1. Convolutional Codes

1.1. **Coding with memory. Introduction.** The concept of a convolutional code may be motivated by the idea of instantaneously encoding strings of variable length, with memory. In this short introduction, we always take $\mathbb{F} = \mathbb{F}_2$, although everything is generalizable, with the obvious adaptations, to any other finite field $\mathbb{F}$.
Let then for $t = 0, 1, 2, \cdots$, $x(t) \in \mathbb{F}^k$ be a string of data to be encoded; the variable t may be thought as representing time: at each time unit, the coding "machine" accepts k bits and produces n bits, but, contrary to what happens in the block codes studied before, the codeword $c(t) \in \mathbb{F}^n$ depends, linearly, on $x(t)$ but also, possibly, in a finite number of previous inputs.

Example 1. *Let C be the code defined by*

$$c_0(t) = u(t) + u(t-1) + u(t-2), \qquad c_1(t) = u(t) + u(t-2).$$

In this example $k = 1$ and $n = 2$; if the input data is defined for $t \geq 0$, we must naturally extend the definition to $t \geq -2$ putting $x(-2) = x(-1) = 0$ in order that the equations apply for $t \geq 0$.
In the same way, we add two extra zero coordinates in the end of the string, so that the entries of the message are read while they stay in the memory of the encoding process.
The string 100111010 *is completed as*

$$00|100111010|00$$

and encoded to

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Example 2. *Another example, now with $k = 2$ and $n = 3$, would be*

$$c_0(t) = u_0(t), \qquad c_1(t) = u_0(t-1) + u_1(t), \qquad c_2(t) = u_0(t) + u_0(t-1) + u_1(t-1);$$

The string, already completed at the beginning and end with the extra coordinate,

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \Big| \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \Big| \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

is encoded to

$$\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Remark 3. *Both the input $x(t) \in \mathbb{F}^k$ and output $c(t) \in \mathbb{F}^n$ are assumed to be accepted and produced by the coding in a unique time unit, and simultaneously.*
In the representation above we represented each entry of the input and codeword as column vectors, in order to distinguish the dimension of these vectors from the other "horizontal" dimension, which is time.
However, we may represent both the input and codeword as strings of 0 *and* 1*, interleaving the coordinates: in the second example above, for example, the input and otput would be represented as*

$$u = 00|1011100001|00, \qquad c = 101100111011010010$$

1.2. **Polynomial and Formal Power Series representation.** We see now how the idea of a coding with memory is conveyed by the following algebraic construction: let $G = [g_{i\,j}(D)]$ be a generator matrix, where the entries are polynomials in a variable D: $g_{ij}(D) \in \mathbb{F}[D]$, with $0 \leq i < k$, $0 \leq j < n$ and $g_{i\,j}(D) = \sum_t g_{i\,j}(t)D^t$. An input message u, with $u(t) \in \mathbb{F}^k$ for every t, is identified with

$$(u_0(D), \cdots, u_{k-1}(D)) \in (\mathbb{F}[D])^k, \qquad u_i(D) = \sum_t u_i(t)D^t,$$

and encoded into

$$(c_0(D), \cdots, c_{n-1}(D))$$

with

$$c_j(D) = \sum_t c_j(t)D^t = \sum_{i=0}^{k-1} u_i(D)g_{i\,j}(D) = \sum_{i=0}^{k-1}\sum_t \left(\sum_{s=0}^{t} g_{i\,j}(s)u_i(t-s)\right) D^t =$$

$$= \sum_t \left(\sum_{s=0}^{t}\sum_{i=0}^{k-1} g_{i\,j}(s)u_i(t-s)\right) D^t.$$

So the output "at time "t is a linear combination of the inputs $u(t-s)$, where, of course, s is bounded by $M = \max_{ij} \deg(g_{ij}(D))$, which is the memory of the code. In fact, and this will be important later, we see that the values of $c_j(t)$ depend, for each $0 \leq i < k$, on the $u_i(t-s)$, with $0 \leq s \leq m_i = \max_j \deg(g_{ij}(D))$.

Remark 4. *The name convolutional code comes from these equations, as a sum of the form $\sum_s g_{ij}(s)u_i(t-s)$ is the t-coordinate of the convolution of the sequences $g_{ij}(t)$ and $u_i(t)$.*
The choice of D as variable is motivated by the idea that it represents a delay operator.

Example 5. *The code from example* 1 *and* 2 *is represented by the generator matrix*

$$G = \begin{bmatrix} 1 + D + D^2 & 1 + D^2 \end{bmatrix}$$

The input string 10011101010 *corresponds to the polynomial*

$$u(D) = 1 + D^3 + D^4 + D^5 + D^7 + D^9$$

$c_0(D) = (1+D^3+D^4+D^5+D^7+D^9)(1+D+D^2) = 1+D+D^2+D^3+D^5+D^8+D^9$, *whose sequence of coefficients corresponds to the sequence of the first coordinates in the codeword computed above. The same happenswith the second coordinate* $c_1(D)$

Example 6. *The generator matrix for the second example is*

$$G = \begin{bmatrix} 1 & D & 1+D \\ 0 & 1 & D \end{bmatrix}.$$

Here the input is the vctor $u(D) \in (\mathbb{F}[D])^2 = \mathbb{F}^2[D]$

$$u(D) = \begin{pmatrix} u_0(D) \\ u_1(D) \end{pmatrix} = \begin{pmatrix} 1 + D + D^2 \\ D + D^4 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} D + \begin{pmatrix} 1 \\ 0 \end{pmatrix} D^2 + \begin{pmatrix} 0 \\ 0 \end{pmatrix} D^3 + \begin{pmatrix} 0 \\ 1 \end{pmatrix} D^4$$

and the codeword $c(D)$ is computed in the same way.

Although in practical applications it may be useful and necessary to establish an upper bound for the length of the input strings, we want to keep the definition free of that type of constraints. One way to formalize the definition of these codes in a manner similar to that of block codes is to use an infinite field:
let $\mathbb{F}[D]$ denote the ring of polynomials with coefficients in $\mathbb{F}$ over a variable D and $\mathbb{F}(D)$ be its field of fractions, ie, the field of rational functions $\frac{p(D)}{q(D)}$ where $p(D)$ and $q(D) \neq 0$ are polynomials.

More generally, we consider the ring of **Formal Power Series** in D

Definition 7. $\mathbb{F}[[D]] = \{a(D) = \sum_{t \geq 0} a_t D^t : a_t \in \mathbb{F}\}$.

