# Week 10

# Golay codes. Classification of perfect codes

Version 2023-11-29. To accessible online version of this chapter

**Synopsis.** *One can explore cyclic codes of a given length over a given finite field in an attempt to find codes with interesting/useful properties. In fact, all types of codes we have considered so far will arise as cyclic codes. In this chapter, we define two new linear equivalence classes of codes called Golay codes. In our approach, these arise as cyclic codes, however, historically they were found in a different way. We give without proof a complete classification of perfect codes over alphabets of prime power size up to parameter equivalence, conjectured by Golay and proved by Tietäväinen and van Lint.*

Recall that:

- the only way to specify a general non-linear code in $\mathbb{F}_q^n$ is to list all the codewords, which consist of a total of $q^k \times n$ symbols;
- a linear code can be specified by a generator matrix, which has $k \times n$ entries;
- a cyclic code can be specified in an even more compact way — by giving its generator polynomial, which corresponds to a single codeword! We only need to specify $n-k$ coefficients of the generator polynomial (its degree is $n-k$ and its leading coefficient is $1$).

**Approach to searching for interesting/perfect/etc codes:**

Look for divisors of $x^n-1$ and hope that the cyclic codes they generate have a large minimum distance. **For example,** among the cyclic codes in $\mathbb{F}_2^7$, there are two perfect, Hamming codes (*Exercise*).

We will now describe two codes found by Marcel Golay in 1949. They are known as the *binary Golay code* $G_{23}$ and the *ternary Golay code* $G_{11}$, respectively.

## The binary Golay code $G_{23}$

In $\mathbb{F}_2[x]$, $x^{23}-1=(x+1)g(x)\overleftarrow{g}(x)$, where $g(x)=x^{11}+x^{10}+x^6+x^5+x^4+x^2+1$ and $\overleftarrow{g}(x)=x^{11}+x^9+x^7+x^6+x^5+x+1$.

*Exercise*: check this! You may use a computer algebra system but it is always instructive to multiply these out by hand.

**Definition: binary Golay code $G_{23}$**

Define a **binary Golay code** to be the cyclic code in $\mathbb{F}_2^{23}$ generated by $g(x)$, or any code linearly equivalent to it. (Any) binary Golay code is denoted $G_{23}$.

**Remark:** The cyclic code generated by $\overleftarrow{g}(x)$ is seen to be linearly equivalent to the cyclic code generated by $g(x)$; the linear equivalence is by writing all the codevectors backwards.

The above definition does not reflect how the code was originally found (see below) but suggests a practical way to construct a $G_{23}$ code if need be: factorise $x^{23}-1$ over $\mathbb{F}_2$ into irreducible factors (e.g., using a computer algebra system) and take one such factor of degree greater than $1$ to be the generator polynomial of a cyclic code.

**Theorem 10.1: parameters of $G_{23}$**

$G_{23}$ is a perfect $[23,12,7]_2$-code.

**Proof of Theorem 10.1 — part 1.** The code is binary ($q=2$) of length $n=23$ by construction. The dimension is $k=23-\deg g=12$.

It is easy to see that the weight of $G_{23}$ is **at most** $7$: indeed, the vector $\underline{g}\in G_{23}$ is $10101110001100000000000$, of weight 7, and so $w(G_{23})\le 7$.

It is more difficult to show that the weight of $G_{23}$ is exactly $7$. We will present a theoretical proof of this result using the extended code $G_{24}$, and will also show how to obtain the same result by a computer calculation.

We now prove that a $[23,12,7]_2$ -code is perfect. The Hamming bound for a binary code in in logarithmic form is $k\le n-\log_2\left(\binom{n}{0}+\cdots+\binom{n}{t}\right)$. Here $t=[(7-1)/2]=3$ so the argument of $\log_2$ is $1+\binom{23}{1}+\binom{23}{2}+\binom{23}{3}=1+23+23\times\frac{22}{2}+23\times\frac{22}{2}\times\frac{21}{3}=1+23(1+11+77)=2048$. One has $12=23-\log_2 2048$ hence the Hamming bound is attained.

The proof that $w(G_{23})=7$ will be given after a series of lemmas **(proof to be continued).**

## Binary vectors: extending, overlaps, weights and orthogonality

To proceed, we need a mini-toolbox containing tools for working with binary vectors.

