Error Control Codes from Algebra and Geometry

John B. Little

Department of Mathematics and Computer Science, College of the Holy Cross

little@mathcs.holycross.edu

TAGS Workshop

College Station, TX

May 17 - 19, 2006

Plan for the Workshop

- §1 – Error - Correcting Codes ("applied algebra")

- §2 – Codes with "extra structure": cyclic and multicyclic codes

- §3 – More on structure of finite fields, Reed-Solomon and BCH codes

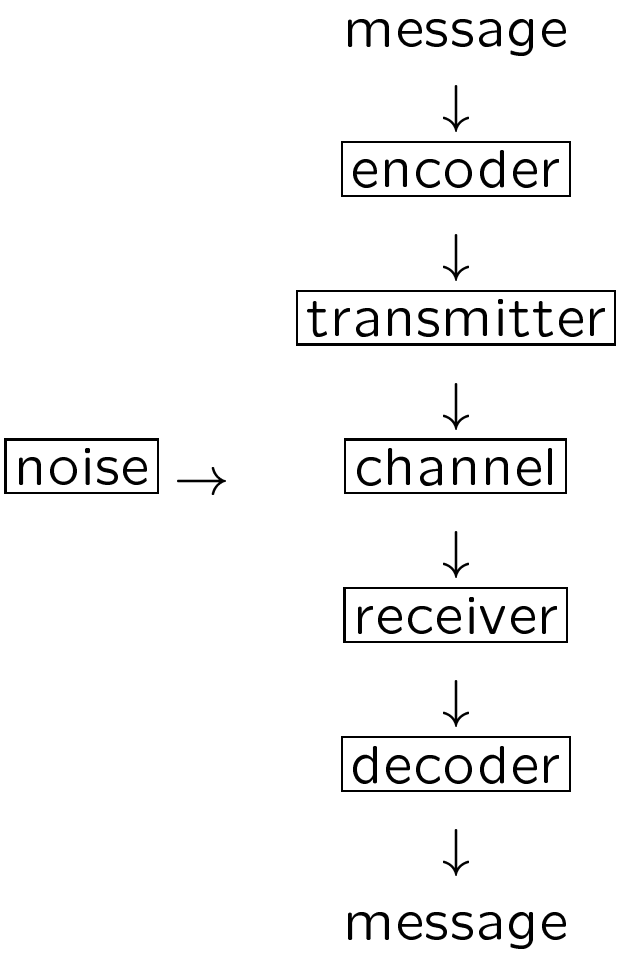
- §4 – Encoding and decoding algorithms via Gröbner bases

- §5 – Algebric-geometric Goppa codes and codes from order domains

- §6 – The Berlekamp-Massey-Sakata decoding algorithm

§1. Error-Correcting Codes

Schematic picture of communication:



- A main goal of coding theory is the design of coding schemes which achieve *error control*: use to detect and correct errors in received messages.

An Example

"If you cen rebd this, then yop're dojng error-corredtion."

In natural languages like English, words are usually "far enough apart" that even if some of a message is corrupted, it is still intelligible.

In the systems used for other types of communication, similar robustness in the presence of "noise" is a very desirable feature.

Mathematical Setting

Messages

- are divided into "words" or blocks of a fixed length, k,

- use symbols from a finite *alphabet* A with q symbols

Simplest case (also best adapted to electronic hardware) is an alphabet with two symbols: $A = \{0, 1\}$, identified with the finite field $\mathbf{F}_2$ (addition and multiplication modulo 2 – for instance $1 + 1 = 0$), but we will see others in a few moments also.

Usually, all strings or k-tuples in A^k are considered as possible words that can appear in a message.

Encoding and Decoding

To correct errors, some *redundancy* must be built into the encoded form of the message.

One way is to make the encoded message consist of strings or n-tuples of length $n > k$ over the same alphabet. Then encoding and decoding operations are functions:

$$E : A^k \to A^n$$

and

$$D : A^n \to A^k$$

where E is 1-1, and $D \circ E = I$ on A^k. (D might also take a "fail" or "not decoded" value on some words in the complement of $Im(E)$ containing too many errors to be decodable.)

We call $C = Im(E)$ the set of *codewords*, or just "the code." Any C of this form is called a *block code* of length n.

Errors

When an error is introduced in a word sent over the channel, the effect is to replace a codeword x by a received word $x' \neq x$.

If the alphabet A has a sum operation satisfying usual algebraic rules (commutativity, associativity, existence of a 0 element and additive inverses), then we can think of x' as a vector sum $x' = x + e$, where $e \in A^n$ is the *error vector*. $wt(e)$ determines how many entries of x are corrupted, and decoding is the same as determining e, then subtracting it off to recover x.

We will restrict to this case from now on, and in particular assume A is a finite field with q elements for some q. The case $q = 2$ is the binary field $\mathbf{F}_2 = \{0, 1\}$ with addition *modulo 2*:

$$0 + 0 = 0, 1 + 0 = 0 + 1 = 1, 1 + 1 = 0.$$

Hamming Distance

For errors to be correctable, codewords must be widely enough "separated" using:

Definition. (Hamming Distance) Let $x, y \in \mathbf{F}_q^n$. Then

$$\begin{aligned} d(x,y) &= |\{i \in \{1,\ldots,n\} : x_i \neq y_i\}| \\ &= \text{number of non-zero entries in } x-y. \end{aligned}$$

Example: $d(11000111, 10100101) = 3$

$d(x,y)$ is a *metric* or *distance function* (on the finite set $\mathbf{F}_q^n$). In particular, the *triangle inequality*:

$$d(x,y) \leq d(x,z) + d(z,y)$$

holds for all $x, y, z \in \mathbf{F}_q^n$.

The *weight* of a word x is $wt(x) = d(x,0) =$ number of nonzero entries. For instance, we have $wt(11000111) = 5$.

Error-Correcting Capacity

The Hamming distance measures a code's error detecting and error correcting capacity:

Proposition 1 *If a code* $C \subset \mathbf{F}_q^n$ *satisfies*

$$d(u, v) \geq s + 1$$

for all distinct $u, v \in C$*, then all error vectors of weight* s *less can be detected. If* $d(u, v) \geq 2t+1$*, all error vectors of weight* t *or less will be corrected by the "nearest-neighbor" decoding function:*

$$D(x) = E^{-1}(c \in C : d(x, c) \text{ is minimal}).$$

We write

$$B(u, s) = \{y \in \mathbf{F}_q^n : d(u, y) \leq s\}.$$

(the closed ball with center u, radius s for the Hamming distance)

The Proof

Proof: If $d(u,v) \geq s+1$ for all $u \neq v$ in C, then changing s entries in a codeword never produces another codeword.

Assume $d(u,v) \geq 2t+1$ for all $u,v \in C$. The closed Hamming balls of radius t about the codewords u,v must be *disjoint*. For if not, and $y \in B(u,t) \cap B(v,t)$, we would have

$$d(u,v) \leq d(u,y) + d(y,v) \leq 2t$$

by the triangle inequality. But this contradicts our assumption.

Shows: If u is sent, weight of error e is $\leq t$, then $u+e$ is still closer to u than any other codeword – nearest neighbor decoding will correct error!

□

Coding Practice

- Nearest neighbor decoding *can fail* if error vector has weight $> t$.

- In most reasonable situations, errors of smaller weight are more likely than errors of larger weight, though.

- Based on type of information to be transmitted, properties of channel, energy "budget" available etc. an engineer would select a code where the *probability* of that happening was acceptably small.

Important Parameters of Codes

- $R = k/n$, the *information rate*

- $d = \min_{x \neq y \in C} d(x, y)$, the *minimum distance* (or d/n the *relative minimum distance*).

There are many known theoretical bounds on these parameters. One such:

Proposition 2 *(Singleton Bound) For each fixed d, the parameters n, k of codes of minimum distance d satisfy*

$$d \leq n - k + 1 \ .$$

Good Codes

"Good" codes are ones for which $R = k/n$ is not too small (so the code is not extremely redundant), but for which d is relatively large.

There is a famous theorem of Claude Shannon (father of this theory) that says very roughly "good codes exist" but gives no explicit way to construct them.

One of the main research directions in coding theory has been to find constructions of "good codes".

Tools: Algebra of polynomials, finite fields, geometry, etc.

Codes with more algebraic structure

All examples we study will be *linear block codes*:

- The alphabet $\mathbf{F}_q$ is a finite *field*,
- The set of codewords C is a k-dimensional *vector subspace* of $\mathbf{F}_q^n$.
- For E, can take any linear mapping with image C – matrix is called a *generator matrix* for C.
- Writing all k- and n-tuples as *row* vectors, can take G to be an $k \times n$ matrix whose rows span C, and $E(x) = xG$.

Note: to describe a linear code we need only k elements in a basis, rather than all q^k codewords!

An Example

In $\mathbf{F}_2^7$ consider the code C given by the generator matrix

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Parameters $n = 7, k = 4, d = 3$.

Since $3 = 2 \cdot 1 + 1$, this code can correct any single bit error in word of length 7.

How we can tell: For linear codes, if $x, y \in C$, then $x - y \in C$ too. Hence

$$\min_{x \neq y \in C} d(x, y) = \min_{x \neq y \in C} d(x-y, 0) = \min_{z \neq 0 \in C} d(z, 0)$$

In other words: For linear codes, the minimum distance is the same as the minimum weight of a nonzero codeword.

Example, continued

There are $2^4 = 16$ different codewords for this code (all linear combinations of the rows of G):

- one: $(0,0,0,0,0,0,0)$ has weight zero,
- seven have weight 3:

  $(1,0,0,0,1,1,0)$ $(1,0,0,1,0,0,1)$
  $(1,1,1,0,0,0,0)$ $(0,1,0,1,0,1,0)$
  $(0,1,0,0,1,0,1)$ $(0,0,1,0,0,1,1)$
  $(0,0,1,1,1,0,0)$
- seven have weight 4
- one has weight 7: $(1,1,1,1,1,1,1)$

(Words of weight 4 are "complements" of the words of weight 3.)

