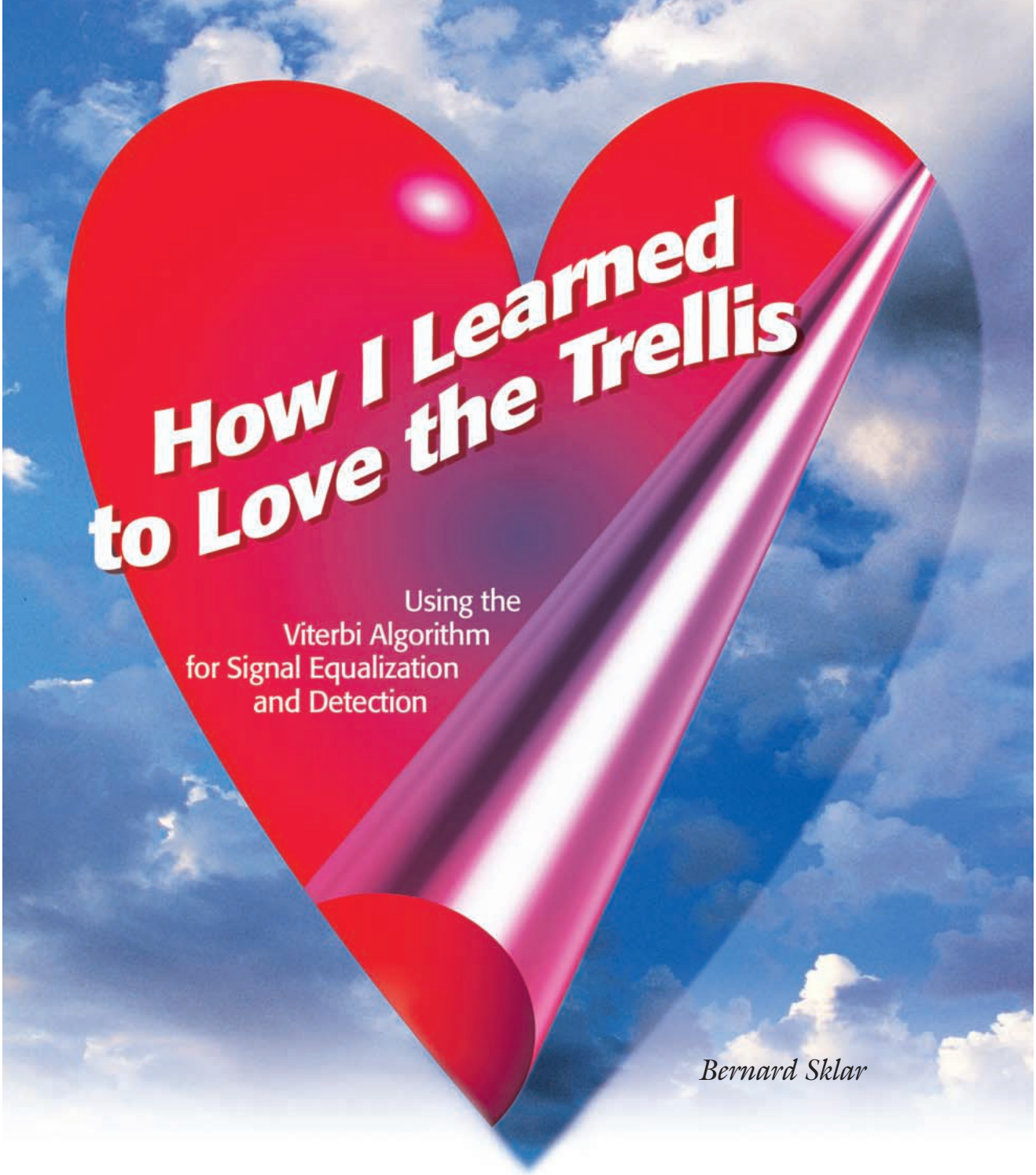


In 1967, Andrew Viterbi first presented his now famous algorithm for the decoding of convolutional codes [1]-[3]. A few years later, what is now known as the Viterbi decoding algorithm (VDA) was applied to the detection of data signals distorted by intersymbol interference (ISI) [4]-[8]. For such applications, the algorithm is often referred to as a Viterbi equalizer (VE). This tutorial focuses on the latter application, because it provides insight into how the VDA (or the VE), which is a maximum-likelihood sequence estimator (MLSE), actually works.

The VDA has become very popular for processing ISI-distorted signals that stem from a linear system with finite memory. Such a system is referred to as a finite-state machine (FSM), which is the general name given to a system (machine)

1053-5888/03/$17.00©2003IEEE

The Viterbi decoding algorithm has become very popular for processing ISI-distorted signals.

whose output signals are affected by (have a memory of) signals that occur earlier and later in time. The term finite refers to the fact that there are only a finite number of unique states that the machine can encounter. The term state is defined as the smallest amount of information that is necessary (and sufficient) to determine the next output for a given current input to the machine. The state provides knowledge of past input signal sequences and the restricted set of possible future outputs. A future state is always constrained by the past state.

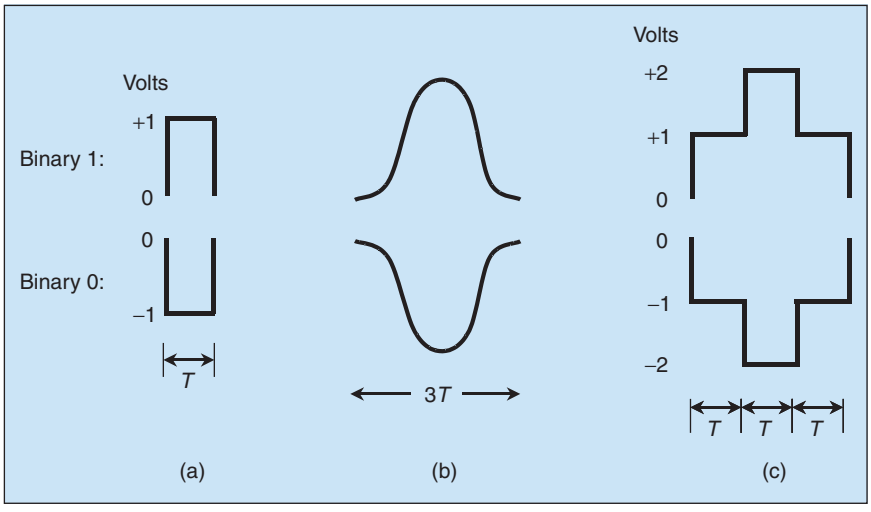
In this tutorial article, we first review the elements and concepts that are needed to demonstrate the VDA, such as FSMs, tree and trellis diagrams, ISI signal distortion, likelihood functions, and decision rules. Then, the algorithm is explained in the context of an equalization/detection application, and a typical implementation is shown using the architecture known as add-compare-select (ACS). Finally, we illustrate an interesting application of this so-called VE that is used in the global system for mobile (GSM) communications.

Intersymbol Interference (Signal Smearing)

In a digital communication system, we transmit waveforms to which digital meanings have been assigned. In general, we choose a waveform set of size M, where $M = 2^m$, enabling us to send m-bits at a time. When $m = 1$ (thus $M = 2$), the signaling is termed binary. When $m > 1$, the signaling is generally referred to as M-ary. Throughout this tutorial, binary signaling is assumed for ease of presentation. An extension of these same concepts to M-ary signaling can similarly be presented.

Ideal bipolar pulses as shown in Figure 1(a) can be used for transmitting binary data, where positive and negative 1 V pulses, each occupying a bit-time interval of T seconds, represent logical ones and logical zeros, respectively (inputs to the FSM). Such a binary signaling set, where one of the waveforms can be represented as the negative of the other, is also referred to as an antipodal signal set. Consider an FSM example where the system's memory is due to ISI. This can occur in signals that propagate over multipath channels, over cables with signal-dispersive characteristics, or via filtering circuits. [Sometimes, signal dispersive characteristics are purposely introduced (e.g., duobinary modulation [9]). Then, the signaling is referred to as partial response signaling or correlative coding.] ISI is quite visible when ideal-rectangular pulses, as seen in Figure 1(a), are applied to the input of a system and are transformed at the output into pulses that are rounded and smeared into neighboring pulse intervals, as shown in Figure 1(b). For ease of computations, we will be working with discretized approximations of the smeared pulses in Figure 1(b), as shown in Figure 1(c). In this equalization application, a parameter K will be used to represent the observation time or support time of the pulse, in units of bit intervals. Then, the product KT represents the support time in units of seconds. In Figure 1(b) and (c), it can be seen that each ISI-distorted pulse is time stretched or "smeared" over a support time of $K = 3$ bit intervals. In Figure 1(b) and (c), a pulse can be partitioned into three segments: a mainlobe (the region where the peak signal amplitude is located), a precursor (the rising edge that precedes the mainlobe), and a postcursor (the falling tail that follows the mainlobe).

For the pulse configurations shown in Figure 1(b) and (c), we can describe the system memory as being $\gamma = K - 1 = 2$ bit intervals long. That is, during a single bit-time interval T, an FSM-output waveform, due to a sequence of input pulses, will typically contain the effect of a current signal event, plus that of $\gamma = 2$ other events due to memory. Imagine an output waveform made up of several smeared pulses as shown in Figure 1(c). Then, in general, during each T-second mainlobe there will also be present the effect of a postcursor from the previous pulse and a precursor of the subsequent pulse. Or another view is that during each T-second precursor, there will also exist segments from two earlier pulses. The ISI-distorted pulses (characterized by a memory of $\gamma = 2$ bit intervals) in



▲ *1. An example of bipolar pulses: (a) transmitted ideal bipolar pulses, (b) received pulses distorted by ISI, and (c) discretized approximation of ISI-distorted pulses.*

Figure 1(c) represent an FSM model that will be assumed in most of the sections that follow.

