# Seminari de Geometria Algebraica

## UB-UAB-UPC

# Error-correcting codes: mathematics and computations

S. Xambó

UPC

5/5/2017

**Abstract**. This talk will have two parts. In the first, the family of alternant block-error correcting codes (and some important subfamilies, like Reed-Solomon, Bose-Chaudhuri-Hocquenghem, and classical Goppa codes) will be reviewed and two general decoding algorithms will be described. The second part will be focused on three aspects of algebraic geometry codes: mathematical structure, search for curves over finite fields with many rational points, and decoding algorithms. All along, some of the computations will be illustrated by means of a package (PyECC) developed in collaboration with Narcís Sayols (work in progress). Rafel Farré is to be acknowledged for his collaboration on an improvement of the PGZ decoder for alternant codes, and Santiago Molina for improved procedures to count points on curves.

**Remark.** In this talk, applications of codes, most notably the use of elliptic curves over finite fields in what is called *post-quantum cryptography*, will not be considered.

# ECC & PyECC

- **Introduction**. A short tour of the RS land (Reed-Solomon), with a quick visit to PyECC. Linear codes over $\mathbb{F}_q$.
- **Alternant codes**. Definition and basic properties. RS revisited. A shortcut approach GRS, BCH, and classical Goppa codes.
- **The simplest decoding algorithm**. Syndromes and the PGZ (Peterson-Gorenstein-Zierler) approach to error-location. An improved version. Error evaluation. On the PyECC implementation.
- **Remarks on other decoders**. The Berlekamp-Massey-Sugiyama approach (BMS). A variant of the PGZ algorithm.
- **Notes on implementations**. Code constructors.
- **Backgroud on AG codes**. The general idea. The Goppa insights. Counting points on curves/$\mathbb{F}_q$. Examples of Goppa codes.
- **Demos**.

- $q$ A prime power (say $p^r$).
- $\mathbb{F}_q$ The field of $q$ elements.
- $\boldsymbol{\alpha} = \alpha_1, \ldots, \alpha_n$ Disctinct non-zero elements of $\mathbb{F}_q$ ($n < q$).
- $G = V_k(\boldsymbol{\alpha})$ Vandermonde matrix of $k$ rows on $\alpha_1, \ldots, \alpha_n$. Has rank $k$ for $k \leqslant n$.
- $C = \mathrm{RS}(\boldsymbol{\alpha}, k)$ Subspace of $\mathbb{F}_q^n$ generated by the rows of $G$.
- $\mathbb{F}_q[X]_k$ Polynomials of degree $< k$ over $\mathbb{F}_q$.

Let $\epsilon : \mathbb{F}_q[X]_k \to \mathbb{F}_q^n$ such that $f \mapsto x = (f(\alpha_1), \ldots, f(\alpha_n))$.

Proposition. If $k \leqslant n$, the map $\epsilon$ is injective and its image is $C$.

Proof. If $x = 0$, then $f$ has $n$ roots. Since $n > k - 1 \geqslant \deg(f)$, $f = 0$.

Now $\epsilon(X^j)$ is the $j$-the row of $G$ for $j = 0, \ldots, k-1$ and so the image of $\epsilon$ is $C$. □

The *weight* of a vector $y \in \mathbb{F}_q^n$, denoted $\mathrm{wt}(y) = |y|$, is *the number of non-zero entries of* $y$.

Proposition The minimum weight of non-zero vectors of $C$ is

$$d = n - k + 1 = r + 1.$$

Proof. Let $x = \epsilon(f)$ be a non-zero vector of $C$. Since the number of roots of $f$ is $\leqslant k - 1$, the number of non-zero entries of $x$ is $\geqslant n - (k-1) = n - k + 1$. On the other hand, the polynomial $f = (X - \alpha_1) \cdots (X - \alpha_{k-1})$ has degree $< k$ and $k - 1$ roots exactly among the $\alpha_j$ and so $|\epsilon(f)| = n - k + 1$. □

- $f \in \mathbb{F}[X]_k$ *information vector*
- $x = \epsilon(f)$ *code vector* (encoding of $f$).
- $e \in \mathbb{F}_q^n$ *error vector* (produced through 'transmission').
- $y = x + e$ *received vector*.
- $t$ $\lfloor (n-k)/2 \rfloor = \lfloor r/2 \rfloor$ (*correcting capacity*)

Theorem. If $|e| \leqslant t$, then $x$ can be recovered from $y$ (and $C$).

Proof. We can find polynomials $P = p_0 + p_1 X + \cdots + p_{n-t-1} X^{n-t-1}$ and $Q = q_0 + q_1 X + \cdots + q_{n-t-k} X^{n-t-k}$, not both 0, such that $P(\alpha_j) + y_j Q(\alpha_j) = 0$ (this amounts to a homogeneous linear system of $n$ equations with $n - t + n - t - k + 1 = n + 1 + (n-k) - 2t > n$). For the $j$ such that $e_j = 0$, we have $P(\alpha_j) + f(\alpha_j) Q(\alpha_j) = 0$, and hence the polynomial $P + f\,Q$ has at least $n - |e| \geqslant n - t$ roots. But its degree is $\leqslant n - t - 1$, and hence $f = -P/Q$. □

The equation $P(\alpha_j) + y_j Q(\alpha_j) = 0$ is equivalent to

$$(p_0, \ldots, p_{n-t-1}) V_{n-t}(\boldsymbol{\alpha}) + (q_0, \ldots, q_{n-t-k-1}) V_{n-t-k}(\boldsymbol{\alpha}) \operatorname{diag}(y_0, \ldots, y_{n-1}) = 0.$$

This justifies the following algorithm (PyECC script):

```
def RSID(a,k,y):
    n = len(a); t = (n-k)//2
    P = vandermonde(a,n-t); Q = vandermonde(a,n-k-t)
    for j in range(n):
        Q[:,j] = y[j]*Q[:,j]
    PQ = left_kernel(stack(P,Q))[0]
    P = PQ[:(n-t)]; Q = PQ[(n-t):]
    [_,T] = polynomial_ring(field(a),'T')
    P = polynomial(P,T); Q = polynomial(Q,T)
    if not P%Q: return "RSID: Error"
    return  -P//Q
```

- A linear code of length $n$ defined over $\mathbb{F}_q$ is a vector subspace $C \subseteq \mathbb{F}_q^n$.
- If $C$ has dimension $k$, we say that $C$ is an $[n, k]$ code.
- The quotient $k/n$ is called the *rate* of $C$.
- The *Hamming distance* $\mathrm{hd}(y, y')$ of $y, y' \in \mathbb{F}_q^n$ is $|y - y'|$ (the number of indices $j \in \{1, \ldots, n\}$ such that $y_j \neq y'_j$).
- The *minimum distance* of $C$, denoted $d$, is the minimum of the distances $\mathrm{hd}(x, x')$ for $x, x' \in C$, $x \neq x'$. *It agrees with the minimum weight*.
- An $[n, k]$ code of minimum distance $d$ is said to be an $[n, k, d]$ code, or an $[n, k, d]_q$ if we need to recall $q$.
- *Singleton bound*: $d + k \leqslant n + 1$ (see [6], p. 25). For the RS codes, it is an equality (maximum distance separable, or just MDS codes).

