Proof of the sampling theorem: By assumption, $s_{\mathcal{F}}(f) = 0$, $f \notin [-B, B]$. Hence, we may think of $s_{\mathcal{F}}(f)$ as being obtained by multiplying its periodic extension, say $\tilde{s}_{\mathcal{F}}(f)$, with $1_{[-B,B]}(f)$. The periodic extension may be written as a Fourier series, yielding

$$s_{\mathcal{F}}(f) = \tilde{s}_{\mathcal{F}}(f) 1_{[-B,B]}(f) = \sum_k A_k e^{+j\frac{2\pi}{2B}fk} 1_{[-B,B]}(f).$$

Taking the Fourier transform on both sides, using

$$1_{[-B,B]}(f) \Leftrightarrow \frac{1}{T}\mathrm{sinc}(\frac{t}{T}), \quad T = \frac{1}{2B},$$

and the time shift property of the Fourier transform

$$h(t-\tau) \Leftrightarrow h_{\mathcal{F}}(f) e^{-j2\pi f\tau},$$

we obtain

$$s(t) = \sum_k \frac{A_k}{T} \,\mathrm{sinc}\left(\frac{t + kT}{T}\right).$$

We still need to determine $\frac{A_k}{T}$. It is straightforward to determine A_k from its definition, but it is easier to observe that if we plug in $t = nT$ on both sides of the expression above we obtain $s(nT) = \frac{A_{-n}}{T}$. This completes the proof. To show that it is straightforward, we also determine A_k from the definition:

$$A_k = \frac{1}{2B}\int_{-B}^{B} s_{\mathcal{F}}(f) e^{-j\frac{2\pi}{2B}kf} df = \frac{1}{2B}\int_{-\infty}^{\infty} s_{\mathcal{F}}(f) e^{-j\frac{2\pi}{2B}kf} df = Ts(-kT),$$

where the first equality is the definition of the Fourier coefficient A_k, the second uses the fact that $s_{\mathcal{F}}(f) = 0$ for $f \notin [-B, B]$, and the third is the inverse Fourier transform evaluated at $t = -kT$. □

Appendix 5.B Power Spectral Density

In this appendix e derive an expression for the power spectral density of a general signal in the form

$$X(t) \;=\; \sum_{i=-\infty}^{\infty} X_i \psi(t - iT - \Theta), \tag{5.10}$$

where $\psi(t)$ is some unit-energy pulse, $\{X_i\}_{i=-\infty}^{\infty}$ is a discrete-time wide-sense stationary sequence, and Θ is a random variable that is uniformly distributed over the interval $[0, T)$. Except for the random delay Θ and the fact that the symbol sequence is infinite, this signal has the general form of symbol-by-symbol on a pulse train. Accounting for a random delay Θ is both realistic and necessary. It is realistic since without it it we would imply that an observer of the signal $X(t)$ uses a time reference known to the sender and that the sender emits the signal in such a way that by the time it reaches the observer it has a predetermined position on the time line. In reality, an observer expects a random shift in time due to clock misalignments but also due to the time it takes for the signal to reach the observer. The reason why we have not mentioned Θ so far is that its presence up until now would have made no difference other than making the notation more combersome. Now we need Θ since without it the stochastic process $X(t)$ is not wide-sense stationary and its power spectral density is not defined.

The first step towards the power spectral density is to compute the correlation $R_X(t+\tau, t) := E[X(t+\tau)X^*(t)]$. It will be expressed as a function of

$$R_X[i] = E[X_{j+i}X_j] \tag{5.11}$$

and

$$R_\psi(v) = \frac{1}{T}\int_{-\infty}^{\infty} \psi(\alpha+v)\psi(\alpha)d\alpha : \tag{5.12}$$

$$\begin{aligned}
R_X(t+\tau,t) &= E[X(t+\tau)X^*(t)] \\
&= E[\sum_{i=-\infty}^{\infty} X_i\psi(t+\tau-iT-\Theta) \sum_{j=-\infty}^{\infty} X_j^*\psi^*(t-jT-\Theta)] \\
&= E[\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} X_iX_j^*\psi(t+\tau-iT-\Theta)\psi^*(t-jT-\Theta)] \\
&= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} E[X_iX_j^*]E[\psi(t+\tau-iT-\Theta)\psi^*(t-jT-\Theta)] \\
&= \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} R_X[i-j]E[\psi(t+\tau-iT-\Theta)\psi^*(t-jT-\Theta)] \\
&= \sum_{k=-\infty}^{\infty} R_X[k]\sum_{i=-\infty}^{\infty}\frac{1}{T}\int_0^T \psi(t+\tau-iT-\theta)\psi^*(t-iT+kT-\theta)d\theta \\
&= \sum_{k=-\infty}^{\infty} R_X[k]\frac{1}{T}\int_{-\infty}^{\infty} \psi(t+\tau-iT-\theta)\psi^*(t-iT+kT-\theta)d\theta.
\end{aligned}$$

Hence

$$R_X(\tau) = \sum_{k=-\infty}^{\infty} R_X[k]\frac{1}{T}R_\psi(\tau-kT), \tag{5.13}$$

where, with a slight abuse of notation, we have written $R_X(\tau)$ instead of $R_X(t+\tau,t)$ to emphasize that $R_X(t+\tau,t)$ depends only on the difference τ between the first and the second variable. It is straightforward to verify that $E[X(t)]$ does not depend on t either. Hence $X(t)$ is a wide-sense stationary process.

The power spectral density S_X is the Fourier transform of R_X. Hence,

$$S_X(f) = \frac{|\psi_{\mathcal{F}}(f)|^2}{T} \sum_k R_X[k] \exp(-j2\pi f T).$$

Notice that the summation is the discrete-time Fourier transform of $\{R_X[k]\}_{k=-\infty}^{\infty}$ evaluated at fT.

In many cases of interest $\{X_i\}_{i=-\infty}^{\infty}$ is a zero-mean iid sequence. Then $R_X[k] = \mathcal{E}\delta_k$ where $\mathcal{E} = E[|X_j|^2]$ and the formulas simplify to

$$R_X(\tau) = \mathcal{E} R_\psi(\tau) \tag{5.14}$$

$$S_X(f) = \mathcal{E}\frac{|\psi_{\mathcal{F}}(f)|^2}{T}. \tag{5.15}$$

Example 47. *When $\psi(t) = \sqrt{\frac{1}{T}} sinc(\frac{t}{T})$ and $R_X[k] = \mathcal{E}\delta_k$, the spectrum is $S_X(f) = \mathcal{E}1_{[-B,B]}(f)$, where $B = \frac{1}{2T}$. By integrating the power spectral density we obtain the power $2B\mathcal{E} = \frac{\mathcal{E}}{T}$. This is consistent with our expectation: When we use the pulse $sinc(\frac{t}{T})$ we expect to obtain a spectrum which is flat for all frequencies in $[-B, B]$ and vanishes outside this interval. The energy per symbol is $\mathcal{E}$. Hence the power is $\frac{\mathcal{E}}{T}$.* □

Chapter 6

Convolutional Coding and Viterbi Decoding

In Chapter 5 we have focused on how to choose the $\psi_i(t) = \psi(t - iT)$ that goes into the general expression $\sum s_i \psi_i(t)$. In this chapter we focus on the symbols s_i.