A binary vector $\underline{v} = (v_1, v_2, \ldots, v_n)$ is **extended** to obtain the vector $\widehat{\underline{v}} = (v_1, \ldots, v_n, v_{n+1})$ where $v_{n+1} = v_1 + \cdots + v_n$ in $\mathbb{F}_2$. That is, a vector is extended by appending one bit so that the resulting vector has even weight. Explicitly, we may write

$$\widehat{\underline{v}} = \begin{cases} (\underline{v}, 0), & \text{if } w(\underline{v}) \text{ is even,} \\ (\underline{v}, 1), & \text{if } w(\underline{v}) \text{ is odd.} \end{cases}$$

By extending each vector in a given binary code, we obtain the *extended code*:

**Definition: extended code**

If $C$ is a binary linear code of length $n$, we define the **extended code** $\widehat{C}$ of length $n+1$ as $\{\widehat{\underline{c}} : \underline{c} \in C\}$.

The following notion is useful:

**Definition: overlap**

If $\underline{u}, \underline{v} \in \mathbb{F}_2^n$, the **overlap** of $\underline{u}$ and $\underline{v}$ is the number of positions $i$ such that $u_i = v_i = 1$.

It is easy to see that

$$w(\underline{u} + \underline{v}) = w(\underline{u}) + w(\underline{v}) - 2 \times \text{overlap}(\underline{u}, \underline{v}) \tag{10.1}$$

and

$$\underline{u} \cdot \underline{v} = 0 \quad \Longleftrightarrow \quad \text{overlap}(\underline{u}, \underline{v}) \text{ is even.} \tag{10.2}$$

It follows that

$$w(\underline{u}), w(\underline{v}) \text{ are multiples of } 4,\ \underline{u} \cdot \underline{v} = 0 \quad \Longrightarrow \quad w(\underline{u} + \underline{v}) \text{ is a multiple of } 4. \tag{10.3}$$

Indeed, by (10.1), $w(\underline{u}+\underline{v})$ is (multiple of $4$) + (multiple of $4$) $-\ 2\times$ overlap($\underline{u}$, $\underline{v}$), and by (10.2), $2\times$ overlap($\underline{u}$, $\underline{v}$) is a multiple of $4$ so the result is a multiple of $4$.

## The extended binary Golay code $G_{24}$

**Definition: the extended binary Golay code $G_{24}$**

The extended code $\widehat{G}_{23}$ is called the **extended binary Golay code** and is denoted $G_{24}$.

The code $G_{24}$ is not cyclic, but we can modify the cyclic code methods used for $G_{23}$ to answer questions about $G_{24}$. For example:

**Example: generator matrix for $G_{24}$**

Write down a generator matrix for $G_{24}$.

**Solution.** Theorem 9.4 gives a generator matrix for $G_{23}$ as follows: the top row is the vector $\underline{g} = 10101110001100000000000$, and the rest of the rows are its cyclic shifts $\underline{xg}, \ldots, \underline{x^{11}g}$. Extending each of these rows (of weight 7 which is odd) by appending 1 gives twelve codevectors of $G_{24}$, forming the matrix

$$G = \begin{bmatrix}
101011100011000000000001 \\
010101110001100000000001 \\
001010111000110000000001 \\
000101011100011000000001 \\
000010101110001100000001 \\
000001010111000110000001 \\
000000101011100011000001 \\
000000010101110001100001 \\
000000001010111000110001 \\
000000000101011100011001 \\
000000000010101110001101 \\
000000000001010111000111
\end{bmatrix}$$

The rows of $G$ are linearly independent, because they give a linearly independent set if you delete the last bit). By definition of extended code, $\#G_{24} = \#G_{23} = 2^{12}$ and so $\dim G_{24} = 12$, same as the number of rows of $G$. Hence $G$ is a generator matrix for $G_{24}$.

The next two propositions establish two main properties of $G_{24}$.

**Proposition 10.2: $G_{24}$ is self-dual**

$G_{24}$ is a self-dual code, that is, $G_{24} = G_{24}^{\perp}$.

*Proof.* It is enough to check that the above generator matrix $G$ for $G_{24}$ satisfies $GG^T = 0$ — that is, its rows $\underline{r}_0, \ldots, \underline{r}_{11}$ are orthogonal to each other — and $n = 2k$. The latter is clear as $24 = 2 \times 12$. The former can be done in two ways.