Let $K = \mathbb{F}_q$ and $\bar{K} = \mathbb{F}_{q^m}$. Let $\alpha_1, \ldots, \alpha_n$ and $h_1, \ldots, h_n$ be elements of $\bar{K}$ such that $h_i, \alpha_i \neq 0$ for all $i$ and $\alpha_i \neq \alpha_j$ for all $i \neq j$. Consider the matrix

$$H = V_r(\alpha_1, \ldots, \alpha_n)\mathrm{diag}(h_1, \ldots, h_n) \in M_n^r(\bar{K}), \tag{1}$$

that is,

$$H = \begin{pmatrix} h_1 & \ldots & h_n \\ h_1\alpha_1 & \ldots & h_n\alpha_n \\ \vdots & & \vdots \\ h_1\alpha_1^{r-1} & \ldots & h_n\alpha_n^{r-1} \end{pmatrix} \tag{2}$$

We say that $H$ is the *alternant control matrix* of order $r$ associated with the vectors

$$\boldsymbol{h} = (h_1, \ldots, h_n) \quad \text{and} \quad \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n).$$

To make explicit that the entries of $\boldsymbol{h}$ and $\boldsymbol{\alpha}$ (and hence of $H$) lie in $\bar{K}$, we will say that $H$ is defined over $\bar{K}$.

The $K$-code $A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$ defined by the control matrix $H$ is the subspace of $K^n$ whose elements are the vectors $x$ such that $xH^T = 0$. Such codes will be called *alternant codes*.

The *$H$-syndrome* of a vector $y \in \bar{K}^n$ is $s = yH^T \in \bar{K}^r$. Note that $A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$ is just the subspace of $K^n$ whose elements are the vectors with zero $H$-syndrome.

Proposition (Alternant bounds). If $C = A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$, then

$$n - r \geqslant \dim C \geqslant n - rm$$

and

$$d \geqslant r + 1$$

(*minimum distance alternant bound*).

Remark. Henceforth, see [6] for omitted proofs.

Proposition. $\mathrm{RS}(\boldsymbol{\alpha}, k) = A_K(\boldsymbol{h}, \boldsymbol{\alpha}, n-k)$, where $\boldsymbol{h} = (h_1, \ldots, h_n)$ is given by

$$h_i = 1/\prod_{j \neq i}(\alpha_j - \alpha_i). \tag{3}$$

Proof. On one hand, $h_i = (-1)^{i-1} D_i/D$, where $D = D(\alpha_1, \ldots, \alpha_n)$ and $D_i = D(\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n)$ are Vandermonde determinants. Now the vanishing for $s = 0, \ldots, n-2$ of the determinant

$$\begin{vmatrix} \alpha_1^s & \cdots & \alpha_n^s \\ 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{n-2} & \cdots & \alpha_n^{n-2} \end{vmatrix}$$

tells us that $\sum_{i=1}^{i=n} (-1)^{i-1} \alpha_i^s D_i = 0$, hence also $\sum_{i=1}^{i=n} \alpha_i^s h_i = 0$. This implies that $\mathrm{RS}(\boldsymbol{\alpha}, k) \subseteq A_K(\boldsymbol{h}, \boldsymbol{\alpha}, n-k)$. Equality follows from the equality of dimensions.

*Remark*

In this case $\bar{K} = K$, hence $m = 1$, and the alternant bounds are sharp. Indeed, we have $r = n - k$, hence $k = n - r$, while $n - k + 1 \geqslant d$ (by the Singleton bound) and $d \geqslant r + 1 = n - k + 1$ by the minimum distance alternant bound. In other words, $C$ is MDS.

An RS code is called *primitive* if the $\alpha_1, \ldots, \alpha_n$ are *all* the non-zero elements of $K$:

- $\mathrm{RS}([1, \alpha, \ldots, \alpha^{n-1}], k)$, $n = q - 1$, $\alpha$ a primitive root of $K$.

*Generalized Reed-Solomon codes*. The vector $\boldsymbol{h}$ in the definition of the code $\mathrm{RS}([\alpha_1, \ldots, \alpha_n], k)$ as an alternant code is obtained from $\boldsymbol{\alpha}$ by the formula (3). If we allow that $\boldsymbol{h}$ can be chosen possibly unrelated to $\boldsymbol{\alpha}$, but still with components in $K$, the resulting codes $A_K(\boldsymbol{h}, \boldsymbol{\alpha}, n - k)$ are called *Generalized Reed–Solomon* (GRS) codes, and we will write $\mathrm{GRS}(\boldsymbol{h}, \boldsymbol{\alpha}, k)$ to denote them. An argument as above shows that such codes have type $[n, k, n - k + 1]$. Notice that the code $A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$ is the intersection of the GRS code $A_{\bar{K}}(\boldsymbol{h}, \boldsymbol{\alpha}, r)$ with $K^n$.

*Bose-(Ray-)Chaudhuri-Hocquenghem codes*. These codes are denoted $\mathrm{BCH}(\alpha, \delta, l)$, where $\alpha \in \bar{K}^*$ and $\delta > 0$, $l \geqslant 0$ are integers (called the *design minimum distance* and the *offset*, respectively). When $l = 1$, we simply write $\mathrm{BCH}(\alpha, \delta)$ and say that the it is a *strict* BCH code. They are the most useful class of the so called *linear cyclic codes*.

Proposition. BCH codes can be defined as alternant codes as follows:

$$\mathrm{BCH}(\alpha, \delta, l) = A_K(\boldsymbol{h}, \boldsymbol{\alpha}, \delta - 1), \tag{4}$$

where $\boldsymbol{h} = (1, \alpha^l, \alpha^{2l}, \ldots, \alpha^{(n-1)l})$, $\boldsymbol{\alpha} = (1, \alpha, \alpha^2, \ldots, \alpha^{(n-1)})$, $n = \mathrm{period}(\alpha)$. In particular, $d \geqslant \delta$.

