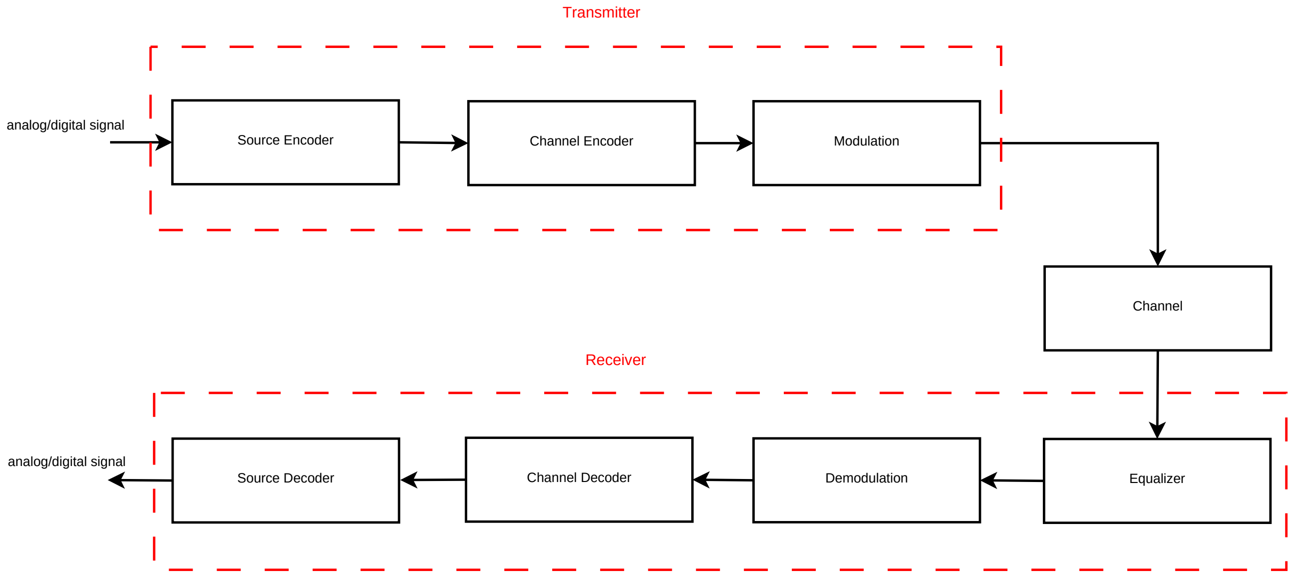
Institute for System Dynamics
Prof. Dr.-Ing. C. Tarín
University of Stuttgart

System Dynamics Laboratory

- Winter Semester 2020-

Experiment 1: Communications and Filter Technology

Version 1.1.8



Contents

1 Introduction

In the *Communication and Filter Technology Laboratory* a speech transmission system is implemented and analyzed. For the implementation and analysis Mathworks Matlab/Simulink is used which provides the Communication Systems and the Digital Signal Processing Toolbox. These toolboxes will be used in the laboratory exercises. The components which will be studied during the laboratory are the following:

- Source Coding
- Channel Coding
- Modulation
- Channel models
- Interleaving
- Equalization

For Analysis the following tools are used:

- Constellation diagram
- Eye diagram
- Bit error rate

Hint:

There are two different types of exercises:

Pre-Laboratory Assignment – V1.0 –
All pre-laboratory assignments have to be done before the laboratory date. They will be used to grade the laboratory. Please upload one solution per group to your group folder in ILIAS. In case of problems with the pre-laboratory assignments, you can ask the supervisors on the laboratory date. Supervisors are free to let you fail the assignment if they feel you haven't prepared anything.

Laboratory Assignment – P1.0 –
The laboratory assignments are done during the experimental part of the laboratory. Note that the last two assignments are optional. In case you are done with the previous assignments and still have some time left, consider challenging yourself with these.

2 Introduction to Communication Systems

In Figure 1 the main components of a communication system are shown. It consists of three parts:

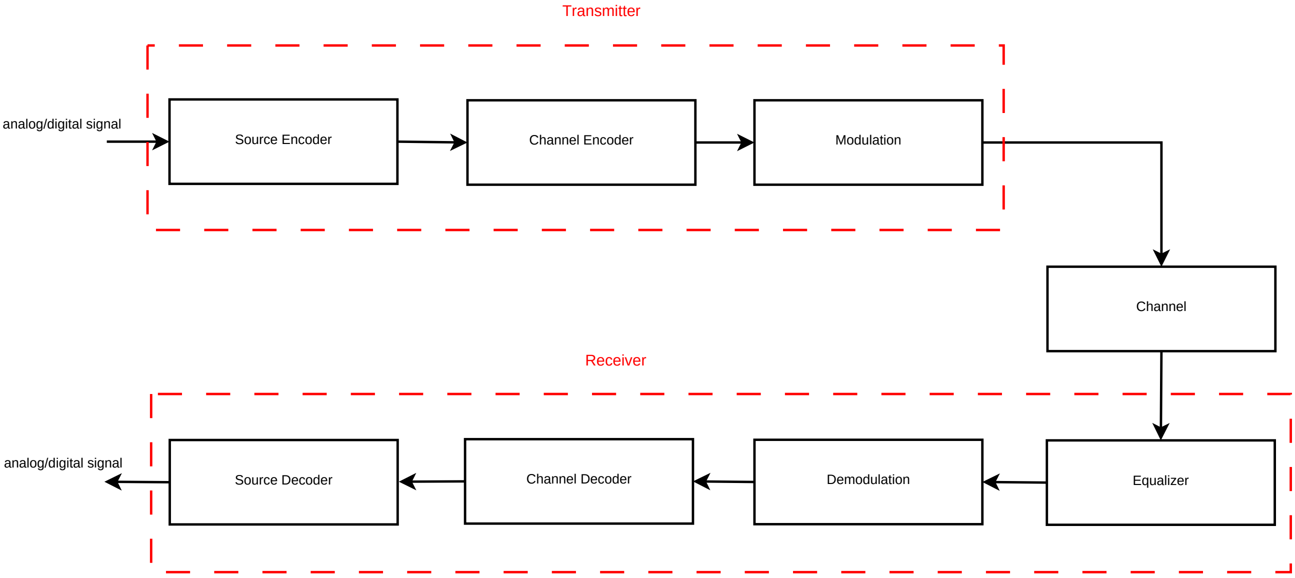


Figure 1: Communication System.

- The **Transmitter**, which encodes signals from a source (analog or digital) and modulates them for transport over the channel. The transmitter consists of three main blocks:
  - **Source Encoder**: Transforms the input signals in a digital bit stream with reduced redundancy.
  - **Channel Encoder**: Adds error detection and correction bits to the bit stream.
  - **Modulation**: Transforms the bit stream into a wave form for transmission over the channel.
- The **Channel**, which connects transports the signal from transmitter to receiver. During the transport the signal gets distorted due to different noise effects.
- The **Receiver**, which decodes and corrects the distorted signal from the channel. The main blocks of the receiver are:
  - **Equalizer**: Reduces the inter symbol interference (ISI) due to the channel distortion.
  - **Demodulation**: Transforms the Waveform back into an bit stream.
  - **Channel Decoder**: Corrects or detects errors in the bit stream.
  - **Source Decoder**: Inverse of the Source Encoder. Transforms the bit stream into output format.

These parts of the communication system will be discussed in the following sections.

2.1 Setting

In the laboratory we will use a recorded wav-file as an input for the speech transmission system. This wav-file can be imported into the Matlab workspace. This gives a sampled continuous valued signal.

Laboratory Assignment – P1.1 –

1. Import the wav-file SillyVoice.wav into the workspace with the Matlab function "audioread()". **Hint:** *For compatibility with the templates in the following exercises either name the signal value vector "w", the frequency "Fs" and the time vector "t" or remember to adapt the templates.*
2. Generate the corresponding time-vector t. The wav-file was sampled with 22.05 kHz.
3. Create a Simulink model with a "From Workspace"-block and an "To Audio Device"-block (from the DSP toolbox). Use the data from the wav-file as input signal, connect both blocks and run the model.

3 Source Coding

Source coding forms efficient descriptions of information sources. It reduces redundancy in the information source and therefore allows transmissions with lower bit-rates.
Source coding for digital transmission consists of a source encoder and a source decoder. The encoder first samples the continuous analog signal, a quantizer then generates a discrete value signal from the sampled signal and as last step a discrete encoder transforms it into a compressed binary signal. The source decoder performs the inverse operations it decodes the compressed bit stream into a sampled signal which is transformed into a continuous analog signal by an analog filter. The components of the source encoder are shown in Figure 2. If the input signal is already digital transformation and quantization are omitted.

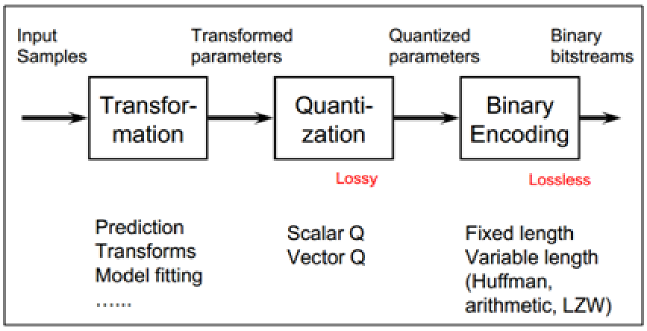


Figure 2: Basic Components in a Source Coding System.

Pulse Code Modulation (PCM) is a very common method to transform a continuous analog signal into a discrete digital signal. It is far spread as digital representation for audio signals. It will be discussed in the following subsection.

3.1 Pulse Code Modulation

A PCM stream is a digital representation of an analog signal which is sampled at uniform intervals. Each sample is rounded to the nearest value within a set of discrete values (quantization). Important properties characterizing the PCM are the sampling rate and the number of bits used for the representation. PCM has no mechanisms for exploiting signal redundancy. There are different forms of PCM:

1. Uniform PCM: Quantization intervals are uniformly distributed over a certain range.
2. Non-uniform PCM: Quantization intervals have different lengths and are fixed.
3. Differential PCM (DPCM): encodes the value as differences between the current value and a predicted value. In the easiest case the predicted value is the former value. The quantization intervals are fixed.
4. Delta modulation: a form of DPCM which uses one bit per sample.

5. Adaptive differential PCM (ADPCM): a form of DPCM where the Quantization intervals can be adapted.

The first two types will be explained in the following paragraphs. For more detail on the other types see [Skl01].

3.1.1 Uniform PCM

Uniform Pulse Code Modulation (PCM) is a memoryless process that quantizes amplitudes by rounding off each sample to one value of a set of discrete values. The difference between adjacent quantization levels, i.e., the step size, is constant in non-adaptive uniform PCM. The number of quantization levels, Q, in uniform PCM binary representations is $Q = 2^{R_b}$, where R_b denotes the number of bits. The performance of uniform PCM can be described in terms of the signal-to-noise ratio (SNR) of the quantizer. This is also called signal-to-quantization-noise-rate (SQNR). The SNR is defined as the quotient of signal variance (power) σ_s^2 and noise variance σ_n^2:

$$SNR = \frac{\sigma_s^2}{\sigma_n^2}, \tag{1}$$

$$SNR_{dB} = 10 \log_{10} \left(\frac{\sigma_s^2}{\sigma_n^2} \right). \tag{2}$$

Uniform quantizers are optimal in the mean square error (MSE) sense for signals with uniform probability density function (PDF).

