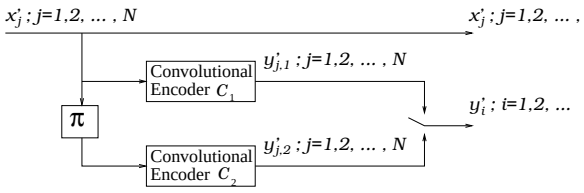
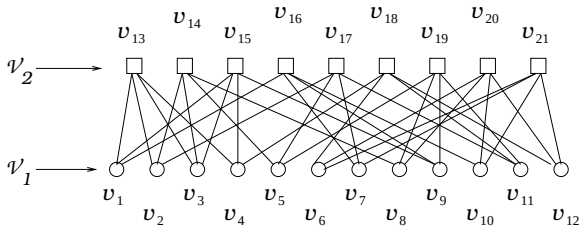
ECE8771
Information Theory & Coding for Digital Communications

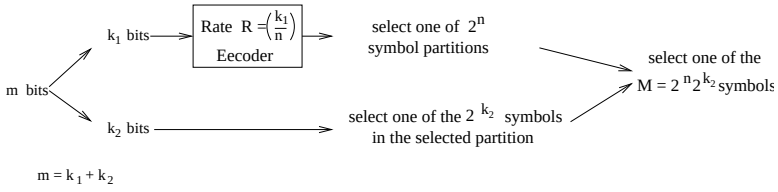
Villanova University
ECE Department

Prof. Kevin M. Buckley

Lecture Set 4
Turbo & LDPC Codes, Space-Time Coding, TCM







Contents

List of Figures

9 Turbo & Low Density Parity Check (LDPC) Codes

To this point in the Course we have established most of the concepts that we need to study modern channel coding methods. We have considered concatenation of constituent codes to effectively build larger codewords. He have discussed interleaving at the transmitter as an approach to facilitate deinterleaving at the receiver for the purpose of decorrelating (spreading out) burst errors. We mentioned iterative decoding, via toggling between the decoder of constituent codes, as an effective suboptimum approach to decoding concatenated codes. These channel coding concepts had been used successfully for years prior to the discovery of turbo codes. In 1993, Berrou, Glavieux and Thitimajshima [1] put these concepts together in a particular way to describe what is called *turbo coding*. They combined a Parallel Concatenated Convolutional Code (PCCC), with deep interleaving (e.g. 2^{16} bits), with an iterative decoder based on a *symbol-by-symbol MAP* procedure (referred to as the BCJR algorithm) to propose *turbo coding*. Turbo coding represents a natural and relatively minor extension of practices that had been established prior to its discovery. This extension, however, is significant because it improves code performance to within a fraction of channel capacity.

The BCJR algorithm is a *forward/backward* symbol-by-symbol MAP block code estimation algorithm named after the four authors, Bahl, Cocke, Jelinek and Raviv [2], who first established it in the open literature in 1974. It is another example of a relatively minor extension of then existing practices. As we will see, a principal attribute that makes it effective as a constituent code decoder for concatenated codes is that it provides *extrinsic or soft information*.

The popularity of turbo codes has lead to an intense interest in other large codes with performance approaching channel capacity. In the literature, the term turbo code has come to represent a range of coding algorithms that employ iterative decoding. One example is what are being termed *Product Turbo Codes (PTC)* [3], which are simply large product codes with iterative decoding (see Subsection 6.8.1 above). Another example is *Low Density Parity Check (LDPC)* codes, also know as Gallagher codes, which were proposed by Gallagher [4] in 1962. Given this expanded connotation of the term turbo code, it can be argued that LDPC codes are the original turbo codes.

We begin in Subsection 9.1 with a description of the BCJR algorithm and an alternative algorithm for iterative decoding, the SOft decision Viterbi Algorithm (SOVA). In Subsection 9.2 we then describe turbo coding. Subsection 9.3, we briefly describe the Product Turbo Code (PTC) approach, which have become a popular alternative to turbo codes. We then overview turbo equalization in Subsection 9.4. Finally, in Subsection 9.5 we overview LDPC codes.

9.1 Decoding Algorithms Which Generate Extrinsic Information

With the advent of the Viterbi algorithm, MLSE has over the last thirty years become a broadly applied approach for convolutional code decoding and for other discrete-parameter estimation problems. There are, however, other design criteria worth considering for the estimation of discrete symbols in digital communication systems with memory. In this Subsection we introduce one, *symbol-by-symbol Maximum A Posterior (MAP)*, which has

received considerable interest over the last decade.

Although *symbol-by-symbol MAP algorithms* provide performance which is only comparable to MLSE and at computational cost that is greater than that of the Viterbi algorithm, they provide a *soft* (continuous-amplitude) output which can be used in concatenated encoder systems as *extrinsic* information that can be passed between component decoders in an iterative decoding scheme. Concatenated encoders, with interleaving, in conjunction with iterative decoders can deliver performance which is significantly better than that of MLSE convolutional code systems, approaching Shannon's capacity limit.

In this Subsection we describe the *BCJR algorithm* for *forward/backward* symbol-by-symbol MAP estimation. BCJR is a block processing algorithm in that it is optimum for a fixed, finite-length block of data and symbols. It is termed a forward/backward algorithm since it sweeps through the data block in the forward and then backward directions to derive the optimum estimate. This forward/backward processing structure is common in optimum block signal processing algorithms for systems modeled with memory. The BCJR algorithm has found broad uses in Turbo decoders which operate on very large blocks of symbols. *Forward-only* symbol-by-symbol MAP algorithms have also been developed (see, for example, Proakis, 4-th ed., Section 5.1.5) for processing ongoing streams of data.

The optimum BCJR symbol-by-symbol MAP algorithm is more computationally intensive than the Viterbi MLSE algorithm. It does, however, provide useful soft extrinsic information. A alternative extrinsic information generating algorithm, *SOft Viterbi Algorithm (SOVA)*, has been developed by Hagenauer and Hoeher [5]. Compared to BCJR, SOVA provides a computationally more attractive option with slightly reduced performance. In this Subsection we also describe SOVA.

9.1.1 Symbol-by-Symbol MAP and the BCJR Algorithm

The Viterbi algorithm, as presented in Section 8, computes the MLSE. That is, it computes the most likely estimate of the information sequence $\mathbf{X} = \{x_j;\ j = 1, 2, \cdots, N\}$ given the data ($\mathbf{R} = \{\mathbf{r}_j;\ j = 1, 2, \cdots, N\}$ for soft decision decoding). The problem statement is

$$\max_{\mathbf{X}} \qquad p(\mathbf{R}/\mathbf{X}) \quad . \tag{1}$$

The solution is also the maximum a posterior (MAP) sequence estimate if the sequences are all equally likely. If the sequences are not all equally likely, the MAP sequence estimator is the solution to

$$\max_{\mathbf{X}} \qquad p(\mathbf{R}/\mathbf{X})\ P(\mathbf{X}) \quad . \tag{2}$$

We now consider an alternative but closely related criterion, symbol-by-symbol MAP, which results in the BCJR algorithm. Our description of the BCJR algorithm follows that in the Bahl, Cocke, Jelinek and Raviv paper, [2]. An alternative description is presented by Lin and Costello [6] which explicitly shows how prior distributions on the information symbols can be incorporated. This is important because it is the key to understanding how the BCJR algorithm and SOVA are used for iterative decoding. The Proakis & Salehi development (in Section 8.8 of the Course Text) follows that in [6]. Below, we will point out where in the BCJR algorithm priors on the information bits are used.

Consider a rate $R_C = \frac{1}{n}$, constraint length K binary convolutional encoder[1] which codes a block of information bits x_j ; $j = 1, 2, \cdots, N$. Let $x_j^{(l)}$; $l = 0, 1$ denote the possible bit values of any x_j. For each stage, the encoder generates an N length block of n-dimensional codewords $\mathbf{C}_j$; $j = 1, 2, \cdots, N$. Assume coherent BPSK transmission[2] At the receiver, the sampled matched filter output is the block of n-dimensional vectors $\mathbf{r}_j$; $j = 1, 2, \cdots, N$, where $\mathbf{r}_j = (2 * \mathbf{C}_j - 1) + \mathbf{n}_j$ and $\mathbf{n}_j$ is zero-mean uncorrelated AWGN. Let $\underline{\mathbf{r}}_{1,N} = [\mathbf{r}_1,\ \mathbf{r}_1,\ \cdots, \mathbf{r}_N]$, where the subscripts on $\underline{\mathbf{r}}_{a,b}$ indicate the starting time a and stopping time b of the data block.

The *symbol-by-symbol MAP* problem statement is, for each and every information bit in the block x_j; $j = 1, 2, \cdots, N$,

$$\max_{x_j^{(l)}} \qquad P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N}) \quad , \tag{3}$$

where $P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N})$ is the *a posterior probability (APP)* of the x_j bit value, $x_j^{(l)}$, given all the data $\underline{\mathbf{r}}_{1,N}$. It is called "a posterior" because it is "after the data" is observed. The solution to this symbol-by-symbol MAP problem provides the *minimum BER information bit estimates*.

Because of the memory in the convolutional encoder, the x_j's are a statistically dependent of all the $\mathbf{r}_j$; $j = 1, 2, \cdots, N$, and all the data must be processed for each x_j. Thus, to solve this optimization problem, for each x_j, we need the $P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N})$; $l = 0, 1$. To calculate the $P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N})$; $l = 0, 1$, consider the trellis representation illustrated below in Figure 88(a). Let $\mathbf{S}_j^{(m)}$; $j = 0,\ ,\ \cdots, N$; $m = 1,\ 2,\ \cdots, M$ represent all the states over all stages. Let $\mathcal{S}_j^{(0)}$ be the set of states at stage j that correspond to the zero symbol at stage j, $x_j^{(0)}$. That is, for the illustration in Figure 88(a), $\mathcal{S}_j^{(0)} = \{\mathbf{S}_j^{(1)},\ \mathbf{S}_j^{(2)}\}$. Similarly, let $\mathcal{S}_j^{(1)}$ be the set of states at stage j that correspond to the one symbol at stage j, $x_j^{(1)}$, i.e. in Figure 88(a) $\mathcal{S}_j^{(1)} = \{\mathbf{S}_j^{(3)},\ \mathbf{S}_j^{(4)}\}$. Consider the *a posterior probabilities (APP's) of the states* at stage j, $P(\mathbf{S}_j^{(m)}/\underline{\mathbf{r}}_{1,N})$; $m = 1,\ 2,\ \cdots,\ M$. By the total probability relationship,

$$P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N}) \ = \ \sum_{\mathcal{S}_j^{(l)}} P(\mathbf{S}_j^{(m)}/\underline{\mathbf{r}}_{1,N}) \ , \qquad l = 0,\ 1 \quad . \tag{4}$$

So, in terms of this trellis diagram representation, for each stage j we need the
$P(\mathbf{S}_j^{(m)}/\underline{\mathbf{r}}_{1,N})$; $m = 1,\ 2,\ \cdots,\ M$, the APP's of the states.
We now develop the BCJR algorithm. Assume that the initial state of the encoder is composed of all zeros, so that the trellis at stage $j = 0$ is constrained to be $\mathbf{S}_0^{(1)}$. Additionally, assume that the state at stage N is constrained to be $S_N^{(1)}$. This latter constraint is equivalent to $x_j = 0$; $j = N - (K - 2),\ \cdots, N$. That is, the trellis is terminated to the zero state. Figure 88(b) illustrates the trellis diagram under these constraints.

To use established notation related to the BCJR algorithm, let

$$\lambda_j^{(l,m)} \ = \ P(\mathbf{S}_j^{(m)}/\underline{\mathbf{r}}_{1,N}) \tag{5}$$

[1]We choose $k = 1$ here to simplify notation. The extension to $k > 1$ is conceptually straightforward.
[2]Modification for other modulation schemes is straightforward.

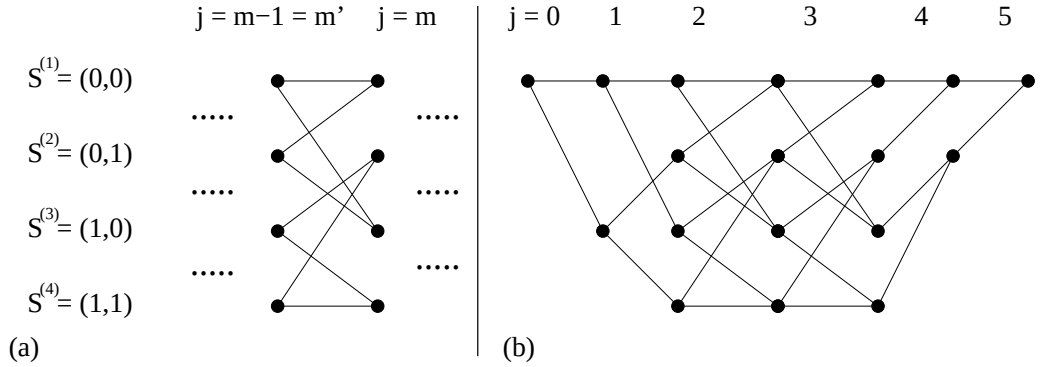


Figure 88: (a) successive trellis diagram stages for a rate $\frac{1}{2}$, $K = 3$ convolutional encoder with $\mathbf{g}_1 = (7)$ and $\mathbf{g}_2 = (5)$.; (b) illustration of trellis termination.

be the state m probability at stage j, conditioned on all the data $\underline{\mathbf{r}}_{1,N}$, for $\mathbf{S}_j^{(m)} \in \mathcal{S}_j^{(l)};\ l = 0\ or\ 1$ (e.g. $\lambda_j^{(0,m)}$ is defined only for $\mathbf{S}_j^{(m)} \in \mathcal{S}_j^{(0)}$). Let

$$\alpha_j^{(l,m)} \ = \ P(\mathbf{S}_j^{(m)}/\underline{\mathbf{r}}_{1,j}) = \ \frac{P(\mathbf{S}_j^{(m)}, \underline{\mathbf{r}}_{1,j})}{P(\underline{\mathbf{r}}_{1,j})} \tag{6}$$

be the *forward* state m probability at stage j, conditioned on the data up to stage j, $\underline{\mathbf{r}}_{1,j}$, for $\mathbf{S}_j^{(m)} \in \mathcal{S}_j^{(l)};\ l = 0\ or\ 1$. Let

$$\beta_j^m \ = \ \frac{P(\underline{\mathbf{r}}_{j+1,N}/\mathbf{S}_j^{(m)})}{P(\underline{\mathbf{r}}_{j+1,N}/\underline{\mathbf{r}}_{1,j})} \tag{7}$$

be a *backward* data probability ratio given $\mathbf{S}_j^{(m)}$. It has been shown, [2], that

$$\lambda_j^{(l,m)} \ = \ \alpha_j^{(l,m)}\ \beta_j^m \quad . \tag{8}$$

This suggests the following algorithm:

- *Forward pass*: For each $j = 1, 2, \cdots, N$, compute the forward state probabilities

$$\alpha_j^{(l,m)} \ = \ \frac{\sum_{m'=1}^{M} \sum_{i=0}^{1} \gamma^l(\mathbf{r}_j, m', m)\ \alpha_{j-1}^{(i,m')}}{\sum_{m=1}^{M} \sum_{l=0}^{1} \sum_{m'=1}^{M} \sum_{i=0}^{1} \gamma^l(\mathbf{r}_j, m', m)\ \alpha_{j-1}^{(i,m')}} \tag{9}$$

  where for each j, m' denotes the state index for stage $j-1$ and m denotes the index for stage j. The

$$\gamma^l(\mathbf{r}_j, m', m) \ = \ \begin{cases} \frac{1}{2} P(\mathbf{r}_j/\mathbf{S}_j^{(m)}, \mathbf{S}_{j-1}^{(m')}) & \mathbf{S}_{j-1}^{(m')}, \mathbf{S}_j^{(m)}\ feasible \\ 0 & otherwise \end{cases} \tag{10}$$

  are state transition probabilities[3]. The Eq (9) denominator normalizes $\alpha_j^{(l,m)}$ over l and m.

