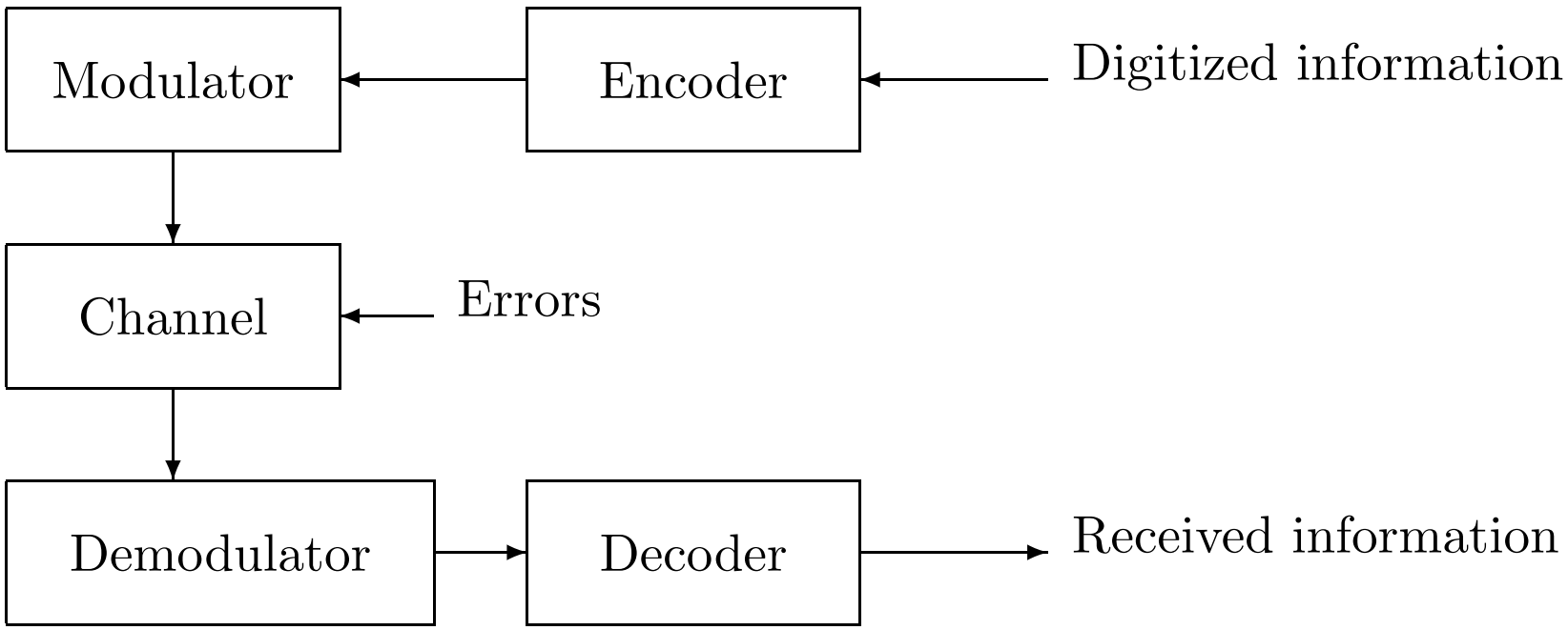
Introduction to Coding Theory- From Decoding Aspect

Yunghsiang S. Han

Graduate Institute of Communication Engineering,
National Taipei University
Taiwan

E-mail: yshan@mail.ntpu.edu.tw

Digital Communication System



Channel Model

1. The *time-discrete memoryless channel* (TDMC) is a channel specified by an arbitrary input space A, an arbitrary output space B, and for each element a in A , a conditional probability measure on every element b in B that is independent of all other inputs and outputs.
2. An example of TDMC is the *Additive White Gaussian Noise channel* (AWGN channel). Another commonly encountered channel is the *binary symmetric channel* (BSC).

AWGN Channel

1. Antipodal signaling is used in the transmission of binary signals over the channel.
2. A 0 is transmitted as $+\sqrt{E}$ and a 1 is transmitted as $-\sqrt{E}$, where E is the signal energy per channel bit.
3. The input space is $A = \{0, 1\}$ and the output space is $B = \boldsymbol{R}$.
4. When a sequence of input elements $(c_0, c_1, \ldots, c_{n-1})$ is transmitted, the sequence of output elements $(r_0, r_1, \ldots, r_{n-1})$ will be

$$r_j = (-1)^{c_j}\sqrt{E} + e_j,$$

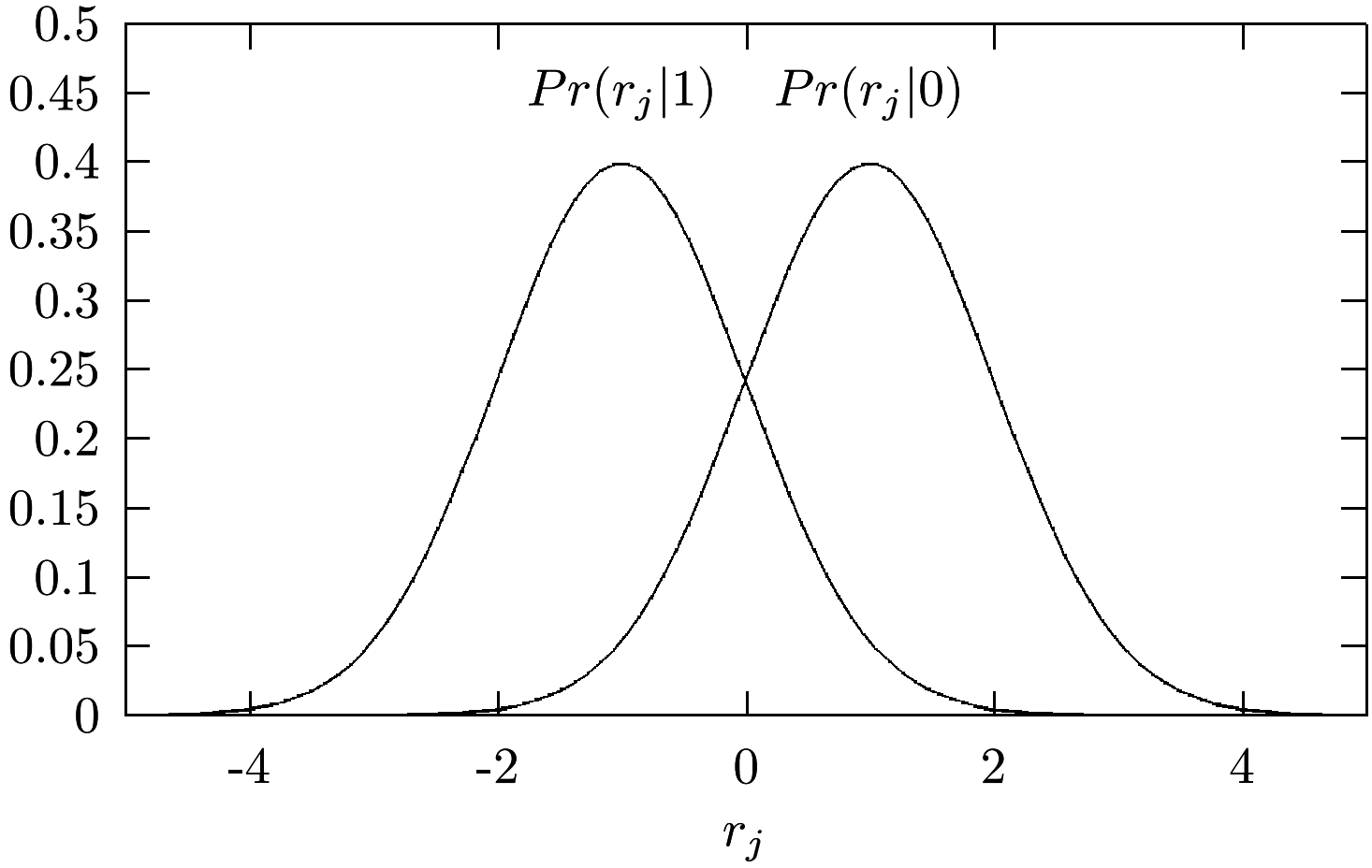
   $j = 0, 1, \ldots, n-1$, where e_j is a noise sample of a Gaussian process with single-sided noise power per hertz N_0.
5. The variance of e_j is $N_0/2$ and the *signal-to-noise ratio* (SNR) for the channel is $\gamma = E/N_0$.

6.

$$\boldsymbol{Pr}(r_j|c_j) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(r_j - (-1)^{c_j}\sqrt{E})^2}{N_0}}.$$

Probability distribution function for r_j

The signal energy per channel bit E has been normalized to 1.