3.1.2 Quantization error

The quantizer subdivides a finite interval $[u_{1,min}, \quad u_{1,max}]$ with dynamic range $\Delta u_1 = u_{1,max} - u_{1,min}$ into Q uniformly spaced steps of size

$$q = \frac{\Delta u_1}{Q}. \tag{3}$$

Each step is related by a one-to-one correspondence with a code word. In this way the continuous signal $u_1(t)$ is mapped to a discrete valued signal $u_2(t)$ with a finite set of code words, see also fig. 4. The quantization error can be modeled by:

$$n_q(t) = u_2(t) - u_1(t) \tag{4}$$

The maximum quantization error is:

$$|n_q| \leq \frac{q}{2} = \frac{\Delta u_1}{2Q} \tag{5}$$

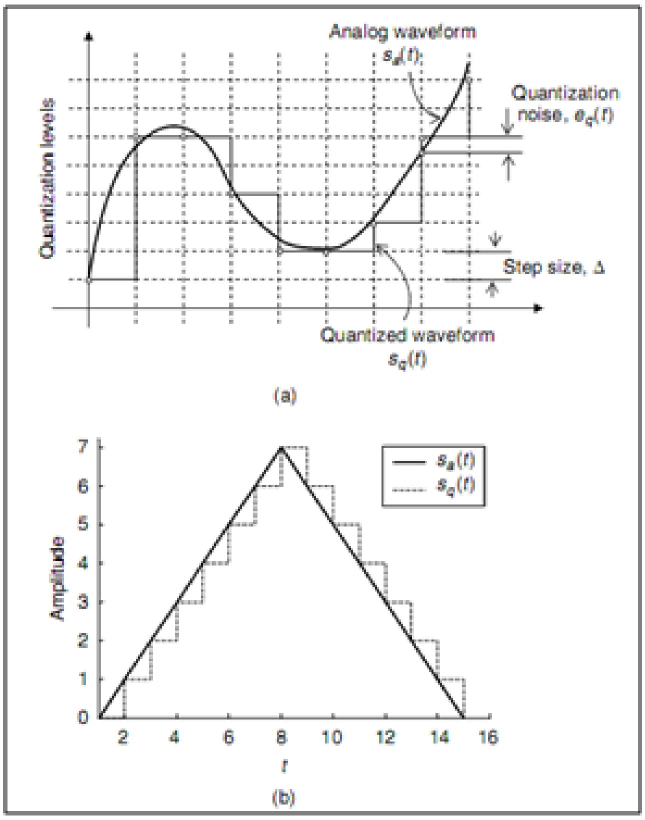


Figure 3: (a) Uniform PCM and (b) uniform quantization of a triangular waveform.

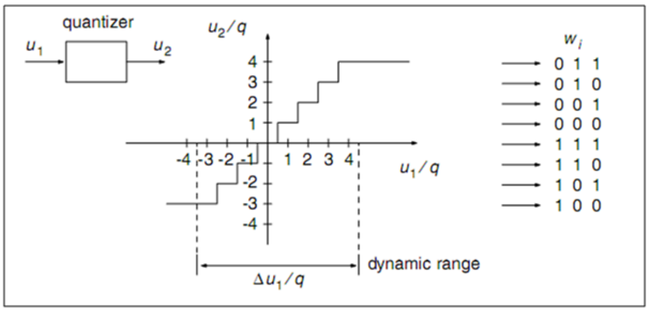


Figure 4: Transfer characteristic of a quantizer for $Q = 8$.

Pre-Laboratory Assignment – V1.1 –

Calculate the Signal-to-Quantization-Noise-Rate (SQNR) of a uniform quantizer depending on the number of bits R_b used to represent the quantized signal. The continuous input signal is assumed to have a uniform distribution and a range from -1 to 1.

Note: For a uniform signal with the range $[x_{min}, \quad x_{max}]$ the variance σ^2 is given by

$$\sigma^2 = \frac{(x_{max} - x_{min})^2}{12}.$$

3.1.3 Non-uniform PCM

Non-uniform PCM quantizers use a non-uniform step size that can be determined from the statistical structure of the signal. PDF-optimized PCM uses fine step sizes for frequently occurring amplitudes and coarse step sizes for less frequently occurring amplitudes. The step sizes can be optimally designed by exploiting the shape of the signal's PDF. A signal with a Gaussian PDF, for instance, can be quantized more efficiently in terms of the overall MSE by computing the quantization step sizes and the corresponding centroids such that the mean square quantization noise is minimized.

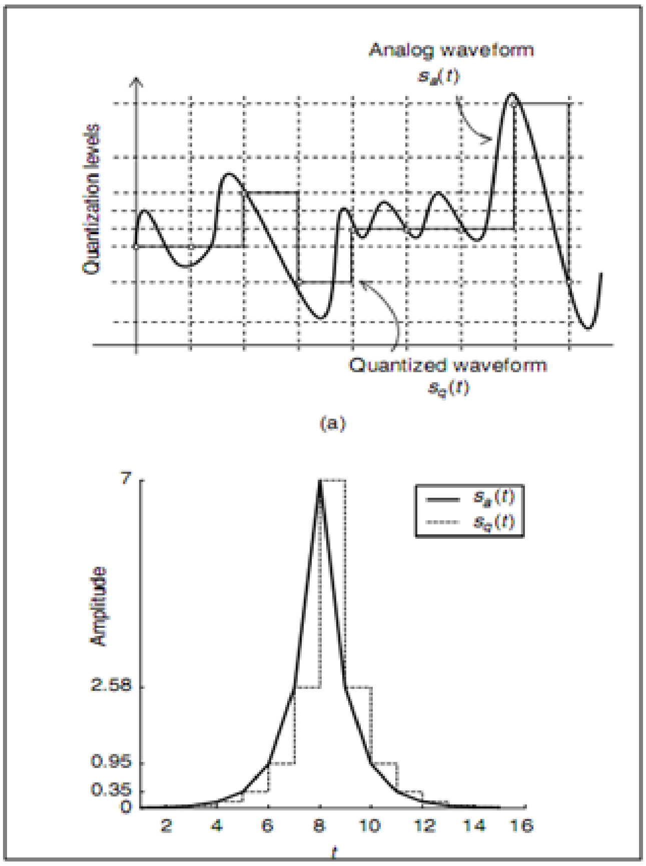


Figure 5: (a) Non-uniform PCM and (b) non-uniform quantization of a decaying-exponential waveform.

3.1.4 A-law and μ-law

Another class of non-uniform PCM relies on logarithmic quantizers that are quite common in telephony applications. For this quantizers a logarithmic compression is performed on the input signal before a uniform quantizer. Two telephony standards have been developed based on logarithmic companding, i.e., the μ-law and the A-law. They are standard forms of audio compression techniques. Compression is always in ratio 2:1. Both compress 16-bit audio in a manner acceptable to

human ears. The main difference between both is that μ-Law attempts to keep the top five bits of precision and uses a logarithmic function to determine the bottom 3 bits, while A-law compression keeps the top 4 bits and uses the logarithmic function to Figure out bottom four.
The μ-law is given by

$$g(x) = \text{sgn}(x)\frac{\ln\left(1+\mu|\frac{x}{x_{max}}|\right)}{\ln(1+\mu)}, \tag{6}$$

where x is an input signal with maximum amplitude x_{max}. For $\mu = 255$(used in the US and Japan), the above equation gives approximately linear mapping for small amplitudes and logarithmic mapping for larger amplitudes. The European A-law ($A = 87.6$) companding standard is slightly different and is based on the mapping

$$g(x) = \text{sgn}(x)\begin{cases} \frac{A\left|\frac{x}{x_{max}}\right|}{1+\ln A}, & \text{for } 0 < \left|\frac{x}{x_{max}}\right| < \frac{1}{A} \\ \frac{1+\ln\left(A\left|\frac{x}{x_{max}}\right|\right)}{1+\ln A}, & \text{for } \frac{1}{A} < \left|\frac{x}{x_{max}}\right| < 1 \end{cases}. \tag{7}$$

The idea with A-law companding is similar with μ-law in that again for signals with small amplitudes the mapping is almost linear and for large amplitudes the transformation is logarithmic. Both of these techniques can yield superior SNRs particularly for small amplitudes. In telephony, the companding schemes have been found to reduce bit rates, without degradation, by as much as 4 bits/sample relative to uniform PCM.

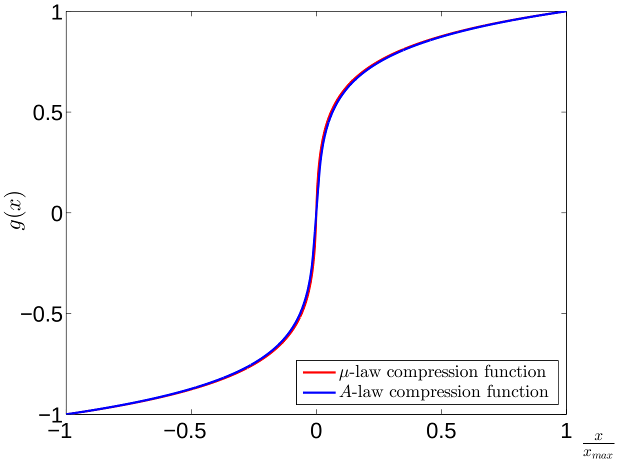


Figure 6: Compressor function for μ- and A-law.

The A-Law and μ-Law expander perform the reverse operation than that of the compressor. The uniform decoder de-quantizes the input.

3.2 Methods to reduce input redundancy

To reduce redundancy in the input stream there exist various compression methods. In the following, some compression methods which are used for digital data in different use cases are given:

- Huffman Code: lossless compression algorithm
- Lempel-Ziv Methods: lossless compression algorithm, variations of it are used in many data compression applications (for example in the zip format)
- JPEG: lossy compression of images
- MPEG: lossy compression for videos (MPEG 2) respectively audio (MP3)

For more details on these see [Skl01].

Laboratory Assignment – P1.2 –

1. Create a Simulink model with a source encoder using uniform encoding, and a source encoder using an A-law and the corresponding decoders. 6 bit quantization should be used in both methods.
2. Add scopes to the original and all decoded signals.
3. Compare the resulting signals and the original signal by using the "To Audio Device"-block.
4. Compare the quantization error of both methods by plotting the errors in a scope.
5. Compare the original signal with the signal processed by uniform quantizer and A-law quantizer in a scope.

4 Modulation

In this section the basic principles of modulation and demodulation are introduced.

4.1 Analog Modulation

For the transmission of analog signals the parameters of a sine wave are changed continuously depending on the input signal. There are three basic types of analog modulation:

- Amplitude Modulation (AM): Amplitude of the carrier signal is changed.
- Frequency Modulation (FM): Frequency of the carrier signal is changed.
- Phase Modulation (PM): Phase of the carrier signal is changed.

