Dgitial Communications 2
Formulary, Summaries, Questions and Answers

~ondelette~

July 6, 2021

Contents

What is this supposed to be?

Glad you asked. A humble student's attempt at coping with this lecture. All summarized stuff below was in one way or another taken and adapted from the official lecture notes, a quite old version of those actually. If you find a page number or similar, pointing at the lecture notes, chances are that they won't help you much if you have a more recent version. But don't be angry, rather put the new page number there and everyone will rejoice!

There is absolutely no warranty whatsoever that the stuff in this document is correct (although it was done with care). It may help you to get an idea of what are the most important things to know and look at.

1 Possible Questions Oral Exams

Those were handed down through generations of students taking this class. Not an exhaustive list. Not all of them are worked out in the following, but the material necessary to answer most of them is summarized further below.

- HISO channel model, how the characteristic is described an how this changes if the channel is memory-less and gaussian. HISO channel and its statistical characterization, how does it change if the channel is memory-less and when it's time-invariant memory-less
- Catastrophic encoding of convolutional codes. So basic of convolutional codes, what does catastrophic mean and what is the criterion.
- Dual code. Check Matrix and Dual Code - just an basic overview not too specific
- Check matrix
- MD and BMD decoder - how they work and error probability
- Polynomial matrix description and non-catastrophic encoder. Polynomial generator matrix of convolutional codes
- Turbo Codes - Encoder, performance, why use IIR filters, weight distribution, why interleaver. Turbo codes: encoder, type of constituent codes, weight distribution, interleaver
- Primitive cyclic codes: principles of cyclic codes, primitive block-length, splitting field, factorization of generator and check polynomial, DFT over GF(q^m), special properties of RS codes

---

Question 1.
HISO channel model, how the characteristic is described an how this changes if the channel is memory-less and gaussian.
HISO channel and its statistical characterization, how does it change if the channel is memory-less and when it's time-invariant memory-less.
(Bascically: know everything about the characterization of the channels.)

Answer 1.
HISO = Hard Input, Soft Output.

TO DO: alternative, equivalent descriptions from symbols, etc., to the soft pseudosymbols.

Different versions and their assumptions

- HISO: transition pdf of the channel is used for characterizing the channel. Channel can be scalar-, vector-valued.
- Memoryless HISO: can be time-variant or time-invariant. Memorylessness is characterized by independent scalar uses of the scalar channel.
- Gaussian Memoryless HISO: assumes memorylessness, time-invariance of the channel, Gaussian noise.

TO DO: what is the connection between memorylessness and ISI? How does Gaussian noise relate to the memorylessness? Also check DC1 lecture notes if necessary.

Question 2.
HIHO channel model, how the characteristic is described an how this changes if the channel is memory-less and ... .
HIHO channel and its statistical characterization, how does it change if the channel is memory-less and when it's time-invariant memory-less.
(Bascically: know everything about the characterization of the channels.)

Answer 2.
HIHO = Hard Input, Hard Output.

TO DO: alternative, equivalent descriptions from symbols, etc.

Different versions and their assumptions

- HIHO: transition pdf of the channel is used for characterizing the channel. Channel can be scalar-, vector-valued.

Question 3.
Dual code. Check Matrix and Dual Code: just basic overview, not too specific.

Answer 3.
p. 134 (Linear Block Codes), p. 184, 194 (Cyclic BC).

Dual Code, Linear Block Codes

A linear $(N, K)_q$ code $\mathcal{C}$ is a K-dimensional linear space (a linear subspace of $[GF(q)]^N$. There exists an orthogonal subspace of dimension $(N - K)$ that is again a code, this is the dual code to the original code. The dual code is defined over the same symbol alphabet, has the same blocklength N, but different dimension, i.e., number of parity symbols.

The rate of the dual code is $1 - R$ of the original code.

The dual of the dual code is the original code.

The check matrix of the dual code is the generator matrix of the original code, and vice-versa.

Weight enumerator of code and dual code are related by MacWilliams identity. This can be used to calculate weight enumarator via the dual code, which can be easier, especially in case of high-rate codes.

Dual Code, Cyclic Block Codes

The dual code of a cyclic code is again a cyclic code. The check polynomial of a cyclic code is closely related to the generator polynomial of the cual code. They are the reciprocal polynomials of each other, respectively

The relations show up as well between the check and generator matrices, when matrix descriptions are used, especially their cyclic structure (they are Toeplitz-matrices).

Question 4.

Basics of convolutional codes, what does catastrophic mean and what is the criterion. Catastrophic encoding of convolutional codes.

Answer 4.

Catastrophic encoders: p. 248f, Noncatastrophic encoders: p. 266f (criterion for obtaining noncatastrophic encoders).

Basics of Convolutional Codes

[p. 242f]

Not a block code, i.e. not mapping finite number of data symbols to finite number of code symbols. Instead, continuous stream of data symbols mapped to continuous stream of code symbols.

Internally, frame structure, each data frame has length K, is mapped to code frame of length N. The sequences of frames (which are finite) may be infinite, i.e., contain infinitely many frames.

For block codes, K and N may be very large, whereas for conv. codes, they are very small, in the range of 1, 2, 3.

Encoding is not memory-less: each data frame depends explicitly on a certain number of past data frames. This is the memory of the encoder.

CHECK: does this yield a channel that is not memoryless? I assume so, as the encoder is part of the channel as well?!

TO DO: check the constraint length of the code ν [p. 248].

Properties:

- linearity
- time-invariance
- causality
- rate
- frame memory order
- constraint length
- systematic encoding
- catastrophic encoding

Different ways of describing conv. codes:

- Matrix description: generator and check matrices
- Polynomial description
- Polynomial matrix description
- Trellis representation, graph searching decoders. Viterbi algorithm for HI and SI decoding.
- (Not relevant for exam) Trellis Coded Modulation (TCM)

Catastrophic encoder

Encodes at least one data sequence of infinite weight into a code sequence of finite weight (that is then transmitted over a channel); property of the encoder, not the code. It could happen that the finite number of (nonzero) code symbols get zeroed during the transmission over the channel. This would results in an infinite number of symbols errors caused by a finite number of transmission errors on the channel (because the encoded data sequence has infinite weight, thus infinite length (?!)).

Note that this is a property of the encoder, not the code.

A catastrophic encoder can be detected when the state diagram exhibits a cycle labeled with input data symbols that are not all zero but code symbols that are all zero [p. 276].

Noncatastrophic encoder

There exists a necessary and sufficient condition on the polynomials formed from the tap constellation vectors. These can be arranged in a polynomial generator matrix.

