Complextoreal.com

Intuitive Guide to Principles of Communications
Charan Langton

Analog Filters

Filters play important role in communications. Filters keep the signal from splashing energy into adjacent channels and conversely protect the user band from unwanted signals and noise from adjacent channels. Filters are also used to shape pulses that represent the baseband symbols.

Here we show a chain of filters that might be used in a multicarrier signal.

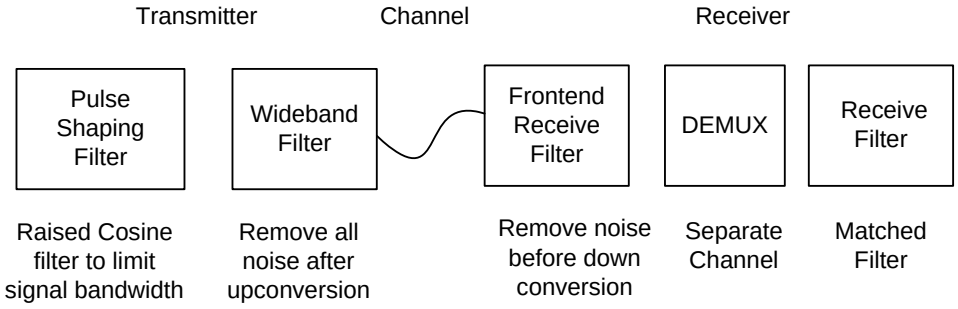


Figure 1- Uses for filters in a communications chain

Important filter characteristics

The first filter used Figure 1 is often called the pulse shaping filter because it converts discrete signals into analog symbol shapes that can be transmitted. Not all shapes can be transmitted efficiently so pulse shaping is very important. After pulse shaping, the signal is modulated and up-converted to a carrier frequency. The signal is them amplified using a High Power Amplifier. At this point, a filter

maybe used to limit the spreading from the HPA. At the receiver there would be a frontend receiver tuned to the carrier frequency. It removes noise picked up by signal through the channel. A de-multiplexer may be used to separate out the user channel, and then a receive or matched filter is used for demodulation.

Whether these filters are realized as analog or digital depends on the application. At rf frequencies, the analog filters are cheaper and lighter and used often.

Frequency Response and Group Delay

The purpose of a filter is to block out the undesirable signals and at the same time keep the pass-band signal as undistorted as possible. We characterize filters by their Frequency response, also called Bode Plots. The frequency response of an analog filter T(s) can be described as a ratio of two voltages E_L and E_S or polynomials in s-domain, N(s) and D(s), where $s = jw$ and where ω is radians per second.

Figure 2 - An LC filter

$$H(s) = \frac{E_L}{E_s} = \frac{N(s)}{D(s)} \tag{0.1}$$

Writing the transfer function for the filter in Fig. 6.4.1

$$H(s) = \frac{1}{s^3 + 2s^2 + 2s + 1} \tag{0.2}$$

If we evaluate this expression for various ω (set $s = jw$), we write the frequency and the phase response as

$$H(s) = \frac{1}{1 - 2\omega^2 + j(2\omega - \omega^3)}, \quad \phi = \tan^{-1}(\omega) \tag{0.3}$$

The order of this filter is given by the highest order of ω, in this case 3. This is a lowpass filter of fairly shallow rejection. The transfer function of Equation (0.30) has three poles and no zeros. Reference to poles and zeros comes up often in filters. Their purpose is to provide a short cut for determining if a filter that has been synthesized is realizable or not. The roots of the numerator are called zeros and those of the denominator, poles. If these roots fail any of the following rules, then filter cannot be built. These rules are: [8]

1. The poles and zeros must occur in pairs which are complex conjugates of each other.
2. The poles and zeros on the real axis do not have to be in pairs.
3. Poles must be restricted to the left half of the complex-frequency plane.

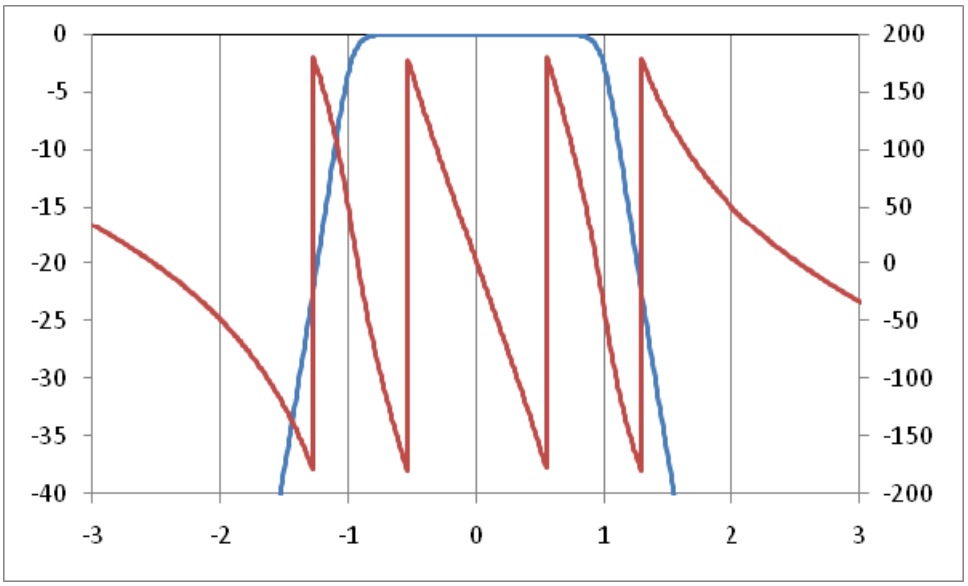


Figure 3 – Frequency and phase response of a general filter.

Phase Response

A passive filter provides no gain to the signal. In fact, most often there is insertion loss and overall power loss in the band of interest. Conversely an active filter is

one that has an amplification function built into it. The filter of Figure 3 is a passive filter as there is no gain in the pass-band. Its phase response seems to jump up and down rapidly, but that's for graphing convenience since the scale of observation is limited to 360 degrees. We see that in the first segment starting at f = -1, phase goes from 50 degrees to -180 degrees and then a full 360 degrees in the next segment and then again from 180 to -40 degrees, adding these up and subtracting 2x360 degrees, we get 90 degrees. Which is exactly the phase difference between a sine wave of frequency -1 Hz and +1 Hz. For this filter, which happens to be a Butterworth filter of order 9, the phase is linear in the pass-band, from -1 Hz to +1 Hz which is a characteristic of Butterworth filters. A great deal of importance is attached to the phase response of the filter since any change in linearity causes signal distortion.

Impulse Response

The Impulse response of a filter is very useful characteristic. Akin to striking a bell to hear the quality of its ring, an impulse response similarly pings the filter with an impulse. Of course, response to a single impulse won't immediately tell you how the filter will respond to a real signal, it does tell us something useful. Since the system response out of a linear filter is itself linear, the response of the filter to a signal is just linear additions of the impulse response for each incoming signal pulse, scaled by its magnitude. For the unit pulse (a delta function) we can say that acting on system H, it produces a response h(n).

$$\delta(n) \xrightarrow{H} h(n)$$

or in time domain, a pulse produces several outputs.

$$\{1,0,0,\ldots 0\} \xrightarrow{H} \{h_0, h_1, h_2, h_3, \ldots\}$$

The time-invariant property tells us that

$$\delta(n-T) \xrightarrow{H} h(n-T)$$

For any delay equal to T, the response is delayed by the same amount. To write the sum of n impulses, we get

$$\delta(n)+\delta(n-1)+\delta(n-2)+\ldots \xrightarrow{H} h(n)+h(n-1)+h(n-2)+\ldots$$

This result in a weighted sum of

$$x(o)\delta(n)+x(1)\delta(n-1)+x(2)\delta(n-2)+\ldots$$
$$\xrightarrow{H}$$
$$x(0)h(n)+x(1)h(n-1)+x(2)h(n-2)+\ldots$$

Which in a more formal way can be written as the familiar discrete-time convolution of the input signal x(n) with the filter tap-weight h(n) to produce the output signal (n).

$$y(n)=\sum_{m} x(m)h(n-m) \tag{0.4}$$

Group Delay of a Filter

The phase response as shown in Figure 3 of a filter is difficult to make sense because of the large excursions over the bandwidth. An alternate metric of interest is the Group Delay. Group delay is a time parameter give in units of time delay. The phase difference is converted to a time delay over the frequencies in the occupied band. The group delay is given as the slope of the phase curve vs. frequency. Group delay is used far more as a measure of filter phase response than pure phase in degrees. A plot of delay vs. signal frequency gives an indication of how the relative frequencies of the signal are being delayed by the filter. The objective is to get as flat a group delay as possible which implies linear behavior.

The expression for group delay is given by ratio of change in phase over change in frequency. Given the phase response of the filter, it is easy to calculate the group delay by this relationship.

$$T_{delay}=-\frac{\partial\phi}{\partial\omega}=-\frac{\nabla\phi}{\nabla\omega} \tag{0.5}$$

From the phase response in Figure 3, we can incrementally compute the slope to get the group delay response. Despite the fact that the phase response looks like straight lines, the group delay plot is more illuminating in showing the important behavior. Group delay is used to compare filters, where such comparison of phase is a difficult task. The effect of group delay on the signal can be explained by this metaphor. Imagine a train where the front is moving faster than the back. The train must stretch and the result is distortion. Similarly group delay tells us how much distortion the filter has introduced in the signal in time domain. If the group delay is much longer than a symbol time, then we have a problem. However if the maximum group delay difference is much smaller than the symbol time (at least one order of magnitude less) the distortion would be small. Wideband signals are much more prone to group delay distortion than narrowband. This is simply because a narrowband signal experiences much less difference in delays between its maximum and minimum frequency. The group delay shows up a function of the linearity of the filter. The filters that offer a linear phase response are generally characterized as linear filters.

Ultimately, we use group delay as a measure of the non-linearity of the system and its potential to distort the signal. There would always be delay through a system, so the absolute value is not very important, just the range of delay over the desired bandwidth.

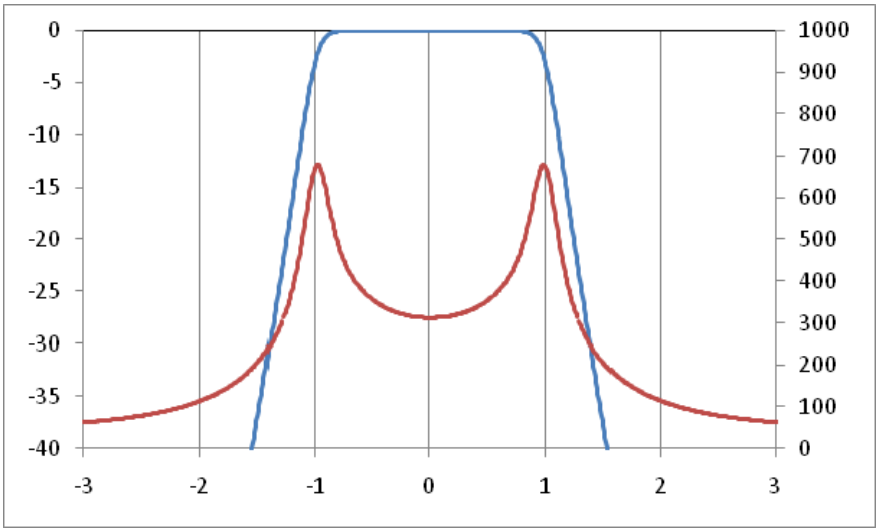


Figure 4 – Frequency and phase response of a filter.

Figure 4 takes the phase of filter in Figure 3 and plots it as group delay. This depiction of group delay has appeal. A perfectly flat group delay tells us that all frequencies in the signal arrive with delta change in time with some given delay through the path. (The front of the train and the back of the train arrive at the station at the same time.) In the center the group delay is somewhat flat. We see the group delay increasing rapidly at the edges of the pass-band. This says that the distortion would be larger for the frequencies at the edge. The concept of group delay can be applied to any device that has a frequency response. Nearly all devices even those that are frequency independent exhibit some group delay. In most cases, the system group delay as opposed to the filter group delay would be measured and compared against some maximum acceptable design specification.

Definition of Bandwidth

Akin to water flowing through a pipe, symbol rate has similar relationship to bandwidth. Larger bandwidth allows larger data rates. But “Bandwidth” is a term fraught with confusion. It is clear what we mean by data rate but what is bandwidth? There are several different ways of defining bandwidth and all lead to different answers. Whenever bandwidth is specified, it is sensible to ask what the specification criteria is.

First thing we need to know is that Bandwidth contains only the positive frequencies of a signal. All frequency specifications start measuring at 0 Hz. This is why lowpass bandwidth is one-half of bandpass, although absolutely nothing about the signal changes as it goes from low-pass to band-pass. It has just shifted to a higher center frequency and all components of the signal have moved into the positive frequency range.

Here are some ways in which bandwidth is specified.[Couch]

1. Absolute Bandwidth – Specifications of bandwidths by regulatory bodies are absolute bandwidths. For example if the bandwidth is given form 12.7 to 13.7 GHz, then this is an absolute bandwidth.
2. 3-dB bandwidth – If the edge frequency is f1, then it is assumed that the magnitude at this point has attenuated to exactly one-half the peak value. However, not all filters have a well-defined 3-dB bandwidth. Chebychev filters that have a ripple in the pass-band are one such example.

