Equalization

John Barry

October 5, 2015

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250
barry@ece.gatech.edu

Contents

1 Motivation

In Chapter 4 there was only one impediment to reliable communication: additive noise. Here we consider the more realistic case where, in addition to adding noise, the channel *filters* the transmitted signal. Such filtering arises in nearly every practical scenario: A wireless channel filters because the transmitted signal bounces off of different reflectors with different path lengths before arriving at the receiver; an electronic cable filters because different frequencies are attenuated by different amounts; a fiber-optic cable filters because different wavelengths propagate at different speeds. This seemingly modest change to the channel model has profound implications on how communications systems are designed and implemented. *Equalization* is broadly defined as any signal processing aimed at counteracting the dispersive effects of such a filtering channel. As we will see, the equalization process can range from a simple linear filter to a sophisticated signal processing algorithm, and it can be performed in advance (at the transmitter) or after the fact (at the receiver).

Equalization plays a central role in most communications applications, ranging from wireless to wired to storage devices. Equalization is thus a fundamental and important topic in its own right. Moreover, the problem of equalization is at its core the problem of mitigating interference, namely interference between one information symbol and another. As such, the concepts developed in this chapter readily extend to a much broader class of interference problems, many of which at a glance have no obvious connection to the linear filter model considered here, including co-channel interference, multiuser interference, intercell interference, and multiple-input multiple-output (MIMO) communications.

2 Models and Metrics

The nature of the equalization problem is strongly impacted by the type of modulation scheme being used. In this chapter we will limit our discussion to a communication system that uses a single-carrier passband version of pulse-amplitude modulation such as quadrature-amplitude modulation (QAM) and phase-shift keying (PSK), as opposed to nonlinear alternatives such continuous-phase frequency-shift keying or pulse-position modulation. Our motivation for this restriction is three-fold: 1) A wide variety of applications use QAM and its variations; 2) the mere presence of a dispersive channel suggests that a spectrally efficient modulation scheme like QAM is an appropriate design choice; and 3) linear modulation schemes like QAM are more amenable to equalization through signal processing.

In a QAM or PSK system, the transmitter conveys a sequence of L *information* or *message* symbols $\{a_0 \cdots a_{L-1}\}$ by transmitting a signal whose complex envelope (see Chap. 4) has the form:

$$x(t) = \sum_{k=0}^{L-1} a_k g(t - kT), \tag{1}$$

where $g(t)$ is the transmit pulse shape, a_k is the k-th information symbol chosen from a complex alphabet $\mathcal{A}$ (typically QAM or PSK), and where T is the signaling period. This equation describes a pulse train, where a new pulse is sent every T seconds, and the amplitude of the k-th pulse is modulated by the k-th symbol a_k.

The impact of a dispersive channel is to filter the transmitted signal. Filtering a PAM signal yields another PAM signal, where the pulse shape after the filter is simply a filtered version of the pulse shape before the filter. Therefore, the complex envelope of the received waveform after a channel that first filters and then adds noise can be written as:

$$y(t) = \sum_{k=0}^{L-1} a_k h(t - kT) + v(t), \tag{2}$$

where the received pulse shape $h(t)$ is the filtered version of the transmit pulse shape $g(t)$, and where $v(t)$ is the complex envelope of the additive noise, assumed to have independent real and imaginary parts, each of which is white and Gaussian with power-spectral density (PSD) $N_0/2$.

2.1 Discrete-Time Intersymbol Interference Model

The change in pulse shape is more problematic than it may at first seem, because it generally leads to interference between neighboring symbols, a phenomenon known as *intersymbol interference (ISI)*. Even when the transmitter carefully chooses its transmit pulse shape $g(t)$ so that the set of translated pulses $\{g(t-kT)\}$ in (1) are mutually orthogonal, the dispersive channel destroys that orthogonality, so that the set of translated pulses $\{h(t-kT)\}$ seen at the receiver are not orthogonal.

Were $\{h(t-kT)\}$ orthogonal, the detection problem would simplify dramatically: The receiver could apply the received waveform to a filter matched to $h(t)$, and sample the matched filter (MF) output at the symbol rate; the k-th sample would be a sufficient statistic for optimal detection of the k-th symbol, and could be passed to a simple memoryless quantizer to arrive at the corresponding symbol decision. In other words, if $\{h(t - kT)\}$ were orthogonal, the optimal receiver could make symbol-by-symbol decisions, considering each symbol in isolation.

It can be shown that a filter matched to $h(t)$ followed by a symbol-rate sampler is an optimal front-end (providing sufficient statistics) even when the received pulses $\{h(t-kT)\}$ are not orthogonal [2]. The cascade of this sampled-matched filter and a subsequent discrete-time noise-whitening filter is known as the whitened-matched filter, and is a common starting point for front-end processing for the general case [5]. However, in this chapter we opt for a simpler

presentation based on an assumption that the transmitter pulse shape is the minimum-bandwidth Nyquist pulse shape, i.e., that $g(t) = \sin(\pi t/T)/(\pi t/T)$. Our motivation for this choice is based on two observations:

- A practical transmitter will often either aim to implement this pulse shape exactly, or will implement a close approximation (such as a square-root-raised-cosine pulse shape with a modest amount of excess bandwidth).
- Using the ideal minimum-bandwidth pulse simplifies our exposition without obscuring the main conceptual problem of equalization. Extensions to handle arbitrary pulse shapes are more cumbersome but straightforward.

Since the received pulse shape is a filtered version of the transmitted pulse shape, the bandwidth of the received pulse will match that of the transmit pulse. Therefore, our assumption that the transmitter uses a minimum-bandwidth pulse, whose bandwidth is $1/(2T)$, implies that the bandwidth of the received pulse $h(t)$ will also be $1/(2T)$. Furthermore, this implies that the received signal (before noise) will similarly be bandlimited to $1/(2T)$. We can thus pass the noisy received signal $r(t)$ through an ideal low-pass filter with cutoff frequency $1/(2T)$ without losing any information about the transmitted symbols; the low-pass filter will only reject out-of-band noise that is irrelevant to the detection problem. Further, the fact that the filter output is bandlimited enables us to sample it at the symbol rate $1/T$ without losing any information. Applying (2) to such a low-pass filter that is scaled to have unit energy, and then sampling at the symbol rate $1/T$ leads to the following equivalent discrete-time model for the channel:

$$r_k = \sum_{i=0}^{L-1} a_i h_{k-i} + n_k, \tag{3}$$

or more compactly $r_k = a_k * h_k + n_k$, where $h_k = \sqrt{T}h(kT)$ is a scaled and sampled version of the received pulse shape, and where $\{n_k\}$ is a complex-valued circularly symmetric white-Gaussian noise process with PSD N_0, so that its real and imaginary parts are mutually independent, each being white and Gaussian with PSD $N_0/2$. A block diagram is shown in Fig. 1.

The discrete-time model of (3) and Fig. 1(c) will be our starting point for the remainder of the chapter. The discrete-time impulse response h_k captures the severity of the ISI, and will be referred to as the ISI response. To be concrete we will assume that the impulse response is causal, starting at time zero, and has memory μ, so that the only nonzero coefficients are h_0 through h_μ. We will at times require that the memory be finite, $\mu < \infty$.

To recap, here are the key assumptions that we make in this chapter, limiting its scope:

- single-carrier passband linear modulation (such as QAM or PSK)
- minimum-bandwidth pulse shape

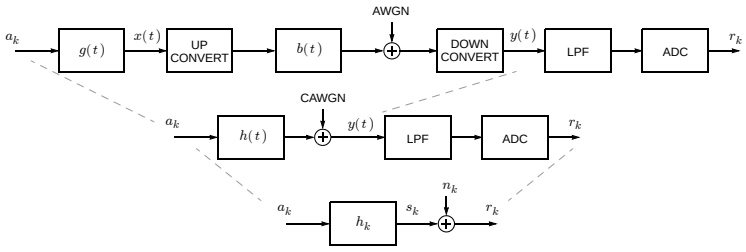

Figure 1: Three equivalent views of the channel model: (a) passband model; (b) baseband continous-time model; (c) baseband discrete-time model.

- linear time-invariant ISI response with additive white Gaussian noise
- perfect channel knowledge available at transmitter (for precoding schemes only) and at receiver (for all schemes)
- perfect synchronization

See Sect. 8 for further reading on the situation where one or more of these assumptions is violated.

2.2 Arithmetic, Harmonic, and Geometric Means

For any real and nonnegative function $S(e^{j\theta})$ over $\theta \in [-\pi, \pi)$, including for example any valid power spectral density, let us define three mean operators as follows:

$$\begin{aligned}
\text{arithmetic mean}: \quad \mathcal{A}\left\{S(e^{j\theta})\right\} &= \frac{1}{2\pi}\int_{-\pi}^{\pi} S(e^{j\theta})d\theta, \\
\text{harmonic mean}: \quad \mathcal{H}\{S(e^{j\theta})\} &= \frac{1}{\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{1}{S(e^{j\theta})}d\theta}, \\
\text{geometric mean}: \quad \mathcal{G}\{S(e^{j\theta})\} &= \exp\{\frac{1}{2\pi}\int_{-\pi}^{\pi}\log S(e^{j\theta})d\theta\}.
\end{aligned}$$

These will prove to be useful throughout the chapter. A few comments:

- In all three cases, the mean of a constant is that same constant.
- All three means can be seen as a special case of the transformation $f^{-1}(\frac{1}{2\pi}\int_{-\pi}^{\pi} f(S(e^{j\theta}))d\theta)$, or equivalently $f^{-1}(\mathcal{A}\{f(\,\cdot\,)\})$, where either $f(x) = x$, $f(x) = 1/x$, or $f(x) = \log(x)$.

- As written the logarithm in the geometric mean is base e, although any other base will yield the same result, as long as the outer exponential operator is changed to be its inverse.

- All three can be seen as the limiting case (as $N \to \infty$) of first taking N equally spaced samples $S_k = S(e^{jk2\pi/N})$ for $k \in 0, \ldots, N-1$, and second applying the corresponding means to the resulting finite set of real numbers. For example, when $N = 2$ the arithmetic mean is $(S_0 + S_1)/2$, the harmonic mean is $2/(1/S_0 + 1/S_1)$, and the geometric mean is $\sqrt{S_0 S_1}$.

- The three means satisfy

$$\mathcal{H}\{S\} \leq \mathcal{G}\{S\} \leq \mathcal{A}\{S\}, \tag{4}$$

  for any real and nonnegative function $S = S(e^{j\theta})$. Equalities are met if and only if $S(e^{j\theta})$ is a constant, independent of θ.

- When $S(e^{j\theta})$ is the PSD of a random sequence x_k, the arithmetic mean reduces to $\mathcal{A}\{S(e^{j\theta})\} = E(|x_k|^2)$, the power in the random sequence.

- When $S(e^{j\theta})$ is the PSD of a random sequence x_k, the geometric mean reduces to the mean-squared error of an optimal linear predictor $\hat{x}_k = \sum_{i=1}^{\infty} p_i x_{k-i}$ whose predictor coefficients $\{p_i\}$ are chosen to minimize the mean-squared error [9], *i.e.*, $G\{S(e^{j\theta})\} = \min_{\{p_i\}} E(|\hat{x}_k - x_k|^2)$.

- The arithmetic mean operator is linear, so that $\mathcal{A}\{a_1 S_1 + a_2 S_2\} = a_1 \mathcal{A}\{S_1\} + a_2 \mathcal{A}\{S_2\}$. In contrast, neither of the harmonic and geometric mean operators is linear.

- The geometric mean involving a product or ratio satisfies $\mathcal{G}\{a\frac{S_1 S_2}{S_3}\} = a\frac{\mathcal{G}\{S_1\}\mathcal{G}\{S_2\}}{\mathcal{G}\{S_3\}}$.

- It can be shown that $\mathcal{G}\{|1 + be^{-j\theta}|^2\} = 1$ for any constant b satisfying $|b| \leq 1$. Combined with the previous fact, this implies that $\mathcal{G}\{|M(e^{j\theta})|^2\} = 1$ for any rational, monic, and minimum phase filter of the form $M(z) = 1 + m_1 z^{-1} + m_2 z^{-2} + \ldots$.

2.3 Shannon Capacity

The Shannon capacity is an upper bound on how fast one can communicate reliably. The Shannon capacity of the ISI channel with AWGN of (3), subject to a power constraint on the input of $E(|a_k|^2) \leq E_a$, can be expressed in terms of the discrete-time Fourier transform $H(e^{j\theta}) = \sum_k h_k e^{-jk\theta}$ of the ISI response, according to [8]:

$$C = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log_2(1 + \frac{S_a(e^{j\theta})|H(e^{j\theta})|^2}{N_0}) d\theta \tag{5}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} \log_2(1+SNR(\theta))d\theta \quad (6)$$

$$= \log_2 \mathcal{G}\{1+SNR(\theta)\} \quad \text{bits/symbol}, \quad (7)$$

where for convenience we have introduced the *SNR spectral density*, defined by:

$$SNR(\theta) = \frac{S_a(e^{j\theta})|H(e^{j\theta})|^2}{N_0},$$

which can be interpreted as the SNR per unit frequency. The capacity formula (5) is not complete until the PSD $S_a(e^{j\theta})$ of the transmitted symbols is specified: The capacity-achieving PSD can be found via a procedure with a geometrical interpretation known as *waterpouring*, according to:

$$S_a(e^{j\theta}) = \max[0, \lambda - N_0|H(e^{j\theta})|^{-2}], \quad (8)$$

where the water-level parameter λ is adjusted until the power constraint is met with equality, $\frac{1}{2\pi}\int_{-\pi}^{\pi} S_a(e^{j\theta})d\theta = E_a$.

Example: The waterpouring procedure is illustrated in Fig. 2 for the case when the channel ISI response is $H(z) = 1+(0.6+0.6j)z^{-1}+0.6jz^{-2}$, the transmit power constraint is $E_a = 1$, and the noise PSD is $N_0 = 0.25$, which corresponds to a channel SNR of 9.2 dB. As the figure shows, the optimal PSD concentrates its power at those frequencies for which the channel gain is large, while avoiding those frequencies for which the channel gain is small.

Rather than adopting the optimal waterpouring spectrum, if the transmitted symbols are instead chosen independently and identically distributed (i.i.d.) with a uniform distribution from an alphabet with energy E_a, so that $S_a(e^{j\theta}) = E_a$, then the formula (5) would no longer be the capacity of the channel, but it would nevertheless represent an upper bound on the achievable rate for any input meeting the i.i.d. constraint.

In the special case of a channel with no ISI, both the channel magnitude response and the optimal waterpouring spectrum are flat, namely $|H(e^{j\theta})| = |h_0|$ and $S_a(e^{j\theta}) = E_a$, so that the SNR spectral density reduces to the constant $SNR_0 = E_a|h_0|^2/N_0$, independent of θ; in this case, the capacity formula of (5) reduces to the familiar form $C = \log_2(1+SNR_0)$.

2.4 SNR Benchmarks: The Matched Filter Bound and Shannon SNR

We will encounter two types of SNR's in this chapter, and it will be important to not confuse the two:

- *Channel SNR* – This is the SNR of the underlying ISI channel, and is by definition the ratio of the power in the received signal (the first term in

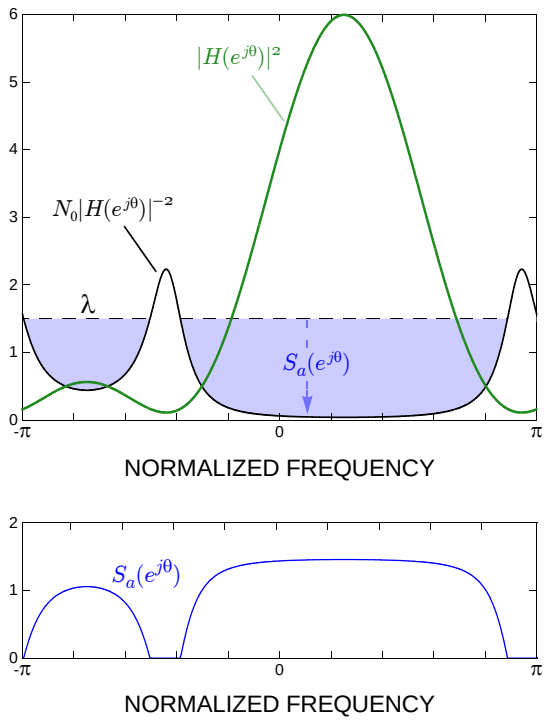

Figure 2: Geometric interpretation of the waterpouring procedure (8) for determining the capacity-achieving transmit spectrum: Water is poured into the "bowl" $N_0|H(e^{j\theta})|^{-2}$ until the average water depth matches the power constraint; the depth of the water at each frequency is then the optimal PSD, illustrated separately in the bottom figure.

(3)) divided by the power in the received noise (the second term in (3)), namely:

$$SNR = \frac{E(|\sum_i a_i h_{k-i}|^2)}{E(|n_k|^2)} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi} S_a(e^{j\theta})|H(e^{j\theta})|^2 d\theta}{N_0} = \mathcal{A}\{SNR(\theta)\}, \tag{9}$$

where $SNR(\theta) = S_a(e^{j\theta})|H(e^{j\theta})|^2/N_0$. This is the SNR of the ISI channel itself, before any attempts at equalization are made.

