Johannes-Kepler-Universität Linz, Altenbergerstraße 69, A-4040 Linz

Research Institute for Symbolic Computation

Diplomarbeit

Algebraic Geometric Codes

Diplomarbeit zur Erlangung des
akademischen Grades eines
Diplomingenieurs
der Studienrichtung Technische Mathematik
(Stzw. Mathematische Computerwissenschaften)

Verfasser : Harald Deischinger
Betreuer : Univ.Prof. Dr. Franz Winkler

Kleinzell, den 3. August 1999

Eidesstattliche Erklärung

Ich, Harald Deischinger, erkläre hier an Eides statt, daß ich die vorliegende Arbeit zur Gänze selbst verfaßt und keine anderen Quellen als die angegebenen verwendet habe.

Kleinzell, August 1999

Contents

As long as algebra and geometry proceeded along separate paths, their advance was slow and their applications limited.
But when these sciences joined company they draw from each other fresh vitality and thenceforward marched on at a rapid pace towards perfection.

J. L. Lagrange

Preface

Not only algebra and geometry drew fresh vitality from each other. Also other former separated fields of mathematics sometimes join to produce new and exciting results. In this work we show some of the results, that came forth from the combination of algebraic geometry and coding theory.

In the first two chapters we summarize the necessary concepts from algebraic geometry and coding theory. The main results in the first chapter is the Riemann-Roch theorem - it is our most important tool for the analysis of algebraic geometric codes.

In Chapter 3 we show, how the methods from algebraic geometry can be applied to construct valuable codes. Algorithms for decoding algebraic geometric codes are presented in Chapter 4.

Finally we present a tutorial implementation, which allows the user to construct algebraic geometric codes, and to experiment with the decoding algorithms.

Acknowledgments

I would like to thank by supervisor Franz Winkler for the support while writing this work. I also would like to thank Michaela Hackl for her valuable help.

Kleinzell, August 1999 Harald Deischinger

Geometry enlightens the intellect and sets one's mind right. All of its proofs are very clear and orderly. It is hardly possible for errors to enter into geometrical reasoning, because it is well arranged and orderly. Thus, the mind that constantly applies itself to geometry is not likely to fall into error. In this convenient way, the person who knows geometry acquires intelligence.

I. Khaldun

Chapter 1

Algebraic Geometry

In this chapter we summarize some definitions and facts from algebraic geometry to fix notation and for easier reference. Proofs of the theorems presented here may be found in [Mor91, Ful69, Sti93, vdW73].

Notation. Throughout this chapter let $\mathbb{F}$ be a field. By $\overline{\mathbb{F}}$ we denote the algebraic closure of $\mathbb{F}$.

1.1 Projective Space

Definition 1.1. *The n-dimensional* affine space *over $\mathbb{F}$ is defined as*

$$\mathbb{A}^n(\mathbb{F}) := \{(a_1, \ldots, a_n) \mid a_i \in \mathbb{F}\}$$

If the field $\mathbb{F}$ is clear from context, we let $\mathbb{A}^n := \mathbb{A}^n(\mathbb{F})$ and $\overline{\mathbb{A}^n} := \mathbb{A}^n(\overline{\mathbb{F}})$.

The elements of $\mathbb{A}^n$ are called points. *$\mathbb{A}^1$ is called the* affine line, *and $\mathbb{A}^2$ is called the* affine plane.

In affine space we often have the problem that certain relations hold only up to some exceptions. So, for instance, two different lines meet in a point, except if they are parallel. Many of those exceptions can be eliminated by moving from affine to projective space.

Definition 1.2. *The n-dimensional* projective space *over $\mathbb{F}$ is defined as*

$$\mathbb{P}^n(\mathbb{F}) := (\mathbb{F}^{n+1} \setminus \{0\})_{/\sim}$$

where

$$(c_1, \dots, c_{n+1}) \sim (d_1, \dots, d_{n+1}) \Leftrightarrow \\ \exists \lambda \in \mathbb{F} \setminus \{0\} : \lambda(c_1, \dots, c_{n+1}) = (d_1, \dots, d_{n+1})$$

If the field $\mathbb{F}$ is clear from context, we let $\mathbb{P}^n := \mathbb{P}^n(\mathbb{F})$ and $\overline{\mathbb{P}^n} := \mathbb{P}^n(\overline{\mathbb{F}})$.

The elements of $\mathbb{P}^n(\mathbb{F})$ are called points. *A representative of a point $P \in \mathbb{P}^n(\mathbb{F})$ denoted by $(c_1 : \dots : c_{n+1})$ is called* homogeneous coordinates *of P. $\mathbb{P}^1$ is called the* projective line, *and $\mathbb{P}^2$ is called the* projective plane.

Note that homogeneous coordinates are determined uniquely up to multiplication by a non-zero constant. By this definition the elements of $\mathbb{P}^n$ correspond to lines through the origin in $\mathbb{A}^{n+1}$.

Definition 1.3. *The set*

$$H_\infty := \{(c_1 : \dots : c_n : 0) \in \mathbb{P}^n\}$$

is called the hyperplane at infinity *(w.r.t. x_{n+1}). The points on H_∞ are called* points at infinity.

Using the bijections

$$\begin{array}{rccc} \rho_1 : & \mathbb{A}^n & \to & \mathbb{P}^n \setminus H_\infty \\ & (a_1, \dots, a_n) & \mapsto & (a_1 : \dots : a_n : 1) \end{array}$$

$$\begin{array}{rccc} \rho_2 : & \mathbb{P}^{n-1} & \to & H_\infty \\ & (c_1 : \dots : c_n) & \mapsto & (c_1 : \dots : c_n : 0) \end{array}$$

$\mathbb{P}^n \setminus H_\infty$ can be identified with $\mathbb{A}^n$ and H_∞ can be identified with $\mathbb{P}^{n-1}$. So we get

$$\mathbb{P}^n \sim \mathbb{A}^n \cup \mathbb{P}^{n-1}.$$

That is $\mathbb{P}^n$ can be viewed as an n-dimensional affine space plus points at infinity, corresponding to the directions in n-dimensional space. Because of this, points in $\mathbb{P}^n \setminus H_\infty$ are called *affine points.*

1.2 Algebraic Sets

Definition 1.4. *A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is called* homogeneous *or a* form *iff all monomials occuring in f have the same degree.*

Definition 1.5. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be of degree d. Then*

$$f^* := x_{n+1}^d f\left(\frac{x_1}{x_{n+1}}, \ldots, \frac{x_n}{x_{n+1}}\right) \quad \in \mathbb{F}[x_1, \ldots, x_{n+1}]$$

is a form of degree d, and it is called the homogenization *of f w.r.t. x_{n+1}.*

Conversely, if f is a form in $\mathbb{F}[x_1, \ldots, x_{n+1}]$ then

$$f_* := f(x_1, \ldots, x_n, 1) \quad \in \mathbb{F}[x_1, \ldots, x_n]$$

is the dehomogenization *of f w.r.t. x_{n+1}.*

Example 1.6.

$$(x^3 + y^2 + y)^* = x^3 + y^2 z + y z^2$$
$$(x^3 + y^2 z + y z^2)_* = x^3 + y^2 + y$$

Definition 1.7. *Let $f \in \mathbb{F}[x_1, \ldots, x_{n+1}]$ be a polynomial and $P \in \overline{\mathbb{P}^n}$. P is called a* root of f *iff $f(c_1, \ldots, c_{n+1}) = 0$ for any choice of homogeneous coordinates $(c_1, \ldots, c_{n+1})$ of P. This is denoted by $f(P) = 0$.*

Lemma 1.8. *Let $f \in \mathbb{F}[x_1, \ldots, x_{n+1}]$, $|\mathbb{F}| > \deg(f)$, $f_0, \ldots, f_d$ forms with $\deg(f_i) = i$ such that $f = \sum_{i=0}^{d} f_i$. P is a root of f iff P is a root for f_i for all $0 \leq i \leq d$.*

Definition 1.9. *For $S \subseteq \mathbb{F}[x_1, \ldots, x_{n+1}]$ a set of forms, we define the* projective algebraic set *defined by S as*

$$V_{\mathbb{F}}(S) := \{P \in \overline{\mathbb{P}^n} \mid \forall f \in S : f(P) = 0\}$$

We omit the $\mathbb{F}$ and write $V(S)$, if the field is clear from context.

A projective algebraic set is called irreducible *iff it cannot be written as the union of two proper algebraic sets. An irreducible algebraic set is called a* variety.

A projective algebraic set of dimension[1] *1 is called a curve.*

A point P of the algebraic set is called an $\mathbb{F}$-rational point *iff $P \in \mathbb{P}^n(\mathbb{F})$.*

[1] We do not go into details here, for the exact definition of the dimension of an algebraic set, see for example [vdW73]

Definition 1.10. *Let $\{f_1, \ldots, f_{n-1}\} \subseteq \mathbb{F}[x_1, \ldots, x_{n+1}]$ be a set of forms, and let $C = V(f_1, \ldots, f_{n-1}) \subseteq \overline{\mathbb{P}^n}$ be a curve. Furthermore let $P \in \overline{\mathbb{P}^n}$ be a point on the curve C.*

Then consider the Jacobian matrix of C

$$J_C = \frac{\partial C}{\partial(x_1, \ldots, x_{n+1})} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n+1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_{n-1}}{\partial x_1} & \cdots & \frac{\partial f_{n-1}}{\partial x_{n+1}} \end{pmatrix}.$$

C is called regular *at P iff $J_C(P)$ has full rank, otherwise P is called a* singular point *of C.*

The curve C is called regular *or* smooth *iff C is regular for all points $P \in \overline{\mathbb{P}^n}$ on the curve.*

For plane curves, that is curves defined over $\overline{\mathbb{P}^2}$ by a single form f, a point P is regular, if at least one of the three partial derivatives $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial z}$ is non-zero at P.

1.3 Plane Curves

For the purpose of coding theory we use a special class of algebraic sets, namely curves. Although curves in higher dimensions play a role in constructing long codes, in this introductory treatment we restrict out attention to plane curves.

A plane curve, in the sense of Definition 1.10, is determined by a square free form $f \in \mathbb{F}[x, y, z]$. This f is determined uniquely up to multiplication by a non-zero constant. Now we are also interested in multiple components of a curve, so we give a more general definition of plane curves.

Definition 1.11. *Let $f, g \in \mathbb{F}[x, y, z]$ be forms. We define the equivalence relation $\sim$ as: $f \sim g :\Leftrightarrow \exists \lambda \in \mathbb{F} \setminus \{0\} : f = \lambda g$.*

Then we define a plane curve *as an equivalence class of non-constant forms under the equivalence relation $\sim$.*

The degree *of the curve is the degree of the defining polynomial of the curve. A curve of degree 1 is called a line.*

If $f = \prod_i f_i^{e_i}$, where f_i are the irreducible factors of f, then f_i are called the components *of f and e_i the multiplicity of the component f_i.*

From $V(f)$ we can determine the components of f, but not the multiplicities of these components.

The following theorem gives us a bound on the number of $\mathbb{F}_q$-rational points a plane curve can have. This is of importance, because for algebraic geometric codes we are interested in curves having many rational points. In Section 1.8 we study a class of curves actually achieving this bound with equality.

Theorem 1.12 (Hasse-Weil Bound). *Let C be a smooth plane curve defined over a finite field $\mathbb{F}_q$ and let m be the degree of C. Then the number of $\mathbb{F}_q$-rational points n satisfies*

$$|n-(q+1)| \leq (m-1)(m-2)\sqrt{q}$$

Proof. A proof of this bound may be found in [Sti93]. Further improvements of this bound may be found in [Sør91]. □

1.3.1 Multiple Points and Tangents

Let C be a plane curve, defined by the form $f \in \mathbb{F}[x,y,z]$, and let P be a point on the curve. If $P=(a:b:c)$ is a regular point of f, then we can do a Taylor expansion of f around P, as

$$f(x,y,z) \approx \frac{\partial f}{\partial x}(P)\cdot(x-a)+\frac{\partial f}{\partial y}(P)\cdot(y-b)+\frac{\partial f}{\partial z}(P)\cdot(z-c).$$

This best-approximating line is called the tangent to the curve at P. Also if P is a singular point of f, we still can do a Taylor expansion, and ask for the terms of lowest degree occuring.

Definition 1.13. *Let C be a plane curve defined by the form $f \in \mathbb{F}[x,y,z]$ of degree d and P be a point. Let m be an integer, s.t. all partial derivatives of f of order less than m vanish at P, and at least one of the partial derivatives of order m does not vanish, that is*

$$\begin{aligned} \forall\, 0 \leq i+j+k < m: \quad & \frac{\partial^{i+j+k} f}{\partial x^i y^j z^k}(P) = 0 \\ \exists\, i+j+k = m: \quad & \frac{\partial^{i+j+k} f}{\partial x^i y^j z^k}(P) \neq 0 \end{aligned}$$

Then m is called the multiplicity *of P, written as $m_P(f)$.*

Remark. With this we distinguish 3 cases for $m_P(f)$:
If $m_P(f)=0$ then P is not a point of the curve f.
If $m_P(f)=1$ then P is a regular point on the curve.
If $m_P(f)>1$ then P is a singular point of f. $m_P(f)$ is then called the order of the singularity.

Theorem 1.14. *Let $f \in \mathbb{F}[x, y, z]$ be a form, and $P = (a : b : 1)$ be a point. Then $m_{(0:0:1)}(f) = m_P(f(x + a, y + b, z))$.*

If P is the origin, we can read off the multiplicity directly from the form f, we just have to find the terms of lowest degree occuring. By the theorem above, we can always perform a shift of coordinates to achieve this situation.

Theorem 1.15. *Let $f \in \mathbb{F}[x, y, z]$ be a form of degree d and let $P = (0 : 0 : 1)$. Write f_* as a sum of forms of increasing degree, that is $f_* = f_m + \cdots + f_d$, where f_i are forms of degree i respectively, and $f_m \neq 0$.*

Then $m = m_P(f)$.

For a regular point the derivatives define the tangent at P, but also if $m_P(f) > 1$, we can define tangents at a point P.

Definition 1.16. *Let $f \in \mathbb{F}[x, y, z]$ be a form and let $P = (0 : 0 : 1)$. Let f_m be the form of lowest degree occuring in f_*, then the homogenization of the linear factors of f_m (f_m considered as an element of $\overline{\mathbb{F}}[x, y]$) are called* tangents *of f at P.*

For $P = (a : b : 1)$, let L_i be the tangents of $f(x+a, y+b, z)$ at the origin. Then $L_i(x - a, y - b, z)$ are the tangents of f at P.

This definition of the tangents coincides with the intuitive notion of a tangent, as the following example demonstrates.

Example 1.17. Consider the tacnode curve defined over $\mathbb{C}$ by the form

$$f = 2x^4 - 3x^2yz + y^2z^2 - 2y^3z + y^4.$$

In order to compute the tangents at the origin $O = (0 : 0 : 1)$ we dehomogenize f and write it as a sum of forms, that is

$$f_* = \underbrace{x^4 + y^4}_{=f_4} + \underbrace{-3x^2y - 2y^3}_{=f_3} + \underbrace{y^2}_{=f_2}.$$

That is $m_O(f) = 2$, with a double tangent $L_1 = L_2 = y$.

We now also want to compute the tangents at the point $P = (0 : 1 : 1)$. We apply a shift of coordinates, dehomogenize and collect the terms of lowest order.

$$f'_* = \underbrace{y^2 - 3x^2}_{=f_2} - 3x^2y + 2y^2 + 2x^4 + y^4$$

That is $m_P(f) = 2$, with the two tangents $L_{1,2} = -3x \pm \sqrt{3}(y - z)$.

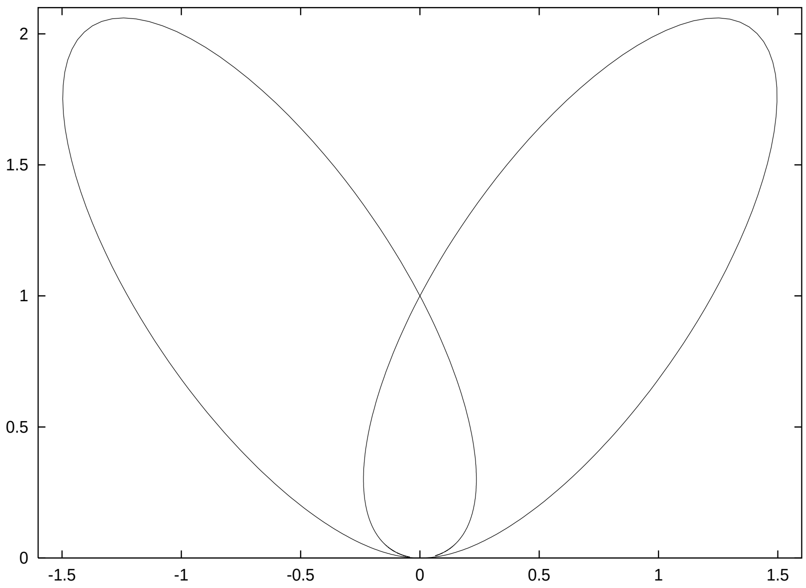


Figure 1.1: Tacnode curve, plotted over $\mathbb{A}^2(\mathbb{R})$.

1.3.2 Intersection Multiplicity

Definition 1.18. *Let now $f', g' \in \mathbb{F}[x, y, z]$ be forms, P an affine point.*

We define the intersection multiplicity *of f and g at P, by listing 7 properties on the dehomogenized polynomials $f = f'_*$, $g = g'_*$. The intersection multiplicity is denoted by $m_P(f \cap g) = m_P(f' \cap g')$.*

1. *$m_P(f \cap g) \in \mathbb{N}_0$ if P is not a point on a common component of f and g. Otherwise we let $m_P(f \cap g) = \infty$.*
2. *$m_P(f \cap g) = 0$ iff $P \notin f \cap g$. Furthermore, $m_P(f \cap g)$ depends only on those components of f and g containing P.*
3. *If T is a change of coordinates and $T(Q) = P$, then $m_Q(f^T \cap g^T) = m_P(f \cap g)$*
4. *$m_P(f \cap g) = m_P(g \cap f)$*
5. *$m_P(f \cap g) \geq m_P(f) \cdot m_P(g)$. Equality holds if and only if f and g do not have a common tangent at P.*
6. *Additivity on the unions of curves: If $f = \prod_i f_i^{r_i}$ and $g = \prod_j g_j^{s_j}$, then*
$$m_P(f \cap g) = \sum_{i,j} r_i s_j \cdot m_P(f_i \cap g_j).$$

7. $m_P(f \cap g) = m_P(f \cap (g + af))$ *for arbitrary* $a \in K[x, y]$.

Remark. Based on these requirements the intersection multiplicity is uniquely defined. (see [Ful69])

We use the intersection multiplicity in the following way: We think of g as a function modulo a curve f. Then we use the intersection multiplicity to define the order of a zero of g (see Definition 1.26). With this the last property becomes less mysterious. It just states, that this order should not depend on the actual representation of g.

Example 1.19. We determine the intersection multiplicity at the origin $O = (0, 0)$ of the curves e and f, defined by

$$\begin{aligned} e(x, y) &= (x^2 + y^2)^2 + 3x^2y - y^3, \\ f(x, y) &= (x^2 + y^2)^3 - 4x^2y^2. \end{aligned}$$

We replace f by the following curve:

$$f(x, y) - (x^2 + y^2) \cdot e(x, y) = y \cdot \underbrace{((x^2 + y^2)(y^2 - 3x^2) - 4x^2y)}_{=:g(x,y)}$$

Using property 7 we have

$$m_O(e \cap f) = m_O(e \cap y) + m_O(e \cap g)$$

We replace g by

$$g + 3e = y \cdot \underbrace{(5x^2 - 3y^2 + 4y^3 + 4x^2y)}_{=:h(x,y)}$$

So

$$\begin{aligned} m_O(e \cap f) &= 2\ m_O(e \cap y) + m_O(e \cap h). \\ m_O(e \cap y) &\underset{4,7}{=} m_O(x^4 \cap y) \underset{6,5}{=} 4 \\ m_O(e \cap h) &\underset{5}{=} m_O(e) \cdot m_O(h) = 6. \end{aligned}$$

Thus, $m_O(e \cap f) = 14$.

Algorithm 1.20. Based on the seven properties listed in Definition 1.18 we can describe a method for determining the intersection multiplicity $m_P(f \cap g)$.

Base cases: By property (3) we can assume that $P = (0, 0)$.

If $\gcd(f, g)(P) = 0$, set $m_P(f \cap g) = \infty$ by property (1).

Check if $P \in f \cap g$. If not set $m_P(f \cap g) = 0$ by property (2).

Let $f(x,0), g(x,0) \in \mathbb{F}[x]$ be of degrees r, s respectively. Because of (4) we can assume that $r \leq s$.

Case 1. $r = 0$: In this case $f = y \cdot h$ for some $h \in \mathbb{F}[x,y]$, because of (6)

$$m_P(f \cap g) = m_P(y \cap g) + m_P(h \cap g).$$

Let m be such that $g(x,0) = x^m(a_0 + a_1 x + \dots), a_0 \neq 0$. Then

$$m_P(y \cap g) \underset{(7)}{=} m_P(y \cap g(x,0)) \underset{(2)}{=} m_P(y \cap x^m) \underset{(6)}{=} m \cdot m_P(y \cap x) \underset{(5)}{=} m$$

Case 2. $r > 0$: We can assume that $f(x,0)$, $g(x,0)$ are monic. Otherwise multiply by suitable constants. Then by property (7)

$$m_P(f \cap g) = m_P(f \cap \underbrace{g - x^{s-r} \cdot f}_{=:h}) \quad \text{and } \deg(h(x,0)) = t < s.$$

Continue this process (we interchange f and h if $t < r$) until a pair of curves is reached, which can be handled by case 1.

Theorem 1.21 (Bézout Theorem). *Let f and g be curves of degree m and n respectively, with* $\gcd(f,g) = 1$. *Then*

$$\sum_{P \in \overline{\mathbb{P}^2}} m_P(f \cap g) = mn.$$

A simple, but also very useful, consequence of the Bézout theorem is the following theorem.

Theorem 1.22. *Let $C \subseteq \overline{\mathbb{P}^2}$ be a regular plane curve. Then C is irreducible.*

Proof. Assume that C is not irreducible. Then $C = fg$ for some curves f and g. Let P be a point on f. Now compute the partial derivatives of C at P (the indices indicate the partial derivative taken).

$$\begin{aligned} C_x(P) &= f_x(P)g(P) + f(P)g_x(P) = f_x(P)g(P) \\ C_y(P) &= f_y(P)g(P) + f(P)g_y(P) = f_y(P)g(P) \\ C_z(P) &= f_z(P)g(P) + \underbrace{f(P)}_{=0}g_z(P) = f_z(P)g(P) \end{aligned}$$

As C is regular, not all three partial derivatives can be zero, therefore $g(P)$ must be non-zero. That is f and g have no points of intersection. But by the Bézout theorem (1.21) this cannot be. So the assumption must be wrong, and C is irreducible. $\square$

1.4 Rational Functions

Definition 1.23. *Let $S \subseteq \mathbb{F}[x_1, \ldots, x_{n+1}]$ be a set of forms and $V = V(S)$ be a variety in $\overline{\mathbb{P}^n}$. The homogeneous coordinate ring of V is defined as*

$$\Gamma(V) := \mathbb{F}[x_1, \ldots, x_{n+1}]/\langle S\rangle$$

The field of rational functions $K(V)$ *over the variety V is defined as those elements from the quotient field of $\Gamma(V)$, which can be written as $\frac{f}{g}$ with f and g forms of the same degree.*

Definition 1.24. *For a variety V, $P \in V$, the rational function $\varphi \in K(V)$ is* defined at P *iff φ can be written as $\varphi = \frac{f}{g}$, where f, g are forms of the same degree and $g(P) \neq 0$.*

If φ is defined at P, the value $\varphi(P)$ is defined as $\varphi(P) := \frac{f(c_1,\ldots,c_{n+1})}{g(c_1,\ldots,c_{n+1})}$ with $P = (c_1 : \cdots : c_{n+1})$.

Note that for f, g forms of degree d

$$\frac{f(\lambda c_1, \ldots, \lambda c_{n+1})}{g(\lambda c_1, \ldots, \lambda c_{n+1})} = \frac{\lambda^d \cdot f(c_1, \ldots, c_{n+1})}{\lambda^d \cdot g(c_1, \ldots, c_{n+1})} = \frac{f(c_1, \ldots, c_{n+1})}{g(c_1, \ldots, c_{n+1})}.$$

So this definition of the value of a rational function is independent from the choice of homogeneous coordinates for the point P.

Example 1.25. Consider the plane curve $H = V(x^3 + y^2z + yz^2)$ over $\mathbb{F}_4$.

Consider now the rational function $\varphi = \frac{y^3+y^2z}{x^3}$. We would like to compute $\varphi(P)$ for $P = (0 : 1 : 1)$. Using this representation of φ we get an expression of the form $\frac{0}{0}$. Using the defining polynomial of H we can represent φ as

$$\varphi = \frac{y^3 + y^2z}{x^3} = \frac{y \cdot (y^2z + yz^2)}{z \cdot x^3} = \frac{y \cdot (y^2z + yz^2)}{z \cdot (-y^2z - yz^2)} = \frac{y}{z}.$$

So $\varphi(P) = 1$.

Definition 1.26. *Let $\varphi = \frac{f}{g}$ be a rational function on a plane curve C, and let P be a point on the curve. Then the* order *of φ at P, denoted by $\nu_P(\varphi)$, is defined as*

$$\nu_P(\varphi) = m_P(C \cap f) - m_P(C \cap g)$$

P is called a pole if $\nu_P(\varphi) < 0$, and a zero if $\nu_P(\varphi) > 0$.

As a direct consequence of the Bézout theorem (1.21) we get the following theorem.

Theorem 1.27. *Any rational function φ has the same number of poles and zeros (counted with multiplicity), that is*

$$\sum_{P \in \overline{\mathbb{P}^2}} v_P(\varphi) = 0.$$

1.5 Divisors

For this section let $C \subseteq \overline{\mathbb{P}^n}$ be a smooth curve.

Definition 1.28. *A* divisor *of C is an element of the free Abelian group generated by the points of C.*

The group of divisors of C is denoted by $\mathrm{Div}(C)$, the group operation will be written additively. Thus any element D in $\mathrm{Div}(C)$ has the form

$$D = \sum_{P \in C} n_P(D) P$$

with $n_P(D) \in \mathbb{Z}$, and only finitely many $n_P(D) \neq 0$. If the curve C is clear from context it will be omitted.

Definition 1.29. *The* degree *of a divisor D is defined as*

$$\deg(D) := \sum_{P \in C} n_P(D).$$

Definition 1.30. *Let D be a divisor on C.*

The support *of D is defined as*

$$\mathrm{supp}(D) := \{P \in C \mid n_P(D) \neq 0\}$$

D is called $\mathbb{F}$-rational *iff* $\mathrm{supp}\, D \subseteq \mathbb{P}^n(\mathbb{F})$*, that is if the points of the divisor have coordinates in $\mathbb{F}$ only.*

Finally we define a partial ordering on $\mathrm{Div}(C)$ by comparing two divisors pointwise.

Definition 1.31. *Let D and E be divisors on C.*

$$D \leq E :\Leftrightarrow \forall P \in C : n_P(D) \leq n_P(E)$$

D is called effective *or* positive *iff* $0 \leq E$.

1.6 Divisors and Rational Functions — The Riemann-Roch Theorem

Now we link the notion of divisors with that of rational functions. For this section let C be a smooth plane curve. Furthermore, let the field $\mathbb{F}$ be finite, that is $\mathbb{F} = \mathbb{F}_q$ for some prime power q.

Definition 1.32. *Let $f \in \mathbb{F}_q[x, y, z]$ be a form. Then the* zero divisor *or* intersection cycle *of f, denoted by (f), is defined as*

$$(f) := \sum_{P \in \overline{\mathbb{P}^2}} m_P(C \cap f)P.$$

Let $\varphi = \left(\frac{f}{g}\right) \in K(C)$ be a rational function. Then the divisor of poles and zeros or principal divisor*, denoted by (φ), is defined as*

$$(\varphi) := (f) - (g) = \sum_{P \in \overline{\mathbb{P}^2}} \nu_P(\varphi)P.$$

Observe that by Theorem 1.27 the degree of a principal divisor is zero, that is $\deg((\varphi)) = 0$.