At the transmitter we will do convolutional coding. The receiver will implement the Viterbi algorithms – a neat and clever way to decode efficiently in many circumstances. To analyze the bit-error probability we will introduce a few new tools, notably detour flow graphs and generating functions.

The signals that we will construct will have the following property: The transmitter and the receiver adapt naturally to the number k of bits that need to be communicated and the duration of the transmission grows linearly with the number of bits; the bandwidth is constant (independently of the number of bits transmitted) and the encoding and decoding operations are done at low complexity and essentially "on the fly." As usual, we assume that signals are passed through the AWGN channel.

6.1 The Transmitter

Like in bit-by-bit on a pulse train, we assume that the entire transmitted signal has the form[1]

$$s_i(t) = \sum_{j=1}^{n} s_{ij} \psi(t - jT),$$

[1]For most of what we do in this chapter we could assume that the signal has the general form $s_i(t) = \sum_{j=1}^{n} s_{ij} \psi_j(t)$. We prefer to be more specific and focus to situations of real practical interest.

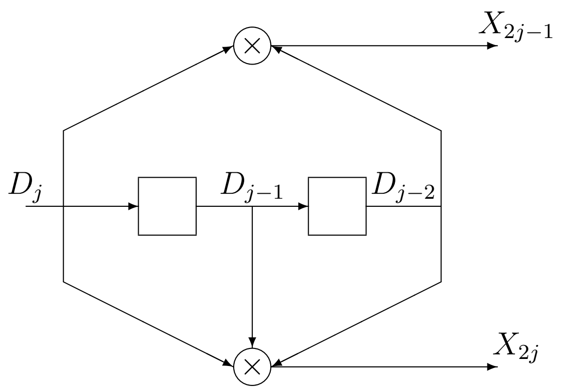


Figure 6.1: Rate $\frac{1}{2}$ convolutional encoder.

where $i \in \{0, 1, \ldots, 2^k - 1\}$ for some integer k,

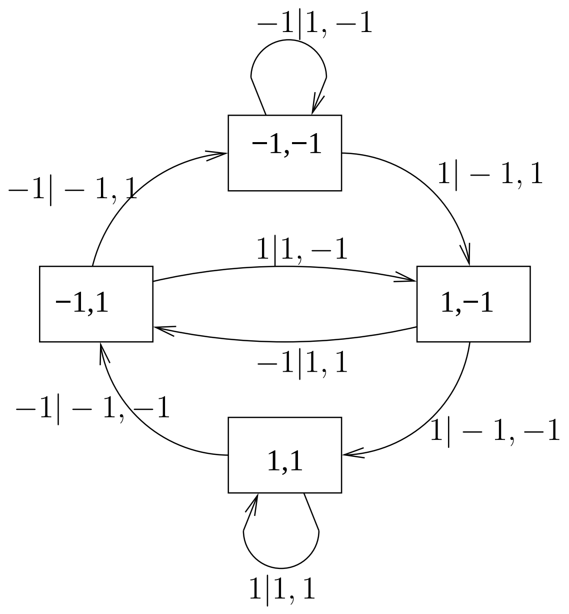
$$s_{ij} = x_{ij}\sqrt{E_s}$$
$$x_{ij} \in \{\pm 1\}.$$

In both cases we will go straight from the bit sequence to the symbol sequence without explicitly passing through the message index i that labels $s_i(t)$. Unlike for bit-by-bit on a pulse train, $k < n$. This implies that when the index i runs over all possible 2^k values, the *codeword* $\boldsymbol{x} = (x_{i1}, \ldots, x_{in})$ does *not* range over all possible 2^n n-length sequences with value in $\{\pm 1\}^n$. Only a subset of such sequences are used. This is what coding is all about. It helps since the fact that not all sequences with components taking value in ± 1 produce a valid signal decreases the chances that a transmitted signal plus noise looks more like an alternative signal rather than the original. In bit-by-bit on a pulse train we would not speak of coding since any n tuple in $\{\pm \mathcal{E}_b\}^n$ is a valid signal space point. For block-orthogonal signaling all components of $\boldsymbol{s}_i$ except one are zero. This is a form of coding.

The next step is to specify the signal space points that we use. We specify our 2^k sequences $x_{i1}, \ldots, x_{in}$ by means of an *encoder*. For our encoder, depicted in Figure **??**, $n = 2k$.

The encoder input is a uniformly distributed k-length random vector $\boldsymbol{D} = (D_1, \ldots D_k)$ that models the information bits (the source output) to be transmitted. The k encoder output pairs are serialized to form an n-length random sequence $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ used to modulate the pulse train. Hence $\boldsymbol{X}\sqrt{E_b}$ is the corresponding signal space point. The multiplication symbol in Figure **??** represents the usual multiplication over $\mathbb{R}$. However, this operation is actually the addition over the field that one can define on the set $\{\pm 1\}$. Hence the encoder is linear. More on this in problem 6.

There are alternative ways to describe the encoder. One such alternative way that will turn out to be useful in determining the error probability is by means of the state diagram shown below.



This is a four state machine. Each box represents a state. States are labeled by the content of the two shift registers of the encoder (previous picture). Arrows represent transitions. There are two possible transitions from each state. The input symbol D_j decides which of the two possible transitions is taken at time j. Transitions are labeled by $D_j|X_{2j-1}, X_{2j}$.

We agree that initially the encoder is in some arbitrary but fixed state. We assume that this state is $(1, 1)$.

The following example shows a source output sequence of length $k = 5$ and the corresponding encoder output sequence of length $n = 10$.

D_j	1	-1	-1	1	1
X_{2j-1}, X_{2j}	$1, 1$	$-1, -1$	$-1, 1$	$-1, 1$	$-1, -1$
j	1	2	3	4	5

6.2 The Receiver

Let $\| s_i \|^2 = \sum_{j=1}^{n} E_s x_{ij}^2 = nE_s$ be the signal's energy (the same for all signals).

A maximum likelihood (ML) decoder decides for one of the i that maximizes

$$\langle r, s_i \rangle - \frac{nE_s}{2},$$

where the second term is irrelevant since it does not depend on i and r is the received signal.

Hence the ML decoder picks (one of) the sequence(s) $s_{i1}, \ldots, s_{in}$ that maximizes

$$\begin{aligned}
&\int r(t) \sum_{j=1}^{n} s_{ij} \psi(t - jT) dt \\
&= \sum_{j=1}^{n} s_{ij} \int r(t) \psi(t - jT) dt \\
&= \sum_{j=1}^{n} s_{ij} y_j \\
&= \sqrt{E_s} \sum_{j=1}^{n} x_{ij} y_j
\end{aligned}$$

where we have defined

$$y_j = \int r(t) \psi(t - jT) dt.$$

Recall that y_j is the output at time jT of the filter with impulse response $\psi(-t)$ and input $r(t)$.

The difficulty in finding one of the i that maximizes $\langle \boldsymbol{x}_i, \boldsymbol{y} \rangle$ is that it appears at first that we have to test all 2^k such inner products. Typically k is much larger than 100. If $k = 100$, with a computer that does 10^9 inner products $\langle \boldsymbol{x}_i, \boldsymbol{y} \rangle$ in a second it takes roughly (using 2^{30} to approximate 10^9) $2^{100}/2^{30} = 2^{70}$ seconds to compute all inner products. This makes almost 4×10^{13} years. The universe is only 20×10^9 years old!