- *Post-Equalization SNR* – Also referred to as the *equalizer* SNR, this is the SNR after equalization. Importantly, this generally applies only to the class of equalizers that strive to *transform the ISI channel into an ISI-free channel*; these include the linear equalizers of Sect. 4, the decision-feedback equalizers of Sect. 5, and precoding strategies of Sect. 6. The performance of any such equalizer is then easily and effectively quantified by the SNR of the resulting ISI-free channel.

The post-equalization SNR is an effective metric for comparing equalizer alternatives. The higher the post-equalization SNR, the better the equalizer. A particularly useful bound for the post-equalization SNR of any practical equalization strategy is the so-called *matched-filter bound* on the post-equalization SNR, which is based on the unrealistic assumption that receiver has (genie-aided) knowledge of all of the interfering symbols. In other words, when making a decision about the i-th symbol a_i, the receiver somehow knows all of the interfering symbols $\{a_{k\neq i}\}$. A receiver with this knowledge can do no better than to reconstruct the ISI from the known interfering symbols and subtract it from the channel output (3), yielding the ISI-free channel $z_k = a_i h_{k-i} + n_k$. Subtracting the ISI in this way clearly transforms the ISI channel into an ISI-free channel whose SNR is:

$$SNR_{\text{MFB}} = \frac{E_a \sum_k |h_k|^2}{N_0}. \tag{10}$$

When the transmit PSD is flat ($S_a(e^{j\theta}) = E_a$), which is often the case, this SNR is identical to the SNR of the underlying channel in (9). Otherwise, in the general case, we use $E_a = \mathcal{A}\{S_a(e^{j\theta})\}$ in (10). The reason that the MFB is a bound for the post-equalization SNR is because it arises from optimal processing with genie-aided knowledge of the interfering symbols; optimal (or suboptimal) processing without such knowledge can only perform worse.

An alternative upper bound on the post-equalization SNR of any practical equalization strategy is the "effective SNR" SNR_{Shannon} achieved by a capacity-achieving system, which can be found by setting (5) equal to $\log_2(1 + SNR_{\text{Shannon}})$ and solving for SNR_{Shannon}, yielding:

$$SNR_{\text{Shannon}} = \mathcal{G}\{1 + SNR(\theta)\} - 1.$$

The inequality in (4) directly leads to the conclusion that the effective SNR of a capacity-achieving system cannot exceed the SNR of the underlying channel:

$$SNR_{\text{Shannon}} \leq SNR,$$

with equality reached only when $SNR(\theta)$ is a constant, independent of θ, which can happen only when the channel has no ISI.

3 Optimum Trellis-Based Detectors

3.1 The Trellis Diagram

Before adding noise, the channel model in (3) applies the sequence of information symbols a_k to a linear filter with impulse response h_k, producing the filter output $s_k = a_k * h_k$. For the case when the channel memory is finite ($\mu < \infty$), it will be convenient to view this filter as a *finite-state machine (FSM)*. In particular, let $\boldsymbol{\theta}_k = [a_{k-1}, a_{k-2}, \ldots a_{k-\mu}]$ denote the filter *state* at time k. Because each symbol $a_k \in \mathcal{A}$ is chosen from a finite alphabet, the number of such states is clearly finite, namely $Q = |\mathcal{A}|^{\mu}$. More than just having a finite number of states, the filter is a FSM because it further satisfies two additional properties. First, the current output s_k is uniquely determined by the pair $(a_k, \boldsymbol{\theta}_k)$, a fact that becomes clear when we rewrite the convolution $s_k = a_k * h_k$ as:

$$s_k = h_0 a_k + [h_1, h_2, \ldots, h_\mu]\boldsymbol{\theta}_k^T. \tag{11}$$

Second, the next state $\boldsymbol{\theta}_{k+1}$ is also uniquely determined by the pair $(a_k, \boldsymbol{\theta}_k)$; this is also clearly true, since the next state $\boldsymbol{\theta}_{k+1}$ can be formed by concatenating the current input a_k with the first $\mu - 1$ entries of the current state $\boldsymbol{\theta}_k$. Key to the FSM formulation is that the state sequence satisfies the Markov property $P(\boldsymbol{\theta}_{k+1}|\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k) = P(\boldsymbol{\theta}_{k+1}|\boldsymbol{\theta}_k)$. Intuitively this means that knowledge of the state at time k tells us everything we need to know about what the next state at time $k+1$ might be; the history of how we got there is irrelevant. Two tapped-delay line models for the finite-state machine are shown in Fig. 3.

So far we have assumed that the transmitter sends a symbol sequence of length L, starting with a_0 and ending with a_{L-1}. To proceed further we need to clarify what happens before a_0 and after a_{L-1}. While it might seem natural to assume that nothing is transmitted before and after, namely to assume that $a_k = 0$ for all $k \notin \{0, \ldots, L-1\}$, this can be problematic when $0 \notin \mathcal{A}$, because it would require that an augmented alphabet $\mathcal{A} \cup \{0\}$ be used to define the states during the transients at the beginning and ending of the message. Besides, in practice it is more common to use preambles or postambles of non-information-bearing symbols for tasks such as frame synchronization and channel estimation. To better capture this reality we will identify one symbol from the alphabet as the *idle* symbol, denoted $a^0 \in \mathcal{A}$, and we will assume that the transmitter uses both a preamble and a postamble, each consisting of a block of μ idle symbols. In other words, we assume that $a_k = a^0$ for $k \in \{-\mu, \ldots -1\}$ and

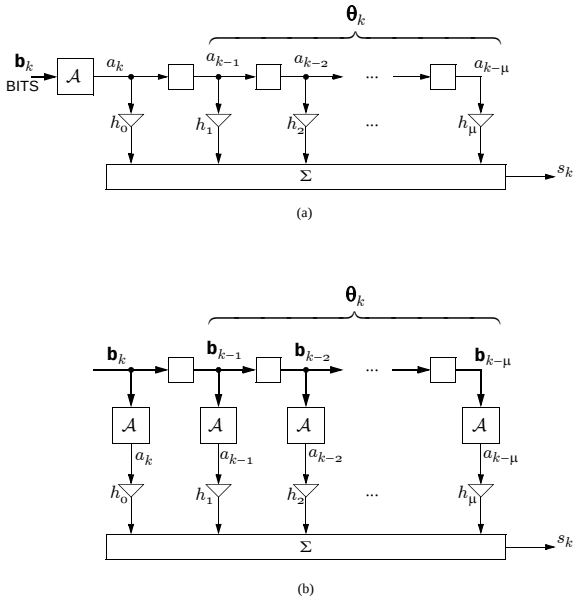

Figure 3: Two equivalent views of the ISI filter as a finite-state machine. In both cases the state of the filter is the contents of its memory elements. In (a) the memory stores the information symbols, while in (b) the memory stores the corresponding message bit blocks, where $\mathbf{b}_k$ denotes the unique block of $\log_2 |\mathcal{A}|$ bits associated with the symbol $a_k \in \mathcal{A}$. The model of (b) is preferred when computing probabilities for the message bits using the BCJR algorithm.

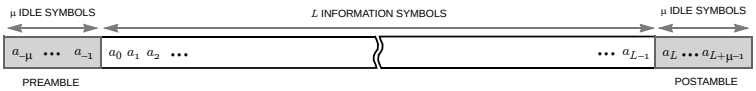

Figure 4: We assume that the L information symbols are sandwiched between a preamble and postamble, both consisting of a block of μ idle symbols.

$k \in \{L, \ldots L+\mu-1\}$. The preamble and postamble are illustrated in (4). The preamble ensures that the state is in the all-idle state at time zero, namely $\boldsymbol{\theta}_0 = [a^0, \ldots a^0]$, while the postamble ensures that the state is in the all-idle state at time $L+\mu$, namely $\boldsymbol{\theta}_{L+\mu} = [a^0, \ldots a^0]$.

In what follows we associate an integer label $p \in 0, \ldots, Q-1$ with each state, and we will will reserve the zero label for the all-ide state; with this notation, the preamble and postamble ensure that both $\boldsymbol{\theta}_0 = 0$ and $\boldsymbol{\theta}_{L+\mu} = 0$. The symbols in the postamble are sometimes referred to as *termination* symbols, since their purpose is to terminate the trellis to a known state (namely, state zero).

The *trellis diagram* is defined simply as a plot of all possible states versus time.

Example: Suppose a block of $L = 5$ binary symbols $\mathcal{A} = \{\pm 1\}$ are transmitted across an ISI channel with memory $\mu = 2$, sandwiched between a preamble and postamble of two idle symbols $a^0 = -1$. There are four possibilities for the state $\boldsymbol{\theta}_k = [a_{k-1}, a_{k-2}]$: it can be $[-1,-1]$, $[+1,-1]$, $[-1,+1]$, or $[+1,+1]$, which are assigned the integer labels 0, 1, 2, and 3, respectively. The corresponding trellis diagram is shown in Fig. 5(a). The preamble ensures that the state begins at state zero at time 0. The state at time 1 is either 0 or 1, depending on the value of a_0. The trellis terminates at state zero at time 10. The path corresponding to the message $[a_0, \ldots a_4] = [-1,+1,+1,-1,+1]$ is highlighted. In all there are $|\mathcal{A}|^L = 2^5 = 32$ distinct paths through this trellis, one for each possible message.

Longer messages, larger alphabets, and higher ISI memory can lead to a more complicated trellis.

Example: Suppose a block of L symbols chosen from an alphabet of size $M = |\mathcal{A}| = 4$ (such as 4-QAM or 4-PSK) is transmitted across an ISI channel with memory $\mu = 2$. In this case the state $\boldsymbol{\theta}_k = [a_{k-1}, a_{k-2}]$ is the previous two symbols, so that the number of states is $Q = |\mathcal{A}|^\mu = 16$. The preamble and postamble ensure that the starting state (at time $k = 0$) and the ending state (at time $k = L+\mu$) are both zero. The corresponding trellis diagram is shown in Fig. 5(b). The trellis diagram has a total of $L+\mu$ stages. The first L stages of the trellis correspond to the L message symbols. In these stages there are $M = |\mathcal{A}| = 4$ *branches* emanating from each node, one for each possible message symbol. The last μ stages of the trellis

correspond to the postamble idle symbols. In these stages there is only one branch emanating from each node, namely the branch corresponding to the idle symbol.

3.2 MAP or ML Sequence Detection: The Viterbi Algorithm

The maximum-a-posteriori (MAP) estimate of the message is the message $\mathbf{a} = [a_0, \ldots, a_{L-1}] \in \mathcal{A}^L$ that maximizes the *a posteriori* probability $P(\mathbf{a}|\mathbf{r}) = f(\mathbf{r}|\mathbf{a})P(\mathbf{a})/f(\mathbf{r})$, or equivalently maximizes just the numerator $f(\mathbf{r}|\mathbf{a})P(\mathbf{a})$. When all messages are equally likely, the MAP sequence decision reduces to that maximum-likelihood (ML) sequence decision, which is the symbol sequence $\mathbf{a} \in \mathcal{A}^L$ that maximizes the likelihood $f(\mathbf{r}|\mathbf{a})$. A brute-force search for either type of decision would require that $f(\mathbf{r}|\mathbf{a})$ be computed for each of the $|\mathcal{A}|^L$ possible message sequences. The complexity of such an exhaustive search would thus grow exponentially with the message length L. In this section we describe an efficient solution to the MAP or ML sequence detection problem whose complexity grows only linearly with L.

Every message $\mathbf{a} \in \mathcal{A}^L$ uniquely determines a "path" through the trellis, as specified by the state sequence $\boldsymbol{\theta} = [\theta_0, \ldots \theta_{L+\mu}] \in \mathcal{A}^{\mu(L+\mu+1)}$. The reverse is true as well, i.e., every path through the trellis uniquely specifies the corresponding message sequence. Therefore, to find the MAP estimate of the message, we need only find the MAP estimate for the *path* through the trellis, which is the state sequence $\boldsymbol{\theta} \in \mathcal{A}^{\mu(L+\mu+1)}$ that maximizes the *a posteriori* probability $P(\boldsymbol{\theta}|\mathbf{r}) = f(\mathbf{r}|\boldsymbol{\theta})P(\boldsymbol{\theta})/f(\mathbf{r})$, or equivalently maximizes just the numerator $f(\mathbf{r}|\boldsymbol{\theta})P(\boldsymbol{\theta})$. Because the noise components are independent, and because the noiseless channel output s_k depends only on the states at time k and time $k+1$, this numerator reduces to:

$$\begin{aligned} f(\mathbf{r}|\boldsymbol{\theta})P(\boldsymbol{\theta}) &= \prod_{k=0}^{L+\mu-1} f(r_k|s_k = s^{(\theta_k,\theta_{k+1})})P(a_k = a^{(\theta_k,\theta_{k+1})}) \\ &= \prod_{k=0}^{L+\mu-1} \gamma_k(\theta_k, \theta_{k+1}), \end{aligned} \tag{12}$$

where $\gamma_k(p,q) = f(r_k|s^{(p,q)})P(a_k = a^{(p,q)})$ can be interpreted as a *branch metric* for the branch in the trellis diagram from state p at time k to state q at time $k+1$. Here we use $a^{(p,q)} \in \mathcal{A}$ to denote the unique input symbol associated with the transition from state $p \in \{0, \ldots, Q-1\}$ to state $q \in \{0, \ldots Q-1\}$. (When such a transition is impossible, we take the corresponding probability to be zero.) Similarly, we use $s^{(p,q)}$ to denote the unique FSM output associated with a transition from state p to state q, as defined by (11). This means that a FSM that starts out in state p will output $s^{(p,q)}$ and move to state q when the input symbol is $a^{(p,q)}$. Intuitively, the branch metric for a branch from one

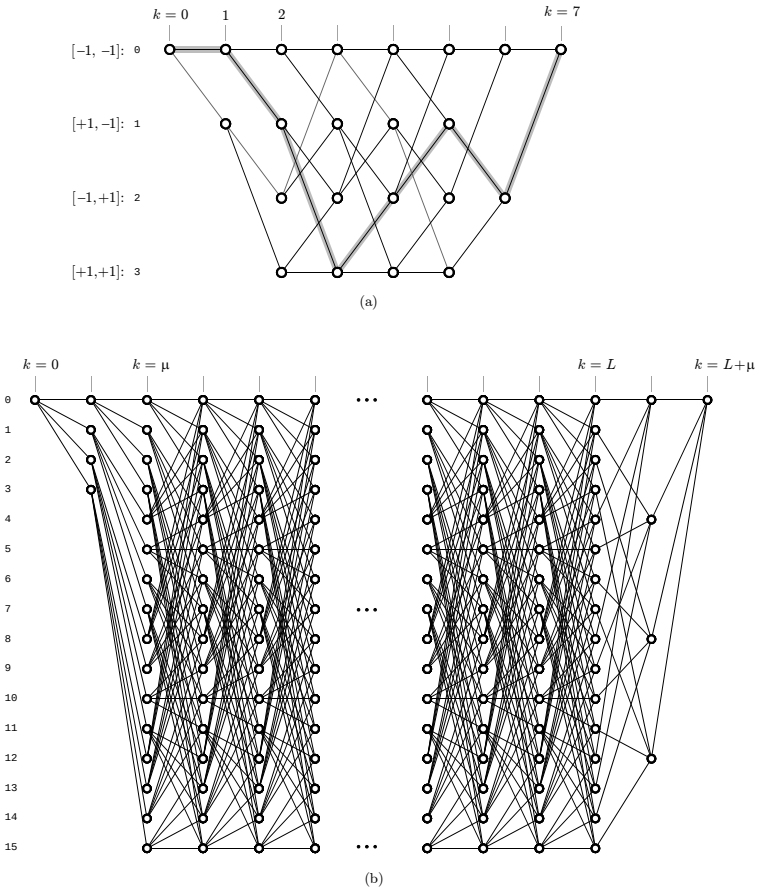

Figure 5: Examples of trellis diagrams: (a) A relatively simple trellis diagram for a message of length $L = 5$, alphabet size $|\mathcal{A}| = 2$, and channel memory $\mu = 2$. There are $Q = |\mathcal{A}|^{\mu} = 4$ states. (b) A more complicated trellis diagram for alphabet size $|\mathcal{A}| = 4$ and channel memory $\mu = 2$, with $Q = |\mathcal{A}|^{\mu} = 16$ states. Each message sequence $\mathbf{a}$ corresponds to a unique "path" through the trellis that begins at state zero at time zero and ends at state zero at time $L+\mu$.

state to another is a measure of how probable that particular transition is; the smaller the branch metric, the less likely the state transition along the corresponding branch occurred. The extreme case of a zero branch metric indicates an impossible state transition; this generally arises because the second factor $P(a_k = a^{(p,q)})$ is zero, meaning that it is impossible to make a transition from state p to state q.

