1 Multipath Propagation & the Need for Channel Estimation and Equalization

In addition to the lack of synchronization between the transmitter and receiver symbol and carrier oscillators, and the noise in the receiver electronics, in most propagation environments between the transmitter and the receiver, there are non-negligible effects due to multipath propagation. In multipath propagation, on its way between the transmitter and receiver, the transmitted signal is “reflected” by multiple objects, some of which may be mobile. The paths between the transmitter and receiver have different lengths, and thus are associated with different delays, carrier phases, and attenuations. What is received can be (ideally) modeled as a weighted sum of the transmitted modulated signal $x(t)$ at different delays.

$$y(t) = \sum_{\ell} \alpha_\ell x(t - \tau_\ell(t))$$

Provided the bandwidth of the transmitted signal is wide compared to the rate of variation of these delays $\tau_k(t)$ (as is quite often the case), this phenomenon is often modeled, at least on a moderate time scale, as a slowly varying linear time invariant system between the transmitter and receiver, and referred to as the frequency selective fading multipath channel. Over a period for which the delays do not appreciably change from $\tau_k(t) = \tau_k$, we have the LTI system

$$y(t) = \sum_{\ell} \alpha_\ell x(t - \tau_\ell) = \int_{-\infty}^{\infty} h(\tau)x(t - \tau)d\tau$$

where

$$h(t) = \sum_{\ell} \alpha_\ell \delta(t - \tau_\ell)$$

This passband channel can be converted into a complex baseband equivalent channel acting on the transmitted signal as discussed during the lecture on quadrature modulation.

$$h_c(t) = \text{LPF} \{h(t) \exp(-j2\pi f_c t)\}$$

The received signal at baseband $y_c(t)$ is the convolution of the baseband complex transmitted signal $x_c(t)$ with this channel $h_c(t)$ plus the noise introduced by the receiver electronics $w(t)$.

$$y_c(t) = \int_{-\infty}^{\infty} h_c(\tau)x_c(t - \tau)d\tau + w(t)$$

Letting $h_c[n] = h_c(nT_p)$ where $T_p = \frac{T_s}{M}$ is sampling rate of the analog to digital convertor at the receiver, and recalling the form of the transmitted signal, we have

$$y_c[n] = h_c[n] * \sum_k c_k p[n - kM] + w[n] = \sum_k c_k(p * h_c)[n - kM] + w[n]$$
After the matched filter \( p^*_k[-n] \) in the timing recovery, the reconstructed signal

\[
z[n] = \sum_k c_k h[n - kM] + w'[n]
\]  

(7)

where the impulse response

\[
h[n] = p^*_k[-n] * p[n] * h_c[n]
\]  

(8)

combines the pulse shape, the matched filter to the pulse shape, and the multipath channel.

The output of the timing recovery (7) will still suffer in general from inter-symbol interference, since the multipath channel is selected by nature, is not known, and hence will not obey the Nyquist criterion \( h[kM] = \delta[k] \), or even a delayed version of it \( h[kM] = \delta[k - \Delta] \). This is to say, the signal \( z[kM] \) will not only contain a contribution from \( c_{k-\Delta} \), but it will also include contributions from \( c_{k'} \) for other \( k' \), and hence a decision device operating on the output of the matched filter will no longer yield decisions which minimize the probability of making an error.

Generally, the problem of determining which QAM symbols were input given the received signal \( z[n] \) with inter-symbol interference is referred to as channel equalization. We will discuss two different methods of channel equalization, one directly designs a linear filter to attempt to “invert” the channel, while dutifully considering the additional effect of the noise in the received signal as well. The second method first estimates the channel with calculations very similar to those used to design the linear equalizer, then performs calculations that, were the channel known, would minimize the probability of making the wrong guess for each \( c_k \), or guessing the wrong sequence \( \{c_k\} \).

We will see that while the later methods are optimal (assuming of course that the channel has been estimated perfectly), they have complexity which is exponential in the length of the combined impulse response \( h[n] \), and hence are unimplementable for channels of moderate to long lengths. This is why the linear methods that we introduce first remain used for equalization as well as for channel estimation.