The GCD of the generator polynomials needs to be 1 (in practice; x^L for some $L \in \mathbb{N}$) for the encoder to be noncatastrophic

---

Question 5.

Polynomial matrix description and non-catastrophic encoder. Polynomial generator matrix of convolutional codes.

Answer 5.

Section 7.4, pp 262, 265.

One can form polynomials for the data and code streams. Also, finite order polynomials for the tap coefficient vectors.

Question 6.

MD and BMD decoder: how they work and error probability.

Answer 6.

Sections 3.3. pp. 85, 89.

Both MD and BMD decoder as based on purely deterministic principles, do not suppose any knowledge of the statistical characteristic of the channel (tranhsmission pdf, pmf) or any statistics of the source pmf. In general sub-optimum performance, but have guaranteed error correction capability t. Can be equal to optimum decoders for certain cases.

Minimum Distance (MD)

Chooses codeword closest to senseword:

$$\hat{\boldsymbol{c}}_{\mathrm{MD}} = \arg\min d_H(\boldsymbol{c}, \boldsymbol{v}), \quad \forall \boldsymbol{c} \in \mathcal{C}. \tag{1}$$

The MD decoder is equal to the ML decoder for the symmetric DMC and the BSC. All these minimize the block error probability in the case that all codewords are equally likely.

If m errors occur, $d_H(\boldsymbol{c}, \boldsymbol{v}) = m$.

The MD decoder makes use of Hamming distance d_H, thus we are interested in the minimum distance of the code $d_{\min}$. From this we can compute the quantity t, the so called random error correction capability (random as in errors can occur at random places in the codeword):

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor. \tag{2}$$

The MD decoder can **correct** all possible sensewords with up to t symbol errors, this is guaranteed. If there are more than t errors, then decoding errors are possible, i.e., the decoder may choose a wrong codeword for the received senseword.

Decoding requires an exhaustive search over all q^K codewords for each given senseword $\boldsymbol{v}$. This is often impractical.

Block Error Probability

A channel has to be assumed to state this? In the notes, a DMC is assumed, here making the MD and the ML decoder equivalent.

The probability of a block error is then approximately given by the mean number of nearest neighbors of a codeword and $\mathcal{Q}(d_{\min,p}$ where p is the prob. of a symbol error. $\mathcal{Q}(d_{\min,p}$ is a sum over products $\binom{d_{\min}}{m} p^m (1-p)^{d_{\min}-m}$.

Bounded Minimum Distance (BMD)

The bounded minimum distance decoder only considers a limited number of sensewords around each codeword as candidates for decoding to the respective codeword. This leads to a smaller search space as compared to the MD decoder that needs to compare a given senseword to each codeword in order to find the one with minimum Hamming distance.

How is this actually done for the BMD, does it have a dictionary of codewords up to some maximum Hamming distance r for each senseword? Such that all codedwords are included in the search? Going through all the codewords is not an option, that would be the same as the MD decoder again.

The MD decoder is able to correct some sensewords with more than t errors, whereas the BMD decoder is not. It will always declare a decoding failure for these sensewords.

BMD only considers sensewords within Haming sphere of radius $r \leq t$ for each codeword. Between the spheres around each of the codewords is the interstitial region. Spheres associated with the codewords are called decoding spheres. Because the minimum distance between any 2 codewords is $d_{\min}$, and $r \leq t$, the decoding spheres never intersect.

BMD can:

1. correct all possible sensewords with up to $r \leq t$ symbol errors but no sensewords with more than r errors; some of these could be decoding failures.
2. the MD decoder can correct all sensewords with up to t errors, and some with more than t errors, as long as they are still closer to the correct codeword than to another codeword.

Block Error Probability

Again assuming the symmetric DMC with error prob. p and correct prob $1 - P$, $P = (|\mathcal{D}| - 1)p$.

$$P_{BMD}\{\mathcal{E}\} = \sum_{m=r+1}^{N} \binom{N}{m} P^m (1-P)^{N-m} = \tag{3}$$

$$1 - P_{BMD}\mathbf{C} = 1 - \sum_{m=0}^{r} \binom{N}{m} P^m (1-P)^{N-m}. \tag{4}$$

Question 7.

Primitive cyclic codes: principles of cyclic codes, primitive block-length, splitting field, factorization of generator and check polynomial, DFT over GF(q^m). Special properties of RS codes.

Answer 7.

Section 6.

Question 8.
Turbo Codes: Encoder, performance, why use IIR filters, weight distribution, why interleaver.
Turbo codes: encoder, type of constituent codes, weight distribution, interleaver.

Answer 8.
Constituent codes: typically terminated cbinary onvolutional codes; usually the same codes with $K_1 = K_2 = K$ are used. Typically, K (also interleaver length!) should be large, and so should be N.

Identical recursive (IIR filter) encoders for systematic encoding. Second encoder preceded by interleaver. This is used to ensure few low-weight codewords of the overall code. If the one branch's parity word has low weight, the other branch's parity word should have high(er) weight, so that the overall codeword has good weight. This effect of few low-weight words is caused spectral thinning.

BER performance shows coding gain of few dB over other codes with encoders of comparable complexity. However, very low BER is not easy to achive (directly) because of error floor performance that is due to not too high $d_{\min}$. Concatenation with an outer code can improve BER performance at cost of slightly reduced rate.

---

2 Notation and Variables

$$\nu = \frac{R_b}{B_T} \ldots \text{spectral efficiency}$$

$\mathcal{D}$ ... data (and code) symbol alphabet
$\mathcal{A}$ ... transmit symbol alphabet
$|\mathcal{D}|$ ... symbol alphabet size
N ... length of codeword $\boldsymbol{c}$
K ... length of dataword $\boldsymbol{u}$, $N \geq K$
u_k ... data, information symbols, $\in \mathcal{D}$, $\quad k = 0, \ldots, K-1$
$\boldsymbol{u}$ ... dataword, $\quad K \times 1$
c_n ... code symbols, $\in \mathcal{D}$, $\quad n = 0, \ldots, N-1$
$\boldsymbol{c}$ ... codeword, $\quad N \times 1$
a_l ... transmit symbols, $\in \mathcal{A}$, $\quad l = 0, \ldots, L-1$
$\boldsymbol{a}$ ... transmit symbol vector, $\quad L \times 1$
q_l ... (receive) pseudo-symbols, $\in \mathbb{R}$ or $\in \mathbb{C}$, $\quad l = 0, \ldots, L-1$
$\boldsymbol{q}$ ... pseudo-symbol vector, $\quad L \times 1$
v_n ... sense symbols, $\in \mathcal{D}$, $\quad n = 0, \ldots, N-1$
$\boldsymbol{v}$ ... senseword, $\quad N \times 1$
$s(t)$ ... transmit signal
$n(t)$ ... noise signal
$y(t)$ ... receive signal
$\mathcal{C}$ ... a block code, set of all codewords
$|\mathcal{C}| = |\mathcal{D}|^K$ ... code size, i.e. number of codewords
$d_{H,\min}(\mathcal{C}) = w_{H,\min}(\mathcal{C})$ ... a property of linear codes

