# Cyclic

George McNinch

2024-03-06 and 2024-03-11

## Maximal distance-separable codes and the Singleton Bound

**Theorem (Singleton Bound)** Let $C$ be an $[n, k, d]_q$-code. Then

$$|C| \le q^{n-d+1}.$$

Put another way,

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

**Proof** Consider a subset $I \subset \{0, 1, \cdots, n-1\}$ with $|I| = n - (d-1)$.

If $v, w \in \mathbb{F}_q^n$ satisfy $v_i = w_i$ for each $i \in I$, then

$$\text{dist}(v, w) = \text{weight}(v - w) < d.$$

Since $d$ is the minimal distance between distinct codewords in $C$, if $v, w \in C$, then $v = w$.

In particular, if $v, w \in C$ with $v \neq w$, then $v_i \neq w_i$ for some $i \in I$.

Now, there are $q^{n-d+1}$ distinct $n - d + 1$ tuples of elements of $\mathbb{F}_q$, so we see that $|C| \le q^{n-d+1}$.

A code is said to be *maximal distance-separable* (MDS) if it *attains* the singleton bound. In other words, a linear MDS code satisfies

$$k = n - d + 1$$

i.e.

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

## Reed-Solomon codes

Important examples of linear MDS codes are the *Reed-Solomon* codes.

Let us enumerate the elements of $\mathbb{F}_q$:

$$\mathbb{F}_q = \{a_1, a_2, \cdots, a_q\}.$$

We fix a natural number $k \le q$. Then $q$ and $k$ determine (restricted) Reed-Solomon:

$$C = \{(f(a_1), f(a_2), \cdots, f(a_q)) \mid f \in \mathbb{F}_q[T], \text{deg}(f) < k\}.$$

Observe that

$$\mathbb{F}_q[T]_{<k} = \{f \in \mathbb{F}_q[T] \mid \text{deg}(f) < k\}$$

is a vector space over $\mathbb{F}_q$ of dimension $k$, since $1, T, \cdots, T^{k-1}$ is a *basis*.

Now, for any $a \in \mathbb{F}_q$, the evaluation mapping

$$(f \mapsto f(a)) : \mathbb{F}_q[T]_{<k} \to \mathbb{F}_q$$

is *linear*.

More generally, the mapping

$$\Phi : \mathbb{F}_q[T]_{<k} \to C$$

given by

$$f \mapsto (f(a_1), f(a_2), \cdots, f(a_q))$$

is linear. Moreover, if $k \leq q$, note that $f \in \ker \Phi$ means that $f$ has at least $q$ roots; in particular, since $f$ has degree $< q$, we may conclude that $f = 0$. This shows that $\Phi$ is *injective* and in particular implies that $\dim C = k$. Thus $C$ is a $[q, k]_q$-code. We now prove:

**Proposition** $C$ is a $[q, k, q - k + 1]_q$-code.

**Proof** We just need to argue that the minimal weight of a non-zero vector in $C$ is given by $d = q + 2 - k$. Let $c \in C$ be a non-zero vector with minimal weight $d$. There is a (unique) polynomial $f \in \mathbb{F}_q[T]_{<k}$ with

$$c = \Phi(f) = (f(a_1), f(a_2), \cdots, f(a_q)).$$

Since $c$ has weight $d$, we see that $f$ has $q - d$ roots.

On the other hand, since $\deg(f) < k$, $f$ has $\leq k - 1$ roots. This shows that $q - d \leq k - 1$, so that $d \geq q - k + 1$.

To show that $d = q - k + 1$, we must *exhibit* a polynomial $f \in \mathbb{F}_q[T]_{<k}$ for which the weight of $c = \Phi(f)$ is precisely $q - k + 1$. Thus, we just need to find a polynomial with exactly $k - 1$ roots. We just take

$$f = \prod_{i=1}^{k-1} (T - a_i) \in \mathbb{F}_q[T]_{<k}.$$

Then

$$c = \Phi(f) = (0, \cdots, 0, *, \cdots, *)$$

has exactly $k - 1$ zeros so that $\text{weight}(c) = q - k + 1$.