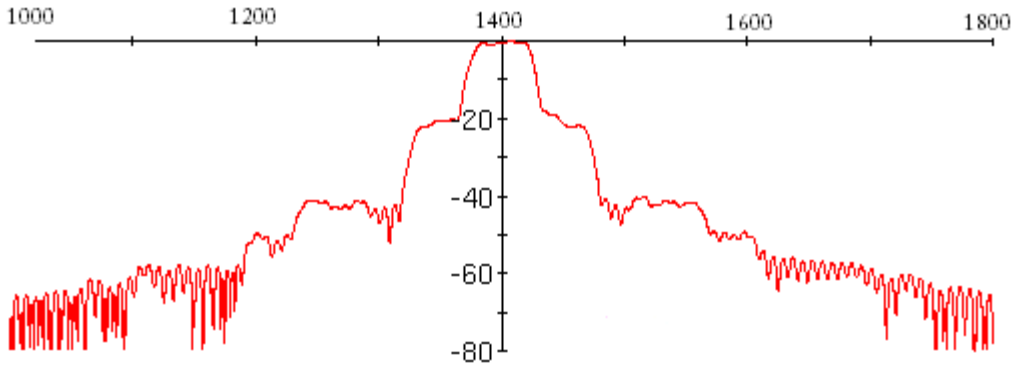


Figure 5 – Definition of Bandwidth

Absolute bandwidth: 28 Hz
3 dB bandwidth: 23.50
98% power bandwidth: 29.50 Hz
99% power bandwidth: 42.75 Hz.
The null-to-null bandwidth is 38.5 Hz

3. **98% and 99% power bandwidths** - This is the bandwidth which contains respective amount of total power of the signal.
4. **Null-to-null bandwidth** – This bandwidth definition is used more commonly in antenna design and specifies the range (f1 – f2), where both edges fall in some predefined level nulls.
5. **Occupied Bandwidth** – This is usually a regulatory specification in terms of a power and attenuation mask, the purpose of which is to control the energy being transmitted out of the band. Figure 6 shows a FCC specified mask for satellite communications. The frequency is given as a percent of the total bandwidth.
6. **Equivalent Noise Bandwidth** – If all power of the signal were to be confined to a rectangular ideal shape, we get a bandwidth measure that is called the Equivalent Noise Bandwidth. The alternate way to specify this is by the term Time-Bandwidth product. For a root-raised cosine, the noise bandwidth is 0.5 and for a Butterworth filter it is approximately 0.53.

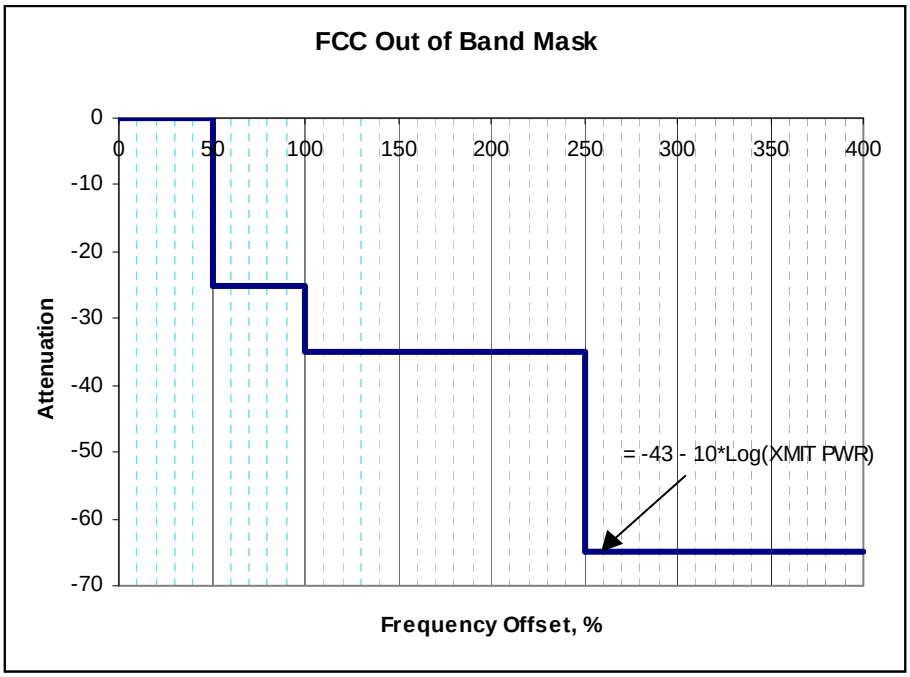


Figure 6 FCC Emissions Mask

Analog filter types

There are four main analog filters and all are used extensively in designs ,particularly at RF frequencies. Analog filters are usually compact, and often inexpensive. The primary principle of an analog filter is that it is based on a RLC circuit, and is inherently a non-linear device but can be made to behave linearly in a particular range. The filter types are: Butterworth, Chebychev, Elliptic, and Bessel. There are of course hybrids that combine best parts of each but they are usually application specific.

Butterworth

Most commonly used analog filter is the Butterworth Filter. It has a set of transfer functions that result in the flattest possible behavior in the pass-band. Butterworth filters are all-pole filters and fall on a circle (limited to the left half) of unit radius so in the pass-band they have nearly linear phase response as in Figure 7. The equation for a Butterworth filter of order 3 is given in Equation (0.33) The frequency response of these filters is consistent as the order goes up [4]. The bandwidth is defined as that point where the attenuation equals 3 dB. The general equation of a Butterworth filter is given by

$$A_{dB} = 10\log\left[1+\left(\frac{\omega}{\omega_c}\right)^{2n}\right] \tag{0.6}$$

Where ω_c the 3 dB cutoff frequency and n is is the order of the filter. A general analog is defined by

$$A_{dB} = 10\log(1+\Omega^{2n)}) \tag{0.7}$$

Where the Ω is given by the Table 6.3. 1. And given by ω_x/ω_c is the ratio of the frequency of interest and the 3 dB cut-of frequency.

Table 1

Filter Type	Ω
Low-pass	ω_x / ω_c
High-pass	ω_c / ω_x
Bandpass	BW_x / BW_{3-dB}
Band-reject	BW_{3-dB} / BW_x

Example: Calculate and plot the frequency response of a Butterworth filter of order 5 , 7 and 9.

Using equation (0.34), we get the following graph.

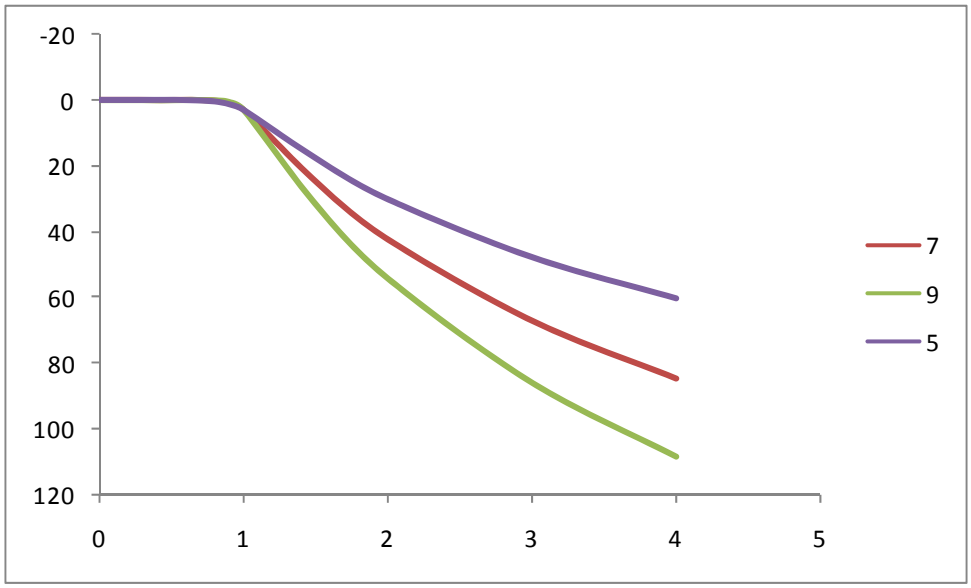


Figure 7 - Butterworth Filter Frequency Response for order 5, 7 and 9.

Butterworth filters are used widely in frontend of low power receivers. There is very little distortion in the pass-band and they are considered most benign of all analog filters. However the out of band attenuation is not as good as that offered by other filters which is often a matter of tradeoff between pass-band behavior and the effect on and from adjacent channels.

Chebychev Filters

There are two types of Chebychev filters. Both types have a ripple either in the pass-band or the stop-band. Type I filter, the one used most often has a ripple in the pass-band which is of course not a desired behavior but it rolls-off much faster than all the others. The ripple of course can be made very small by design. Type II does not have a ripple in the pass-band but does not roll off as rapidly as type I. Both are used depending on which band is of concern The Chebychev (There are numerous way to spell this name, for a story about the correct spelling see book [6] by Paul Davies.) filters offer a steeper stop-band response but at the expense of introducing the ripple.

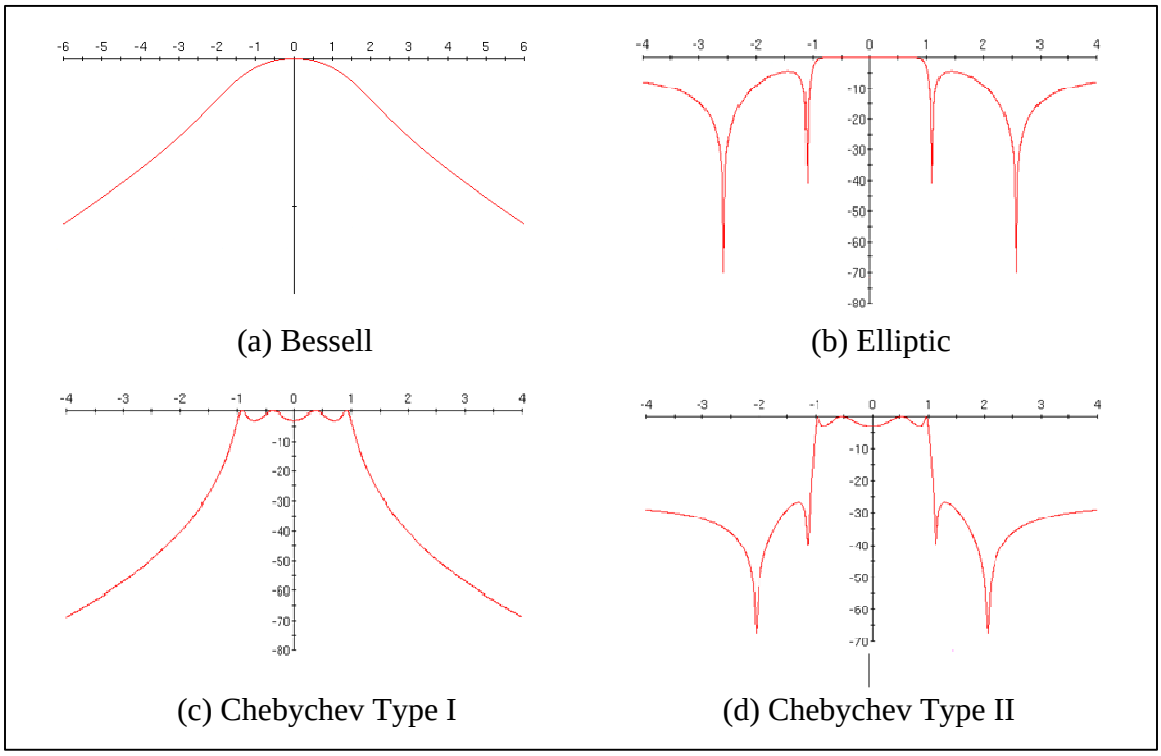
Figure 8 Chebychev Filter Types

Although the magnitude response of the Chebychev is very attractive, its group delay behavior is not so. In Figure 9 we present the group delay of this filter for several filter orders. Beyond order 7, the group delay becomes too large for practical use.

Elliptic

Both Butterworth and Chebychev are all-pole filters and as such their rejection does not roll off rapidly and can only be zero at the far end of the stop band.

Elliptic filters on the other hand have zeros in the stopband and for this reason this filter has the fastest roll-off of all analog filters. As Chebychev has ripple in either the passband or stopband, the elliptic has ripple in both. It is of course related to both Chebychev and Butterworth filters. Setting the ripple to zero in stopband causes Elliptic filter to become a Chebychev and suppression of ripple in both bands causes it to degenerate to a Butterworth filter. This filter has equiripple behavior in both the passband and the stopband. The ripple can be changed in both bands independently by design. A form of elliptic filter that limits the ripple in the stopband is called quasi-elliptic filter. The improved performance from doing that comes with sidelobes but they are usually quite far down. For applications where large and fast rejection is required, Elliptic filters stand out. They are used in satellite communications for both as input multiplexer and output multiplexers after HPA amplification. In Figure 9, we see the comparison of the Elliptic and see how much better is its stopband response.



(a) Bessell

(b) Elliptic

(c) Chebychev Type I

(d) Chebychev Type II

Figure 9 - The frequency response of (a) Bessell, (b) Elliptic, (c) Chebychev Type I and (d) Chebychev Type II analog filters.