**Way 1 (manual):** recall (10.2). Manually check that the overlap of $\underline{r}_i$ and $\underline{r}_j$ is even for all $i, j$. It is enough to check the overlap of the top row with the other rows — the rest follows by cyclic shifts of the first 23 bits.

**Way 2 (working with polynomials):** Write the rows of $G$ as $\underline{r}_i = \widehat{\underline{x^i g}} = (\underline{x^i g}, 1)$ for $i = 0, 1, \ldots, 11$. We calculate the inner product, $(\underline{x^i g}, 1) \cdot (\underline{x^j g}, 1) = \underline{x^i g} \cdot \underline{x^j g} + 1$, of two rows of $G$. Recall from the proof of Theorem 9.4 that the inner product of vectors $\underline{a}$ and $\overleftarrow{\underline{b}}$

is the coefficient of $x^{n-1}$ in the polynomial $a(x)b(x)$. The vector $\underline{x^j g}$ written backwards is seen to be $\underline{x^{11-j}\overleftarrow{g}(x)}$, so

$$\underline{x^i g} \cdot \underline{x^j g} + 1 = \big(\text{coef. of } x^{22} \text{ in } x^{i+11-j} g(x)\overleftarrow{g}(x)\big) + 1.$$

Note that

$$g(x)\overleftarrow{g}(x) = \frac{x^{23}-1}{x-1} = x^{22} + x^{21} + \cdots + x + 1$$

is a polynomial where the coefficients of $x^0, \ldots, x^{22}$ are all $1$ and so $x^{22}$ appears in the polynomial $x^{i+11-j} g(x)\overleftarrow{g}(x)$ with coefficient $1$. Thus, $\underline{x^i g} \cdot \underline{x^j g} + 1 = 1 + 1 = 0$. □

Self-duality of $G_{24}$ allows us to deduce other further properties of this code.

**Proposition 10.3: weights in $G_{24}$**

The weight of every codevector of $G_{24}$ is a multiple of $4$.

*Proof.* Each row $\underline{r}_i$ of the generator matrix $G$ constructed in the proof of Proposition 10.2 has weight $8$ which is a multiple of $4$. By Proposition 10.2, rows of $G$ are mutually orthogonal, so by (10.3), a sum $\underline{r}_i + \underline{r}_j$ of two rows of $G$ also has weight divisible by $4$.

We can now apply (10.3) to a sum of $\underline{r}_i + \underline{r}_j$ and $\underline{r}_k$ (both are codevectors of $G_{24}$ so their inner product is zero by Proposition 10.2) to show that a sum of three rows of $G$ has weight divisible by $4$. Continuing in the same way, we show that a sum of any number of rows of $G$, i.e., any codevector of $G_{24}$, has weight divisible by $4$. □

**Remark: rest of proof of Theorem 10.1**

We are now ready to finish the proof of Theorem 10.1 about the parameters of $G_{23}$.

**Proof of Theorem 10.1 — part 2 (final).** We are left to prove that the binary Golay code $G_{23}$ does not contain non-zero codevectors of weight less than $7$.

We will take $G_{23}$ to be cyclic with generator polynomial $g(x)$, and will interchangeably use vectors and polynomials. Assume $\underline{v} \in G_{23}$. If a vector $\underline{v}'$ obtained from $\underline{v}$ by applying the cyclic shift $m$ times, then $v' \in G_{23}$; note that a term $x^i$ in the polynomial $v(x)$ is shifted to $x^{i+m}$ in $v'(x)$ if $i + m < n$, more generally to $(i+m) \bmod n$, where $n = 23$.

**$w(\underline{v})$ cannot be 1, 2, 5 or 6.** If $\underline{v} \in G_{23}$ has weight $1$, $2$, $5$ or $6$, then the extended vector $\widehat{\underline{v}} \in G_{24}$ has weight $2$ or $6$, not divisible by $4$, contradicting Proposition 10.3.