**Remark** The Proposition shows that the (restricted) Reed-Solomon code $C$ is an MDS code.

**Remark** We can also consider the (extended) Reed-Solomon code.

First recall that given $f \in \mathbb{F}_q[T]$ we can *homogenize* $f$ to produce $f^h \in \mathbb{F}_q[X, Y]$ where if

$$f = \sum_{i=0}^{N} a_i T^i$$

with $a_N \neq 0$ then

$$f^h = \sum_{i=0}^{N} a_i X^{N-i} Y^i.$$

Notice that $f = f^h(1, T)$. Recall that $\mathbb{P}^1_{\mathbb{F}_q} = \{(1 : a) \mid a \in \mathbb{F}_q\} \cup \{\infty = (0 : 1)\}$. Then $f^h(1, a) = f(a)$ and $f^h(0, 1) = a_N$. We view $f(\infty) = f^h(0, 1)$ as the "*value of $f$ at the point at infinity in $\mathbb{P}^1$*."

Now, the extended Reed-Solomon code is given by

$$\widetilde{C} = \{(f(a_1), \cdots, f(a_n), f(\infty)) \mid f \in \mathbb{F}_q[T]_{<k}\}.$$

Essentially the same argument as above shows that $\widetilde{C}$ is a $[q + 1, k, q + 1 - k + 1]_q = [q + 1, k, q + 2 - k]_q$-code, so that $\widetilde{C}$ is also an MDS code.

Indeed, we need to argue that the minimal weight of a non-zero vector in $\widetilde{C}$ is $q + 2 - k$. If $f(\infty) = 0$, we observe that $\deg(f) \leq k - 2$, so that $f$ has no more than $k - 2$ roots in $\mathbb{F}_q$; thus $f$ has no more than $k - 1$ roots in $\mathbb{P}^1$. On the other hand if $f(\infty) \neq 0$, then $\deg(f) \leq k - 1$ again shows that $f$ has no more than $k - 1$ roots in $\mathbb{P}^1$. With this observation, the proof is straightforward.

Finally, we note a nice feature of Reed-Solomon codes; there is an effective decoding algorithm for them. More precisely, we have the following:

**Theorem** There is a decoding algorithm for the restricted Reed-Solomon which corrects up to $\dfrac{n-k}{2}$ errors and completes in a number of steps which is polynomial in $q$.

**Proof** Suppose that we have received the vector

$$\mathbf{y} = (y_1, y_2, \cdots, y_q) \in \mathbb{F}_q^n.$$

We must find a polynomial $f \in \mathbb{F}_{[}T]_{<k}$ such that

$$(y_1, y_2, \cdots, y_q) = (f(a_1), f(a_2), \cdots, f(a_q)) + \mathbf{e}$$

for an *error vector* $\mathbf{e} \in \mathbb{F}_q^n$ with weight$(e) \leq \dfrac{q-k}{2}$.

Let $M_1 = \lfloor (q-k)/2 \rfloor$ and $M_2 = k + \lceil (q-k)/2 \rceil - 1$.

We consider an arbitrary polynomial $h$ of degree $\leq M_1$ and an arbitrary polynomial $g$ of degree $\leq M_2$; here by *arbitrary* we mean that we view the *coefficients* of these polynomials as *unknowns*.

We want to solve the equations

$$(\clubsuit) \quad g(a_j) - h(a_j)y_j = 0 \quad \text{for } j = 1, \cdots.$$

Write $g = \sum_{i=0}^{M_2} z_i T^i$ and $h = \sum_{i=0}^{M_1} w_i T^i$. First observe that $M_1 + M_2 = k + (q-k) - 1 = q - 1$. Thus there are $q + 1 = q - 1 + 2$ coefficients in the list $z_0, \cdots, z_{M_2}, w_0, \cdots, w_{M_1}$.

We see that $(\clubsuit)$ amounts to the linear equation

$$z_0 + z_1 a_j + z_2 {a_j}^2 + \cdots + z_{M_1} {a_j}^{M_2} + w_0 y_j + w_1 y_j a_j + w_2 y_j {a_j}^2 + \cdots w_{M_2} y_j {a_j}^{M_1} = 0$$

in the variables $z_i$ and $w_i$ for each $j = 1, \cdots, q$.

