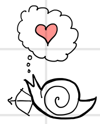
CS 250/EE387 - LECTURE 15 - REED-MULLER CODES!

AGENDA

1. Recall Reed-Muller Codes
2. Decoding binary RM codes: Reed's Algorithm.
3. Larger field sizes?

GASTROPOD FACT

When slugs or snails mate, they shoot each other with "love darts" as part of the courtship ritual. It's not well understood the function that these serve, but it's thought that they increase the likelihood of fertilization.



1. RECALL REED-MULLER CODES

Reed-Muller codes are the generalization of RS codes to multiple variables.

Recall that $\mathbb{F}_q[X_1, \ldots, X_m]$ is the space of m-variate polynomials over $\mathbb{F}_q$.
The (total) DEGREE of a monomial $X_1^{i_1} X_2^{i_2} \cdots X_m^{i_m}$ is $\sum_{j=1}^m i_j$.
The DEGREE of $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ is the largest degree of any monomial in f.

DEF. The m-VARIATE REED-MULLER CODE of DEGREE r over $\mathbb{F}_q$ is

$$RM_q(m,r) = \left\{ \left(f(\vec{\alpha}_1), \ldots, f(\vec{\alpha}_{q^m})\right) : f \in \mathbb{F}_q[X_1, \ldots, X_m],\ \deg(f) \le r \right\}$$

REMARK. Note that we may assume that each X_i has degree $< q$, since $\alpha = \alpha^q$ for all $\alpha \in \mathbb{F}_q$.

We saw BINARY RM CODES back in Lecture 6 when we were trying to figure out how to get good binary codes.

PROPERTIES of $RM_q(m,r)$:

- **Block length:** q^m [number of pts in $\mathbb{F}_q^m$]

- **Dimension:** #coefficients in a degree-r m-variate polynomial.

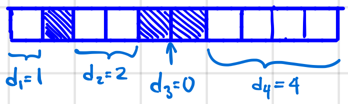
  - If $q=2$: The possible monomials are $\prod_{i \in S} X_i$ for $S \subseteq [m]$, $|S| \leq r$.

    There are $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r}$ of those, so that's the dimension.

  - If $q>r$: The possible monomials are $\prod_{i \in [m]} X_i^{d_i}$ for $\sum_{i=1}^m d_i \leq r$.

    There are $\sum_{j \leq r} \binom{j+m-1}{m-1}$ of those, so that's the dimension.

    ← ASIDE: To see this, notice that picking $d_1, \ldots, d_m$ is the same as picking $m-1$ "dividers" in an array of length $r+m-1$:



    (eg, if $m=4$ and $r=7$)

  - If $2<q<r$: Then some of the monomials like ↑ are not allowed since $d_i < q$.
    In this case there is no nice expression for dim, it's just $|\{(i_1, \ldots, i_m) : \sum_{j=1}^m i_j \leq r \text{ and } i_j < q \ \forall j\}|$.

- **Distance**

Just as with RS codes, RM codes have decent distance because low-degree (multivariate) polynomials don't have too many roots.

The SCHWARTZ–ZIPPEL LEMMA tells us how many they have, and the result is that

↖ See ESSENTIAL CODING THEORY for details.

$$\text{DISTANCE}(RM_q(m,d)) = q^{m-a}\left(1 - b/q\right) \quad \text{where } r = a(q-1)+b, \quad 0 \leq b < q-1.$$

PARSING THIS: if $q=2$, this is q^{m-r} [what we saw before; $\delta = 1/q^r$]

if $q>r$, this is $q^m(1-r/q)$ [$\delta = (1-r/q)$].

EXAMPLES of RM CODES we have ALREADY SEEN:

- $RM_2(m, r)$ is the binary RM code from LECTURE 6.

  Rate $= \frac{1}{2^m} \cdot \left(\binom{m}{0} + \cdots + \binom{m}{r} \right) \approx 2^{-m(1 - H_2(r/m))}$.

