Term Project

Adaptive Channel Equalization

The purpose of adaptive channel equalization is to compensate for signal distortion in a communication channel. Consider a digital communication system that transmits data from one terminal to another through an unknown wireless multipath channel with additive white Gaussian noise (AWGN). During the transmission process, a transmitted signal that contains information bits becomes distorted at the receiver. For the sake of recovering the information bits (or called original data), one can apply an adaptive filter to the received signal as a way of distortion compensation.

As illustrated in Figure 1, an adaptive filter works as a channel equalizer (i.e., adaptive equalizer) with two operating modes as follows:

1) **Training Mode**: The adaptive equalizer works with a known training sequence to identify the channel characteristics.
2) **Decision-Directed Mode**: The output of the decision device is used to generate the error signal for adaptation of the filter coefficients.

At the transmitter, the training sequence followed by a sequence of binary phase shift keying (BPSK) data is transmitted over the channel. At the receiver, the received signal $v(n)$ is passed through the adaptive equalizer, whose output $y(n)$ is fed into the decision device for symbol-by-symbol detection with the following rule:

$$z(n) = \begin{cases} +1, & \text{if } y(n) \geq 0 \\ -1, & \text{otherwise.} \end{cases}$$

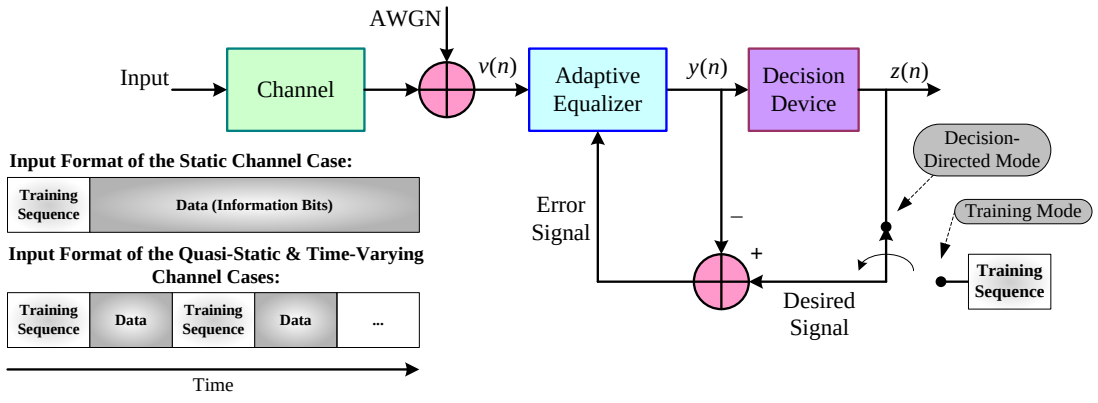

Figure 1. An adaptive equalizer for digital communications.

Initially, the adaptive equalizer operates in the training mode, using the known training sequence as the desired signal. Then, it switches to the decision-directed mode, where the output of the decision device is directly used as the desired signal.

In this project, please design adaptive channel equalizers to recover the original data for each of the following channels: a static channel, a quasi-static channel, and a time-varying channel. Also, compare the output of the decision device in the decision-directed mode with the transmitted data to evaluate the bit error rate (BER) performance of the designed adaptive equalizers. You will have to showcase your design in terms of the algorithm design/analysis, BER performance, and computational complexity.

I. Static Channel Case:

For the static case, the channel is fixed during the transmission, and the channel input consists of a *1000-bit training sequence followed by a 200000-bit data sequence.*

II. Quasi-Static Channel Case:

For the quasi-static case, the channel remains unchanged within a block interval, where the channel input for each block interval consists of a *200-bit training sequence followed by a 1000-bit data sequence*. In this case, the training mode and the decision-directed mode are required for each block interval.

III. Time-Varying Channel Case:

For the time-varying case, the channel changes rapidly and may change during a block interval, where the channel input for each block interval consists of a *50-bit training sequence followed by a 400-bit data sequence*. In this case, the training mode and the decision-directed mode are required for each block interval.

Detailed Project Information:

BPSK modulation: +1, −1

Data file: **project_data2021.mat** (You can use MATLAB function '`load`' to open it)

Part 1: Static Channel Case

- trainseq_static_1 and trainseq_static_2: The known training sequence with length of 1000 bits
- data_static_1 and data_static_2: The received sequence, which is given in the format of data_static = [**a**, **b**], where **a** is the received training sequence and **b** the received data (information bits) sequence.
- Design an adaptive equalizer to recover the original data.
- Save the recovered data as ***ans_static.mat*** (including **ans_static_1** and **ans_static_2**).

Part 2: Quasi-Static Channel Case

- trainseq_qstatic_1 and trainseq_qstatic_2: The training sequence with length of 200 bits. Note that this training sequence is applied whenever operating in the training mode.

- data_qstatic_1 and data_qstatic_2: The received sequence, given as the format of data_qstatic = [$\mathbf{a}_1$, $\mathbf{b}_1$, $\mathbf{a}_2$, $\mathbf{b}_2$,…, $\mathbf{a}_{200}$, $\mathbf{b}_{200}$], where $\mathbf{a}_n$ are the received training sequences; $\mathbf{b}_n$ are the received data (information bits) sequences, 1000 bits for each.
- Design an adaptive equalizer to recover the original data. (Note: There are 200 times that the system operates in the training mode, and 200 times in the decision-directed mode, where each training sequence of 200 bits is followed by a data sequence of 1000 bits.
- Save sequentially the recovered data as ***ans_qstatic.mat*** (including **ans_qstatic_1** and **ans_qstatic_2**).

Part 3: Time-Varying Channel Case

- trainseq_varying_1 and trainseq_varying_2: The training sequence, with a length of 50 bits, applied in the training mode.
- data_varying_1 and data_varying_2: The received sequence, given as the format of data_varying = [$\mathbf{a}_1$, $\mathbf{b}_1$, $\mathbf{a}_2$, $\mathbf{b}_2$,…, $\mathbf{a}_{500}$, $\mathbf{b}_{500}$], where $\mathbf{a}_n$ is the *n*-th received training sequences; $\mathbf{b}_n$ is the *n*-th received data (information bits) sequences, 400 bits for each.
- Design an adaptive equalizer to recover the original data. (Note: There are 500 times that the system operates in the training mode, and 500 times in the decision-directed mode, where each training sequence of 50 bits is followed by a data sequence of 400 bits.)
- Save sequentially the recovered data as ***ans_varying.mat*** (including **ans_varying_1** and **ans_varying_2**).

Note:

- The sizes of data used for simulations are given as follows:

  trainseq_static: 1×1000; data_static: 1×201000
  trainseq_qstatic: 1×200; data_qstatic: 1×240000
  trainseq_varying: 1×50; data_varying: 1×225000

- The simulation results have to be recorded in the following format:

  ans_static: ans_static_1 (1×200000) and ans_static_2 (1×200000);
  ans_qstatic: ans_qstatic _1 (1×200000) and ans_qstatic _2 (1×200000);
  ans_varying: ans_varying _1 (1×200000) and ans_varying _2 (1×200000).