This is a commutative ring, whose units (ie, invertible elements) are determined (**HW**) by the condition

$$a(D) \text{ is a unit } \Leftrightarrow a_0 \neq 0,$$

and its field of fractions is the field of **Laurent Series**

Definition 8. $\mathbb{F}((D)) = \{\sum_{t \geq m} a_t D^t : a_t \in \mathbb{F};\ m \in \mathbb{Z}\}$.

We have the inclusions

$$\mathbb{F}[D] \subset \mathbb{F}[[D]],\ \mathbb{F}[D] \subset \mathbb{F}(D) \subset \mathbb{F}((D)); \mathbb{F}[[D]] \subset \mathbb{F}((D)).$$

Moreover, $a(D) = \frac{p(D)}{q(D)} \in \mathbb{F}[[D]] \cap \mathbb{F}(D)$ iff $q(0) \neq 0$.

Exercise 9. *Find the formal power series expression for* $a(D) = \frac{1+D}{1+D+D^2}$.

As we have seen with polynomials, the vector space $(\mathbb{F}((D)))^n$ may be identified with $\mathbb{F}^n((D))$, ie, the vector space of Laurent series with coefficients from $\mathbb{F}^n$.

We define

Definition 10. *A convolutional binary code with length n and dimension k is a k dimensional subspace of $(\mathbb{F}((D)))^n$ that has a basis $v_0, \cdots, v_{n-1}$ with $v_i \in \mathbb{F}^n[D]$. $R = k/n$ is the information rate of the code. If $G = [g_{ij}]$ is a generator matrix, $M = \max_{ij} \deg(g_{ij})$ is the **memory** of the generator.*

Remark 11. *Distinct generator matrices of the same code may have different memories. A more intrinsic notion of memory will be discussed later.*

1.2.1. *Scalar representation by interleaving.* As we saw before, both the input and codewords of a $[n, k]$ convolutional code may be transformed into strings with elements from $\mathbb{F}$, interleaving the coordinates of the vectors. A generator matrix may be written as

$$G = G_0 + G_1 D + \cdots + G_M D^M = \sum_t G_t D^t,$$

where each G_t is a $k \times n$ matrix over $\mathbb{F}$, and $G_t = 0$ if $t < 0$ or $t > M$.
Define the infinite matrix $I(G)$ in the following way: $I(G) = [B_{i\,j}]_{0 \leq i}^{0 \leq j}$ where each

$B_{i\,j}$ is a $k \times n$ matrix and $B_{i\,j} = G_{j-i}$.
Replacing $x = (x_0(D), \cdots, x_{k-1}(D))$ by

$$I(x) = (x_0(0), x_1(0), \cdots, x_{k-1}(0), x_0(1), \cdots, x_{k-1}(1), \cdots)$$

ie, interleaving the coefficients of coordinates of x,we obtain that

$$I(x)I(G) = (c_0(0), c_1(0), \cdots, c_{n-1}(0), c_0(1), \cdots, c_{n-1}(1), \cdots)$$

which is the interleaving of the coefficients of the $c_j(D)$ (**HW**).

As mentioned earlier, in concrete applications, we deal with finite data. This means that we restrict the source messages $x(D)$ to polynomials with degree less than $L < +\infty$, ie, the restricted coding is a map

$$(P_L[D])^k \to (P_{L+M}[D])^n\,,$$

where $P_m[D]$ denotes the set of polynomials with degree less than m and coefficients in $\mathbb{F}$.
To this restriction corresponds the truncation $I_L(G)$ of the scalar matrix $I(G)$, which is a generator matrix of the truncated $[n(M+L), kL]$ binary code, with information rate

$$R_L = \frac{kL}{n(M+L)}.$$

By taking L much larger than M, we obtain codes with information rate very close to R.

1.3. **Encoding and State Diagrams.** The algorithmic implementation of the computation with polynomials is sometimes represented by shift registers (or physical encoders) and also by state diagrams, in order to capture the notion of a step by step encoding, in which ,at each time t, we receive and encode the input $u(t)$. We describe that last representation, as it is relevant for the decoding algorithm studied below.
A state diagram for the convolutional code C may be visualized as a finite state automaton: a directed graph whose vertices are the possible states of memory; an edge corresponds to an input $u(t)$ that changes the state and produces an output $c(t)$; the input and output are usually represented as a label of the edge. We illustrate the concept with a simple example:

Example 12. *Let C be the code from example 1; the possible states are*

$$00\,01\,10\,11.$$

If at time t we are at state 00, ie $u(t-1) = u(t-2) = 0$, for example, the input $u(t) = 1$ takes us to the state 01 and produces the output $c(t) = (1,1)$; if the initial state is 01 and $u(t) = 1$, the final state is 11 and the output is $c(t) = (0,1)$.
So the encoding of a source message is read through a path in the graph. For a finite message, this path naturally starts and ends in the 00 state. More precisely, assuming, without loss of generality, that the message is $u(0), \cdots, u(L-1)$, the sequence of states starts with $u(-2)u(-1) = 00$ and ends with $u(L)u(L+1) = 00$, independently from the value of the last entry in the message. This implies that if the source message has length L, the codeword has length $2(L+2)$.
The encoding procedure works with 3 variables a, b, and x, initialized as $a = b = 0$,

$x = u(0)$; at step t, the procedure has $a = u(t-2)$, $b = u(t-1)$ already defined, accepts the input value $u(t)$, creates the output

$$(c_0(t), c_1(t)) = (a + b + x, a + x),$$

which is appended to the string of previous outputs, and replaces $a = b$ and $b = x$.

In the general case of a (binary) convolutional $[n, k]$ code with generator matrix $G = [g_{i\,j}]$, we must consider the maximal degree m_i of the entries of $\mathrm{row}_i(G)$, $0 \le i < k$: if $(u_i(t))$ is the input at time t, the corresponding output $(c_j(t))$ depends on the $u_i(t-s)$ with $0 \le s \le m_i$.
So the number of states is $2^{m_0+\cdots+m_{k-1}}$. An input $(u_i(t))_{0\le i<k}$ determines an edge with initial vertex

$$(u_i(t-s)), \qquad 0 \le i < k, \qquad 1 \le s \le m_i$$

and final vertex

$$(u_i(t-s)), \qquad 0 \le i < k, \qquad 0 \le s < m_i,$$

and an output $c(t)$ with

$$c_j(t) = \sum_{i=0}^{k-1} \sum_{s=0}^{m_i} g_{i\,j}(s) x_i(t-s).$$

Example 13. *Let C be the $[3, 2]$ binary convolutional code with generator*

$$G = \begin{bmatrix} 1 & 0 & 1+D \\ 0 & 1 & D^2 \end{bmatrix}.$$

The input at each time t is $(x, y) = (u_0(t), u_1(t))$ and we have memory variables

$$a = u_0(t-1), \qquad b = u_1(t-2), \qquad c = u_1(t-1),$$

(all initialized as zero).
The encoding produces the output

$$(c_0(t), c_1(t), c_2(t)) = (x, y, x + a + b),$$

and updates the memory variables

$$a = x, \qquad b = c, \qquad c = y.$$

Exercise 14. *Suppose the input is the binary string (interleaving of $u(D)$)*

$$0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,0$$

Determine the sequence of states visited by the corresponding path in the state diagram, and the output.