Conversely we can associate a vector space of rational functions with any divisor.

Definition and Theorem 1.33. *Let D be a divisor. The space $\mathcal{L}(D)$ is defined as*

$$\mathcal{L}(D) := \{\varphi \in K(C) \mid (\varphi) + D \geq 0\} \cup \{0\}$$

$\mathcal{L}(D)$ is a finite dimensional vector space over the field $\mathbb{F}_q$.

With this we define the dimension of a divisor as the dimension of this vector space, that is

$$\dim(D) := \dim(\mathcal{L}(D)).$$

The space $\mathcal{L}(D)$ can also be expressed without using divisors, namely as

$$\mathcal{L}(D) = \{\varphi \in K(C) \mid \forall P \in C : \nu_P(\varphi) \geq -n_P(D)\} \cup \{0\}.$$

That is the space $\mathcal{L}(D)$ consists of rational functions, satisfying certains constraints on the order of the poles and zeros, namely

$n_P(D) > 0$: φ can have a pole of order at most $n_P(D)$ at P.

$n_P(D) = 0$: φ has no pole at P.

$n_P(D) < 0$: φ must have a zero of order at least $-n_P(D)$ at P.

Now we state some simple, but useful properties of $\mathcal{L}(D)$.

Theorem 1.34. *Let D and D' be divisors. Then*

$$\begin{array}{lr} a) & D \leq D' \Rightarrow \mathcal{L}(D) \subseteq \mathcal{L}(D') \\ b) & \deg(D) < 0 \Rightarrow \mathcal{L}(D) = \{0\} \end{array}$$

Proof. a) trivial

b) Assume there exists an element $0 \neq \varphi \in \mathcal{L}(D)$. Then

$$0 \leq \deg(\underbrace{(\varphi) + D}_{\geq 0}) = \underbrace{\deg((\varphi))}_{=0} + \deg(D) < 0,$$

which is impossible.

□

Theorem 1.35. *A non-zero rational function $\varphi \in \mathcal{L}(D)$ has at most* $\deg(D)$ *zeros.*

Proof. Let $\varphi \in \mathcal{L}(D)$ be a rational function having zeros $P_1, \ldots, P_z$, where $z > \deg(D)$. Then $\varphi \in \mathcal{L}(D - (P_1 + \cdots + P_z))$. But $\deg(D) - z < 0$, so by Theorem 1.34 $\varphi = 0$. □

It is intuitively clear, that the dimension and the degree of a divisor are closely related.

Definition 1.36. *The* genus *of a curve C is defined by*

$$g := \max_{D \in \mathrm{Div}(C)} (\deg(D) - \dim(D)) + 1$$

Theorem 1.37 (Plücker Formula). *Let C be a smooth plane curve of degree m. Then the genus g of C is given by*

$$g = \frac{(m-1)(m-2)}{2}$$

Some authors define the genus of a curve using the Plücker formula, taking into account the orders of the singularities of the curve C. This approach is more natural, when one is interested in the singularities. Details of this may be found in [vdW73, Win96]. Of course this is an equivalent approach, but we are more interested in the relation between dimension and degree of a divisor.

Theorem 1.38 (Riemann-Roch Theorem). *Let D be a divisor on a curve of genus g. Then*

$$\dim(D) = \deg(D) + 1 - g + i(D)$$

where $i(D)$ is the index of speciality *of D. $i(D)$ satisfies the following properties*

$$0 \leq i(D) \leq g, \text{ and } i(D) = 0 \text{ for } \deg(D) > 2g - 2.$$

Several interpretations for $i(D)$ may be found in [Sti93].

1.7 A Basis of the Space $\mathcal{L}(D)$

Let C be a smooth plane curve of degree d, and let $D = \sum_P n_P P$ be a divisor on C. We now give a method for constructing the space $\mathcal{L}(D)$. Details may be found in [HI94], a brief description may also be found in [Gop88].

Definition 1.39. *A divisor D is called* very simple *iff $n_P(D) \in \{-1, 0, 1\}$ for all $P \in C$.*

The following lemma provides a basic tool for the construction of a basis of $\mathcal{L}(D)$.

Lemma 1.40 (Simplification Lemma). *Let D and E be divisors on C. There exists a rational function φ, such that $(\varphi) = D + A$, and A is a very simple divisor. Moreover, φ can be constructed such that A and $D + E$ have disjoint supports.*

Sketch of the proof. We construct φ as a product $\varphi = \varphi_1 \cdots \varphi_k$, where each φ_i removes one instance of a point from D. Let P_i be an enumeration of the points in $\operatorname{supp}(D)$, counted with multiplicity. φ_i is constructed such that it satisfies the following conditions:

a) φ_i is the quotient of linear forms L_i^+ and L_i^-.

b) (φ_i) is very simple

c) $\operatorname{supp}((\varphi_i)) \cap \operatorname{supp}(D) = P_i$, and $\operatorname{supp}((\varphi_i)) \cap \operatorname{supp}(E) = \emptyset$.

d) φ_i should not increase the multiplicities of some points already occuring in $D_i = D + (\prod_{j<i} \varphi_j)$.

The linear forms needed in this construction may be found by testing possible choices for φ_i.

Details of the construction, and how this testing can be done, may be found in [HI94]. $\square$

Algorithm 1.41. Let D be an effective divisor, that is $D \geq 0$.

1. Build a polynomial G such that $(G) = D + A$, where A is very simple and effective. G can be constructed using the simplification lemma. G will be the denominator of the basis.

2. Now construct the numerators. To do this, begin with the $\frac{(d+1)(d+3)}{2}$-dimensional space of forms of degree d in $\mathbb{F}_q[x, y, z]$.

From this space, select out the subspace $\mathcal{B}$ of polynomials F, where $F \geq A$. Since A is very simple, $(F) \geq A$ if and only if $m_P(F) \geq 1$ for all $P \in \operatorname{supp}(A)$. Hence is suffices to ensure that $F(P) = 0$ for all such points. This imposes $\deg(A)$ linear constraints.

3. Let $\{F_1, \ldots, F_k\}$ be a basis for $\mathcal{B}$. Then $\{F_1/G, \ldots, F_k/G\}$ is a basis of $\mathcal{L}(D)$.

It is important, that in Step 1 we construct a very simple divisor. Otherwise we would introduce additional constraints in Step 2, and we would not get a complete basis.

The disadvantage of the algorithm sketched above is, that the divisor A must be computed explicitly. The field extension in which all the points of A are rational may in fact have degree $\mathcal{O}(d!^{\deg(D)})$.

But one can overcome the high complexity of this algorithm, when representing divisors implicitly using socalled *Chow forms*.

Definition 1.42. *Let $D = \sum_{P \in C} n_P P$ be an intersection cycle on C. Then the* Chow form *or* associated form *of D is the polynomial $R \in \overline{\mathbb{F}_q}[u_x, u_y, u_z]$ where*

$$R(u) = \prod_{P=(X,Y,Z) \in \operatorname{supp}(D)} (Xu_x + Yu_y + Zu_z)^{n_P}.$$

Let G be a form, then we can compute the Chow form of an intersection cycle $D = (G)$ directly, without first computing the support of D. The Chow form $R(u)$ of D can be computed as

$$R(u) = \text{u-res}(F, C) = \operatorname{res}(F, C, u_x x + u_y y + u_z z).$$

A brief description, how to compute such a multivariate resultant may be found in [CS97].

Using this representation of intersection cycles the tests for intersection, as they are needed in the simplification lemma, are now reduced to simple gcd computations. For the construction of the numerators some additional work has to be done. Some details on this not given in [HI94] may be found in [GH91].

1.8 Hermitian Curves

In this section we apply the theorems collected before to a family of curves, which will be very useful later.

Definition 1.43. *Let m be a prime power. The plane curve defined by the form*

$$H_m := x^{m+1} + y^m z + y z^m$$

over the field $\mathbb{F}_q = \mathbb{F}_{m^2}$ is called a Hermitian curve.

Theorem 1.44. *H_m is smooth, of genus $g = \frac{m(m-1)}{2}$. The $\mathbb{F}_{m^2}$-rational points of H_m are given as a single point at infinity $Q = (0:1:0)$, and m^3 affine points $P_1, \ldots, P_{m^3}$.*

Proof. To find the singular points of H_m we need to solve the following system.

$$\begin{aligned}
H_m &= x^{m+1} + y^m z + y z^m = 0\\
\frac{\partial H_m}{\partial x} &= \underbrace{(m+1)}_{=1} x^m = 0\\
\frac{\partial H_m}{\partial y} &= \underbrace{m y^{m-1}}_{=0} z + z^m = 0\\
\frac{\partial H_m}{\partial z} &= y^m + \underbrace{m y z^{m-1}}_{=0} = 0
\end{aligned}$$

From this we can read off that $x = 0,\ y = 0,\ z = 0$ is the only solution. But that is not a point in projective space. That is H_m is smooth and its genus can be computed by the Plücker formula (1.37).

To get the affine $\mathbb{F}_q$-rational points of H_m we fix $x \in \mathbb{F}_{m^2}$ and dehomogenize w.r.t z. With that we have to solve the equation

$$\underbrace{y^m + y}_{:=L(y)} = \underbrace{-x^{m+1}}_{:=a}. \tag{1.1}$$

Notice that for finite fields (see [LN94] for details)

$$\begin{aligned}
x^{m+1} &= x^m \cdot x = \mathrm{Norm}_{\mathbb{F}_{m^2}/\mathbb{F}_m}(x)\\
y^m + y &= \mathrm{Trace}_{\mathbb{F}_{m^2}/\mathbb{F}_m}(y)
\end{aligned}$$

For F, K fields with $K \leq F$ both, $\mathrm{Norm}_{F/K}$ and $\mathrm{Trace}_{F/K}$, take values in K only and $\mathrm{Trace}_{F/K}$ is linear.

We can solve (1.1) using a linear system as described in [LN94] for L-polynomials. Let $\{1, \xi\}$ be a basis of $\mathbb{F}_{m^2}$ ($\mathbb{F}_{m^2}$ considered as a 2-dimensional vector space over $\mathbb{F}_m$). Evaluate $L(y)$ at the basis to get

$$\begin{aligned}
L(1) &= 1 + 1 = 2 + 0\xi\\
L(\xi) &= \xi^m + \xi =: v + 0\xi
\end{aligned}$$

With this we get all solutions of (1.1) in $\mathbb{F}_{m^2}$ as $c_1+c_2\xi$ with (c_1, c_2) a solution of the following linear system.

$$\begin{pmatrix} 2 & v \\ 0 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} a \\ 0 \end{pmatrix}$$

The m solutions of this system are given by

$$\begin{cases} (c_1, c_2) = (\frac{a-vc_2}{2}, c_2) & \text{for } 2 \neq 0, \quad c_2 \in \mathbb{F}_m \\ (c_1, c_2) = (c_1, \frac{a}{v}) & \text{for } 2 = 0, \quad c_1 \in \mathbb{F}_m \end{cases}$$

That is for each of the m^2 values for $x \in \mathbb{F}_{m^2}$ we get exactly m solution for $y \in \mathbb{F}_{m^2}$, that is in total m^3 affine points.

Finally for $z = 0$, H_m simplifies to $x^{m+1} = 0$, that is $(0 : 1 : 0)$ is the only point of H_m at infinity. □

Note that by the Hasse-Weil bound (1.12)H_m has the maximal possible number of $\mathbb{F}_{m^2}$-rational points for a curve of genus $\frac{m(m-1)}{2}$.

Theorem 1.45. *The vector space $\mathcal{L}(aQ)$ over the Hermitian curve H_m, with $Q = (0 : 1 : 0)$ the single point at infinity, is spanned by*

$$\left\{ \frac{x^i y^j}{z^{i+j}} \;\middle|\; im + j(m+1) \leq a, \; i \leq m \right\}.$$

Proof. Let $\varphi_{i,j} = \frac{x^i y^j}{z^{i+j}}$ for $im + j(m+1) \leq a, \; i \leq m$.

Obviously $\varphi_{i,j}$ can have a singularity at the point Q only. So we compute $\nu_Q(\varphi_{i,j}) = m_Q(H_m \cap x^i y^j) - m_Q(H_m \cap z^{i+j})$ (we dehomogenization w.r.t y).

$$\begin{aligned} & m_{(0,0)}(x^i \cap x^{m+1} + z + z^m) = \\ & \qquad = i \cdot m_{(0,0)}(x \cap x^{m+1} + z + z^m) = i \cdot m_{(0,0)}(x \cap z + z^m) = i \\ & m_{(0,0)}(z^{i+j} \cap x^{m+1} + z + z^m) = \\ & \qquad = (i+j) \cdot m_{(0,0)}(z \cap x^{m+1} + z + z^m) = \\ & \qquad = (i+j) \cdot m_{(0,0)}(z \cap x^{m+1}) = (i+j)(m+1) \end{aligned}$$

That is $\nu_Q(\varphi_{i,j}) = -im - j(m+1)$ and therefore $\varphi_{i,j} \in \mathcal{L}(aQ)$.

For $i \leq m$ the monomials $\varphi_{i,j} = \frac{x^i y^j}{z^{i+j}}$ are linearly independent, so we count how many there are. For simplicity we only show the case when $(m+1)|a$ and $a > 2g - 2$. Let $a = a'(m+1)$. Then for each $0 \leq i \leq m$:

$$0 \leq j \leq \left\lfloor \frac{a - im}{m+1} \right\rfloor = a' + \left\lfloor \frac{-im}{m+1} \right\rfloor = a' - i$$

So

$$|\{\varphi_{i,j}\}| = \sum_{i=0}^{m} a' - i + 1 = \underbrace{a'(m+1)}_{=a} \underbrace{- \frac{m(m+1)}{2} + m}_{=-g} + 1,$$

that is by the Riemann-Roch theorem (1.38) we already found enough functions to form a basis. □

1.9 Curves in n-dimensional Space

It is possible to extend the theory presented above also to n-dimensional space and to non-smooth curves. Such curves are actually needed in the construction of long algebraic geometric codes.

But we do not need such curves for the presentation of the general ideas and algorithms in algebraic geometric coding theory. Actually it would be more appropriate to use the theory of algebraic function fields for space curves.

Details for this may be found in [Pre98, Sti93, Mor91]

Errors using inadequate data are much less than those using no data at all.

C. Babbage

Chapter 2

Coding Theory

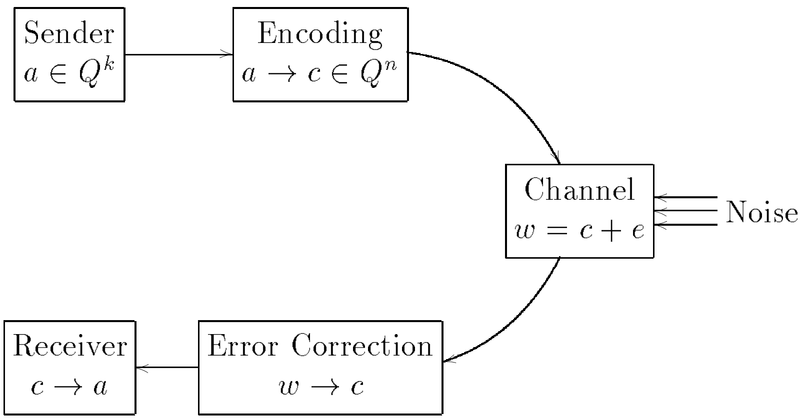
In this chapter we summarize some basic definitions and facts from coding theory to fix notation and for easier reference.

Details and proofs may be found in [vLvdG88, vL92, LP98].

Notation. Throughout this chapter let $\mathbb{F}_q$ be a finite field, for some prime power q.

2.1 Linear Codes

The basic problem one is faced with in coding theory is to transmit messages over a noisy channel. In order to do that, control symbols are added to the original message, so that errors during the transmission can be detected and possibly corrected by the receiver.



Definition 2.1. *Let Q be a non-empty set. A (block)* code C *over the alphabet Q is a non-empty subset of Q^n. Elements of Q^n are called* words *or* code words *if they lie in C.*

Definition 2.2. *Let C be a code over Q. The* Hamming distance $d(c, c')$ *is the number of positions by which two words $c, c' \in Q^n$ differ, that is*

$$d(c, c') := |\{0 < i \leq n \mid c_i \neq c'_i\}|$$

Using the Hamming distance we can formulate an important value, describing the quality of a code.

Definition 2.3. *Let C be a code over Q. The* minimal distance d *of the code C is the minimal Hamming distance of two distinct codewords, that is*

$$d = \min\{d(c, c') \mid c, c' \in C, c \neq c'\}$$

With a code of minimal distance d, we can detect up to $d - 1$ errors in a received word. But if d errors occured during the transmission, the word received might be a code word, different from the one sent out.

We restrict our attention now to a special class of codes only, namely linear codes.

Definition 2.4. *A* linear $[n, k, d]$-code C *over the finite field $\mathbb{F}_q$ is a subspace of $\mathbb{F}_q^n$, where k is the dimension of C and d is the minimal distance of the code.*

Computing the minimal distance of a code by comparing all pairs of code words can be very complicated. For linear codes it is sufficient to compare all code words with the zero-word.

Definition 2.5. *Let C be a (not necessarily linear) code over a finite field $\mathbb{F}_q$. The* weight $\mathrm{wt}(c)$ *of a word C is the distance of c to the zero word, that is*

$$\mathrm{wt}(c) := d(c, 0).$$

Theorem 2.6. *For a linear code C the minimal distance d equals the minimal weight of a non-zero code word, that is*

$$d = \min\{\mathrm{wt}(c) \mid c \in C, c \neq 0\}$$

Because a linear code is a k-dimensional subspace of $\mathbb{F}_q^n$, it can be represented as the image of a linear map, or as the null space of a linear map.

Definition 2.7. *Let C be a linear $[n,k,d]$-code. A $k \times n$ matrix G is called a* generator matrix *for C iff C is the image of G, that is*

$$C = \left\{ a \cdot G \mid a \in \mathbb{F}_q^k \right\}.$$

A $(n-k) \times n$ matrix H is called a parity check matrix *for C iff C is the null space of H, that is*

$$C = \left\{ c \in \mathbb{F}_q^n \;\middle|\; H \cdot c = 0 \right\}.$$

Often we also allow H to contain more rows than necessary by this definition.

Note that some authors use G^T as the generator matrix or H^T as the parity check matrix.

Definition 2.8. *Let C be a linear $[n,k,d]$-code. Then the linear $[n, n-k, d']$-code $C^\perp$ is called the* dual code *of C, where*

$$C^\perp := \left\{ c \in \mathbb{F}_q^n \;\middle|\; \forall c' \in C : c \cdot c' = 0 \right\}$$

where $c \cdot c' = \sum_{i=1}^n c_i \cdot c'_i$ is the scalar product in $\mathbb{F}_q^n$.

One should not think of $C^\perp$ as an orthogonal complement in the sense of vector spaces over $\mathbb{R}$. In the case of a finite field, the subspaces C and $C^\perp$ can have an intersection larger than $\{0\}$ and in fact they can even be equal. If $C = C^\perp$ then C is called a *self-dual* code.

It is not hard to see, that a generator matrix of C is a parity check matrix of $C^\perp$ and that a parity check matrix of C is a generator matrix of $C^\perp$.

Another important method to construct new codes out of known ones, is to consider only those code words having coordinates in a subfield.

Definition and Theorem 2.9. *Let C be a linear $[n,k,d]$-code over $\mathbb{F}_q^n$, and let K be a subfield of $\mathbb{F}_q$. Then the* subfield code $C|_K$ *is defined as*

$$C|_K := C \cap K^n$$

$C|_K$ is a linear $[n, k', d']$-code, where $k' \leq k$ and $d' \geq d$.

2.2 Encoding Linear Codes

For this section let C be a linear $[n,k,d]$-code, with generator matrix G.

From the definition of the generator matrix G we immediately get a method for encoding messages. A simple matrix multiplication does the job. To simplify this process it is appropriate to normalize the generator matrix.

Definition 2.10. *A generator matrix G is called a* canonical *generator matrix, iff*

$$G = \begin{pmatrix} 1 & 0 & \dots & 0 & g_{k+1,1} & \dots & g_{n,1} \\ 0 & 1 & \dots & 0 & g_{k+1,2} & \dots & g_{n,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & g_{k+1,k} & \dots & g_{n,k} \end{pmatrix}$$

If the matrix is in this form the message symbols make up the first part of the code word, followed by parity check symbols. Any encoder that gives code words of this form is called a *systematic* encoder.

Using matrix multiplication for encoding, the encoding takes $\mathcal{O}(k^2)$ arithmetic operations. Certain classes of codes might be encoded faster, for example cyclic codes might be encoded using $\mathcal{O}(n \log n)$ operations.

2.3 Decoding Linear Codes

For this section let C be a linear $[n, k, d]$-code, with generator matrix G and parity check matrix H.

For decoding, socalled *nearest neighbor decoding* is performed, that is, for a received word w the code word c is determined, for which the distance $d(c, w)$ is minimal. If several such code words exist, the decoding of c is not possible, we only know that an error occured.

Consider the spheres $S_r(c) := \left\{ w \in \mathbb{F}_q^n \;\middle|\; d(w, c) \leq r \right\}$ around the code words. If a received word w belongs to precisely one of the spheres $S_r(c)$ we decode w to the center of the sphere, that is to the code word c. By choosing r maximal, such that the spheres do not overlap, we get the following theorem.

Theorem 2.11. *A (not necessarily linear) code C with minimal distance d can correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors and detect up to $d - 1$ errors.*

For reasons of fairness, and for simplicity, words that do not lie in any of the spheres $S_r(c)$ are usually not decoded to a code word, even if a unique nearest neighbor would exist. Only the occurrence of an error is noted.

Linearly searching through all code words would be a method for decoding, of course not a very efficient one. So one of the problems in coding theory is to find efficient decoding algorithms. For some special codes very fast decoding algorithms are known. For example for Reed-Solomon codes the Berlekamp-Massey algorithm is available.

Once the code word is found it is easy to reconstruct the message. For systematic encoders the message can be read off immediately, otherwise the linear system $a \cdot G = c$ must be solved. Solving this linear system still can be time consuming. For cyclic codes, the code word can be recovered using $\mathcal{O}(n \log n)$ arithmetic operations.

2.4 Bounds on Linear Codes

For this section let C be a linear $[n,k,d]$-code over $\mathbb{F}_q$. Furthermore let $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ be the number of errors this code can correct.

Obviously we can not choose the parameters of a linear code completely free. If two parameters, for example n and k, are fixed, the third parameter can only vary in a certain interval. We now collect some bounds on n, k, and d.

Definition 2.12. *Let $A_q(n,d)$ be the maximal number of code words a code over $\mathbb{F}_q$ with length n and minimal distance d can have.*

This definition is useful both for linear and non-linear codes. For linear codes $\log_q A_q(n,d)$ is the maximal dimension a code over $\mathbb{F}_q$ of length n and minimal distance d can have.

Theorem 2.13 (Hamming Bound). *The parameters q, n, t and k satisfy the inequality*

$$q^k \left(1 + (q-1)\binom{n}{1} + \cdots + (q-1)^t \binom{n}{t}\right) \leq q^n$$

Definition 2.14. *A t-error correcting code over $\mathbb{F}_q$ is called* perfect *iff equality holds in Theorem 2.13.*

The Hamming bound states, that there cannot be more words in the spheres of radius t around the code words, than there are words in the whole space. So for perfect codes all words received can be decoded uniquely.

Theorem 2.15 (Gilbert-Varshamov Bound). *If*

$$q^{n-k} > \sum_{i=0}^{d-2} \binom{n-1}{i} (q-1)^i$$

then there exists a linear $[n,k,d']$-code with $d' \geq d$.

Theorem 2.16 (Singleton Bound).

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

Definition 2.17. *A code for which equality holds in Theorem 2.16 is called a* maximum distance separable *(MDS) code.*

The bounds presented so far give a relation between the three parameters n, k and d. Now one is also interested in the case when $n \to \infty$.

Definition 2.18. *Let C be a linear $[n, k, d]$-code.*
The information rate *R of a linear code is defined as $R = \frac{k}{n}$.*
The relative minimal distance *δ of a linear code is defined as $\delta = \frac{d}{n}$.*
Furthermore define for arbitrary codes

$$\alpha(\delta) := \limsup_{n\to\infty} \frac{\log_q A_q(n, \delta n)}{n}$$

For linear codes, $\alpha(\delta)$ is the maximal information rate a code of relative minimal distance δ can have.

Theorem 2.19 (Asymptotic Gilbert-Varshamov Bound). *Let $H_q(x)$ be the* entropy function *defined by*

$$H_q(x) := \begin{cases} x \log_q(q-1) - x \log_q x - (1-x)\log_q(1-x) & \text{for } 0 < x < \frac{q-1}{q} \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\alpha(\delta) \geq 1 - H_q(\delta)$$

Theorem 2.20 (Plotkin Bound).

$$\alpha(\delta) \leq \begin{cases} 1 - \frac{q\delta}{q-1} & \text{for } 0 < \delta < \frac{q-1}{q} \\ 0 & \text{otherwise} \end{cases}$$

The asymptotic Gilbert-Varshamov Bound gives us a lower bound for $\alpha(\delta)$, while the Plotkin bound states that $\alpha(\delta)$ can not be too high. But there is still a region inbetween, as can be seen in Figure 2.1. Also observe, that the region between the two bounds gets smaller as the field size increases.

2.5 Cyclic Codes

Definition 2.21. *A linear code C is called* cyclic *iff for all code words $c \in C$ the cyclic shift of c is again a code word. That is $(c_1, \ldots, c_n) \in C$ implies that $(c_n, c_1, \ldots, c_{n-1}) \in C$.*

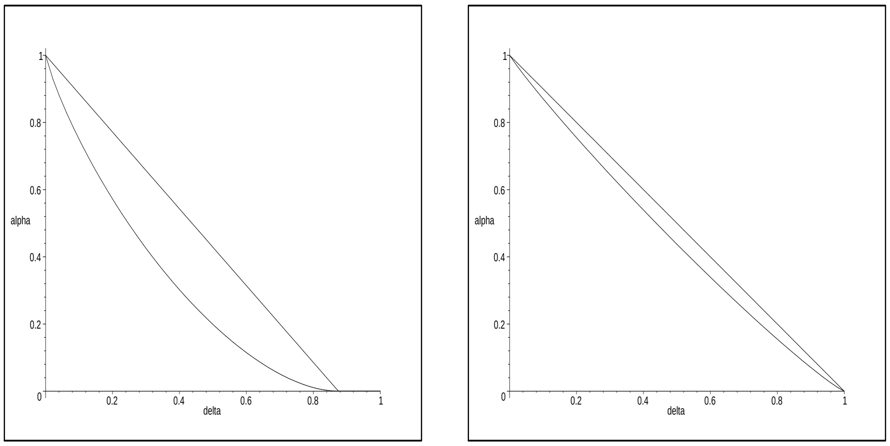


Figure 2.1: Gilbert-Varshamov and Plotkin Bound for $q = 8$ and $q = 65536$

For cyclic codes one identifies the vectors in $\mathbb{F}_q^n$ with the elements of $W_n := \mathbb{F}_q[x]/\langle x^n - 1\rangle$ using the bijection

$$\begin{array}{cccc} \tau : & \mathbb{F}_q^n & \to & W_n \\ & (c_0, \ldots, c_{n-1}) & \mapsto & c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \end{array}$$

The cyclic shift in $\mathbb{F}_q^n$ then corresponds to the multiplication by x in W_n. So one sees that a cyclic code C has the structure of an ideal in the ring W_n.

Theorem 2.22. *A linear code $C \subseteq W_n$ is cyclic iff C is an ideal in W_n.*

Theorem 2.23. *Let C be an ideal $\neq \{0\}$ of W_n. Then there exists a unique $g \in W_n$ with the following properties*

1. *$g | x^n - 1$ in $\mathbb{F}_q[x]$*
2. *$C = \langle g \rangle$*
3. *g is monic*

Hence every cyclic code C can be generated by a unique monic polynomial g. This g is called the generator polynomial *of C.*

The following theorem gives us a relation between the generator polynomial and the parity check matrix of a cyclic code.