3 Basic Properties of Codes

- Symbol alphabet size $q = |\mathcal{D}|$
- blocklength N, dataword length K
- Rate $R = \frac{K}{N}$
- minimum distance $d_{\min}$
- Algebraic structure: linear, cyclic, primitive cyclic, etc.
- Equivalent code: codes equal up to a fixed permutation of the codes symbols within a block/frame/...
- Dual code

Random error **correction** capability:

$$t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor . \tag{5}$$

t is also called the **packing radius** of the block code. This is the largest radius such that each senseword $\boldsymbol{v}$ is in at most one decoding sphere. These are spheres around the codewords $\boldsymbol{c}$ with a certain radius r. For $r = t$, we get the largest possible spheres that do not intersect.

There is also the **covering radius**, that yields spheres such that each senseword is in at least on sphere. Typically the covering radius is larger than the packing radius, unless for perfect codes where it is equal.

There is also the random error **detection** capability. For all patterns of up to m symbol errors, detection is possible iff:

$$1 \leq m \leq d_{\min} - 1. \tag{6}$$

It may be possible to detect more error patterns even if m $\geq d_{\min}$.

For the BMD it is possible to trade off error correction and error detection.

For **perfect codes**, the interstitial region is empty.

In general, for m errors and ρ erasures, a suitable MD decoder can detect / fill this many errors / erasures, iff:

$$2m + \rho \leq d_{\min} - 1. \tag{7}$$

4 Types of Codes considered in class

Block Codes

Block code is the set $\mathcal{C}$ of all codewords $\boldsymbol{c}^{(1)}$ to $\boldsymbol{c}^{(M)}$. The code itself does not specify an encoding mapping from the datawords to the codewords. There are $M = q^K$ different codewords, where $q = |\mathcal{D}|$. CHECK

Denoted $(N, K, d)_q$, $d = d_{H,\min}(\mathcal{C})$

$q \ldots$ characteristic of GF(q). $q = p^m$, $p \in \mathbb{P}$, $m \in \mathbb{N}$.

Linear Block Codes

- structure: weighted sum of codewords is again a codeword, weights from underlying GF(q)
- the $\mathbf{0}$ codeword is part of any linear code
- $\mathcal{C}$ is a K-dimensional linear subspace of $[\mathrm{GF}(q)]^N$
- standard array: way of tabulating all sensewords $\boldsymbol{v}$ relative to given linear block code $\mathcal{C}$
- offer matrix description: encoding, check matrix
- dual codes

- $d_{H,\min}(\mathcal{C}) = w_{H,\min}(\mathcal{C})$. Typically it is much easier to determine $w_{H,\min}(\mathcal{C})$ than the minimum distance, so this is a useful property.
- Weight distribution: number A_w of codewords of weight w. The weight enumerator is another representation of the weight distribution, using a polynomial.

Cyclic Block Codes

All cyclic block codes are linear codes (p. 114).

A block code $\mathcal{C}$ is cyclic if cyclic shift of codeword $\boldsymbol{c}$ is again a codeword in $\mathcal{C}$.

Descriptions:

- Polynomial: codeword, generator, check polynomials
- Matrix: generator, check matrices
- Shift-register circuits:

Variants, with possible additional representations:

- Primitive Cyclic Codes: DFT representation
- Reed-Solomon (RS) Codes
- BCH Codes

Primitive Cyclic Codes

Have primitive blocklength $N = q^m - 1$, $m \in \mathbb{N}, m \geq 1$. $q \ldots$ size of the symbol alphabet, $N \ldots$ is the number of entries in the codeword $\boldsymbol{c} \in \mathcal{C}$.

Allow for describing zeros of generator-polynomial $g(x)$ in closed form.

Allows for "frequency domain" description of codes using DFT.

Convolutional Codes

Encoding with memory, recursive (FIR) or non-recursive (IIR).

Matrix description, polynomial description, trellis description

Turbo Codes

TO DO

5 Error detection, Error correction capabilities

- Random errors: errors can occur in any position, all positions independent of each other
- Random erasures: as above, can occur in any position
- Burst errors: errors are confined to some bust interval $l \ll N$

Random error channels:

- line of sight wireless channels
- deep space channels

- many satellite channels

Note that burst-error channels are **channels with memory**. Types of burst-error channels:

- mobile wireless channels
- cable channels
- magnetic recording channels

There are specific codes designed for burst-error channels.

6 Linear Block Codes

Generator matrix $\boldsymbol{G} : N \times K$:

$$\boldsymbol{G} = (\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{K-1}) \tag{8}$$

The column vectors in the matrix are linearly independent and thus form a basis of the code. A generator matrix is a quite efficient way to generate all codewords of a code from the datawords. Much less storage is needed for storing the matrix elements than e.g. all codewords.

Encoding from the datawords $\boldsymbol{u}$:

$$\boldsymbol{c} = \boldsymbol{G}\boldsymbol{u} \tag{9}$$

Systematic encoder (any generator matrix can be brought to this form using elementary column and row permutations):

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{I}_K \\ \boldsymbol{P} \end{pmatrix}, \tag{10}$$

where $\boldsymbol{P}$ is $(N-K) \times K$.

There is an orthogonal space $\mathcal{C}^\perp$ that is an ($N-K$-dimensional) subspace of $[\mathrm{GF}(q)]^N$ that holds all vectors orthogonal to all codewords and thus to $\mathcal{C}$. There may be vectors that are orthogonal to themselves, thus can occur in both subspaces $\mathcal{C}$ and $\mathcal{C}^\perp$, those are thus not disjoint.

Check matrix $\boldsymbol{H} : N \times (N-K)$ (for the systematic encoder):

$$\boldsymbol{H} = \begin{pmatrix} -\boldsymbol{P}^\mathsf{T} \\ \boldsymbol{I}_{N-K} \end{pmatrix} \tag{11}$$

The check matrix can also be used for decoding. The check matrix is built from linear independent column vectors from $\mathcal{C}^\perp$.

$$\boldsymbol{H}^\mathsf{T}\boldsymbol{c} = \boldsymbol{0}, \quad \forall \boldsymbol{c} \in \mathcal{C}, \tag{12}$$

$$\boldsymbol{H}^\mathsf{T}\boldsymbol{v} = \boldsymbol{0}, \quad \iff \boldsymbol{v} \in \mathcal{C} \tag{13}$$

The latter allows to determine whether the senseword $\boldsymbol{v}$ is a codeword of $\mathcal{C}$ or not. This thus allows **error detection**.

