

Chapter 10

Trellis representations of binary linear block codes

We now return to binary linear block codes and discuss trellis representations, particularly minimal trellis representations. The three main reasons for doing so are:

- (a) Trellis-based (Viterbi algorithm) decoding is one of the most efficient methods known for maximum-likelihood (ML) decoding of general binary linear block codes;
- (b) The complexity of a minimal trellis gives a good measure of the complexity of a code, whereas the parameters (n, k, d) do not;
- (c) Trellis representations are the simplest class of graphical representations of codes, which will be a central concept in our later discussion of capacity-approaching codes.

The topic of trellis complexity of block codes was an active research area in the 1990s. We will summarize its main results. For an excellent general review, see [A. Vardy, “Trellis structure of codes,” in HANDBOOK OF CODING THEORY, Elsevier, 1998.]

10.1 Definition

We saw in the previous chapter that certain binary linear block codes could be represented as terminated convolutional codes, and therefore have trellis representations.

Example 1. (SPC codes) Any $(n, n - 1, 2)$ single-parity-check (SPC) code has a two-state trellis representation like that shown in Figure 1 (see Exercise 9.4).

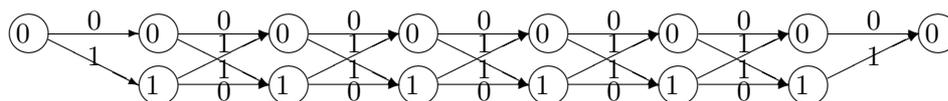


Figure 1. Two-state trellis for a binary $(n, n - 1, 2)$ single-parity-check code ($n = 7$).

In this chapter, we will show that all binary linear block codes have trellis (finite-state) representations. Indeed, we will show how to find minimal trellis representations.

In general, a trellis representation of a block code is a directed graph like that of Figure 1 in which there is one starting state (root) at time 0, one ending state (goal, “toor”) at time N , and state spaces \mathcal{S}_k of size greater than one at all $N - 1$ intermediate times k , $1 \leq k < N$. All edges (branches, state transitions) go from a state at some time k to a state at the next time $k + 1$. Each edge is labelled by an n -tuple of output symbols. The set of all possible codewords is in one-to-one correspondence with the set of all possible paths through the trellis. The codeword associated with any particular path is the sequence of corresponding n -tuple labels.

Example 2. ((8, 4, 4) RM code.) Figure 2 shows a trellis representation of an (8, 4, 4) binary Reed-Muller code with generators $\{11110000, 10101010, 11001100, 11111111\}$. Here two output symbols (bits) are associated with each branch. It can be seen that each of the 16 codewords in this code corresponds to a unique path through this trellis.

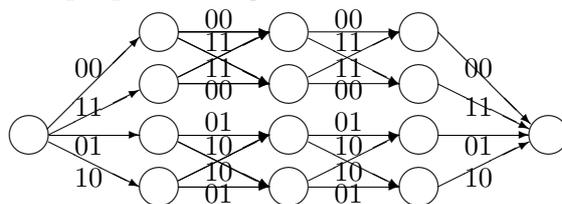


Figure 2. Four-state trellis for (8, 4, 4) Reed-Muller code.

Given a trellis representation of a block code and a sequence of received symbol log likelihoods, the Viterbi algorithm (VA) may be used to find the trellis path with greatest log likelihood—*i.e.*, to perform ML decoding of the code.

There are various measures of the complexity of a trellis, which are all related to VA decoding complexity. The most common is perhaps the *state complexity profile*, which is just the sequence of state space sizes. For example, the state complexity profile of the trellis of Figure 1 is $\{1, 2, 2, 2, 2, 2, 2, 1\}$, and that of the trellis of Figure 2 is $\{1, 4, 4, 4, 1\}$. The *state complexity* of a trellis is often defined as the maximum state space size; *e.g.*, 2 or 4, respectively, for these two trellises. The *branch complexity profile* is the sequence of numbers of branches in each section—*e.g.*, $\{2, 4, 4, 4, 4, 4, 2\}$ and $\{4, 8, 8, 4\}$ for the trellises of Figures 1 and 2, respectively—and the *branch complexity* is the maximum number of branches in any section—*e.g.*, 4 or 8, respectively. The complexity of VA decoding is more precisely measured by the branch complexity profile, but any of these measures gives a good idea of VA decoding complexity.¹

10.2 Minimal trellises and the state space theorem

A *minimal trellis* for a given linear block code will be defined as a trellis that has the minimum possible state space size at each time. It is not immediately clear that there exists a single trellis that minimizes every state space size, but for linear block codes, we will see that there exists a canonical minimal trellis with this property. It turns out that the canonical minimal trellis also minimizes every other trellis complexity measure, including VA decoding complexity, so that we need not worry whether we are minimizing the right quantity.

¹The most precise measure of VA decoding complexity is $2|E| - |V| + 1$, where $|E|$ and $|V|$ denote the numbers of edges and vertices in the graph of the trellis, respectively.

10.2.1 The state space theorem for linear block codes

The state space theorem is an easy but fundamental theorem that sets a lower bound on the state space size of a trellis for a linear block code at any time, and also indicates how to construct a canonical minimal trellis. For simplicity, we will prove it here only for binary linear block codes, but it should be clear how it generalizes to arbitrary linear codes. (In fact, it generalizes to arbitrary codes over groups.)

The reader should think of a trellis as defining a state-space realization of a time-varying finite-state discrete-time linear system, where these terms are used as in system theory. The time axis \mathcal{I} of the system is some subinterval of the integers \mathbb{Z} ; e.g., a finite subinterval $\mathcal{I} = [0, N] \subseteq \mathbb{Z}$.

In a state-space realization, a state space \mathcal{S}_k is defined at each time $k \in \mathcal{I}$. The defining property of a state space is the *Markov property*:

Markov property. The state space \mathcal{S}_k of a system at time k has the Markov property if, given that the system is in a certain state $s_k \in \mathcal{S}_k$ at time k , its possible future trajectories depend only on s_k and not otherwise on the previous history of the system. In other words, the state $s_k \in \mathcal{S}_k$ is a sufficient statistic for the past with respect to prediction of possible futures.