Theorem 2.24. *Let C be a cyclic code with generator polynomial g,* $\deg(g) = n-k$*, and let $\alpha_1, \ldots, \alpha_{n-k}$ be the roots of g. Then $v \in W_n$ is a code polynomial iff the coefficient vector $(v_0, \ldots, v_{n-1})$ of v is in the null space of the matrix*

$$H = \begin{pmatrix} 1 & \alpha_1 & {\alpha_1}^2 & \ldots & {\alpha_1}^{n-1} \\ 1 & \alpha_2 & {\alpha_2}^2 & \ldots & {\alpha_2}^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_{n-k} & {\alpha_{n-k}}^2 & \ldots & {\alpha_{n-k}}^{n-1} \end{pmatrix}$$

We describe now an important class of cyclic codes which has been introduced by R. C. Bose, D. K. Ray-Chaudhuri and independently by A. Hocquenghem. The codes are known as BCH codes.

Definition and Theorem 2.25. *Let ξ be a primitive nth root of unity in $\mathbb{F}_{q^m}$, where m is the multiplicative order of q modulo n. Let $2 \leq d \leq n$.*

Then the BCH code *C of designed distance d is a cyclic code over $\mathbb{F}_q$ of length n. It is the subfield code $C := C'\big|_{\mathbb{F}_q}$ of the cyclic code C' defined by the parity check matrix*

$$H = \begin{pmatrix} 1 & \xi^c & \ldots & \xi^{c(n-1)} \\ 1 & \xi^{c+1} & \ldots & \xi^{(c+1)(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi^{c+d-2} & \ldots & \xi^{(c+d-2)(n-1)} \end{pmatrix}$$

For $c = 1$ the code is called a narrow sense BCH code. *If $n = q^m - 1$, the BCH code is called* primitive. *The dimension of C is $\geq n - m(d-1)$.*

For $n = q - 1$ and $c = 1$ we get a special class of BCH codes.

Definition and Theorem 2.26. *Let $k < q$, $n = q - 1$ and $d = q - k$. Furthermore let α be a primitive element in $\mathbb{F}_q$. The* Reed-Solomon (RS) code *$RS(k,q)$ is given by the parity check matrix*

$$H_{RS(k,q)} := \begin{pmatrix} 1 & \alpha & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & \ldots & \alpha^{2(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{d-1} & \ldots & \alpha^{(d-1)(n-1)} \end{pmatrix}$$

The code $RS(k,q)$ is a cyclic $[n,k,d]$-code.

The importance of BCH and RS codes is, that we can construct codes for any given minimal distance d. But in order to do that we also have to increase the length n of the code. In order to keep the complexity of encoding and decoding low one is interested in codes with n as small as possible.

Practical application is found by not looking for it, and one can say that the whole progress of civilization rests on that principle.

H. Hadamard

Chapter 3

Construction of Algebraic Geometric Codes

Around 1980 *V.D.Goppa* [Gop81] showed how methods from algebraic geometry can be used for the construction of linear codes — because of this, algebraic geometric codes are often refered to as Goppa codes.

Already in 1982 *Tsfasman, Vlăduţ and Zink* [TVZ82] used Goppa's construction to prove the existence of codes exceeding the Gilbert-Varshamov bound – which was believed to be the best possible bound for over 20 years. Also, several well known classes of codes, for example Reed-Solomon codes, can be seen as special cases of algebraic geometric codes.

Notation. From this chapter on let $\mathbb{F}_q$ be a finite field, for a prime power q, and let α be a primitive element of $\mathbb{F}_q$, i.e $\mathbb{F}_q = \{0\} \cup \{\alpha^i \mid 0 \leq i \leq q-2\}$.

For $f_1, \ldots, f_{n-1} \in \mathbb{F}_q[x_1, \ldots, x_{n+1}]$ forms, we let $F = V(f_1, \ldots, f_{n-1}) \subseteq \mathbb{P}^n(\overline{\mathbb{F}_q})$ be a smooth projective algebraic curve of genus g. Usually this will be a plane variety $F = V(f) \subset \overline{\mathbb{P}^2}$. All functions mentioned are rational functions on this fixed curve, that is elements of $K(F)$.

3.1 Functional Goppa Code

Definition 3.1 (Functional Goppa Code). *Let $D := \sum_{i=1}^{n} P_i$ be an $\mathbb{F}_q$-rational divisor and G be a divisor with support distinct from D. Then the* (geometric) functional Goppa code $C_{\mathcal{L}}(D, G)$ *is defined by evaluating the* *functions* $\varphi \in \mathcal{L}(G)$ *at the points of* D*:*

$$C_{\mathcal{L}}(D,G) := \{(\varphi(P_1),\ldots,\varphi(P_n)) \mid \varphi \in \mathcal{L}(G)\} \tag{3.1}$$

Theorem 3.2. *$C_{\mathcal{L}}(D,G)$ is a linear $[n,k,d]$-code. For $\deg(G) < \deg(D)$ the length n, the dimension k and the minimal distance d satisfy*

$$\begin{aligned} n &= \deg(D) \\ k &= \dim(G) = \deg(G) + 1 - g + i(G) \\ d &\geq d_\Gamma = n - \deg(G) \end{aligned}$$

d_Γ is called the designed distance *of the code.*

Proof. Consider the evaluation map

$$\begin{aligned} \mathrm{ev}_D : \mathcal{L}(G) &\to \mathbb{F}_q^n \\ \varphi &\mapsto (\varphi(P_1),\ldots,\varphi(P_n)) \end{aligned} \tag{3.2}$$

Because all points of D have coordinates only in $\mathbb{F}_q$, and D and G have distinct support, ev_D takes values only in $\mathbb{F}_q^n$. Furthermore it is easy to see that ev_D is linear.

So $C_{\mathcal{L}}(D,G) = \mathrm{ev}_D(\mathcal{L}(G))$ is a subspace of $\mathbb{F}_q^n$, that is it is a linear code of length n.

From linear algebra we know that its dimension can be computed as $k = \dim(\mathcal{L}(G)) - \dim(\ker(\mathrm{ev}_D))$. Since $\mathcal{L}(G-D)$ is the space of those functions $\varphi \in \mathcal{L}(G)$ which have at least a simple zero at all points in D,

$$\mathcal{L}(G-D) = \ker(\mathrm{ev}_D).$$

By assumption $\deg(G-D) = \deg(G) - n < 0$, so the kernel contains just the zero element (by Theorem 1.34). The dimension of $\mathcal{L}(G)$ is given by the Riemann-Roch theorem (1.38). So $k = \dim(\mathcal{L}(G)) = \deg(G) + 1 - g + i(G)$.

By Theorem 1.35 the number of zeros of a non-zero function in $\mathcal{L}(G)$ is at most $\deg(G)$. Therefore the number of zeros for a code word is also at most $\deg(G)$. So the minimal weight of a non-zero code word is at least $n - \deg(G)$, which is also the minimal distance d (by Theorem 2.6). ☐

Corollary 3.3. *If $2g-2 < \deg(G) < \deg(D)$ then the dimension of $C_{\mathcal{L}}(D,G)$ is given by*

$$k = \deg(G) + 1 - g.$$

In order to produce a generator matrix for this code we can fix a basis $\{\varphi_1, \ldots, \varphi_k\}$ of $\mathcal{L}(G)$, evaluate the basis functions at the points of D and collect the results as the rows of a matrix:

$$G := \begin{pmatrix} \varphi_1(P_1) & \ldots & \varphi_1(P_n) \\ \vdots & \ddots & \vdots \\ \varphi_k(P_1) & \ldots & \varphi_k(P_n) \end{pmatrix} \tag{3.3}$$

Example 3.4. Take now the Hermitian curve

$$F = V(x^3 + y^2 z + y z^2)$$

over the field $\mathbb{F}_4$. As this curve is smooth the genus is given by the Plücker formula (1.37) as $g = \frac{(3-1)(3-2)}{2} = 1$.

As we have seen in Section 1.8 this curve has 9 points over $\mathbb{F}_4$, which are

$$\begin{array}{lll} Q = (0:1:0) & & \\ P_1 = (0:0:1) & P_2 = (0:1:1) & P_3 = (1:\alpha:1), \\ P_4 = (1:\alpha^2:1) & P_5 = (\alpha:\alpha:1) & P_6 = (\alpha:\alpha^2:1) \\ P_7 = (\alpha^2:\alpha:1) & P_8 = (\alpha^2:\alpha^2:1) & \end{array}$$

Let D be the divisor of the sum of the 8 affine points, that is

$$D = P_1 + \cdots + P_8.$$

We consider now the code $C_{\mathcal{L}}(D,\ 4Q)$. The 4-dimensional space $\mathcal{L}(4Q)$ is spanned by the following basis functions. The numbers in parentheses indicate the order of pole at the point Q.

$$\varphi_1 = 1 \quad (0), \quad \varphi_2 = \frac{x}{z} \quad (2), \quad \varphi_3 = \frac{y}{z} \quad (3), \quad \varphi_4 = \frac{x^2}{z^2} \quad (4)$$

With this information we can already give a generator matrix for the code $C_{\mathcal{L}}(D,\ 4Q)$ by evaluating the functions $\varphi_1, \ldots, \varphi_4$ at the points $P_1, \ldots, P_8$:

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

By Theorem 3.2 we know that the minimal distance of this code is $d \geq n - \deg(4Q) = 8 - 4 = 4$. By comparing the second and fourth row of the generator matrix, we see that the minimal distance actually equals 4.

3.2 Primary Goppa Code

Definition 3.5 (Goppa code). *The dual code of the functional Goppa code* $C_\Omega(D,G) := C_\mathcal{L}(D,G)^\perp$ *is called the* (primary) Goppa code.

This code is also known as *residual Goppa code*, because the original construction used residuals on the space of differentials (denoted by Ω).

Definition 3.6. *Let* $c \in \mathbb{F}_q^n$*, and* φ *a rational function. The* syndrome of c w.r.t. φ*, denoted by* $c \cdot \varphi$*, is defined as*

$$c \cdot \varphi := \sum_{i=1}^{n} c_i \cdot \varphi(P_i).$$

Using this notation the code $C_\Omega(D,G)$ can be constructed without refering to the functional code.

$$\begin{aligned} C_\Omega(D,G) &= C_\mathcal{L}(D,G)^\perp \\ &= \left\{ (c_1,\dots,c_n) \in \mathbb{F}_q^n \mid \forall_{d \in C_\mathcal{L}(D,G)} c \cdot d = 0 \right\} \\ &= \left\{ (c_1,\dots,c_n) \in \mathbb{F}_q^n \mid \forall_{\varphi \in \mathcal{L}(G)} c \cdot \varphi = 0 \right\} \end{aligned}$$

Theorem 3.7. *The code* $C_\Omega(D,G)$ *is a linear* $[n,k,d]$*-code. For* $\deg(G) < \deg(D)$ *the length* n*, the dimension* k *and the minimal distance* d *satisfy*

$$\begin{aligned} n &= \deg(D) \\ k &= n - \dim(G) = n - \deg(G) - 1 + g - i(G) \\ d &\geq d_\Gamma = \deg(G) - 2g + 2 \end{aligned}$$

d_Γ *is called the* designed distance *or* Goppa bound *of the code.*

Proof. As $C_\Omega(D,G)$ is the dual code of $C_\mathcal{L}(D,G)$ the dimension can be determined immediately from the functional Goppa code.

To estimate the minimal distance a little bit more work is needed. We will show that the assumption $d < \deg(G) - 2g + 2$ leads to a contradiction.

Let c be a non-zero code word with minimal weight. W.l.o.G. let the first d positions be non-zero. That is $c_1, \dots, c_d \neq 0$ and $c_{d+1} = \dots = c_n = 0$.

With $T_j := \sum_{i=1}^{j} P_i$ we get a sequence of increasing spaces

$$\mathcal{L}(G - T_d) \leq \mathcal{L}(G - T_{d-1}) \leq \mathcal{L}(G)$$

Using the Riemann-Roch theorem (1.38) we show now that $\mathcal{L}(G-T_d) \neq \mathcal{L}(G-T_{d-1})$ by showing that their dimensions are not equal.

$$\dim(G-T_d) = \deg(G) - d + 1 - g + \underbrace{i(G-T_d)}_{=0}$$
$$\dim(G-T_{d-1}) = \deg(G) - d + 1 + 1 - g + \underbrace{i(G-T_{d-1})}_{=0}$$

In both cases the index of speciality is zero because in both cases the degree of the divisor is greater than $2g-2$ (Here the assumption on d is used):

$$\deg(G-T_d) \geq \deg(G) - d \qquad > \deg(G) - \deg(G) + 2g - 2 = 2g - 2$$
$$\deg(G-T_{d-1}) \geq \deg(G) - (d-1) > \deg(G) - \deg(G) + 2g - 2 = 2g - 2$$

Therefore $\mathcal{L}(G-T_d) \subsetneq \mathcal{L}(G-T_{d-1}) \leq \mathcal{L}(G)$. Take now $\varphi \in \mathcal{L}(G-T_{d-1}) \setminus \mathcal{L}(G-T_d)$. That is $\varphi(P_1) = \cdots = \varphi(P_{d-1}) = 0$ and $\varphi(P_d) \neq 0$.

By choice of c and φ:

$$c \cdot \varphi = \underbrace{\sum_{i=1}^{d-1} c_d \cdot \underbrace{\varphi(P_i)}_{=0}}_{=0} + \underbrace{c_d}_{\neq 0} \cdot \underbrace{\varphi(P_d)}_{\neq 0} + \underbrace{\sum_{i=d+1}^{n} \underbrace{c}_{=0} \cdot \varphi(P_i)}_{=0} \neq 0$$

But c is taken to be a code word and $\varphi \in \mathcal{L}(G)$ which means that $c \cdot \varphi = 0$. So the assumption that $d < m - 2g + 2$ must be wrong. □

Remark. This bound on the minimal distance of the code is not the best possible bound. For small values of $\deg(G)$ the *Feng-Rao bound* d_{FR} is superior to the Goppa bound stated in Theorem 3.7. The Feng-Rao bound is based on more subtle properties of the space $\mathcal{L}(G)$. Details may be found in [HP95].

Theorem 3.8. *For any primary Goppa code $C_\Omega(D,G)$ there exists a divisor W, s.t. $C_\Omega(D,G) = C_\mathcal{L}(D, D-G+W)$.*

Proof. To show this theorem the residue representation of Goppa codes is needed. A proof may be found in [Sti93]. □

This theorem states that there is actually only one class of Goppa codes.

Definition 3.9. *A Goppa code of the form $C_\Omega(D,G)$ or $C_\mathcal{L}(D,G)$ with $G = aQ$ for a single $\mathbb{F}_q$-rational point Q, $a \in \mathbb{N}$ is called a* one point Goppa code*.*

Many interesting examples of Goppa codes, e.g. Hermitian codes, are of this form.

3.3 Examples

3.3.1 Reed-Solomon Codes

We have already defined Reed-Solomon codes in Section 2.5. Now we are going to construct this class of codes with the methods presented above.

Let us consider the straight line $\mathbb{P}^1$, which can be identified with the curve $y = 0$ in the projective plane, so all the notions developed for plane curves can also be applied to the projective line.

The genus of this curve is 0, and it has exactly $q + 1$ $\mathbb{F}_q$-rational points: q affine points of the form $(x : 1)$ with $x \in \mathbb{F}_q$, and one point at infinity $Q := (1 : 0)$.

As the divisor D we use the affine points, except the origin, that is

$$D = \sum_{x \in \mathbb{F}_q \setminus \{0\}} (x : 1) = \sum_{i=0}^{q-2} (\alpha^i : 1).$$

With $G := (d-1)Q - (0 : 1)$ the space $\mathcal{L}(G)$ consists of those functions having a pole at infinity of order at most $d - 1$ and having a zero at the origin. That is $\mathcal{L}(G)$ consists of polynomials of degree at most $d - 1$, with no constant part. Therefore a basis for $\mathcal{L}(G)$ (written in affine form) is given by $\{x, x^2, ..., x^{d-1}\}$.

A parity check matrix of the code $C_\Omega(D, G)$ can be obtained by evaluating these basis functions at the points of the divisor D. Because $x^i(\alpha^j : 1) = (\alpha^j)^i = \alpha^{i \cdot j}$ this matrix takes the form:

$$H_{C_\Omega(D,G)} = \begin{pmatrix} 1 & \alpha & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \dots & \alpha^{2(n-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{d-1} & \dots & \alpha^{(d-1)(n-1)} \end{pmatrix}$$

This matrix equals the parity check matrix of the Reed-Solomon code $RS(k, q)$, therefore

$$RS(k, q) = C_\Omega \left(\sum_{i=0}^{q-2} (\alpha^i : 1),\ (d-1)Q - (0 : 1) \right)$$

where $d = q - k$.

3.3.2 BCH Codes

One can use basically the same construction as it was used for Reed-Solomon codes also to obtain BCH codes as subfield codes of Goppa codes. We take the divisor G as

$$G = (c + d - 2)(1 : 0) - c(0 : 1),$$

and the divisor D as

$$D = \sum_{i=0}^{n-1} (\xi^i : 1),$$

with q, n, d, c, ξ as in Section 2.5. Then the BCH code for the given parameters is given as $C_\Omega(D, G)\,\big|_{\mathbb{F}_q}$

3.3.3 Classical Goppa Codes

Some years before finding the algebraic geometric codes presented here, V.D. Goppa introduced a class of codes now known as classical Goppa codes. Those codes can be seen as a special case of primary Goppa codes.

Definition 3.10. *Let $g \in \mathbb{F}_{q^m}[x]$ be a polynomial of degree t. Let $L = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_{q^m}$, such that $|L| = n$ and $g(\alpha_i) \neq 0$ for $1 \leq i \leq n$, and $1 \leq t \leq n - 1$.*

Then the classical Goppa code $\Gamma(L, g)$ *with* Goppa polynomial $g(z)$ *is defined as the set of codewords $c = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$ for which*

$$\sum_{i=1}^{n} \frac{c_i}{x - \alpha_i} = 0 \mod g(x).$$

$\Gamma(L, g)$ is a linear code of length n.

Theorem 3.11. *The classical Goppa code $\Gamma(L, g)$ is the $\mathbb{F}_q$-subfield code of the code $\Gamma'(L, g)$ defined by the parity check matrix*

$$H := \begin{pmatrix} g(\alpha_1)^{-1} & g(\alpha_2)^{-1} & \ldots & g(\alpha_n)^{-1} \\ \alpha_1 \cdot g(\alpha_1)^{-1} & \alpha_2 \cdot g(\alpha_2)^{-1} & \ldots & \alpha_n \cdot g(\alpha_n)^{-1} \\ {\alpha_1}^2 \cdot g(\alpha_1)^{-1} & {\alpha_2}^2 \cdot g(\alpha_2)^{-1} & \ldots & {\alpha_n}^2 \cdot g(\alpha_n)^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ {\alpha_1}^{t-1} \cdot g(\alpha_1)^{-1} & {\alpha_2}^{t-1} \cdot g(\alpha_2)^{-1} & \ldots & {\alpha_n}^{t-1} \cdot g(\alpha_n)^{-1} \end{pmatrix} \tag{3.4}$$

That is

$$\Gamma(L, g) = \Gamma'(L, g)\,\big|_{\mathbb{F}_q} = \left\{ c \in \mathbb{F}_{q^m}^n \;\middle|\; H \cdot c = 0 \right\}\big|_{\mathbb{F}_q}$$

Proof. A proof may be found in [vL92]. $\square$

Now we express this code as a primary Goppa code over the projective line $\mathbb{P}^1(\mathbb{F}_{q^m})$.

Theorem 3.12. *In addition to the notation of Definition 3.10 let* $D_L = \sum_{i=1}^{n}(\alpha_i : 1)$, *and* $P_\infty = (1:0)$. *Let* $G_0 = (g)$ *be the zero divisor of* $g(x)$ *(actually we work with the homogenization of* $g(x)$*). Then*

$$\Gamma'(L,g) = C_\Omega\left(D_L,\ G_0 - P_\infty\right).$$

The minimal distance of $\Gamma(L,g)$ *is at least* $1 + \deg(g(x))$.

Proof. For $0 \le j \le t-1$ we show that $\varphi_j := \frac{x^j z^{t-j}}{g(x)}$ is in $\mathcal{L}(G_0 - P_\infty)$. We compute the principal divisor of φ_j as

$$\left(\frac{x^j z^{t-j}}{g(x)}\right) = (x^j z^{t-j}) - (g(x)) = j \cdot (0:1) + \underbrace{(t-j)}_{\ge 1}\underbrace{(1:0)}_{=P_\infty} - G_0 \ge -G_0 + P_\infty$$

The functions φ_j are linearly independent and since $\dim(\mathcal{L}(G_0 - P_\infty)) = t$, a basis of $\mathcal{L}(G_0 - P_\infty)$ is given by $\{\varphi_0, \dots, \varphi_{t-j}\}$. So the parity check matrix (3.4) of $\Gamma'(L,g)$ is also a parity check matrix for $C_\Omega\left(D_L,\ G_0 - P_\infty\right)$, and the codes are equal.

To estimate the minimal distance of this code we apply Theorem 3.7 and obtain $d \ge d_\Gamma = \deg(G_0 - P_\infty) + 2 = \deg(g(x)) + 1$. $\square$

3.3.4 Hermitian Codes

In the examples above the curve used to define the code was very simple. We now analyze a class of codes from more interesting curves, namely Hermitian curves. Hermitian codes are a very popular example of algebraic geometric codes. Hermitian codes are treated for example in [JLJ+89, LSH97]

As we have seen in Section 1.8 the Hermitian curve H_m has m^3 affine $\mathbb{F}_m$-rational points $P_1, \dots, P_{m^3}$ and a single point at infinity $Q = (0:1:0)$. Its genus is given by $g = \frac{m(m-1)}{2}$. And the vector space $\mathcal{L}(aQ)$ is spanned by $\left\{\frac{x^i y^j}{z^{i+j}} \mid im + j(m+1) \le a,\ i \le m\right\}$.

Definition 3.13. *Let* $a \in \mathbb{N}$. *The one point Goppa code defined over the Hermitian curve* H_m

$$C_{m,a} := C_{\mathcal{L}}(P_1 + \dots + P_{m^3},\ aQ)$$

is called a Hermitian code.

Theorem 3.14. *For $2g-2 < a < m^3$ the Hermitian code $C_{m,a}$ has parameters $n = m^3$, $d \geq n - a$ and $k = a - g + 1$.*

Proof. This is a direct consequence of Theorem 3.2, taking into account that $\deg(aQ) = a$ and that the index of speciality $i(G)$ is zero for $\deg(G) > 2g-2$. $\square$

Theorem 3.15. *For $2g-2 < a < m^3 - 2g + 2$ the dual of a Hermitian code is again a Hermitian code. That is*

$$C_\Omega\left(P_1 + \cdots + P_{m^3},\ a'Q\right) = C_{\mathcal{L}}(P_1 + \cdots + P_{m^3},\ aQ) = C_{m,a}$$

with $a' = m^3 - 2 + 2g - a$.

Proof. Let $D = P_1 + \cdots + P_{m^3}$. The bounds imposed on a make sure that also $a' > 2g-2$. So the dimension k' of $C_\Omega\left(D,\ a'D\right)$ can be computed as $k' = m^3 - a' - 1 + g$, while the dimension k of $C_{\mathcal{L}}(D,\ aD)$ is given by $k = a - g + 1$. Because the dimensions must agree we get that $a' = m^3 - 2 + 2g - a$.

To proof the statement it is sufficient to show, that if $F(x,y) \in K[x,y]$ is any polynomial with degree $s < m^3$, then

$$\sum_{(x:y:1)} F(x,y) = 0$$

where the sum is taken over the affine $\mathbb{F}_q$-rational points of H_m.

Write $F(x,y)$ in the form

$$F(x,y) = \sum_{i=1}^{s} a_i x_i + \sum_{j=0}^{s} b_j y_j + \sum_{i,j>0}^{i+j\leq s} c_{i,j} x^i y^j$$

For fixed $y \in \mathbb{F}_{m^2}$ let ξ be a solution for x of $x^{m+1} + y^m + y = 0$, that is $(y : \xi : 1) \in H_m$. If $y^m + y = 0$ then also $\xi = 0$, and if $y^m + y \neq 0$ then ξ takes $m+1$ different values in $\mathbb{F}_{m^2}$.

$$\sum_{(x:y:1)} F(x,y) = \sum_{y\in\mathbb{F}_{m^2}} \sum_{\xi} F(x,y) =$$

$$\underbrace{\sum_{y} \sum_{i=1}^{s} a_i \sum_{\xi} x^i}_{=:A} + \underbrace{\sum_{j=0}^{s} b_j \sum_{y} y^j \sum_{\xi} 1}_{=:B} + \underbrace{\sum_{i+j>0}^{i+j\leq s} c_{i,j} \sum_{y} y^j \sum_{\xi} x^i}_{=:C}$$

A and C contribute 0 to the sum, because in both cases the innermost sum is the sum over the roots of $x^{m+1} = (-y^m - y)^i$. The term B also contributes 0, since

$$B = \underbrace{b_0 \cdot m^3}_{=0} + \sum_{j=1}^{m} b_j \sum_{y} y^j \underbrace{\sum_{\xi} 1}_{=1} = \sum_{j=1}^{m} b_j \underbrace{\sum_{y \in \mathbb{F}_{m^2}} y^j}_{=0} = 0$$

□

Example 3.16. Let us now compare the Hermitian codes with Reed-Solomon codes. We consider codes over a field with $q = 64$ elements. For both codes we choose the information rate R as $R = \frac{1}{2}$.

For the Hermitian code we get with $m = q^{1/2} = 16$

$$\begin{aligned} n &= m^3 = 512 \\ g &= \frac{m(m-1)}{2} = 28 \\ k &= a - g + 1 = \frac{n}{2} \Rightarrow a = \frac{n}{2} + g - 1 = 283 \\ d_\Gamma &= n - a = 229 \end{aligned}$$

From this we compute the relative minimal distance δ as $\delta = \frac{229}{512} \approx 0.447$.

A Reed-Solomon code over the field $\mathbb{F}_{64}$ has the parameters

$$\begin{aligned} n &= q - 1 = 63 \\ d &= q - k = q - \frac{n}{2} = 64 - 32 = 32 \end{aligned}$$

So for the Reed-Solomon code we get the relative minimal distance δ as $\delta = \frac{32}{63} \approx 0.5$.

At the first glance it seems, that the Reed-Solomon code is slightly better than the Hermitian code, but we should notice, that the Reed-Solomon code is much shorter than the Hermitian code for a fixed field. To transmit the same amount of data with both codes, we need 4 times as many words using the Reed-Solomon code than we need words using the Hermitian code. So we should actually compare the Reed-Solomon code with an 8 times repeated Reed-Solomon code. But the simple repetition of a code does not increase the minimal distance d of the code, only the length n and the dimension k are raised.

Now let us also compute the probability of an uncorrectable error for both codes. That is an error of weight at least $t = \left\lfloor \frac{d-1}{2} \right\rfloor$. The probability for an error with weight l equals

$$P_l = \binom{n}{l}(1-p)^{n-l}p^l$$

where p is the probability for an error in the transmission of one character. To find the probability P of an uncorrectable error we should compute P as one of sums

$$P = \sum_{l=t}^{n} P_l = 1 - \sum_{l=0}^{t-1} P_l,$$

but we get already a good estimate for this probability, if we just look at P_t and P_{t+1}. After actually computing the values we see that $P \approx P_t$, for t sufficiently large.