**$w(\underline{v})$ cannot be 3.** Assume $w(\underline{v}) = 3$ so that $v(x) = x^i + x^j + x^k$. Out of the $22$ possible cyclic shifts of $\underline{v}$, at most six can have non-zero overlap with $\underline{v}$: these shift $x^a$ to $x^b$ for

some $a, b \in \{i, j, k\}$. Hence there exists a shift $\underline{v}'$ of $\underline{v}$ which has zero overlap with $\underline{v}$. Then $\underline{v} + \underline{v}' \in G_{23}$ has weight 6, contradicting the previous case.

$w(\underline{v})$ **cannot be 4.** Suppose it can, and shift $\underline{v}$ so that $v(x) = 1 + x^a + x^b + x^c$ with $0 < a < b < c$. Pick a code polynomial of weight 4 of least possible degree $c$.

Shifting $v(x)$ to the left $a$ times gives $v'(x) = 1 + x^{b-a} + x^{c-a} + x^{n-a}$. Note that $(\underline{v}, 0)$ and $(\underline{v}', 0)$ lie in $G_{24} = G_{24}^{\perp}$ and so have inner product 0, hence $\underline{v} \cdot \underline{v}' = 0$ and by (10.2) the overlap of $\underline{v}$ and $\underline{v}'$ must be even. The overlap is not 4 because $v'(x) \neq v(x)$: otherwise one would have $b = 2a$, $c = 3a$ and $n = 4a$, impossible as $n = 23$. The overlap is not 0 as $v(x)$ and $v'(x)$ have term 1 in common. Hence the overlap of $\underline{v}$ and $\underline{v}'$ is 2.

Observe that $n - a = c$ is impossible, as it would give the code polynomial $v(x) - v'(x)$ of degree less than $c$ and weight 2 (impossible by earlier cases) or 4 (contradicts minimality of $c$), so $n - a > c$. Neither $x^{n-a}$ nor $x^c$ contribute to the overlap of $\underline{v}$ and $\underline{v}'$, which leaves three cases of how overlap 2 could be achieved.

**Case** $c - a = b$**.** Then $v(x) = 1 + x^a + x^b + x^{a+b}$ which factorises as $(1 + x^a)(1 + x^b)$. The code polynomial $v(x)$ is divisible by the generator polynomial $g(x)$ which is irreducible, so $1 + x^a$ or $1 + x^b$ must be divisible by $g(x)$. But this means a codevector of weight 2, a contradiction.

**Case** $c - a = a$**.** Writing $b = a + d$, we have $v(x) = 1 + x^a + x^{a+d} + x^{2a}$. Shift $d$ times to obtain $v''(x) = x^d + x^{a+d} + x^{a+2d} + x^{2a+d}$ (since $2a < n - a$, we have $2a + d < n$). The polynomials $v(x)$ and $v''(x)$ have the term $x^{a+d}$ in common, and the only possibility for the overlap of $\underline{v}$ and $\underline{v}''$ to be 2 is $a + 2d = 2a$, that is, $a = 2d$. Then $v(x) = 1 + x^{2d} + x^{3d} + x^{4d}$ which factorises as $(1 + x^d)(1 + x^d + x^{3d})$. As above, either $1 + x^d$ or $1 + x^d + x^{3d}$ must be divisible by $g(x)$, so there is a codevector of weight 2 or 3, a contradiction.

**Case** $b - a = a$**.** We have $v(x) = 1 + x^a + x^{2a} + x^c$, and shift $2a$ times gives $v''(x) = x^{2a} + x^{3a} + x^{4a} + x^{(c+2a) \bmod n}$. The overlap of $\underline{v}$ and $\underline{v}''$ must be 2, and the two polynomials have the term $x^{2a}$ in common, so there must be another common term. This is only possible in two subcases.

**Subcase** $c = 4a$**.** We have $v(x) = 1 + x^a + x^{2a} + x^{4a}$ which factorises as $(1 + x^a)(1 + x^{2a} + x^{3a})$. As above, this means a codevector of weight 2 or 3, contradicting earlier results.

**Subcase** $(c+2a) \bmod n = 0$**.** Here we have the code polynomial $v''(x) = 1 + x^{2a} + x^{3a} + x^{4a}$, which factorises in the same way as in the case $c - a = a$ above, so that we arrive at the same contradiction.