[3]These "gamma" terms are likelihoods (i.e. data PDFs, conditioned on the states and therefore codeword

- *Backward pass*: For each $j = N, N-1, \cdots, 1$, compute the backward data probabilities

$$\beta_j^{(m)} \;=\; \frac{\sum_{m'=1}^{M}\sum_{i=0}^{1}\gamma^l(\mathbf{r}_{j+1},m,m')\;\beta_{j+1}^{(m')}}{\sum_{m=1}^{M}\sum_{l=0}^{1}\sum_{m'=1}^{M}\sum_{i=0}^{1}\gamma^l(\mathbf{r}_{j+1},m',m)\;\alpha_j^{(i,m')}} \quad , \tag{12}$$

and the state probabilities

$$\lambda_j^{(l,m)} \;=\; \alpha_j^{(l,m)}\;\beta_j^m \quad . \tag{13}$$

Eqs(9 - 13) constitute the BCJR algorithm, where as stated earlier

$$\lambda_j^{(l,m)} = P(\mathbf{S}_j^{(m)}/\underline{\mathbf{r}}_{1,N}) \tag{14}$$

for $\mathbf{S}_j^{(m)} \in \mathcal{S}_j^{(l)}$. So the bit probabilities, required to determine the symbol-by-symbol MAP bit estimates, are

$$P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N}) \;=\; \sum_{\mathcal{S}_j^{(l)}} \lambda_j^{(l,m)} \;, \qquad l=0,\;1\;; j=1,2,\cdots N \quad . \tag{15}$$

Upon completion of the BCJR algorithm (after the backward pass), for each stage j, the information bit APP's $P(x_j^{(l)}/\underline{\mathbf{r}}_{1,N});\; l=0,1$ are compared, and the output information bit corresponds to the maximum. The information bit APP's provide additional information. Specifically, consider the *Log Likelihood Ratios (LLRs)*,

$$L(x_j) \;=\; \log_e\left(\frac{P(x_j^{(1)}/\underline{\mathbf{r}}_{1,N})}{P(x_j^{(0)}/\underline{\mathbf{r}}_{1,N})}\right) \;; \qquad j=1,2,\cdots,N \quad . \tag{16}$$

$L(x_j) >> 0$ indicates that the $x_j = x_j^1$ output is highly reliable. Likewise, $L(x_j) << 0$ indicates $x_j = x_j^0$ output is highly reliable. Conversely, $L(x_j)$ close to zero means that the output is not very reliable. $L(x_j)$ is often called the *reliability*.

As opposed to the Viterbi algorithm, which just provides "hard" bit information (i.e. as the elements of the estimated sequence), this symbol-by-symbol MAP algorithm provides both hard and "soft" A Posterior Probabilities (APP's) of the bits, as well as the LLRs. This soft information, referred to as *extrinsic information*, is vital for effective iterative decoding.

---

values, with data plugged in). For Gaussian noise, they are exponential functions of the data and the information bits represented by the two states being transitioned. Note that, since these state transition probabilities are conditioned on the states, they are not effected by prior distributions on the information bits the states represent. In an alternative derivation of the BCJR algorithm, presented in Lin and Costello [6] Section 12.6, the "alpha", "beta" and "gamma" functions are defined slightly differently, such that priors on the information bits are incorporated into the gammas. In particular, therein

$$\gamma^l(\mathbf{r}_j,m',m) \;\equiv\; P(\mathbf{S}_j^{(m)},\mathbf{r}_j/\mathbf{S}_{j-1}^{(m')}) \;=\; P(\mathbf{S}_j^{(m)}/\mathbf{S}_{j-1}^{(m')})\;P(\mathbf{r}_j/\mathbf{S}_j^{(m)},\mathbf{S}_{j-1}^{(m')}) \quad , \tag{11}$$

so that priors on the information bits are incorporated into $P(\mathbf{S}_j^{(m)}/\mathbf{S}_{j-1}^{(m')})$. The resulting algorithm is the same (i.e. BCJR), and has the same forward/backward structure.

9.1.2 A Comparison Between MLSE with Viterbi & Symbol-by-Symbol MAP with BCJR

In this Subsection we use Monte Carlo simulations to compare soft-decision convolutional code decoding using MLSE implemented with the Viterbi algorithm and symbol-by-symbol MAP implemented with the BCJR algorithm. We consider a rate $R_c = \frac{1}{2}$ constraint length $K = 5$ convolutional encoder with generators $\mathbf{g}_1 = (37)$ and $\mathbf{g}_2 = (21)$. This is the encoder first considered in Example 9.1. We assume BPSK modulation with coherent reception. Figure 89 shows the simulation results – BER plotted vs. SNR/bit. As noted earlier, performance for these two decoder criteria is similar.

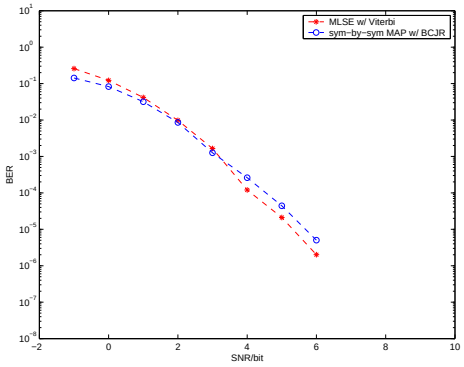


Figure 89: Monte Carlo simulation comparison of MLSE and symbol-by-symbol MAP.

9.1.3 The Soft Output Viterbi Algorithm (SOVA)

Motivated by the desire to generate soft bit estimates in a decoder (to be used as extrinsic information by another decoder) and by the sometimes impractical computational requirements of BCJR, a soft version of the Viterbi algorithm, called SOVA, has been developed [5]. A good description of this algorithm is presented in Lin and Costello [6] Section 12.5.

Say, for example, the modulation/demodulation scheme is coherent binary PSK. As mentioned in Subsection 8.5.2, as a practical matter, with the Viterbi algorithm we usually *truncate the trellis*. For SOVA, after pruning the paths at stage j and noting the costs of the survivor paths, we take the lowest cost path, trace it back δ stages, and declare the estimate of the information bit $x_{j-\delta}$ to be the bit indicated by the state at stage $j-\delta$ associated with this currently (stage j) best path. Call this estimated bit $\hat{x}_{j-\delta}$. In tracing back to stage $j-\delta$ through this best path at stage j, we cross $\delta+1$ states. At each of these states we have discarded a path. So along this trace we have discarded $\delta+1$ paths. Tracing all these discarded paths back to stage $j-\delta$, say that D of these correspond to the opposite bit of $\hat{x}_{j-\delta}$, which we denote $\hat{\overline{x}}_{j-\delta}$. For these D paths at stage $j-\delta$, let $\Delta_d; d = 1, 2, \cdots D$ be the differences between the cost of these paths at stage $j-\delta$ and the cost of the best path at stage j. Let $\Delta_{j,min} = \min\{\Delta_d; d = 1, 2, \cdots, D\}$ be the minimum of these cost differences. Then, as argued in [6] Section 12.5,

$$\hat{P}_{e,j-\delta} = \frac{e^{\Delta_{j,min}}}{1+e^{\Delta_{j,min}}} \tag{17}$$

is an approximation of the probability of error of bit estimate $\hat{x}_{j-\delta}$.

The established rule-of-thumb is that the trellis truncation depth δ is four to five times the memory depth $k(K-1)$ of the encoder. Hard bit estimates, $\hat{x}_j$, and corresponding probability of bit error estimates, $\hat{P}_{e,j-\delta}$, are outputted with delay (latency) δ. The extrinsic data passed on to another decoder in a concatenated code decoding system is the log likelihood ratio or reliability

$$L(x_{j-\delta}) \;=\; \ln\left(\frac{\hat{P}_{e,j-\delta}}{1-\hat{P}_{e,j-\delta}}\right) \;=\; \Delta_{j,min} \quad . \tag{18}$$

The SOVA algorithm outputs the MLSE (just as does the Viterbi algorithm) plus the log likelihood ratios $L(x_j);\ j = 1, 2, \cdots, N-K+1$ (remember that the trellis is terminated with known bits).

9.2 Turbo Codes

In this Subsection we consider turbo codes. At this point we take a fairly narrow perspective, covering the original concept and approach proposed by Berrou, Glavieux and Thitimajshima [1] in 1993, and minor variations of it. That is, we limit our discussion in this Subsection to Parallel Concatenated Convolutional Coding (PCCC) with interleaving and iterative decoding using soft outputs from constituent decoders. We begin with a description of the specific encoder and decoder described in [1]. We then discuss variations of these and consider performance and implementation issues.

9.2.1 PCCC with Interleaving and Iterative Decoding

Encoding

Consider the encoder illustrated in Figure 90. It consists of two equivalent constituent convolutional encoders, $\mathcal{C}_1$ and $\mathcal{C}_2$, along with a interleaver $\mathbf{\Pi}$. The encoder processes input information bits in blocks of size N, where N should be large to achieve the best possible performance. Thus, the encoder implements a block encoding scheme, where the block size is large. As shown in Figure 90, the output is systematic. Letting $\mathbf{X}' = [x_1',\ x_2',\ \cdots,\ x_N']$ represent the N-dimensional information vector, N elements of the output codeword $\mathbf{C}$ are the elements of $\mathbf{X}'$. These systematic bits are intertwined with the convolutional encoder outputs, as described below, to complete $\mathbf{C}$.

The two constituent convolutional codes proposed in [1] are both constraint length $K = 5$, rate $R_c = \frac{1}{2}$, with generators $\mathbf{g}_1 = (37)$ and $\mathbf{g}_2 = (21)$. The canonical nonrecursive encoder form of this code is shown in Figure 91(a). This was the convolution code considered earlier in the Course in Example 8.1. The systematic recursive version of this code is shown in Figure 91(b). The generator matrices for these two encoder forms are

$$\mathbf{G}(\mathcal{D}) \;=\; [\mathcal{D}^4+\mathcal{D}^3+\mathcal{D}^2+\mathcal{D}+1\ ,\ \ \mathcal{D}^4+1] \tag{19}$$

for the nonrecursive form, and

$$\mathbf{G}^1(\mathcal{D}) \;=\; \left[1\ ,\ \ \frac{\mathcal{D}^4+1}{\mathcal{D}^4+\mathcal{D}^3+\mathcal{D}^2+\mathcal{D}+1}\right] \tag{20}$$

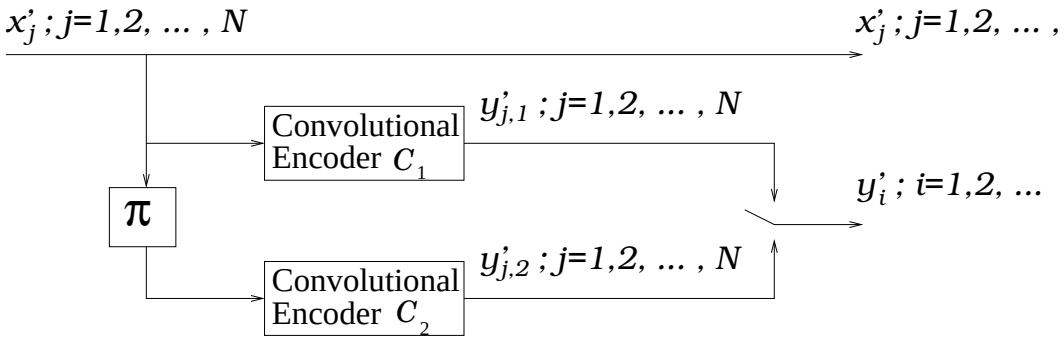


Figure 90: The turbo encoder proposed by Berrou, Glavieux and Thitimajshima.

for the systematic recursive form. The systematic recursive encoder is used as the constituent encoders for the turbo codes described in [1]. Since, as shown in Figure 90, the systematic bits are already provided (outside of the constituent convolutional encoders), the systematic outputs of the two convolutional encoders are not used.

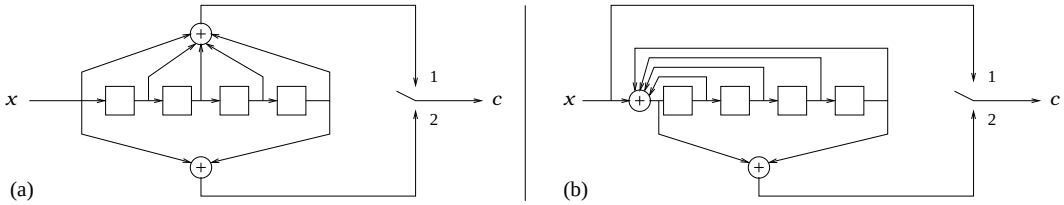


Figure 91: (a) the nonrecursive, and (b) the systematic recursive encoder used as the constituent encoders for the turbo code described by Berrou, Glavieux and Thitimajshima.

In [1], a block length of $N = 2^{16} = 65,536$ is used in simulations to illustrate performance, and a pseudo-random interleaver is suggested. The constituent encoder outputs are punctured and combined with the systematic output to form the codeword. As formulated in [1] and illustrated in Figure 90, the puncturing scheme is general. For example, a common approach used to form a rate $R_c = \frac{1}{2}$ turbo coder, is to take every other constituent encoder output, i.e.

$$y_j^{'} \;=\; \begin{cases} y_{1,j}^{'} & j = 1\ ,3\ ,5\ ,\ \cdots,\ N-1 \\ y_{2,j}^{'} & j = 2\ ,4\ ,6\ ,\ \cdots,\ N \end{cases} \quad . \tag{21}$$

For this case, the resulting codeword (usually termed a sequence since N is very large) is $\mathbf{C} = [c_1,\ c_2,\ c_3,\ \cdots,\ c_{2N}]$, where

$$c_i \;=\; \begin{cases} x_{(i+1)/2}^{'} & i = 1\ ,3\ ,5\ ,\ \cdots,\ 2N-1 \\ y_{i/2}^{'} & i = 2\ ,4\ ,6\ ,\ \cdots,\ 2N \end{cases} \quad . \tag{22}$$

For an AWGN channel with coherent BPSK transmission, the received sequence (after matched filtering and symbol rate sampling) is

$$r_i \;=\; (2 * c_i - 1) \;+\; n_i \quad , \tag{23}$$

where n_i is zero-mean AWGN with variance $\sigma_n^2 = N_0/2$. For the puncturing scheme described above, for the rate $R_c = \frac{1}{2}$ code, we denote these outputs as follows:

$$x_j \;=\; r_{(j+1)/2} \;;\; j = 1, 2, \cdots, N \qquad\qquad y_j \;=\; r_{j/2} \;;\; j = 1, 2, \cdots, N \quad . \tag{24}$$

Below, to describe the iterative decoder, we denote the received data as

$$\mathbf{r}_j \;=\; [x_j,\; y_{j,1},\; y_{j,2}] \;; \quad j = 1, 2, \cdots, N \quad . \tag{25}$$

If puncturing is implemented at the encoder, then some of the $y'_{k,l}s$ are zero.

Decoding

Here we provide a basic description of the iterative decoding approach used for turbo coding. The main computational algorithm is the soft decision decoding algorithm used for each of the two constituent decoders. For this, the BCJR algorithm described in Subsection 9.1, or some variation of it, or SOVA can be employed. Here we only describe the inputs and outputs to these constituent decoders, and the flow of data between them. We refer to the constituent decoders as BCJR algorithms. For detailed derivations and descriptions of iterative decoding algorithms for turbo decoding see Berrou, Glavieux and Thitimajshima [1], and Lin and Costelli [6] Chapters 12 and 16.

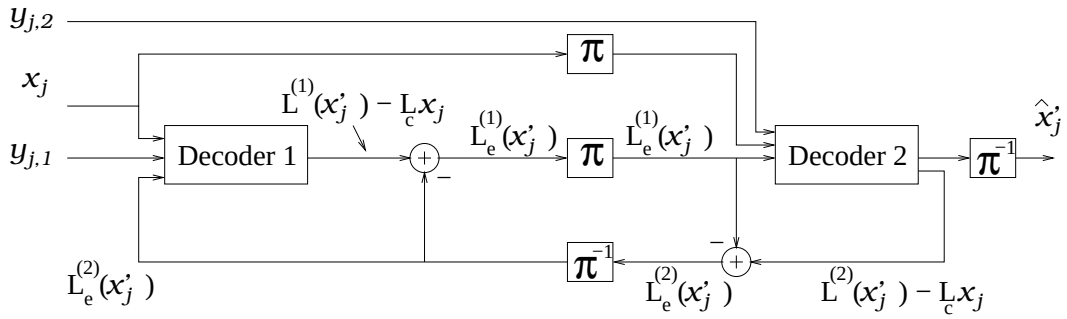


Figure 92: Turbo code iterative decoder.

The iterative decoder structure is illustrated in Figure 92. It consists of two constituent decoders iteratively processing data and feeding extrinsic information to one another through interleavers. Consider decoder 1. Its inputs are:

1. the x_j's;
2. the $y_{j,1}$'s; and
3. the extrinsic information values, the $L_e^{(2)}(x'_j)$'s, derived from the decoder 2 generated log likelihood ratios as illustrated in Figure 92. (These extrinsic information values are set to zero for the first iteration).