More On This Example

The code given by G above has another interesting property: every word in $\mathbf{F}_2^7$ is either a codeword, or Hamming distance 1 from a unique codeword. (Reason: $16 \cdot (1+7) = 128$.)

A rudimentary decoder would use a "check matrix" H for C such as

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

for which the codewords $x \in C$ (written as columns) are the solutions of $xH = 0$.

Decoding

Assuming error has weight ≤ 1,

- Given received y compute $s = yH$;
- If $s = 0$, then $y \in C$;
- If not, then repeat $x' := y$ with one bit toggled until $x'H = 0$. Then x' is a codeword within Hamming distance 1 of y, and that was the transmitted word, *assuming no more than 1 bit in the codeword was corrupted*.

As before, this method *may not correctly decode* a received word y if weight of error is too large.

Hamming Codes

- The $[7, 4, 3]$ binary code in this example is an example of a *Hamming code*.

- The general binary Hamming code C_r has parameters $[n, k, d] = [2^r - 1, 2^r - r - 1, 3]$.

- Parity check matrix H_r is the $2^r - 1 \times r$ matrix whose rows are the $2^r - 1$ nonzero vectors of length r over $\mathbf{F}_2$.

- $x \in C_r \Leftrightarrow xH_r = 0$, or equivalently the entries of x are coefficients in a linear dependence between rows of H_r

- $\Rightarrow d = 3$.

Some Other Famous Codes

- The *Golay* code – a remarkable $[23, 12, 7]$ binary code. The Hamming balls of radius 3 about 2^{12} codewords completely fill out $\mathbf{F}_2^{23}$:

$$2^{12} \cdot (1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}) = 2^{23}$$

- BCH codes – give a way to construct binary codes of length $2^s - 1$ with any *designed minimum distance* δ; actual minimum distance satisfies $d \geq \delta$. Construction uses much of the same algebra as the Reed-Solomon codes we will study in §3.

- Low-Density Parity-Check codes – intensively studied recently.

§2. Cyclic and Multi-cyclic codes, Finite Fields

- A linear code $C \subset \mathbf{F}_q^n$ is said to be *cyclic* if it is invariant under the cyclic shift mapping

  $$\sigma(c_0, c_1, c_2, \ldots, c_{n-1}) = (c_{n-1}, c_0, c_1, \ldots, c_{n-2})$$

- Often assume n is relatively prime to q here to avoid some algebraically "interesting" behavior; not necessary, though.

- This property has another nice algebraic interpretation and leads to interesting encoding and decoding methods.

Algebraic translation

Define

$$\begin{array}{rcl} \varphi : \mathbf{F}_q^n & \to & \mathbf{F}_q[t]/\langle t^n - 1\rangle \\ (c_0, \ldots, c_{n-1}) & \mapsto & c_0 + c_1 t + \cdots + c_{n-1} t^{n-1} \end{array}$$

(right hand side mod $t^n - 1$.)

Proposition 3 *If C is a cyclic code, then $\varphi(C)$ is an ideal in the ring $\mathbf{F}_q[t]/\langle t^n - 1\rangle$, and conversely.*

Proof: $\varphi(C)$ is a vector subspace, since φ is the standard linear isomorphism between $\mathbf{F}_q^n$ and the quotient ring.

Proof, continued

So, we only need show $\varphi(C)$ closed under products by elements of $\mathbf{F}_q[t]/\langle t^n - 1\rangle$. First note that if $c \in C$ is a vector as above,

$$\begin{aligned} t \cdot \varphi(c) &= c_0 t + c_1 t^2 + \cdots + c_{n-1} t^n \\ &\equiv c_{n-1} + c_0 t + \cdots + c_{n-2} t^{n-1} \bmod t^n - 1 \\ &= \varphi(\sigma(c)) \end{aligned}$$

This is in $\varphi(C)$ because C is invariant under σ. But now by distributivity, $\varphi(C)$ is closed under arbitrary products. //

Note: Can also view $R = \mathbf{F}_q[t]/\langle t^n - 1\rangle$ as the group algebra of the cyclic group of order n: $\mathbf{F}_q[\mathbf{Z}_n]$.

More on the structure of cyclic codes

- Easy to see that the ring $R = \mathbf{F}_q[t]/\langle t^n - 1 \rangle$ is a principal ideal ring, since $\mathbf{F}_q[t]$ is.

- Every ideal in R is generated by the coset of some $g(t)$ of degree $n - 1$ or less.

- Also, can take $g(t)|t^n - 1$ in $\mathbf{F}_q[t]$.

- Call $g(t)$ a *generator polynomial* for the cyclic code.

- Can always normalize $g(t)$ to make it *monic*.

Some Examples

- In $\mathbf{F}_2[t]$, the polynomial $t^{15}-1 = t^{15}+1$ has irreducible factors: $(t+1), (t^2+t+1), (t^4+t+1), (t^4+t^3+1), (t^4+t^3+t^2+t+1)$

- $\Rightarrow 2^5 = 32$ different divisors, each of which can be taken as the generator polynomial for a binary cyclic code with $n = 15$.

- For example, code with $g(t) =$

$$(t+1)(t^4+t+1) = t^5+t^4+t^2+1$$

has $n = 15, k = 10, d = 4$.

- One generator matrix is formed from cyclic shifts of:

$$(0,0,0,0,0,0,0,0,0,1,1,0,1,0,1)$$

A First Generalization

The algebraic interpretation of cyclic codes given above suggests several generalizations. The first is relatively straightforward.

- Replace $\mathbf{F}_q[t]/\langle t^n - 1\rangle$ by a finite-dimensional quotient of a polynomial ring in more than one variable, of the form

$$R = \mathbf{F}_q[t_1, \ldots, t_m]/\langle t_1^{n_1} - 1, \ldots, t_m^{n_m} - 1\rangle$$

- Consider ideals $I \subset R$.

The resulting codes are called *abelian codes*, and *m-dimensional cyclic codes* in the literature.

2-D Cyclic Codes

- For example, with $m = 2$, the codewords could be viewed as either as polynomials in two variables (linear combinations of monomials $t_1^{e1} t_2^{e2}$, with $0 \leq e_i \leq n_i$), or as rectangular arrays.

- Closure under multiplication by t_1 means code is invariant under simultaneous row-wise cyclic shifts.

- Similarly, closure under multiplication by t_2 means the code is invariant under simultaneous column-wise cyclic shifts.

- Could also study these codes as ideals in the group algebra $\mathbf{F}_q[\mathbf{Z}_{n_1} \times \mathbf{Z}_{n_2}]$.

An Example

For instance, we can ask: What is the smallest 2D-cyclic code C of size $n_1 \times n_2 = 3 \times 3$ over $\mathbf{F}_2$ containing the codeword:

$$X = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} ?$$

X corresponds to the polynomial:

$$p_X = 1 + t_1^2 + t_1 t_2 + t_2^2 + t_1^2 t_2^2$$

(or, better, the coset of this polynomial in $R = \mathbf{F}_2[t_1, t_2]/\langle t_1^3 + 1, t_2^3 + 1 \rangle$).

Example, continued

We claim that the ideal generated by p_X in R is equal to the ideal generated by $p_1 = t_1^2 + t_1 + 1$ and $p_2 = t_2^2 + t_2 + 1$. This is equivalent to the claim that

$$I = \langle p_X, t_1^3 + 1, t_2^3 + 1 \rangle$$

equals

$$J = \langle p_1, p_2 \rangle$$

in $\mathbf{F}_2[t_1, t_2]$. We see:

$$\begin{aligned} p_X &= p_1 p_2 + t_2 p_1 + t_1 p_2 \\ t_i^3 + 1 &= (t_i + 1) p_i \end{aligned}$$

$\Rightarrow I \subseteq J$. Similarly, can see $p_1, p_2 \in I$ (Exercise!), so $I = J$.

(Gröbner bases give a nice way to test equality of ideals. Can also deduce $I = J$ here from properties of the Galois field $\mathbf{F}_4$ – next lecture!)

Example, concluded

From the description of the code as the ideal J above, can see:

$$n = 9, k = 5, d \leq 3$$

(Estimate on d comes from the weight of the codewords corresponding to p_1 and p_2.)

We have a basis for the code consisting of the words corresponding to:

$$p_1, t_2 p_1, t_2^2 p_1, t_1 p_2, t_1^2 p_2$$

Looking Ahead

The algebra of ideals $I \subset R$ is more interesting than in the case $m = 1$.

Not every ideal is principal, for instance, as we see already in the example done before!

For all $m \geq 1$, the theory of Groebner bases can be applied, for instance, to construct encoding and decoding algorithms. We will study these for cyclic and abelian codes in §4 and §6.

Finite Fields

To describe our next examples, we need to introduce explicit fields larger than $\mathbf{F}_2 = \{0, 1\}$ or the "prime fields" $\mathbf{F}_p$ (integers mod p) to be used as code alphabets.

To see idea, we might want set of *strings* of $0, 1$'s of a fixed length r to be the alphabet. $r = 4$ would give 2^4 distinct symbols:

$0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,$
$1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111.$

In order to work with the set-up of linear codes, though, this set must be given the structure of *a field*.

A Field With 16 Elements

We can interpret a string $\beta_3\beta_2\beta_1\beta_0$ ($\beta_i \in \mathbf{F}_2$) as a polynomial in a new variable α:

$$\beta_3\alpha^3 + \beta_2\alpha^2 + \beta_1\alpha + \beta_0$$

The set of all 16 such expressions will be denoted by $\mathbf{F}_{16}$.

Addition operation: usual polynomial addition (same as vector addition in $\mathbf{F}_2^4$).

Multiplication operation: we know how to multiply polynomials:

$$(\alpha^3 + 1)(\alpha^2 + 1) = \alpha^5 + \alpha^3 + \alpha^2 + 1.$$

But of course the degree of the product is *too large* here. To reduce to the proper range of degrees, we *divide* by some polynomial of degree 4 in α and take the *remainder* of the product.