In general, we have:

$$\boldsymbol{G}^{\mathsf{T}}\boldsymbol{H} = \mathbf{0}_{(N-K)\times K}. \tag{14}$$

The check matrix of a code has w linearly dependent rows **iff** there exists a codeword of Hamming weight w. Since minimum Hamming distance and weight are equal for a linear code, with a given check matrix, the minimum distance can be found by finding the minimum number of linearly dependent rows of the check matrix.

Dual Codes

For a $(N, K)_q$ code, there exists a $(N, N-K)_q$ code, the dual code.

$$\boldsymbol{G} = \boldsymbol{H}_{\perp} \tag{15}$$

$$\boldsymbol{H} = \boldsymbol{G}_{\perp} \tag{16}$$

$$R_{\perp} = 1 - R \tag{17}$$

There may be benefits for finding weight enumerator and distribution of the dual code and then get those quantities also for the original code from them.

Syndromes

Can be used for decoding (error correction):

$$\boldsymbol{s} = \boldsymbol{H}^{\mathsf{T}}\boldsymbol{v}, \quad \boldsymbol{s} = \mathbf{0} \iff \boldsymbol{v} \in \mathcal{C} \tag{18}$$

Syndrome $\boldsymbol{s}$, error word $\boldsymbol{e}$:

$$\boldsymbol{s} = \boldsymbol{H}^{\mathsf{T}}\boldsymbol{e} \tag{19}$$

Syndrome decoding, equivalent to MD decoding, but with much less complexity (but still impractical for too large $N-K$):

1. calculate syndrome from senseword
2. lookup minimum weight error word $\hat{\boldsymbol{e}}$ that yields the same syndrome via the check matrix as the senseword
3. decoder output: $\hat{\boldsymbol{c}} = \boldsymbol{v} - \hat{\boldsymbol{e}}$

7 Cyclic Codes

Linear block codes with additional property: cyclic shift of codeword is again a codeword. Description of code using polynomials. The additional structure offers more efficient implementations based on polynomial calculations, implemented using shift-registers, filters.

Important examples:

- Reed-Solomon (RS) codes
- Cyclic Redundancy Check (CRC) codes

Polynomial division:

$$a(x) : g(x) = q(x) + \frac{r(x)}{g(x)}. \tag{20}$$

The remainder of the polynomial division always has smaller degree than the polynomial divisor $g(x)$.

Some stuff about polynomials:

- $a(x)$ is divisible by $g(x)$ if $a(x) = q(x)g(x)$, i.e., with zero remainder $r(x)$ of the polynomial division of $a(x)$ by $g(x)$. Degree of $g(x)$ is $\leq$ to degree of $a(x)$.
- A polynomial that is not divisible by any polynomial of degree ≥ 1 is **irreducible**. Then, no factorization is possible in the underlying field. If it is also a monic polynomial, it is a **prime** polynomial.
- Any polynomial can be uniquely factorized into prime polynomials and a factor, but it may be only one single prime polynomial in case the polynomial to be factorized is irreducible.

7.1 Polynomials for Cyclic Codes

Codewords $\boldsymbol{c}$ as polynomials $c(x)$.

Generator polynomial

Generator polynomial $g(x)$: unique monic nonzero codeword polynomial of smallest degree ($=N-K$). The generator polynomial is itself a codeword. (CHECK)

For any cyclic code of blocklength N, $g(x)$ is a factor of $x^N - 1$ (the famous *magic polynomial*), because $g(x)h(x) = x^N - 1$, see below.

Codewords $c(x)$ can be generated using any polynomial $a(x)$ with max. degree $K-1$ and the generator polynomial:

$$c(x) = a(x)g(x) \tag{21}$$

The properties and performance of the code rely on $g(x)$, e.g. minimum distance, weight distribution. The above equation can be directly used for non-systematic encoding.

Equivalently, using $\boldsymbol{u}'$, i.e. the dataword $\boldsymbol{u}$ zero padded to length N:

$$\boldsymbol{c} = \boldsymbol{u}' \circledast \boldsymbol{g} = \boldsymbol{u}' * \boldsymbol{g} \tag{22}$$

where $\circledast$ is cyclic convolution.

Check polynomial

Check polynomial $h(x)$, of degree K:

$$g(x)h(x) = x^N - 1. \tag{23}$$

Equivalently ($\boldsymbol{h}$ is zero padded length N vector version of $h(x)$):

$$\boldsymbol{g} \circledast \boldsymbol{h} = \boldsymbol{0}. \tag{24}$$

Also (useful for decoding):

$$\boldsymbol{v} \circledast \boldsymbol{h} = \boldsymbol{0}. \tag{25}$$

Another check equation:

$$R_{x^N-1}\{c(x)h(x)\} = 0 \tag{26}$$

This is a necessary and sufficient condition for $v(x)$ to be a codeword (needs to have degree at most $N-1$).

$$R_{x^N-1}\{v(x)h(x)\} = 0 \tag{27}$$

Syndrome Polynomials

There are also syndrome polynomials.

With $v(x) = c(x) + e(x)$:

$$s(x) = R_{g(x)}\{c(x) + e(x)\} = R_{g(x)}\{e(x)\} \tag{28}$$

There is a 1:1 correspondence between each error polynomial of weight $m \leq t$ and a single syndrome polynomials. This is the basis for a syndrome decoding approach.

Alternative syndrome polynomial $\widetilde{s}(x)$:

$$\widetilde{s}(x) = s(x)h(x) \tag{29}$$

Then

$$\widetilde{s}(x) = R_{x^N-1}\{v(x) + h(x)\}, \tag{30}$$

equivalently ($\widetilde{\boldsymbol{s}}$ is of length N, $\boldsymbol{s}$ of length $N-K$):

$$\widetilde{\boldsymbol{s}} = \boldsymbol{v} \circledast \boldsymbol{h}. \tag{31}$$

For decoding using the syndromes, again a decoding table with the syndrome polynomials and their associated minimum weight error polynomials needs to be created and stored. For decoding, the syndrome is computed from the senseword (by dividing by $g(x)$), then the minimum weight error word is found from the table, subtracted from the senseword to form the estimated codeword. This is an implementation of the BMD decoder with decoding radius t (smaller ones also possible), similar as to the one for block codes (using syndromes). More efficient than for block codes, but still too costly for large blocklengths.

