Communication Systems - Assignment 3

vers. 1.0 13/11/2021

Roberto Garello - Politecnico di Torino
roberto.garello@polito.it

5G SSS Gold set

Definition

The official document **3GPP TS 38.211 V16.3.0 (2020-09) - 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; Physical channels and modulation (Release 16)** describes the 5G New Radio SSS sequences generation in this way:

The sequence $d_{SSS}(n)$ for the secondary synchronization signal is defined by

$$d_{SSS}(n) = [1 - 2x_0((n + m_0) \bmod 127)][1 - 2x_1((n + m_1) \bmod 127)]$$

$$m_0 = 15\left\lfloor \frac{N_{ID}^{(1)}}{112} \right\rfloor + 5N_{ID}^{(2)}$$

$$m_1 = N_{ID}^{(1)} \bmod 112$$

$$0 \le n < 127$$

where

$$x_0(i+7) = (x_0(i+4) + x_0(i)) \bmod 2$$
$$x_1(i+7) = (x_1(i+1) + x_1(i)) \bmod 2$$

and

$$[x_0(6)\ x_0(5)\ x_0(4)\ x_0(3)\ x_0(2)\ x_0(1)\ x_0(0)] = [0\ 0\ 0\ 0\ 0\ 0\ 1]$$
$$[x_1(6)\ x_1(5)\ x_1(4)\ x_1(3)\ x_1(2)\ x_1(1)\ x_1(0)] = [0\ 0\ 0\ 0\ 0\ 0\ 1]$$

Labeling and product

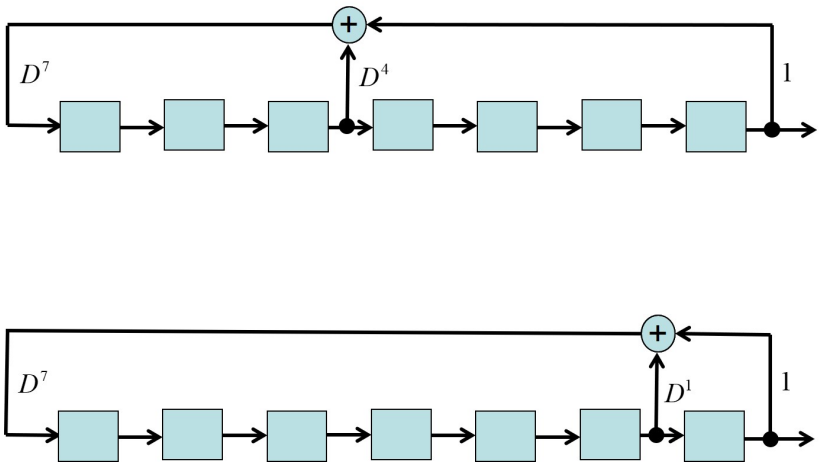
The sequence $d_{SSS}(n)$ reported in the 3GPP document is a bipolar ($\{-1/+1\}$) sequence.

It is obtained by multiplying two bipolar sequences obtained by two binary sequences with the bit/symbol association $0 \rightarrow +1, 1 \rightarrow -1$.

The two binary sequences are m-sequences.

5G SSS Gold set

The SSS sequences of length $N = 127$ are obtained by using two m-sequences generated by these two 7-cell Linear Feedback Shift Registers (LFSR) with polynomial description $D^7 + D^4 + 1$ and $D^7 + D + 1$:



The starting seed for both of them is 0 0 0 0 0 0 1

Since the product of two bipolar sequences is equivalent to the ex-or sum of the two original binary sequences, the SSS sequences are Gold sequences with length $N = 127$ bits.

5G SSS Gold set

Both the two m-sequences are shifted by a quantity that depends on the PCID.