Let the **past** at time k be the set $\mathcal{P} = (-\infty, k) \cap \mathcal{I}$ of time indices in \mathcal{I} prior to time k , and the **future** at time k be the set $\mathcal{F} = [k, \infty) \cap \mathcal{I}$ of time indices at time k or later. Given a linear code \mathcal{C} , the state space theorem may be expressed in terms of certain linear codes defined on the past \mathcal{P} and future \mathcal{F} , as follows.

Given any subset $\mathcal{J} \subseteq \mathcal{I}$, the **subcode** $\mathcal{C}_{\mathcal{J}}$ is defined as the set of codewords whose components are equal to 0 on the complement of \mathcal{J} in \mathcal{I} . It is possible to think of $\mathcal{C}_{\mathcal{J}}$ either as a code of length $|\mathcal{J}|$ or as a code of length $|\mathcal{I}|$ in which all symbols not in \mathcal{J} equal 0. It should be clear from the context which of these two viewpoints is used.

The subcode $\mathcal{C}_{\mathcal{J}}$ is evidently a subset of the codewords in \mathcal{C} that has the group property, and therefore is a linear code. In coding theory, $\mathcal{C}_{\mathcal{J}}$ is called a *shortened code* of \mathcal{C} . Because $\mathcal{C}_{\mathcal{J}} \subseteq \mathcal{C}$, the minimum Hamming distance of $\mathcal{C}_{\mathcal{J}}$ is at least as great as that of \mathcal{C} .

Similarly, given $\mathcal{J} \subseteq \mathcal{I}$, the **projection** $\mathcal{C}_{|\mathcal{J}}$ is defined as the set of all projections of codewords of \mathcal{C} onto \mathcal{J} . By projection, we mean either zeroing of all coordinates whose indices are not in \mathcal{J} , or throwing away (puncturing) all such coordinates. Correspondingly, it is possible to think of $\mathcal{C}_{|\mathcal{J}}$ either as a code of length $|\mathcal{J}|$ or as a code of length $|\mathcal{I}|$ in which all symbols not in \mathcal{J} equal 0. In coding theory, $\mathcal{C}_{|\mathcal{J}}$ is called a *punctured code* of \mathcal{C} .

The projection $\mathcal{C}_{|\mathcal{J}}$ evidently inherits the group property from \mathcal{C} , and therefore is also a linear code defined on \mathcal{J} . Moreover, $\mathcal{C}_{\mathcal{J}}$ is evidently a subcode of $\mathcal{C}_{|\mathcal{J}}$.

Example 2 (cont.) For the $(8, 4, 4)$ code illustrated in Figure 2, regarding the “past” as the first two time units or first four bits, the subcode $\mathcal{C}_{\mathcal{P}}$ consists of the two codewords $\mathcal{C}_{\mathcal{P}} = \{00000000, 11110000\}$. This code may be regarded as effectively a $(4, 1, 4)$ binary repetition code defined on the past subinterval $\mathcal{P} = [0, 1, 2, 3]$. The projection on this subinterval is the set $\mathcal{C}_{|\mathcal{P}} = \{0000, 0011, 1100, 1111, 0101, 0110, 1001, 1010\}$, which is a $(4, 3, 2)$ binary linear SPC code that has the $(4, 1, 4)$ code as a subcode. \square

Now for any time $k \in \mathcal{I}$, let \mathcal{P} and \mathcal{F} denote the past and future subintervals with respect to k , and let $\mathcal{C}_{\mathcal{P}}, \mathcal{C}_{|\mathcal{P}}, \mathcal{C}_{\mathcal{F}}$ and $\mathcal{C}_{|\mathcal{F}}$ be the past and future subcodes and projections, respectively.

Then \mathcal{C} must have a generator matrix of the following form:

$$\begin{bmatrix} G(\mathcal{C}_{\mathcal{P}}) & 0 \\ 0 & G(\mathcal{C}_{\mathcal{F}}) \\ G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_{\mathcal{P}}) & G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_{\mathcal{F}}) \end{bmatrix},$$

where $[G(\mathcal{C}_{\mathcal{P}}), 0]$ is a generator matrix for the past subcode $\mathcal{C}_{\mathcal{P}}$, $[0, G(\mathcal{C}_{\mathcal{F}})]$ is a generator matrix for the future subcode $\mathcal{C}_{\mathcal{F}}$, and $[G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_{\mathcal{P}}), G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_{\mathcal{F}})]$ is an additional set of linearly independent generators that together with $[G(\mathcal{C}_{\mathcal{P}}), 0]$ and $[0, G(\mathcal{C}_{\mathcal{F}})]$ generate \mathcal{C} . Moreover, it is clear from the form of this generator matrix that $G(\mathcal{C}_{\mathcal{P}})$ and $G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_{\mathcal{P}})$ together generate the past projection $\mathcal{C}_{|\mathcal{P}}$, and that $G(\mathcal{C}_{\mathcal{F}})$ and $G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_{\mathcal{F}})$ generate the future projection $\mathcal{C}_{|\mathcal{F}}$.

The **state code** \mathcal{S} will be defined as the linear code generated by this last set of generators, $[G(\mathcal{C}_{|\mathcal{P}}/\mathcal{C}_{\mathcal{P}}), G(\mathcal{C}_{|\mathcal{F}}/\mathcal{C}_{\mathcal{F}})]$. The dimension of the state code is evidently

$$\dim \mathcal{S} = \dim \mathcal{C} - \dim \mathcal{C}_{\mathcal{P}} - \dim \mathcal{C}_{\mathcal{F}}.$$

Moreover, by the definitions of $\mathcal{C}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{F}}$, the state code cannot contain any codewords that are all-zero on the past or on the future. This implies that the projections $\mathcal{S}_{|\mathcal{P}}$ and $\mathcal{S}_{|\mathcal{F}}$ on the past and future are linear codes with the same dimension as \mathcal{S} , so we also have

$$\begin{aligned} \dim \mathcal{S} &= \dim \mathcal{S}_{|\mathcal{P}} = \dim \mathcal{C}_{|\mathcal{P}} - \dim \mathcal{C}_{\mathcal{P}}; \\ \dim \mathcal{S} &= \dim \mathcal{S}_{|\mathcal{F}} = \dim \mathcal{C}_{|\mathcal{F}} - \dim \mathcal{C}_{\mathcal{F}}. \end{aligned}$$

