# 5ETC0 - Communication I

# Lab 2

May 27, 2024

**NOTE:** Before you start the lab, make sure you have the following toolboxes installed:**Communications Toolbox**, **Control System Toolbox**, **DSP System Toolbox**, **Data Acquisition Toolbox**, **Instrument Control Toolbox**, **Optimization Toolbox**, **Signal Processing Toolbox**, **Symbolic Math Toolbox**, **System Identification Toolbox**. You may already have them installed, but double-check by running `ver` in the command window. This will allow you to see all of the Toolboxes that have already been installed.
Furthermore, exercises 2 and 3 require you to use the `designfilt` MATLAB function. Focus on the 'StopbandFrequency', 'PassbandFrequency' and 'StopbandAttenuation' variables. Make sure to use FIR filters, because they have a constant delay. An example is shown in Fig. 2 and can be obtained using `fvtool`.

## Exercise 1: Raised Cosine

The application `appChapter_3.6`, located in the 'Exercise 1' folder, should be downloaded, installed, and opened in MATLAB. To install it, double-click the `appChapter_3.6` file, and then select 'Install' when MATLAB opens.

1. Upload an integer message with a bit time $T_b$ of 0.001 s plot the channel output without filtering it. Now turn filtering on and filter the same signal with a roll-off factor of 0.2 and plot the output again. Repeat this for $r = 0.6$ and $r = 1$ and again for a bit time of $T_b = 0.00005$s (so 6 plots total). Compare the output of the channel for each $r$ and $T_b$.
   How does the spectrum change when the roll-off factor is increased/decreased? What does the bit rate change?
2. Is it practical to make the roll-off factor as small as possible? Why or why not?
3. What are the advantages of using raised cosine signals?

## Exercise 2: Filtering

Open `lab2_ex2.m`. This file is loading `noisySignal.mat` containing `signal`. This is a gated PAM signal that passed through a channel that is contaminated with low-frequency noise.

1. Plot it's PSD using `periodogram(signal, [], [], R_s, 'centered')`.
2. Try to demodulate this signal to achieve the original spectrum. This can be done by multiplying it with the local oscillator (cosine with correct frequency), and applying low pass filtering, see Fig 1.
3. Plot signal, s_c and s_f to get an idea of what is happening at each stage of demodulation.

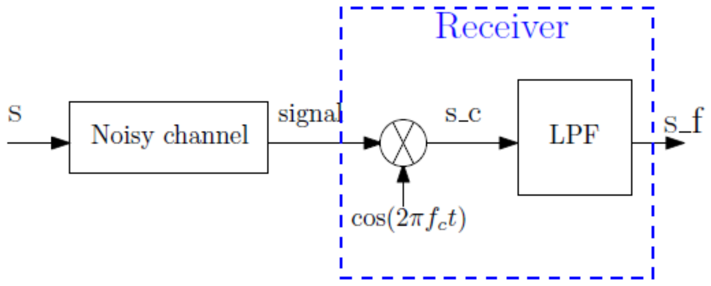

Figure 1: Block diagram of the most simple receiver circuit. This is the basis of demodulation of PAM that has been contaminated with low-frequency noise.

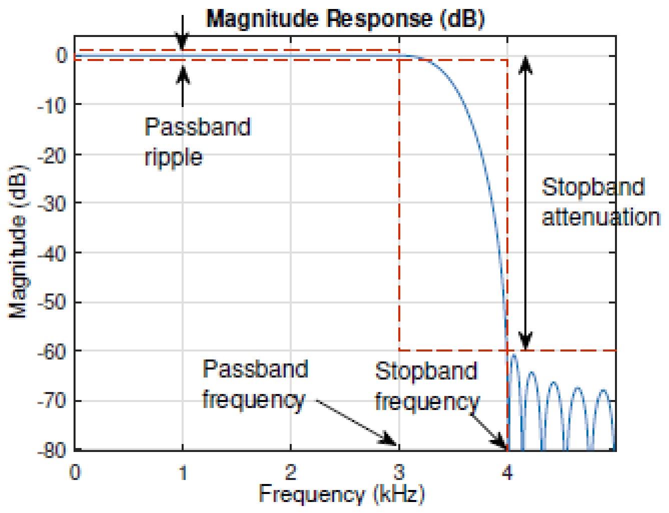

Figure 2: Filter response of filter `lpfilt = designfilt('lowpassfir', 'StopbandFrequency', 4e3, 'PassbandFrequency', 3e3, 'PassbandRipple', 2, 'StopbandAttenuation', 60, 'SampleRate', 10e3, 'DesignMethod', 'kaiserwin')`, obtained using `fvtool`.

**NOTE:** The method shown in the caption of Fig. 2 is a more convenient way to define the filter parameters in the MATLAB code.

## Exercise 3: Demodulating and decoding line coded messages

Open `lab2_ex3.m`. Three messages are sent through a channel at different carrier frequencies, using different line coding, see Fig. 3. Load the data from `sharedChannel.mat` and plot the spectrum of the waveform using the `periodogram` function. For each filter, you have to define the Bandwidth of the signal. Due to the limited accuracy of the decoder the filter needs to include more information of the signal, therefore the filter settings have been predefined.