What we need is a method that finds the maximum $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ by making a number of operations that grows linearly (as opposed to exponentially) in k. By cleverly making use of the encoder structure we will see that this can be done. The result is the Viterbi algorithm.

To describe the Viterbi algorithm (VA) we need the notion of a trellis. The trellis is an unfolded transition diagram that keeps track of the passage of time. If we assume that we start at state $(1, 1)$, that we transmit $k = 5$ source digits, and that we append 2 *dummy* source digits to ensure that at the end of the transmission the encoder is again in state $(1, 1)$, the trellis diagram looks as in the top drawing of figure 6.2. For each index i, there is exactly one path in the trellis between the *root* (the leftmost node) and the *toor*[2] (the rightmost node) which is labeled by the sequence $\boldsymbol{x}_i$.

Unlike in the state diagram, in the mentioned trellis we have labeled transitions with the encoder output symbols only. We have dropped the encoder input from the label since we have ordered states in such a way that the lower branch leaving a state at depth j corresponds to $D_j = 1$. Hence it is straightforward to associate a path in the trellis with the corresponding source sequence.

To decode we do the following. Let $\boldsymbol{y} = (y_1, y_2 \ldots y_n)^T$ be the n tuple of matched filter output symbols. Each $\boldsymbol{x}_i$ corresponds to a path in the trellis and the sequence of labels

[2]Toor is root read backwards.

that we read out along that path is exactly $\boldsymbol{x}_i$. We use $\boldsymbol{y}$ to relabel the path in the trellis corresponding to $\boldsymbol{x}_i$. Specifically instead of $x_{i,2j-1}, x_{i,2j}$ we now write the *branch metric* $\langle (x_{i,2j-1} x_{i,2j})^T, (y_{2j-1}, y_{2j})^T \rangle$. Then, by adding the path metric along the path corresponding to $\boldsymbol{x}_i$ we obtain $\sum_j \langle (x_{i,2j-1} x_{i,2j})^T, (y_{2j-1}, y_{2j})^T \rangle = \langle \boldsymbol{x}_i, \boldsymbol{y} \rangle$. We call this the *path metric.* The path metric is the sum of the branch metric along a specific path between root and toor. The second trellis in Fig. 6.2 has been labeled with branch metrics.

Maximum likelihood decoding now amounts to finding (one of) the path(s) with the largest path metrics. The VA is an efficient way to do this. It is convenient to think of the trellis as a road map with branch metrics instead of distances. The job of the VA is to find the longest (not the shortest) path between root and toor. How to do this is best explained by means of an example. (See example in class and bottom two trellises in Fig. 6.2).

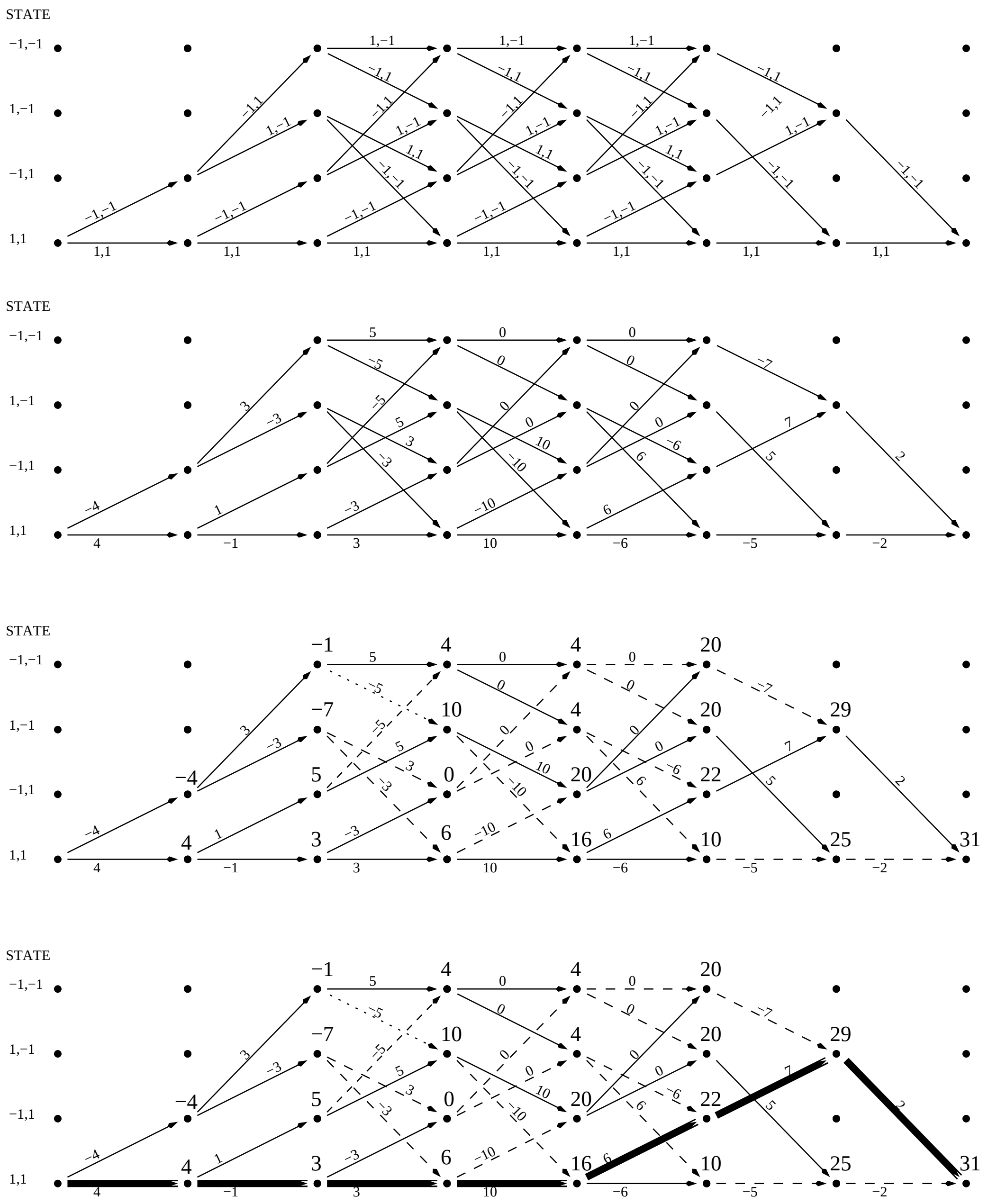


Figure 6.2: The Viterbi algorithm. Top figure: Trellis representing the encoder. The upper transition leaving a state corresponds to source symbol -1, the lower transition to source symbol 1. Transitions are labeled with the corresponding output symbols; Second figure: Transitions have been labeled with the branch metric corresponding to the received sequence $(1,3),(-2,1),(4,-1),(5,5),(-3,-3),(1,-6),(2,-4)$ (parentheses have been inserted for convenience only); Third figure: Each state has been labeled with the metric of the surviving path and non-surviving transitions have been dashed; Fourth figure: Tracing back from the end we find the decoded path (bold). It corresponds to source sequence $1,1,1,1,-1,1,1$.

