ELEC-H-401: Modulation and Coding

François Horlin

Contents

1 HFC architecture

Figure 1 gives a block diagram of a common network configuration utilised by cable operators to deliver broadcast television, internet access, and telephone service based on voice-over-IP (VoIP) protocols. This network architecture, known as HFC (Hybrid Fiber Co-ax), combines the use of both fiber-optic (FO) and co-axial cables.

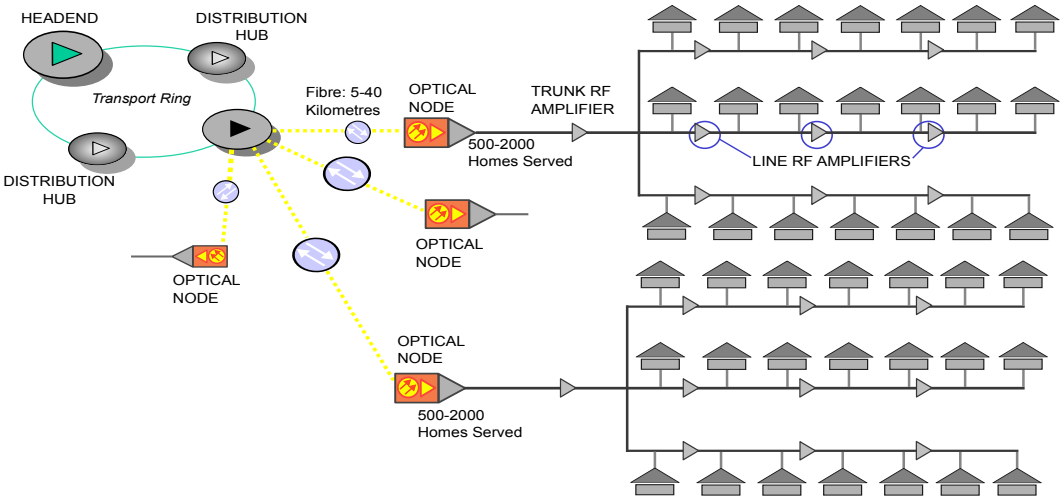

Figure 1: Typical HFC network

The network begins at the headend, which houses the links to the service providers. Initially, the signal is transmitted over fiber-optic cables, then subsequently through trunk cables, distribution cables, and finally, each subscriber is connected via a dedicated drop cable. The adoption of fiber-optic cables for the initial distribution is advantageous due to their lower loss characteristics compared to co-axial cables. Moreover, fiber-optic cables are more compact, lightweight, and cost-effective, efficiently covering long ranges while minimising signal degradation. To further extend the range of the system, trunk cables are employed. As each amplifier introduces noise and distortion, trunk cables have generally a large diameter to minimise signal loss. To compensate for cable attenuation, an amplifier is required every 1 km along the co-axial trunk cable. Distribution cables, which traverse each street and pass every house, are utilised to deliver the signal to individual subscribers. Multiple amplifiers are strategically placed along the distribution cable to account for the loss of signal power mainly caused by the splitters. Each subscriber connects to a splitter on the distribution cable using a flexible co-axial drop cable, typically less than 50 m in length.

Depending on the co-axial cable and its length, up to approximately 1 GHz bandwidth is available for communication. Television and internet signals are multiplexed in frequency on channels of 6 to 8 MHz bandwidth each. For transmission on the fiber-optic cable, the frequency-multiplexed signal modulates in amplitude a laser-generated optical carrier. At the receiver, a photodiode recovers the original electrical signal. Note that even though the electrical signal itself contains many modulated carriers at frequencies up to about 1 GHz, it is seen as a single low-pass baseband signal by the optical modulator/demodulator.

The HFC architecture is still continuously evolving for improved capacity and easier management. All modern cable systems are bidirectional to support interactive services: downstream direction from the headend to the subscriber, upstream direction from the subscriber to the headend. The recent evolution further involves the push of the optical fiber deeper in the network to lower the number of subscribers sharing the same co-axial cable and the fiber digitalisation.