Of course, we are working in the quotient ring $\mathbf{F}_2[\alpha]/\langle h(\alpha)\rangle$.

A Field with 16 Elements, continued

This much works for any divisor polynomial $h(\alpha)$ of degree 4. But, do we always get a field this way?

The answer is *no*, since $h(\alpha)$ must be irreducible in order for the $\langle h(\alpha) \rangle$ to be a maximal ideal and the quotient ring to be a field.

For instance, it can be checked that both $h(\alpha) = \alpha^4+\alpha+1$ and $h(\alpha) = \alpha^4+\alpha^3+1$ are irreducible in $\mathbf{F}_2[\alpha]$.

A Field with 16 Elements, concluded

Another more elementary way to see that we have a field is to check $1, \alpha, \alpha^2, \ldots, \alpha^{14}$ are all distinct, and $\alpha^{15} = 1$. Hence

- The powers of α give all the nonzero elements of $\mathbf{F}_{16}$, and

- Each element α^k has a multiplicative inverse α^{15-k}. (That is, the multiplicative group $\mathbf{F}_{16}^*$ is a cyclic group of order 15.)

α is called a *primitive element* for $\mathbf{F}_{16}$.

Finite (Galois) Fields

Theorem 1 *Let p be prime in $\mathbf{Z}$.*

1. *The set of integers mod p is a field denoted $\mathbf{F}_p$ (a prime field).*

2. *For all p and all $r \geq 1$, there are irreducible $h(\alpha)$ of degree r in $\mathbf{F}_p[\alpha]$.*

3. *Let $h(\alpha)$ be irreducible of degree r. Then $\mathbf{F}_p[\alpha]/\langle h(\alpha)\rangle$ is an extension field of the prime field $\mathbf{F}_p$ with p^r elements.*

4. *Different choices of irreducible h of the same degree yield isomorphic fields, called $\mathbf{F}_{p^r}$.*

5. *Every field $\mathbf{F}_{p^r}$ has a primitive element.*

§3. Applications: Reed-Solomon and BCH Codes

Suppose we want to construct codes attaining the Singleton bound. (These are called MDS, or "maximum distance separable," codes).

Restricting to codes of length $n \leq q$, here is a way if the code alphabet is $\mathbf{F}_q$. Fix an integer $k \leq q$. Polynomials of degree $< k$ have at most $k-1$ roots in $\mathbf{F}_q$, and some have precisely that many roots.

Write L_k for the set of all polynomials in $\mathbf{F}_q[x]$ of degree $< k$. We will always assume $k < q$ here.

Application, continued

L_k is a vector space over $\mathbf{F}_q$ of dimension k. For each polynomial $f \in L_k$, we construct a word in $\mathbf{F}_q^q$ by *evaluating* f at the elements of $\mathbf{F}_q$, to get a q-tuple (letting α be a primitive element),

$$(f(0), f(1), f(\alpha), \ldots, f(\alpha^{q-2}))$$

(recall $\alpha^{q-1} = 1$). When we do this, we get a word with

- at most $k - 1$ zero entries, hence

- at least $q - (k - 1) = q - k + 1 = n - k + 1$ nonzero entries (and some have exactly $k - 1$ zero entries).

Application, continued

The set of *all* such words is a linear code since:

- L_k is a vector space, and
- the evaluation mapping is linear.

Hence the resulting code will have

- dimension k and
- *minimum distance* $d = n - k + 1$ when $k < q$.

Application, concluded

- Using all q elements of $\mathbf{F}_q$, we get *extended Reed-Solomon codes*.

- Standard Reed-Solomon codes come from evaluating only at the nonzero elements of the field (omitting the $f(0)$ entry to get a word in $\mathbf{F}_q^{q-1}$).

- Can also take the polynomials vanishing at some fixed subset of nonzero elements, evaluate them, and delete the zeroes to form evaluation vectors of length $n < q-1$ – yields the *shortened* Reed-Solomon codes. (The actual codes used in the CD audio system are examples of these.)

In Summary ...

Theorem 2 *Pick a primitive element α for $\mathbf{F}_q$, and write the nonzero elements of $\mathbf{F}_q$ as*

$$1, \alpha, \ldots, \alpha^{q-2}$$

Let $k < q$ and $L_k = \{f \in \mathbf{F}_q[x] : \deg f < k\}$. Write

$$\begin{aligned} ev : L_k &\rightarrow \mathbf{F}_q^{q-1} \\ f &\mapsto (f(1), f(\alpha), \ldots, f(\alpha^{q-2})). \end{aligned}$$

Then $Im(ev)$ is a linear code with $n = q - 1$, dimension k, and minimum distance $d = n - k + 1 = q - k$, called a Reed-Solomon code, $RS(k, q)$. All Reed-Solomon codes are MDS codes with $d = n - k + 1$.

An RS Example

For example, using the standard monomial basis

$$\{1, x, x^2, x^3, \ldots, x^{k-1}\}$$

for L_k, the Reed-Solomon code $RS(3, 16)$ (parameters: $n = 15, k = 3, d = 13$ over $\mathbf{F}_{16}$, so $16^3 = 4096$ distinct codewords) has generator matrix

$$G = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ 1 & \alpha & \alpha^2 & \cdots & \alpha^7 & \alpha^8 & \cdots & \alpha^{14} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{14} & \alpha & \cdots & \alpha^{13} \end{pmatrix}.$$

How RS Codes are Used

Reed-Solomon codes are commonly used in situations where errors tend to occur in "bursts" rather than randomly.

This includes communication to and from deep-space exploration craft, the CD digital audio system, and many other applications.

Reed-Solomon and other block codes over $\mathbf{F}_{2^r}$ can correct relatively long bursts of errors on the bit level, even if d is relatively small.

Burst Error Correction

Can "go back" and think of $\beta_{r-1}\alpha^{r-1} + \cdots + \beta_1\alpha + \beta_0$ in $\mathbf{F}_{2^r}$ as the vector $(\beta_{r-1}, \ldots, \beta_1, \beta_0) \in \mathbf{F}_2^r$.

Then a Reed-Solomon codeword is represented by a string of $(2^r - 1)r$ bits.

A burst of $s \cdot r$ consecutive bit errors, for instance, will change at most $s + 1$ of the entries of the codeword, as elements of $\mathbf{F}_{2^r}$.

Hence, $RS(3, 16)$ above can correct burst errors of bit length up to $20 = 5 \cdot 4$. Random errors of weight ≤ 6 are correctable since $d = 13$.

More on Structure of RS Codes

Reed-Solomon codes also have additional algebraic structure as in §2.

This greatly facilitates the encoding and decoding operations. To see the idea, consider the generator matrix G for the Reed-Solomon code $RS(k,q)$ constructed by evaluating the monomials $\{1, x, x^2, \ldots, x^{k-1}\}$ at the $\alpha^\ell \in \mathbf{F}_q^*$. The ith row of G has the form

$$((1)^{i-1}, (\alpha)^{i-1}, (\alpha^2)^{i-1}, \ldots, (\alpha^{q-2})^{i-1}).$$

Cyclically permuting this row, we obtain

$$((\alpha^{q-2})^{i-1}, (1)^{i-1}, (\alpha)^{i-1}, \ldots, (\alpha^{q-3})^{i-1}),$$

which is equal to

$$\alpha^{(i-1)(q-2)} \cdot ((1)^{i-1}, (\alpha)^{i-1}, (\alpha^2)^{i-1}, \ldots, (\alpha^{q-2})^{i-1})$$

because $\alpha^{q-1} = 1$.

RS Codes are Cyclic

Thus, a cyclic permutation of the ith row yields a scalar multiple of the same row—it is also one of the Reed-Solomon codewords!

The cyclic permutation is a linear mapping S on $\mathbf{F}_q^{q-1}$, and we have just seen that there is a basis of $RS(k,q)$ consisting of *eigenvectors* for S.

It follows that the Reed-Solomon code $RS(k,q)$ is *invariant* under S, since all the codewords are linear combinations of the rows of G. These observations give the proof of the following fact.

Theorem 3 *For all q and all $k < q$, the Reed-Solomon code $RS(k,q)$ is cyclic.*

To the Generator Polynomial

Consider the mapping

$$\begin{array}{rcl} \psi : RS(k,q) & \rightarrow & \mathbf{F}_q[t]/\langle t^{q-1}-1\rangle \\ (c_0, c_1, \ldots, c_{q-2}) & \mapsto & c_0 + c_1 t + \cdots + c_{q-2} t^{q-2} \end{array}$$

as in §2.

Caution: that there are *two* rings of polynomials "in play" now. The first is the ring $\mathbf{F}_q[x]$ containing the polynomials that are evaluated to form the Reed-Solomon codewords. The second is the ring $\mathbf{F}_q[t]$ (or $\mathbf{F}_q[t]/\langle t^{q-1}-1\rangle$) containing the polynomial forms $\psi(c)$ of the Reed-Solomon codewords. Care should be taken not to confuse these(!)

To the Generator Polynomial, continued

Every element of $\psi(C)$ has the form

$$\psi(c) = c_0 + c_1 t + \cdots + c_{q-2} t^{q-2}$$

where we obtain the coefficients

$$c_i = \sum_{j=0}^{k-1} a_j (\alpha^i)^j$$

by evaluating some fixed $f(x) = \sum_{j=0}^{k-1} a_j x^j$ at $x = \alpha^i$ for $i = 0, \ldots, q-2$.

Substituting these expressions for c_i, interchanging the order of summation:

$$\begin{aligned} \psi(c) &= \sum_{i=0}^{q-2} \left(\sum_{j=0}^{k-1} a_j (\alpha^i)^j \right) t^i \\ &= \sum_{j=0}^{k-1} a_j \left(\sum_{i=0}^{q-2} (\alpha^j t)^i \right) \end{aligned}$$

The Generator Polynomial, Finally

In $\mathbf{F}_q$, roots of $0 = 1 + z + z^2 + \cdots + z^{q-2}$ are all $z \neq 0, 1$. Hence the inner sum in previous is equal to zero provided that $a^j t \neq 0, 1$.