In other words, $(\clubsuit)$ amounts to a system of $q$ linear equations in the $q+1$ unknowns $z_0, \cdots, z_{M_2}, w_0, \cdots, w_{M_1}$.

Since $q + 1 > q$, this system of linear equations always has a non-zero solution. Moreover, *Gaussian elimination* will arrive at a solution after a number of operations that is $O(q^3)$; in particular, this algorithm is polynomial in $q$.

Now suppose that $z_0, \cdots, z_{M_2}, w_0, \cdots, w_{M_1} \in \mathbb{F}_q$ reflect a non-zero solution to $(\clubsuit)$, and set

$$g = \sum_{i=0}^{M_2} z_i T^i, \quad h = \sum_{i=0}^{M_1} w_i T^i \in \mathbb{F}_q[T].$$

Now, by assumption there is a polynomial $f$ for which $y_j = f(a_j)$ for at least $q - \lfloor (q-k)/2 \rfloor$ values of $j$. For these values of $j$, the element $a_j \in \mathbb{F}_q$ is a root of the polynomial $g(T) - h(T)f(T)$.

Note that since deg $h(T)f(T) \leq M_1 + k \leq M_2$, we have

$$\deg(g(T) - h(T)f(T)) \leq M_2.$$

Since $q - \lfloor (q-k)/2 \rfloor > k + \lceil (q-k)/2 \rceil - 1 = M_2$ it folllows that $g(T) - h(T)f(T)$ has more roots than its degree and is thus identically zero. Thus $h(T) \mid g(T)$ in $\mathbb{F}_q[T]$, and we find $f = \dfrac{g}{h} \in \mathbb{F}_q[T]$ as required.

## Implementation of Reed-Solomon decoding

Here is an implementation of the decoding algorithm in `Sage-Math`:

```
class ReedSolomon:
    def __init__(self,q,k):
        self.q = q
        self.F = GF(q)
        self.k = k
        self.t = floor((q-k)/2)  # we should be able to correct this many errors!

        self.V = VectorSpace(self.F,self.q)

        self.R = self.F['T']
        self.T = self.R.0

    def vectorize(self,f):
        if f.degree() < self.k:
            return self.V([ f(a) for a in self.F ])
        else:
            print("error")

    def decode(self,v):
        q = self.q
        k = self.k

        M1 = floor((1/2)*(q-k))
        M2 = k + ceil((1/2)*(q-k)) - 1

        # the elements of the field F
        elts = [ a for a in self.F ]

        def mkRow(i):
            return [ elts[i]^j for j in range(M2+1) ] + [-elts[i]^j * v[i] for j in range(M1+1) ]

        M = MatrixSpace(self.F,q,q+1).matrix([ mkRow(i) for i in range(q) ])

        # get a vector in the null space of V.
        coeffs = M.right_kernel().basis()[0]

        T = self.T

        # use the vector in the null space of V to build polynomials g and h
        g = sum([ coeffs[j] * T^j      for j in range(M2+1) ])
        h = sum([ coeffs[M2+1+j] * T^j for j in range(M1+1) ])

        # our solution is the quotient f = g/h
        # h divides g, so we "coerce" the ratio g/h to be element of self.R -- i.e. a polynomial
        # and not just a rational function of self.T
        f = self.R(g/h)

        # return the vector corresponding to f - this the decoded vector.
        return self.vectorize(f)


    def random_field_element(self):
        elts = [ a for a in self.F ]
        return choice(elts)

    def random_poly(self):
        # return a random polynomial of degree < k

        T = self.T
        return sum([ self.random_field_element()*T^j for j in range(self.k) ])

    def random_error(self):
        q = self.q
        k = self.k
        t = self.t

        # choose t random indices
        ll = [ choice(range(q)) for _ in range(t) ]

        # return a random error vector, according to the indices in ll
        return self.V([ self.random_field_element() if j in ll else 0 for j in range(q) ])


    def display_result(self,res):
        print("sent/received/error: \n")
        M = MatrixSpace(self.F,3,self.q).matrix([ res['sent'], res['received'], res['error'] ])
        print(M)
        print(f"\ndecoded correctly? {res['passed']}")


    def test(self):
        f = self.random_poly()
        v = self.vectorize(f)
        e = self.random_error()
        return { "sent": v,
                 "poly": f,
                 "received": v + e,
                 "error": e,
                 "passed": v == self.decode(v+e)
               }


    def run_tests(self,m):
        print(f"Reed Solomon code with q = {self.q} and k = {self.k}")
        print(f"Can correct t = {self.t} errors.")
        print(f"test decoding of {m} random vectors in C each with <={self.t} random errors\n")
        test_results = [ self.test() for _ in range(m) ]

        all_passed = all([ res['passed'] for res in test_results ])
        print(f"all passed? {all_passed}")
        print("------\n")

        for res in test_results:
            self.display_result(res)
            print("-----\n")
```