2 DVB-C and DOCSIS standards

In Europe, broadcasters and consumer electronics companies formed a consortium for the development of digital television, known as Digital Video Broadcasting (DVB). The DVB project designed and approved several digital television standards for terrestrial, satellite and cable transmission. The Digital Video Broadcasting-Cable (DVB-C) standard in particular specifies the physical layer signalling for the transmission of digital television on co-axial cable networks. Compared to other DVB channels, the co-axial cable is characterised by a high signal-to-noise ratio (SNR) and does not suffer from multipath propagation. High transmission capacity is possible with high order modulations up to 256-QAM applied on single carrier channels.

The cable originally foreseen to support television services only has more recently been modernised to further provide access to internet. Data Over Cable Service Interface Specification (DOCSIS) is the telecommunications standard developed by the cable research consortium CableLabs to permit bi-directional data transfers over existing HFC infrastructure. DOCSIS covers the three first communication layers (physical, control, network) and has progressed through various versions, each one providing improved data rates, enhanced Quality of Service, and security. Notably, the recent evolution steps involved the channel bonding (DOCSIS 3.0), the use of orthogonal frequency-division multiplexing (OFDM) modulation (DOCSIS 3.1) and the full-duplex operation (DOCSIS 4.0). Channel bonding consists in allocating multiple channels to the subscribers. OFDM is a wideband modulation that defines the channel frequency grid digitally. Full-duplex consists in re-using the same frequencies for upstream and downstream channels.

Today, the DVB-C and DOCSIS signals co-exist on the HFC infrastructure. Over the years, a clear convergence is observed between the DVB-C and DOCSIS technologies.

3 Project objective and organisation

The objective of the project is to design and simulate a representative DVB-C transmission chain in Matlab The HFC channel can be well modelled as an ideal channel only corrupted by additive white Gaussian noise (AWGN). Your simulation will include all the functionality usually found in a typical modem, and is organised in three main steps:

- the optimal communication chain over the ideal channel;
- the time/frequency synchronisation algorithms;
- the Low-density parity check code (LDPC) channel encoder and decoder.

Once the transmission chain is working in Matlab, it is nice to validate it by communicating the signals on a real-life HFC setup. Orange Belgium has made such a setup available at ULB premises for learning purposes. To interface the setup with your Matlab code, the Adalm Pluto hardware developed by Analog Instruments will be used. It supports the full-duplex communication (simultaneous transmission and reception) of signals a 20 MHz bandwidth.

A good understanding of the theory is necessary to start implementing the project. The present document describes the content of the project. The explanations are mainly intended to draw your attention on the most important parts, not to give a detailed introduction to the theory. Questions are asked at the end of each section to make the link with the theory presented during the lectures.

The project is organised in groups of 2 to 3 students. The deliverable is a written report of no more than 20 pages (ReportELEC401_your_names.pdf) and a .zip file of your Matlab code

(CodeELEC401_your_names.pdf). They have to be sent to *francois.horlin@ulb.be* before the exam session. The report should focus on the presentation of the simulation results along with a clear explanation of the observations. A brief answer to the questions found at the end of each section is also required.

4 Optimal communication chain over the ideal channel

A block diagram of the communication chain simulated in the first phase of this project is represented in Figure 2. The LDPC encoder and hard/soft decoder blocks are not activated yet.

4.1 Symbol mapping

The binary sequence is first transformed in a sequence of complex symbols in order to improve the spectral efficiency of the system. High order symbol constellations up to 256-QAM are supported in the DVB-C systems. The symbols are transmitted at a typical 5 Msymb/s rate on the channel. The bit rate is equal to the symbol rate multiplied by the number of bits per symbol.

At the receiver, the constellation points are corrupted by additive noise. The detection consists in selecting the symbol constellation points the closest to the received points.