In effect, (12) rephrases the MAP sequence detection problem as the problem of finding a path through the trellis with the largest path metric, where the metric for a path is defined as the product of the branch metrics in its path, according to (12). The *Viterbi algorithm* is an efficient solution to this problem whose complexity grows only linearly with the message length L [5]. Let us define the *survivor* for the node (p, k) as the partial path to that node (i.e., the partial path starting at node $(0, 0)$ and ending at node (p, k)) with maximal metric. Further, let $\hat{\alpha}_k(p)$ denote this maximal survivor metric.[1] The decision path can thus be thought of as the survivor path for ending node of the trellis, namely the node $(0, L + \mu)$. The key to the Viterbi algorithm is the observation that the survivors for the states at time $k + 1$ will always build on the survivors at time k. In particular, the survivors for the states $q \in \{0, 1, \ldots Q - 1\}$ at time $k + 1$ can be computed recursively from the Q survivors at time k according to:

$$\hat{\alpha}_{k+1}(q) = \max_p \{\hat{\alpha}_k(p)\gamma_k(p, q)\}, \tag{13}$$

with initialization $\hat{\alpha}_0(p) = \delta_p$, to account for the fact that the state is initially $p = 0$ at time 0.

The Viterbi algorithm starts at the zero-th stage of the trellis, computes the survivors for the nodes at time $k = 1$ based on the above recursion, then moves on to the next stage, computes the survivors for the nodes at time $k = 2$, and so on. It continues moving through the trellis from left to right and computes the survivors recursively, according to the above recursion, and further stores for each node the index of the previous state that was selected by the maximization of (13). At any time k the Viterbi algorithm need only track Q survivors, independent of k, one for each possible state at time k. This represents a significant amount of pruning, since the number of partial paths to any node at time k grows exponentially with k. Further, observe that the complexity at the k-th stage of the trellis is fixed, independent of the value of k; the number of computations required to implement (13) does not depend on k. This makes the overall complexity of the Viterbi algorithm a *linear* function of the sequence length L, in stark contrast to the exponential dependence on L that would be required by an exhaustive search for the best message.

The Viterbi algorithm described above is *multiplicative* because the path metric (12) is the product of all of the branch metrics in its path; because the logarithm is monotonic, we can equivalently maximize the logarithm of (12), *i.e.* maximize $\sum_k g_k(\theta_k, \theta_{k+1})$, which leads to an *additive* version of (13):

[1] The reason for the "hat" is that this metric can be interpreted as an estimate or approximation of a quantity that will be denoted $\alpha_k(p)$ in the next section.

$$\lambda_{k+1}(q) = \max_p\{\lambda_k(p) + g_k(p,q)\}, \tag{14}$$

where $\lambda_k(p) = \log \hat{\alpha}_k(p)$ and $g_k(p,q) = \log \gamma_k(p,q)$. Exploiting the fact that the noise is Gaussian, the branch metric $\gamma_k(p,q) = f(r_k|s^{(p,q)})P(a^{(p,q)})$ reduces to:

$$\gamma_k(p,q) = \frac{1}{\pi N_0} e^{-|r_k - s^{(p,q)}|^2/N_0} P(a_k = a^{(p,q)}), \tag{15}$$

so that a negatively scaled version $\mu_k(p,q) = -N_0 g_k(p,q)$ of the additive branch metric is:

$$\mu_k(p,q) = |r_k - s^{(p,q)}|^2 - N_0 \log \frac{P(a_k = a^{(p,q)})}{\pi N_0}. \tag{16}$$

The negative scaling factor means that, instead of maximizing the sum $\sum_k g_k(\theta_k, \theta_{k+1})$ of the original additive branch metrics, the MAP detector can now equivalently *minimize* the sum $\sum_k \mu_k(\theta_k, \theta_{k+1})$ of the new branch metrics. Furthermore, when all symbols are equally likely, the subtracted term in (16) will be independent of p and q and thus the same for every branch in the trellis, meaning that the MAP detector (which reduces to the ML detector in the case when all symbols are equally likely) can be based on an additive version of the Viterbi algorithm with the simplified branch metric $\hat{\mu}_k(p,q) = |r_k - s^{(p,q)}|^2$. This last branch metric has a simple geometric interpretation as the squared Euclidean distance between the k-th observation and what that observation should have been, had the particular transition from state p to state q actually occurred. The Viterbi algorithm with this branch metric is also commonly known as the minimum-distance sequence detector, or the maximum-likelihood sequence detector (MLSD) when the noise is white and Gaussian. The solution is the message sequence that maximizes the likelihood $f(\mathbf{r}|\mathbf{a})$, or equivalently minimizes the energy in the error between the observation sequence r_k and the filtered message $a_k * h_k$.

Example: Suppose a sequence $\mathbf{a} = [a_0, \ldots a_4]$ of $L = 5$ binary symbols chosen independently and uniformly from the BPSK alphabet $\mathcal{A} = \{\pm 1\}$ is transmitted across an ISI channel with real-valued impulse response $H(z) = 3 + 2z^{-1} + z^{-2}$, which has memory $\mu = 2$, with a preamble and postamble each consisting of a pair of idle symbols $a^0 = -1$. In this example we illustrate how to use the Viterbi algorithm to find the ML decision sequence, given that the noisy channel output[2] (after additive white Gaussian noise) is $\mathbf{r} = [r_0, \ldots r_6]$ $= [1, 4, 1, 1, 5, 2, -4]$.

The state $\boldsymbol{\theta}_k = [a_{k-1}, a_{k-2}]$ can be $[-1,-1]$, $[+1,-1]$, $[-1,+1]$, or $[+1,+1]$, which are labeled by the integers 0, 1, 2, and 3, respectively. A transition from state $\boldsymbol{\theta}_k = [a_{k-1}, a_{k-2}]$ at time k to state

[2] In practice neither the channel ISI coefficients nor the noisy channel outputs will be integer-valued; nevertheless, we assume they are integers in this example to simplify the branch metric computations.

$\boldsymbol{\theta}_{k+1} = [a_k, a_{k-1}]$ at time $k+1$ uniquely determines the "expected" filter output $s_k = 3a_k + 2a_{k-1} + a_{k-2}$ at time k (see (11)). The expected filter outputs for all possible state transitions are summarized in Fig. 6(a). On the left of the figure is a set of four nodes, one for each possible value for the state $p \in \{0, 1, 2, 3\}$ at time k. On the right is another set four nodes, one for each possible state $q \in \{0, 1, 2, 3\}$ at time $k+1$. A branch from state p to state q indicates that a transition from state p to state q is possible. The label on each such branch in Fig. 6(a) is the corresponding expected output $s_k = 3a_k + 2a_{k-1} + a_{k-2}$ at time k.

The diagram in Fig. 6(a) can be used as an alternative to convolution for computing the ISI filter output in response to an input sequence. For example, consider the input sequence $\mathbf{a} = [a_0, \ldots a_4]$ $= [-1, +1, +1, -1, +1]$. The path for this sequence is highlighted in Fig. 5(a). Each branch in this path has an expected output, as indicated in Fig. 6(a), so that the expected ISI filter output in response to this input sequence can be read off from the branches in Fig. 6(a) as $\mathbf{s} = [s_0, \ldots s_6] = [-6, 0, 4, 0, 2, -2, 4]$.

Each message has its own unique path, and hence from Fig. 6(a), its own expected output sequence $\mathbf{s}$. The ML sequence detection problem for this AWGN channel boils down to finding the message $\mathbf{a}$ whose expected filter output $\mathbf{s}$ is closest to the observation $\mathbf{r} = [r_0, \ldots r_6]$, in the sense that it minimizes $\|\mathbf{r} - \mathbf{s}\|^2 = \sum_{k=0}^{6} |r_k - s_k|^2$. Let us label the branch from state p at time k to state q at time $k+1$ with the additive branch metric $|r_k - s^{(p,q)}|^2$ (which is equivalent to (16) given our assumption that the symbol distribution is uniform), where $s^{(p,q)}$ is the expected output associated with a transition from state p to state q, as summarized in Fig. 6(a). Then the ML cost $\|\mathbf{r} - \mathbf{s}\|^2$ for a particular message can be computed by summing the branch metrics in its corresponding path; this branch label thus reduces the ML sequence detection problem to the problem of finding a path through the trellis (from state 0 at time 0 to state 0 at time 7) whose path metric (equal to the sum of its branch metrics) is as small as possible.

The Viterbi algorithm efficiently finds the path through the trellis with minimum cost. For each node (p, k) in the trellis, the algorithm keeps track of both the survivor path to that node (which is the lowest-cost path to that node) and its corresponding cost, call it $\alpha_k(p)$. The algorithm is depicted in Fig. 6(b) through Fig. 6(h). Fig. 6(b) highlights stage $k = 0$ of the trellis, with the corresponding noisy observation $r_0 = 1$ written below it. The two branch metrics for this stage are computed by computing the square of the

difference between the *actual* observation ($r_0 = 1$) and the *expected* observation ($s^{(p,q)}$) for the transition, as indicated in Fig. 6(a). In particular, since $s^{(0,0)} = -6$ in Fig. 6(a), the upper branch metric in Fig. 6(b) is $|r_k - s^{(0,0)}|^2 = |1 - (-6)|^2 = 49$. Similarly, since $s^{(0,1)} = 0$ in Fig. 6(a), the lower branch metric in Fig. 6(b) is $|r_k - s^{(0,1)}|^2 = |1 - 0|^2 = 1$.

In the figure, the number written inside each node (p, k) is the survivor metric $\alpha_k(p)$ for that node's survivor; i.e., it is the cost of the lowest-cost partial path to that node. The cost at the beginning of the trellis (state 0 at time 0) is initialized to zero. At time 1 only two of the four states are reachable, with costs 49 and 1, as shown in Fig. 6(b).

In Fig. 6(c) we highlight stage $k = 1$ of the trellis, with the corresponding noisy observation $r_1 = 4$ written below. As before, the branches are labeled by the squared difference between the actual observation $r_1 = 4$ and the expected observation for the corresponding transition, as indicated in Fig. 6(a). For example, since $s^{(0,0)} = -6$ in Fig. 6(a), the upper branch metric in Fig. 6(c) is $|r_k - s^{(0,0)}|^2 = |4-(-6)|^2 = 100$, and since $s^{(1,2)} = -2$ in Fig. 6(a), the corresponding branch metric in Fig. 6(b) is $|r_k - s^{(1,2)}|^2 = |4 - (-2)|^2 = 36$. The survivor metrics are then computed and stored by adding the previous survivor metrics to the corresponding branch metrics.

It is not until stage $k = 2$ of the trellis, as highlighted in Fig. 6(d), that the pruning of paths begins. The noisy observation for this stage is $r_2 = 1$, which is written below. The branch metrics are labeled as before (by computing the square of the difference between r_2 and the expected outputs from Fig. 6(a)). Next the survivors for the states at time $k = 3$ are computed. For example, the figure shows that there are two ways to get to node $(0, 3)$:

- we could start at node $(0, 2)$, which has cost 149, and traverse the upper branch, which has cost 49, for a total cost of 198;
- we could start at node $(2, 2)$, which has cost 37, and traverse the lower branch, which as cost 25, for a total cost of 62.

The Viterbi algorithm selects the second option because of its lower cost. In the figure we indicate this selection by crossing out the branch that was not selected (the upper branch in this case). We further store the new metric by writing the cost 62 into node $(0, 3)$. A similar *add-compare-select* procedure is implemented for the remaining three nodes at time $k = 3$: the lower branch is selected for node $(1, 3)$, since $37 + 1 = 38 < 149 + 1 = 150$; the lower branch is selected for node $(2, 3)$, since $1 + 1 = 2 < 65 + 9 = 74$; and the lower

branch is selected for node $(3,3)$, since $1+25=26<65+9=74$.

The algorithm then moves on to stage $k=3$, as shown in Fig. 6(e), performing four add-compare-select operations, one for each possible state, and storing the new survivor metrics. The same operations are performed for stage 4, as shown in Fig. 6(f), and then again for stage 5, as shown in Fig. 6(g), and then again for the last stage, as shown in Fig. 6(h).

The decision path is the survivor path for the last node of the trellis. It can be found by starting at the ending node of the trellis and tracing backwards, at each stage choosing the branch that was not discarded (*i.e.*, not crossed out); the resulting decision path is highlighted in Fig. 6(h). The decision path in turn determines the decision message, since there is a one-to-one mapping between paths and messages. In particular, since the states were ordered so that the lower branch emanating from any node always corresponds to an input of $+1$, the fact that the decision path transitions are lower-lower-upper-lower-lower-upper-upper indicates that the ML decision sequence is $\hat{\mathbf{a}}=[+1,+1,-1,+1,+1]$. (The last two transitions correspond to the postamble and are not a part of the message decision.)

Observe that the survivor metric for the last node of the trellis is $\alpha_7(0)=8$. This is the smallest cost $\|\mathbf{r}-\mathbf{s}\|^2$ that is achieved by the decision sequence. Indeed, we can confirm this value by computing the expected output sequence from the decision sequence $\hat{\mathbf{a}}$ (via Fig. 6(a)) as $\mathbf{s}=[0,4,0,2,4,0,-4]$, and verifying that the squared distance to $\mathbf{r}=[1,4,1,1,5,2,-4]$ is indeed $\|\mathbf{r}-\mathbf{s}\|^2=8$.

We close this section with a summary of the key properties of the Viterbi algorithm, as described so far:

- it makes a decision about the entire sequence all at once, either the MAP or ML decision, depending on how the branch metrics are defined.
- It waits until the entire message has been transmitted before making its decision.
- it is built on a trellis diagram consisting of $L+\mu$ stages (in the horizontal dimension) and $Q=|\mathcal{A}|^\mu$ states (in the vertical dimension).
- The storage requirements for the survivor paths is roughly QL.
- The complexity of each stage is roughly proportional to the number of branches in each stage, namely $|\mathcal{A}|^{\mu+1}$, so that it requires a constant computation rate that is independent of the message length.

To reduce the storage requirements and delay, in practice the Viterbi algorithm is typically modified to operate over a window of say D stages of the trellis,

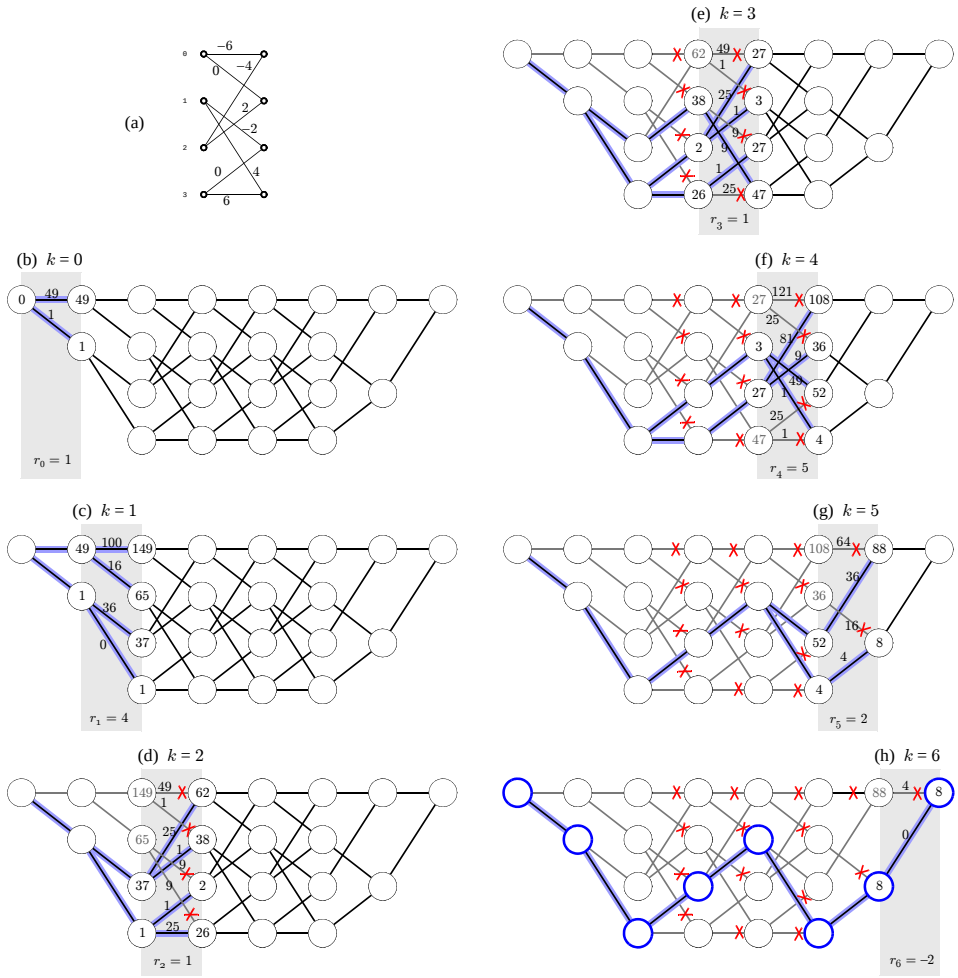

Figure 6: The Viterbi algorithm example: (a) shows the expected outputs $\{s^{(p,q)}\}$, while (b) through (h) show the branch metrics and add-compare-select results for stage 0 through stage 6, respectively.

from time $k - D$ to time k. At each time k, the survivor node with the best survivor metric is traced back to determine the decision at time $k - D$. This approach reduces the decoding delay from approximately L to D, and it reduces the storage requires from approximately QL to approximately QD. The window depth parameter D can then can then be tuned to trade-off performance versus complexity and delay. The optimal performance of the original Viterbi algorithm is achieved as D grows large, but small values are often sufficient; values of D on the order of 5μ are often sufficient to make the performance degradation negligible.

The Viterbi algorithm is an effective solution to the problem of estimating the inputs to a FSM based on a noisy observation of the FSM output, and is used in a wide variety of applications beyond equalization of ISI channels, including error-control decoding for convolutional and trellis codes, and demodulation of continuous-phase modulation [5].