7.2 Matrix Description for Cyclic Codes

Basis for the code is built from single generator or check polynomials by cyclic shift. This is possible here due to the cyclic structure of the code. Thus, a single polynomial defines the whole basis and in turn generator (or check) matrix.

$$g_k(x) = x^k g(x), \quad k = 0, \ldots, K-1. \tag{32}$$

From the $g_(x)$, one can form $\boldsymbol{g}_k$, the column vectors of the generator matrix. The check matrix is formed from vectors created from cyclically shifted versions of the reciprocal check polynomial:

$$h_k(x) = x^k \, \overline{h}(x) = x^{K+k} \, h(x^{-1}), \quad k = 0, \ldots, N-K-1. \tag{33}$$

The resulting matrices are Toeplitz matrices. They are at first in non-systematic form but can be brought to systematic form by elementary column and row operations.

7.3 Dual Codes

Dual code of a cyclic code is again a cyclic code.

$$g^{\perp}(x) = \overline{h}(x) = x^K \, h(x^{-1}) \tag{34}$$

$$h^{\perp}(x) = \overline{g}(x) = x^{N-K} \, g(x^{-1}) \tag{35}$$

8 Primitive Cyclic Codes

For q the size of the symbol alphabet, the block-length is now related to it: $N = q^m - 1$, where $m \in \mathbb{N}$ and $m \geq 1$. For primitive codes, the zeros of the generator polynomial can be characterized in closed form. Also, such codes can intuitively be described using the Fourier Transform.

Examples:

- CRC codes
- Reed-Solomon codes
- BCH codes

8.1 Primitive Elements and Stuff

Every Galois-field has at least one primitive element. If $q - 1 \in \mathbb{P}$, i.e., a prime number, then *every* element of the GF not 0 or 1 is a primitive element. Iff an element is a primitive element, one can generate all elements of the GF via powers of a primitive element: $x = z^k$ for $k = 0, 1, \ldots, q-2$. Every GF can thus be represented using powers of its at least one existing primitive element.

Because of

$$z^{q-1} = 1, \tag{36}$$

the field elements repeat for higher powers than $q - 2$, i.e.,

$$z^k = z^{k \mod (q-1)}. \tag{37}$$

8.2 Extension and Splitting Fields

Consider GF(q) with characteristic $p \in \mathbb{P}$ and $q = p_0^m$ for $m_{\in}\mathbb{N}$. We can also have $q' = q^m = (p^{m_0})^m = p^{m_0\,m}$, i.e., $q' > q$. We can also have $q'' = p^{m_0\,n\,m}$, with $n \in \mathbb{N}$. Then:

- GF(q'') = GF($p^{m_0\,n\,m}$) is an extension field to both GF(q) and GF(q')
- GF(q') is an extension field to GF(q)
- both GF(q) and GF(q') are subfields of GF(q'').

In general, some polynomial $a(x)$ will have more zeros in an extension field GF(q^m) of its underlying field GF(q). For any given polynomial $a(x)$ with degree D over some GF(q), there exists an extension field GF(q^m) where the polynomial has D zeros and thus can be completely factorized into *linear factors*. Such an extension field is called a **splitting field**. Every polynomial has a splitting field.

8.3 Primitive Polynomials

A primitive polynomial of degree m is irreducible over GF(q) (cannot be factored into non-constant polynomials with coefficients in the underlying field GF(q)). Page 169 says: irreducible means that a polynomial is not divisible (i.e., division with zero remainder) by any polynomial of degree ≥ 1.

A primitive polynomial $f(x)$ over GF(q) of degree m is:

- irreducible over GF(q)
- all its m zeros (they are distinct) $z_l \in$ GF(q^m) (extension field, a splitting field for the polynomial) are primitive elements of GF(q).

$f(x)$ can be factorized completely in GF(q^m) into linear factors, using its zeros $z_l = z^{(q^l)}$ for $l = 0, \ldots, m-1$:

$$f(x) = \prod_{l=0}^{m-1} (x - z^{(q^l)}) \tag{38}$$

8.4 Defining Set Stuff

For primitive blocklength $N = q^m - 1$, in GF(q^m), $x^N - 1$ can be completely factorized into linear factors, where the N distinct zeros are all the nonzero elements of GF(q^m).

$$x^N - 1 = \prod_{k=0}^{N-1} (x - z^k), \quad N = q^m - 1, \tag{39}$$

where z is a primitive element of GF(q^m) and all zeros are $z_k = z^k$.

Because of $x^N - 1 = g(x)h(x)$, $g(x)$ must consist of some of the zeros of $x^N - 1$, i.e., some z^{k_i} for $i = 1, \ldots, N - K$:

$$g(x) = \prod_{i=1}^{N-K} (x - z^{k_i}) \tag{40}$$

The zeros $\beta_i = z_{k_i}$ define the generator polynomial and thus the whole code $\mathcal{C}$ completely (and the remaining K zeros define the check polynomial $h(x)$?). They are thus called the **defining set** $\mathcal{D} = \{\beta_1, \ldots, \beta_{N-K}\}$.

Checking Sensewords using Zeros
$v(x)$ is a codeword **iff** all β_i are zeros of $v(x)$:

$$v(x) \in \mathcal{C} \Leftrightarrow v(\beta_i) = 0, \quad \forall \beta_i \in \mathcal{D} \tag{41}$$

This also holds for $c(x)$: all zeros of $g(x)$ are also zeros of $c(x)$, but $c(x)$ may have $K-1$ additional zeros, as it has max. degree $N-1$.

8.5 DFT Representation

The DFT uses a primitive element z as the "basis function" for the projection. The z in all of the following equations is the same primitive element used for the DFT.

Vectors $\boldsymbol{v}$ ○—● $\boldsymbol{V}$ can be described as polynomials $v(x)$ ○—● $V(x)$ over GF(q) and GF(q^m) in general, respectively, of degree at most $N-1$: (TO DO: CHECK)

$$v(x) = v_0 + v_1 x + v_2 x^2 + \ldots + v_{N-1} x^{N-1} \tag{42}$$
$$V(x) = V_0 + V_1 x + V_2 x^2 + \ldots + V_{N-1} x^{N-1} \tag{43}$$

Note that the vector representations may need to be zero padded appropriately if the degree is smaller than $N-1$.

$$V_k = v(z^k) \tag{44}$$
$$v_n = \frac{1}{\widetilde{N}} V(z^{-n}) \tag{45}$$

Zeros in the polynomials

$$v(z^k) = 0 \quad \Leftrightarrow \quad V_k = 0 \tag{46}$$
$$V(z^{-n}) = 0 \quad \Leftrightarrow \quad v_n = 0 \tag{47}$$