Binary Symmetric Channel

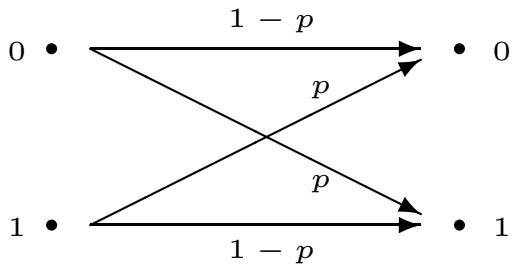
1. BSC is characterized by a probability p of bit error such that the probability p of a transmitted bit 0 being received as a 1 is the same as that of a transmitted 1 being received as a 0.

2. BSC may be treated as a simplified version of other symmetric channels. In the case of AWGN channel, we may assign p as

$$\begin{aligned} p &= \int_0^\infty \boldsymbol{Pr}(r_j|1)dr_j \\ &= \int_{-\infty}^0 \boldsymbol{Pr}(r_j|0)dr_j \\ &= \int_0^\infty \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(r_j+\sqrt{E})^2}{N_0}} dr_j \\ &= Q\left((2E/N_0)^{\frac{1}{2}}\right) \end{aligned}$$

where

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} dy$$



Binary symmetric channel

Binary Linear Block Code (BLBC)

1. An (n, k) binary linear block code is a k-dimensional subspace of the n-dimensional vector space
   $\boldsymbol{V}_n = \{(c_0, c_1, \ldots, c_{n-1}) | \forall c_j \; c_j \in \boldsymbol{GF(2)}\}$; n is called the length of the code, k the dimension.

2. Example: a $(6, 3)$ code

$$\begin{aligned} \boldsymbol{C} &= \{000000, 100110, 010101, 001011, \\ &\quad 110011, 101101, 011110, 111000\} \end{aligned}$$

Generator Matrix

1. An (n, k) BLBC can be specified by any set of k linear independent codewords $\boldsymbol{c_0}$,$\boldsymbol{c_1}$,. . . ,$\boldsymbol{c_{k-1}}$. If we arrange the k codewords into a $k \times n$ matrix $\boldsymbol{G}$, $\boldsymbol{G}$ is called a *generator matrix* for $\boldsymbol{C}$.

2. Let $\boldsymbol{u} = (u_0, u_1, \ldots, u_{k-1})$, where $u_j \in \boldsymbol{GF(2)}$.

$$\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) = \boldsymbol{uG}.$$

3. The generator matrix $\boldsymbol{G'}$ of a systematic code has the form of $[\boldsymbol{I_k A}]$, where $\boldsymbol{I_k}$ is the $k \times k$ identity matrix.

4. $\boldsymbol{G'}$ can be obtained by permuting the columns of $\boldsymbol{G}$ and by doing some row operations on $\boldsymbol{G}$. We say that the code generated by $\boldsymbol{G'}$ is an equivalent code of the generated by $\boldsymbol{G}$.

Example

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

be a generator matrix of $\boldsymbol{C}$ and

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

be a parity-check matrix of $\boldsymbol{C}$.

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G}$$

$$\boldsymbol{u} \in \{000, 100, 010, 001, 110, 101, 011, 111\}$$

$$\begin{aligned} \boldsymbol{C} &= \{000000, 100110, 010101, 001011, \\ &\quad 110011, 101101, 011110, 111000\} \end{aligned}$$

Parity-Check Matrix

1. A *parity check* for $\boldsymbol{C}$ is an equation of the form

$$a_0c_0 \oplus a_1c_1 \oplus \ldots \oplus a_{n-1}c_{n-1} = 0,$$

   which is satisfied for any $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) \in \boldsymbol{C}$.

2. The collection of all vectors $\boldsymbol{a} = (a_0, a_1, \ldots, a_{n-1})$ forms a subspace of $\boldsymbol{V}_n$. It is denoted by $\boldsymbol{C}^\perp$ and is called the *dual code* of $\boldsymbol{C}$.

3. The dimension of $\boldsymbol{C}^\perp$ is $n-k$ and $\boldsymbol{C}^\perp$ is an $(n, n-k)$ BLBC. Any generator matrix of $\boldsymbol{C}^\perp$ is a *parity-check matrix* for $\boldsymbol{C}$ and is denoted by $\boldsymbol{H}$.

4. $\boldsymbol{cH}^T = \boldsymbol{0}_{1\times(n-k)}$ for any $\boldsymbol{c} \in \boldsymbol{C}$.

5. Let $\boldsymbol{G} = [\boldsymbol{I}_k \boldsymbol{A}]$. Since $\boldsymbol{cH}^T = \boldsymbol{uGH}^T = \boldsymbol{0}$, $\boldsymbol{GH}^T$ must be $\boldsymbol{0}$. If

$\boldsymbol{H} = \left[-\boldsymbol{A}^T \boldsymbol{I}_{n-k}\right]$, then

$$\begin{aligned} \boldsymbol{G}\boldsymbol{H}^T &= [\boldsymbol{I}_k \boldsymbol{A}] \left[-\boldsymbol{A}^T \boldsymbol{I}_{n-k}\right]^T \\ &= [\boldsymbol{I}_k \boldsymbol{A}] \begin{bmatrix} -\boldsymbol{A} \\ \boldsymbol{I}_{n-k} \end{bmatrix} = -\boldsymbol{A} + \boldsymbol{A} = \boldsymbol{0}_{k\times(n-k)} \end{aligned}$$

Thus, the above $\boldsymbol{H}$ is a parity-check matrix.

6. Let $\boldsymbol{c}$ be the transmitted codeword and $\boldsymbol{y}$ is the binary received vector after quantization. The vector $\boldsymbol{e} = \boldsymbol{c} \oplus \boldsymbol{y}$ is called an *error pattern*.

7. Let $\boldsymbol{y} = \boldsymbol{c} \oplus \boldsymbol{e}$.

$$\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T = (\boldsymbol{c} \oplus \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{e}\boldsymbol{H}^T$$

which is called the *syndrome* of $\boldsymbol{y}$.

8. Let $S = \{\boldsymbol{s} | \boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T \text{ for all } \boldsymbol{y} \in \boldsymbol{V}_n\}$ be the set of all syndromes. Thus, $|S| = 2^{n-k}$.

Hamming Weight and Hamming Distance (1)

1. The Hamming weight (or simply called weight) of a codeword $\boldsymbol{c}$, $W_H(\boldsymbol{c})$, is the number of 1's (the nonzero components) of the codeword.
2. The Hamming distance between two codewords $\boldsymbol{c}$ and $\boldsymbol{c'}$ is defined as $d_H(\boldsymbol{c}, \boldsymbol{c'})$ = the number of components in which $\boldsymbol{c}$ and $\boldsymbol{c'}$ differ.
3. $d_H(\boldsymbol{c},\boldsymbol{0}) = W_H(\boldsymbol{c})$.
4. Let HW be the set of all distinct Hamming weights that codewords of $\boldsymbol{C}$ may have. Furthermore, let $HD(\boldsymbol{c})$ be the set of all distinct Hamming distances between $\boldsymbol{c}$ and any codeword. Then, $HW = HD(\boldsymbol{c})$ for any $\boldsymbol{c} \in \boldsymbol{C}$.
5. $d_H(\boldsymbol{c}, \boldsymbol{c'}) = d_H(\boldsymbol{c} \oplus \boldsymbol{c'}, \boldsymbol{0}) = W_H(\boldsymbol{c} \oplus \boldsymbol{c'})$

6. If $\boldsymbol{C}$ and $\boldsymbol{C'}$ are equivalent to each other, then the HW for $\boldsymbol{C}$ is the same as that for $\boldsymbol{C'}$.

7. The smallest nonzero element in HW is referred to as d_{min}.

8. Let the column vectors of $\boldsymbol{H}$ be $\{\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_{n-1}\}$.

$$\begin{aligned} \boldsymbol{c}\boldsymbol{H}^T &= (c_0, c_1, \ldots, c_{n-1}) \left[\boldsymbol{h}_0 \ \boldsymbol{h}_1 \ \cdots \ \boldsymbol{h}_{n-1}\right]^T \\ &= c_0\boldsymbol{h}_0 + c_1\boldsymbol{h}_1 + \cdots + c_{n-1}\boldsymbol{h}_{n-1} \end{aligned}$$

9. If $\boldsymbol{c}$ is of weight w, then $\boldsymbol{c}\boldsymbol{H}^T$ is a linear combination of w columns of $\boldsymbol{H}$.

10. d_{min} is the minimum nonzero number of columns in $\boldsymbol{H}$ where a nontrivial linear combination results in zero.