PCI

The Physical Layer Cell ID (PCID) is given by:

$$N_{ID}^{cell} = \{0, \ldots, 1007\}$$

$$N_{ID}^{cell} = 3N_{ID}^{(1)} + N_{ID}^{(2)}$$

$$N_{ID}^{(1)} = \{0, \ldots, 335\}$$

$$N_{ID}^{(2)} = \{0, 1, 2\}$$

Matlab

Even if there exists this Matlab function for Gold sequence generation:

Matlab Gold function

```
m=7;
L=2^m-1;
goldseq = comm.GoldSequence('FirstPolynomial','x^7+x^4+1',...
'SecondPolynomial','x^7+x+1',...
'FirstInitialConditions',[0 0 0 0 0 0 1],...
'SecondInitialConditions',[0 0 0 0 0 0 1],... 'Index',1,'SamplesPerFrame',L);
x1 = goldseq()';
x1b=1-2*x1;
```

for this exercise we suggest to generate separately the two m-sequences by using the comm.PNSequence function of Assignment 2 (because for the 5G Gold set the first m-sequence is shifted, too)

Matlab Assignment A3 - exercise 3.1

Write a Matlab program to:

1. Generate all the 1008 5G SSS Gold sequences.
2. Take the first two sequences, compute their periodic cross-correlation, **plot it** and verify the 3-value Gold code cross-correlation property.
3. Take the first sequence, compute its periodic cross-correlation against all the other 1007 SSS sequences. For each pair compute the maximum of the absolute value of the periodic cross-correlation. **Discuss the result**.

Definitions

Given a set $\{\underline{c}_i\}_{i=1}^{K}$ of K sequences of length N we define:
The cyclic autocorrelation function

$$r_{c_i}(\tau) = \sum_{n=1}^{N} c_i(n) c_i(n+\tau)$$

where $\underline{c}_i$ is viewed as periodic with period N.
The cyclic crosscorrelation function

$$r_{c_i c_j}(\tau) = \sum_{n=1}^{N} c_i(n) c_j(n+\tau)$$

where $\underline{c}_i$ and $\underline{c}_j$ are viewed as periodic with period N.

Definitions

Given a set $\{\underline{c}_i\}_{i=1}^{K}$ of K sequences of length N we define

- the maximal out-of-phase cyclic auto-correlation magnitude

$$r_A = \max\{|r_{c_i}(\tau)| \quad 1 \leq \tau \leq N-1 \quad 1 \leq i \leq K \quad \}$$

- the maximal cyclic cross-correlation magnitude

$$r_C = \max\{\left|r_{c_i c_j}(\tau)\right| \quad 0 \leq \tau \leq N-1 \quad 1 \leq i,j \leq K \quad i \neq j\}$$

- the maximal cyclic correlation magnitude

$$r_M = \max\{r_C, r_A\}$$

Welch bound

It holds for complex sequences of symbols with unitary magnitude:

$$r_M \geq N \sqrt[2s]{\frac{1}{KN-1}\left[\frac{KN}{\binom{N+s-1}{s}}-1\right]} \qquad \forall s \in \mathbb{Z}^+ = \{1,2,3,.....\}$$

(you can stop at s=10, if the root argument is negative, set it to 0)

It is often simplified as:

$$r_M \geq N\sqrt{\frac{K-1}{KN-1}}$$

that, for large K, can be approximated to:

$$\sqrt{N}$$

(Since we are working with binary sequences, use ceil.)

