

# Spectrum Access System: Comparison of Different Equalizers

*Anant Sahai  
John Wawrzynek  
Felicity Zhao*

Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2017-87

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-87.html>

May 12, 2017



Copyright © 2017, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

#### Acknowledgement

Many Thanks to my supervisor Professor Anant SAHAI and John WAWRZYNEK. And many thanks to my teammates Cindy Chen and Heyi Sun.

# **Spectrum Access System**

**Zhuangyi (Felicity) Zhao**

**University of California, Berkeley**

**Student ID:3032166476**

# Abstract

Intersymbol interference is an unwanted phenomenon that makes communication less reliable. However, an equalizer can reduce the bad influence of intersymbol interference. In general, my work is to compare different equalizers in various scenarios. In the first part of this paper, I explain the motivation of using equalization in detail, followed by introducing prerequisite knowledge of equalization. Then, I will introduce three main adaptive algorithms, least mean square (LMS), Recursive least square (RLS) and Constant Modulus Algorithms (CMA). Then, various structures of equalizers are depicted as well, including Linear Equalizer, Decision-Feedback Equalizer and Maximum-Likelihood Sequence Estimation. Finally, simulation results are used to compare their performances.

*Keywords: Equalizer Comparison, Least Mean Square Algorithm, Recursive Least Square Algorithm, Constant Modulus Algorithm, Linear Equalizer, Decision-Feedback Equalizer, Maximum-Likelihood Sequence Estimation*

# 1. Motivation

In wireless communication, intersymbol interferences (ISI) is a common phenomenon, which reduces the signal transmission accuracy. Luckily, with the help of equalizers, intersymbol distortion will be reduced so that communication transmission performs better.

## 1.1 Intersymbol Interference (ISI)

Suppose a discrete input signal  $x(t)$  is transmitted over an analog channel with channel response  $h(t)$ . According to David Smalley (1994)<sup>[1]</sup>, the received signal  $r(t)$  is a convolution of the input sequence by a continuous time channel response.

$$r(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau) d\tau \quad (1.1.1)$$

Yet, the input signal is discrete and should be transmitted at  $t = kT$  so the resulting signal should be sampled on signal hardware at  $t = nT$ . So, formula 1.1.1 can be rewrite as

$$r(nT) = \sum_{k=-\infty}^{\infty} x_k h(nT - kT) = x_n h(0) + \sum_{k \neq n} x_k h(nT - kT) \quad (1.1.2)$$

The first term is the component of  $r(t)$  due to the  $N^{\text{th}}$  symbol, which is multiplied by the center tap of the channel-impulse response  $h(0)$ . The other product terms in the summation are intersymbol interference terms.

## 1.1 Intersymbol Interference Simulation

Next, I will use an example to illustrate the importance of equalization.

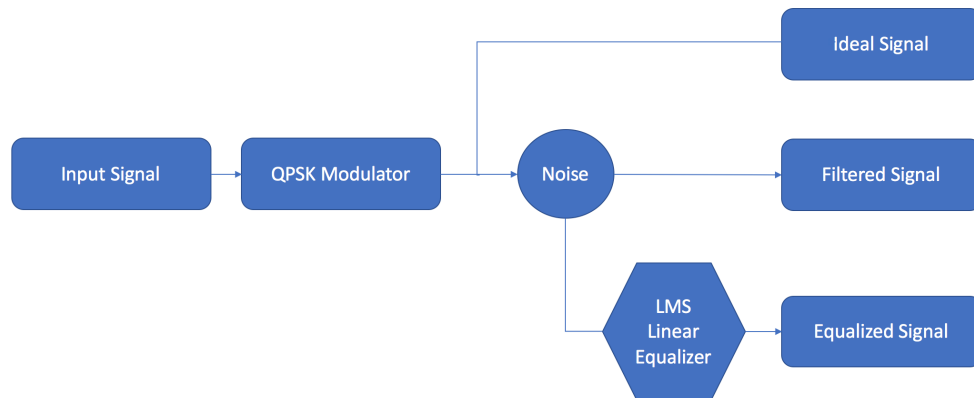


Figure 1.2.1 Matlab program to illustrate the idea signal and equalized signal.

Consider the model depicted in figure 1.2.1. The input signals are 1500 random samples among 0, 1, 2, 3. Then these signals are modulated using Quadrature Phase Shift Keying (QPSK). After that, these symbols are transmitted to the channel:  $H(z) = 0.986 + 0.845z^{-1} + 0.237z^{-2} + (0.123 + 0.31i)z^{-3}$  to simulate the signal distortion.

Without noise, the ideal signals should be settled at 1,  $-i$ ,  $i$ ,  $-1$  as shown in figure 1.1.2.

However, with noise and intersymbol interference, the filtered signal creates a large difference from the ideal signal. Luckily, with the help of LMS Equalizer, step size 0.01 and 8 taps, the equalized signal should be closer to the ideal signal as shown in figure 1.1.3.

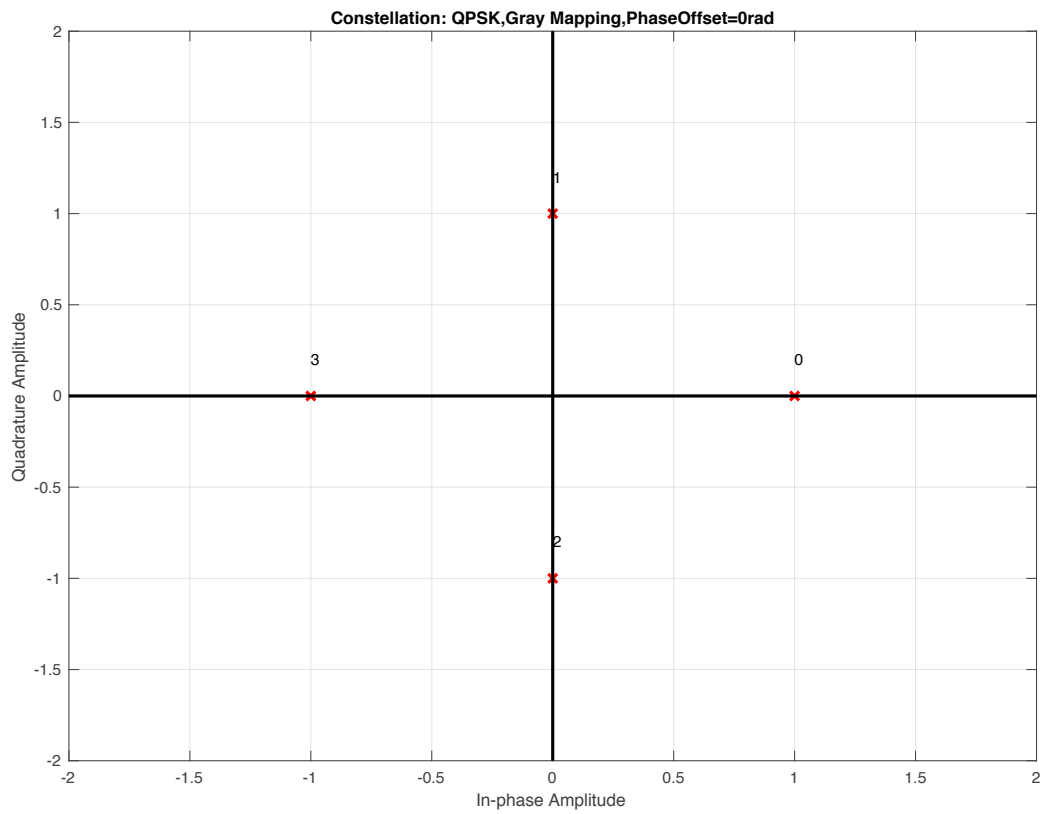


Figure 1.1.2 Idea signal constellation plot for QPSK.