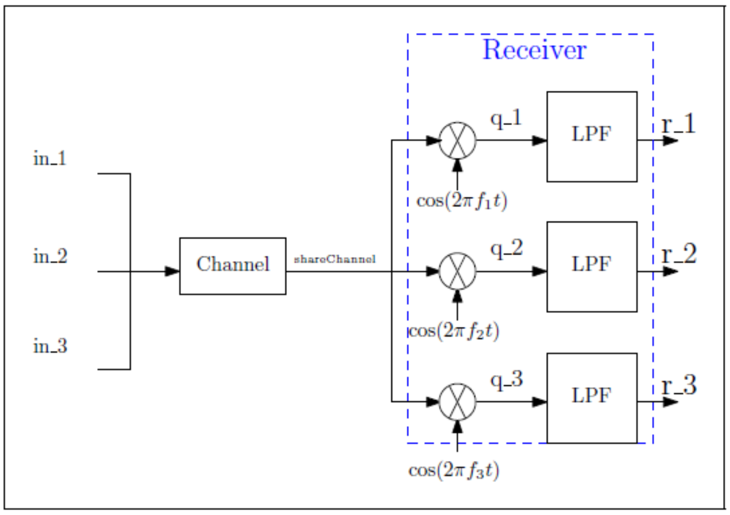

Figure 3: Block diagram Exercise 2

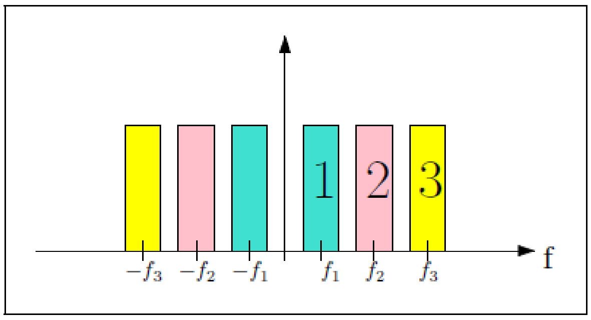

Figure 4: A simplified diagram of the PSD of the signal `shareChannel`

1. Demodulate the carriers, as in the last exercise, by multiplying them with appropriate local oscillator, and filtering.

2. What line codes are used in each case?

3. What are the messages encoded by these signals? Use the function `DecodeMessage.p` provided in the exercise folder to decode the digital messages into sentences.

   The function `DecodeMessage` is defined as:

   ```
   output = DecodeMessage(received message, time, number, Linecode)
   ```

   Here:

   (a) `received message` is the message you get after low-pass filtering.

   (b) `time` is a predefined parameter that you don't have to change

   (b) `number` is the number assigned to the spectra you're decoding (seen in Fig. 4).

   (c) `Linecode` is a string defining the type of line coding used. `type` can be equal to 'unipolarNRZ', 'unipolarRZ', 'polarNRZ', 'bipolarRZ', 'manchesterNRZ'.

## Exercise 4: Choosing a line code for a channel

Open `Lab2_LineCodes.m` and `Lab2_LineCoding.slx`.
You are given two channels with different spectrum properties. To test what these properties are, you will first pass a noise signal through the channels, and inspect their frequency response with the Spectrum Analysis block. The blocks to use are in the file and can be dragged and attached to the channel.

**NOTE:** Unused blocks may generate errors. To avoid this, you can comment out the unnecessary blocks with **Ctrl+Shift+X**. If the Simulink simulation does not compile, delete the **'slprj'** file in the 'Exercise 4' folder and run the simulation again.

1. What characteristics can be attributed to Channel 1?
2. What kind of line code would you use to transmit data over this channel?

   Channel 2 represents a frequency band that is chosen for a certain transmission system.
3. What characteristics can be attributed to Channel 2?
4. What is the bandwidth and center frequency of the channel?
5. What is the significance of the -3dB point of a signal spectrum?
6. Channel 2 only has two voltage levels - therefore, you can only use 1s and 0s for transmission. Which line code(s) can you send through the channel?
7. Do you need to modulate the signal of the previous question to pass it through this channel successfully?
8. In the MATLAB (`.m`) file, the function `lineCode` can encode a sequence of random bits into a line code of your choosing:

   ```
   [encode, fs] = lineCode(binary, type, Tb)
   ```

   Here, `binary` is the random binary vector, `Tb` is the bit time, and `type` is the chosen line code as a string.

   Line codes can be chosen from 'unipolarNRZ', 'polarNRZ', 'unipolarRZ', 'bipolarRZ' and 'manchesterNRZ'.

   (a) For Channel 1, choose a line code you specified in question 2. What bit time `Tb` should you choose?

   (b) `encode` is now your up-sampled, line coded bit sequence. You can now use the `Signal From Workspace` block in the Simulink model to pass the signal through the channel. If you use the carrier signal, make sure to specify the carrier frequency, `Fc`, for the sine wave in the `Multiplication with carrier block`.

   Inspect the spectrum at the output of the channel and compare with the input spectrum.

   i. What is the SNR of this channel?

   ii. What is the attenuation of this channel?

   iii. Could the signal be restored without information loss?

   (c) For Channel 2, choose a line code you specified in question 5. What bit time `Tb` should you choose?

   (d) Repeat steps i-iii as in (b), but this time for Channel 2.

## Appendix

| **MATLAB function** | **Description** |
|---|---|
| $t$ | Vector representation of time. |
| $r = randi([min(r)\ \ max(r)], rows, columns)$ | Returns an array of dimensions *rows* x *columns* composed of integers drawn from the interval [imin,imax]. |