The $L_e^{(2)}(x'_j)$'s are employed as prior information by decoder 1. That is, the decoder 1 symbol-by-symbol MAP metric for each information symbol incorporates that symbol's APP's as prior symbol value probabilities, as noted above in our discussion in Subsection 9.1.1 on the

state transition probabilities $\gamma^l(\mathbf{r}_j, m', m)$ (i.e. the gamma's) used in the BCJR algorithm. It is this incorporation of extrinsic information as priors the results in improved performance iteration after iteration. The inputs $x'_j s$, $y'_{j,1} s$ are scaled by $L_c = \frac{4\mathcal{E}_s}{N_0}$ prior to processing by the BCJR algorithm so that their contribution to the log metric can be added directly to $L_e^{(2)}(x_j^{'})$. L_c, termed the *channel reliability factor*, needs to be provided or estimated. The decoder 1 outputs are:

1. the information bit log likelihood ratios, the $L^{(1)}(x_j^{'})$'s, generated by the decoder 1 BCJR algorithm, to be used as shown in Figure 92 to compute the extrinsic information for decoder 2; and

2. the data x_j, is passed along to compute the extrinsic information for decoder 2.

The decoder 1 outputs are then used to compute the extrinsic information for decoder 2 as follows:

$$L_e^{(1)}(x_j^{'}) \;=\; L^{(1)}(x_j^{'}) \;-\; L_c \cdot x_j \;-\; L_e^{(2)}(x_j^{'}) \;;\; j = 1, 2, \cdots, N \quad . \tag{26}$$

Subtracting $L_c \cdot x_j + L_e^{(2)}(x_j^{'})$ from $L^{(1)}(x_j^{'})$ to form the extrinsic information $L_e^{(1)}(x_j^{'})$ removes the effect of $x_j^{'}$, so as to provide decoder 2 a log likelihood ratio estimate which is independent of decoder 2.

The $L_e^{(1)}(x_j^{'})$'s and the x_j's are interleaved so as to be presented to decoder 2 in the same order as the $y_{j,2}$'s. Decoder 2 then operates just as decoder 1. The decoder 2 derived extrinsic information, calculated as

$$L_e^{(2)}(x_j^{'}) \;=\; L^{(2)}(x_j^{'}) \;-\; L_c \cdot x_j \;-\; L_e^{(1)}(x_j^{'}) \;;\; j = 1, 2, \cdots, N \quad , \tag{27}$$

is deinterleaved and passed to decoder 1 for the next iteration.

Upon completion, decoder 2 provides the final bit estimates $\hat{x}_j^{'};\; i = 1, 2, \cdots, N$ after deinterleaving.

Comments on Implementation, Performance and Variations

1. Berrou, Glavieux and Thitimajshima [1] reported 10^{-5} BER at SNR/bit only 0.7dB above channel capacity using the turbo coder described above. This level of performance has subsequently been well established.

2. Although turbo coding provides moderate BER levels (e.g. 10^{-5}) at SNRs near channel capacity, it is not very effective at providing very good BER levels (e.g. $< 10^{-6}$). This phenomenon, referred to as the *error floor* problem, has been attributed to the relatively small d_{min} value for a code with such large codewords.

3. Outer codes can be used in conjunction with inner turbo codes (i.e. in cascade) to provide very good BER levels (e.g. $< 10^{-6}$) at SNRs approaching channel capacity.

4. Increased N (e.g. $N = 2^{16}$) results in better performance. Large N results in significant latency, which limits the utility of turbo coding for some applications.

5. Pseudo-random interleavers result in better performance than block and other more structured interleavers, because they result in codewords which are less structured (i.e. more random) with generally higher weight distribution.

6. As constituent convolutional encoders, systematic recursive encoders outperform non-recursive encoders. Moderate constraint lengths (e.g. $K \leq 5$) encoders outperform larger K encoders.

7. Parity bit puncturing is used to reduce rate, to a certain extent without significantly sacrificing performance (e.g. $R_c = \frac{1}{2}$ works well with the turbo coder described in [1]).

8. Concerning iterative decoding, typically some improvement can be observed after each iteration up to about the 18^{th} iteration. However, after about 6 iterations performance is already very close to channel capacity. Stopping rules have been developed.

9. Besides SOVA, modifications to BCJR have been proposed which reduce computational requirements. For example, it has been suggested that a $\ln(e^x+e^y) = max(x,y)+\ln(1+e^{-|x-y|})$ calculation required for each metric used in the BCJR algorithm be replaced with $max(x,y)$. The resulting algorithm is termed Max-log-MAP. The regular BCJR algorithm, called log-MAP, performs better than SOVA or Max-log-MAP.

10. Alternative turbo coding structures have been considered. These include PCBCs with interleaving and multiple (> 2) constituent encoders.

9.3 Turbo Product Codes

Shortly after the introduction of their turbo coding scheme in 1993 by Berrou, Glavieux and Thitimajshima [1], Pyndiah, Glavieux, Picart and Jacq [3] suggested Turbo Product Codes (TPCs). The basic idea, to decode product codes by iterating between constituent decoders, had been around (see our earlier discussion in Subsection 7.8.1). The key contribution in [3] is the suggestion that BCJR soft decision algorithms be employed for the constituent decoders.

As noted in Subsection 7.8.1, given a $k_2 \times k_1$-dimensional information array, the minimum distance of a $(n_1 \cdot n_2, k_1 \cdot k_2)$ product code is $d_{min} = d_{1,min} \cdot d_{2,min}$, where $d_{1,min}$ is the minimum distance of the (n_1, k_1) block code $\mathcal{C}_1$ that operates on the array rows, and $d_{2,min}$ is the minimum distance of the (n_2, k_2) block code $\mathcal{C}_2$ that then operates on the array columns. The TPC rate is $R_c = \frac{k_1 \cdot k_2}{n_1 \cdot n_2} = R_{1,c} \cdot R_{2,c}$, i.e. the product of the constituent code rates. The d_{min} attribute of TPCs is attractive, but the rate attribute is a shortcoming. Nonetheless, for a reasonable size information array (e.g. $k_1 \cdot k_2 = 4096$), good BER can be achieved at SNRs within a dB of channel capacity.

9.4 Turbo Equalization

Channels often disperse a transmitted signal over time. Using systems terminology, we say that this is due to channel memory. For example, in a multipath channel scenario, a signal is received via more than a single path. Some paths are longer than others, so the receiver sees a superposition of versions of the transmitted delayed by different amounts. This is channel memory. When the channel memory is on the order of a symbol duration or greater, there is intersymbol interference (ISI) at the receiver. In the presence of ISI, detection of a symbol based on data received over only that symbol duration is not optimum, and often not effective.

Strictly speaking, a channel equalizer is a preprocessing filter designed to mitigate ISI in the received data stream prior to symbol detection, sequence estimation and/or decoding. However, the term *channel equalization* has come have a more general connotation. It implies any approach to dealing with ISI, including: equalizer filtering; and modeling ISI so as to directly perform symbol detection, sequence estimation and/or decoding. Being a broad and deep issue, channel equalization is the topic of a whole different course (for an overview of this topic, see Proakis & Salehi [7], Chapters 9 & 10). Here, as an illustration of the possible interaction between channel equalization and decoding, we introduce turbo equalization, which combines equalization and channel decoding in an iterative process.

To introduce the idea, consider two serially concatenated convolutional codes (SCCC's) illustrated in Figure 93(a). We call the first encoder the outer encoder and the second the inner encoder. (This is not an uncommon channel coding approach, see Subsections 7.8.3 & 8.7.) Both encoders have memory, and both compute outputs as linear combinations of inputs (is a GF field). Assume that the channel is memoryless with AWGN, and that the receiver front end matched-filters and samples to form a discrete-time sequence r_k. Figure 93(b) depicts this channel/receiver using a discrete-time model. Figure 93(c) shows a suboptimum but potentially effective decoding structure, consisting of an inner decoder followed by an outer decoder. Of course, we can iterate between these decoders to improve decoding

performance.

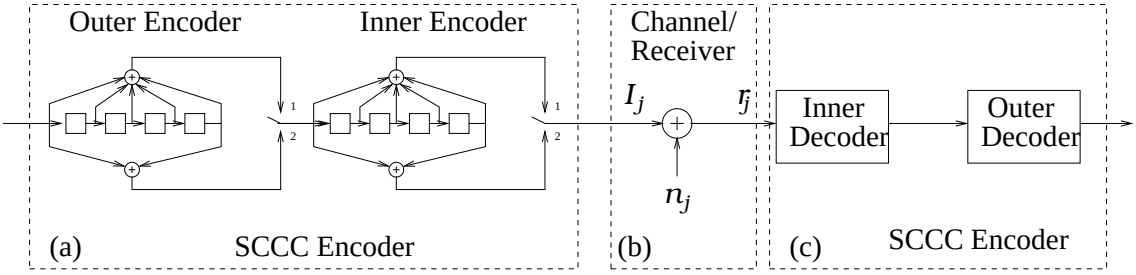


Figure 93: Serial concatenated convolutional code (SCCC): (a) encoder, (b) memoryless AWGN channel, (c) decoder.

Now consider that, instead of an inner encoder, we have an ISI channel. An equivalent discrete-time ISI model is shown in Figure 94(a) (see Proakis & Salehi, [7], Subsection 9.8.2). The discrete-time channel filter is usually FIR, corresponding the a finite memory channel such as those encountered in multipath cellular phone applications. In the respect most important for receiver processing, the inner encoder in Figure 93(a) and the ISI channel in Figure 94(a) similar. That is, they both have finite memory and combine delayed inputs as weighted sums. Because of this similarity, the inner decoder and channel equalization processor shown in Figure 94(b) function similarly. Thus an iterative decoder for the SCCC (i.e. a turbo) decoder, with relatively minor modification, can be considered as a *turbo equalizer*.

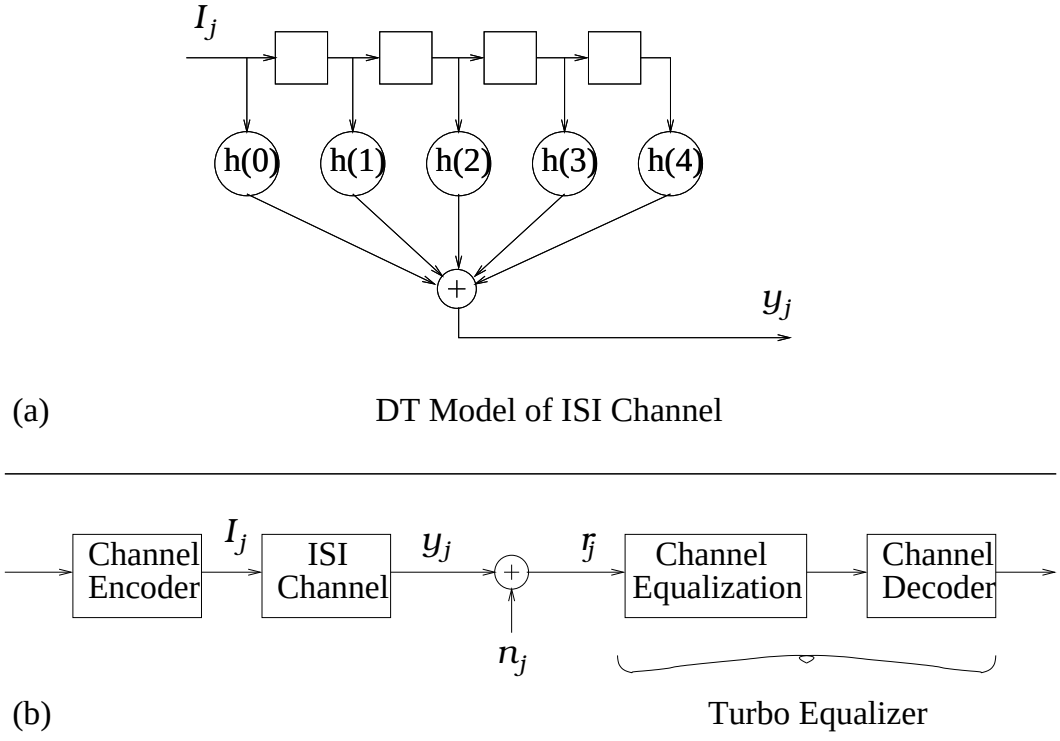


Figure 94: (a) discrete-time ISI channel model; (b) turbo equalization.

9.5 Low Density Parity Check (LDPC) Coding & Decoding

Back in 1962, Gallagher [4] introduced Low Density Parity Check (LDPC) block codes. He described these in terms of the parity check matrix $\mathbf{H}$, proposed code design procedures and discussed suboptimum decoding. The principal LDPC code characteristics are the large size and sparseness of $\mathbf{H}$. That is, an (n, k) LDPC code is a large block code, with a therefore large parity check matrix which is composed of mostly zeros. Being large block codes, it was realized that LDPC codes had the potential for excellent performance, at the expense of decoding complexity. In the 1960's and 1970's, computational resources for practical decoding of LDPC codes did not exist, so this class of codes went largely ignored. In 1981, Tanner [8] formulated LDPC codes in terms of graph theory, which led him to iterative decoding algorithms. Still, LDPC decoding was considered impractical, and this work went virtually unnoticed until the mid 1990's. With the popularity of turbo codes since the 1990's, there has been a growing interest in the application and iterative decoding of LDPC codes. LDPC codes can provide very low BER to within a fraction of channel capacity. They offer several advantages over turbo codes, in that: they don't require a long interleaver to achieve near capacity performance; decoders are not trellis based; their error floor occurs at lower SNRs; and they tend to have better d_{min} properties for the same (n, k).

Example 9.2: As a little example of a LDPC code, consider the following 9×12-dimensional parity check matrix:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} . \tag{28}$$

The null space of this matrix is the code space. Since the row space for this matrix is 9-dimensional (i.e. it is full row rank), the code is a $(12, 3)$ block code. Let m denote the number of rows in $\mathbf{H}$. In this example, $m = 9$. In general, $m \geq n - k$. That is, $\mathbf{H}$ can have more rows than the dimension of the code null space. Although in this example we have $m = n-k$, $m > n-k$ is not uncommon for LDPC codes. Note that each row of $\mathbf{H}$ contains $\rho = 4$ 1's and each column has $\gamma = 3$ 1's. That is, all rows are weight 4 and all columns weight 3. The density of 1's is $r = \frac{\rho}{n} = \frac{\gamma}{m} = \frac{1}{3}$.

A LDPC code is called (ρ, γ)-*regular* if it meets the following two conditions: 1) its parity check matrix has a constant row weight, ρ, and a constant column weight, γ; and 2) any two columns have at most one position where both have value 1. Otherwise, the LDPC code is called *irregular*. The code in Example 9.2 is irregular because it parity check matrix does not satisfy condition 2).

9.5.1 Basic Graph Theory Concepts

To understand LDPC code decoding, it is helpful to couch the codeword estimation problem in graph theory terms. In this Subsection we introduce just enough graph theory concepts to facilitate this.

A *graph* is a collection of *vertices* $\mathcal{V}$ (also called nodes) connected by a set of *edges* $\mathcal{E}$ (also called branches). Figure 95(a) illustrates a graph consisting of a set of 3 vertices, $\mathcal{V} = \{v_1,\ v_2,\ v_3\}$, and and a set of 5 edges $\mathcal{V} = \{e_1,\ e_2,\ e_3,\ e_4,\ e_5\}$. A graph with a finite number of vertices and edges is called a *finite graph*. Each edge is terminated with two vertices. The *degree* of a vertex is the number of edges that connect to it. Vertex v_2 in Figure 95(a) has degree 3. Two edges that connect to the same vertex are said to be *connected*. In Figure 95(a), edges e_2 and e_4 are connected by vertex v_2. A *path* through a graph is a sequence of alternating vertices and edges, where successive edges are connected by the vertex listed in between. The *length* of a path is its number of edges. In Figure 95(a), path $\{v_1,\ e_3,\ v_3,\ e_4,\ v_2,\ e_2,\ v_1\}$ has length 3. A closed path, termed a *cycle*, is a path that begins and ends at the same vertex, but otherwise contains no repeated vertices. In Figure 95(a), path $\{v_1,\ e_3,\ v_3,\ e_4,\ v_2,\ e_2,\ v_1\}$ is a cycle. An edge that begins and ends at the same vertex is a *self-loop*. Edge e_1 in Figure 95(a) is a self-loop. The length of the shortest cycle in a graph is called the *girth* of the graph. The girth of the graph in Figure 95(a) is 1. A *acyclic* graph is one without any cycles. The graph in Figure 95(a) is not acyclic. An acyclic graph has infinite girth. A graph is *connected* if one or more paths exists between every pair of vertices. The graph in Figure 95(a) is connected.