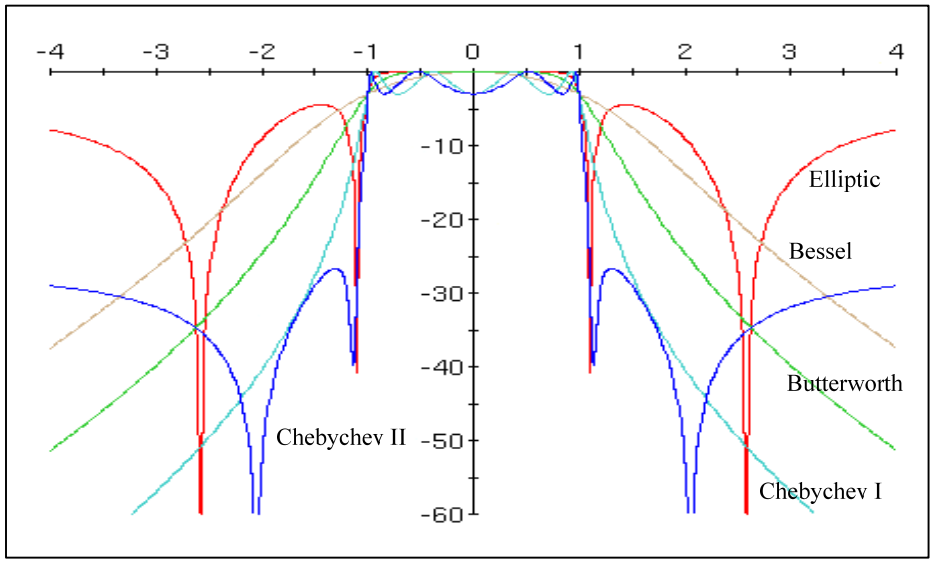


Figure 10- In-band response comparison

Digital Filters

Digital filters can provide the same frequency characteristics as analog filters and can do even better in phase response. They are a design created from shift-register networks and can be built with memory and Arithmetic units. They offer many advantages over analog filters: [5]

1. Most digital structures offer unconditional stability.
2. The digital shift registers are easy to control and coefficient values can be stored and easily changed for applications such as adaptive filtering.
3. They can be manufactured from ASICs.
4. Digital filters can operate over wide frequencies and dynamic range.
5. We can design perfectly linear phase response.
6. Digital filters don't suffer from age related degradations.

Finite Impulse response (FIR) Filters

Digital filters can be classified in two forms: FIR and IIR, depending on the length of their impulse response, finite or infinite. Both of these forms are

created with delay/memory/shift registers. The FIR has an impulse response that extends over a finite number of terms only such that we can write its impulse response as

$$\{h_0 \quad h_1 \quad \cdots \quad h_M\}$$

Where M is the order of the filter. The length of the impulse response is equal to M+1. In Figure 11 is shown one such FIR filter. The signal is tapped off after each delay element and multiplied by a constant, called filter coefficient, filter weight or tap-weight. All scaled components of the signal are added together to produce the output.

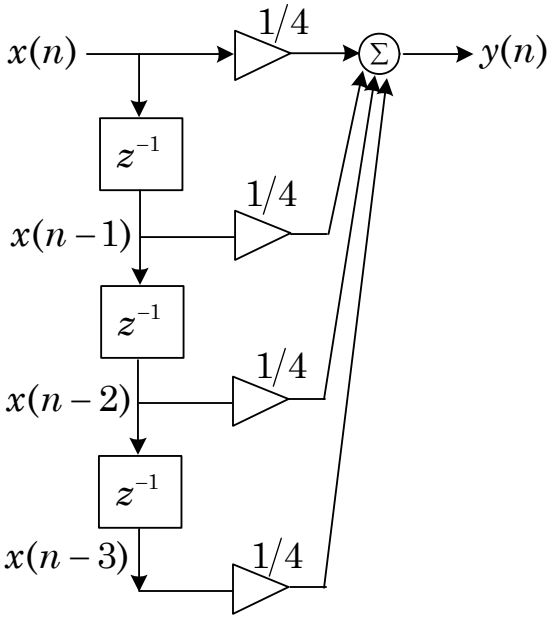


Figure 11- A Moving Average FIR filter

In Figure 11 we see a very simple FIR filter. All tap-weights are equal to 0.25 and this is an M = 4 order filter. We determine the output by

$$y(n) = \frac{1}{4}(x) + \frac{1}{4}(x-1) + \frac{1}{4}(x-2) + \frac{1}{4}(x-3).$$

If the first inputs were 1, 2, 6, 4, 3, 2, our first few outputs would be

$$(1\times 0.25+0\times 0.25+0\times 0.25+0\times 0.25)=0.2$$
$$(2\times 0.25+1\times 0.25+0\times 0.25+0\times 0.25)=0.75$$
$$(6\times 0.25+2\times 0.25+1\times 0.25+0\times 0.25)=0.983$$
$$(4\times 0.25+6\times 0.25+2\times 0.25+1\times 0.25)=1.983$$
$$(3\times 0.25+4\times 0.25+6\times 0.25+2\times 0.25)=$$
$$(2\times 0.25+3\times 0.25+4\times 0.25+6\times 0.25)=$$

Let's determine the impulse response of this filter using the same procedure as above but with a signal equal to 1,0,0,0,0. What we get for first five times is 0.25, 0.25, 0.25, 0.25. This is the impulse response of this FIR filter and it is exactly equal to the filter tap-weights.

$$(1\times 0.25+0\times 0.25+0\times 0.25+0\times 0.25)=0.25$$
$$(0\times 0.25+1\times 0.25+0\times 0.25+0\times 0.25)=0.25$$
$$(0\times 0.25+0\times 0.25+1\times 0.25+0\times 0.25)=0.25$$
$$(0\times 0.25+0\times 0.25+0\times 0.25+1\times 0.25)=0.25$$

This fact is true for all FIR filters, no matter how many tap-weight it has long as it has this forward-feeding **s**tructure. If we trace the path of the impulse through the FIR filters, its impulse response will always be equal to its tap-weights or coefficients. The tap-weights imply a frequency response. Changing the tap-weights means changing the filter response. If we know what type of frequency response we want, we can design it by finding coefficients that provide it. This is the process of the design of a FIR filter; going from a given frequency response to determining coefficients.

To generalize the expression for an arbitrary FIR filter of order M, we can write the equation of a FIR filter as

$$y(n)=\sum_{k=0}^{M}h(k)x(n-k) \qquad (0.8)$$

The filter has an easy to understand structure. This type of design is called a feed-forward structure. It has the wonderful quality of always being stable as it has no denominator and can never be singular.

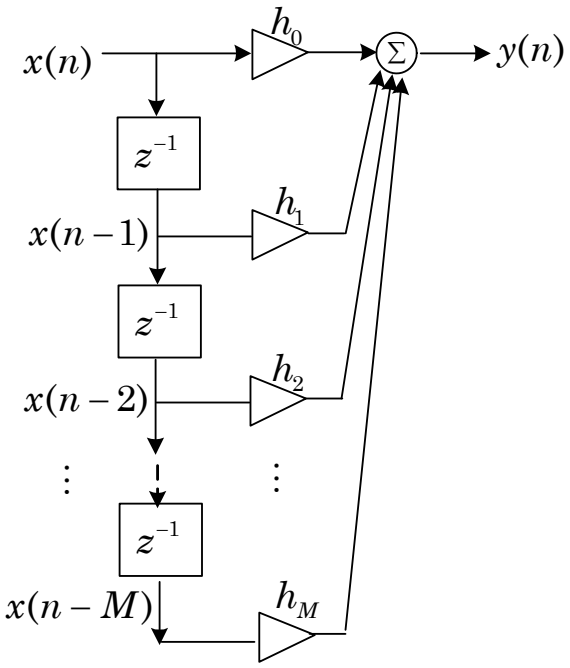


Figure 12- Generic FIR filter of M+1 taps

Where M+1 equals the numbers of taps. This is the basic structure of a FIR filter, also called transversal, or tap-delay line filter. That's all there is to a FIR filter. The output of the filter, Equation (0.35) is a convolution of the coefficients and the incoming signal values. We only show a scalar version in this figure but of course, the coefficients and the signal can both be complex.

FIR filters find their uses in many fields other than communications. One example is the technical analysis done on stock prices. Moving average, exponential moving average, MACD, RSI, are all example of FIR filters applied to a sequence of data either in order to shape it, or to extract useful information such as trends or intelligent information free of instantaneous noise.

Condition for Phase Linearity

One of the advantageous qualities of FIR filters is that they exhibit linear phase. A kth order FIR has a frequency response as given by (0.35). We say it has a linear phase if the phase as a function of the frequency equals the phase change through the filter at the center tap plus a constant.

$$\begin{aligned} &\theta(\omega) = -h(\omega) + p \\ &h = (M-1)/2 \quad and \quad p = \pm\pi/2 \end{aligned} \tag{0.9}$$

Group delay which is a derivative of the phase is given as

$$T_{gd} = -d(\theta(\omega))/d\omega = (M-1)/2 \tag{0.10}$$

That says that group delay is a constant and a function of the number of taps. The other condition that is implied is that the impulse response of the filter must be symmetric, otherwise equation (0.37) cannot hold:

$$h(n) = \pm h(M-1-n) \tag{0.11}$$

In describing adaptive filters, we will show that although we can design the taps to be symmetrical, in order to equalize a channel, we need to come up with tap-weights that may not be symmetrical because of the non-linear nature of the incoming signal. The property of being able to change the impulse response by changing the tap-weights is remarkably powerful and versatile.

Infinite Impulse Response (IIR) Filter

The difference between a FIR filter and an IIR filter is that IIR filter can produce an infinite response. We can write its impulse response as

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) \tag{0.12}$$

Of course, the infinite number of terms is a problem. In a sub-class of IIR filters, the requirements of infinite number of taps is gotten around by selecting tap-weights that are related to each other, so that the summation can be estimated as the sum of an infinite series. [3]

Compare the FIR structure with only forward going taps to the structure of an IIR filter. This structure adds an additional section (on the right in Figure 13) that takes the past values of the output signal d(n) and feeds them back into computing the current value. The second section on the right is called feed-backward. [See Lyons 1 for more on FIR, IIR].

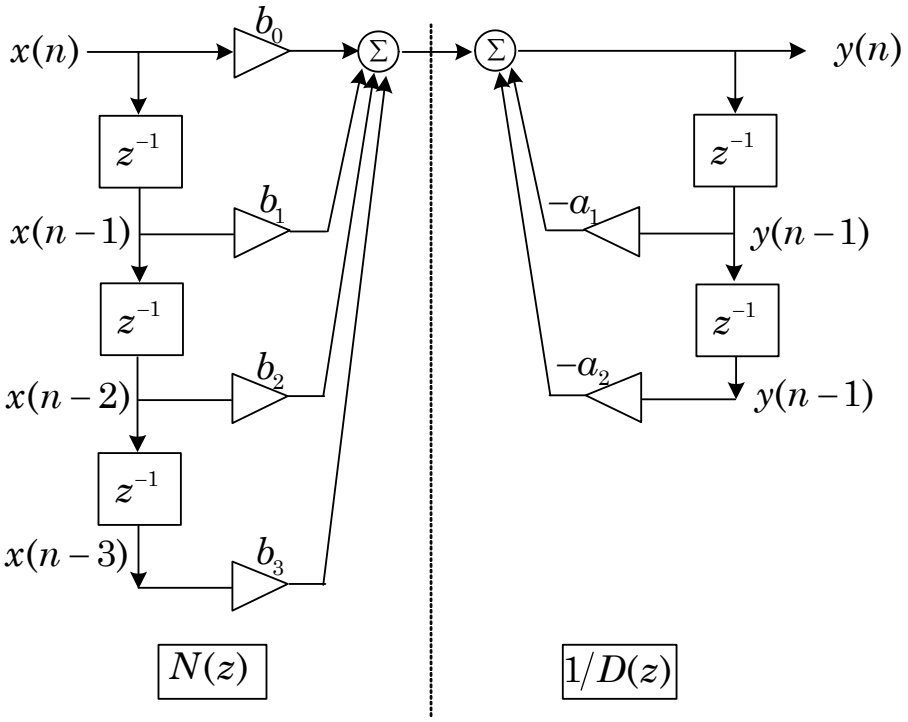


Figure 13- An IIR filter with both feed-forward and feedback parts.

This is one such IIR filter. It has two sections, the right one is called non-recursive and the left is recursive section. The other names for these sections are: feed-forward and feed-backward. The response of this filter is

$$H(z) = N(z)\frac{1}{D(z)} = \frac{N(z)}{D(z)}$$

The output y(n) is sum of the terms from Figure 13.

$$
\begin{aligned}
y(n) &= \left[feed - forward\ output \right] + \left[feed - backward\ output \right] \\
&= \left[b_1 x(n) + b_2 x(n-1) + b_3 x(n-2) + b_4 x(n-3) \right] \\
&+ \left[a_1 y(n-1) + a_2 y(n-2) \right]
\end{aligned}
$$

Where the terms with the 'b' coefficients are feeding forward, and those in the second line with 'a" coefficients are feeding backward, hence the names. The feed-backward terms require the previous values of the signal created by the feed-forward section. Note that the values of the coefficients in the feed-backward section expression are negative of the tap-weights..

The channel response can be written as

$$
H(z) = \frac{\left[b_1 x(n) + b_2 x(n-1) + b_3 x(n-2) + b_4 x(n-3) \right]}{\left[a_1 y(n-1) + a_2 y(n-2) \right]} \qquad (0.13)
$$

This forward and backward process makes this type of filter complex. It can become unstable and has the problem of propagating errors. But this structure allows us to model analog filters and specialized digital filters such as those used in Decision Feedback Equalization.

Adaptive Filtering

What is Adaptive Filtering

The general name for adaptive filtering is **Equalization**. In the broad sense, the word "equalization" refers to any signal processing or filtering technique that is designed to eliminate or reduce channel distortions. How well the channel effects can be equalized depends on how well we know the channel transfer function. Equalization has wide applications in communications from echo cancelling in telephone lines, multipath mitigation in cell phones, beam forming and signal recognition.

The simplest type is equalizer is one called a graphic equalizer, where the bandwidth of interest is divided in a certain number of bands with sliding controls.