4.2 Nyquist filter

The transmitted signal is shaped with a halfroot Nyquist filter, that limits the signal occupancy to the desired frequency spectrum. We assume that the Nyquist filter has 0.2 roll-off factor, which implies that the communication bandwidth is equal to 6 MHz. Before filtering, the symbol sequence is over-sampled with a factor $M > 1$. The sample rate fixes the bandwidth simulated in Matlab, that must be high enough to simulate the halfroot Nyquist filtering.

A filter matched to the transmitter filter is applied at the receiver - it is therefore also a halfroot Nyquist filter - so that the signal-to-noise power ratio (SNR) is maximised at its output. The two halfroot Nyquist filters together form a Nyquist filter, that reduces to a dirac pulse when it is sampled at the symbol rate from its maximum. The interference between the successive symbols is therefore completely cancelled out. The time synchronisation is here limited to the selection of the correct samples at the receiver.

4.3 Noise addition

The communication performance is limited by AWGN. It is a common model to represent the combined contribution of the effects that corrupt the received signal.

The aim of the simulation is to assess the bit error rate (BER) as a function of the bit energy on noise one-sided power spectrum density ratio (E_b/N_0). The simulations should confirm the theoretical results provided in the course. Since the simulation is only performed at baseband (the frequency up-conversion at the transmitter and down-conversion at the receiver is not simulated), the baseband equivalent model of the noise must be used.

4.4 Steps

- The first step is to implement the symbol mapping and demapping. A stream of random bits is generated and transformed in a stream of complex symbols. The Matlab functions "mapping" and "demapping" may be used for this purpose.

