Lecture notes on coding theory

Raymond van Bommel

Curves over finite fields, Autumn 2017, Leiden

1 Introduction

When one agent tries to transfer information to another agent through a noisy channel, errors can occur. Error detection and correction is used to resolve this problem. These methods will be illustrated in the following simple examples.

Example 1 (parity bit)**.** Alice wants to send three bits (i.e., either 0 or 1 inside the finite field $\mathbb{F}_2$) to Bob. To these three bits, she adds one extra bit, such that the four bits add up to 0. For example, is she wants to send the message 100, then she actually sends 1001, where the extra bit 1 at the end is called the *parity bit*.

Now, suppose the second bit got flipped during the communication process and Bob receives the message 1101. Then he can recognise that something went wrong during the communication. Even if we tell Bob that only one bit has been flipped, he does not have enough data to actually know which one has been flipped. The best he could do is ask Alice to send her message again.

If two bits were flipped in the communication, Bob would actually not notice that something is off. We say, using this code, Bob can detect up to 1 bit error.

Exercise 2. In the situation of the previous example, suppose that, for each bit independently, there is a probability of 10% that this bit has been flipped during the transmission.

(a) Calculate the probability that Bob receives the message correctly.

(b) Calculate the probability that Bob receives an invalid message (i.e. a message for which the four bits do not add up to 0).

(c) Calculate the probability that Bob receives another message that is also valid (i.e. a message for which the four bits add up to 0, but which is not

Alice's message).

*(d) If Bob is receiving an invalid message, he will ask Alice to resend her message until Bob does receive a valid message. Calculate the probability that Bob receives Alice's message correctly.

Example 3 (repetition code)**.** This time Alice wants to send one bit to Bob. Alice will send the bit to Bob thrice. So she either sends the message 000 or 111 to Bob.

Now, suppose she wanted to send 000 and the second bit got flipped during the communication process. Then Bob receives the message 010. Now Bob could still guess that Alice's intended message was 000, assuming that it is very unlikely that two errors occured during the transmission.

Using this code, Bob could either detect up to 2 bit errors (if he does not care about correcting the error), or he could correct up to 1 bit error.

Exercise 4. In the situation of the previous example, suppose that, for each bit independently, there is a probability of 10% that this bit has been flipped during the transmission.

(a) Calculate the probability that, after correction, Bob receives the message correctly.

*(b) Suppose that Bob does not try to correct the error himself, but actually asks Alice to resend her message in case he does not receive either 000 or 111. Calculate the probability that Bob receives Alice's message correctly (possibly after multiple iterations).

Error detection and correction is used in many places in daily life. For example, 7-bit long ASCII characters are often appended with one parity bit. More complicated error correcting codes are used for communication on the internet, or on compact discs[1]. Another example is the so-called ISBN, which is used to uniquely identify books.

Example 5 (ISBN-10)**.** An old ISBN consists of 9 digits $x_1, \ldots, x_9$ and a single *check digit*. The check digit is determined by calculating

$$x_1 + 2x_2 + \ldots + 9x_9 \mod 11.$$

If this is between 0 and 9, this is the check digit, if this is 10, then the check digit is the letter 'X'. An example of a valid ISBN is 0345391802. We can

[1] A compact disc is an archaic optical data storage system designed in the 1980's. One disc was able to contain up to 74 or 80 minutes of music, depending on who you ask.

check that this ISBN is valid by calculating

$$\mathbf{0} + 2 \cdot \mathbf{3} + 3 \cdot \mathbf{4} + \ldots + 9 \cdot \mathbf{0} + 10 \cdot \mathbf{2} \equiv 0 \mod 11.$$

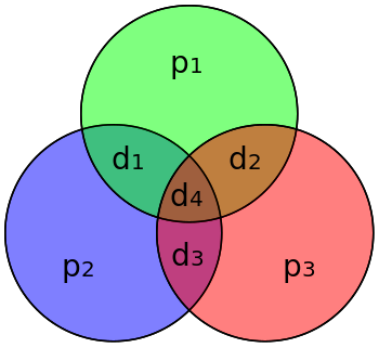
The system protects its users against the most two common errors made while copying these numbers:

- if you change any one digit in a valid ISBN, the resulting number will not be a valid ISBN;
- if you swap any two digits in a valid ISBN, the resulting number will not be a valid ISBN.

Exercise 6. Prove the latter two claims made in the previous example.