If $\alpha$ is a primitive element of $K$, and hence $n = q - 1$, we have the equality

$$\mathrm{BCH}(\alpha, n - k + 1) = \mathrm{RS}([1, \alpha, \ldots, \alpha^{n-1}], k).$$

Let $g \in \bar{K}[T]$ be a polynomial of degree $r > 0$ and let $\boldsymbol{\alpha} = \alpha_1, \ldots, \alpha_n \in \bar{K}$ be distinct non-zero elements such that $g(\alpha_i) \neq 0$ for all $i$.

Proposition. The (classical) *Goppa code* over $K$ associated with $g$ and $\boldsymbol{\alpha}$, $\Gamma(g, \boldsymbol{\alpha})$, can be defined as $A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$, where $\boldsymbol{h}$ is the vector $(1/g(\alpha_1), \ldots, 1/g(\alpha_n))$. Thus the minimum distance of $\Gamma(g, \boldsymbol{\alpha})$ is $\geqslant r + 1$ and its dimension $k$ satisfies $n - rm \leqslant k \leqslant n - r$. The minimum distance bound can be improved to $d \geqslant 2r + 1$ in the case that $K = \mathbb{F}_2$ and the roots of $g$ are distinct.

Let $C = A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$ be an alternant code. Let $t = \lfloor r/2 \rfloor$, that is, the highest integer such that $2t \leqslant r$. For reasons that will become apparent below, $t$ is called the *error-correction capacity* of $C$ (compare with what we have seen for the interpolation decoder of RS codes).

*Transmission channel terminology*

- $x \in C$ — *sent vector*
- $e \in \bar{K}^n$ — *error vector*
- $y = x + e$ — *received vector*
- Decoding problem — *to recover $x$ from $y$ (and $C$) when $l = |e| \leqslant t$.*

If $e_m \neq 0$, $m$ is an *error position*. Let $\{m_1, \ldots, m_l\}$ be the error positions and $\{e_{m_1}, \ldots, e_{m_l}\}$ the corresponding *error values*. The *error locators* $\eta_1, \ldots, \eta_l$ are defined by $\eta_k = \alpha_{m_k}$. Since $\alpha_1, \ldots, \alpha_n$ are distinct, the *knowledge of the $\eta_k$ is equivalent to the knowledge of the error positions*.

The monic polynonial $L(z)$ whose roots are the error locators is called the *error-locator polynomial*:

$$L(z) = \prod_{i=1}^{l}(z - \eta_i) = z^l + a_1 z^{l-1} + a_2 z^{l-2} + \cdots + a_l, \qquad (5)$$

where $a_j = (-1)^j \sigma_j$, $\sigma_j = \sigma_j(\eta_1, ..., \eta_l)$ the $j$-th elementary symmetric polynomial in the $\eta_i$ ($0 \leq j \leq l$, $a_0 = \sigma_0 = 1$ by convention).

The *syndrome* of $y$ is the vector $s = yH^T$, say $s = (s_0, \ldots, s_{r-1})$. Thus $s = 0$ if and only if $y \in C$.

Since $xH^T = 0$, we have $s = eH^T$. Inserting the definitions, we find:

$$s_j = \sum_{i=0}^{n-1} e_i h_i \alpha_i^j = \sum_{k=1}^{l} h_{m_k} e_{m_k} \alpha_{m_k}^j = \sum_{k=1}^{l} h_{m_k} e_{m_k} \eta_k^j \tag{6}$$

We will use the following notations:

$$A_l = \begin{pmatrix} s_0 & s_1 & \ldots & s_{l-1} \\ s_1 & s_2 & \ldots & s_l \\ \vdots & \vdots & \ddots & \vdots \\ s_{l-1} & s_l & \ldots & s_{2l-2} \end{pmatrix} \tag{7}$$

and the vector

$$\boldsymbol{b}_l = (s_l, \ldots, s_{2l-1}). \tag{8}$$

Proposition. If $\boldsymbol{a}_l = (a_l, ..., a_1)$ (see the fomula (5)), then

$$\boldsymbol{a}_l A_l + \boldsymbol{b}_l = 0. \tag{9}$$

Proof. Substituting $z$ by $\eta_i$ in the identity

$$\prod_{i=1}^{l}(z - \eta_i) = z^l + a_1 z^{l-1} + ... + a_l$$

we obtain the relations

$$\eta_i^l + a_1 \eta_i^{l-1} + \cdots + a_l = 0,$$

where $i = 1, ..., l$. Multiplying by $h_{m_i} e_{m_i} \eta_i^j$ and adding with respect to $i$, we obtain (using (6)) the relations

$$s_{j+l} + a_1 s_{j+l-1} + \cdots + a_l s_j = 0,$$

where $j = 0, ..., l-1$, and these relations are equivalent to the stated matrix relation. □

*Remark.* If we knew $l$, and that the matrix $A_l$ in equation (9) were non-singular, then we could determine $\boldsymbol{a}_l$ and hence $L(z)$.

In the usual approach, one proves that $l$ is the first integer in the sequence $t, t-1, \ldots, 1, 0$ such that $\det(A_l) \neq 0$ ([6], §4.4). This allows to determine $l$, then $\boldsymbol{a}_l$ and $L(z)$, and finally the error locators (and locations) by finding the roots of $L(z)$.

But there is a more efficient and neat way that we explain next (here I will follow [2]).

The main object is the matrix

$$S = \left(\begin{array}{cccccc} s_0 & s_1 & \cdots & s_{l-1} & s_l & \cdots & s_t \\ s_1 & s_2 & \cdots & s_l & s_{l+1} & \cdots & s_{t+1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ s_{l-1} & s_l & \cdots & s_{2l-2} & s_{2l-1} & \cdots & s_{t+l-1} \\ \hline \vdots & \vdots & & \vdots & \vdots & & \vdots \\ s_{t-1} & s_t & \cdots & s_{t+l-2} & s_{t+l-1} & \cdots & s_{2t-1} \end{array}\right) \tag{10}$$

Note that $2t - 1 \leqslant r - 1$, so that all components are well defined. Note also that the $l \times l$ submatrix at the upper left corner is the matrix $A_l$ defined in (7) an that the column $(s_l, s_{l+1}, \ldots, s_{2l-1})^T$ to its right is the vector $\boldsymbol{b}_l$ defined in (8).