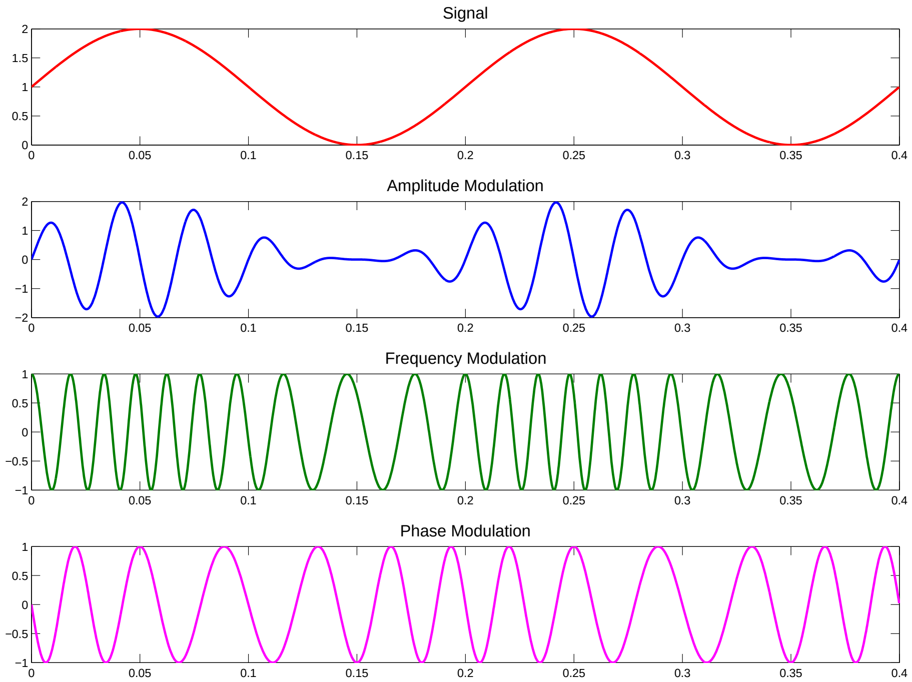


Figure 7: Example for different analog modulation methods.

Since we are working with digital signals in the laboratory we don't go into detail. For more details about analog modulation and demodulation see [MG02].

4.2 Digital Modulation

In order to transmit digital signals over an analog channel digital modulation is performed. There are three basic types of digital modulation techniques:

- Amplitude-Shift Keying (ASK)
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)

These techniques are very similar to those for AM, FM and PM respectively. Parameters of a sine wave are changed according to a digital input signal. Examples, where every bit is modulated separately, are shown in Figure 8.

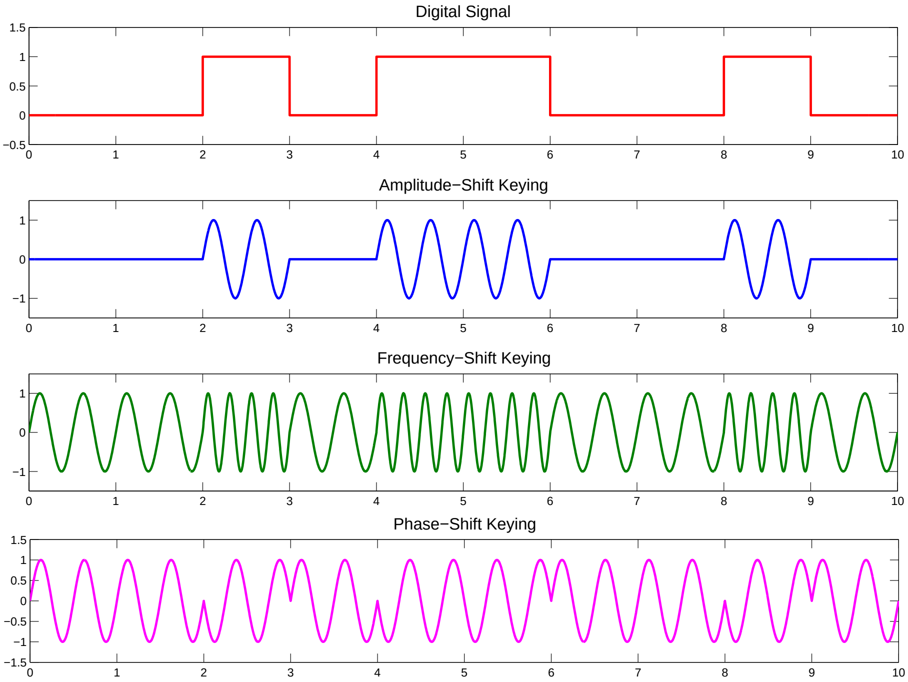


Figure 8: Example for different digital modulation methods.

For these one bit modulation methods the following equations hold:

- Binary Amplitude-Shift Keying (BASK):

$$s(t) = \begin{cases} 0, & \text{bit:0} \\ A\cos(\omega t), & \text{bit:1} \end{cases} . \tag{8}$$

- Binary Frequency-Shift Keying (BFSK):

$$s(t) = \begin{cases} A\cos(\omega_1 t), & \text{bit:0} \\ A\cos(\omega_2 t), & \text{bit:1} \end{cases} . \tag{9}$$

- Binary Phase-Shift Keying (BPSK):

$$s(t) = \begin{cases} A\cos(\omega t + 0), & \text{bit:0} \\ A\cos(\omega t + \pi), & \text{bit:1} \end{cases}. \tag{10}$$

4.2.1 I-Q channels and Constellation Diagram

The carrier bandpass signal can be written as follows

$$s(t) = A(t)\cos(\omega(t)t + \phi(t)). \tag{11}$$

It's phasor is

$$\tilde{A}(t) = A(t)e^{j\phi(t)}, \tag{12}$$

which leads to

$$s(t) = Re\left\{\tilde{A}(t)e^{j\omega(t)t}\right\}. \tag{13}$$

This equation can be split into the so called in-phase and quadrature component:

$$s(t) = \underbrace{Re\{\tilde{A}(t)\}\cos(\omega(t)t)}_{\text{in-phase component}} - \underbrace{Im\{\tilde{A}(t)\}\sin(\omega(t)t)}_{\text{quadrature component}} = I(t)\cos(\omega(t)t) - Q(t)\sin(\omega(t)t). \tag{14}$$

With this the equivalent complex low-pass signal is:

$$\tilde{s}_l(t) = I(t) + jQ(t) = \sqrt{I^2(t) + Q^2(t)}e^{j\arctan\frac{I(t)}{Q(t)}}. \tag{15}$$

$I(t)$ and $Q(t)$ are used to draw the so called constellation diagram, where I(t) is the value on the x-axis and Q(t) the value on the y-axis. The constellation diagram shows the modulation alphabet. It can be used to study the distortion of the symbols. The constellation has to show the modulation symbols as distinct sets to allow demodulation without errors. Figure 9 shows a constellation plot of a duo-binary modulation. It can be seen that the sets are far spread and some of the points are between two sets which is an indicator of a bad SNR and a higher error possibility for the demodulation.

Pre-Laboratory Assignment – V1.2 –
Draw the constellations for BASK and BPSK. Note that both have no quadrature component $Q(t)$.

4.2.2 M-ary Modulation

With different levels of modulation these methods can be used to transmit signals with M values: If $M = 4$, 2 bits can be transmitted in one sample.
Common modulation methods are:

- M-PSK
- M-ASK
- M-Quadrature Amplitude Modulation(M-QAM)

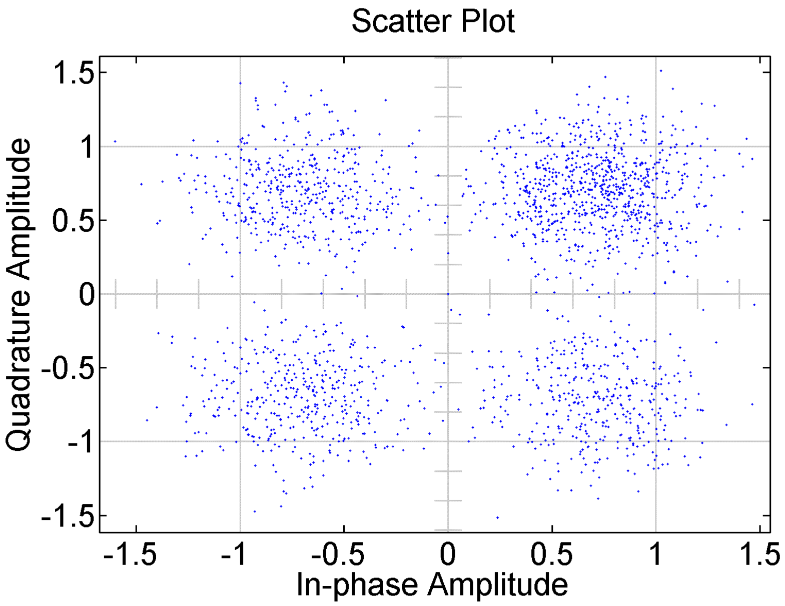


Figure 9: Example of a Constellation Plot.

M-PSK

M-PSK uses M different phases for the M values. This results in the following equation for the modulated signal of the i-th value:

$$s_i(t) = A\cos\left(\omega t + \frac{2\pi i}{M} + \phi\right),\ i = 0, 1, ...M - 1. \tag{16}$$

With the trigonometric identity

$$\cos(A + B) = \cos(A)\cos(B) - \sin(A)\sin(B), \tag{17}$$

we get the following form

$$s_i(t) = I_i(t)\cos(\omega t) - Q_i(t)\sin(\omega t) \tag{18}$$

and can draw the constellation diagram. An constellation diagram for an 16-PSK is shown in Figure 10.

Pre-Laboratory Assignment – V1.3 –
Draw a constellation diagram for a 4-PSK for $\phi = \frac{\pi}{4}$. This case is also called Quadrature Phase-Shift Keying(QPSK).

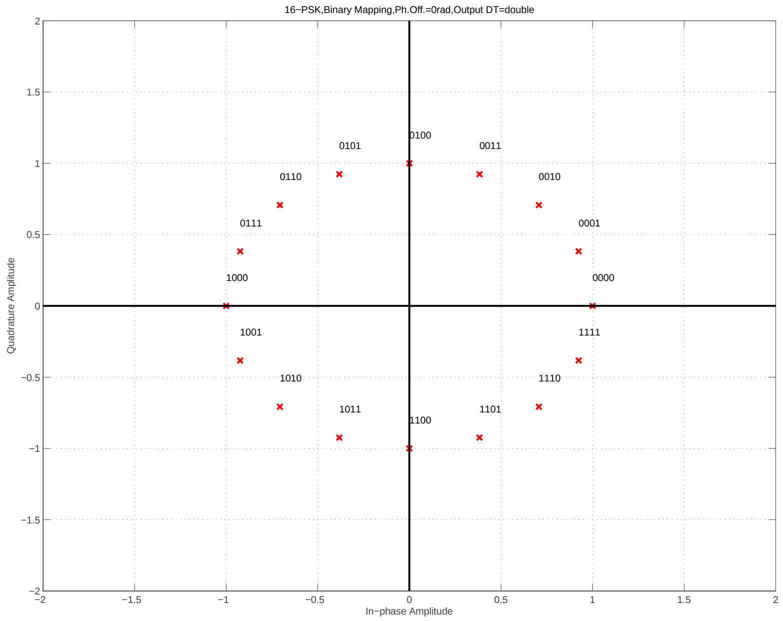


Figure 10: Constellation diagram of a 16-PSK.