The whole sum $= 0$ if $\alpha^j t \neq 0, 1$ for all $j = 0, \ldots, k-1$, $\Leftrightarrow t \in \{\alpha, \alpha^2, \ldots, \alpha^{q-k-1}\}$. Consequently, every $\psi(c)$ is divisible by

$$g(t) = (t - \alpha)(t - \alpha^2) \cdots (t - \alpha^{q-k-1}).$$

In fact, by comparing dimensions, we have:

Theorem 4 *The polynomial*

$$g(t) = (t - \alpha)(t - \alpha^2) \cdots (t - \alpha^{q-k-1})$$

is the generator polynomial for $RS(k, q)$.

An Alternate Form

Since the minimum distance of a Reed-Solomon code satisfies $d = q - k$, the generator polynomial can also be written as

$$g(t) = (t - \alpha)(t - \alpha^2) \cdots (t - \alpha^{d-1}).$$

For example, the Reed-Solomon code $RS(3, 16)$ from above has

$$g(t) = (t - \alpha)(t - \alpha^2)(t - \alpha^3) \cdots (t - \alpha^{12})$$

since $d = 15 - 3 + 1 = 13$.

The BCH Codes

The second class of codes we will consider were discovered at almost the same time as the Reed-Solomon codes, independently by Bose and also by Chaudhuri and Hocquenghem.

They are known as BCH codes, and they are of interest because they give examples of codes of *general* block length over $\mathbf{F}_q$ (not just $n \leq q$ as for the Reed-Solomon examples) that can be designed to satisfy $d \geq \delta$, for a given δ.

They are also cyclic, so they can be described most simply by giving a method for constructing their generator polynomials. As we will see, they are also very closely connected to Reed-Solomon codes.

Motivation for the Construction

BCH construction uses a smidgen of Galois theory to construct codes over $\mathbf{F}_q$ with good d. If $\gcd(n, q) = 1$, then the nth roots of unity in $\overline{\mathbf{F}_q}$ are distinct.

Theorem 5 *(The BCH Bound) Let β be a primitive nth root of unity in an extension field $\mathbf{F}_{q^r}/\mathbf{F}_q$, and assume that $g(t) \in \mathbf{F}_q[t]$ is the generator polynomial of a cyclic code C of length n over $\mathbf{F}_q$. Consider the set of powers β^j such that $g(\beta^j) = 0$. Assume this contains a string of consecutive powers:*

$$\beta^{i_0}, \beta^{i_0+1}, \ldots, \beta^{i_0+\delta-2}$$

of length $\delta - 1$. Then the minimum distance of C satisfies $d \geq \delta$.

In RS case, $n = q - 1$, so a primitive nth root of unity is just a primitive element of $\mathbf{F}_q$, and there is no field extension involved.

The Construction

BCH codes in general are constructed by "cooking up" generator polynomials that are guaranteed to have consecutive strings of powers of a primitive nth root of unity among their roots.

From Galois theory for finite fields, we know that $Gal(\mathbf{F}_{q^r}/\mathbf{F}_q)$ is cyclic of order r, generated by the Frobenius automorphism $F(x) = x^q$.

If $g(t) \in \mathbf{F}_q[t]$ and β is a root of g, then since F fixes the coefficients in g, $F(\beta) = \beta^q$ is also a root of g. This observation can be used to simplify the description of the desired $g(t)$.

BCH Codes over $\mathbf{F}_2$

The binary BCH codes (i.e. BCH codes over $\mathbf{F}_2$) are the most commonly encountered ones, and make the idea clear.

Take n odd. To get a cyclic code with minimum distance $d \geq \delta = 2s + 1$, take β any primitive nth root of 1 in an extension $\mathbf{F}_{2^r}/\mathbf{F}_2$ (with r as small as possible, of course).

Let $m_{\beta^k}(t)$ be the *minimal polynomial* of β^k in $\mathbf{F}_2[t]$. Then full set of roots of $m_\beta(t)$ is

$$\beta^k, F(\beta) = \beta^{2k}, F(F(\beta)) = \beta^{4k}, \ldots, \beta^{2^{r-1}k}$$

(all exponents mod n since $\beta^n = 1$). So if

$$g(t) = \text{lcm}(m_\beta, m_{\beta^3}, m_{\beta^5}, \ldots, m_{\beta^{2s-1}})$$

then all the β^j for $1 \leq j \leq 2s$ will be among the roots. Hence the BCH bound will imply $d \geq \delta = 2s + 1$. There is a similar statement for all q.

Conclusion

It is not too difficult to find examples where the actual minimum distance of a BCH code is strictly larger than the "designed distance" δ.

Example: take $n = 31, \delta = 9$ over $\mathbf{F}_2$.

Other more refined lower bounds on d for cyclic codes are also known, but this is an area where research continues and our understanding is not complete yet.

§4. Encoding and Decoding via Gröbner Bases

The extra symmetry of a cyclic code means that less information is required to specify a systematic encoder than for a general linear code of the same dimension.

In fact all we need to know is the $n-k$ coefficients in a monic generator polynomial $g(t)$. (Note: if $\dim C = k$, then the degree of $g(t)$ will be $n-k$.)

Moreover, the encoding process can be described very succinctly using a standard algebraic algorithm – polynomial division!

Systematic Encoding via Division

For this description of systematic encoding,

- Coefficients of $t^{n-k}, \ldots, t^{n-1}$ are the he *information positions*, and

- Coefficients of $1, \ldots, t^{n-k-1}$ are the *parity checks*.

Input : $g(t)$, generator poly
information symbols $c_1, ..., c_k$
Output : y, a codeword

$$\begin{aligned} p &:= c_1 t^{n-k} + ... + c_k t^{n-1}; \\ y &:= p - Rem(p, g, t); \end{aligned}$$

Since g has degree $n-k$, the remainder will contain only $1, t, \ldots, t^{n-k-1}$. The other terms will be the same as in p.

Generalization to m-dimensional Cyclic Codes

The method here generalizes immediately to m-dimensional cyclic codes (viewed as ideals I in $\mathbf{F}_q[t_1, \ldots, t_m]/\langle t_1^{n_1} - 1, \ldots, t_m^{n_m - 1} \rangle$).

Generator polynomial $g(t)$ is replaced by any Gröbner basis $\mathcal{G}$ for I.

Univariate polynomial division replaced by multivariate division w.r.t. $\mathcal{G}$.

RS Decoding

Several different but related extremely efficient decoding algorithms for Reed-Solomon and BCH codes have been developed – one major reason for the Reed-Solomon codes' popularity.

- Berlekamp-Massey algorithm, very commonly used in practice

- Another algorithm paralleling the Euclidean algorithm for the GCD of two polynomials is also known

We'll study the first approach here.

The Setup

For simplicity, assume Reed-Solomon code C satisfies $d = 2s + 1$. Then by Proposition 1, any s or fewer errors in a received word should be correctable.

Let $c = \sum_{j=0}^{q-2} c_j t^j$ be a codeword of C.

In $\mathbf{F}_q[t]$, c is divisible by

$$g = (t - \alpha)(t - \alpha^2) \cdots (t - \alpha^{d-1}).$$

Suppose that c is transmitted, but some errors are introduced, so that the received word is $r = c + e$ for some $e = \sum_{i \in L} e_i t^i$. $L =$ *error locations*, and we assume $|L| \leq s$. The coefficients e_i are the *error values*.

The Decoding Problem

Given the received word r, determine the set of error locations L and the error values e_i for the error polynomial e with s or fewer nonzero terms (if such a polynomial exists).

First, we compute $s_j = r(\alpha^j)$ for all $j = 1, \ldots, d-1$, called the *syndromes* of the received word.

- If these are all zero, then r is divisible by g, and assuming $wt(e) \leq s$, r must be the codeword we sent.

- If some syndromes are nonzero,

$$s_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j) = e(\alpha^j),$$

since c is a multiple of g.

- Hence we can try to use the information included in the syndromes to determine $e(t)$.

The Syndrome Polynomial

The syndromes may be used as the coefficients in a polynomial

$$S(u) = \sum_{j=1}^{d-1} s_j u^{j-1},$$

called the *syndrome polynomial* for the received word r. Its degree is $d-2$ or less.

By extending the definition of $s_j = e(\alpha^j)$ to *all* exponents j we can also consider the formal power series

$$\widehat{S}(u) = \sum_{j=1}^{\infty} s_j u^{j-1}.$$

Another form for $\widehat{S}$

Suppose we knew $e(t)$ for a received word with $wt(e) \leq s$. Then, $s_j = \sum_{i \in L} e_i (\alpha^j)^i = \sum_{i \in L} e_i (\alpha^i)^j$. Exchanging the order of summation, then summing formal geometric series,

$$\begin{aligned}
\widehat{S}(u) &= \sum_{j=1}^{\infty} s_j u^{j-1} \\
&= \sum_{i \in L} e_i \left(\sum_{j=1}^{\infty} (\alpha^i)^j u^{j-1} \right) \\
&= \sum_{i \in L} \frac{e_i \alpha^i}{(1 - \alpha^i u)} \\
&= \frac{w(u)}{\ell(u)},
\end{aligned}$$

where

$$\ell = \prod_{i \in L} (1 - \alpha^i u), \quad w = \sum_{i \in L} e_i \alpha^i \prod_{\substack{j \neq i \\ j \in L}} (1 - \alpha^j u).$$

The Error Locator

The roots of ℓ are precisely the α^{-i} for $i \in L$.

Since the error locations can be determined easily from these roots, ℓ is called the *error locator polynomial*.

Turning to the numerator w, we see that

$$\deg w \leq \deg \ell - 1.$$

In addition, if $i \in L$,

$$w(\alpha^{-i}) = e_i \alpha^i \prod_{j \neq i, j \in L} (1 - \alpha^j \alpha^{-i}) \neq 0.$$

Hence w has no roots in common with ℓ. From this we deduce the important observation that the polynomials w and ℓ must be *relatively prime*.