For $p = 0.01$ the probabilities for an uncorrectable error are

$$\begin{array}{rrl} P \approx P_{114} \approx 0.6 \cdot 10^{-113} & P_{115} \approx 0.2 \cdot 10^{-114} & \text{for the Hermitian code} \\ P \approx P_{16} \approx 0.2 \cdot 10^{-17} & P_{17} \approx 0.6 \cdot 10^{-19} & \text{for the RS code} \end{array}$$

3.4 Long Algebraic Geometric Codes

In order to construct very good codes, that is codes with a high information rate, and a high relative minimal distance, it is necessary to consider codes of great length. If we consider a fixed field $\mathbb{F}_q$, then for plane curves the number of $\mathbb{F}_q$-rational points is at most $q^2 + 2q$. So if we want to construct long codes from plane curves, we also have to consider bigger fields.

But we can overcome this bound, if we consider curves in higher dimensional space. Then we can fix the field and increase the dimension of the space to produce longer codes. We do not go into details, how the basic notions from algebraic geometry are generalized to higher dimensional space. The details may be found in [Mor91, Sti93]. The theorems on the dimension and minimal distance of Goppa codes still hold for curves in higher dimensions.

Consider a Goppa code $C_\Omega(D, G)$ defined over some curve of genus g, with $\deg(D) = n$, $n > \deg(G) = \alpha \geq 2g - 2$. From Theorem 3.7 we get for the information rate R and the relative minimal distance δ the following bounds

$$\begin{aligned} R &= \frac{n - \alpha - 1 + g}{n} \approx 1 - \frac{\alpha}{n} + \gamma \\ \delta &\geq \frac{\alpha - 2g + 2}{n} \approx \frac{\alpha}{n} - 2\gamma \end{aligned}$$

where $\gamma = \frac{g}{n}$. By choosing α, we can construct codes with arbitrary δ and R, such that $\delta + R$ asympotitcally equals $1 - \gamma$.

Let us now compare these codes with the Gilbert-Varshamov bound (2.19). First we compute when the Goppa line $R = 1-\delta-\gamma$ is tangent to the Gilbert-Varshamov curve $R = 1 - H_q(\delta)$.

$$H_q(x) = x \log_q(q-1) - x \log_q x - (1-x)\log_q(1-x)$$
$$H_q'(x) = \log_q(q-1) - \log_q x + \log_q(1-x)$$

We obtain $-H_q'(\delta_0) = R'(\delta_0) = -1$ when $\delta_0 = \frac{q-1}{2q-1}$. Consequently the Goppa line is tangent if

$$\gamma = H_q(\delta_0) - \delta_0 = \log_q(2q-1) - 1.$$

In other words, whenever $\gamma < \log_q(2q-1) - 1$, there is an interval $[\delta_1, \delta_2]$ inside with Goppa codes are better than those of Gilbert-Varshamov.

In 1982 *Tsfasman, Vlăduţ and Zink* [TVZ82] showed the following theorem.

Theorem 3.17. *Let p be prime, $q = p^2, p \geq 7$ or $q = p^4, p \geq 3$. Then there exist one-point Goppa codes which are better than the Gilbert-Varshamov bound in some interval $[\delta_1, \delta_2]$.*

Proof. In [TVZ82] it is shown, that there exist Shimura curves over $\mathbb{F}_q$ with $\gamma \approx (\sqrt{q} - 1)^{-1}$. Consider now the a one-point Goppa code $C_\Omega(D, G)$ over such a curve. In the remarks before this theorem, we have already seen, that this code will be better than the Gilbert-Varshamov bound, if $\gamma < \log_q(2q-1) - 1$. So it remains to verify that

$$\frac{1}{\sqrt{q}-1} < \log_q(2q-1) - 1.$$

For $p = 7, q = p^2$ we get that $0.166 < 0.175$. After taking in inequality to the qth power, we see that the left hand side is decreasing, while right hand side is increasing. So the inequality remains true for $p > 7$. □

The curves used by Tsfasman, Vlăduţ and Zink are far too complicated for any practical application. *Garcia* and *Stichtenoth* describe a sequence of curves giving explicit equations. *Voss* and *Høholdt* [VH97] make the construction even more explicit and give generator matrices for the first codes in the family. It is interesting to notice that in this construction the first steps are the well known Reed-Solomon codes and Hermitian codes.

When you have eliminated the impossible, what ever remains, however improbable must be the truth.

Sir A. C. Doyle

Chapter 4

Decoding of Algebraic Geometric Codes

Notation. Again let $\mathbb{F}_q$ be a finite field for some prime power q, and let $\overline{\mathbb{F}_q}$ be the algebraic closure of $\mathbb{F}_q$. Throughout this chapter we fix the code $C_\Omega(D,G)$ defined over a curve V of genus g. Let the divisor D be of the form $D = P_1 + \cdots + P_n$. All functions in this chapter are rational functions on the curve V, with no poles at the divisor D.

Also assume that the word $c \in \mathbb{F}_q^n$ was transmitted, and the word $w = c+e$ was received. The word e is called the *error word*.

4.1 Introduction

According to [HP95] the development of an algorithm has three steps

$$E^{E^E}.$$

First one proves its *existence*, thereafter one tries to find an *effective* algorithm, and finally one needs an *efficient* algorithm for practical use.

The decoding of algebraic geometric codes has reached the second stage. Effective algorithms are available, but their complexity is still too high for practical use. The first steps to find efficient algorithms are already done, but many more must follow.

One of the first methods for decoding was given by Skorobogatov and Vlăduţ. An elegant solution of the decoding problem for algebraic geometric codes by a *majority scheme* was suggested by Feng and Rao. This algorithm was further generalized by *Duursma* [Duu93]. We presented the details of this method in the next sections.

The second important method for decoding Goppa codes is bases on *Sakata*'s multidimensional generalization of the Berlekamp-Massey algorithm. We do not consider this approach here. Details of this may be found in [SJM+95, SJH95, JLJH92]

For decoding Goppa codes (and linear codes in general) it is important to obtain some information on the error word e from the received word w.

Theorem 4.1. *For any $\varphi \in \mathcal{L}(G)$*

$$e \cdot \varphi = w \cdot \varphi \tag{4.1}$$

Proof.

$$e \cdot \varphi = \underbrace{c \cdot \varphi}_{=0} + e \cdot \varphi = (c + e) \cdot \varphi = w \cdot \varphi$$

□

4.2 Error Locator Decoding

The algorithms described next are based on the idea of first determining at which positions the error occured and with that knowledge to compute the actual error values.

Definition 4.2. *For an error word $e = (e_1, \ldots, e_n)$ the point P_l is called an* error location *if $e_l \neq 0$.*

A non-zero rational function θ is called an error locator *iff*

$$e_l \neq 0 \implies \theta(P_l) = 0$$

and θ has no poles among $P_1, \ldots, P_n$.

Note that the error locator can also be zero for positions where no error occured. This is more general then the similar notation of an error locator polynomial when decoding Reed-Solomon codes.

Theorem 4.3 (Existence of an Error Locator). *Let e have weight at most t and let A be a divisor with support distinct from D. If $\dim(A) > t$ then there exists an error locator in $\mathcal{L}(A)$.*

Proof. Let $s = \dim(A)$ and let $\{\psi_1, \ldots, \psi_s\}$ be a basis of $\mathcal{L}(A)$. Then a function $\theta = b_1\psi_1 + \cdots + b_s\psi_s$ is an error locator if and only if

$$b_1\psi_1(P_l) + \cdots + b_s\psi_s(P_l) = 0$$

for all error locations P_l. This is a linear system of at most t equations in s unknowns. By assumption $t < s = \dim(A)$, so it has a non-zero solution. □

In order to find an error locator we take an approach that is similar to finding a weak formulation of a partial differential equation. For an error locator θ we have that

$$\forall P_i \in D : \quad e_i\, \theta(P_i) = 0$$

Now we multiply this equation by a *test function* χ from some space $\mathcal{L}(C)$ and take the sum over all points P_i. That is θ is a solution of the system

$$\text{Find } \theta \in \mathcal{L}(A) : \quad e \cdot (\theta\chi) = 0 \qquad \forall \chi \in \mathcal{L}(C) \tag{4.2}$$

The next theorem states that if θ is a solution of (4.2) for a big enough space $\mathcal{L}(C)$, then θ is an error locator.

Theorem 4.4 (Computation of an Error Locator). *Let e have weight at most t and let C be a divisor with support distinct from D and* $\deg(C) \geq t + 2g - 1$.

Then a function θ without poles in D is an error locator if and only if

$$e \cdot (\theta\chi) = 0 \qquad \forall \chi \in \mathcal{L}(C) \tag{4.3}$$

Proof. Let $e' := (e_1\theta(P_1), \ldots, e_n\theta(P_n))$. This word as weight at most t and

$$e \cdot (\theta\chi) = \sum_{i=1}^{n} e_i\, \theta(P_i)\chi(P_i) = e' \cdot \chi$$

For an error locator θ we have that $e' = 0$. Because χ has no poles in D also $e' \cdot \chi = 0$, that is θ is a solution of (4.3).

Now let θ be a solution of (4.3). We will show that θ then also is an error locator. $e' \cdot \chi = 0 \quad \forall \chi \in \mathcal{L}(C)$, that is e' is a code word of $C_\Omega(D, C)$. By Theorem 3.7 e' is either zero or has weight at least $\deg(C) - 2g + 2 = t + 1$. By construction e' has weight at most t, therefore e' must be zero, that is θ is an error locator. □

Corollary 4.5. *Let e have weight at most t, A as in Theorem 4.3, $s = \dim(A)$, $\{\psi_1, \ldots, \psi_s\}$ be a basis of $\mathcal{L}(A)$. Furthermore let C be as in the theorem above, $c = \dim(C)$ and let $\{\chi_1, \ldots, \chi_c\}$ be a basis of $\mathcal{L}(C)$.*

Then $\theta = b_1\psi_1 + \cdots + b_s\psi_s$ is an error locator in $\mathcal{L}(A)$ if and only if $(b_1, \ldots, b_s)$ is a solution of

$$\begin{pmatrix} e \cdot \psi_1\chi_1 & e \cdot \psi_2\chi_1 & \ldots & e \cdot \psi_s\chi_1 \\ e \cdot \psi_1\chi_2 & e \cdot \psi_2\chi_2 & \ldots & e \cdot \psi_s\chi_2 \\ \vdots & \vdots & \ddots & \vdots \\ e \cdot \psi_1\chi_c & e \cdot \psi_2\chi_1 & \ldots & e \cdot \psi_s\chi_c \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.4}$$

Note that for $A + C \leq G$ the syndromes needed in (4.4) can be computed from the received word as $w \cdot (\psi_i\chi_j)$.

Theorem 4.6 (Computation of the Error Values). *Let e have weight at most t and let A and B be divisors with support distinct from D. If $\deg(B) \geq \deg(A) + 2g - 1$ and θ an error locator in $\mathcal{L}(A)$ then e can be uniquely determined by the syndromes $e \cdot \varphi$ for $\varphi \in \mathcal{L}(B)$.*

Note that if $B \leq G$ then the syndromes required in this theorem can be computed from the received word as $w \cdot \varphi$.

Proof. Let θ be an error locator in $\mathcal{L}(A)$ and let $M := \{P_l \in D \mid \theta(P_l) = 0\}$ be the set of possible error locations. By Theorem 1.35 we know that θ has at most $\deg(A)$ zeros. Among these are the true error positions. Therefore we get for any code word with support in M only and for any function φ, especially for $\varphi \in \mathcal{L}(B)$, that

$$e \cdot \varphi = \sum_{P_l \in M} \varphi(P_l)e_l \tag{4.5}$$

By construction the error word e is a solution of this system.

It remains to show that e is the only solution. Assume that e and e' are two solutions of (4.5), that is $e \cdot \varphi = e' \cdot \varphi$ for all $\varphi \in \mathcal{L}(B)$. That is $(e - e') \cdot \varphi = 0$ for all $\varphi \in \mathcal{L}(B)$ and so $e - e'$ is a code word of $C_\Omega(D, B)$. By Theorem 3.7 $wt(e - e') \geq \deg(G) - 2g + 2 = \deg(A) + 1$ or $e - e'$ is the zero word. Both e and e' have at most $\deg(A)$ nonzero entries, so $e - e' = 0$. Which shows that (4.5) has e as unique solution. □

Corollary 4.7. *Let e, A and B be as in the theorem above, $s := \dim(B)$, $M := \{P_l \in D \mid \theta(P_l) = 0\}$ be as in the proof and let $\{\varphi_1, \ldots, \varphi_s\}$ be a basis of $\mathcal{L}(B)$.*

Then e is the unique solution of the linear system

$$\sum_{P_l \in M} \varphi_i(P_l) e_l = e \cdot \varphi_i \qquad i = 1, \ldots, s \tag{4.6}$$

Remark. For one point Goppa codes $C_\Omega(D,G)$ with $G = mQ$ for single point Q, the divisors needed above can be taken as $A = aQ$, $B = bQ$ and $C = cQ$. Also all the basis needed can be chosen as subsets of a single basis.

4.3 The SV-Algorithm

If we combine the theorems just presented we get the *Skorobogatov-Vlăduţ Error Processor* (for short SV-Algorithm) for the code $C_\Omega(D,G)$, with $n = \deg(D)$ and $n > \deg(G) > 2g-2$.

For the SV-Algorithm we choose A, B and C so, that $B \leq G$ and $A+C \leq G$, that is we demand that all the syndromes occuring can be computed from the received word.

We will now determine the degrees of the divisors A, B and C. And with that also get the maximal number of errors this algorithm can correct.

- In order to assure the existence of an error locator in $\mathcal{L}(A)$ we need that $\dim(A) > t$. By the Riemann-Roch theorem (1.38) we get that $\deg(A) \geq \dim(A) + g - 1 \geq t + g$.

  In order to compute the syndromes needed to find the error locator we demand that $A+C \leq G$. So $\deg(C) \leq \deg(G-A) = \deg(G) - \deg(A) \leq \deg(G) - t - g$.

- In order to apply Theorem 4.4 we need that $\deg(C) \geq t + 2g - 1$.

Putting these two inequalities together we get

$$\deg(G) - t - g \geq \deg(C) \geq t + 2g - 1$$

Solving for t we obtain that $2t \leq \deg(G) - 3g + 1$.

In order to compute the error values we need that $\deg(B) \geq \deg(A) + 2g - 1 \geq t + 3g - 1$ and we need that $\deg(B) \leq \deg(G)$, that is $t \leq \deg(G) - 3g + 1$. This is no further restriction.

So for the code $C_\Omega(D,G)$ with designed distance $d_\Gamma = \deg(G) - 2g + 2$ up to

$$t \leq \left\lfloor \frac{d_\Gamma - 1 - g}{2} \right\rfloor \tag{4.7}$$

error can be corrected. Note that in general this is less than $\lfloor \frac{d_\Gamma - 1}{2} \rfloor$, the number of errors we could correct at least with a code of designed distance d_Γ.

4.3.1 Summary of the SV-Algorithm

Consider the code $C_\Omega(D, G)$ defined over a curve of genus g over some finite file $\mathbb{F}_q$. Let $\varphi_1, \dots, \varphi_k$ be a basis of $\mathcal{L}(G)$, and let D be of the form $D = P_1 + \dots + P_n$.

Initialization – once for a given code

1. Choose a divisor A, distinct from D, with $\deg(A) = t + g$. Let $C = G - A$. Typically we choose $A \leq G$.
2. Choose a divisor B, distinct from D, with $\deg(B) = t + 3g + 1$ and $B \leq G$.
3. Choose basis of $\mathcal{L}(A)$, $\mathcal{L}(B)$ and $\mathcal{L}(C)$. $\mathcal{L}(B)$ can be chosen as a subset of the basis of $\mathcal{L}(G)$, w.l.o.g. we assume that $\varphi_1, \varphi_2, \dots$ is a basis of $\mathcal{L}(B)$, so

$$\begin{aligned}
\mathcal{L}(A) &= \operatorname{span}\{\theta_1, \theta_2, \dots\} \\
\mathcal{L}(C) &= \operatorname{span}\{\chi_1, \chi_2, \dots\} \\
\mathcal{L}(B) &= \operatorname{span}\{\varphi_1, \varphi_2, \dots\}
\end{aligned}$$

Decoding

Let w be the word received.

1. Compute the syndrome matrix S, that is compute

$$S = \begin{pmatrix} w \cdot \theta_1\chi_1 & w \cdot \theta_2\chi_1 & w \cdot \theta_3\chi_1 & \dots \\ w \cdot \theta_1\chi_2 & w \cdot \theta_2\chi_2 & w \cdot \theta_3\chi_2 & \dots \\ w \cdot \theta_1\chi_3 & w \cdot \theta_2\chi_3 & w \cdot \theta_3\chi_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

   If all syndromes are zero, no error has occured and the received word is a code word.

2. Find a nonzero vector $(b_1, b_2, \dots)$ in the null space of S. If no such vector exists we can not decode the received word.

   Otherwise $\theta = b_1\theta_1 + b_2\theta_2 + \dots$ is an error locator.

3. Find the error locations M

$$M = \{1 \leq i \leq n \mid \theta(P_i) = 0\}$$

4. Find the error values e_i by solving the linear system

$$\sum_{l \in M} \varphi_i(P_l) e_l = w \cdot \varphi_i \quad \forall \varphi_i \in \mathcal{L}(B)$$

   Let $e_i = 0$ for $i \notin M$.

   Then $c = w - e$ is the nearest neighbor code word of w.

Remark. By evaluating the rational functions occuring in advance, once for a given code, we can perform all the operations using vectors in $\mathbb{F}_q^n$.

Note that the matrix occuring in the last step of the algorithm is a submatrix of the parity check matrix of the code.

4.3.2 Decoding Reed-Solomon Codes

We have already seen in Section 3.3.1, that we can consider Reed-Solomon codes as primary Goppa codes over the projective line, that is a curve of genus 0. So we can apply the SV-Algorithm to decode RS codes.

Let $t > 0$ and let $d = 2t + 1$. Consider now a Reed-Solomon code $C_\Omega(D, G)$, with $G = (d-1)Q - (0:1)$, $Q = (1:0)$ the single point at infinity, and D the sum of the $q-1$ remaining points of the projective line. We have already seen, that $\mathcal{L}(G)$ is spanned by $\varphi_i = x^i$ for $1 \leq i \leq d-1$. Here we write the basis functions in affine form.

For decoding we choose the divisor A as

$$A = (t+1)Q - (0:1).$$

The divisor C must then be $C = G - A = (t-1)Q$. As the divisor B we choose $B = (t+2)Q - (0:1)$. Basis of the corresponding vector spaces are given by

$$\begin{aligned}
\mathcal{L}(A) &= \operatorname{span}\left\{ \theta_i = x^i \;\middle|\; 1 \leq i \leq t+1 \right\} \\
\mathcal{L}(C) &= \operatorname{span}\left\{ \chi_i = x^i \;\middle|\; 0 \leq i \leq t-1 \right\} \\
\mathcal{L}(B) &= \operatorname{span}\left\{ \varphi_i = x^i \;\middle|\; 1 \leq i \leq t+2 \right\}
\end{aligned}$$

Note that here $C \not\leq G$. That is not necessary, we only need that $A + C \leq G$, which is of course the case.

So the syndromes needed to find an error locator are determined by the matrix

$$S = \begin{pmatrix} x & x^2 & x^3 & \dots & x^{t+1} \\ x^2 & x^3 & x^4 & \dots & x^{t+2} \\ x^3 & x^4 & x^5 & \dots & x^{t+3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x^{t-1} & x^t & x^{t+1} & \dots & x^{2t} \end{pmatrix}$$

As we can see this matrix is symmetric. Even more this matrix is a so-called Hankel matrix, that is $S_{i,j} = s_{i+j}$ for some vector s. In our case $s = (1, x, x^2, \dots, x^{2t})$. For Hankel matrices special variants of Gaussian elimination are known. Those special algorithms perform the elimination using only $\mathcal{O}(t^2)$ operations. ([FWRT94]).

4.3.3 Decoding Hermitian Codes

Because Hermitian codes are defined using curves of genus different from zero, this algorithm will not be able to exploit the full error correction capacity of the code.

Consider the Hermitian curve $H_m = V(x^{m+1} + y^m z + yz^m)$ over the field $\mathbb{F}_{m^2}$, and let g be the genus of this curve. Let D be the divisor of the m^3 affine points and let $Q = (0 : 1 : 0)$ be the single point at infinity. We use the primary Goppa code $C_\Omega(D,\ aQ)$ for some integer a satisfying $2g - 2 < a < m^3 - 2g - 2$.

As we have seen in Section 1.8 the space $\mathcal{L}(aQ)$ for an integer a and $Q = (0 : 1 : 0)$ is spanned by monomials, more precisely

$$\mathcal{L}(aQ) = \operatorname{span}\left\{ x^i y^j \;\middle|\; 0 \le im + j(m+1) \le a,\ i \le m \right\}$$

Again we write the rational functions occuring in affine form.

Naturally we choose the divisors A and B for the SV-Algorithm as multiples of the point Q.

Let now $m = 2$ and $a = 6$, then the genus of the curve equals $g = 1$, and we obtain the designed distance d_Γ as $d_\Gamma = 6 - 2g + 2 = 6$. The SV-algorithm can correct up to $t \le \left\lfloor \frac{d_\Gamma - 1 - g}{2} \right\rfloor = 2$ errors. Let the basis of $\mathcal{L}(6Q)$ be given by

$$\mathcal{L}(6Q) = \operatorname{span}\{1, y, y^2, x, xy, x^2\}$$

A parity check matrix for this code is given by

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

The syndrome matrix takes the form

$$S = e \cdot \begin{pmatrix} 1 & y & x \\ y & y^2 & xy \\ x & xy & x^2 \end{pmatrix}$$

Let the word received be

$$w = (\alpha, \alpha, \alpha, \alpha^2, 0, \alpha, 1, 1).$$

Then

$$S = \begin{pmatrix} \alpha^2 & \alpha^2 & \alpha \\ \alpha^2 & 0 & 0 \\ \alpha & 0 & 0 \end{pmatrix},$$

and $b = (0, \alpha^2, 1)$ is an element in the null space of S. That is an error locator for w is given by

$$\theta = \alpha^2 y + x.$$

We obtain the set of error locations as $M = \{1, 3, 6\}$.

It remains to compute the error values using the following system

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & \alpha & \alpha^2 \\ 0 & \alpha^2 & \alpha \\ 0 & 1 & \alpha \\ 0 & \alpha & 1 \\ 0 & 1 & \alpha \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_3 \\ e_6 \end{pmatrix} = \begin{pmatrix} \alpha^2 \\ \alpha^2 \\ 0 \\ \alpha \\ 0 \\ 0 \end{pmatrix}.$$

We obtain $e_1 = 0$, $e_3 = 1$, and $e_6 = \alpha$. Thus the code word searched for is

$$c = w - e = (\alpha^2, \alpha^2, \alpha, \alpha, 1, 1, 0, 0)$$

Now also consider the case $m = 4$. The genus of H_4 equals $g = 6$, so for $a = 19$ we get a code of designed distance $d_\Gamma = 19 - 2g + 2 = 9$. This is a 4-error correcting code. But the SV-algorithm is only capable of correcting single errors. The situation gets even worse, if we increase m further.

4.4 The Duursma-Algorithm

A major drawback of the SV-algorithm is, that it does not exploit the full error correction capacity of the code. The algorithm presented now extends the SV-Algorithm to correct up to $\lfloor \frac{d_\Gamma - 1}{2} \rfloor$ errors. This is done by determining additional syndromes by a majority voting scheme. This decoding procedure was developed by *Iwan M. Duursma* [Duu93], generalizing ideas by *Feng* and *Rao* [FR93].

Notation. Fix the primary Goppa code $C_\Omega(D, G)$, defined over some curve of genus g. Let $\deg(G) = 2t + 2g - 1$, that is we consider a code of designed distance $d_\Gamma = 2t + 1$, capable of correcting t errors.

Let Q be a point on the curve, $Q \notin \mathrm{supp}(D)$. Typically Q is taken as one of the points of G.

Furthermore all rational functions mentioned are assumed to have no pole at $\mathrm{supp}(D)$.

Finally let $w = c + e$ be the word received, for $c \in C_\Omega(D, G)$ and $\mathrm{wt}(e) \leq t$.

4.4.1 Divisors for Error Locator and Test Functions

Let A be a divisor of degree t, e.g. $A = tQ$, and let $A' := G - A - (2g - 1)Q$, which also has degree $\deg(A') = t$.

a) First we determine integers m and m', such that we find an error locator in $\mathcal{L}(A + mQ)$ using the functions in $\mathcal{L}(A' + m'Q)$ as test functions.

   For the existence of an error locator θ in $\mathcal{L}(A + mQ)$, we need that $\dim(A + mQ) > t$ (by Theorem 4.3). So $\deg(A + mQ) \geq t + g$. Thus $m \geq g$.

   To compute the error locator using $\mathcal{L}(A' + m'Q)$ as the space of test functions, we need that $\deg(A' + m'Q) > t + 2g - 1$ (by Theorem 4.4). So $m' \geq 2g - 1$.

   So the syndromes needed in Theorem 4.4 will be computed using rational functions from the space $\mathcal{L}((A + gQ) + (A' + (2g - 1)Q))$. Since

   $$(A + gQ) + (A' + (2g - 1)Q) = G + gQ.$$

   we need a method to find syndromes, which can not be computed directly from the received word.

b) Similarly we can also ask for an error locator η in $\mathcal{L}(A' + m'Q)$, using $\mathcal{L}(A + mQ)$ as the space of test functions. We then get that $m' \geq g$, $m \geq 2g - 1$.

c) Putting a) and b) together, we obtain the following spaces to work with

$$\mathcal{L}(A + (2g - 1)Q)$$
$$\mathcal{L}(A' + (2g - 1)Q)$$
$$\mathcal{L}(A + A' + (3g - 1)Q) = \mathcal{L}(G + gQ)$$

4.4.2 H-Order and Special Basis

We can not expect to find the unknown syndromes for arbitrary bases. But if the basis functions are "well behaved", the unknown entries can be computed one after the other.

For convenience we introduce the following notation.

Definition 4.8. *Let Q be a point on the curve, H be a divisor and θ a rational function. Then the* H-order *of θ, written as $\mu_{H,Q}(\theta)$, is the smallest integer m, such that $\theta \in \mathcal{L}(H + (m - \deg(H))Q)$, and ∞ if no such integer exists.*

In the whole of this chapter the point Q will remain unchanged. So we omit Q and write $\mu_H(\theta)$.

Theorem 4.9. *Let $\mu_H(\theta) = r < \infty$, $\mu_{H'}(\theta') < \infty$.*

a) Then $r \geq 0$ and $\nu_Q(\theta) = \deg(H) - n_Q(H) - \mu_H(\theta)$.

b) $\mu_{H+H'}(\theta \cdot \theta') = \mu_H(\theta) + \mu_{H'}(\theta')$

Proof. a) $\mathcal{L}(H + (m - \deg(H)Q)) = \{0\}$ for $m < 0$, so $\mu_H(\theta) \geq 0$.

The hypothesis $\mu_H(\theta) = r$ implies that

$$\theta \in \mathcal{L}(H + (r - \deg(H))Q) \setminus \mathcal{L}(H + (r - 1 - \deg(H))Q).$$

So $\nu_Q(\theta) = -(n_Q(H) + r - \deg(H))$.

b) Certainly $\theta \cdot \theta' \in \mathcal{L}(H + H' + (r + r' - \deg(H + H')Q))$ and $\nu_Q(\theta \cdot \theta') = \nu_Q(\theta) + \nu_Q(\theta')$, from which $\mu_{H+H'} = r + r'$ follows.

□

Theorem 4.10. *Let H be a divisor, r be an integer. Then there exists a basis $\{\psi_1, \ldots, \psi_n\}$ of $\mathcal{L}(H + rQ)$, such that*

$$\mu_H(\psi_1) < \mu_H(\psi_2) < \cdots < \mu_H(\psi_n)$$

is a complete list of the possible values for $\mu_H(\psi) \leq \deg(H) + r$.

Such a basis is called a special *basis of $\mathcal{L}(H + rQ)$.*

Proof. Let $D_i := H + (i - \deg(H))Q$. Then $\deg(D_{i+1}) = \deg(D_i) + 1$, and $\dim(D_{i+1}) \leq \dim(D_i) + 1$. Furthermore $D_{r+\deg(H)} = H + rQ$.