  Rel. Distance $= 2^{-r}$

- $RM_q(1, r)$ is just $RS_q(\mathbb{F}_q, n=q, r+1)$

  Rate $= (r+1)/q$

  Rel. dist $= 1 - r/q$

- $RM_q(m, 1)$ is the HADAMARD CODE (which we saw on HW; dual of the Hamming code if $q=2$)

  Rate is m/q^m

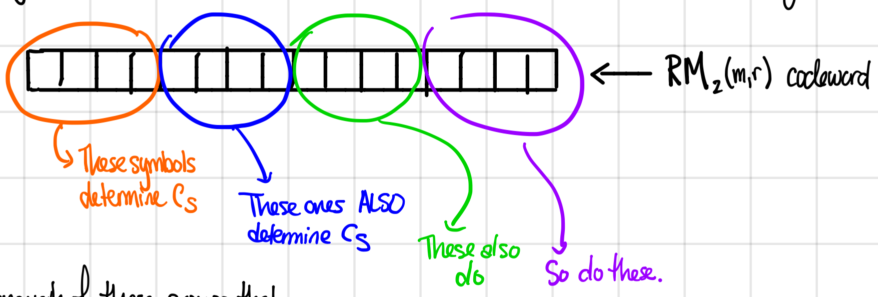
  Rel. dist $= 1 - 1/q$

① DECODING BINARY RM CODES: Reed's Algorithm.

Consider an m-variate poly $f \in \mathbb{F}_2[X_1, \ldots, X_m]$, $\deg \leq r$.
It looks like this:

$$f(X_1, \ldots, X_m) = \sum_{\substack{T \subseteq [m] \\ |T| \leq r}} c_T \cdot X^T, \quad \text{where } X^T := \prod_{i \in T} X_i \text{ and } c_T \in \mathbb{F}_2.$$

THE PLAN: We will try to figure out each coefficient c_S, one-at-a-time.
More precisely, we will see that for each S, there is some partition of symbols s.t.:



There will be enough of these groups that $< \frac{1}{2}$ of them have any errors in them.

So we can just take the "MAJORITY VOTE" of all the groups.

1A WARM-UP: HADAMARD CODES $(r=1)$.

Let's try to decode $RM_2(m,1)$ from $< \frac{2^{m-1}}{2} = 2^{m-2}$ errors.

aka, distance/2

A codeword $w \in RM_2(m,1)$ is given by

$$w = \left(f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_{2^m}) \right) \text{ for some deg-1 } f.$$

Any* degree-1 poly f looks like $f(X_1, \ldots, X_m) = \sum_i c_i X_i$ for $c_i \in \mathbb{F}_2$.

CHEATING! I'm not allowing a constant term. We'll see how to fix this in a moment.

So
$$w = \left(\boxed{\vec{c}} \begin{bmatrix} 0\\0\\0\\0\\0 \end{bmatrix}, \boxed{\vec{c}} \begin{bmatrix} 0\\0\\0\\0\\1 \end{bmatrix}, \boxed{\vec{c}} \begin{bmatrix} 0\\0\\0\\1\\0 \end{bmatrix}, \ldots \right)$$

$\vec{c} = (c_1, \ldots, c_m)$

iterating through all binary vectors.

$$= \left(\langle c, \alpha_1 \rangle, \langle c, \alpha_2 \rangle, \ldots, \langle c, \alpha_{2^m} \rangle \right).$$

IDEA: Recover each c_i, one-at-a-time.

OBSERVATION: $c_i = \langle c, e_i \rangle = \langle c, \beta \rangle + \langle c, \beta + e_i \rangle$ for any $\beta \in \mathbb{F}_2^m$.

This is true by linearity.

This inspires an algorithm:

ALG.

Input: $g: \mathbb{F}_2^m \to \mathbb{F}_2$ s.t. $\Delta(g,f) < 2^{m-2}$ for some $f \in RM_2(m,1)$

Output: f.