The `vectorize` method of the `ReedSolomon` class simply returns the vector

$$(f(a_1), \cdots, f(a_q))$$

when given the polynomial $f$.

The `decode` method of the class implements the decoding algorithm; given a vector in $\mathbb{F}_q^q$ it computes the polynomial $f$ as in the proof of the Theorem, and returns the vector

$$(f(a_1), \cdots, f(a_q)).$$

The remainder of the code is just implements some testing. We provide facility for generating random field elements, random polynomials, and random “error vectors” – see `random_field_element`, `random_poly`, `random_error`.

Then `test` method generates a random polynomial `f`, a random error `e`, and then it calls the decode method on the vector `c.vectorize(f) + e`. The test is *passed* if `c.vectorize(f) == c.decode(c.vectorize(f) + e)`

Here is some sample output

```
rs = ReedSolomon(19,5)
rs.run_tests(10)

Reed Solomon code with q = 19 and k = 5
Can correct t = 7 errors.
test decoding of 10 random vectors in C each with <=7 random errors

all passed? True
------

sent/received/error:

[10  1  6 13 12 14 13  5  7  0  5  7 12  9  8  2  5 14  9]
[ 3  1  6 13 12  3 13  5  7  0  5  4 14  7  8  2  5 12  9]
[12  0  0  0  0  8  0  0  0  0  0 16  2 17  0  0  0 17  0]

decoded correctly? True
-----

sent/received/error:

[ 1 17  9 13 14  3 15 15 12  2  6 13 18  3 13  4 14 11  7]
[13 17  9 13 14  3 15 15 12  2 17 13 18  9  2  4 11 11  5]
[12  0  0  0  0  0  0  0  0  0 11  0  0  6  8  0 16  0 17]

decoded correctly? True
-----

sent/received/error:

[ 5  0 13 16  5  0  7 18 11  7 13  3 13  8 15  9  8 16  4]
[ 5  1 13 16  5  0  7 10 11  7 13  3  9  8  3  9  8  1 14]
[ 0  1  0  0  0  0  0 11  0  0  0  0 15  0  7  0  0  4 10]

decoded correctly? True
-----

sent/received/error:

[ 3  9  5  4 17 15  5 11 17  5 12 16 12 12  7  5 12 13 10]
[ 3 14 10  4 17 15 14 11 17  5 12 16 12 12  7  5  5 13  1]
[ 0  5  5  0  0  0  9  0  0  0  0  0  0  0  0  0 12  0 10]

decoded correctly? True
-----

sent/received/error:

[18 15  1 11 11 12 13  1  8 16 14 17  9  0  7 16  1  0  1]
[18 15 10 11 12 12 13  1  8 18 17 17  9  0  7 16 17 10  1]
[ 0  0  9  0  1  0  0  0  0  2  3  0  0  0  0  0 16 10  0]
```

```
decoded correctly? True
-----

sent/received/error:

[ 2  7  7 14  3  7  3  7 17 13 14  2 17  5  8 12  4 10  0]
[ 2  7  7 14  3  7 17  7  7 13 14 18 17  6  8 13  4 10  0]
[ 0  0  0  0  0  0 14  0  9  0  0 16  0  1  0  1  0  0  0]

decoded correctly? True
-----

sent/received/error:

[ 4  3  3 13 15  2 16 15  6  7  9 14 16  1  4 14 12  9  8]
[ 9  3  3 13 15  2 16 15  6  6  9 14 16  1  4  9 12  9  8]
[ 5  0  0  0  0  0  0  0  0 18  0  0  0  0  0 14  0  0  0]

decoded correctly? True
-----

sent/received/error:

[14 17  1 11  1  5  4  2  7 12 14 14 17 13 15  2 14  0  8]
[11 17  9 11  1  7  4  8 15 12 14 14 17 13 15  2 14 14  8]
[16  0  8  0  0  2  0  6  8  0  0  0  0  0  0  0  0 14  0]

decoded correctly? True
-----

sent/received/error:

[ 6  3 13 10 15  1  7  5 14  5 15 14  0 18  8 14 13 10  0]
[ 6 13 13 11 15  1  7  5 14  9 15 11  0 18  1  4 13 14  0]
[ 0 10  0  1  0  0  0  0  0  4  0 16  0  0 12  9  0  4  0]

decoded correctly? True
-----

sent/received/error:

[15  0 14 15  7  2  1 13 17  0 14  5  3  8  9  3 14 17 14]
[15  0 13 15  7  2 12 13  7  0 11 18 12  8 17  3 14 17 14]
[ 0  0 18  0  0  0 11  0  9  0 16 13  9  0  8  0  0  0  0]

decoded correctly? True
-----
```

