ENEL 671 **PROJECT # 3** **Fall 2016**

ADAPTIVE EQUALIZATION OF A COMMUNICATION CHANNEL using RLSL ALGORITHM

In this project you will study the use of the Recursive least squares lattice (RLSL) algorithm for adaptive equalization of a linear dispersive channel that produces (unknown) distortion. We assume that the data are all real valued. Figure 1 shows a block diagram of the system to be used for the purpose of carrying out the study. The data source generates the signal $\{a(n)\}$ used for probing the channel while the Gaussian noise source generates white noise $\{\nu(n)\}$, with zero mean, that corrupts the channel output. These two random-number generators are statistically independent of each other. The adaptive linear equalizer, which is a tap-delay line filter, has the task of correcting for the distortion introduced by the channel in the presence of the additive white noise. The signal $\{a(n)\}$, after suitable delay, supplies the desired response $\{d(n)\}$ applied to the adaptive equalizer for computing the estimation error $\{e(n)\}$ sequence. This delay is equal to the time it takes the signal to propagate through the channel (the medium) and the adaptive filter.

The random sequence $\{a(n)\}$ applied to the channel input is in polar form (BPSK), that is, $a(n) = \pm 1$, so the sequence $\{a(n)\}$ has zero mean and variance equal to 1. Four channels, each consisting of three paths, will be investigated in this project. The impulse responses of the channels are given in Table 1.

Table 1

	$h(1)$	$h(2)$	$h(2)$
Channel 1	0.2194	1.0	0.2194
Channel 2	0.2798	1.0	0.2798
Channel 3	0.3365	1.0	0.3365
Channel 4	0.3887	1.0	0.3887

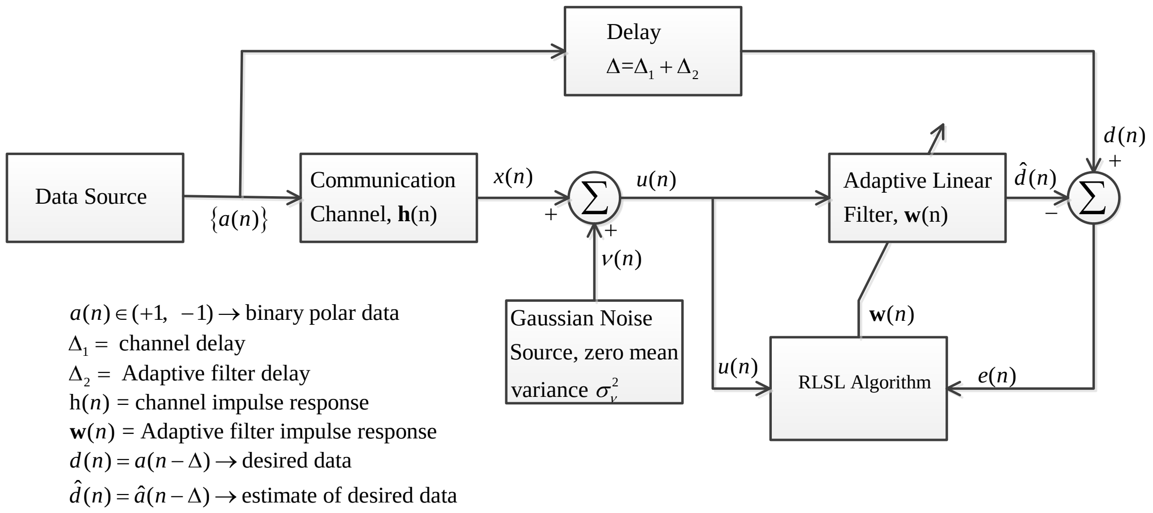
Delay
$\Delta = \Delta_1 + \Delta_2$
Data Source
$\{a(n)\}$
Communication Channel, $\mathbf{h}(n)$
$x(n)$
$u(n)$
$\nu(n)$
Adaptive Linear Filter, $\mathbf{w}(n)$
$\hat{d}(n)$
$d(n)$
Gaussian Noise Source, zero mean variance σ_ν^2
$\mathbf{w}(n)$
RLSL Algorithm
$e(n)$
$a(n) \in (+1, -1) \to$ binary polar data
$\Delta_1 =$ channel delay
$\Delta_2 =$ Adaptive filter delay
$h(n)$ = channel impulse response
$\mathbf{w}(n)$ = Adaptive filter impulse response
$d(n) = a(n-\Delta) \to$ desired data
$\hat{d}(n) = \hat{a}(n-\Delta) \to$ estimate of desired data