Example 2 (cont.) Taking \mathcal{P} as the first four symbols and \mathcal{F} as the last four, the $(8, 4, 4)$ code has a generator matrix

$$\begin{bmatrix} 1111 & 0000 \\ 0000 & 1111 \\ 1010 & 1010 \\ 1100 & 1100 \end{bmatrix}.$$

Here the state code \mathcal{S} is generated by the last two generators and has dimension 2. The past and future projections of the state code evidently also have dimension 2. \square

In view of the generator matrix above, any codeword $\mathbf{c} \in \mathcal{C}$ may be expressed uniquely as the sum of a past codeword $\mathbf{c}_{\mathcal{P}} \in \mathcal{C}_{\mathcal{P}}$, a future codeword $\mathbf{c}_{\mathcal{F}} \in \mathcal{C}_{\mathcal{F}}$, and a state codeword $\mathbf{s} \in \mathcal{S}$. We then say that the codeword \mathbf{c} is associated with the state codeword \mathbf{s} . This allows us to conclude that the state codewords have the Markov property, and thus may be taken as states:

Lemma 10.1 (Markov property) *For all codewords $\mathbf{c} \in \mathcal{C}$ that are associated with a given state codeword $\mathbf{s} \in \mathcal{S}$, the past projection $\mathbf{c}_{|\mathcal{P}}$ has the same set of possible future trajectories. On the other hand, if two codewords \mathbf{c} and \mathbf{c}' are associated with different state codewords, then the sets of possible future trajectories of $\mathbf{c}_{|\mathcal{P}}$ and $\mathbf{c}'_{|\mathcal{P}}$ are disjoint.*

Proof. If \mathbf{c} is associated with \mathbf{s} , then $\mathbf{c} = \mathbf{c}_{\mathcal{P}} + \mathbf{c}_{\mathcal{F}} + \mathbf{s}$ for some $\mathbf{c}_{\mathcal{P}} \in \mathcal{C}_{\mathcal{P}}$ and $\mathbf{c}_{\mathcal{F}} \in \mathcal{C}_{\mathcal{F}}$. Hence $\mathbf{c}_{|\mathcal{P}} = (\mathbf{c}_{\mathcal{P}})_{|\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}$ and $\mathbf{c}_{|\mathcal{F}} = (\mathbf{c}_{\mathcal{F}})_{|\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$. Thus, for every such $\mathbf{c}_{|\mathcal{P}}$, the set of possible future trajectories is the same, namely the coset $\mathcal{C}_{\mathcal{F}} + \mathbf{s}_{|\mathcal{F}} = \{(\mathbf{c}_{\mathcal{F}})_{|\mathcal{F}} + \mathbf{s}_{|\mathcal{F}} \mid \mathbf{c}_{\mathcal{F}} \in \mathcal{C}_{\mathcal{F}}\}$.

If \mathbf{c} and \mathbf{c}' are associated with different state codewords \mathbf{s} and \mathbf{s}' , then the sets of possible future trajectories are $\mathcal{C}_{\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$ and $\mathcal{C}_{\mathcal{F}} + \mathbf{s}'_{|\mathcal{F}}$, respectively; these cosets are disjoint because the difference $\mathbf{s}_{|\mathcal{F}} - \mathbf{s}'_{|\mathcal{F}}$ is not a codeword in $\mathcal{C}_{\mathcal{F}}$. \square

In other words, a codeword $\mathbf{c} \in \mathcal{C}$ is in the coset $\mathcal{C}_{\mathcal{P}} + \mathcal{C}_{\mathcal{F}} + \mathbf{s}$ if and only if $\mathbf{c}_{|\mathcal{P}} \in \mathcal{C}_{|\mathcal{P}}$ is in the coset $\mathcal{C}_{\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}$ and $\mathbf{c}_{|\mathcal{F}} \in \mathcal{C}_{|\mathcal{F}}$ is in the coset $\mathcal{C}_{\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$.

This yields the following picture. The set of past projections associated with a given state codeword \mathbf{s} is $\mathcal{C}_{\mathcal{P}} + \mathbf{s}_{|\mathcal{P}} = \{(\mathbf{c}_{\mathcal{P}})_{|\mathcal{P}} + \mathbf{s}_{|\mathcal{P}} \mid \mathbf{c}_{\mathcal{P}} \in \mathcal{C}_{\mathcal{P}}\}$. For any past projection in this set, we have the same set of possible future trajectories, namely $\mathcal{C}_{\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$. Moreover, these past and future subsets are disjoint. Therefore the entire code may be written as a disjoint union of Cartesian products of past and future subsets:

$$\mathcal{C} = \bigcup_{\mathbf{s} \in \mathcal{S}} (\mathcal{C}_{\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}) \times (\mathcal{C}_{\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}).$$

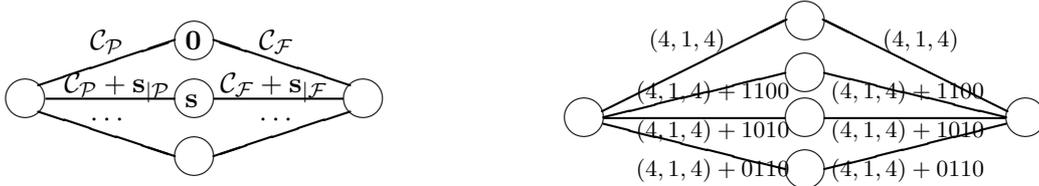


Figure 3. (a) Two-section trellis for generic code; (b) two-section trellis for (8, 4, 4) code.

Figure 3(a) is a two-section trellis that illustrates this Cartesian-product decomposition for a general code. The states are labelled by the state codewords $\mathbf{s} \in \mathcal{S}$ to which they correspond. Each edge represents an entire set of parallel projections, namely $\mathcal{C}_{\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}$ for past edges and $\mathcal{C}_{\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$ for future edges, and is therefore drawn as a thick line. The particular path corresponding to $\mathbf{s} = \mathbf{0}$ is shown, representing the subcode $\mathcal{C}_{\mathcal{P}} + \mathcal{C}_{\mathcal{F}}$, as well as a generic path corresponding to a general state \mathbf{s} , representing $\mathcal{C}_{\mathcal{P}} + \mathcal{C}_{\mathcal{F}} + \mathbf{s}$.