3.3 APP Detection: The BCJR algorithm

The MAP decision found by the Viterbi algorithm of the previous section is optimal in the sense that it minimizes the probability that the message decision is incorrect. What could be better? Note that the MAP decision produced by the Viterbi algorithm is a *hard* decision about the *entire* message sequence. Here we describe the *Bahl-Cocke-Jelinek-Raviv (BCJR)* algorithm [1][18] that differs on both fronts:

- instead of making one decision about the *entire* message, the BCJR algorithm makes *separate* decisions about each bit of the message.
- instead of making *hard* decisions about the message bits, the BCJR algorithm makes *soft* decisions in the form of *a posteriori* probabilities.

The "soft" decisions of the BCJR algorithm are the *a posteriori probabilities (APP's)* $P(b_{k,i} = 1|\mathbf{r})$ for each message bit $b_{k,i}$, for $k \in \{0, \ldots, L-1\}$ and $i \in \{0, \ldots, \log_2 |\mathcal{A}| - 1\}$. The BCJR algorithm is thus an APP computer. If we were to quantize each bit's APP we would arrive at hard decisions that would at times disagree with those of the Viterbi algorithm. In fact, these quantized APP's would be optimal in the sense that they minimize the probability that each message bit is incorrect.

The fact that the quantized APP's from BCJR minimize the *bit*-error probability, as opposed to the *word*-error probability minimized by the Viterbi decision, is not what makes BCJR valuable. Indeed, in practice the performance difference between the two is often negligible. In other words, the fact that BCJR makes separate decisions about each bit is not what makes it valuable. Instead, the value of BCJR is the soft outputs (APP's) themselves, not their quantized versions. An equalizer that passes hard decisions to another receiver computation block (such as synchronization, estimation, or error-control decoding) is throwing away useful information. Overall performance can be improved

significantly when the equalizer instead passes soft decisions to these other receiver blocks. For example, an error-control decoder operating on soft decisions can significantly outperform a decoder operating on hard decisions. Extraordinary gains can be achieved when such blocks iteratively cooperate by sharing soft decisions according to the turbo principle [16]. Indeed, the BCJR algorithm is a key building block for turbo decoding of convolutional codes [3] and turbo equalization of ISI channels [16].

We now describe the BCJR algorithm, which is built on precisely the same trellis as the Viterbi algorithm. Let us associate with each node (p,k) in the trellis the *forward metric* $\alpha_k(p) = f(\theta_k = p, r_0^{k-1})$, which is within a scaling constant of being the *a posteriori* probability $P(\theta_k = p|r_0^{k-1})$ of being in that state p at that time k, given all of the "past" observations $r_0^{k-1} = \{r_0, r_1, \ldots, r_{k-1}\}$. These metrics can be computed recursively, *i.e.* knowledge of the Q metrics $\alpha_k(0)$ through $\alpha_k(Q-1)$ at time k can be used to compute the Q metrics $\alpha_{k+1}(0)$ through $\alpha_{k+1}(Q-1)$ at time $k+1$, according to:

$$\alpha_{k+1}(q) = \sum_{p=0}^{Q-1} \alpha_k(p)\gamma_k(p,q), \qquad \text{for } q \in \{0,1,\ldots Q-1\} \tag{17}$$

where again $\gamma_k(p,q)$ denotes the branch metric from state p at time k to state q at time $k+1$, and is precisely the same as the branch metric for the Viterbi algorithm, namely $\gamma_k(p,q) = f(r_k|s^{(p,q)})P(a^{(p,q)})$. The forward recursion of (17) is initialized in the same way as the Viterbi algorithm, namely $\alpha_0(p) = \delta_p$, to account for the fact that the state is initially $p = 0$ at time 0.

Comparing the recursion (17) to the Viterbi recursion in (13) we see that they are nearly the same; the only difference is that the maximum operation $\max_p$ of (13) has been replaced by a sum operation $\sum_p$ in (17). This difference means that, while only the largest term contributes in the Viterbi recursion, *all* terms contribute in the BCJR recursion. The difference is not always significant, however, because at high SNR it is common for one term to dominate anyway. We can thus view the Viterbi survivor metric $\hat{\alpha}_k(p)$ in (13) as a high-SNR approximation of the BCJR metric $\alpha_k(p)$ in (17).

Similarly, let us associate with each node (p,k) in the trellis the *backward* metric $\beta_k(p) = f(r_k^{L+\mu-1}|\theta_k = p)$, which is a function of the "future" observations $r_k^{L+\mu-1} = \{r_k, \ldots, r_{L+\mu-1}\}$. These metrics can also be computed recursively, using the same type of recursion as in in (17), except that it starts at the end of the trellis and moves backwards, so that the backward metrics at time k can be computed from those at time $k+1$ according to:

$$\beta_k(p) = \sum_{q=0}^{Q-1} \gamma_k(p,q)\beta_{k+1}(q), \quad \text{for } p \in \{0,1,\ldots Q-1\} \tag{18}$$

where again $\gamma_k(p,q) = f(r_k|s^{(p,q)})P(a^{(p,q)})$. The backward recursion of (18) is initialized by $\beta_{L+\mu}(q) = \delta_q$, to account for the fact that the ending state at time $k = L+\mu$ is fixed at $q = 0$.

Key to the BCJR algorithm is the fact that the *a posteriori* probability of a state transition, say a transition from state p at time k to state q at time $k+1$, can be expressed in terms of the forward, backward, and branch metrics as:

$$P(\theta_k = p, \theta_{k+1} = q|\mathbf{r}) = \alpha_k(p)\gamma_k(p,q)\beta_{k+1}(q)/f(\mathbf{r}). \tag{19}$$

Therefore, we can compute the *a posteriori* probability for a particular bit, say the i-th bit, of the k-th symbol by summing over all state transitions for which the i-th bit is a one:

$$P(b_{k,i} = 1|\mathbf{r}) = \frac{1}{f(\mathbf{r})} \sum_{(p,q)\in\mathcal{B}_i} \alpha_k(p)\gamma_k(p,q)\beta_{k+1}(q), \tag{20}$$

where $\mathcal{B}_i$ denotes the subset of branches $\{(p,q) : p,q \in \{0,\ldots,Q-1\}\}$ for which the i-th bit of the symbol associated with that transition is one.

Rather than compute (20) directly, it is much more convenient (and hence much more common) to compute the so-called "L" values, which are defined as the logarithm of the ratio of the a posteriori probabilities [6]:

$$L_{k,i} = \log(\frac{P(b_{k,i} = 1|\mathbf{r})}{P(b_{k,i} = 0|\mathbf{r})}). \tag{21}$$

These L values are the soft information provided by the BCJR algorithm. They have several useful properties:

- the sign of the L values determines the optimal MAP decision (i.e., the decision that minimizes the probability of being incorrect), according to $\hat{b}_{k,i}^{\text{MAP}} = 1_{L_{k,i}>0}$.
- a zero value ($L_{k,i} = 0$) indicates complete uncertainty in the value of the corresponding bit; *i.e.*, that $b_{k,i}$ is equally likely to be a 0 and a 1.
- more generally, the magnitude $|L_{k,i}|$ is a measure of certainty regarding the hard decision; small magnitudes indicate a small amount of confidence that the hard decision is correct, while large magnitudes indicate a high amount of confidence that the hard decision is correct.
- the a posteriori bit probability in (20) can be recovered from the L value via $P(b_{k,i} = 1|\mathbf{r}) = 1/(1 + e^{-L_{k,i}})$.

Substituting (20) and its complement into (21) yields the following expression for the L values, expressed in terms of the α, β, and γ parameters:

$$L_{k,i} = \log(\frac{\sum_{(p,q)\in\mathcal{B}_i} \alpha_k(p)\gamma_k(p,q)\beta_{k+1}(q)}{\sum_{(p,q)\in\bar{\mathcal{B}}_i} \alpha_k(p)\gamma_k(p,q)\beta_{k+1}(q)}). \tag{22}$$

The set $\bar{\mathcal{B}}_i$ in the denominator denotes the complement of $\mathcal{B}_i$, i.e., the set of branches corresponding to a zero bit. The scaling factor $1/f(\mathbf{r})$ in (19) and (20) can be ignored because it is common to both the numerator and the denominator in (22), so it will cancel.

We can now summarize the BCJR algorithm for APP computation after an ISI channel:

1. Compute the forward metrics $\{\alpha_k(p)\}$ recursively using the forward recursion (17), moving through the trellis from left to right.

2. Compute the reverse metrics $\{\beta_k(p)\}$ recursively using the backward recursion (18), moving through the trellis from right to left.

3. Use (22) to compute the APP values for each bit of the message.

Each pass through the trellis has complexity roughly comparable to that of the Viterbi algorithm, making the BCJR algorithm roughly twice as complex.[3] The computation of the APP L values has a further cost in complexity.

Example: Suppose a sequence of symbols chosen from the 4-QAM alphabet $\mathcal{A} = \{\pm 1 \pm j\}$ are transmitted over the ISI channel $H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2}$, where $h_0 = 1 + 0.3j$, $h_1 = 0.2 + 0.7j$, and $h_2 = 0.05 - 0.1j$, and with noise power $N_0 = 0.18$, so that $SNR = E_a E_h / N_0 = 12.6$ dB. The message bits are i.i.d. uniform over $\{0, 1\}$. A block diagram is shown in Fig. 7, along with two empirically measured constellations: one for the output r_k of the noisy channel, and another for the outputs $L_{k,i}$ of the BCJR algorithm. The ISI in this example is severe enough to cause significant overlap of neighboring clouds. Some sort of equalization is clearly needed to reliably recover the transmitted message. The trellis diagram for this example is exactly as shown in Fig. 5(b). In particular, since the alphabet size is $|\mathcal{A}| = 4$ and the channel memory is $\mu = 2$, the number of states is $Q = |\mathcal{A}|^\mu = 16$. Also shown in the figure is a constellation for the fictitious signal $L_{k,1} + jL_{k,2}$ after the BCJR algorithm. We see that it resembles a 4-QAM constellation with no ISI, and with circularly symmetric Gaussian noise. The conditional distribution after the BCJR is approximately consistent Gaussian, meaning that the variance in each dimension is twice the conditional mean. Measuring the SNR after the BCJR algorithm yields SNR = 12.6 dB, which is approximately the same as the underlying SNR. Evidently, in this example, the BCJR algorithm is able to eliminate the effects of the ISI without an appreciable penalty in SNR. This conclusion can be confirmed in terms of capacity [10].

4 Linear Equalization

The trellis-based detectors of the previous section perform exceptionally well, but their complexity can be high. In particular, the complexity of both Viterbi

[3] Although the traceback operation of the Viterbi algorithm can be viewed as a backward pass through the trellis, analogous to the backward recursion of BCJR, it does not require any computations.

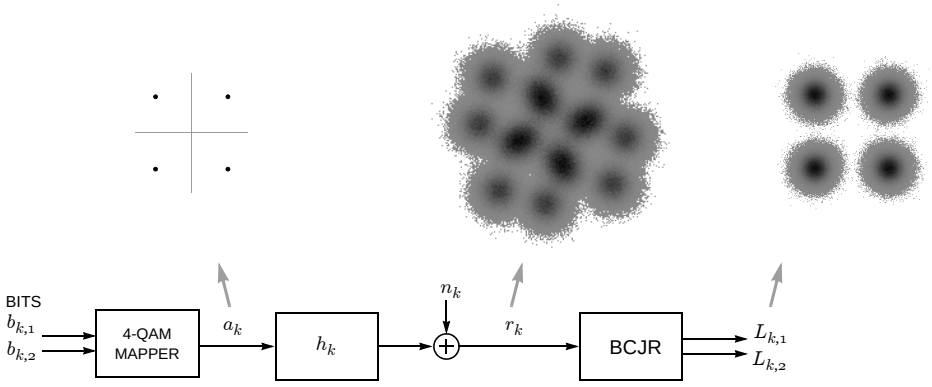

Figure 7: An example of equalization via BCJR for 4-QAM over the ISI channel $H(z) = (1 + 0.3j) + (0.2 + 0.7j)z^{-1} + (0.05 - 0.1j)z^{-2}$, with SNR = 12.6 dB.

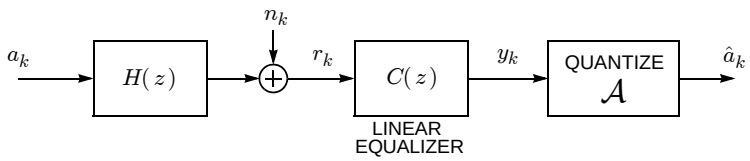

Figure 8: A linear equalizer $C(z)$ followed by a memoryless quantizer results in decisions $\hat{a}_k$ about the message symbols a_k.

and BCJR algorithms is exponential in both the spectral efficiency ($\log_2 |\mathcal{A}|$) and channel memory (μ), and can be prohibitive when the alphabet is large, the channel memory is large, or both. As an alternative we consider in this section the class of *linear* detectors, which feature a complexity that is nearly independent of the alphabet size and channel memory, albeit with performance that can at times fall significantly short of the trellis-based detectors.

A linear equalizer is simply a linear time-invariant filter applied to the channel output. In this section we assume that the linear equalizer is designed with the expectation that the equalizer output will be applied to a *memoryless* quantizer that rounds the equalizer output to the nearest element of the symbol alphabet $\mathcal{A}$. In other words, the aim of the linear equalizer is to eliminate ISI, so that subsequent receiver processing can ignore ISI.[4]

A block diagram of a linear equalizer is shown in Fig. 8. In the following three sections we describe three special cases of the linear equalizer: (1) the matched filter, (2) the zero-forcing (ZF) linear equalizer, and (3) the minimum-mean-squared-error (MMSE) linear equalizer.

[4]This is in contrast to a partial-response equalizer, which aims not to eliminate ISI but to mold it to match a desired target [4].

4.1 The Matched Filter

If the transmitter were to send only a single symbol in isolation, so that $L = 1$ in (1) and the channel model (3) reduces to $r_k = a_0 h_k + n_k$, then the SNR after equalization (at time zero) is maximized when the linear equalizer $C(z)$ in Fig. 8 is *matched* to the channel. In the time domain, the matched filter impulse response is a time-reversed and conjugated version of the ISI channel:

$$c_k = h^*_{-k}.$$

In the z domain[5] this means that $C(z) = H^*(1/z^*)$, while in the frequency domain it means that $C(e^{j\theta}) = H^*(e^{j\theta})$. The MF equalizer gain at a particular frequency matches that of the channel at that frequency; frequencies that are amplified by the channel will be further and similarly amplified by the equalizer, while frequencies that are attenuated by the channel will be further and similarly attenuated by the equalizer.

In practice the transmitter will send a sequence of symbols, not a single symbol in isolation. In this case the MF receiver may not at first glance appear to be an equalization strategy at all, since the net effect of the matched filter will be to *accentuate* the severity of the ISI, rather than to eliminate it. However, as we will see below, the matched filter is an optimal linear equalizer in the limit of low SNR. Intuitively, this can be understood by observing that the ISI, no matter how severe, will eventually become negligible (falling below the noise floor) at low enough SNR. Furthermore, in a direct-sequence code-division-multiple-access (DS-CDMA) application for which the transmit pulse shape $g(t)$ of (1) has a bandwidth that is orders of magnitude greater that the signaling rate $1/T$, an oversampled version[6] of the matched filter is known as the RAKE receiver and is an effective method for compensating for the effects of a dispersive channel, at any SNR.

4.2 Zero-Forcing Linear Equalization

The *zero-forcing* strategy for linear equalizer design is to choose the equalizer to be the inverse of the channel, when it exists, which will completely eliminate the ISI:

$$C_{ZF}(z) = \frac{1}{H(z)}. \tag{23}$$

This is called a zero-forcing equalizer because the ISI is forced to zero. The ZF linear equalizer is in some sense the opposite of the matched filter: It attenuates

[5] The z transform of a sequence x_k is a mapping from the complex z plane to $X(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k}$, for those values of z for which the sum converges. Evaluating the z transform at $z = e^{j\theta}$ results in the Fourier transform $X(e^{j\theta})$, when it exists.

[6] The broadband nature of a DS-CDMA signal violates the minimum-bandwidth assumption made in Sect. 2.1, and hence the baud-rate sampled model considered in this chapter would need to be replaced by an oversampled model in order to accommodate the expanded bandwidth.

at frequencies that are amplified by the channel, and it amplifies at frequencies that are attenuated by the channel.

Because the ZF linear equalizer completely eliminates ISI, its output can be written as:

$$y_k = a_k + e_k,$$

where $e_k = n_k * c_k$ is the noise after being filtered by the equalizer. The PSD of the filtered noise is:

$$S_e(e^{j\theta}) = N_0|C(e^{j\theta})|^2 = \frac{N_0}{|H(e^{j\theta})|^2}.$$

The power of the filtered noise is $E(|e_k|^2) = \mathcal{A}\{S_e(e^{j\theta})\}$. Therefore, the SNR after the ZF equalizer, as seen by the memoryless quantizer, is simply:

$$SNR_{\text{ZF}} = \frac{E(|a_k|^2)}{E(|e_k|^2)} = \frac{E_a}{\mathcal{A}\{S_e(e^{j\theta})\}} = \frac{1}{\mathcal{A}\{\frac{1}{SNR(\theta)}\}}, \tag{24}$$

or more compactly,

$$SNR_{\text{ZF}} = \mathcal{H}\{SNR(\theta)\}. \tag{25}$$

This is the post-equalization SNR of the ZF linear equalizer. Note that this SNR does not account for the correlation in the noise after the linear equalizer, because the memoryless quantizer has no way of exploiting this correlation. Thus, the SNR measure implicitly takes the memoryless constraint into account. (In contrast, an optimal way of exploiting the noise correlation would be to follow the equalizer by the inverse of the equalizer (!), which reverts back to to the original channel output, and then to apply a trellis-based detector.)