If a rational function $\psi_i \in \mathcal{L}(D_i) \setminus \mathcal{L}(D_{i-1})$ exists, then it has H-order $\mu_H(\psi_i) = i$. So we can construct a special basis by successively finding elements in $\mathcal{L}(D_i) \setminus \mathcal{L}(D_{i-1})$. □

The H-order and the notion of a special basis, are generalizations of Weierstrass gaps. If H consists of just a single point P, we use the following notions.

Definition 4.11. *Let P be a point. An integer $n \geq 0$ is called a* pole number *of P, iff $\mathcal{L}(nP) \neq \mathcal{L}((n-1)P)$. Otherwise n is called a* gap number *or* Weierstrass gap *of P.*

Clearly n is a pole number iff there exists a rational function in $\mathcal{L}(nP)$ having a pole of order n at P. For a gap number no such function exists. The set of pole numbers of P is an additive sub-semigroup of $\mathbb{N}_0$.

Theorem 4.12 (Weierstrass Gap Theorem). *Let P be a point on a smooth curve of genus $g > 0$. Then there are exactly g gap numbers $i_1 < \cdots < i_g$ of P. Furthermore $i_1 = 1$ and $i_g \leq 2g - 1$.*

Proof. That there are exactly g gaps, and that $i_g \leq 2g - 1$ is a consequence of the Riemann-Roch theorem (1.38).

Suppose that 1 is a pole number of P. Since the pole numbers form an additive semigroup, any $n \in \mathbb{N}$ is a pole number, and there are no gaps at all. This is a contradiction because $g > 0$. □

4.4.3 Syndrome Matrix

Now we use special bases for $\mathcal{L}(A + (2g-1)Q)$, $\mathcal{L}(A' + (2g-1)Q)$ and $\mathcal{L}(D + gQ)$ to define a syndrome matrix. For convenience we index the bases functions by their H-order. That is, we fix special basis

$$\begin{aligned}
\mathcal{L}(A + (2g-1)Q) &= \operatorname{span}\{\psi_i \mid 0 \leq i \leq t + 2g - 1\} \\
\mathcal{L}(A' + (2g-1)Q) &= \operatorname{span}\{\chi_i \mid 0 \leq i \leq t + 2g - 1\} \\
\mathcal{L}(D + gQ) &= \operatorname{span}\{\varphi_i \mid 0 \leq i \leq \deg(D) + g\}
\end{aligned}$$

With this we define the $(t+g) \times (t+g)$ syndrome matrix as

$$S_{i,j} := e \cdot \psi_i \chi_j.$$

We index the rows and columns of this matrix by the H-order of the generating basis functions. We call these indices the *order* of the row and column.

Some entries in this matrix can not be computed from the received word, those entries are called *unknown* entries. We shall call a row or column *known* if all its entries are known.

We denote the submatrix of rows order up to r and columns up to order r' by $S\,|_{r,r'}$.

Example 4.13. Take now the Hermitian code $C_\Omega(D,G)$ defined on the curve $H_3 = V(x^4+y^3z+yz^3)$ over $\mathbb{F}_9$. The genus of H_3 equals $g=3$. We let $t=6$ and obtain G as

$$G = (2t+2g-1)Q = 17Q.$$

As usual D is taken to be the sum of the 27 affine $\mathbb{F}_9$-rational points of the curve.

$$\begin{array}{rccccccccc}
D = & (0:0:1) & + & (0:\alpha^2:1) & + & (0:\alpha^6:1) & + & (1:1:1) & + \\
& (1:\alpha^5:1) & + & (1:\alpha^7:1) & + & (\alpha:\alpha:1) & + & (\alpha:\alpha^3:1) & + \\
& (\alpha:\alpha^4:1) & + & (\alpha^2:1:1) & + & (\alpha^2:\alpha^5:1) & + & (\alpha^2:\alpha^7:1) & + \\
& (\alpha^3:\alpha:1) & + & (\alpha^3:\alpha^3:1) & + & (\alpha^3:\alpha^4:1) & + & (\alpha^4:1:1) & + \\
& (\alpha^4:\alpha^5:1) & + & (\alpha^4:\alpha^7:1) & + & (\alpha^5:\alpha:1) & + & (\alpha^5:\alpha^3:1) & + \\
& (\alpha^5:\alpha^4:1) & + & (\alpha^6:1:1) & + & (\alpha^6:\alpha^5:1) & + & (\alpha^6:\alpha^7:1) & + \\
& (\alpha^7:\alpha:1) & + & (\alpha^7:\alpha^3:1) & + & (\alpha^7:\alpha^4:1) & & &
\end{array}$$

We choose $A = tQ$. With this also $A' = tQ$, and all the basis occuring can be chosen as subsets of a single basis of $\mathcal{L}(20Q)$. Furthermore we have $\mu_A(\varphi) = \mu_{A+A'}(\varphi) = -\nu_Q(\varphi)$. A special basis (written in affine form) of $\mathcal{L}(20Q)$ is given in the following table.

i	0	3	4	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
φ_i	1	x	y	x^2	xy	y^2	x^3	x^2y	xy^2	y^3	x^3y	x^2y^2	xy^3	y^4	x^3y^2	x^2y^3	xy^4	y^5

The syndrome matrix is computed using the functions given in Figure 4.1. The functions are again written in affine form. Unused entries are left black. Initially unknown entries are underlined. Numbers at the border indicate the order of the row and column respectively.

We express the functions in this table as linear combinations of the basis functions. Most of the entries are already basis functions. The remaining entries can be expressed as

$$\begin{aligned}
x^4 &= 2y+2y^3 & \qquad x^4y &= 2y^2+2y^4 \\
x^5 &= 2xy+2xy^3 & \qquad x^4y^2 &= 2y^3+2y^5 \\
x^6 &= 2x^2y+2x^2y^3 & \qquad x^5y &= 2xy^2+2xy^4
\end{aligned}$$

	0	3	4	6	7	8	9	10	11
0	1	x	y	x^2	xy	y^2	x^3	x^2y	xy^2
3	x	x^2	xy	x^3	x^2y	xy^2	x^4	x^3y	x^2y^2
4	y	xy	y^2	x^2y	xy^2	y^3	x^3y	x^2y^2	xy^3
6	x^2	x^3	x^2y	x^4	x^3y	x^2y^2	x^5	x^4y	x^3y^2
7	xy	x^2y	xy^2	x^3y	x^2y^2	xy^3	x^4y	x^3y^2	$\underline{x^2y^3}$
8	y^2	xy^2	y^3	x^2y^2	xy^3	y^4	x^3y^2	$\underline{x^2y^3}$	$\underline{xy^4}$
9	x^3	x^4	x^3y	x^5	x^4y	x^3y^2	$\underline{x^6}$	$\underline{x^5y}$	$\underline{x^4y^2}$
10	x^2y	x^3y	x^2y^2	x^4y	x^3y^2	$\underline{x^2y^3}$	$\underline{x^5y}$	$\underline{x^4y^2}$	
11	xy^2	x^2y^2	xy^3	x^3y^2	$\underline{x^2y^3}$	$\underline{xy^4}$	$\underline{x^4y^2}$		

Figure 4.1: Functions to compute the syndrome matrix in Example 4.13

4.4.4 Error Locator Candidates

We now describe a method, how to determine the initially unknown entries in the syndrome matrix. First we try to find an error locator from the known entries in the syndrome table. This also gives us some important relations on the unknown entries.

Definition 4.14. *a) A rational function θ is called a* (r, r')-row-candidate *iff $\mu_A(\theta) = r$, and*

$$e \cdot \theta\chi = 0 \qquad \forall \chi \in \mathcal{L}(A' + r'Q).$$

The candidate is called normalized iff $\theta = \psi_r + \theta'$ for $\mu(\theta') < r$.

b) A rational function η is called a (r, r')-column-candidate *iff $\mu_{A'}(\eta) = r'$, and*

$$e \cdot \psi\eta = 0 \qquad \forall \psi \in \mathcal{L}(A + rQ).$$

The candidate is called normalized iff $\eta = \chi_{r'} + \eta'$ for $\mu(\eta') < r'$.

Every candidate can be normalized by multiplication by a suitable non-zero scalar. Obviously a (r, r')-row-candidate is also a $(r, r'-1)$-row-candidate, but the first index r can not be changed. Similarly a (r, r')-column-candidate is also a $(r-1, r')$-columns-candidate, here the second index r' can not be changed.

Theorem 4.15. *Let $\theta = \sum_{j<r} \alpha_r \psi_j + \underbrace{\alpha_r}_{=1} \psi_r$ be a rational function.*

a) θ is a normalized (r, r')-row-candidate iff $S\,|_{r,r'} \cdot \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_r \end{pmatrix} = 0.$

b) If θ is a $(r, t + 2g - 1)$-row-candidate, then θ is an error locator.

c) If θ is a normalized $(r, r' - 1)$-row-candidate and η is a normalized $(r - 1, r')$-column-candidate then

$$e \cdot \theta\chi_{r'} = e \cdot \psi_r \eta.$$

d) If θ_1, θ_2 are two distinct normalized $(r, r' - 1)$-row-candidates with

$$e \cdot \theta_1\chi_{r'} \neq e \cdot \theta_2\chi_{r'}$$

then there exists no $(r - 1, r')$-column-candidate.

Similar results also hold for column candidates.

Proof. a) By construction, $\mu_A(\theta) = r$ and θ is normalized.

By linearity the condition for an error locator candidate

$$\forall\chi \in \mathcal{L}(A' + r'Q) : e \cdot \theta\chi = 0$$

is equivalent to

$$\forall 0 \leq i < r' : e \cdot \theta\chi_i = 0,$$

which is again equivalent to the linear system in the theorem.

b) This is a direct consequence of Theorem 4.4

c) Let $\theta = \psi_r + \theta'$, $\eta = \chi_{r'} + \eta'$ with $\mu_A(\theta') < r$ and $\mu_{A'}(\eta') < r'$. Then using these relations and the properties of candidates we obtain the following sequence of equations:

$$e \cdot \theta\chi_{r'} = e \cdot \theta\eta - \underbrace{e \cdot \theta\eta'}_{=0} = e \cdot \theta\eta = e \cdot \psi_r\eta + \underbrace{e \cdot \theta'\eta}_{=0} = e \cdot \psi_r\eta$$

d) Assume there exists a column candidate η of order $(r - 1, r')$. We can assume that η is normalized. By part c we get that

$$e \cdot \theta_1\chi_{r'} = e \cdot \psi_r\eta = e \cdot \theta_2\chi_{r'},$$

contradicting the hypothesis.

□

4.4.5 Estimating Additional Syndrome Entries

Let the syndrome entries of order up to $s-1$ be known. Then we consider the unknown entries of order $r+r'=s$. Because of the special order of the basis functions all element in $S\,|_{r,r'}$, except the entry at position r,r' are known.

Let θ be a normalized $(r,r'-1)$-row-candidate and let η be a normalized $(r-1,r')$-column-candidate, if such candidates exist. We would like θ and η also to be (r,r')-candidates.

Express θ as $\theta=\psi_r+\theta'$, with $\mu_A(\theta')<r$. Now we test θ using a test function of order r', that is we take a closer look at the r'th row of the syndrome matrix.

$$0=e\cdot\theta\chi_{r'}=\underbrace{e\cdot\psi_r\chi_{r'}}_{=S_{r,r'}}+e\cdot\theta'\chi_{r'}=0.$$

So our first guess for $S_{r,r'}$ is $S_{r,r'}=-e\cdot\theta'\chi'_r$.

Using the columns candidate $\eta=\chi_{r'}+\eta'$ we obtain the additional guess $S_{r,r'}=-e\cdot\psi_r\eta'$.

From Theorem 4.15 we can conclude, that the two guesses agree, if both candidates exist.

$$e\cdot\theta'\chi_{r'}=e\cdot\psi_r\chi_{r'}-e\cdot\theta\chi_{r'}=_{(4.15)}\ e\cdot\psi_r\chi_{r'}-e\cdot\psi_r\eta=e\cdot\psi_r\eta'$$

We call the value $S_{r,r'}$ found by this method a *test entry*.

Using such a guess for $S_{r,r'}$ we can compute a guess for $e\cdot\varphi_s$. From $\mu_{A+A'}(\psi_r\chi_{r'})=r+r'=s$ we get that $\psi_r\chi_{r'}=\alpha\varphi_s+\varphi'$, where $\mu_{A+A'}(\varphi')<s$, and α is some non-zero scalar. So we compute

$$e\cdot\varphi_s=\alpha^{-1}(\underbrace{e\cdot\psi_r\chi_{r'}}_{=S_{r,r}}-e\cdot\varphi'). \tag{4.8}$$

Definition 4.16. *We call a value computed in (4.8) a* vote *for $e\cdot\varphi_s$.*

Example 4.17. We continue Example 4.13.

Assume that received word w is zero, except for the following 6 entries:

$$\begin{array}{lll} w_1=1 & w_{10}=\alpha & w_{15}=\alpha^5 \\ w_{17}=1 & w_{21}=\alpha & w_{27}=1 \end{array}$$

It is clear, that in this situation the code word searched for is the zero-word, and that the received word w already is the error word e.

We should note, that the computations following depend heavily on the actual representation of the finite field. A different choice of the primitive element α will result in different estimates. Here we use $\mathbb{F}_9 = \mathbb{F}_3[x]/_{x^2+x+2}$ and $\alpha = 2x$.

First we compute those syndromes, which can be found directly from the received word. That are the syndromes up to order 17. After this the syndrome matrix S take the following form. The entries of order 18, 19, and 20 are still unknown.

$$S = \begin{pmatrix} \alpha & \alpha^6 & \alpha^5 & \alpha^6 & \alpha^3 & \alpha^5 & \alpha^3 & \alpha^6 & 0 \\ \alpha^6 & \alpha^6 & \alpha^3 & \alpha^3 & \alpha^6 & 0 & \alpha & \alpha^2 & \alpha^3 \\ \alpha^5 & \alpha^3 & \alpha^5 & \alpha^6 & 0 & 0 & \alpha^2 & \alpha^3 & \alpha^3 \\ \alpha^6 & \alpha^3 & \alpha^6 & \alpha & \alpha^2 & \alpha^3 & \alpha^3 & \alpha & \alpha^3 \\ \alpha^3 & \alpha^6 & 0 & \alpha^2 & \alpha^3 & \alpha^3 & \alpha & \alpha^3 & \mathrm{S}_{7,11} \\ \alpha^5 & 0 & 0 & \alpha^3 & \alpha^3 & 0 & \alpha^3 & \mathrm{S}_{8,10} & \mathrm{S}_{8,11} \\ \alpha^3 & \alpha & \alpha^2 & \alpha^3 & \alpha & \alpha^3 & \mathrm{S}_{9,9} & \mathrm{S}_{9,10} & \mathrm{S}_{9,11} \\ \alpha^6 & \alpha^2 & \alpha^3 & \alpha & \alpha^3 & \mathrm{S}_{10,8} & \mathrm{S}_{10,9} & \mathrm{S}_{10,10} & \\ 0 & \alpha^3 & \alpha^3 & \alpha^3 & \mathrm{S}_{11,7} & \mathrm{S}_{11,8} & \mathrm{S}_{11,9} & & \end{pmatrix}$$

First we check, if we can find an error locator using only the first 4 basis functions, that is basis functions up to order 6. This is not the case, so we continue to find the unknown entries.

We start with the entry of order $s = 18$, that is we try to find estimates for the entries $S_{7,11}$, $S_{8,10}$, $S_{9,9}$, $S_{10,8}$, and $S_{11,7}$.

- For the entry $S_{7,11}$ there exists no row-candidate, and by symmetry there is no column-candidate for the entry $S_{11,7}$.
- A row-candidate θ_1 for the entry $S_{8,10}$ is given as

  $$\theta_1 = y^2 + \alpha^7 y + \alpha^2 x^2 + \alpha^6 xy.$$

  For this entry we can also find a column-candidate η_1 as

  $$\eta_1 = x^2 y + 1 + \alpha^7 x + \alpha y + \alpha^7 x^2 + \alpha xy + y^2 + x^3.$$

  Using these candidates we find an estimate for $S_{8,10}$ as $S_{8,10} = \alpha^3$. By symmetry, this is also an estimate for the entry $S_{10,8}$.

- For the entry $S_{9,9}$ we can compute a row- and column-candidate, as

$$\theta_2 = x^3 + \alpha + \alpha^2 x + \alpha^5 y + \alpha^6 x^2 + \alpha^6 xy + \alpha^4 y^2$$
$$\eta_2 = x^3 + \alpha^5 + \alpha^6 x + \alpha y + \alpha^2 x^2 + \alpha^2 xy + y^2$$

  Using these candidates, we can compute an estimate for the entry $S_{9,9}$ as $S_{9,9} = 0$.

Using this estimates we compute votes for the unknown syndrome $e \cdot \varphi_{18}$. From the entries $S_{8,10}$ and $S_{10,8}$ we obtain the vote α^3, while from the entry $S_{9,9}$ we compute the vote α^2. Obviously one of the two votes must we wrong.

4.4.6 Majority Voting

Suppose that all entries in the syndrome matrix of order up to $s-1$ are known. Initially entries up to order $2t+2g-1$ are known, so $s \geq 2t+2g$. We can now compute test entries for the unknown entries $S_{r,r'}$, for $r+r'=s$, by the method described above. From each of the test entries we compute a vote for the unknown syndrome $e \cdot \varphi_s$. It is a remarkable fact the majority of those estimates will be correct.

To show this, we first need a technical lemma.

Lemma 4.18. *Let $t \leq u \leq g+t$. Assume that there are no error locators θ, η with $\mu_A(\theta) < u$, $\mu_{A'}(\eta) < u$. Furthermore let $r' = 2g-1+t+u-r$.*

Let

$$p := \left| \left\{ r \,\middle|\, \begin{array}{l} \textit{both error locators } \theta, \eta \textit{ exist, with} \\ \mu_A(\theta) = r,\ \mu_{A'}(\eta) = r' \ \&\ u \leq r \leq 2g-1+t \end{array} \right\} \right|$$

and let

$$q := \left| \left\{ r \,\middle|\, \begin{array}{l} \textit{neither error locator } \theta, \eta \textit{ exist, with} \\ \mu_A(\theta) = r,\ \mu_{A'}(\eta) = r' \ \&\ u \leq r \leq 2g-1+t \end{array} \right\} \right|$$

Then

$$p - q \geq u - t.$$

Proof. Let $J := \{r \mid u \leq r \leq 2g-1+t\}$, and let

$$I := \{r \in J \mid \text{error locator } \theta,\ \mu_A(\theta) = r \text{ exists}\}$$

$$I' := \{r \in J \mid \text{error locator } \eta,\ \mu_{A'}(\eta) = r' \text{ exists}\}$$

then $p = |I \cap I'|$ and $q = |J \setminus (I \cup I')| = 2g + t - u - |I \cup I'|$, and

$$p - q = \underbrace{|I \cap I'| + |I \cup I'|}_{=|I|+|I'|} - 2g - t + u.$$

Now we estimate $|I|$ and $|I'|$. Let e be the error word received, and let $B_e := \sum_{e_l \neq 0} P_l$ be the divisor of error positions.

Let θ be an error locator with $u \leq \mu_A(\theta) \leq 2g - 1 + t$. Then

$$\theta \in \mathcal{L}(A + (2g - 1 + t - \underbrace{\deg(A)}_{=t})Q - B_e) =: L_1$$

By assumption, that there is no error locator of order less than u, that is

$$\{0\} = \mathcal{L}(A + (u - 1 - \underbrace{\deg(A)}_{=t})Q - B_e) \subseteq L_1$$

By assumption $wt(e) \leq t$, so $\deg(A + (2g-1)Q - B_e) \geq t + 2g - 1 - t = 2g - 1$, and therefore $\dim(L_1) \geq g$. Let $\{\theta_i\}$ be a special basis of L_1, with $\mu_{A-B_e}(\theta_i) = i$, for $i \leq 2g - 1 + t - \deg(B_e)$. Then $u \leq \mu_A(\theta_i) = i + \deg(B_e) \leq 2g - 1 + t$. So each of the θ_i is an element of I. Thus $|I| \geq g$.

Similarly $|I'| \geq g$ and so it follows that

$$p - q = |I| - |I'| + u - t - 2g \geq u - t.$$

□

Note that it is important for this lemma, that there exists no error locator of order less than u.

Theorem 4.19. *Let the entries of the syndrome matrix of order up to $s - 1$ be known, and assume that there is no error locator of order less than $s - 2g - t + 1$.*

The number of test entries $S_{r,r'}$ with $r + r' = s$ producing correct votes for the value $e \cdot \varphi_s$ exceeds those producing incorrect votes by at least $s - 2g - 2t + 1 > 0$. In particular there is at least one such test entry.

Proof. Let $u = s - 2g - t + 1$. We have already seen, that $s \geq 2g + 2t$, so $u > t$. All the entries in rows and columns of order up to $u-1$ are known, and there are no error locators θ, η with $\mu_A(\theta) < u$ or $\mu_{A'}(\eta) < u$. The numbers $r \geq u, r' \geq u$ with $r + r' = s$ span the interval $u \leq r, r' \leq u + 2g + t - 1$. Thus u, r, and r' satisfy the conditions of Lemma 4.18.

a) Suppose the test entries $S_{r,r'}$ produces an incorrect vote, then $e \cdot \theta\chi_{r'} \neq 0$, with θ the candidate used to compute the test entry. Assume that there exists any error locator θ', with $\mu_A(\theta') = r$. By Theorem 4.15 we obtain $e \cdot \theta'\chi_{r'} = e \cdot \theta\chi_{r'} \neq 0$. Thus there can not be an error locator θ' of order r. Similarly there can not be an error locator η' of order r'.

   Thus the number of incorrect votes is at most the number q of pairs (r, r') for which there are not error locators θ, η with $\mu_A(\theta) = r$ and $\mu_{A'}(\eta')$.

b) Now assume that there are error locators θ' and η', with $\mu_A(\theta) = r$ and $\mu_{A'}(\eta') = r'$. Then again by Theorem 4.15 $e \cdot \theta'\chi_{r'} = e \cdot \theta\chi_{r'} = 0$, and similarly $e \cdot \chi_{r'}\eta' = e \cdot \chi_{r'}\eta = 0$. Hence the test entry produces a correct vote, and the number of test entries producing correct votes is at least the number p of pairs for which there are error locators of both orders.

Hence the difference between the number of correct votes and the number of incorrect votes is at least $p - q$. By Lemma 4.18 we know that $p - q \geq u - t = s - 2g - 2t + 1 > 0$ as claimed. □

Example 4.20. We continue Example 4.17.

For the entries of order $s = 18$ we found three votes for the unknown syndrome $e \cdot \varphi_{18}$, namely α^2 (from the estimate for $S_{9,9}$), and α^3 (from the estimates for $S_{8,10}$ and $S_{10,8}$). From the theorem above we know, that the most frequently occuring vote is the correct one, that is $e \cdot \varphi_{18} = \alpha^3$. We now also know, that the estimate for the entry $S_{9,9}$ was wrong. We use the value just found for the syndrome $e \cdot \varphi_{18}$ to update that entry. The correct value is $S_{9,9} = \alpha^5$.

We now continue this processes. First we check, if we can find an error locator using the syndrome matrix computed so far. If this is still not the case, we compute additional entries. If we have computed all unknown entries, and still can not find an error locator, the decoding of the received word is not possible.

In this example we need entries up to order $s = 19$ to find an error locator θ. The syndrome matrix with entries up to order 19 is given in Figure 4.2. The error locator θ may be found as

$$\theta = \alpha^7 y + \alpha^2 x^2 + \alpha^6 xy + y^2.$$

The set of error locations may then be found as

$$M = \{1, 10, 14, 17, 21, 27\}.$$

Again we utilize Theorem 4.6 to build a linear system for the error values.

$$S = \begin{pmatrix}
\alpha & \alpha^6 & \alpha^5 & \alpha^6 & \alpha^3 & \alpha^5 & \alpha^3 & \alpha^6 & 0 \\
\alpha^6 & \alpha^6 & \alpha^3 & \alpha^3 & \alpha^6 & 0 & \alpha & \alpha^2 & \alpha^3 \\
\alpha^5 & \alpha^3 & \alpha^5 & \alpha^6 & 0 & 0 & \alpha^2 & \alpha^3 & \alpha^3 \\
\alpha^6 & \alpha^3 & \alpha^6 & \alpha & \alpha^2 & \alpha^3 & \alpha^3 & \alpha & \alpha^3 \\
\alpha^3 & \alpha^6 & 0 & \alpha^2 & \alpha^3 & \alpha^3 & \alpha & \alpha^3 & \alpha^3 \\
\alpha^5 & 0 & 0 & \alpha^3 & \alpha^3 & 0 & \alpha^3 & \alpha^3 & \alpha^6 \\
\alpha^3 & \alpha & \alpha^2 & \alpha^3 & \alpha & \alpha^3 & \alpha^5 & \alpha^2 & \mathsf{S}_{9,11} \\
\alpha^6 & \alpha^2 & \alpha^3 & \alpha & \alpha^3 & \alpha^3 & \alpha^2 & \mathsf{S}_{10,10} & \\
0 & \alpha^3 & \alpha^3 & \alpha^3 & \alpha^3 & \alpha^6 & \mathsf{S}_{11,9} & &
\end{pmatrix}$$

Figure 4.2: Final syndrome matrix in Example 4.20

4.4.7 Summary of the Duursma Algorithm

Consider the code $C_\Omega(D, G)$ defined on a curve of genus g. Assume that $\deg G = 2t + 2g - 1$.

Initialization – once for a given code

1. Choose a point Q, $Q \notin \mathrm{supp}(D)$. Typically we choose $Q \in \mathrm{supp}(G)$.

2. Choose a divisor A, distinct from D, with $\deg(A) = t$.
   Let $A' = D - A - (2g-1)Q$.

3. Choose special bases of $\mathcal{L}(G+gQ)$, $\mathcal{L}(A+(2g-1)Q)$, and $\mathcal{L}(A'+(2g-1)Q)$. Index the bases by their H-order.

$$\begin{aligned}
\mathcal{L}(G+gQ) &= \mathrm{span}\,\{\varphi_i \mid 0 \le i \le 3g+2t-1\} \\
\mathcal{L}(A+(2g-1)Q) &= \mathrm{span}\,\{\psi_i \mid 0 \le i \le t+2g-1\} \\
\mathcal{L}(A+(2g-1)Q) &= \mathrm{span}\,\{\chi_i \mid 0 \le i \le t+2g-1\}
\end{aligned}$$

4. Express the entries of the syndrome matrix as linear combinations of the basis of $\mathcal{L}(G+gQ)$, that is find scalars $\alpha_k^{i,j}$ such that

$$\psi_i\chi_j = \alpha_0^{i,j}\varphi_0 + \cdots + \alpha_{i+j}^{i,j}\varphi_{i+j} \quad \text{for } i+j \le 3g+2t-1$$

Decoding

Let w be the word received.

1. Set up the $(t+g) \times (t+g)$ syndrome matrix S, as
$$S_{i,j} = e \cdot \chi_i \psi_j.$$
Index the rows and columns by the orders of the basis functions ψ_i, χ_j. Compute the syndromes up to order $\deg(G) = 2g + 2t - 1$ from the received word, that is compute
$$S_{i,j} = w \cdot \chi_i \psi_j \quad \text{for } 0 \leq i + j \leq 2g + 2t - 1.$$
The remaining entries are initially left unknown.
Additionally compute the syndromes $e \cdot \varphi_i = w \cdot \varphi_i$ for $i \leq 2g + 2t - 1$.
If all the syndromes computes so far all equal zero, then w is a code word.

2. Suppose all entries up to order $s - 1$ are known, but the entries of order s are unknown. Then the rows and column of order less than $u = s - 2g - t + 1$ are known in the sense that all their entries are known.
Try to find an error locator $\theta \in \mathcal{L}(A + mQ)$, that is find a non-zero solution b of
$$\sum_{j<u} S_{j,k} b_j = 0, \quad k \leq 2g + t - 1.$$
If such an element exists, let $\theta = \sum_j b_j \psi_j$. Then θ is an error locator for e. Continue with Step 5.
Try to find an error locator $\eta \in \mathcal{L}(A' + m'Q)$, that is find a non-zero solution β of
$$\sum_{k<u} S_{j,k} \beta_k = 0, \quad j \leq 2g + t - 1.$$
If such an element exists, let $\eta = \sum_j \beta_j \chi_j$. Then η is an error locator for e. Continue with Step 5.
If $s = 3g + 2t$, and still none of the two systems has a solution, we can not decode w.