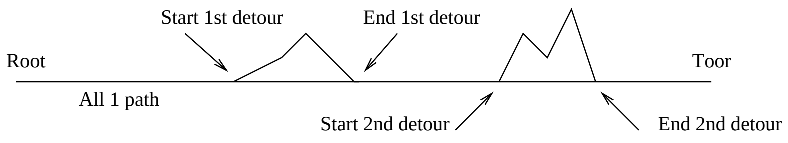
6.3 Bit-Error Probability

We assume that the initial state is $(1, 1)$ and we transmit a number k of bits.

As we have done so far, we determine (an upper bound to) the probability of error by first conditioning on a fixed transmitted signal. It will turn out that our expression does not depend on the selected transmitted signal.

Each signal that can be produced by the transmitter corresponds to a path in the trellis. The path corresponding to the signal that we use to bound the bit error probability will be called the *reference* path. For now the reference path is the *all-one* path. This is the path generated by the all-one source symbols. The corresponding encoder output sequence also consists of all ones.

The first new concept needed is that of a *detour*. Detours are those segments of the path selected by the Viterbi decoder that do not correspond to the reference path.[3] A detour starts at a node where the decoded path diverges from the reference path and ends at some later node where the decoded path merges again with the reference path. (See the figure below.)



6.3.1 Counting Detours

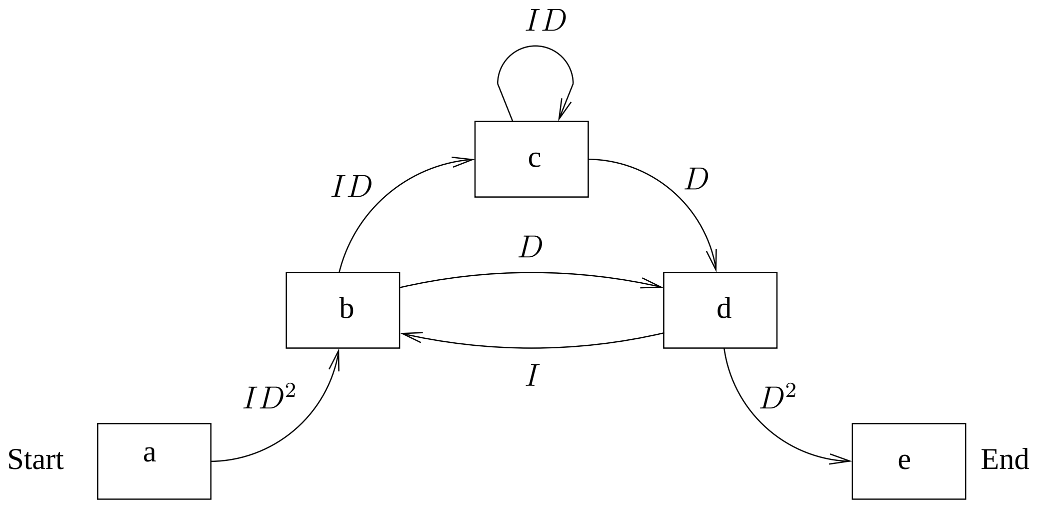
The basic idea to determine the bit error probability is simple. For each detour we determine the probability that the Viterbi decoder takes this detour and the number of information bit errors we make when this happens. These two quantities allow us, in principle, to determine the average bit-error probability. We will actually work with an upper bound to the probability that the Viterbi decoder takes a given detour. For a given detour, our upper bound will depend on the number d of discrepancies between the encoder output sequence corresponding to the detour and that of the reference path.

Example 48. *Regardless of the reference path, for our example there is a shortest detour starting at any state in the trellis (provided we are sufficiently away from the final state to avoid "edge effect"). This shortest detour spans 3 trellis sections. (A trellis section is the portion of the trellis between all states at one depth and all states at the next depth). The corresponding parameters are* $i = 1$ *and* $d = 5$. □

[3]For an analogy, think of the trellis as a road map, of the reference path as of an intended road for your journey, and the path selected by the Viterbi decoder as of the actual road that you take. Once on a while you are forced to take a detour with respect to the intended road.

To avoid edge effects, we will assume that the trellis is semi infinite, i.e., it extends to infinity to the right. For any given point (depth) on the reference path, how many detours are there with given parameters i and d? We now proceed to find this number, denoted by $a(i,d)$. It will be one of the main ingredient in determining our upper bound to the bit error probability.

There is a one-to-one correspondence between a detour with respect to the all-one path and a path between state a and e in the following *detour flow graph.*



The label I^iD^d, (i and d nonnegative integers), on a transition denotes that this transition increases the discrepancies between the two input sequences (reference path and detour) by i and between the two output sequences by d.

Now we show how to determine $a(i,d)$. We will actually determine the *generating function* $T(I,D)$ of $a(i,d)$ defined as

$$T(I,D) = \sum_{i,d} I^iD^d a(i,d)$$

You should think of I and D as “place holders” without any physical meaning. It is like describing the coefficients $a_0, a_1, \ldots, a_{n-1}$ by means of the polynomial $p(x) = a_0 + a_1x + \ldots + a_{n-1}x^{n-1}$. In our case, as we will see, having the generating function $T(I,D)$ is more convenient than having $a(i,d)$ for all i and all d. We determine $T(I,D)$ recursively as follows. We introduce auxiliary generating functions, one for each intermediate state of the detour flow graph, namely:

$$T_b(I,D) = \sum_{i,d} I^iD^d a_b(i,d) \tag{6.1}$$

$$T_c(I,D) = \sum_{i,d} I^iD^d a_c(i,d) \tag{6.2}$$

$$T_d(I,D) = \sum_{i,d} I^iD^d a_d(i,d) \tag{6.3}$$

As $a(i,d)$ is the number of paths in the detour flow graph that start at Start, end at End, and have parameters i and d, so $a_b(i,d)$ is the number of paths in the detour flow graph that start at Start, end at b, and have parameters i and d. Similar definitions apply for $a_c(i,d)$ and $a_d(i,d)$. From the detour flow graph we immediately see that the following relationships hold:

Writing T_c instead of $T_c(I,D)$ we have the following relationships:

$$\begin{aligned}T_b &= T_a\, ID^2 + T_d\, I\\ T_c &= T_b\, ID + T_c\, ID\\ T_d &= T_b\, D + T_c\, D\\ T &= T_d\, D^2\end{aligned}$$

The above system may be solved for T by pure formal manipulations. (Like solving a system of equations). The result is

$$T(I,D) = \frac{ID^5}{1-2ID}.$$

The above expression $T(I,D)$ is what we need. However, to show that one can indeed obtain $a(i,d)$ from $T(I,D)$, using the expansion $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots$ we write

$$T(I,D) = \frac{ID^5}{1-2ID} = ID^5(1 + 2ID + (2ID)^2 + (2ID)^3 + \cdots \tag{6.4}$$

$$= ID^5 + 2I^2D^6 + 2^2I^3D^7 + 2^3I^4D^8 + \cdots \tag{6.5}$$

This means that there is one path with parameters $d=5$, $i=1$, that there are two paths with $d=6$, $i=2$, etc. In general, for $i=1,2,\ldots$ we have

$$a(i,d) = \begin{cases} 2^{i-1}, & d = i+4\\ 0, & \text{otherwise.}\end{cases}$$

Next we show that $a(i,d)$ does not depend on the reference path, provided that the encoder is linear.