The ZF linear equalizer performance is never better than the matched filter bound when the transmit spectrum is flat, since in this case the inequalities in (4) imply that $SNR_{\text{ZF}} = \mathcal{H}\{SNR(\theta)\} \leq \mathcal{A}\{SNR(\theta)\} = SNR_{\text{MFB}}$. The inequality becomes an equality if and only if there is no ISI, *i.e.*, if and only if $SNR(\theta)$ is independent of θ. Therefore, for any channel with ISI, the ZF linear equalizer always falls short of the MFB. The nature of the harmonic mean implies that the gap in performance can be significant when the channel frequency response is small at one or more frequencies.

Example: Consider the performance of the ZF linear equalizer for a sequence of i.i.d. symbols chosen from the 4-QAM alphabet $\mathcal{A} = \{\pm 1 \pm j\}$ over a channel with frequency response $H(z) = 1 + bz^{-1}$ and noise PSD $N_0 = 2$; in this case the SNR spectral density reduces to $SNR(\theta) = |1 + be^{-j\theta}|^2$, so that the SNR after a ZF linear equalizer is:

$$\begin{aligned} SNR_{\text{ZF}} &= \mathcal{H}\{SNR(\theta)\} \\ &= \frac{1}{\frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{|1+be^{-j\theta}|^2} d\theta} \\ &= 1 - |b|^2. \end{aligned}$$

If $|b|$ is infinitesimal, this SNR is not much smaller than the MFB, namely $SNR_{\text{MFB}} = 1 + |b|^2$. However, as $|b|$ grows large, the SNR decreases dramatically. In fact, as $|b|$ approaches unity, the SNR approaches zero! The ZF linear equalizer thus incurs an infinite SNR penalty when $|b| = 1$. The reason can be easily understood in the frequency domain, for the channel magnitude response $|H(e^{j\theta})| = |1 + be^{-j\theta}|$ has a *spectral null* at some frequency θ (that depends on the angle of b) when $|b| = 1$. In turn, this implies that the gain of the equalizer $|C(e^{j\theta})| = \frac{1}{|H(e^{j\theta})|}$ grows to infinity at that same frequency. This amplifies the noise by an infinite amount, so that the noise totally swamps the desired signal.

It is generally true that a ZF linear equalizer suffers infinite noise enhancement whenever the channel magnitude response is zero at one or more frequencies, and that it performs poorly whenever the channel has a near spectral null at one or more frequencies.

4.3 MMSE Linear Equalization

The fatal flaw of the zero-forcing equalizer is its insistence on forcing ISI to zero, regardless of what impact it has on the noise. Forcing ISI to zero is overkill. In contrast, here we describe the *minimum-mean-squared-error (MMSE)* equalizer, which chooses its coefficients so as to minimize the mean-squared error between the equalizer output and what we want it to be, namely to minimize $MSE = E(|e_k|^2)$, where $e_k = y_k - a_k$. The Fourier transform of the solution is

$$C_{MMSE}(e^{j\theta}) = \frac{H^*(e^{j\theta})}{|H(e^{j\theta})|^2 + \frac{N_0}{E_a}}. \tag{26}$$

The numerator represents a matched filter, which transforms the channel frequency response from its original form $H(e^{j\theta})$ to $|H(e^{j\theta})|^2$. The presence of the constant N_0/E_a term in the denominator is the only thing that prevents the remainder of the MMSE equalizer from inverting this effective response. As such, this constant is the only thing that prevents the MMSE linear equalizer from being the ZF linear equalizer. Intuitively we can think of the constant as a way of preventing the denominator from being close to zero, even when the channel itself is zero or near zero at certain frequencies.

Two extremes are of special interest: high SNR and low SNR. At high SNR, high enough that the N_0/E_a term in the denominator is negligible, the MMSE equalizer reduces to the ZF equalizer $C_{ZF}(e^{j\theta}) = 1/H(e^{j\theta})$ of (23). This makes intuitive sense, because at high enough SNR any residual ISI after the equalizer will eventually dominate. At the other extreme of low SNR, so low that the N_0/E_a term in the denominator dominates, the MMSE equalizer reduces to within a constant scaling factor of the matched filter $H^*(e^{j\theta})$. This also makes sense intuitively, because at low enough SNR the ISI will eventually fall below the noise floor and will become negligible.

The two extreme cases considered above help to clarify the tradeoff achieved by the MMSE solution. The error after the equalizer will have two components, one being residual ISI and the other being filtered noise. The aim of the MMSE equalizer is to minimize the sum of the power of both. In stark contrast, the ZF equalizer minimizes ISI while *ignoring* the noise. Similarly, the matched filter can be thought of as an equalizer that maximizes SNR while *ignoring* the ISI. Neither of these extremes achieves the optimal balance obtained by the MMSE equalizer. One final intuitive view of how the MMSE equalizer compares to the ZF equalizer: While the ZF equalizer forces ISI to zero at the expense of a potentially large enhancement of the noise power, the MMSE equalizer merely pushes the ISI to be roughly below the noise floor, without as much noise enhancement.

In order to accurately quantify the post-equalization SNR of the MMSE equalizer, we need to account for the equalizer bias. Let $\beta_k = h_k * c_k$ denote the impulse response of the cascade of the channel and any equalizer c_k. We say that an equalizer is *biased* whenever the zero-th coefficient β_0 satisfies $\beta_0 \neq 1$, because the conditional mean of the equalizer output satisfies $E(y_k|a_k) = \beta_0 a_k$. For example, the ZF equalizer of the previous section is unbiased, because in that case $\beta_0 = 1$. On the other hand, the MMSE equalizer is biased because $\beta_0 \neq 1$, namely:

$$\beta_0 = \frac{1}{1 + 1/SNR_{\text{MMSE-LE}}},$$

where $SNR_{\text{MMSE-LE}}$ will be defined shortly (see (28)). Scaling the equalizer by $1/\beta_0$ removes the bias, leading to the so-called *unbiased* MMSE linear equalizer:

$$C_{\text{MMSE,U}}(e^{j\theta}) = \frac{\beta_0^{-1} H^*(e^{j\theta})}{|H(e^{j\theta})|^2 + \frac{N_0}{E_a}}. \tag{27}$$

At high SNR there is not much difference in performance between the unbiased and biased equalizer. However, the unbiased version makes it easy to compute the post-equalization SNR of the MMSE equalizer, because – unlike for the biased case – the equalizer error $e_k = y_k - a_k$ after the unbiased equalizer will be *independent* of a_k, so that the SNR after this equalizer can be computed simply as:

$$\begin{aligned} SNR_{\text{MMSE-LE}} &= \frac{E(|a_k|^2)}{E(|e_k|^2)} && (28) \\ &= \mathcal{H}\{1 + SNR(\theta)\} - 1. && (29) \end{aligned}$$

Comparing this SNR to that of the ZF linear equalizer, and exploiting the inequality (4), we conclude that:

$$SNR_{\text{MMSE-LE}} \geq SNR_{\text{ZF-LE}},$$

with equality if and only if the channel has no ISI. Thus, when faced with ISI, the MMSE linear equalizer always outperforms the ZF linear equalizer. The

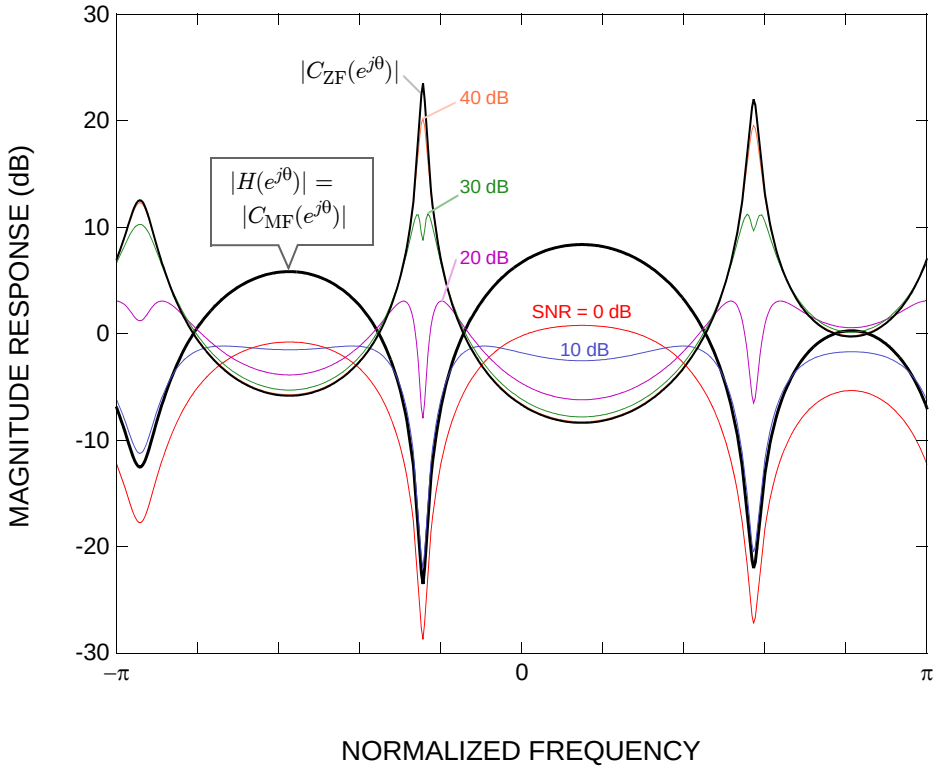

Figure 9: An illustration of how the magnitude response of the MMSE linear equalizer ranges from that of a matched filter to that of a ZF linear equalizer, depending on the SNR.

difference in performance can be significant at low SNR, or for channels with severe ISI, while the difference can be small at high SNR, or for channels with mild ISI.

Example: Consider a channel with transfer function $H(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$, where $h_1 = 0.4 - 0.1j$, $h_2 = 0.1 + 0.5j$, and $h_3 = 0.3 + 0.8j$. The magnitude response $|H(e^{j\theta})|$ is shown in Fig. 9, where we see deep spectral nulls that are attenuated by as much as 32 dB relative to the peak response. Since the matched filter and the channel share the same magnitude response, this same curve also depicts the magnitude response of the matched filter. Also shown in the figure is the magnitude response $|C_{ZF}(e^{j\theta})| = 1/|H(e^{j\theta})|$ of the zero-forcing linear equalizer. Observe that the ZF linear equalizer has a high gain of over 20 dB at the frequency most attenuated by the channel. The remaining five curves show the magnitude response of the unbiased MMSE linear equalizer for five different SNR values,

ranging from 0 dB to 40 dB. At low SNR, the MMSE equalizer has roughly the same shape as the matched filter. At high SNR, the MMSE equalizer closely approximates the ZF linear equalizer. At moderate SNR, the MMSE linear equalizer resembles neither the MF nor the ZF linear equalizer over the entire frequency range. Instead, at 20 dB SNR, for example, we see that the MMSE linear equalizer behaves like the MF at frequencies near the channel null, while it behaves more like the ZF linear equalizer at all other frequencies.

5 Decision-Feedback Equalization

The decision-feedback equalizer is a nonlinear equalization strategy that can significantly outperform a linear equalizer, with comparable complexity. The DFE is based on the concept of *interference cancellation*, where interference is estimated at the receiver and subtracted. As a motivating example, let us begin our discussion by supposing that the channel impulse response has most of its energy concentrated in the zero-th coefficient h_0, so that the channel output $r_k = a_k * h_k + n_k$ can be broken down into three terms:

$$r_k = \underbrace{h_0 a_k}_{\text{desired}} + \underbrace{\sum_{i=1}^{\mu} h_i a_{k-i}}_{\text{ISI}} + \underbrace{n_k}_{\text{noise}}.$$

The first term represents the desired signal, the second term is the ISI, and the third term is the channel noise. Suppose further that, at time k, the receiver has access to the prior decisions $\{\hat{a}_{k-1}, \hat{a}_{k-2}, \ldots\}$. In this case, the receiver can reconstruct an estimate $\sum_{i=1}^{\mu} h_i \hat{a}_{k-i}$ of the ISI and subtract this estimate from the channel output, and further scale by $1/h_0$, resulting in:

$$\begin{aligned} z_k &= \frac{1}{h_0}(r_k - \sum_{i-1}^{\mu} h_i \hat{a}_{k-i}) \qquad (30) \\ &= a_k + \frac{1}{h_0}\sum_{i=1}^{\mu} h_i (a_{k-i} - \hat{a}_{k-i}) + n_k/h_0. \end{aligned}$$

When the μ relevant decisions are correct, this reduces to:

$$z_k = a_k + n_k/h_0.$$

Like the zero-forcing linear equalizer, this DFE has completely eliminated ISI. Unlike the linear equalizer, however, the noise here has not been enhanced. Instead, the noise term n_k in the above equation is the noise n_k of the original channel. This DFE thus eliminates ISI without any noise enhancement.

In the general case when h_0 is not large, the receiver can first apply the channel output to a linear "forward" filter whose purpose is to transform the

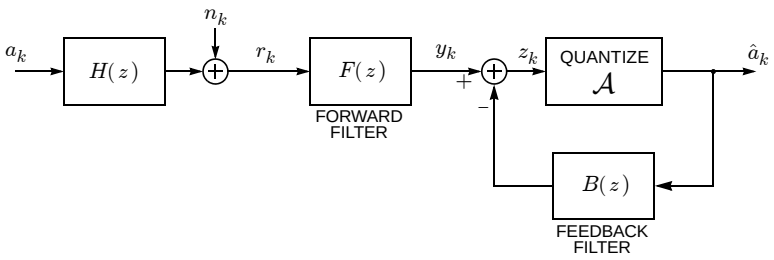

Figure 10: The DFE has two filters, a forward filter for mitigating ISI from future symbols, and a feedback filter for mitigating ISI from past symbols.

cascade of the channel and the equalizer into an effective channel whose zero-th tap *is* large. This leads to the DFE structure shown in Fig. 10. The equalizer output (or equivalently, the input to the decision device) for the DFE is:

$$z_k = r_k * f_k - \sum_{i=1}^{\infty} b_i \hat{a}_{k-i}, \tag{31}$$

where f_k is the impulse response of the forward filter, and where b_k is the impulse response of the feedback filter. To be implementable, the feedback filter must be *strictly* causal, i.e., it must satisfy $b_k = 0$ for all $k \in \{0, -1, -2, \ldots\}$, so that $B(z) = \sum_{k=1}^{\infty} b_k z^{-k}$. The motivating example of the previous paragraph is a special case when the forward filter is the scaling factor $F(z) = 1/h_0$, and the feedback filter is the strictly causal "tail" of the normalized channel, $B(z) = (H(z) - h_0)/h_0 = h_0^{-1} \sum_{k=1}^{\mu} h_k z^{-k}$.

5.1 The Zero-Forcing DFE

A zero-forcing DFE is one for which the DFE output z_k depends only on the k-th symbol a_k, with no residual ISI from the others. Of the many DFE's that satisfy this ZF property, we will prefer the one that results in the highest SNR, or equivalently that minimizes the $MSE = E(|z_k - a_k|^2)$. This version of the ZF DFE is unique, and is given by [2]:

$$F(z) = \frac{H^*(1/z^*)}{\gamma^2 M^*(1/z^*)}, \tag{32}$$

$$B(z) = M(z) - 1, \tag{33}$$

which are defined in terms of the following spectral factorization of $H(z)H^*(1/z^*)$:

$$H(z)H^*(1/z^*) = \gamma^2 M(z) M^*(1/z^*), \tag{34}$$

where $M(z) = 1 + m_1 z^{-1} + m_2 z^{-2} \ldots$ is monic (i.e., satisfying $m_0 = 1$), loosely minimum phase (i.e., all of its poles are inside the unit circle, and all of its zeros are either inside or on the unit circle), and where:

$$\gamma^2 = \mathcal{G}\{|H(e^{j\theta})|^2\}$$

is the geometric mean of the magnitude squared of the channel frequency response. With this choice for the forward filter, it is easy to see that the cascade of the channel and the forward filter reduces to $H(z)F(z) = M(z)$. Thus, the forward filter transforms the original channel with transfer function $H(z)$ into an effective channel with transfer function $M(z)$. Of all transfer functions having the same magnitude response as the original channel, the minimum-phase factor $\gamma M(z)$ has the desirable property that its energy is maximally concentrated at time zero. Observe further that the optimal ZF forward filter is a scaled all-pass filter, which ensures that it does not enhance the noise; the noise after the forward filter is still white. One final observation: If the original channel happens to be minimum phase from the beginning, then the minimum-phase factor reduces to $\gamma M(z) = H(z)$. This means that the forward filter reduces to a scaling constant $F(z) = 1/\gamma = 1/h_0$, and the feedback filter reduces to $B(z) = H(z)/h_0 - 1$, so that the optimal ZF DFE reduces to the original motivating example provided at the beginning of this section.