For $i = 1, \ldots, m$:

  For each $\beta \in \mathbb{F}_2^m$:

    Let $\hat{c}_i(\beta) = g(\beta) + g(\beta + e_i)$

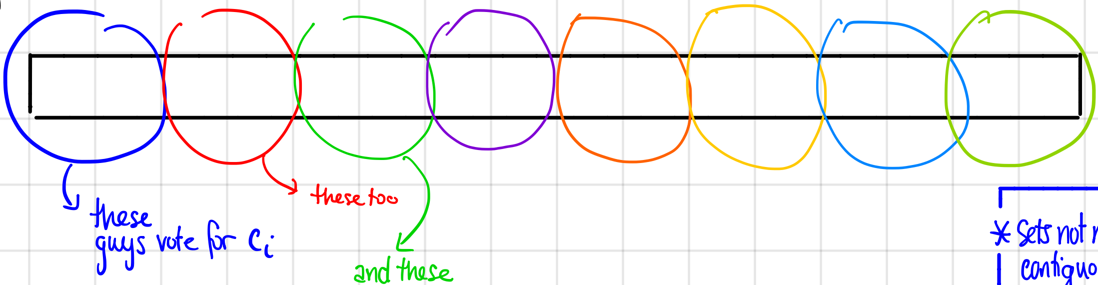
  Set $\hat{c}_i = \mathrm{MAJ}\{ \hat{c}_i(\beta) : \beta \in \mathbb{F}_2^m \}$

RETURN $f(X_1, \ldots, X_m) = \sum_i \hat{c}_i X_i$.

Why does this algorithm work?

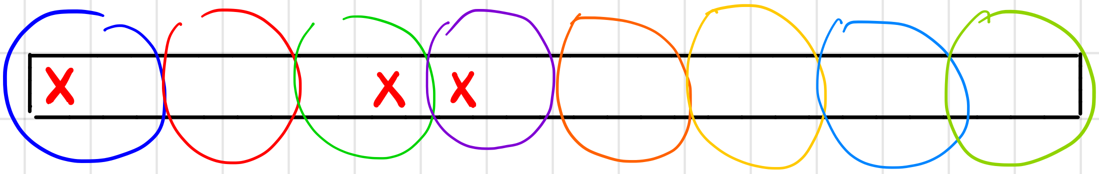
- A vote $\hat{c}_i(\beta)$ is correct as long as neither of the queries β or $\beta + e_i$ were corrupted.

- Notice that the collection of sets $\left\{ \{\beta, \beta+e_i\} : \beta \in \mathbb{F}_2^m \right\}$ partition $\mathbb{F}_2^m$ into 2^{m-1} sets of size 2:



* Sets not necessarily contiguous.

Now there are 2^{m-1} sets and $< 2^{m-2} = \frac{2^{m-1}}{2}$ errors.
So $< 2^{m-1}/2$ of the sets have errors.



So $< \frac{1}{2}$ of the votes are INCORRECT, meaning $> \frac{1}{2}$ are CORRECT!
So the majority vote is always correct and we win.

Now let's extend this to $m > 1$.

Once again, for each coefficient c_T in $\sum_{\substack{|T| \le r \\ T \subseteq [m]}} c_T X^T$,
we will come up with a bunch of disjoint groups of symbols which will cast a vote for c.

The groups we choose will be

"all the evaluation pts that look like $\underbrace{\overbrace{[\,\text{something}\,}^{S}\,|\,\overbrace{\text{— }\beta\text{ —}\,]}^{[m]\setminus S}}_{m}$" for various $\beta \in \mathbb{F}_2^{m-r}$

LEMMA Let $S, T \subseteq [m]$, $|S| = r$ and $|T| \leq r$. Then $\forall\ \beta \in \mathbb{F}_2^{m-|S|}$,

$$\sum_{\substack{\alpha \in \mathbb{F}_2^m \\ \alpha|_{[m]\setminus S} = \beta}} \alpha^T = \begin{cases} 1 & \text{if } S = T \\ 0 & \text{otherwise} \end{cases}$$

Proof. First, suppose that $T = S$. Then

$$\alpha = [\,\overbrace{\gamma}^{S=T}\,|\,\overbrace{\beta}^{m\setminus[S]}\,] \quad (\text{IF } S=T)$$

$$\sum_{\substack{\alpha \in \mathbb{F}_2^m \\ \alpha|_{[m]\setminus S} = \beta}} \alpha^S = \sum_{\gamma \in \mathbb{F}_2^S} \gamma^S = \mathbb{1}^S = 1$$

β doesn't matter at all

If γ has a 0 in it then $\gamma^S = 0$, so only $\mathbb{1}$ survives.