ISI-Distorted Output Waveform in Terms of the System Impulse Response

Figure 2 represents a visual way of determining the output waveform that results from a specific 4-bit input sequence 0001 to our model FSM, where the leftmost bit is the earliest. The response of the FSM to each input bit has the discretized form as presented in Figure 1(c). As shown in Figure 2, we first align, according to their time occurrence, the four ISI-distorted responses (to the input 0001), and then we sum them to form the output waveform. The reader may recognize that the process of alignment and summation, yielding an output waveform in Figure 2, is a pictorial representation of convolution. Consider a mathematical representation of the process depicted in Figure 2. Let us start by denoting bipolar binary data with impulse functions, $\pm\delta(t)$. Such impulses can be shaped into the ideal rectangular pulses shown in Figure 1(a), which are then further shaped into the ISI-distorted pulses shown in Figure 1(c). We can combine both shaping functions into a single system. When an impulse is applied at the input of this overall system, the resulting output waveform, as shown in Figure 1(c), is termed the impulse response $h(t)$ of the system. The output $u(t)$ of such a system can be expressed as a sequence of data impulses convolved with the system's impulse response [9], as follows:

$$u(t)=\left[\sum_k d_k\,\delta(t-kT)\right]*h(t) \tag{1a}$$

where the symbol * denotes convolution, and $\sum_k d_k\,\delta(t-kT)$ describes a sequence of data impulses having values of $d_k=+1$ for a logical one, $d_k=-1$ for a logical zero, and occurring at times $t=kT$. In the 4-bit example of Figure 2, we denote these time instants as t_k and apply the inputs at times $t_k=t_{-2},\ldots,t_1$; assume that there is no propagation delay between input and output. In (1a), the convolution operation is only represented symbolically. We now express it in greater detail as

$$u(t)=\int_{-\infty}^{+\infty}\sum_k d_k\,\delta(\tau-kT)\,h(t-\tau)\,d\tau \tag{1b}$$

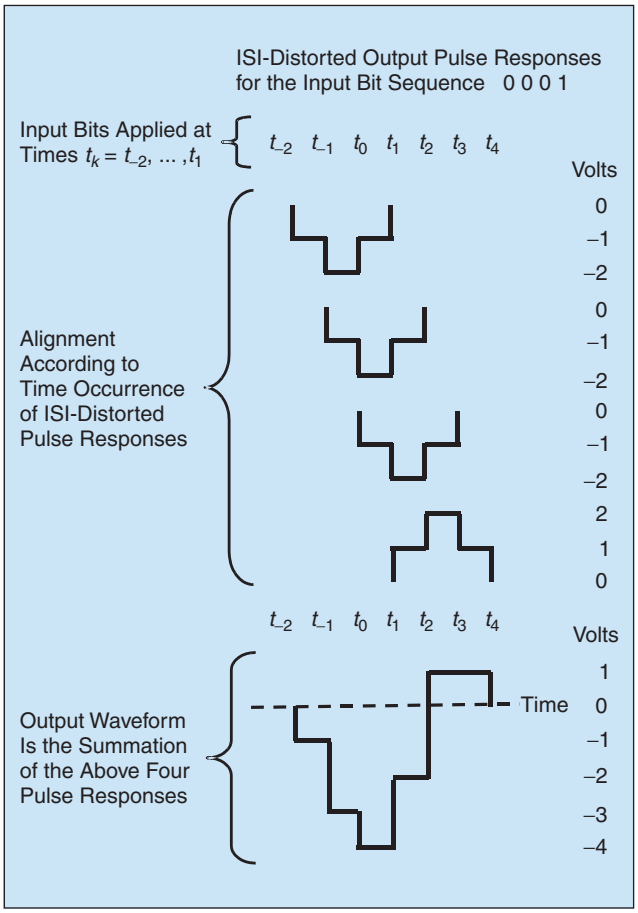
where the integral in (1b) is known as the convolution integral [9]. We next interchange the operations of integration and summation (permissible for linear systems), and we obtain

$$\begin{aligned}u(t)&=\sum_k\int_{-\infty}^{+\infty}d_k\,\delta(\tau-kT)\,h(t-\tau)\,d\tau\\&=\sum_k d_k\,h(t-kT),\end{aligned} \tag{1c}$$

The trellis diagram has become the "workhorse" for characterizing a finite state machine.

which is the mathematical representation of the output waveform (produced by the alignment and summation of the smeared bipolar pulses) shown in Figure 2.

Figure 3 shows eight output-waveform shapes (selected from the possible 16) that can result from different 4-bit input sequences. Notice that a memory of only $\gamma=2$ bit intervals causes considerable output-pulse smearing. In such cases, performing signal detection over just a single bit interval at a time can lead to significant error-performance degradation. If there is a great deal of ISI distortion (large value of γ), then detecting one bit at a time without equalization may result in unacceptable performance. In such cases, an algorithmic approach such as the VDA is well suited for finding an estimate of the most likely input sequence to the FSM, thereby implementing equalization and detection simultaneously.

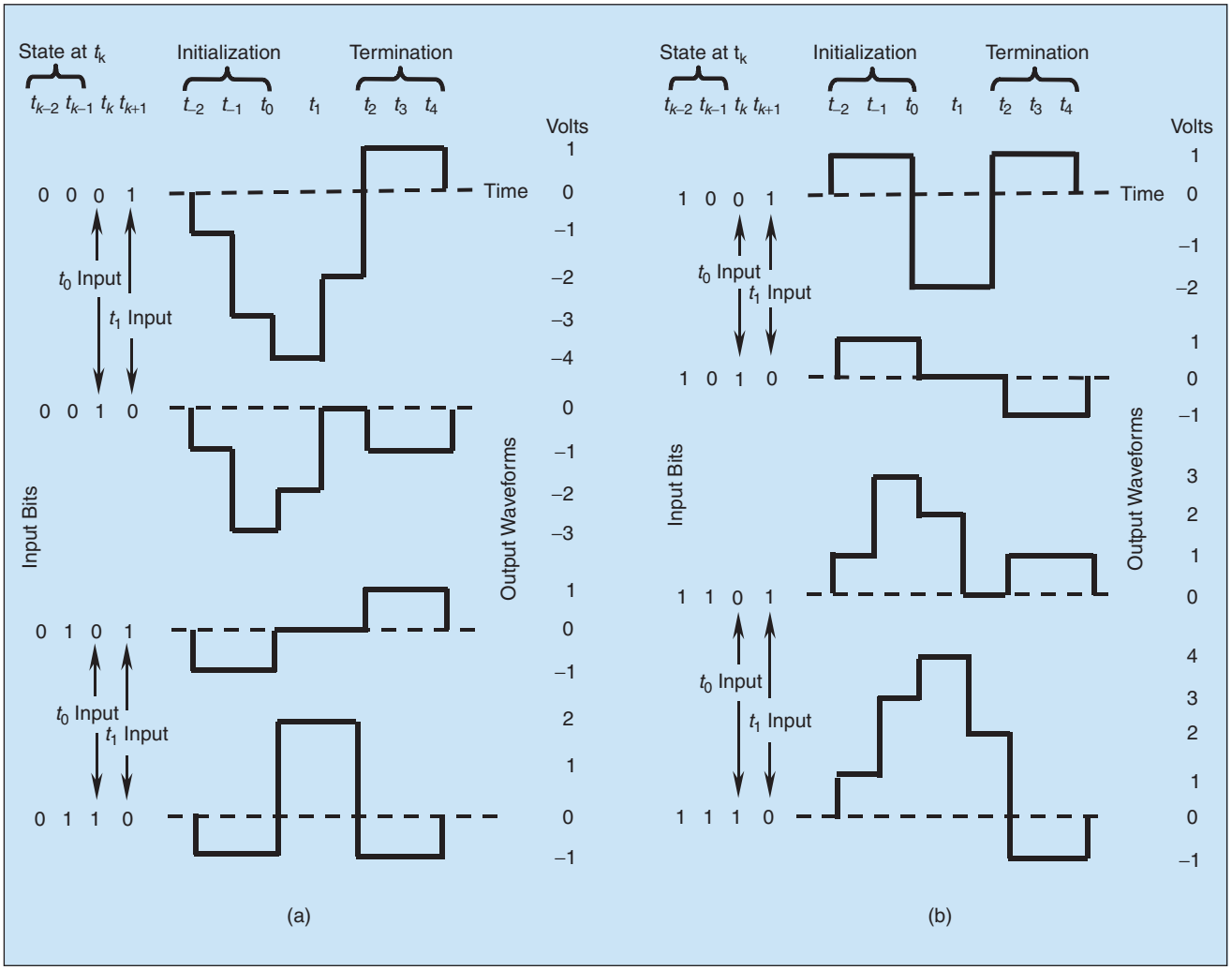


▲ *2. FSM output waveform for the input sequence 0 0 0 1.*

State of the FSM

In general, for any value of γ, an output starting at time t_k is influenced by the binary inputs at earlier times $t_{k-\gamma}, t_{k-\gamma+1}, \ldots, t_{k-1}$. We designate the input sequence of events at these γ time instants in the past as the state (memory of past input bits) of the FSM at time t_k. Since the events are binary, such a system has 2^γ distinct states; and given that $\gamma = 2$ in our model FSM, this model has four possible states. As seen in Figure 3, the state at time t_k (t_0 for this example) is specified by the input bits at times t_{k-2} and t_{k-1} (t_{-2} and t_{-1} for this example). In Figure 3, we use the same convention introduced earlier that the leftmost bit is the earliest bit. The two rightmost bits represent the inputs at times t_k and t_{k+1} (t_0 and t_1 for this example). At time t_{k+1}, the state changes to reflect the input bits at times t_{k-1} and t_k. For a memory of $\gamma = 2$, during the first two bit-time intervals, the FSM-output waveform experiences initialization. During the final two bit-time intervals, the FSM-output waveform, reflecting the ($\gamma = 2$) contents stored in the FSM memory, experiences termination. This explains why each waveform in Figure 3 is longer than its corresponding input by two bit-time intervals. We shall refer to the FSM-output time intervals, between initialization and termination, as the quiescent intervals.

Consider the time interval between t_0 and t_1, which corresponds to the time interval of the third input bit. At the output (see Figure 2), this interval contains the postcursor of the first output pulse (−1 V), the main lobe of the second output pulse (−2 V), and the precursor of the third output pulse (−1 V) yielding a tally of −4 V (assuming no time delay between input and output). In Figure 3, observe that, for this example, the possible voltage levels that an output waveform may assume belong to the set {−4,−3,−2,−1,0,1,2,3,4}. If one were to apply a long sequence of logical zeros to the input of such a system, the resulting output voltage sequence would be {−1,−3,−4,−4,−4,…}. In other words, there would be a buildup to the −4 V level, and then the −4 V level would persist (because the system only has a memory of $\gamma = 2$). Similarly, if one were to apply a long sequence of logical ones to the input of such a system, the resulting output voltage sequence would be {1,3,4,4,4,…}. One can verify this, by starting with the output pulse shapes from Figure 1(c) and appropriately aligning and summing them to



▲ *3. Output waveforms exhibiting the effects of ISI distortion.*

form output waveforms, as was done in Figures 2 and 3. Note that during any quiescent intervals, the output waveform in this example can only take on levels belonging to the even-voltage set {–4,–2,0,2,4}.

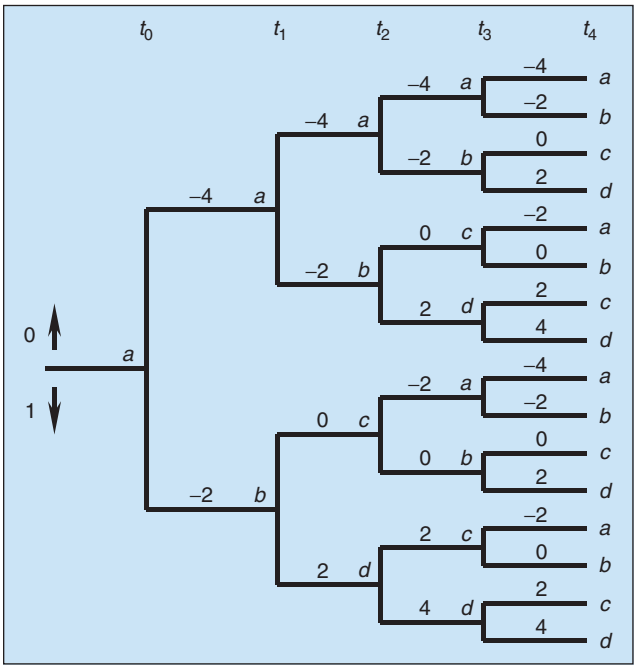
Tree Diagram

Figure 4 shows a so-called *tree diagram*. It is used here to specifically display the possible FSM outputs in the form of transitions (or tree branches) for the purpose of characterizing the system ISI. Each of the nodes labeled a, b, c, or d refers to a state, where $a = 00$, $b = 01$, $c = 10$, and $d = 11$. Time progresses from left to right as shown, and we assume that the system has been initialized such that it starts at time t_0 in the $a = 00$ state. This is equivalent to having preceded each input data sequence with two logical zeros at times t_{-2} and t_{-1} (not shown) so that the output due to any message sequence is always preceded by two of the negative pulses shown in Figure 1(c). The legend on the left of Figure 4 indicates that, from any node, a logical zero FSM input results in an upward tree transition toward a subsequent node. The voltage output level (called a branch word), produced by a transition, is written on its corresponding (horizontal) tree branch. In a similar manner, a logical one FSM input results in a downward tree transition. Note that, for a binary system such as this, only two possible transitions can leave each node.