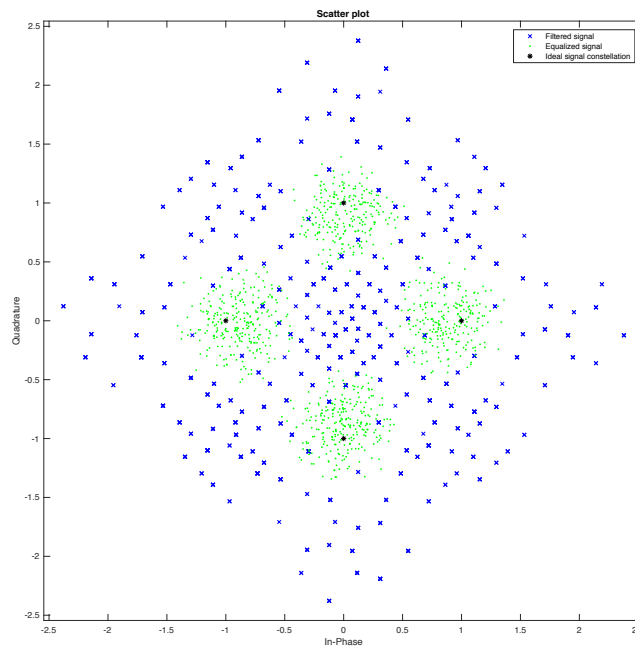


Figure 1.1.2 Constellation comparing filtered signal with equalized signal and the ideal signal.

The above figure shows the difference between the ideal points, which are four black points located at 1, -1,  $i$ ,  $-i$ , the received points after equalization, which are the green points located around the desired signal, and the received points without equalization, which are the blue points scattered all around the surface. It is clear that the equalizer creates a huge impact in signal transmission. This is why I focused on the equalizer for this capstone project.



## 2. Equalization

### 2.1 Equalization Process

As depicted in Figure 2.1.1, the task of the equalization process is to apply a filter that results in a signal having less ISI.

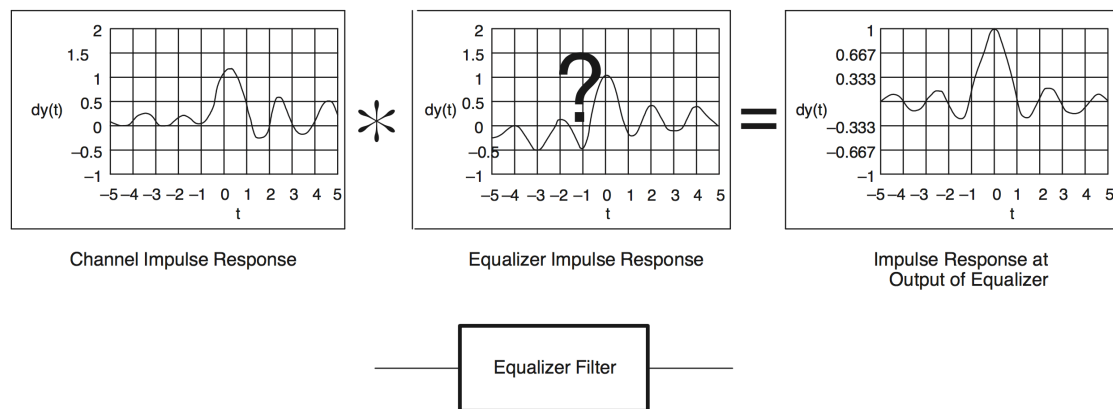


Figure 2.1.1 Equalization depicted as filtering.

### 2.2 Formulate the Equalizer Coefficients

There are various types of equalizers that will be implemented in the following sections.

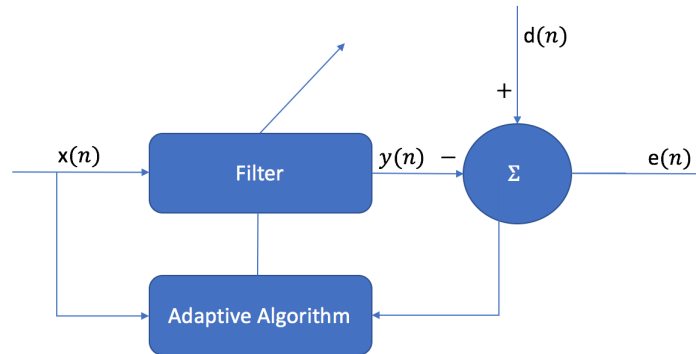
However, no matter which equalizer we choose, we always need to formulate the equalizer coefficients. According to David Smalley (1994)<sup>[1]</sup>, two main techniques are employed to solve this problem: automatic synthesis and adaptation.

In automatic synthesis method, a copy of the undistorted input signal is stored, which is the training signal. By comparing the received signal to the training signal, the error signal can be determined. Then we can use the training signal to calculate the coefficient of an inverse filter. There are two methods for finding the inverse of a filter, based on which domain does the inversion. First, the inversion is accomplished strictly in the time-domain, as is done in the LMS systems, which will be discussed in the next section. Second, we need to perform two conversions. The first one is to converse the training signal to its spectral representation to compensate for the channel response. Then, this inverse spectrum is then converted back to a time-domain representation so that filter tap weights may be extracted.

In adaptation method, the equalizer endeavors to minimize the error signal based on the difference between the output of the equalizer  $z_k$  and the estimate of the transmitted signal  $\widehat{x}_k$ , which is generated by a decision device.

The main drawback of automatic synthesis is that the transmission of a training signal must be at least as long as the filter tap length<sup>[2]</sup>. Typically, training is used to converge a filter at startup as part of the initialization overhead. Adaptation techniques can then be employed to track and compensate for minor variations in channel response on the fly.

### 2.3 Adaptive Filter Structure



**Figure 2.3.1 Typical Adaptive Filter.**

An adaptive filter is a computational device that iteratively models the relationship between the input and output signals of a filter<sup>[3]</sup>. An adaptive filter self-adjusts the filter coefficients according to an adaptive algorithm. Figure 2.3.1 shows the diagram of a typical adaptive filter where  $x(n)$  is the input signal to the filter,  $y(n)$  is the corresponding output signal,  $d(n)$  is an additional input signal to the adaptive filter and  $e(n)$  is the error signal that denotes the difference between  $d(n)$  and  $y(n)$ . The filter can be different filter types, such as finite impulse response (FIR) or infinite impulse response (IIR). An adaptive algorithm adjusts the coefficients of the linear filter iteratively to minimize the power of  $e(n)$ .

## 3. Adaptation Algorithms

In this section, three major adaptive algorithms are introduced, the Least Mean Square (LMS) algorithm, the Recursive least square (RLS) algorithm and the Constant Modulus Algorithm(CMA).

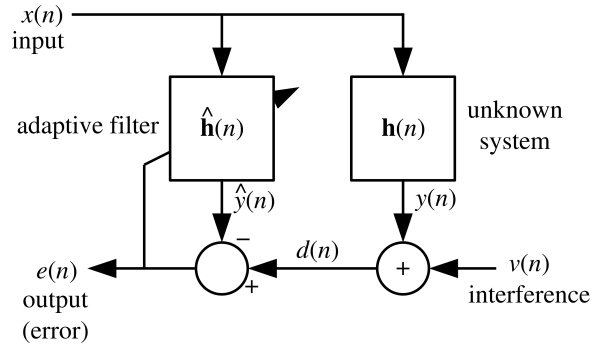


Figure 3.1 General equalizer structure.