Each band has bandpass filter and the slider adjusts power gain settings in that band. Equalization in this case is performed by adjusting power in selected bands. A general type of equalizer, called parametric equalizer can adjust the signal with all three of its parameters; gain, frequency and phase. We can make these adjustments in frequency domain or time. Most of the equalization does rely on filtering, performed most commonly by FIR filters, either a transversal or a lattice type with adjustable tap-weights. The equalization process is based on the knowledge that the impulse response of an FIR filter is same as its tap-weights. So once we know or have estimated the channel impulse response, we apply the inverse of this to the filter, such that by changing the tap-weights of the filter, signal is distorted in the opposite direction of the channel impulse response hence equalizing it. The filter and the algorithm that is used to adjust the tap-weights is called the equalizer. The equalization can be done with knowledge of the channel or blind without knowledge of the channel.

There are many variations on this basic theme, some equalizers operate continuously, some only part of the time. The equalizer filters can be described as to whether they are linear devices that contain only feed-forward elements, or whether they are nonlinear devices that contain both feed-forward and feedback elements (Decision Feedback Equalizers). They can be grouped according to the automatic nature of their operation, which may be either preset or adaptive. They can also be grouped according to the filter's resolution or update rate. The equalization can take place on the symbol or on a sequence.

To understand equalization and the various ways it can be accomplished we start first with the channel impulse response. Assume that the transmitted pulse had a root-raised cosine shape. We want the received signal response to be the root-raised cosine so the overall system transfer function $H(f)$ is equal to the raised-cosine filter, designated $H_{RC}(f)$. Thus, we write it as the product in frequency domain of the response of the transmitter and the receiver.

$$H_{RC}(f) = H_t(f)H_r(f) \tag{1.1}$$

In this way $H_t(f)$ and $H_r(f)$, the response of the transmitter and receiver respectively, each have frequency transfer functions that are the square root of the

raised cosine. Then, the equalizer transfer function needed to compensate for channel distortion is simply the inverse of the channel transfer function:

$$H_e(f) = \frac{1}{H_{RC}(f)} = \frac{1}{\left|H_{RC}(f)\right|} e^{-j\theta_c(f)} \tag{1.2}$$

Sometimes a system frequency transfer function manifesting ISI at the sampling points is purposely chosen (e.g., a Gaussian filter transfer function). The motivation for such a transfer function is to improve bandwidth efficiency, compared with using a raised-cosine filter. When such a design choice is made, the role of the equalizing filter is not only to compensate for the channel-induced ISI, but also to compensate for the ISI brought about by the transmitter and receiver filters [7].

The basic principle is simple. Apply a filter E that mitigates the expected ISI. We can do this in one of two ways: 1. Design E(f) so that all ISI is eliminated, or 2. Design it such that we minimize the mean squared error. The equalizer designed to eliminate all ISI is called a zero-forcing Equalizer and one designed to minimize the squared error (or the variance) is called Minimum Squared Error Equalizer.

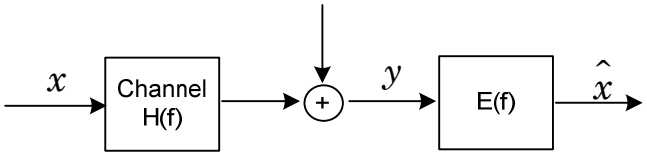


Figure 14 - Basic idea of an equalizer

Linear Estimation Principles

The incoming signal to an equalizer is a random process. There are two possibilities when estimating a random variable, 1. We have no observed samples of the process and 2. We have a set of limited numbers of samples. In the first case, an example helps intuitive understanding. You need to estimate the number of number of hours it will take to complete your homework The best estimate under these conditions would be to say that it will you take about the same amount of time as the average of all your past times (past observations). So if it takes you on the average 3 hours to do your home work, the best current estimate would be

the same. Although this is intuitive, we really made this decision by applying the mean-square error criterion which says that the least-mean-squares estimate of a random variable x given only its average and variance from past observations, is equal to its past average or the best estimate is $\hat{x} = \overline{x}$. The resulting minimum error is $\mathrm{E}e^2 = \sigma_x^2$. For this example, selection of the past average as the estimate minimizes the error, however that error is still large and equal to the variance of this process. So if it was taking you on the average 3 hours, but the variance was 1 hour, then on the average, you could be wrong by that amount. When we do have observations available, the problem becomes that of estimate of the function that relates the input to the output with the least error. The process of equalization tries to achieve this objective.

In most books, two different sets of terminology are used to differentiate between linear estimation and adaptive estimation. Here we will use only one set of terminology for both types of processes. We define terminology we will use in this section:

$x(i)$	*T*ransmitted unknown symbol
σ_x^2	Variance of the input signal
$n(i)$	Additive noise
σ_n^2	Variance of noise
y(i)	Received distorted symbol
$\hat{y}(i)$	Equilizer's estimate of $x(i)$
$e(i)$	Error between estimate and actual symbol
$\hat{x}(i)$	Final decision about $x(i)$
$k(n)$	*Tap* – weight of i_{th} tap
L	Number of taps, 2N + 1

Assume all are column vectors unless shown with transpose symbol, T.

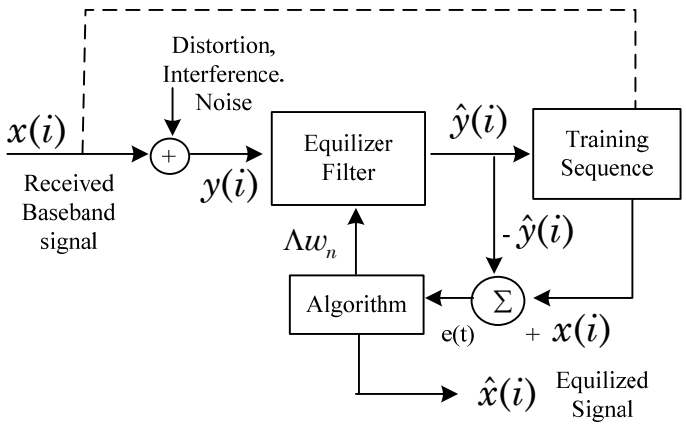


Figure 15 - Equalization based on a transversal filter

In figure 15, we show the general process for equalization. A symbol x(i) is transmitted passing through an unknown channel. We call the observed or the received signal, y(i). Keep in mind that y(i) may contain parts of past symbols, such as the simple transfer relationship shown in Figure 16 where the received signal y(i) is the sum of two transmitted symbols and as such contains ISI. The signal y(i) goes through the equalizer which filters it according to its taps weights. The filtered version, also the estimation called $\hat{x}$ is compared with the transmitted symbol by the algorithm. Based on the difference, the algorithm directs the filter to adjust the tap-weights and continue in this loop until the error is acceptably small.

One of the big questions which we will answer later on is that in real systems we do not know the channel response as neatly as shown in Figure 16. The dotted line in Figure 16 from the transmitter to the receiver is not there.

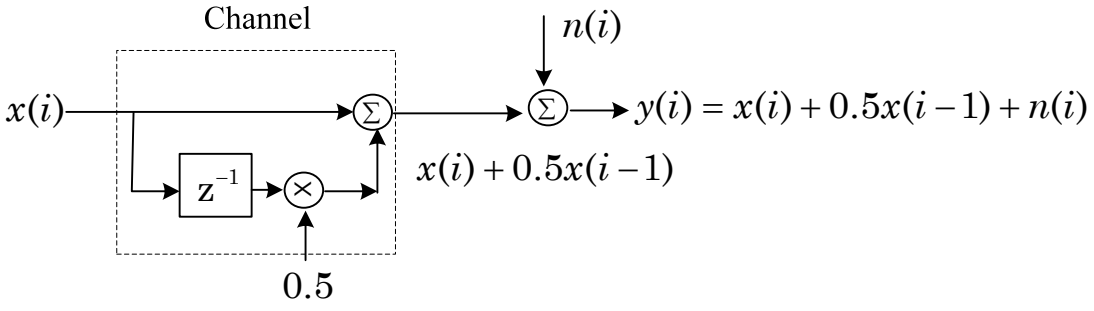


Figure 16 - A hypothetical channel with additive noise

In Figure 16, we see an example of an ISI channel. Each observed instance of y is given by

$$y(i) = x(i) + 0.5x(i-1) + n(i) \tag{1.3}$$

The received symbol *y(i)* consists of contribution from more than one transmitted symbol ($x(i), x(i-1)$), hence this channel introduces ISI as well as noise. The task of the equalizer is to take this signal and filter it in such a way that the filtered symbol is as close to the transmitted symbol as possible. The equalizer performs the linear mapping as stated in Eq. (1.4). We define the transfer function of the equalizer, K as a linear function of tap-weights of the filter. The equalizer takes the inputs and multiplies them by the vector (a row vector) of tap-weights (only one tap is shown in Figure 17) and produces an output which is the estimated symbol.

$$\hat{x} = K^T y \tag{1.4}$$

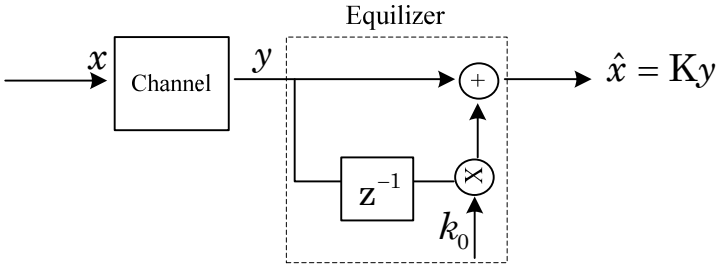


Figure 17 - The input and output of an equalizer

In equalization process that is continuous, there are two separate phases, 1. The training phase, and 2. The tracking phase. The training sequence as shown in Figure 16 provides the missing transmitted data, at least for a short while. This is done via a preset sequence that is appended to the start of the transmitted data. The sequence used is often chosen to be a noise-like, and "rich" in spectral content, which is needed to estimate the channel frequency response. Alternately the training can also consist of sending a single narrow pulse, approximating an ideal impulse and thereby learning the impulse response of the channel. In practice, a

pseudonoise (PN) signal is preferred over a single pulse for the training sequence because the PN signal has larger average power and hence larger SNR for the same peak transmitted power.

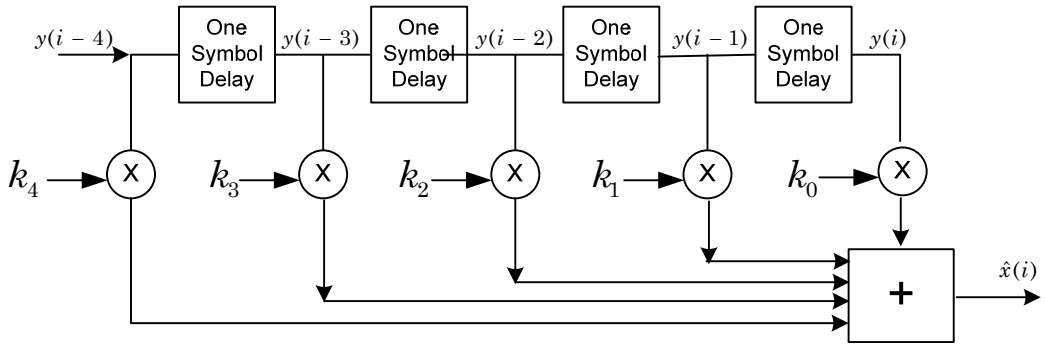


Figure 18 - Feed-forward filter structure used as an equalizer

The transversal filter, depicted in Figure 18, is the most popular form of an easily adjustable equalizing filter consisting of a delay line with T-second taps (where T is the symbol duration). In such an equalizer, the current and past values of the received signal are linearly weighted with equalizer coefficients or tap weights $\{k_n\}$ and are then summed to produce the output. Note that k_n are scalar values but change as a set for each sample. The main contribution is from a central tap, with the other taps contributing echoes of the main signal at symbol intervals on either side of the main signal. If it were possible for the filter to have an infinite number of taps, then the tap weights could be chosen to force the system impulse response to zero at all but one of the sampling times, thus making $H_e(f)$ correspond exactly to the inverse of the channel transfer function in Equation (1.2) Even though an infinite length filter is not realizable, one can still specify practical filters that approximate the ideal case.