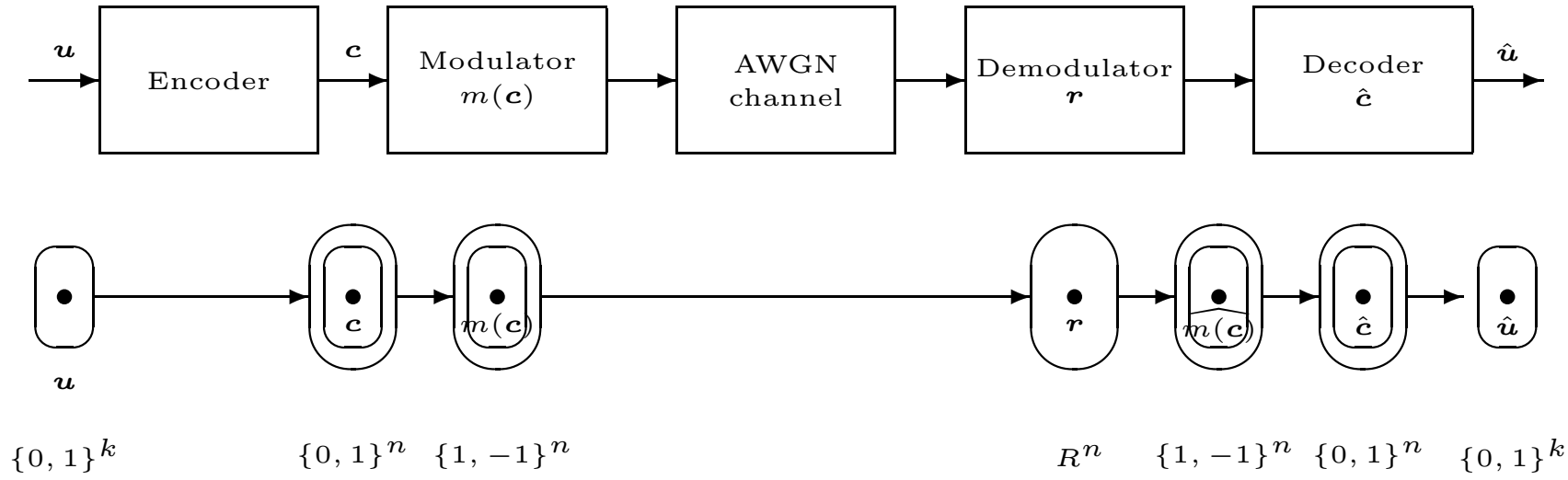
Hamming Weight and Hamming Distance (2)

$$\begin{aligned} \boldsymbol{C} &= \{000000, 100110, 010101, 001011, \\ &\quad 110011, 101101, 011110, 111000\} \end{aligned}$$

$HW = HD(\boldsymbol{c}) = \{0, 3, 4\}$ for all $\boldsymbol{c} \in \boldsymbol{C}$

$$d_H(001011, 110011) = d_H(111000, 000000) = W_H(111000) = 3$$

Digital Communication System Revisited



Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (1)

1. The goal of decoding:

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{c}_{\boldsymbol{\ell}} \text{ where } \boldsymbol{c}_{\boldsymbol{\ell}} \in \boldsymbol{C} \text{ and}$$

$$\boldsymbol{Pr}(\boldsymbol{c}_{\boldsymbol{\ell}}|\boldsymbol{r}) \geq \boldsymbol{Pr}(\boldsymbol{c}|\boldsymbol{r}) \text{ for all } \boldsymbol{c} \in \boldsymbol{C}.$$

2. If all codewords of $\boldsymbol{C}$ have equal probability of being transmitted, then to maximize $\boldsymbol{Pr}(\boldsymbol{c}|\boldsymbol{r})$ is equivalent to maximizing $\boldsymbol{Pr}(\boldsymbol{r}|\boldsymbol{c})$, where $\boldsymbol{Pr}(\boldsymbol{r}|\boldsymbol{c})$ is the probability that $\boldsymbol{r}$ is received when $\boldsymbol{c}$ is transmitted, since

$$\boldsymbol{Pr}(\boldsymbol{c}|\boldsymbol{r}) = \frac{\boldsymbol{Pr}(\boldsymbol{r}|\boldsymbol{c})\boldsymbol{Pr}(\boldsymbol{c})}{\boldsymbol{Pr}(\boldsymbol{r})} \ .$$

3. A maximum-likelihood decoding rule (**MLD** rule), which minimizes error probability when each codeword is transmitted

equiprobably, decodes a received vector $\boldsymbol{r}$ to a codeword $\boldsymbol{c_\ell} \in \boldsymbol{C}$ such that

$$\boldsymbol{Pr}(\boldsymbol{r}|\boldsymbol{c_\ell}) \geq \boldsymbol{Pr}(\boldsymbol{r}|\boldsymbol{c}) \text{ for all } \boldsymbol{c} \in \boldsymbol{C}.$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (2)

For a time-discrete memoryless channel, the **MLD** rule can be formulated as

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{c_\ell}$$

$$\text{where } \boldsymbol{c_\ell} = (c_{\ell 0}, c_{\ell 1}, \ldots, c_{\ell(n-1)}) \in \boldsymbol{C} \text{ and}$$

$$\prod_{j=0}^{n-1} \boldsymbol{Pr}(r_j|c_{\ell j}) \geq \prod_{j=0}^{n-1} \boldsymbol{Pr}(r_j|c_j) \text{ for all } \boldsymbol{c} \in \boldsymbol{C}.$$

Let $S(\boldsymbol{c},\boldsymbol{c_\ell}) \subseteq \{0, 1, \ldots, n-1\}$ be defined as $j \in S(\boldsymbol{c}, \boldsymbol{c_\ell})$ iff $c_{\ell j} \neq c_j$. Then the **MLD** rule can be written as

$$\text{set } \hat{\boldsymbol{c}} = \boldsymbol{c_\ell} \text{ where } \boldsymbol{c_\ell} \in \boldsymbol{C} \text{ and}$$

$$\sum_{j \in S(\boldsymbol{c}, \boldsymbol{c_\ell})} \ln \frac{\boldsymbol{Pr}(r_j | c_{\ell j})}{\boldsymbol{Pr}(r_j | c_j)} \geq 0 \text{ for all } \boldsymbol{c} \in \boldsymbol{C}.$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (3)

1. The bit log-likelihood ratio of r_j

$$\phi_j = \ln \frac{\boldsymbol{Pr}(r_j|0)}{\boldsymbol{Pr}(r_j|1)} .$$

2. let $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_{n-1})$. The absolute value of ϕ_j is called the *reliability* of position j of received vector.

3. For AWGN channel

$$\boldsymbol{Pr}(r_j|0) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(r_j - \sqrt{E})^2}{N_0}}$$

and

$$\boldsymbol{Pr}(r_j|1) = \frac{1}{\sqrt{\pi N_0}} e^{-\frac{(r_j + \sqrt{E})^2}{N_0}} .$$

Since

$$\phi_j = \ln \frac{\boldsymbol{Pr}(r_j|0)}{\boldsymbol{Pr}(r_j|1)} = \frac{4\sqrt{E}}{N_0} r_j,$$

then

$$\boldsymbol{\phi} = \frac{4\sqrt{E}}{N_0} \boldsymbol{r}.$$

4. For BSC,

$$\phi_j = \begin{cases} \ln \frac{1-p}{p} & : \quad \text{if } r_j = 0; \\ \ln \frac{p}{1-p} & : \quad \text{if } r_j = 1. \end{cases}$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (4)

$$\sum_{j \in S(\boldsymbol{c},\boldsymbol{c_\ell})} \ln \frac{\boldsymbol{Pr}(r_j|c_{\ell j})}{\boldsymbol{Pr}(r_j|c_j)} \geq 0$$

$$\iff \quad 2 \sum_{j \in S(\boldsymbol{c},\boldsymbol{c_\ell})} \ln \frac{\boldsymbol{Pr}(r_j|c_{\ell j})}{\boldsymbol{Pr}(r_j|c_j)} \geq 0$$

$$\iff \quad \sum_{j \in S(\boldsymbol{c},\boldsymbol{c_\ell})} ((-1)^{c_{\ell j}} \phi_j - (-1)^{c_j} \phi_j) \geq 0$$

$$\iff \quad \sum_{j=0}^{n-1} (-1)^{c_{\ell j}} \phi_j \geq \sum_{j=0}^{n-1} (-1)^{c_j} \phi_j$$

$$\iff \quad \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{\ell j}})^2 \leq \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_j})^2$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (5)

1. The **MLD** rule can be written as

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{c}_\ell, \text{ where } \boldsymbol{c}_\ell \in \boldsymbol{C} \text{ and}$$

$$\sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{\ell j}})^2 \le \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_j})^2$$

$$\text{for all } \boldsymbol{c} \in \boldsymbol{C}.$$

2. we will say that $\boldsymbol{c_\ell}$ is the "closest" codeword to $\boldsymbol{\phi}$.

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (6)

1. Let m be a function such that

$$m(\boldsymbol{c}) = ((-1)^{c_0}, \ldots, (-1)^{c_{n-1}}).$$

2. Let $\langle \boldsymbol{\phi}, m(\boldsymbol{c}) \rangle = \sum_{j=0}^{n-1} (-1)^{c_j} \phi_j$ be the inner product between $\boldsymbol{\phi}$ and $m(\boldsymbol{c})$.