As illustrated in figure 3.1, since the input signal is streaming in, we denote  $\bar{\mathbf{u}}(n)$  to be the filter input vector and  $\bar{\mathbf{u}}(n) = [x(n), x(n-1), \dots, x(n-N+1)]^T$ . Let us suppose the filter coefficients  $\hat{\mathbf{h}}(n) = [h_0(n), h_1(n), \dots, h_{N-1}(n)]^T$  and  $d(n)$  is the desired response at time  $n$ . The implementation procedure requires the following steps<sup>[4]</sup>.

1. Calculate the output signal  $\hat{y}(n)$  from the filter.

$$\hat{y}(n) = \bar{\mathbf{u}}^T(n) \cdot \hat{\mathbf{h}}(n)$$

2. Calculate the error signal  $e(n) = d(n) - \hat{y}(n)$

Finally, we update the weights in order to minimize the error signal. The above steps are the ones found in the below adaptive algorithms.

### 3.1 Least Mean Squares Algorithm (LMS)

The basic idea behind LSM algorithm is to approach the optimum filter weights by minimizing the cost function.

$$C(w_n) = \sum_{i=0}^n e^2(i) \quad (3.1.1)$$

The algorithm starts by assuming small weights (zero in most cases) and, at each step, by finding the gradient of the mean square error, the weights are updated <sup>[4]</sup>. So the weight update equation is:

$$W_{n+1} = W_n - \mu \nabla C(w_n) \quad (3.1.2)$$

### 3.2 Recursive Least Square Algorithm (RLS)

The RLS filter is an algorithm which recursively find the filter coefficients that minimize a weighted linear least square cost function  $C$  relating to the input signals <sup>[5]</sup>.

$$C(w_n) = \sum_{i=0}^n \lambda^{n-i} e^2(i) \quad (3.2.1)$$

where  $0 < \lambda \leq 1$  is the “forgetting factor” which gives exponentially less weight to older error samples. This contrasts with the LMS algorithm which aims to reduce the mean square error.

The weight update equation is:

$$W_{n+1} = W_n - \mu \nabla C(w_n) \quad (3.2.2)$$

Compared to most of its competitors, the RLS exhibits extremely fast convergence. However, this benefit comes at the cost of high computational complicity, and potentially poor tracking performance when the filter to be estimated changes.

### 3.3 Constant Modulus Algorithm (CMA)

The motivation of CMA is to find a filter  $w$  to restore the constant modulus property without knowing the sources. What need to mention is that CMA requires the constant modulus property to perform well, which will be discussed in detail using the simulation result.

The cost function of CMA is defined by

$$J_{CMA}(f) = E\{[e(n)]^2\} \quad (3.3.1)$$

where  $E[\cdot]$  indicates statistical expectation and  $e(n)$  is the error function of CMA, defined by

$$e(n) = |y(n)|^2 - R^2 \quad (3.3.2)$$

Where  $R^2$  is a constant which is defined by

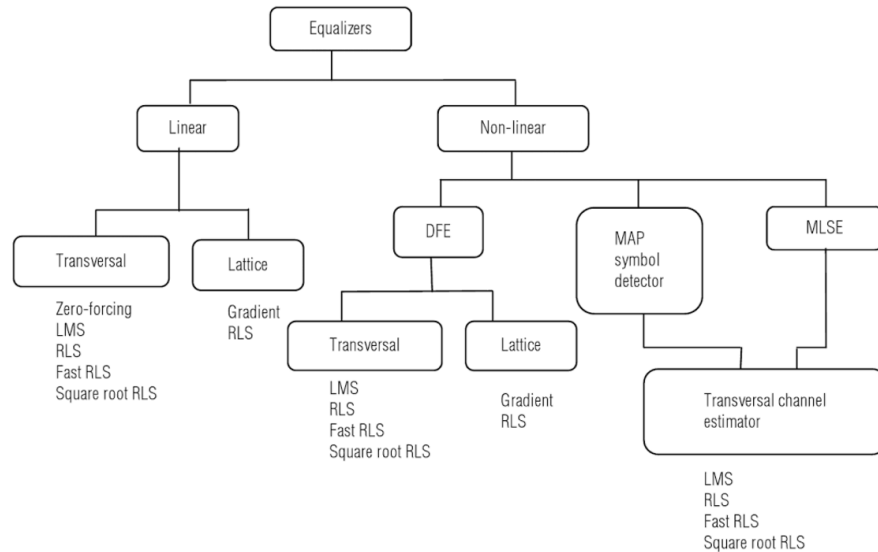
$$R^2 = E[|a(k)|^4]/E[|a(k)|^2] \quad (3.3.3)$$

The weight vector is updated by where  $\mu$  is the step size.

$$f(k+1) = f(k) - \mu x^*(k)y(k)(|y(k)|^2 - R^2) \quad (3.3.4)$$

## 4. Equalizer Classification

Figure 4.1 depicts equalization category in which equalizers can be divided into linear equalizers and non-linear equalizers. The decision feedback equalizer (DFE) and Maximum-Likelihood Sequence Estimation (MLSE) will be introduced in section 4.2.



**Figure 4.1 Categories of equalizers.**

#### 4.1 Linear Equalizer

A linear equalizer is the simplest type of equalizer. The general idea of linear equalization is that the present and the past values of the received signals are linearly weighted by the filter coefficients and summed up to produce the output. The linear equalizer can be implemented either as the simple transversal filter or as a complicated lattice filter. Linear equalizers have the potential of increasing the noise, so they are not very effective on channels having severe distortion. Linear equalizers try to compromise between the ISI and noise enhancement.

#### 4.2 Non-linear equalization

Linear equalizers have the drawback of enhancing channel noise while trying to eliminate ISI. As a result, satisfactory performance is unattainable with linear equalizers for channels having severe amplitude distortion. Linear equalization techniques are not preferred for wireless communication systems; whereas non-linear techniques, such as DFE and MLSE, are commonly

used for wireless systems. Of the non-linear techniques, the choice for use in wireless systems is usually DFE since MLSE requires an increased computational complexity and knowledge of the channel characteristics.

When channel distortion is too severe, then non-linear equalizers are used. The basic limitation of a linear equalizer such as transversal filter is the poor performance on the channel having spectral nulls. These equalizers do not perform well on channels that have deep spectral nulls in the pass band. The most commonly used non-linear equalizers are DFE and MLSE equalizer <sup>[7]</sup>.

#### **4.2.1 Decision Feedback Equalization**

A decision-feedback equalizer is a nonlinear equalizer that contains a forward filter and a feedback filter. The forward filter is similar to the linear equalizer, while the feedback filter contains a tapped delay line whose inputs are the decisions made on the equalized signal. The purpose of a DFE is to cancel intersymbol interference while minimizing noise enhancement. By contrast, noise enhancement is a typical problem with the linear equalizers described earlier.



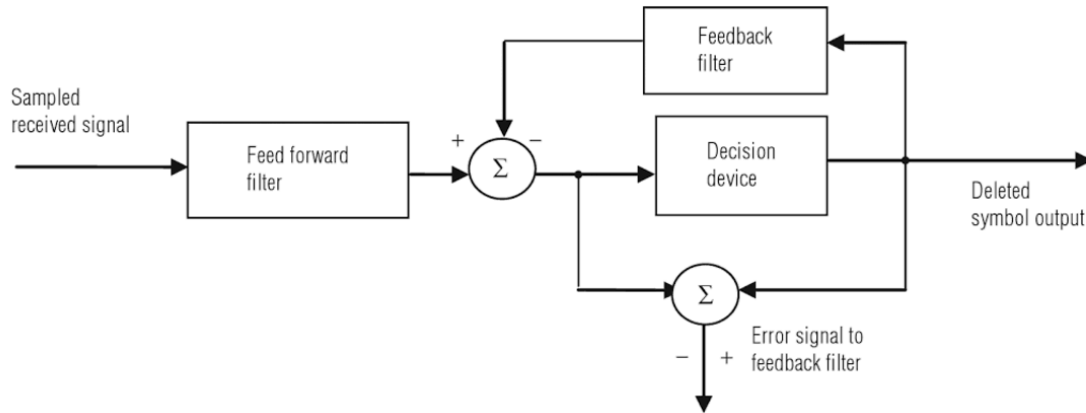


Figure 4.2.1.1 Block diagram of DFE.