Instead of

$$\boldsymbol{c} = \boldsymbol{a} \circledast \boldsymbol{g}, \tag{48}$$

we can use, with the DFT representations (this would be non-systematic encoding):

$$C_k = A_k G_k, \quad k = 0, 1, \ldots, N-1. \tag{49}$$

$g(x)$ has $N-K$ zeros $\beta_i = z^{k_i}$ in GF(q^m) (z any primitive element of it), where $q^m - 1 = N$ is the primitive block-length of the primitive cyclic code. The set of zeros of $g(x)$ is the **defining set** of the code, as it fixes $g(x)$ and thus the whole code $\mathcal{C}$.

$$g(z^{k_i}) = 0, \quad i = 1, \ldots, N-K. \tag{50}$$

The zeros of $g(x)$ are also zeros of any codeword, and any polynomial is a codeword iff it has the same zeros as $g(x)$. With $G_k = g(z^k)$ it follows that the zeros of $g(x)$ are zeros of the DFT representations, at frequencies $k = k_i$, called check frequencies, and there are again $N - K$ zeros.

The frequency domain representations of the polynomials from GF(q) exhibit a conjugacy property:

$$C_k^q = C_{qk \mod N}, \quad k = 0, \ldots, N-1, \tag{51}$$

which is necessary and sufficient for $c(x)$ to be from GF(q).

Cyclic shift of codeword results in another codeword, this also works in the frequency domain:

$$C'_k = z^{lk} C_k, \quad k = 0, \ldots, N-1. \tag{52}$$

$$G_k H_k = 0, \quad k = 0, \ldots, N-1. \tag{53}$$

Any $V(x)$ satisfying the conjugacy property is the DFT of a codeword iff:

$$V_k H_k = 0, \quad k = 0, \ldots, N-1. \tag{54}$$

The DFT of any codeword satisfies:

$$C_k H_k = 0, \quad k = 0, \ldots, N-1. \tag{55}$$

Dual Code

Dual code of primitive cyclic code is again a primitive cyclic code. There is a relation between the check frequencies of the code and its dual code.

9 Reed-Solomon-Codes

RS-code: primitive cyclic codes over GF(q), with $q > 2$, i.e. non-binary codes. Popular for error correction. Used and analyzed in the frequency domain. They are maximum-distance codes, thus they attain the singleton-bound. The weight distribution for RS codes has a closed form expression.

- minimum (designed) distance $d_{\min} = d$ is a design parameter
- primitive block length $N = q - 1$, i.e. $m = 1$
- closed-form weight distribution has been derived
- good burst-error correction performance
- efficient encoding and decoding techniques available
- $K = N - d + 1$
- $N - K = d - 1$
- $R = 1 - \frac{d-1}{N}$
- $d_{\min} = d = N - K + 1$

Code $\mathcal{C}$ consists of all codewords $\boldsymbol{c} \in [\mathrm{GF}(q)]^N$, whose DFT over GF($q$), denoted $\boldsymbol{C} \circ\!\!-\!\!\bullet\ \boldsymbol{c}$, is zero in a fixed frequency band $\mathcal{B}$ of $d-1$ cyclically consecutive frequencies.

Block-length N and symbol alphabet q size are related via primitive block-length equation $N = q - 1$.

RS codes use DFT represntation over the base field GF(q), not an extension field.

The check frequencies correspond to the zero band $\mathcal{B}$.

The zeros of the generator polynomial are all from the base field. Thus, $g(x)$ can be completely factorized into linear factors in the base field, that is a splitting field of $g(x)$!

Codewords can be "built" in the frequency domain, filling the K nonzero positions appropriately with elements from GF(q). This gives a possibility for filling these positions directly with the DFT representation of the dataword, thus achieving systematic encoding in the frequency domain.

The conjugacy property does not have to be checked/fulfilled as the elements in the codeword are already from the base field GF(q).

Zeros, Defining set

There is the zero band $\mathcal{B}$ of consecutive zeros in the frequency domain, all elements of GF(q)

$$\beta_i = z^{(k_1+i-1) \mod N}, \quad i = 1, \ldots, d-1. \tag{56}$$

stating the zero band completely specifies $g(x)$ and thus the code.

10 Convolutional Codes

We now look at streams of data and code symbols. The possibly infinite streams are split up into frames. There are data frames of length K and code frames of length N. Encoding of "new" data frames depends on past data frames, i.e., this is encoding with **memory**.

The encoders can be seen as FIR filters, or feed-forward shift registers, but also recursive encoders, i.e., IIR filters, can be used. Systematic encoders exist, but may need to be implemented using IIR filters.

For an N, K encoder there are:

- K internal data subsequences; these are "parallel" paths starting from the input S/P converter, indexed with $j = 0, \ldots, K-1$.
- K shift registers (memory cell chains) of not necessarily the same length m_j, $j = 0, \ldots, K-1$.
- N internal code subsequences. These are ending in the output P/S converter. Indexd with $i = 0, \ldots, N-1$.
- KN tap constellation vectors $\boldsymbol{g}^{(i,j)}$ describing the encoder structure

There are different clocks / rates involved with the decoder:

- input data clock k

- internal frame clock l
- output code clock n

For N, K, we have these relationships between the clocks:

$$nN = kK \tag{57}$$

$$n = lN \tag{58}$$

$$k = lK \tag{59}$$

Tap constellation vectors:

$$\boldsymbol{g}^{(i,j)} = (g_0^{(i,j)}, g_1^{(i,j)}, \ldots, g_{m_j}^{(i,j)}) \tag{60}$$

Each of these describes the "path" from one output of the input S/P-converter to one input of the output P/S-converter, i.e., one tap constellation (with some memory cells somewhere along the path).

The encoding relation using the tap coefficients can be described as:

$$c_l^{(i)} = \sum_{j=0}^{K-1} \left(\sum_{l'=0}^{m} g_{l'}^{(i,j)} u_{l-l'}^{(j)} \right) = \sum_{l'=0}^{m} \left(\sum_{j=0}^{K-1} g_{l'}^{(i,j)} u_{l-l'}^{(j)} \right), \tag{61}$$

the latter formulation is suitable for a certain matrix description, see below.

Some properties:

- encoding is with memory, the past m data frames and the current data frame (so $m+1$ in total) influence the l-th code frame $\boldsymbol{c}_l$
- frame order memory $m = \max m_j, \quad \forall j = 0, \ldots K-1$
- property of the code: constraint length $\nu = \min \sum_{j=0}^{K-1} m_j$ over all possible encoders for the same code
- convolutional code encoding is linear
- encoding is time-invariant at the frame, but not at the symbol level
- encoding is causal at the frame level
- rate $R = \frac{K}{N}$, as for block code, typically quite small, as K often 1.