OTOH, say $T \neq S$.
Then $S \setminus T$ is nonempty, since $|T| \leq r = |S|$.
Then

$$\alpha = [\,\overbrace{\underbrace{\square}_{S\setminus T \text{ is nonempty.}}\ \underbrace{\square\square}_{T}}^{S}\,|\,\overbrace{\quad}^{m\setminus[S]}\,] = \alpha$$

$$\alpha = [\,\overbrace{\delta}^{S\setminus T}\,|\,\overbrace{\gamma}^{T\cap S}\,|\,\overbrace{\beta}^{\bar{S}}\,] \quad (\text{IF } S \neq T)$$

$$\sum_{\substack{\alpha \in \mathbb{F}_2^m \\ \alpha|_{[m]\setminus S} = \beta}} \alpha^T = \sum_{\substack{\delta \in \mathbb{F}_2^{S\setminus T} \\ \gamma \in \mathbb{F}_2^{S\cap T}}} \gamma^{T\cap S} \cdot \beta^{\bar{S}} \quad \text{(see picture)}$$

$$= \sum_{\delta \in \mathbb{F}_2^{S\setminus T}} \underbrace{\left(\sum_{\gamma \in \mathbb{F}_2^{S\cap T}} \gamma^{T\cap S} \cdot \beta^{\bar{S}} \right)}_{\text{doesn't depend on } \delta}$$

$$= \left(\sum_{\gamma \in \mathbb{F}_2^{S\cap T}} \gamma^{T\cap S} \cdot \beta^{\bar{S}} \right) \cdot \underbrace{\left(\sum_{\delta \in \mathbb{F}_2^{S\setminus T}} 1 \right)}_{\substack{\equiv 0 \text{ in } \mathbb{F}_2, \text{ since } |S\setminus T| \geq 1, \\ \text{and so } |\mathbb{F}_2^{S\setminus T}| \text{ is even.}}}$$

$$= 0.$$

COR. Let $f(X_1,\dots,X_m) = \sum'_{\substack{T\subseteq[m]\\|T|\leq r}} c_T X^T$. Then

$$\sum_{\substack{\alpha\in\mathbb{F}_2^m\\ \alpha|_{\bar{S}}=\beta}} f(\alpha) = c_S .$$

Proof.

$$\sum_{\substack{\alpha\in\mathbb{F}_2^m\\ \alpha|_{\bar{S}}=\beta}} f(\alpha) = \sum_{\substack{\alpha\in\mathbb{F}_2^m\\ \alpha|_{\bar{S}}=\beta}} \sum_{\substack{T\subseteq[m]\\|T|\leq r}} c_T\alpha^T$$

$$= \sum_{\substack{T\subseteq[m]\\|T|\leq r}} c_T \left(\sum_{\substack{\alpha\in\mathbb{F}_2^m\\ \alpha|_{\bar{S}}=\beta}} \alpha^T \right)$$

This vanishes for all $T\neq S$, and is 1 for $T=S$

$$= c_S$$

This inspires an algorithm:

ALG. 1 (NOT the final version)

Input: $g:\mathbb{F}_2^m\to\mathbb{F}_2$ s.t. $\Delta(g,f) < 2^{m-r-1}$ for some $f\in\mathbb{F}_2[X_1,\dots,X_m]$ w/ $\deg(f)\leq r$, (and the parameter r)

