How big does q have to be? OPEN QUESTION!

example that this is possible: $G = \begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & 0 \\ \alpha_1 & \alpha_2 & & \alpha_q & 0 & 1 \\ \alpha_1^2 & \alpha_2^2 & & \alpha_q^2 & 1 & 0 \end{bmatrix}^T$ is MDS if $q = 2^h$, and $n = q+2$.

CONJECTURE ("MDS CONJECTURE"). If $k \le q$, then $n \le q+1$, unless ($q = 2^h$ and $k=3$) or $k = q-1$, in which case $n \le q+2$.
(from 1955)

aka, RS codes basically have the smallest alphabet size w/ $n = q$.

③ DUAL VIEW of RS CODES

What is the parity-check matrix of an RS code?
We'll need a bit more algebra.

DEF $\mathbb{F}_q^*$ is the multiplicative group of nonzero elements in $\mathbb{F}_q$.

Aka, $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$ as a set, and I can define multiplication and division everywhere in $\mathbb{F}_q^*$.

EXAMPLE. $\mathbb{F}_5 = \{0,1,2,3,4\}$ mod 5 equipped w/ $+$ and $*$
$\mathbb{F}_5^* = \{1,2,3,4\}$ mod 5 equipped w/ just $*$.

FACT. $\mathbb{F}_q^*$ is CYCLIC, which means there's some $\gamma \in \mathbb{F}_q^*$ so that

$$\mathbb{F}_q^* = \{\gamma, \gamma^2, \gamma^3, \ldots, \gamma^{q-1}\}$$

γ is called a PRIMITIVE ELEMENT of $\mathbb{F}_q$.

EXAMPLE. 2 is a primitive element of $\mathbb{F}_5$, and

$$\mathbb{F}_5^* = \{2,\ 2^2=4,\ 2^3=3,\ 2^4=1\}$$

4 is NOT a primitive element, since $4^2=1, 4^3=-1, 4^4=1, 4^5=-1, \ldots$ and we'll never generate 2 or 3 as a power of 4.

FUN EXERCISE:
If you haven't seen this before, play around w/ this and other examples. What elements of $\mathbb{F}_p$ are primitive? If an element isn't primitive, what can you say about its ORBIT $\{\gamma^i : i=1,2,3,\ldots\}$?

FACT / LEMMA. For any $0 < d < q-1$, $\sum_{\alpha \in \mathbb{F}_q} \alpha^d = 0$.

Proof. $\sum_{\alpha \in \mathbb{F}_q} \alpha^d = \sum_{\alpha \in \mathbb{F}_q^*} \alpha^d$

$= \sum_{j=0}^{q-2} (\gamma^j)^d$ for a primitive element γ.

$= \sum_{j=0}^{q-2} (\gamma^d)^j$

For any $x \neq 1$,
$(1-x) \cdot (\sum_{j=0}^{n-1} x^j) = 1 - x^n$,
and so $\sum_{j=0}^{n-1} x^j = \frac{1-x^n}{1-x}$
for any n. Apply this with $x = \gamma^d$.

$= \dfrac{1 - (\gamma^d)^{q-1}}{1 - \gamma^d}$

$(\gamma^d)^{q-1} \cdot \gamma^d = (\gamma^d)^q = \gamma^d$,
using $(*)$ again.
So $(\gamma^d)^{q-1} = 1$. (since $\gamma^d \neq 0$).

$= \dfrac{1-1}{1-\gamma^d} = 0.$

Now we can answer our question about the parity-check matrix of RS codes.

PROP. Let $n = q-1$, and let γ be a primitive element of $\mathbb{F}_q$.

$$RS_q((\gamma^0, \gamma^1, \gamma^2, \ldots, \gamma^{n-1}), n, k)$$

$$= \{ (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_q^n : c(\gamma^j) = 0 \text{ for } j = 1, 2, \ldots, n-k \}$$

where $c(X) = \sum_{i=0}^{n-1} c_i \cdot X^i$.

COR. The parity check matrix of $RS_q((\gamma^0, \ldots, \gamma^{n-1}), n, k)$ is

$$H = \begin{bmatrix} 1 & \gamma & \gamma^2 & \cdots & \gamma^n \\ 1 & \gamma^2 & \gamma^4 & \cdots & \gamma^{2(n-1)} \\ \vdots & & & & \\ 1 & \gamma^{n-k} & \gamma^{2(n-k)} & \cdots & \gamma^{(n-k)(n-1)} \end{bmatrix} \in \mathbb{F}_q^{(n-k) \times n}$$

Proof of PROP. It suffices to show that

$$
n-k\left\{\underbrace{\begin{bmatrix} 1 & \gamma & \gamma^2 & \cdots & \gamma^n \\ 1 & \gamma^2 & \gamma^4 & & \gamma^{2(n-1)} \\ \vdots & & H & & \\ 1 & \gamma^{n-k} & \gamma^{2(n-k)} & \cdots & \gamma^{(n-k)(n-1)} \end{bmatrix}}_{n=q-1}\right.
\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \gamma & \gamma^2 & \cdots & \gamma^{k-1} \\ & \gamma^2 & \gamma^4 & \cdots & \gamma^{2(k-1)} \\ \vdots & & & & \\ & & G & & \\ 1 & \gamma^{n-1} & \cdots & & \gamma^{(n-1)(k-1)} \end{bmatrix}}_{k} = 0
$$

So let's just consider the (i,j) entry of the product. This is

$$
\begin{bmatrix} 1 & \gamma^i & \gamma^{2i} & \gamma^{3i} & \cdots & \gamma^{(n-1)i} \end{bmatrix} \cdot \begin{bmatrix} \gamma^{0\cdot j} \\ \gamma^j \\ \gamma^{2\cdot j} \\ \vdots \\ \gamma^{(n-1)j} \end{bmatrix} = \sum_{l=0}^{n-1} \gamma^{li} \cdot \gamma^{lj}
$$

$$
\begin{aligned}
&= \sum_{l=0}^{n-1} \gamma^{l(i+j)} \\
&= \sum_{l=0}^{n-1} (\gamma^l)^{(i+j)} \\
&= \sum_{\alpha \in \mathbb{F}_q^*} \alpha^{(i+j)} \\
&= 0
\end{aligned}
$$

since $i+j \leq (n-k)+k = n = q-1 < q$.

[and $i+j > 0$ since $i > 0$] ∎

NOTICE: $RS(n,k)^\perp$ has generator matrix H^T, which again looks a lot like a Vandermonde matrix! So $RS(n,k)^\perp$ is again (kind of) an RS code!

This particular derivation used the choice of eval. pts heavily. However, a statement like this is true in general.

$\vec{\lambda} \in (\mathbb{F}_q^*)^n$

DEF. A GENERALIZED RS CODE $GRS_q(\vec{\alpha}, n, k; \vec{\lambda})$ is

$$GRS_q(\vec{\alpha}, n, k; \vec{\lambda}) := \left\{ \left(\lambda_0 f(\alpha_0), \lambda_1 f(\alpha_1), \dots, \lambda_n f(\alpha_n) \right) \,\middle|\, f \in \mathbb{F}_q[X],\ \deg(f) \leq k-1 \right\}.$$

THM.

$$GRS_q(\vec{\alpha}, n, k; \vec{\lambda})^{\perp} = GRS_q(\vec{\alpha}, n, n-k, \vec{\sigma})$$

for some $\vec{\sigma} \in (\mathbb{F}_q^*)^n$.

Proof: Fun exercise!

QUESTIONS TO PONDER

① How would you modify RS codes to make them binary?

② How would you decode RS codes from errors efficiently? Can we do this?