Project

In this project assignment, in all parts consider a received signal model

$$y_k = \sum_{l=0}^{L} h_\ell x_{k-\ell} + n_k$$

where f is the *causal* ISI channel impulse response, x_k is the binary data symbol at time k ($x_k \in \{-1, 1\}$), and n_k is the AWGN noise sample ($n_k \sim N(0, \frac{N_0}{2})$). Consider the ISI channel

$$h_\ell|_{\ell=0}^{4} = \begin{bmatrix} 0.74 & -0.514 & 0.37 & 0.216 & 0.062 \end{bmatrix}$$

that is normalized ($\sum h_\ell^2 = 1$).

1. In class we have seen ZF, MMSE and DFE equalization techniques, which are all based on digital filtering approaches. Eventually, all of these approaches are suboptimal in the sense that at their output, minimum distance (or maximum likelihood) rule is employed for detection despite the presence of residual ISI and colored noise. To circumvent this problem, one would have to employ equalization based on maximum-likelihood sequence estimation (MLSE) as we discussed in class as well. Notice that the brute force approach to MLSE requires M^N computations where M is the symbol alphabet size and N is the length of the sequence and this exponentially increasing complexity quickly makes the usage of this technique unviable. The Viterbi algorithm is an elegant method to perform optimum MLSE without the exponential complexity. In this part, you are required to investigate the fundamentals of the Viterbi Algorithm, write a Matlab code implementing the Viterbi equalizer for the channel given above. Also simulate the 10-tap causal MMSE and ZF equalizers for the same channel using your codes from the previous homework assignment and compare the three performances.

2. In class, we have seen the MMSE equalization in time-domain. An alternative alternative approach is to perform equalization in the frequency domain. In this approach, called frequency-domain equalization (FDE), you consider blockwise transmission of symbols (say N symbols per block) and add a cyclic prefix (CP) to each block. With proper addition of the CP, at the receiver side you can see the linear convolution of the ISI channel with the transmitted symbol sequence being equivalent to a circular convolution. Then you can take the discrete Fourier transform (DFT) of the received signal and perform the equalization in the DFT domain. In this part, you are required to learn the principles of FDE and write a Matlab code implementing the FDE with 1, 3, 5, 10 cyclic prefix symbols. You can assume that in each block you are sending 1000 data symbols. Evaluate the performance and compare it with that of a 10-tap time-domain MMSE equalizer. Would there be a performance difference as you change the cyclic prefix length? Why?

3. Consider the ISI channel that is given above but contrary to what we did in class assume that the receiver does not know the channel. In this case, pilot based adaptive equalization is often an effective solution. In this approach you insert some pilot symbols, which are known to both the transmitter and receiver, to the beginning of each transmitted symbol block. Then you can design an adaptive equalizer which starts training the equalizer coefficients with these known pilot symbols and then updates the equalizer coefficients in every symbol instant. In this part, you are required to investigate the fundamentals of adaptive equalization and implement two adaptive equalizers, one based on the least mean quares (LMS) algorithm and the other based on the recursive least squares (RLS) algorithm. You can assume that in each block you are sending 1000 data symbols. Consider the addition of 10, 20 and 30 pilot symbols to each of your data block. How do the performances of LMS and RLS adaptive equalizer compare? How do the performances of each equalizer change based on the number of pilots?

Before generating the figures and preparing your report, go through the next page.

Instructions on Figures, Report and Submission:

Your final project should include 2 out of the 3 parts given above. The choice of which two parts is up to your choice. Interested individuals may submit all three parts for an added bonus.

You are expected to implement the equalizers on your own, therefore do not use the equalization toolbox. Your code has to be understandable (commenting, meaningful variable names etc).

In each part, create semi-logarithmic BER vs E_b/N_0 figures by following the instructions below.

- Name each curve, show the legend and make sure the legend does not block the curves. (*semilogy(...,'DisplayName',CurveName)*).
- All curves will be straight lines (no dashed or else.)
- Axes have to be squared (*axis square*).
- Background have to be gridded (*grid on*).
- Set the font size of the figure to 14 (*set(gcf,'FontSize',14)*).
- Label both axes. No need for a title, instead caption the figures in your report.
- Make sure your curves go down to at least sufficiently small BER levels, e.g., 10^{-4} BER level.
- Limit the axes appropriately (no blank spaces).

For each part you submit, you are expected to write a report explaining the algorithm and implementation details and to comment on the change in performances of different equalizers. Zip the m-files with your report pdf and upload it on the moodle. The due date for the projects is December 31st, 2019. Late submission is penalized with 10% per day.