3. The **MLD** rule can be written as

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{c}_\ell, \text{ where } \boldsymbol{c}_\ell \in \boldsymbol{C} \text{ and}$$

$$\langle \boldsymbol{\phi}, m(\boldsymbol{c}_\ell) \rangle \geq \langle \boldsymbol{\phi}, m(\boldsymbol{c}) \rangle \text{ for all } \boldsymbol{c} \in \boldsymbol{C}.$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (7)

Let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ be the hard-decision of $\boldsymbol{\phi}$. That is

$$y_j = \begin{cases} 1 & : \quad \text{if } \phi_j < 0; \\ 0 & : \quad \text{otherwise.} \end{cases}$$

$$\sum_{j=0}^{n-1} (-1)^{c_{\ell j}} \phi_j \geq \sum_{j=0}^{n-1} (-1)^{c_j} \phi_j$$

$$\iff \quad \frac{1}{2} \sum_{j=0}^{n-1} [(-1)^{y_j} - (-1)^{c_{\ell j}}] \phi_j \leq$$

$$\frac{1}{2} \sum_{j=0}^{n-1} [(-1)^{y_j} - (-1)^{c_j}] \phi_j$$

$$\iff \quad \sum_{j=0}^{n-1} (y_j \oplus c_{\ell j}) |\phi_j| \leq \sum_{j=0}^{n-1} (y_j \oplus c_j) |\phi_j|$$

$$\iff \quad \sum_{j=0}^{n-1} e_{\ell j} |\phi_j| \leq \sum_{j=0}^{n-1} e_j |\phi_j|$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Word-by-Word Decoding (8)

1. Let $\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^T$ be the syndrome of $\boldsymbol{y}$.
2. Let $E(\boldsymbol{s})$ be the collection of all error patterns whose syndrome is $\boldsymbol{s}$. Clearly, $|E(\boldsymbol{s})| = |\boldsymbol{C}| = 2^k$.
3. The **MLD** rule can be stated as

$$\text{set } \widehat{\boldsymbol{c}} = \boldsymbol{y} \oplus \boldsymbol{e}_{\boldsymbol{\ell}}, \text{ where } \boldsymbol{e}_\ell \in E(\boldsymbol{s}) \text{ and}$$

$$\sum_{j=0}^{n-1} e_{\ell j}|\phi_j| \le \sum_{j=0}^{n-1} e_j|\phi_j| \quad \text{for all } \boldsymbol{e} \in E(\boldsymbol{s}).$$

Distance Metrics of MLD Rule for Word-by-Word Decoding

Let m be a function such that $m(\boldsymbol{c}) = ((-1)^{c_0}, \ldots, (-1)^{c_{n-1}})$ and let $\boldsymbol{y} = (y_0, y_1, \ldots, y_{n-1})$ be the hard decision of $\boldsymbol{\phi}$. That is

$$y_j = \begin{cases} 1 & : \quad \text{if } \phi_j < 0; \\ 0 & : \quad \text{otherwise.} \end{cases}$$

1. $d_E^2(m(\boldsymbol{c}), \boldsymbol{\phi}) = \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_j})^2$

2. $\langle \boldsymbol{\phi}, m(\boldsymbol{c}) \rangle = \sum_{j=0}^{n-1} (-1)^{c_j} \phi_j$

3. $d_D(\boldsymbol{c}, \boldsymbol{\phi}) = \sum_{j=0}^{n-1} (y_j \oplus c_j)|\phi_j| = \sum_{j \in T} |\phi_j|$, where $T = \{j | y_j \neq c_j\}$.

4. Relations between the metrics:

   - $d_E^2(m(\boldsymbol{c}), \boldsymbol{\phi}) = \sum_{j=0}^{n-1} \phi_j^2 - 2\langle \boldsymbol{\phi}, m(\boldsymbol{c}) \rangle + n =$
     $\|\boldsymbol{\phi}\|^2 + n - 2\langle \boldsymbol{\phi}, m(\boldsymbol{c}) \rangle$
   -

$$\begin{aligned} \langle \boldsymbol{\phi}, m(\boldsymbol{c}) \rangle &= \sum_{j \in T^c} |\phi_j| - \sum_{j \in T} |\phi_j| \\ &= \sum_{j \in (T \cup T^c)} |\phi_j| - 2 \sum_{j \in T} |\phi_j| \\ &= \sum_{j=0}^{n-1} |\phi_j| - 2 d_D(\boldsymbol{c}, \boldsymbol{\phi}) \end{aligned}$$

   where T^c is the complement set of T.

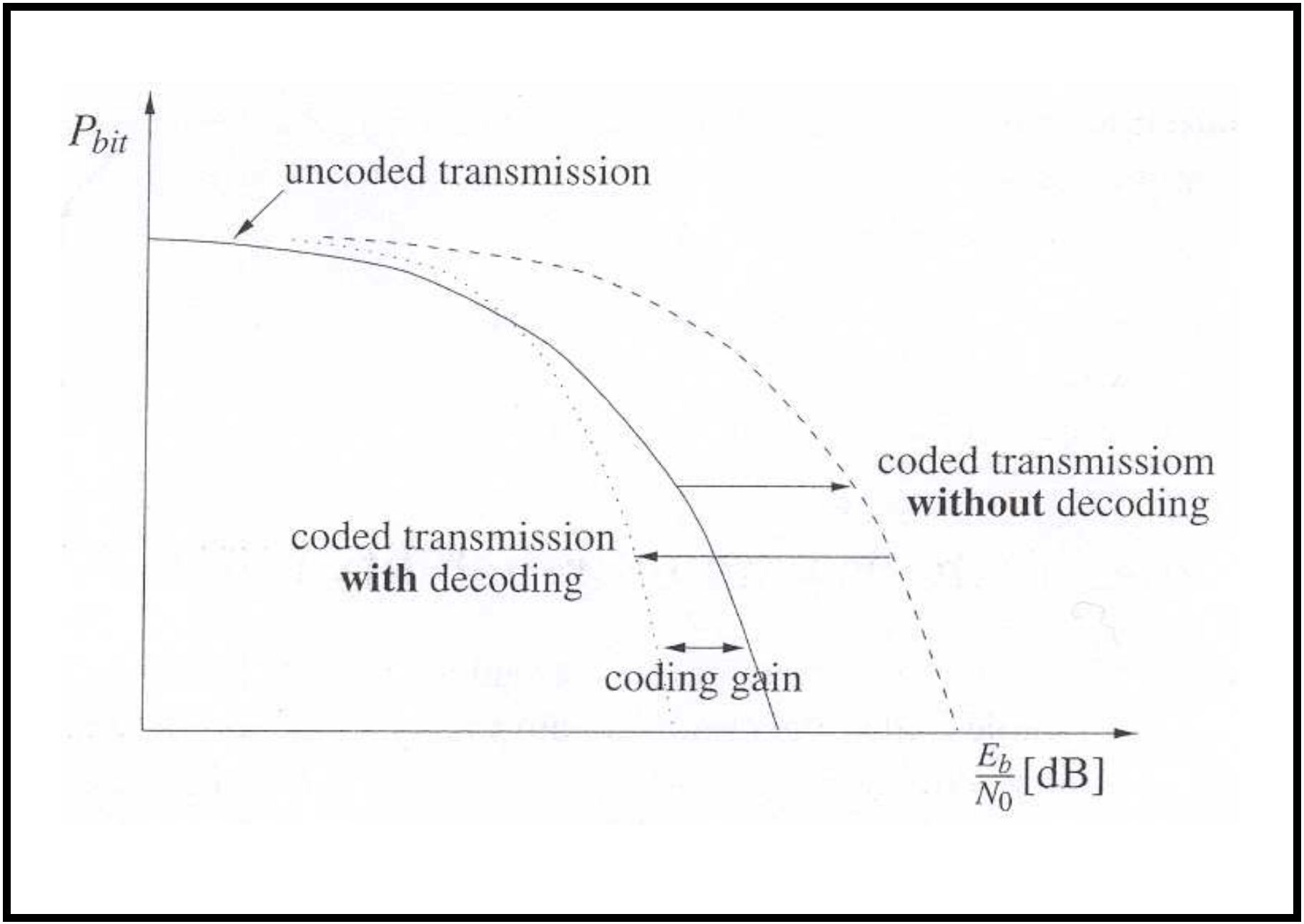
5. When one only considers BSC, $|\phi_j| = |\ln \frac{1-p}{p}|$ will be the same for all positions. Thus,

$$d_D(\boldsymbol{c}, \boldsymbol{\phi}) = \sum_{j=0}^{n-1} (y_j \oplus c_j)|\phi_j| = |\ln \frac{1-p}{p}| \sum_{j=0}^{n-1} (y_j \oplus c_j)$$

In this case, the distance metric will reduce to the Hamming distance between $\boldsymbol{c}$ and $\boldsymbol{y}$, i.e., $\sum_{j=0}^{n-1} (y_j \oplus c_j) = d_H(\boldsymbol{c}, \boldsymbol{y})$. Under this condition, a decoder is called a *hard-decision* decoder; otherwise, the decoder is called a *soft-decision* decoder.