Let $\boldsymbol{a} \in \mathcal{D}^*$, be the reference encoder-input sequence. Let $f : \mathcal{D}^* \to \mathcal{D}^*$ be the encoder map. Hence $f(\boldsymbol{a})$ is the encoder output corresponding to input sequence $\boldsymbol{a}$.

For a sequence $\boldsymbol{a} \in \mathcal{D}^*$ and positive integers k, l, we define $\boldsymbol{a}_k^l = (a_k, a_{k+1}, \ldots, a_l)$ to be the subsequence that starts with index k and ends with index l, $k \le l$. The set of all such sequences is denoted by $\{\pm 1\}_k^l$.

We will be interested in knowing how many "-1"are contained in a given sequence $\boldsymbol{a} \in \{\pm 1\}_k^l$. Let $w(\boldsymbol{a})$ denote this number.

$$w(\boldsymbol{a}) = \mid \{j : a_j = -1\} \mid$$

Let $\mathcal{A}_{j,i,d}$ be the set of detours that leave the all-one sequence at depth j $(j = 0, 1, \ldots)$ and has parameters i and d, i.e.

$$\mathcal{A}_{i,j,d} = \left\{ \boldsymbol{c} \in \{\pm 1\}_1^\infty : w(\boldsymbol{c}_1^j) = 0, c_{j+1} = -1, w(\boldsymbol{c}) = i, w(f(\boldsymbol{c})) = d \right\}$$

Then $a(i,d) = \mid \mathcal{A}_{j,i,d} \mid$, where the right hand side does not depend on the depth j since the encoder is time-invariant in the following sense. If input $\boldsymbol{a}$ produces a detour to the all-one path that starts at depth j and has parameters i and d, then $1\boldsymbol{a}$ produces a detour that starts at depth $j+1$ and has parameters i and d.

Similarly define $\mathcal{A}_{i,j,d}(\boldsymbol{a})$ as the set of detours with respect to the reference path determined by the input sequence $\boldsymbol{a}$:

$$\mathcal{A}_{j,i,d}(\boldsymbol{a}) = \left\{ \boldsymbol{e} \in \mathcal{D}^* : w((\boldsymbol{ea})_1^j) = 0,\ (\boldsymbol{ea})_{j+1} = -1,\ w(\boldsymbol{ea}) = i,\ w(f(\boldsymbol{e})f(\boldsymbol{a})) = d \right\}.$$

Let $a_{j,i,d}(\boldsymbol{a}) \overset{\triangle}{=} \mid \mathcal{A}_{j,i,d}(\boldsymbol{a}) \mid$ be the number of such detours. We want to show that for all $\boldsymbol{a} \in \mathcal{D}^*$, $a_{j,i,d}(\boldsymbol{a}) = a(i,d)$. It suffices to show that there exists a one-to-one correspondence between the elements of $\mathcal{A}_{j,i,d}(\boldsymbol{a})$ and those of $\mathcal{A}_{j,i,d}$.

We claim that the mapping

$$g : \mathcal{A}_{j,i,d}(\boldsymbol{a}) \to \mathcal{A}_{j,i,d}$$

that sends $\boldsymbol{e} \in \mathcal{A}_{j,i,d}(\boldsymbol{a})$ to $\boldsymbol{ea}$ is such a correspondence.

If we let $\boldsymbol{c} = \boldsymbol{ea}$ and use the definition of $\mathcal{A}_{j,i,d}(\boldsymbol{a})$ and the linearity of f which implies $f(\boldsymbol{e})f(\boldsymbol{a}) = f(\boldsymbol{ea}) = f(\boldsymbol{c})$, we see immediately that $\boldsymbol{c} \in \mathcal{A}_{j,i,d}$. Now let $\boldsymbol{c} \in \mathcal{A}_{j,i,d}$. The inverse mapping g^{-1} maps $\boldsymbol{c}$ to $\boldsymbol{e} = \boldsymbol{ca}$. Using the definition of $\mathcal{A}_{j,i,d}$, the fact that $\boldsymbol{ea} = \boldsymbol{caa} = \boldsymbol{c}$ and the linearity of f which implies $f(\boldsymbol{e})f(\boldsymbol{a}) = f(\boldsymbol{ea}) = f(\boldsymbol{c})$ we immediately verify that $\boldsymbol{c} \in \mathcal{A}_{j,i,d}(\boldsymbol{a})$. This completes the proof.

6.4 Upper Bound to P_b

We are now ready for the final step, namely the derivation of a tight upper bound to the bit-error probability.

Fix an arbitrary encoder input sequence, let $\boldsymbol{x} = x_1, x_2 \ldots, x_n$ be the corresponding encoder output sequence and $\boldsymbol{s} = \sqrt{E_s}\boldsymbol{x}$ be the corresponding point in signal space. The transmitted signal is

$$s(t) = \sum s_i \psi(t - iT).$$

We transmit this signal over an AWGN channel with power spectral density $N_0/2$. Let $r(t) = s(t) + z(t)$ be the received signal (where $z(t)$ is a sample path of the noise process $Z(t)$) and let

$$\boldsymbol{y} = (y_1, \ldots, y_n)^T, \quad y_i = \langle r, \psi_i \rangle$$

be a sufficient statistic.

The Viterbi algorithm labels each branch in the trellis with the corresponding branch metric and finds the path through the trellis with the largest path metric. A branch from depth j to $j+1$ corresponding to output symbols x_{2j-1}, x_{2j} is assigned the branch metric $y_{2j-1}x_{2j-1} + y_{2j}x_{2j}$.

Consider the sample path through the trellis selected by the Viterbi algorithm.

Let $w_j, j = 0, 1, \ldots, k-1$, be the number of errors made on a detour that *begins* at depth j. If at depth j the VD is on the correct path or if it follows a detour started earlier then $w_j = 0$. Let W_j be the corresponding random variable (over all possible noise realizations).

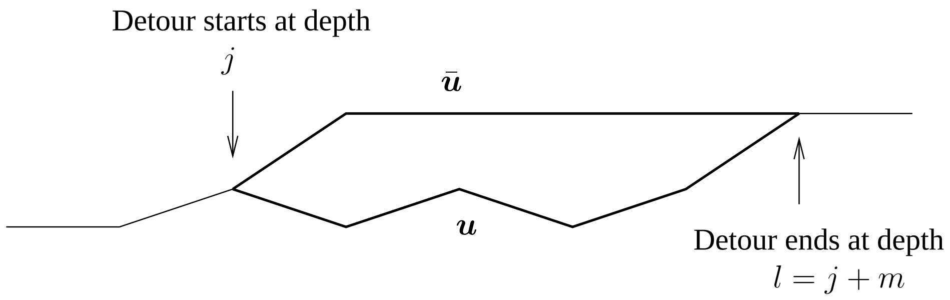
The total of source symbol errors for the path selected by the VD is

$$\sum_{j=0}^{k-1} w_j$$

and $\frac{1}{k}\sum_{j=0}^{k-1} w_j$ is the fraction errors with respect to the k source symbols. Hence we define the bit-error probability

$$P_b \triangleq E\frac{1}{k}\left[\sum_{j=0}^{k-1} W_j\right] = \frac{1}{k}\sum EW_j$$

Let us now focus on a detour. If it starts at depth j and ends at depth $l = j + m$, then the corresponding encoder-output symbols are some $2m$ tuple $\bar{\boldsymbol{u}} \in \{\pm 1\}^{2m}$. Let $\boldsymbol{u} = (x_{2j}, \ldots, x_{2l-1})^T \in \{\pm 1\}^{2m}$ be the corresponding sequence of the correct path and let $\boldsymbol{\rho}$ be the portion of $\boldsymbol{y}$ corresponding to the same interval, i.e., $\boldsymbol{\rho} = (y_{2j}, \ldots, y_{2l-1})^T$.