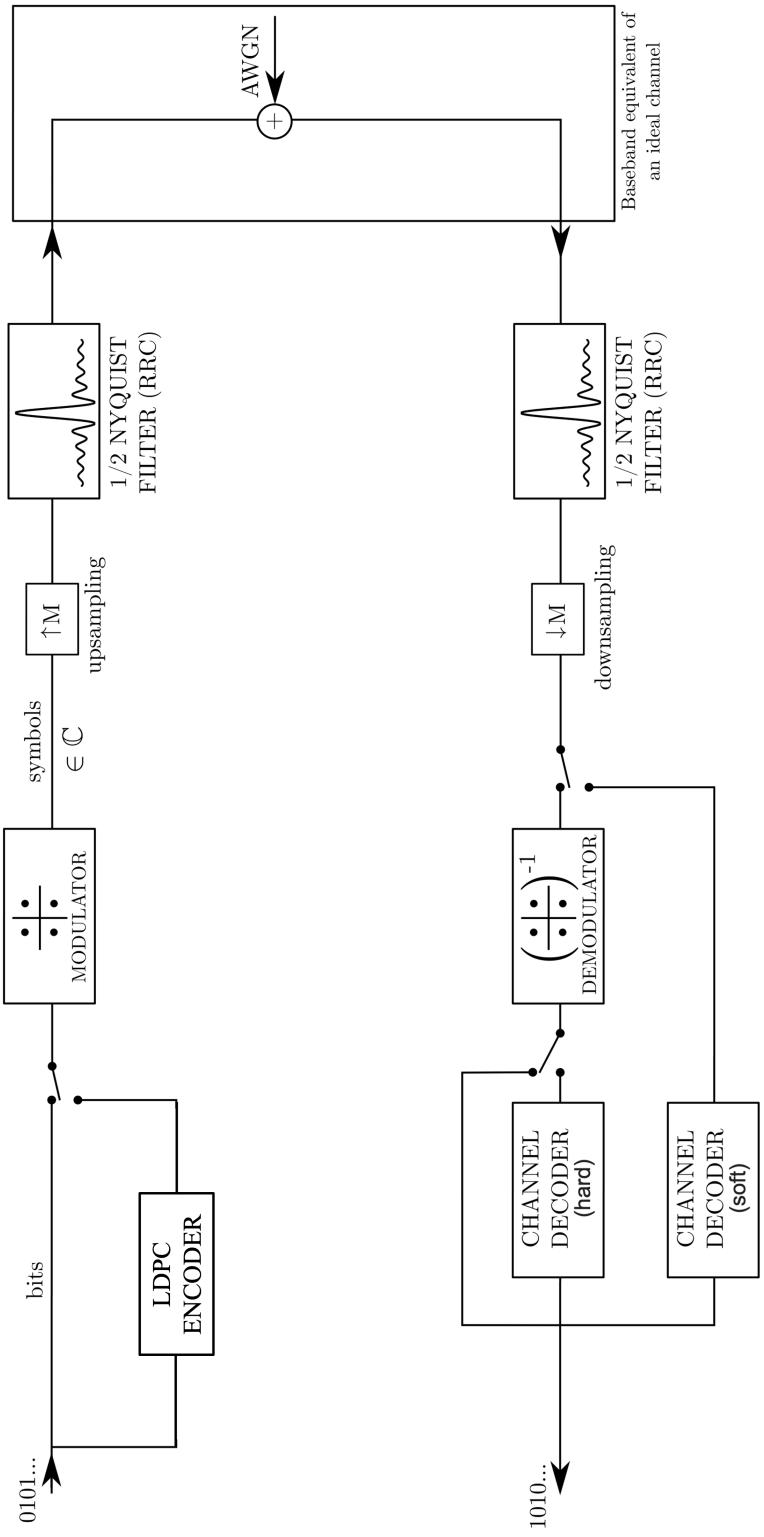

Figure 2: Block diagram of the communication system

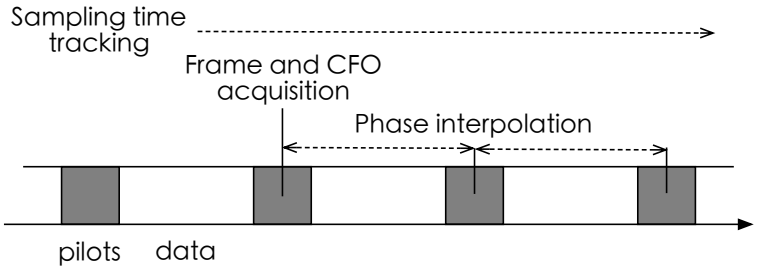

Figure 3: DVB-C frame

- The second step is to write a new function that computes the halfroot Nyquist filter. The function should implement the expression of the filter in the frequency domain and translate the result to the time domain. The transmit and receive signals can afterwards be generated. **Illustrate the two main properties of the Nyquist filter by simulations: the limited communication bandwidth, the cancellation of the inter-symbol interference.**
- The third step consists in adding noise in the communication chain and in evaluating its impact. You should correctly settle the power of the noise relatively to the power of the signal. **For the different constellation sizes, validate the theoretical BER curves by simulations.**

4.5 Questions

Regarding the simulation:

- It is proposed to use the baseband equivalent model of the AWGN channel. Would it be possible to work with a bandpass implementation of the system?
- How do you choose the sample rate in Matlab?
- How do you make sure you simulate the desired E_b/N_0 ratio?
- How do you choose the number of transmitted data packets and their length?

Regarding the communication system:

- Determine the supported (uncoded) bit rate as a function of the physical bandwidth.
- Explain the trade-off communication capacity/reliability achieved by varying the constellation size.
- Why do we choose the halfroot Nyquist filter to shape the complex symbols?
- How do we implement the optimal demodulator? Give the optimisation criterion.
- How do we implement the optimal detector? Give the optimisation criterion.

5 Time and frequency synchronisation

Before the communication can take place, it is important to take care of the time and frequency differences existing between the two sides of the link. In this project, we are mainly interested in the following effects:

- the carrier frequency error (CFO) and the sample clock offset (SCO), existing because of the limited accuracy of the local oscillators;
- the carrier phase error and the sample time shift, existing because the transmitter and receiver are physically not at the same location.