Theory behind equalization

Minimization Criteria

Referring to Figure 18, we have two random variables **x**, and **y** of zero-mean. We assume that they are Wide-Sense Stationary, this assumption allows us to use the

ensemble covariance (of the training sequence) as the total channel response for the duration of the communication. It is obvious that the random variables **x** and **y** are correlated in some way. This correlation of course is necessary otherwise no estimation is possible. Signal **y** enters the equalizing transversal filter of tap-weight vector K, where K is column vector of *p* taps. We are interested in estimating the signal **x** by equalizing the channel output **y**. That equalizer estimate is $\hat{x}$ and is the equalized version of **y.** We can write the relationship between y and $\hat{x}$ as a linear relationship

$$\begin{bmatrix} \hat{x}(0) \\ \hat{x}(1) \\ \vdots \\ \hat{x}(p-1) \end{bmatrix} = \begin{bmatrix} k_0^T y \\ k_1^T y \\ \vdots \\ k_{p-1}^T y \end{bmatrix} \tag{1.5}$$

Alternately we can write this in vector form as

$$\hat{x} = ky \tag{1.6}$$

The input data **y** is a vector of *p* values since each estimated $\hat{x}$ is a sum of *p* values of **y**. Since each k_i is a column vector by assumption, we transpose it in Equation (1.5). For example, the first value of $\hat{x}(0)$, *for* $i = 0$ will be calculated as

$$\begin{aligned} \hat{x}(0) &= \begin{bmatrix} k_{p-1} & k_{p-2} & \cdots & k_o \end{bmatrix} \begin{bmatrix} y(i-p) \\ y(i-(p-1)) \\ \vdots \\ y(0) \end{bmatrix} \\ &= k_{p-1}y(-p) + k_{p-2}y(-(p-1)) + \cdots k_0 y(0) \end{aligned}$$

The error between the actual and the equalized signal can be written as

$$Error \approx \left| x - \hat{x} \right|^2 \tag{1.7}$$

To design an optimum equalizer, we reduce this error as small as possible. For this we use a criterion called Minimum Mean Squares Error (MMSE)[15]. This criterion is the one most often used in the design of equalizer filters. Minimizing MSE requires no more than second degree statistics such as covariance and leads

to easy to implement designs. Sometimes the terminology Least Mean Squares Error (LMSE) is also used when under certain conditions the minimum cannot be guaranteed. We will square both sides of (1.7) substituting Equation (1.6) and since **x** is a random variable, we take expectations of both sides to specify the mean error.

$$E\left|e(i)\right|^2 = E\left|x(i) - k_i^T y\right|^2 \tag{1.8}$$

Expanding the right hand side,

$$\mathrm{E}\left|e(i)\right|^2 = \mathrm{E}\left|x(i)\right|^2 + \left[\,\mathrm{E}\left|k_i^T y\right|^2\right] - 2\mathrm{E}\left[x(i)k_i^T y\right] \tag{1.9}$$

The expected value of $\mathrm{E}\left|x(i)\right|^2$ is the variance of the input signal. The second term can be written as $k_i^T R_y k_i$ and the third term as $k_i^T R_{yx,i}$. Now let's call the error, a cost operator J and making substitutions,

$$J(k_i) \triangleq \sigma_x^2(i) - R_{xy,i}k_i - k_i^T R_{yx,i} + k_i^T R_y k_i \tag{1.10}$$

The objective is to minimize J where J is a function of the unknown tap-weight vector k. Differentiate J with respect to k_i, and the cost function reduces to

$$J(k_i) = -R_{xy,i} + k_i^T R_y \tag{1.11}$$

Set equation (1.11) equal to zero to minimize the cost and then rewrite in matrix form, to get a very simple equation for determining optimum tap-weights of the equalizing filter.

$$K_O R_y = R_{xy} \tag{1.12}$$

K_O Represents optimum set of coefficients that result in zero error. This is called the MMSE solution. Solution for optimum tap weights requires knowledge of $R_y \; and \; R_{xy}$. To find estimates, we use training sequences and develop this covariance over the length of the sequence. The taps are computed at each sample of the training sequence and once the error has been reduced down to zero, can

either be set for some time or can be changed periodically depending on the channel variability. The resulting MSE is given by

$$MMSE = J(K_0) = R_y - R_{xy}K_0^T \tag{1.13}$$

The output signal is given by,

$$\hat{x} = k_o y \tag{1.14}$$

Example of a MMSE 3-tap equalizer

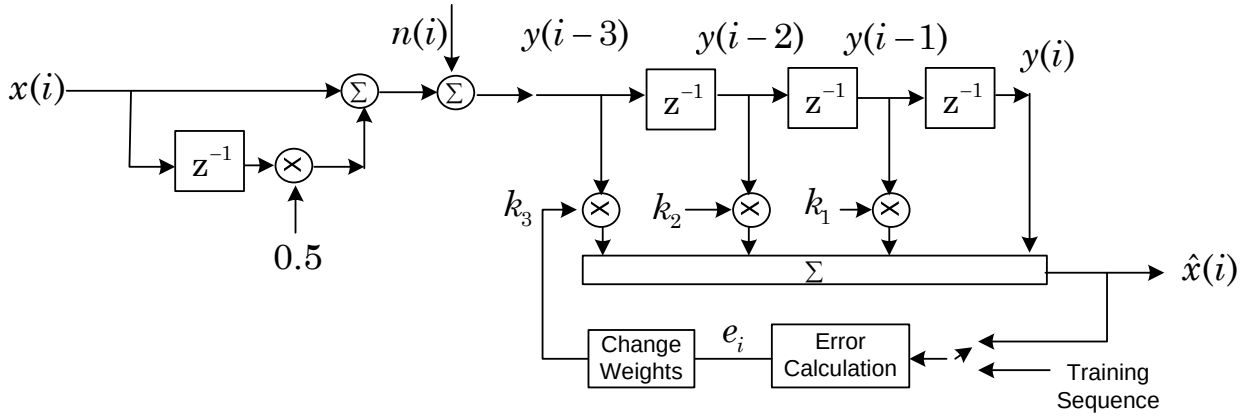


Figure 19 – A transversal feed-forward adaptive filter

In this example borrowed from [15, Sayed, Adaptive filtering which BTW is the best book I have ever seen on this subject,] we use the channel from Figure 19 with ISI and noise. For the equalizing filter, we will use a 3-tap feed-forward structure. Both the signal and noise are zero-mean signals of variance equal to 1. The signal goes through an equalizer with three taps. Given input to the equalizer of $x(i), x(i-1), x(i-2), x(i-4)$, we compute the tap-weights that will equalize the incoming signal with MMSE.

The equalizer tap vector at time *i = 0* is given by

$$k_0^T = \begin{bmatrix} k_1 & k_2 & k_3 \end{bmatrix}$$

From Equation (1.12), we write in matrix form

$$K^T R_y = R_{xy} \tag{1.15}$$

Where K^T is a 1x3 matrix consisting of just one row of three tap-weights. Next compute matrix R_y. Note that individual values in this auto-correlation matrix can be written as $r_y(k) \triangleq E[y(i)y^T(i-k)]$.

$$R_y = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \end{bmatrix} \begin{bmatrix} y(0) & y(1) & y(2) \end{bmatrix} = E \begin{bmatrix} |y(0)|^2 & y(0)y(1) & y(0)y(2) \\ y(1)y(0) & |y(1)|^2 & y(1)y(2) \\ y(2)y(0) & y(2)y(1) & |y(2)|^2 \end{bmatrix}$$

Which is equal to

$$= \begin{bmatrix} r_y(0) & r_y(1) & r_y(2) \\ r_y{}^T(1) & r_y(0) & r_y(1) \\ r_y{}^T(2) & r_y{}^T(1) & r_y(0) \end{bmatrix} \tag{1.16}$$

We can compute each of these correlations by noting that the output signal is given by this equation

$$y(i) = s(i) + 0.5s(i-1) + n(i) \tag{1.17}$$

Multiplying (1.17) by $y^T(i)$ from the right and taking expectations we get

$$\mathrm{E}\left(y(i)y_i{}^T(i)\right) = \mathrm{E}\left(s(0)y_i{}^T(i)\right) + \mathrm{E}\left(0.5s(1)y_i{}^T(i)\right) + \mathrm{E}\left(n(i)y_i{}^*(i)\right) \tag{1.18}$$

The correlations of the second two terms are zeros because the points are uncorrelated, so their cross-correlations are zero.

$$\begin{aligned} \mathrm{E}\left(y(i)y_i{}^T(i)\right) &= \mathrm{E}\left(s(0)y_i{}^T(i)\right) + \mathrm{E}\left(0.5s(1)y_i{}^T(i)\right) + \mathrm{E}\left(n(i)y_i{}^*(i)\right) \\ &= 1 + 0 + 0 = 1 \end{aligned} \tag{1.19}$$

$$\begin{aligned} \mathrm{E}\left(s(i-1)y^T(i)\right) &= \mathrm{E}\,s(i-1)\left[s(i) + 0.5s(i-1) + n(i)\right]^T \\ &= 0 + 0.5 \times 1 + 0 = 0.5 \end{aligned} \tag{1.20}$$

$$\begin{aligned} \mathrm{E}\left(s(i-1)y^T(i)\right) &= \mathrm{E}\,s(i-1)\left[s(i)+0.5s(i-1)+n(i)\right]^T \\ &= 0+0.5\times 1+0=0.5 \end{aligned} \tag{1.21}$$

Multiply Eq. (1.17) by $y_i^T(i)$ and then taking expectation of the product to get the correlation values,

$$\begin{aligned} r_y(0) &= \mathrm{E}\left(y(i)y_i^T(i)\right) \\ &= \mathrm{E}\left(s(0)y_i^T(i)\right)+\mathrm{E}\left(0.5s(1)y_i^T(i)\right)+\mathrm{E}\left(n(i)y_i^T(i)\right) \\ &= 1+0.5\times 0.5+1 \\ &= 2.25 \end{aligned} \tag{1.22}$$

Similarly compute $r_y(1)\ and\ r_y(2)$ by multiplying Equation (1.17) with $y^T(i-1)$ to get

$$\begin{aligned} r_y(1) &= \mathrm{E}\left(y(i)y_i^T(i-1)\right) \\ &= \mathrm{E}\left(s(i)y_i^T(i-1)\right)+\mathrm{E}\left(0.5s(i-1)y_i^T(i-1)\right)+\mathrm{E}\left(n(i)y_i^T(i-1)\right) \\ &= \mathrm{E}\left(s(1)y_i^T(0)\right)+\mathrm{E}\left(0.5s(0)y_i^T(0)\right)+\mathrm{E}\left(n(1)y_i^T(0)\right) \\ &= 0+\mathrm{E}\left(0.5s(0)(s(0)\right)+0=0.5 \end{aligned}$$

$$\begin{aligned} r_y(1) &= \mathrm{E}\left(y(i)y_i^T(i-1)\right) \\ &= \mathrm{E}\left(s(i)y_i^T(i-1)\right)+\mathrm{E}\left(0.5s(i-1)y_i^T(i-1)\right)+\mathrm{E}\left(n(i)y_i^T(i-1)\right) \\ &= \mathrm{E}\left(s(1)y_i^T(0)\right)+\mathrm{E}\left(0.5s(0)y_i^T(0)\right)+\mathrm{E}\left(n(1)y_i^T(0)\right) \\ &= 0+\mathrm{E}\left(0.5s(0)(s(0)\right)+0=0.5 \end{aligned}$$

Compute $r_y(2)$ similarly,

$$
\begin{aligned}
r_y(2) &= \mathrm{E}\left(y(i)y_i^T(i-1)\right) \\
&= \mathrm{E}\left(s(2)y_i^T(1)\right) + \mathrm{E}\left(0.5s(1)y_i^T(1)\right) + \mathrm{E}\left(n(2)y_i^T(1)\right) \\
&= \mathrm{E}\left(s(2)y_i^T(1)\right) + \mathrm{E}\left(0.5s(2)y_i^T(1)\right) + \mathrm{E}\left(n(1)y_i^T(1)\right) \\
&= 0 + 0 + 0 = 0.0
\end{aligned}
$$

Form the matrix from individually computed values of cross-correlations:

$$
R_y = \begin{bmatrix} 2.25 & 0.5 & 0 \\ 0.5 & 2.25 & 0.5 \\ 0 & 0.5 & 2.25 \end{bmatrix} \tag{1.23}
$$

Now we compute R_{xy} which is equal to

$$
R_{xy} = \mathrm{E}xy^T = \begin{bmatrix} \mathrm{E}s(i)y^T(i) & \mathrm{E}s(i)y^T(i-1) & \mathrm{E}s(i)y^T(i-2) \end{bmatrix} \tag{1.24}
$$

Now multiply Equation (1.17) by $x^T(i)$, and taking expectation, we get

$$
\begin{aligned}
\mathrm{E}y(i)x^T(i) &= \mathrm{E}\,x(i)x^T(i) + \mathrm{E}\,0.5x(i-1)x^T(i) + \mathrm{E}\,n(i)x^T(i) \\
&= 1 + 0 + 0
\end{aligned}
$$

The second and the third terms on the right are zero, because individual symbols are uncorrelated with each other and with noise. Similarly with the remaining two terms in Equation (1.24) can be obtained by multiplying the Equation (1.17) by $x^T(i+1)$ and $x^T(i+2)$.

$$
E\,y(i)x^T(i+1) = 0 \quad and \quad E\,y(i)x^T(i+2) = 0
$$

We assume that the channels are WSS and hence the cross correlations are not sensitive to the order of the symbols, we can rewrite these two equivalences as

$$
\begin{aligned}
\mathrm{E}\,y(i)x^T(i+1) &= \mathrm{E}\,y(i-1)x^T(i) = (\mathrm{E}\,x(i)y^T(i-1))^T \\
\mathrm{E}\,y(i)x^T(i+2) &= \mathrm{E}\,y(i-2)x^T(i) = (\mathrm{E}\,x(i)y^T(i-2))^T
\end{aligned}
$$

Now we have all the data to put together R_{xy}

$$R_{xy} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{1.25}$$

Compute the optimum tap-weights by equation (1.15)

$$k_0^T = R_{xy} R_y^{-1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2.25 & 0.5 & 0 \\ 0.5 & 2.25 & 0.5 \\ 0 & 0.5 & 2.25 \end{bmatrix}^{-1}$$
$$= \begin{bmatrix} 0.4688 & -0.1096 & 0.0244 \end{bmatrix}$$

There we have it, the values of each of the three taps-weights. The error in this equalization is equal to

$$MMSE = \sigma_x^2 - R_{xy} k_0 = .5312$$

We can write the cost function as a function of the 3 tap-weights which would result in a three-dimensional error space.

