EECE 442 - Communication Systems
American University of Beirut, Fall 2025–2026

October 22, 2025
Due date: November 30, 2025

Course Project - Part II

Project Requirements

- Work in a groups of two (except one group of three in case odd).
- Matlab coding is recommended due to its robust communication system toolbox.
- Avoid using data that require special approval for animal or human subjects.
- The final grade will be based on the accuracy and completeness of your work, the efficiency and organization of the code, the quality of the report, and the oral presentation. Ensure that the code is testable, user-friendly, and well-documented. The percentage distribution of the grade on the different components is detailed in the course syllabus.

Description

This course project aims to strengthen your research and programming skills by guiding you through the **design** and **analysis** of digital communication systems. The project is structured, over two parts, into several modules, each covering essential baseband operations and offering hands-on insights into the foundations of digital communications.

By completing the modules, you will construct a full physical-layer communication simulation chain, along with an application for experimentation. As illustrated in Fig. 1, a continuous-time message signal $m(t)$, representing a target application, is first sampled into $m(nT)$ or expressed through a finite set of Fourier series coefficients (assuming communication of time-limited chunks of $m(t)$). The sampled sequence is then quantized into a vector $\mathbf{v}$, mapped into bits $\mathbf{b}$, channel-coded into a codeword $\mathbf{c}$, and modulated into complex symbols $\mathbf{x}$ for transmission. These symbols are affected by the channel $\mathbf{h}$ and additive noise $\mathbf{n}$, producing a received vector $\mathbf{y}$. On the receiver side, the inverse operations are performed: after equalization and detection, an estimate $\hat{\mathbf{x}}$ is obtained, followed by demodulation and decoding into $\hat{\mathbf{c}}$, $\hat{\mathbf{b}}$, and $\hat{\mathbf{v}}$, eventually reconstructing an approximate message signal $\hat{m}(t)$.

In this second stage of the project, you will focus on building and testing the channel encoder and decoder.

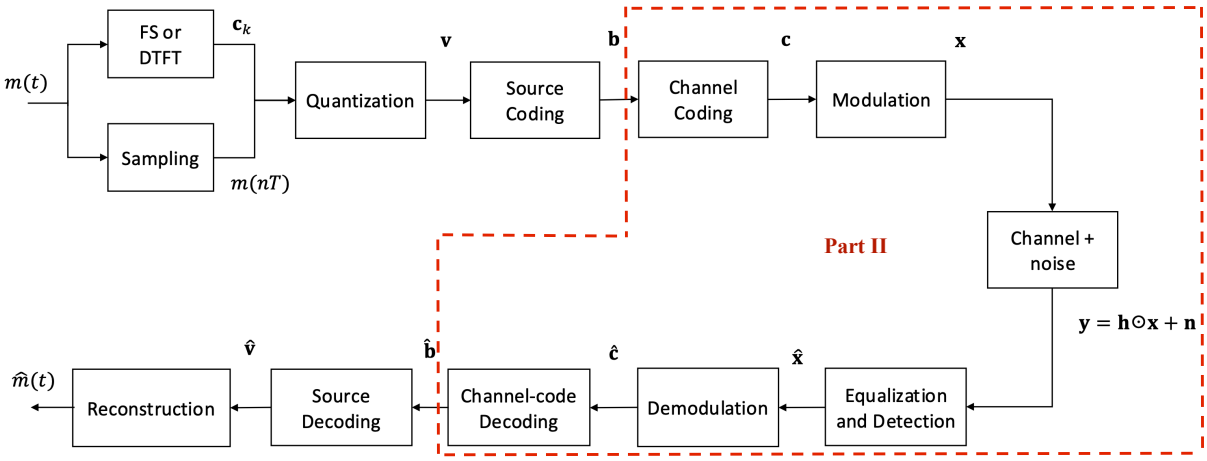

Figure 1: A physical-layer baseband communication system chain.