This is half the distance of $RM_2(m,r)$

Output: c_S for $|S|=r$, where $f(X) = \sum_{\substack{S\subseteq[m]\\|S|\leq r}} c_S X^S$.

for $S\subseteq\mathbb{F}_2^m$ with $|S|=r$:

  for $\beta\in\mathbb{F}_2^{m-r}$:

    compute a guess $\hat{c}_S(\beta) = \sum_{\substack{\alpha\in\mathbb{F}_2^m\\ \alpha|_{\bar{S}}=\beta}} g(\alpha)$

  set $\hat{c}_S = \mathrm{MAJ}\{\hat{c}_S(\beta) : \beta\in\mathbb{F}_2^{m-r}\}$

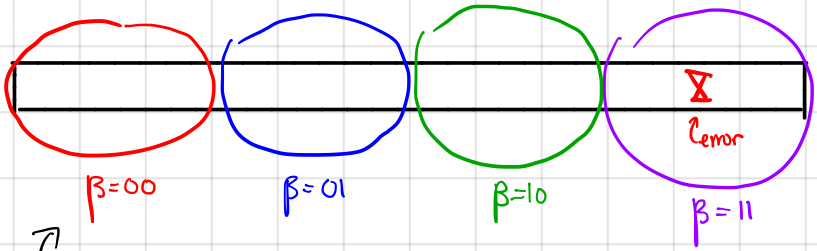
.

Notice that ALG1 doesn't necessarily find f, it only finds c_S with $|S|=r$. We'll come back to that.

PROP. ALG1 is correct.

Proof. Let $E \subseteq \mathbb{F}_2^m$ be the set of errors between f and g, so $|E| < 2^{m-r-1}$.
Notice that the guess $\hat{c}_S(\beta)$ is correct provided that all the points α s.t. $\alpha|_{\bar{S}} = \beta$ were not in error, aka if $E \cap \{\alpha \in \mathbb{F}_2^m : \alpha_{\bar{S}} = \beta\} = \phi$.

The sets $\{\alpha \in \mathbb{F}_2^m : \alpha|_{\bar{S}} = \beta\}$ are all disjoint, and there are 2^{m-r} of them.
Since $|E| < 2^{m-r-1}$, strictly fewer than ½ of these sets intersect E.



There are 2^{m-r} disjoint sets, and $< 2^{m-r-1}$ errors,
so $< 2^{m-r-1}$ sets have an error in them.

So $> \frac{1}{2}$ of the $\hat{C}_S(\beta)$ are correct, and MAJ returns the correct answer.

Now we just need to be able to recover ALL of the coeffs...

ALG. (REED'S MAJORITY LOGIC DECODER)

Input: $g: \mathbb{F}_2^m \to \mathbb{F}_2$ s.t. $\Delta(g,f) < 2^{m-r-1}$ for some $f \in \mathbb{F}_2[X_1,\dots,X_m]$ w/ $\deg(f) \leq r$,
(and the parameter r)

Output: c_S for $|S| \leq r$, where $f(X) = \sum_{\substack{S \subseteq [m] \\ |S| \leq r}} c_S X^S$.

for $j = r, r-1, \dots, 1$:

- Run **ALG 1** on $g(X_1,\dots,X_m)$ with degree $= j$ to find all coefficients $\hat{c}_S$, $|S| = j$.
- $g(\bar{X}) \leftarrow g(\bar{X}) - \sum_{|S|=j} \hat{c}_S \bar{X}^S$

Return $\sum_{|S| \leq r} \hat{c}_S \cdot X^S$

Running time? $O\Big(\sum_{j=1}^{r} \underbrace{\binom{m}{j}}_{\text{Iterate over all sets } T} \cdot \underbrace{2^{m-j}}_{\text{Iterate over all } \beta\text{'s}} \cdot \overbrace{2^j}^{\text{summing up } 2^j\ \alpha\text{'s}} \Big) = O\left(2^m \cdot 2^{m H_2(r/m)}\right)$

$= O(N^2)$ where $N = 2^m$ is the block length.