Sidelnikov bound

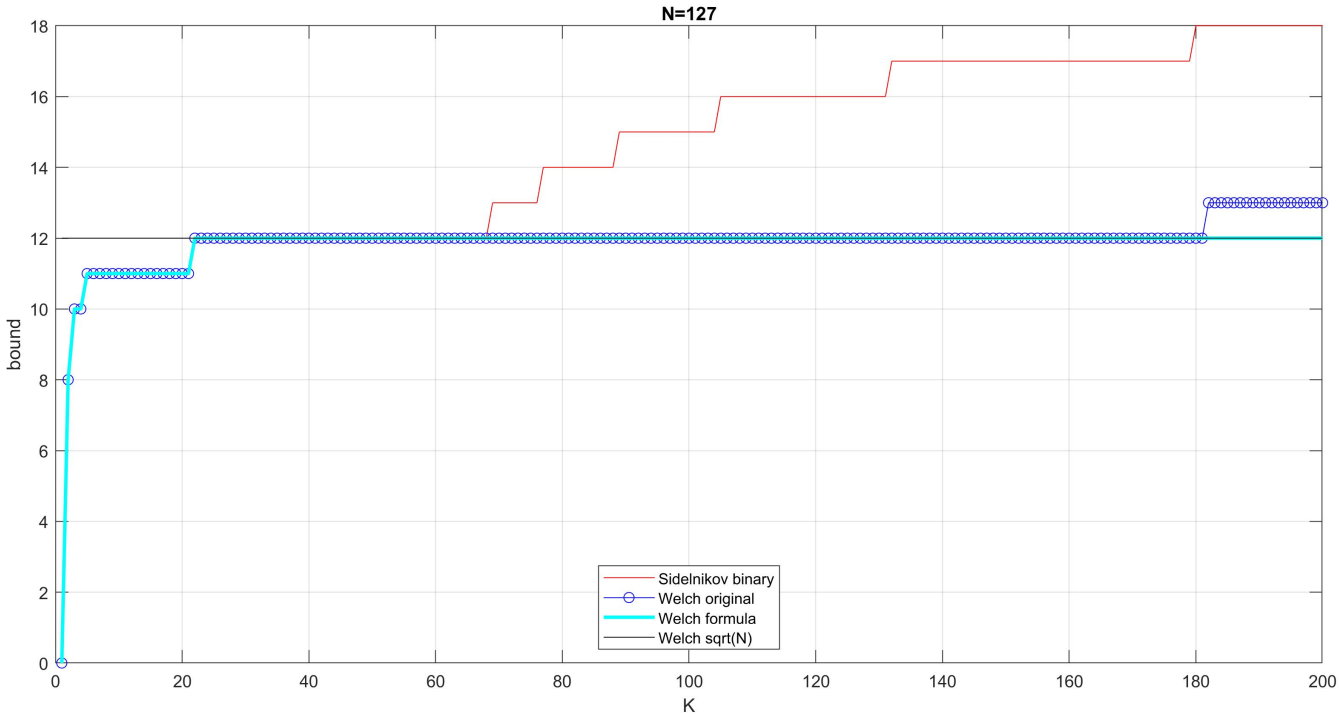
It holds for binary sequences:

$$r_M \geq \left\lfloor \sqrt{\left[(2s+1)(N-s) + \frac{s(s+1)}{2} - \frac{2^s N^{2s+1}}{K(2s)!\binom{N}{s}} \right]} \right\rfloor$$

$$\forall s \in \mathbb{N} = \{0, 1, 2, 3, ...\} \qquad 0 \leq s < \frac{2N}{5}$$

(you can stop at s=10, if the root argument is negative, set it to 0)

Since the Sidelnikov bound applies to binary sequences, it is tighter than the Welch bound.



Matlab Assignment A3 - exercise 3.2

Write a Matlab program to:

4. Plot the three versions of the Welch bound and the Sidelnikov bound.
5. On the same figure, plot a point corresponding to the Gold set with $N = 127$ and $K = 129$.
6. Consider the Gold set of $K = 129$ sequences, but reduce the length to $N = 126$. Compute r_M. On the same figure used before, plot a point corresponding to the value of r_M for $N = 126$ and $K = 129$.
7. Comment the result.