1 Channel Coding and Decoding

We did not study channel coding in class, but it was mentioned on a few occasions. You are not required to implement capacity-achieving codes or optimal channel-code decoders. Instead, you are asked to consider a basic form of channel coding which is *repetition coding*, which consists of adding redundancy bits to the output of the source encoder. The objective is to show that this type of coding reduces the overall errors. For example, you can experiment with repetition coding at a given rate and majority-vote decoding. Many off-the-shelf block codes are available that you can easily experiment with.

After completing the simulation chain, compare the overall coded and uncoded performance. The uncoded bit error rate (BER) is computed by tracking the average number of bit errors after demodulation and before channel-code decoding (between $\mathbf{c}$ and $\hat{\mathbf{c}}$). The coded BER is computed by tracking the average number of bit errors after channel-code decoding (between $\mathbf{b}$ and $\hat{\mathbf{b}}$). With variable source and channel codeword lengths, maintaining bit synchronization may require the addition of filler bits where needed.

2 Modulation and Demodulation

Here, you are required to construct modulation and demodulation functions at the level of constellation design, with pulse shaping as an optional bonus. For instance, implement Binary Phase Shift Keying (BPSK) modulation and demodulation using:

```
function x = bpsk_mod(b, A);   function b = bpsk_demod(x);
```

where b is a bit value, x is a real or complex symbol, and A is the amplitude. The modulation rule is:

$$x = \begin{cases} A, & \text{if } b = 1, \\ -A, & \text{if } b = 0. \end{cases}$$

The demodulation rule is:

$$b = \begin{cases} 1, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0. \end{cases}$$

Similarly, implement Quadrature Phase Shift Keying (QPSK) modulation and demodulation using:

`function x = qpsk_mod(b, A);` `function b = qpsk_demod(x);`

where the modulation rule is:

$$x = \begin{cases} \frac{A}{\sqrt{2}}(1+i), & b = 11, \\ \frac{A}{\sqrt{2}}(-1+i), & b = 10, \\ \frac{A}{\sqrt{2}}(-1-i), & b = 00, \\ \frac{A}{\sqrt{2}}(1-i), & b = 01, \end{cases}$$

and the demodulation rule is:

$$b = \begin{cases} 11, & \text{if } \mathrm{Re}(\tilde{x}) \geq 0, \mathrm{Im}(\tilde{x}) \geq 0, \\ 10, & \text{if } \mathrm{Re}(\tilde{x}) < 0, \mathrm{Im}(\tilde{x}) \geq 0, \\ 00, & \text{if } \mathrm{Re}(\tilde{x}) < 0, \mathrm{Im}(\tilde{x}) < 0, \\ 01, & \text{if } \mathrm{Re}(\tilde{x}) \geq 0, \mathrm{Im}(\tilde{x}) < 0. \end{cases}$$

After completing the simulation chain, compare the overall performance under different modulation schemes. You are encouraged to experiment beyond BPSK and QPSK. Note that different schemes yield varying bits per symbol, impacting the number of channel uses for the same message signal.

3 Channel, Noise, and Detection

3.1 AWGN channel

You will first test the communication link by transmitting modulated signals over an additive white Gaussian noise (AWGN) channel. The energy per symbol is denoted as E_s, and the amplitude A is given by $A = \sqrt{E_s}$. The channel model is expressed as:

$$y = x + n,$$

where $n \sim \alpha + j\beta$ is a sample of complex Gaussian noise, with $\alpha, \beta \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2 = 1/2)$.

Before interfacing with the application of interest, simulate error probabilities by generating a random bit sequence of length 10^6. Modulate these bits using BPSK first then using QPSK, add AWGN to the modulated symbols, detect the transmitted symbols from the received noisy signals, and demodulate the detected signals. Calculate the BER by counting the discrepancies between the transmitted and received bit sequences, then dividing by 10^6.

Plot the resultant uncoded BER against the signal-to-noise ratio (SNR), given by:

$$\mathrm{SNR} = 10\log_{10}(E_s).$$

Repeat the experiment with channel coding, and plot the coded BER under AWGN.