Example 2 (cont.) Figure 3(b) is a similar illustration for our running example (8, 4, 4) code. Here $\mathcal{C}_{\mathcal{P}} = \mathcal{C}_{\mathcal{F}} = (4, 1, 4) = \{0000, 1111\}$, so each edge represents two paths that are binary complements of one another. The reader should compare Figure 3(b) to Figure 2 and verify that these two trellises represent the same code. \square

It should now be clear that the code \mathcal{C} has a trellis representation with a state space \mathcal{S}_k at time k that is in one-to-one correspondence with the state code \mathcal{S} , and it has no trellis representation with fewer states at time k . Indeed, Figure 3(a) exhibits a two-section trellis with a state space \mathcal{S}_k such that $|\mathcal{S}_k| = |\mathcal{S}|$. The converse is proved by noting that no two past projections $\mathbf{c}_{|\mathcal{P}}, \mathbf{c}'_{|\mathcal{P}}$ that do not go to the same state in Figure 3(a) can go to the same state in any trellis for \mathcal{C} , because by Lemma 10.1 they must have different sets of future continuations.

We summarize this development in the following theorem:

Theorem 10.2 (State space theorem for binary linear block codes) *Let \mathcal{C} be a binary linear block code defined on a time axis $\mathcal{I} = [0, N]$, let $k \in \mathcal{I}$, and let $\mathcal{C}_{\mathcal{P}}, \mathcal{C}_{|\mathcal{P}}, \mathcal{C}_{\mathcal{F}}$ and $\mathcal{C}_{|\mathcal{F}}$ be the past and future subcodes and projections relative to time k , respectively. Then there exists a trellis representation for \mathcal{C} with $|\mathcal{S}_k| = 2^{(\dim \mathcal{S})}$ states at time k , and no trellis representation with fewer states, where $\dim \mathcal{S}$ is given by any of the following three expressions:*

$$\begin{aligned} \dim \mathcal{S} &= \dim \mathcal{C} - \dim \mathcal{C}_{\mathcal{P}} - \dim \mathcal{C}_{\mathcal{F}}; \\ &= \dim \mathcal{C}_{|\mathcal{P}} - \dim \mathcal{C}_{\mathcal{P}}; \\ &= \dim \mathcal{C}_{|\mathcal{F}} - \dim \mathcal{C}_{\mathcal{F}}. \quad \square \end{aligned}$$

We note that if we subtract the first expression for $\dim \mathcal{S}$ from the sum of the other two, then we obtain yet another expression:

$$\dim \mathcal{S} = \dim \mathcal{C}_{|\mathcal{P}} + \dim \mathcal{C}_{|\mathcal{F}} - \dim \mathcal{C}.$$

Exercise 1. Recall the $|u|u+v|$ construction of a Reed-Muller code $\text{RM}(r, m)$ with length $n = 2^m$ and minimum distance $d = 2^{m-r}$:

$$\text{RM}(r, m) = \{(\mathbf{u}, \mathbf{u} + \mathbf{v}) \mid \mathbf{u} \in \text{RM}(r, m-1), \mathbf{v} \in \text{RM}(r-1, m-1)\}.$$

Show that if the past \mathcal{P} is taken as the first half of the time axis and the future \mathcal{F} as the second half, then the subcodes $\mathcal{C}_{\mathcal{P}}$ and $\mathcal{C}_{\mathcal{F}}$ are both effectively equal to $\text{RM}(r-1, m-1)$ (which has the same minimum distance $d = 2^{m-r}$ as $\text{RM}(r, m)$), while the projections $\mathcal{C}_{|\mathcal{P}}$ and $\mathcal{C}_{|\mathcal{F}}$ are both equal to $\text{RM}(r, m-1)$. Conclude that the dimension of the minimal central state space of $\text{RM}(r, m)$ is

$$\dim \mathcal{S} = \dim \text{RM}(r, m-1) - \dim \text{RM}(r-1, m-1).$$

Evaluate $\dim \mathcal{S}$ for all RM codes with length $n \leq 32$.

Similarly, show that if the past \mathcal{P} is taken as the first quarter of the time axis and the future \mathcal{F} as the remaining three quarters, then the subcode $\mathcal{C}_{\mathcal{P}}$ is effectively equal to $\text{RM}(r-2, m-2)$, while the projection $\mathcal{C}_{|\mathcal{P}}$ is equal to $\text{RM}(r, m-2)$. Conclude that the dimension of the corresponding minimal state space of $\text{RM}(r, m)$ is

$$\dim \mathcal{S} = \dim \text{RM}(r, m-2) - \dim \text{RM}(r-2, m-2).$$

Using the relation $\dim \text{RM}(r, m) = \dim \text{RM}(r, m-1) + \dim \text{RM}(r-1, m-1)$, show that $\dim \text{RM}(r, m-2) - \dim \text{RM}(r-2, m-2) = \dim \text{RM}(r, m-1) - \dim \text{RM}(r-1, m-1)$. \square

Exercise 2. Recall that the dual code to a binary linear (n, k, d) block code \mathcal{C} is defined as the orthogonal subspace \mathcal{C}^\perp , namely the set of all n -tuples that are orthogonal to all codewords in \mathcal{C} , and that \mathcal{C}^\perp is a binary linear block code whose dimension is $\dim \mathcal{C}^\perp = n - k$.

Show that for any partition of the time axis \mathcal{I} of \mathcal{C} into past \mathcal{P} and future \mathcal{F} , the subcode $(\mathcal{C}^\perp)_{\mathcal{P}}$ is equal to the dual $(\mathcal{C}_{|\mathcal{P}})^\perp$ of the projection $\mathcal{C}_{|\mathcal{P}}$, and *vice versa*. [Hint: notice that $(\mathbf{a}, \mathbf{0})$ is orthogonal to (\mathbf{b}, \mathbf{c}) if and only if \mathbf{a} is orthogonal to \mathbf{b} .]

Conclude that the minimal state spaces of \mathcal{C} and \mathcal{C}^\perp at any time k have the same size. \square

10.2.2 Canonical minimal trellis representation

We now show that we can construct a single trellis for \mathcal{C} such that the state space is minimal according to the state space theorem at each time k .