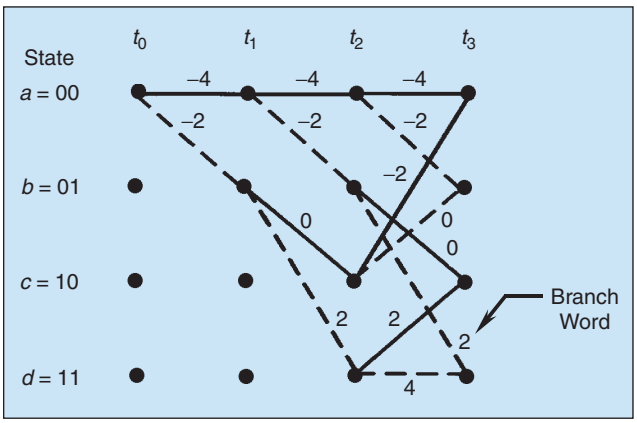
The reader is urged to verify that the output-voltage levels resulting from particular state transitions (see Figure 3) are the same as the branch words corresponding to those same state transitions appearing on the tree diagram of Figure 4. For example, consider the input sequence 0001 in Figure 3. At time t_0, the state is $a = 00$ and the input to the FSM at that time is a logical zero. Hence, at time t_1, the state will still be $a = 00$, and in the interval between times t_0 and t_1 we say that the state has transitioned from a to a. We see that the output voltage during that interval is –4. The next input at time t_1 is a logical one. Hence, in the interval between times t_1 and t_2, the state makes a transition from $a = 00$ to $b = 01$, and the output voltage during that interval is –2. On the tree diagram of Figure 4, we verify that the transition from state a to state a is characterized by the branch word –4 and the transition from state a to state b is characterized by the branch word –2, corresponding to the output-voltage levels seen in Figure 3.

Trellis Diagram

One of the difficulties in working with the tree diagram is that, as time increases, the number of tree branches increases exponentially. The desire to generate a compact pictorial presentation has given rise to the formation of the trellis diagram shown in Figure 5. This diagram can be visualized as a tree diagram (starting at t_0 in the $a = 00$ state) that has been modified by showing only the transitions from t_0 to t_3 and by joining common nodes (a in the lower half of the tree with a in the upper half, b in the lower half with b in the upper half, and so forth). The trellis diagram showing transitions and outputs (branch words) between t_2 and t_3 is repeated for as many clock cycles as desired. This trellis segment, during any time interval (after initialization), is identical to the trellis segment during any other time interval. Thus, for the purpose of characterizing a system, only one such interval of the trellis is needed. Note that a state transition is drawn as a solid line to represent an FSM logical zero input and as a dashed line for a logical one input. Based on the current state and the current input, the encoder moves to the next state along the trellis branch corresponding to the input signal. During this state transition, the encoder outputs the branch word according to the label of that branch. Later, it will be shown how the solid-line/dashed-line convention provides some helpful "bookkeeping" for carrying out the signal equalization and de-



▲ *4. A tree diagram; the transition branches are labeled with output voltage values.*

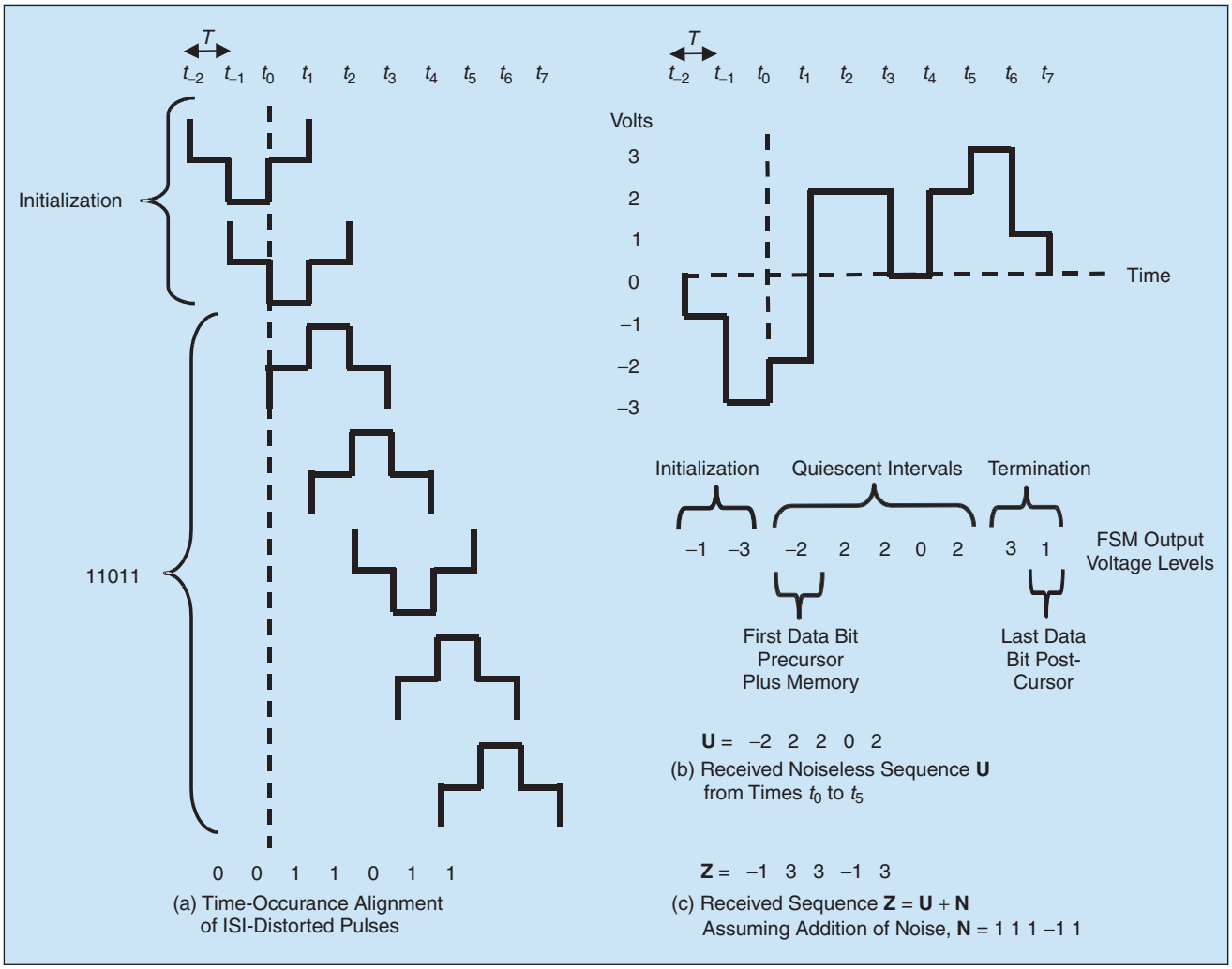


▲ *5. An encoder trellis diagram.*

tection. The process of applying bipolar-pulse inputs to the FSM and generating outputs with memory is very similar to the process of convolutionally encoding a binary data sequence. Therefore, the diagram of Figure 5 (used extensively to describe convolutional encoding) will be referred to as the encoder trellis. This trellis diagram has become the "workhorse" for characterizing a finite state machine.

Example of Using the Encoder Trellis with an ISI-Distorted Data Sequence

Consider a 5-bit data sequence or message, $\mathbf{m} = 1\,1\,0\,1\,1$, represented with the ideal bipolar pulses shown in Figure 1(a). Suppose that each input pulse of the sequence applied to the FSM suffers ISI so that its output is time-stretched over $K = 3$ bit intervals exhibiting a smeared "staircase-like" shape like one of those shown in Figure 1(c). In Figure 6(a) the smeared output pulses, corresponding to data sequence **m** out of the FSM, are aligned according to their occurrence in time to illustrate the superposition of pulse segments at each bit-time interval T. The system is initialized to start at time t_0 in the $a = 00$ state. Hence, the output waveform starts with two smeared negative pulses, one initiated at time t_{-2} and one at time t_{-1}. This is followed by the five smeared pulses corresponding to the message **m**. During each of the nine bit-time intervals (from t_{-2} to t_7) shown in Figure 6(a), the voltage components are algebraically added. In this example with $\gamma = 2$, there are never more than $K = 3$ voltage components in each time interval that contribute to a signal sum. Figure 6(b) shows the resulting nine FSM-output voltage levels. Neglecting initialization and termination, we then denote the five levels during the quiescent intervals as the received distorted (but noiseless) sequence $\mathbf{U} = \{-2,2,2,0,2\}$. (The values of **U** could also have been developed using the concepts of convolution presented earlier.) The multiple levels of the sequence **U** can be thought of as an encoded message that can be displayed as a sequence of transitions (a path) on an encoder trellis like the one in Figure 5, where each unique message **m** will give rise to a unique path on the trellis. For this example, Figure 7 shows the five voltage levels of sequence **U** appearing as a path on such an encoder trellis. In Figure 6(c), we have postulated additive noise levels $\mathbf{N} = \{1,1,1,-1,1\}$ that further corrupt **U** so that



▲ *6. Example of a message distorted by effects of ISI.*

the received sequence, $\mathbf{Z}=\mathbf{U}+\mathbf{N}$, becomes $\mathbf{Z}=\{-1,3,3,-1,3\}$. Values of the noise sequence $\mathbf{N}$ were chosen from the set $\{\pm 1\}$ to keep $\mathbf{Z}$ integer valued, thereby simplifying computations. But in general, the components of $\mathbf{N}$ are characterized by a Gaussian distribution and might take on any value from $-\infty$ to $+\infty$. Likewise for the case of continuous impulse-response waveforms, the values of $\mathbf{U}$ are generally made up of real-valued numbers (not simply integers).