Example 15. *If the sum of maximal degrees of the rows of the generator satisfies $\sum_i m_i < k$, the state diagram will have multiple arrows between pairs of states.*
A particularly simple example is given by the $[4, 2]$ binary code with generator

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1+D & D & 1 \end{bmatrix}.$$

The state diagram has only two states $S(0)$ and $S(1)$, corresponding, at each time t, to the value of $u_1(t-1)$. And we have the arrows

$$S(0) \xrightarrow[01|0101]{} S(1), \qquad S(0) \xrightarrow[11|1010]{} S(1),$$

where, for example, the label $01|0101$ *represents the input and corresponding output, both written in interleaved form.*

Exercise 16. *Complete the state diagram for this example.*

The graphical representation of the state diagram as a directed labelled graph is useful to vizualise the encoding procedure, but is only feasible for small values of k, n and m_i. The usefulness of the concept, however, goes beyond the algorithmic implementation of the encoding, as we'll see.

1.4. **Free Distance.** The appropriate notion of distance for convolutional codes is called **free distance** and denoted $d_f(C)$. For each coordinate $c_j = \sum_{t\geq m} c_j(t)D^t$ of a codeword, we define its **weight** $w(c_j)$ as the number of nonzero coefficients $c_j(t)$. Naturally, c_j may have infinite weight. The weight of a codeword is defined as $w(c) = \sum_j w(c_j)$, and

Definition 17. *The **free distance** of a convolutional code C is the minimal weight of a nonzero finite weight codeword.*

So, the free distance of C coincides with the usual Hamming distance of the code obtained from C by interleaving the coeficients of the coordinates, as described above.

When a convolutional code C with free distance d_f and generator matrix G is used to encode an input $u(D) = \sum_{t=0}^{L-1}$ as a codeword $c(D) = \sum_{t=0}^{L+M-1}$, and an output $y(D) = \sum_{t=0}^{L+M-1}$ is received. If the number of errors created by the transmission process is less or equal than $\frac{d_f-1}{2}$, we know that $c(D)$ is the unique codeword at minimal distance from $y(D)$; so, applying some form of Minimal Distance Decoding, it is possible to correct the errors.

As it happens with block codes, the computation of $d_f(C)$ is in general a difficult problem. The state diagram associated with the generator, described in the last section, may be used, in many cases (see remark below), to compute the free distance and much more, in the following way:

We construct a modified state diagram, by splitting the state with all entries equal to 0, which we denote here as $S(0)$, into a initial state I and a final state T; the arrows leaving $S(0)$ have I as its source, while the arrows terminating at $S(0)$ now terminates at T; the loop at $S(0)$ that corresponds to a zero input is eliminated.
Each arrow of this modified state diagram is then labeled with a monomial X^s, where s is the weight of the output. If we multiply these monomials along a path in the state diagram, starting at I and ending at T, we obtain a monomial X^u, where u is the total weight of the output, ie, of the codeword resulting from the encoding of that input.

From this, we compute the generating function $W(X) = \sum_{i\geq 0} a_i X^i$ where a_i is the number of **fundamental paths** of weight i, ie, paths starting and ending at the zero state but with no intermediate passages by it.
The encoding of a finite weight input corresponds to the concatenation of finitely many such fundamental paths, and possibly of loops at the zero state of the original diagram. So the least i such that $a_i \neq 0$ gives us the free distance of the code.

This construction may be modified to give even more information if each arrow is labeled with a monomial $X^s Y^r Z$, where r is the weight of the input. The monomial associated to a $I-T$ path gives is $X^u Y^v Z^l$, where, u is the weight of the codeword, v the weight of the corresponding input, and l the length of the path.
The generating function $W(X, Y, Z)$ classifies all codewords with respect to those three parameters.
We may recover the simpler generating function as $W(X, 1, 1)$.

Example 18. *We consider again the* $[2, 1]$ *binary code with generator*

$$G = [D^2 + D + 1 \quad D^2 + 1].$$

Let ψ_S *denote the function of* X, Y *and* Z *determined by paths starting at* I *and ending at state* S*. Reading the diagram we find (**HW**) the equations*

$$\begin{cases} \psi_T = X^2 Z \psi_{01} \\ \psi_{10} = XZ(\psi_{11} + \psi_{01}) \\ \psi_{11} = XYZ(\psi_{11} + \psi_{01}) \\ \psi_{01} = X^2 YZ + YZ\psi_{10} \end{cases}$$

Notice that in the last equation we put $\psi_I = 1$*, which is coherent with the definition. We want to obtain the explicit expression for* $W(X, Y, Z) = \psi_T$*, by eliminating the other unknown functions. We have from the third equation*

$$\psi_{11} = \frac{XYZ}{1 - XYZ}\psi_{01},$$

and replacing in the second,

$$\psi_{10} = \left(\frac{X^2 YZ^2}{1 - XYZ} + XZ\right)\psi_{01} = \frac{XZ}{1 - XYZ}\psi_{01};$$

using the last equation, we arrive at

$$\psi_{10} = \frac{XZ}{1 - XYZ}\left(X^2 YZ + YZ\psi_{10}\right) \Leftrightarrow \psi_{10}\left(1 - \frac{XYZ^2}{1 - XYZ}\right) = \frac{X^3 YZ^2}{1 - XYZ} \Leftrightarrow$$

$$\Leftrightarrow \psi_{10} = \frac{X^3 YZ^2}{1 - XYZ(1 + Z)}$$

and so,

$$W(X, Y, Z) = \frac{X^5 YZ^2}{1 - XYZ(1 + Z)} = (X^5 YZ^2)\sum_{j\geq 0}(XYZ(1 + Z))^j.$$

Putting $Y = Z = 1$ *we get*

$$W(X, 1, 1) = X^5 \sum_{j\geq 0}(2X)^j = X^5 + 2X^6 + 4X^7 + 8X^8 + \cdots$$

Remark 19. *It must be stressed that the generating function discussed above counts the number of fundamental paths in the modified state diagram. This allow us to count the number of finite weight codewords that result from the encoding of finite weight inputs, of a given length, by the given generator matrix. The number of codewords of a given finite weight may be infinite.*
The function may be computed if the modified state diagram does not contain a loop or, more generally, a cycle with output of weight zero. The existence of such a cycle implies that, for the corresponding generator matrix, there are finite weight codewords that result from the encoding of a infinite weight input.
The relation between different generator matrices is discussed in another section. The following simple example illustrates these remarks.