Coding Gain [2]

1. $R = \frac{k}{n}$.
2. $$kE_b = nE_s \rightarrow E_b = \frac{E_s}{R}.$$
   where E_b is the energy per information bit and E_s the energy per received symbol.
3. For a coding scheme, the *coding gain* at a given bit error probability is defined as the difference between the energy per information bit required by the coding scheme to achieve the given bit error probability and that by uncoded transmission.


P_{bit}
uncoded transmission
coded transmissiom
without decoding
coded transmission
with decoding
coding gain
$\frac{E_b}{N_0}$ [dB]

Error Probability for AWGN Channel

1. The word error probability $\mathcal{E}_s$ with ML decoder is

$$\begin{aligned} \boldsymbol{Pr}(\mathcal{E}_s) &\approx \sum_{w=d_{min}}^{n} A_w Q((2wE_s/N_0)^{1/2}) \\ &= \sum_{w=d_{min}}^{n} A_w Q((2wRE_b/N_0)^{1/2}), \end{aligned}$$

   where A_w is the number of codewords with Hamming weight w.

2. The bit error probability $\mathcal{E}_b$ with ML decoder is

$$\boldsymbol{Pr}(\mathcal{E}_b) \approx \sum_{w=d_{min}}^{n} \frac{w}{n} A_w Q((2wRE_b/N_0)^{1/2})$$

3. At moderate to high SNRs, the first term of the above upper bound is the most significant one. Therefore, the minimum

distance and the number of codewords of minimum weight of a code are two major factors which determine the bit error rate performance of the code.

4. Since $Q(x) \leq \frac{1}{\sqrt{2\pi}x} e^{-x^2/2}$ for all $x > 0$,

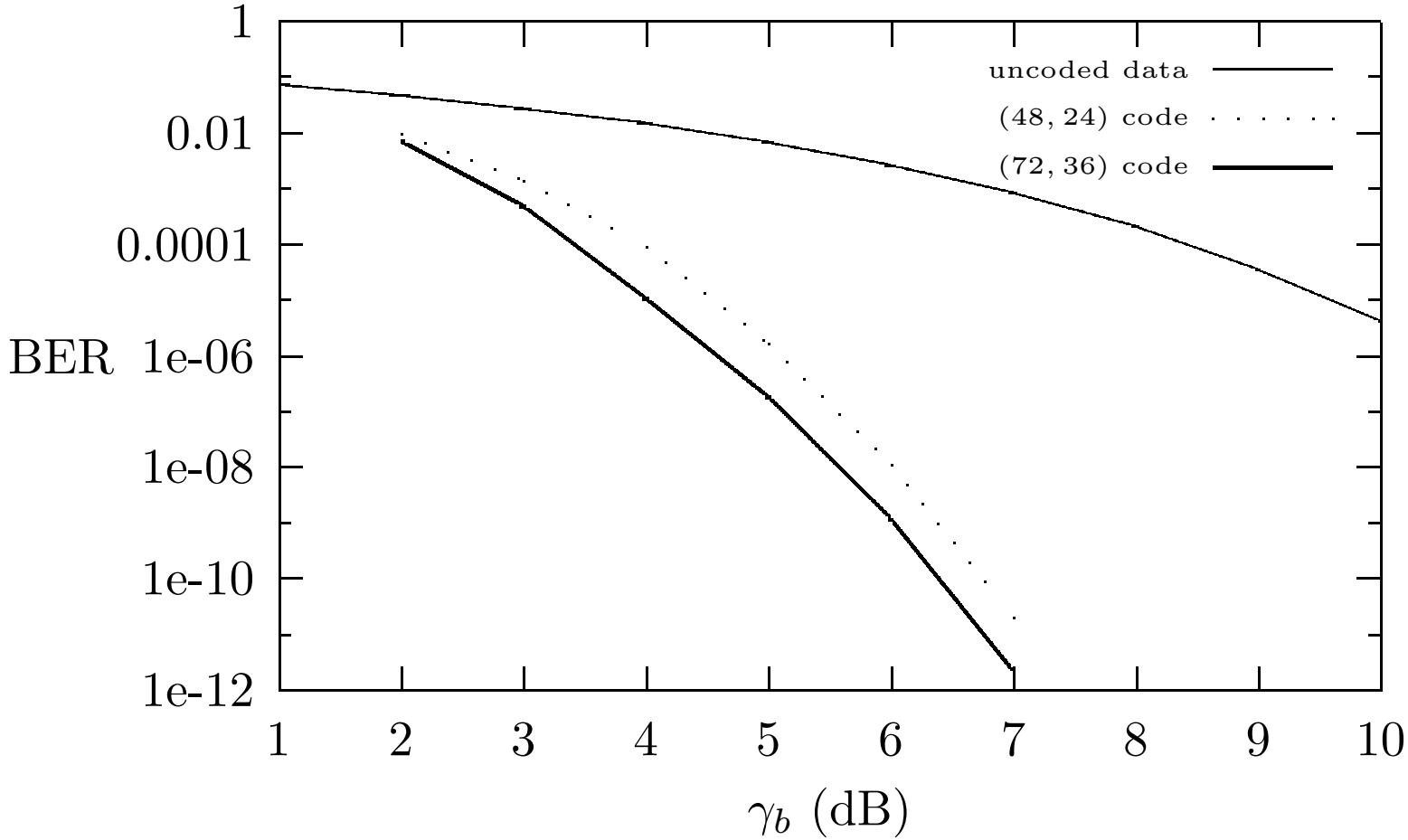
$$\boldsymbol{Pr}(\mathcal{E}_b) \approx A_{d_{min}} \sqrt{\frac{d_{min}}{4\pi n k \gamma_b}} e^{-(R d_{min} \gamma_b)} \tag{1}$$

where $\gamma_b = E_b/N_0$.

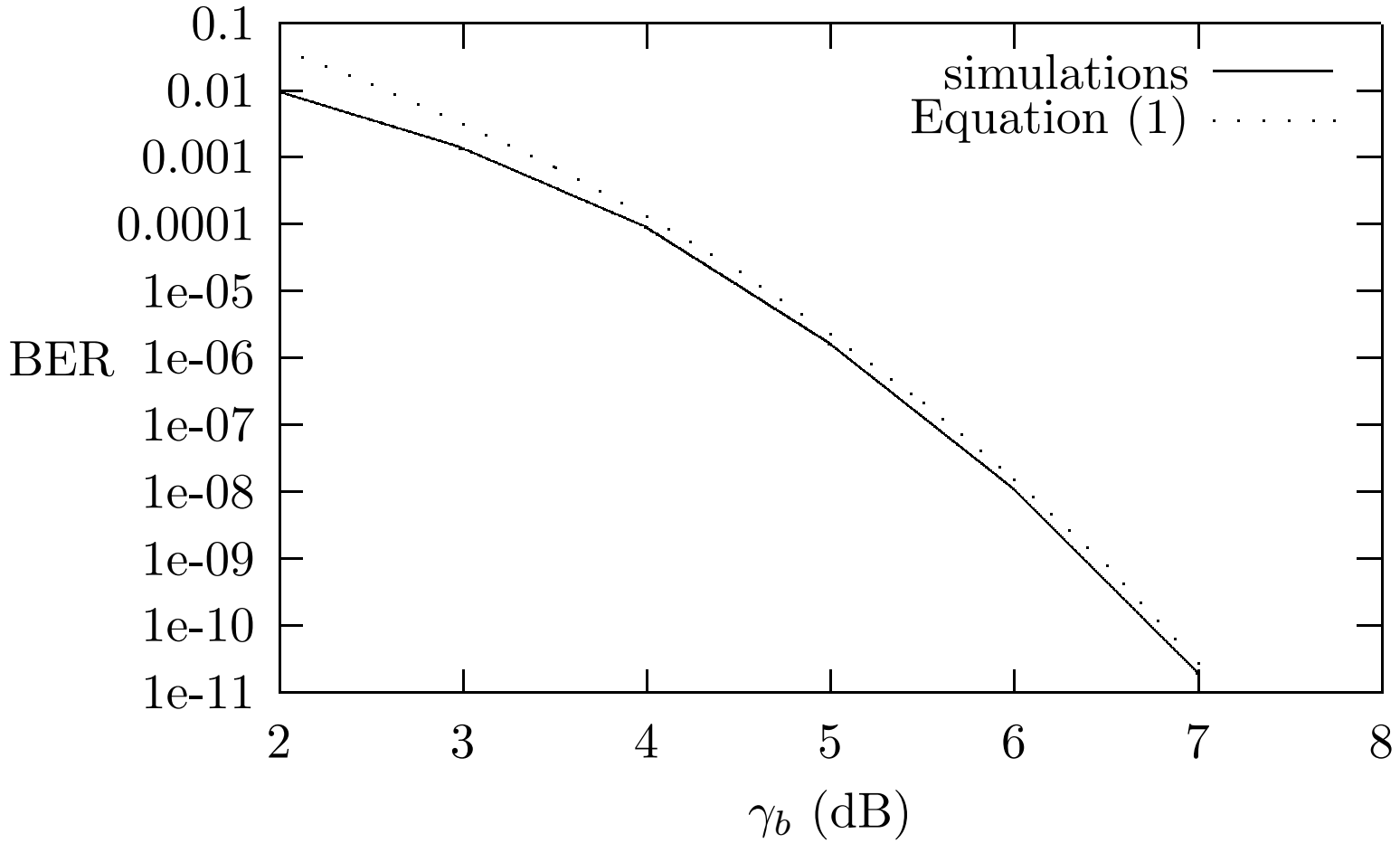
5. We may use Equation 1 to calculate bit error probability of a code at moderate to high SNRs.