The Key Equation

Similarly, if we consider the "tail"

$$\begin{aligned}\widehat{S}(u) - S(u) &= \sum_{j=d}^{\infty} \left(\sum_{i \in L} e_i (\alpha^i)^j \right) u^{j-1} \\ &= u^{d-1} \cdot \frac{g(u)}{\ell(u)},\end{aligned}$$

where

$$g = \sum_{i \in L} e_i \alpha^{id} \prod_{\substack{j \neq i \\ j \in L}} (1 - \alpha^j u).$$

The degree of g is also at most $\deg \ell - 1$.

Combining these, and writing $d - 1 = 2s$ we obtain the relation

$$(KE) \qquad w = \ell S + u^{2s} g.$$

This is the *key equation* for decoding.

A "Key" bservation

The derivation of (KE) assumed the error polynomial e was known. But now consider the actual situation. Given the received word r, S is computed. We consider (KE) as a relation between the known polynomials S, u^{2s}, and *unknowns* Ω, Λ, Γ:

(KE') $$\Omega = \Lambda S + u^{2s}\Gamma.$$

Suppose a solution $(\Omega, \Lambda, \Gamma)$ of (KE') is found, which satisfies the *degree conditions*: $\deg \Lambda \leq s$, and $\deg \Omega < \deg \Lambda$,

and in which Ω, Λ are relatively prime. We claim that in such a solution Λ must be a factor of $u^{q-1} - 1$, and its roots give the inverses of the error locations.

A Uniqueness Statement

The last claim is a consequence of:

Proposition 4 *Suppose* $wt(e) \leq s$, *and let* S *be the corresponding syndrome polynomial. Up to a constant multiple, there exists a unique solution* $(\Omega, \Lambda, \Gamma)$ *of (KE') that satisfies the degree conditions above, and for which* Ω *and* Λ *are relatively prime.*

Proof: The actual error locator ℓ and the corresponding w, g give one solution. Let $(\Omega, \Lambda, \Gamma)$ be any other. Start with

$$\begin{aligned} w &= \ell S + u^{2s} g \\ \Omega &= \Lambda S + u^{2s} \Gamma, \end{aligned}$$

multiply the second by ℓ, the first by Λ and subtract. We obtain

$$w\Lambda = \Omega\ell + u^{2s}(g\Lambda - \ell\Gamma).$$

Proof, continued

By the degree conditions, $w\Lambda$ and $\Omega\ell$ are actually polynomials of degree at most $2s-1$, so it follows that

$$w\Lambda = \Omega\ell$$

(and $g\Lambda = \ell\Gamma$). Since both pairs (w, ℓ) and (Ω, Λ) are relatively prime, they can differ only by a constant multiple.//

Any solution of (KE') for which the degree conditions are satisfied can be used to decode:

- Solve $\Lambda(u) = 0$ in $\mathbf{F}_q \backslash \{0\}$, and get the error locations.

- Then find the error values as follows.

The Forney Formula

Let (w, ℓ, g) be the solution of (KE') in which the actual error locator ℓ (with constant term 1) appears. If $i \in L$, then

$$w(\alpha^{-i}) = \alpha^i e_i \chi_i(\alpha^{-i})$$

where $\chi_i = \prod_{j \neq i}(1 - \alpha^j u)$. (This is called the *Forney formula*.) Hence we can solve for e_i, once we know the error locations.

The preceding discussion shows that solving the decoding problem can be accomplished by solving the key equation (KE').

Recasting the Key Equation

From now on, for our purposes, it will be more convenient to regard (KE) as a congruence:

$$\Omega \equiv \Lambda S \bmod u^{2s}.$$

Congruences of the same form are studied in numerical analysis in the context of Padé approximation.

We will now follow Fitzpatrick and see how computational commutative algebra can be used to solve the key equation.

Given the integer s and $S \in \mathbf{F}_q[u]$, consider set of *all pairs* $(\Omega, \Lambda) \in \mathbf{F}_q[u]^2$ satisfying:

$$K = \{(\Omega, \Lambda) : \Omega \equiv \Lambda S \bmod u^{2s}\}.$$

Module of Solutions

K is an $\mathbf{F}_q[u]$-submodule of $\mathbf{F}_q[u]^2$. In addition, every element of K can be written as a combination (with polynomial coefficients) of the two generators $(u^{2s}, 0)$ and $(S, 1)$ for K, which involve only *known polynomials*.

We use the theory of Gröbner bases in modules over a polynomial ring.

Monomials in $\mathbf{F}_q[u]^2$ are simply monomials in u times one of the standard basis vectors $\mathbf{e}_1$ or $\mathbf{e}_2$.

Let $r \in \mathbf{Z}$, and define an order $>_r$ by the following rules. First, $u^m \mathbf{e}_i >_r u^n \mathbf{e}_i$ if $m > n$ and $i = 1$ or 2. Second, $u^m \mathbf{e}_2 >_r u^n \mathbf{e}_1$ if and only if $m + r \geq n$.

An Example; Some General Observations

For example, with $r = 2$, the monomials in $\mathbf{F}_q[u]^2$ are ordered by $>_2$ as follows:

$$\mathbf{e}_1 <_2 u\mathbf{e}_1 <_2 u^2\mathbf{e}_1 <_2 \mathbf{e}_2 <_2 u^3\mathbf{e}_1 <_2 u\mathbf{e}_2 <_2 \cdots .$$

Gröbner bases for submodules of $\mathbf{F}_q[u]^2$ with respect to the $>_r$ orders have very special forms.

Proposition 5 *Let M be a submodule of the free module $\mathbf{F}_q[u]^2$, and fix $r \in \mathbf{Z}$. Assume $\langle LT_{>_r}(M)\rangle$ is generated by $u^a\mathbf{e}_1 = (u^a, 0)$ and $u^b\mathbf{e}_2 = (0, u^b)$ for some $a, b \geq 0$. Then a subset $\mathcal{G} \subset M$ is a reduced Gröbner basis of M with respect to $>_r$ if and only if $\mathcal{G} = \{g_1 = (g_{11}, g_{12}), g_2 = (g_{21}, g_{22})\}$, satisfying the following two properties:*

a) $LT(g_1) = u^a\mathbf{e}_1$ (in g_{11}), and $LT(g_2) = u^b\mathbf{e}_2$ (in g_{22}) for a, b as above.

b) $\deg(g_{21}) < a$ *and* $\deg(g_{12}) < b$.

A Corollary; Minimal Elements

Proposition 6 *Let $\mathcal{G} = \{(S,1),(u^{2s},0)\}$ be the generators for the module K. Then $\mathcal{G}$ is a Gröbner basis for K with respect the order $>_{\mathsf{deg}(S)}$. Note that $LT_{>_{\mathsf{deg}(S)}}((S,1)) = (0,1) = \mathbf{e}_2$.*

A *minimal element* of a module M with respect to $>$ is a $g \in M$ such that $LT(g)$ is minimal with respect to $>$. For instance, the module element $(S,1)$ is minimal with respect to the order $>_{\mathsf{deg}(S)}$ in $K = \langle (S,1),(u^{2s},0)\rangle$ since

$$\mathbf{e}_2 = LT((S,1)) <_{\mathsf{deg}(S)} LT((u^{2s},0)) = u^{2s}\mathbf{e}_1.$$

Minimal elements of $M \subset \mathbf{F}_q[u]^2$ are *unique*, up to a (nonzero) constant multiple.

Proposition 7 *Fix any $>_r$ order on $\mathbf{F}_q[u]^2$, and let M be a submodule. Every Gröbner basis for M with respect to $>_r$ contains a minimal element of M with respect to $>_r$.*

The Particular Solution e ant

Proposition 8 *Each solution* (Ω, Λ) *of the key equation and the degree conditions* $\deg \Omega \leq \deg \Lambda - 1$ *with relatively prime components is a minimal element of* K *under the* $>_{-1}$ *ordering.*

The Berlekamp-Massey approach builds up the minimal element *iteratively* by solving the congruences $\Omega \equiv \Lambda S \bmod u^m$ for $m = 0, \ldots, 2s$.

The Iterative Step

Proposition 9 *Let K_m be the module of solutions of $\Omega \equiv \Lambda S \bmod u^m$, and let*

$$\mathcal{B} = \{(a_1, b_1), (a_2, b_2)\}$$

be a $>_r$ Gröbner basis of K_m, with the first element minimal. Let $\overline{S} = S \bmod u^{m+1}$ and let c_i be the coefficient of u^m in $b_i\overline{S} - a_i$. Define $\mathcal{B}' = \{(a'_1, b'_1), (a'_2, b'_2)\}$: If $c_1 = 0$, then

$$\begin{aligned} (a'_1, b'_1) &= (a_1, b_1) \\ (a'_2, b'_2) &= (ua_2, ub_2). \end{aligned}$$

If $c_1 \neq 0$, then

$$\begin{aligned} (a'_1, b'_1) &= (ua_1, ub_1) \\ (a'_2, b'_2) &= (a_2, b_2) - \frac{c_2}{c_1}(a_1, b_1). \end{aligned}$$

Then $\mathcal{B}'$ is a $>_r$ Gröbner basis for K_{m+1}.

Apply the Proposition repeatedly with $r = -1$, starting from the $>_{-1}$ Gröbner basis for K_0: $\{(0,1),(1,0)\}$.

An Example

The final values $(a_1(u), b_1(u))$ are $(g(u), \ell(u))$ from (KE), up to a constant multiple. So the roots of $b_1(u)$ are the inverses of the error locations.