Example 20. *We consider the binary convolutional code C with generator*

$$G = [1 \quad 1+D].$$

*It is easy to see (**HW**) that the generating function for the number of fundamental paths in the modified state diagram is*

$$W(X) = \frac{X^3}{1-X} = \sum_{j \geq 3} X^j.$$

In particular, the free distance of the code is 3, as it is easy to confirm directly. It is also easy to verify that C contain infinitely many codewords of weight, for example 6: consider the inputs $u(D) = 1 + D^s$ for any $s \geq 2$. The state diagram may be helpful to conclude that in fact the same is true for any weight $w \geq 6$.
*On the other hand, it is clear (**HW**) that a finite weight codeword must result from the encoding of a finite weight input.*

Now

$$G_1 = [D+1 \quad D^2+1]$$

is also a generator:

$$c(D) = u(D)G \Leftrightarrow c(D) = \frac{u(D)}{1+D} G_1.$$

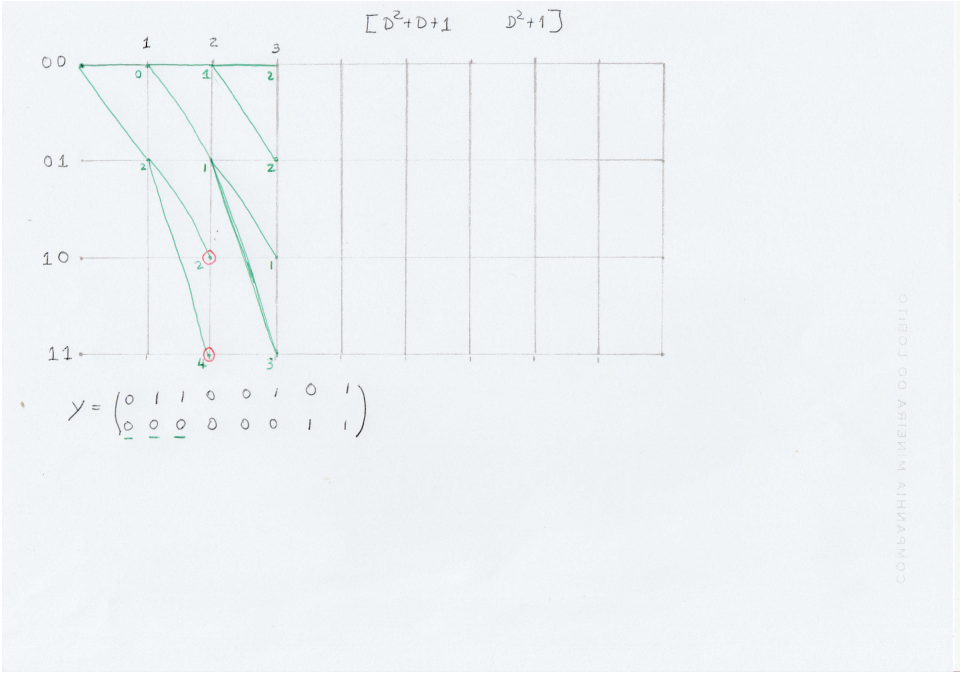
The unique codeword of weight 3 results, for G_1, from the encoding of

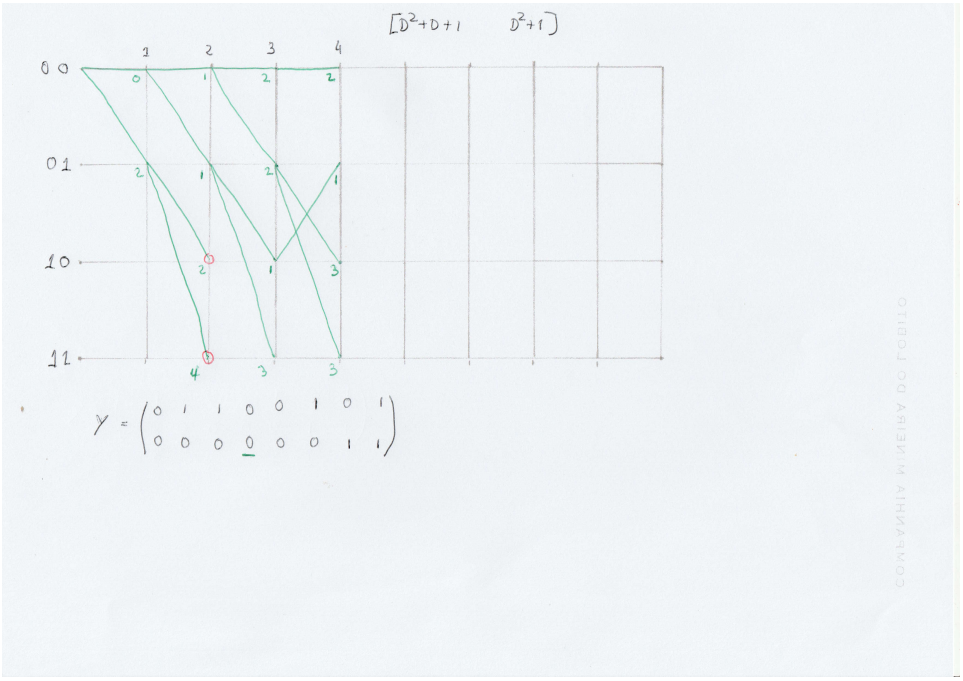
$$u(D) = \frac{1}{1+D} = \sum_{j \geq 0} D^j.$$