6. The bit error probability of $(48, 24)$ and $(72, 36)$ binary extended quadratic residue (QR) codes are given in the next two slides [5]. Both codes have $d_{min} = 12$. $A_{d_{min}}$ for the $(48, 24)$ code is 17296 and for the $(72, 36)$ code is 2982.

Bit Error Probability of Extended QR Codes for AWGN Channel



Bit Error Probability of the $(48, 24)$ Extended QR Codes for AWGN Channel



Asymptotic Coding Gain for AWGN Channel

1. At large x the Q function can be overbounded by

$$Q(x) \leq e^{-x^2/2}$$

2. An uncoded transmission has a bit error probability of

$$\begin{aligned} \boldsymbol{Pr}(\mathcal{E}_b) &= Q\left((2E_b'/N_0)^{1/2}\right) \\ &\leq \exp\left(-\frac{E_b'}{N_0}\right) \end{aligned}$$

3. At high SNRs for a linear block code with minimum distance d_{min} the bit error probability is approximated by the first term

of the union bound. That is,

$$\begin{aligned} \boldsymbol{Pr}(\mathcal{E}_b) &\approx \frac{d_{min}}{n} A_{d_{min}} Q\left((2d_{min}RE_b/N_0)^{1/2}\right) \\ &\leq \frac{d_{min}}{n} A_{d_{min}} \exp\left(-\frac{d_{min}RE_b}{N_0}\right) \end{aligned}$$

4. Let
$$\frac{d_{min}}{n} A_{d_{min}} \exp\left(-\frac{d_{min}RE_b}{N_0}\right) = \exp\left(-\frac{E_b'}{N_0}\right)$$

5. Taking the logarithm of both sides and noting that $\log\left[\frac{d_{min}}{n} A_{d_{min}}\right]$ is negligible for large SNR we have
$$\frac{E_b'}{E_b} = d_{min}R$$

6. The asymptotic coding gain is
$$G_a = 10\log[d_{min}R]$$

Soft-Decision Decoding vs Hard-Decision Decoding

1. It can be shown that the asymptotic coding gain of hard-decision decoding is $G_a = 10\log\left[R(d_{min}+1)/2\right]$.
2.

$$\begin{aligned} G_{diff} &= 10\log[d_{min}R] - 10\log\left[R(d_{min}+1)/2\right] \\ &= 10\log\left[\frac{d_{min}R}{R(d_{min}+1)/2}\right] \\ &\approx 10\log[2] = 3\ dB \end{aligned}$$

3. Soft-decision decoding is about 3 dB more efficient than hard-decision decoding at very high SNRs. At realistic SNR, 2 dB is more common.
4. There exist fast algebraic decoding algorithms for powerful linear

block codes such as BCH codes and Reed-Solomon codes.

5. The soft-decision decoding algorithms usually are more complex than the hard-decision decoding algorithms.

6. There are tradeoffs on bit error probability performance and decoding time complexity between these two types of decoding algorithms.

Trellis of Linear Block Codes [1]

1. Let $\boldsymbol{H}$ be a parity-check matrix of $\boldsymbol{C}$, and let $\boldsymbol{h_j}$, $0 \le j \le n-1$ be the column vectors of $\boldsymbol{H}$.
2. Let $\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1})$ be a codeword of $\boldsymbol{C}$. With respect to this codeword, we recursively define the states $\boldsymbol{s_t}$, $0 \le t \le n$, as

$$\boldsymbol{s}_0 = \boldsymbol{0}$$

and

$$\boldsymbol{s}_t = \boldsymbol{s}_{t-1} + c_{t-1}\boldsymbol{h}_{t-1} = \sum_{j=0}^{t-1} c_j \boldsymbol{h}_j, \quad 1 \le t \le n.$$

3. $\boldsymbol{s}_n = \boldsymbol{0}$ for all codewords of $\boldsymbol{C}$.
4. The above recursive equation can be used to draw a trellis diagram.
5. In this trellis, $\boldsymbol{s}_0 = \boldsymbol{0}$ identifies the start node at level 0; $\boldsymbol{s}_n = \boldsymbol{0}$

identifies the goal node at level n; and each state $\boldsymbol{s}_t, 1 \leq t \leq n-1$ identifies a node at level t.

6. Each transition (branch) is labelled with the appropriate codeword bit c_t.

7. There is a one-to-one correspondence between the codewords of $\boldsymbol{C}$ and the sequences of labels encountered when traversing a path in the trellis from the start node to the goal node.

Example of Trellis

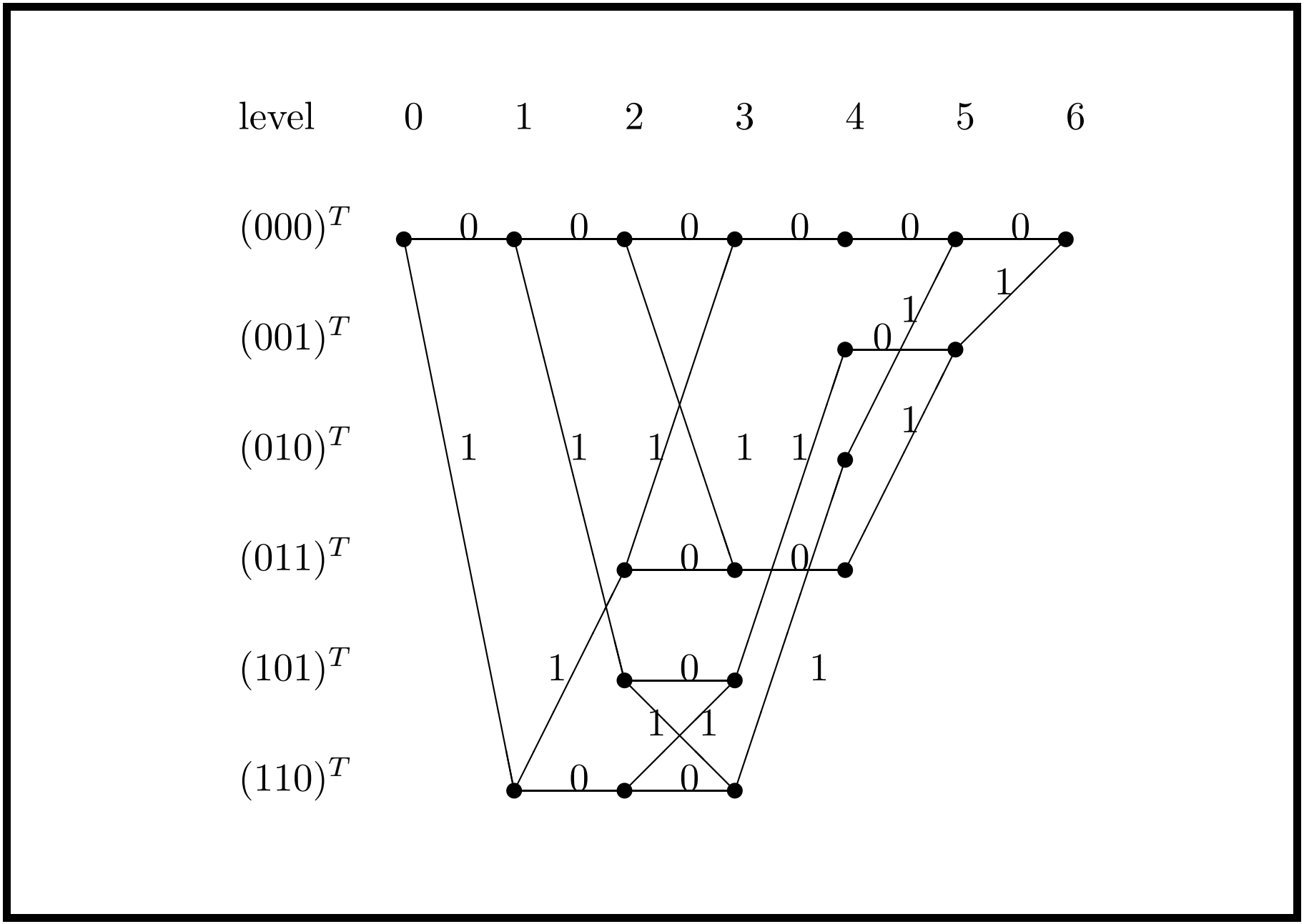
Let

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

be a generator matrix of $\boldsymbol{C}$ and let

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

be a parity-check matrix of $\boldsymbol{C}$.


level 0 1 2 3 4 5 6
$(000)^T$
$(001)^T$
$(010)^T$
$(011)^T$
$(101)^T$
$(110)^T$

Convert a Decoding Problem to a Graph-Search Problem

1. According to the MLD rule presented, we want to find a codeword $\boldsymbol{c_\ell}$ such that $d_E^2(m(\boldsymbol{c}_\ell), \boldsymbol{\phi}) = \sum_{j=0}^{n-1} (\phi_j - (-1)^{c_{\ell j}})^2$ is the minimum among all the codewords.
2. If we properly specify the branch costs (branch metrics) on a trellis, the **MLD** rule can be written as follows:

   Find a path from the start node to a goal node such that the cost of the path is minimum among all the paths from the start node to a goal node, where a cost of a path is the summation of the cost of branches in the path. Such a path is called an *optimal path*.
3. In the trellis of $\boldsymbol{C}$ the cost of the branch from $\boldsymbol{s}_{t-1}$ to $\boldsymbol{s}_t =$

$\boldsymbol{s}_{t-1} + c_{t-1}\boldsymbol{h}_{t-1}$ is assigned the value $(\phi_{t-1} - (-1)^{c_{t-1}})^2$.