To compute the SNR of the ZF DFE, observe that the net effect of the forward filter $F(z)$ from (32) is to transform the channel from $H(z)$ to the channel-filter cascade $H(z)F(z) = M(z)$. Because the forward filter is a scaled all-pass filter, the noise after the forward filter is still white and Gaussian. The scaling constant changes the noise PSD from N_0 to N_0/γ^2. The feedback filter eliminates the ISI without any impact on this noise, so that the SNR after the ZF DFE, assuming correct decisions, is simply:

$$SNR_{\mathrm{ZF-DFE}} = \frac{E_a}{N_0/\gamma^2} = \frac{E_a}{N_0}\mathcal{G}\{|H(e^{j\theta})|^2\} = \mathcal{G}\{SNR(\theta)\}. \tag{35}$$

(The last equality requires that the transmit spectrum be flat, $S_a(e^{j\theta}) = E_a$.) This is the post-equalization SNR for the ZF DFE. Exploiting the inequality (4), we conclude that:

$$SNR_{\mathrm{ZF-DFE}} \geq SNR_{\mathrm{ZF-LE}},$$

with equality if and only if the channel has no ISI. The DFE thus always outperforms the linear equalizer for channels with ISI. One caveat to keep in mind for the DFE is that the above SNR measure is based on an assumption that decisions are correct; the actual SNR (that accounts for occasional errors) will be slightly lower. The impact of errors is considered next.

5.2 Error Propagation

The DFE was designed and analyzed in the previous section with an assumption that the receiver decisions are correct. In practice there will be occasional errors

that tend to increase the probability of a subsequent error, a phenomenon known as *error propagation*. A single error will often lead to a burst of many errors. Looking closer at (30), we see that any error amongst $\{\hat{a}_{k-1}, \ldots, \hat{a}_{k-\mu}\}$ will lead to the corresponding ISI being *added* (in effect, doubled) instead of subtracted, which will in turn significantly increase the probability of a subsequent error. The error burst effect ends whenever μ consecutive correct decisions are made. The impact of error propagation on the overall performance depends on the value of SNR: At high SNR the impact can be negligible, while at low SNR the impact can be more severe.

Fortunately we can analyze the impact of error propagation in a straightforward way, and quantify the overall performance that takes into account the possibility of error propagation. We illustrate this analysis with a simple example.

Example: Consider the channel $r_k = a_k + a_{k-1} + n_k$, where the symbols are chosen uniformly and independently from the BPSK alphabet $\mathcal{A} = \{\pm 1\}$, and where the independent noise is white with variance $\sigma^2 = N_0/2$. The forward filter of the ZF DFE in this case is the identity, $F(z) = 1$, while the feedback filter is the impulse response tail, namely $B(z) = z^{-1}$. The k-th decision is then $\hat{a}_k = \text{sign}(z_k)$. Let $p_0 = P(\hat{a}_k \neq a_k | \hat{a}_{k-1} = a_{k-1})$ denote the conditional probability of error for the k-th decision, given that the previous decision was correct. Under this condition, (31) reduces to $z_k = a_k + n_k$, from which we conclude that $p_0 = Q(1/\sigma)$. Similarly, let $p_1 = P(\hat{a}_k \neq a_k | \hat{a}_{k-1} \neq a_{k-1})$ denote the conditional probability of error for the k-th decision, given that the previous decision was incorrect. Under this condition, (31) reduces to $z_k = a_k + a_{k-1} - \hat{a}_{k-1} + n_k = a_k + 2a_{k-1} + n_k$. The doubling of the ISI in this case is clearly evident. From this equation we conclude that the conditional error probability for $\hat{a}_k = \text{sign}(z_k)$ is $p_1 = \frac{1}{2}(1 - Q(1/\sigma) + Q(3/\sigma))$. From the law of total probability, the overall probability of error $P_e = P(\hat{a}_k \neq a_k)$ can then be written as:

$$\begin{aligned} P_e &= P(\hat{a}_k \neq a_k | \hat{a}_{k-1} = a_{k-1}) P(\hat{a}_{k-1} = a_{k-1}) \\ &\quad + P(\hat{a}_k \neq a_k | \hat{a}_{k-1} \neq a_{k-1}) P(\hat{a}_{k-1} \neq a_{k-1}) \\ &= p_0(1 - P_e) + p_1 P_e. \end{aligned}$$

Solving this equation for P_e yields:

$$P_e = \frac{p_0}{1 + p_0 - p_1} = \frac{2Q(1/\sigma)}{1 + 3Q(1/\sigma) - Q(3/\sigma)}.$$

The denominator quickly approaches unity as SNR increases, so the dominant impact of error propagation can be seen from the numerator: Error propagation approximately doubles the overall

error probability in this example. This result is plotted in Fig. 11 versus SNR (upper curve), along with the performance of the ideal DFE (lower curve) that somehow has access to correct decisions when canceling the ISI. While it is true that the error propagation effectively doubles the error probability at moderate to high SNR, which may sound severe, the actual degradation when measuring the horizontal distance between the two curves is small. This thanks to the fact that the Q function decays rapidly. In fact, the horizontal distance between the two curves is only 0.17 dB at $P_e = 10^{-7}$, implying that the SNR penalty due to error propagation is only 0.17 dB at this probability.

The broad conclusions from the above example generalize to a wide range of alphabets and ISI responses: The impact of error propagation is rarely if ever catastrophic. Instead it results in a modest SNR penalty when compared to the ideal DFE, a penalty generally small enough that even with error propagation the DFE will outperform the linear equalizer. (We will see another example in Sect. 7 where the error propagation penalty is about 0.5 dB.)

5.3 The Minimum-MSE DFE

The DFE output z_k will generally contain both ISI and noise. The ZF DFE described in the previous section forces the ISI to zero, without regard to the impact on the noise. In this section we describe the MMSE DFE, which chooses the forward and feedback filters to minimize MSE instead, thus accounting for both ISI and noise. In particular, the MMSE DFE chooses $F(z)$ and $B(z)$ to minimize $MSE = E(|z_k - a_k|^2)$, where again the DFE output z_k is given by (31). The MMSE solution is [2]:

$$F(z) = \frac{H^*(1/z^*)}{\tilde{\gamma}^2 \tilde{M}^*(1/z^*)}, \tag{36}$$

$$B(z) = \tilde{M}(z) - 1, \tag{37}$$

where $\tilde{\gamma}$ and $\tilde{M}(z)$ are defined by the following spectral factorization:

$$H(z)H^*(1/z^*) + \frac{N_0}{E_a} = \tilde{\gamma}^2 \tilde{M}(z)\tilde{M}^*(1/z^*), \tag{38}$$

where again $\tilde{M}(z)$ is monic and minimum phase. Like the linear MMSE equalizer, this MMSE DFE is biased; removing the bias means scaling the forward filter by $1 + 1/SNR_{\text{MMSE-DFE}}$. The resulting SNR of the MMSE DFE is:

$$SNR_{\text{MMSE-DFE}} = G\{1 + SNR(\theta)\} - 1. \tag{39}$$

In light of the inequalities in (4) we conclude that, for any channel with ISI, the MMSE DFE will always outperform the ZF DFE, the MMSE LE, and the ZF LE.

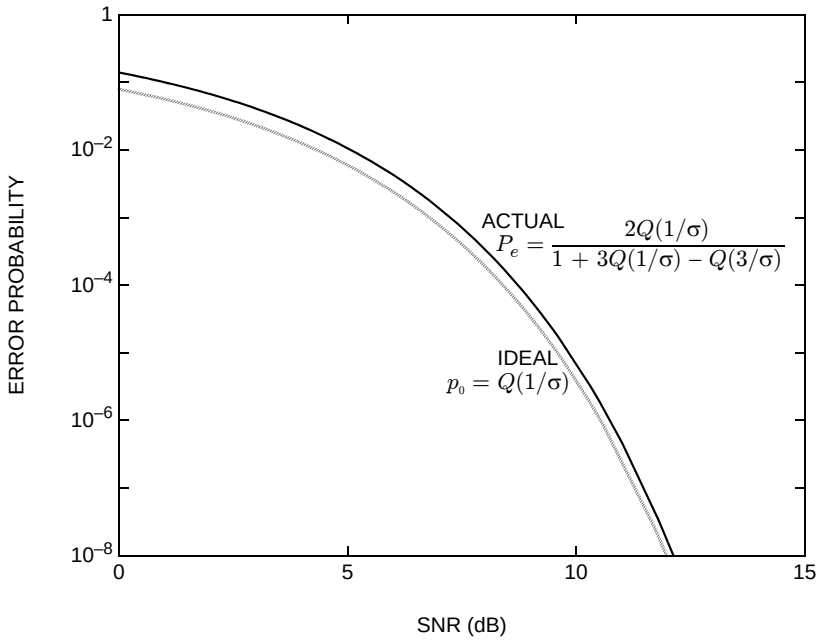

Figure 11: The impact of error propagation on the performance of the ZF DFE, for the channel $H(z) = 1 + z^{-1}$. The lower curve is the performance of the genie-aided (ideal) DFE that somehow has access to correct decisions. The upper curve is the performance that takes into account the impact of decision errors in the feedback process. The SNR penalty due to error propagation is 0.17 dB at $P_e = 10^{-7}$.

	FORWARD $F(z)$	FEEDBACK $B(z)$	SNR
MFB			$\mathcal{A}\{SNR(\theta)\}$
SHANNON			$\mathcal{G}\{1+SNR(\theta)\}-1$
MMSE DFE	$\frac{\beta_0^{-1}H^*}{\tilde{\gamma}^2\tilde{M}^*}$	$(\tilde{M}(z)-1)/\beta_0$	$\mathcal{G}\{1+SNR(\theta)\}-1$
ZF DFE	$\frac{H^*}{\gamma^2 M^*}$	$M(z)-1$	$\mathcal{G}\{SNR(\theta)\}$
MMSE LINEAR	$\frac{\beta_0^{-1}H^*}{HH^*+\frac{N_0}{E_a}}$	0	$\mathcal{H}\{1+SNR(\theta)\}-1$
ZF LINEAR	$\frac{1}{H(z)}$	0	$\mathcal{H}\{SNR(\theta)\}$

Table 1: Summary of structure and performance of the linear and DFE equalizers. The first row shows the matched-filter bound on SNR for the case of a flat transmit spectrum ($S_a(e^{j\theta}) = E_a$), for comparison. The bias coefficient β_0 of both MMSE equalizers is related to the corresponding SNR (last column) by $\beta_0^{-1} = 1 + 1/SNR$.

A summary of the linear and DFE equalizers is provided in table 1. The rows are sorted roughly from best to worst, according to their SNR. The sorting is rough because it is possible for an MMSE linear equalizer to outperform a ZF DFE, if the channel ISI response and SNR are carefully chosen. But the other rankings are universal; we always have:

$$SNR_{\text{MFB}} \geq SNR_{\text{Shannon}} = SNR_{\text{MMSE-DFE}} \geq SNR_{\text{ZF-DFE}},$$

and

$$SNR_{\text{MMSE-LE}} \geq SNR_{\text{ZF-LE}}.$$

Example: Consider a 4-QAM alphabet $\mathcal{A} = \{\pm 1 \pm j\}$ over the ISI channel $H(z) = 1 + bz^{-1}$ with noise PSD $N_0 = 2$, so that the SNR spectral density is $SNR(\theta) = |1 + be^{-j\theta}|^2$. The severity of the ISI in this example is captured by the value of the ISI coefficient $h_1 = b$. At one extreme, $b = 0$ corresponds to the case of an ideal ISI-free channel. At the other extreme, $b = 1$ corresponds to severe ISI, where the energy of the interfering symbol is the same as the energy of the desired symbol. In Fig. 12 we plot the SNR after equalization versus the channel coefficient b, as b ranges from 0 to 1. For small values of b we see that all equalizers perform about the same. For large values of b, however, we see that the MMSE versions significantly outperform the ZF counterparts, and further we see that the DFE significantly outperforms the LE. Of particular note is the poor performance of the ZF LE as b grows large, which is the result of noise enhancement: As b approaches 1, the channel frequency response develops a spectral null; an equalizer that attempts to invert this null will end up amplifying the noise by an infinite gain.

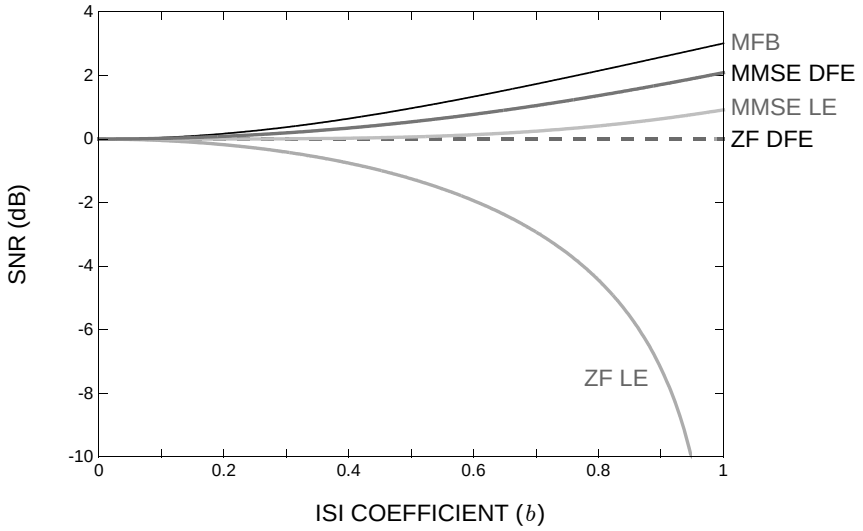

Figure 12: SNR after equalization versus the severity of the ISI, as captured by the value of the ISI coefficient $h_1 = b$.

5.4 DFE via Noise Prediction

The DFE was introduced above as an instance of interference cancellation, where the feedback filter reconstructs the ISI from past decisions and subtracts it. Here we describe an alternative viewpoint, namely that the DFE can be viewed as a combination of linear equalization and *noise prediction*. To see how, let us start with the ZF *linear* equalizer $C(z) = 1/H(z)$, whose output is $y_k = a_k + \eta_k$, where η_k is the filtered (enhanced) noise:

$$\eta_k = n_k * c_k,$$

which has the PSD:

$$\begin{aligned} S_\eta(z) &= N_0 C(z) C^*(1/z^*) \\ &= \frac{N_0}{H(z)H^*(1/z^*)}. \end{aligned}$$

The fact that this noise PSD is not white implies that the filtered noise values $\{\eta_k\}$ are correlated, and that knowledge of the past values can be used to predict the next value. Let $\hat{\eta}_k = \sum_{i=1}^{\infty} p_i \eta_{k-i}$ be a linear predictor. In terms of the spectral factorization $H(z)H^*(1/z^*) = \gamma^2 M(z)M^*(1/z^*)$ of (34), the prediction coefficients $\{p_i\}$ that minimize the mean-squared error $E(|\hat{\eta}_k - \eta_k|^2)$ are defined by $P(z) = 1 - M(z)$. While the receiver does not have access to the filtered noise η_k directly, it does have access to the receiver decisions $\hat{a}_k$, and the difference

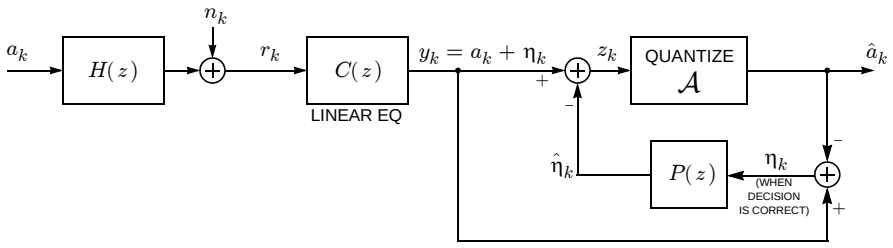

Figure 13: An alternative interpretation of the DFE based on linear prediction of the noise. Both the ZF and MMSE DFE have this interpretation, depending on whether the front-end linear equalizer is the ZF or MMSE linear equalizer.

$y_k - \hat{a}_k$ reduces to η_k whenever the corresponding decision is correct. Therefore, under the assumption that the past decisions are correct, the linear predictor $\hat{\eta}_k = \sum_{i=1}^{\infty} p_i \eta_{k-i}$ for the k-th filtered noise sample can be rewritten as:

$$\hat{\eta}_k = \sum_{i=1}^{\infty} p_i (y_{k-i} - \hat{a}_{k-i}).$$

The receiver can then subtract this predicted value for η_k from the equalizer output, as shown in the block diagram of Fig. 13. The impact of this subtraction changes the noise that is seen at the decision device: Instead of being η_k, which has power $E(|\eta_k|^2) = \mathcal{A}\{N_0 |H(e^{j\theta})|^{-2}\}$, the noise after subtraction is $\eta_k - \hat{\eta}_k$, which has power $E(|\eta_k - \hat{\eta}_k|^2) = \mathcal{G}\{N_0 |H(e^{j\theta})|^{-2}\}$, when the prediction coefficients $\{p_i\}$ are chosen optimally, and assuming the relevant past decisions are correct. The geometric mean is always less than the arithmetic mean whenever there is ISI, so the use of noise prediction will always lead to a performance improvement.