3. Estimate addition syndromes. For each pair r, r' with $r + r' = s$, we compute an estimate $S_{r,r'}$.

   Calculate a $(r, r'-1)$-row-candidate by solving

   $$\sum_{j<r} S_{j,k} b_j = S_{r,k}, \quad k < r'$$

   If a non-zero solution b exists, put $S_{r,r'} = \sum_{j<r} b_j S_{j,r}$.

   Similarly compute a $(r-1, r')$-columns-candidate by solving

   $$\sum_{k<r'} S_{j,k} \beta_k = S_{j,r'}, \quad j < r$$

   If a non-zero solution β exists, put $S_{r,r'} = \sum_{k<r'} \beta_k S_{r,k}$.

   If both estimates exist, they are unique and agree, and we call $S_{r,r'}$ a *test entry*. Otherwise we discard the estimate.

4. Majority voting

   For each test entry found in Step 3, compute a vote for $e \cdot \varphi_s$, that is compute

   $$e \cdot \varphi_s = \frac{1}{\alpha_s^{r,r'}} \left(S_{r,r'} - \sum_{k<s} \alpha_k^{r,r'} \cdot e \cdot \varphi_k \right).$$

   The correct value for $e \cdot \varphi_s$ is the one most frequently occuring. Use this value for calculate the entries $S_{r,r'}$ which gave no estimate, or an incorrect vote. Compute $S_{r,r'}$ as

   $$S_{r,r'} = e \cdot \psi_r \chi_{r'} = \sum_{0 \le k \le s} \alpha_k^{r,r'} \cdot e \cdot \varphi_k$$

   If additional rows or columns are known return to Step 2, else return to Step 3.

5. Find the error locations M

   $$M = \{1 \le i \le n \mid \theta(P_i) = 0\}$$

6. Find the error values e_i by solving the linear system

   $$\sum_{l \in M} \varphi_i(P_l) e_l = w \cdot \varphi_i, \quad i < s$$

Let $e_i = 0$ for $i \notin M$.

Then $e = (e_1, \ldots, e_n)$ is the error word, and $c = w - e$ is the code word searched for.

Remark. If we only consider one point codes, this algorithm can be simplified a little bit. Let us fix the one point Goppa code $C_\Omega(D,\ aQ)$, for an integer a, and a single point Q. Let the underlying curve be of genus g.

- First we fix $A = tQ$. Then also $A' = tQ$.
- All the basis occuring can be chosen as subsets of a single special basis $\{\varphi_0, \ldots, \varphi_{t+2g}\}$ of $\mathcal{L}((t+g)Q)$.
- If the H-order $\mu_A(\theta)$ for a rational function θ is finite, we can compute it easily as

$$\mu_A(\theta) = \mu_{A+A'}(\theta) = -\nu_Q(\theta).$$

- The syndrome matrix will be symmetric. So any computation on row-candidates for the entry $S_{i,j}$ is at the same time also valid for column-candidates of the entry $S_{j,i}$.

On two occasions I have been asked [by members of Parliament], 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.

C. Babbage

Chapter 5

A Tutorial Implementation

5.1 Summary of the Implementation

The purpose of this implementation is to allow the user to experiment with the codes presented above, in the environment of the Maple computer algebra system. It is distributed as a part of the CASA package, developed by the Research Institute for Symbolic Computation (RISC Linz) at the University of Linz under the direction of Franz Winkler [HHW99]. CASA may be obtain from the CASA home page

`http://www.risc.uni-linz.ac.at/software/casa`

or by writing to `casa@risc.uni-linz.ac.at`.

After initializing the package using the Maple call

```
with(casa);
```

the full functionality of the implementation is available. Because this is a tutorial implementation, many functions print extra information. With the assignment

```
infolevel['casa/finite'] := level;
```

the amount of extra information to be printed is controlled. Higher values of `level` give more information. A value of 0 disables any messages, except warnings an errors. A level of 9 prints (almost) all messages. Matrices are printed depending on their size, bigger matrices require a larger value of `level` to be printed.

Example 5.1.

```
> with(casa):
           |__|
           |  |         Welcome to CASA 2.4.3 for Maple V.5
     |   /\|  |/\
    /=\__|  []  |
   /           \_       Copyright (C) 1990-1999 by Research Institute
   |             \      for Symbolic Computation (RISC-Linz), the
   \   CASA 2.4.3 |     University of Linz, A-4040 Linz, Austria.
    |             |
   _|     |||     |     For help type '?casa' or '?casa,<topic>'.
 __/      |||     |_

> infolevel[`casa/finite`] := 9;
```

$$infolevel_{casa/finite} := 9$$

5.1.1 Finite Fields

The first step is to create a finite field $\mathbb{F}_q$ and a smooth plane curve over this field. Maple supports finite fields using `RootOf`, but this method is not very practical. So the first part of the implementation was to supply an easier interface to work with finite fields.

The maple call

```
Fq := finiteField(q);
```

prepares the necessary functions. The returned table then provides methods for conversion, and some other useful functions (e.g. normalization) in the context of finite fields.

You can think of the function `finiteField(q)` as a constructor of the class `finiteField`. The returned table `Fq` can then be interpreted as an instance of this class.

A detailed description of the methods supplied by this object can be found in the online documentation. The two most important methods are:

- `Fq['In'](x)`: This functions converts an element of the finite field to Maple representation. This is done by replacing all occurrences of `alpha` in `x` by the RootOf representation of the primitive element.

  To see, how the field is represented, and which element is takes as the primitive element, one can use `Fq['In'](alpha)`.

- `Fq['Out'](x)`: This function takes an element in RootOf representation, and tries to express it as a power of the primitive element `alpha`. If this is not possible (for example if `x` is a variable) it is returned unchanged.

Example 5.2.

```
> F16 := finiteField(16);
```

$$F16 := Field$$

```
> F16['In'](alpha);
```

$$\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^2+1$$

```
> a := F16['In'](alpha^4 + alpha + 1);
```

$$a := (\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^2+1)^4$$
$$+\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^2+2$$

```
> F16['N'](a);
```

$$\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^3+\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^2+1$$

```
> F16['Out'](a);
```

$$\alpha^{10}$$

```
>
```

5.1.2 Smooth Plane Curves over Finite Fields

In order to keep the data associated with a plane curve over a finite field together the type `'casa/finite/Curve'` is supported. The function

```
C := finiteCurve(f, Fq);
```

is used for initialization, it takes as arguments the polynomial `f` defining the curve, and the finite field `Fq` over which the curve is defined.

It is assumed that the curve `C` is smooth. The system does not test this.

- `Genus(C)`: This functions computes the genus of the curve `C`. The genus is computed using the Plücker formula.

- `RationalPoints(C)`: Finds the `Fq`-rational points of the curve `C`. As you can see in the example below, points in projective space are represented as Maple lists of length 3.

Example 5.3.

```
> H2 := finiteCurve( x^3 + y^2 + y, finiteField(4));
```

$$H2 := x^3 + y\,z^2 + y^2\,z$$

```
> Genus(H2);
```

$$1$$

```
> RationalPoints(H2);
```

$[[0, 0, 1], [0, 1, 1], [1, \alpha, 1], [1, \alpha^2, 1], [\alpha, \alpha, 1], [\alpha, \alpha^2, 1], [\alpha^2, \alpha, 1],$
$[\alpha^2, \alpha^2, 1], [0, 1, 0]]$

```
>
```

5.1.3 Rational Functions

For algebraic geometric codes, rational functions on a smooth plane curve are of importance. In the implementation rational functions are initialized by the call

```
phi := finiteRational(f, curve);
```

The result is of the type `casa/finite/Rational`. `f` will be homogenized if necessary. The most important functions to work with rational functions are implemented as:

- `RatOrder(P, phi)`: This function computes the order $\nu_{\mathtt{P}}(\mathtt{phi})$.
- `RatEv(phi, P)`: This functions evaluates the rational function `phi` at the point `P`.
- `RatMul(phi1, phi2)`: Multiplies the two rational functions `phi1` and `phi2`.

Example 5.4.

```
> phi := finiteRational( (y^3 + y^2)/x^3, H2);
```

$$\phi := \frac{y^3 + y^2\,z}{x^3}$$

```
> RatOrder( [0,0,1], phi);
```

$$3$$

```
> RatEv(phi, [0,1,1]);
```

$$1$$

```
>
```

5.1.4 Divisors

What we still need for constructing Goppa codes are divisors. A divisor is initialized using the function

```
Di := makeDivisor(Ps, nPs);
```

Where `Ps` is a list of the points of the divisor and `nPs` is a list of the corresponding multiplicities.

- `DivPretty(Di)`: This function is supplied for pretty printing a divisor as a string.
- `DivAdd(D1, D2)`, `DivSub(D1, D2)`: These two functions add/subtract two divisors.
- `DivLe(D1, D2)`, `DivGe(D1, D2)`: These functions are used for the comparison of two divisors. `DivLe` tests for `D1` $\leq$ `D2`, while `DivGe` tests for `D1` $\geq$ `D2`.
- `DivDeg(Di)`: This function computes the degree of a divisor.

Example 5.5.

```
> D1 := makeDivisor( [ [0,0,1], [0,1,1] ], [1,-2]);
                          D1 := Divisor
> DivPretty(D1);
                     "+1 (0:0:1) -2 (0:1:1) "
> D2 := makeDivisor( [ [0,0,1], [1,1,0] ], [2, 1]);
                          D2 := Divisor
> DivPretty(D2);
                     "+2 (0:0:1) +1 (1:1:0) "
> DivLe(D1, D2);
                              true
> DivPretty( DivAdd(D1, D2) );
                "+3 (0:0:1) -2 (0:1:1) +1 (1:1:0) "
> DivPretty( DivSub(D1, D2) );
                "-1 (0:0:1) -2 (0:1:1) -1 (1:1:0) "
> DivDeg(D1);
                               -1
>
```

For the construction of a basis $\mathcal{L}(G)$, only a heuristic is implemented. To construct a basis of $\mathcal{L}(G)$ only monomials are tested. This is sufficient for most of the examples considered here, especially this is enough for Reed-Solomon and Hermitian codes.

When a basis of $\mathcal{L}(G)$ is needed, this heuristic can be tried, and if it fails a basis must be supplied by the user. For an example of this see the example session on the Klein Quartic (A.3).

Also note, that this heuristic does not test, whether the functions found for the basis are linearly independent. This should be checked by the user. If G is of the form $G = aQ$, for an integer a and a single point Q, then it is often enough to compute the orders $\nu_P(\varphi)$ of the rational functions φ found by the heuristic at this point. If all functions take a different values, then they are linearly independed.

Example 5.6.

```
> Di := makeDivisor( [[0,1,0]], [4]):
> DivPretty(Di);
```

"+4 (0:1:0) "

```
> DivBasisL(Di, H2, "monomial");
```

$$\left[1, \frac{y}{z}, \frac{x}{z}, \frac{x^2}{z^2}\right]$$

```
>
```

5.1.5 Goppa Codes

Now we have all the tools to construct primary and functional Goppa codes. The implementation allows the construction of both types of codes, but only for the primary Goppa codes, a decoding algorithm is available.

The codes $C_{\mathcal{L}}(D, G)$ and $C_{\Omega}(D, G)$ are constructed using the functions `GoppaFunctional` and `GoppaPrimary` respectively. The arguments required are the curve C, and divisors D and G. It is not necessary to specify all arguments explicitly, for example it possible to choose the divisor D as all affine points of the curve C.

For decoding, the error locator used must be initialized separately using `GoppaPrepareSV` or `GoppaPrepareDu`. The first functions prepares a Skorobogatov-Vlăduţ error processor, while the second initializes a Duursma error processor.

Details may again be found in the online documentation, and in the examples following in the next section.

Appendix A

Example Sessions

To demonstrate the tutorial implementation, and to make the theory of algebraic geometric codes clearer, we include some example sessions.

In all the subsequent examples we first construct the code, and then encode a simple message a to obtain some code word c. Some random errors are added to the code word c, so we obtain a word w. Then one of the two decoding algorithms presented in Chapter 4, is used to find the the code word c and with that the message a.

A.1 Reed-Solomon Code

The first example session following, will be the Reed-Solomon code of length 15. We let

$$\begin{aligned} n &= 15 & t &= 3 \\ q &= n + 1 = 16 & d &= 2t + 1 = 7 \end{aligned}$$

We define the code over the line $y = 0$. The divisor G is taken as

$$G = -1\ (0 : 0 : 1) + (d - 1)\ (1 : 0 : 0)$$

The divisor D is taken as the remaining $\mathbb{F}_{16}$-rational points of the line.

For decoding, using the SV-algorithm, the divisor A is taken as

$$A = 3(1 : 0 : 0).$$

Another natural choice would be $A = -1(0 : 0 : 1) + 4(1 : 0 : 0)$.

The Maple output starts on page 77.

A.2 Hermitian Code

The second example is the Hermitian code, defined over the curve

$$H_4 = V(x^5 + y^4z + yz^4)$$

over the field $\mathbb{F}_{16}$.

This is a code of length $n = 4^3 = 64$. For this tutorial implementation this is about the maximal length useful. For longer codes the construction, and decoding take too much time.

The Maple output starts on page 81.

A.3 The Klein Quartic

As the last example we consider the Klein quartic over $\mathbb{F}_{16}$. The curve is defined by the equation

$$x^3y + y^3z + z^3x = 0.$$

It is easy to see, that this is a smooth curve of genus $g = 3$.

This curve is mainly interesting for educational purposes, as it has less $\mathbb{F}_q$-rational points than a comparable Hermitian curve. Also it does not have a single point at infinity, so we can not use a simple monomial basis to define the code.

We construct the code $C_\Omega(D, G)$, with $G = 11(0:0:1)$ and D the sum of the remaining 16 $\mathbb{F}_q$-rational points of the curve. With this choice we obtain a code of dimension $k = 7$ and designed distance $d_\Gamma = 7$.

In the example we show how a basis of $\mathcal{L}(14(0:0:1))$ by be found using the methods supplied by this implementation. A basis of this space can also be found using paper and pencil without much difficulties. The advantage of doing this using the computer algebra system is, that also a user not familiar with the computation of the order of a rational function can apply this method. But still some knowledge on projective geometry is needed.

The Maple output starts on page 92.

Maple Session – Reed-Solomon Code

```
> with(casa):
            |__|
            |  |          Welcome to CASA 2.4.2 for Maple V.5
     |    /\|  |/\
    /=\__|  []  |
   /            \_        Copyright (C) 1990-1999 by Research Institute
   |              \       for Symbolic Computation (RISC-Linz), the
   \   CASA 2.4.2 |       University of Linz, A-4040 Linz, Austria.
    |             |
   _|     |||     |       For help type '?casa' or '?casa,<topic>'.
__/       |||     |_
```

Construction of the Code

First we prepare the finite field, and take a look, how the primitive element is represented.

```
> F16 := finiteField(16):
> F16['In'](alpha);
```

$$\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^2+1$$

The second step is to prepare the curve. We already know, that this curve is smooth.

```
> H4 := finiteCurve(y, F16, [x,y,z]);
```

$$H4 := y$$

```
> g := Genus(H4);
```

$$g := 0$$

This variable contols the verbosity of the implementation.

```
> infolevel[`casa/finite`] := 9;
```

$$infolevel_{casa/finite} := 9$$

Now we can already construct the Reed-Solomon code as a primary Goppa code.

```
> G := makeDivisor( [ [0,0,1], [1,0,0] ], [-1, 6] );
> Code := GoppaPrimary(H4, "rest", G);
```

$$G := Divisor$$

$$G = \textit{-1 (0:0:1) +6 (1:0:0)}$$

$$D = [\alpha^5,0,1] + [\alpha^6,0,1] + [\alpha^7,0,1] + [\alpha^8,0,1] + [\alpha,0,1] + [\alpha^{11},0,1]$$
$$+ [\alpha^{10},0,1] + [\alpha^9,0,1] + [\alpha^2,0,1] + [\alpha^{14},0,1] + [\alpha^{13},0,1]$$
$$+ [\alpha^{12},0,1] + [\alpha^3,0,1] + [\alpha^4,0,1] + [1,0,1]$$

$$L(G) = \operatorname{span}\left(\frac{x}{z}, \frac{x^2}{z^2}, \frac{x^3}{z^3}, \frac{x^4}{z^4}, \frac{x^5}{z^5}, \frac{x^6}{z^6}\right)$$

$$H = \begin{bmatrix} \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha & \alpha^{11} & \alpha^{10} & \alpha^9 & \alpha^2 & \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^3 & \alpha^4 & 1 \\ \alpha^{10} & \alpha^{12} & \alpha^{14} & \alpha & \alpha^2 & \alpha^7 & \alpha^5 & \alpha^3 & \alpha^4 & \alpha^{13} & \alpha^{11} & \alpha^9 & \alpha^6 & \alpha^8 & 1 \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^3 & \alpha^3 & 1 & \alpha^{12} & \alpha^6 & \alpha^{12} & \alpha^9 & \alpha^6 & \alpha^9 & \alpha^{12} & 1 \\ \alpha^5 & \alpha^9 & \alpha^{13} & \alpha^2 & \alpha^4 & \alpha^{14} & \alpha^{10} & \alpha^6 & \alpha^8 & \alpha^{11} & \alpha^7 & \alpha^3 & \alpha^{12} & \alpha & 1 \\ \alpha^{10} & 1 & \alpha^5 & \alpha^{10} & \alpha^5 & \alpha^{10} & \alpha^5 & 1 & \alpha^{10} & \alpha^{10} & \alpha^5 & 1 & 1 & \alpha^5 & 1 \\ 1 & \alpha^6 & \alpha^{12} & \alpha^3 & \alpha^6 & \alpha^6 & 1 & \alpha^9 & \alpha^{12} & \alpha^9 & \alpha^3 & \alpha^{12} & \alpha^3 & \alpha^9 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} \alpha^{14} & \alpha^9 & \alpha^8 & \alpha^{10} & \alpha^9 & \alpha^{10} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha^{10} & \alpha^{10} & \alpha^{14} & \alpha^8 & \alpha^9 & \alpha^{12} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \alpha^3 & \alpha^{10} & \alpha^{10} & \alpha^6 & \alpha^6 & \alpha^2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & \alpha^{13} & \alpha & \alpha^7 & \alpha^5 & \alpha^4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \alpha & 1 & \alpha^7 & 1 & \alpha^5 & \alpha^{13} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \alpha^7 & \alpha^4 & 1 & \alpha & \alpha^5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \alpha^6 & \alpha^2 & \alpha^3 & \alpha^{10} & \alpha^6 & \alpha^{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \alpha^{10} & \alpha^{11} & \alpha^6 & \alpha^3 & \alpha^6 & \alpha^{10} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \alpha^8 & \alpha^{12} & \alpha^{10} & \alpha^{14} & \alpha^9 & \alpha^9 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[n, k, d_\Gamma] = 15, 9, 7$$

$$Code := Goppa/primary$$

Decoding using the SV-Algorithm

To decode, we must initialize an error locator. Since this code is defined on a curve of genus 0, we can apply the SV-algorithm.

Since we do not have a one point code here, we must specify the divisor A to use. The space L(A) is used to find an error locator. As the space of test functions L(G-A) is used.

```
> A := makeDivisor( [ [1,0,0] ], [3] );
> SV := GoppaPrepareSV(Code, A):
```

$$A := Divisor$$

$$A = +3\ (1{:}0{:}0)$$

$$C = -1\ (0{:}0{:}1)\ +3\ (1{:}0{:}0)$$

$$\psi = \left[1, \frac{x}{z}, \frac{x^2}{z^2}, \frac{x^3}{z^3} \right]$$

$$\chi = \left[\frac{x}{z}, \frac{x^2}{z^2}, \frac{x^3}{z^3} \right]$$

$$\psi * \chi = \begin{bmatrix} \frac{x}{z} & \frac{x^2}{z^2} & \frac{x^3}{z^3} & \frac{x^4}{z^4} \\ \frac{x^2}{z^2} & \frac{x^3}{z^3} & \frac{x^4}{z^4} & \frac{x^5}{z^5} \\ \frac{x^3}{z^3} & \frac{x^4}{z^4} & \frac{x^5}{z^5} & \frac{x^6}{z^6} \end{bmatrix}$$

The message to be encoded is generated by random.

```
> R1 := rand(1..16):
> a := [ seq( F16['seq'][R1()], i = 1.. Code['k']) ];
```

$$a := [1, \alpha^{14}, \alpha^{10}, \alpha^8, 0, \alpha^9, \alpha^4, \alpha^{11}, \alpha^4]$$

Encoding is preformed by a simple matrix multiplication.

```
> c := GoppaEncode(a, Code);
```

$$c := [\alpha^9, \alpha^3, \alpha^{14}, \alpha^{14}, \alpha^5, \alpha^{10}, \alpha^{14}, \alpha^{10}, 1, \alpha^4, \alpha^{11}, \alpha^8, 0, \alpha^9, \alpha^4]$$

Now some errors are added to the code word just found. Our code as a minimal distance of 7, so it is no problem, if we add 3 errors.

```
> w := evalm(c):
> R2 := rand(1 .. Code['n'] ):
> R3 := rand(0 .. 15):
> for i from 1 to 3 do  r:= R2(); w[r] := w[r] +
  alpha^R3(); od:
> print(w);
```

$$[\alpha^9, \alpha^3, \alpha^{14}, \alpha^{14} + \alpha^5, \alpha^5, \alpha^{10}, \alpha^{14}, \alpha^{10}, 1 + \alpha^6, \alpha^4 + \alpha^6, \alpha^{11}, \alpha^8, 0, \alpha^9, \alpha^4]$$

For deocing the received word, the code, and the error locator must be given.

Finally the system a G = c is solved to find the message sent out.

```
> a1 := GoppaDecode(w, Code, SV);
```

$$S = \begin{bmatrix} \alpha^{12} & \alpha^{14} & \alpha^{7} & \alpha^{6} \\ \alpha^{14} & \alpha^{7} & \alpha^{6} & 1 \\ \alpha^{7} & \alpha^{6} & 1 & \alpha^{7} \end{bmatrix}$$

$$Nullspace(S) = ([\alpha^{9}, \alpha^{6}, \alpha^{13}, 1])$$

$$error\ locations = \{4, 9, 10\}$$

$$Error\ Value\ System: \begin{bmatrix} \alpha^{8} & \alpha^{2} & \alpha^{14} \\ \alpha & \alpha^{4} & \alpha^{13} \\ \alpha^{9} & \alpha^{6} & \alpha^{12} \\ \alpha^{2} & \alpha^{8} & \alpha^{11} \\ \alpha^{10} & \alpha^{10} & \alpha^{10} \\ \alpha^{3} & \alpha^{12} & \alpha^{9} \end{bmatrix} * e = \begin{bmatrix} \alpha^{12} \\ \alpha^{14} \\ \alpha^{7} \\ \alpha^{6} \\ 1 \\ \alpha^{7} \end{bmatrix}$$

$$e = [\alpha^{5}, \alpha^{6}, \alpha^{6}]$$

$$c = [\alpha^{9}, \alpha^{3}, \alpha^{14}, \alpha^{14}, \alpha^{5}, \alpha^{10}, \alpha^{14}, \alpha^{10}, 1, \alpha^{4}, \alpha^{11}, \alpha^{8}, 0, \alpha^{9}, \alpha^{4}]$$

$$a1 := [1, \alpha^{14}, \alpha^{10}, \alpha^{8}, 0, \alpha^{9}, \alpha^{4}, \alpha^{11}, \alpha^{4}]$$

To check the computations performed above, we can compare the message sent out, and the result of the decoding algorithm.

```
> evalm( a1 - a );
```

$$[0, 0, 0, 0, 0, 0, 0, 0, 0]$$

```
>
```

Maple Session – Hermitian Code

```
> with(casa):
            |__|
            |  |          Welcome to CASA 2.4.2 for Maple V.5
     |   /\|  |/\
    /=\__|  []  |
   /           \_         Copyright (C) 1990-1999 by Research Institute
   |             \        for Symbolic Computation (RISC-Linz), the
   \   CASA 2.4.2 |       University of Linz, A-4040 Linz, Austria.
    |             |
   _|     |||     |       For help type '?casa' or '?casa,<topic>'.
 __/      |||     |_
```

Construction of the Code

First we prepare the finite field, and take a look, how the primitive element is represented.

```
> F16 := finiteField(16):
> F16['In'](alpha);
```

$$\mathrm{RootOf}(\_Z^4+\_Z^3+\_Z^2+\_Z+1)^2+1$$

The second step is to prepare the curve. We already know, that this curve is smooth.

```
> H4 := finiteCurve(x^5 + y^4 + y, F16);
```

$$H4 := x^5 + y\,z^4 + y^4\,z$$

```
> g := Genus(H4);
```

$$g := 6$$

This variable contols the verbosity of the implementation.

```
> infolevel[`casa/finite`] := 9;
```

$$infolevel_{casa/finite} := 9$$

Now we can already construct a primary Goppa code. The divisor D is taken as the sum of the affine Fq-rational points. The third argument specifies, that the divisor G should be taken as 25 times the single point at infinty.

The basic aprameters (n,k,d) of the code are printed at the end of the construction.

```
> Code := GoppaPrimary(H4, "affine", 25);
```

$$G = +25\ (0{:}1{:}0)$$

$$D = [\alpha^5, \alpha^8, 1] + [\alpha^8, \alpha^8, 1] + [\alpha^7, \alpha^{12}, 1] + [\alpha^9, \alpha^7, 1] + [\alpha^9, \alpha^{13}, 1]$$
$$+ [\alpha^9, \alpha^{14}, 1] + [\alpha^7, \alpha, 1] + [\alpha^8, \alpha^9, 1] + [\alpha^9, \alpha^{11}, 1] + [\alpha^7, \alpha^3, 1]$$
$$+ [\alpha^6, \alpha^{14}, 1] + [\alpha^6, \alpha^{13}, 1] + [\alpha^4, \alpha^4, 1] + [\alpha^6, \alpha^7, 1] + [\alpha^4, \alpha, 1]$$

$+[\alpha^4,\alpha^3,1]\ +[\alpha^8,\alpha^6,1]\ +[\alpha^8,\alpha^2,1]\ +[\alpha^5,\alpha^9,1]\ +[\alpha^6,\alpha^{11},1]$
$+[\alpha^4,\alpha^{12},1]\ +[0,1,1]\ +[\alpha^5,\alpha^2,1]\ +[\alpha^2,\alpha^8,1]\ +[\alpha^3,\alpha^7,1]$
$+[\alpha^2,\alpha^6,1]\ +[\alpha^2,\alpha^9,1]\ +[\alpha,\alpha,1]\ +[\alpha,\alpha^3,1]\ +[1,\alpha^{11},1]$
$+[\alpha^2,\alpha^2,1]\ +[\alpha^{12},\alpha^{14},1]\ +[1,\alpha^{13},1]\ +[0,0,1]\ +[1,\alpha^7,1]$
$+[\alpha^3,\alpha^{14},1]\ +[\alpha^{13},\alpha,1]\ +[0,\alpha^{10},1]\ +[\alpha^{13},\alpha^3,1]\ +[0,\alpha^5,1]$
$+[\alpha^{12},\alpha^7,1]\ +[\alpha^{12},\alpha^{13},1]\ +[\alpha^{11},\alpha^2,1]\ +[\alpha^3,\alpha^{13},1]\ +[\alpha^{11},\alpha^9,1]$
$+[\alpha^{11},\alpha^8,1]\ +[\alpha^{10},\alpha^4,1]\ +[\alpha^{10},\alpha^{12},1]\ +[\alpha^{14},\alpha^9,1]\ +[\alpha^{14},\alpha^6,1]$
$+[\alpha^{14},\alpha^8,1]\ +[1,\alpha^{14},1]\ +[\alpha,\alpha^4,1]\ +[\alpha,\alpha^{12},1]\ +[\alpha^{10},\alpha^3,1]$
$+[\alpha^3,\alpha^{11},1]\ +[\alpha^7,\alpha^4,1]\ +[\alpha^{10},\alpha,1]\ +[\alpha^{13},\alpha^4,1]\ +[\alpha^{14},\alpha^2,1]$
$+[\alpha^{13},\alpha^{12},1]\ +[\alpha^5,\alpha^6,1]\ +[\alpha^{12},\alpha^{11},1]\ +[\alpha^{11},\alpha^6,1]$

$$L(G) = \mathrm{span}\left(1, \frac{y}{z}, \frac{y^2}{z^2}, \frac{y^3}{z^3}, \frac{y^4}{z^4}, \frac{y^5}{z^5}, \frac{x}{z}, \frac{x\,y}{z^2}, \frac{x\,y^2}{z^3}, \frac{x\,y^3}{z^4}, \frac{x\,y^4}{z^5}, \frac{x^2}{z^2}, \frac{x^2\,y}{z^3}, \frac{x^2\,y^2}{z^4}, \frac{x^2\,y^3}{z^5}, \frac{x^3}{z^3}, \frac{x^3\,y}{z^4}, \frac{x^3\,y^2}{z^5}, \frac{x^4}{z^4}, \frac{x^4\,y}{z^5}\right)$$

$$[n, k, d_\Gamma] = 64, 44, 15$$

$$Code := Goppa/primary$$

Decoding using the Duursma-Algorithm

To decode, we must initialize an error locator. Since this code is defined on a curve of genus greater than zero, we should use the Duursma-algorithm.