Matlab Assignment A3 - exercise 3.3 - CRC

Consider the polynomial $g(D) = (D^{16} + D^{12} + D^5 + 1)$.
Fix $k = 10$.
Write a Matlab program to

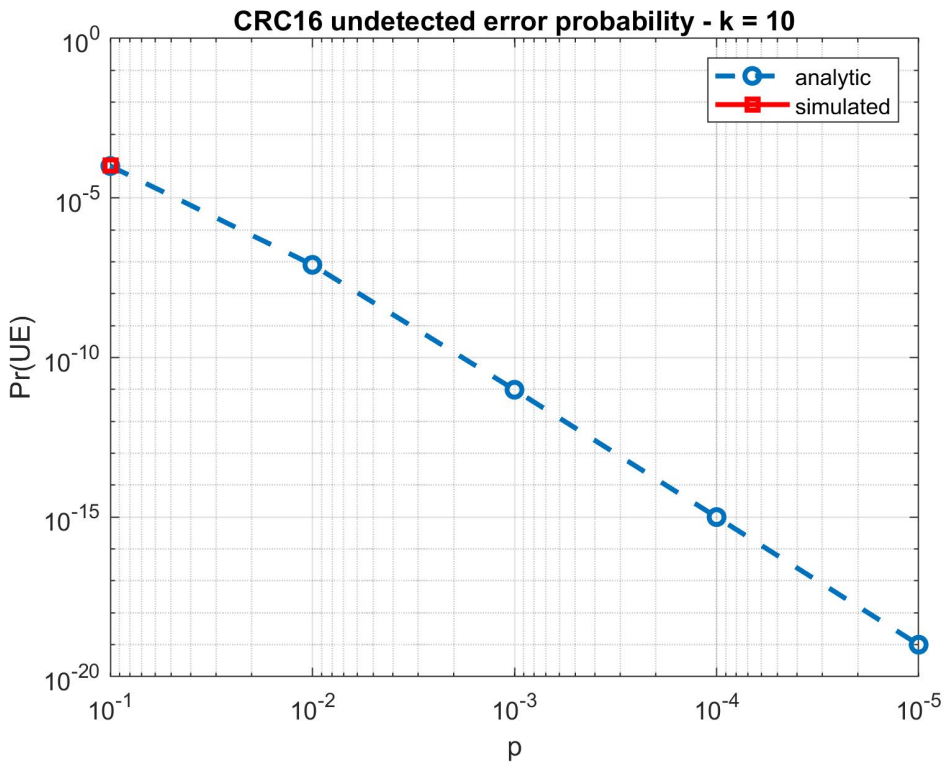
1. Encode the entire H^k by using $g(D)$.
2. Given the codebook, compute the Hamming weight of each non-zero codeword, compute the multiplicity values A_i (number of codewords with Hamming weight i, for $1 \leq i \leq n = k + 16$). Verify that all the weights are even. Verify that all the weights are larger or equal to 4.
3. Given the codeword weights, compute and plot the probability of undetected error vs. p by using

$$Pr(UE) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i}$$

(use $p = 1e-1, 1e-2, 1e-3, 1e-4, 1e-5$).

4. Consider $p = 1e-1$. Evaluate $Pr(UE)$ by simulation and add this point to the figure.

$k = 10$


CRC16 undetected error probability - k = 10
analytic
simulated
Pr(UE)
p
10^0
10^-5
10^-10
10^-15
10^-20
10^-1
10^-2
10^-3
10^-4
10^-5

Matlab

CRC encoder and syndrome check

```
crc_encoder = comm.CRCGenerator('Polynomial','z^16 + z^12 + z^5 + 1');
crc_detector = comm.CRCDetector('Polynomial','z^16 + z^12 + z^5 + 1');
```

First part = codebook generation

- Generate all the $2^k - 1$ non-zero binary vectors (for example, use `de2bi(i,k)`)
- For each vector generate the codeword (use `codeword=crc_encoder(v)`)
- For each codeword compute the Hamming weight
- Compute A_i (number of codewords with Hamming weight $1 \leq i \leq n = k + 16$
- Compute $Pr(UE)$ by using the formula for $p = 1e-1, 1e-2, 1e-3, 1e-4, 1e-5$
- Plot the curve

Second part = simulation

- Generate a k-bit random binary information vector
- Encode it with `crc_encoder`
- Transmit it over the BSC channel with $p = 1e-1$(use `rx_vector=bsc(codeword,p)`)
- Compute the syndrome (use `[~,syndrome]=crc_detector(rx_vector);`)
- If the syndrome is zero, check if there is an undetected error
- Repeat until you observe at least 25 (if your PC is fast enough use 100 to have a better reliability) undetected error events
- Compute the undetected error probability (number of undetected error events/number of transmitted codewords)