We demonstrate this with an example. Use the field $\mathbf{F}_8$ (with $h(\alpha) = \alpha^3 + \alpha + 1$), and the Reed-Solomon code $RS(3, 8)$, which has $d = n - k + 1 = 7 - 3 + 1 = 5$, so $s = 2$. Suppose the codeword

$$c = ev(1) = (1, 1, 1, 1, 1, 1, 1)$$

is sent, but it is corrupted by errors to yield

$$r = (1, \alpha, 1, 1, 1, 1, \alpha^2 + 1),$$

or in polynomial form

$$r = 1 + \alpha t + t^2 + t^3 + t^4 + t^5 + (\alpha^2 + 1)t^6.$$

Example, continued

The first step is to compute the syndromes and the corresponding syndrome polynomial $S(u)$. For instance,

$$\begin{aligned} s_1 &= r(\alpha) \\ &= 1 + \alpha^2 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 + \alpha^6(\alpha^2 + 1) \\ &= 1 + (\alpha + 1) + (\alpha^2 + \alpha) + (\alpha^2 + \alpha + 1) \\ &\quad + (\alpha^2 + \alpha + 1) \\ &= \alpha^2 \end{aligned}$$

Similarly,

$$s_2 = \alpha^4, s_3 = 0, s_4 = \alpha^4,$$

So

$$S(u) = \alpha^2 + \alpha^4 u + \alpha^4 u^3$$

(note the shift in indexing, as in the definition of S above).

Example, continued

Start with

$$K_0 = \{(0,1),(1,0)\}$$

Step 1:

$$\begin{aligned} \overline{S} &= \alpha^2 \\ c_1 &= \alpha^2 \cdot 1 + 0 = \alpha^2 \neq 0 \\ c_2 &= \alpha^2 \cdot 0 + 1 = 1 \\ K_1 &= \{(1,\alpha^5),(0,u)\} \end{aligned}$$

(with the $>_{-1}$-minimal element first).

Step 2:

$$\begin{aligned} \overline{S} &= \alpha^2 + \alpha^4 u \\ c_1 &= \left[(\alpha^2 + \alpha^4 u) \cdot \alpha^5 + 1\right]_u = \alpha^2 \\ c_2 &= \left[(\alpha^2 + \alpha^4 u) \cdot u + 0\right]_u = \alpha^2 \\ K_2 &= \{(1,u+\alpha^5),(u,\alpha^5 u)\} \end{aligned}$$

(with the $>_{-1}$-minimal element first).

Decoding, continued

Step 3:

$$\begin{aligned}
\overline{S} &= \alpha^2 + \alpha^4 u + 0u^2 \\
c_1 &= \left[(\alpha^2 + \alpha^4 u) \cdot (u + \alpha^5) + 1\right]_{u^2} = \alpha^4 \\
c_2 &= \left[(\alpha^2 + \alpha^4 u) \cdot \alpha^5 u + u\right]_{u^2} = \alpha^2 \\
K_3 &= \{(u + \alpha^5, \alpha^3), (u, u^2 + \alpha^5 u)\}
\end{aligned}$$

(with the $>_{-1}$-minimal element first).

Step 4:

$$\begin{aligned}
\overline{S} &= \alpha^2 + \alpha^4 u + a^4 u^3 \\
c_1 &= 1 \\
c_2 &= \alpha^4
\end{aligned}$$

So $(u, u, u^2 + \alpha^5 u) + \alpha^4 (u + \alpha^5, \alpha^3)$

$$= (\alpha^5 u + \alpha^2, u^2 + \alpha^5 u + 1)$$

is the minimal element of K_4.

Error Locations and Values

The minimal element in K_4 has constant coefficient 1 and gives (w, ℓ) exactly (not just up to a scalar multiple).

The roots of the error locator are $u = a, \alpha^6$, so the errors occurred in locations $-1 \equiv 6$ and $-6 \equiv 1 \pmod 7$.

The error values are found with the Forney formula:

$$(\alpha^5\alpha^6+\alpha^2) = \alpha e_1(1+\alpha^6\alpha^6) \Rightarrow e_1 = \alpha^3 = \alpha+1$$

and

$$(\alpha^5\alpha + \alpha^2) = \alpha^6 e_1(1 + \alpha\alpha) \Rightarrow e_6 = \alpha^2$$

(Compare with the codeword and received word!)

§5. Codes From rder Domains

Around 1980 – a big new development in coding theory:

- Start with a smooth curve X defined over $\mathbf{F}_q$. Let G and D be effective divisors on X, sums of $\mathbf{F}_q$-rational points, w/ disjoint supports.

- Take $L(G) = \{f \in \mathbf{F}_q(X) : (f) + G \geq 0\} \cup \{0\}$, an $\mathbf{F}_q$-vector subspace of $\mathbf{F}_q(X)$.

- Define for $D = P_1 + \cdots + P_n$,

$$\begin{aligned} ev : L(G) &\to \mathbf{F}_q^n \\ f &\mapsto (f(P_1), \ldots, f(P_n)) \end{aligned}$$

- Let $C = C_L(D, G) = im(ev) \subset \mathbf{F}_q^n$. (Note: ev is linear so C is a linear code.)

An Example

Called AG Goppa codes.

Let $q = 4$, $\mathbf{F}_4 = \mathbf{F}_2[\alpha]/\langle \alpha^2 + \alpha + 1 \rangle$ (α is primitive).

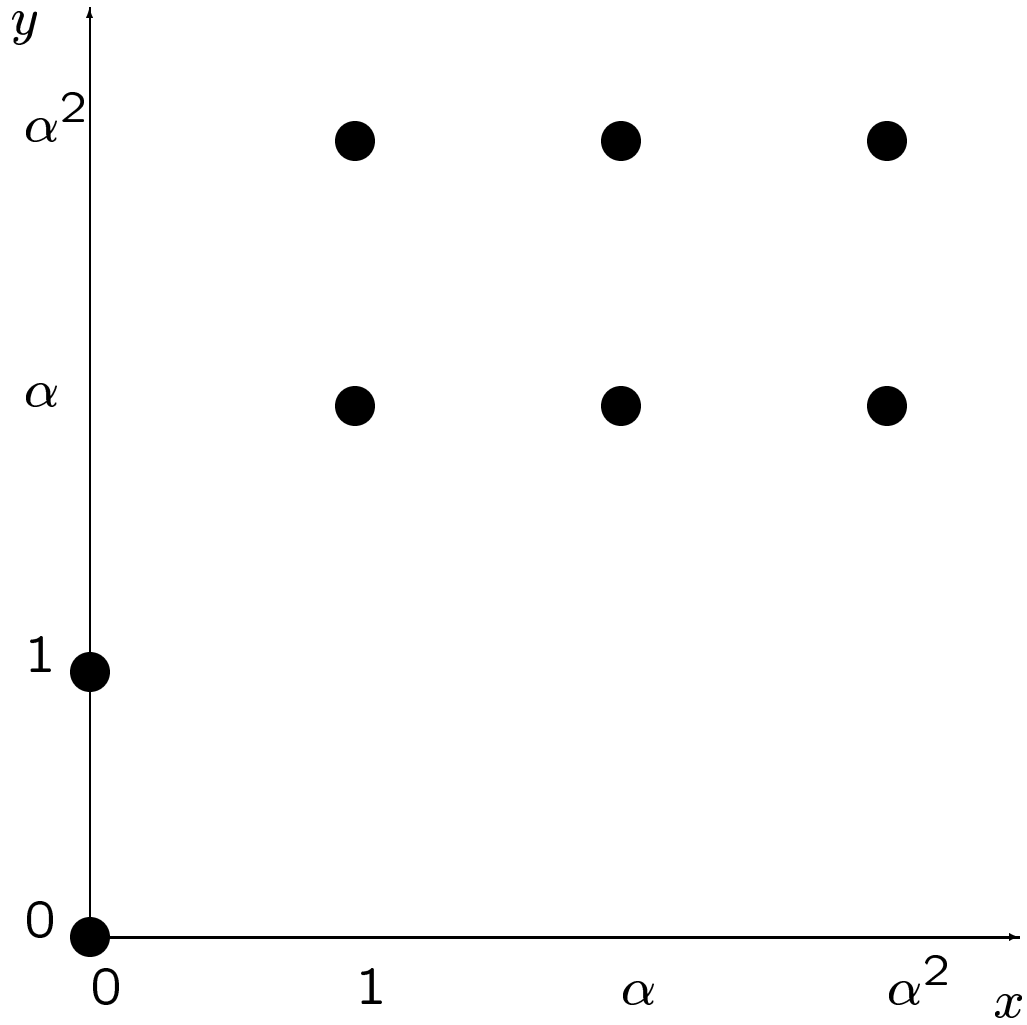
Take X to be the Hermitian curve over $\mathbf{F}_4$: $X = V(x^3 + y^2z + yz^2) \subset \mathbf{P}^2$. X is smooth, genus $g = 1$. There are 9 $\mathbf{F}_4$-rational points on X: $Q = (0 : 1 : 0)$, and 8 affine points.

(Note: This is the maximum possible for a curve of genus 1 over $\mathbf{F}_4$, by the Hasse-Weil bound:

$$|X(\mathbf{F}_q)| \leq 1 + q + 2g\sqrt{q}.)$$

Want X to have "many" $\mathbf{F}_q$-rational points in this construction.)

$X(\mathbf{F}_4),\ X = V(x^3 + y^2 + y)$



A Goppa Code from

Take $G = mQ$, $D = \sum_{i=1}^{8} P_i$.

It can be seen easily that $x \in L(2Q)$ and $y \in L(3Q)$. In fact $L(3Q) = \mathsf{Span}\{1, x, y\}$.

The Goppa code $C_L(D, 3Q)$ is the span of the rows of the matrix:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & \alpha & \alpha & \alpha^2 & \alpha^2 \\ 0 & 1 & \alpha & \alpha^2 & \alpha & \alpha^2 & \alpha & \alpha^2 \end{pmatrix}$$

By Bézout's theorem, this code has parameters $[8, 3, 5]$ over $\mathbf{F}_4$ (any ≤ 2 errors in a received word can be corrected by nearest neighbor decoding).