A more sophisticated example of an error correcting code is the so-called Hamming code.

Example 7 (Hamming code)**.** Alice wants to send Bob four bits d_1, d_2, d_3 and d_4. She puts these bits in a circle diagram as below. Now she chooses the parity bits p_1, p_2 and p_3 in such a way that the sum of the bits inside each circle is 0.



For example, if Alice wants to send the message 1010, then p_1, p_2 and p_3 must be 0, 0 and 1, respectively. So Alice sends the message 1010001 to Bob.

Now suppose that the third bit get flipped, and that Bob receives the message 1000001. Bob can then put the 7 numbers in the diagram and check that the sum in the three coloured circles. The blue and red circle will yield sum 1, while the green circle will yield sum 0. So now Bob knows that he has to flip the bit that is in both the blue and red, but not in the green circle, which is indeed the third bit.

This code allows Bob to correct any one error that occured in the transmission. So in order to correct up to 1 error in 4 bits, Bob has to send 3 bits extra.

This is much better than the repetition code in which Bob would have to send 8 bits extra (but could correct some more errors).

The outline of the rest of these lecture notes is as follows. First we will treat the basic theory behind codes. Then we will see some classical examples, before continuing with the examples using curves over finite fields. Finally, we will do actual calculations with these codes, using `Magma`.

2 Basic definitions and terminology

For our all our codes we will be working over an *alphabet*. The alphabet is assumed to be a finite field, whose order is called q. We will be working mostly with $\mathbb{F}_q^n$, the vector space of *words* of *length* n.

Definition 8. Let $a = (a_1, \ldots, a_n) \in \mathbb{F}_q^n$ and $b = (b_1, \ldots, b_n) \in \mathbb{F}_q^n$ be two words of lenght n. Then the *Hamming distance* between a and b is defined as

$$D(a, b) = |\{i \in \{1, \ldots, n\} : a_i \neq b_i\}|,$$

the number of positions in which a and b differ.

Exercise 9. Show that the Hamming distance defines a metric on $\mathbb{F}_q^n$.

In this course, we will only consider linear codes as defined below. There exists a more general notion of code, but we will not consider it for this course. Remark that the ISBN, as in Example 5 is technically not an example of this.

Definition 10. An $[n, k, d]$-*code* C is a linear subspace $C \subset \mathbb{F}_q^n$, such that $\dim_{\mathbb{F}_q}(C) = k \geq 1$ and

$$d = \min_{\substack{x,y \in C \\ x \neq y}} D(x, y).$$

The elements of C are called *codewords*. The parameters n, k and d are also called the *length*, the *dimension* and the *minimum distance* of C, respectively. Sometimes n, k and d are implicit and C is just called a code.

Exercise 11. Let C be an $[n, k, d]$-code. Prove that

$$d = \min_{x \in C \setminus \{0\}} D(0, x).$$

This might be useful in order to determine the minimum distance of a code in practice. You can try this yourself for the following examples.

Example 12. The parity bit code described in Example 1, given by

$$C = \{(x_1, x_2, x_3, x_4) \in \mathbb{F}_2^4 : x_1 + x_2 + x_3 + x_4 = 0\} \subset \mathbb{F}_2^4,$$

is a $[4, 3, 2]$-code.

Example 13. The repetition code described in Example 3, given by

$$C = \{(0, 0, 0), (1, 1, 1)\} \subset \mathbb{F}_2^3,$$

is a $[3, 1, 3]$-code.

Example 14. The Hamming code described in Example 7 is a $[7, 4, 3]$-code.

3 Construction of codes

There are several ways to construct codes from existing ones.

Definition 15. Let C_1 be an $[n_1, k_1, d_1]$-code and let C_2 be an $[n_2, k_2, d_2]$-code, then the *direct sum* of C_1 and C_2 is the $[n_1 + n_2, k_1 + k_2, \min(d_1, d_2)]$-code $C_1 \oplus C_2 \subset \mathbb{F}_q^{n_1+n_2}$.

Definition 16. Let C be an $[n, k, d]$-code and let

$$\langle \cdot, \cdot \rangle : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q : ((x_1, \ldots, x_n), (y_1, \ldots, y_n)) \mapsto x_1y_1 + x_2y_2 + \ldots + x_ny_n.$$

Then the *dual code* of C is the code

$$C^\vee = \{x \in \mathbb{F}_q^n : \forall y \in C : \langle x, y \rangle = 0\}.$$

Exercise 17. Describe the dual of the repetition code (see Ex. 3) and the parity bit code (see Ex. 1).