Let d be the number of positions in which $\boldsymbol{u}$ and $\bar{\boldsymbol{u}}$ differ (also called Hamming distance $d(\boldsymbol{u}, \bar{\boldsymbol{u}})$ between $\boldsymbol{u}$ and $\bar{\boldsymbol{u}}$).

Notice that the Euclidean distance between the corresponding waveforms is the distance between $\sqrt{E_s}\boldsymbol{u}$ and $\sqrt{E_s}\bar{\boldsymbol{u}}$ which is $d_E = 2\sqrt{E_s d}$.

A necessary (but not sufficient) condition for the Viterbi decoder to take the detour under consideration is that

$$\langle \boldsymbol{\rho}, \sqrt{E_s}\boldsymbol{u} \rangle \leq \langle \boldsymbol{\rho}, \sqrt{E_s}\bar{\boldsymbol{u}} \rangle.$$

This condition is satisfied iff

$$\| \boldsymbol{\rho} - \sqrt{E_s}\boldsymbol{u} \|^2 \geq \| \boldsymbol{\rho} - \sqrt{E_s}\bar{\boldsymbol{u}} \|^2 .$$

If $\boldsymbol{u}$ is the correct sequence and $\bar{\boldsymbol{u}}$ is the only alternative, then the above event happens with probability

$$Q\left(\frac{d_E}{2}\frac{1}{\sigma}\right)$$

where $\sigma^2 = \frac{N_0}{2}$ and $d_E = 2\sqrt{E_s d}$.

Recall (the Bhattacharyya bound for the Gaussian channel) that

$$Q\left(\frac{d_E}{2\sigma}\right) = Q\left(\sqrt{\frac{2E_s d}{N_0}}\right) \leq e^{-\frac{E_s d}{N_0}} = z^d,$$

where we have defined $z = e^{-\frac{E_s}{N_0}}$. Using the union of events bound

$$Pr\{W_j = i\} \leq \sum_d a(i,d)z^d$$

and

$$EW_j = \sum_i iPr\{W_j = i\} \leq \sum_{i,d} ia(i,d)z^d,$$

we obtain

$$P_b = \frac{1}{k}\sum_{j=0}^{k-1} EW_j \leq \frac{1}{k}\sum_{j=0}^{k-1}\sum_{i,d} a(i,d)z^d i = \sum_{i,d} a(i,d)z^d i$$

Recall that $T(I,D) = \sum_{i,d} I^i D^d a(i,d)$. Thus

$$P_b \leq \left.\frac{\partial}{\partial I}T(D,I)\right|_{I=1,\, D=z}$$

$$P_b \leq \left.\frac{\partial}{\partial I}T(D,I)\right|_{I=1,\, D=z}$$

This result can be generalized to an encoder that, in each trellis section, takes k_0 information symbols and produces n_0 channel symbols, $n_0 \geq k_0$ (in our example $k_0 = 1$ and $n_0 = 2$), and for any memoryless channel (not just the AWGN).

In our particular example

$$T(D,I) = \frac{ID^5}{1-2ID}$$
$$\frac{\partial T}{\partial I} = \frac{D^5}{(1-2ID)^2}.$$

Thus

$$P_b \leq \frac{z^5}{(1-2z)^2},$$

where $z = e^{-\frac{E_b}{2N_0}}$ and $E_s = \frac{E_b}{2}$ (we are transmitting two channel symbols per information bit).

6.5 Concluding Remarks

What have we done and how does it compare to what we have done before?

It is convenient to think of the bit-by-bit as our starting point.

$$s(t) = \sum_{i=0}^{n-1} s_i \psi(t - iT)$$

$$s_i \in \{\pm\sqrt{E_s}\}.$$

The relevant design choices for this system are:

$$\begin{aligned} R_b &= R_s = \frac{1}{T} \quad \text{bit rate} \\ E_b &= E_s \quad \text{energy per bit} \\ P_b &= Q\left(\frac{\sqrt{E_s}}{\sigma}\right), \quad \text{bit-error probability.} \end{aligned}$$

Using $\sigma = \sqrt{\frac{N_0}{2}}$ and the upper bound $Q(x) \le e^{-\frac{x^2}{2}}$ we obtain

$$P_b \le e^{-\frac{E_b}{N_0}}.$$

The novelty of this chapter was to have an encoder in front of the bit-by-bit on a pulse train. The encoder trades P_b for R_b. The new parameters are:

$$\begin{aligned} R_b &= \frac{R_s}{2} = \frac{1}{2T_s} \\ E_b &= \frac{E_s}{r} = 2E_s \\ P_b &\le \frac{z^5}{(1-2z)^2} \text{ where } z = e^{-\frac{E_b}{2N_0}}. \end{aligned}$$

where r is the dimensionless rate of the encoder in $\frac{\text{bits}}{\text{symbol}}$ ($r = \frac{1}{2}$ in our example). It should be emphasized that source symbols are called *bits* since they carry one bit of information each. The notion of bits as a unit of information will be introduced to you in the Information Theory class (7th semester). For now it suffices to say that a binary random variable carries one bit of information iff it is uniformly distributed, and each symbol of a sequence of binary random variables carries one bit of information if, in addition, the symbols are i.i.d.

For a fixed encoder output rate, the power spectral density of the transmitted signal is the same as that of an uncoded system with the same symbol rate (see Homework). However, the encoder output has a symbol rate which is twice that of the encoder input rate. Hence, in effect, we are occupying twice as much bandwidth. We have reduced the bit-error probability but we have increased the bandwidth by a factor two. With more powerful codes we can further decrease the bit error probability without further expanding the bandwidth. As already mentioned in the previous chapter, there is a fundamental limit, studied in information theory (seen next semester) that says that we can make the error probability arbitrarily small as long as the bit rate is smaller than a computable number called *channel capacity*. For the AWGN channel with bandwidth B, the channel capacity is $B\log_2\left(1+\frac{P}{BN_0}\right)$ [bits/sec], where P is the transmitted power. The probability of error may be made arbitrarily small by a clever choice of signaling method. In fact, across a bandlimited channel, one may always transmit signals of the form

$$s_i(t) = \sum_j s_{ij}\psi(t-jT)$$

where $\psi(t) = \frac{1}{\sqrt{T}}\text{sinc}(\frac{t}{T})$. This follows from the sampling theorem. In general, however, the s_{ij} is not constrained to be of the form $\{\pm\sqrt{E_s}\}$. Like in the example of this chapter, the signal space points used to make the probability of error arbitrary small are obtained from an encoder.