Starting with a received noisy sequence $\mathbf{Z}$, the first goal of any receiving system, demodulation, is to overcome noise corruption and thereby recover an estimate of the transmitted (and distorted) waveform. This task corresponds to finding the most-likely path that represents the FSM output (i.e., sequence $\hat{\mathbf{U}}$) which we anticipate will be a good estimate of the path $\mathbf{U}$ shown on the encoder trellis in Figure 7. The next goals, equalization and detection, are to respectively mitigate the ISI and to make data decisions by relating the chosen path $\hat{\mathbf{U}}$ to a data sequence $\hat{\mathbf{m}}$. In many systems, equalization is accomplished by a filter (typically incorporated within the demodulator) that reverses the channel's nonoptimum (dispersive) properties. In such cases, it is usually quite straightforward to clearly partition the functions of demodulation, equalization, and detection into separate blocks. When the VDA is used for equalization, the functions of demodulation, equalization, and detection are all quite coupled. However, for notational simplicity, we shall refer to the task of selecting $\hat{\mathbf{U}}$ as demodulation and the task of selecting $\hat{\mathbf{m}}$ as equalized detection. Our overall goal, of course, is that $\hat{\mathbf{m}}$ should be a good estimate of the transmitted sequence $\mathbf{m}$. These steps are treated in the sections that follow.

Likelihood Functions

The mathematical foundation of statistical decision theory rests on Bayes' theorem [9]. For communications engineering, the most useful form of Bayes' theorem expresses the a posteriori probability (APP) of a decision, conditioned on a received continuous-valued random variable z usually termed the observation or the test statistic, in the following form:

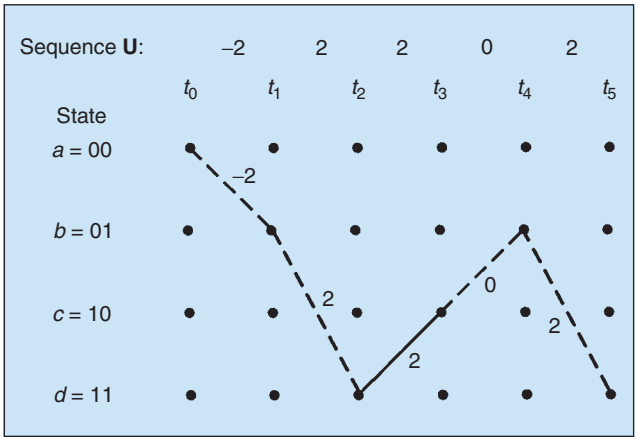
$$P(d=i|z)=\frac{p(z|d=i)P(d=i)}{p(z)}\,i=1,\ldots,M \tag{2}$$

and

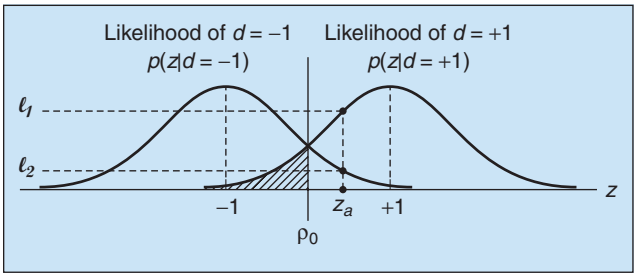
$$p(z)=\sum_{i=1}^{M}p(z|d=i)P(d=i). \tag{3}$$

The APP $P(d=i|z)$ is a conditional probability that the data d belongs to the ith signal class (from a set of M classes). Further, $p(z|d=i)$ is the probability density function (pdf) of the observation z conditioned on the signal class $d=i$. $P(d=i)$, called the a priori probability, is the known (or assumed) probability of occurrence of the ith signal class. Typically, the observation z is a random variable produced at the output of a signal processor such as a matched filter [9], [10] in a receiver, having signal plus noise at its input. The denominator term, $p(z)$ in (2), is the unconditional pdf of the received signal z, taken over the entire space of signal classes, as described by (3). In (2), $p(z)$ is simply a scaling factor that can be neglected, as explained in the next section. For the case of two possible signal classes represented with ideal bipolar pulses such that $i=\{+1,-1\}$ transmitted over an additive white Gaussian noise (AWGN) channel, Figure 8 shows the conditional pdfs referred to as likelihood functions. Note that the AWGN assumption is not necessary for describing Bayes' theorem or the general rules stemming from the theorem. However, since AWGN is the most commonly used noise model in communication systems, we shall use it here and therefore show the likelihood functions in Figure 8 drawn with the familiar Gaussian shapes. The rightmost function $p(z|d=+1)$, called the likelihood of $d=+1$, shows the pdf of the random variable z conditioned on $d=+1$ being sent. The leftmost function

The Viterbi decoding algorithm represents a technique for finding the shortest route through a graph.



▲ *7. Sequence* **U** *shown as a path on the encoder trellis.*



▲ *8. Likelihood functions.*

$p(z|d=-1)$, called the likelihood of $d=-1$, illustrates a similar pdf conditioned on the data $d=-1$ being sent. The abscissa shows the full range, from $-\infty$ to $+\infty$, of possible values of the test statistic z generated at the receiver (because Gaussian noise can take on any value in that range). In Figure 8, one such arbitrary received sample value z_a is shown. A line subtended from z_a intercepts the two likelihood functions yielding the likelihood values $\ell_1 = p(z_a|d=+1)$ and $\ell_2 = p(z_a|d=-1)$. The APP of a received signal can be thought of as the result of an experiment. Before the experiment, there generally exists (in nature) a priori information $P(d=i)$ about the process being examined. The experiment consists of generating a sample z_a of the observable random variable and finding (from Figure 8) the likelihoods ℓ_1 and ℓ_2 of the underlying data d. In (2), the APP values $P(d=+1|z)$ and $P(d=-1|z)$, functions of ℓ_1 and ℓ_2, respectively, can be thought of as "refinements" of our prior knowledge of the data, brought about by having observed the test statistic $z=z_a$.

The Maximum a Posteriori Decision

For the case of only two signal classes, a reasonable starting point for establishing a receiver decision rule from Bayes' theorem in (2) is to decide that the kth data bit d_k (in a sequence) is equal to $+1$ if the APP of the data belonging to each class, $d_k=+1$ and $d_k=-1$, obey the inequality

$$P(d_k=+1|z_k) > P(d_k=-1|z_k). \tag{4}$$

Otherwise, decide that $d_k=-1$. In (4), note that the time index k has been introduced, so that the random variable z and the data d are now expressed in terms of a sample value z_k observed at a particular time t_k and a transmitted data bit d_k corresponding to that same time. We can replace the APPs of (4) with their equivalent expressions stemming from Bayes' theorem in (2), yielding

$$p(z_k|d_k=+1)P(d_k=+1) > p(z_k|d_k=-1)P(d_k=-1) \tag{5}$$

where the only difference is that the inequality in (5) is in the form of likelihoods, and the $p(z)$ term has been cancelled out since it appears on both sides of the inequality. Equation (5) is often rearranged in the form

$$\frac{p(z_k|d_k=+1)}{p(z_k|d_k=-1)} > \frac{P(d_k=-1)}{P(d_k=+1)}, \tag{6}$$

where the left-hand ratio (ℓ_1/ℓ_2) is known as the likelihood ratio, and the decision rule becomes: decide $d_k=+1$ if $\ell_1/\ell_2 > P(d_k=-1)/P(d_k=+1)$. Otherwise decide $d_k=-1$. Equation (6) is often referred to as the likelihood ratio test. It corresponds to making a decision based on comparing a function of the observation z_k to a threshold (right-hand ratio of prior probabilities). That is, decide $d_k=+1$ if the likelihood ratio is greater than the threshold, and otherwise decide $d_k=-1$. Since the test is based on choosing the signal class with maximum a posteriori (MAP) probability, the decision criterion is called the maximum APP (or the MAP) criterion. It is also called the minimum-error criterion, since this criterion yields, on the average, the minimum number of incorrect decisions [10].

The Maximum Likelihood Decision

Very often there is no knowledge available about the a priori probabilities of the signal classes. In such instances, decisions are usually made by assuming that the classes are equally likely. When this is done, the MAP criterion becomes what is known as the ML criterion, and (6) can be written as

$$\frac{p(z_k|d_k=+1)}{p(z_k|d_k=-1)} > 1 \tag{7a}$$

or

$$p(z_k|d_k=+1) > p(z_k|d_k=-1) \tag{7b}$$

or

$$\ell_1 > \ell_2. \tag{7c}$$

Similar to the decision rule developed earlier, the ML decision rule is: decide that $d_k=+1$ if the inequality $\ell_1 > \ell_2$ in (7) is true. Otherwise decide that $d_k=-1$. Equation (7) represents the optimum way of making binary decisions when the a priori probabilities are *not* available. For each data bit at time t_k, this is tantamount to deciding, in the context of Figure 8, that the decision $d_k=+1$ is made if the kth test statistic falls on the right side of the line labeled ρ_0. Otherwise the decision is $d_k=-1$. Note that ρ_0 is the decision threshold mentioned earlier, taking on a zero value for symmetric likelihood functions, bipolar data $d_k=\pm 1$, and zero-mean noise. However, in general, ρ_0 may be nonzero due to, for example, nonequal-magnitude signaling.

Equation (7) describes an optimum rule for making a decision based on an observation (a random variable) made up of binary data plus noise. For systems having a memory of neighboring signals (due to ISI distortion), however, the receiver's signal-processing task needs to be expanded to include equalization because the transmitted pulses can be corrupted by both noise and ISI. The first step for the ML processor is to overcome the noise corruption by choosing the most likely distorted waveform out of the FSM (demodulation). The next step is to mitigate the distortion and relate the chosen waveform to the transmitted data sequence (equalized detection). Note that for this application, demodulation entails choosing from a received sequence $\mathbf{Z}=\{z_k, z_{k+1}, \ldots\}$, the most

likely random sequence $\mathbf{U}=\{u_k, u_{k+1}, \ldots\}$ (not just the most likely random variable u_k), where u_k is a voltage level out of the FSM during the time interval (t_k, t_{k+1}). Thus, the ML rule in (7) needs to be modified slightly so that the demodulation process "looks at" sequences that are longer than one bit in duration. For such sequences, the ML rule can be expressed in the following form [9]:

$$p\left(\mathbf{Z}|\mathbf{U}^{(j')}\right) = \max_j p\left(\mathbf{Z}|\mathbf{U}^{(j)}\right) \tag{8}$$

where $\mathbf{Z}$ is a sequence of observables, $\mathbf{U}^{(j)}$ is the jth candidate sequence (one of the possible ISI-distorted sequences), and $p(\mathbf{Z}|\mathbf{U}^{(j)})$ is the likelihood of $\mathbf{U}^{(j)}$, denoted $\ell(\mathbf{U}^{(j)})$. Assuming binary signaling, the index $j = 1,2,...,2^{K_0}$ identifies a particular binary sequence, where K_0 is the number of bit intervals spanned by the ISI-distorted sequence. The decision rule calls for choosing the j'th sequence $\mathbf{U}^{(j')}$ if its likelihood is the maximum over all possible likelihoods. One can imagine a host of likelihood functions $p(\mathbf{Z}|\mathbf{U}^{(j)})$, one for each possible ISI-distorted sequence $\mathbf{U}^{(j)}$ out of the FSM. Figure 9 represents a simplified visualization wherein a sequence of observables $\mathbf{Z}=\{z_1, z_2\}$ corresponds to two consecutive receptions. An arbitrary received sequence $\mathbf{Z}_a$ is represented as a point in the $z_1 - z_2$ plane. For the purpose of this diagram, consider that the candidate (uncorrupted by noise) sequences $\mathbf{U}^{(j)}$ are the points $\{-1-1, -1+1, +1-1, +1+1\}$ on the plane. Plotted as a surface centered about each candidate sequence is a (Gaussian-shaped) likelihood function $p(\mathbf{Z}|\mathbf{U}^{(j)})$. From the sequence $\mathbf{Z}_a$, a subtended line perpendicular to the $z_1 - z_2$ plane intercepts each of the likelihood functions. The vertical line piercing the surfaces produces four interception points, the highest of which yields the maximum value $p(\mathbf{Z}_a|\mathbf{U}^{(j')})$. When the sequences $\mathbf{Z}$ and $\mathbf{U}$ entail more than two elements, we can only imagine the extension of Figure 9 to a higher-dimensional space (which, of course, we cannot draw).