The construction is straightforwardly based on the development of the previous subsection. We define a state space \mathcal{S}_k at each time $k \in \mathcal{I} = [0, N]$ corresponding to the state code \mathcal{S} at time k . Note that \mathcal{S}_0 is trivial, because $\mathcal{C}_{|\mathcal{F}} = \mathcal{C}_{\mathcal{F}} = \mathcal{C}$, so $\dim \mathcal{S}_0 = 0$; similarly $\dim \mathcal{S}_N = 0$.

Every codeword $\mathbf{c} \in \mathcal{C}$ passes through a definite state in \mathcal{S}_k corresponding to the state codeword $\mathbf{s} \in \mathcal{S}$ in the unique decomposition $\mathbf{c} = \mathbf{c}_{\mathcal{P}} + \mathbf{c}_{\mathcal{F}} + \mathbf{s}$. It thus goes through a well-defined sequence of states (s_0, s_1, \dots, s_N) , which defines a certain path through the trellis. Each edge in each such path is added to the trellis. The result must be a trellis representation of \mathcal{C} .

Example 1 (cont.) We can now see that the two-state trellis of Figure 1 is a canonical minimal trellis. For each time $k, 0 < k < N$, $\mathcal{C}_{\mathcal{P}}$ is the set of all even-weight codewords that are all-zero on \mathcal{F} , which is effectively a $(k, k-1, 2)$ SPC code, and $\mathcal{C}_{|\mathcal{P}}$ is the universe $(k, k, 1)$ code consisting of all binary k -tuples. There are thus two states at each time $k, 0 < k < N$, one reached by all even-weight pasts, and the other by all odd-weight pasts. A given codeword passes through a well-defined sequence of states, and makes a transition from the zero (even) to the nonzero (odd) state or *vice versa* whenever it has a symbol equal to 1. \square

Example 2 (cont.) Similarly, the four-state trellis of Figure 2 is a canonical minimal trellis. For $k = 1$, $\mathcal{C}_{\mathcal{P}}$ is the trivial $(2, 0, \infty)$ code, so each of the four past projections at time $k = 1$ leads to a different state. The same is true at time $k = 3$ for future projections. Each of the 16 codewords goes through a well-defined state sequence $(s_0, s_1, s_2, s_3, s_4)$, and the set of all these sequences defines the trellis.

10.2.3 Trellis-oriented generator matrices

It is very convenient to have a generator matrix for \mathcal{C} from which a minimal trellis for \mathcal{C} and its parameters can be read directly. In this subsection we define such a generator matrix, show how to find it, and give some of its properties.

The *span* of a codeword will be defined as the interval from its first to last nonzero symbols. Its *effective length* will be defined as the length of this span.

A *trellis-oriented* (or *minimum-span*) generator matrix will be defined as a set of $k = \dim \mathcal{C}$ linearly independent generators whose effective lengths are as short as possible.

Concretely, a trellis-oriented generator matrix may be found by first finding all codewords with effective length 1, then all codewords of effective length 2 that are not linearly dependent on codewords of effective length 1, \dots , all codewords of effective length i that are not linearly dependent on codewords of lower effective length, \dots , until we have k independent generators.

The following theorem shows how to check whether a given generator matrix is trellis-oriented, and also suggests how to reduce any given generator matrix to one that is trellis-oriented.

Theorem 10.3 (Trellis-oriented generator matrices) *A set of k linearly independent generators is a trellis-oriented generator matrix if and only if the starting times of all spans are distinct and the ending times of all spans are distinct.*

Proof. If all starting times and ending times are distinct, then given a linear combination (with nonzero coefficients) of a certain subset of generators, the starting time and ending time of the combination are the least and greatest starting and ending times of the given subset of generators. It follows that the generators that combine to form any non-generator codeword have effective lengths no greater than the effective length of that codeword, so the given generators are indeed a set of generators whose effective lengths are as short as possible.

Conversely, if two starting or ending times are not distinct, then the sum of the two corresponding generators is a codeword whose effective length is shorter than that of at least one of the two generators. If this generator is replaced by this codeword, then we obtain a set of linearly independent generators of which one has a shorter effective length, so the original set was not trellis-oriented. \square

The second part of the proof suggests a simple greedy algorithm for finding a trellis-oriented generator matrix from a given generator matrix. If the starting or ending times of two generators in the given matrix are not distinct, then replace the generator with greater effective length by the sum of the two generators, which must necessarily have a shorter effective length. This algorithm reduces the aggregate effective length in each step, and therefore must terminate after a finite number of steps in a trellis-oriented generator matrix.

Example 2 (cont.) The standard generator matrix for the $(8, 4, 4)$ RM code is as follows:

$$\begin{bmatrix} 1111 & 0000 \\ 1010 & 1010 \\ 1100 & 1100 \\ 1111 & 1111 \end{bmatrix}.$$

The ending times are all distinct, but the starting times are all the same. Adding the first generator to all others results in

$$\begin{bmatrix} 1111 & 0000 \\ 0101 & 1010 \\ 0011 & 1100 \\ 0000 & 1111 \end{bmatrix}.$$

All starting and ending times are now distinct, so this generator matrix is trellis-oriented. \square

Exercise 3. Consider the following generator matrix for the $(16, 5, 8)$ RM code, which follows directly from the $|u|u + v|$ construction:

$$\begin{bmatrix} 1111111100000000 \\ 1111000011110000 \\ 1100110011001100 \\ 1010101010101010 \\ 1111111111111111 \end{bmatrix}.$$

Convert this generator matrix to a trellis-oriented generator matrix. \square

Exercise 4 (minimum-span generators for convolutional codes).

(a) Let \mathcal{C} be a rate- $1/n$ binary linear convolutional code generated by a rational n -tuple $\mathbf{g}(D)$, and let $\mathbf{g}'(D)$ be the canonical polynomial n -tuple that generates \mathcal{C} . Show that the generators $\{D^k \mathbf{g}'(D), k \in \mathbb{Z}\}$ are a set of minimum-span generators for \mathcal{C} .