Perhaps unexpectedly, the noise prediction strategy described above and illustrated in Fig. 13 is actually precisely equivalent to the ZF DFE described earlier. In other words, linear prediction does not lead to a new receiver strategy, it merely provides an alternative viewpoint for interpretation of the DFE strategy. Indeed, looking closer at the prediction architecture in Fig. 13, we see that it can be viewed as an instance of the DFE structure in Fig. 10, when the forward filter is $F(z) = C(z)(1 - P(z))$ and the feedback filter is $B(z) = -P(z)$. Furthermore, when substituting the optimal linear prediction coefficients $P(z) = 1 - M(z)$, the forward and feedback filters reduce to the optimal forward and feedback filters of the ZF DFE, as described by (32) and (33).

While we have restricted our attention here to the ZF DFE, the noise predictive architecture of Fig. 13 applies equally well to the MMSE DFE; the only change is that the front-end linear equalizer is not the ZF but the MMSE linear equalizer, and the predictor is predicting the resulting error at the output, which includes residual ISI as well as noise. When the prediction coefficients are

chosen optimally, the predictive architecture is equivalent to the MMSE DFE as defined by (36) and (37).

6 Tomlinson-Harashima Precoding

The theory of the previous section shows that the ideal performance of the MMSE DFE is sufficient to achieve the Shannon capacity of the ISI channel. This is a remarkable result because it implies that the more complicated trellis-based detectors (such as Viterbi and BCJR) are not ideally necessary to approach capacity. However, this theoretical result is based on an assumption that the DFE has *instantaneous* access to *correct* decisions, which is nearly impossible to achieve in practice. Decisions that are instantaneous cannot tolerate the long decoding delay of powerful error-control codes, and are hence unlikely to be correct; decisions that are likely to be correct require a significant decoding delay that prevents them from being instantaneous. In this sense, the DFE is fundamentally incompatible with powerful error-control codes.

Tomlinson-Harashima precoding (THP) is an alternative to the DFE with similar performance that is compatible with error-control coding [14][7]. In a sense, THP can be thought of as a way of implementing the DFE *at the transmitter* instead of at the receiver. Since the transmitter knows the symbols it transmits, there is no chance for error propagation. However, as we will see, this benefit comes at the cost of (1) a slight increase in transmit power, which translates to a small SNR penalty, and (2) the requirement that the channel ISI response be known precisely at the transmitter. THP is thus not a good match for rapidly varying channels as well as other situations where it is difficult to get precise channel knowledge to the transmitter.

The relationship between THP and DFE is illustrated in Fig. 14. The top of the figure shows the channel model followed by the DFE of Fig. 10. As a reminder, for the ZF case, the forward filter transforms the channel into its monic and minimum-phase equivalent form, so that $H(z)F(z) = M(z) = 1 + m_1 z^{-1} + m_2 z^{-2} + ...$, and the feedback filter is the tail $B(z) = M(z) - 1$ of this channel.

The Tomlinson-Harshima precoder (THP) is shown at the bottom of Fig. 14. The THP principle requires that the message symbols $\{a_k\}$ be chosen from an M-ary QAM alphabet, say $\mathrm{Re}\{\mathcal{A}\} = \mathrm{Im}\{\mathcal{A}\} = \{\pm 1, \pm 3, \ldots, \pm(K-1)\}$, where $K = \sqrt{M}$. Instead of transmitting $\{a_k\}$ directly, the k-th symbol αx_k transmitted by THP is computed recursively according to:

$$x_k = [a_k - \sum_{i=1}^{\infty} m_i x_{k-i}]_{2K}, \tag{40}$$

where m_k is the impulse response of the monic and minimum-phase equivalent channel $M(z)$, and where we have introduced the complex modulo operator:

$$[z]_{2K} = z + 2K(p + jq),$$

where p and q are integers chosen so that the result $[z]_{2K}$ falls within the complex square centered at the origin with sides of length $2K$. Looking at it another way: the modulo operator adds an integer multiple of $2K$ to both the real and imaginary parts of its input, and it chooses these two integers so that the *modulo output is as close to the origin as possible*. The modulo operator thus maps any point in the complex plane to a point inside the $2K$-by-$2K$ square centered at the origin.

THP is known as "DFE at the transmitter" because the mapping from a_k to x_k of (40) is identical to the the second-half of the DFE, with feedback filter $M(z)-1$, except that the decision device of the DFE is replaced by the complex modulo operation.

One way to motivate the modulo operation is to recognize that, without the modulo operator, the mapping in (40) would be a recursive implementation of a filter with transfer function $1/M(z)$. In other words, without the modulo operator, the mapping in (40) would be a *ZF linear equalizer at the transmitter*. And just like linear equalization at the receiver suffers from noise enhancement, linear equalization at the transmitter suffers from what might be called *signal* enhancement: the filter could potentially amplify the transmit signal power by a significant amount, especially when the channel ISI is severe. One way to compensate for this signal amplification is to follow the linear filter by a compensating attenuator. (Doing so would lead to the conclusion that linear equalization at the transmitter performs identically to linear equalization at the receiver; performance-wise it is neither better nor worse.) THP compensates for the signal amplification in a nonlinear way using the modulo operation, which removes nearly all of the signal amplification, while at the same time allowing the receiver to compensate (with high probability) for the modulo operation with another modulo operation.

For most channels the output of the THP modulo operator will be approximately uniformly distributed over the $2K$-by-$2K$ square in the complex plane, which makes its energy $E_x = E(|x_k|^2) = 2K^2/3$. In contrast, the energy before the precoder is $E_a = 2(K^2-1)/3$. The THP precoder thus includes an attenuation factor of $\alpha = \sqrt{E_a/E_x}$, and transmits αx_k as the k-th transmitted symbol. This attenuation makes for a fair comparison, because it ensures that the energy of the symbols transmitted by a THP precoder is the same as the energy of the original alphabet $\mathcal{A}$. We see that α will be a number only slightly less than unity, and will approach unity as the size of the alphabet grows large. For example, the value of α is 0.866, 0.968, 0.992, and 0.998 for QAM alphabets $\mathcal{A}$ of size 4, 16, 64, and 256, respectively.

At the receiver, the THP uses a scaled version of the same front end as the DFE, namely the linear filter $\alpha^{-1}F(z)$, where $H(z)F(z) = M(z)$. The purpose of the gain factor α^{-1} in the receiver is to cancel the attenuation factor of α that was introduced at the transmitter. Since the forward filter satisfies $H(z)F(z) = M(z)$, and since the precoder inverts $M(z)$ as well as effectively adding a complex integer $2K(p_k + jq_k)$ to each information symbol a_k, it follows that the output of the scaled forward filter at the receiver will be:

$$y_k = a_k + 2K(p_k + jq_k) + n_k/\alpha. \tag{41}$$

The first term is the desired information symbol, which lies within the $2K$-by-$2K$ square. When the noise n_k/α is small enough, the second term can be cleanly wiped away by a modulo operator, yielding $[y_k]_{2K} = a_k + n_k/\alpha$. Motivated by this observation, the receiver applies the forward filter output to the same modulo operator that was used at the transmitter. It then follows with a standard memoryless quantizer, which rounds each input to the nearest element of the alphabet $\mathcal{A}$.

Example: Consider a $K^2 = 16$ QAM alphabet with $\mathrm{Re}\{\mathcal{A}\} = \mathrm{Im}\{\mathcal{A}\} = \{\pm 1, \pm 3\}$, so that $K = 4$. Assume the ISI channel is $H(z) = 1 + (0.4 - 0.1j)z^{-1} + (0.1 + 0.5j)z^{-2} + (0.3 + 0.8j)z^{-3}$, and the SNR is 16 dB. Shown in Fig. 14 are the constellations for several of the signals at various points in the THP system. First, the constellation for the information symbols a_k is shown, along with the 8-by-8 square. The output x_k of the modulo operator is empirically measured to be approximately uniform over the 8-by-8 square, so that its constellation is represented by a shaded square. After the forward filter, the constellation for y_k of (41) is shown. The constellation clearly extends well beyond the 8-by-8 square. Applying y_k to another modulo operator yields the final decision variable z_k, whose constellation is strictly confined to the 8-by-8 square. It closely resembles a simple 16-QAM alphabet that has been perturbed by Gaussian noise.

7 Comparing Performance: A Case Study

There is no definitive ranking of the various equalization strategies described in this chapter; they all have their merits, and it will be the specific details of the application scenario that will dictate which choice is best. For example, fiber-optic links and other high-data-rate applications will often not have enough computational power to implement trellis-based detectors, and hence will favor the reduced-complexity alternatives. Rapidly varying channels may not easily acquire knowledge of the channel at the transmitter, and hence will avoid transmitter-based equalization strategies like THP. High-performance links that operate at low SNR and use powerful error-control strategies will avoid DFE, since the decoding delay prevents the DFE from having timely and reliable decisions. The severity of the ISI also plays a major role when comparing various strategies. For example, linear equalization is generally not a viable option when the ISI is severe, while linear equalization is generally a preferred option when the ISI is mild (and there is not much to be gained by the more complex alternatives).

To close out this chapter we will compare equalizer performance for one particular example. The reader is cautioned against drawing any universal con-

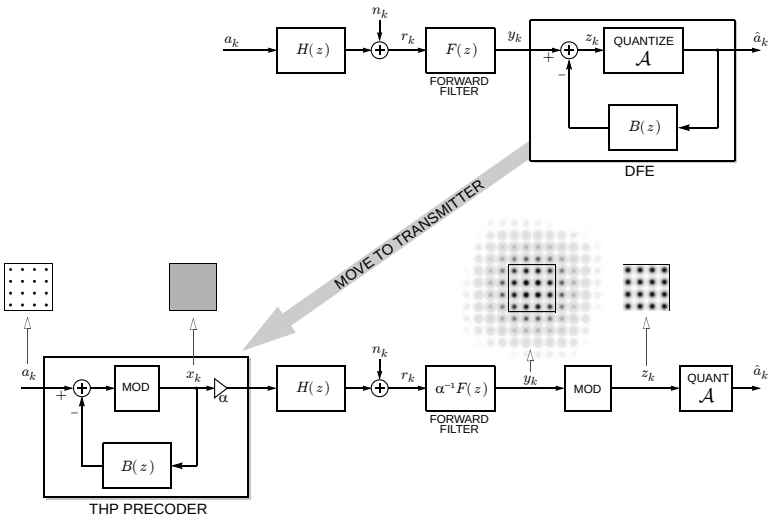

Figure 14: The relationship between DFE and THP. The top of the figure shows the channel model followed by the DFE from Fig. 10. The bottom part of the figure shows Tomlinson-Harashima precoder (THP) implemented at the transmitter, before the channel, with additional processing performed by the receiver. The feedback filter of the DFE has been moved to the transmitter, but with two modifications: (1) the quantizer of the DFE is replaced by a modulo device for THP; (2) there is a scaling constant α in THP to avoid any signal power amplification. Also shown in the figure are constellations at various points of the THP system, for a 16-QAM alphabet and a particular ISI response at SNR = 16 dB.

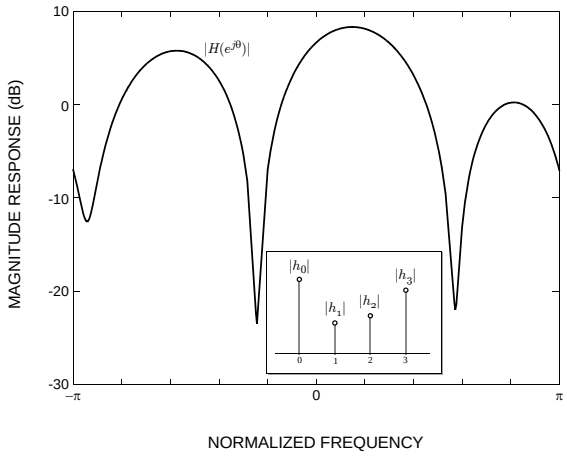

Figure 15: The channel magnitude response for the case study example, which varies by 32 dB. The inset shows the magnitude of the impulse response coefficients in the time domain.

clusions from this one example; as the previous paragraph points out, a different example might reveal very different quantitiative results. Nevertheless the example we consider will be a valuable tool for pointing out the various qualitative differences between approaches.

Suppose a sequence of independent and uniformly distributed 16-QAM symbols is transmitted over an ISI channel with AWGN, whose impulse response is $H(z) = 1 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$, where $h_1 = 0.4 - 0.1j$, $h_2 = 0.1 + 0.5j$, and $h_3 = 0.3 + 0.8j$. The same ISI response was considered in the example of Sect. 4.3. This channel is monic ($h_0 = 1$) and minimum phase, which avoids the need for forward filters for the zero-forcing DFE and THP strategies. The monic and minimum-phase condition is chosen for convenience, but is not itself a limiting constraint on the channel, because any nonmonic and non-minimum-phase channel with AWGN can always be transformed into a monic and minimum-phase channel by a front-end all-pass filter. In effect, by starting with the minimum-phase channel, we are absorbing this all-pass filter into the definition of the channel. The magnitude response of the channel is shown in Fig. 15, where we see deep spectral nulls that are 32 dB below the peak magnitude response. The ISI in this example is severe.

Because the transmit PSD is flat, the underlying channel SNR and the MFB (10) on post-equalization SNR are identical, namely $SNR = E_a \sum_k |h_k|^2 / N_0$. Some equalization strategies, like the linear and decision feedback equalizers, strive to transform the ISI channel into an effective memoryless (ISI-free) channel. When successful, the performance after such an equalizer can be quantified using the classical closed-form expression for the symbol-error probability of an ML detector in AWGN for 16-QAM (see, e.g., (5.113) in [2]):

$$P_e = 3Q(\sqrt{SNR_{\text{eq}}/5}) - 2.25Q^2(\sqrt{SNR_{\text{eq}}/5}), \tag{42}$$

where SNR_{eq} is the SNR at the output of the equalizer. In the process of this transformation, the equalizer typically introduces a penalty in SNR, so that SNR after the equalizer is strictly less than the SNR of the underlying channel. In the previous sections we derived theoretical expressions for this post-equalizer SNR for various strategies, and compared them. Here we instead look at the performance as measured by the symbol-error probability P_e.

The error-probability performance of eight different equalizers is shown in Fig. 18. In the following we will discuss the results for each strategy, one by one:

ZF Linear Equalizer

- *Implementation* — Because the channel is FIR, its linear inverse (i.e., the ZF linear equalizer of (23)) cannot be implemented as an FIR transversal filter. (It would require an infinite number of coefficients.) However, because the channel has been reduced to its monic and minimum-phase form, its inverse can be implemented recursively via $y_k = r_k - \sum_{i=1}^{\mu} h_i y_{k-i}$, where r_k is the equalizer input and y_k is the equalizer output, so that the ZF linear equalizer can be implemented with only $\mu = 3$ (feedback) coefficients.

- *Performance* — The right-most curve in Fig. 18 shows the error probability performance of the zero-forcing linear equalizer. It performs worse than all other options, requiring nearly 21 dB to achieve a symbol-error probability of $P_e = 10^{-5}$. This curve (like most of the others) was generated via Monte Carlo simulations over millions of independent realizations of the symbols and noise. However, because the ZF linear equalizer exactly transforms the ISI channel into an ISI-free channel, we could instead use the closed-form equation of (42) for this error-probability curve, with SNR_{eq} set to the SNR after the ZF linear equalizer, namely the harmonic mean SNR_{ZF} of the SNR spectral density, as specified in (25).

MMSE Linear Equalizer:

- *Implementation* — Unlike the ZF case, the MMSE linear equalizer of (27) cannot be implemented in exact form using a finite number of coefficients. Instead, a finite-coefficient version of the MMSE was implemented, with the number of coefficients chosen to be 200, large enough to mimic the performance of the theoretical infinite-coefficient case. The magnitude of the 200 MMSE linear equalizer coefficients for this example is shown in Fig. 15 for the case when $SNR = 15$ dB.

- *Performance* — The MMSE linear equalizer outperforms the ZF linear equalizer in this example at all SNR values. The advantage of MMSE over

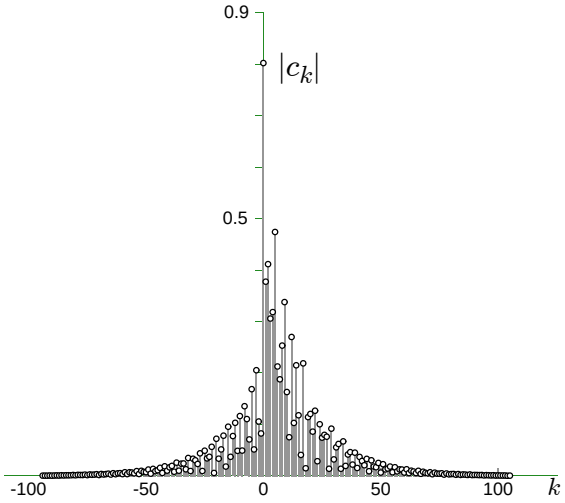

Figure 16: The magnitude of the 200 coefficients of the MMSE linear equalizer for the case study, when SNR = 15 dB. The leading and trailing coefficients are close to zero, suggesting that 200 coefficients is enough to mimic the infinite-coefficient theoretical MMSE linear equalizer.