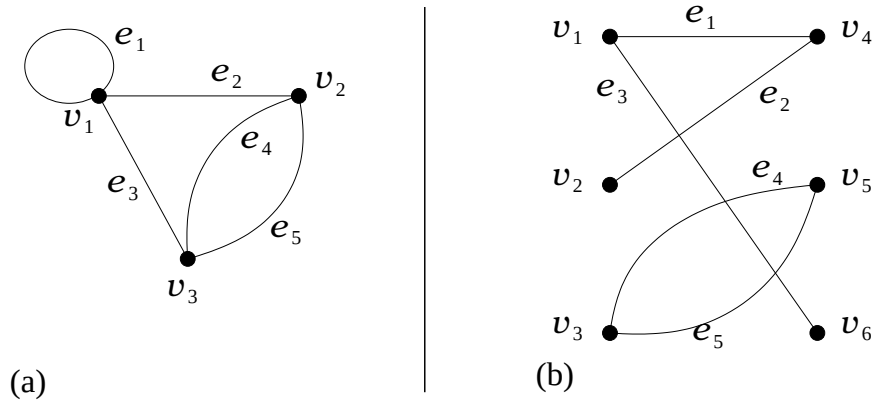


Figure 95: (a) an illustration of a graph, (b) a bipartite graph.

Of interest for representing LDPC codes is a particular type of graph, called a *bipartite graph*. This is a graph with vertices that are partitioned into two sets, where considering each set alone, no vertices are connected. To connect two vertices in the same partition, a path must go through a vertex in the other partition. Figure 95(b) shows a bipartite graph. If a bipartite graph has any cycles, these cycles have even length (greater than or equal to 2).

9.5.2 Graph Representation of LDPC Codes

Linear codes can be represented using graphs. We have seen this already with tree, trellis and state representations of convolutional codes. Block codes can be represented with a bipartite graph called a *Tanner graph* [8]. In a Tanner graph, the vertices $\mathcal{V}$ are partitioned into two sets, the *code-bit vertices* $\mathcal{V}_1$ and the *check-sum vertices* $\mathcal{V}_2$. Figure 96 shows the Tanner graph for the LDPC code in Example 9.2. The 12 code-bit vertices are represented by the circular nodes. As the name implies, these represent the individual code-bits. The 9 check-sum vertices are represented by the squares. These represent the check-sums (i.e. the parity bits) for the codeword. Each check-sum vertex is directly connected by an edge to each vertex of a code-bit used to compute the check-sum. That is, each check-sum vertex represent a row of $\mathbf{H}$. For example, v_{13} is connected directly by edges to v_1, v_2, v_3 and v_4, since the first row of $\mathbf{H}$ has nonzero entries only in the first 4 columns. Similarly, each code-bit vertex is connected through an edge to each vertex of a check-sum uses the code-bit. That is, each code-bit vertex represents a column of $\mathbf{H}$. For example, v_1 is connected directly by edges to v_{13}, v_{15}d and v_{16}, since the first column of $\mathbf{H}$ has nonzero entries only in rows 1, 3 and 4.

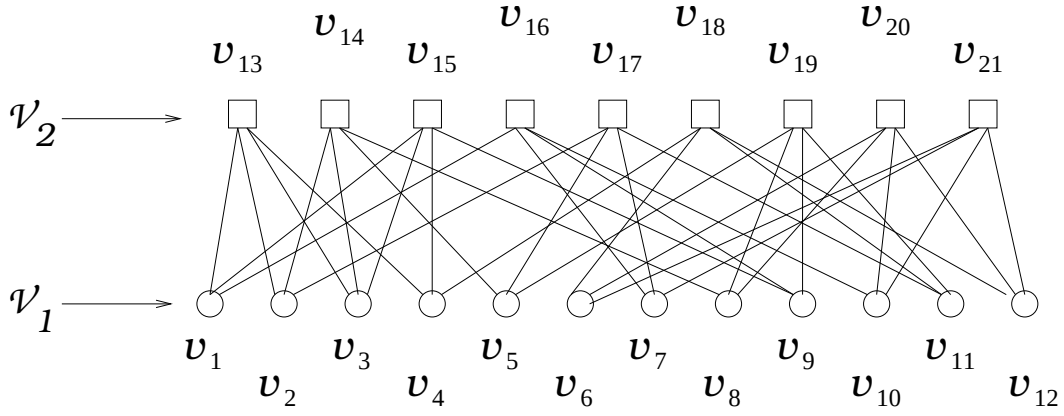


Figure 96: The bipartite graph for a small $(4, 3)$-regular LDPC code.

9.5.3 Decoding LDPC Codes

A number of LDPC code decoding algorithms have been proposed. These include two hard decision algorithms, a *majority-logic decoder* and a *bit-flipping* algorithm, and a soft decision iterative belief propagation algorithm which, in graph theory terminology, is a *Sum-Product Algorithm (SPA)*. Here we first describe the majority-logic decoder, which is the simpler (though lower performing) of the two hard decision decoders. We then discuss the SPA.

A Majority-Logic LDPC Code Decoder

Earlier in this Course, in a discussion on Reed-Muller block codes, we eluded to a majority-logic based decoder which uses check-sums. Majority-logic decoding is widely used. Here we describe its use for LDPC decoding.

Consider a (ρ, γ)-regular (n, k) LDPC code. Consider a code-bit position l, and let

$$\mathcal{A}_l \;=\; \{\mathbf{h}_1^{(l)},\; \mathbf{h}_2^{(l)} \;\cdots,\; \mathbf{h}_\gamma^{(l)}\} \tag{29}$$

be the set of rows of $\mathbf{H}$ that test code-bit position l (i.e the γ rows that have a 1 in position l – $\mathbf{h}_j^{(l)}$ is the j^{th} row with a 1 in the l^{th} position). We say that two rows are orthogonal to the j^{th} code-bit position if, besides the j^{th} position, they have no position where both have value 1. Consider $\mathcal{A}_l;\; l = 1, 2, \cdots, n$. Since, by assumption, the code is (ρ, γ)-regular, for each $\mathcal{A}_l$, all pairs of rows are orthogonal to the l^{th} position.

Let $\mathbf{Y}$ be the n-dimensional received hard decision vector. Consider the set of check-sums (i.e. syndromes) for $\mathcal{A}_l$:

$$\mathcal{S}_l \;=\; \{s_j^{(l)} = \mathbf{h}_j^{(l)} \cdot \mathbf{Y}^T;\; j = 1, 2, \cdots, \gamma\} \quad . \tag{30}$$

With a *majority-logic* algorithm, the check-sum set $\mathcal{S}_l$ is used to estimate the error, e_l, in Y at position l. To see this, note that each $s_j^{(l)};\; j = 1, 2, \cdots, \gamma$ is effected by a possible error at position l, and one other possible error. $s_j^{(l)} = 1$ means that either $e_l = 1$ or the other error is 1. Let K_l be the weight of $\mathcal{S}_l$. The majority-logic decoder rule is simply

$$\hat{e}_l \;=\; \begin{cases} 1 & K_l \geq \gamma/2 \\ 0 & K_l < \gamma/2 \end{cases} \qquad l = 1, 2, \cdots, n \quad . \tag{31}$$

Then, with $\hat{\mathbf{e}} = [\hat{e}_1,\; \hat{e}_2, \cdots,\; \hat{e}_n]$, the codeword estimate is

$$\hat{\mathbf{C}} \;=\; \mathbf{Y} \;+\; \hat{\mathbf{e}} \quad . \tag{32}$$

This simple majority-logic block code decoder can be used for any block code. It does not perform as well as syndrome decoding (i.e. it is not a hard decision ML decoder). However the price may be right. For the (ρ, γ)-regular (n, k) LDPC codes considered here, this simple decoder is guaranteed to correct any $\lfloor \gamma/2 \rfloor$ or less errors. This performance attribute requires that all row pairs in any $\mathcal{A}_l$ by orthogonal to the l^{th} position. Other majority-logic algorithms have been designed for codes that do not satisfy this orthogonality requirement.

A Sum-Product Algorithm (SPA) LDPC Code Decoder

To take full advantage of the performance potential of a LDPC code, soft decision decoding is required. The sum-product algorithm is a general suboptimum approach to iteratively computing, on a graph, a complex multivariate function by partitioning it into a product of local functions on the graph. Applied to LDPC code decoding, it provides a soft decision algorithm which yields performance to within a small fraction of a dB of channel capacity. Here we briefly describe the Sum-Product Algorithm (SPA) for LDPC code decoding. For details see Lin and Costello [6] Subsection 17.6.4.

Consider a (n, k) LDPC code and a parity check matrix $\mathbf{H}$ with m parity check rows, $\{\mathbf{h}_1,\; \mathbf{h}_2, \cdots,\; \mathbf{h}_m\}$, a transmitted codeword $\mathbf{C}$, and a soft received soft decision vector $\mathbf{r}$. The objective is to iteratively compute the code-bit posterior probabilities,

$$P(c_l = 1/\mathbf{r}), \quad P(c_l = 0/\mathbf{r}) \;; \qquad i = l, 2, \cdots, n \quad , \tag{33}$$

and to use these after the final iteration to determine the estimated code-bits as

$$\hat{c}_l^{(I_{max})} = \begin{cases} 1 & P^{(I_{max})}(c_l = 1/\mathbf{r}) > 0.5 \\ 0 & otherwise \end{cases} \qquad l = 1, 2, \cdots, n \quad , \tag{34}$$

where $P^{(i)}(c_l = 1/\mathbf{r})$ is the estimated posterior probability that $c_l = 1$, after the i^{th} iteration, and I_{max} is the final iteration.

For parity check matrix row $\mathbf{h}_j$, let $B(\mathbf{h}_j)$ be the set of 1 positions. For example, for the second row of the $\mathbf{H}$ in Example 9.2, $B(\mathbf{h}_2) = \{2,\ 3,\ 5,\ 8\}$. Let $B(\mathbf{h}_j)|l$ be the set of these indices, excluding the position l. In Example 9.2, $B(\mathbf{h}_2)|5 = \{2,\ 3,\ 8\}$. For code-bit position l and $\mathbf{h}_j \in \mathcal{A}_l\ ;\ j = 1, 2, \cdots \gamma$, let $\mathcal{S}_l^{(i)}|j$ denote the check-sums for $\mathcal{A}_l$ (see Eq (30)), excluding the j^{th} one. Let $\hat{C}^{(i)}$ be the codeword estimate at the i^{th} iteration (see Eq (34)). Let $x \in \{0, 1\}$ represent the possible code-bit values. Given these definitions, let

$$q_{j,l}^{x,(i)} = P(c_l = x/\mathcal{S}_l^{(i)}|j)\ ; \quad l = 1, 2, \cdots, n; \quad j = 1, 2, \cdots, m; \quad x = 0, 1 \tag{35}$$

each denote the probability, at iteration i, that code-bit $c_l = x$, given the check-sums $\mathcal{S}_l^{(i)}|j$. Also let

$$\sigma_{j,l}^{x,(i)}\ ; \quad l = 1, 2, \cdots, n; \quad j = 1, 2, \cdots, m; \quad x = 0, 1 \tag{36}$$

each denote the probability, at iteration i, that the check-sum s_j in the set $\mathcal{S}_l^{(i)}$ is zero, given that $c_l = x$ and that the code-bits in $B(\mathbf{h}_j)$ have a separable distribution. To put it another way,

$$\sigma_{j,l}^{x,(i)} = \sum_{\{v_t : t \in B(\mathbf{h}_j)|l\}} P(s_j = 0/v_l = s, \{v_t : t \in B(\mathbf{h}_j)|l\}) \cdot \prod_{t \in B(\mathbf{h}_j)|l} q_{j,t}^{v_t,(i)} \quad . \tag{37}$$

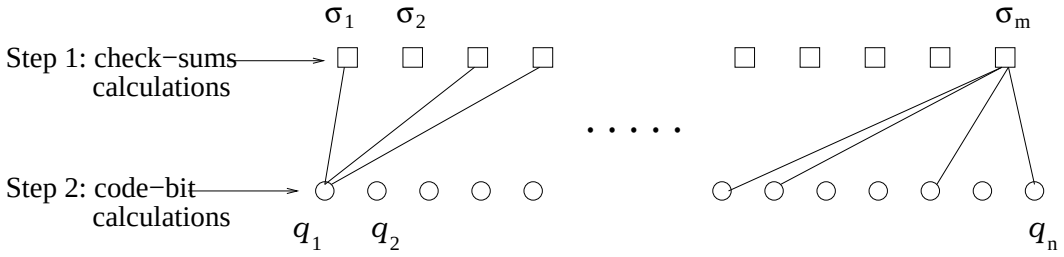


Figure 97: A Tanner graph for a small a LDPC code.

Is this clear!? Perhaps a Tanner graph will help. Figure 97 illustrates a Tanner graph for a LDPC code. The upper vertices represent check-sums, and the lower represent code-bits. In other words, the upper vertices represent $\sigma_{j,l}^{x,(i)}$ calculations, and lower represent $q_{j,l}^{x,(i)}$ calculations. The vertices are labeled to indicate this. Only the edges connected to q_1 and σ_m are shown.

Consider the conditional check-sum probabilities computed at the vertex labeled σ_j. From Eq (37), the $\sigma_{j,l}^{x,(i)};\ l = 1, 2, \cdots, n$ are computed using only information from the code-bit vertices that σ_j is directly connected to. This information is composed of the $q_{j,t}^{v_t,(i)};\ t \in B(\mathbf{h}_j)|l$ and $P(s_j = 0/v_l = s, \{v_t : t \in B(\mathbf{h}_j)|l\})$.

The conditional code-bit probabilities are computed iteratively as

$$q_{j,l}^{x,(i+1)} \;=\; \alpha_{j,l}^{x,(i+1)}\; P(c_l = x) \prod_{\mathbf{h}_t \in \mathcal{A}_l | \mathbf{h}_j} \sigma_{t,l}^{x,(i)} \quad , \tag{38}$$

where $\alpha_{j,l}^{x,(i+1)}$ is a normalizing factor such that $q_{j,l}^{0,(i+1)} + q_{j,l}^{1,(i+1)} = 1$. So, the $q_{j,l}^{x,(i+1)};\; l = 1, 2, \cdots, m$ are computed using only information from the check-sum vertices q_l is directly connected to. This information is composed of the $\sigma_{t,l}^{x,(i)};\; \mathbf{h}_t \in \mathcal{A}_l | \mathbf{h}_j$.

The sum-product algorithm, so called because sum/product calculations are involved, is an iterative procedure as described below:

- Initialize: Set the conditional code-bit probabilities to prior values, i.e. $q_{j,l}^{0,(0)} = P(c_l = 0)$ and $q_{j,l}^{1,(0)} = P(c_l = 1)$ for all (j, l) pairs (i.e. for all the 1's in $\mathbf{H}$). Set $i = 1$ and specify I_{max}.

- *Step 1*: Calculate the conditional check-sum probabilities, as represented by the upper vertices in the Tanner graph illustrated in Figure 97;

- *Step 2*: Calculate the conditional code-bit probabilities, as represented by the lower vertices in the Tanner graph illustrated in Figure 97:

  - If $\hat{\mathbf{C}}^{(i)} \cdot \mathbf{H}^T = \underline{0}_m$ or $i = I_{max}$, stop.
  - otherwise, set $i = i + 1$ and go to Step 1.

The SPA algorithm will typically converge to the soft decision ML solution as long as the LDPC code has a Tanner graph with no short cycles. Recall that a Tanner graph is a bipartite graph, so that cycle lengths are positive even integers. As long as there is not more than a single edge connecting any two vertices, which there are not for LDPC codes, the Tanner graph will have a minimum cycle length of 4. In designing or selecting a LDPC code, length 4 cycles should be avoided. If the LDPC code is (ρ, γ)-regular, then the Tanner graph will have a minimum cycle length of 6 (i.e. there will be no cycles of length 4).

The sum-product algorithm is a general class of iterative algorithms on a particular type of graph called a *factored graph*. The algorithm described above for LDPC codes is one example. The Viterbi algorithm, the BCJR algorithm, the Kalman filter, and some FFT's can be considered other examples of the sum-product algorithm. This unifying view of these iterative (or recursive) algorithms, along with the interpretation that turbo codes are long block codes, has led some information theorists and code designers to argue that, in fact, LDPC codes are the original turbo codes. It's a small world.

10 Space-Time Coding

In this Section of the Course we present an overview of the multipath fading problem and some diversity techniques which compensate for it.