M-ASK

The M-ASK is used as a basis for the M-QAM. It is the enhancement of the ASK to M different amplitude levels. The corresponding modulated signal can be written as:

$$s(t) = A(t)cos(\omega t + \phi), \text{ where } A(t) \in \{(2i - 1 - M)A_c,\ i = 1, ...M\}. \qquad (19)$$

The constellation points are all in-phase if $\phi = 0$ holds.

M-QAM

For M-QAM the form of equation (14) is exploited. Since the in-phase and the quadrature component are orthogonal they can be seen as uncorrelated. For an input with M signals a $\frac{\mathrm{M}}{2}$-ASK is performed for the in-phase and the quadrature component. Both components are added. The result can be seen as amplitude and phase shift keying of one carrier signal. In Figure 12 the schema of an M-QAM modulation is shown. Here also the division in in-phase and quadrature component can be seen. An example constellation diagram of a 16-QAM is shown in Figure 11. If more than 8 bits have to be transmitted per symbol usually M-QAM is preferred over M-PSK since the distance between the symbols in the I-Q-plane is greater in QAM and therefore it is less sensitive to noise. For more details see [Kam11] and [PS05].

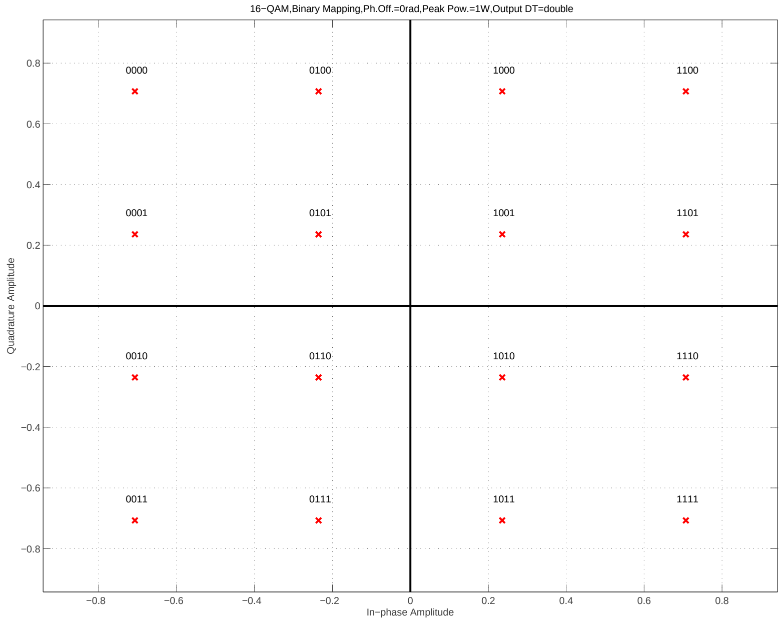


Figure 11: Constellation diagram of a 16-QAM.

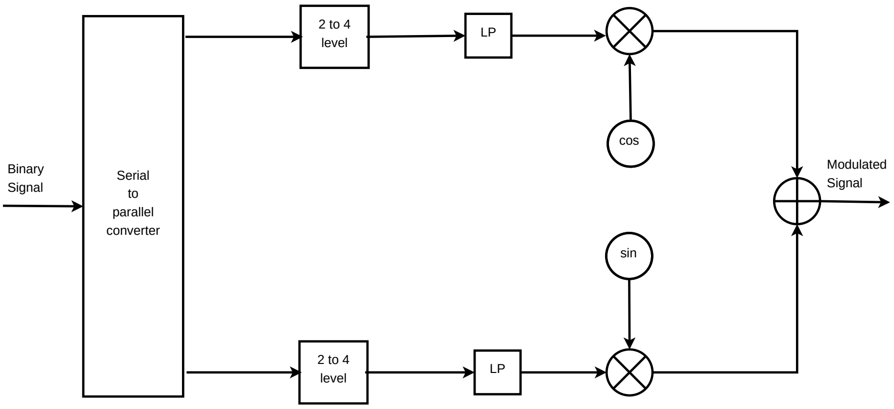


Figure 12: Schema of a M-QAM modulation.

4.2.3 Other Modulation Methods

There are many other digital modulation methods like:

- Minimum Shift Keying (MSK)
- Gaussian Minimum Shift Keying (GMSK)
- Trellis Coded Modulation (TCM)
- Offset QPSK (OQSPK)
- ...

These are not be discussed here. For details see [MG02], [Skl01] and [Kam11].

4.2.4 Demodulation

There are two different types of demodulations: demodulations which need carrier reproduction, also called coherent demodulations, and demodulations which do not need them (non-coherent demodualtions). A coherent schema for M-QAM demodulation is shown in Figure 13. It can be seen that there is a "carrier recovery"- and a "symbol timing recovery"-block. These blocks are also called synchronization and are used to retrieve information about the carrier signal from the modulated signal. Details about the synchronization and other demodulation methods can be found in [Skl01] and [PS05]

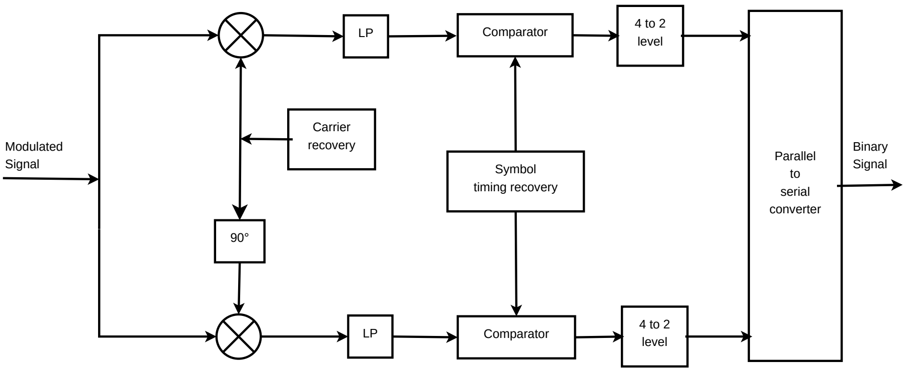


Figure 13: Schema of a M-QAM demodulation.

Laboratory Assignment – P1.3 –

1. Use the Matlab command *bertool* to open the Bit Error Rate Analysis Tool.
2. Compare the theoretical error rate for M-QAM and M-PSK modulations with different numbers of M.
3. A-law source encoding quantization with 8bit is used from now on. Use "Integer to Bit Converters" to pass the digital signal to the modulators.
4. Create a Simulink model with M-QAM and M-PSK modulations and demodulations ($M = \{4, 16, 256\}$). The normalization method should be set to average power for better comparability. Note that the Output data type of the demodulation blocks has to be set to "uint8" (→"Bit") manually.
5. Use an Additional White Gaussian Noise (AWGN) channel model with a "Signal to noise ratio (E_b/N_0)" of 10 dB. Adapt the symbol sample time and the symbol bit properties to the corresponding modulation.
6. Add "Error Rate Calculation"-blocks (in Comm Sinks) to analyze the bit error rates (BER). Use "Display"-blocks to directly evaluate the error.
7. Add To-Workspace-blocks for the modulated signals with noise of 4-QAM, 16-QAM and 16-PSK.

8. Add a "Discrete-Time Scatter Plot Scope" (Simulink: "constellation diagram") for 16-QAM.
9. Simulate the model and compare the BER.
10. Plot the Constellation diagrams of the signals saved in the workspace with the command "scatterplot()" and compare them.
11. Repeat the simulations with an E_b/N_0 of 16 dB and analyze them.
12. Add a 256-QAM modulation and demodulation with AWGN and change the Constellation ordering from "gray" to "binary". Compare the BER and watch the constellation ordering ("view constellation"). Explain the difference in BER.

4.3 Eye Diagram

An eye diagram is a tool for studying the effects of inter symbol interference (ISI) and other channel impairments in digital transmission. The procedure to construct an eye diagram is to plot the received signal against time on a fixed interval axis. This is repeated at the end of the fixed time interval, the result is many overlapping plots. The information included in an eye diagram can be seen in Figure 14.

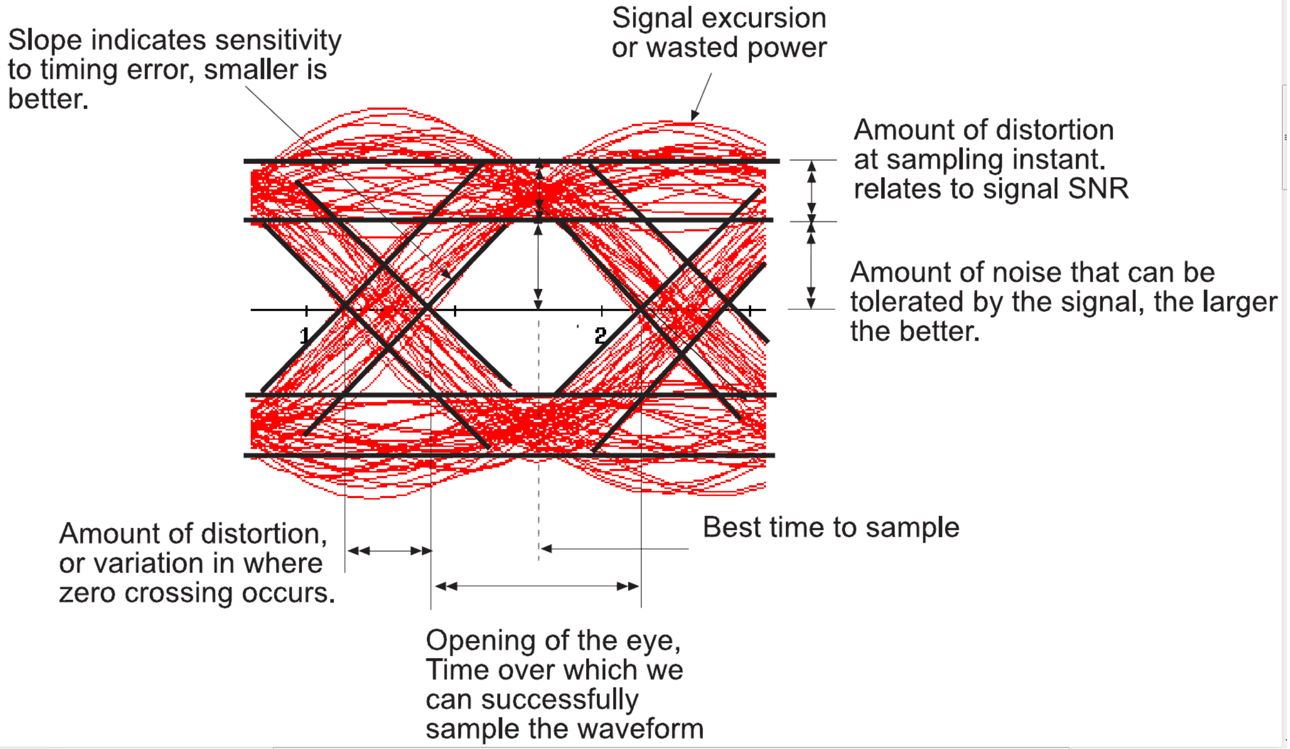

Slope indicates sensitivity to timing error, smaller is better.
Signal excursion or wasted power
Amount of distortion at sampling instant. relates to signal SNR
Amount of noise that can be tolerated by the signal, the larger the better.
1
2
Amount of distortion, or variation in where zero crossing occurs.
Best time to sample
Opening of the eye, Time over which we can successfully sample the waveform

Figure 14: Explanation of an Eye Diagram.