Exercise 18. Determine the length, dimension and minimum distance of the dual of the Hamming code (see Ex. 7).

Definition 19. Let V and W be vector spaces over a field K. Let $\{v_i\}_{i\in I}$ and $\{w_j\}_{j\in J}$ be bases of W. Then the *tensor product* of V and W is a vector space over K whose basis elements are formal symbols $v_i \otimes w_j$ (for all $(i,j) \in I \times J$).

Example 20. If $V = K^n$ and $W = K^m$, then $V \otimes W$ will be a vector space isomorphic to K^{mn}.

Remark 21. In Definition 19, it looks like the tensor product might depend on the choice of bases for V and W, but it does not. A more canonical construction (but probably a bit harder to grasp when you see it for the first time), would be to take the free vector space generated by the symbols $v \otimes w$ for all $(v,w) \in V \times W$, and then quotient out the relations

$$\begin{aligned}(v_1 + v_2) \otimes w_1 &= v_1 \otimes w_1 + v_2 \otimes w_1\\ v_1 \otimes (w_1 + w_2) &= v_1 \otimes w_1 + v_1 \otimes w_2\\ c(v_1 \otimes w_1) &= (cv_1) \otimes w_1\\ c(v_1 \otimes w_1) &= v_1 \otimes (cw_1)\end{aligned}$$

for all $v_1, v_2 \in V$, $w_1, w_2 \in W$ and $c \in K$.

Definition 22. If $v = \sum_{i\in I} c_i v_i$ and $w = \sum_{j\in J} d_j w_j$ are vectors in V and W (and all but finitely many c_i and d_j are zero), then $v \otimes w$ is the vector $\sum_{(i,j)\in I\times J} c_i d_j \cdot v_i \otimes w_j$ inside $V \otimes W$. This is consistent with the notation used in Remark 21.

Exercise 23.

(a) Construct an isomorphism between $V \otimes W$ and the space constructed in Remark 21.

(b) Prove that the map $\varphi : V \times W \to V \otimes W : (v,w) \mapsto v \otimes w$ is bilinear.

(c*) Prove that $V \otimes W$ and φ satisfy the following universal property: for any K-vector space T and any bilinear map $\rho : V \times W \to T$ there exists a unique linear map $\eta : V \otimes W \to T$, such that the following diagram commutes:

$$\begin{array}{ccccc} V \times W & \xrightarrow{\varphi} & V \otimes W & \overset{\eta}{\dashrightarrow} & T \\ & \searrow^{\rho} & & & \nearrow \end{array}$$

Definition 24. If $C_1 \subset \mathbb{F}_q^n$ and $C_2 \subset \mathbb{F}_q^m$ are codes, then their *tensor product* is $C_1 \otimes C_2 \subset \mathbb{F}_q^n \otimes \mathbb{F}_q^m \cong \mathbb{F}_q^{mn}$, using Definition 22 to define $C_1 \otimes C_2$ as a subspace of $\mathbb{F}_q^n \otimes \mathbb{F}_q^m$.

Exercise 25. Describe the tensor product of the parity bit code (see Ex. 1) with the Hamming code (see Ex. 7), i.e. give a basis for the code words and give its length, dimension and minimum distance.

4 Bounds on codes

Using an $[n, k, d]$-code, we need n bits to send a message that originally consisted of k bits. We can detect up to $d-1$ (or correct up to $\lfloor \frac{d-1}{2} \rfloor$) bit erros that might have occured during the transmission. The question is now: how large can d become, while keeping the difference $n-k$ as small as possible? The following theorem provides an upper bound.

Theorem 26 (Singleton bound)**.** *For any $[n, k, d]$-code C we have*

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

Proof. For $i = 1, \ldots, k-1$, consider the hyperplanes

$$V_i = \{(x_1, \ldots, x_n) \in \mathbb{F}_q^n : x_i = 0\} \subset \mathbb{F}_q^n.$$

Then $C \cap \bigcap_{i=1}^{k-1} V_i$ has dimension at least $k - (k-1) = 1$. It contains a non-zero vector v having at most $n - k + 1$ non-zero entries. Hence, we have the inequality $d \leq D(0, v) \leq n - k + 1$. □

Definition 27. An $[n, k, d]$-code is called a *maximal distance separable code*, or short-hand *MDS code*, if it satisfies $d = n - k + 1$.