To “ rder Domains”

- We took $G = mQ$ and $D =$ sum of all other $\mathbf{F}_q$-rational points to maximize n (gives the class of “one-point Goppa codes”)

- Also, X in a “special position” (only one point at infinity) $\Rightarrow$ polynomials in affine coords give elements of $L(mQ)$, $m \geq 1$. The pole orders of the affine coord funs generate the Weierstrass semigroup of X at Q. (Can always achieve this by reembedding X by $|kQ|$, $k \geq 2g-1$ if we want.)

- Can completely describe construction of codes and a good decoding algorithm (Berlekamp-Massey-Sakata) via algebra of the ring $R = L(\infty Q) = \cup_{m=0}^{\infty} L(mQ)$, and the discrete valuation v_Q on $\mathbf{F}_q(X)$.

A Generalization

Høholdt, van Lint, and Pellikaan (building on *a lot* of previous work) recently introduced the following idea:

Def. Let R be a $\mathbf{F}_q$-algebra. Let $(\Gamma, +, \prec)$ be a well-ordered semigroup. An *order, or weight, function* is a surjective mapping $\rho : R \to \{-\infty\} \cup \Gamma$ satisfying:

1. $\rho(f) = -\infty \Leftrightarrow f = 0$

2. $\rho(cf) = \rho(f)$ for all $f \in R$, all $c \neq 0$ in $\mathbf{F}_q$

3. $\rho(f + g) \preceq \max_{\prec}\{\rho(f), \rho(g)\}$

4. if $\rho(f) = \rho(g) \neq -\infty$, then $\exists\ c \neq 0$ in $\mathbf{F}_q$ such that $\rho(f - cg) \prec \rho(f)$

5. $\rho(fg) = \rho(f) + \rho(g)$

First Properties

- Axioms 1 and 5 imply that R must be a domain; a ring with an order function is called an *order domain*.

- Let $K = QF(R)$.

- From now on, restrict to case Γ a sub-semigroup of $\mathbf{Z}^r_{\geq 0}$, some $r \geq 1$, so *finitely generated*.

- Then WLOG, may assume $r = tr.deg._{\mathbf{F}_q}(K)$.

- "order" refers to the ordered $\mathbf{F}_q$ basis of R with distinct ρ-values guaranteed by axiom 4

Some Examples

- As above, $R = L(\infty Q)$ is an order domain for any point $Q \in X$, a smooth curve. $\Gamma =$ Weierstrass semigroup of X at Q, $\rho(f) = -v_Q(f)$. (Goppa)

- $R = \mathbf{F}_q[X_1, \ldots, X_r]$ is an order domain taking $\Gamma = \mathbf{Z}^r_{\geq 0}$, $\prec$ a monomial order, $\rho(f) = \alpha$ if $LT_\prec(f) = X^\alpha$ for $f \neq 0$. (Reed-Muller)

- Can construct all order domains with a given Γ, as in following example. Take $r = 2$, $\Gamma = \langle (0,2), (1,1), (3,0) \rangle \subset \mathbf{Z}^2_{\geq 0}$ ordered by graded lex (for example).

Examples, cont.

- 3 generators for $\Gamma \Rightarrow$ there is a surjective ring homomorphism:

  $$\phi : \mathbf{F}_q[X, Y, Z] \to R,$$

  where if $\phi(X) = x$, $\phi(Y) = y$, etc. $\rho(x) = (0, 2)$, $\rho(y) = (1, 1)$, $\rho(z) = (3, 0)$.

- Easy to see that all relations between $\rho(x)$, $\rho(y)$,$\rho(z)$ are generated by $\rho(x^3z^2) = \rho(y^6)$

- For axioms to hold, must have $\rho(y^6 - cx^2z^3) < \rho(y^6)$ for some $c \neq 0$. $\Rightarrow R \cong \mathbf{F}_q[X, Y, Z]/I$, where $I = \langle F \rangle$,

  $$F = Y^6 - cX^2Z^3 + \text{lower order terms}$$

- Can check all such R are order domains (and all deformations of the *monomial algebra* $\mathbf{F}_q[\Gamma] = \mathbf{F}_q[v^2, uv, u^3]$).

An "Extrinsic" Characterization

Theorem 6 *(Geil-Pellikaan) Let R be an order domain with a given finitely-generated value semigroup $\Gamma \subset \mathbf{Z}^r_{\geq 0}$. Let*

$$R_\Gamma = \mathbf{F}_q[\Gamma] \cong \mathbf{F}_q[X_1, \ldots, X_s]/I_\Gamma$$

be the monomial (or toric) algebra associated to Γ. Then R has a flat deformation to R_Γ (coming from a presentation of R similar to our last example above).

rder Domains and Valuations

On the other hand, there is a close connection between order domains and *valuations* on function fields (recall the Goppa code case!)

- If R is an order domain with field of fractions K, then

  $$S_\rho = \{f/g \in K : \rho(g) \geq \rho(f)\}$$

  is a *valuation ring* in K (that is, for all $h \neq 0$ in K, either $h \in S_\rho$, or $1/h \in S_\rho$).

- S_ρ is a local ring with maximal ideal $M_\rho = \{f/g \in K : \rho(g) > \rho(f)\}$.

- R is in a special "$\mathbf{F}_q$-complementary position to S_ρ" in K: $S_\rho \cap R = \mathbf{F}_q$ and $S_\rho = S_\rho \cap R + M_\rho$.

rder Domains and Valuations, cont.

- Conversely, using results of Mosteig and Sweedler, can show

  Theorem 7 *Any valuation on a function field, with rational rank equal to the transcendence degree, with center a point "at infinity" on a projective model yields a corresponding order domain in the function field.*

- see `arXiv:math.AC/0304292` for more details.

- In particular, applies to varieties of interest in coding theory (Hermitian hypersurfaces, Grassmannians, flag varieties, ...)

Codes From rder Domains

To construct codes from an order domain $R = \mathbf{F}_q[x_1, \ldots, x_t]/I$, generalize Goppa's construction:

- Let Δ be the ordered basis of R (ordered by ρ value, or equiv. w-weight) given by the monomials in complement of $LT_>(I)$
- Let $X = V(I)$, and $X(\mathbf{F}_q) = \{P_1, \ldots, P_n\}$ be the set of $\mathbf{F}_q$-rational points on X
- Let V_ℓ be the span of the first ℓ elements of Δ
- Let $ev : R \to \mathbf{F}_q^n$: $ev(f) = (f(P_1), \ldots, f(P_n))$
- Get codes $E_\ell = ev(V_\ell)$, $C_\ell = Ev_\ell^\perp$.

Codes From rder Domains, cont.

- Results of O'Sullivan $\Rightarrow$ many features of one-point Goppa codes generalize to these codes from *all* order domains: Good bounds on minimum distance of C_ℓ codes (Feng-Rao-Duursma) tied to an efficient decoding algorithm (B-M-S)

- Construction of good codes by this method still requires finding X with many $\mathbf{F}_q$-rational points, good minimum distance properties, ...

- On the other hand, there is the possibility of exploiting known higher-dimensional varieties of interest.

Final Example

Consider the Hermitian surface:

$$\mathcal{H} = V(X_0^{q+1} + X_1^{q+1} + X_2^{q+1} - X_3^{q+1})$$

in $\mathbf{P}^3$ over the field $\mathbf{F}_{q^2}$.

- $V_1 = V(X_0)$ is a smooth Hermitian curve on $\mathcal{H}$

- $V_2 = \{(0 : 1 : \delta : 0)\}$ is a point on V_1 if $\delta^{q+1} = -1$.

- From standard construction of *composite divisorial valuations* corresponding to flags of subvarieties, we get an order domain structure on the ring $L(\infty V_1)$ on $\mathcal{H}$ (the subring of the function field consisting of functions with poles only along V_1 (arbitrary order).

Hermitian Surface Codes

- Can also derive explicit presentation as deformation of the toric algebra for

$$\Gamma = \langle (1, q), (1, q+1), (1, 0) \rangle$$

- $\mathcal{H}$ has $(q^2+1)(q^3+1)$ $\mathbf{F}_{q^2}$-rational points, of which q^3+1 lie on the plane section V_1. $\Rightarrow$ we get generalized Goppa codes from $\mathcal{H}$ with $n = q^2(q^3+1) = q^5 + q^2$.

- With $\ell = 4$, for instance, evaluating $1, X_1, X_2, X_3$ yields E_4 code with $k = 4$ and $d = q^5 - q^3$ (maximum number of zeroes is $(q+1)q^2 = q^3 + q^2$).

- With $q = 2$, $d = 2^5 - 2^3 = 24$. (Best known code with $n = 36, k = 4$ over $\mathbf{F}_4$ has $d = 25$.)

Conclusion

- Ironically, when order domains were introduced by Høholdt, van Lint, and Pellikaan, their goal was to "take the (hard) algebraic geometry out of the theory of Goppa codes" (!)

- As it turns out, their synthesis of that theory has made it possible to use even more commutative algebra and algebraic geometry to construct new examples of error control codes, generalize the existing decoding algorithms, etc.

§6. The BMS Decoding Algorithm

Berlekamp-Massey-Sakata algorithm applies to *duals of evaluation codes* from order domains (and similar m-dimensional cyclic codes).

Can formulate it

- "internally" in $R \cong \mathbf{F}_q[x_1, \ldots, x_t]/I$ (approach in lecture notes), or

- "externally" in polynomial ring $\mathbf{F}_q[x_1, \ldots, x_t]$.

We'll take second approach here (and consider a very idealized version of the decoding problem that puts some of the complicating features in the background – indicate how those are addressed at end.)

Set-up

In §5, the evaluation code E_ℓ was defined to be the image of

$$\begin{aligned} ev_S : L_\ell &\to \mathbf{F}_q^n \\ f &\mapsto (f(P_1), \ldots, f(P_n)), \end{aligned}$$

where $S = \{P_1, \ldots, P_n\}$ = the $\mathbf{F}_q$-rational points of $\mathbf{V}(I)$ and L_ℓ is the $\mathbf{F}_q$-span of the first ℓ elements of the ordered basis Δ for R.