As already mentioned, the encoder is just meant to form appropriate signal points in n dimensions, where n is typically large. Intuitively speaking, here is what a large n buys us. Let us start first from the opposite situation and let us assume that we are operating in one dimension like in pulse amplitude modulation (PAM). Independently of the transmitted point s, the received point $y = s + Z$, can be anywhere in $\mathbb{R}$ (certain regions are more likely than other and if we choose the decoding region according to the MAP rule then we minimize the error probability, but there is a limit to how small we can make the probability of error). If we move to n dimensions, then we send some $\boldsymbol{s} \in \mathbb{R}^n$ and receive $\boldsymbol{y} = \boldsymbol{s} + \boldsymbol{Z}$, where $\boldsymbol{Z} \sim \mathcal{N}(0, I_n\sigma^2)$. By the law of large numbers, $\sqrt{\frac{1}{n}\sum z_i^2}$ goes to σ as n goes to infinity. This means that with probability going to 1, $\boldsymbol{Y}$ will be in a thin shell of radios $\sqrt{n}\sigma$ around $\boldsymbol{s}$. Intuitively, the key here is that in n dimensional space there are points in $\mathbb{R}^n$ that are "off limit" for $\boldsymbol{Y} = \boldsymbol{s}+\boldsymbol{Z}$. This was not the case for the corresponding one-dimensional problem. By choosing the signal points cleverly, there is a hope that we can find a large number of such points that can be distinguished from one another with hight probability even after they have been sent through the channel.

6.6 Problems

Problem 1. (Power Spectral Density of the convolutional code that was considered in class.)

Block-orthogonal signaling may be the simplest coding method that achieves $Pr\{e\} \to 0$ as $N \to \infty$ for a non-zero data rate. However, we have seen in class that the price to pay is that block-orthogonal signaling requires infinite bandwidth to make $Pr\{e\} \to 0$.

This may be a small problem for one space explorer communicating to another; however, for terrestrial applications, there are always constraints on the bandwidth consumption. Therefore, in the examination of any coding method, an important issue is to compute its bandwidth consumption. Compute the bandwidth occupied by the rate$-1/2$ *convolutional code studied in this chapter. The signal that is put onto the channel is given by*

$$X(t) = \sum_{i=-\infty}^{\infty} X_i \sqrt{E_s}\psi(t - iT_s), \tag{6.6}$$

where $\psi(t)$ *is some unit-energy function of duration* T_s *and we assume that the trellis extends to infinity on both ends, but as usual we actually assume that the signal is the wide-sense stationary signal*

$$\tilde{X}(t) = \sum_{i=-\infty}^{\infty} X_i \sqrt{E_s}\psi(t - iT_s - T_0), \tag{6.7}$$

where T_0 *is a random delay which is uniformly distributed over the interval* $[0, T_s)$.

(i) *Find the expectation* $E[X_i X_j]$ *for* $i = j$, *for* $(i,j) = (2n, 2n+1)$ *and for* $(i,j) = (2n, 2n+2)$ *for the convolutional code that was studied in class. Then give the autocorrelation function* $R_X[i-j] = E[X_i X_j]$ *for all* i *and* j. Hint: *Consider the infinite trellis of the code. Recall that the convolution code studied in the class can be defined as*

$$\begin{aligned} X_{2n} &= D_n D_{n-2} \\ X_{2n+1} &= D_n D_{n-1} D_{n-2} \end{aligned}$$

(ii) *Find the autocorrelation function of the signal* $\tilde{X}(t)$, *that is*

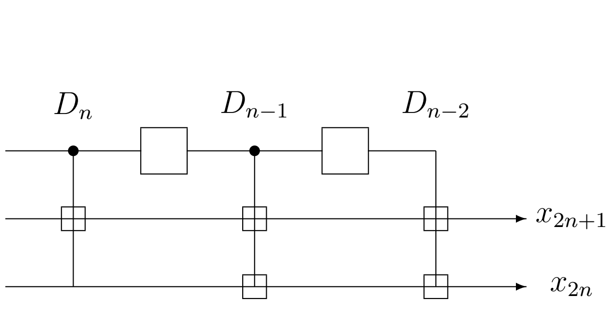
$$R_{\tilde{X}}(\tau) = E[\tilde{X}(t)\tilde{X}(t+\tau)] \tag{6.8}$$

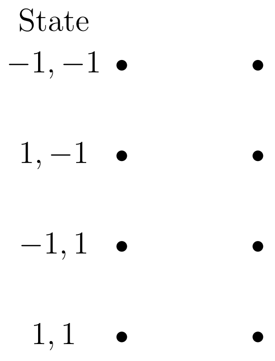
in terms of $R_X[k]$ *and* $R_\psi(\tau) = \frac{1}{T_s}\int_0^{T_s} \psi(t+\tau)\psi(t)dt$.

(iii) *Give the expression of power spectral density of the signal* $\tilde{X}(t)$.

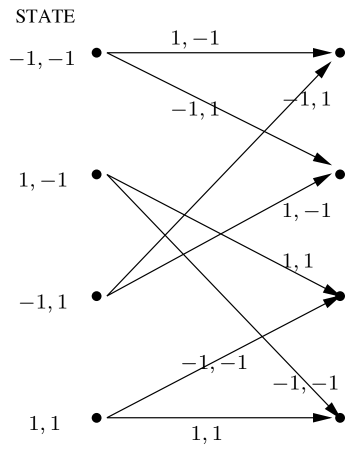
(iv) *Find and plot the power spectral density that results when* $\psi(t)$ *is a rectangular pulse of width* T_s *centered at* 0.

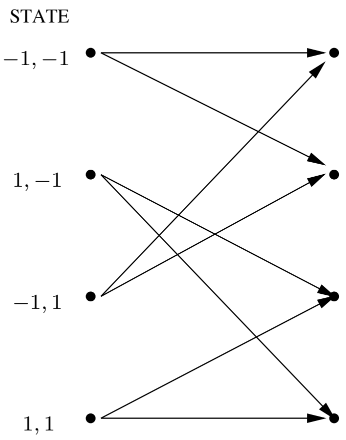
PROBLEM 2. *For the convolutional encoder shown below on the left, fill in the section of the trellis shown below on the right, that is, find the correct arrows and label them with the corresponding output value pairs* (x_{2n}, x_{2n+1}). *The input sequence* d_n *takes values in* $\{\pm 1\}$.