In next theorem we use the following notation: $V_s = V_s(\eta_1, \ldots, \eta_l)$. Thus the $i$-th row of $V_s$, for $0 \leqslant i \leqslant s - 1$, is the vector $(\eta_1^i, \ldots, \eta_l^i)$. We also write $D = \operatorname{diag}(h_{m_1} e_{m_1}, \ldots, h_{m_l} e_{m_l})$.

**Theorem.** $S = V_t D V_{t+1}^T$.

**Proof.** Let $0 \leqslant i \leqslant t-1$ and $0 \leqslant j \leqslant t$. Then the $j$-th column of $DV_{t+1}^T$ is the column vector $(h_{m_1} e_{m_1} \eta_1^j, \ldots, h_{m_l} e_{m_l} \eta_l^j)^T$. It follows that the element in row $i$ column $j$ of $V_t D V_{t+1}^T$ is $h_{m_1} e_{m_1} \eta_1^{i+j} + \cdots + h_{m_l} e_{m_l} \eta_l^{i+j} = s_{i+j}$ (by the equation (6)). □

**Corollary.** The rank of $S$ is $l$ and the matrix $A_l$ is non-singular.

**Proof.** Since $D$ has rank $l$, the rank of $S$ is at most $l$. On the other hand, the theorem shows that $A_l = V_l D V_l^T$ and therefore

$$\det(A_l) = \det(V_l)^2 \det(D) \neq 0.$$

Note that $\det(V_l)$ is the Vandermonde determinant of $\eta_1, \ldots, \eta_l$, which is non-zero because the error locators are distinct. □

Corollary. The Gauss-Jordan algorithm applied to the matrix $S$ returns a matrix that has the form

$$\left(\begin{array}{ccccc} 1 & 0 & \cdots & 0 & -a_l & * \\ 0 & 1 & \cdots & 0 & -a_{l-1} & * \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_1 & * \\ \hline \vdots & \vdots & & \vdots & \vdots & \vdots \end{array}\right) \tag{11}$$

where $*$ denotes unneeded values *(if any)* and the vertical dots below the horizontal line denote that all its elements *(if any)* are zero. This matrix gives at the same time $l$, the number of errors, and the coefficients of the error-locator polynomial. □

*Remark*. We write GJ($S$) for the PyECC form of the Gauss-Jordan procedure that delivers the column on the right of the matrix $Id_l$.

According to (6),the error values $e_{m_1}, ..., e_{m_l}$ satisfy the following system of linear equations:

$$h_{m_1} e_{m_1} \eta_1^j + h_{m_2} e_{m_2} \eta_2^j + ... + h_{m_l} e_{m_l} \eta_l^j = s_j \ (0 \leqslant j \leqslant l-1).$$

In matrix form, this is becomes de relation

$$\begin{pmatrix} h_{m_1} & h_{m_2} & \dots & h_{m_l} \\ h_{m_1}\eta_1 & h_{m_2}\eta_2 & \dots & h_{m_l}\eta_l \\ h_{m_1}\eta_1^2 & h_{m_2}\eta_2^2 & \dots & h_{m_l}\eta_l^2 \\ \vdots & \vdots & \ddots & \vdots \\ h_{m_1}{\eta_1}^{l-1} & h_{m_2}{\eta_2}^{l-1} & \dots & h_{m_l}{\eta_l}^{l-1} \end{pmatrix} \begin{pmatrix} e_{m_1} \\ e_{m_2} \\ e_{m_3} \\ \vdots \\ e_{m_l} \end{pmatrix} = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_{l-1} \end{pmatrix}. \quad (12)$$

Since the matrix is the alternant matrix

$$H([h_{m_1}, \dots, h_{m_l}], [\eta_1, \dots, \eta_l], l),$$

the error values are uniquely determined and the vector $x$ can be recovered.

At computation time, this system is solved with GJ.

## *Algorithm PGZm*

- Get the syndrome $s = (s_0, ..., s_{r-1}) = yH^T$. If $s = 0$, return $y$.
- Form the matrix $S$ as in the equation (10).
- Set $\boldsymbol{a} = -\mathrm{GJ}(S)$ (equation (11)). After this we have $a_1, ..., a_l$, hence also the error-locator polynomial $L$.
- Find the elements $\alpha_j$ that are roots of the polynomial $L$. If the number of these roots is $< l$, return *Error*. Otherwise let $\eta_1, ..., \eta_l$ be the error-locators corresponding to the roots and set $M = \{m_1, \ldots, m_l\}$, where $\eta_i = \alpha_{m_i}$.
- Find the error values $e_m$, for all $m \in M$, by solving (12). If any of the values of $e_m$ is not in $K$, return *Error*.
- Return $y - e$.

Theorem. PGZm corrects up to $t$ errors. □

```
def PGZm(y,C):
    h = h_(C); r = r_(C);
    alpha = a_(C); H = H_(C); K = K_(C)
    s = y * transpose(H)
    if is_zero(s): return y
    S = hankel_matrix(s)
    c0 = S[:,0]   # first column of S
    a = -GJ(S); l = len(a)
    a = reverse(a)
    K1 = K_(H)
    [_,z] = polynomial_ring(K1,'z','K1[z]')
    L = hohner([1]+a,z)
    R = [s for s in alpha if evaluate(L,s)==0]
    if len(R) < l: return "PGZm: Error"
    M = [alpha.index(r) for r in R]
    h1 = [h[m] for m in M]
    V = alternant_matrix(h1,R,l)
    v = c0[:l]
    V1 = splice(V,v)
    w = GJ(V1)
    for t in w:
        t = pull(t,K)
        if not belongs(t,K):
            return "PGZ: Error"
    for j in range(len(M)):
        y[M[j]] -= w[j]
    return pull(y,K)
```

- $L(z) = (z - \eta_1) \cdots (z - \eta_l)$ *error-locator polynomial*
- $S(z) = s_0 z^{r-1} + \cdots + s_{r-1}$ *syndromy polynomial*
- $E(z) = -\sum_{i=1}^{i=l} h_{m_i} e_{m_i} \eta_i^r \prod_{j \neq i} (z - \eta_i)$ *error-evaluator*
- $E(z) = L(z)S(z) \mod z^r$ *key equation*
- $e_{m_k} = -E(\eta_k)/h_{m_k}\eta_k^r L'(\eta_k)$ *Forney's formula*