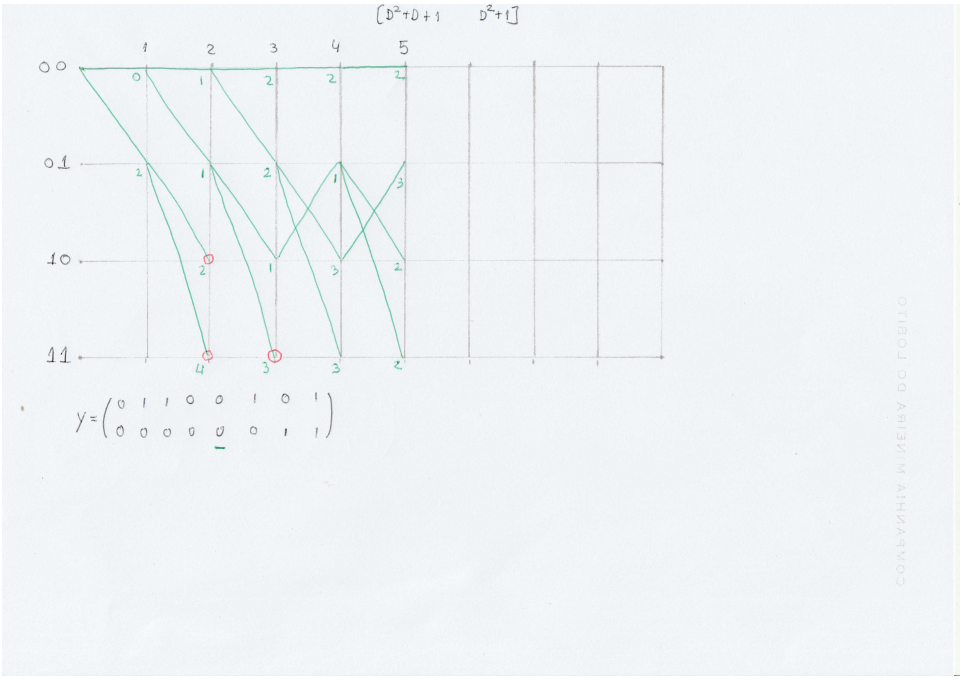
In the corresponding modified state diagram, this is realized as an infinite path, starting at state 00, going to state 01, then to state 11 and then looping infinitely many times at this state.

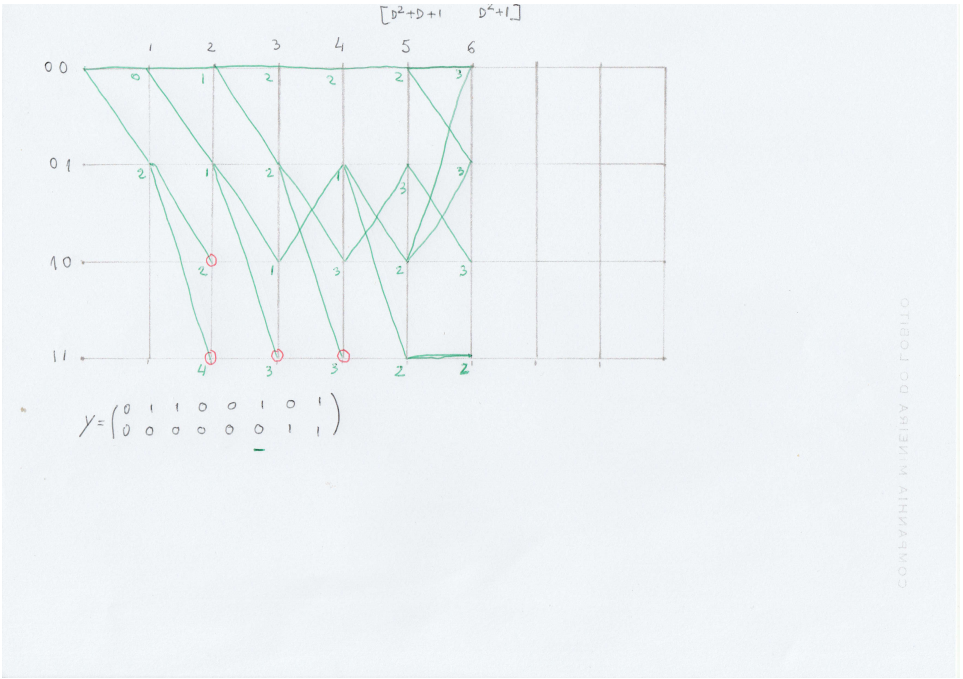
1.5. **Viterbi's algorithm.** In order to explain an algorithm for the decoding of finite messages we develop the state automaton described above into a **trellis diagram**:
A trellis diagram is a table with rows indexed by the states and columns indexed by time units; an edge with $u(t)|c(t)$ label joins (a,t) to $(b,t+1)$ where a is the initial state and b the final state; both $u(t)$ and $c(t)$ are presented in interleaved form.
Viterbi's algorithm goes as follows: suppose a message $y(t)$ with $0 \leq t < L+M$ is received, corresponding to an input $u(t)$ with $0 \leq t < L$.
We construct step by step the trellis diagram, replacing the label of an edge from t to $t+1$ with weight equal to the Hamming distance between $y(t)$ and the output $c(t)$; a survivor path at time t is a path that has minimal weight among all paths starting at the zero state and ending at a given state at time t. After each time step, we identify the survivor paths and only continue these to the next step. A possible decoding of y is a survivor path at time $L+M$ ending at the zero state.
An induction argument on the length of the path shows that the survivor path found is a closest codeword to the output y. This means that the algorithm performs minimal distance decoding.

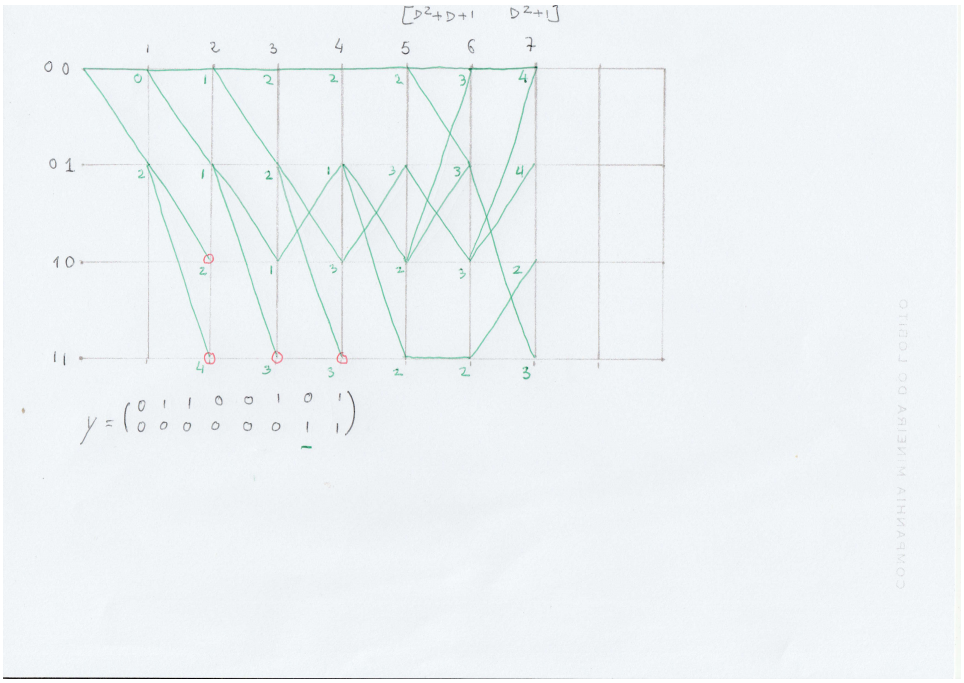
The images below illustrate the application of Viterbi's algorithm for the code with generator $[D^2+D+1 \quad D^2+1]$: in the first image the algorithm was applied up to step 3; in each step only the surviving paths are shown. The last picture presents the decoding and corresponding input.