Synchronisation algorithms are designed to estimate and compensate for those effects. They are usually organised in two main steps: the acquisition implemented before the data communication takes place to provide a rough estimate of the effects; the tracking implemented when the data are communicated to refine the estimates and follow the time variation of the effects. Figure 3 illustrates the construction of the frames designed to support both acquisition and tracking steps. Pilot symbols are regularly interleaved in the data symbols. The synchronisation is organised as follows:

- the acquisition and tracking of the sampling time instants (Gardner algorithm);
- the acquisition of the frame starting time;
- the acquisition of the carrier frequency offset;
- the interpolation of the phase drift between the pilot sequences.

The third phase of this project consists in assessing the impact of the synchronisation errors on the system performance and designing algorithms to compensate for them.

5.1 Impact of the synchronisation errors

In order to design the synchronisation algorithms, it is necessary to know the gap existing between the initial value of the effects and the robustness of the communication chain to those effects. This fixes the specifications on the design of synchronisation algorithms.

The carrier and sampling clock frequency accuracy mostly depends on the hardware used to implement the local oscillators. The local oscillators often integrate a crystal of accuracy higher than 10 ppm (parts per million). Therefore, the CFO is at most equal to the carrier frequency assumed equal to 600 MHz times 10^{-5} and the SCO is at most equal to the sample clock frequency times 10^{-5}. Obviously, the frequency phase error can take any value between 0 and 2π and the sampling time shift can take any value between 0 and the symbol duration.

On the other hand, the impact of the synchronisation errors on the system performance must be assessed by simulations. The baseband model of the communication should be updated to include the synchronisation errors. The CFO and the carrier phase error are implemented by multiplying the received signal by $\exp\left(j(2\pi\Delta f t + \phi)\right)$ where Δf is the CFO and ϕ is the phase error. Implementing the SCO and sample time shift is much more difficult as it requires a high complexity interpolation between the samples. In this project, it is proposed to significantly increase the sampling rate to be able to simulate sampling time shifts with a sufficient accuracy. The SCO is neglected to simplify the discussion.

5.2 Gardner algorithm

The Gardner algorithm is a non-data aided (NDA) feedback algorithm used to compensate for the sampling time errors. It can not only deal with the initial sampling time shift but also track the variations over the time due to the SCO (not simulated here). As it is robust to the other synchronisation effects, and especially to the CFO and phase offset, it is applied first.

The algorithm works by computing one estimate of the error per symbol and by correcting it progressively over the time in a feedback structure. The time shift estimate at the next time instant $\hat{\epsilon}[n+1]$ is the current estimate $\hat{\epsilon}[n]$ plus a weighted version of the error. By properly selecting the error weight κ, the algorithm averages the error over the time and converges therefore to the correct value. The error is estimated by multiplying the output signal midway between two symbols by the signal slope evaluated by subtracting the value between the two symbols. The sign of the error gives the direction of the time shift while its magnitude gives the importance of the time shift. We get:

$$\hat{\epsilon}[n+1] \;=\; \hat{\epsilon}[n] \;-\; \kappa \cdot \Re\Big[y_{\hat{\epsilon}[n]}[n-0.5]\,(y^*_{\hat{\epsilon}[n]}[n] - y^*_{\hat{\epsilon}[n-1]}[n-1])\Big] \tag{1}$$

The algorithm requires to work with two samples per symbol.

5.3 Frame and frequency acquisition

The frame and frequency acquisition is performed by using data-aided (DA) algorithms in a feedforward structure.