Appendix 1 - Gold sequences construction and properties

Construction:

- An m-sequence u_1 generated by a Linear Feedback Shift Registers (LFSR) with n cells and primitive polynomial $p_1(D)$.
- Another m-sequence u_2 generated by a Linear Feedback Shift Registers (LFSR) with n cells and primitive polynomial $p_2(D)$ paired to $p_1(D)$.
- The set of Gold codes is made by the $(2^n + 1)$ sequences obtained as:

$$\begin{cases} u_1 \\ u_2 \\ u_1 + T^i(u_2) \quad 0 \leq i < 2^n - 1 \end{cases}$$

where T^i is the cyclic shift by i positions and $+$ is the binary sum (ex-or) in GF(2).

Gold sequences

Basic properties:

- Sequence length (period):
$$N = 2^n - 1$$
- Number of Gold sequences:
$$2^n + 1$$
- cross-correlation property: three possible values
$$\begin{cases} -t \\ -1 \\ t-2 \end{cases}$$

$t = 2^{\frac{n+1}{2}} + 1$ for n odd, $t = 2^{\frac{n+2}{2}} + 1$ for n even

Appendix 2 - CRC

For students without a background on coding

Basic properties of codes:
A binary linear code $C(n, k)$ is defined by its $k \times n$ generator matrix G. If we denote by H^k and H^n the set of k and n-bit vectors (with binary sum and multiplication), we can associate information vectors $\underline{v} \in H^k$ to codewords $\underline{c} \in H^n$, by

$$\underline{c} = \underline{v}G.$$

Usually, the generator matrix is systematic, i.e., it has form

$$G = [I_k|P]$$

where I_k is a $k \times k$ identity matrix and P is a $k \times r = n - k$ binary matrix. Then, the codeword associated to the information vector $\underline{v}$ has form $\underline{c} = (\underline{v}|\underline{p})$, where $\underline{p}$ contains r redundancy (or parity) bits.
Since there are 2^k different k-bit information vectors and the systematic G maps them into different codewords, there are 2^k different n-bit codewords.

Parity check matrix and syndrome:
Given G, a parity check matrix H is a binary matrix with dimension $n \times r$ that satisfies

$$GH = 0_{k \times r}.$$

The parity check matrix is not unique (it is easy to show that $H = \begin{bmatrix} P \\ I_r \end{bmatrix}$ is a parity check matrix that can be easily obtained from the systematic G).

A key property of linear codes is that, given an n-bit vector $\underline{y}$, its r-bit syndrome obtained by multiplying $\underline{y}$ and H is zero if and only if $\underline{y}$ is a codeword:

$$\underline{s} = \underline{y}H = \underline{0}_r \iff \underline{y} \in C$$

CRC

Error vectors:
Given a codeword $\underline{c} \in H^n$ transmitted over a noisy channel, let us denote by $\underline{y} \in H^n$ the corresponding received vector. Some bits of $\underline{y}$ may be wrong. We model this by introducing an error vector $\underline{e} \in H^n$: $\underline{y} = \underline{c} + \underline{e}$, where given

$$\underline{c} = (c_1, ..., c_i, ..., c_n) \;\; \underline{y} = (y_1, ..., y_i, ..., y_n) \;\; \underline{e} = (e_1, ..., e_i, ..., e_n)$$

$$e_i = 0 \text{ if } y_i = c_i$$

$$e_i = 1 \text{ if } y_i \neq c_i$$

If we model the channel transmission by a Binary Symmetric Channel with error probability p, the probability of a given error vector is

$$Pr\left(\underline{e}\right) = p^{w(\underline{e})} \left(1 - p\right)^{n - w_H(\underline{e})}$$

Then the most probable vectors have Hamming weight 1, then 2, and so on....