Figure 4.2.1.1 depicts a typical structure of DFE comprising of two filters, referred to as the feed forward and the feedback equalizers. The received signal is the input to the forward equalizer. The input to the feedback equalizer is the stream of the detected symbols. The tap gains of this section are the estimates of the channel sampled impulse response, including the forward equalizer. Due to past examples, this section cancels the ISI. Decision directed mode means that the equalizer uses a detected version of its output signal when adapting the weights. Adaptive equalizers typically start with a training sequence and switch to decision directed mode after exhausting all symbols in the training sequence.

A delay can be taken into account by truncating data appropriately. The DFE is particularly useful for channels with severe amplitude distortions and has been widely used in wireless communications. There is an improved performance since the addition of the feedback filter allows more freedom in the selection of feed forward coefficients. The exact inverse of the channel response does not need to be synthesized in the feed forward filter. Therefore, excessive noise enhancement is avoided and sensitivity to sampler phase is decreased.

The main advantage of a DFE implementation is the feedback filter, which is additionally working to remove ISI, which operates on noiseless quantized levels, and thus its output is free of channel noise. One drawback of the DFE structure surfaces when an incorrect decision is applied to the feedback filter. The DFE output reflects this error during the next few symbols as the incorrect decision propagates through the feedback filter. Under this condition, there is a greater likelihood for more incorrect decisions following the first one, producing a condition known as error propagation.

#### 4.2.2 Maximum-Likelihood Sequence Estimation (MLSE)

The optimal equalizer, in the sense that it with the highest probability correctly detects the transmitted sequence, is the maximum-likelihood sequence estimator.

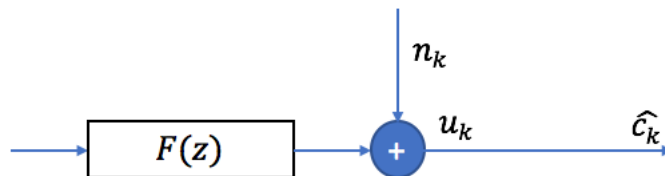


Figure 4.2.2 Block Diagram of MLSE.

We now look at the entire sequence of transmitted symbols. We need to compare the received noisy sequence  $n_k$  with all possible noise free received sequences and select the closest one. For sequences of length  $N$  bits, this requires comparison with  $2^N$  different noise free sequences.

Since we know the  $L+1$  tap impulse response  $f_j, j = 0, 1, \dots, L$ , of the channel, the receiver can, given a sequence of symbols  $\{c_m\}$ , create the corresponding “noise free signal alternative” as

$$u_m^{NF} = \sum_{j=0}^L f_j c_{m-j} \quad (4.2.2.1)$$

where NF denotes Noise Free.

The squared Euclidean distance (optimal for white Gaussian noise) to the received sequence  $\{u_m\}$  is

$$d^2(\{u_m\}, \{u_m^{NF}\}) = \sum_m |u_m - u_m^{NF}|^2 = \sum_m |u_m - \sum_{j=0}^L f_j c_{m-j}|^2 \quad (4.2.2.2)$$

The MLSE decision is then the sequence of symbols  $\{c_m\}$  minimizing this distance

$$\{\hat{c}_m\} = \underset{\{c_m\}}{\operatorname{argmin}} \sum_m |u_m - \sum_{j=0}^L f_j c_{m-j}|^2 \quad (4.2.2.3)$$

Then, we can build a trellis and use the Viterbi algorithm to efficiently calculate the best path <sup>[8]</sup>.

## 5. Implementation

### 5.1 Overall Structure

I simulated the equalization process using various types of equalizers as follows:

#	Equalizer Type	Explanation
1	CMA Equalizer	Equalize using constant modulus algorithm
2	MLSE Equalizer	Equalize using Viterbi algorithm
3	LMS Linear Equalizer	Equalize using linear equalizer that meditorsupdates weights with LMS algorithm

4	Normalized LMS Linear Equalizer	Equalize using linear equalizer that updates weights with normalized LMS algorithm
5	RLS Linear Equalizer	Equalize using linear equalizer that updates weights using RLS algorithm
6	Sign LMS Linear Equalizer	Equalize using linear equalizer that updates weights with signed LMS algorithm
7	Variable Step LMS Linear Equalizer	Equalize using linear equalizer that updates weights with variable-step-size LMS algorithm
8	LMS Decision Feedback Equalizer	Equalize using decision feedback equalizer that updates weights with LMS algorithm
9	Normalized LMS Decision Feedback Equalizer	Equalize using decision feedback equalizer that updates weights with normalized LMS algorithm
10	RLS Decision Feedback Equalizer	Equalize using decision feedback equalizer that updates weights with RLS algorithm
11	Sign LMS Decision Feedback Equalizer	Equalize using decision feedback equalizer that updates weights with signed LMS algorithm
12	Variable Step LMS Decision Feedback Equalizer	Equalize using decision feedback equalizer that updates weights with variable-step-size LMS algorithm

## 5.2 Equalizer Testbech

First, the basic scenario for every equalizer is constructed. Here is an example of a LMS

Decision Feedback Equalizer:

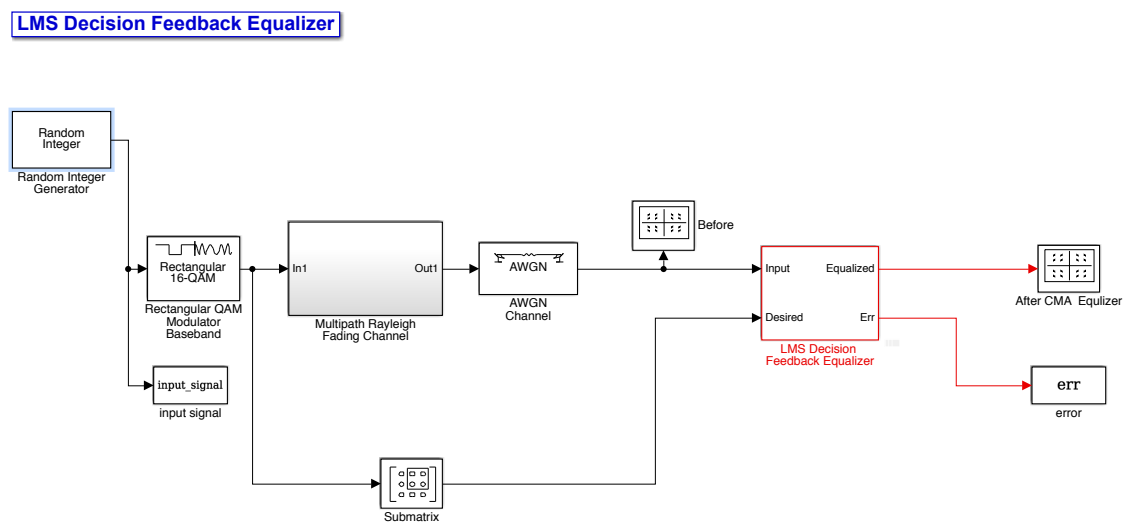


Figure 5.2.1 Block Diagram of LSM Decision Feedback Equalizer.