## 2 Linear Equalization

The first class of equalization methods utilize a linear filter \( f[n] \) called a linear equalizer, followed by decimation down to the symbol rate, and a decision device (selecting the nearest QAM symbol from the QAM constellation) to undo the effects of the ISI channel \( h[n] \). If there were no noise, and if we had not oversampled the signal, so that \( M = 1 \) (for instance, if we select only the sample instants after performing baud timing synchronization) the output of the equalizer would have z-transform

\[
D(z) = F(z)H(z)C(z)
\]  

(9)

where \( F(z) \), \( H(z) \), and \( C(z) \) are the transforms of the equalizer \( f[n] \), the combined channel impulse response \( h[n] \), and the QAM symbols \( c_k \). From (9), in this non-oversampled (aka baud-spaced) context, we can crudely understand the purpose of the linear equalizer to be to invert the channel, so that we would like to design \( F(z) \) to be \( 1/F(z) \), as this would allow \( D(z) = C(z) \). A problem with not oversampling the signal and using such a baud spaced equalizer is that even if \( h[n] \) is finite length (i.e. a FIR filter), the inverse \( 1/F(z) \), requiring an equalizer which is either infinite length, which will come with stability issues, or a finite length approximation of it. However, as we shall see next, oversampling effectively introduces several parallel channels between the transmitter and receiver, and this or any other means of introducing these parallel channels (for instance through the use of more receive antennas), allows for finite length (FIR) equalizers to perfectly equalize finite length channels.

### 2.1 FIR Linear Equalization: Benefits of Oversampling & other Multi-channel Designs

Now, let’s return to supposing that the input to the equalizer is oversampled by a factor of \( M = 2 \), so that the analog to digital convertor at the receiver is operating at a rate twice the rate at which QAM symbols are transmitted, and none of the processing before the equalizer has decimated the signal. The input to the linear equalizer is modeled from (7) as

\[
y[n] = \sum_k c_k h[n - kM] + w[n]
\]  

(10)
Neglecting the noise \( w[n] \) again to gain a crude understanding of the purpose of the equalizer, we see that the output of the linear equalizer, when still observed at the oversampled rate is

\[
A(z) = F(z)H(z)C(z^2)
\]

where \( C(z^2) \) is the z-transform of the “upsampled” QAM symbols \( c[n] = \sum_k c_k \delta[n-kM] \) and \( A(z) \) is the z-transform of the output of the equalizer. The output \( a[n] \) of the linear equalizer is then decimated to yield \( d_k = a[Mk] \), so that when \( M = 2 \), on the samples at even \( n \) are kept. Equivalently, only the coefficients of \( z \) to an even power are kept in the z-transform after decimation. Since the upsampled signal \( c[n] \) already only had non-zero coefficients at even \( n \) time instants, only the coefficients of even powers of \( z \) of the product \( F(z)H(z) \) will contribute to the decimated output. These coefficients of even powers of \( z \) in \( F(z)H(z) \), in turn, are derived from the product of coefficients of two even powered coefficients of each \( F(z) \) and \( H(z) \), or two odd powered coefficients in each of \( F(z) \) and \( H(z) \), because the sum of an even and an odd number is always odd while the sum of two even or two odd numbers is always even.

This means we can write the z-transform of the output of the decimator \( d_k \) following the linear equalizer as

\[
D(z) = (F_{\text{even}}(z)H_{\text{even}}(z) + F_{\text{odd}}(z)H_{\text{odd}}(z))C(z)
\]

where

\[
H_{\text{even}}(z) = \sum_{n=-\infty}^{\infty} h[2n]z^{-n}, \quad H_{\text{odd}}(z) = \sum_{n=-\infty}^{\infty} h[2n+1]z^{-n},
\]

and \( F_{\text{even}}(z) \) and \( F_{\text{odd}}(z) \) are similarly defined with \( f[n] \). This movement from (11) the input to a decimator to (12) the output of a decimator, and the associated reasoning involving in this instance \( M = 2 \) even and odd numbers, is an instance of a general concept in DSP called a polyphase decomposition.