## Algebraic Geometry Codes

Considering *algebraic curves* defined over finite fields provides a vast generalization of the Reed-Solomon codes. We are going to *sketch* a few details of this construction. Note that our treatment is very rapid is mainly intended to communicate some hints about these codes. See e.g. (Ball 2020) for some more details in the context of *codes*. Further references with relatively minimal dependencies include (Goldschmidt 2003) and (Rosen 2002). Also see (Serre 2020).

Let $A = \mathbb{F}_q[X,Y,Z]/\langle \psi \rangle$ where $\psi$ is a homogeneous polynomial which is *absolutely irreducible*.

We consider the *field*

$$K = \left\{ \frac{g}{h} \mid g,h \in A \quad \text{homogeneous of the same degree} \right\}.$$

**Examples**

- $A = \mathbb{F}_q[X,Y] \simeq \mathbb{F}_q[X,Y,Z]/\langle Z \rangle$. In this case $K = \mathbb{F}_q\left(\frac{X}{Y}\right) \simeq \mathbb{F}_q(t)$ is the field of rational functions in a single variable $t$.
- Suppose that $p \neq 2$ and let $A = \mathbb{F}_q[X,Y,Z]/\langle Y^2 Z - X(X-Z)(X+Z) \rangle$. Then $K = \mathbb{F}_q\left(\frac{X}{Z}, \frac{Y}{Z}\right)$. Write $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$. Then $K = \mathbb{F}_q(x)(y)$ where $y^2 = x(x-1)(x+1)$. This is an *ellpitic curve* over $\mathbb{F}_q$.

Let $\overline{F_q}$ be an algebraic closure of $\mathbb{F}_q$ and let

$$X = \{(x:y:z) \in \mathbb{P}^3_{\overline{F_q}} \mid \psi(x,y,z) = 0\}.$$

Then $X$ is an *algebraic plane curve*, $A$ is an $\mathbb{F}_q$-form of its homogeneous coordinate ring, and $K = \mathbb{F}_q(X)$ is the field of $\mathbb{F}_q$-defined rational functions on $X$.

## Divisors and rational functions

By a *divisor* we mean a $\mathbb{Z}$-linear combination of points of $X$:

$$D = \sum_{x \in X} n_p[x] \quad \text{with } n_x \in \mathbb{Z} \text{ and almost all } n_x = 0.$$

One source of divisors is the divisor $\operatorname{div}(f)$ of a rational function $f \in K = \mathbb{F}_q(X)$: we have

$$\operatorname{div}(f) = \sum_{x \in C} \operatorname{ord}_x(f)[x]$$

where $\operatorname{ord}_x(f)$ denotes the *order of vanishing* of $f$ at $x \in X$. Note that $\operatorname{ord}_x(f) < 0$ precisely when $f$ has *a pole* at $x$.