10.1 Multipath Fading Channels and Diversity Techniques

10.1.1 Multipath Fading Channels

Figure 98 illustrates at multipath digital communications channel. Multipath propagation spreads a symbol over time, as observed at the receiver. Depending on the amount of spread, relative to the signal duration, this may or may not result in significant ISI. Additionally, the transmitter and/or receiver may be moving, resulting in a time-varying channel. Depending on the rate of motion (or the Doppler frequency) and the symbol duration (i.e. the signal bandwidth), the channel may or may not appear constant over the symbol duration.

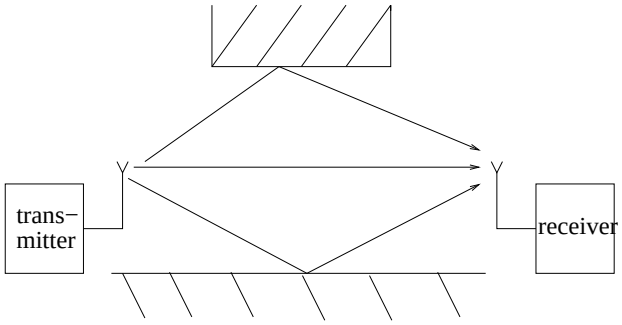


Figure 98: A multipath channel.

Here we model the multipath channel as time-varying and linear. The channel impulse response, at time t, is denoted $c^{'}(\tau;t)$, where τ represents the delay or memory of the channel. For a multipath channel with a countable number of distinct paths, we have that the lowpass equivalent channel impulse response

$$c(\tau;t) \;=\; \sum_{n} \; \alpha_n(t) \; e^{-j2\pi f_c \tau_n(t)} \; \delta(\tau - \tau_n(t)) \quad , \tag{1}$$

where n is the path index, f_c is the carrier frequency, $\alpha_n(t)$ is the n^{th} path gain at time t and $\tau_n(t)$ is the n^{th} path delay at time t. For a general channel, we have

$$c(\tau;t) \;=\; \alpha(\tau;t) \; e^{-j2\pi f_c \tau} \quad . \tag{2}$$

The time varying channel frequency response is

$$C(f;t) \;=\; \int_{-\infty}^{\infty} \; c(\tau;t) \; e^{-j2\pi f \tau} \; d\tau \quad , \tag{3}$$

The CT Fourier transform of the impulse response at time t.

Since the channel characteristics depend on the transmission environment, it is modeled as random. That is, the impulse response $c(\tau;t)$ or equivalently the path gains and delay are considered random. If there are a large number of random paths, the impulse response $c(\tau;t)$ is often modeled as a complex-valued Gaussian process, in which case the envelope of $c(\tau;t)$, $|c(\tau;t)| = \sqrt{Re\{c(\tau;t)\}^2 + Im\{c(\tau;t)\}^2}$, is Rayleigh distributed. If there are several strong, fixed paths, then the envelope is often modeled as Ricean.

For design purposes, multichannel fading channels are characterized in terms of several parameters associated with the channel. For example:

1. T_m is the multipath temporal spread of the channel. It is the memory depth of the channel impulse response.

2. $(\Delta f)_c \approx \frac{1}{T_m}$ is the coherence bandwidth of the channel. For frequencies separated by $\Delta f > (\Delta f)_c$, the channel frequency response $C(f;t)$ is effectively uncorrelated at any time t.

3. B_d is the Doppler spread of the channel. It is the magnitude of the maximum variation of the carrier frequency due to transmitter/receiver motion.

4. $(\Delta t)_c \approx \frac{1}{B_d}$ is the coherence time of the channel. For times separated by $\Delta t > (\Delta t)_c$, the channel impulse response $c(\tau;t)$ is effectively uncorrelated for any memory time τ.

Consider a digital communication modulation scheme that has symbol duration T and bandwidth W (in Hz.) Flat fading, or frequency nonselective fading, occurs when the impulse response is modeled as $c(\tau;t) = c(t)$. It refers to the situation where $T >> T_m$ (i.e. the symbol duration is much greater than the channel spread). For modulations schemes where $W \approx \frac{1}{T}$, the flat fading condition is equivalent to $W << (\Delta f)_c$ (i.e. the signal bandwidth is much less than the coherence bandwidth of channel).

A slowly fading or quasi-static channel refers to a channel that is effectively time invariant over the symbol or processing interval. That is, when effectively $c(\tau;t) = c(\tau)$ which implies that $T << \Delta t_c$.

For a flat fading, slowly fading channel, $c(\tau;t) = c = \alpha\, e^{-j\phi}$. This implies that $T_m\, B_d << 1$. In this case, and for a Rayleigh fading channel, α is Rayleigh distributed and α^2 (the symbol energy gain or loss) is chi-squared distributed.

10.1.2 Diversity Techniques

Channel fading can result in severe degradation in the performance of a digital communication system. This is due to the fact the channel may be in a deep fade (i.e. the channel attenuation may be large) when some bits are transmitted. These bits will not be reliably received. Diversity is used to mitigate channel fading. Generally, diversity refers to transmitting information over different channel conditions. If diversity is designed properly, then these different channel conditions are uncorrelated, and the diversity is referred to a maximum diversity. There are various approaches to diversity. Three common approaches are:

1. Temporal diversity – for which the information is sent through a channel spread out over time. That is, it is sent over the same channel at different times. If the channel is time varying, and if the different times are spread further than the channel coherence time, we can think of the channel at different times as different channels or as having different channel conditions. Temporal diversity typically involves channel coding and interleaving.

2. Frequency diversity – for which the information is sent through a channel, spread over frequency. For example, Frequency Division Multiplexing (FDM) is used for this purpose, where to take full advantage of frequency diversity the different frequency bins should be separated by at least the channel coherence bandwidth. Channel coding techniques (e.g. OFDM) have been adapted for this application.

   Broadband modulation schemes, where $W >> (\Delta f)_c$ so that $T_m << T$, are also used to provide frequency diversity. If in addition to using broadband symbols, if $T_m << T << (\Delta t)_c$ so that the channel is quasi-static and there is no appreciable ISI, a RAKE receiver can be used. The RAKE receiver is essentially a discrete-time filter matched to the symbol shape.

3. Spatial diversity – where multiple transmit and/or receiver antennae are used to transmit information over different physical channels. We discuss this diversity approach in more detail in the next Subsection.

4. Space-time diversity – which combines spatial & temporal diversity techniques.

In the following subsections we first discuss an approach which is applicable for known, flat-fading channels. This is referred to as spatial coding. Subsequently we consider space-time coding, for:

1. known, time-invariant, flat-fading channels;

2. unknown, time-varying, flat-fading channels; and

3. frequency-selective fading channels (unknown/unknown, time-invariant/time-varying).

10.2 A Spatial Diversity Technique

As a prelude to our study of space-time coding, we consider a spatial processing technique for known, flat-fading channels. The parallels between this and temporal coding will be clear, suggesting that coding approaches we have already covered might be considered for space-time coding.

Assume that there are N transmitting antennae, and M receiver antennae. There are $N \times M$ channels, one from each transmitter antenna to each receiver antenna. We assume each channel is flat-fading and constant over a symbol duration, so that the channel from the n^{th} transmitter to the m^{th} receiver is a complex constant α_{mn}. We assume that the $M \times N$

matrix

$$\underline{A} \;=\; \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1N} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M1} & \alpha_{M2} & \cdots & \alpha_{1MN} \end{bmatrix} \tag{4}$$

of channel response constants is known. This Multiple Input Multiple Output (MIMO) channel is illustrated in Figure 99.

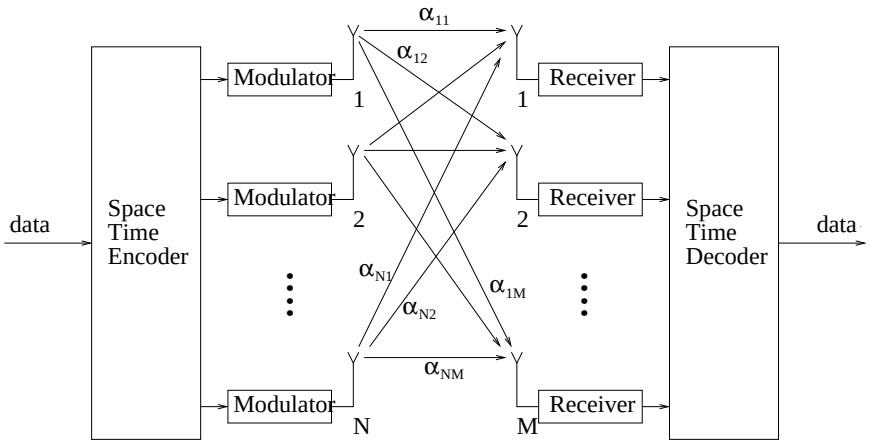


Figure 99: A MIMO flat-fading communications channel.

The input to the transmitter antenna array at time k is the set of N symbols in the vector $\underline{d}$. At the receiver, at time k, our observation across the receiver antenna array after matched filtering and sampling is the M dimensional observation vector

$$\underline{y} \;=\; \underline{A}\,\underline{d} \;+\; \underline{n} \tag{5}$$

where the additive noise vector $\underline{n}$ is assumed to be temporally uncorrelated, zero-mean complex Gaussian with correlation matrix $\sigma_n^2 \underline{I}_M$. The noise $\underline{n}$ and symbols $\underline{d}$ are assumed to be mutually uncorrelated. For now we assume that the sequence of $\underline{d}\,'s$ is statistically independent over time, and since the channels have no memory and the noise is temporally uncorrelated, at the receiver we process our observations, $\underline{y}$, one symbol duration at a time.

Figure 99 illustrates how the transmitter generates a symbol vector from input information bits. Let M_s be the number of symbols of the modulation scheme, common to all transmitters. There are M_s^N possible symbol vectors $\underline{d}$. Let K be the number of bits to be transmitted at a given symbol time, and let $\underline{b}_i; i = 1, 2, \cdots, M_v$ represent the possible vectors of K bits to be transmitted over a symbol time ($M_v = 2^K$). The "serial to parallel" converter takes each $\underline{b}_i$ and maps it to a unique $\underline{d}_i$ symbol vector for transmission. (It is assumed that $M_v \leq M_s^N$.) The rule for this mapping, called the spatial code, is not within the scope of this Course.

At the receiver, the objective is to decide, from the observation $\underline{y}$, which symbol vector from $\underline{d}_i; i = 1, 2, \cdots, M_v$ and corresponding information bit vector $\underline{b}_i$ was sent. This is the receiver detection problem.

Several receiver detection approaches have been suggested. Here we consider the Maximum Likelihood (ML) detector. Assume $\underline{A}$ is known. Under the Gaussian receiver noise assumption stated above, the joint pdf of the received data, $\underline{y}$, conditioned on a transmitted symbol vector $\underline{d}_i$, is

$$p(\underline{y}/\underline{d}_i) \;=\; \frac{1}{(\pi\sigma_n)^{2N}}\; e^{-||\underline{y}-\underline{A}\;\underline{d}_i||^2/\sigma_n^2} \quad . \tag{6}$$

The ML detection problem is

$$\max_{\underline{d}_i} \qquad p(\underline{y}/\underline{d}_i) \tag{7}$$

or

$$\min_{\underline{d}_i} \qquad ||\underline{y} - \underline{A}\;\underline{d}_i||^2 \quad . \tag{8}$$

10.3 Space-Time Block Codes

Space-Time Processing for Known, Time-Invariant, Flat-Fading Channels

Though there had been some isolated developments in joint spatial and temporal diversity earlier in the 1990's (e.g. the delay diversity transmitter), the origin of the general space-time coding approach in the open literature dates back only to 1998 with the publication of V. Tarokh, N. Seshadri and A. R. Calderbank [9]. In this paper, space-time coding criteria were developed, and trellis space-time codes for flat-fading (no ISI), time-invariant (quasi-static) channels were considered in detail. Two space-time code design/performance criteria were established, which deal with the differences between pairs of space-time code matrices. These criteria, a rank criterion and a determinant criterion, will be presented below. They can be thought of as generalizations of the distance criterion for block channel codes.

Below, we will focus on space-time block codes (STBC's), which compared to trellis space-time codes are conceptually simpler and have reduced receiver complexity. STBC's and decoding can be thought of as a generalization of the spatial diversity approach described above.

Consider transmitting a block of T symbols (time slots) over each of N transmit antennae using a M_s symbol modulation scheme. Let $c_{n,t}$ be the symbol transmitted from the n^{th} antenna at symbol time t. The matrix

$$\underline{C} \;=\; \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,T} \\ c_{2,1} & c_{2,2} & \cdots & \alpha_{2,T} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N,1} & \alpha_{N,2} & \cdots & \alpha_{N,T} \end{bmatrix} \tag{9}$$

is a code matrix. There are M_s^{NT} possible code matrices. Let K be the number of bits to be represented by a code matrix $\underline{C}$. Figure 100 depicts the STBC transmitter.

With $M_b = 2^K$, let $\underline{b}_m; m = 1, 2, \cdots, M_b$ represent all possible K-bit vectors. A STBC is a mapping for each $\underline{b}_m$ to a unique $\underline{C}_m$ code matrix. Let $\{\underline{C}_m;\; m = 1, 2, \cdots, M_b\}$ be the set code matrices for a STBC. From the M_s^{NT} possible code matrices, we should choose the $\underline{C}_m; m = 1, 2, \cdots, M_b$ to be in some meaningful sense as different from one another as possible.

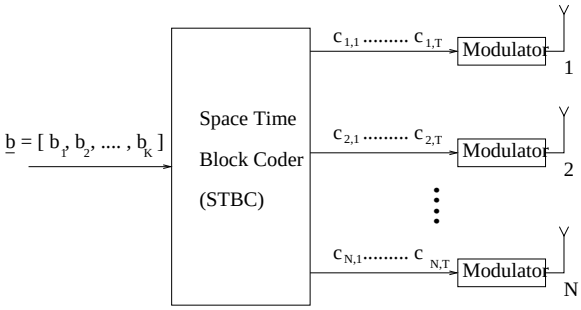


Figure 100: The STBC transmitter.

Recall that for block codes, the principal measure dictating performance is d_{min}, the minimum Hamming distance between codewords. We now describe analogous measures for STBC's.

Assume the channels are memoryless and time-invariant. Let α_{ij} be the complex *Gaussian* channel gain from transmitter i to receiver j. For zero mean α_{ij}, the channel is Rayleigh fading. Let $\underline{B}_{mk} = \underline{C}_m - \underline{C}_k$, and let $\underline{A}_{mk} = \underline{B}_{mk}\underline{B}_{mk}^H$. Denote as $\lambda_i; i = 1, 2, \cdots, R$ the nonzero eigenvalues of $\underline{A}_{mk}$. For Rayleigh fading channels, using an ML receiver, the probability of detecting $\underline{C}_k$ given that $\underline{C}_m$ was transmitted, is bounded as

$$P(\underline{C}_k/\underline{C}_m) \;\leq\; \left(\prod_{i=1}^{R}\lambda_i\right)^{-N} \left(\frac{\mathcal{E}_s}{4N_0}\right)^{-RM} \tag{10}$$

where, as before, M is the number of receiver antennae.

- The *diversity advantage*, (RM), is the power on the inverse SNR $\left(\frac{\mathcal{E}_s}{4N_0}\right)^{-1}$. The bigger, the better.
- The *rank criterion* states that the difference matrix $\underline{B}_{mk}$ should be full rank for all $m \neq k;\ m, k = 1, 2, \cdots, M_b$. Then $R = N$ is maximum.
- The *coding advantage*, $\left(\prod_{i=1}^{R}\lambda_i\right)^{-N}$, is the gain over an uncoded system with same diversity.
- The *determinant criterion* states that if $R = N$, the minimum determinant $|\underline{A}_{mk}|$, over all $m \neq k;\ m, k = 1, 2, \cdots, M_b$, should be maximized.

Let $r_j(t);\ j = 1, 2, \cdots, M;\ t = 1, 2, \cdots, T$ be the data received by the j^{th} receiver at symbol time t over the duration of a code matrix transmission. Let $\underline{r}$ be the NT-dimensional received data vector for a transmitted code matrix. The ML estimator is the solution to:

$$\hat{\underline{C}}_m \;=\; \arg\max_{\underline{C}_i} \;\; p(\underline{r}/\underline{C}_i) \tag{11}$$

A popular STBC for $N = 2$ transmitting antennae and $T = 2$ time slots was proposed by Alamouti [10]. Assuming an M_s symbol modulation scheme, let $c_i;\ i = 1, 2$ be two symbols

that represent $K = 2 \cdot \log_2(M_s)$ bits. The M_v code matrices representing the K bits are of the form

$$\underline{C} \;=\; \left[\begin{array}{cc} c_1 & -c_2^* \\ c_2 & c_1^* \end{array} \right] \quad . \tag{12}$$

This STBC has a code rate equal to 1 (i.e. on the average, one symbol per symbol-time is transmitted). Besides having good diversity characteristics (i.e. it satisfies the rank criterion), this STBC, called the Alamouti code, results is a particularly simple ML decoding structure.