The Duursma-algorithm is implemented here only for one point codes, so no extra arguments are necessary - it is sufficient to specify which code is used.

The functions used to define the syndrome matrix are printed below, also how to express the entries in the matrix as linear combinations of the basis functions of L(G + gQ).

```
> Du := GoppaPrepareDu(Code):
```

$$G + gQ = +31\ (0{:}1{:}0)$$

$$L(G{+}gQ) = \mathrm{span}\left(1, \frac{x}{z}, \frac{y}{z}, \frac{x^2}{z^2}, \frac{x\,y}{z^2}, \frac{y^2}{z^2}, \frac{x^3}{z^3}, \frac{x^2\,y}{z^3}, \frac{x\,y^2}{z^3}, \frac{y^3}{z^3}, \frac{x^4}{z^4}, \frac{x^3\,y}{z^4}, \frac{x^2\,y^2}{z^4}, \frac{x\,y^3}{z^4}, \frac{y^4}{z^4},\right.$$

$$\left. \frac{x^4\,y}{z^5}, \frac{x^3\,y^2}{z^5}, \frac{x^2\,y^3}{z^5}, \frac{x\,y^4}{z^5}, \frac{y^5}{z^5}, \frac{x^4\,y^2}{z^6}, \frac{x^3\,y^3}{z^6}, \frac{x^2\,y^4}{z^6}, \frac{x\,y^5}{z^6}, \frac{y^6}{z^6}, \frac{x^4\,y^3}{z^7} \right)$$

$\psi * \chi =$

$$\left[\begin{array}{ccccccccccccc}
1 & \frac{x}{z} & \frac{y}{z} & \frac{x^2}{z^2} & \frac{x y}{z^2} & \frac{y^2}{z^2} & \frac{x^3}{z^3} & \frac{x^2 y}{z^3} & \frac{x y^2}{z^3} & \frac{y^3}{z^3} & \frac{x^4}{z^4} & \frac{x^3 y}{z^4} & \frac{x^2 y^2}{z^4} \\
\frac{x}{z} & \frac{x^2}{z^2} & \frac{x y}{z^2} & \frac{x^3}{z^3} & \frac{x^2 y}{z^3} & \frac{x y^2}{z^3} & \frac{x^4}{z^4} & \frac{x^3 y}{z^4} & \frac{x^2 y^2}{z^4} & \frac{x y^3}{z^4} & \frac{x^5}{z^5} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} \\
\frac{y}{z} & \frac{x y}{z^2} & \frac{y^2}{z^2} & \frac{x^2 y}{z^3} & \frac{x y^2}{z^3} & \frac{y^3}{z^3} & \frac{x^3 y}{z^4} & \frac{x^2 y^2}{z^4} & \frac{x y^3}{z^4} & \frac{y^4}{z^4} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} \\
\frac{x^2}{z^2} & \frac{x^3}{z^3} & \frac{x^2 y}{z^3} & \frac{x^4}{z^4} & \frac{x^3 y}{z^4} & \frac{x^2 y^2}{z^4} & \frac{x^5}{z^5} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} & \frac{x^6}{z^6} & \frac{x^5 y}{z^6} & \frac{x^4 y^2}{z^6} \\
\frac{x y}{z^2} & \frac{x^2 y}{z^3} & \frac{x y^2}{z^3} & \frac{x^3 y}{z^4} & \frac{x^2 y^2}{z^4} & \frac{x y^3}{z^4} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} & \frac{x y^4}{z^5} & \frac{x^5 y}{z^6} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} \\
\frac{y^2}{z^2} & \frac{x y^2}{z^3} & \frac{y^3}{z^3} & \frac{x^2 y^2}{z^4} & \frac{x y^3}{z^4} & \frac{y^4}{z^4} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} & \frac{x y^4}{z^5} & \frac{y^5}{z^5} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} & \frac{x^2 y^4}{z^6} \\
\frac{x^3}{z^3} & \frac{x^4}{z^4} & \frac{x^3 y}{z^4} & \frac{x^5}{z^5} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} & \frac{x^6}{z^6} & \frac{x^5 y}{z^6} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} & \frac{x^7}{z^7} & \frac{x^6 y}{z^7} & \frac{x^5 y^2}{z^7} \\
\frac{x^2 y}{z^3} & \frac{x^3 y}{z^4} & \frac{x^2 y^2}{z^4} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} & \frac{x^5 y}{z^6} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} & \frac{x^2 y^4}{z^6} & \frac{x^6 y}{z^7} & \frac{x^5 y^2}{z^7} & \frac{x^4 y^3}{z^7} \\
\frac{x y^2}{z^3} & \frac{x^2 y^2}{z^4} & \frac{x y^3}{z^4} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} & \frac{x y^4}{z^5} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} & \frac{x^2 y^4}{z^6} & \frac{x y^5}{z^6} & \frac{x^5 y^2}{z^7} & \frac{x^4 y^3}{z^7} & \\
\frac{y^3}{z^3} & \frac{x y^3}{z^4} & \frac{y^4}{z^4} & \frac{x^2 y^3}{z^5} & \frac{x y^4}{z^5} & \frac{y^5}{z^5} & \frac{x^3 y^3}{z^6} & \frac{x^2 y^4}{z^6} & \frac{x y^5}{z^6} & \frac{y^6}{z^6} & \frac{x^4 y^3}{z^7} & & \\
\frac{x^4}{z^4} & \frac{x^5}{z^5} & \frac{x^4 y}{z^5} & \frac{x^6}{z^6} & \frac{x^5 y}{z^6} & \frac{x^4 y^2}{z^6} & \frac{x^7}{z^7} & \frac{x^6 y}{z^7} & \frac{x^5 y^2}{z^7} & \frac{x^4 y^3}{z^7} & & & \\
\frac{x^3 y}{z^4} & \frac{x^4 y}{z^5} & \frac{x^3 y^2}{z^5} & \frac{x^5 y}{z^6} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} & \frac{x^6 y}{z^7} & \frac{x^5 y^2}{z^7} & \frac{x^4 y^3}{z^7} & & & & \\
\frac{x^2 y^2}{z^4} & \frac{x^3 y^2}{z^5} & \frac{x^2 y^3}{z^5} & \frac{x^4 y^2}{z^6} & \frac{x^3 y^3}{z^6} & \frac{x^2 y^4}{z^6} & \frac{x^5 y^2}{z^7} & \frac{x^4 y^3}{z^7} & & & & &
\end{array}\right]$$

$$\mathrm{order}(\psi * \chi) = \begin{bmatrix} 0 & 4 & 5 & 8 & 9 & 10 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 4 & 8 & 9 & 12 & 13 & 14 & 16 & 17 & 18 & 19 & 20 & 21 & 22 \\ 5 & 9 & 10 & 13 & 14 & 15 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 8 & 12 & 13 & 16 & 17 & 18 & 20 & 21 & 22 & 23 & 24 & 25 & 26 \\ 9 & 13 & 14 & 17 & 18 & 19 & 21 & 22 & 23 & 24 & 25 & 26 & 27 \\ 10 & 14 & 15 & 18 & 19 & 20 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ 12 & 16 & 17 & 20 & 21 & 22 & 24 & 25 & 26 & 27 & 28 & 29 & 30 \\ 13 & 17 & 18 & 21 & 22 & 23 & 25 & 26 & 27 & 28 & 29 & 30 & 31 \\ 14 & 18 & 19 & 22 & 23 & 24 & 26 & 27 & 28 & 29 & 30 & 31 & \\ 15 & 19 & 20 & 23 & 24 & 25 & 27 & 28 & 29 & 30 & 31 & & \\ 16 & 20 & 21 & 24 & 25 & 26 & 28 & 29 & 30 & 31 & & & \\ 17 & 21 & 22 & 25 & 26 & 27 & 29 & 30 & 31 & & & & \\ 18 & 22 & 23 & 26 & 27 & 28 & 30 & 31 & & & & & \end{bmatrix}$$

$$\frac{x^5}{z^5} = \frac{y}{z} + \frac{y^4}{z^4}$$

$$\frac{x^6}{z^6} = \frac{x\,y}{z^2} + \frac{x\,y^4}{z^5}$$

$$\frac{x^5\,y}{z^6} = \frac{y^2}{z^2} + \frac{y^5}{z^5}$$

$$\frac{x^7}{z^7} = \frac{x^2\,y}{z^3} + \frac{x^2\,y^4}{z^6}$$

$$\frac{x^6\,y}{z^7} = \frac{x\,y^2}{z^3} + \frac{x\,y^5}{z^6}$$

$$\frac{x^5\,y^2}{z^7} = \frac{y^6}{z^6} + \frac{y^3}{z^3}$$

The message to be encoded is generated by random.

```
> R1 := rand(1..16):
> a := [ seq( F16['seq'][R1()], i = 1.. Code['k']) ];
```

$$a := [\alpha^8, \alpha^5, 1, \alpha^{10}, 0, \alpha^9, \alpha^4, \alpha^7, 1, 1, \alpha^{14}, \alpha^{10}, \alpha^8, 0, \alpha^9, \alpha^4, \alpha^{11}, \alpha^4, \alpha^{10}, \alpha^4, \alpha^5, \alpha^{13}, \alpha^9, \alpha^{12}, \alpha^{12}, \alpha^4, \alpha^{13}, \alpha^5, 1, \alpha^5, \alpha^6, \alpha^8, \alpha^3, \alpha^2, \alpha^9, 1, \alpha^{11}, \alpha^3, \alpha^4, \alpha^6, \alpha^{10}, \alpha, \alpha^4, 0]$$

Encoding is preformed by a simple matrix multiplication.

```
> c := GoppaEncode(a, Code);
```

$c := [\alpha^3, \alpha^7, \alpha^8, 1, \alpha^{13}, \alpha^{14}, \alpha^6, \alpha^{11}, \alpha, \alpha^{10}, \alpha^6, \alpha^6, \alpha^{10}, \alpha^{13}, \alpha^{13}, \alpha^5, \alpha^{13}, \alpha, \alpha^{13}, \alpha^5, \alpha^7, \alpha^8, \alpha^8, \alpha^{12}, \alpha^{14}, \alpha^5, \alpha^4, \alpha^{11}, \alpha^9, \alpha^6, \alpha^9, 1, \alpha^6, \alpha^5, 0, \alpha^{14}, \alpha^9, \alpha^3, \alpha^8, \alpha^{11}, \alpha^2, \alpha^3, \alpha^{10}, \alpha, \alpha^4, 0, 1, 1, \alpha^{10}, 1, \alpha^{10}, \alpha^7, 0, \alpha^4, \alpha^{12}, \alpha^4, 1, \alpha^4, \alpha^8, \alpha^9, \alpha^{10}, \alpha^4, \alpha^4, \alpha^{13}]$

Now some errors are added to the code word just found. Our code as a minimal distance of at least 15, so it is no problem, if we add 7 errors.

```
> w := evalm(c):
> R2 := rand(1 .. Code['n'] ):
> R3 := rand(0 .. 15):
> for i from 1 to 7 do  r:= R2(); w[r] := w[r] +
  alpha^R3(); od:
> print(w);
```

$[\alpha^3, \alpha^7, \alpha^8, 1, \alpha^{13}, \alpha^{14}, \alpha^6, \alpha^{11}, \alpha, \alpha^{10}, \alpha^6, \alpha^6, \alpha^{10}, \alpha^{13}, \alpha^{13}, \alpha^5, \alpha^{13}, \alpha, \alpha^{13}, \alpha^5, \alpha^7, \alpha^8, \alpha^8, \alpha^{12}, \alpha^{14}, \alpha^5, \alpha^4, \alpha^{11}, \alpha^9, \alpha^6, \alpha^9, 1, \alpha^6 + \alpha^5, \alpha^5 + \alpha^{12}, \alpha, \alpha^{14}, \alpha^9, \alpha^3, \alpha^8, \alpha^{11}, \alpha^2, \alpha^3, \alpha^{10}, \alpha, \alpha^4, \alpha^2, 1, 1 + \alpha^6, \alpha^{10}, 1, \alpha^{10} + \alpha^{12}, 2\,\alpha^7, 0, \alpha^4, \alpha^{12}, \alpha^4, 1, \alpha^4, \alpha^8, \alpha^9, \alpha^{10}, \alpha^4, \alpha^4, \alpha^{13}]$

For the decoding we would like to get even more information.

```
> infolevel[`casa/finite`] := 19;
```

$$infolevel_{casa/finite} := 19$$

For decoding the received word, the code, and the error locator must be given.

For the entries of order s=26 we really need the voting process. This is not the "typical" case - it may take some tries to find an example where not all error locator candidates give the same estiamte.

At the end we see the set of error locations, and the system used to find the error values.

Finally the system a G = c is solved to find the message sent out.

```
> a1 := GoppaDecode(w, Code, Du);
```

$$S = \begin{bmatrix}
\alpha^9 & \alpha^{12} & \alpha^6 & \alpha^{10} & \alpha^2 & \alpha^4 & \alpha & \alpha^7 & \alpha^{14} & \alpha^6 & \alpha^{11} & \alpha^9 & \alpha^7 \\
\alpha^{12} & \alpha^{10} & \alpha^2 & \alpha & \alpha^7 & \alpha^{14} & \alpha^{11} & \alpha^9 & \alpha^7 & 1 & \alpha^3 & \alpha^3 & \alpha^4 \\
\alpha^6 & \alpha^2 & \alpha^4 & \alpha^7 & \alpha^{14} & \alpha^6 & \alpha^9 & \alpha^7 & 1 & \alpha^7 & \alpha^3 & \alpha^4 & \alpha \\
\alpha^{10} & \alpha & \alpha^7 & \alpha^{11} & \alpha^9 & \alpha^7 & \alpha^3 & \alpha^3 & \alpha^4 & \alpha & \alpha & \alpha^2 & ?_{4,13} \\
\alpha^2 & \alpha^7 & \alpha^{14} & \alpha^9 & \alpha^7 & 1 & \alpha^3 & \alpha^4 & \alpha & \alpha^{13} & \alpha^2 & ?_{5,12} & ?_{5,13} \\
\alpha^4 & \alpha^{14} & \alpha^6 & \alpha^7 & 1 & \alpha^7 & \alpha^4 & \alpha & \alpha^{13} & \alpha^{11} & ?_{6,11} & ?_{6,12} & ?_{6,13} \\
\alpha & \alpha^{11} & \alpha^9 & \alpha^3 & \alpha^3 & \alpha^4 & \alpha & \alpha^2 & ?_{7,9} & ?_{7,10} & ?_{7,11} & ?_{7,12} & ?_{7,13} \\
\alpha^7 & \alpha^9 & \alpha^7 & \alpha^3 & \alpha^4 & \alpha & \alpha^2 & ?_{8,8} & ?_{8,9} & ?_{8,10} & ?_{8,11} & ?_{8,12} & ?_{8,13} \\
\alpha^{14} & \alpha^7 & 1 & \alpha^4 & \alpha & \alpha^{13} & ?_{9,7} & ?_{9,8} & ?_{9,9} & ?_{9,10} & ?_{9,11} & ?_{9,12} & \\
\alpha^6 & 1 & \alpha^7 & \alpha & \alpha^{13} & \alpha^{11} & ?_{10,7} & ?_{10,8} & ?_{10,9} & ?_{10,10} & ?_{10,11} & & \\
\alpha^{11} & \alpha^3 & \alpha^3 & \alpha & \alpha^2 & ?_{11,6} & ?_{11,7} & ?_{11,8} & ?_{11,9} & ?_{11,10} & & & \\
\alpha^9 & \alpha^3 & \alpha^4 & \alpha^2 & ?_{12,5} & ?_{12,6} & ?_{12,7} & ?_{12,8} & ?_{12,9} & & & & \\
\alpha^7 & \alpha^4 & \alpha & ?_{13,4} & ?_{13,5} & ?_{13,6} & ?_{13,7} & ?_{13,8} & & & & &
\end{bmatrix}$$

$$e * \phi = [\alpha^9, \alpha^{12}, \alpha^6, \alpha^{10}, \alpha^2, \alpha^4, \alpha, \alpha^7, \alpha^{14}, \alpha^6, \alpha^{11}, \alpha^9, \alpha^7, 1, \alpha^7, \alpha^3, \alpha^4, \alpha, \alpha^{13}, \alpha^{11}, ?_{21}, ?_{22}, ?_{23}, ?_{24}, ?_{25}, ?_{26}]$$

estimating entries of order $s = 26$

no row-candidate for entry $_{4,13}$

no row-candidate for entry $_{5,12}$

no row-candidate for entry $_{6,11}$

$$\textit{row-candidate for } S_{7,9} = \frac{x^3}{z^3} + \alpha^{13} + \frac{\alpha^{12} x}{z}$$

col-candidate for $S_{7,9} =$

$$\frac{x\,y^2}{z^3} + \alpha^{10} + \frac{\alpha^{12} x}{z} + \frac{\alpha^{14} y}{z} + \frac{\alpha^7 x^2}{z^2} + \frac{\alpha\, x\, y}{z^2} + \frac{\alpha^{12} y^2}{z^2} + \frac{x^3}{z^3} + \frac{x^2 y}{z^3}$$

$$\textit{row, col-estimate for } S_{7,9} = \alpha^{10}, \alpha^{10}$$

$$\textit{row-candidate for } S_{8,8} = \frac{x^2 y}{z^3} + \alpha^2 + \frac{\alpha^2 x}{z} + \frac{\alpha^6 y}{z} + \frac{\alpha^{12} x^2}{z^2} + \frac{\alpha^4 x\, y}{z^2} + \frac{\alpha^2 y^2}{z^2} + \frac{x^3}{z^3}$$

$$\textit{col-candidate for } S_{8,8} = \frac{x^2 y}{z^3} + \alpha^2 + \frac{\alpha^2 x}{z} + \frac{\alpha^6 y}{z} + \frac{\alpha^{12} x^2}{z^2} + \frac{\alpha^4 x\, y}{z^2} + \frac{\alpha^2 y^2}{z^2} + \frac{x^3}{z^3}$$

$$\textit{row, col-estimate for } S_{8,8} = \alpha^{11}, \alpha^{11}$$

row-candidate for $S_{9,7} =$

$$\frac{x y^2}{z^3} + \alpha^{10} + \frac{\alpha^{12} x}{z} + \frac{\alpha^{14} y}{z} + \frac{\alpha^7 x^2}{z^2} + \frac{\alpha x y}{z^2} + \frac{\alpha^{12} y^2}{z^2} + \frac{x^3}{z^3} + \frac{x^2 y}{z^3}$$

$$\textit{col-candidate for } S_{9,7} = \frac{x^3}{z^3} + \alpha^{13} + \frac{\alpha^{12} x}{z}$$

$$\textit{row, col-estimate for } S_{9,7} = \alpha^{10}, \alpha^{10}$$

no col-candidate for entry $S_{11,6}$

no col-candidate for entry $S_{12,5}$

no col-candidate for entry $S_{13,4}$

$$\textit{votes } = ([\alpha^{10}, 2], [\alpha^{11}, 1])$$

$$\textit{most frequent vote } = \alpha^{10}$$

$$\textit{new entry } S_{4,13} = \alpha^{10}$$

$$\textit{new entry } S_{5,12} = \alpha^{10}$$

$$\textit{new entry } S_{6,11} = \alpha^{10}$$

$$\textit{new entry } S_{7,9} = \alpha^{10}$$

$$\textit{new entry } S_{8,8} = \alpha^{10}$$

$$\textit{new entry } S_{9,7} = \alpha^{10}$$

$$\textit{new entry } S_{11,6} = \alpha^{10}$$

$$\textit{new entry } S_{12,5} = \alpha^{10}$$

$$\textit{new entry } S_{13,4} = \alpha^{10}$$

estimating entries of order $s = 27$

no row-candidate for entry $_{5,13}$

no row-candidate for entry $_{6,12}$

$$\textit{row-candidate for } S_{7,10} = \frac{x^3}{z^3} + \alpha^{13} + \frac{\alpha^{12} x}{z}$$

$$\textit{col-candidate for } S_{7,10} = \frac{y^3}{z^3} + \alpha + \frac{x}{z} + \frac{x^2}{z^2} + \frac{\alpha^2 x y}{z^2} + \frac{\alpha^{10} y^2}{z^2} + \frac{x^3}{z^3} + \frac{x^2 y}{z^3} + \frac{x y^2}{z^3}$$

$$\textit{row, col-estimate for } S_{7,10} = \alpha^{10}, \alpha^{10}$$

$$\textit{no row-candidate for entry }_{8,9}$$

$$\textit{no col-candidate for entry } S_{9,8}$$

$$\textit{row-candidate for } S_{10,7} = \frac{y^3}{z^3} + \alpha + \frac{x}{z} + \frac{x^2}{z^2} + \frac{\alpha^2 x y}{z^2} + \frac{\alpha^{10} y^2}{z^2} + \frac{x^3}{z^3} + \frac{x^2 y}{z^3} + \frac{x y^2}{z^3}$$

$$\textit{col-candidate for } S_{10,7} = \frac{x^3}{z^3} + \alpha^{13} + \frac{\alpha^{12} x}{z}$$

$$\textit{row, col-estimate for } S_{10,7} = \alpha^{10}, \alpha^{10}$$

$$\textit{no col-candidate for entry } S_{12,6}$$

$$\textit{no col-candidate for entry } S_{13,5}$$

$$\textit{votes } = ([\alpha^{10}, 2])$$

$$\textit{most frequent vote } = \alpha^{10}$$

$$\textit{new entry } S_{5,13} = \alpha^{10}$$

$$\textit{new entry } S_{6,12} = \alpha^{10}$$

$$\textit{new entry } S_{7,10} = \alpha^{10}$$

$$\textit{new entry } S_{8,9} = \alpha^{10}$$

$$\textit{new entry } S_{9,8} = \alpha^{10}$$

$$\textit{new entry } S_{10,7} = \alpha^{10}$$

$$\textit{new entry } S_{12,6} = \alpha^{10}$$

$$\textit{new entry } S_{13,5} = \alpha^{10}$$

$$\textit{estimating entries of order } s = 28$$

$$\textit{estimating entries of order } s = 29$$

$$\textit{estimating entries of order } s = 30$$

$$S = \begin{bmatrix}
\alpha^9 & \alpha^{12} & \alpha^6 & \alpha^{10} & \alpha^2 & \alpha^4 & \alpha & \alpha^7 & \alpha^{14} & \alpha^6 & \alpha^{11} & \alpha^9 & \alpha^7 \\
\alpha^{12} & \alpha^{10} & \alpha^2 & \alpha & \alpha^7 & \alpha^{14} & \alpha^{11} & \alpha^9 & \alpha^7 & 1 & \alpha^3 & \alpha^3 & \alpha^4 \\
\alpha^6 & \alpha^2 & \alpha^4 & \alpha^7 & \alpha^{14} & \alpha^6 & \alpha^9 & \alpha^7 & 1 & \alpha^7 & \alpha^3 & \alpha^4 & \alpha \\
\alpha^{10} & \alpha & \alpha^7 & \alpha^{11} & \alpha^9 & \alpha^7 & \alpha^3 & \alpha^3 & \alpha^4 & \alpha & \alpha & \alpha^2 & \alpha^{10} \\
\alpha^2 & \alpha^7 & \alpha^{14} & \alpha^9 & \alpha^7 & 1 & \alpha^3 & \alpha^4 & \alpha & \alpha^{13} & \alpha^2 & \alpha^{10} & \alpha^{10} \\
\alpha^4 & \alpha^{14} & \alpha^6 & \alpha^7 & 1 & \alpha^7 & \alpha^4 & \alpha & \alpha^{13} & \alpha^{11} & \alpha^{10} & \alpha^{10} & \alpha^{12} \\
\alpha & \alpha^{11} & \alpha^9 & \alpha^3 & \alpha^3 & \alpha^4 & \alpha & \alpha^2 & \alpha^{10} & \alpha^{10} & \alpha^2 & \alpha^{13} & \alpha^4 \\
\alpha^7 & \alpha^9 & \alpha^7 & \alpha^3 & \alpha^4 & \alpha & \alpha^2 & \alpha^{10} & \alpha^{10} & \alpha^{12} & \alpha^{13} & \alpha^4 & ?_{8,13} \\
\alpha^{14} & \alpha^7 & 1 & \alpha^4 & \alpha & \alpha^{13} & \alpha^{10} & \alpha^{10} & \alpha^{12} & \alpha^{10} & \alpha^4 & ?_{9,12} & \\
\alpha^6 & 1 & \alpha^7 & \alpha & \alpha^{13} & \alpha^{11} & \alpha^{10} & \alpha^{12} & \alpha^{10} & \alpha^{13} & ?_{10,11} & & \\
\alpha^{11} & \alpha^3 & \alpha^3 & \alpha & \alpha^2 & \alpha^{10} & \alpha^2 & \alpha^{13} & \alpha^4 & ?_{11,10} & & & \\
\alpha^9 & \alpha^3 & \alpha^4 & \alpha^2 & \alpha^{10} & \alpha^{10} & \alpha^{13} & \alpha^4 & ?_{12,9} & & & & \\
\alpha^7 & \alpha^4 & \alpha & \alpha^{10} & \alpha^{10} & \alpha^{12} & \alpha^4 & ?_{13,8} & & & & &
\end{bmatrix}$$

$$e * \phi = [\alpha^9, \alpha^{12}, \alpha^6, \alpha^{10}, \alpha^2, \alpha^4, \alpha, \alpha^7, \alpha^{14}, \alpha^6, \alpha^{11}, \alpha^9, \alpha^7, 1, \alpha^7, \alpha^3, \alpha^4, \alpha, \alpha^{13}, \alpha^{11}, \alpha^{10}, \alpha^{10}, \alpha^{12}, \alpha^{10}, \alpha^{13}, ?_{26}]$$

$$\mathit{Nullspace(S)} = ([\alpha^{13}, \alpha^{12}, 0, 0, 0, 0, 1])$$

$$\mathit{theta} = \alpha^{13} + \frac{\alpha^{12} x}{z} + \frac{x^3}{z^3}$$

$$\mathit{error\ locations} = \{33, 34, 35, 36, 45, 46, 47, 48, 49, 50, 51, 52\}$$