4. The solution of the decoding problem is converted into finding a path from the start node to a goal node in the trellis, that is, a codeword $\boldsymbol{c}= (c_0, c_1, \ldots, c_{n-1})$ such that $d_E^2(m(\boldsymbol{c}), \boldsymbol{\phi})$ is minimum among all paths from the start node to a goal node.

5. The *path metric* for a codeword $\boldsymbol{c}= (c_0, c_1, \ldots, c_{n-1})$ is defined as

$$\sum_{j=0}^{n-1} (\phi_j - (-1)^{c_j})^2$$

6. The *ℓth partial path cost (metric)* $M^{\ell}(\boldsymbol{P}_m)$ for a path $\boldsymbol{P}_m$ is obtained by summing the branch metrics for the first ℓ branches of the path, where node m is the ending node of $\boldsymbol{P}_m$.

7. If $\boldsymbol{P}_m$ has labels $\overline{v}_0, \overline{v}_1, \ldots, \overline{v}_\ell$, then

$$M^\ell(\boldsymbol{P}_m) = \sum_{i=0}^{\ell} \left(\phi_i - (-1)^{\overline{v}_i}\right)^2$$

Viterbi Decoding Algorithm (Wolf Algorithm) (1)

1. Wolf algorithm [7] finds an optimal path by applying the Viterbi algorithm [6] to search through the trellis for $\boldsymbol{C}$.
2. In Wolf algorithm, each node in the trellis is assigned a cost. This cost is the partial path cost of the path starting at start node and ending at that node.
3. When more than one path enters a node at level ℓ in the trellis, one chooses the path which has the best (minimum) $(\ell-1)$th partial path cost as the cost of the node. The path with the best cost is the *survivor*.
4. The algorithm terminates when all nodes in the trellis have been labelled and their entering survivors determined.

5. Viterbi algorithm and Wolf algorithm are special types of dynamic programming, consequently, they are ML decoding algorithms.

6. This decoding algorithm uses a breadth-first search strategy to accomplish this search. That is, the nodes will be labelled level by level until the last level containing the goal node in the trellis.

7. The time and space complexities of Wolf algorithm are of $O(n \times \min(2^k, 2^{n-k}))$ [3], since it traverses the entire trellis.

8. Forney has given a procedure to reduce the number of states in the trellis for $\boldsymbol{C}$ [4] and now it becomes an active research area.

Viterbi Decoding Algorithm (Wolf Algorithm)(2)

Initialization: $\ell = 0$. Let $g_\ell(m)$ be the partial path cost assigned to node m at level ℓ. $g_\ell(m) = 0$.

Step 1: Compute the partial metrics for all paths entering each node at level ℓ.

Step 2: Set $g_\ell(m)$ equal to the best partial path cost entering the node m. Delete all non survivors from the trellis.

Step 3: If $\ell = n$, then stop and output the only survivor ending at the goal node; otherwise $\ell = \ell + 1$, step 1.

Viterbi Decoding Algorithm (Wolf Algorithm) (3)

Let

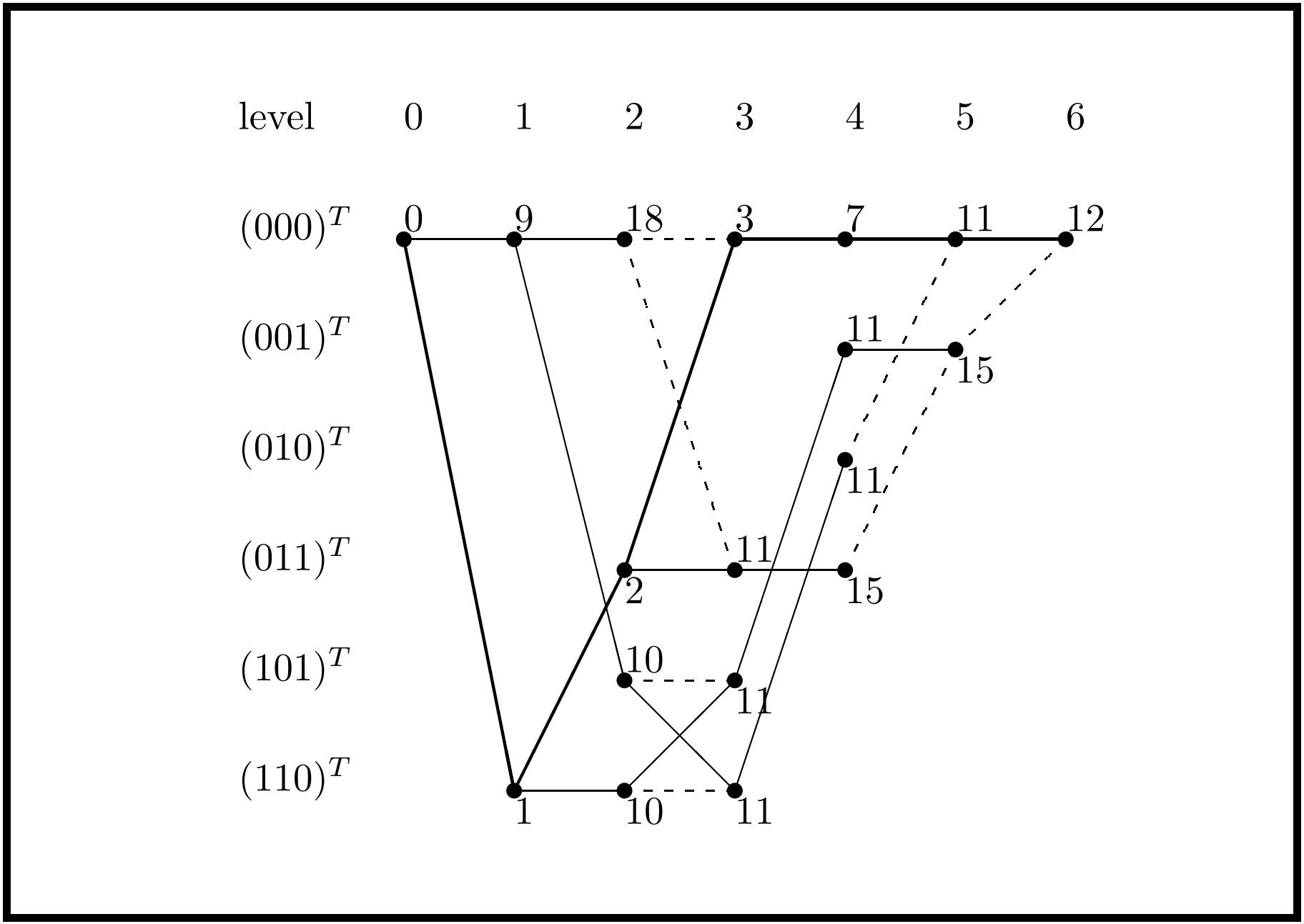
$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

be a generator matrix of $\boldsymbol{C}$ and let

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

be a parity-check matrix of $\boldsymbol{C}$. Assume the received vector is

$$\boldsymbol{\phi} = (-2, -2, -2, -1, -1, 0)$$


level 0 1 2 3 4 5 6
$(000)^T$ 0 9 18 3 7 11 12
$(001)^T$ 11 15
$(010)^T$ 11
$(011)^T$ 11 2 15
$(101)^T$ 10 11
$(110)^T$ 1 10 11

Maximum-Likelihood Decoding Rule (MLD Rule) for Symbol-by-Symbol Decoding (1)

1. The goal of decoding: for $0 \le i \le k-1$,

$$\text{set } \widehat{u_i} = s \text{ where}$$

$$Pr(u_i = s|\boldsymbol{r}) \ge Pr(u_i = 1 \oplus s|\boldsymbol{r}).$$

2.

$$\begin{aligned} Pr(u_i = s|\boldsymbol{r}) &= \sum_{\boldsymbol{c} \in T_i^s} Pr(\boldsymbol{c}|\boldsymbol{r}) \\ &= \sum_{\boldsymbol{c} \in T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}) \frac{Pr(\boldsymbol{c})}{Pr(\boldsymbol{r})}, \end{aligned}$$

where $T_i^s = \{c \in \boldsymbol{C} | c = (u_0, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_{k-1})G\}$.