3.2 Linear Gaussian Channel

We consider the linear time-invariant ISI channel

$$y_i \;=\; 2x_i + x_{i-1} + n_i, \qquad n_i \sim \mathcal{CN}(0,1),$$

with QPSK modulation (Gray mapping) and unit symbol energy $E_s = 1$. Hence $E_b = E_s/2 = 1/2$. The constellation is $\mathcal{S} = \{\frac{1}{\sqrt{2}}(\pm 1 \pm j)\}$. We plot BER versus $\mathcal{E}_\mathrm{b}/N_0$ (in dB). The AWGN (no-ISI) uncoded QPSK reference is

$$P_b^{\mathrm{AWGN}}(\mathcal{E}_\mathrm{b}/N_0) \;=\; Q\big(\sqrt{2\,\mathcal{E}_\mathrm{b}/N_0}\big).$$

3.2.1 Option 1: Viterbi Equalizer (MLSE)

With channel memory 1, the ML sequence detector can be implemented by a Viterbi equalizer with 4 states corresponding to the previous symbol $s_{i-1} \in \mathcal{S}$. For each transition $s_{i-1} \to s_i$ the branch metric is

$$\big|\, y_i - (2s_i + s_{i-1}) \,\big|^2,$$

and the path metric is the accumulated sum. Survivor paths are updated according to the time index i and traceback yields $\{\hat{s}_i\}$. Finally, $\hat{s}_i$ are mapped back to bits (Gray de-mapping).

3.2.2 Option 2: Zero-Forcing (ZF) Linear Equalizer (BONUS)

Let the channel taps be $h = [h_0, h_1] = [2, 1]$ and choose an FIR equalizer of length L_g (e.g., $L_g = 5$). Define the convolution matrix $H \in \mathbb{C}^{(L_g+L_h-1)\times L_g}$ built from h, and let e_d be a unit vector with 1 at the target delay index d (typically near the center). The ZF filter is obtained via least squares:

$$g_{\mathrm{ZF}} \;=\; \arg\min_{g} \;\|Hg - e_d\|_2^2 \;=\; (H^*H)^{-1}H^*e_d \quad \text{(or via QR)}.$$

The equalized sequence is $z_i = (g_{\mathrm{ZF}} * y)_i$, and decisions are made by nearest-neighbor detection to $\mathcal{S}$.

Noise enhancement: the channel frequency response $H(e^{j\omega}) = 2 + e^{-j\omega}$ has small magnitude near $\omega = \pi$, so ZF may significantly amplify noise at those frequencies, degrading BER.

3.2.3 BER Estimation Procedure (Simulation)

(a) Generate i.i.d. bits, map to QPSK symbols $x_i \in \mathcal{S}$ with $\mathcal{E}_s = 1$.

(b) ISI channel: $y_i = 2x_i + x_{i-1} + n_i$, with $n_i \sim \mathcal{CN}(0, \sigma^2)$. Using $\mathcal{E}_b = 1$, set $N_0 = 1/(\mathcal{E}_b/N_0)_{\text{lin}}$ and $\sigma^2 = N_0$ per complex symbol.

(c) **MLSE**: run the Viterbi equalizer with the above branch metric, detect the symbols, and compute BER.

(d) **AWGN baseline**: simulate $y_i = x_i + n_i$ and compare to $Q(\sqrt{2\mathcal{E}_b/N_0})$.

(e) **ZF** (optional): design g_{ZF} (once per SNR or fixed), equalize $z_i = (g_{\text{ZF}} * y)_i$, detect to QPSK, calculate BER.

Discussion Points:

- **Uncoded ZF vs. MLSE:** MLSE typically outperforms ZF at low–moderate SNR due to ZF's noise enhancement.
- **High SNR:** MLSE approaches the AWGN benchmark; ZF may show a persistent gap resulting from residual ISI and spectral inversion loss.
- **Complexity:** With memory 1 and QPSK, Viterbi has 4 states and 16 branches per time step; computationally modest.