The frame acquisition consists in estimating the pilot sequence position in the sequence of received samples so as to know when the data can be received in the frame. The position of the pilots can be obtained by correlating the received sequence with the known pilot sequence and by selecting the peak in magnitude. Even if the algorithm is optimal according to the maximum likelihood (ML) criterion in the absence of CFO, its performance is strongly degraded when there is CFO in the system. Therefore a differential cross-correlator is instead implemented of output metric given by:

$$D_k[n] \;=\; \frac{1}{N-k} \sum_{l=k}^{N-1} (y^*[n+l]\,a[l])\;(y^*[n+l-k]\,a[l-k])^* \tag{2}$$

where $y[n]$ is the received sequence and $a[n]$ is the pilot sequence. The metric is built by multiplying two time-shifted windows of the correlation with the pilot sequence. The pilot position is selected to maximise an average of the output metric over K values of the time shift:

$$\hat{n} \;=\; \arg\max_n \sum_{k=1}^{K} |D_k[n]| \tag{3}$$

The CFO can also be estimated by observing the metric at the output of the differential cross-correlator. The phase difference existing due to the CFO between the two windows of the cross-correlation with the pilot sequence is averaged taking the varying time shift into account:

$$\hat{\Delta f} \;=\; -\frac{1}{K} \sum_{k=1}^{K} \frac{\angle D_k[\hat{n}]}{2\pi k T} \tag{4}$$

where T is the symbol duration. Once the CFO is estimated, it can be compensated on the upcoming data symbols.

5.4 Phase interpolation

The CFO estimation achieved during the acquisition phase if accurate enough to get rid of the inter-symbol interference (ISI). However there still remains a small CFO that causes a linearly varying phase over the time. Also the initial carrier phase offset has not yet been tackled.

A simple carrier phase tracking scheme consists in linearly interpolating the phase existing between two consecutive pilot sequences. The length of the pilot sequence is selected to ensure an accurate estimation of the phase. The length of the data sequence is selected to avoid ambiguities in the phase interpolation between two pilot sequences. Both parameters are on the other hand also selected to prevent a too large pilot overhead in the data communication.

5.5 Steps

- The first step consists in updating the channel baseband model to include the CFO and the carrier phase error and in assessing their impact on the BER performance. Your analysis should separately investigate the impact of the phase drift over the time, easily compensated on the received symbols, and of the ISI, hardly removed from the received symbols. **Illustrate the BER degradation for increasing values of the CFO and carrier phase error.**
- The second step consists in updating the channel baseband model to include the sample time shift and in assessing its impact on the BER performance. The SCO is neglected in the analysis. **Illustrate the BER degradation for increasing values of the sample time shift.**
- The third step is to write a new function implementing the Gardner algorithm so as to obtain the correct samples at the output of the matched filter. Starting from the signal sampled at a high rate, linear interpolations between adjacent samples can be performed to estimate the signal at arbitrary time positions. **Illustrate the convergence of the algorithm at the desired SNR for different values of the error weight. The robustness of the algorithm to the CFO should also be checked.**
- The fourth step is to write a new function implementing the frame and frequency acquisition. **Illustrate the remaining time/frequency error variances at the desired SNR for a varying pilot sequence length and parameter K. The robustness of the frame acquisition to the CFO should also be checked.**

5.6 Questions

Regarding the simulation:

- Derive analytically the baseband model of the channel including the synchronisation errors.
- How do you separate the impact of the carrier phase drift and ISI due to the CFO in your simulation?
- How do you simulate the sampling time shift in practice?
- How do you select the simulated E_b/N_0 ratio?
- How do you select the lengths of the pilot and data sequences?

Regarding the communication system:

- In which order are the synchronisation effects estimated and compensated. Why?
- Explain intuitively how the error is computed in the Gardner algorithm. Why is the Gardner algorithm robust to CFO?
- Explain intuitively why the differential cross-correlator is better suited than the usual cross-correlator? Isn't interesting to start the summation at $k = 0$ (no time shift)?
- Are the frame and frequency acquisition algorithms optimal? If yes, give the optimisation criterion.