(b) Show that the greedy algorithm of Section 9.2.4 chooses a set of minimum-span generators for a rate- k/n binary linear convolutional code. \square

The key property of a trellis-oriented generator matrix, used in the first part of the proof of Theorem 10.3, is that the starting and ending times of a linear combination (with nonzero coefficients) of a subset of generators are the earliest and latest starting times of the component generators, respectively. We state this important observation as a lemma:

Lemma 10.4 (Generators for subcodes) *Given a trellis-oriented generator matrix for a linear code \mathcal{C} , if $[k, k'] \subseteq \mathcal{I}$ is any subinterval of the time axis \mathcal{I} , then the subcode $\mathcal{C}_{[k, k']}$ is the set of all linear combinations of generators whose spans are contained in $[k, k']$.*

Thus the dimensions of each past and future subcode may be read directly from a trellis-oriented generator matrix. Moreover, for any partition into past and future, the state subcode \mathcal{S} is generated by those generators which lie neither wholly in the past nor wholly in the future. The dimension of the minimal state space is the number of such *active* generators.

Example 1 (cont.) A trellis-oriented generator matrix for the $(7, 6, 2)$ SPC code of Figure 1 is

$$\begin{bmatrix} 1100000 \\ 0110000 \\ 0011000 \\ 0001100 \\ 0000110 \\ 0000011 \end{bmatrix}.$$

At each cut time k , only one generator is active, so each state space \mathcal{S}_k has dimension 1. \square

Example 2 (cont.) For the $(8, 4, 4)$ code, we constructed the following trellis-oriented generator matrix:

$$\begin{bmatrix} 1111 & 0000 \\ 0101 & 1010 \\ 0011 & 1100 \\ 0000 & 1111 \end{bmatrix}.$$

There are two active generators at each of the three cut times corresponding to a nontrivial state space in the trellis of Figure 2, so the four-state trellis of Figure 2 is minimal.

Notice from this matrix that the complete state complexity profile of a minimal 8-section trellis for this code is as follows: $\{1, 2, 4, 8, 4, 8, 4, 2, 1\}$. The maximum state complexity is 8, so a 4-section trellis somewhat masks the state complexity of a full minimal trellis. \square

Exercise 3 (cont.). For the $(16, 5, 8)$ code given earlier, determine the state complexity profile of a minimal trellis. \square

Exercise 5. (Trellis complexity of MDS codes, and the Wolf bound)

Let \mathcal{C} be a linear $(n, k, d = n - k + 1)$ MDS code over a finite field \mathbb{F}_q . Using the property that in an MDS code there exist $q - 1$ weight- d codewords with support \mathcal{J} for every subset $\mathcal{J} \subseteq \mathcal{I}$ of size $|\mathcal{J}| = d$, show that a trellis-oriented generator matrix for \mathcal{C} must have the following form:

$$\begin{bmatrix} xxxx0000 \\ 0xxxx000 \\ 00xxxx00 \\ 000xxxx0 \\ 0000xxxx \end{bmatrix},$$

where $xxxx$ denotes a span of length $d = n - k + 1$, which shifts right by one position for each of the k generators (*i.e.*, from the interval $[1, n - k + 1]$ to $[k, n]$).

For example, show that binary linear $(n, n - 1, 2)$ and $(n, 1, n)$ block codes have trellis-oriented generator matrices of this form.

Conclude that the state complexity profile of any $(n, k, d = n - k + 1)$ MDS code is

$$\{1, q, q^2, \dots, |\mathcal{S}|_{\max}, |\mathcal{S}|_{\max}, \dots, q^2, q, 1\},$$

where $|\mathcal{S}|_{\max} = q^{\min(k, n-k)}$.

Using the state space theorem and Exercise 2, show that this is the worst possible state complexity profile for a (n, k) linear code over \mathbb{F}_q . This is called the Wolf bound. \square

10.2.4 Branch complexity

Most of the work on trellis complexity has focussed on state complexity. However, branch complexity is in some respects more fundamental. It is a better measure of Viterbi algorithm decoding complexity. Also, as we shall see, it cannot be reduced by sectionalization.

The time axis for branches is not the same as the time axis $\mathcal{I} = [0, N]$ for states. Branches occur *at* symbol times, whereas states occur *between* symbol times. Thus there are only N branch times, say $[0, N)$, whereas there are $N + 1$ state times.

A branch at time k may be identified by a triple (s_k, c_k, s_{k+1}) , where (s_k, s_{k+1}) is a valid state transition, and c_k is a valid code symbol that may be generated during that transition. Thus there may be more than one branch (*parallel transition*) associated with a given state transition, if there is more than one output possible during that transition. The branch space at time k is the set of all possible branches, $\mathcal{B}_k = \{(s_k, c_k, s_{k+1})\}$.

Theorem 10.5 (Branch space theorem) *Let \mathcal{C} be a binary linear block code defined on a time axis $\mathcal{I} = [0, N]$. Then in any minimal trellis for \mathcal{C} , for any $k \in \mathcal{I}$, the branch space $\mathcal{B}_k = \{(s_k, c_k, s_{k+1})\}$ is a linear vector space with dimension*

$$\dim \mathcal{B}_k = \dim \mathcal{C} - \dim \mathcal{C}_{[0,k)} - \dim \mathcal{C}_{[k+1,N)}.$$

where $\mathcal{C}_{[0,k)}$ and $\mathcal{C}_{[k+1,N)}$ are the subcodes defined on $[0, k)$ and $[k + 1, N)$, respectively.

Proof In a minimal trellis, the state spaces \mathcal{S}_k and \mathcal{S}_{k+1} are linear vector spaces of minimum dimension. The set of all codewords that pass through a given branch (s_k, c_k, s_{k+1}) is the set that have a past projection $\mathbf{c}_{|\mathcal{P}} = (\mathbf{c}_{\mathcal{P}})_{|\mathcal{P}} + \mathbf{s}_{|\mathcal{P}}$ consistent with the state s_k associated with $\mathbf{s}_{|\mathcal{P}}$, a projection $\mathbf{c}_{|\{k\}}$ at time k equal to c_k , and a future projection (with respect to time $k + 1$) $\mathbf{c}_{|\mathcal{F}} = (\mathbf{c}_{\mathcal{F}})_{|\mathcal{F}} + \mathbf{s}_{|\mathcal{F}}$ consistent with the state s_{k+1} associated with $\mathbf{s}_{|\mathcal{F}}$. Thus two codewords go through the same branch at time k if and only if they differ only by an element of the past subcode $\mathcal{C}_{[0,k)}$ and/or an element of the future subcode $\mathcal{C}_{[k+1,N)}$. It follows that the branch space is a linear vector space with dimension $\dim \mathcal{B}_k = \dim \mathcal{C} - \dim \mathcal{C}_{[0,k)} - \dim \mathcal{C}_{[k+1,N)}$. \square

Since $\dim \mathcal{C}_{[0,k)}$ is equal to the number of generators in a trellis-oriented generator matrix whose span lies in $[0, k)$, and $\dim \mathcal{C}_{[k+1,N)}$ is the number that lie in $[k + 1, N)$, we can read the branch complexity profile by inspection from a trellis-oriented generator matrix. In other words, $\dim \mathcal{B}_k$ is equal to the number of trellis-oriented generators that are active at symbol time k .