As noted earlier, the number of possible binary sequences $\{\mathbf{U}^{(j)}\}$ is 2^{K_0}, where K_0 is the number of bit intervals spanned by the sequence. Therefore, if "brute force" ML detection was attempted, it would require an exhaustive comparison of 2^{K_0} likelihoods, representing all the possible distorted sequences that could have been generated by the FSM. Then, the sequence having the largest likelihood would be chosen. This would generally be computationally unfeasible, since the number of possible sequences increases exponentially with K_0. For example, there are over 30 billion possible sequences of length $K_0 = 35$ bits. Later we show how the VDA makes it possible to configure an equalizer/detector that can quickly discard sequences that are extremely unlikely to be the $\mathbf{U}^{(j')}$ sequence of (8). By using the VDA, the detected path (sequence) is chosen from some reduced set (survivor paths) of all possible paths. The VDA configuration is still optimum in the sense that the detected path is the same as the one obtained from an exhaustive search of all possible paths, but the early rejection of unlikely paths reduces the computational complexity.

The Viterbi Decoding Algorithm

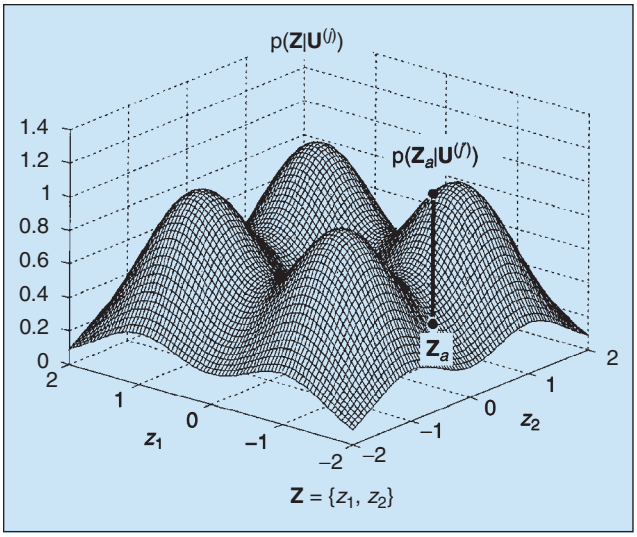
For developing the ML decision rules, consider that bipolar data $d_k = \pm 1$ is transmitted using the ideal pulses in Figure 1(a). Also, consider that ISI-distortion transforms a message sequence $\mathbf{m}=\{d_k, d_{k+1}, \ldots\}$ into the distorted sequence $\mathbf{U}=\{u_k, u_{k+1}, \ldots\}$ and that the noise sequence $\mathbf{N}=\{n_k, n_{k+1}, \ldots\}$ further corrupts the message so that a received signal (observation z_k) at time t_k can be described as

$$z_k = u_k + n_k. \tag{9}$$

Assuming that transmission occurs over an AWGN channel, then n_k is a zero-mean Gaussian random variable with variance σ_n^2 [9]. The pdf of the observation z_k conditioned on u_k can be expressed as

$$p\left(z_k|u_k^{(j)}\right) = \frac{1}{\sqrt{2\pi}\,\sigma_n} \exp\left[-\frac{(z_k - u_k^{(j)})^2}{2\sigma_n^2}\right] \tag{10}$$

where z_k is the kth observation of the received sequence $\mathbf{Z}$, and $u_k^{(j)}$ is the kth value of the ISI-distorted sequence $\mathbf{U}^{(j)}$. Although the channel manifests signal memory, as characterized by the smearing of each transmitted signal, the channel is nevertheless memoryless in the context of the AWGN model. Because the noise is white (thus uncorrelated) and Gaussian, each noise event n_k at time t_k is independent of the noise at any other time [10]. Since the noise affects each transmitted pulse independently of all the other pulses and there are K_0 bit intervals spanned by the ISI-distorted sequence, we can express the likelihood $\ell(\mathbf{U}^{(j)})$ as the product of K_0 marginal pdfs as follows:



▲ 9. Likelihood functions for a random sequence $\mathbf{Z} = z_1, z_2$.

One of the difficulties with the tree diagram is that the number of tree branches increases exponentially with message length.

$$\ell\left(\mathbf{U}^{(j)}\right)=p\left(\mathbf{Z}|\mathbf{U}^{(j)}\right)=\prod_{k=0}^{K_0-1}p\left(z_k|u_k^{(j)}\right)$$
$$=\prod_{k=0}^{K_0-1}\frac{1}{\sqrt{2\pi}\,\sigma_n}\exp\left[-\frac{(z_k-u_k^{(j)})^2}{2\sigma_n^2}\right]$$
$$=\left(\frac{1}{\sqrt{2\pi}\,\sigma_n}\right)^{K_0}\exp\left[-\sum_{k=0}^{K_0-1}\frac{(z_k-u_k^{(j)})^2}{2\sigma_n^2}\right]. \quad (11)$$

Given a received (distorted and noisy) sequence $\mathbf{Z}$, the first task (demodulation) of the MLSE is to determine which of the possible distorted sequences $\{\mathbf{U}^{(j)}\}$ maximizes $\ell(\mathbf{U}^{(j)})=p(\mathbf{Z}|\mathbf{U}^{(j)})$. In the context of a trellis diagram, the problem consists of choosing a path through the trellis (see Figures 5 and 7) such that $\prod_{k=0}^{K_0-1}p(z_k|u_k^{(j)})$ is maximized. Generally, it is computationally more convenient to use the logarithm of the likelihood function since the product of terms in (11) will then become a sum of terms. This transformation is permitted because the logarithm is a monotonically increasing function and thus will not alter the solution to the maximization of (8). Using (11), we can define the log-likelihood function $L(\mathbf{U}^{(j)})$ as follows:

$$L\left(\mathbf{U}^{(j)}\right)=\log\left[\ell\left(\mathbf{U}^{(j)}\right)\right]$$
$$=\log p\left(\mathbf{Z}|\mathbf{U}^{(j)}\right)$$
$$=-\sum_{k=0}^{K_0-1}\left(z_k-u_k^{(j)}\right)^2 \quad (12)$$

where the logarithm has been taken with respect to the base e, and the terms that are independent of $\mathbf{Z}$ have been neglected. The decoder problem can now be described as choosing a path through a trellis such that $L(\mathbf{U}^{(j)})$ is maximized.

From (12) we see that finding the $\mathbf{U}^{(j')}$ which has maximum likelihood over all $\mathbf{U}^{(j)}$ gives rise to describing the ML criterion in terms of minimizing the sum of terms of the form $(z_k-u_k^{(j)})^2$. Thus, we can equivalently describe the ML criterion as one that selects the sequence $\mathbf{U}^{(j')}$ that is closest in Euclidean distance to $\mathbf{Z}$ (minimizes the Euclidean distance between $\mathbf{Z}$ and $\mathbf{U}^{(j)}$). We can express a distance metric D_E as

$$D_E\left(\mathbf{Z},\mathbf{U}^{(j)}\right)=\sum_{k=0}^{K_0-1}\left(z_k-u_k^{(j)}\right)^2. \quad (13a)$$

For ease of computation (and implementation), let us define another convenient measure of similarity or distance, denoted D_V, as follows:

$$D_V\left(\mathbf{Z},\mathbf{U}^{(j)}\right)=\sum_{k=0}^{K_0-1}\left|z_k-u_k^{(j)}\right|. \quad (13b)$$

Using the metric D_E or D_V should give the same decision results because minimizing one minimizes the other. For the MLSE example described in "Example of Using the Encoder Trellis with an ISI-Distorted Data Sequence," we choose a simple measurement of distance between the kth observable z_k and a value $u_k^{(j)}$ conforming to (13b), namely the magnitude of the voltage difference between them. We do this for each candidate u_k at each time t_k, and in the next section, we show how such voltage values are accumulated and used for choosing the ML sequence.

For the AWGN channel, another way to describe the ML criterion is seen by expanding (12), which yields

$$L\left(\mathbf{U}^{(j)}\right)=\sum_{k=0}^{K_0-1}-z_k^2+2\,z_k\,u_k^{(j)}-\left(u_k^{(j)}\right)^2. \quad (14)$$

We can drop the z_k^2 term because it is the same for each candidate decision. Also, $(u_k^{(j)})^2$ is a bias term that can be dropped for all equal-magnitude types of signaling such as antipodal signaling and M-ary phase shift keying (M-PSK). However, it cannot be dropped for non-equal-magnitude signaling such as M-ary pulse-amplitude modulation (M-PAM) and M-ary quadrature-amplitude modulation (M-QAM) [9], [10]. Therefore, for equal-magnitude signaling, we can say that maximizing $\ell(\mathbf{U}^{(j)})=p(\mathbf{Z}|\mathbf{U}^{(j)})$ or maximizing $L(\mathbf{U}^{(j)})$ is equivalent to maximizing the inner product between a sequence $\mathbf{U}^{(j)}$ and the received sequence $\mathbf{Z}$. (The inner product of vectors $A=(a1,a2,...)$ and $B=(b1,b2,...)$ is defined by $a1b1+a2b2+\cdots$.) Thus, for such signals (14) can be simplified so that the ML decision rule becomes: choose the sequence $\mathbf{U}^{(j')}$ that maximizes the correlation between $\mathbf{Z}$ and $\mathbf{U}^{(j)}$ as expressed by

$$\sum_{k=0}^{K_0-1}z_k\,u_k^{(j)}. \quad (15)$$

Equalization/Detection Example

The encoder trellis diagram in Figure 5 was used to describe the candidate transitions (ISI-distorted signals) that can emanate from a four-state machine. A similar trellis, called the decoder trellis shown in Figure 10, is used to describe the equalizer/detector function. At time t_0, the FSM starts in the $a=00$ state by initializing each message in the manner depicted in Figure 6 to provide the detector with starting-state knowledge. (The VDA typically converges to the solution without any starting state knowledge. However, such knowledge makes for a more