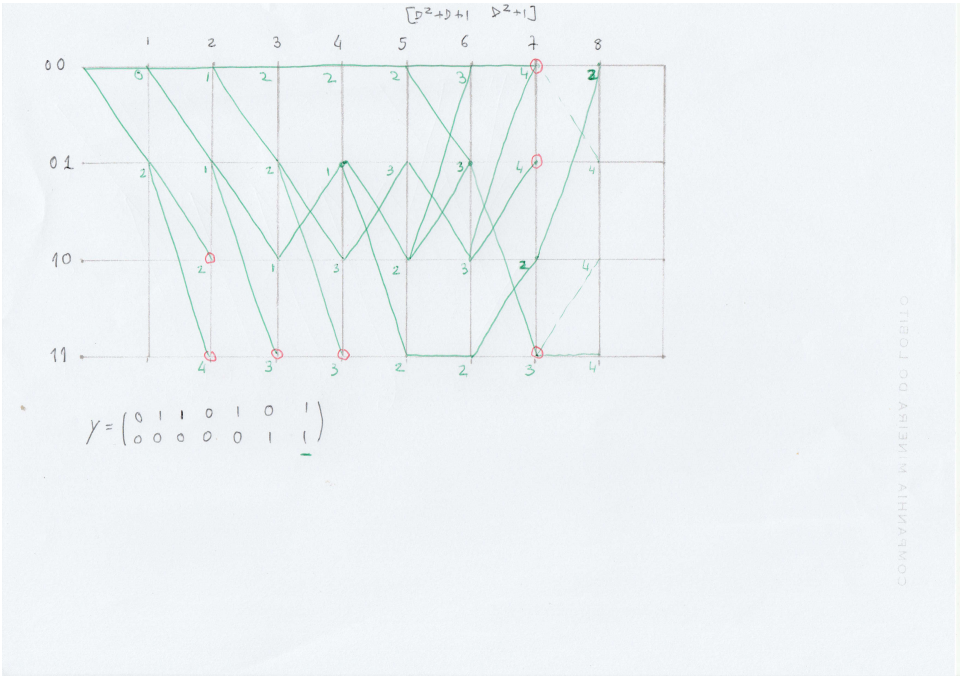


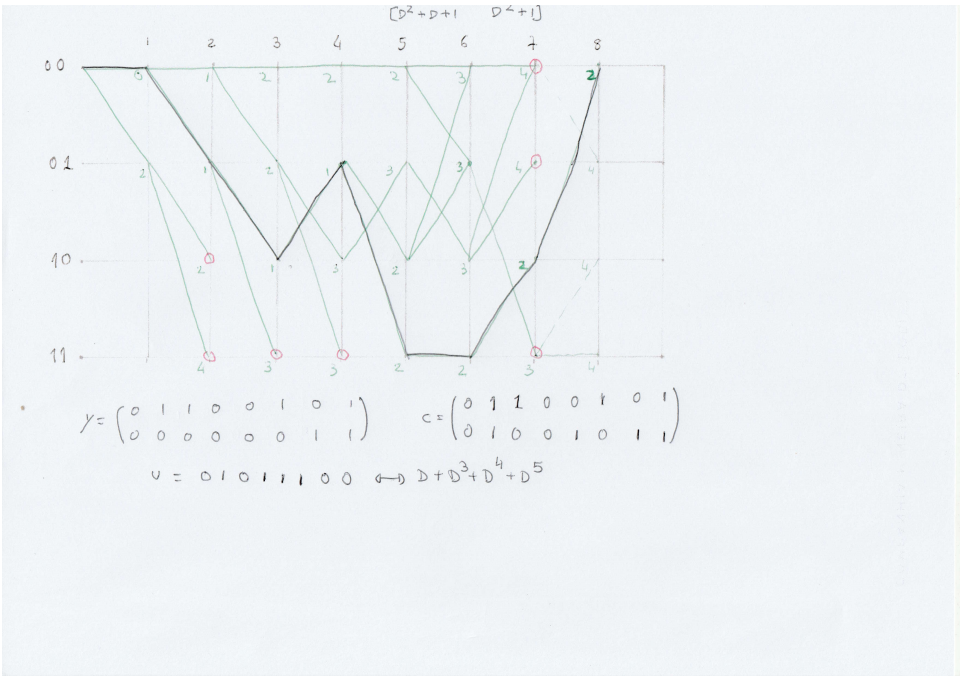

[D²+D+1 D²+1]
1 2 3 4
00
01
10
11
y = (0 1 1 0 0 1 0 1
0 0 0 0 0 0 1 1)


[D²+D+1 D²+1]
1 2 3 4 5
00
01
10
11
y = (0 1 1 0 0 1 0 1
0 0 0 0 0 0 1 1)


[D²+D+1 D²+1]
1 2 3 4 5 6
00
01
10
11
Y = (0 1 1 0 0 1 0 1
0 0 0 0 0 0 1 1)


[D²+D+1 D²+1]
1 2 3 4 5 6 7
00
01
10
11
Y = (0 1 1 0 0 1 0 1
0 0 0 0 0 0 1 1)


[D²+D+1 D²+1]
1 2 3 4 5 6 7 8
00
01
10
11
y = (0 1 1 0 1 0 1)
(0 0 0 0 0 1 1)


[D²+D+1 D²+1]
1 2 3 4 5 6 7 8
00
01
10
11
y = (0 1 1 0 0 1 0 1)
(0 0 0 0 0 0 1 1)
c = (0 1 1 0 0 1 0 1)
(0 1 0 0 1 0 1 1)
u = 0 1 0 1 1 1 0 0 ⟷ D + D³ + D⁴ + D⁵

A modified trellis diagram may be used to compute the free distance of the code in the following way:

- eliminate the edge from zero state at time 0 to the zero state at time 1;
- label each remaining edge starting at state zero and time 0 with the weight of the corresponding output;
- for each $s > 0$, label each state a at time s with the least weight of a path ending there and starting at the zero state and time 0;
- stop at time s_0 if the label w of state zero at time s_0 is less or equal than the labels of the remaining states at the same time.

The free distance is then equal to w, because we found a fundamental path with weight w and any other fundamental path will have weight larger or equal than w.