Proof of the key equation:

$$E(z) = L(z) \sum_{i=1}^{i=l} h_{m_i} e_{m_i} \eta_i^{r-1}/(1 - z/\eta_i)$$

$$= \sum_{i=1}^{i=l} h_{m_i} e_{m_i} \eta_i^{r-1}(1 + z/\eta_i + \cdots + z^{r-1}/\eta_i^{r-1}) \mod z^r$$

$$= \sum_{i=1}^{i=l} h_{m_i} e_{m_i} (\eta_i^{r-1} + \eta_i^{r-2} z + \cdots + z^{r-1}) = S(z).$$

*Sugiyama's procedure*. Let $r_0 = z^r$, $r_1 = S(z)$, $v_0 = 0$, $v_1 = 1$. For $j \geqslant 2$, define recursively $v_j = v_{j-2} - q_j v_{j-1}$, with $q_j$ and $r_j$ the quotient and remainder of the Euclidean division of $r_{j-2}$ by $r_{j-1}$:

$$r_{j-2} = q_j r_{j-1} + r_j, \ \deg(r_j) < \deg(r_{j-1}).$$

Stop as soon as $\deg(r_j) < t$ and return the pair $(v_j, r_j)$.

PyECC implementation:

```
def sugiyama(r0,r1,t):
    v0 = 0; v1 = 1
    while t <= degree(r1):
        [q,r] = quo_rem(r0,r1)
        v = v0 - q*v1
        r0 = r1; r1 = r
        v0 = v1; v1 = v
    return (v1,r1)
```

Theorem. $L(z) = \rho v_j$ and $E(z) = \rho r_j$ for some $\rho \in \bar{K}^*$. □

Corollary. $L(z)$ and $v_j$ have the same roots, so $v_j$ can be used as error locator. Moreover, Forney's formula tells us that the error evaluation can be done using $v_j$ and $r_j$ instead of $L(z)$ and $E(z)$.

1. Compute $[s_0, \ldots, s_{r-1}] = yH^T$ (syndrome vector).
2. Define $S(z) = s_0 z^{r-1} + \cdots + s_{r-2} z + s_{r-1}$.
3. Compute $[L, E] = \text{sugiyama}(z^r, S, t)$.
4. Make a list $M = [m_1, \ldots, m_l]$ of the indexes $m$ such that $L(\alpha_m) = 0$. If $l < \deg(L)$, return *Error*.
5. For each $m \in M$, compute $e = (\alpha_m \cdot E(\alpha_m)/(h_m \alpha_m^r L'(\alpha_m))$. If $e \notin K$, return *Error*, otherwise replace $y_m$ by $y_m + e$ and continue the loop.
6. If $y$ is not a code vector, return *Error*. Otherwise return $y$.

Theorem. The BMS algorithm corrects up to $t$ errors. □

In the PGZm algorithm, the error evaluation (based on solving a linear system of equations) can be replaced by a Forney evaluation.

Indeed, the key congruence

$$E(z) = L(z)S(z) \mod z^r$$

gives $E(z)$, as the degree of this polynomial is at most $t$. Finally, having $E$ and $L$, we can find the error values with Forney's formula.

- In PyECC a code C is represented by a record-like structure with fields that allow to get data from the code or store new information about it (*dynamic table*).
- The labels of the table entries end with an underscore, but otherwise tend to mimic the mathematical symbols.
- For example, for an alternant code $\mathrm{C}$, there is a field labeled $\mathrm{H\_}$ to hold the alternating matrix and the expression H_(C) delivers that matrix. Similarly, h_(C) and a_(C) yield the vectors $\boldsymbol{h}$ and $\boldsymbol{\alpha}$.

- AC(**h**,**a**,r,K): This constructs the alternating code $A_K(\boldsymbol{h}, \boldsymbol{\alpha}, r)$. It is our *main constructor*, as the others (described below) are in fact defined as special calls to AC following the procedures explained on slides 11, 14 and 15.
- RS(**a**,k): This yields the RS code $\mathrm{RS}(\boldsymbol{\alpha}, k)$, an $[n, k, n-k+1]$ code defined over the field to which the elements of $\boldsymbol{\alpha}$ belong.
- GRS(**h**,**a**,k): As RS, but we have to supply $\boldsymbol{h}$ as a first argument.
- PRS(F,k): The *primitive RS code* of the finite field $F$. It is defined as RS(**a**,k), but taking as $\boldsymbol{\alpha}$ the list of non-zero elements of $F$.
- BCH(a,$\delta$,l): Supplies the code $\mathrm{BCH}(\alpha, \delta, l)$, where here a stands for an element $\alpha$ in a finite field. If $l = 1$, it can be omitted.
- Goppa(g,**a**): The Goppa code $\Gamma(g, \boldsymbol{\alpha})$.

A generalization of the alternant codes *setup* ($K = \mathbb{F}_q$, $\bar{K} = \mathbb{F}_{q^m}$):

- $X$ — *a 'space'*
- $\boldsymbol{f} = f_1, \ldots, f_k$ — *$\bar{K}$-valued 'functions' defined on $X$*
- $\boldsymbol{P} = P_1, \ldots, P_n$ — *distinct 'points' of $X$*
- $G = G(\boldsymbol{f}, \boldsymbol{P})$ — *the $k \times n$ matrix $(f_i(P_j))$*
- $C = C_K(\boldsymbol{f}, \boldsymbol{P})$ — *the code $\langle G \rangle_{\bar{K}} \cap K^n$*

In the case of alternant codes, $X$ is $\bar{K}^*$ and $f_i(x) = h_i x^{i-1}$.

*Curve/* $\mathbb{F}_q$: a non-singular absolutely irreducible projective curve defined over $\mathbb{F}_q$.

The Goppa (primal) setup:

- $X$ — *a curve/* $\mathbb{F}_q$
- $\boldsymbol{P} = P_1, \ldots, P_n$ — *distinct* $\mathbb{F}_q$*-rational points of* $X$
- $\Pi$ — *divisor* $P_1 + \cdots + P_n$
- $\Delta$ — *an* $\mathbb{F}_q$*-rational divisor of* $X$ *disjoint from* $\Pi$
- $\boldsymbol{f} = f_1, \ldots, f_k$ — *a basis of* $L(\Delta)$
- $C = \Gamma(\Delta, \boldsymbol{P})$ — *the code* $C(\boldsymbol{f}, \boldsymbol{P})$

*Remark.* The code $C$ is the image of the linear map $\epsilon : L(\Delta) \to \mathbb{F}_q^n$, $f \mapsto (f(P_1), \ldots, f(P_n))$.