Zero-Forcing Solution

Consider that a single pulse was transmitted over a system designated to have a raised-cosine transfer function $H_{RC}(f) = H_t(f) H_r(f)$. Also consider that the channel induces ISI, so that the received demodulated pulse exhibits distortion, as shown in Figure 20, such that the pulse sidelobes do not go through zero at sample times adjacent to the mainlobe of the pulse. The distortion can be viewed as positive or negative echoes occurring both before and after the mainlobe. To achieve the desired raised-cosine transfer function, the equalizing filter should have a frequency response $H_e(f)$, as shown in Equation (1.2), such that the actual channel response when multiplied by $H_e(f)$ yields HRC(f). In other words, we would like the equalizing filter to generate a set of canceling echoes using the process shown in Figure 20.

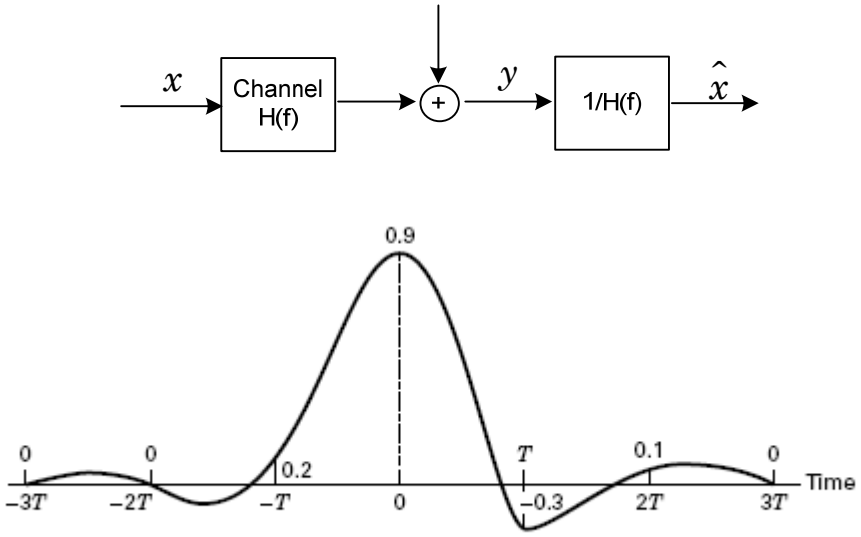


Figure 3.25 Received pulse exhibiting distortion.

Figure 20 - Received pulse exhibiting distortion.

Since we are interested in sampling the equalized waveform at only a few predetermined sampling times, the design of an equalizing filter can be a straightforward task.

We will use a zero-forcing equalizer to do that. The idea is to force the signal to zero at the sample boundary. This method proposed by Lucky [5, 14] does not use MMSE as its criteria but instead the minimization of peak distortion. The solution also ignores noise and can have problems with amplification of noise. For such an equalizer with finite length, the peak distortion is guaranteed to be minimized only if the eye pattern is initially open. However, for high-speed transmission and channels introducing much ISI, the eye is often closed before equalization [8]. Since the zero-forcing equalizer neglects the effect of noise, it is not always the best system solution. Most high-speed telephone line modems use an MMSE criterion because it is superior to a zero-forcing criterion; it is more robust in the presence of noise and large ISI [8].

Example of a Zero-Forcing Three-Tap Equalizer

Consider that the tap weights of an equalizing transversal filter are to be determined by transmitting a single impulse as a training signal. Let the equalizer circuit in be made up of just three taps. Given a received distorted set of pulse samples {x(k)}, with voltage values 0.0, 0.2, 0.9, 0.3, 0.1, as shown in Figure 20, use a zero-forcing solution to find the weights $\{k_0, k_1, k_2\}$ that reduce the ISI so that the equalized pulse samples $\hat{x}(i)$ have the values, {x(-1)= 0, x(0)= 1, x(1)= 0}. Using these weights, calculate the ISI values of the equalized pulse at the sample times $k = \pm 2, \pm 3$. What is the largest magnitude sample contributing to ISI, and what is the sum of all the ISI magnitudes?

Solution

For the channel impulse response specified, Equation (1.14) yields

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} y(0) & y(-1) & y(-2) \\ y(1) & y(0) & y(-1) \\ y(2) & y(1) & y(0) \end{bmatrix} \begin{bmatrix} k_{-1} \\ k_0 \\ k_1 \end{bmatrix} \tag{1.26}$$

$$= \begin{bmatrix} 0.9 & 0.2 & 0 \\ -0.3 & 0.9 & 0.2 \\ 0.1 & -0.3 & 0.9 \end{bmatrix} \begin{bmatrix} k_{-1} \\ k_0 \\ k_1 \end{bmatrix}$$

Solving these three simultaneous equations results in the following weights:

$$\begin{bmatrix} k_{-1} \\ k_0 \\ k_1 \end{bmatrix} = \begin{bmatrix} -0.2140 \\ 0.9631 \\ 0.3448 \end{bmatrix}$$

The values of the equalized pulse samples {x(k)} corresponding to sample times $k = -3, -2, -1, 0, 1, 2, 3$ are computed by using the preceding weights in Equation (1.26), yielding

0.0000, 0.0428, 0.0000, 1.0000, 0.0000, 0.0071, 0.0345

The sample of greatest magnitude contributing to ISI equals 0.0428, and the sum of all the ISI magnitudes equals 0.0844. It should be clear that this three-tap equalizer has forced the sample points on either side of the equalized pulse to be zero. If the equalizer is made longer than three taps, more of the equalized sample points can be forced to a zero value.

Whereas the MMSE equalizer removes most of the ISI but limits the amplification of noise, ZFE enhances the noise

MSE Equalization Types

Symbol-Spaced Equalizers

Equalizer filters are classified by the rate at which the input signal is sampled. A transversal filter with taps spaced T seconds apart, where T is the symbol time, is called a symbol-spaced equalizer. The process of sampling the equalizer output at a rate 1/T causes aliasing if the signal is not strictly bandlimited to 1/T hertz—that is, the signal's spectral components spaced 1/T hertz apart are folded over and super-imposed. The aliased version of the signal may exhibit spectral nulls [8].

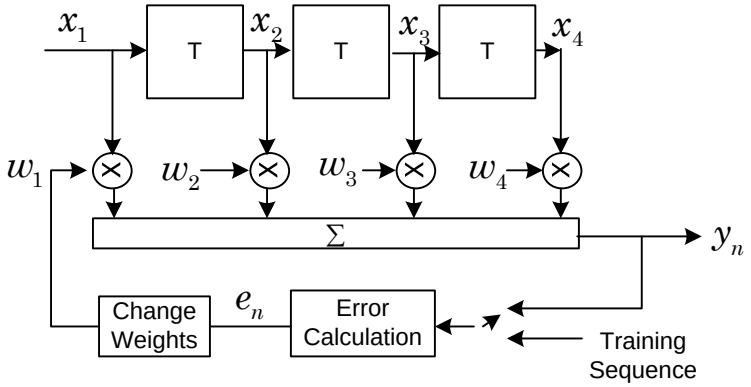


Figure 21 – Symbol Spaced Equalizer

Fractionally Spaced Equalizers (FSE)

A fractionally spaced equalizer is very similar to a symbol-spaced linear equalizer. The major difference is that a fractionally spaced equalizer receives K input samples before it produces one output sample. The signal is oversampled by factor K/T, where T is the symbol and the sample rate. Often K is 2 and the equalizer is

referred to as T/2 FSE. The output sample rate for FSE is 1/T, while the input sample rate is K/T. The weight-updating also occurs at a higher rate.

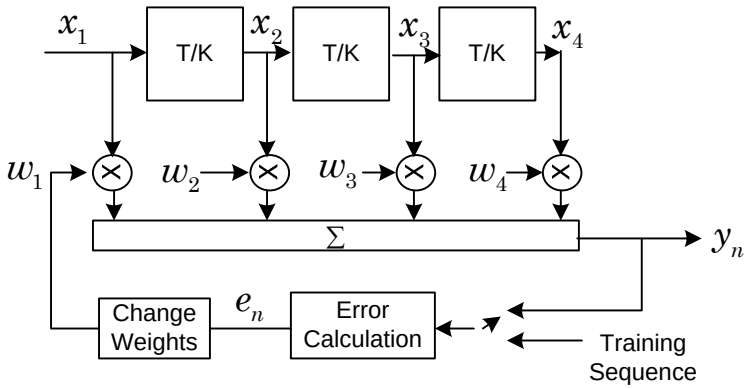


Figure 22 – A K-fractionally Space Equalizer

A filter update rate that is greater than the symbol rate helps to mitigate the difficulty of finding spectral nulls when a smaller sampling rate is used. With a fractionally spaced equalizer, the filter taps are spaced at

$$T' \leq \frac{T}{(1+r)}$$

seconds apart, where r denotes the excess bandwidth. In other words, the received signal bandwidth is

$$W \leq \frac{(1+r)}{T}$$

The goal is to choose T so that the equalizer transfer function $H_e(f)$ becomes sufficiently broad to accommodate the whole signal spectrum. Note that the signal at the output of the equalizer is still sampled at a rate 1/T, but since the tap weights are spaced T seconds apart (the equalizer input signal is sampled at a rate 1/T '), the equalization action operates on the received signal before its frequency components are aliased. Equalizer simulations over voice-grade telephone lines, with T = T/ 2, confirm that such fractionally-spaced equalizers outperform symbol-spaced equalizers [14].

Decision Feedback Equalizer

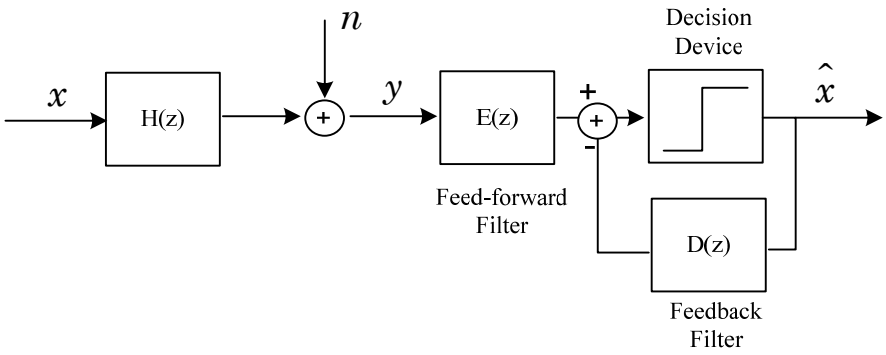


Figure 23- Dual section DFE Equalizer

The basic limitation of a linear equalizer, such as the transversal filter, is that it performs poorly on channels having spectral nulls [11]. Such channels are often encountered in mobile radio applications. A decision feedback equalizer (DFE) is a nonlinear equalizer that uses previous detected decisions to eliminate the ISI on pulses that are currently being demodulated. The ISI being removed was caused by the tails of previous pulses; in effect, the distortion on a current pulse that was caused by previous pulses is subtracted. Two types of DFE are possible: one where the feedback filter is sued with a zero-forcing equalizer (ZF-DFE) so that the symbols coming in to it are ISI-free and 2. DFE used with a MMSE equalizer (MMSE-DFE), in which case the symbols coming in have minimum ISI. Same as in a ZFE, the noise enhancement problem exists in a ZF-DFE. A MMSE-DFE can minimize noise enhancement as compared to the ZF-DFE. Performance of MMSE-DFE is usually better than other types, but we can still get error propagations.

Figure 23 shows a simplified block diagram of a DFE where the forward filter and the feedback filter can each be a linear filter, such as a transversal filter. The figure also illustrates how the filter tap weights are updated adaptively. The nonlinearity of the DFE stems from the nonlinear nature of the feed-back filters.

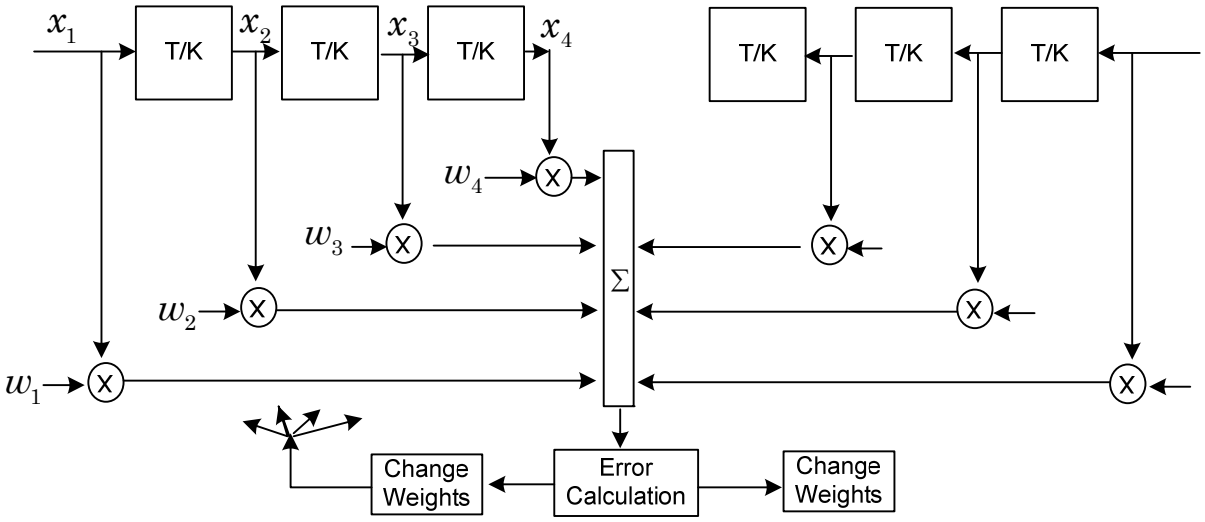


Figure 24- Decision Feedback Equalizer

The principal behind a DFE is that if the symbols previously detected are known (past decisions are assumed to be correct), then the ISI contributed by these symbols can be canceled out exactly at the output of the feed-forward filter by subtracting past symbol values with appropriate weighting. The forward and feedback tap weights can be adjusted simultaneously to fulfill a criterion such as minimizing the MSE.