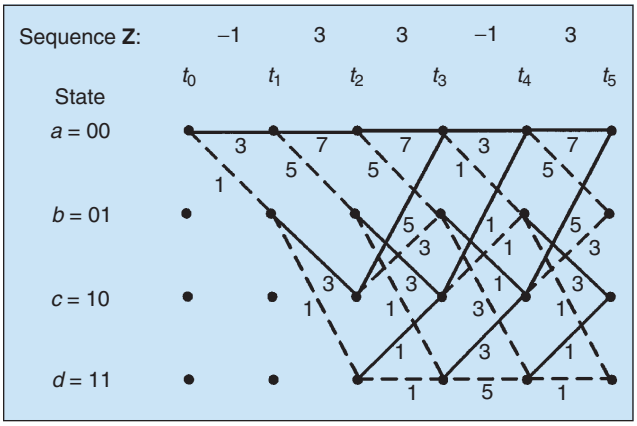
straightforward description of the process.) In a binary example such as this, there are only two possible transitions leaving a state; thus in Figure 10, the full trellis structure need only be shown after γ intervals of time (after time t_2 in this example). For the received sequence $\mathbf{Z}=\{-1,3,3,-1,3\}$ from the example in "Example of Using the Encoder Trellis with an ISI-Distorted Data Sequence," we examine the encoder trellis of Figure 5 while simultaneously working with the decoder trellis of Figure 10. For each time t_k, we label the branches of the decoder trellis, in the interval (t_k, t_{k+1}), with the voltage "distance" between the value z_k received in that interval and the branch word u_k (corresponding to the same transition on the encoder trellis of Figure 5). Each branch word in Figure 5 represents the signal level that would be expected at the FSM output as a result of that transition. In the Figure 10 decoder trellis, the numerical values on the branches are computed "on the fly." That is, as a signal is received, each branch of the decoder trellis is labeled with a metric of similarity (voltage distance) between that received signal and each of the branch words for that time interval. These labels are termed branch metrics. From the sequence $\mathbf{Z}$, shown in Figure 10, we see that the signal received during the first interval (t_0, t_1) is $z_k = z_0 = -1$. From the encoder trellis of Figure 5, we see that a state $a \to a$ transition yields an output branch word of $u_k = -4$. But since we actually received -1, then on the decoder trellis of Figure 10 we label the state $a \to a$ transition during this first interval with the magnitude of the voltage distance $|z_k - u_k|$ between them, namely, the branch metric of 3. Looking at the encoder trellis of Figure 5 again, we see that a state $a \to b$ transition yields an output branch word of $u_k = -2$. But, since we received $z_0 = -1$, then on the decoder trellis of Figure 10, we label the state $a \to b$ transition during this first interval with the branch metric of 1. In summary, the metric entered on a decoder-trellis branch represents the difference between what was received and what "should have been" received had the branch word associated with that branch actually been transmitted. The branch metrics describe distance measurements between a received signal and each of the candidate branch words. We continue labeling the decoder trellis branches in this manner for each time interval. Next, these distance metrics will be summed, as in (13b), in an effort to find the most likely (minimum distance or maximum correlation) path through the trellis.

Once the ML path is selected and its corresponding ISI-distorted voltage values identified, demodulation has been completed. The next task, equalized detection, reconstructs an estimate of the transmitted bit sequence by tracing back through the particular states that the selected path has traversed. The state history is sufficient to yield an estimate of the transmitted bit sequence. For example, during a given time interval, a state $a \to a$ transition could only have been effected by having sent a logical zero, just as a state $a \to b$ transition could only have been effected by having sent a logical one. Notice how the convention of drawing trellis branches with a solid line for an input logical zero, and a dashed line for an input logical one, makes it easy to identify the message.

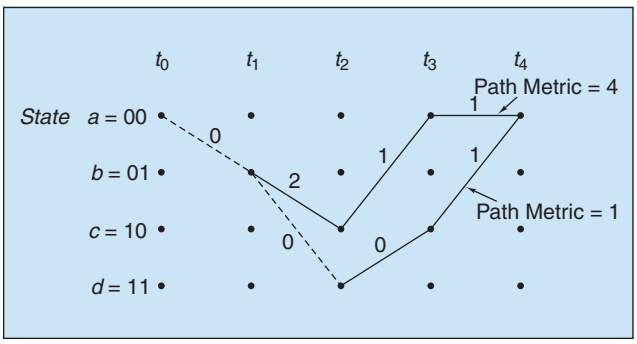
The basis of the VDA is the following observation: If any two distinct paths along the trellis merge to a single state, one of them can always be eliminated in the search for an optimum path. We thereby "prune" the decoder trellis diagram of those paths that are unlikely to have been sent. For example, Figure 11 shows a path that diverges into two distinct paths at time t_1, which then remerge at time t_4 to state $a = 00$. We define the cumulative path metric of a given path at time t_k as the sum of the branch metrics along that path up to time t_k. In Figure 11 the upper and lower paths have cumulative path metrics of four and one, respectively. The upper path cannot be a portion of the ML (minimum distance) path because the lower path, which enters the same state, has a lower cumulative path metric. Thus, the upper path is eliminated, and the cumulative metric of the surviving or "winning" path is termed the path metric or state metric Γ_x; the subscript x labels the state where the path has terminated.

Add-Compare-Select Implementation

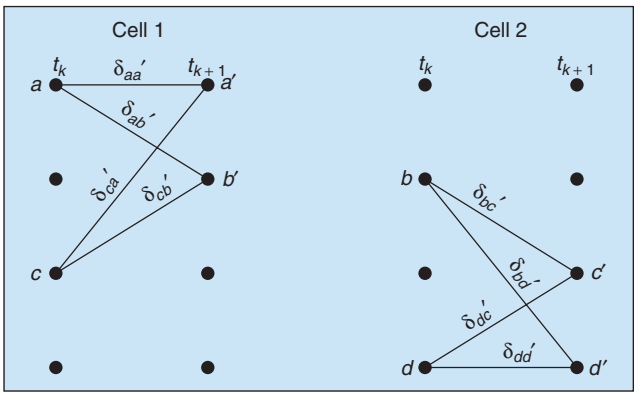
In the context of the decoder trellis diagram of Figure 10, transitions during any one time interval (after time t_2) can be grouped into $2^{\gamma-1}$ disjoint cells, with each cell (sometimes called a butterfly) depicting four possible transitions. For $\gamma = 2$ as in the example being considered,



▲ *10. A decoder trellis diagram.*



▲ *11. Path metrics for two merging paths.*



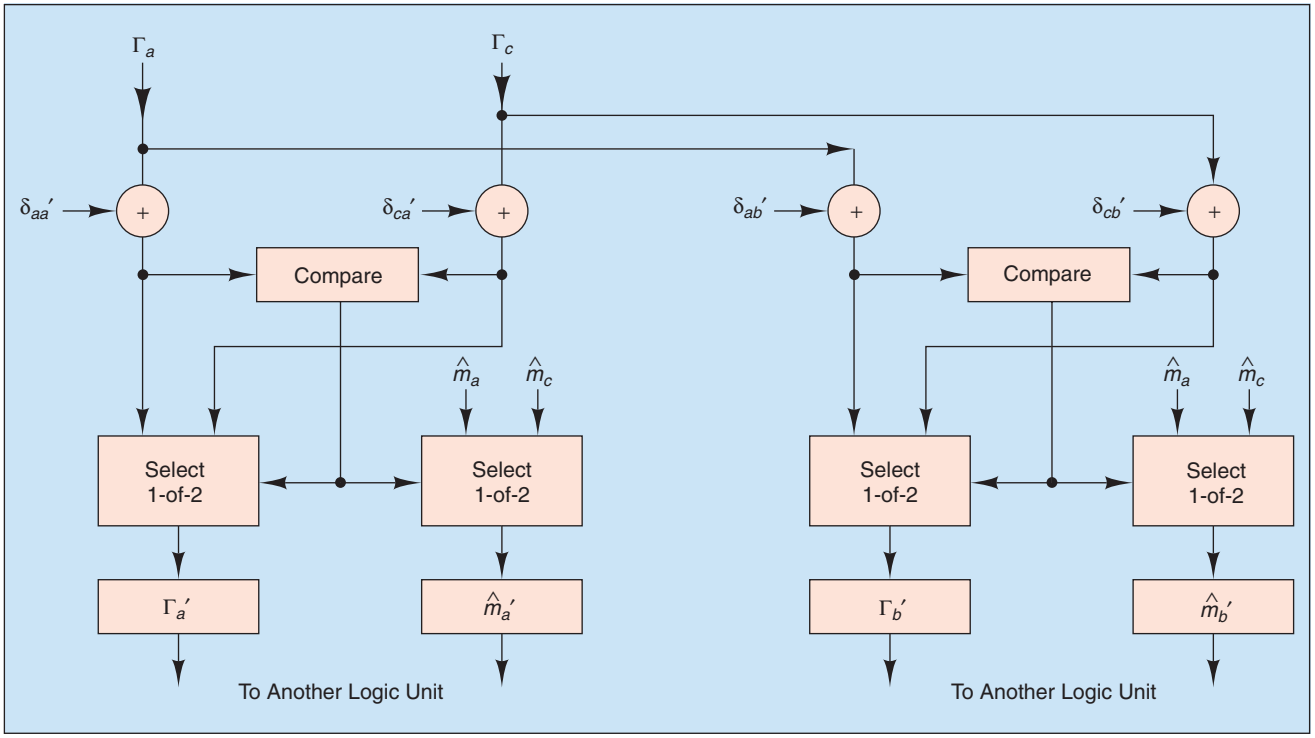
▲ *12. Example of decoder cells.*

there are two cells. These cells are shown in Figure 12, where a, b, c, and d refer to the states at time t_k while a', b', c', and refer to the states at time t_{k+1}. For each transition, the branch metric δ_{xy} is shown, where the subscripts indicate that the metric corresponds to the transition from state x to state y. These cells and the associated logic units that update the state metrics $\{\Gamma_a, \Gamma_b, \ldots\}$, form the basic building blocks of the decoder.

Figure 13 illustrates the logic unit corresponding to cell 1, which executes the special purpose computation called ACS. The state metric $\Gamma_{a'}$ is calculated by first finding two candidate state metrics. One candidate is obtained by adding the state metric Γ_a of state a at the previous time to the branch metric $\delta_{aa'}$. The other candidate is obtained by adding the state metric Γ_c of state c at the previous time to the branch metric $\delta_{ca'}$. The two candidate metrics are then compared in the logic unit of Figure 13, and the largest likelihood (smallest distance) of the two is stored as the new state metric $\Gamma_{a'}$ for state a'. Also stored is the new path history $\hat{m}_{a'}$ for state a', where $\hat{m}_{a'}$ is the message-path history of the state augmented by the data bit associated with the winning path. Also shown in Figure 13 is the ACS logic that yields the new state metric $\Gamma_{b'}$ and the new path history $\hat{m}_{b'}$ for cell 1. At a given time, starting with the state having the smallest Γ_x, the decoder output is formed by tracing the winning branches back to the oldest bit on the path.