At the receiver, a simple preprocessor, termed as space-time matched filter, can be used to decouple the ML estimation of c_1 and c_2. Figure 101 illustrates the decoder for the $M = 1$ receiver antenna case. (We now use h to represent channel response.)

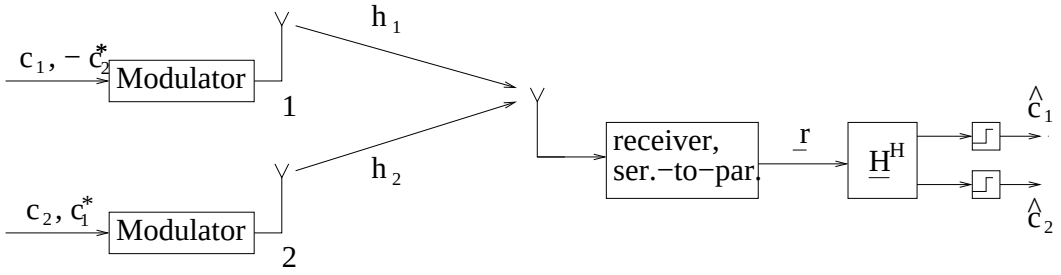


Figure 101: The Alamouti STBC transmitter and ML receiver for $M = 1$ receiving antenna.

The received data vector is

$$\underline{r} \;=\; \left[\begin{array}{c} r_1 \\ r_2^* \end{array} \right] \;=\; \left[\begin{array}{cc} h_1 & h_2 \\ -h_1^* & h_2^* \end{array} \right] \left[\begin{array}{c} c_1 \\ c_2 \end{array} \right] \;+\; \left[\begin{array}{c} n_1 \\ n_2^* \end{array} \right] \;=\; \underline{H}\,\underline{c} \;+\; \underline{n} \quad . \tag{13}$$

Since the columns of $\underline{H}$ are orthogonal, $\underline{H}^{-1} \propto \underline{H}^H$, and

$$\underline{y} \;=\; \left[\begin{array}{c} y_1 \\ y_2 \end{array} \right] \;=\; \underline{H}^H\,\underline{r} \;=\; (|h_1|^2 + |h_2|^2) \left[\begin{array}{c} c_1 \\ c_2 \end{array} \right] \;+\; \left[\begin{array}{c} n_1^{'} \\ n_2^{'} \end{array} \right] \quad , \tag{14}$$

where $n_1^{'}$ and $n_2^{'}$ are uncorrelated because n_1 and n_2^* are uncorrelated and equal-variance, and $\underline{H}^H$ is unitary to within a scalar. Thus, y_1 and y_2 can be processed separately to optimally estimate c_1 and c_2, respectively. $\underline{H}^H$ is the space-time matched filter.

The Alamouti STBC was extended to $N = 4$ and $N = 8$ transmit antennae in a paper by V. Tarokh, H. Jsfsrkhsni and A. R. Calderbank [11], 1999, on orthogonal STBC's. Good nonorthogonal codes for $N = 3, 5, 6$ *and* 7 codes are also described.

Space-Time Processing for Unknown, Time-Varying, Flat-Fading Channels

We just discussed STBC's for flat-fading, quasi-static channels. Quasi-static means that the channel responses did not change over the temporal length of a STBC matrix. For example, the STBC proposed by Alamouti was designed for channels that can be assumed to be constant over two successive symbol time slots, and for decoupled optimum decoding the channel responses were assumed constant over the block, and known at the receiver. Even though in the design of these STBC's, quasi-static channels are assumed, these codes can be used for channels which are varying over a code block duration. However, the decoder structures designed under the quasi-static assumption will likely not be optimum. We now consider STBC decoding for unknown, time-varying, flat-fading channels. Specifically, as an example, we consider decoding of the Alamouti STBC. We will consider both the exact ML decoder and a computationally efficient suboptimum decoder.

First we need a little background on MLSE for unknown, time-varying channels. Consider the single transmitter antenna, single receiver antenna case. Let h_k be the unknown, time-varying, flat-fading channel response at time k, which is modeled as

$$h_{k+1} \; = c \cdot h_k \; + \; w[k] \quad , \tag{15}$$

where c is a known complex constant with $|c| < 1$, and $w[k]$ is an uncorrelated Gaussian random process. This is a first order Gauss-Markov channel model. Consider transmission of n symbols I_k; $k = 1, 2, \cdots, n$, received in AWGN, as

$$r_k \; = \; h_k \; I_k \; + \; n_k \; ; \qquad k = 1, 2, \cdots, n \quad . \tag{16}$$

The joint pdf of the received data, conditioned on the channel coefficients and symbols, is

$$p(\underline{r}/\underline{I}, \underline{h}) \; = \; \frac{1}{(2\pi\sigma^2)^n} \; e^{-\frac{1}{\sigma^2}\sum_{k=1}^{n} |r_k - h_k \cdot I_k|^2} \tag{17}$$

where σ^2 is the variance of the noise n_k. Marginalizing over $\underline{h}$, the joint pdf of $\underline{r}$ conditioned of $\underline{I}$ is

$$p(\underline{r}/\underline{I}) \; = \; \int_{\underline{h}} p(\underline{r}/\underline{I}, \underline{h}) \; p(\underline{h}) \; d\underline{h} \quad . \tag{18}$$

The MLSE problem is

$$\max_{\underline{I}} \quad p(\underline{r}/\underline{I}) \tag{19}$$

where the data $\underline{r}$ has been plugged into $p(\underline{r}/\underline{I})$ (i.e. maximize the likelihood function).

The key point here is that Eq (19) does not reduce to a decoupled symbol-by-symbol detection receiver. This is because of the correlation over time of the unknown channel response. Because of this, data at all time can be effectively used to estimate the symbol at any time.

We know that for a M_s symbol modulation scheme, the number of possible $\underline{I}$ sequences up to time n is M_s^n. Figure 102 shows the trellis representation of these possible sequences for $M_s = 2$. Each sequence is a path through the trellis. It has been shown, for example in Iltis [12] and Chen, Perry and Buckley [13], that for each possible sequence, the cost $p(\underline{r}/\underline{I}) = \int_{\underline{h}} p(\underline{r}/\underline{I}, \underline{h}) p(\underline{h}) d\underline{h}$ is computed using a channel estimator which is conditioned on

that sequence. The Iltis paper shows that, for the Gauss-Markov channel model such as described above, the conditional channel estimator can be implemented as a Kalman filter (see Iltis for details). The conditional channel estimate generated for each sequence will be a function of the entire sequence and data history. So, in terms of the trellis representation, a Kalman channel estimator must be run for each path through the trellis. Since this Kalman estimator has memory back to time $k = 0$, each path through a branch will have its own branch cost, which means that any path into a state, regardless of its current cost relative to other paths into that state, might turn out to be the best path sometime in the future. Thus, no optimum pruning of paths is possible. The Viterbi algorithm can not be optimally applied, and the MLSE solution requires computation of all M_s^n sequence costs. This, of course, is not practical for large n.

In Chen, Perry and Buckley [13] a suboptimum but effective pruning algorithm, based on the List Viterbi algorithm, is described. The algorithm is in the general sequence estimator class of Per-Survivor Processing (PSP) algorithms, because a conditional Kalman channel estimator is used for each survivor (unpruned) sequence.

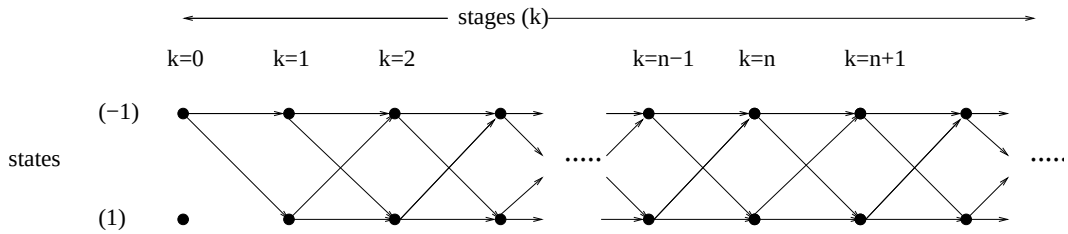


Figure 102: Trellis diagram representation of symbol sequences for a flat-fading, $M_s = 2$ symbol modulation scheme.

Returning to decoding algorithms for the Alamouti STBC for unknown, time-varying, flat-fading channels, consider a sequence of transmitted STBC matrices, $\underline{C}_k;\ k = 1, 2, \cdots, n$, and received data $\underline{r}_k;\ k = 1, 2, \cdots, n$, where $\underline{r}_k$ is the received data for block k. For notational simplicity, assume $M = 1$ receiving antenna, so that $\underline{r}_k = [r_{1,k}, r_{2,k}^*]^T$, the two receiver antenna outputs for the two time slots corresponding to the reception of the $\underline{C}_k$ transmitted STBC matrix. Let the two channels between the two transmitter antennae and the one receiver antenna be modeled as

$$h_i[l+1] \ = c_i \cdot h_i[l] \ + \ w_i[l]; \quad i = 1, 2 \tag{20}$$

i.e. two Gauss-Markov channels with known $c_i's$. Let

$$\underline{H}_k = \begin{bmatrix} h_1[2k-1] & h_2[2k-1] \\ -h_1^*[2k] & h_2^*[2k] \end{bmatrix} \ . \tag{21}$$

Assume that the receiver noise is AWGN.

Let $\underline{\mathcal{C}} = [\underline{C}_1, \underline{C}_2, \cdots, \underline{C}_n]$, $\underline{R} = [\underline{r}_1, \underline{r}_2, \cdots, \underline{r}_n]$, and $\underline{\mathcal{H}} = [\underline{H}_1, \underline{H}_2, \cdots, \underline{H}_n]$. The MLSE problem is:

$$\max_{\underline{\mathcal{C}}} \quad p(\underline{R}/\underline{\mathcal{C}}) \tag{22}$$

$$p(\underline{R}/\underline{\mathcal{C}}) \;=\; \int_{\underline{\mathcal{H}}} p(\underline{R}/\underline{\mathcal{C}},\underline{\mathcal{H}})\; p(\underline{\mathcal{H}})\; d\underline{\mathcal{H}} \tag{23}$$

and

$$p(\underline{R}/\underline{\mathcal{C}},\underline{\mathcal{H}}) \;=\; \frac{1}{(2\pi\sigma^2)^{2n}}\; e^{-\frac{1}{\sigma^2}\sum_{k=1}^{n}|\underline{r}_k-\underline{H}_k\cdot\underline{c}_k|^2} \tag{24}$$

Because of the correlation across time of the unknown, time-varying channel responses, solution of the MLSE problem, Eq (22), requires an exhaustive search of all STBC matrix sequences $\underline{\mathcal{C}}$. Since this is impractical, a PSP based algorithm similar to the one described in the Chen paper can be employed.

Another suboptimum decoding algorithm for the Alamouti STBC for unknown, time-varying, flat-fading channels is described in Liu, Ma and Giannakis [14]. This algorithm, instead of using a conditional Kalman channel estimator for each sequence, runs a single Kalman filter channel estimator conditioned on a single estimated sequence.

Space-Time Processing for Unknown, Time-Varying, Frequency-Selective Fading Channels

For unknown, quasi-static, frequency selective (ISI) channels, a generalization of the Alamouti STBC has recently been proposed in Lindskog and Paulraj [15]. The proposed code is referred to as the Time-Reversed STBC (TR-STBC). Code matrices are $2 \times P$ dimensional, where $P \geq 2$ ($P = 2$ for the Alamouti STBC). For $P > 2$, code matrices contain more than two information symbols, and pilot (known) symbols are inserted and used at the receiver to estimate the ISI channel, which is assumed constant over the P time slot transmission duration of a code matrix. As with the Alamouti STBC, code matrices are constructed so that a simple space-time matched filter can be used at the receiver to decouple the estimation of two sets of symbols.

For ISI channels that vary over a code matrix duration, a TR-STBC can still be used. However, the decoupling receiver can not be employed, and the pilot bit approach to channel estimation can no longer be exploited. Alternatively, for any STBC, whether designed specifically for ISI channels or not, a MLSE based PSP algorithm can be used. (See, for example, Wang, Buckley and Perry [16].)

11 Trellis Coded Modulation (TCM)

In this Section of the Course we consider bandwidth efficient modulation in conjunction with channel coding, and we specifically describe Trellis Coded Modulation (TCM) which is a bandwidth efficient joint channel-encoding/modulation approach to digital communication.

In Sections 7 & 8 0f this Course, we learned that block and convolutional channel encoding can be used to improve BER performance, though this comes with a price of increased transmission bit rate, which translates to increased transmission bandwidth. For example, if BPSK modulation is used in conjunction with a rate R_c code, then for a fixed SNR/bit (γ_b), BER is improved. However, compared to uncoded transmission the required bandwidth W is increased by a factor of R_c^{-1}. So, this is not bandwidth efficient, and and any improvement realized should be measured against channel capacity, which increases with W. On the other hand, in our consideration of modulation schemes in Section 3 of the Course, we observed that higher order schemes such as M-ary PSK and M-ary QAM exploit bandwidth efficiently. These schemes have bandwidth efficiency, measured in bits/sec./Hz., of $\log_2(M)$. However, with increasing order M, greater SNR/bit γ_b is required to achieve the same BER performance. So, this is not power efficient, and again it is not so clear if a significant improvement has been realized as measured against channel capacity, which increases with γ_b.

It is natural to consider combining channel coding and bandwidth efficient modulation. It provides us with greater flexibility in controlling BER by trading off bandwidth and SNR/bit. However, in addressing this tradeoff, it appears that coding and higher order modulation work against each other. TCM combines the encoding and modulation scheme design problems. It thereby effectively provides improved performance (i.e. reduces BER) for fixed γ_b and W compared to digital communication systems in which the coder and modulator are designed separately. Thus, it gets us closer to channel capacity. In TCM, a convolutional encoder, termed a trellis encoder, is used in conjunction with a bandwidth efficient higher order modulation scheme such as M-ary PSK or QAM. The critical TCM design feature is the assignment of encoder outputs to modulation symbols. Qualitatively, encoder outputs which are not easily differentiated by the decoder are assigned symbols which are easily differentiated by the demodulator.

In Subsection 11.1 we first organize basic concepts from previous Course discussions which are subsequently relevant. Then, in Subsection 11.2, we describe *trellis encoding*, which is basically just convolutional encoding described using the trellis diagram representation. Next, in Subsection 11.3, we introduce *set partitioning*, which leads to the assignment of encoder outputs to modulation symbols, and is thus the principal feature of TCM. In Subsection 11.4 we describe TCM. In Subsection 11.5 we cover TCM decoding and performance.

Finally, it should be noted that over the past decade another approach to bandwidth efficient coded digital communication has received a great deal of attention. This approach is Space-Time Coding (STC), just covered in Section 10.

11.1 Introduction

Recall from Subsection 3.3 of this Course that Figure 103 provides a comparison, in terms of bandwidth efficiency and SNR/bit, of a number of modulation schemes to channel capacity. Specifically, the channel capacity of a memoryless Gaussian channel is plotted (in bits/sec./Hz. (C/W) vs. γ_b) along with R/W vs. γ_b for the modulation schemes for a symbol error probability rate of $P_e = 10^{-5}$. The channel capacity curve is a bound. For a fixed γ_b it tells us the highest bit rate per Hertz that can be transmitted reliably (without error). Compared to this bound, for $P_e = 10^{-5}$, we see that none of the modulation schemes considered come very close to achieving channel capacity. For a fixed R/W, none of the considered modulations schemes are within 7dB SNR/bit of channel capacity. For a higher level of performance (i.e. $P_e < 10^{-5}$) it only gets worse.

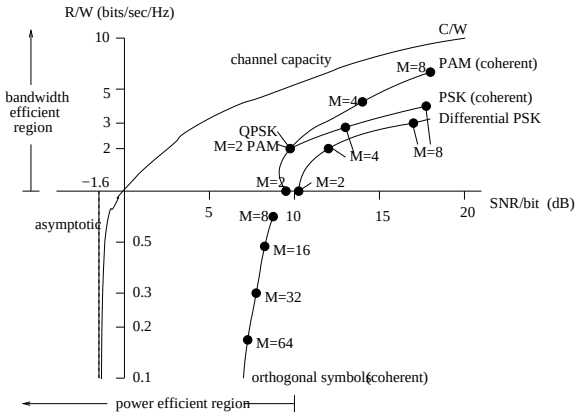


Figure 103: Bandwidth efficiency and power requirements for several modulation schemes.