1.6. **Canonical Generator Matrices.** A convolutional code has many polynomial generator matrices. The complexity of the encoding and decoding processes depend heavily on the maximal degree of the entries of the generator matrix G. In fact, the number of states of the state diagram induced by G is the important parameter to determine that complexity. This number equals the sum, over all rows, of the maximal degree of the entries of each row of G, which is called the **external degree** of G. Formally:

Definition 21. *If $G = [g_{i\,j}(D)]$ is a polynomial generator matrix, $m_i = \deg(\mathrm{row}_i(G)) = \max_j \deg(g_{i\,j}(D))$, and the external degree of G is*

$$\mathrm{extdeg}(G) = \sum_i m_i.$$

Definition 22. *A polynomial generator matrix G of the convolutional code C is **canonical** if it has minimal external degree among all polynomial generator matrices for C, ie*

$$\mathrm{extdeg}(G) \leq \mathrm{extdeg}(MG)$$

for all nonsingular matrices $M \in M_{k\times k}(\mathbb{F}_2(D))$ such that MG is polynomial.

A matrix $M(D) \in M_{k\times k}(\mathbb{F}_2[D])$ is **unimodular** if it has determinant equal to 1. This is equivalent to $M(D)$ being invertible with polynomial inverse, ie $(M(D))^{-1} \in M_{k\times k}(F_2[D])$: in general a square matrix over a ring has inverse with entries in the ring if and only if its determinant is a unit (ie, invertible element) of the ring; the units of the ring $\mathbb{F}[D]$ are the nonzero elements of $\mathbb{F}$. So, for a more general field $\mathbb{F}$, unimodularity of $M(D) \in M_{k\times k}(\mathbb{F}[D])$ means $\det(M(D)) \in \mathbb{F}^\times$.

Definition 23. *A polynomial generator matrix G is **reduced** if*

$$\mathrm{extdeg}(G) \leq \mathrm{extdeg}(MG)$$

for all unimodular matrices M.

Obviously, being reduced is a necessary condition for a generator matrix to be canonical. However, it is not sufficient.

Definition 24. *The* ***internal degree*** $\operatorname{intdeg}(G)$ *of* G *is defined as the maximum of the degrees of all* $k \times k$ *minors of* G.
A polynomial generator matrix G *is* ***basic*** *if it has minimal internal degree among all polynomial generator matrices for* C.

Lemma 25. *Let* G *be a generator matrix.*

a) *If* T *is a nnsingular polynomial matrix (ie,* T *has entries in* $\mathbb{F}[D]$ *and* $\det(T) \neq 0$*) then* $\operatorname{intdeg}(TG) = \operatorname{intdeg}(G) + \deg(\det(T))$. *In particular,* $\operatorname{intdeg}(G) \leq \operatorname{intdeg}(TG)$ *with equality iff* T *is unimodular.*
b) $\operatorname{intdeg}(G) \leq \operatorname{extdeg}(G)$.

Proof. **HW**. □

Example 26.

$$G = \begin{bmatrix} 1 & D & 1+D^2 \\ D & 1+D^2 & 1+D+D^2 \end{bmatrix}$$

$$G_1 = \begin{bmatrix} D^2 & 1+D+D^3 & 1+D \\ 1+D+D^2 & D^2+D^3 & 1 \end{bmatrix} = \begin{bmatrix} D & 1+D \\ 1+D & D \end{bmatrix} G$$

are two generator matrices for the same code. The matrix

$$\begin{bmatrix} D & 1+D \\ 1+D & D \end{bmatrix}$$

is unimodular. The internal degree of G *and* G_1 *is* 4. *The externa degrees differ:* $\operatorname{extdeg}(G) = 4$ *and* $\operatorname{extdeg}(G_1) = 6$.

The importance of the last definitions lies in the following theorem, whose proof is given later:

Theorem 27. *A polynomial generator matrix* G *is canonical if and only if it is basic and reduced.*

Of course, this result would have no practical consequences if it was not for the existence of equivalent properties that make it possible to verify if a matrix is basic and/or reduced. We state some of these conditions in the next two propositions. Before that, we recall, without proof, some concepts and results from the algebraic theory of matrices over $\mathbb{F}[D]$, which are valid for any Euclidean domain:

Proposition 28. *Let* A *be a* $m \times n$ *matrix over* $\mathbb{F}[D]$, $m \leq n$. *Then there exist invertible matrices (over* $\mathbb{F}[D]$*)* P *and* Q, *such that the entries* s_{ij} *of* PAQ *satisfy*

i) $s_{ii} = d_i$, *for all* $1 \leq i \leq m$,
ii) $s_{ij} = 0$, *for all* $i \neq j$, *and*
iii) *the* d_i *are either monic polynomials or zero and* $d_i \mid d_{i+1}$, *for all* $1 \leq i < m$.

The coefficients d_i are called the **invariant factors** of A and are uniquely determined by the matrix.

Proposition 29. *For any* $1 \leq l \leq m$, $\prod_{i=1}^{l} d_i$ *is equal to the greatest common divisor of the* $l \times l$ *minors of* A.

Remark 30. *The matrix* PAQ *is usually called the* ***Smith Normal Form*** *of* A.

Proposition 31. *The following conditions are equivalent:*

1- *G is basic;*
2- *the invariant factors $d_1, \cdots, d_k$ of G are all equal to 1;*
3- *The greatest common divisor of the $k \times k$ minors of G is 1;*
4- *G has a polynomial right inverse: there exists a $n \times k$ polynomial matrix U such that $GU = I_k$;*
5- *if $x(D) = u(D)G \in (\mathbb{F}_2[D])^n$ then $u(D) \in (\mathbb{F}_2[D])^k$ (polynomial output implies polynomial input).*

Proof. The properties of the invariant factors stated above show that $2 \Leftrightarrow 3$ (**HW**). Notice also (**HW**) that, because G has rank k, its invariant factors are all nonzero. Suppose that G is basic and let $S = PGQ$ be its Smith Normal Form. Then G and $G_1 = PG = SQ^{-1}$ generate the same code and have the same internal degree, because P is unimodular. We may multiply G_1 on the left by the diagonal matrix D, with $D_{ii} = \frac{1}{d_i}$ and obtain a new generator

$$G_2 = \left[I_k | 0_{k\times(n-k)}\right] Q^{-1} = DPG;$$

but then (**HW**) we have

$$\text{intdeg}(G_2) = \text{intdeg}(G) - \sum_i \deg(d_i);$$

if some d_i is not 1, this contradicts the hypothesis of G being basic.
Assuming 2, we know that $PGQ = \left[I_k | 0_{k\times(n-k)}\right]$. We leave as an exercise (**HW**) to find a $n \times k$ polynomial matrix U satisfying 4.
The proof of $4 \Rightarrow 5$ is straightforward and is left as an exercise (**HW**).
Finally, assuming 5, if TG is a polynomial matrix, each of its rows is the encoding of a row of T and we conclude that T must in fact be a polynomial matrix. But then, by lemma 25, $\text{intdeg}(TG) \geq \text{intdeg}(G)$, ie, G is basic. □