So Reed's Alg runs in time polynomial in the block length.

FUN EXERCISE: Can Reed's Alg be modified to work over larger fields?

Notice that this algorithm has an additional nice property: it's LOCAL, in the sense that we recover one symbol c_S at a time.

This idea will come back in the next lecture.

② Larger fields.

NOTE. We did not get to part ②, in class, it will be partially re-hashed in Lecture 16.

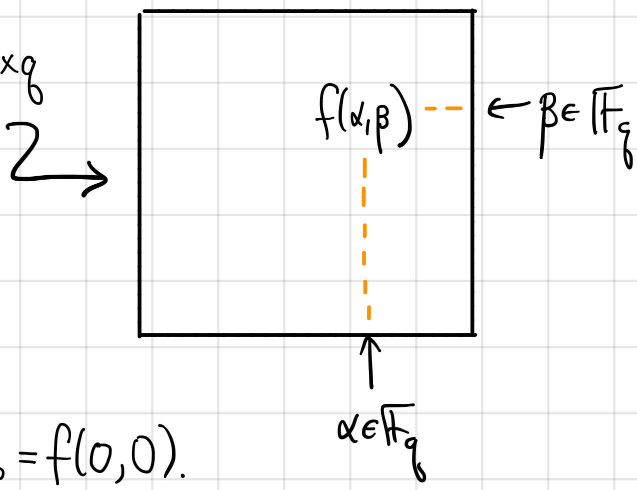
Now let's find a way to generalize this basic framework to larger fields.
(We'll deviate a bit from the specific approach we just saw).

Let's say that $q > r$, so we're in that other regime where $\delta = (1 - r/q)$.

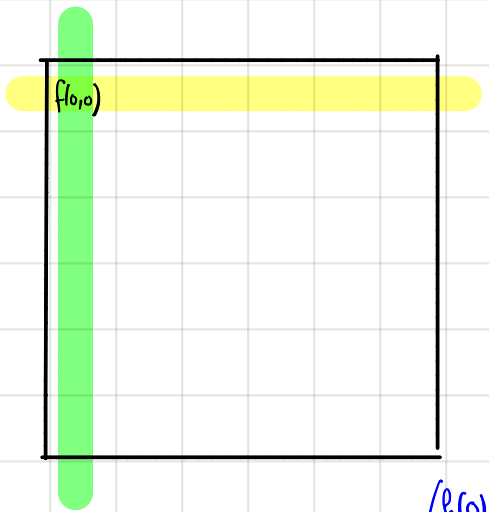
Consider $RM_q(2, r)$, so bivariate polynomials:

$$f(X, Y) = \sum_{i+j \leq r} c_{i,j} X^i Y^j .$$

We can think of codewords as $q \times q$ grids of evaluation points.



Suppose I want to recover $c_{00} = f(0,0)$.
As before, we want to find a bunch of LOCAL, LINEAR relationships involving $f(0,0)$.



$\leftarrow$ This row is $\left(f(0,0),\ f(0,\gamma), \ldots, f(0,\gamma^{q-1}) \right)$

$=: \left(g(0),\ g(\gamma), \ldots, g(\gamma^{q-1}) \right)$

where $g(Y) := f(0, Y) = \sum_{i+j \leq r} c_{ij} 0^i \cdot Y^j$

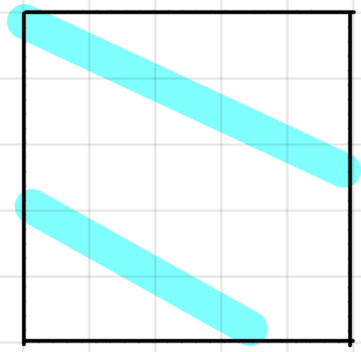
$= \sum_{j \leq r} c_{0j} Y^j$

Similarly, this column is $\begin{pmatrix} h(0) \\ h(\gamma) \\ \vdots \\ h(\gamma^{q-1}) \end{pmatrix}$ where $h(X) = \sum_{j \leq r} c_{j0} X^j$

Hey, those are RS codewords! That's a real nice linear relationship!