Under this crude noiseless model, we see that the goal is to select the linear equalizer through \( F_{\text{even}}(z) \) and \( F_{\text{odd}}(z) \) to have the property that \( G(z) = 1 \), so that \( d_k \) is just the original QAM symbols \( c_k \). In truth, we settle for a little less, that \( G(z) = z^{-\Delta} \), so that \( d_k \) is just a delayed version of the original QAM symbols.

Another way to phrase this without z-transforms observes that, under this crude noiseless understanding, one would like the combined channel equalizer response \( g[n] = h[n] * f[n] \) to obey the delayed "Nyquist" criterion we mentioned for pulse shapes \( g[kM] = \delta[k-\Delta] \), as this will enable the output of the equalizer to be decimated, and the subsequent signal \( d_k = a[Mk] \) will contain only a contribution from \( c_k \) and no other \( c_{k'} \).

Unlike the baud-spaced case \( M = 1 \), oversampling allows \( G(z) = 1 \) to happen in principle for an equalizer \( f[n] \) of finite (sufficient) length, owing to Bézout’s theorem.

Bézout’s Theorem states that for any two polynomials \( a(z) \) and \( b(z) \), there always exist two other polynomials \( c(z) \) and \( d(z) \) such that

\[
a(z)c(z) + b(z)d(z) = \gcd(a(z), b(z))
\]

where \( \gcd(a(z), b(z)) \) is a polynomial whose roots are those roots that are common to both \( a(z) \) and \( b(z) \).

Hence, when \( a(z) \) and \( b(z) \) are relatively prime, meaning that they share no common zeros, there always exist polynomials \( c(z) \) and \( d(z) \) (read: FIR filters!) such that

\[
a(z)c(z) + b(z)d(z) = 1
\]

This fact applied to \( G(z) \) in (12) shows that, provided the even \( H_{\text{even}}(z) \) and odd \( H_{\text{odd}}(z) \) parts of the channel share no common roots, a finite impulse response channel \( H(z) \), when used with fractionally spaced equalization, can be perfectly equalized with a finite length equalizer.

For this reason, it is beneficial to utilize fractionally spaced or multi-channel linear equalizers, rather than baud spaced. Having understood the utility of oversampling in a linear equalization context, we now pass to how to design linear equalizers, given the additional difficulties that the channel is unknown and that there is receiver noise.
2.2 Trained Equalization

Because the channel impulse response \( h[n] \) is initially unknown to both the transmitter and receiver, in order to equalize the channel easily, typically the transmitter will periodically transmit a “training sequence” of QAM symbols bearing no information and known previously to the receiver. The linear equalizer can then be selected to make the equalizer output be as close to the training signal (appropriately delayed) as possible, as this will allow the subsequent output of the equalizer during the data period to be as close (on average) to the original QAM symbols as possible, and yield hence decisions which have the lowest error probability possible (assuming Gaussian noise) from a receiver built with a linear equalizer followed by a decision device.

This notion of selecting the equalizer to minimize the average distance between the delayed training symbols and the (decimated) output of the equalizer can be accomplished in a number of ways. If the channel \( h[n] \) is assumed static over the training period and the subsequent data transmission period, so that it does not change, then a block least squares approach can be used, in which the average squared error between the equalizer output and the delayed training signal is minimized by selecting the equalizing filter. In other words, the equalizer \( f \) is selected to solve the optimization problem

\[
\mathbf{f} = \arg \min_{f \in \mathbb{C}^{L_f}} \sum_{k=0}^{K} |c_k - \sum_{\ell=0}^{L_f} f[\ell] y_{2k+\Delta - \ell}|^2 = \arg \min_{f \in \mathbb{C}^{L_f}} \| \mathbf{c} - \mathbf{Yf} \|^2_F
\]

where \( K+1 \) is the length of the training period, and we have introduced the decimated convolution matrix