$$\textit{Error Value System:}\ \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\alpha^{7} & \alpha^{7} & \alpha^{7} & \alpha^{7} & \alpha^{10} & \alpha^{10} & \alpha^{10} & \alpha^{10} & \alpha^{11} & \alpha^{11} & \alpha^{11} & \alpha^{11} \\
\alpha & \alpha^{3} & \alpha^{4} & \alpha^{12} & \alpha & \alpha^{3} & \alpha^{4} & \alpha^{12} & \alpha^{2} & \alpha^{6} & \alpha^{8} & \alpha^{9} \\
\alpha^{14} & \alpha^{14} & \alpha^{14} & \alpha^{14} & \alpha^{5} & \alpha^{5} & \alpha^{5} & \alpha^{5} & \alpha^{7} & \alpha^{7} & \alpha^{7} & \alpha^{7} \\
\alpha^{8} & \alpha^{10} & \alpha^{11} & \alpha^{4} & \alpha^{11} & \alpha^{13} & \alpha^{14} & \alpha^{7} & \alpha^{13} & \alpha^{2} & \alpha^{4} & \alpha^{5} \\
\alpha^{2} & \alpha^{6} & \alpha^{8} & \alpha^{9} & \alpha^{2} & \alpha^{6} & \alpha^{8} & \alpha^{9} & \alpha^{4} & \alpha^{12} & \alpha & \alpha^{3} \\
\alpha^{6} & \alpha^{6} & \alpha^{6} & \alpha^{6} & 1 & 1 & 1 & 1 & \alpha^{3} & \alpha^{3} & \alpha^{3} & \alpha^{3} \\
1 & \alpha^{2} & \alpha^{3} & \alpha^{11} & \alpha^{6} & \alpha^{8} & \alpha^{9} & \alpha^{2} & \alpha^{9} & \alpha^{13} & 1 & \alpha \\
\alpha^{9} & \alpha^{13} & 1 & \alpha & \alpha^{12} & \alpha & \alpha^{3} & \alpha^{4} & 1 & \alpha^{8} & \alpha^{12} & \alpha^{14} \\
\alpha^{3} & \alpha^{9} & \alpha^{12} & \alpha^{6} & \alpha^{3} & \alpha^{9} & \alpha^{12} & \alpha^{6} & \alpha^{6} & \alpha^{3} & \alpha^{9} & \alpha^{12} \\
\alpha^{13} & \alpha^{13} & \alpha^{13} & \alpha^{13} & \alpha^{10} & \alpha^{10} & \alpha^{10} & \alpha^{10} & \alpha^{14} & \alpha^{14} & \alpha^{14} & \alpha^{14} \\
\alpha^{7} & \alpha^{9} & \alpha^{10} & \alpha^{3} & \alpha & \alpha^{3} & \alpha^{4} & \alpha^{12} & \alpha^{5} & \alpha^{9} & \alpha^{11} & \alpha^{12} \\
\alpha & \alpha^{5} & \alpha^{7} & \alpha^{8} & \alpha^{7} & \alpha^{11} & \alpha^{13} & \alpha^{14} & \alpha^{11} & \alpha^{4} & \alpha^{8} & \alpha^{10} \\
\alpha^{10} & \alpha & \alpha^{4} & \alpha^{13} & \alpha^{13} & \alpha^{4} & \alpha^{7} & \alpha & \alpha^{2} & \alpha^{14} & \alpha^{5} & \alpha^{8} \\
\alpha^{4} & \alpha^{12} & \alpha & \alpha^{3} & \alpha^{4} & \alpha^{12} & \alpha & \alpha^{3} & \alpha^{8} & \alpha^{9} & \alpha^{2} & \alpha^{6} \\
\alpha^{14} & \alpha & \alpha^{2} & \alpha^{10} & \alpha^{11} & \alpha^{13} & \alpha^{14} & \alpha^{7} & \alpha & \alpha^{5} & \alpha^{7} & \alpha^{8} \\
\alpha^{8} & \alpha^{12} & \alpha^{14} & 1 & \alpha^{2} & \alpha^{6} & \alpha^{8} & \alpha^{9} & \alpha^{7} & 1 & \alpha^{4} & \alpha^{6} \\
\alpha^{2} & \alpha^{8} & \alpha^{11} & \alpha^{5} & \alpha^{8} & \alpha^{14} & \alpha^{2} & \alpha^{11} & \alpha^{13} & \alpha^{10} & \alpha & \alpha^{4} \\
\alpha^{11} & \alpha^{4} & \alpha^{8} & \alpha^{10} & \alpha^{14} & \alpha^{7} & \alpha^{11} & \alpha^{13} & \alpha^{4} & \alpha^{5} & \alpha^{13} & \alpha^{2} \\
\alpha^{5} & 1 & \alpha^{5} & 1 & \alpha^{5} & 1 & \alpha^{5} & 1 & \alpha^{10} & 1 & \alpha^{10} & 1 \\
1 & \alpha^{4} & \alpha^{6} & \alpha^{7} & \alpha^{12} & \alpha & \alpha^{3} & \alpha^{4} & \alpha^{3} & \alpha^{11} & 1 & \alpha^{2} \\
\alpha^{9} & 1 & \alpha^{3} & \alpha^{12} & \alpha^{3} & \alpha^{9} & \alpha^{12} & \alpha^{6} & \alpha^{9} & \alpha^{6} & \alpha^{12} & 1 \\
\alpha^{3} & \alpha^{11} & 1 & \alpha^{2} & \alpha^{9} & \alpha^{2} & \alpha^{6} & \alpha^{8} & 1 & \alpha & \alpha^{9} & \alpha^{13} \\
\alpha^{12} & \alpha^{7} & \alpha^{12} & \alpha^{7} & 1 & \alpha^{10} & 1 & \alpha^{10} & \alpha^{6} & \alpha^{11} & \alpha^{6} & \alpha^{11} \\
\alpha^{6} & \alpha^{3} & \alpha^{9} & \alpha^{12} & \alpha^{6} & \alpha^{3} & \alpha^{9} & \alpha^{12} & \alpha^{12} & \alpha^{6} & \alpha^{3} & \alpha^{9}
\end{bmatrix} * e = \begin{bmatrix}
\alpha^{9} \\ \alpha^{12} \\ \alpha^{6} \\ \alpha^{10} \\ \alpha^{2} \\ \alpha^{4} \\ \alpha \\ \alpha^{7} \\ \alpha^{14} \\ \alpha^{6} \\ \alpha^{11} \\ \alpha^{9} \\ \alpha^{7} \\ 1 \\ \alpha^{7} \\ \alpha^{3} \\ \alpha^{4} \\ \alpha \\ \alpha^{13} \\ \alpha^{11} \\ \alpha^{10} \\ \alpha^{10} \\ \alpha^{12} \\ \alpha^{10} \\ \alpha^{13}
\end{bmatrix}$$

$$e = [\alpha^{5}, \alpha^{12}, \alpha, 0, 0, \alpha^{2}, 0, \alpha^{6}, 0, 0, \alpha^{12}, \alpha^{7}]$$

$$c = [\alpha^{3}, \alpha^{7}, \alpha^{8}, 1, \alpha^{13}, \alpha^{14}, \alpha^{6}, \alpha^{11}, \alpha, \alpha^{10}, \alpha^{6}, \alpha^{6}, \alpha^{10}, \alpha^{13}, \alpha^{13}, \alpha^{5}, \alpha^{13}, \alpha, \alpha^{13}, \alpha^{5}, \alpha^{7}, \alpha^{8}, \alpha^{8}, \alpha^{12}, \alpha^{14}, \alpha^{5}, \alpha^{4}, \alpha^{11}, \alpha^{9}, \alpha^{6}, \alpha^{9}, 1, \alpha^{6}, \alpha^{5}, 0, \alpha^{14}, \alpha^{9}, \alpha^{3}, \alpha^{8}, \alpha^{11}, \alpha^{2}, \alpha^{3}, \alpha^{10}, \alpha, \alpha^{4}, 0, 1, 1, \alpha^{10}, 1, \alpha^{10}, \alpha^{7}, 0, \alpha^{4}, \alpha^{12}, \alpha^{4}, 1, \alpha^{4}, \alpha^{8}, \alpha^{9}, \alpha^{10}, \alpha^{4}, \alpha^{4}, \alpha^{13}]$$

$$a1 := [\alpha^{8}, \alpha^{5}, 1, \alpha^{10}, 0, \alpha^{9}, \alpha^{4}, \alpha^{7}, 1, 1, \alpha^{14}, \alpha^{10}, \alpha^{8}, 0, \alpha^{9}, \alpha^{4}, \alpha^{11}, \alpha^{4}, \alpha^{10}, \alpha^{4}, \alpha^{5}, \alpha^{13}, \alpha^{9}, \alpha^{12}, \alpha^{12}, \alpha^{4}, \alpha^{13}, \alpha^{5}, 1, \alpha^{5}, \alpha^{6}, \alpha^{8}, \alpha^{3}, \alpha^{2}, \alpha^{9}, 1, \alpha^{11}, \alpha^{3}, \alpha^{4}, \alpha^{6}, \alpha^{10}, \alpha, \alpha^{4}, 0]$$

To check the computations performed above, we can compare the message sent out, and the result of the decoding algorithm.

```
> evalm( a1 - a );
```

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
> 
```

Maple Session – Klein Quartic

```
> with(casa):
              |__|
              |  |           Welcome to CASA 2.4.2 for Maple V.5
        |   /\|  |/\
       /=\__|  []  |
      /           \_         Copyright (C) 1990-1999 by Research Institute
      |             \        for Symbolic Computation (RISC-Linz), the
      \   CASA 2.4.2 |       University of Linz, A-4040 Linz, Austria.
       |             |
      _|     |||     |       For help type '?casa' or '?casa,<topic>'.
   __/       |||     |_
```

```
> F16 := finiteField(16);
```

$$F16 := Field$$

```
> F16['In'](alpha);
```

$$\mathrm{RootOf}(\_Z^4 + \_Z^3 + \_Z^2 + \_Z + 1)^2 + 1$$

```
> curve := finiteCurve(x^3*y + y^3 + x, F16);
```

$$curve := x\,z^3 + x^3\,y + y^3\,z$$

```
> RationalPoints(curve);
```

$$[[0, 0, 1], [\alpha^3, \alpha^{10}, 1], [\alpha^5, \alpha^2, 1], [\alpha^5, \alpha^8, 1], [\alpha^5, \alpha^{10}, 1], [\alpha^6, \alpha^5, 1], [\alpha^7, \alpha^{12}, 1], [\alpha^9, \alpha^5, 1], [\alpha^{10}, \alpha, 1], [\alpha^{10}, \alpha^4, 1], [\alpha^{10}, \alpha^5, 1], [\alpha^{11}, \alpha^6, 1], [\alpha^{12}, \alpha^{10}, 1], [\alpha^{13}, \alpha^3, 1], [\alpha^{14}, \alpha^9, 1], [0, 1, 0], [1, 0, 0]]$$

For this curve there is not just a single point at infinty, so we can not use a simple monomial basis.

We construct a basis of L(aQ) for Q = (0:0:1). Functions of the form 1/x^i are a good guess for a basis of this space. But we must multiply by y^j to make sure, that there is no pole at the point (0:1:0).

```
> B0 := [ seq( seq( y^j / x^i, i = 0..6), j = 0..6) ]:
> infolevel[`casa/finite`] := 5:
```

We construct the basis up to order 14, using the functions choosen above. Later we only need functions up to order 14.

```
> G := makeDivisor( [ [0,0,1] ], [14] );
> B1 := DivBasisL(G, curve, B0);
```

$$G := Divisor$$

Functions found =

$$\left(1, \frac{z}{x}, \frac{z^2}{x^2}, \frac{z^3}{x^3}, \frac{z^4}{x^4}, \frac{y\,z}{x^2}, \frac{y\,z^2}{x^3}, \frac{y\,z^3}{x^4}, \frac{y\,z^4}{x^5}, \frac{y^2\,z}{x^3}, \frac{y^2\,z^2}{x^4}, \frac{y^2\,z^3}{x^5}, \frac{y^3\,z^2}{x^5}, \frac{y^4\,z^2}{x^6}\right)$$

```
Error, (in DivBasisL) Found too many basis functions
```

```
for divisor +14 (0:0:1)
expected 12, but found 14 functions.
```

Too many functions are found. This indicates, that the functions specified are not linearly independend. We now take the functions found, and compte the order at the point (0:0:1).

```
> B1 := map( finiteRational,
>   [1, z/x, z^2/(x^2), z^3/(x^3), z^4/(x^4), y*z/(x^2),
  y*z^2/(x^3), y*z^3/(x^4),    y*z^4/(x^5), y^2*z/(x^3),
  y^2*z^2/(x^4), y^2*z^3/(x^5), y^3*z^2/(x^5),
  y^4*z^2/(x^6)], curve);
```

$$B1 := \left[1, \frac{z}{x}, \frac{z^2}{x^2}, \frac{z^3}{x^3}, \frac{z^4}{x^4}, \frac{y\,z}{x^2}, \frac{y\,z^2}{x^3}, \frac{y\,z^3}{x^4}, \frac{y\,z^4}{x^5}, \frac{y^2\,z}{x^3}, \frac{y^2\,z^2}{x^4}, \frac{y^2\,z^3}{x^5}, \frac{y^3\,z^2}{x^5}, \frac{y^4\,z^2}{x^6}\right]$$

```
> map2( RatOrder, [0,0,1], B1);
```

$$[0, -3, -6, -9, -12, -5, -8, -11, -14, -7, -10, -13, -12, -14]$$

We must eliminate two functions and we see, that the last two functions in this list have an order already occuring. So we can simply delete them. The remaining functions all have different orders, so they must be linearly independent.

```
> B2 := B1[ 1.. 12];
```

$$B2 := \left[1, \frac{z}{x}, \frac{z^2}{x^2}, \frac{z^3}{x^3}, \frac{z^4}{x^4}, \frac{y\,z}{x^2}, \frac{y\,z^2}{x^3}, \frac{y\,z^3}{x^4}, \frac{y\,z^4}{x^5}, \frac{y^2\,z}{x^3}, \frac{y^2\,z^2}{x^4}, \frac{y^2\,z^3}{x^5}\right]$$

Now we can use this basis to construct the code.

```
> G := makeDivisor( [ [0,0,1] ], [11] );
> code := GoppaPrimary( curve, "rest", G, B2);
```

$$G := Divisor$$

$$G = +11\ (0{:}0{:}1)$$

$$\begin{aligned} D = {} & [\alpha^5, \alpha^2, 1] + [\alpha^3, \alpha^{10}, 1] + [\alpha^5, \alpha^8, 1] + [\alpha^6, \alpha^5, 1] + [\alpha^5, \alpha^{10}, 1] \\ & + [\alpha^7, \alpha^{12}, 1] + [\alpha^{10}, \alpha, 1] + [\alpha^9, \alpha^5, 1] + [\alpha^{10}, \alpha^5, 1] + [\alpha^{10}, \alpha^4, 1] \\ & + [\alpha^{11}, \alpha^6, 1] + [\alpha^{12}, \alpha^{10}, 1] + [\alpha^{13}, \alpha^3, 1] + [\alpha^{14}, \alpha^9, 1] + [0, 1, 0] \\ & + [1, 0, 0] \end{aligned}$$

$$L(G) = \mathrm{span}\left(1, \frac{z}{x}, \frac{z^2}{x^2}, \frac{z^3}{x^3}, \frac{y\,z}{x^2}, \frac{y\,z^2}{x^3}, \frac{y\,z^3}{x^4}, \frac{y^2\,z}{x^3}, \frac{y^2\,z^2}{x^4}\right)$$

$$[n, k, d_\Gamma] = 16, 7, 7$$

$$code := Goppa/primary$$

An prepare an error locator for this code, using the basis functions found above.

```
> Du := GoppaPrepareDu(code, B2):
```

$$G + gQ = +14\ (0{:}0{:}1)$$

$$L(G+gQ) = \operatorname{span}\left(1, \frac{z}{x}, \frac{y\,z}{x^2}, \frac{z^2}{x^2}, \frac{y^2\,z}{x^3}, \frac{y\,z^2}{x^3}, \frac{z^3}{x^3}, \frac{y^2\,z^2}{x^4}, \frac{y\,z^3}{x^4}, \frac{z^4}{x^4}, \frac{y^2\,z^3}{x^5}, \frac{y\,z^4}{x^5}\right)$$

$$\psi * \chi = \begin{bmatrix} 1 & \frac{z}{x} & \frac{y\,z}{x^2} & \frac{z^2}{x^2} & \frac{y^2\,z}{x^3} & \frac{y\,z^2}{x^3} \\ \frac{z}{x} & \frac{z^2}{x^2} & \frac{y\,z^2}{x^3} & \frac{z^3}{x^3} & \frac{y^2\,z^2}{x^4} & \frac{y\,z^3}{x^4} \\ \frac{y\,z}{x^2} & \frac{y\,z^2}{x^3} & \frac{y^2\,z^2}{x^4} & \frac{y\,z^3}{x^4} & \frac{y^3\,z^2}{x^5} & \frac{y^2\,z^3}{x^5} \\ \frac{z^2}{x^2} & \frac{z^3}{x^3} & \frac{y\,z^3}{x^4} & \frac{z^4}{x^4} & \frac{y^2\,z^3}{x^5} & \frac{y\,z^4}{x^5} \\ \frac{y^2\,z}{x^3} & \frac{y^2\,z^2}{x^4} & \frac{y^3\,z^2}{x^5} & \frac{y^2\,z^3}{x^5} & \frac{y^4\,z^2}{x^6} & \\ \frac{y\,z^2}{x^3} & \frac{y\,z^3}{x^4} & \frac{y^2\,z^3}{x^5} & \frac{y\,z^4}{x^5} & & \end{bmatrix}$$

$$\operatorname{order}(\psi * \chi) = \begin{bmatrix} 0 & 3 & 5 & 6 & 7 & 8 \\ 3 & 6 & 8 & 9 & 10 & 11 \\ 5 & 8 & 10 & 11 & 12 & 13 \\ 6 & 9 & 11 & 12 & 13 & 14 \\ 7 & 10 & 12 & 13 & 14 & \\ 8 & 11 & 13 & 14 & & \end{bmatrix}$$

$$\frac{y^3\,z^2}{x^5} = \frac{y\,z}{x^2} + \frac{z^4}{x^4}$$

$$\frac{y^4\,z^2}{x^6} = \frac{y^2\,z}{x^3} + \frac{y\,z^4}{x^5}$$

Finally we encode and decode a simple message.

```
> a := [ 1 $ 7];
```

$$a := [1, 1, 1, 1, 1, 1, 1]$$

```
> c := GoppaEncode(a, code);
```

$$c := [\alpha^{14}, \alpha^9, \alpha^9, 0, \alpha^{12}, \alpha^2, \alpha^8, \alpha^{10}, \alpha^8, 1, 1, 1, 1, 1, 1, 1]$$

```
> w := evalm( c + [ 0,0,0,1, 0,1,0,0, 0,0,0,0, 0,1,0,0 ]
  );
```

$$w := [\alpha^{14}, \alpha^9, \alpha^9, 1, \alpha^{12}, \alpha^2 + 1, \alpha^8, \alpha^{10}, \alpha^8, 1, 1, 1, 1, 2, 1, 1]$$

```
> a1 := GoppaDecode(w, code, Du);
```

$$S = \begin{bmatrix} 1 & \alpha^4 & \alpha^9 & \alpha^8 & 0 & \alpha^3 \\ \alpha^4 & \alpha^8 & \alpha^3 & \alpha^8 & \alpha^3 & \alpha^7 \\ \alpha^9 & \alpha^3 & \alpha^3 & \alpha^7 & ?_{3,5} & ?_{3,6} \\ \alpha^8 & \alpha^8 & \alpha^7 & ?_{4,4} & ?_{4,5} & ?_{4,6} \\ 0 & \alpha^3 & ?_{5,3} & ?_{5,4} & ?_{5,5} & \\ \alpha^3 & \alpha^7 & ?_{6,3} & ?_{6,4} & & \end{bmatrix}$$

$$e * \phi = [1, \alpha^4, \alpha^9, \alpha^8, 0, \alpha^3, \alpha^8, \alpha^3, \alpha^7, ?_{10}, ?_{11}, ?_{12}]$$

$$\textit{estimating entries of order } s = 12$$

$$\textit{votes} = ([\alpha, 1])$$

$$\textit{most frequent vote} = \alpha$$

$$\textit{estimating entries of order } s = 13$$

$$\textit{estimating entries of order } s = 14$$

$$S = \begin{bmatrix} 1 & \alpha^4 & \alpha^9 & \alpha^8 & 0 & \alpha^3 \\ \alpha^4 & \alpha^8 & \alpha^3 & \alpha^8 & \alpha^3 & \alpha^7 \\ \alpha^9 & \alpha^3 & \alpha^3 & \alpha^7 & \alpha^7 & \alpha^{11} \\ \alpha^8 & \alpha^8 & \alpha^7 & \alpha & \alpha^{11} & 0 \\ 0 & \alpha^3 & \alpha^7 & \alpha^{11} & 0 & \\ \alpha^3 & \alpha^7 & \alpha^{11} & 0 & & \end{bmatrix}$$

$$e * \phi = [1, \alpha^4, \alpha^9, \alpha^8, 0, \alpha^3, \alpha^8, \alpha^3, \alpha^7, \alpha, \alpha^{11}, 0]$$

$$\textit{Nullspace(S)} = ([\alpha^{12}, \alpha^4, \alpha^3, 1])$$

$$\textit{theta} = \alpha^{12} + \frac{\alpha^4 z}{x} + \frac{\alpha^3 y z}{x^2} + \frac{z^2}{x^2}$$

$$\textit{error locations} = \{4, 6, 7, 14\}$$

$$\text{Error Value System: } \begin{bmatrix} 1 & 1 & 1 & 1 \\ \alpha^9 & \alpha^8 & \alpha^5 & \alpha \\ \alpha^8 & \alpha^{13} & \alpha^{11} & \alpha^{11} \\ \alpha^3 & \alpha & \alpha^{10} & \alpha^2 \\ \alpha^7 & \alpha^3 & \alpha^2 & \alpha^6 \\ \alpha^2 & \alpha^6 & \alpha & \alpha^{12} \\ \alpha^{12} & \alpha^9 & 1 & \alpha^3 \\ \alpha & \alpha^{11} & \alpha^7 & \alpha^7 \\ \alpha^{11} & \alpha^{14} & \alpha^6 & \alpha^{13} \\ \alpha^6 & \alpha^2 & \alpha^5 & \alpha^4 \\ \alpha^{10} & \alpha^4 & \alpha^{12} & \alpha^8 \\ \alpha^5 & \alpha^7 & \alpha^{11} & \alpha^{14} \end{bmatrix} * e = \begin{bmatrix} 1 \\ \alpha^4 \\ \alpha^9 \\ \alpha^8 \\ 0 \\ \alpha^3 \\ \alpha^8 \\ \alpha^3 \\ \alpha^7 \\ \alpha \\ \alpha^{11} \\ 0 \end{bmatrix}$$

$$e = [1, 1, 0, 1]$$

$$c = [\alpha^{14}, \alpha^9, \alpha^9, 0, \alpha^{12}, \alpha^2, \alpha^8, \alpha^{10}, \alpha^8, 1, 1, 1, 1, 1, 1, 1]$$

$$a1 := [1, 1, 1, 1, 1, 1, 1]$$

[>

Bibliography

[CS97] D. A. Cox and B. Sturmfels (eds.). *Applications of Computational Algebraic Geometry*, volume 53 of *Proceedings of Symposia in Applied Mathematics*. AMS (1997).

[Duu93] I. M. Duursma. Majority Coset Decoding. *IEEE Transactions on Information Theory*, 39:1067–1070 (1993).

[FR93] G.-L. Feng and T. R. N. Rao. Decoding Algebraic-Geometric Codes up to the Designed Minimum Distance. *IEEE Transactions on Information Theory*, 39:37–45 (1993).

[Ful69] W. Fulton. *Algebraic Curves*. Mathematics Lecture Note Series. The Benjamin/Cummings Publishing Company (1969).

[FWRT94] G.-L. Feng, V. K. Wei, T. R. N. Rao, and K. K. Tzeng. Simplified Understanding and Efficient Decoding of a Class of Algebrac-Geometric Codes. *IEEE Transactions on Information Theory*, 40:981–991 (1994).

[GH91] M. Giusti and J. Heintz. Là détermination des points isolés et de la dimension d'une variété algébrique peut se faire en temps polynomial, pages 216–256. In: *Computational Algebraic Geometry and Commutative Algebra*, D. Eisenbud and L. Robbiano (eds.), Cambridge University Press (1991).

[Gop81] V. D. Goppa. Codes on algebraic curves. *Soviet Math. Dokl. (translation)*, pages 207–214 (1981).

[Gop88] V. D. Goppa. *Geometry and Codes*. Number 24 in Mathematics and Its Applications (Soviet Series). Kluwer Academic Publishers (1988).

[HHW99] R. Hemmecke, E. Hillgarter, and F. Winkler. The CASA System. In: *Handbook for Computer Algebra*, J. Grabmeier, E. Kalkofen, and V. Weispfenning (eds.), Springer-Verlag (1999).

[HI94] M.-D. Huang and D. Ierardi. Efficient Algorithms for the Riemann-Roch Problem and for Addition in the Jacobian of a Curve. *J. Symbolic Computation*, 18:519–539 (1994).

[HP95] T. Høholdt and R. Pellikaan. On the Decoding of Algebraic-Geometric Codes. *IEEE Transactions on Information Theory*, 41:1589–1614 (1995).

[JLJ+89] J. Justesen, K. J. Larsen, H. E. Jensen, A. Havemose, and T. Høholdt. Construction and Decoding of a Class of Algebraic Geometry Codes. *IEEE Transactions on Information Theory*, 35:811–821 (1989).

[JLJH92] J. Justesen, K. J. Larsen, H. E. Jensen, and T. Høholdt. Fast Decoding of Codes from Algebraic Plane Curves. *IEEE Transactions on Information Theory*, 38:111–119 (1992).

[LN94] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, revised edition (1994).

[LP98] R. Lidl and G. Pilz. *Applied Abstract Algebra*. Undergraduate Texts in Mathematics. Springer-Verlag, second edition (1998).

[LSH97] J. Little, K. Saints, and C. Heegard. On the structure of Hermitian codes. *Journal of pure and applied algebra*, 121:293–314 (1997).

[Mor91] C. Moreno. *Algebraic curves over finite fields*. Cambridge University Press (1991).

[Pre98] O. Pretzel. *Codes and Algebraic Curves*. Number 8 in Oxford lecture series in mathematics and its applications. Oxford Science Publications (1998).

[SJH95] S. Sakata, H. E. Jensen, and T. Høholdt. Generalized Berlekamp-Massey Decoding of Algebraic-Geometric Codes up to Half the Feng-Rao Bound. *IEEE Transactions on Information Theory*, 41:1762–1768 (1995).

[SJM+95] S. Sakata, J. Justesen, Y. Madelung, H. E. Jensen, and T. Høholdt. A Fast Decoding Method of AG Codes from Miura-Kamiya Curves C_{ab} up to Half the Feng-Rao Bound. *Finite Fields and their Applications*, 1:83–101 (1995).

[Sør91] A. B. Sørensen. *Rational points on hypersurfaces, Reed-Muller codes and algebraic geometric codes.* PhD thesis, University of Aarhus (1991).

[Sti93] H. Stichtenoth. *Algebraic Function Fields and Codes.* Springer-Verlag (1993).

[TVZ82] M. A. Tsfasman, S. G. Vlăduţ, and Th. Zink. Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound. *Math. Nachr.*, 109:21–28 (1982).

[vdW73] B. L. van der Waerden. *Einführung in die algebraische Geometrie.* Number 51 in Die Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen. Springer-Verlag, second edition (1973).

[VH97] C. Voss and T. Høholdt. An Explicit Construction of a Sequence of Codes Attaining the Tsfasman-Vladut-Zink Bound The First Steps. *IEEE Transactions on Information Theory*, 43:128 – 135 (1997).

[vL92] J. H. van Lint. *Introduction to Coding Theory.* Number 86 in Graduate Texts in Mathematics. Springer-Verlag, second edition (1992).

[vLvdG88] J. H. van Lint and G. van der Geer. *Introduction to Coding Theory and Algebraic Geometry.* DMV Seminar Band 12. Birkhäuser Verlag (1988).

[Win96] F. Winkler. *Polynomial Algorithms in Computer Algebra.* Texts and Monographs in Symbolic Computation. Springer-Verlag (1996).

Index