The *dual code* is $C_a = E_a^\perp$. Codewords of E_a furnish *parity check equations* for the codewords of C_a: For $y \in \mathbf{F}_q^n$,

$$y \in C_a \Leftrightarrow \langle y, ev(f) \rangle = \sum_{j=1}^{n} y_j f(P_j) = 0, \text{ all } f \in L_\ell.$$

Syndromes

Analogs of the *syndromes* used in §4 for decoding Reed-Solomon codes. One way to package: *syndrome mapping* associated to $y \in \mathbf{F}_q^n$.

$$\begin{aligned} S_y : \mathbf{F}_q[x_1, \ldots, x_t] &\to \mathbf{F}_q \\ f &\mapsto \sum_{j=1}^{n} y_j f(P_j) \end{aligned}$$

If $x \in C_a$ is sent and $y = x + e$ is received, can use $S_e(f)$ to correct.

$$S_y(f) = S_x(f) + S_e(f) = S_e(f)$$

for all $f \in L_a$.

Decoding

- Components of vectors c, e, y are indexed by the $\mathbf{F}_q$-rational points of X, in particular $e = (e_P : P \in X(\mathbf{F}_q))$ ($X = \mathbf{V}(I)$ from presentation of R).

- The error vector e is determined by *locations* of the nonzero entries $\leftrightarrow$ a subset of $X(\mathbf{F}_q)$, and the *error values* $e_P \neq 0$.

- Strategy: Determine the *error-locator ideal* $I_e =$

$$\{f \in \mathbf{F}_q[x_1, \ldots, x_t] : f(P) = 0 \text{ if } e_P \neq 0\}$$

- In fact, BMS produces a Gröbner basis $G = \{g_1, \ldots, g_k\}$ for I_e with respect to the order $>$ used for the presentation of R.

Syndrome Series

Another way to package the syndromes: Given e, and $u \in \mathbf{Z}^t_{\geq 0}$ (t from pres'n of R as a quotient of a polynomial ring), we let

$$E_u = \langle e, ev(x^u) \rangle = \sum_P e_P x^u(P)$$

be the syndromes of the error vector.

Define:

$$\mathcal{S}_e = \sum_{u \geq 0} E_u x^{-u}$$

in $T = \mathbf{F}_q[[x_1^{-1}, \ldots, x_s^{-1}]]$.

Also let $S = \mathbf{F}_q[x_1, \ldots, x_t]$.

Algebraic Context

Elements of T act as *linear functionals* on S. Let $A = \sum_u a_u x^{-u} \in T$ and $B = \sum_v b_v x^v \in S$. Then $A(B) \in \mathbf{F}_q$ is the degree 0 part of the product:

$$A(B) = \sum_u a_u b_u.$$

T also has the structure of S-module according to the following product operation: if A, B above and $C = \sum_w c_w x^w$, then $(C \cdot A)(B) = A(CB)$.

For monomials: $x^\alpha \in S$, $x^{-\beta} \in T$, then

$$x^\alpha \cdot x^{-\beta} = \begin{cases} x^{\alpha-\beta} & \text{if in } T \\ 0 & \text{otherwise} \end{cases}$$

(extend by linearity).

(Same as one version of the theory of *duality* and Macaulay *inverse systems*. In fact T is the linear dual space of S, under the pairing defined above.)

The Key Equation for C_ℓ

Theorem 8 *Let $\mathcal{S}_e$ be as above. Then $f \in I_e$ if and only if*

(KE) $$f \cdot \mathcal{S}_e = 0.$$

proof: Let $f = \sum_m f_m x^m$. Then

$$\begin{aligned} f \cdot \mathcal{S}_e &= \Big(\sum_m f_m x^m\Big) \cdot \Big(\sum_{u \geq 0} E_u x^{-u}\Big) \\ &= \sum_{r \geq 0} \Big(\sum_m f_m E_{m+r}\Big) x^{-r} \end{aligned}$$

Hence $f \cdot \mathcal{S}_e = 0 \Leftrightarrow \sum_m f_m E_{m+r} = 0$ for all $r \geq 0$.

Proof, cont.

By the definition of the E_u,

$$\begin{aligned}
\sum_m f_m E_{m+r} &= \sum_m \left(f_m \sum_P e_P x^{m+r}(P) \right) \\
&= \sum_P e_P x^r(P) \left(\sum_m f_m x^m(P) \right) \\
&= \sum_P e_P x^r(P) f(P)
\end{aligned}$$

If $f \in I_e$, then sum is zero for all $r \geq 0$, so $f \cdot \mathcal{S}_e = 0$. Conversely, if $f \cdot \mathcal{S}_e = 0$, then

$$\sum_P e_P f(P) x^r(P) = 0$$

for all $r \geq 0$. Hence $e_P f(P) = 0$ for all P, so $f \in I_e$.//

Idealized Version of Decoding

Suppose we (somehow) knew the full syndrome series $\mathcal{S}_e$.

Note: in actual decoding, we only know the truncation containing terms $E_u x^u$ corresponding to basis x^u of L_ℓ. But because of equations $x_i^q - x_i = 0$ over $\mathbf{F}_q$, a finite initial segment determines the rest (in fact $\mathcal{S}_e$ represents a rational function of the x_i as was true in RS case in §4).

From duality setup, the S-submodule of T generated by $\mathcal{S}_e$ contains all the information needed to recover I_e, and here's one way to do it!

Same approach could also be used for RS decoding (gives version of Berlekamp-Massey).

Decoding algorithm, (VERY) rudimentary version

Algorithm.

Input: $\mathcal{S}_e$, monomial order $>$ on S
compatible with order function in R
Output: $G =$ Gröbner basis for I_e,
$B =$ monomial basis for S/I_e.
Initialize: $G := \{\}$; $B := \{1\}$;
$List := \{L_0 := x^0 \cdot \mathcal{S}_e\}$
`for` monomials x^α in $>$ order `do`
$L_\alpha := x^\alpha \cdot \mathcal{S}_e$
`if` $\{L_\alpha\} \cup List$ linearly dependent `then`
$G := G \cup \{x^\alpha + \sum_\beta c_\beta x^\beta\}$
`else`
$B := B \cup \{x^\alpha\}$;
$List := List \cup \{x^\alpha \mathcal{S}_e\}$

Comments

- Reason this works: $(\sum c_\alpha x^\alpha) \cdot \mathcal{S}_e = 0 \in T$ is equivalent to $\sum c_\alpha x^\alpha \in I_e$ by theorem on (KE)

- Experienced "Gröbnerians" will recognize the outline of *Buchberger-Möller*, or *FGLM* here. BMS is in same family of algorithms, as Teo Mora has observed(!)

- Deliberately vague about stopping criteria. After a finite number of steps B will stabilize at size wt(e)

- "Real" BMS algorithm is more economical in that it only stores a set of maximal elements for B and minimal elements for complement of B, updates those as it proceeds

Further Comments

- But, main simplification here is actually the assumption that full $\mathcal{S}_e$ is known (not just the truncation).

- In "real" BMS algorithm, this is handled by an ingenious idea of Feng and Rao.

- Additional terms in $\mathcal{S}_e$ are computed as needed

- Uses a "majority voting" idea to determine syndrome values not known directly from received word. See references for the details here(!)

- Also means that elements of the Gröbner basis "in progress" might be modified as the algorithm proceeds and more terms added to $\mathcal{S}_e$.

Final Example

For our final example we take the code $C_5 = E_5^{\perp}$ from the order domain

$$R \cong \mathbf{F}_4[x,y]/\langle x^3 + y^2 + y \rangle$$

as in §5 (the Hermitian curve over $\mathbf{F}_4$). C_5 has parameters $[n,k,d] = [8,3,5]$.

As above we will assume the full $\mathcal{S}_e$ is known for some error of weight ≤ 2. Writing α for a primitive element of $\mathbf{F}_4$, suppose

$$\begin{aligned}\mathcal{S}_e \;=\; & \alpha + \alpha^2 x^{-1} + x^{-2} + \alpha y^{-2} \\ & + \alpha x^{-3} + \alpha^2 x^{-1} y^{-2} + \alpha y^{-3} \\ & + x^{-2} y^{-2} + \alpha^2 x^{-1} y^{-3} \\ & + \alpha x^{-3} y^{-2} + x^{-2} y^{-3} + \alpha x^{-3} y^{-3} \\ & + \cdots\end{aligned}$$

(Rest is determined from this via relations $x^4 - x = y^4 - y = 0$.)

Example, continued

We will use the $>_{(2,3),lex}$ order ($x > y$ for the lex), so monomials will be processed in the order:

$$1 < x < y < x^2 < xy < y^2 < x^3 < x^2y < \cdots$$

Already with $1 \cdot \mathcal{S}_e$ and $x \cdot \mathcal{S}_e$, we find a linear dependence:

$$x \cdot \mathcal{S}_e + \alpha \cdot \mathcal{S}_e = 0 \in T$$

So we add $x + \alpha$ to G.

Next, $y \cdot \mathcal{S}_e$ and $\mathcal{S}_e$ are linearly independent, so we add y to B.

With the next two monomials that are processed: x^2 and xy, we find consequences of the first relation added to G: $x^2 + \alpha^2$ (in characteristic 2!), and $xy + \alpha y$.

Example, continued

These are added to G in the basic form of the algorithm, though of course they are unnecessary (will not be a part of a reduced Gröbner basis for I_e). Could avoid this by only using monomials that are not multiples of leading terms of elements of G already found.

With y^2, we find a new relation:

$$(y^2 + y + 1) \cdot \mathcal{S}_e = 0 \in T$$

So $y^2 + y + 1$ is also inserted in G.

In fact, we have found the Gröbner basis of I_e at this point:

$$\{x + \alpha, y^2 + y + 1\}$$

Error locations are the solutions of these:

$$P = (\alpha, \alpha), (\alpha, \alpha^2)$$

Then determine error values by generalized Forney formulas!