Example 2 (cont.) For the $(8, 4, 4)$ code, the trellis-oriented generator matrix given above shows that the branch complexity profile of a minimal 8-section trellis is as follows: $\{2, 4, 8, 8, 8, 8, 4, 2\}$. The maximum branch complexity is 8, which here equals the maximum state complexity. \square

Exercise 3 (cont.) Find the branch complexity profile of a minimal trellis for the $(16, 5, 8)$ code.

10.2.5 Average dimension bounds, and asymptotics

Each generator in a trellis-oriented generator matrix contributes one dimension to the state and branch spaces during the time that it is active; *i.e.*, between its start and its end. More precisely, if its span has length L , then it contributes to $L - 1$ state spaces and L branch spaces.

The one-dimensional code generated by each generator may in fact be realized by a small time-varying state machine that has a state space of size 1 when it is inactive and of size 2 when it is active, illustrated for the generator 11110000 in Figure 4.

In this section we address these two problems. Permutations can make a big difference in trellis complexity, but finding the optimum permutation is intractable (NP-hard). Nonetheless, a few results are known. Sectionalization typically makes little difference in trellis complexity, but optimum sectionalization is fairly easy.

10.3.1 The permutation problem

Finding the optimum coordinate permutation from the point of view of trellis complexity is the only substantive outstanding issue in the field of trellis complexity of linear codes. Since little is known theoretically about this problem, it has been called “the art of trellis decoding” [Massey].

To illustrate that coordinate permutations do make a difference, consider the $(8, 4, 4)$ code that we have used as a running example. As an RM code with a standard coordinate ordering, we have seen that this code has state complexity profile $\{1, 2, 4, 8, 4, 8, 4, 2, 1\}$. On the other hand, consider the equivalent $(8, 4, 4)$ code generated by the following trellis-oriented generator matrix:

$$\begin{bmatrix} 11101000 \\ 01110100 \\ 00111010 \\ 00010111 \end{bmatrix}.$$

We see that the state complexity profile of this code is $\{1, 2, 4, 8, 16, 8, 4, 2, 1\}$, so its maximum state space size is 16.

In general, generator matrices that have a “cyclic” structure have poor state complexity profiles. For example, Exercise 5 shows that a trellis-oriented generator matrix for an MDS code always has such a structure, so an MDS code has the worst possible state complexity profile.

Finding the coordinate permutation that minimizes trellis complexity has been shown to be an NP-hard problem. On the other hand, various fragmentary results are known, such as:

- The Muder bound (see next subsection) applies to any coordinate permutation.
- The optimum coordinate ordering for RM codes is the standard coordinate ordering that results from the $|u|u + v|$ construction.
- A standard coordinate ordering for the $(24, 12, 8)$ binary Golay code achieves the Muder bound on the state complexity profile (see Exercise 6, below) everywhere.

10.3.2 The Muder bound

The Muder bound is a simple lower bound which shows that certain trellises have the smallest possible state space sizes. We show how this bound works by example.

Example 2 (cont.) Consider the $(8, 4, 4)$ code, with the first and last 4-tuples regarded as the past and future. The past subcode $\mathcal{C}_{\mathcal{P}}$ is then effectively a binary linear block code with length 4 and minimum distance at least 4. An upper bound on the largest possible dimension for such a code is $\dim \mathcal{C}_{\mathcal{P}} \leq 1$, achieved by the $(4, 1, 4)$ repetition code. A similar argument holds for the future subcode $\mathcal{C}_{\mathcal{F}}$. Thus the dimension of the state code \mathcal{S} is lowerbounded by

$$\dim \mathcal{S} = \dim \mathcal{C} - \dim \mathcal{C}_{\mathcal{P}} - \dim \mathcal{C}_{\mathcal{F}} \geq 4 - 1 - 1 = 2.$$

Thus no $(8, 4, 4)$ code can have a central state space with fewer than 4 states. □

Example 3 Consider any $(32, 16, 8)$ binary linear block code. If we partition the time axis into two halves of length 16, then the past subcode $\mathcal{C}_{\mathcal{P}}$ is effectively a binary linear block code with length 16 and minimum distance at least 8. An upper bound on the largest possible dimension for such a code is $\dim \mathcal{C}_{\mathcal{P}} \leq 5$, achieved by the $(16, 5, 8)$ biorthogonal code. A similar argument holds for the future subcode $\mathcal{C}_{\mathcal{F}}$. Therefore $\dim \mathcal{S}$ is lowerbounded by

$$\dim \mathcal{S} = \dim \mathcal{C} - \dim \mathcal{C}_{\mathcal{P}} - \dim \mathcal{C}_{\mathcal{F}} \geq 16 - 5 - 5 = 6.$$

Thus no $(32, 16, 8)$ code can have a central state space with fewer than 64 states. Exercise 1 showed that the $(32, 16, 8)$ RM code has a trellis whose central state space has 64 states. \square

In general, define $k_{\max}(n, d)$ as the largest possible dimension of code of length n and minimum distance d . There exist tables of $k_{\max}(n, d)$ for large ranges of (n, d) . The **Muder bound** is then (with k denoting the time index k and $\dim \mathcal{C}$ denoting the dimension of \mathcal{C}):

$$\dim \mathcal{S}_k \geq \dim \mathcal{C} - k_{\max}(k, d) - k_{\max}(n - k, d).$$