ZF is seen to decrease as the SNR grows; the horizontal spacing between the MMSE and ZF linear curves is only about 0.2 dB at $P_e = 10^{-5}$. This trend confirms the fact that MMSE and ZF converge at high SNR, as expected because of the fact the MMSE equalizer of (26) reduces to the ZF equalizer as $N_0 \to 0$. We can use (42) to predict performance for the MMSE linear equalizer, with SNR_{eq} set to SNR_{MMSE} from (27). The resulting curve is no longer exact, however, because the MMSE equalizer only approximately transforms the ISI channel into an ISI-free channel, leaving some residual ISI that is subsequently treated as noise. The fact that the distribution of this residual ISI is not precisely Gaussian is what makes the expression in (42) an approximation. Nevertheless, the central-limit theorem ensures that it is close to Gaussian, and the curve predicted by (42) (not shown) is indistinguishable from the actual performance curve (generated via Monte Carlo simulation).

ZF DFE:

- *Implementation* — The ZF DFE is described by (31). Because the channel is minimum-phase and monic, the forward filter of (32) reduces to the identity, $F(z) = 1$ (requiring no coefficients), and the feedback filter of (33) reduces to the impulse response tail $B(z) = H(z) - 1 = h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$, requiring only $\mu = 3$ coefficients. The complexity (as measured by the number of coefficients) of the ZF DFE is thus no greater

than that required by the ZF linear equalizer. Furthermore, by the same complexity metric, the complexity of the ZF DFE is significantly *less* than that of the MMSE linear equalizer (which requires in theory an infinite number of coefficients, or as implemented 200 coefficients.)

- *Performance* — The ZF DFE is seen to significantly outperform both linear equalizers. Remarkably, this performance advantage comes at essentially no cost in complexity. The dashed curve labeled "ideal ZF DFE" shows the performance predicted by (42) with SNR_{eq} set to $SNR_{\text{ZF-DFE}}$ from (35); we see that this bound is not an accurate predictor of performance, because (35) does not take into account the effects of error propagation. The impact of error propagation can be quantified by comparing the dashed and solid curves for the ZF DFE. Error propagation is especially detrimental at low SNR, as might be expected, but its impact decreases rapidly at high SNR. At $P_e = 10^{-5}$, the SNR penalty due to error propagation for the ZF DFE is 0.5 dB. Comparing ZF DFE to the linear equalizers, we see that the ZF DFE always outperforms the ZF linear equalizer in this example, regardless of SNR, and regardless of error propagation. Comparing the ZF DFE to the MMSE linear equalizer, we see a crossover point: At high enough SNR the ZF DFE is better, while at low enough SNR the MMSE linear equalizer is better.

MMSE DFE:

- *Implementation* — The MMSE DFE is described by (31), (36), and (37). Because the channel $H(z)$ in this example is minimum phase, the forward filter $F(z)$ of (36) is anticausal, satisfying $f_k = 0$ for all $k > 0$. In theory this $F(z)$ requires an infinite number of equalizer coefficients. Like the linear case, however, we can approximate the infinite-coefficient case by using a finite number of coefficients, and choosing this number to be large. For this example, the MMSE DFE was implemented using a forward filter having 200 coefficients, that approximates the infinite-coefficient filter of (36). The MMSE DFE feedback filter from (37) has only $\mu = 3$ nonzero coefficients. The magnitudes of the forward and feedback coefficients for the MMSE are shown in Fig. 17, where they are compared to those for the ZF case.

- *Performance* — The MMSE DFE is seen to outperform the ZF DFE at all values of SNR. The performance gain is most significant at low SNR, and decreases as SNR increases. At $P_e = 10^{-5}$, the advantage of MMSE DFE over ZF DFE is only 0.1 dB. This gain comes at the cost of an increase in complexity due to the increased number of filter coefficients. The dashed curve labeled "ideal MMSE DFE" shows the performance predicted by (42) with SNR_{eq} set to $SNR_{\text{MMSE-DFE}}$ from (39); as for the ZF case, this bound is not an accurate predictor of performance because it neglects

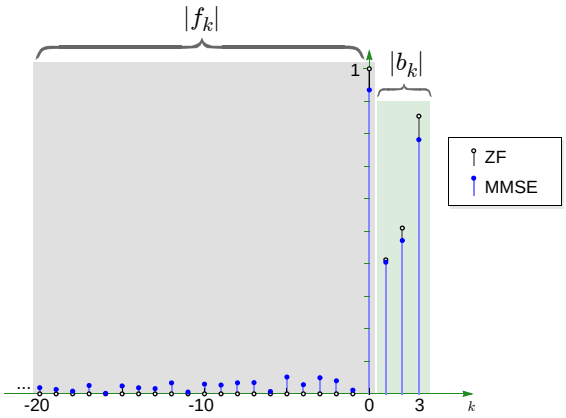

Figure 17: Comparison of the ZF and MMSE forward and feedback coefficients (magnitudes) for the case study example, when $SNR = 15$ dB. The coefficients in the left box are the anticausal forward filter coefficients f_k. For the ZF the only nonzero forward coefficient is $f_0 = 1$, while for the MMSE case there are 200 nonzero coefficients (not all are shown). The coefficients in the right box are the strictly causal feedback filter coefficients b_1 through b_3.

error propagation. Furthermore, again as in the ZF case, the penalty due to error propagation at $P_e = 10^{-5}$ for the MMSE DFE is 0.5 dB.

ZF Tomlinson-Harashima Precoding (THP):

- *Implementation* — The ZF THP precoded symbols x_k are generated according to (40), where — because $H(z)$ is already monic and minimum phase in this example — $M(z)$ is set to $H(z)$. This will only be possible when the transmitter knows the channel, perhaps through a feedback mechanism from the receiver. To avoid the signal-enhancement effect of the modulo device, these symbols are attenuated in amplitude by a factor of $\alpha = \sqrt{E_a/E_x} = \sqrt{10/(32/3)} \approx 0.9682$ before transmitting (see Fig. 14). The forward filter $F(z)$ reduces to unity in this example. The receiver thus directly scales the channel output and applies a modulo device, yielding $z_k = [r_k/\alpha]_8$, which is then rounded to the nearest alphabet symbol to arrive at the decision.

- *Performance* — ZF THP is similar to ZF DFE but with one big advantage: by moving the feedback filter from the receiver to the transmitter (see Fig. 14), THP avoids the problem of error propagation. In Fig. 18 we see that that ZF THP outperforms ZF DFE at all SNR values, and that the gap in performance is especially large at low SNR, where error propagation is most severe. However, the performance of ZF THP is seen to fall short of the performance of the ideal ZF DFE (dashed curve) that feeds back

correct decisions; this gap is due to a combination of the attenuation factor α and the loss inherent in the modulo device at the receiver.

MMSE THP:

- *Implementation* — In theory, the MMSE THP forward filter $F(z)$ is the MMSE DFE forward filter of (36), and the precoded symbols x_k are generated according to (40), where the feedback filter $M(z) - 1$ is the MMSE DFE feedback filter of (37). As before, the attenuation factor is $\alpha = \sqrt{E_a/E_x} \approx 0.9682$. The forward filter was implemented using 200 coefficients, enough to emulate the infinite-coefficient filter specified by (36). The receiver scales by $1/\alpha$ and filters by the MMSE forward filter $F(z)$, as shown in Fig. 14, applies a modulo device, and then rounds to the nearest alphabet symbol to arrive at the decision.

- *Performance* — In Fig. 18 we see that that MMSE THP is slightly but consistently better than ZF THP at all SNR values. As in the ZF case, the MMSE THP closely approaches the performance of the ideal MMSE DFE; it falls short because of a combination of the attenuation factor α and the modulo loss. It is worth emphasizing that THP is a transmitter-based equalization strategy that requires knowledge of the channel at the transmitter; as such it will be incompatible with many rapidly varying applications, as well as with many broadcast (one-to-many) applications.

Viterbi Algorithm:

- *Implementation* — The Viterbi algorithm is implemented according to (14), where in place of $g_k(p, q)$ we use the simplified additive branch metric $\hat{\mu}_k(p, q) = |r_k - s^{(p,q)}|^2$. There is no *a priori* term in the branch metrics, because all transmitted symbols are independent and uniformly distributed over the 16-QAM alphabet. Because the channel memory is $\mu = 3$ and the alphabet size is $M = 16$, the number of states in the trellis is $M^\mu = 4096$. Furthermore, there will be $M = 16$ branches emanating from each node in the trellis. The overall complexity is thus extremely high, significantly higher than the suboptimal equalizers considered above.

- *Performance* — The gray curve in Fig. 18 shows the performance of the Viterbi algorithm. At high SNR, the Viterbi algorithm significantly outperforms all of the linear, decision feedback, and THP strategies. At $P_e = 10^{-5}$, the SNR advantage of Viterbi over MMSE THP is 1.7 dB, and the advantage over ZF LE is more than 5.3 dB. Interestingly, the performance advantage of Viterbi is not as significant at extremely low SNR values. In fact, when the SNR is so low that the error probability exceeds 20%, the Viterbi algorithm is seen to perform slightly worse than the significantly less complex THP strategies. The dashed curve labeled "MFB" shows the error probability of the matched-filter bound, which is

computed by substituting (10) into (42). The Viterbi algorithm closely approaches the MFB at high SNR, falling only 0.2 dB short at $P_e = 10^{-5}$.

BCJR Algorithm:

- *Implementation* — The BCJR is implemented using the same 4096-state trellis as Viterbi, with a multiplicative branch metric $\gamma_k(p,q) = e^{-|r_k - s^{(p,q)}|^2/N_0}$ that is a simplified version of (15). (Factors in the branch metric that are common to all branches – like the *a priori* probability factor – are ignored, to reduce complexity.) The forward and backward recursions are implemented according to (17) and (18). The scaled *a posteriori* probabilities for each QAM symbol are computed according to $P(a_k = a|\mathbf{r}) = \sum_{(p,q)\in\mathcal{B}_a} \alpha_k(p)\gamma_k(p,q)\beta_{k+1}(q)$, where $\mathcal{B}_a$ is the set of branches (p,q) corresponding to an input symbol of $a \in \mathcal{A}$. The maximum such *a posteriori* probability determines the MAP decision, $\hat{a}_k = \arg\max_{a\in\mathcal{A}} P(a_k = a|\mathbf{r})$.

- *Performance* — The hard-output BCJR detector, which minimizes symbol-error probability, is seen to be almost indistinguishable from the Viterbi algorithm, which minimizes sequence-error probability. At high SNR they perform identically, while at low SNR we see a slight advantage for BCJR over Viterbi. This result reinforces a point that was made earlier: hard-output BCJR is not typically worth the trouble, the Viterbi algorithm is simpler to implement and performs nearly the same. The story would change if we were to expand the case study to include error-control coding, however; in that case the soft outputs of BCJR would become extremely valuable, and the performance after error-control decoding would be significantly better with BCJR than with Viterbi.

8 Summary

This chapter has reviewed an array of strategies for dealing with the intersymbol interference that arises because of dispersive channels. Linear and decision-feedback equalizers transform the ISI channel into an effective ISI-free channel, which enables us to quantify their performance by the post-equalization SNR. This post-equalization SNR was in turn shown to be related in a simple way to the harmonic and geometric means of the SNR spectral density. These results are tabulated in Table 1. In practice the DFE SNR falls short of the SNR predicted by this theory because of error propagation; in the case study example the penalty was seen to be about 0.5 dB. The Viterbi and BCJR detectors have no notion of a post-equalization SNR, although their performance is bounded and typically close to the matched-filter bound at low error probabilities. Which strategy to choose for a particular design scenario will depend on many factors, including the severity of the ISI, the computational resources, and the availability of reliable channel knowledge at the transmitter.

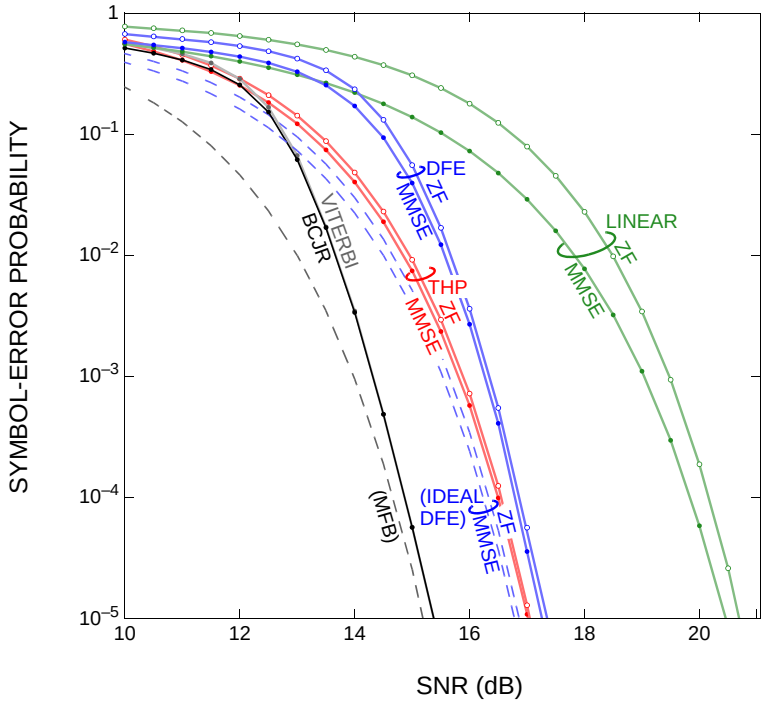

Figure 18: Error probability comparison of various equalizers for the case study example.

This chapter's focus was limited to classical equalization strategies for single-carrier systems. Space limitations prevented us from exploring several alternative equalizers and related concepts. For example, we assumed throughout that the channel response was known. Channel estimation [17] and adaptive equalization [12] for finite-coefficient equalizers [2] are important topics for the realistic case. We assumed perfect synchronization at all levels (e.g., frame, symbol, and carrier phase). Synchronization strategies are explored in Chap. 7. Fractionally spaced equalizers [15] are robust to timing errors and to signals with excess bandwidth (beyond the minimum bandwidth assumed in this chapter), while passband equalizers [13] are robust to carrier phase errors. A combination of partial-response linear equalization and trellis-based detection has been effectively used for hard-disk drives for decades [4]. Single-carrier frequency-domain equalization (as used in LTE-A uplink) is a reduced-complexity alternative for implementing either a linear equalizer or the forward filter of a DFE [11]. Multicarrier strategies such as orthogonal-frequency-division multiplexing (OFDM) are immensely important strategies for communication over ISI channels; conceptually the idea is to transmit data independently across numerous subchannels, each of whose bandwidth is so narrow that a simple one-coefficient linear "equalizer" is enough to compensate for the dispersive channel. Chap. 10 is devoted to OFDM and related concepts.

References

[1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *Information Theory, IEEE Transactions on*, 20(2):284–287, Mar 1974.

[2] J.R. Barry, E.A. Lee, and D.G. Messerschmitt. *Digital Communication*. Springer-Verlag GmbH, 2004.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record, IEEE International Conference on*, volume 2, pages 1064–1070 vol.2, May 1993.

[4] R.D. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, and W. Schott. A prml system for digital magnetic recording. *Selected Areas in Communications, IEEE Journal on*, 10(1):38–56, Jan 1992.

[5] Jr. Forney, G.D. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[6] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *Information Theory, IEEE Transactions on*, 42(2):429–445, Mar 1996.

[7] H. Harashima and H. Miyakawa. Matched-transmission technique for channels with intersymbol interference. *Communications, IEEE Transactions on*, 20(4):774–780, Aug 1972.

[8] W. Hirt and J.L. Massey. Capacity of the discrete-time gaussian channel with intersymbol interference. *Information Theory, IEEE Transactions on*, 34(3):38–38, May 1988.

[9] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[10] R. Muller and W.H. Gerstacker. On the capacity loss due to separation of detection and decoding. *Information Theory, IEEE Transactions on*, 50(8):1769–1778, Aug 2004.

[11] F. Pancaldi, G.M. Vitetta, R. Kalbasi, N. Al-Dhahir, M. Uysal, and H. Mheidat. Single-carrier frequency domain equalization. *Signal Processing Magazine, IEEE*, 25(5):37–56, September 2008.

[12] S.U.H. Qureshi. Adaptive equalization. *Proceedings of the IEEE*, 73(9):1349–1387, Sept 1985.

[13] J. F. Hayes R. D. Gitlin and S. B. Weinstein. *Data Communication Principles*. Springer-V, 1992.

[14] M. Tomlinson. New automatic equaliser employing modulo arithmetic. *Electronics Letters*, 7(5):138–139, March 1971.

[15] J.R. Treichler, I. Fijalkow, and C.R. Johnson. Fractionally spaced equalizers. *Signal Processing Magazine, IEEE*, 13(3):65–81, May 1996.

[16] M. Tuchler and A.C. Singer. Turbo equalization: An overview. *Information Theory, IEEE Transactions on*, 57(2):920–952, Feb 2011.

[17] J.K. Tugnait, Lang Tong, and Zhi Ding. Single-user channel estimation and equalization. *Signal Processing Magazine, IEEE*, 17(3):16–28, May 2000.

[18] A.J. Viterbi. An intuitive justification and a simplified implementation of the map decoder for convolutional codes. *Selected Areas in Communications, IEEE Journal on*, 16(2):260–264, Feb 1998.