The bandwidth efficient region of this Figure, where $R/W > 1$, is of interest in this discussion. Higher order PSK, PAM and QAM modulation schemes allow us higher R/W, but always at an expense of higher γ_b for a given P_e. Increasing the number of symbols M increases R without increasing bandwidth W, but since the symbols are more tightly packed in the signal space, γ_b must be increased to maintain a constant minimum Euclidean distance. The curves do suggest that for QAM things improve with increased M, but this is with increased complexity and there is still an appreciable gap between the required γ_b and the bound.

For a channel encoder, the *coding gain* is the reduction in SNR/bit, γ_b, allowed through the employment of the code, to achieve the same error performance compared to the uncoded system. This gain can be identified, for a given level of performance, by comparing performance as indicated by performance equations and curves discussed previously.

Example 11.1: Consider employing uncoded binary PSK with coherent detection. A BER of $P_b = 10^{-5}$ is achieved with a SNR/bit of $\gamma_b = 9.5dB$ (see Figure 21 of these Course notes). Alternatively, using a rate $R_c = \frac{1}{3}$ convolutional encoder with constraint length $K = 3$, along with coherent, binary PSK and soft decision decoding, a BER of $P_b = 10^{-5}$ is achieved with $\gamma_b = 7db$ (see Subsection 7.4.1 of these Course notes, and Proakis & Salehi [7] Figure 8.2-1). Thus the coding gain is 2.5dB. Note that the price of this coding gain is a tripling of bandwidth (along with increased transmitter and receiver complexity).

To compensate for the tripling of bit rate from the encoder input to output, $M = 8$-PSK can be considered instead of binary PSK. From Proakis Salehi [7] Figure 4.3-5, this would appear to require an additional SNR/bit of about 4dB (to achieve the same symbol error probability ($P_e = 10^{-5}$)).

For a given code rate R_c, coding gain can be improved to a certain extent by increasing the constraint length K. This is in effect the idea behind Turbo coding. Another basic approach to reducing P_b while keeping R_c large, which directly considers joint channel code and modulation scheme design, is TCM.

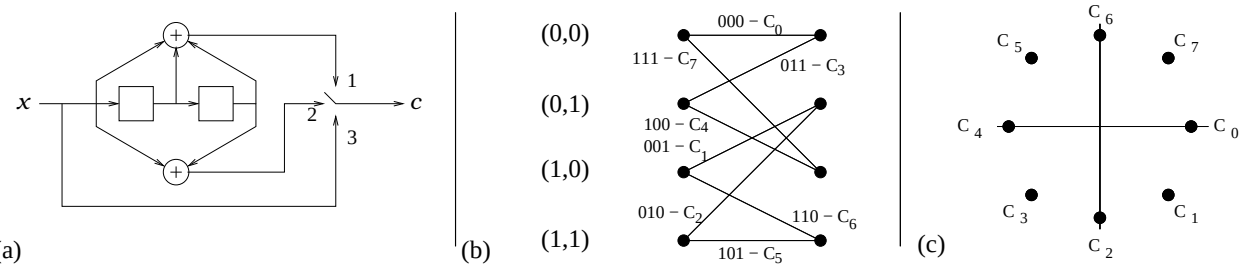


Figure 104: (a) a rate $R_c = \frac{1}{3}$ convolutional encoder; (b) its corresponding trellis representation; c) the 8-PSK signal constellation.

11.2 Trellis Coding with Higher Order Modulation

Figure 104(a) is the block diagram representation of a rate $R_c = \frac{1}{3}$ convolutional encoder. Figure 104(b) is a stage of the corresponding trellis diagram. It shows directly, as branch labels, the output bits that are generated given a previous state and a current input. We call this encoder a trellis encoder in that its output is dictated by trellis branches labels. In Section 8 of this Course, in our discussions on convolutional code decoding and performance, we used the BPSK modulation scheme as an example. More generally consider the use of a M^{th} order modulation scheme. Specifically, since $n = 3$ code bits are generated each encoder cycle time, resulting in one of 8 possible 3-bit words, consider $M = 8$. Figure 104(c) shows the 2-dimensional signal space representation of the 8-PSK (i.e. its constellation). Since M, the number of symbols, is equal to 2^n, the number of possible encoder outputs at a given time, we can assign a modulation symbol to each encoder output. Note that for this example the information bit bandwidth efficiency (i.e. the number of information bits per second per Hertz) is $R_c \log_2(M) = 1$. We have not increased the required bandwidth beyond that required for BPSK and uncoded bits.

Since in this example there are 8 branches/stage for this encoder, each branch is assigned a unique symbol. This would not be the case, for example, for a rate $\frac{2}{3}$ encoder which has 16 branches per stage. The mapping of encoder 3-bit outputs to modulation symbols is not unique. In Figure 104(c) we show one such assignment, selected arbitrarily. This assignment is not a good one since, for example, the nearest symbol neighbors C_0 and C_7 differ in all 3 bits. Thus, is C_7 is detected instead of transmitted C_0, which is relatively likely since they are nearest neighbors, 3 code bit errors are made, which the code will have trouble correcting.

In summary, a basic point of TCM is that we can improve BER performance through intelligent selection of this mapping in conjunction with the trellis code used to generate the codewords.

11.3 Set Partitioning

In [17] Ungerboeck proposed an approach to mapping trellis code outputs to modulation symbols which is based on: 1) *set partitioning* of the symbols; and 2) a particular channel coding strategy. In this Subsection, we describe the approach to set partitioning.

Start with a M order modulation scheme, where we assume that $M = 2^n$ and n is a whole number. We wish to successively partition the symbols, subdividing each partition into two new partitions at each stage, in such a way that symbols within the new partitions have as the largest minimum Euclidean distance as possible. In this way, probability of error in differentiating between symbols within a partition will be as small as possible.

Figures 105 and 106 illustrate the approach for 8-PSK and 16-QAM, respectively. Note that at each stage of partitioning, symbols within a partition are further separated. For 8-PSK and Figure 105, the original minimum symbol distance is $d_0 = \sqrt{(2-\sqrt{2})\mathcal{E}}$ where $\mathcal{E}$ is the symbol energy. After the first level of partitioning, the minimum symbol distance is increased to $d_1 = \sqrt{2\mathcal{E}}$, and after the second partitioning, it is $d_2 = 2\sqrt{\mathcal{E}}$. For 16-QAM and Figure 106, the original minimum symbol distance is $d_0 = 2\sqrt{\mathcal{E}}$. After the first level of partitioning, the minimum symbol distance is increased to $d_1 = 2\sqrt{2\mathcal{E}}$. After the second partitioning, it is $d_2 = 4\sqrt{\mathcal{E}}$.

11.4 Trellis Coded Modulation

As illustrated in Figure 107, with TCM, m information bits are first divided into two groups, one of k_1 bits and one with k_2 bits. The k_1 bits are encoded with a rate $R = \frac{k_1}{n}$ trellis encoder, and the k_2 bits are uncoded. The n coded bits are used to select one of the 2^n partitioned sets, and the k_2 uncoded bits are used to select a element of the selected partition. So, the required number of symbols in the modulation scheme is $M = 2^n \cdot 2^{k_2}$. Set partitioning is used to divide these symbols into 2^n partitions of 2^{k_2} symbols each. The key point here is that the k_2 uncoded bits are represented by symbols that are well separated from one another. And, though the Euclidean distances between the partitions can be small, because the bits used to select the partition are coded, correct selection of the partition is enhanced.

With TCM, one symbol from an M symbol constellation is selected for each $m = k_1 + k_2$ bits. The number of bits/symbol, m, is less than $\log_2(M)$ since $n + k_2 = \log_2(M)$. So

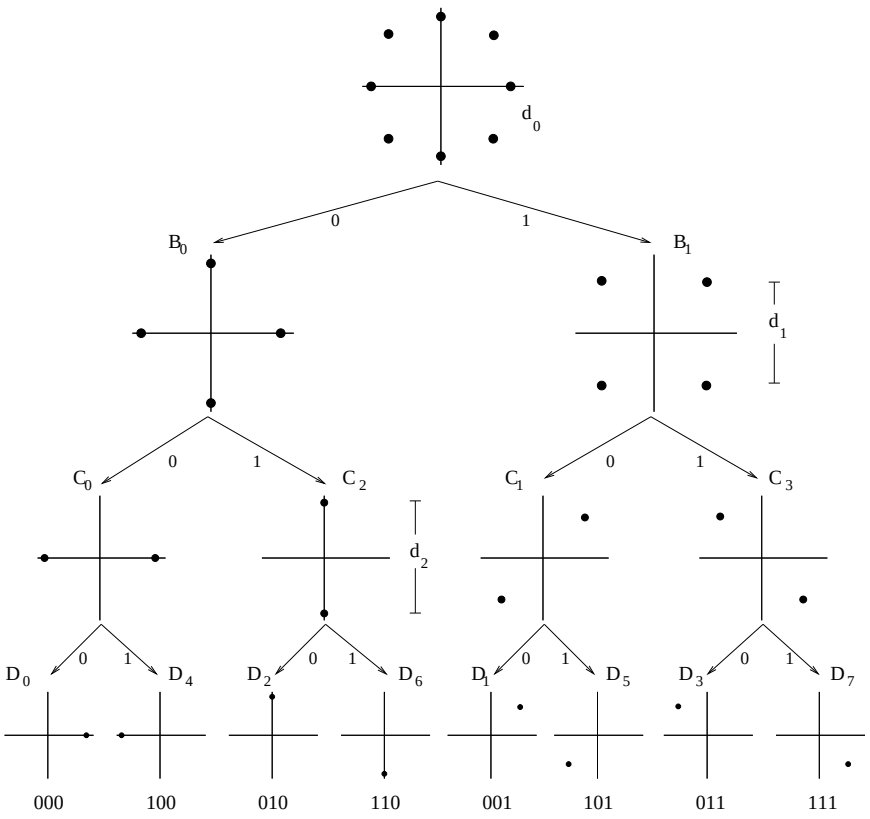


Figure 105: Set partitioning for 8-PSK.

the bandwidth efficiency, m bits/sec./Hz., is less than that of uncoded M-ary modulation. However, the coding gain will more than compensate for this.

To completely describe the TCM approach, one critical issue remains. That is, how are the n bit outputs of the encoder assigned to the partitioned sets so as to take full advantage of the encoding? Ungerboeck suggested a procedure based on the three rules which follow. We will use the rate $R = \frac{k_1}{n} = \frac{2}{3}$ constraint length $K = 2$ convolutional encoder[4] illustrated in Figure 108 as an example to emphasize the rationale behind these rules. $n = 3$ often implies that there are $2^n = 8$ partitioned sets which we will denote $D_i; i = 0, 1, \cdots, 7$. These could, for example, be the 8 1-bit partitioned sets shown in Figure 105 for 8-PSK or the 8 2-bit partitioned sets shown in Figure 106 for 16-QAM.

Rule 1: Use all partitioned sets with equal frequency. In the example, there are 16 branches per trellis stage, and $8 = 2^3$ 3-bit encoder outputs. Note that each 3-bit output is used twice (i.e. each is the label for 2 branches). This is an encoder characteristic. Since there are more branches per stage than partitioned sets (16 vs. 8), at least some of the partitioned sets must be used to represent more than one branch. If each partitioned set represents 2 branches, than all partitioned set will be used with equal probability (assuming al state transitions are equally probable).

[4]The original TCM approach employs a convolutional encoder. More recently, block encoders have been considered for *Block Coded Modulators (BCM's)*.

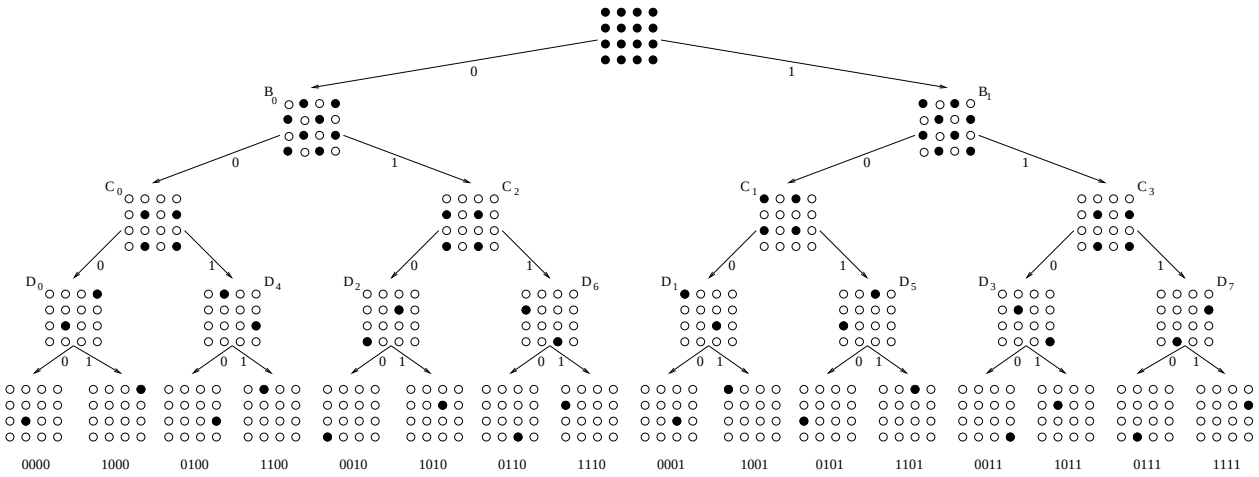


Figure 106: Set partitioning for 16-QAM.

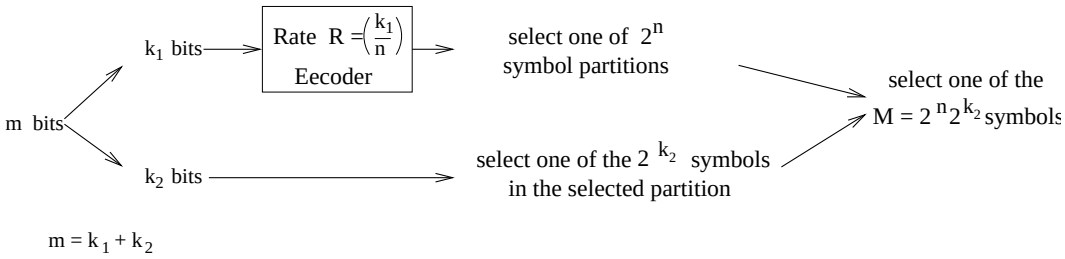


Figure 107: TCM higher order modulation scheme symbol selection.

Rule 2: Branches that originate from the same state or merge to the same state should be assigned partitioned sets that are as far apart as possible in Euclidean distance. This will force the Euclidean distance between two trellis path segments that diverge from a state an then merge at a later state to be large, thus making the path segments more distinguishable. In our example, states have 4 branches both in and out. So we need groups of 4 partitioned sets that are far apart. With either the 8-PSK or the 16-QAM modulation schemes shown in Figures 105 and 106, the groups $\{D_0, D_2, D_4, D_6\}$ and $\{D_1, D_3, D_5, D_7\}$ of partitioned sets are the clear choices. Groups $\{D_0, D_1, D_6, D_7\}$ and $\{D_2, D_3, D_4, D_5\}$ are not so good. If we assign groups to branches as shown in Figure 108(b), then for the branches originating from the states, Rule 2 is well adhered to. However, for the branches merging to the states, the assignment is not so good.

Rule 3: Parallel branches (i.e. same state to same state) should be assigned partitioned sets that are as far apart as possible in Euclidean distance. Parallel transitions do not occur in the example.

For the rate $\frac{2}{3}$ trellis encoder we have been considering, Rule 2 can not be well satisfied.

In the example carried through above in our discussion on partitioned set assignment to branches, we considered a trellis encoder that can be used as a component of a TCM encoder (e.g. if $k_2 = 1$ and 16-QAM is employed) or as a complete TCM encoder (e.g. if $k_2 = 0$ and 8-PSK is used). We now consider a complete TCM encoder for a $k_2 > 0$ case.