*Example.* If $X = \mathbf{P}^1_q$, $\Delta = (k-1)[\infty]$, and $P_i = [\alpha_i]$, $\alpha_i \in K$, then $\Gamma(\Delta, \boldsymbol{P}) = \mathrm{RS}(\boldsymbol{\alpha}, k)$. The key fact is that $L((k-1)[\infty]) \simeq \mathbb{F}_q[X]_k$, as $\mathrm{ord}_\infty(X^i) = -i \geqslant -(k-1)$ for $i \leqslant k-1$.

Theorem (Goppa). $\Gamma(\Delta, \Pi) \sim [n, \ell(\Delta) - \ell(\Delta - \Pi), d \geqslant n - \deg(\Delta)]$.

Proof. By definition, the length of $\Gamma(\Delta, \Pi)$ is $n$. Since $\epsilon$ is surjective, its dimension is $k = \ell(\Delta) - \dim \ker \epsilon$. But $\ker \epsilon = L(\Delta - \Pi)$ (because $\Delta$ and $\Pi$ are disjoint) and hence $k = \ell(\Delta) - \ell(\Delta - \Pi)$.

As for the minimum distance bound (*Goppa bound*), we can argue as follows. Let $f \in L(\Delta)$, $f \neq 0$ and assume that $f$ has $m$ zeros among the $\boldsymbol{P}$. Then

$$0 = \deg(f) = \sum_{P \in \boldsymbol{P}} v_P(f) + \sum_{P \notin \boldsymbol{P}} v_P(f) \geqslant m - \sum_{P \notin \boldsymbol{P}} v_P(\Delta) = m - \deg(\Delta).$$

This shows that $m \leqslant \deg(\Delta)$. Thus the weight of $\epsilon(f)$ is $\geqslant n - \deg(\Delta)$ and therefore the minimum weight of the code (and hence the minimum distance $d$) is not less than $n - \deg \Delta$. □

Corollary. If $\deg \Delta) < n$, then

$$k \geqslant \deg(\Delta) + 1 - g.$$

In particular $k + d \geqslant n + 1 - g$.

If in addition $\deg(\Delta) > 2g - 2$, then $k = \deg(\Delta) + 1 - g$

Proof. If $\deg(\Delta) < n = \deg(\Pi)$, $\ell(\Delta - \Pi) = 0$ and hence $k = \ell(\Delta) = \deg(\Delta) + 1 - g + \ell(\Omega - \Delta)$ by RR ($\Omega$ a canonical divisor). Therefore, $k \geqslant \deg(\Delta) + 1 - g$, with equality if $2g - 2 = \deg(\Omega) < \deg(\Delta)$. □

Remark. $\Gamma(\Delta, \Pi)$ is MDS for $g = 0$ (this case essentially agrees with classical Goppa codes). For $g \geqslant 1$, in general it is not MDS, but for $n$ large with respect to $g$ it is nearly so (when $\deg(\Delta) < n$). On the other hand, we have ample freedom in choosing $n$. We turn to this question (how large can $n$ be) in next slides.

We will follow [4] (and references therein). Here $X$ denotes a curve/$\mathbb{F}_q$, and $g = g(X)$ its genus.

- $\nu_r = \nu_r(X)$ — $\#X(\mathbb{F}_{q^r})$
- $Z = Z(T) = \exp\left(\sum_{r=1}^{\infty} \nu_r \frac{T^r}{r}\right)$ — *Weil zeta function of X*

  $\nu_r = \frac{1}{(r-1)!}\frac{d^r}{dT^r} \log Z(T)|_{T=0}$.
- $Z(T) = \frac{P(T)}{(1-T)(1-qT)}$, $P(T) \in \mathbf{Z}[T]$ — *rationality*
- $P(T) = q^g T^{2g} P(1/qT)$ — *functional equation*

  $\deg(P) = 2g$
- $P(T) = \prod_{j=1}^{2g}(1 - \alpha_j T)$, $|\alpha_j| = \sqrt{q}$ — *'Riemann hypothesis' for X*
- $\nu_r = q^r + 1 - S_r$, $S_r = \sum_{j=1}^{2g} \alpha_j^r$

*Notations.* $c_0 = 1$ and $c_j = (-1)^j \sigma_j(\alpha_1, \ldots, \alpha_{2g})$ for $j = 1, \ldots, 2g$. Thus $P(T) = c_0 + c_1 T + \cdots + c_{2g} T^{2g}$ and $c_{2g} = q^g$.

*Input*: $\nu_1, \ldots, \nu_{2g}$ and $r > 2g$.

*Output*: $\nu_{2g+1}, \ldots, \nu_r$.

- For $j = 1, \ldots, 2g$, set $S_j = q^j + 1 - \nu_j$.
- Use the Girard-Newton formulas to recursively compute $c_1, \ldots, c_{2g}$:

$$c_j = -(S_j + c_1 S_{j-1} + \cdots + c_{j-1} S_1)/j.$$

- Use the Girard-Newton relation

$$S_j = -\left(c_1 S_{j-1} + \cdots + c_{2g-1} S_{j-(2g-1)} + c_{2g} S_{j-2g}\right)$$

to successively get $S_j$ and $\nu_j = q^j + 1 - S_j$ for $j = 2g+1, \ldots, r$.

**Proposition.** $c_{g+l} = q^l c_{g-l}$.

**Proof.** If $\alpha_j$ is a root, $\bar{\alpha}_j = q/\alpha_j$ is a root ($P$ has real coefficients). Possible real roots of $P$: $\pm\sqrt{q}$ (an even number). The multiplicity of $-\sqrt{q}$ is even (the coefficient of $T^{2g}$ is $q^g$, by the functional equation). Index the roots of $P$ so that $\alpha_{2g-j+1} = \bar{\alpha}_j = q/\alpha_j$, $j = 1, \ldots, g$. Now $\alpha_j \mapsto q/\alpha_j$ exchanges $\alpha_1, \ldots, \alpha_g$ and $\alpha_{2g}, \ldots, \alpha_{g+1}$. If we set

$$f(T) = \prod_{j=1}^{2g}(T - \alpha_j) = c_0 T^{2g} + c_1 T^{2g-1} + \cdots + c_{2g-1} T + c_{2g},$$

then $T^{2g} f(q/T)$ has the same roots as $f(T)$ and therefore $T^{2g} f(q/T) = c_{2g} f(T) = q^g f(T)$. Now the claim follows by equating the coefficients of $T^{g+l}$ on both sides: on the right we get $q^g c_{g-l}$ and on the left $q^{g-l} c_{g+l}$. □

*Input*: $\nu_1, \ldots, \nu_g$ and $r > g$.