4.4 Pulse Shaping

In a band-limited system signals with unlimited bandwidth may lead to inter symbol interference (ISI). There are two criteria for non-interference systems where pulse shaping is employed.

1. The pulse shape exhibits a zero crossing at the sampling point of all pulse intervals except its own. This is also called the first Nyquist condition.
2. The shape of the pulses be such that the energy of the frequency response decays rapidly outside of the pulse interval to avoid ISI by reflections due to the limited bandwidth.

A rectangular pulse satisfies the first criterion (where it contains zero crossing – see Figure 15) but not the second criterion (the energy of the rectangular pulse does not decay rapidly outside the pulse interval and in fact it extends to infinite bandwidth). A raised cosine pulse is designed to satisfy these two criterions and thus provides an ISI free system.

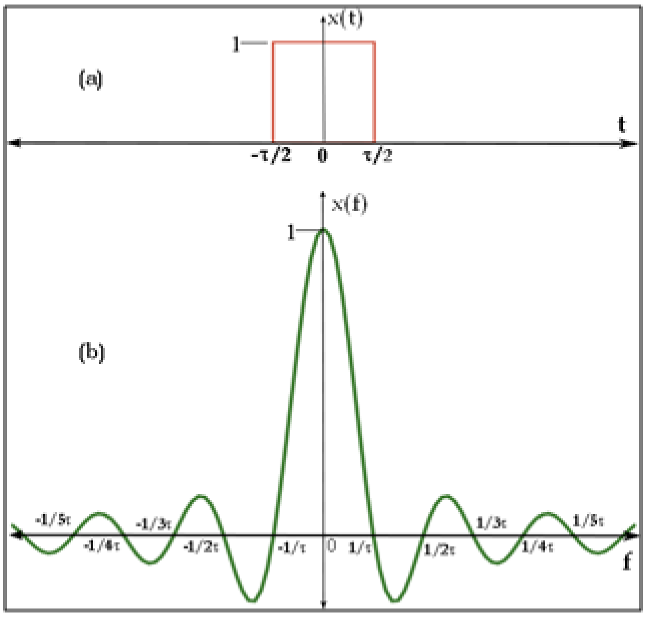


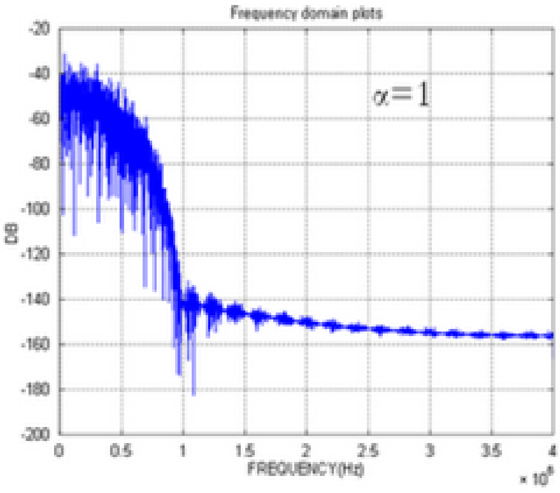
Figure 15: Rectangular pulse and its frequency domain signal.

4.4.1 Raised Cosine Filter

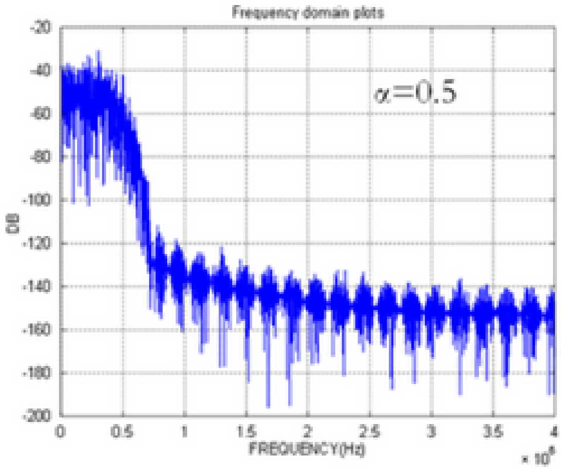
Raised Cosine Filters are used mainly for pulse shaping and matched filtering.
A Raised Cosine pulse takes on the shape of a sinc pulse in time domain and its energy decreases fast with with increasing frequency. It is given by the following function

$$H(f) = \begin{cases} 1, & |f| \leq \frac{1-\alpha}{2T} \\ \cos^2\left(\frac{\pi T}{2\alpha}\left(|f| - \frac{1-\alpha}{2T}\right)\right), & \frac{1-\alpha}{2T} < |f| < \frac{1+\alpha}{2T} \\ 0, & |f| \geq \frac{1+\alpha}{2T} \end{cases} . \tag{20}$$

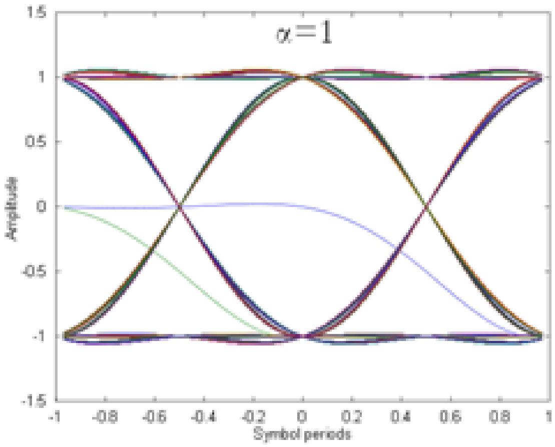
Here T is the symbol sample time, α is a parameter that governs the bandwidth occupied by the pulse and the rate at which the tails of the pulse decay. A value of $\alpha = 0$ offers the narrowest bandwidth, but the slowest rate of decay in the time domain. When $\alpha = 1$, the bandwidth is $\frac{1}{T}$, but the time domain tails decay rapidly.
The frequency response and the eye-diagram for the Raised Cosine Filtered output for $\alpha = 1$ and $\alpha = 0.5$ are plotted below for comparison. The frequency response shows minimized side lobes for $\alpha = 1$, whereas for the case $\alpha = 0.5$ the side lobes' energy is higher when compared to $\alpha = 1$. Also, for $\alpha = 1$, the eye opening is clearer when compared to $\alpha = 0.5$. This implies that by increasing the value of α from 0 to 1, the ISI decreases where as the excess bandwidth of the signal increases. An optimum balance between the value of α and the bandwidth of the band-limited channel is necessary for ISI free reliable communication between the transmitter and receiver. In most application an α in the range of 0.2 to 0.5 is used.
Another filter used for pulse shaping is the Gaussian Filter (used for GMSK).



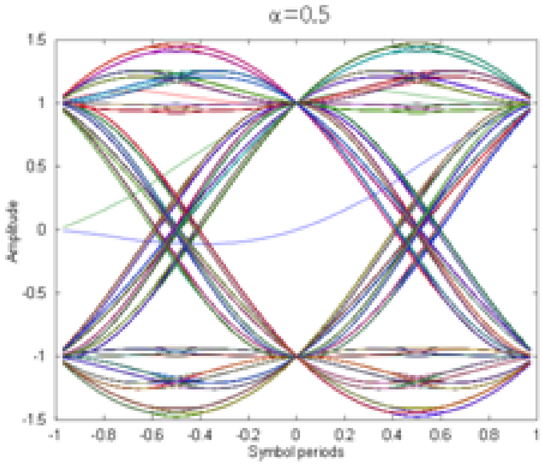
(a) Raised Cosine Filter Frequency Response for $\alpha = 1$.



(b) Raised Cosine Filter Frequency Response for $\alpha = 0.5$.



(c) Raised Cosine Filter Eye Diagram for $\alpha = 1$.



(d) Raised Cosine Filter Eye Diagram for $\alpha = 0.5$.

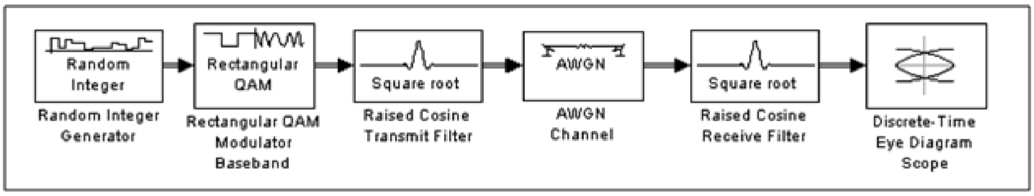

Random Integer
Random Integer Generator
Rectangular QAM
Rectangular QAM Modulator Baseband
Square root
Raised Cosine Transmit Filter
AWGN
AWGN Channel
Square root
Raised Cosine Receive Filter
Discrete-Time Eye Diagram Scope

Figure 17: Example for Raised Cosine Filter in Simulink.

4.4.2 Group Delay of Filters

The raised cosine and Gaussian filter blocks implement realizable filters by delaying the peak response. Group delay is defined as the length of time between filters initial response and its peak response.
The power gain of square root raised cosine transmit filter is 1/N, where N is the up-sampling factor of the filter.
Roll off factor determines the excess bandwidth of the filter.
For e.g a roll off factor of 0.5 means bandwidth of filter is 1.5 times the input sampling frequency.
The group delay influences the size of the output, as well as the order of the filter.

Laboratory Assignment – P1.4 –

1. Create a Simulink model with A-law source encoding 4-QAM and 16-QAM modulation.
2. Add "Discrete-Time Eye Diagram Scopes" (Comm Sinks).
3. Start the simulation and compare the eye diagrams.
4. Open the Simulink model "CFT_pulse_shaping.mdl". Here the A-law source encoding a 16-QAM modulation and a low pass channel model is implemented.
5. Simulate the model and study the constellation and eye diagram.
6. Add a second 16-QAM modulation with channel followed by a "Raised Cosine Transmit Filter" and a "Raised Cosine Receive Filter". Set the Decimation factor in "Raised Cosine Receive Filter" to 1.
7. Add an eye diagram and a constellation diagram after the Receive Filter. Change the "Samples per Symbol" property of the eye diagram to 16 and add "Downsample"-block (DSP, Signal Operations) with downsample factor 8 in front of the constellation diagram.
8. Simulate the model for different rolloff factors of the raised cosine filters and compare the scopes.

5 Channel Coding

Channel coding is used to detect and correct transmission errors.

5.1 Error Detection and Correction

Error-control coding techniques detect, and possibly correct, errors that occur when messages are transmitted in a digital communication system. To accomplish this, the encoder transmits not only the information symbols but also extra redundant symbols. The decoder interprets what it receives, using the redundant symbols to detect and possibly correct errors which occurred during transmission.
In coding theory, a linear code is an error-correcting code for which any linear combination of code words is another code word of the code. Linear codes are traditionally partitioned into block codes and convolution codes, although Turbo codes can be seen as a hybrid of these two types. Linear codes allow for more efficient encoding and decoding algorithms than other codes.