The *degree* of a divisor $D = \sum_{p \in X} n_p$ is the integer $\deg(D) = \sum_{p \in X} n_p$.

An important result states that for any $f \in \mathbb{F}_q(X) = K$, we have

$$\deg(\operatorname{div}(f)) = 0.$$

Finally, a divisor $D$ is non-negative - written $D \geq 0$ - provided that $n_p \geq 0$ for each $p \in X$.

## Riemann-Roch spaces and the Riemann-Roch Theorem

For a divisor $D$ we consider

$$L(D) = \{f \in K \mid \operatorname{div}(f) + D \geq 0\}.$$

This is an $\mathbb{F}_q$-vector space, and it turns out to always be finite dimensional.

Observe that $L(D)$ consists of rational functions on $X$ whose poles are constrained by the divisor $D$.

The *genus* of an algebraic curve is a certain natural number $g$; it satisfies

$$g = \dim L(C), \quad \text{and} \quad 2g - 2 = \deg(C)$$

where $C$ is a so-called *canonical divisor*. This means that $C$ is a divisor of the form $C = \operatorname{div}(\omega)$ where $\omega \in \Omega_{K/\mathbb{F}_q}$ is a non-zero *differential form* on $X$.

If we worked over the complex numbers rather than over $\mathbb{F}_q$, then if the equation $\phi$ is *non-singular*, topologically, the curve $X$ is an *orientable surface* and hence is homeomorphic to either the sphere (case $g = 0$) or the connected sum of $g$ tori (case $g \geq 1$). So the genus $g$ "counts holes in the surface".

The important result here is:

**Theorem (Riemann-Roch)** If $\deg(D) \geq 2g - 1$ then

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

More generally, for any divisor $D$, we have

$$\dim L(D) = \deg D + 1 - g + \dim L(C - D).$$

## Codes from data on curves

Let $X$ be an algebraic curve over $\mathbb{F}_q$ as before.

By an $\mathbb{F}_q$-rational point of $X$ we mean a point $w = (x : y : z) \in X \subset \mathbb{P}^3_{\overline{\mathbb{F}_q}}$ with $x, y, z \in \mathbb{F}_q$.

Let us fix a divisor $D = \sum_{x \in C} n_x[x]$ on $X$ with the property that $n_x \neq 0$ implies that $x$ is $\mathbb{F}_q$-rational. Also, choose $n$ distinct $\mathbb{F}_q$-rational points $P_1, P_2, \cdots, P_n \in X$ for which $n_{P_i} = 0$ for $1 \leq i \leq n$.

**Theorem** Suppose that $n > \deg D > 2g - 1$. Then

$$C(D) = \{(f(P_1), \cdots, f(P_n)) \mid f \in L(D)\}$$

is a $[n, \deg D - g + 1, \geq n - \deg D]_q$-code.

# Bibliography

---

# Bibliography

Ball, Simeon. 2020. *A Course in Algebraic Error-Correcting Codes*. Compact Textbooks in Mathematics. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-41153-4.

Goldschmidt, David M. 2003. *Algebraic Functions and Projective Curves*. Vol. 215. Graduate Texts in Mathematics. Springer-Verlag, New York. https://doi.org/10.1007/b97844.

Rosen, Michael. 2002. *Number Theory in Function Fields*. Vol. 210. Graduate Texts in Mathematics. Springer-Verlag, New York. https://doi.org/10.1007/978-1-4757-6046-0.

Serre, Jean-Pierre. 2020. *Rational Points on Curves over Finite Fields*. Edited by Alp Bassa, Elisa Lorenzo García, Christophe Ritzenthaler, and René Schoof. Vol. 18. Documents Mathématiques (Paris) [Mathematical Documents (Paris)]. Société Mathématique de France, Paris.