*Output*: $\nu_{g+1}, \ldots, \nu_r$.

- For $j = 1, \ldots, g$, set $S_j = q^j + 1 - \nu_j$.
- For $j = 1, \ldots, g$,
$$c_j = -(S_j + c_1 S_{j-1} + \cdots + c_{j-1} S_1)/j.$$
- For $j = g+1, \ldots, \min(r, 2g)$, set $c_j = q^{j-g} c_{2g-j}$, get
$$S_j = -(c_1 S_{j-1} + \cdots + c_{j-1} S_1 + j c_j),$$
and set $\nu_j = q^j + 1 - S_j$.
- If $r > 2g$, proceed as in the basic algorithm: for $j = 2g+1, \ldots, r$,
$$S_j = -(c_1 S_{j-1} + \cdots + c_{2g} S_{j-2g})$$
and set $\nu_j = q^j + 1 - S_j$.

The parameter X of the function XN denotes the list $[\nu_1, \ldots, \nu_g]$.

```
def XN(q,X,r):
    g = len(X)
    if r<=g: return X[:r]
    X = [0]+X   # trick so that X[j] refers to GF(q^j)
    X = [x>>Q_ for x in X] # Q_: rational field
    S = [q**(j)+1-X[j] for j in range(1,g+1)]
    S = [0]+S   # similar trick
    # Computation of c1,...,cg; set c0=1
    c = [1>>Q_] # Computations in Q_
    for j in range(1,g+1):
        cj = S[j]
        for i in range(1,j):
            cj += c[i]*S[j-i]
        c += [-cj/j]
```

```
# Add c_{g+i}, for i=1,...,g
for i in range(1,g+1):
    c += [q**i*c[g-i]]
# Find Sj for j = g+1,...,r
for j in range(g+1,r+1):
    if j>2*g:
        Sj=0
    else:
        Sj = j*c[j]
    for i in range(1,j):
        if i>2*g: break
        Sj += c[i]*S[j-i]
    S += [-Sj]
# Find X[i] for i = g+1,...,r
for i in range(g+1,r+1): X += [q**i+1-S[i]]
return X[1:]
```

- $N_q(g)$: maximum of $\#X(\mathbb{F}_q)$ taken over all curves $X$ of genus $g$.
- *Hasse-Weil-Serre bound* (HWS): $N_q(g) \leqslant q + 1 + g\lfloor 2\sqrt{q} \rfloor$.
- $X$ of genus $g$ is *maximal* if $\#X(\mathbb{F}_q) = N_q(g)$.
- *Deuring algorithm*: Yields the list of all possible $\#E(\mathbb{F}_q)$ for elliptic curves $E/\mathbb{F}_q$.

```
 q  m
 2  2 [1, 2, 3, 4, 5]
 3  3 [1, 2, 3, 4, 5, 6, 7]
 4  4 [1, 2, 3, 4, 5, 6, 7, 8, 9]
 5  4 [2, 3, 4, 5, 6, 7, 8, 9, 10]
 7  5 [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
 8  5 [4, 5, 6, 8, 9, 10, 12, 13, 14]*
 9  6 [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
11  6 [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
```

Missing values in the HWS range for elliptic curves. Here $q$ is a prime power up to $5^3$, $m = \lfloor 2\sqrt{q} \rfloor$ and $d$ is the length of the Deuring list when less than $2m + 1$.

```
  q 2m+1 d
  8   11  9 [7, 11]
 16   17 13 [11, 15, 19, 23]
 25   21 20 [26]
 27   21 17 [22, 25, 31, 34]
 32   23 15 [23, 27, 29, 31, 35, 37, 39, 43]
 49   29 27 [43, 57]
 64   33 21 [51, 53, 55, 59, 61, 63, 67, 69, 71, 75, 77, 79]
 81   37 29 [67, 70, 76, 79, 85, 88, 94, 97]
125   45 37 [106, 111, 116, 121, 131, 136, 141, 146]
```

Over $\mathbb{F}_2 = \mathbf{Z}_2$ there are 32 cubic polynomials in *normal form*

$$E = y^2 + a_1xy + a_3 + x^3 + a_2x^2 + a_4x + a_6$$

of which precisely 16 are non-singular. For these cases, $g = 1$, the HWS bound is $q + 1 + m = 5$ (as $m = \lfloor 2\sqrt{2} \rfloor = 2$) and we have seen that all the integers in the HWS interval $[1, 5]$ occur as $\nu_1(E)$ for some $E$. Now a straighforward computation yields the following distribution:

| $\nu_1$ | $E$ |
|---|---|
| 1 | $y^2+y+x^3+x+1,\ y^2+y+x^3+x^2+1$ |
| 2 | $y^2+xy+x^3+x^2+1,\quad y^2+xy+x^3+x^2+x,$ |
| | $y^2+(x+1)y+x^3+1,\quad y^2+(x+1)y+x^3+x+1$ |
| 3 | $y^2+y+x^3,\quad y^2+y+x^3+1$ |
| | $y^2+y+x^3+x^2+x,\quad y^2+y+x^3+x^2+x+1$ |
| 4 | $y^2+xy+x^3+1,\quad y^2+xy+x^3+x$ |
| | $y^2+(x+1)y+x^3+x^2,\quad y^2+(x+1)y+x^3+x^2+x$ |
| 5 | $y^2+y+x^3+x,\quad y^2+y+x^3+x^2$ |

The sequences of values returned by XN with inputs $q=2$ and $[\nu_1]$, for $\nu_1=1,\ldots,5$, and $r=20$ are the following (the top row $S$ is the maximum value $N_q(1)$ of $\#E(\mathbb{F}_q)$ supplied by "Serre's procedure":

| $r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $S(2^r)$ | 5 | 9 | 14 | 25 | 44 | 81 | 150 | 289 | 558 | 1089 |
| $\nu_r$ | 1 | 5 | 13 | 25 | 41 | 65 | 113 | 225 | 481 | 1025 |
| | 2 | 8 | 14 | 16 | 22 | 56 | 142 | 288 | 518 | 968 |
| | 3 | 9 | 9 | 9 | 33 | 81 | 129 | 225 | 513 | 1089 |
| | 4 | 8 | 4 | 16 | 44 | 56 | 116 | 288 | 508 | 968 |
| | 5 | 5 | 5 | 25 | 25 | 65 | 145 | 225 | 545 | 1025 |

| $r$ | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| $S$ | 2139 | 4225 | 8374 | 16641 | 33131 | 66049 | 131797 | 263169 | 525737 | 1050625 |
| $\nu_r$ | 2113 | 4225 | 8321 | 16385 | 32513 | 65025 | 130561 | 262145 | 525313 | 1050625 |
| | 1982 | 4144 | 8374 | 16472 | 32494 | 65088 | 131174 | 263144 | 525086 | 1047376 |
| | 2049 | 3969 | 8193 | 16641 | 32769 | 65025 | 131073 | 263169 | 524289 | 1046529 |
| | 2116 | 4144 | 8012 | 16472 | 33044 | 65088 | 130972 | 263144 | 523492 | 1047376 |
| | 1985 | 4225 | 8065 | 16385 | 33025 | 65025 | 131585 | 262145 | 523265 | 1050625 |