5.2 Properties of Correction Codes

The minimum distance d_{min} of a code C is $\min d(c_1, c_2)\ \forall c_1, c_2 \in C \wedge c_1 \neq c_2$, that is the smallest of the Hamming distances between distinct pairs of codewords. The Hamming distance d is the number of bits which are different in two codewords. The Hamming distance d is a metric, i.e. it satisfies the following properties:
For all vectors c_1,c_2 and c_3 holds

$$d(c_1, c_2) \geq 0 \tag{21}$$
$$d(c_1, c_2) = 0, \text{ if and only if } c_1 = c_2 \tag{22}$$
$$d(c_1, c_2) = d(c_2, c_1) \tag{23}$$
$$d(c_1, c_2) \leq d(c_1, c_3) + d(c_3, c_2). \tag{24}$$

Similarly, the minimum weight w_{min} of a code C is $\min w(c)$ with $c \in C, c \neq 0$, the smallest weight w among non-zero codewords, where the weight w of a nonzero code word is the number of its "1"s. The minimum distance of a code determines its error detection/correction capacity, as the following theorem shows:

Let C be a code with minimum distance d_{min}. Then C can detect any $e_d = d_{min} - 1$ errors, and can correct any $e_c = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ (the floor of $\frac{d_{min}-1}{2}$) errors. Additionally in the case of correction the conditions

$$e_c + e_d \leq d_{min} - 1 \tag{25}$$
$$e_d \geq e_c \tag{26}$$

have to hold. This means that if the number of errors is $d_{min} - 1$ and at least one error correction is attempted, the correction changes a wrong bit and falsely assumes to a successful correction with no errors.

5.3 Block Codes

Block coding is a special case of error-control coding. Block-coding techniques map a fixed number of message bits to a fixed number of code bits. A block coder treats each block of data independently and is a memoryless device.

The codes in a linear block code are blocks of bits which are encoded using more bits than the original value to be sent. Information is divided into blocks of length k. r parity bits or check bits are added to each block. The total length n becomes $k + r$. A linear code of length n transmits blocks containing n bits. The code rate R is defined as the ratio of n to k. The Decoder looks for the code word closest to received vector (code vector + error vector), n is the length of the code word, in bits. There are many types of linear block codes:

- Cyclic codes (e. g., Hamming codes)
- Repetition codes
- Parity codes
- Polynomial codes (e.g., BCH codes)
- Reed-Solomon codes
- Golay codes

5.3.1 Hamming codes

Hamming codes are special codes with $k = 2^r - r - 1$ and $n = 2^r - 1$ for $r \geq 2$. These codes are high rate codes and have a constant minimum distance $d_{min} = 3$. This means they can detect and correct one error or if no error correction is made detect two errors.

For the generation and the parity check of Hamming codes the parity check matrix H and the generator matrix G are used. The matrix H can be constructed by writing the all bit words with the length r except the zero word in the columns of H. By ordering the columns a structure

$$H = [P^T | I_r] \in \{0, 1\}^{r \times (k+r)} \tag{27}$$

can be obtained, where I_r is the $r \times r$ Identity matrix and P^T is a $r \times k$ matrix where the entries are the columns with more than one "1". The generator matrix can be derived from this with the equation

$$G = [I_k | P] \in \{0, 1\}^{k \times (k+r)}. \tag{28}$$

For a bit word $x \in \{0, 1\}^k$ the code word $c \in \{0, 1\}^n$ is generated by the matrix multiplication $c = xG$.

For any admissible code word $p = cH^T = 0$ holds. If there are detectable errors $p = cH^T \neq 0$ holds.

For one error the parity vector p gives information about the position of the error. p is the same as one column of H. The number of this column is the position of the bit error in the faulty code word. The bit error is corrected by simply flipping this bit. The decoding matrix D has the form $D = [I_k|0]^T \in \{0,1\}^{k \times n}$ and gives the decoded data with $x' = c'D$, where c' is the corrected code word. Note that all additions during the calculations are modulo-2 additions.

Pre-Laboratory Assignment – V1.4 –

1. Calculate the matrices G, H and D for $r = 3$.
2. Calculate the code word c for the input word $x = 1100$.
3. Flip the fourth bit of c, calculate the parity vector p, correct the code word and decode it. Is the error correctable?
4. Flip two bits of c, calculate the parity vector p, correct the code word and decode it. Is the error correctable?

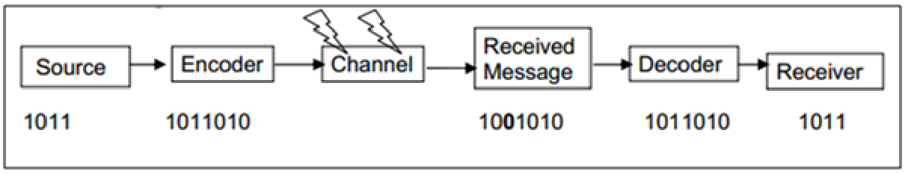


Figure 18: Example for Hamming Source Coding.

5.4 Convolutional Codes

The "codes" in convolutional codes involve encoding of information bits rather than information blocks. The value of a certain information symbol also affects the encoding of the next m information symbols. It is easily implemented using shift registers. There are k input bits and n output bits. Decoding is mostly performed by the Viterbi Algorithm. The essential difference between block codes and convolution codes is the fact that rate $1/n$ convolution codes have a memory of previous $K-1$ inputs bits, where K is the constraint length. With such a memory the encoding of each input bit not only depends on the value of that bit but also on the values of $m = K-1$ input bits that preceded it.

5.4.1 Encoder Structure

A convolutional code introduces redundant bits into the data stream through the use of linear shift registers as shown in Figure 19.

The information bits are inputs into shift registers and the output encoded bits are obtained by modulo-2 addition of the input information bits and the contents of the shift registers. The code

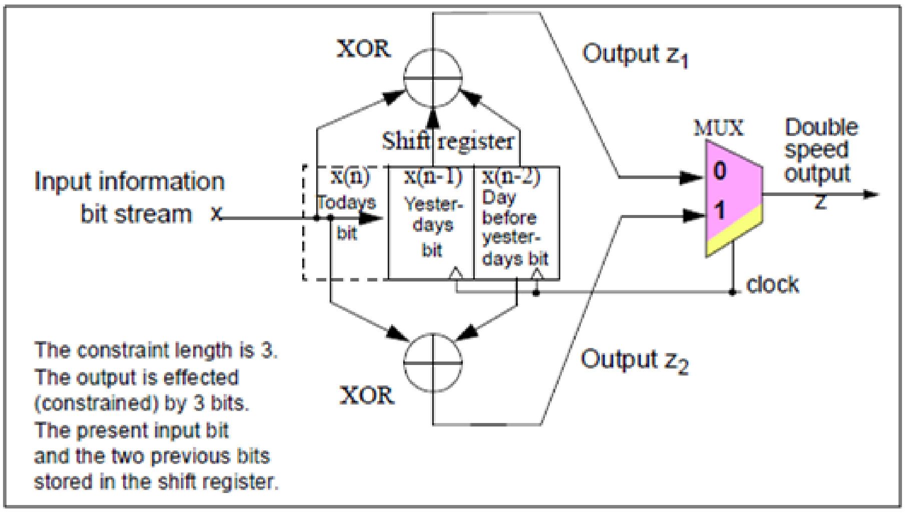

XOR
Output z1
Shift register
MUX
Double speed output
Input information bit stream x
x(n) Todays bit
x(n-1) Yester-days bit
x(n-2) Day before yester-days bit
0
1
z
clock
The constraint length is 3.
The output is effected (constrained) by 3 bits.
The present input bit and the two previous bits stored in the shift register.
XOR
Output z2

Figure 19: Convolution code generation using shift register (rate = 1/2).

rate R for a convolutional code is defined as $R = k/n$. where k is the number of parallel input information bits and n is the number of parallel output encoded bits at one time interval. The constraint length K for a convolutional code is defined as $K = m + 1$ where m is the maximum number of stages (memory size) in any shift register. The shift registers store the state information of the convolutional encoder and the constraint length is related to the number of bits upon which the output depends.

5.4.2 Generator Representation

The generator representation shows the hardware connection of the shift register taps to the modulo-2 adders. A generator vector represents the position of the taps for an output. A "1" represents a connection and a "0" represents no connection. For example, the two generator vectors for the encoder in Figure 19 are $g_1 = [111]$ and $g_2 = [101]$ where the subscripts 1 and 2 denote the corresponding output terminals. The vectors start with the actual input as first element and end with the longest saved input.

5.4.3 State Diagram Representation

The state diagram shows the state information of a convolutional encoder. The state information is the bit word stored in the shift registers. Figure 20 shows the state diagram of the encoder in Figure 19. In the state diagram, the state information is shown in the circles. Each new input information bit causes a transition from one state to another. The path information between the states, denoted as x/c, represents input information bit x and output encoded bits c. It is customary to begin convolutional encoding from the "all zero" state. For example, the input information sequence $x = 1011$ leads to the state transition sequence $s = \{10, 01, 10, 11\}$ and produces the output encoded sequence $c = \{11, 10, 00, 01\}$.

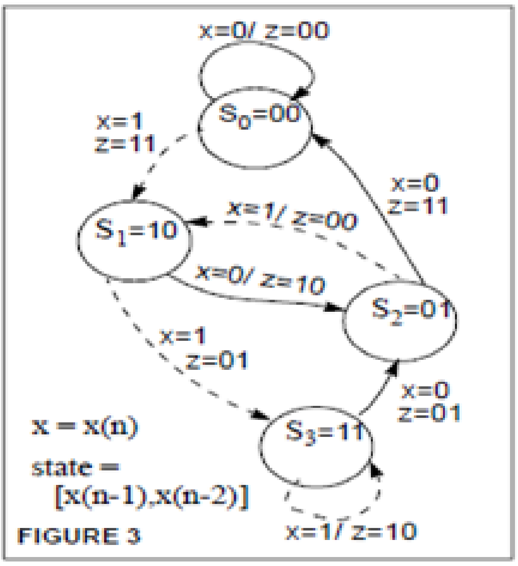


Figure 20: State Diagram Representation of the encoder.

5.4.4 Trellis Diagram Representation

The trellis diagram is basically a redrawing of the state diagram. It shows all possible state transitions at each time step. Frequently, a legend accompanies the trellis diagram to show the state transitions and the corresponding input and output bit mappings (x/c). This compact representation is very helpful for decoding convolutional codes as discussed later. Figure 21 shows the trellis diagram for the encoder in Figure 19.

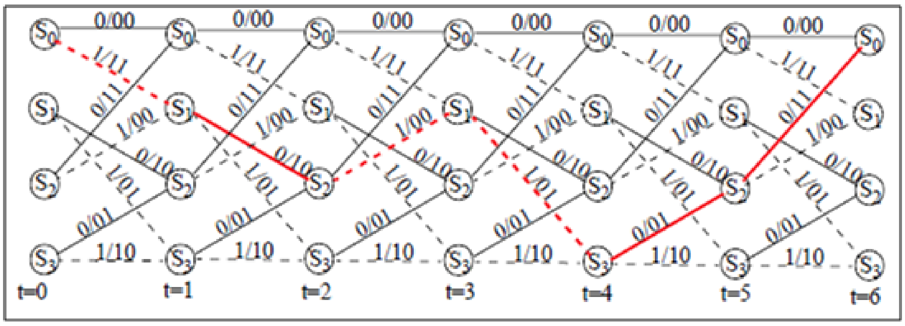


Figure 21: Complete trellis diagram.