Figure 1 Block diagram for the equalization project

The noise sequence $\{\nu(n)\}$, has zero mean and variance σ_ν^2. The input to the equalizer filter is given by the convolution sum:

$$u(n) = \mathbf{h}^T \mathbf{a}(n) + v(n); \qquad n = 1, 2, \ldots, N$$
$$= h_1 a(n-1) + h_2 a(n-2) + h_3 a(n-3) + v(n)$$
$$a(n-i) = 0, \qquad n - i < 0$$

In this project you are required to solve the equalization problem using the joint process estimator. You will recursively compute the lattice filter parameters and the regression coefficients using the RLSL algorithm. You will evaluate the mean-square error of the adaptive equalizer with respect to changes in the eigenvalue spread χ. An approximate mean-square error curve (the learning curve) is obtained by averaging the instantaneous squared error versus over K independent runs (or experiments) of the RLS algorithm as you did in Projects 1 and 2. The a priori MSE is computed as follows (for the final stage):

$$MSE_p(n) = \frac{1}{K} \sum_{k=1}^{K} e_k^2(n); \qquad n = 1, 2, \ldots, N$$

$e_k(n)$ is the a priori error for the final stage of the k^{th} experiment. The a posteriori error for the final stage, $e_M(n)$, is related to the a priori error, $\alpha_k(n)$, as follows:

$$\alpha_M(n) = \frac{e_M(n)}{\gamma_M(n)}$$

The a posteriori MSE, that you need to plot is therefore given by

$$MSE(n) = \frac{MSE_p(n)}{\gamma_M(n)}$$

The factor N is the number of data samples $\{a(n)\}$, for a single run (or experiment) of the algorithm and K (the outer loop) is the number of independent runs required for the averaging. To ensure that the runs are independent, the seeds used to generate the data and noise samples must be different for each experiment k. The curves must be plotted on the same graph in semi-log scale like those for the LMS and RLS algorithms.

1. Effect of Eigenvalue Spread

(a) For the parameters given in the table below, generate and plot the learning curves for Channels 1, 2, 3, 4 and 5, all on one graph. The curves must all be plotted on semi-log scale. If you plot $MSE_p(n)$ and $MSE(n)$, side-by-side, on the same graph, you will understand why the normalization is necessary, that is,

$$MSE(n) = \frac{MSE_p(n)}{\gamma_M(n)}.$$

δ	0.01
Prediction Filter Order	$M = 10$
Number of data samples N	≥ 500
Number of independent runs (experiments) K	≥ 250
SNR	**40 dB**

Note: Since the regression coefficients start from $m = 0$, you must choose $M = 10$ so that the overall number of coefficients is 11 and you can compare with the tap-delay line filters.

(b) For Channel 1, do the following (Note: All the parameters are averaged):

(i) Plot the reflection coefficients, $\Gamma_{f,M}(n)$ and $\Gamma_{b,M}(n)$, as functions of iteration n for the final stage. Note that the final stage is $M = 10$. You need to comment on the behavior of these coefficients.

(ii) Plot the likelihood parameter, $\gamma_M(n)$, for the final stage, as a function of iteration n. You need to comment on the behavior of these coefficients.

(iii) Sketch the **steady state values** of the regression coefficients, that is κ_i, $i = 0,1,\ldots,M$. In your report, you need to talk about how the regression coefficients and the tap weight coefficients obtained for the RLSL compare.