\[
\mathbf{Y} = \begin{bmatrix}
|y[\Delta]| & |y[\Delta - 1]| & \cdots & |y[\Delta - L_f]|
|y[2 + \Delta] & |y[2 + \Delta - 1]| & \cdots & |y[2 + \Delta - L_f]|
|y[4 + \Delta] & |y[4 + \Delta - 1]| & \cdots & |y[4 + \Delta - L_f]|
\vdots & \vdots & \ddots & \vdots
|y[2k + \Delta] & |y[2k + \Delta - 1]| & \cdots & |y[2k + \Delta - L_f]|
\vdots & \vdots & \ddots & \vdots
\end{bmatrix}
\]

and defined the norm

\[
\|x\|^2_F = \mathbf{x}^H \mathbf{x}
\]

where \( \mathbf{x}^H \) denotes the conjugate transpose, and we have again assumed the oversampling factor to be 2.

Solving the optimization by taking the gradient and setting it equal to zero, we arrive at the equation

\[
\mathbf{f} = (\mathbf{Y}^H \mathbf{Y})^{-1} \mathbf{Y}^H \mathbf{c}
\]

(19)

In other instances, either the channel slowly varies over the duration of the training period, or the complexity and memory required for the matrix calculation are beyond the capabilities of the hardware in use or deemed too inefficient. If this is the case, the least mean squares adaptive equalizer is of interest, as it allows the receiver to minimize the average squared error, while also tracking slow variations in the channel.

The LMS equalizer performs a stochastic gradient descent (discussed in the previous lecture notes) on the cost function

\[
J(f) = \text{AVG}\{ |c_k - \sum_{\ell=0}^{L_f} f[\ell] y_{2k + \Delta - \ell}|^2 \}
\]

As outlined in the synchronization lecture, the stochastic gradient descent replaces the derivative of the cost in a gradient descent on the cost function \( J(f) \) above with a derivative of the instantaneous cost \( |c_k - \sum_{\ell=0}^{L_f} f[\ell] y_{2k + \Delta - \ell}|^2 \), yielding a recursion

\[
f_{k+1} = f_k + \mu (c_k - \mathbf{Y}_{k,\Delta} \mathbf{f}_k) \mathbf{Y}_{k,\Delta}^H (-f_k - \mu \nabla_f J(f_k))
\]

(21)

where \( \mathbf{y}_{k,\Delta} = [y_{2k+\Delta}, y_{2k+\Delta-1}, \cdots, y_{2k+\Delta-L_f}] \).

Finding the equalizer via the iterative recursion rather than the block calculation \([19]\) has the benefit that it can track slow changes to the channel as the minimum \( J(f) \) moves, while the block calculation \([19]\) both requires more memory and assumes the channel is fixed.
2.3 Blind Equalization

In between training periods, the channel continues to vary. Hence it is desirable to continue to adapt the equalizer during the data portion of the frame. Provided the equalizer had converged to equalizing the channel correctly by the end of the training, the decisions at the output of the decision device can provide a continued surrogate for the training signal with which to adapt the equalizer. This is called decision directed equalization, and works well provided it is initialized with a equalizer (at the end of the training period) which is yielding accurate decisions, and that the equalizer is able to track the variations in the channel sufficiently quickly. The associated cost function for decision direction is

$$J(f) = \text{AVG} \left\{ Q \left\{ \sum_{\ell=0}^{L_f} f[\ell]y[2k + \Delta - \ell] \right\} - \sum_{\ell=0}^{L_f} f[\ell]y[2k + \Delta - \ell] \right\}^2 \right\}$$

(22)

where \(Q\{d\} = \arg\min_{c \in C} |d-c|^2\) is the decision device, and the stochastic gradient descent is

$$f_{k+1} = f_k + \mu \left( Q\{y_{k,\Delta}f_k\} - y_{k,\Delta}f_k \right) y^{H}_{k,\Delta}$$

(23)