Pre-Laboratory Assignment – V1.5 –

1. Draw the encoder structure for the generator vectors $g_1 = [110]$ and $g_2 = [101]$.
2. Draw the corresponding state diagram.

3. Draw the corresponding trellis diagram up to $t = 3$.
4. Calculate the code word for the input “110” starting at the “all zero” state and mark the path in the trellis diagram.

5.4.5 Viterbi Algorithm

For optimal decoding of a modulation scheme with memory, we need to search for input sequence in the set of possible combinations. This becomes prohibitively complex if the code is large. However, Andrew J. Viterbi described a scheme for reducing the complexity to more manageable levels. Some of the key assumptions are as follows:

1. Any state can be reached from only 2 possible previous state.
2. Though we reach each state from 2 possible states, only one of the transitions is valid. We can find the transition which is more likely (based on the received coded bits) and ignore the other transition.
3. The errors in the received coded sequence are randomly distributed and the probability of error is small.

Based on the above assumptions, the decoding scheme proceed as follows:

- Assume that there are coded bits.
- Calculate the Branch Metrics, Path Metric and the Survivor Path.
- Perform a trace back on the Survivor Path and calculate output bits.

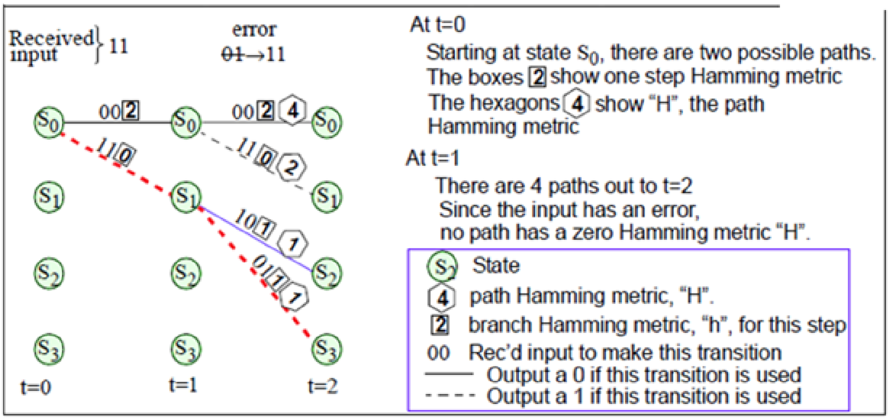
The different parts are explained in the following subsections.

5.4.6 Branch Metric and Path Metric Computation

The branch metric is a measure for the distance between the transmitted sequence and the output for a state transition at that time (see Figure 22). Usually the hamming distance is used as branch metric. The path metric is the sum of branch metrics on the path between two states for a fixed number of time steps. Since there could be different paths only the one with the smallest value of the sum is saved. If two paths to one state have the same path metric, only one is saved. This operation is also referred to as Add Compare and Select (ACS).
The path with the lowest path metric at the given length is the survivor path.

5.4.7 Traceback Unit

Once the survivor path is computed, the decoding algorithm can start trying to estimate the input sequence. So, starting from the computed survivor path, find the previous state corresponding to the current state. From the knowledge of current state and previous state, the input sequence can be determined due to the encoder trellis diagram. Continue tracking back through the survivor path



At t=0
Starting at state S_0, there are two possible paths.
The boxes 2 show one step Hamming metric
The hexagons 4 show “H”, the path Hamming metric

At t=1
There are 4 paths out to t=2
Since the input has an error,
no path has a zero Hamming metric “H”.

S_2 State
4 path Hamming metric, “H”.
2 branch Hamming metric, “h”, for this step
00 Rec’d input to make this transition
—— Output a 0 if this transition is used
- - - - Output a 1 if this transition is used

Figure 22: Branch and Path Metric computation for Viterbi decoder.

and estimate the input sequence till the first state. Since the whole code word of a convolutional encoder is very long not the whole code word is considered due to the resulting delay and memory issues. Usually codewords with a length of $5K$ are considered which is suboptimal but still gives good results.
For more details see [PS05].

Laboratory Assignment – P1.5 –

1. Open the Simulink model “CFT_Channel.mdl”. In this model A-law source coding, 16-QAM modulation, demodulation, raised cosine filter and a low pass AGWN channel model is already implemented.
2. Simulate the model and observe the scopes, bit error rate and symbol error rate.
3. Add a Hamming encoder and decoder to the model. Be careful with delays. For calculation of delays “Find Delay” blocks (Utility Blocks) are used.
4. Save the model as a separate Simulink file.
5. Simulate the model and compare the BER before and after channel decoding.
6. Compare the voice signals with and without channel coding by using the “sound()” function in Matlab.
7. Replace the Hamming encoder and decoder by a convolutional encoder and a Viterbi decoder and run the simulation again.
Hint: *Set the decision type of the Viterbi decoder to “hard decision” and adjust the traceback depth.*

6 Interleaving

Interleavers randomize the data sequence. This is necessary to distribute burst errors among many codewords. Burst errors are characteristics of some channels (wireless) and also occur in concatenated codes.
There are different kinds of data interleavers:

- Block interleaver: Data is in raw format and read in column.
- Pseudo interleaver: Data is stored at positions that are determined randomly.
- Convolutional interleaver: Shift of data, usually applied in a fixed and cumulative way.
- Linear interleaver: Block interleavers where the data positions are altered by following a linear law.

The amount of error protection based on the length of noise bursts determines the span length or depth of interleaving required.

6.1 Block Interleaver

The number of rows should ideally be larger than the longest burst of errors expected. The number of columns is normally selected to be equal to or larger than the block or decoding length of the code that is used.

6.1.1 Matrix Interleaver

The Matrix Interleaver block accomplishes block interleaving by filling a matrix with the input symbols row by row and then sending the matrix contents to the output port column by column. For example, if the interleaver uses a 2-by-3 matrix to do its internal computations, then for an input of [123456], the block produces an output of [142536].

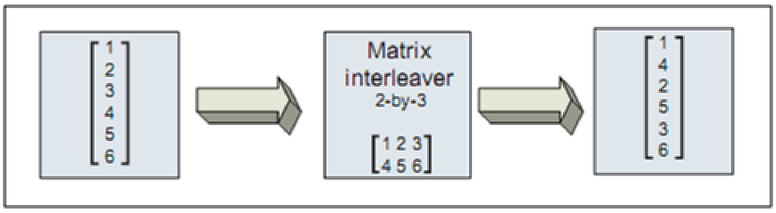


Figure 23: Matrix Interleaver.

6.2 Convolutional Interleaver

Convolutional interleavers reduces memory requirements over block interleavers by about one half. The delay is a function of interleaver depth and the data rate.
It is formed with a set of N registers that are multiplexed in such a way that each register has L symbols more than the previous register.
The multiplexers commute through the different register output and take out the “oldest” symbol stored in each register while another symbol is input to that register at the same time.

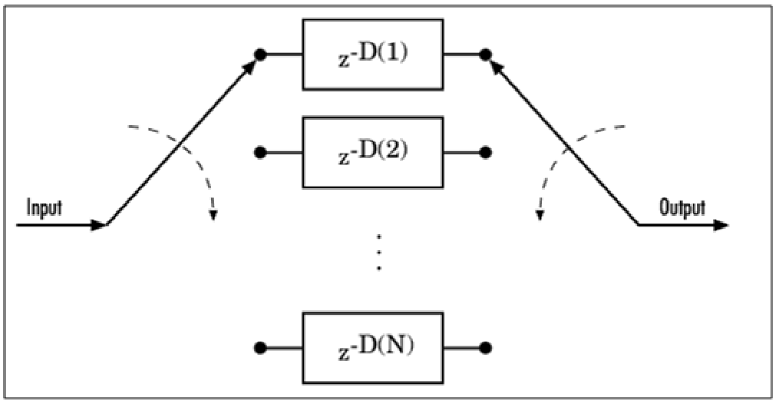


Figure 24: Convolutional Interleaver.

6.3 Random Interleaver

The Random interleaver block chooses a permutation table randomly using the initial seed parameter that is provided in the block mask. By using the same initial seed value in the corresponding random deinterleaver block, you can restore the permuted symbols to their original ordering.
Note: The block is predictable for a given seed, but different seeds produce different permutations. Hence, if we have same initial seed value at both random interleaver and random deinterleaver, the two blocks are inverses of each other, and hence we are able to get back original data sequence.
Number of elements parameter: how many numbers are in the input vector.
Initial seed: initialize the random number generator that the block uses to determine the permutation.

7 Equalizer

In communications, an equalizer is a device that attempts to reverse the distortion incurred by transmission through a channel.
In Digital Communication, it is used to reduce inter symbol interference (ISI) to allow recovery of the transmitted symbols. It may be a simple linear filter or complex algorithm.
Types of equalizers:

- Linear Equalizer: Process incoming signal with a linear filter.
  - Minimum Mean Square Error (MMSE) Equalizer: designs the filter to minimize $E[|e|^2]$, where e is the error signal, which is the filter output minus the transmitted signal.
  - Zero Forcing Equalizer: approximates the inverse of the channel with a linear filter.
- Decision Feedback: augments a linear equalizer by adding a filtered version of previous symbol estimates to the original filter output.
- Blind Equalizer: estimates the transmitted signal without knowledge of the channel statistics, using only knowledge of the transmitted signal's statistics.
- Adaptive Equalizer: automatically adapts to time varying properties of the communication channel. It is used with coherent modulations such as PSK, mitigating the effects of multipath propagation and Doppler spreading.
- Viterbi Equalizer: Finds the maximum likelihood optimal solution to the equalization problem. Its goal is to minimize the probability of making an error over the entire sequence.

7.1 Equalizers in Matlab and Simulink

- Linear equalizer:
  - Symbol spaced: consists of a tapped delay line that stores samples from the input signal.
  - Fractionally spaced: similar to symbol spaced, but a fractionally spaced equalizer receives K input samples before it produces one output sample and updates the weights, where K is an integer.
- Decision Feedback Equalizer: A non-linear equalizer that contains a forward filter and a feedback filter.
  Purpose of DFE: It cancels ISI while minimizing noise enhancement.
- MLSE (Maximum Likelihood Sequence Estimation): uses Viterbi algorithm.

The Toolbox supports the following adaptive algorithms:

- Least mean square (LMS)
- Signed LMS, including these types: sign LMS, signed regressor LMS, and sign-sign LMS
- Normalized LMS
- Variable-step-size LMS

- Recursive least squares (RLS)
- Constant modulus algorithm (CMA)

7.1.1 LMS Algorithm

Least mean squares (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method where the filter is only adapted based on the error at the current time.

An unknown system $h(n)$ is to be identified and the adaptive filter attempts to adapt the filter $\hat{h}(n)$ to make it as close as possible to $h(n)$, while using only the observable signals $x(n)$, $d(n)$ and $e(n)$. $y(n)$, $v(n)$ and $h(n)$ are not directly observable.