When only a forward filter is used, the output of the filter contains channel noise contributed from every sample in the filter. The advantage of a DFE implementation is that the feedback filter, which is additionally working to remove ISI, operates on noiseless quantized levels, and thus its output is free of channel noise.

Pre-set equalization

On channels whose frequency responses are known but are mildly time invariant, the channel characteristics can be measured and the filter's tap weights adjusted accordingly. If the weights remain fixed during transmission of data, the equalization is called pre- set equalization; one very simple method of preset equalization consists of setting the tap weights according to some average knowledge of the channel. This is used for data transmission over voice-grade telephone lines at less than 2400 bit/s. The significant aspect of any preset method

is that it is done once at the start of transmission or seldom (when transmission is broken and needs to be reestablished).

Adaptive equalization

When the equalization is capable of tracking a slowly time-varying channel response, it is known as adaptive equalization. It can be implemented to perform tap-weight adjustments periodically or continually. Periodic adjustments are accomplished by periodically transmitting a preamble or short training sequence of digital data that is known in advance by the receiver. The receiver also uses the preamble to detect start of transmission, to set the automatic gain control (AGC) level, and to align internal clocks and local oscillator with the received signal. Continual adjustments are made by replacing the known training sequence with a sequence of data symbols estimated from the equalizer output and treated as known data. When performed continually and automatically in this way, the adaptive procedure (the most popular) is referred to as decision directed [11]. The name "decision directed" is not to be confused with decision feedback (DFE). Decision directed only addresses how filter tap weights are adjusted—that is, with the help of a signal from the detector. DFE, however, refers to the fact that there exists an additional filter that operates on the detector output and recursively feeds back a signal to the detector input. Thus, with DFE there are two filters, a feed-forward filter and a feedback filter that process the data and help mitigate the ISI.

A disadvantage of preset equalization is that it requires an initial training period that must be invoked at the start of any new transmission. Also, a time- varying channel can degrade system performance due to ISI, since the tap weights are fixed. Adaptive equalization, particularly decision-directed adaptive equalization, successfully cancels ISI when the initial probability of error exceeds one percent, (rule of thumb). If the probability of error exceeds one percent, the decision directed equalizer might not converge. A common solution to this problem is to initialize the equalizer with an alternate process, such as a preamble to provide good channel-error performance, and then switch to the decision-directed mode. To avoid the overhead represented by a preamble, many systems designed to operate in a continuous broadcast mode use blind equalization algorithms to form

initial channel estimates. These algorithms adjust filter coefficients in response to sample statistics rather than in response to sample decisions [11].

Algorithms for Adaptive Equalization

Steepest Descent Algorithm

The steepest Descent Algorithm is a class of algorithms that solves the problem of finding the coefficients in an iterative manner. Eq. (1.27) gives the form of weight updates we used before. In steepest Descent Method, we replace the covariance matrices by an *approximation.*

$$k_i = k_{i-1} + \mu\left[R_{xy} - R_y k_{i-1} \right] \qquad (1.27)$$

We replace the term $\left[R_{xy} - R_y k_{i-1} \right]$ in Eq. (1.27) by an approximation p.

$$k_i = k_{i-1} + \mu p \qquad (1.28)$$

The problem of finding optimum tap-weights can now be solved iteratively by starting with a guess for the first weight k_{-1}, and an initial step size μ, change based on the gradient of the vector at that point. Steepest Descent Method is based on the principle that the error calculated at any particular spot on the error surface will decrease only if we move opposite to the gradient at that point, scaled by a step size. We will try to make that clear by example.

The cost function J is a scalar function of the filter coefficients. The partial derivatives or the gradients of a scalar function in matrix form are defined as a column vector in Equation (1.29)

$$J(k_1, k_2, \ldots k_N) \quad \nabla J = \left[\frac{\partial J}{\partial k_1} \quad \frac{\partial J}{\partial k_2} \quad \cdots \quad \frac{\partial J}{\partial k_N} \right]^T \qquad (1.29)$$

To show how this algorithm works, [20] we pick a 2-tap cost function. We picked 2-taps because we can plot the function and immediately have an idea where the minimum lies. The cost function picked is

$$J(k_1, k_2) = 1.0 - k_1 + .75k_2 + k_1 k_2 + 2(k_1^2 + k_2^2) \qquad (1.30)$$

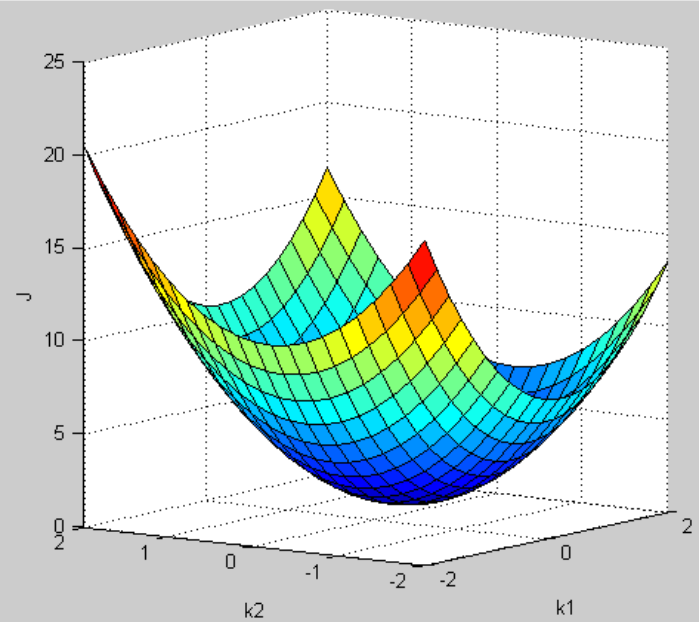


Figure 25 - The Error Surface

We compute the gradient of the error surface by matrix differentiation rules from Equation (1.29)

$$\nabla J = \begin{bmatrix} -1.0 + k_2 + 4k_1 & 0.75 + k_1 + 4k_2 \end{bmatrix} \tag{1.31}$$

Let's assume starting values of $k_1 = 1$ and $k_2 = -1$, calculate the gradient for these values of coefficients using expression (1.31)

$$\nabla J = \begin{bmatrix} 2 \\ -2.25 \end{bmatrix}$$

These gradients are the slope of the surface defined by the two chosen tap-weights, 1 and -1. The gradient is uniquely defined once we know R_x and R_{xy} which are used to form the error function, Equation (1.30). We will normalize these values to simplify the math, and compute a normalized version of the gradient and the associated cost. We compute the cost at each iteration to see if it is has decreased enough and depending on a preset convergence factor, if we can we stop the computations. Normalized increment is give by

$$\begin{bmatrix}\Delta k_1\\ \Delta k_2\end{bmatrix}=\frac{1}{\sqrt{\left(\frac{\partial J}{\partial k_1}\right)^2+\left(\frac{\partial J}{\partial k_2}\right)^2}}\begin{bmatrix}\frac{\partial J}{\partial k_1}\\ \frac{\partial J}{\partial k_2}\end{bmatrix} \tag{1.32}$$

$$\frac{1}{\sqrt{2^2+2.25^2}}\begin{bmatrix}2\\ -2.25\end{bmatrix}=\begin{bmatrix}0.665\\ -0.747\end{bmatrix}$$

This vector becomes the delta change in the tap-weights for the next trial. Here we apply a step-size or a scaling factor to provide better granularity. The estimate for the tap-weights for the next step is the starting tap-vector weight minus the scaled gradient.

$$\begin{bmatrix}k_1^1\\ k_2^1\end{bmatrix}=\begin{bmatrix}1\\ -1\end{bmatrix}-0.1\begin{bmatrix}0.664\\ -0.747\end{bmatrix}=\begin{bmatrix}0.933\\ -0.925\end{bmatrix}$$

With these new values, go to Equation (1.31) and re-compute the gradients and the next increment. The next step gives

$$\begin{bmatrix}k_1^2\\ k_2^2\end{bmatrix}=\begin{bmatrix}0.933\\ -0.925\end{bmatrix}-0.1\begin{bmatrix}0.667\\ -0.744\end{bmatrix}=\begin{bmatrix}0.866\\ -0.850\end{bmatrix}$$

Notice that the gradient vector has not changed much. Continue until the error function decreases and then starts increasing. The optimum values for this case are approximately 0.2705 and -0.3312

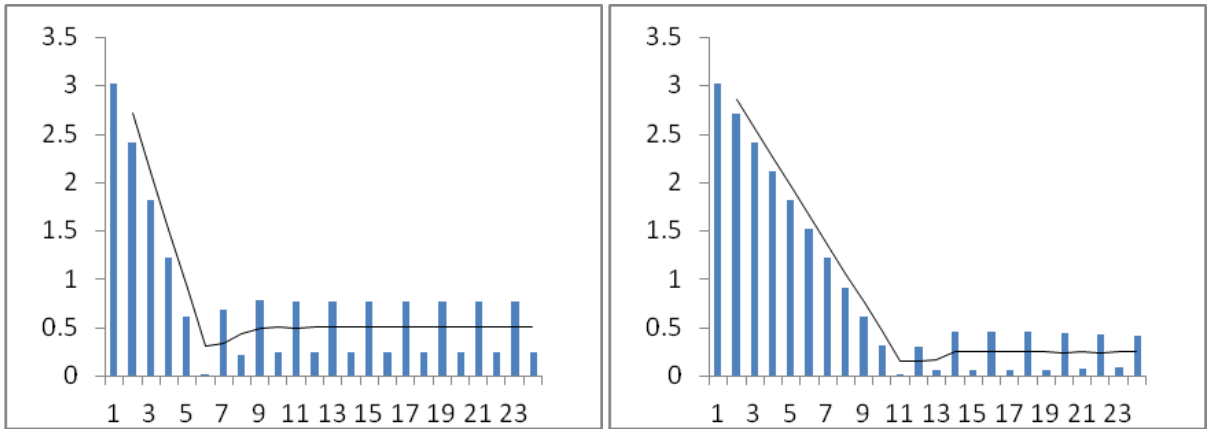


Figure 26 - Convergence time vs. step size for Steepest Descent Method

The algorithm is sensitive to step size as can be seen in the Figure 26 for same starting point but with different step sizes. The oscillations in cost values occur because the step size in too large. An alternate method where the step size is large at first and then reduced as the solution starts to converge, works better. The error surface of a transversal filter working on a W.S.S stochastic process is a bowl shape quadratic with order equal to number of taps [17]. Of course, the bowl becomes hard to graph with anything more than two taps. The steepest descent method step size must meet the following criteria.[4]

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{1.33}$$

Where $\lambda_{\max}$ is the largest eigenvalue of the matrix R_{xy}. The method is deterministic and theoretically can take an infinite number of steps to converge but in most cases it reaches the optimum solution fairly quickly. Formally we can write the SDM as a recursion

$$k_i = k_{i-1} + \mu(R_{xy} - R_y k_{i-1}) \tag{1.34}$$

We can derive this by noting that we have selected each new vector by this general relationship.

$$k_i = k_{i-1} + \mu p \quad for\ i > 0 \tag{1.35}$$

Now calculate the cost function for this new value of k_i by substituting (1.35) into the previous value of the cost function at $(i-1)$,

$$\begin{aligned} J(k)_i &\triangleq \sigma_x^2(i) - R_{xy}^T\left(k_{i-1} + \mu p\right) - \left(k_{i-1} + \mu p\right)^T R_{xy} + \left(k_{i-1} + \mu p\right)^T R_y \left(k_{i-1} + \mu p\right) \\ &= J(k)_{i-1} + \mu\underline{\left(k_{i-1}^T R_y - R_{xy}^T\right)}p + \mu p^T\underline{\left(R_y k_{i-1} - R_{xy}\right)} + \mu^2 p^T R_y p \end{aligned} \tag{1.36}$$

Note that gradient at any step is equal to

$$\nabla J(k) = k^T R_y - R_{xy}^T \tag{1.37}$$

Substitute (1.37) the gradient, at step *i* into (1.36), we get

$$J(k)_i = J(k)_{i-1} + 2\mu \,\mathrm{Re}\left[\nabla J(k_{i-1})p\right] + \underline{\mu^2 p^T R_y p} \tag{1.38}$$

The last term is always positive so we can say that the error function at the next step is always less than at the previous step, assuming the middle term is also positive which becomes a condition for the algorithm to converge.

We can now set the value of the next step as opposite of the gradients

$$p = -\left[\nabla J(k_{i-1})\right]^T = R_{xy} - R_y k_{i-1} \tag{1.39}$$

Which is the gradient at the previous step, which is exactly what we did in the example in this section.

Least Mean Square LMS

Steepest Gradient Descent although simple requires knowledge of variance matrix. Another class of algorithms called **Stochastic Gradient algorithms** further simplify the estimation process by making estimates for the covariance and cross variances. The advantages of using approximations is that we no longer need to know the covariance and cross variances of the actual signals which are not available anyway once we are past the training phase. These algorithms are often used in the tracking mode if the channel is to continue to adapt to the channel after the training sequence is exhausted. These are true adaptive algorithms in that they learn and adjust to the channel. This most popular of all equalization algorithms developed by Widrow [20, 21] is a variation of stochastic-gradient algorithm, called least-mean-square (LMS) algorithm.