Example 28. Both the parity bit code (see Example 1) and the repetition code (see Example 3) are MDS codes.

Exercise 29. Does there exist a $[7, 4, 4]$-code over $\mathbb{F}_2$?

There are also positive results about the existence of codes.

Theorem 30 (Gilbert-Varshamov bound)**.** *Let q be a prime power and let $n, k, d \in \mathbb{Z}_{\geq 1}$ satisfy*

$$\sum_{i=0}^{d-1} \binom{n}{i} \cdot (q-1)^i < q^{n-k+1}.$$

Then there exists an $[n, k, \geq d]$-code over $\mathbb{F}_q$.

Proof. Let ℓ be maximal such that there exists an $[n, \ell, \geq d]$-code C over $\mathbb{F}_q$. Suppose, on the contrary, that $\ell < k$. For each code word $x \in C$ consider the ball $B_{d-1}(x) \subset \mathbb{F}_q^n$ consisting of all words at Hamming distance at most $d-1$ from x. Then

$$|B_{d-1}(x)| = \sum_{i=0}^{d-1} \binom{n}{i} \cdot (q-1)^i.$$

The code C has q^ℓ code words, hence

$$\left| \bigcup_{x \in C} B_{d-1}(x) \right| \leq q^\ell \cdot \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i < q^{n+\ell-k+1},$$

by assumption. As $\ell \leq k-1$, the latter is less than or equal to q^n. In other words, there is a word $y \in \mathbb{F}_q^n$ such that $D(x, y) \geq d$ for each $x \in C$.

Now the claim is that $C + \mathbb{F}_q \cdot y$ is a $[n, \ell+1, \geq d]$-code, contradicting the maximality of ℓ. Hence, $\ell \geq k$, and then the existence of an $[n, k, \geq d]$-code follows immediately. □

Exercise 31. Prove the claim in the proof of Theorem 30.

There are several practical problems with this bound. On the one hand, it is quite far from sharp. But, more problematically, it does not provide codes that can be used in a practical way. Especially the decoding (i.e. the process of determining which code word is most likely after having received a message with errors) is not efficient for these codes. This is one of the reasons why people study geometric codes.

5 Geometric codes

In this section, we will discuss some codes that use curves over finite fields, but first we start off with a very classical code.

Example 32 (Reed-Solomon code)**.** Let $\mathbb{F}_q = \{a_1, \ldots, a_q\}$ be the finite field consisting of q elements. Let n and k be integers such that $1 \leq k \leq n \leq q$. Consider the vector space P_k of polynomials in $\mathbb{F}_q[x]$ of degree at most $k-1$. Let

$$\varphi : P_k \to \mathbb{F}_q^n : f \mapsto (f(a_i))_{i=1}^n$$

Now let C be the image of P_k inside $\mathbb{F}_q^n$. We claim that C is a $[n, k, n-k+1]$-code. This is an MDS code.

Exercise 33. Prove the claim in the example above.

Remark 34. A very naive decoding algorithm for this Reed-Solomon code would be as follows. Suppose you received the values $(x_i)_{i=1}^n$. Then, for each k-element subset

$$S = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\},$$

use Lagrange interpolation to find the polynomial f_S that goes through the points (a_{i_j}, x_{i_j}) for $j = 1, \ldots k$. For each such f_S, calculate the number of indices i, such that f_S passes through (a_i, x_i), and then pick the S for which this number is maximum.

As there are $\binom{n}{k}$ subsets S consisting of k elements, this decoding algorithm is not very efficient. For example, for $n = 255$ and $k = 249$ (which are numbers that are being used in practice), there are already more than 300 billion subsets to check.

In fact, using a slightly alternative interpretation of the encoding scheme, there does exist a decoding algorithm that runs in reasonable (i.e. polynomial) time.

Definition 35 (Goppa code)**.** Let $C/\mathbb{F}_q$ be a smooth projective curve. Let $P_1, \ldots, P_n$ be $\mathbb{F}_q$-rational points of C. Let D be a divisor on C, such that the P_i are not in the support of D (i.e. they do not occur in the sum). Then the *Goppa code* associated to (C, D) is the image of the map

$$\mathcal{L}(D) \to \mathbb{F}_q^n : f \mapsto (f(P_1), \ldots, f(P_n)),$$

where $\mathcal{L}(D) = \{f \in k(X) : \operatorname{div}(f) + D \geq 0\}$ is the Riemann-Roch space associated to D.