Proposition 32. *The following conditions are equivalent:*

1- *G is reduced;*
2- *$\text{extdeg}(G) = \text{intdeg}(G)$;*
3- *If $\bar{G}$ is the binary $k \times n$ matrix defined by $\bar{G}_{i\,j}$ = the coefficient of D^{m_i} in $G_{i\,j}$, where $m_i = \deg(row_i(G))$, then $\bar{G}$ has rank k;*
4- *For any $u(D) \in (\mathbb{F}_2[D])^k$,*

$$\deg\left(u(D)G\right) = \max_i \left(\deg(u_i(D)) + \deg(row_i(G))\right)$$

Proof. We prove first that $1 \implies 3$. In order to better understand the proof, we notice that the definition of $\bar{G}$ means that, for each i,

$$\text{row}_i(G) = D^{m_i}\,\text{row}_i\,\bar{G} + \text{a row of polynomials with degree less than } m_i.$$

We assume, without loss of generality, that the m_i are ordered in nondescending order, and suppose that 3 is false: there exists then a nonzero vector $\mathbf{a} = (a_0 \quad \cdots \quad a_{k-1})$ such that $\mathbf{a}\bar{G} = 0$; let l be the highest index such that $a_l \neq 0$; then we conclude (**HW**) that

$$\sum_{t=0}^{l} a_t D^{m_l - m_t}\,\text{row}_t(G)$$

is a linear combination of rows of G with maximal degree strictly less than m_l. We may replace $\mathrm{row}_l(G)$ by this linear combination, by elementary row operations, obtaining a matrix with smaller external degree, a contradiction.
To prove that 3 $\implies$ 2, we notice that, assuming 3, there must exist a $k \times k$ submatrix V of $\bar{G}$ with nonzero determinant. For the corresponding $k \times k$ submatrix of G, the coefficient of $D^{m_0+\cdots+m_{k-1}}$ in its determinant is $\det(V) \neq 0$, showing that

$$\mathrm{intdeg}(G) \geq \sum_{i=0}^{k-1} m_i = \mathrm{extdeg}(G).$$

As the other inequality is always true, the proof is complete.
Now we verify that 2 $\implies$ 1: let T be a unimodular $k \times k$ matrix. Then

$$\mathrm{extdeg}(TG) \geq \mathrm{intdeg}(TG) = \mathrm{intdeg}(G) = \mathrm{extdeg}(G),$$

where we used Lemma 25 and the hypothesis 2.
Finally, we prove 3 $\Leftrightarrow$ 4. Let $u(D) \in \mathbb{F}^k[D]$ be an input and

$$c(D) = \sum_{i=0}^{k-1} u_i(D)\,\mathrm{row}_i(G)$$

the codeword. If $\deg(u_i(D)) = d_i$, the degree of $c(D)$, ie, the maximal degree of its entries, is at most $d = \max_i(d_i + m_i)$. Now, the vector of coefficients of D^d in $c(D)$ is (**HW**) $\mathbf{b}\bar{G}$, where $\mathbf{b} = (b_0, \cdots, b_{k-1})$ and b_i is the coefficient in $u_i(D)$ of D^{d-m_i}. At least one of these b_i is nonzero, by definition of d, and so $\mathbf{b}\bar{G}$ is not the zero vector, ie, condition 4 is satisfied, if and only if $\bar{G}$ has rank k, ie 3 is satisfied. □

Exercise 33. *Show that the matrix G in example 26 is canonical, while G_1 is basic but not reduced. find a polynomial right inverse for each one of them.*

We give now the proof of Theorem 27:

Proof. Suppose tht G is canonical. We must show that G is basic; let G_0 be a basic generator for the same code with minimum external degree, among all basic generators.
We verify that G_0 is necessarily reduced, because, if T is a unimodular matrix,

$$\mathrm{extdeg}(TG_0) \geq \mathrm{extdeg}(G_0).$$

Moreover, Lemma 25 shows that $\mathrm{intdeg}(TG_0) = \mathrm{intdeg}(G_0)$ and so TG_0 is also basic, for any unimodular T. We have

$$\mathrm{intdeg}(G_0) \leq \mathrm{intdeg}(G) = \mathrm{extdeg}(G) \leq \mathrm{extdeg}(G_0) :$$

the first inequality follows from the hypothesis that G_0 is basic, the equality and the last inequality from the hypothesis that G is canonical, and so reduced.
But we confirmed that G_0 is also reduced and so $\mathrm{intdeg}(G_0) = \mathrm{extdeg}(G_0)$, and all the values are equal, in particular, $\mathrm{intdeg}(G)$ is minimal, ie, G is basic.
Conversely, if G is basic and reduced, and G_1 is any other generator of the same code, Lemma 25 and the hypothesis imply

$$\mathrm{extdeg}(G_1) \geq \mathrm{intdeg}(G_1) \geq \mathrm{intdeg}(G) = \mathrm{extdeg}(G),$$

ie, G has minimal external degree. □

It is possible to prove also the following

Theorem 34. *Suppose G is a canonical generator of a code and G_1 is any generator of the same code. If the row degrees m_i of G and t_i of G_1 are both in nodescending order*

$$m_0 \leq \cdots \leq m_{k-1}, \qquad t_0 \leq \cdots \leq t_{k-1},$$

then $m_i \leq t_i$ for all $0 \leq i < k$.
In particular, the canonical generator matrices of a code have all the same sequence of row degrees.

Proof. (A slightly more difficult **HW**).
Hint: assume that, for some choice of canonical G and any G_1, there exists an i_0 such that $t_{i_0} < m_{i_0}$ and $m_i \leq t_i$ for all $i < i_0$. By proposition 31, the rows $\mathrm{row}_i(G_1)$ are linear combinations of the rows $\mathrm{row}_i(G)$ with polynomial coefficients; by Proposition 32, the rows $\mathrm{row}_i(G_1)$ with $i < i_0$ are linear combinations of the $\mathrm{row}_i(G)$, with $i < i_0$. Justify these claims and deduce a contradiction.
The last statement of the Theorem is immediate. □

Remark 35. *The common row degrees of all canonical generator matrices of a code are called the* ***Forney indices*** *of the code.*