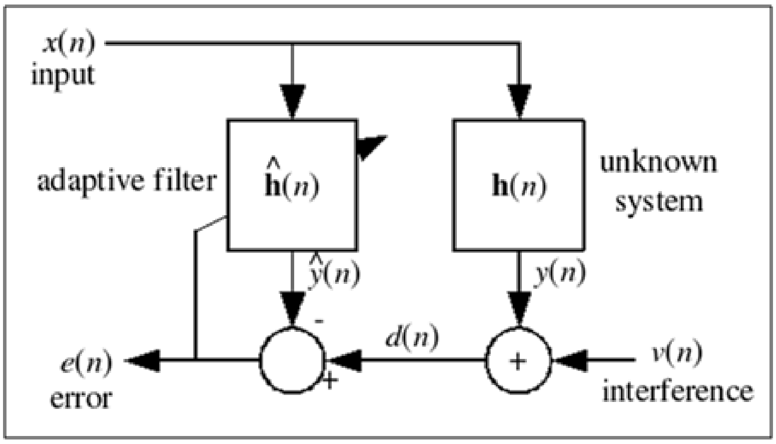


Figure 25: Schema of an LMS-Equalizer.

Implementation of LMS Equalizer in Simulink:

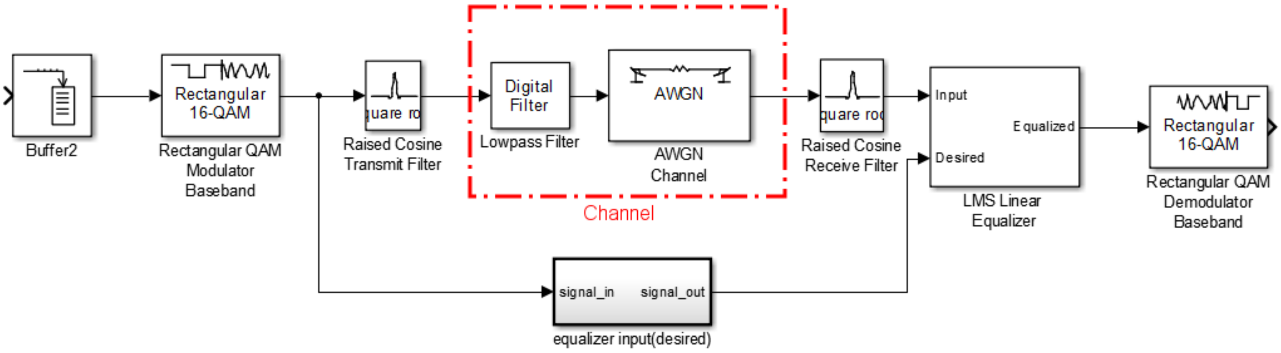


Figure 26: Equalizer Implementation in Simulink.

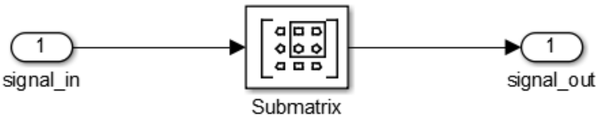


Figure 27: Equalizer Input (Desired) Implementation.

Laboratory Assignment – P1.6 – (optional / for the motivated ones)

1. Add a LMS Linear Equalizer to the previously created model with Hamming coding. Parameters for the LMS Linear Equalizer are:
   - Number of taps: 31
   - Number of Samples per symbol: 1
   - Signal constellation: *according to modulation and demodulation*
   - Reference tap: 15
   - Step size: 0.005
   - Leakage factor 1
   - Initial weights 0
2. Change the buffer size in front of the modulation to 224 Bits and adapt the rest of the model.
3. Set the Submatrix block for the desired equalizer input to output half of the original input signal.
4. Add a constellation diagram before and after equalization.
5. Check and if necessary adapt delays.
6. Introduce computation delays for the error rate calculations to compensate for the tuning phase of the equalizer to achieve a better comparability.
7. Simulate the model and compare the result to the ones without equalizer.
8. Change the step size of the equalizer to 0.0005 and simulate the model. What are the differences?

8 Channel Models

The term channel refers to the medium between the transmitting antenna and the receiving antenna. The characteristics of wireless signal changes as it travels from the transmitter antenna to the receiver antenna. These characteristics depend upon the distance between the two antennas, the path(s) taken by the signal, and the environment (buildings and other objects) around the path. The profile of received signal can be obtained from that of the transmitted signal if we have a model of the medium between the two. This model of the medium is called channel model.

In general, the power profile of the received signal can be obtained by convolving the power profile of the transmitted signal with the impulse response of the channel. Convolution in time domain is equivalent to multiplication in the frequency domain. Therefore, the transmitted signal x, after propagation through the channel H becomes y:

$$y(f) = H(f)x(f) + n(f) \tag{29}$$

Here $H(f)$ is channel response, and $n(f)$ is the noise. x, y, H, and n are all functions of the signal frequency f.

Key components of the channel response:

- Path loss: The simplest channel is the free space line of sight (LOS) channel with no objects between the receiver and the transmitter or around the path between them. In this simple case, the transmitted signal attenuates since the energy is spread spherically around the transmitting antenna. The presence of ground causes some of the waves to reflect and reach the transmitter. These reflected waves may sometime have a phase shift of 180° and so may reduce the net received power.
- Shadowing: If there are any objects (such buildings or trees) along the path of the signal, some part of the transmitted signal is lost through absorption, reflection, scattering, and diffraction. This effect is called shadowing.

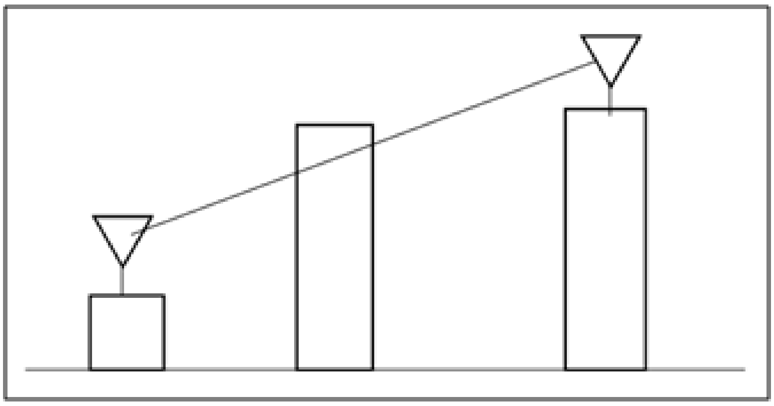


Figure 28: Shadowing.

- Multipath: The objects located around the path of the wireless signal reflect the signal. Some of these reflected waves are also received at the receiver. Since each of these reflected signals takes a different path, it has a different amplitude and phase.

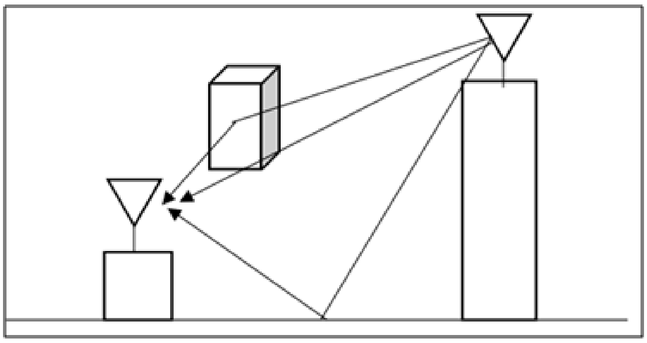


Figure 29: Multipath.

Depending upon the phase, these multiple signals may result in increased or decreased received power at the receiver. Even a slight change in position may result in a significant difference in phases of the signals and so in the total received power.

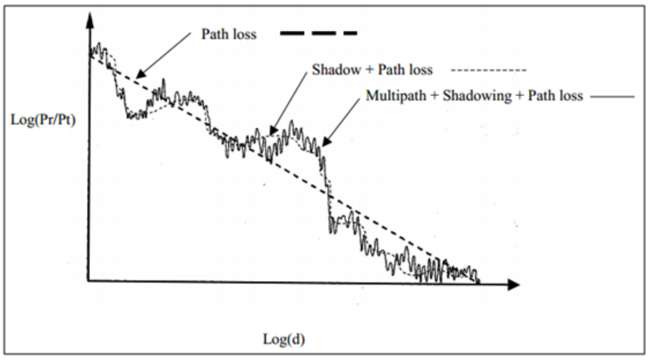


Figure 30: Pathloss, Shadowing and Multipath.

8.1 Additive White Gaussian Noise (AWGN)

AWGN is a channel model in which the only impairment to communication is a linear addition of wideband or white noise with a constant spectral density (expressed as watts per hertz of bandwidth) and a Gaussian distribution of amplitude. The model does not account for fading, frequency selectivity, interference, non-linearity or dispersion.

8.2 Rayleigh fading

Rayleigh fading is a statistical model for the effect of a propagation environment on a radio signal, such as that used by wireless devices. Rayleigh fading models assume that the magnitude of a signal that has passed through such a transmission medium (also called a communications channel) will vary randomly, or fade, according to a Rayleigh distribution – the radial component of the sum of two uncorrelated Gaussian random variables.
Rayleigh fading is most applicable when there is no dominant propagation along a line of sight between the transmitter and receiver.
Rayleigh fading is exhibited by the assumption that the real and imaginary parts of the response are modeled by independent and identically distributed zero-mean Gaussian processes, so that the amplitude of the response is the sum of such two processes.

8.3 Rician fading

Rician fading is a stochastic model for radio propagation anomaly caused by partial cancellation of a radio signal by itself – the signal arrives at the receiver by several different paths (hence exhibiting multipath interference), and at least one of the paths is changing (lengthening or shortening). Rician fading occurs when one of the paths, typically a line of sight signal, is much stronger than the others. In Rician fading, the amplitude gain is characterized by a Rician distribution.
The Rician k-factor specifies the ratio of specular-to-diffuse power for a direct LOS path. Usually, k varies from 1 to 10. A k-factor of 0 corresponds to Rayleigh fading.

8.4 Channel Modeling in Simulink

To model a channel that involves both fading and AWGN noise, use a fading channel block connected in series with the AWGN channel block, where the fading channel block comes first.
Compensate for Fading Response:

- Differential modulation or a one-tap equalizer helps compensate for a frequency flat fading channel.
- An equalizer with multiple taps helps compensate for a frequency selective fading channel.

Laboratory Assignment – P1.7 – (optional / for the motivated ones)

1. Open the model with Hamming coding and without equalizer.
2. Replace the low pass filter by the “Multipath Rayleigh Fading Channel”. Change the following properties of this block: Maximum Doppler shift to 5Hz, the discrete path delay vector to [0 0] and the average path gain vector to [0 -1] .

3. Simulate the model and compare the BER and constellation diagram to previous simulations.
4. Add a LMS equalizer to the model, adapt the model appropriately and rerun the simulation. Use the same parameters as in **P1.6** and step size 0.0005.
5. Set the step size of the LMS equalizer to 0.005 and compare the results. How can the differences be explained?
6. (Optional) Add a Random interleaver and deinterleaver and simulate the model.

References

[Kam11] Kammeyer, Karl-Dirk: *Nachrichtenübertragung*. Vieweg+Teubner Verlag, 2011

[MG02] Mäusel, Rudolf ; Göbel, Jürgen: *Analoge und digitale Modulationsverfahren: Basisband und Trägermodulation*. Hüthig Verlag Heidelberg, 2002

[PS05] Proakis, John ; Salehi, Masoud: *Fundamentals of Communication Systems*. Pearson Prentice Hall, 2005

[Skl01] Sklar, Bernard: *Digital Communications: fundamentals and applications*. Prentice Hall PTR, 2001