Moreover, the restriction of f to ANY line is an RS codeword!

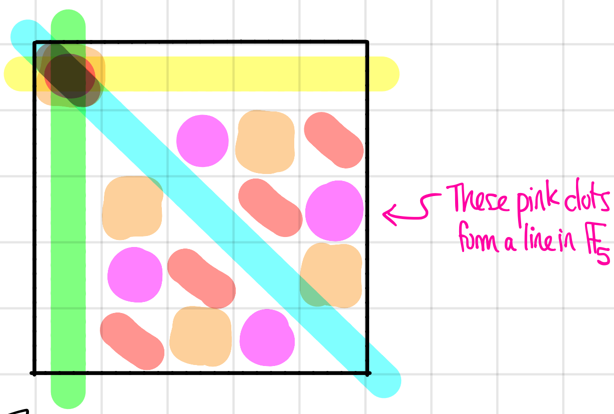
Consider the line $L(Z) = (a_1 Z + b_1,\ a_2 Z + b_2)$, $a_i, b_i \in \mathbb{F}_q$.



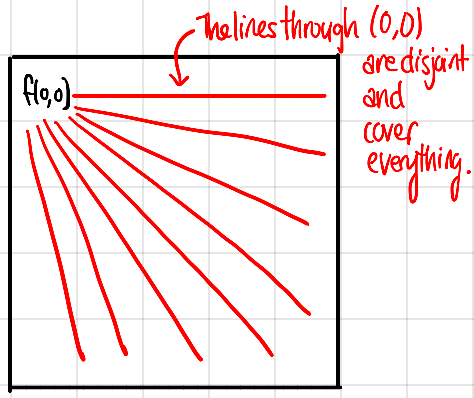
Then $f(L(Z)) = \sum_{i+j \le r} (a_1 Z + b_1)^i (a_2 Z + b_2)^j$

$= \text{some degree} \le r \text{ polynomial in } Z.$

So if we are looking for lots of disjoint sets through $f(0,0)$ that tell us something about $f(0,0)$, here they are!



Confusing but more accurate



Less accurate but hopefully more clear.

This inspires an algorithm.

A bit worse than half the distance...

ALG. Input: $h : \mathbb{F}_q^2 \to \mathbb{F}_q$ s.t. $\exists f \in RM_q(2,r)$, $\Delta(f,g) < \frac{1}{4} q^2 (1 - r/q)$

and a position α, β

Output: $f(\alpha, \beta)$.

For each line $L : \mathbb{F}_q \to \mathbb{F}_q^2$ so that $L(0) = (\alpha, \beta)$:

- Let $g(z) := h(L(z))$
- Find the unique deg $\le r$ polynomial $p(z)$ s.t. $\Delta(p,g) < \frac{q-r}{2}$ (if it exists), using your favorite RS decoding alg.
- Set $\hat{f}_{\alpha,\beta}(L) \leftarrow p(0)$

RETURN $\mathrm{MAJ}\{ \hat{f}_{\alpha,\beta}(L) : \text{lines } L \}$.

Same analysis as before:

- There are q lines through (α,β), disjoint except for (α,β).
- The RS decoder is correct if there are $< \frac{q-r}{2}$ errors on a line.
- There are $< q\left(\frac{q-r}{4}\right)$ errors total
- So $< \frac{1}{2}$ the lines return the wrong answer, so MAJ is correct.

CONCLUSION: This alg can correct up to $\frac{1}{4} q^2 (1 - r/q) = \text{distance}/4$ errors in $RM_q(2,r)$.

This is NOT optimal (and there are algs that do better), but it's a good warm-up for next time, when we'll observe that this algorithm is REALLY local.

QUESTIONS to PONDER

1. Can you get Reed's algorithm to run faster?
2. Can you adapt Reed's alg to larger fields?
3. Can you fix the large-field alg. we gave to work up to distance/2 ?