**Conclusion.** We showed that $G_{23}$ has no codevectors of weight $1, 2, 3, 4, 5, 6$ and so $w(G_{23}) \geq 7$ as claimed. This completes the proof of Theorem 10.1. □

The above theoretical proof that $w(G_{23}) = 7$ gives the taste of how Coding Theory was done in the last century. Today, the weight of $G_{23}$ can be easily found using a computer —

consider for example the following code written for the computer algebra system **SageMath**:

```
sage: R.<x>=GF(2)[]
sage: factor(x^23 - 1)
(x+1)*(x^11+x^9+x^7+x^6+x^5+x+1)*(x^11+x^10+x^6+x^5+x^4+x^2+1)
sage: g = factor(x^23 - 1)[1][0]
sage: messagepolynomials = R.monics( max_degree=23-g.degree()-1 )
sage: codepolynomials = [ u*g for u in messagepolynomials ]
sage: min([ len(c.coefficients()) for c in codepolynomials ])
7
```

Is the above code a proof? Many mathematicians would accept it as the source code can be checked, and the calculation reproduced.

**Remark: trivia**

The code $G_{24}$ was used by Voyager 1 & 2 spacecraft to transmit information back to Earth (NASA, Jupiter and Saturn, 1979–81).

### The ternary Golay code $G_{11}$

In $\mathbb{F}_3[x]$, $x^{11}-1=(x-1)g(x)g_1(x)$ where $g(x)=x^5+x^4+2x^3+x^2+2$ and $g_1(x)=-\overleftarrow{g}(x)=x^5+2x^3+x^2+2x+2$.

**Definition: the ternary Golay code $G_{11}$**

A **ternary Golay code** is the the cyclic code in $\mathbb{F}_3^{11}$ generated by $g(x)$, or any code linearly equivalent to it. (*Notation*: $G_{11}$.)

**Theorem 10.4: paremeters of $G_{11}$**

$G_{11}$ is a perfect $[11,6,5]_3$ code.

The crucial, and difficult, step in a theoretical proof of Theorem 10.4 is showing that $G_{11}$ does not contain non-zero codevectors of weight less than $5$. We omit the proof.

An alternative approach is a computer-based calculation:

**Exercise.** Prove Theorem 10.4, modifying the computer code provided after the proof of Theorem 10.1 to calculate the weight of $G_{11}$.

### Historical notes

Golay found his two perfect codes in 1949, before cyclic codes were discovered. He wrote check matrices for $G_{23}$ and $G_{11}$. Crucially, Golay observed that $\binom{23}{0}+\binom{23}{1}+\binom{23}{2}+\binom{23}{3}$

is a power of two. From the proof of perfectness above one can see that the condition $\binom{n}{0} + \cdots + \binom{n}{t} = 2^r$ is necessary for the existence of a perfect $t$-error-correcting binary code of length $n$. This condition is not sufficient: e.g., in his 1949 paper Golay also observes that $\binom{90}{0} + \binom{90}{1} + \binom{90}{2} = 2^{12}$ but this does not lead to any perfect binary code of length 90.

Amazingly, Golay's 1949 paper where he constructs all the Hamming codes and the two Golay codes, is barely half a page long.

Now we can state the classification result about perfect codes.

**Definition: parameter equivalence**

We say that two codes are **parameter equivalent**, if they both are $[n, k, d]_q$-codes for some $n, k, d$ and $q$.

The following theorem was proved by Tietäväinen and van Lint in 1973, more than twenty years since Golay gave a conjectural list of perfect codes in alphabets of prime power size. We will not give its proof here, but you should learn the statement of the theorem.

**Theorem 10.5: classification of perfect codes where $q$ is a prime power**

Let $q$ be a power of a prime number. A perfect $[n, k, d]_q$-code is parameter equivalent to one of the following:

- a trivial code: $n$ arbitrary, $k = n$, $d = 1$, $q$ any prime power;
- a binary repetition code of odd length: $n$ odd, $k = 1$, $d = n$, $q = 2$;
- a Hamming code $\mathrm{Ham}(r, q)$: $n = \dfrac{q^r - 1}{q - 1}$, $k = n - r$, $d = 3$, $q$ any prime power;
- the Golay code $G_{23}$, which is a $[23, 12, 7]_2$-code;
- the Golay code $G_{11}$ which is an $[11, 6, 5]_3$-code.

**Remark: perfect codes over general alphabets**

Classification of perfect codes over alphabets of size not equal to a prime power is, in general, an open problem.