Syndrome check and undetected errors:
At the receiver side, given the received vector $\underline{y}$ we want to determine if it is equal to the transmitted codeword or it contains errors. We know that every codeword has zero syndrome. We compute the syndrome of $\underline{y}$ and we accept it if the syndrome is zero.
Unfortunately, we have problem. When we compute the syndrome of $\underline{y}$, we have:

$$\underline{s} = \underline{y}H = \underline{c}H + \underline{e}H = \underline{e}H$$

Then the syndrome of $\underline{y}$ is the syndrome of the error vector $\underline{e}$. If we are unlucky and the error vector introduced by the channel is a non-zero codeword, its syndrome is zero, we accept $\underline{y}$ but this decision is wrong because it is a codeword different from the transmitted one. We call this event an **undetected error**.

Code property:
Clearly, we would like to reduce the probability of undetected error. To do this we need to prevent the error vectors with low weight (the most probable ones) to cause undetected error.
The minimum distance of a code is the minimum Hamming weight of a non-zero codeword

$$d_{min} = \min_{\underline{c} \in C, \underline{c} \neq \underline{0}} w_H(\underline{c})$$

All the error vectors with $w_H(\underline{e}) < d_{min}$ do not belong to the code and cannot produce undetected error, then a possible design criterion is to design codes with large minimum distance.

CRC

CRC

Cyclic Redundancy Check codes are specifically designed for error detection. Typically:

- the number r of redundancy bits is small (usually $r \leq 24$)
- they are able to prevent low-weight error vectors to produce undetected error
- they are designed by using binary polynomials instead of binary vectors

Binary vectors and binary polynomials

Given H^k (set of all possible k-bit vectors), we can map each vector into a binary polynomial with max degree $k-1$ by this rule:

$$\underline{u} = (u_1, ..., u_i, ..., u_k) \rightarrow u(D) = u_1 D^{k-1} + ... + u_i D^{k-i} + ...u_k D^0$$

Let us denote by P^k the set of all possible polynomials of max degree $k-1$.

$H^3 \rightarrow P^k$		
000	$\rightarrow$	0
001	$\rightarrow$	1
010	$\rightarrow$	D
011	$\rightarrow$	$D+1$
100	$\rightarrow$	D^2
101	$\rightarrow$	D^2+1
110	$\rightarrow$	D^2+D
111	$\rightarrow$	D^2+D+1

Generator polynomial

Given a code $C(n,k)$ with $r = n - k$ we can realize the encoding process by multiplying the polynomial $u(D) \in P^k$ by a generator polynomial $g(D)$ with degree r:

$$c(D) = u(D)\, g(D)$$

This way we obtain a polynomial $c(D) \in P^n$, i.e., its max degree is $n-1$, i.e., it corresponds to an n-bit binary codeword $\underline{c} \in H^n$.

Non-systematic encoder

The multiplication by $g(D)$ does not correspond to a systematic encoder, as shown in this example with $g(D) = D + 1$.

$\underline{u}$	$\rightarrow$	$u(D)$	$\rightarrow$	$c(D) = u(D)g(D)$	$\rightarrow$	$\underline{c}$
000	$\rightarrow$	0	$\rightarrow$	0	$\rightarrow$	0000
001	$\rightarrow$	1	$\rightarrow$	$D+1$	$\rightarrow$	0011
010	$\rightarrow$	D	$\rightarrow$	D^2+D	$\rightarrow$	0110
011	$\rightarrow$	$D+1$	$\rightarrow$	D^2+1	$\rightarrow$	0101
100	$\rightarrow$	D^2	$\rightarrow$	D^3+D^2	$\rightarrow$	1100
101	$\rightarrow$	D^2+1	$\rightarrow$	D^3+D^2+D+1	$\rightarrow$	1111
110	$\rightarrow$	D^2+D	$\rightarrow$	D^3+D	$\rightarrow$	1010
111	$\rightarrow$	D^2+D+1	$\rightarrow$	D^3+1	$\rightarrow$	1001