First, we need to initialize prerequisite parameters. In this simulation, the sampling frequency is set to  $2 \times 10^7 \text{ Hz}$ , the number of samples simulated is 10000, the maximum Doppler shift is  $3 \text{ Hz}$ . The standard constellation points are defined through rectangular QAM signal constellation. I also constructed a multipath Rayleigh fading propagation channel model to simulate channel distortion. The coefficient and delays are calculated by cost207RAx4 model. And the simulation results are as follows:

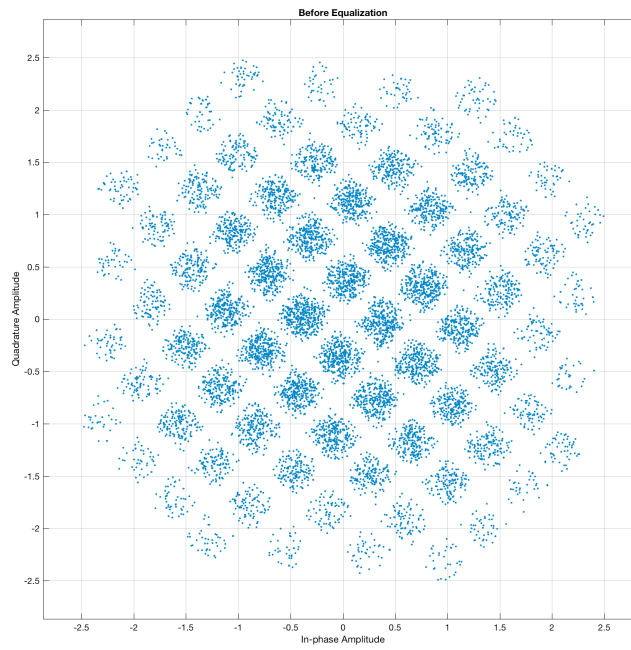


Figure 5.2.2 Points before Equalization.

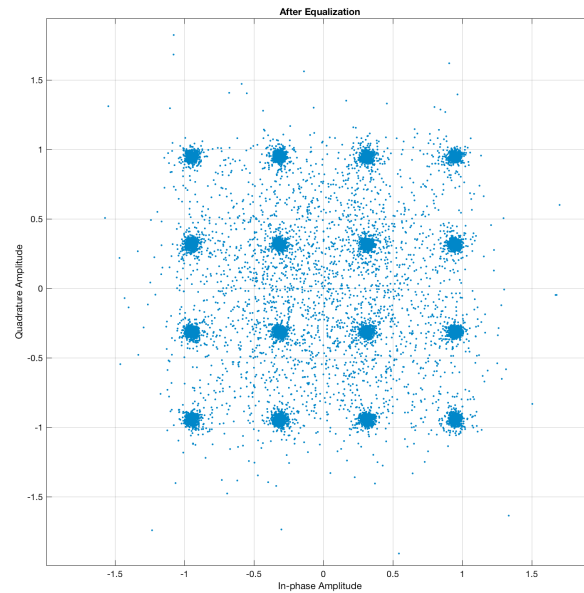
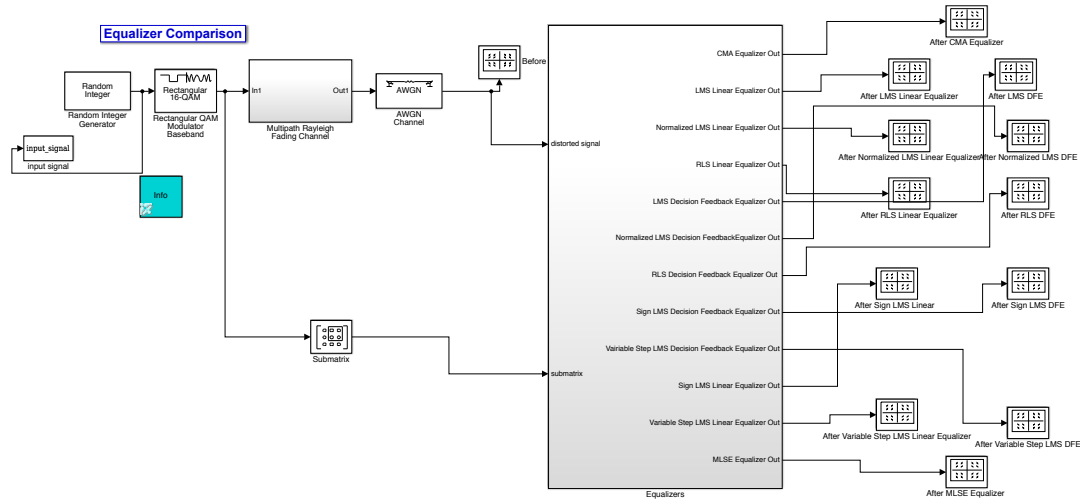


Figure 5.2.3 Points after Equalization

In figure 5.2.2, I plotted 10000 points before the equalization. Figure 5.2.3 shows these 10000 points after the equalization. It is s clear that after applying the LMS decision feedback equalizer, the signal has been equalized to our desired constellation points.

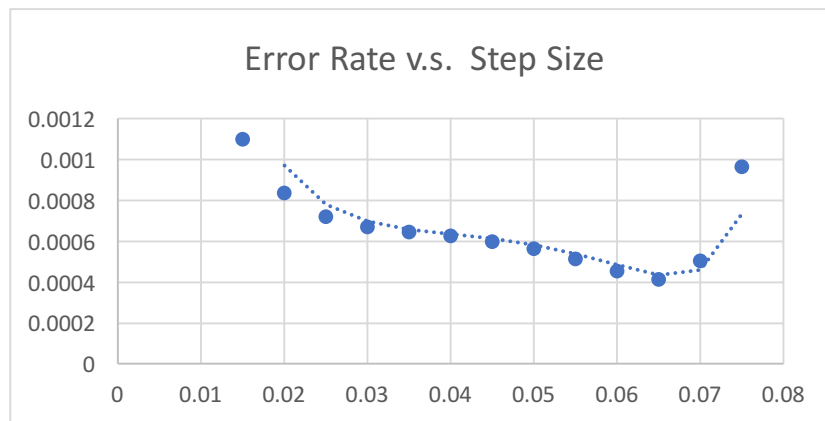
### 5.3 Equalizers Comparison

Now I combine different equalizers together. I implemented a combined framework using the same initialize parameters with section 5.2 as follows.



**Figure 5.3.1 Equalizer Comparison in Simulink.**

Then, I tried range of parameters to find parameters that gives the best performance (shown in Table 5.3.1 and Table 5.3.2). For example, given the limited run time, for the LMS Decision Feedback Equalizer, the step size is an important parameter to adjust. The step size values and corresponding error rate are as follows. So, finding the best parameters are critical for customizing the best equalizer.