10.1 Matrix Description

$$\boldsymbol{c} = \boldsymbol{G}\boldsymbol{u} \tag{62}$$

All of these are of infinite size.

Matrix of tap constellation coefficients for all (i,j) for a single time-step l with $l = 0, \ldots, m+1$, has shape $N \times K$:

$$\boldsymbol{G}_l = (g_l^{(i,j)}) \tag{63}$$

An infinitely large overall generator matrix $\boldsymbol{G}$ can then be created as a block-Toeplitz matrix from the $m+1$ matrices $\boldsymbol{G}_l$.

There is a special form of the $\boldsymbol{G}_l$ for systematic encoding.

Similar formulations are available for check matrices $\boldsymbol{H}_l$ and an overall infinitely large matrix $\boldsymbol{H}$.

Truncation, Termination
Truncation: (N, K) convolutional code to $(N_b, K_b) = (PN, PK)$ block code. The overall generator matrix is then a truncated block-Toeplitz matrix of shape $PN \times PK$. It is no longer infinite in size.

Termination: (N, K) convolutional code to $(N_b, K_b) = ((P + m)N, PK)$ block code, additional length due to zero forcing frames. The overall matrix again has a block-Toeplitz structure. It is no longer infinite in size.

10.2 Polynomial representation

Data and code sequences can be represented as power series (infinite degree polynomials):

$$u^{(j)}(x) = \sum_{l=0}^{\infty} u_l^{(j)} x^l, \quad j = 0, \ldots, K-1. \tag{64}$$

$$c^{(i)}(x) = \sum_{l=0}^{\infty} c_l^{(i)} x^l, \quad i = 0, \ldots, N-1. \tag{65}$$

Note that there is a polynomial for each branch leaving a S/P or ending in a P/S converter.

Tap coefficient sequences of finite length are written as finite length polynomials of maximum degree m_j:

$$g^{(i,j)}(x) = \sum_{l=0}^{m_j} g_l^{(i,j)} x^l, \quad i = 0, \ldots, N-1,\ j = 0, \ldots, K-1. \tag{66}$$

Encoding convolution relation can then be written as (sum of) products of data and tap coefficient polynomials:

$$c^{(i)}(x) = \sum_{j=0}^{K-1} g^{(i,j)}(x) u^{(j)}(x), \quad i = 0, \ldots, N-1. \tag{67}$$

Polynomial Generator Matrix
Is of shape $N \times K$ and holds as elements the polynomials (!) $g^{(i,j)}(x)$ (of finite degree m_j):

$$\boldsymbol{G}(x) = (g^{(i,j)}(x)) \tag{68}$$

There is also a systematic form available for the generator matrix.

The polynomials $u^{(j)}(x)$ can $\forall j$ then be stacked into a $K \times 1$ vector $\boldsymbol{u}(x)$ of polynomials, and equivalently for the codeword polynomials, yielding $\boldsymbol{c}(x)$. They can also be viewed as polynomials of dataframes $\boldsymbol{u}_l$, equivalently for the codeframes. Furthermore: $\boldsymbol{G}(x) = \sum_{l=0}^{m} \boldsymbol{G}_l x^l$, i.e., a matrix polynomial of the per-time-step l defined matrices of tap constellation coefficients.

Encoding:

$$\boldsymbol{c}(x) = \boldsymbol{G}(x)\boldsymbol{u}(x) \tag{69}$$

10.3 Non-catastrophic Encoder

In practice (avoiding unnecessary delays in the memory chains), a necessary and sufficient condition for a polynomial generator matrix $\boldsymbol{G}(x)$ to describe a non-catastrophic encoder is that the GCD of all tap coefficient polynomials is 1 (corresponding to $x^L = x^0 = 1$).

Note that a systematic encoder is non-catastrophic as it always includes the non-zero data symbol sequences directly! If the input sequences have infinite weight, then so will the output sequences as a consequence of that.

Non-catastrophic property of a non-recursive encoder is necessary for existence of a left inverse of the polynomial generator matrix $\boldsymbol{G}(x)$ and thus an non-recursive inverse encoder.

11 Turbo Codes

Two fundamental ideas:

1. design code with random-like properties
2. decoder that uses 2 or more SISO constituent decoders within iterative scheme

Design of the interleaver for the second (or multiple) branch(es) is quite important to achieve pseudorandom properties, as is large length K of the interleaver.

Ideally, spectral thinning is achieved, i.e., lower count of low-weight codewords.

12 Modifications of (Block) Codes

Important? TO DO.

13 HISO Channels Family Tree

13.1 HISO

Soft pseudosymbols Q, transmit symbols A. We use a transition pdf of the channel for description.

In general, we consider the "block" channel transition pdf:

$$f_{\boldsymbol{Q}|\boldsymbol{A}}(\boldsymbol{q}|\boldsymbol{a}). \tag{70}$$

If we assume the output symbols Q_l at different times l to be statistically independent, the pdf can be factored as:

$$f_{\boldsymbol{Q}|\boldsymbol{A}}(\boldsymbol{q}|\boldsymbol{a}) = \prod_{l=0}^{L-1} f_{Q_l|\boldsymbol{A}}(q_l|\boldsymbol{a}) \tag{71}$$

If we furthermore assume that each output symbol Q_l only depends on the input symbol A_l, and is statistically independent of all other A_i, $i \neq l$, we can write for the single output symbol Q_l:

$$f_{Q_l|\boldsymbol{A}}(q_l|\boldsymbol{a}) = f_{Q_l|A_l}(q_l|a_l) \tag{72}$$

Both equations above together yield the description of a memoryless HISO channel, where each Q_l only depends on its corresponding A_l. Note that the pdf could change with each time-step l.

13.2 Memoryless HISO

Time-variant and time-invariant versions exist.

Memoryless, time-variant:

$$f_{\boldsymbol{Q}|\boldsymbol{A}}(\boldsymbol{q}|\boldsymbol{a}) = \prod_{l=0}^{L-1} f_{Q_l|A_l}(q_l|a_l) \tag{73}$$

Memoryless, time-invariant; L independent uses of scalar HISO channel $A \to Q$:

$$f_{\boldsymbol{Q}|\boldsymbol{A}}(\boldsymbol{q}|\boldsymbol{a}) = \prod_{l=0}^{L-1} f_{Q|A}(q_l|a_l) \tag{74}$$

13.3 Gaussian Memoryless HISO

Assumes time-invariance, no ISI.