Example 36. If you take $C = \mathbb{P}^1_{\mathbb{F}_q}$ and $D = (k-1) \cdot (\infty)$, then you get the Reed-Solomon code descibred before.

Exercise 37. Verify this.

Example 38. Let $C \subset \mathbb{P}^2_{\mathbb{F}_2}$ be the smooth curve given by the homogeneous equation $x^3 + y^3 + z^3 = 0$. There are three rational points on this curve $P = (1 : 0 : 1)$, $Q = (0 : 1 : 1)$ and $R = (1 : 1 : 0)$. Consider the divisor $D = 3 \cdot P$. We are going to contruct the Goppa code associated to (C, D). To do this, we first need to calculate $\mathcal{L}(D)$.

Consider the affine patch $z \neq 0$ (also denoted by "$z = 1$" by Griffon). On this patch the curve is given by $X_z^3 + Y_z^3 + 1 = 0$. The function Y_z vanishes at the point P and it only vanishes once there. Hence, the functions, on this

affine patch, with at most a pole of order 3 are of the shape

$$\frac{F(X_z, Y_z)}{Y_z^3}$$

for some $F(X_z, Y_z) \in \mathbb{F}_2[X_z, Y_z]/(X_z^3 + Y_z^3 + 1)$.

Now we need to find those functions that do not have a pole outside of the affine patch $z \neq 0$. There are three points on C with $z = 0$: R, $(1 : \zeta_3 : 0)$ and $(1 : \zeta_3^2 : 0)$, where $\zeta_3 \in \mathbb{F}_4$ is a third root of unity.

Hence, we go to the affine patch $x \neq 0$. On this patch the curve is given by $1 + Y_x^3 + Z_x^3 = 0$, and the coordinates are related to X_z and Y_z by $X_z = 1/Z_x$ and $Y_z = Y_x/Z_x$. So, we are considering the function

$$\frac{F(X_z, Y_z)}{Y_z^3} = \frac{F(1/Z_x, Y_x/Z_x)Z_x^3}{Y_x^3}.$$

As $Y_x \neq 0$ for all three points in question, it suffices if $F(X_z, Y_z)$ would consists of terms of "degree 3" at most, i.e. F can only consist of the monomials $1, X_z, Y_z, X_z^2, X_zY_z, Y_z^2, X_z^3, X_z^2Y_z, X_zY_z^2$ and $Y_z^3 = X_z^3 + 1$.[a]

The function Y_z^3 also vanishes (with order 3) at the points $(\zeta_3 : 0 : 1)$ and $(\zeta_3^2 : 0 : 1)$. Hence F should also vanish with order at least 3 in these two points. This can only be achieved if F is a multiple of Y_z^3 or $X_z^2 + X_z + 1$, or a linear combination thereof. Hence, the Riemann-Roch space $\mathcal{L}(D)$ is generated by 1, $\frac{X_z^2+X_z+1}{Y_z^3} = \frac{x^2z+xz^2+z^3}{y^3}$ and $\frac{Y_z(X_z^2+X_z+1)}{Y_z^3} = \frac{x^2+xz+z^2}{y^2}$.

Now if we evaluate these three functions in Q and R, then we get 1, 1, 1, and 1, 0, 1, respectively. Hence, the Goppa code associated to (C, D) is the vector space spanned by $(1, 1), (1, 0), (1, 1)$ inside $\mathbb{F}_2^2$.

[a] This is not obvious at all. One could potentially allow extra factors of Z_x in the denominator, as long as it does not give rise to extra poles at any of the points three points. The only way to achieve this, is if the numerator is a multiple of $Y_x - 1$, $Y_x - \zeta_3$ and $Y_x - \zeta_3^2$, but then the numerator will be a multiple of $Y_x^3 - 1 = Z_x^3$.

The advantage of using the Goppa code, instead of the Reed-Solomon code, is that one can choose the field $\mathbb{F}_q$ somewhat smaller, as the curve $C/\mathbb{F}_q$ could potentially have up to $q + 1 + 2g\sqrt{q}$ points (approximately), cf. the Hasse-Weil bounds. When doing computations with error probabilities as we did in Exercise 2 and 4, one finds that longer codes (i.e. codes for which n is relatively large in comparison to q) can generally perform better than shorter codes.