6 Low-density parity check code

Structured redundant information is added at the transmitter in the channel encoder to improve the system robustness to transient effects. The channel decoder exploits the structure of the redundant information to correct the errors caused at the receiver. The second phase of this project consists in implementing the LDPC encoder and decoder. It is proposed to proceed progressively: first a small size example is considered to study the main principles of the LDPC encoder/decoder; second a larger size LDPC code is investigated based on which the performance gain can be assessed.

6.1 Channel encoder

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

$$\underline{\underline{H}} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{5}$$

In the case of LDPC codes, the parity check matrix is sparse: it is composed of a lot of zeros and a few ones in order to keep the complexity of the decoder sufficiently low.

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:

$$\underline{\underline{H}} = \begin{bmatrix} \underline{\underline{I}} & \underline{\underline{P}}^T \end{bmatrix} \tag{6}$$

and the generator matrix is easily deduced:

$$\underline{\underline{G}} = \begin{bmatrix} \underline{\underline{P}} & \underline{\underline{I}} \end{bmatrix} \tag{7}$$

Therefore the rows of the proposed parity check matrix must first be linearly combined in order to create an identity matrix at the left side of the matrix. In that case, the codeword is composed of the parity bits and of the message itself.

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.

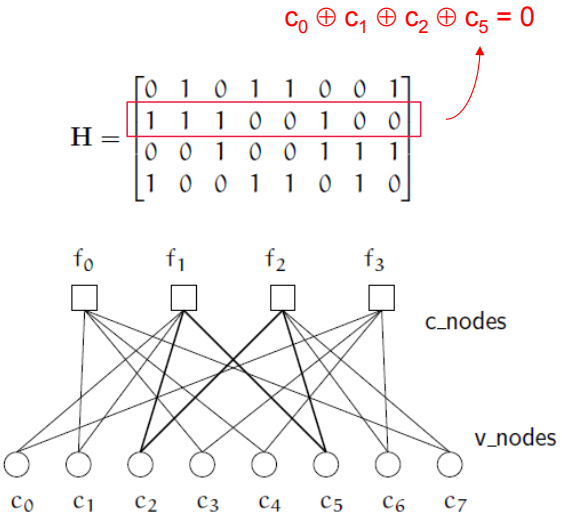

Figure 4: Construction of Tanner graph

6.2 Iterative decoding

The LDPC decoder relies on the construction of the Tanner graph that represents the multiplication of the received block (in the variable nodes) with the parity check matrix (in the check nodes). It is shown in Figure 4. In case of hard decoding, the iterative process consists in exchanging binary information between the variable nodes and the check nodes:

- The variables nodes transmit their most probable bit to the check nodes they are connected to;
- The check nodes reply to each variable node the most probable bit for them based on the binary information received from the other variable nodes.

The iterative process stops when all check equalities are verified.

Because of the small size of the encoder, the observed channel coding gains are negligible. The next step consists in implementing a LDPC encoder/decoder of much larger dimension.

6.3 DVB-C LDPC code

The sizes of the generator and parity check matrices increase drastically in the case of the DVB-C LDPC code. It is therefore too complex to implement the channel encoder by a matrix product. In this project, we propose to make use of the Matlab functions 'generate_ldpc' and 'encode_ldpc' as follows:

```
% Create initial parity check matrix of size 128 x 256
H0 = generate_ldpc(128, 256,0,1,3);

% Compute parity bits (128 x nb of packets) and generate final parity check matrix
[paritybits, H] = encode_ldpc(infobits, H0, 0);
```

The decoder constructed based on the Tanner graph can on the other hand still be used for large size LDPC codes. The performance of the DVB-C LDPC hard decoder can now be assessed. The soft decoder significantly outperforms the hard decoder and should therefore also be considered.