**Figure 5.3.2 The effect of step size.**



Besides step size, each equalizer has its own unique parameters which are listed in the table below. So I tried a range of different parameters, for example the same input and the same multipath Rayleigh fading channel with seed size 67. The best equalizer coefficients are as follows:

**Table 5.3.1 Parameters for linear equalizers.**

Linear Equalizer Type	Number of taps	Number of samples per symbol	Signal Constellation	Step Size	Leakage factor	Initial weights	Reference Tap	Forgetting factor	Error Rate
CMA	12	1	$-0.9487 + 0.9487i$ $-0.9487 + 0.3162i$ $-0.9487 - 0.3162i$ $-0.9487 - 0.9487i$ $-0.3162 + 0.9487i$ $-0.3162 + 0.3162i$ $-0.3162 - 0.3162i$ $-0.3162 - 0.9487i$	0.001	1	$[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]$	2		0.0115
LMS	12	1		0.015	1		2		0.00033385
Normalized LMS	12	1		0.9	1		2		0.0032
RLS Linear	12	1					2	.97	0.0014
Sign LMS Linear	12	1		0.05	1		2		0.001
Variable Step LMS Linear	12	1	$0.9487 + 0.3162i$ $0.9487 - 0.3162i$ $0.9487 - 0.9487i$	Initial step size: 0.001, Increment step size: 0.001 Minimum step Size:0.001 Maximum step size: 0.01	1		2		0.0016

**Table 5.3.2 Parameters for non-linear equalizers.**

Non-linear Equalizer Type	Number of Forward Taps	Number of feedback taps	Number of samples per symbol	Signal Constellation	Reference Tap	Step Size	Leakage factor	Initial Weights	Forgetting factor	Error Rate
LMS	8	4	1	Constellation Points Constellation Points Constellation Points Constellation Points Constellation Points Constellation Points	1	0.015	1	[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]		0.000076995
Normalized LMS	8	4	1		1	0.1	1	[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]		0.0017 0.1
RLS	8	4	1		1			[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]	0.97	0.0010
Sign LMS	8	4	1		1	0.015	1	[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]		0.0056
Variable Step LMS	8	4	1		1	Initial step size: 0.01, Increment step size: 0.001 Minimum step Size:0.001 Maximum step size: 0.1	1	[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0]		0.00064991
MLSE						Initial step size: 0.01, Increment step size: 0.001 Minimum step Size:0.001 Maximum step size: 0.1	1	Channel coefficient [1.0000 0.7943 0.3162 0.1000]'		

Based on the best parameters, the equalized signal of the same input are as follows:

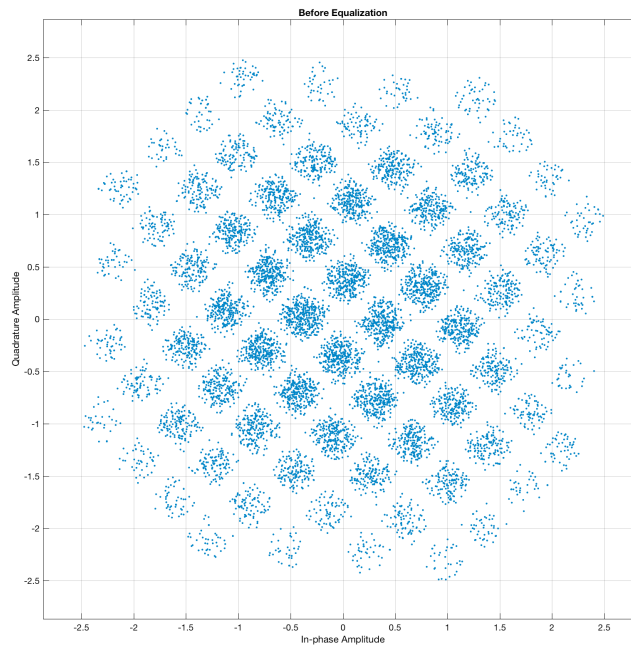
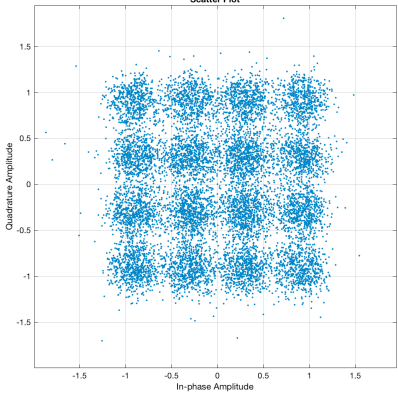
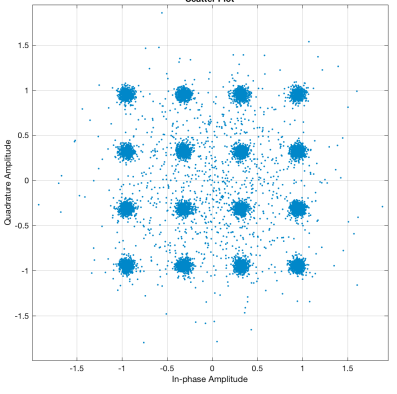
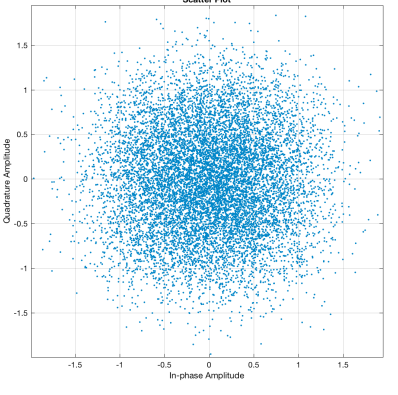
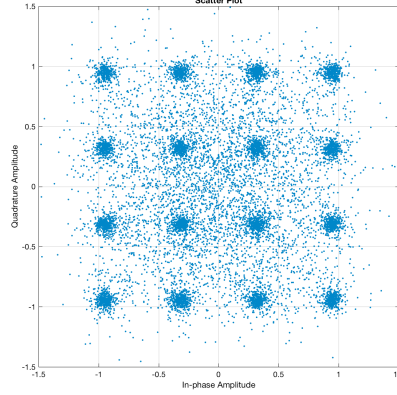
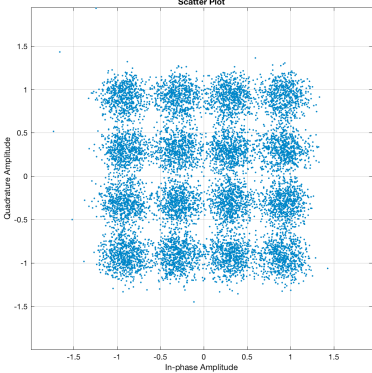
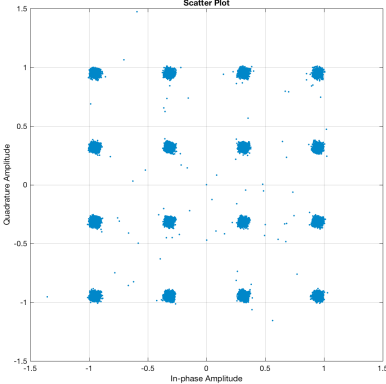
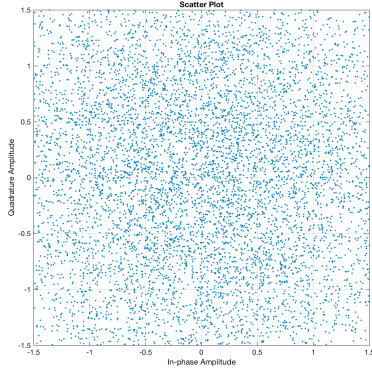
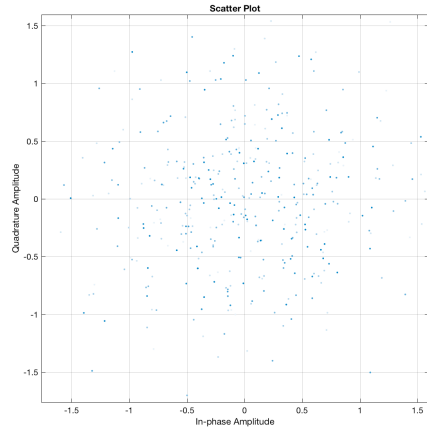
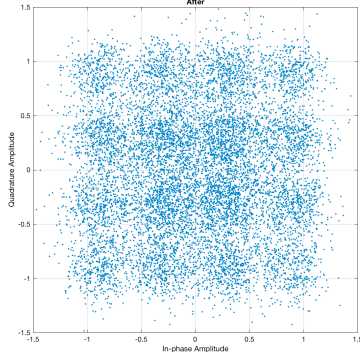
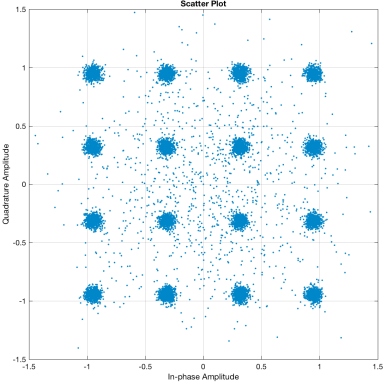
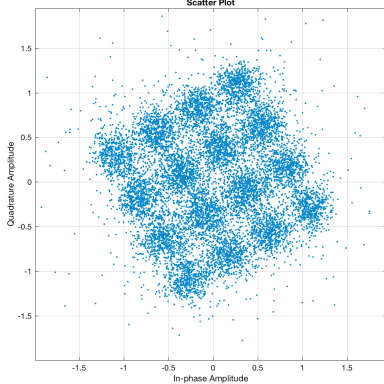
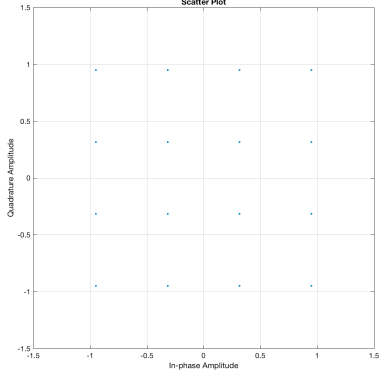