Although the Goppa code is not MDS in general, it is still fairly close. One can use the bounds in the next lemma to improve upon the Gilbert-Varshamov bounds (see Theorem 30), see for example [vL99].

Lemma 39. *Assume that D is a divisor on C of degree at least $2g-1$ and at most $n-1$. Then the Goppa code G associated to (C,D) has dimension $\deg(D)-g+1$ and minimum distance $d \geq n-\deg(D)$.*

Proof. By Riemann-Roch $\mathcal{L}(D)$ has dimension $\deg(D)-g+1$. Hence, for the first claim it suffices to prove that $\alpha : \mathcal{L}(D) \to \mathbb{F}_q^n$ is injective. Any $f \in \ker \alpha$ is contained in $\mathcal{L}(D-P_1-\ldots-P_n)$, which is the zero space as $\deg(D-P_1-\ldots-P_n)$ is negative by assumption.

If $\alpha(f) \neq 0$ has distance d from 0, then f has at least $n-d$ zeros, say in $P_{i_1},\ldots,P_{i_{n-d}}$. Then $0 \neq f \in \mathcal{L}(D-P_{i_1}-\ldots-P_{i_{n-d}})$. Hence, the divisor $D-P_{i_1}\ldots-P_{i_{n-d}}$ must have non-negative degree and $\deg(D) \geq n-d$. □

6 Using Magma to study codes

In this section, we will try to construct the code from Example 38 using Magma. Documentation for Magma can be found online. A lot of the documentation has been put inside Magma. For example, you can see the different ways to construct a projective space by typing:

```
?ProjectiveSpace
?1
?2
```

First we define the projective space and the curve inside it:

```
P2<x,y,z> := ProjectiveSpace(GF(2), 2);
C := Curve(P2, x^3 + y^3 + z^3);
```

We can find the rational points:

```
L := RationalPoints(C);
print L;
P := L[1];
Q := L[2];
R := L[3];
```

Next one defines the divisor D and we calculate its Riemann-Roch space:

```
D: = 3*Divisor(P);
B := Basis(D);
print B;
```

The output,

```
[
    1,
    1/$.1^3*$.1^2 + 1/$.1^3*$.1 + 1/$.1^3,
    1/$.1^2*$.1^2 + 1/$.1^2*$.1 + 1/$.1^2
]
```

is completely incomprehensible. Here there are two symbols `$.1` occurring, and they are different elements of the function field of C. Fortunately, there is a function `ProjectiveFunction` that does give human-readable output. This gives exactly the basis we found in Example 38:

```
print [* ProjectiveFunction(f) : f in B *];
```

Now we can let `Magma` evaluate these functions in the other two points:

```
M := Matrix([ [ f(p) : f in B ] : p in [Q,R] ]);
print M;
```

Then we can generate a code in `Magma` (be aware, this is not the Goppa code, the matrix has been transposed to make it more interesting) and take random code word:

```
Co := LinearCode(M);
v := Random(Co);
print v;
```

`Magma` is also able to decode words using a (slow) algorithm for linear codes:

```
V := AmbientSpace(Co);
print V;
v := V![0,1,1];
print Decode(Co, v);
```

Exercise 40. Use `Magma` to construct an $[n, k, d]$-code using algebraic geometry with $k, d \geq 200$ and $n \leq 450$, just as we did above. Generate a random code word, randomly change 10 entries of the vector and try to decode the word.

Exercise 41. Use `Magma` again to repeat the previous exercise, but this time use the specific algebraic-geometric procedures that are implemented in `Magma`. Look up the functions `AGDualCode` and `AGDecode` in the `Magma` handbook.

References

[Magma] W. Bosma, J. Cannon, C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.* **24** (1997), no. 3–4, 235–265.

[Hin11] M. Hindry, *Arithmetics.* Translated from French. *Springer, London,* 2011.

[vL99] J. H. van Lint, *Introduction to coding theory.* Third edition. Graduate Texts in Mathematics, 86. *Springer-Verlag, Berlin,* 1999.

[NX09] H. Niederreiter, C. Xing, *Algebraic geometry in coding theory and cryptography. Princeton University Press, Princeton, NJ,* 2009.

[Sti09] H. Stichtenoth, *Algebraic function fields and codes.* Graduate Texts in Mathematics, 254. *Springer-Verlag, Berlin,* 2009.