Add-Compare-Select as Shown on the Decoder Trellis

Consider the example that was used for describing the VDA in "Equalization/Detection Example." Figure 14 depicts the decoding trellis diagram of Figure 10, where each branch metric δ_{xy} is the voltage distance between the received signal z_k and that of the corresponding branch word u_k at the encoder trellis. Additionally in Figure 14, the trellis indicates a state metric value Γ_x (written as a large bold number) at each state x and for each time from t_1 to t_5. We perform the ACS operation when there are two transitions entering a state as there are for times t_3 and onward. For example, at time t_3 the value of the state metric for state a is obtained by incrementing the state metric $\Gamma_a = 10$ at time t_2 with the branch metric $\delta_{aa'} = 7$ yielding a candidate value of 17. Simultaneously, the state metric $\Gamma_c = 4$ at time t_2 is incremented with the branch metric $\delta_{ca'} = 5$ yielding a candidate value of nine. The select operation of the ACS process selects the largest-likelihood (minimum distance) path metric as the new state metric; hence, for state a at time t_3, the new state metric is



▲ *13. Logic unit that implements the ACS functions corresponding to cell 1.*

$\Gamma_{a'} = 9$. The winning (survivor) path is shown with a heavy line, and the eliminated path is shown with a lighter line. Observe the state metrics from left to right on the trellis of Figure 14. Verify that, at each time, the value of each state metric is obtained by incrementing the connected state metric from the previous time along the winning path (heavy line) with the branch metric between them. Typically, decoding the oldest bit(s) proceeds after the state metric computations have advanced a depth of several times the memory γ into the trellis (resulting in a startup delay). As a simplified (shorter depth) example, we start at time t_5 in Figure 14 and see that the minimum state metric Γ_d has a value of 5. From this state d, the winning path can be traced back to time t_0, and one can verify that the decoded data bits $\hat{m} = 11011$ are the same as those in the original message, by the convention that dashed and solid lines represent logical ones and zeros, respectively. Of course, even though the VDA is optimum [3] in the ML sense, it will occasionally produce errors, particularly for low values of the signal-to-noise ratio.

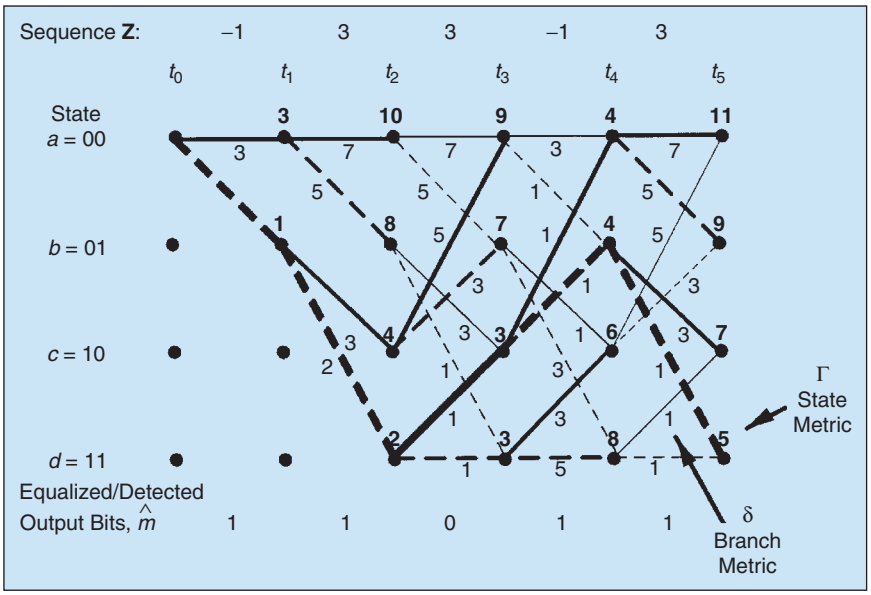
The VDA represents a technique for finding the shortest route through a graph. An interesting solution to this problem was described in 1957 by Minty [11] (long before the development of the VDA). In Minty's solution, he postulated an example of a travel network modeled with strings, where knots represent cities and string lengths represent distances (or costs). He then instructed the reader, "Seize the knot Los Angeles in your left hand and the knot Boston in your right hand, and pull them apart." He went on to suggest, "If the model becomes entangled, have an assistant untie and re-tie knots until the entanglement is resolved. Eventually one or more paths will stretch tight—they then are alternative shortest routes." In 1973, Forney [3] quoted this solution and went on to say (tongue in cheek), "Unfortunately, the Minty algorithm is not well adapted to modern methods of machine computation, nor are assistants as pliable as formerly. It therefore becomes necessary to move on to the Viterbi algorithm ..."

Implementation Complexity and Equalizer Performance

The hardware implementation of the VE typically involves passing information among the following functional units: branch metric unit, ACS arithmetic unit, path memory unit, and output unit. At each discrete time step t_k, the branch metric unit calculates a branch metric δ_{xy} for each distinct trellis branch in the interval (t_k, t_{k+1}). The basic calculations (add, compare, select) take place in the ACS arithmetic unit. One such ACS function must be performed for each of the 2^γ states, where γ is the system memory in bit intervals. In general, the speed of the ACS unit places an upper bound on the speed of the decoder. The storage requirements of the hardware are mostly dictated by the path memory unit that needs to store the decision histories of each state for a length of about five times γ [12]. The complexity of the path memory and ACS units increase exponentially with γ. That is, there is a doubling of complexity for each increase of γ by 1 bit interval. The essence of the Viterbi algorithm is a conceptually simple procedure of identical ACS and traceback operations. The computational burden arises because this relatively simple set of operations must be applied to a large number of states at each discrete time step. The precision of these operations (i.e., the number of bits used in quantizing the values) also plays a role in the overall complexity.

The most straightforward implementation of the algorithm is a completely sequential realization where every state is evaluated in sequence using a single ACS unit. This approach minimizes the required circuitry but also provides the smallest throughput. The other extreme is a fully parallel implementation requiring 2^γ ACS units. This approach requires the most circuitry but provides the largest throughput. Other architectures falling between these two extremes are also possible [13], giving designers a wide range of tradeoff choices. In developing VE hardware for pulses that are each smeared over $K = 9$ bit intervals (memory of $\gamma = 8$), the number of states is $2^8 = 256$ and the implementation is fairly complex, considering that commercially available Viterbi decoder chips are typically limited to 64 states.

Equalizing methods fall into three major classes: 1) linear equalizers (LEs), 2) decision-feedback equalizers (DFEs), and 3) VEs using the MLSE algorithm described in this article. These methods appear in the rank order of least complex to most complex (and least benefi-

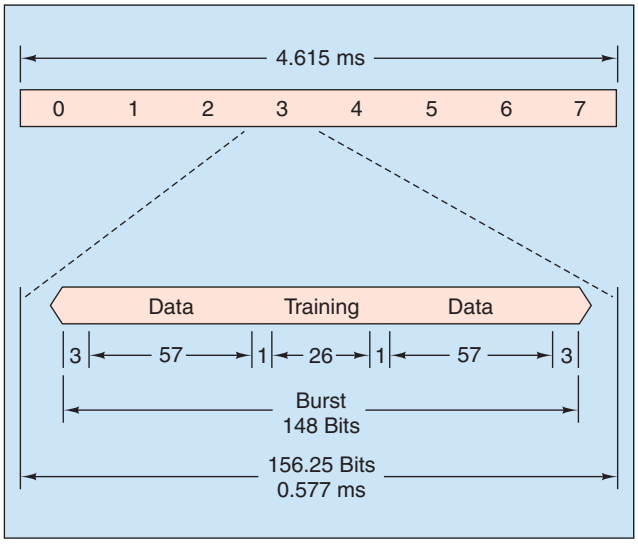


▲ *14. ACS as seen on the decoder trellis.*

cial to most beneficial). A linear equalizer is most often implemented with a transversal filter [9] consisting of a delay line with T-second taps. The tap weights can be adaptively adjusted to improve the shape of the system impulse response compared to that of the unequalized channel. If the channel is described by some nonoptimum frequency-transfer function $H_c(f)$, then one can say that the LE attempts to invert the channel or reverse the ISI distortion by applying the inverse of $H_c(f)$, namely $H_c^{-1}(f)$. Such equalizers yield good performance on channels such as telephone lines where the spectral characteristics are well behaved (no spectral nulls). When such nulls are present, as in mobile wireless communication systems, a linear equalizer performs very poorly. In general spectral nulls in the channel result in a large noise enhancement at the output of the LE [10].

The limitation of the LE to mitigate severe ISI led to the development of nonlinear equalizers with low complexity such as the DFE. This device is made up of two filters: a feedforward filter (very similar to the linear transversal filter) and a feedback filter, which feeds back decisions on previously detected bits so as to remove that part of the ISI from pulses that are currently being demodulated. The ISI being removed was caused by the tails (postcursors) of previous pulses. In effect, the DFE subtracts away from a current pulse, the distortion stemming from earlier pulses. The performance of the DFE is considerably better than that of the LE with the same number of taps.

The advantage of the VE (MLSE algorithm) is that its performance is much superior to the DFE and very much superior the LE. The disadvantage is that its implementation is much more complex. Performance comparisons depend on the exact impulse response of the channel, modulation, and particular circuit implementation. Analysis to determine bit-error probability versus SNR in the presence of ISI is much more involved than finding such analytical expressions in the presence of Gaussian noise.



▲ *15. The GSM and TDMA frame and time slot containing a normal burst.*

Simulations are often preferred to analysis. The interested reader is urged to review the comparative performance results presented by Proakis [10]. The performance advantage of MLSE has motivated a significant amount of research in the area of suboptimal Viterbi algorithms that retain most of the superior performance of MLSE but do so with reduced complexity. These algorithms, referred to as reduced-state sequence estimation techniques, circumvent the exponential growth in complexity by operating on fewer states than are needed in the full trellis (recall that the number of ACS operations is proportional to the number of states). To reduce complexity, the algorithms either combine or discard states according to some criterion [14]-[19], typically carried out adaptively. The combining or discarding operations are responsible for some loss in performance.