Memoryless, time-invariant; L independent uses of scalar HISO channel $A \to Q$ (as above):

$$f_{\boldsymbol{Q}|\boldsymbol{A}}(\boldsymbol{q}|\boldsymbol{a}) = \prod_{l=0}^{L-1} f_{Q|A}(q_l|a_l) \tag{75}$$

Specialized here to the Gaussian noise:

$$f_{\boldsymbol{Q}|\boldsymbol{A}}(\boldsymbol{q}|\boldsymbol{a}) = \frac{1}{\pi N_0 E_h} \exp\left(-\frac{|q - E_h a|^2}{N_0 E_h}\right). \tag{76}$$

Here: $N_0 \ldots$ CHECK: one or two-sided?, $E_h \ldots$?, . This is scalar, $\mathbb{C}$ Gaussian noise.

14 HIHO Channels Family Tree

14.1 HIHO

Senseword $\boldsymbol{V}$, codeword $\boldsymbol{C}$. We use a transition pmf of the channel for description:

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}). \tag{77}$$

If we assume the output symbols V_n at different times n to be statistically independent, the pmf can be factored as:

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}) = \prod_{n=0}^{N-1} p_{V_n|\boldsymbol{C}}(v_n|\boldsymbol{c}). \tag{78}$$

If we furthermore assume that each output symbol V_n only depends on the input symbol C_n, and is statistically independent of all other C_i, $i \neq n$, we can write for the single output symbol V_n:

$$p_{V_n|\boldsymbol{C}}(v_n|\boldsymbol{c}) = p_{V_n|C_n}(v_n|c_n). \tag{79}$$

Both equations above together yield the description of a memoryless HIHO channel, where each V_n only depends on its corresponding C_n. Note that the pmf could change with each time-step n, as indicated by V_n, C_n in the factors of the total pmf for the memoryless, time-variant HIHO channel:

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}) = \prod_{n=0}^{N-1} p_{V_n|C_n}(v_n|c_n). \tag{80}$$

14.2 Discrete Memoryless Channel (DMC)

This is a memoryless, time-invariant HIHO channel.

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}) = \prod_{n=0}^{N-1} p_{V|C}(v_n|c_n) \tag{81}$$

This corresponds to N independent uses of a scalar HIHO channel $C \to V$ [p. 38].

14.3 Symmetric DMC

Assumes time-invariance. All symbol errors are equally likely, with error probability p for all error cases $n \neq c$.

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}) = \prod_{n=0}^{N-1} p_{V|C}(v_n|c_n) = (1-P)^{N-d_H(\boldsymbol{c},\boldsymbol{v})}\, p^{d_H(\boldsymbol{c},\boldsymbol{v})}. \tag{82}$$

The first part in last equation is correct symbols, the second part for the erroneous symbols.

$P = (|\mathcal{D}| - 1)p$. Binary case: $P = p$, $1 - P = 1 - p$.

14.4 Binary Symmetric Channel (BSC)

Assumes time-invariance. All symbol errors are equally likely, with error probability p. Symbol alphabet is binary.

Scalar

$$p_{V|C}(v|c) = \begin{cases} 1-p, & v|c = \begin{cases} 0|0 \\ 1|1 \end{cases}, \\ p, & v|c = \begin{cases} 0|1 \\ 1|0 \end{cases}, \end{cases} \tag{83}$$

Vector

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}) = (1-P)^{N-w_H(\boldsymbol{e})}\, p^{w_H(\boldsymbol{e})}. \tag{84}$$

Alternatively:

$$p_{\boldsymbol{V}|\boldsymbol{C}}(\boldsymbol{v}|\boldsymbol{c}) = \left(\frac{p}{1-p}\right)^{d_H(\boldsymbol{c},\boldsymbol{v})} (1-p)^N. \tag{85}$$

15 Galois-Fields

In general, we denote a Galois-field as GF(q) with $q = p^m$, with $m \in \mathbb{N}$, and $p \in \mathbb{P}$, i.e. p is a prime number, called the characteristic of GF(q). If we find $m = 1$, i.e., $q = p$, we will just write GF(p). Note that for $m \neq 1$, $q \notin \mathbb{P}$. For some q, we can have $q - 1 \in \mathbb{P}$.

Some examples:

- GF(2): $q = p = 2$, $q - 1 = 1 \notin \mathbb{P}$
- GF(3): $q = p = 3$, $q - 1 = 2 \in \mathbb{P}$
- GF(4): $q = 4$, $p = 2$, $m = 2$, $q - 1 = 3 \in \mathbb{P}$
- GF(5): $q = p = 5$, $q - 1 = 4 \notin \mathbb{P}$
- GF(6): not relevant for us
- GF(7): $q = p = 7$, $q - 1 = 6 \notin \mathbb{P}$
- GF(8): $q = 8$, $p = 2$, $m = 3$, $q - 1 = 7 \in \mathbb{P}$

Some properties:

- For GF(p): Addition ($+$) and multiplication ($\cdot$) are to be executed with $\mod p$.
- For GF(2^m) = GF(q), i.e. fields with characteristic $p = 2$, addition and subtraction are identical. This also means $a = -a$, for field element a.
- For GF(q): If $q-1 \in \mathbb{P}$, then every element of GF(q) that is not $\{0, 1\}$ is a primitive element.
- Every GF(q) has at least 1 primitive element.
- Every non-zero element of a GF(q) can be represented as a power of a primitive element z. $z^k = x$, for every $x \in \text{GF}(q) \setminus \{0\}$, $k = 0, 1, 2, \ldots, q-2$. As there can be multiple primitive elements, there may be different ways to get the elements.

15.1 Constructing a Galois-field

TO DO: CHECK THE NOTATION HERE! q vs. q'.
For constructing a $\mathrm{GF}(q^m) = \mathrm{GF}(q')$:

- Get a primitive polynomial $f(x)$ of order m (irreducible, all zeros $z_l \in \mathrm{GF}(p^m)$) over $\mathrm{GF}(q)$.
- For a primitive element z, we know that $f(z) = 0$. All non-zero elements of the GF are powers of z, $k = 0, 1, \ldots, q' - 2$.
- Example: $f(x) = x^3 + x + 1$, $\Rightarrow f(z) = z^3 + z + 1 = 0$, i.e. $z^3 = z + 1$. Present all other elements as powers of z, get all $z^k = \ldots$ equations. The easiest way is to multiply $z^3 = z + 1$ consecutively with $z, z^2, \ldots$. From these equations, construct the addition table.
- In case $q^m = q' - 1$ is a prime number, any element not in $\{0, 1\}$, i.e. $\{2, 3, \ldots, q' - 2\}$, is a primitive element that can be chosen as z, for constructing the table.
- $z^k = z^{k \mod (q-1)}$