Systematic encoder

A systematic encoder can be obtained by first multiplying the information polynomial by D^r (r shifts to left) then dividing by $g(D)$ and appending the remainder:

$$D^r u(D) = q(D)g(D) + r(D) \rightarrow c(D) = D^r u(D) + r(D)$$

It is fundamental to note that $c(D)$ is still a multiple of $g(D)$:

$$c(D) = D^r u(D) + r(D) = q(D)g(D)$$

$\underline{u}$	$\rightarrow$	$u(D)$	$\rightarrow$	$c(D) = D^r u(D) + r(D)$	$\rightarrow$	$\underline{c}$
000	$\rightarrow$	0	$\rightarrow$	0	$\rightarrow$	0000
001	$\rightarrow$	1	$\rightarrow$	$D+1$	$\rightarrow$	0011
010	$\rightarrow$	D	$\rightarrow$	D^2+1	$\rightarrow$	0101
011	$\rightarrow$	$D+1$	$\rightarrow$	D^2+D	$\rightarrow$	0110
100	$\rightarrow$	D^2	$\rightarrow$	D^3+1	$\rightarrow$	1001
101	$\rightarrow$	D^2+1	$\rightarrow$	D^3+D	$\rightarrow$	1010
110	$\rightarrow$	D^2+D	$\rightarrow$	D^3+D^2	$\rightarrow$	1100
111	$\rightarrow$	D^2+D+1	$\rightarrow$	D^3+D^2+D+1	$\rightarrow$	1111

CRC

Properties of $g(D) = D + 1$ As shown in the previous example, encoding by $g(D) = D + 1$ corresponds to a single parity check code. The only $(r = 1)$ redundancy bit is a parity bit, i.e., all the codewords contains an even number of bits equal to 1.
In fact, $D = 1$ is a zero of $g(D) = D + 1$: $g(D = 1) = 1 + 1 = 0$. But $c(D)$ is a multiple of $g(D)$, then $D = 1$ is a zero of $c(D)$, too: $c(D = 1) = 0$. But this is possible if and only if $c(D)$ contains an even number of terms, i.e., the corresponding codeword has an even number of bits equal to 1.
Important: This way we are sure that the code only contains codewords with even weight. Then all the error vectors with odd weight cannot produce undetected error:

$$\text{all } \underline{e} \text{ with } w(\underline{e}) = 1, 3, 5, \ldots \text{ cannot produce undetected error.}$$

This is the reason why all CRCs used in practical applications have $(D + 1)$ as a factor of the generator polynomial.

Primitive polynomials

A binary primitive polynomial $p(D)$ is the equivalent of an integer prime number: it can be divided only by 1 and by itself.

A key property of primitive polynomials of degree m is that the smallest index J such that $p(D)$ divides $D^J + 1$ is $J = 2^m - 1$.

Example: $p(D) = D^3 + D + 1$ is a primitive polynomial of degree $m = 3$.

It cannot divide $D^4 + 1$ or $D^5 + 1$ or $D^6 + 1$ but it divides $D^7 + 1$.

CRC

CRC and primitive polynomials

Suppose to use a primitive polynomial $p(D)$ of degree m as the generator polynomial of a CRC. All the codeword polynomials are a multiple of $p(D)$. Then all polynomials $D^x + 1$ with $x < J = 2^m - 1$ cannot belong to the code. Then if the codeword length is $n \leq J$, the code cannot contain weight-2 codewords:

all weight-2 error vectors cannot produce undetected error.

CRC

CRC using $(D+1)$ and primitive polynomials
Given a primitive polynomial $p(D)$ of degree m, let use $g(D) = p(D)(D+1)$ as the generator polynomial of a CRC. All the codeword polynomials are a multiple of $g(D)$. The code cannot contain codewords with odd weight and, if the codeword is not too long, codewords with weight 2.
Then all error vectors $\underline{e}$ with $w(\underline{e}) = 1, 2, 3, 5, ...$ cannot produce undetected error.
This is very good because they are the most probable ones. This is the most popular CRC design rule.