Figure 5.3.3 Points before equalization.

	Linear Equalizer	Decision Feedback Equalizer
LMS	 <p>A scatter plot titled "Scatter Plot" showing the distribution of data points in the In-phase Amplitude vs. Quadrature Amplitude plane. The points are densely clustered in a circular pattern centered at (0,0), indicating a lack of distinct signal constellation points.</p>	 <p>A scatter plot titled "Scatter Plot" showing the distribution of data points in the In-phase Amplitude vs. Quadrature Amplitude plane. The points are clearly clustered into a 4x4 grid of 16 distinct locations, indicating successful signal constellation recovery.</p>
Normalized LMS	 <p>A scatter plot titled "Scatter Plot" showing the distribution of data points in the In-phase Amplitude vs. Quadrature Amplitude plane. The points are densely clustered in a circular pattern centered at (0,0), indicating a lack of distinct signal constellation points.</p>	 <p>A scatter plot titled "Scatter Plot" showing the distribution of data points in the In-phase Amplitude vs. Quadrature Amplitude plane. The points are clearly clustered into a 4x4 grid of 16 distinct locations, indicating successful signal constellation recovery.</p>

RLS	 <p>Scatter Plot</p> <p>This scatter plot shows the distribution of data points for the RLS algorithm. The x-axis is labeled 'In-phase Amplitude' and the y-axis is labeled 'Quadrature Amplitude', both ranging from -1.5 to 1.5. The data points are concentrated into four distinct, dense clusters arranged in a square pattern, indicating successful signal separation.</p>	 <p>Scatter Plot</p> <p>This scatter plot shows the distribution of data points for the RLS algorithm. The x-axis is labeled 'In-phase Amplitude' and the y-axis is labeled 'Quadrature Amplitude', both ranging from -1.5 to 1.5. The data points are concentrated into four distinct, dense clusters arranged in a square pattern, indicating successful signal separation.</p>
Sign LMS	 <p>Scatter Plot</p> <p>This scatter plot shows the distribution of data points for the Sign LMS algorithm. The x-axis is labeled 'In-phase Amplitude' and the y-axis is labeled 'Quadrature Amplitude', both ranging from -1.5 to 1.5. The data points form a dense, uniform cloud across the entire plot area, indicating no significant clustering or separation.</p>	 <p>Scatter Plot</p> <p>This scatter plot shows the distribution of data points for the Sign LMS algorithm. The x-axis is labeled 'In-phase Amplitude' and the y-axis is labeled 'Quadrature Amplitude', both ranging from -1.5 to 1.5. The data points form a dense, uniform cloud across the entire plot area, indicating no significant clustering or separation.</p>
Variable Step LMS	 <p>After</p> <p>This scatter plot shows the distribution of data points for the Variable Step LMS algorithm. The x-axis is labeled 'In-phase Amplitude' and the y-axis is labeled 'Quadrature Amplitude', both ranging from -1.5 to 1.5. The data points form a dense, uniform cloud across the entire plot area, indicating no significant clustering or separation.</p>	 <p>Scatter Plot</p> <p>This scatter plot shows the distribution of data points for the Variable Step LMS algorithm. The x-axis is labeled 'In-phase Amplitude' and the y-axis is labeled 'Quadrature Amplitude', both ranging from -1.5 to 1.5. The data points are concentrated into four distinct, dense clusters arranged in a square pattern, indicating successful signal separation.</p>

Equalizer Type	CMA Equalizer	MLSE Equalizer
After Equalization		

**Table 5.3.3**

From the table we can observe that in general, decision feedback equalizers perform better than linear equalizers. Some equalizer do not converge at all, such as the Sign LMS Equalizer. The MLSE has the perfect performance which is not difficult to deduce from the calculating process.

We can also calculate error rate of these equalizers and the table below is the table with performances from the best to the worst.

**Table 5.3.4 Error rate of different equalizers.**

Ranking	Equalizer	Error Rate
1	MLSE	0.00000000

2	LMS Linear Equalizer	0.00033385
3	LMS Decision Feedback Equalizer	0.00048399
4	Variable Step LMS DFE	0.00064991
5	RLS DFE	0.0010101
6	RLS Linear Equalizer	0.0014011
7	Variable Step LMS Linear Equalizer	0.0016417
8	Normalized LMS DFE	0.0016912
9	Normalized LMS Linear Equalizer	0.0042491
10	Sign LMS DFE	0.0085833
11	Sign LMS Linear Equalizer	0.0099704
12	CMA Equalizer	0.011463

Based on my implementation, MLSE has the best performance because it basically calculates every possibility and selects the best one. The CMA equalizer perform worst since the 16-QAM does not have the constant modulus property. In general, DFE equalizers are better than Linear equalizer because they used more information to deduce the original signal.

## 6. Conclusion

From the previous analysis, linear equalizers suffer from noise enhancement. Decision-feedback equalizers (DFEs) use decisions on data to remove parts of the ISI, allowing the linear equalizer part to be less powerful and thereby suffer less from noise enhancement. Incorrect decisions can cause error-propagation in DFEs, since an incorrect decision may add ISI instead of removing it. Maximum-likelihood sequence estimation (MLSE) is optimal in the sense of having the lowest probability of detecting the wrong sequence. Brute-force MLSE is prohibitively complex and the Viterbi-equalizer (detector) implements the MLSE with considerably lower complexity. CMA requires the constant modulus property to perform well.