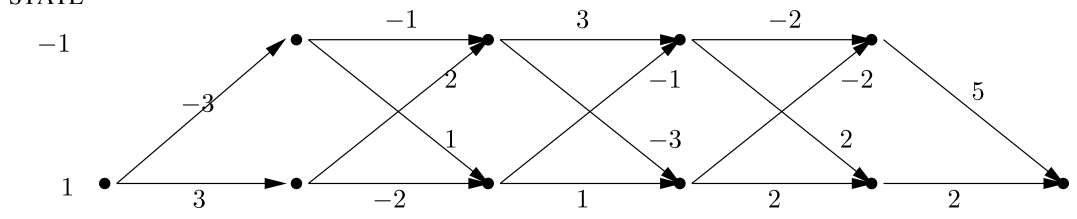


PROBLEM 3. *Consider the convolutional code described by the trellis section below on the left. You may assume that each of the encoder output symbols* (x_{2n}, x_{2n+1}), *are mapped into orthogonal waveforms,* $\phi_1(t)$ *if* $x_i = +1$ *and* $\phi_2(t)$ *if* $x_i = -1$. *The waveforms are of equal energy* E_s. *At the receiver we perform matched filtering with the filters matched to* $\phi_1(t)$ *and* $\phi_2(t)$. *Suppose the output of the matched filter at time* n *are* $(y_{2n}, y_{2n+1}) = (1,-2)$. *Find the branch metric values to be used by the Viterbi algorithm and enter them into the trellis section on the right.*





PROBLEM 4. *In the trellis below, the received sequence has already been preprocessed. The labels on the edges of the trellis are the branch metric values. Find the maximum likelihood path.*



Problem 5. (Intersymbol Interference)

An information sequence $\underline{U} = (U_1, U_2, \ldots, U_5)$, $U_i \in \{0,1\}$ *is transmitted over a noisy intersymbol interference channel. The* i*th channel output is*

$$lsY_i = S_i + Z_i,$$

where the noise Z_i *forms an independent and identically distributed (i.i.d.) sequence of Gaussian random variables,*

$$S_i = \sum_{j=0}^{\infty} U_{i-j} h_j, \qquad i = 1, 2, \ldots$$

and the channel impulse response is given by

$$h_i = \begin{cases} 1, & i = 0 \\ -2, & i = 1 \\ 0, & \text{otherwise.} \end{cases}$$

You may assume that $U_i = 0$ *for* $i \geq 6$ *and* $i \leq 0$.

(a) *Rewrite* S_i *in a form that explicitly shows which information symbols are relevant for the* i*th output.*

(b) *Sketch a trellis representation of a finite state machine that produces the output sequence* $\underline{S} = (S_1, S_2, \ldots, S_6)$ *from the input sequence* $\underline{U} = (U_1, U_2, \ldots, U_5)$. *Label each trellis transition with the specific value of* $U_i | S_i$.

(c) *Specify a metric* $f(\underline{s}, \underline{y}) = \sum_{i=1}^{6} f(s_i, y_i)$ *whose minimization or maximization with respect to* $\underline{s}$ *leads to a maximum likelihood decision on* $\underline{S}$. *Specify if your metric needs to be minimized or maximized.* Hint: *Think of a vector channel* $\underline{Y} = \underline{S} + \underline{Z}$, *where* $\underline{Z} = (Z_1, \ldots, Z_6)$ *is a sequence of i.i.d. components with* $Z_i \sim \mathcal{N}(0, \sigma^2)$.

(d) *Assume* $\underline{Y} = (Y_1, Y_2, \cdots, Y_5, Y_6) = (2, 0, -1, 1, 0, -1)$. *Find the maximum likelihood estimate of the information sequence* $\underline{U}$. Please: *Do* not *write into the trellis that you have drawn in Part* (b)*; work on a copy of that trellis.*

Problem 6. (Linear Transformations.)

(i)(a) *First review the notion of a field. (See e.g. K. Hoffman and R. Kunze,* Linear Algebra, *Prentice Hall or your favorite linear algebra book.)*

Now consider the set $\mathcal{F} = \{0, 1\}$ *with the following addition and multiplication tables:*

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

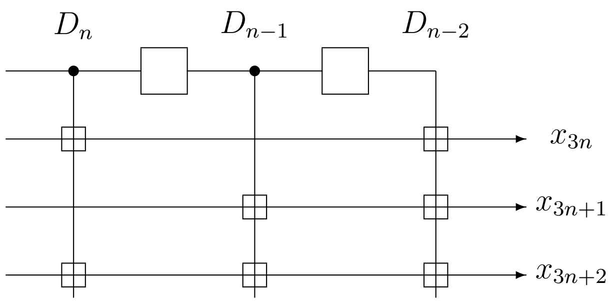


Figure 6.3: Convolutional encoder. $x_{3n} = d_n d_{n-2}$; $x_{3n+1} = d_{n-1} d_{n-2}$; $x_{3n+2} = d_n d_{n-1} d_{n-2}$.

Does $\mathcal{F}$, "$+$", and "$\times$" form a field?

(i)(b) *Repeat using $\mathcal{F} = \{\pm 1\}$ and the following addition and multiplication tables:*

$+$	1	-1
1	1	-1
-1	-1	1

$\times$	1	-1
1	1	1
-1	1	-1

(ii)(a) *Now first review the notion of a vector space.*

Let $\mathcal{F}$, $+$ and $\times$ be as defined in (i)(a). *Let $\mathcal{V} = \mathcal{F}^\infty$. (The latter is the set of infinite sequences with components in $\mathcal{F}$. Does $\mathcal{V}$, $\mathcal{F}$, $+$ and $\times$ form a vector space?*

(ii)(b) *Repeat using $\mathcal{F}$, $+$ and $\times$ as in* (i)(b).

(iii)(a) *Review the concept of linear transformation from a vector space $\mathcal{I}$ to a vector space $\mathcal{O}$. Now let $f : \mathcal{I} \to \mathcal{O}$ be the mapping implemented by the encoder described in this chapter. Specifically, let $\boldsymbol{x} = f(\boldsymbol{d})$ be specified by*

$$\begin{aligned} x_{2j-1} &= d_{j-1} \oplus d_{j-2} \oplus d_{j-3} \\ x_{2j} &= d_j \oplus d_{j-2}. \end{aligned}$$

Is this encoder linear?

Problem 7. (Rate$-1/3$ Convolutional Code.)

Consider the following convolutional code, to be used for the transmission of a data sequence $d_i \in \{-1, 1\}$:

(i) *Draw the state diagram for this encoder.*

(ii) *Suppose that this code is decoded using the Viterbi algorithm. Draw the detour flowgraph.*

(iii) *This encoder/decoder is used on an AWGN channel. The energy available per source digit is E_b and the power spectral density of the noise is $N_0/2$. Give an upper bound on the bit error probability P_b as a function of E_b/N_0.*

PROBLEM 8. (Convolutional Code.)

The following equations define a convolutional code for a data sequence $d_i \in \{-1, 1\}$:

$$x_{3n} = d_{2n} \cdot d_{2n-1} \cdot d_{2n-2} \tag{6.9}$$

$$x_{3n+1} = d_{2n+1} \cdot d_{2n-2} \tag{6.10}$$

$$x_{3n+2} = d_{2n+1} \cdot d_{2n} \cdot d_{2n-2} \tag{6.11}$$

(i) *Draw an implementation of the encoder of this convolutional code, using only delay elements D and multipliers.* Hint: *Split the data sequence d into two sequences, one containing only the even-indexed samples, the other containing only the odd-indexed samples.*

(ii) *What is the rate of this convolutional code?*

(iii) *Draw the state diagram for this convolutional encoder.*

(iv) *Does the formula for the upper bound on P_b that was derived in class still hold? If not, make the appropriate changes.*

(v) (optional) *Now suppose that the code is used on an AWGN channel. The energy available per source digit is E_b and the power spectral density of the noise is $N_0/2$. Give the detour flowgraph, and derive an upper bound on the bit error probability P_b as a function of E_b/N_0.*