**Part II.**  **Low-density parity check code**

The second phase of this project consists in implementing the LDPC encoder and decoder. is proposed to proceed progressively: first a small size example is considered to study the main principles of the LDPC encoder/decoder; second the actual LDPC code used in the DVB-S2 standard is investigated based on which the performance gain can be assessed.

The following steps are discussed:

1. The small-size LDPC encoder and hard decoder is implemented. The encoder simply consists in modulo-2 multiplying blocks of bits with the generator matrix computed from the parity check matrix. A function "decoder" implementing the iterative hard decoder based on the Tanner graph is written.

2. The DVB-S2 LDPC encoder and hard decoder is simulated. For the encoder, the Matlab functions "makeLDPC" and "makeParityChk" are used.

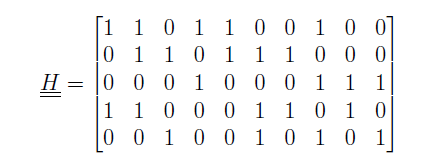
As for the decoder, the function developed in the first step is used. The channel coding gain as a function of the number of iterations is illustrated.

**3.1 Channel encoder**

For a small size of the LDPC encoder and decoder example, the example H matrix in the lab notes is used. First the function "gen2par" is used for changing the parity check matrix into generator matrix in the standard form. Then, for encoding, the input bits’ stream should be divided into blocks to meet the same dimensions with the generator matrix and then, a simple multiplication between the generator matrix G and the bits stream should be performed.

A small-size block code of rate 1/2 is first considered. It is defined by the parity check matrix

provided by:

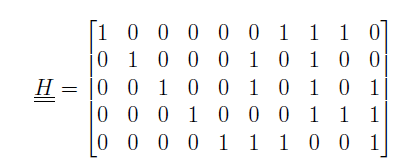


In the case of LDPC codes, the parity check matrix is sparse: it is composed of a lot of zeros and a few ones.

Based on the knowledge of the parity check matrix, the generator matrix can

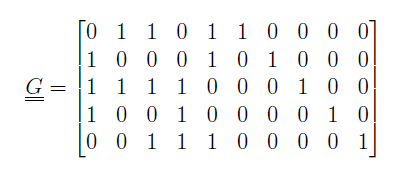
be computed. In the case of a systematic code, the parity check matrix can be written as:





and the generator matrix is easily deduced:





Therefore the rows of the proposed parity check matrix must be first linearly combined in order

to create an identity matrix at the left side of the matrix.

When the generator matrix is known, the encoder simply consists in dividing the bit stream into

blocks of bits and by modulo-2 multiplying each block with the generator matrix. It should be

noted that the implementation of the encoder can be optimised when the size of the generator

matrix is large to avoid full matrix products. This is beyond the scope of this project.

**3.2 Iterative decoding**

By using the encoder and decoder, the BER is different from the one obtained before. The encoder and decoder perform corrections for the transmitted sequences, so the BER is smaller for the same value of *Eb/N*0, compared to the situation without the encoder and decoder.

!! here we can talk about channel coding gain for multiple iterations

Also we can make a comparison between the encoding and decoding efficiency, and the BER curve with a big parity check matrix [128x256] and a small one [5x10]!! I 🡪 depends on the result of the code . !!

1. **Key Points**
   1. **Simulation Part**
2. ***When building the new BER curves, do you consider the uncoded or coded bit energy on the x-axis?***

The uncoded bit energy is considered on the x-axis, because the bit-energyremains the same when using the encoder and decoder.

1. ***How do you limit the number of decoder iterations****?*

By limiting the number of iterations. Theoretically the iteration should stop when the estimated codeword satisfies all the check equations, so that *u.H* = 0

1. ***Why is it much simpler to implement the soft decoder for BPSK or QPSK than for 16-QAM or 64-QAM?***

The soft decoder is not implemented in our case, instead a theoretical explanation is given. From the block diagram shown in Figure 13, the soft decoder does not use the data after the demodulation, and instead, it uses the symbols after the down-sampling operation. As opposed to the soft decoder, the hard decoder must deal with the demodulated symbols(bits); but for soft decoder, it has to consider the symbols and the modulation type. The judgments of soft decoder are based on the Euclidean distance

between the received symbols and the transmitted symbols. For BPSK and QPSK modulation, it is quite easy to calculate the distance but for the QAM, it is quite hard to judge because many symbols laid on one direction and the distribution is pretty intensive.

* 1. **The Communication System Part**

1. ***Demonstrate analytically that the parity check matrix is easily deduced from the generator***

***matrix when the code is systematic.***

A systematic code is any error-correcting code in which the input data is embedded in the encoded output. For this case, the generator matrix consists of an identical matrix and a transformed parity matrix. So the parity check matrix H can be deduce by just get the parity matrix of generator matrix transformed back and then put the identical matrix on the other side. So it is very easy to deduce.

1. ***Explain why we can apply linear combinations on the rows of the parity check matrix to***

***produce an equivalent systematic code.***

A generator matrix is a matrix whose rows form a basis for a linear code.

The code words are linear combinations of the rows of this matrix, that is, the linear code is the row space of its generator matrix. The rows of a parity check matrix are the coefficients of the parity check equations.

That is, they show how linear combinations of certain components of each codeword equal zero.

1. ***Why is it especially important to have a sparse parity check matrix (even more important***

***than having a sparse generator matrix)?***

The generator matrix can be deduced from the H matrix if we rearrange the matrix into an identical matrix with the parity matrix; The decoder is using H matrix to decode and also, we need to use the H matrix to see how’s the corrections going on and decide when we can stop the iteration based on that correction.

1. ***Explain why the check nodes only use the information received from the other variable***

***nodes when they reply to a variable node.***

The check nodes only use the information received from the other variable nodes when they reply to a variable node, because the check nodes think the variable node they are replying to is nor right nor wrong.