6.4 Steps

- The first step is to implement the small-size LDPC encoder and hard decoder. The encoder simply consists in modulo-2 multiplying blocks of bits with the generator matrix computed from the parity check matrix (5). A function implementing the iterative hard decoder based on the Tanner graph must be written. At this point, you can only check that your functions are working properly. A special care should be taken to the efficiency of your implementation in Matlab.
- The second step is to simulate the DVB-C LDPC encoder and hard decoder. As for the encoder, the Matlab functions 'generate_ldpc' and 'encode_ldpc' may be used. As for the decoder, you should use your own function developed in the first step. **Illustrate the channel coding gain as a function of the number of iterations.**
- The third step consists in comparing the hard and soft decoder performance. To simplify the implementation of the soft symbol detector, you can limit the discussion to the BPSK modulation. **Illustrate the channel coding gain achieved with hard and soft decoding after convergence.**

6.5 Questions

Regarding the simulation:

- When building the new BER curves, do you consider the uncoded or coded bit energy on the x-axis?
- How do you limit the number of decoder iterations?
- Why is it much simpler to implement the soft decoder for BPSK or QPSK than for larger QAM constellation orders?

Regarding the communication system:

- Demonstrate analytically that the parity check matrix is easily deduced from the generator matrix when the code is systematic.
- Explain why we can apply linear combinations on the rows of the parity check matrix to produce an equivalent systematic code.
- Why is it especially important to have a sparse parity check matrix (even more important than having a sparse generator matrix)?
- Explain why the check nodes only use the information received from the other variable nodes when they reply to a variable node.

7 Real-life experimentation on the HFC setup

The Adalm-Pluto SDR Active Learning Module can directly be interfaced with Matlab after installing the proper driver. In Matlab, the following piece of code should be used. It starts by defining the transmitter and receiver system objects, mainly settling the hardware parameters:

```
% Transmitter system object
txPluto = sdrtx('Pluto',...
                'RadioID', 'usb:0',...
                'Gain', -10,...  % -90 to 0 dB
                'CenterFrequency', fcarrier,... % 335e6 to 3.8e9 [Hz]
                'BasebandSampleRate', Rsamp); % 60e3 to 60e6 [Hz]

% Receiver system object
rxPluto = sdrrx('Pluto',...
                'RadioID', 'usb:0',...
                'CenterFrequency', fcarrier,... % 335e6 to 3.8e9 [Hz]
                'GainSource', 'Manual',... % Manual, AGC Slow Attack
                'Gain', 10,... % -4 to 71 [dB]
                'BasebandSampleRate', Rsamp,... % 60e3 to 60e6 [Hz]
                'EnableBurstMode', true,...
                'NumFramesInBurst', 1,...
                'SamplesPerFrame', 200000,...
                'OutputDataType', 'double' );
```

Next the function transmitRepeat is used to start a continuous periodic transmission of the signal buffer communicated in the parameters. Note that since the buffer is periodically transmitted, the transients effects at the beginning and at the end of the signal due to the filter memory should first be removed.

```
% Periodic transmission
txPluto.transmitRepeat(buffer_tx);
```

Finally, a few periods of the signal are recorded by calling the method associated with the receiver system object. It enables the detection of overflows in case the samples are not properly received in Matlab. Since the transmitted signal is infinitely periodic, no difficult sample synchronisation needs to be foreseen when recording the signal.

```
% Burst reception
[data_rx,datavalid,overflow] = rxPluto();

if (overflow),
    disp('Samples dropped');
end
```

8 Useful Matlab functions

```
rand  % uniform random variable
randi  % integer random variable
```

```
randn  % Gaussian random variable

abs  % module
angle  % phase

fft  % fast Fourier transform
ifft  % inverse fast Fourier transform
fftshift  % spectrum centering around "0"

reshape  % matrix element re-organization
permute  % matrix dimension permutation

figure  % open a figure
subplot  % divide a Figure
plot  % illustrate a function
stem  % illustrate a sequence
grid  % add a grid
axis  % define the axes
xlabel/ylabel  % give a name to the axes
title  % add a title
legend  % add a legend
hold  % illustrate multiple functions
```