Remark. $\mathrm{XN}(q, [\nu_1, \ldots, \nu_g], \infty) = \{\nu_j(X/\mathbb{F}_q)\}_{j \geqslant 1}$. Given a positive integer $s$, the subsequence $\{\nu_{sj}(X/\mathbb{F}_q)\}_{j \geqslant 1}$ is $\{\nu_j(X/\mathbb{F}_{q^s})\}_{j \geqslant 1}$ and therefore it must agree with $\mathrm{XN}(q^s, [\nu_s, \ldots, \nu_{sg}], \infty)$.

Summary. The tables above show that the elliptic curves $E_i$ $(i = 1, ..., 5)$ are maximal in 11 occasions over $\mathbb{F}_{2^r}$ in the range $r = 1, ..., 20$, and that they are close to the maximal value in the remaining cases:

- $E_1$ is maximal for $r = 4, 12, 20$, and is submaximal for $r = 19$.
- $E_2$ is maximal for $r = 3, 13$, and is submaximal for $r = 16$.
- $E_3$ is maximal for $r = 2, 6, 10, 14, 18$.
- $E_4$ is maximal for $r = 5$, and is submaximal for $r = 8, 11, 15, 16$ (the first and last tie with $E_2$).
- $E_5$ is maximal for $r = 1$, and is submaximal for $r = 7, 9, 16$.

The *Klein quartic* $C/\mathbb{F}_2$ ($g = 3$) is given by the equation

$$F(x, y, z) = x^3y + y^3z + z^3x. \tag{13}$$

In this case $\nu_1 = 3$, $\nu_2 = 5$, $\nu_3 = 24$.

Indeed, $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ are the only points of $C$ that satisfy $xyz = 0$ (the first two are at infinity). If $xyz \neq 0$, then we can look at the affine curve $C_z = x^3y + y^3 + x$. Over $\mathbb{F}_2$ it is clear that there are no more points, hence $\nu_1 = 3$. Over $\mathbb{F}_4$, there are two more points: $(\alpha, \alpha^2, 1)$ and $(\alpha^2, \alpha, 1)$, where $\alpha^2 = \alpha + 1$, and so $\nu_2 = 5$. To get $\nu_3$, let $\mathbb{F}_8$ be generated by $\beta$ with $\beta^3 = \beta + 1$. Since $y^3 = y^{10}$, on dividing $C_z$ by $y^3$ we get $(x/y^3)^3 + 1 + x/y^3 = 0$. Since $\xi^3 + \xi + 1 = 0$ has three solutions in $\mathbb{F}_8$ $(\beta, \beta^2, \beta^4)$, we conclude that $C_z$ has $7 \times 3 = 21$ poins other than $(0, 0)$ that are $\mathbb{F}_8$-rational and therefore $\nu_3 = 24$. With this, the values for $\nu_r$ supplied by XN (for $r \leqslant 12$) are the following:

| $r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\nu_r$ | 3 | 5 | 24 | 17 | 33 | 38 | 129 | 257 | 528 | 1025 | 2049 | 4238 |

Over $\mathbb{F}_5$, one finds that $\nu_1 = 6$, $\nu_2 = 26$ and $\nu_3 = 126$. With this, we get a similar table (for $r = 1, \ldots, 9$):

| $r$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $\nu_r$ | 6 | 26 | 126 | 626 | 3126 | 16376 | 78126 | 390626 | 1953126 |

Let $E/\mathbb{F}_q$ be an (plane) elliptic curve ($g = 1$), $O$ its improper point, $\Delta = mO$, with $0 < m < n$, and $\boldsymbol{P} = P_1, \ldots, P_n$ the list of $\mathbb{F}_q$-rational points of $E$. Then the Goppa code $\Gamma(\Delta, \Pi)$ has type $[n, m, d \geqslant n - m]$. So $k + d = n$ or $k + d = n + 1$(because of the Singleton bound).

Remark (Driencourt and Michon). If $E/\mathbb{F}_{2^m}$ is elliptic, and $\nu = m//2$, $\nu' = (m-3)//2$, then $1, x, \ldots, x^\nu, y, yx, \ldots, yx^{\nu'}$ is a basis of $L(mO)$.

- AG block ECC: implementing best known decoders, say [3].
- Standard convolutional ECC: implementing constructors and decoders (mainly Viterbi).
- Similarly for convolutional Goppa codes, after [5] and references therein.
- Concatenated codes (for example turbo codes), ...

- Complete and optimize the package.
- Fast multivar polynomials over $\mathbf{Z}_n$ would be a welcome addition (J. Tuitman, private communication, April 2017).
- Complete the set of jupyter notebooks.
- Upload the system to github.

# References I

[1] M. Bras and S. Xambó-Descamps.
OMEGA decoders for Goppa codes.
In A. Montes, editor, *Actas del Sexto Encuentro de Álgebra Computational y Aplicacicones*, page 14 pages, 2001.

[2] R. Farré, N. Sayols, and S. Xambó-Descamps.
On the PGZ decoding algorithm for alternant codes.
arXiv:1704.05259, 2017.

[3] K. Lee, M. Bras-Amorós, and M. E. O'Sullivan.
Unique decoding of general AG codes.
*IEEE Transactions on Information Theory*, 60(4):2038–2053, 2014.

# References II

[4] S. Molina, N. Sayols, and S. Xambó-Descamps.
A bootstrap for the number of $\mathbb{F}_{q^m}$-rational points on a curve over $\mathbb{F}_q$.
arXiv:1704.04661, 2017.

[5] G. Serrano.
Métodos de Geometría Algebraica en Teoría de Códigos Convolucionales, 20/1/2014.
Universidad de Salamanca.

[6] S. Xambó-Descamps.
*Block error-correcting codes: a computational primer*.
Univesitext. Springer, 2003.