# References

- [1] David Smalley (1994). *Equalization Concepts: A Tutorial*. Retrieved from <http://dsp-book.narod.ru/spra140.pdf>
- [2] Mansoor A.Chishtie (1994). *Telecommunication Applications with the TMS320C5x DSPs*. Retrieved from <http://www.ti.com/lit/an/spra033/spra033.pdf>
- [3] Least Mean Square(LMS) Adaptive Filter (2013). Retrieved from <http://www.ni.com/example/31220/en/>
- [4] Hayes, Monson H. (1996). "9.4: Recursive Least Squares". *Statistical Digital Signal Processing and Modeling*. Wiley. p. 541. ISBN 0-471-59431-8.
- [5] Hayes, Monson H. (1996). "9.4: Recursive Least Squares". *Statistical Digital Signal Processing and Modeling*. Wiley. p. 541. ISBN 0-471-59431-8.
- [6] Wei Rao (2008). "A simple Constant Modulus Algorithm for blind equalization suitable for 16-QAM signal". ISBN 0-471-59431-8.
- [7] Gottapu Sasibhushana Rao, "Cellular Mobile Communication". ISBN: 9788131773611
- [8] Lecture slides from [http://www.eit.lth.se/fileadmin/eit/courses/etin15/slides/Lec08\\_Equalizers.pdf](http://www.eit.lth.se/fileadmin/eit/courses/etin15/slides/Lec08_Equalizers.pdf)

# Appendix

The following code initializes required parameters and need to be run before the Simulink.

```
fs = 20e6;% sampling frequency
numberOfSamples = 10000; % number of samples
fd = 3;% maximum Doppler shift
mod = comm.RectangularQAMModulator('NormalizationMethod','Average power','ModulationOrder',16,
'AveragePower',1);
constellationPoints = constellation(mod);

% Create a 802.11g channel object.
chan = stdchan(1/fs,fd,'cost207RAx4');
delays = chan.getPathDelays * fs
gainDB = chan.AvgPathGaindB;
gain = 10.^(gainDB./20);
simulation_time = numberOfSamples/fs
```

The following code compares performances of different equalizers so that it need to be run after the Simulink.

```
% 1
CMAEqualizer_err=abs(sum(CMAEqualizer(:)))/length(CMAEqualizer(:));
% 2
LMSLinearEqualizer_err=abs(sum(LMSLinearEqualizer(:)))/length(LMSLinearEqualizer(:));
% 3
NormalizedLMSLinearEqualizer_err=abs(sum(NormalizedLMSLinearEqualizer(:)))/length(NormalizedLMSLinear
Equalizer(:));
% 4
RLSLinearEqualizer_err=abs(sum(RLSLinearEqualizer(:)))/length(RLSLinearEqualizer(:));
% 5
LMSDecisionFeedbackEqualizer_err=abs(sum(LMSDecisionFeedbackEqualizer(:)))/length(LMSDecisionFeedback
Equalizer(:));
```

```

% 6
NormalizedLMSDecisionFeedbackEqualizer_err=abs(sum(NormalizedLMSDecisionFeedbackEqualizer(:))/length
(NormalizedLMSDecisionFeedbackEqualizer(:)));

% 7
RLSDecisionFeedbackEqualizer_err=abs(sum(RLSDecisionFeedbackEqualizer(:))/length(RLSDecisionFeedback
Equalizer(:)));

% 8
SignLMSDecisionFeedbackEqualizer_err=abs(sum(SignLMSDecisionFeedbackEqualizer(:))/length(SignLMSDeci
sionFeedbackEqualizer(:)));

% 9
VariableStepLMSDecisionFeedbackEqualizer_err=abs(sum(VariableStepLMSDecisionFeedbackEqualizer(:))/le
ngth(VariableStepLMSDecisionFeedbackEqualizer(:)));

% 10
VariableStepLMSLinearEqualizer_err =
abs(sum(VariableStepLMSLinearEqualizer(:))/length(VariableStepLMSLinearEqualizer(:)));

% 11
SignLMSLinearEqualizer_err = abs(sum(SignLMSLinearEqualizer(:))/length(SignLMSLinearEqualizer(:)));

errors = [CMAEqualizer_err, LMSLinearEqualizer_err, NormalizedLMSLinearEqualizer_err,
RLSLinearEqualizer_err,
LMSDecisionFeedbackEqualizer_err, NormalizedLMSDecisionFeedbackEqualizer_err,
RLSDecisionFeedbackEqualizer_err, SignLMSDecisionFeedbackEqualizer_err,
VariableStepLMSDecisionFeedbackEqualizer_err,
VariableStepLMSLinearEqualizer_err, SignLMSLinearEqualizer_err ];
sorted = sort(errors);

% MLSE Equalizer    Equalize using Viterbi algorithm
% Sign LMS Linear Equalizer Equalize using linear equalizer that updates weights with signed LMS
algorithm
% Variable Step LMS Linear Equalizer
for i = 1: 11
    for j = 1 :11
        if (sorted(i) == errors(j))
            if j == 1
                disp('CMAEqualizer: ' + CMAEqualizer_err)

```

```
end

if j == 2
    disp("LMSLinearEqualizer: " + LMSLinearEqualizer_err)
end

if j == 3
    disp("NormalizedLMSLinearEqualizer: " + NormalizedLMSLinearEqualizer_err)
end

if j == 4
    disp("RLSLinearEqualizer: " + RLSLinearEqualizer_err)
end

if j == 5
    disp("LMSDecisionFeedbackEqualizer: " + LMSDecisionFeedbackEqualizer_err)
end

if j == 6
    disp("NormalizedLMSDecisionFeedbackEqualizer: " +
NormalizedLMSDecisionFeedbackEqualizer_err)
end

if j == 7
    disp("RLSDecisionFeedbackEqualizer: " + RLSDecisionFeedbackEqualizer_err)
end

if j == 8
    disp("SignLMSDecisionFeedbackEqualizer: " + SignLMSDecisionFeedbackEqualizer_err)
end

if j == 9
    disp("VariableStepLMSDecisionFeedbackEqualizer: " +
VariableStepLMSDecisionFeedbackEqualizer_err)
end

if j == 10
    disp("VariableStepLMSLinearEqualizer: " + VariableStepLMSLinearEqualizer_err)
end

if j == 11
    disp("SignLMSLinearEqualizer: "+ SignLMSLinearEqualizer_err)
end

end

end

end
```

The following code is used to generates figure 1.1.2.

```
% Set up parameters and signals.
M = 4; % Alphabet size for modulation
msg = randi([0 M-1],1500,1); % Random message
hMod = comm.QPSKModulator('PhaseOffset',0);
modmsg = step(hMod,msg); % Modulate using QPSK.
trainlen = 500; % Length of training sequence
chan = [.986; .845; .237; .123+.31i]; % Channel coefficients
filtmsg = filter(chan,1,modmsg); % Introduce channel distortion.

% Equalize the received signal.
eq1 = lineareq(8, lms(0.01)); % Create an equalizer object.
eq1.SigConst = step(hMod,(0:M-1)'); % Set signal constellation.
[symbolest,yd] = equalize(eq1,filtmsg,modmsg(1:trainlen)); % Equalize.

% Plot signals.
h = scatterplot(filtmsg,1,trainlen,'bx'); hold on;
scatterplot(symbolest,1,trainlen,'g.',h);
scatterplot(eq1.SigConst,1,0,'k*',h);
legend('Filtered signal','Equalized signal',...
      'Ideal signal constellation');
hold off;

% Compute error rates with and without equalization.
hDemod = comm.QPSKDemodulator('PhaseOffset',0);
demodmsg_noeq = step(hDemod,filtmsg); % Demodulate unequalized signal.
demodmsg = step(hDemod,yd); % Demodulate detected signal from equalizer.
hErrorCalc = comm.ErrorRate; % ErrorRate calculator
ser_noEq = step(hErrorCalc, ...
               msg(trainlen+1:end), demodmsg_noeq(trainlen+1:end));
reset(hErrorCalc)
ser_Eq = step(hErrorCalc, msg(trainlen+1:end),demodmsg(trainlen+1:end));
disp('Symbol error rates with and without equalizer:')
disp([ser_Eq(1) ser_noEq(1)])
```