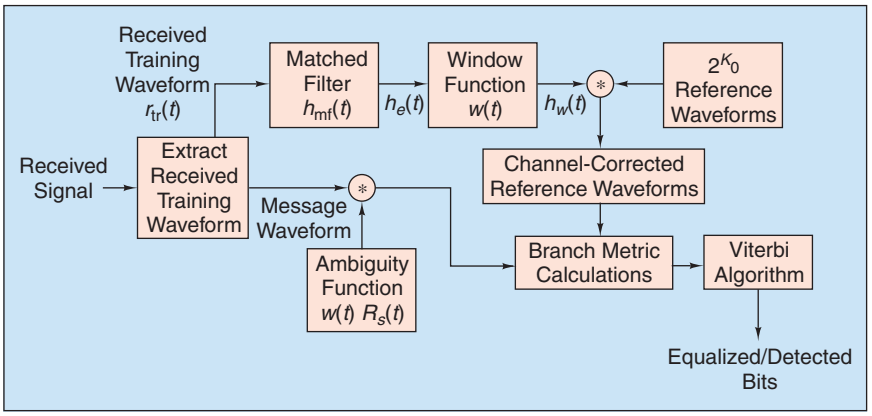
The Viterbi Equalizer as Applied to GSM

Figure 15 shows the structure of a GSM time-division multiple access (TDMA) frame, having a duration of 4.615 ms and comprising eight time slots, one assigned to each active mobile user. A normal transmission burst occupying one slot contains 57 message bits on each side of a 26-bit midamble, called a training or sounding sequence. The slot-time duration is 0.577 ms (corresponding to a slot rate of 1733 slots/s). The purpose of the midamble is to assist the receiver in adaptively estimating the impulse response of the channel (during the time duration of each 0.577 ms slot). For the equalization technique to be effective, the fading characteristics of the channel must not change appreciably during the time duration of one slot. If there are significant fading changes during a time slot, the receiver analysis of the midamble distortion will not yield an accurate estimate of the channel distortion, and compensation efforts will not be effective. Consider, for example, a GSM receiver used aboard a high-speed train, traveling at a constant velocity of $V = 200$ km/h (55.56 m/s). Assume that the carrier frequency is 900 MHz (thus, wavelength $\lambda = 0.33$ m). The distance corresponding to a half-wavelength is traversed in the time

$$T_0 \approx \frac{\lambda/2}{V} \approx 3 \text{ ms}. \tag{16}$$

T_0, referred to as the channel coherence time [9], corresponds approximately to the average time duration that the channel is in a given condition (fading or not fading). Here, T_0 is more than five times greater than the slot time duration of 0.577 ms. In this example, the length of time (3 ms on the average) before significant changes in channel fading characteristics occur is relatively long compared to the time duration of one slot. The choices made in the design of the GSM TDMA frame structure were undoubtedly influenced by the need to preclude fading changes occurring several times within a slot, which could cause the equalizer to be ineffective. Because of the

(fading) nature of a mobile radio system, the GSM receiver must utilize some form of mitigation to combat the fading effects (channel-induced ISI distortion). To accomplish this goal, the VE is typically chosen. Figure 16 illustrates the basic functional blocks that a GSM receiver using a VE implements for estimating the channel impulse response. This estimate is used to provide the equalizer/detector with channel-corrected reference waveforms [20], as explained below. In the final step, the VDA is used to obtain the MLSE of the message bits.



▲ *16. Equalization with the VDA as applied to the GSM system.*

A received signal can be described as a transmitted signal convolved with the impulse response of the channel $h_c(t)$. Let $s_{tr}(t)$ denote the transmitted midamble training sequence previously described and $r_{tr}(t)$ denote the corresponding received training waveform. Thus, ignoring the noise

$$r_{tr}(t) = s_{tr}(t) * h_c(t) \tag{17}$$

where * denotes convolution. At the receiver, since $r_{tr}(t)$ is part of the received normal burst, it is removed from the frame and sent to a filter having impulse response $h_{mf}(t)$ that is matched to $s_{tr}(t)$. This matched filter yields at its output an estimate of $h_c(t)$, denoted $h_e(t)$, developed with the use of (17) as follows:

$$\begin{aligned} h_e(t) &= r_{tr}(t) * h_{mf}(t) \\ &= s_{tr}(t) * h_c(t) * h_{mf}(t) \\ &= R_s(t) * h_c(t) \end{aligned} \tag{18}$$

where $R_s(t) = s_{tr}(t) * h_{mf}(t)$ is the autocorrelation function [9] of $s_{tr}(t)$. If $s_{tr}(t)$ is designed to have a highly peaked (impulse-like) autocorrelation function $R_s(t)$, then $h_e(t)$ will approximately equal $h_c(t)$. Recall, from "ISI-Distorted Output Waveform in Terms of the System Impulse Response" and Figure 1(c), that a system impulse response with a smeared-staircase appearance was assumed for all our examples. In the GSM receiver, however, the channel impulse response $h_c(t)$ must be estimated (at frequent intervals) through the use of the training sequence $r_{tr}(t)$; this estimate is recomputed during each received 0.577 ms slot. Using a windowing function $w(t)$, the estimate $h_e(t)$ is time truncated to form a computationally affordable function $h_w(t)$. This is needed because the complexity of the VE grows exponentially with the number of signaling intervals. The time duration of $w(t)$ must be large enough to compensate for the effect of typical channel-induced ISI. We designate the term K_0 in units of bit intervals to represent the sum of two dispersion (smearing) contributions. The first contribution is K_{CISI}, corresponding to the controlled and fairly predictable ISI caused by Gaussian-shaped filtering of the baseband waveform. (GSM uses minimum-shift keying (MSK) modulation of the carrier [9], after baseband Gaussian-shaped filtering.) The second contribution is K_C, corresponding to the channel-induced ISI caused by multipath propagation. Therefore, K_0 can be written as

$$K_0 = K_{CISI} + K_C. \tag{19}$$

The GSM receiver is required to provide mitigation against any ISI-distortion which might cause signal dispersion of about 15 to 20 μs. Since the bit duration is 3.69 μs in GSM, then it follows that K_0 entails a dispersion of about 4 to 6 bit intervals. For each K_0-bit grouping in the message, the function of the VE is to find the most likely K_0-bit sequence out of the 2^{K_0} possible sequences that might have been transmitted. Determining the most likely transmitted K_0-bit message sequence requires that 2^{K_0} meaningful references be created by modifying (or distorting) the 2^{K_0} candidate sequences (reference waveforms) in the same way that the channel has distorted the transmitted slot. Therefore, as indicated in Figure 16, the 2^{K_0} reference waveforms are convolved with the windowed estimate of the channel impulse response $h_w(t)$ to generate the distorted or so-called channel-corrected reference waveforms.

Next, the channel-corrected reference waveforms are compared with the received message waveforms to yield metric calculations. Before the comparison takes place, however, a "fine-tuning" step preconditions the message waveforms to compensate for the fact that $h_e(t)$ is only approximately equal to $h_c(t)$. The message waveforms are convolved with the known windowed autocorrelation function $w(t)R_s(t)$ (often called the ambiguity function), transforming them in a manner comparable to the transformation applied to the reference waveforms. This preconditioned message signal is compared with all possible 2^{K_0} channel-corrected reference signals. Metrics are computed using the VDA in a manner similar to that described earlier as the VE, and the ML estimate of the transmitted message sequence is thereby obtained.

Note that many equalizing techniques use filters to compensate for the nonideal properties of a channel. That is, equalizing filters at the receiver attempt to modify the distorted waveforms. However, the operation of a VE is quite different. It entails making channel measurements to estimate $h_c(t)$ and then adjusting the receiver by modifying its reference waveforms according to the channel environment. The goal of such adjustments is to enable the receiver to make good data estimates from the received message waveforms. With a VE, the distorted message waveforms are not reshaped or directly modified (with the exception of the preconditioning step); instead the mitigating technique is for the receiver to "adjust itself" in such a way that it can better deal with the distorted waveforms.

Conclusion

This article shows how the VDA can be used for signal equalization and detection in a way that is quite different from the usual equalization approach of adjusting a received signal via filtering to "reverse" the distortion. With the VDA, the receiver "adjusts itself" so as to make good data estimates from a distorted sequence of waveforms. The main goal of this article has been to provide insight as to how the VDA works and why it is a useful tool for equalizing and detecting signals that can be modeled as outputs from an FSM. For further study of the VE as well as equalization in general, Proakis [10] offers detailed analytical treatments. Additionally, [21]-[23], in the context of decoding trellis codes, provide comprehensive overviews of the VDA and trellises in general.

Bernard Sklar has 50 years of electrical engineering experience at companies that include Hughes Aircraft, Litton Industries, and The Aerospace Corporation. At Aerospace, he helped develop the MILSTAR satellite system and was the principal architect for EHF Satellite Data Link Standards. He is currently the head of Advanced Systems at Communications Engineering Services. He has taught engineering courses at several universities including the University of California, Los Angeles and the University of Southern California and has presented numerous training programs throughout the world. He has published and presented many technical papers. He is the recipient of the 1984 Prize Paper Award from the IEEE Communications Society for his tutorial series on digital communications, and he is the author of the book, *Digital Communications: Fundamentals and Applications* (2nd ed., Prentice-Hall, 2001). His academic credentials include a B.S. in math and science from the University of Michigan; an M.S. in electrical engineering from the Polytechnic Institute of Brooklyn, New York; and a Ph.D. in engineering from the University of California, Los Angeles.

References

[1] A.J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[2] A.J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 751-772, Oct. 1971.

[3] G.D. Forney, Jr., "The Viterbi algorithm," *Proc IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[4] J.F. Hayes, "The Viterbi algorithm applied to digital data transmission," *IEEE Commun. Mag.*, vol. 13, no. 2,pp. 15-20, Mar. 1975.

[5] H. Kobayashi, "Application of probabilistic decoding to digital magnetic recording systems," *IBM J. Res. Develop.*, vol. 15, pp. 64-74, Jan. 1971.

[6] H. Kobayashi, "Correlative level coding and maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 586-594, Sept. 1971.

[7] J.K. Omura, "Optimal receiver design for convolutional codes and channels with memory via control theoretical concepts," *Info. Sci.*, vol. 3, pp. 243-266, July 1971.

[8] G.D. Forney, Jr., "Maximum likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363-378, May 1972.

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

[10] J.G. Proakis, *Digital Communications*, 4th ed. New York: McGraw Hill, 2001.

[11] G.J. Minty, "A comment on the shortest-route problem," *Oper. Res.*, vol. 5, pp. 724, Oct. 1957.

[12] J.A. Heller and I.M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 835-848, Oct. 1971.

[13] P. Gulak and T. Kailath, "Locally connected VLSI architectures for the Viterbi algorithm," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 527-537, Apr. 1988.

[14] T. Hashimoto, "A list-type reduced constraint generalization of the Viterbi algorithm," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 866-876, Nov. 1987.

[15] J. Livingston and S.G. Wilson, "Reduced state sequence estimation: Performance loss and trellis structure," in *Proc. Conf. Info. Science and Systems*, Johns Hopkins University, Mar. 1989, pp. 193-198.

[16] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Trans. Commun.*, vol. 37, pp. 428-436, May 1989.

[17] M.V. Eyuboglu and S.U.H. Quereshi, "Reduced-state sequence estimation for coded modulation on intersymbol interference channels," *IEEE J. Select. Areas Commun.*, vol. 7, no. 6, pp. 989-995, Aug. 1989.

[18] G.J. Pottie and D.P. Taylor, "A comparison of reduced complexity decoding algorithms for trellis codes," *IEEE J. Select. Areas Commun.*, vol. 7, no. 9, pp. 1369-1380, Dec. 1989.

[19] G. Allan and S. Simmons, "A VLSI implementation of an adaptive effort low-power Viterbi decoder for wireless communications," in *Proc. IEEE Canadian Conf. Electrical and Computer Engineering*, Toronto, Canada, 2001, pp. 1183-1188.

[20] L. Hanzo and J. Stefanov, "The Pan-European digital cellular mobile radio system—Known as GSM," in *Mobile Radio Communications*, R. Steele, Ed. London: Pentech Press, 1992, chap. 8.

[21] C. Schlegel, *Trellis Coding*. New York: IEEE Press, 1997

[22] S.B. Wicker, *Error Control Systems for Digital Communication and Storage*. Upper Saddle River, NJ: Prentice Hall, 1995.

[23] E. Biglieri, D. Divsalar, P. McLane, and M. Simon, *Introduction to Trellis-Coded Modulation with Applications*. New York: Macmillan, 1991.