However, once bad decisions start being made the decision directed mode no longer can converge to the correct location. Furthermore it is not recommended to initially determine the equalizer in decision directed mode. In the case when decision direction has failed to sufficiently track the channel between training periods, or when training information is not available at all to the receiver, a purely blind method of equalization is necessary. One such method of purely blind equalization is called the constant modulus algorithm, its name is derived from the fact that it attempts to restore the property the input QAM constellation has all of its points on the unit circle, and hence has a constant modulus. A cost function that attempts to restore this property is as follows

$$J(f) = \text{AVG} \left\{ \left( |y_{k,\Delta}f|^2 - 1 \right) \right\}$$

(24)

this yields the stochastic gradient descent of the form

$$f_{k+1} = f_k - \mu \left( |y_{k,\Delta}f_k|^2 - 1 \right) (y_{k,\Delta}f_k) y^{H}_{k,\Delta}$$

(25)

2.4 Channel Estimation

The same algorithms employed for trained equalization can also be used to estimate the unknown multipath channel. While channel estimation is not necessary with a linear equalizer, the optimal methods for sequence and symbol detection require knowledge of the channel, and hence if using one of these more complex equalizers, channel estimation is necessary.

To use one of the trained linear equalization algorithms as a channel estimation algorithm, one only needs to swap the roles between the input to the equalizer and the training signals. In particular, assuming again an oversampling factor of 2, the block least-squares channel estimation algorithm will become

$$h = \arg\min_{h \in C^{L_h+1}} \sum_n |y[n + \Delta] - \sum_{\ell=0}^{L_h} h[\ell]c[n - \ell]|^2 = \arg\min_{h \in C^{L_h+1}} \|y_{\Delta,\text{even}} - Ch_{\text{even}}\|^2 + \|y_{\Delta,\text{odd}} - Ch_{\text{odd}}\|^2$$

(26)

where we have simplified the latter expression by noting that the odd samples in \(c[n]\) (the upsampled training signal) are zero, and assuming \(\Delta\) is even and the channel order \(L_h\) is odd, we have defined

$$y_{\Delta,\text{even}} = \begin{bmatrix} y_{\Delta} \\ y_{\Delta+2} \\ y_{\Delta+4} \\ \vdots \\ y_{\Delta+2(k - \frac{L_h-1}{2})} \end{bmatrix}, \quad y_{\Delta,\text{odd}} = \begin{bmatrix} y_{\Delta+1} \\ y_{\Delta+3} \\ y_{\Delta+5} \\ \vdots \\ y_{\Delta+2(k - \frac{L_h-1}{2})+1} \end{bmatrix}, \quad h_{\text{odd}} = \begin{bmatrix} h_1 \\ h_3 \\ h_5 \\ \vdots \\ h_{L_h} \end{bmatrix}, \quad h_{\text{even}} = \begin{bmatrix} h_0 \\ h_2 \\ h_4 \\ \vdots \\ h_{L_h-1} \end{bmatrix}$$

(27)
and
\[
C = \begin{bmatrix}
  c_L^{-1} & c_L^{-1} - 1 & \cdots & c_0 \\
  c_L^{-1} + 1 & c_L^{-1} & \cdots & c_1 \\
  c_L^{-1} + 2 & c_L^{-1} + 1 & \cdots & c_2 \\
  \vdots & \vdots & \ddots & \vdots \\
  c_K & c_K - 1 & \cdots & c_{K - L/2 - 1}
\end{bmatrix}.
\]

(28)

Solving the optimization problem by taking derivatives and setting them equal to zero yields the channel estimate
\[
h_{\text{even}} = (C^H C)^{-1} C^H y_{\Delta, \text{even}}, \quad h_{\text{odd}} = (C^H C)^{-1} C^H y_{\Delta, \text{odd}}
\]

(29)

Again, as with the case of linear equalization, if the channel is deemed likely to vary over the length of the training period, an adaptive channel estimate may be formed with the stochastic gradient descent on the cost function
\[
J(h) = \text{AVG} \left\{ |y[\Delta + n] - \sum_{\ell=0}^{L_h} h[\ell] c[n - \ell]|^2 \right\}
\]