In LMS algorithm, we start with the steepest descent method, Equation (1.34) We need two variance matrices, R_y and R_{xy}. But instead of the actual matrices, we use their instantaneous values as an estimate. We rewrite the steepest descent equation by making the following substitutions and then rewriting

$$R_{xy} \cong x(i)^T y_i \quad \textit{and} \quad R_y \cong y_i^T y_i \tag{1.40}$$

$$k_i = k_{i-1} + \mu y_i (x^T(i) - y_i^T k_{i-1}) \tag{1.41}$$

This simple relationship is a consequence of the orthogonality principle that states that the error formed by an optimal solution is orthogonal to the data. Since this is a recursive algorithm, care must be exercised to assure algorithm stability. Stability is assured if the parameter μ is smaller than the reciprocal of the energy of the data in the filter. When stable, this algorithm converges in the mean to the optimal solution but exhibits a variance proportional to the parameter. Thus, while we want the convergence parameter μ to be large for fast convergence but not so large as to be unstable, we also want it to be small enough for low variance. The parameter is usually set to a fixed small amount [12] to obtain a low-variance steady-state tap-weight solution. Schemes exist that permit μ to change from large values during initial acquisition to small values for stable steady-state solutions [13].

The rule of thumb for selecting step size in slowly-varying channel according to [15] is

$$\Delta = \frac{1}{(5N)SNR_{rcvd}} \tag{1.42}$$

Where N is number of taps and SNR is for the received signal. For a signal of SNR 2 and 5 taps, the step size should be 0.02. The example shown in Figure 24 uses a step size of .02.

Example of the LMS algorithm

Figure 27 shows an example of equalization performed by the LMS equalizer. A sine wave experiences ISI and noise as shown. The LMS converges quickly and is successful in removing a great deal of the noise. Figure 28 shows the values of the tap-weights oscillating which is normal part of this process as they are trying to track the signal and one would expect them to change along with the signal.

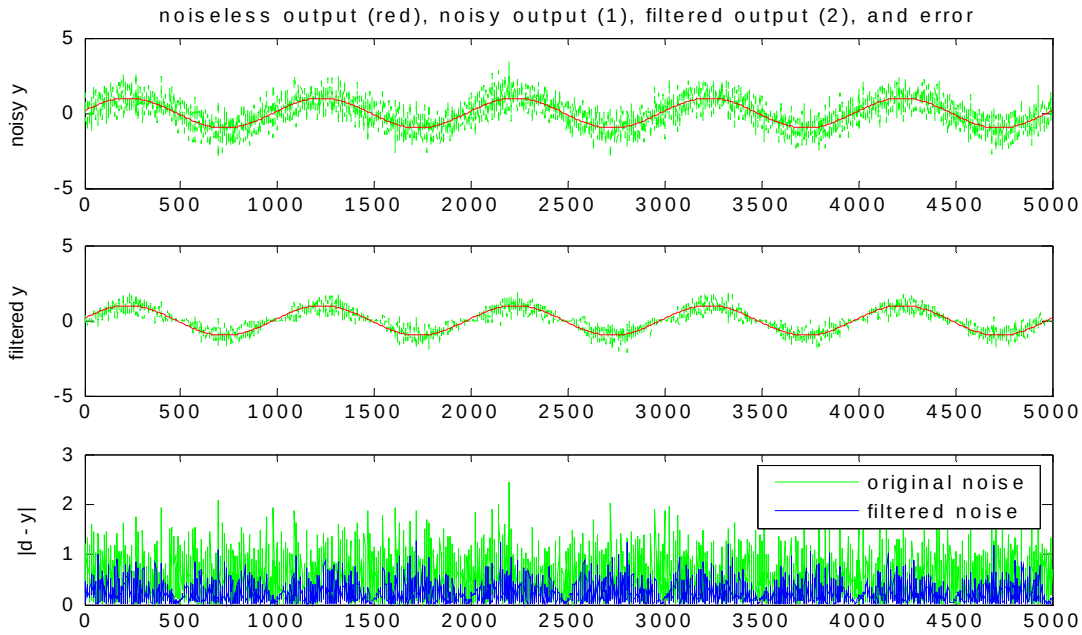


Figure 27 - LMS equalizer input and the equalized signal.

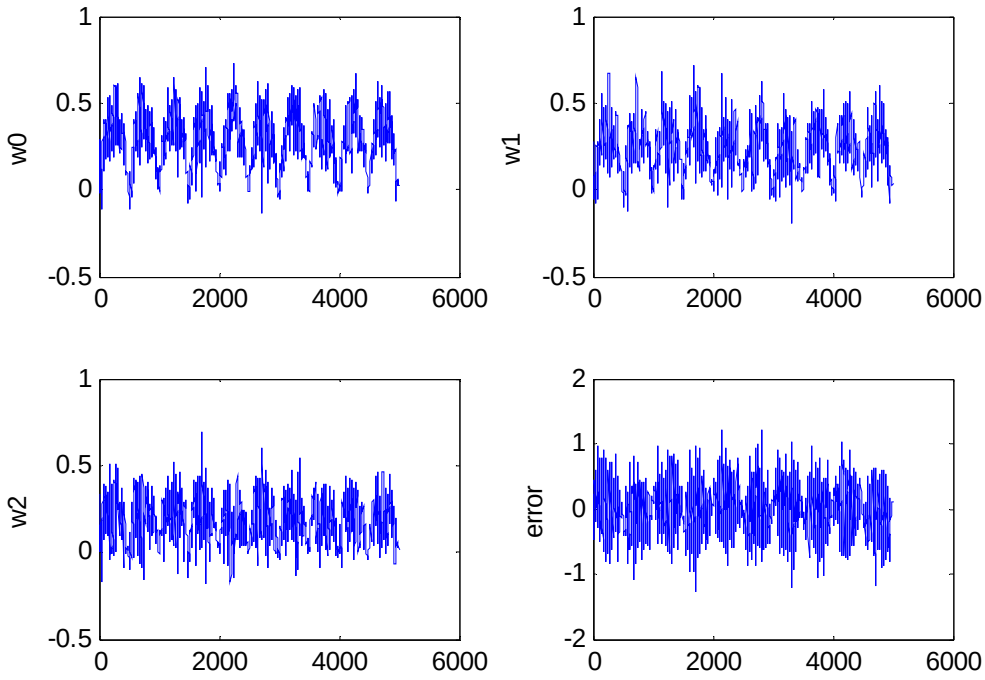


Figure 28 -The convergence of the filter tap-weights

Normalized LMS

One problem with LMS is that once the signal tracking has approached steady state, we ought be able to reduce the size of the step since ostensibly only minor changes are needed. That cannot be done with LMS. The normalized LMS overcomes this.

The normalized LMS (NLMS) algorithm is a modified form of the standard LMS algorithm where the step size is made a function of the received power as well a function of time. It is no longer fixed as in LMS.

$$k(n+1) = k(n) + \mu e(n) \frac{y(n)}{\left\| y(n) \right\|^2} \tag{1.43}$$

You also can rewrite the above equation to the following equation: NLMS usually converges faster than LMS and maintains better tracking.

6.6.5. Minimum Least Sequence Estimation (MLSE)

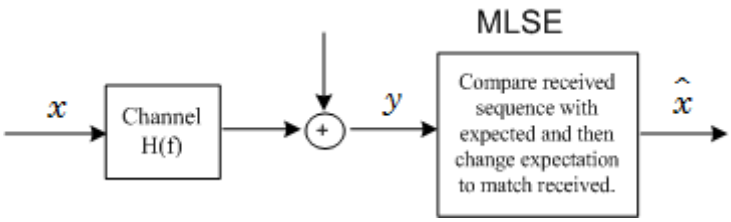


Figure 29 – Sequence adaptation

Previous sections covered equalization performed on the symbol. There is another techniques that works on sequences of symbol rather than on one symbol at a time. In 1967, Andrew Viterbi first presented his now famous algorithm for the decoding of convolutional codes [1] - [3]. A few years later, what is now known as the Viterbi decoding algorithm (VDA) was applied to the detection of data signals distorted by intersymbol interference (LSI) [4]-[8]. For such applications, the algorithm is often referred to as a Viterbi equalizer (VE). Note that many equalizing techniques use filters to compensate for the non-ideal properties of a channel. That is, equalizing filters at the receiver attempt to modify the distorted waveforms. However, the operation of a VE is quite different. It entails making

channel measurements to estimate hc(t) and then adjusting the receiver by modifying its reference waveforms according to the channel environment. The goal of such adjustments is to enable the receiver to make good data estimates from the received message waveforms. With a VE, the distorted message waveforms are not reshaped or directly modified (with the exception of the preconditioning step); instead the mitigating technique is for the receiver to "adjust itself" in such a way that it can better deal with the distorted waveforms.

The VDA has become very popular for processing ISI-distorted signals that stem from a linear system with finite memory. Such a system is referred to as a finite-state machine (FSM), which is the general name given to a system whose output signals are affected by signals that occur earlier and later in time.

Your comments, corrections are welcome. Please post them on www.complextoreal.com

REFERENCES

1. FCC website

2. ITU website.

3. ETSI website

4. Nyquist, H., “Certain Topics of Telegraph Transmission Theory,” Trans. Am. Inst. Electr. Eng., vol. 47, Apr. 1928, pp. 617–644.

5. Glover, I. A., Grant P.M., Digital Communications, Prentice Hall, 1998

6. Wozencraft, J. M. and Jacobs, I. M., Principles of Communication Engineering, John Wiley & Sons, Inc., New York, 1965.

7. TBD

8. Gentile, Ken, Digital Pulse Shaping Filter Basics, Analog Devices, AN-922 Application Note

9. Rappaport T.S., Wireless Communications Principles and Practice, 2nd Edition, Prentice Hall, 2001.

10. Krishnapura N., Pavan S., Mathiazhagan C., Ramamurthi B., "A Baseband Pulse Shaping Filter for Gaussian Minimum Shift Keying," *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, 1998.

11. Hanzo, L. and Stefanov, J., "The Pan-European Digital Cellular Mobile Radio System—Known as GSM," Mobile Radio Communications, edited by R. Steele, Chapter 8, Pentech Press, London, 1992.

7. Lender, A. "correlative Coding for Binary Data transmission", IEEE Spectrum, Vol. 3, No. 2, pp. 104-115, February 1966

8. Digital Transmission Systems, David R. Smith, Ist Edition, 1985 Von Nostrand Reinhold Company

9. Aulin, T. Rydbeck, N. Sundberg, C.-E. TBD

10. Continuous Phase Modulation--Part II: Partial Response Signaling Dept. of Telecommunication Theory, Univ. of Lund, Fack, Lund, Sweden; This paper appears in: Communications, IEEE Transactions on Publication Date: Mar 1981 Volume: 29, Issue: 3

10. Kabal, P.; Pasupathy, S., **Partial-Response Signaling** Communications, IEEE Transactions on Volume 23, Issue 9, Date: Sep 1975, Pages: 921 - 934

11. Proakis, J. G., Digital Communications, McGraw-Hill Book Company, New York, 1983.

12. Lyons, Richard, Understanding Digital Signal Processing, Prentice Hall

13. Sophocles J., Orfanidis, Introduction to Signal Processing, Prentice Hall. Englewood Cliff, 1996

14. Taub, Herbert, Schilling, Donald L., Principles of Communication Systems, Mcgraw Hill Publishing, Second Edition, 1986

15. Couch, Leon, Digital Communication,

16. K. W. HENDERSON, AND W. H. KAUTZ Transient Responses of Conventional Filters, , IRLG TRANSACTIONS ON CIRCUIT THEORY, December 1956

17. Williams, Arthur B., Taylor, Fred J. Electronic Filter Design Handbook, McGraw Hill, 3rd Edition, 1995

18. Davies, Paul, A Mathematical Yarn

19. B. Sklar, Digital Communications: Fundamentals and Applications, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2001.

20. Harris, F., and Adams, B., "Digital Signal Processing to Equalize the Pulse Response of Non Synchronous Systems Such as Encountered in Sonar and Radar," Proc. of the Twenty-Fourth Annual ASILOMAR Conference on Signals, Systems, and Computers, Pacific Grove, California, November 5–7, 1990.

21. Benedetto, S., Biglieri, E., and Castellani, V., Digital Transmission Theory, Prentice Hall, 1987.

22. Lucky, R.W., Automatic Equalization for Digital Communications, Bell Systems technology Journal, Page 547, April 1965

23. Lucky, R. W., Salz, J., and Weldon, E. J., Jr., Principles of Data Communications, Mc-Graw Hill Book Co., New York, 1968.

24. Poularikas, Alexander D. and Ramadan, Zayed M., Adaptive Filtering Primer with Matlab . Taylor & Francis Publishers, 2006

25. Sayed, Ali, Fundamentals of Adaptive Filtering, Wiley, 2003

26. Haykin, S., Adaptive Filter Theory, Upper Saddle River, NJ: Prentice Hall, 2001.

27. Proakis, John G., Salehi, Masoud. Contemporary Communication Systems with Matlab, Brookside/Cole, 2000

28. Qureshi, S. U. H., "Adaptive Equalization," Proc. IEEE, vol. 73, no. 9, September 1985, pp. 1340–1387.

29. Feuer, A., and Weinstein, E., "Convergence Analysis of LMS Filters with Uncorrelated Gaussian Data, "IEEE Trans. on ASSP, vol. V-33 pp. 220–230, 1985.

30. Woodrow, Thinking About Thinking, IEEE, April 2005

31. Macchi, O., Adaptive Processing: Least Mean Square Approach With Applications in Transmission, John Wiley & Sons, New York, 1995.

32. Liu, Kefu http://u.cs.biu.ac.il/~treismr/steepest.pdf, web paper on Steepest descent Method

33. Ta-Hong, Tuan Do-Hong , LMS software module for LMS