Similarly, for branch complexity, we have the Muder bound

$$\dim \mathcal{B}_k \geq \dim \mathcal{C} - k_{\max}(k - 1, d) - k_{\max}(n - k, d).$$

Exercise 6. The maximum possible dimension of a binary linear $(n, k, d \geq 8)$ block code is

$$k_{\max} = \{0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 4, 5, 5, 6, 7, 8, 9, 10, 11, 12\}$$

for $n = \{1, 2, \dots, 24\}$, respectively. [These bounds are achieved by $(8, 1, 8)$, $(12, 2, 8)$, $(16, 5, 8)$ and $(24, 12, 8)$ codes and shortened codes thereof.] Show that the best possible state complexity profile of any $(24, 12, 8)$ code (known as a binary Golay code) is

$$\{1, 2, 4, 8, 16, 32, 64, 128, 64, 128, 256, 512, 256, 512, 256, 128, 64, 128, 64, 32, 16, 8, 4, 2, 1\}.$$

Show that the best possible branch complexity profile is

$$\{2, 4, 8, 16, 32, 64, 128, 128, 128, 256, 512, 512, 512, 512, 256, 128, 128, 128, 64, 32, 16, 8, 4, 2\}.$$

A standard coordinate ordering that achieves both these bounds exists. \square

10.3.3 The sectionalization problem

The sectionalization problem is the problem of how many symbols to take at a time in the construction of a trellis. For example, we have seen that if we take one symbol at a time with our example $(8, 4, 4)$ code, then we obtain a state complexity profile of $\{1, 2, 4, 8, 4, 8, 4, 2, 1\}$ and a branch complexity profile of $\{2, 4, 8, 8, 8, 8, 4, 2\}$, whereas if we take two symbols at a time we obtain a state complexity profile of $\{1, 4, 4, 4, 1\}$ and a branch complexity profile of $\{4, 8, 8, 4\}$. The latter trellis has less apparent state complexity and a nicer-looking trellis, but its branch complexity is not significantly less.

Sectionalization affects to some degree the order of operations in VA decoding. For example, taking symbols two at a time means that the VA computes metrics of pairs of symbols before making comparisons. For the two trellises that we have considered for the $(8, 4, 4)$ code, it is apparent that this makes no material difference in VA decoding complexity.

Sectionalization may reduce the apparent state complexity, as we have seen. In fact, if the whole trellis is clustered into one section, then the state complexity profile becomes $\{1, 1\}$.

However, clustering cannot reduce and may increase branch complexity. For example, with a one-section trellis, there are 2^k parallel transitions from the starting to the ending state.

To prove this, we generalize the branch space theorem from intervals $\{k\} = [k, k+1)$ of length 1 to general clustered intervals $[k, k')$. A branch is then identified by a triple $(s_k, \mathbf{c}_{|[k, k')}, s_{k'})$, where $s_k \in \mathcal{S}_k$, $s_{k'} \in \mathcal{S}_{k'}$, and $\mathbf{c}_{|[k, k')}$ is the projection of a codeword onto the interval $[k, k')$. The branch space $\mathcal{B}_{[k, k')}$ is the set of all such branches such that there exists a codeword \mathbf{c} that passes through states $(s_k, s_{k'})$ and has the projection $\mathbf{c}_{|[k, k')}$.

By a similar argument to that in the proof of Theorem 10.5, we can conclude that in a minimal trellis two codewords go through the same branch in $\mathcal{B}_{[k, k')}$ if and only if they differ only by an element of the past subcode $\mathcal{C}_{[0, k)}$ and/or an element of the future subcode $\mathcal{C}_{[k', N)}$. This implies that the branch space $\mathcal{B}_{[k, k')}$ is a linear vector space with dimension

$$\dim \mathcal{B}_{[k, k')} = \dim \mathcal{C} - \dim \mathcal{C}_{[0, k)} - \dim \mathcal{C}_{[k', N)}.$$

Since $\dim \mathcal{C}_{[0, k)}$ is nonincreasing with decreasing k , and $\dim \mathcal{C}_{[k', N)}$ is nonincreasing with increasing k' , this shows that $\dim \mathcal{B}_{[k, k')} \geq \dim \mathcal{B}_{[\kappa, \kappa')}$ for $[\kappa, \kappa') \subseteq [k, k')$; in other words,

Theorem 10.6 *Clustering cannot decrease branch complexity.*

A good elementary rule for sectionalization is therefore to cluster as much as possible without increasing branch complexity. A heuristic rule for clustering as much as possible without increasing branch complexity is therefore as follows:

HEURISTIC CLUSTERING RULE: Extend sections toward the past as long as $\dim \mathcal{C}_{[0, k)}$ does not decrease, and toward the future as long as $\dim \mathcal{C}_{(k', N)}$ does not decrease; *i.e.*, to the past up to the next trellis-oriented generator stop time, and to the future up to the next trellis-oriented generator start time.

Example 2 (cont.) For our example $(8, 4, 4)$ code, the time-3 section may be extended back to the beginning and the time-4 section to the end without violating our rule, so the optimum sectionalization has just one boundary, at the center of the trellis—*i.e.*, we take symbols four at a time, as in Figure 3(b). The state complexity profile with this sectionalization is $\{1, 4, 1\}$, and the branch complexity profile is $\{8, 8\}$, so this sectionalization simplifies the state complexity profile as much as possible without increasing the maximum branch complexity. \square

Exercise 6 (cont.). Assuming that an optimum coordinate ordering for the $(24, 12, 8)$ binary Golay code exists such that the Muder bounds are met everywhere (it does), show that application of our heuristic clustering rule results in section boundaries at $k = \{0, 8, 12, 16, 24\}$, state complexity profile $\{0, 64, 256, 64, 0\}$ and branch complexity profile $\{128, 512, 512, 128\}$. \square

Lafourcade and Vardy have shown that, for any reasonable definition of optimality, there exists a polynomial-time algorithm for optimal sectionalization. They further observe that the following simple rule appears always to yield the optimal sectionalization:

LV RULE: Between any time when branches merge and the next subsequent time when branches diverge in an unclustered trellis, insert one and only one section boundary.

The LV rule and our rule give the same sectionalizations for the $(8, 4, 4)$ and $(24, 12, 8)$ codes.