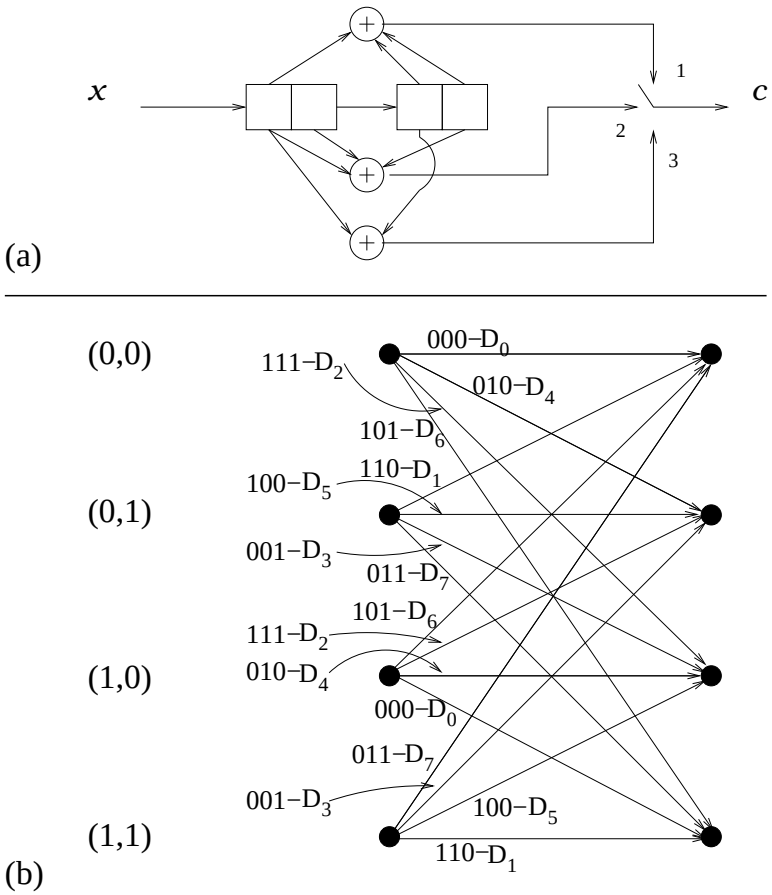


Figure 108: The rate $\frac{2}{3}$ trellis encoder used to illustrate Ungerboeck's partitioned set assignment rules.

Example 11.2: Consider a $k_1 = 1$, $n = 2$, $k_2 = 1$ TCM encoder used in conjunction with an 8-PSK modulator. Figure 109(a) shows the TCM encoder with uncoded bit c_3 and a rate $R_c = \frac{1}{2}$ trellis encoder generating output bits $\{c_1, c_2\}$. The constituent encoder trellis diagram is shown in Figure 109(b). Note that in the branch to partitioned set assignments, the Ungerboeck rules are adhered to. Each branch will represent a trellis encoder output plus an uncoded bit. For example, the top branch in a stage represents the transition from state (0,0) to (0,0), along with the c_3 bit, which can be 0 to 1. So the branch represents two TCM encoder outputs. Figure 109(c) illustrates this. The 8-PSK constellation is shown in Figure 109(d). The $\{c_3, c_2, c_1\}$ assignments to the symbols, as dictated by Figure 109(b), are shown.

This TCM encoder/modulator has a bandwidth efficiency of 2 bits/sec/Hz.

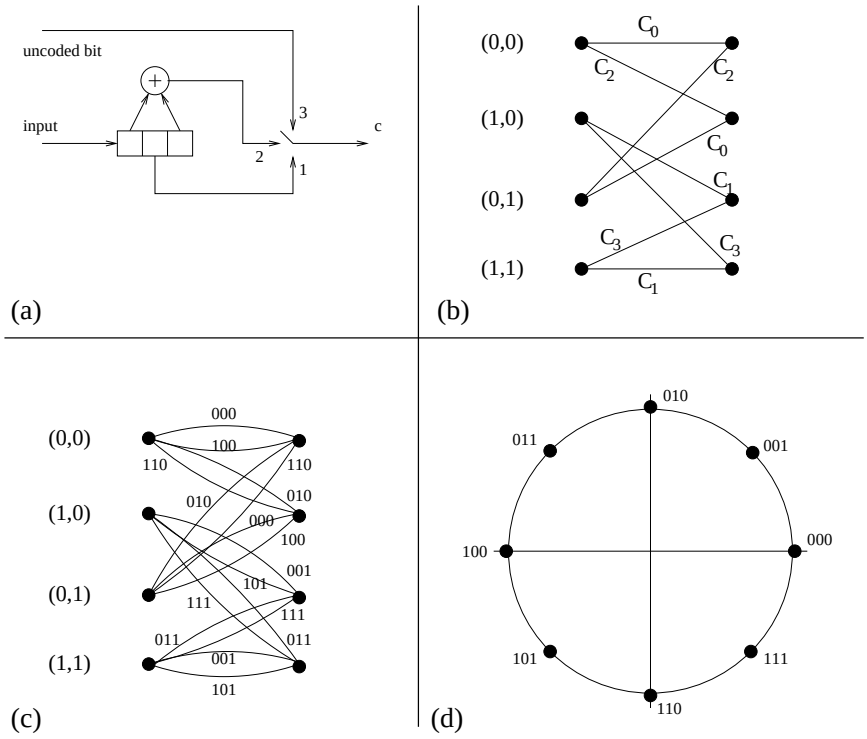


Figure 109: A $k_1 = 1$, $n = 2$, $k_2 = 1$ TCM encoder used with a 8-PSK modulator.

11.5 TCM Decoding and Performance

11.5.1 TCM Decoding

The Viterbi algorithm can be used to implement soft decision MLSE for TCM. Each path through the trellis corresponds to a sequence of partitioned sets. In general, each branch represents several trellis encoder output vectors, each with a corresponding assigned symbol.

The Viterbi algorithm operates optimally as it does for regular convolutional code decoding, pruning all but the best path into each trellis state. The one difference is in the computation of the branch costs. Since each branch represents several encoder output, the cost for each output is computed and the best output and cost is assigned to that branch. For example, if the modulation scheme is 8-PSK (a 2-dimensional scheme) and each branch represents two encoder outputs (i.e. two symbols) as in Figure 109(b), for each branch and stage the 2-dimensional sampled matched filter output (the observation) for that stage is compared to the signal space representation for the symbols associated with the branch, and the symbol is kept that is closest to the observation in Euclidean distance.

11.5.2 TCM Performance

We will illustrate the performance advantage associated with TCM with an example. Consider a specified 2 bit/symbol communication rate. We will compare performance of uncoded PSK with several TCM's which employ PSK. Coherent reception is assumed.

First, for reference, consider uncoded PSK. For the specified rate, 4-PSK gives us the specified 2 bit/symbol representation. Figure 110(a) shows the symbol constellation. We are

interested in the ratio

$$\frac{d^2_{min}}{E_s} \;=\; \frac{E_s + E_s}{E_s} \;=\; 2 \quad , \tag{25}$$

which is minimum distance normalized by symbol energy.

Second, consider 8-PSK modulation. The symbol constellation is shown in Figure 110(b). We have that

$$\frac{d^2_{min}}{E_s} \;=\; 4\sin^2(\pi/8) \quad . \tag{26}$$

The constellation is partitioned as in Figure 105. To achieve the specified rate, the TCM must map 2 bits to 8 symbols. So a rate $\frac{2}{3}$ TCM encoder is required. Figure 110(b) shows the 2 state encoder employed, along with the corresponding trellis and branch to partitioned set assignments. Note the we are unable to satisfy Rule 2 very well. As with the performance of a regular convolutional coder/decoder, free Euclidean distance d_{fed}, the minimum Euclidean distance between a trellis zero path segment and any path segment that diverges from and then merges to the all-zero path, will dominate performance as pointed out earlier in Section 7 of this Course. The highlighted path in the trellis in Figure 110(b) results in the d_{fed}. This leads to

$$\frac{d^2_{fed}}{E_s} \;=\; \frac{2E_s + 4\sin^2(\pi/8)E_s}{E_s} \;= 2 + 4\sin^2(\pi/8) \;=\; 2.586 \quad . \tag{27}$$

Compared to uncoded 4-PSK, the coding gain is

$$\gamma \;=\; \frac{2.586}{2} \;=\; 1.293 \;=\; 1.1dB \quad . \tag{28}$$

A little, but not much, is gained here.

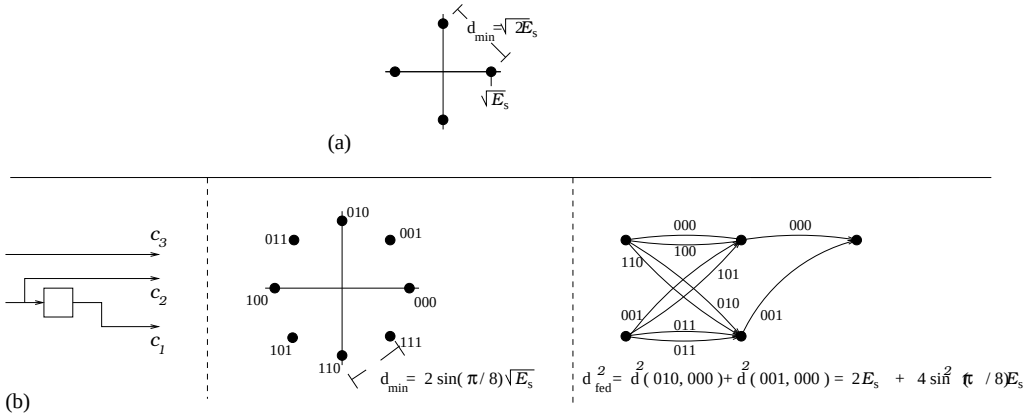


Figure 110: (a) encoded 4-PSK; (b) TCM with a 2 state encoder and 8-PSK.

Third, again consider 8-PSK modulation with

$$\frac{d^2_{min}}{E_s} \;=\; 4\sin^2(\pi/8) \quad . \tag{29}$$

But now consider the 4 state TCM encoder shown in Figure 109 with the corresponding trellis and branch to partitioned set assignments. Figure 111(a) shows the trellis path segment

corresponding to d_{fed}. Note that it is a parallel branch (between states (0,0) and (0,0)). From this, we have that

$$\frac{d_{fed}^2}{E_s} \;=\; \frac{4E_s}{E_s} \;= 4 \quad , \tag{30}$$

and the coding gain is

$$\gamma \;=\; \frac{4}{2} \;=\; 2 \;=\; 3dB \quad , \tag{31}$$

which is a more substantial gain.

Finally, again consider 8-PSK modulation with

$$\frac{d_{min}^2}{E_s} \;=\; 4\sin^2(\pi/8) \quad . \tag{32}$$

But now consider the 8 state TCM encoder shown in Figure 111(b) with the corresponding trellis and branch to partitioned set assignments. With this encoder, there is no unencoded bit (i.e. $k_2 = 0$). There are only $2^3 = 8$ states because the output of the last delay is not used. The rate $\frac{2}{3}$ encoder has 3 outputs which are used to select one of 8 1-bit partitioned "sets". The advantage here, compared to the previous TCM encoder, is that there will be no parallel transitions because there are no uncoded bits. Recall that for the previous encoder d_{fed} was due to a parallel transition. Figure 111(b) shows the corresponding trellis, the branch to partition assignments, and the path corresponding to d_{fed}. From this, we have that

$$\frac{d_{fed}^2}{E_s} \;=\; \frac{2E_s + 4\sin^2(\pi/8)E_s + 2E_s}{E_s} \;=\; 4.585 \quad , \tag{33}$$

and the coding gain is

$$\gamma \;=\; \frac{4.585}{2} \;=\; 2 \;=\; 3.6dB \quad , \tag{34}$$

which is again an improvement over the previous example.

References

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes (1)," *Proc. of International Conference on Communications*, pp. 1064–1070, 1993.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. on Info. Theory*, pp. 284–287, Mar. 1974.

[3] R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, "Near optimum product codes," *Globecom*, pp. 339–343, Nov. 1994.

[4] R. Gallagher, "Low-density parity check codes," *IRE Trans. Info. Theory*, pp. 21–28, 1962.

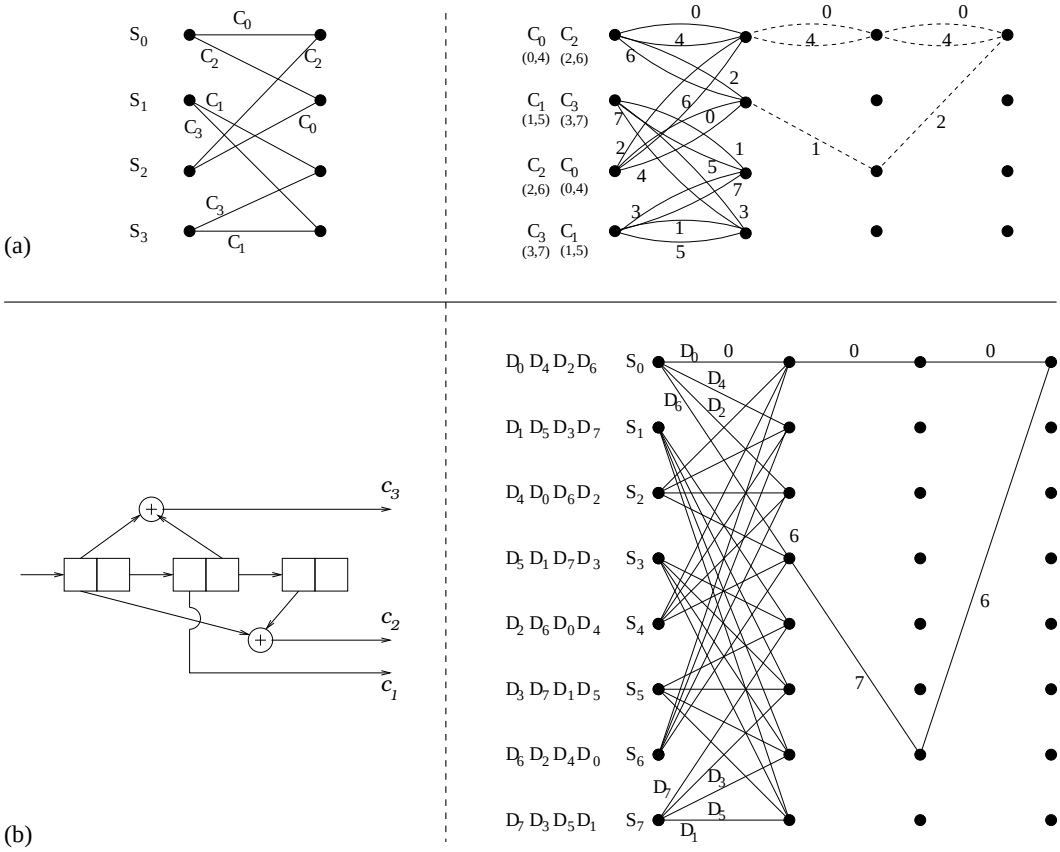


Figure 111: (a) TCM with a 4 state encoder and 8-PSK; (b) TCM with an 8 state encoder and 8-PSK.

[5] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," *Proc. of Globecom*, pp. 1680–1686, 1989.

[6] S. Lin and D. J. Costello, *Error Control Coding 2-nd ed.* Pearson Prentice Hall, 2004.

[7] J. Proakis and M. Salehi, *Digital Communications 5-th ed.* McGraw Hill, 2008.

[8] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Info. Theory*, vol. IR-27, pp. 533–547, Sept 1981.

[9] V.Tarokh, N.Seshadri, and A.R.Calderbank, "Space-time codes for high data rate wireless communications: Performance criterion and code construction," *IEEE Trans. Inform. Theory*, vol. 44, pp. 744–765, Mar. 1998.

[10] S.M.Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1451–1458, Oct 1998.

[11] V.Tarokh, H.Jafarkhani, and A.R.Calderbank, "Space-time block codes from orthogonal designs," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1456–1467, July 1999.

[12] R.A.Iltis, "A bayesian maximum likelihood sequence estimation algorithm for a priori unknown channels with symbol timing," *IEEE Jour. on Select. Areas in Comm.*, vol. 10, pp. 579–588, Apr 1992.

[13] H. Chen, R. Perry, and K. Buckley, "Direct and em based map sequence estimation with unknown time-varying channels," *ICASSP-2001*, April 2001.

[14] Z.Liu, X.Ma, and G.B.Giannakis, "Space-time coding and kalman filtering for time-selective fading channels," *IEEE Trans. on Communications*, vol. 50, pp. 183–186, Feb 2002.

[15] E.Lindskog and A.Paulraj, "A transmit diversity scheme for for channels with inter-symbol interference," *IEEE Intl Conf. on Commun.*, pp. 307–311, Jun. 2000.

[16] C. Wang, K. Buckley, and R. Perry, "Space-time block coding over unknown frequency and time selective channels," *CISS Conf.*, April 2003.

[17] G. Ungerboeck, "Channel coding with multilevel/phase signaling," *IEEE Trans. on Info. Theory*, pp. 55–67, Jan. 1982.