(30)
yielding the iterative channel estimate
\[
h_{n+1} = h_n + \mu (y[\Delta + n] - v_n h_n) v_n^H
\]

or equivalently the iteration
\[
h_{k+1}^{\text{even}} = h_k^{\text{even}} + \mu (y[\Delta + 2k] - c_k h_k^{\text{even}}) c_k^H
\]

(32)
\[
h_{k+1}^{\text{odd}} = h_k^{\text{odd}} + \mu (y[\Delta + 2k + 1] - c_k h_k^{\text{odd}}) c_k^H
\]

(33)
\[
c_k = [c_k, c_k - 1, \ldots, c_k - L/2 - 1]
\]

(34)

which reflects the fact that the upsampled training signal \(c[n]\) is zero for odd samples in (31).

If the channel is deemed to continue to vary during the data portion of the frames, a linear equalizer can be trained up to equalize the channel during the training period, and this linear equalizer can continue to be adapted with decision direction during the data period. The outputs of the linear adaptive equalizer during the data period, in turn, can be used as the training signals for an adaptive channel estimation algorithm during the data portion of the frame.

3 Optimal Detection

In this section, we aim to generalize the optimal detectors for QAM in noise developed in the previous lecture notes to the frequency selective fading case. The received signal model is the same as in those lecture notes, with the only difference being that now instead of the pulseshape \(p[n]\) we have in its place the multipath channel \(h[n]\), as discussed above, because we are no longer designing this impulse response we cannot selected to obey the criterion we used for the pulse shape in order to simplify the complexity of the detector in the case of just a noisy channel.

Additionally, because the different QAM symbols decisions no longer decouple in the optimization, we have two different senses of minimizing the probability of being wrong. In the first sense, called maximum likelihood sequence detection, we wish to decide the entire sequence of QAM symbols all at once, and minimize the probability of selecting the wrong sequence. In the second sense, called maximum a posteriori symbol detection, we make decisions about each of the QAM symbols individually and we want to minimize the individual error probabilities of selecting the wrong symbol.

3.1 Maximum Likelihood Sequence Detection: Viterbi Algorithm

3.2 Maximum A Posteriori Symbol Detection: Forward Backward (BCJR) Algorithm

3.3 A Look at Complexity
particular, the complexity of these methods is on the order of $N|C|^{L_c}$ where $N$ is the block length in symbols, and $L_c$ is the channel length, and $|C|$ is the size of the QAM constellation. Additionally, in order to keep the metrics for the states required, a memory with at least $|C|^{L_c}$ elements is necessary. While this reflects gigantic savings over naive implementations of maximum likelihood which require on the order of $|C|^N$ calculations, the exponential growth in the length of the channel remains a problem. This exponential growth of complexity renders Viterbi or BCJR equalization unimplementable in all but the shortest of channels and smallest of constellations. For instance, what may appear to be a relatively short channel order of 75, all taps non-zero, together with a 16-QAM constellation requires a memory with $2^{4\times75} = 2^{300} \approx 10^{90}$ elements in it, and even more computations for every element in the block of received samples.

If that number doesn’t look big, note that even the most aggressive estimates for the number of particles in the universe cap out around $10^{87}$, so you would need more elements in the memory than there are particles in the universe.

If for some reason you don’t trust physics and are more computer oriented, you may consider that the biggest companies deal with distributed storage systems on the order of a petabytes ($10^{15}$) or exabytes ($10^{18}$) or that estimates of annual total internet traffic (not saved) by 2015 don’t exceed a zettabyte ($10^{21}$). Even when cubed these numbers are astronomically smaller than the amount of memory you would need just to equalize a single channel with 75 non-zero taps for 16-QAM.

The point is, exponentially growing complexity is bad, and a force to be reckoned with. Linear equalization remains in use precisely because, despite its suboptimality, it has complexity which is linear in the length of the channel and the block length.

Additionally, just as synchronization can be significantly aided via the exploitation of the forward error correction coding of the data, equalization sees a similar performance benefit, albeit with a net complexity increase. Depending on the way that they are implemented, such equalizers can be called turbo equalizers or decision feedback equalizers.