3. If all codewords of $\boldsymbol{C}$ have equal probability of being

transmitted, then

$$\begin{aligned} & Pr(u_i = s|\boldsymbol{r}) \geq Pr(u_i = 1 \oplus s|\boldsymbol{r}) \\ \Longleftrightarrow \quad & \sum_{\boldsymbol{c} \in T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}) \frac{Pr(\boldsymbol{c})}{Pr(\boldsymbol{r})} \geq \sum_{\boldsymbol{c} \in C \setminus T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}) \frac{Pr(\boldsymbol{c})}{Pr(\boldsymbol{r})} \\ \Longleftrightarrow \quad & \sum_{\boldsymbol{c} \in T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}) \geq \sum_{\boldsymbol{c} \in C \setminus T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}). \end{aligned}$$

4. A maximum-likelihood decoding rule (**MLD** rule), which minimizes error probability when each codeword is transmitted equiprobably, decodes a received vector $\boldsymbol{r}$ to an information sequence $\boldsymbol{u} = (u_0, \ldots, u_{k-1})$ such that

$$\sum_{\boldsymbol{c} \in T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}) \geq \sum_{\boldsymbol{c} \in C \setminus T_i^s} Pr(\boldsymbol{r}|\boldsymbol{c}).$$

Maximum-Likelihood Decoding Rule (MLD Rule) for Symbol-by-Symbol Decoding (2)

1. For a time-discrete memoryless channel, the **MLD** rule can be formulated as: for $0 \le i \le k-1,\ s \in \{0,1\}$,

$$\text{set } \ \widehat{u_i} = s \text{ where}$$

$$\sum_{\boldsymbol{c} \in T_i^s} \prod_{j=0}^{n-1} Pr(r_j|c_j) \ge \sum_{\boldsymbol{c} \in C \setminus T_i^s} \prod_{j=0}^{n-1} Pr(r_j|c_j).$$

2. Define the bit likelihood ratio of r_j as

$$\phi_j = \frac{Pr(r_j|0)}{Pr(r_j|1)}.$$

3. Assume that $\prod_{j=0}^{n-1} Pr(r_j|1) \neq 0$. Then we can rewrite the above

inequality as

$$\sum_{\boldsymbol{c}\in T_i^s}\prod_{j=0}^{n-1} Pr(r_j|c_j) \geq \sum_{\boldsymbol{c}\in C\setminus T_i^s}\prod_{j=0}^{n-1} Pr(r_j|c_j)$$

$$\Longleftrightarrow \quad \sum_{\boldsymbol{c}\in T_i^s}\prod_{j=0}^{n-1} \phi_j^{c_j\oplus 1} \geq \sum_{\boldsymbol{c}\in C\setminus T_i^s}\prod_{j=0}^{n-1} \phi_j^{c_j\oplus 1}$$

Binary Convolutional Codes

1. A binary convolutional code is denoted by a three-tuple (n, k, m).
2. n output bits are generated whenever k input bits are received.
3. The current n outputs are linear combinations of the present k input bits and the previous $m \times k$ input bits.
4. m designates the number of previous k-bit input blocks that must be memorized in the encoder.
5. m is called the *memory order* of the convolutional code.

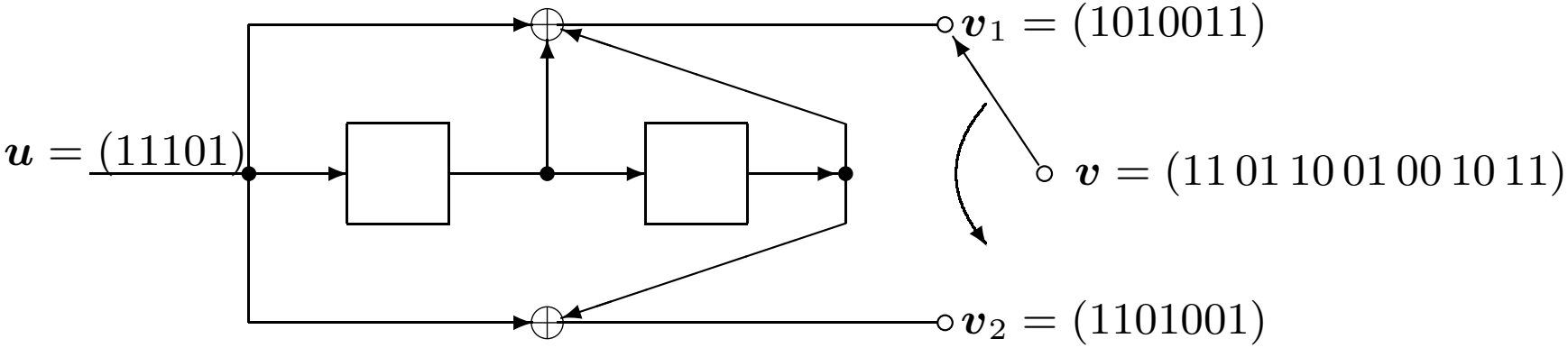
Encoders for the Convolutional Codes

1. A binary convolutional encoder is conveniently structured as a mechanism of shift registers and modulo-2 adders, where the output bits are modular-2 additions of selective shift register contents and present input bits.
2. n in the three-tuple notation is exactly the number of output sequences in the encoder.
3. k is the number of input sequences (and hence, the encoder consists of k shift registers).
4. m is the maximum length of the k shift registers (i.e., if the number of stages of the jth shift register is K_j, then $m = \max_{1 \le j \le k} K_j$).
5. $K = \sum_{j=1}^{k} K_j$ is the total memory in the encoder (K is sometimes called the *overall constraint lengths*).

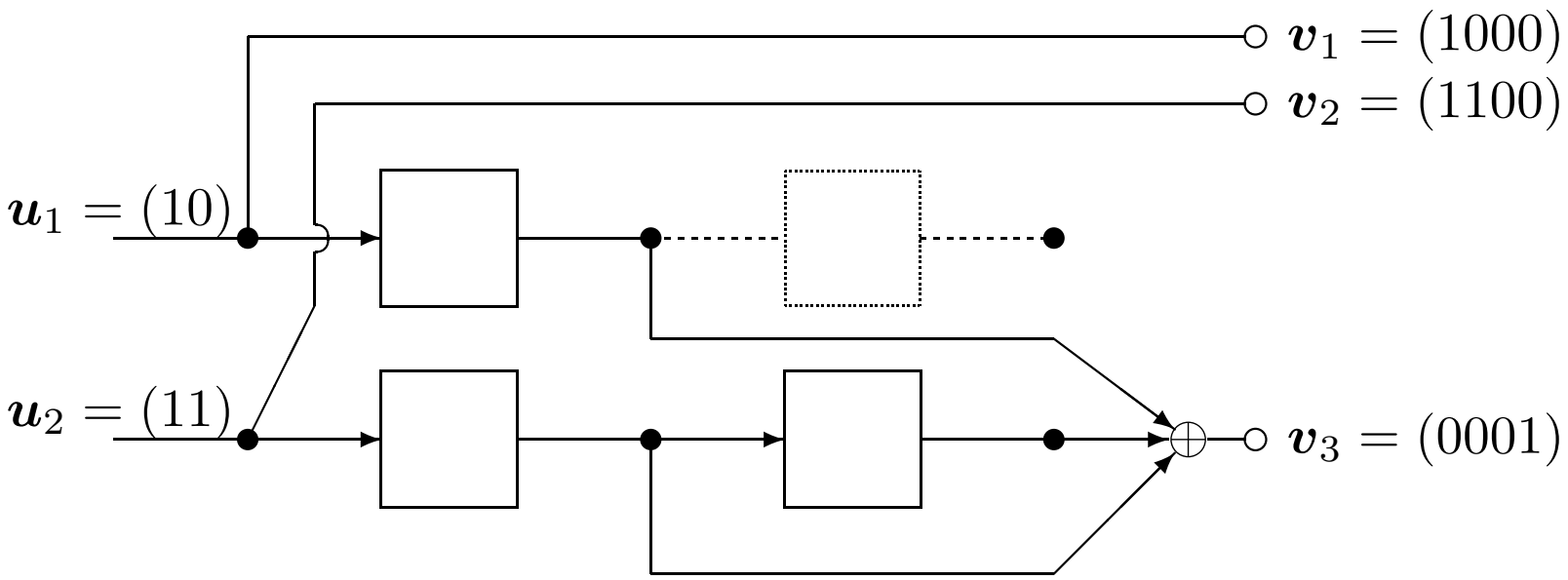
6. The definition of constraint length of a convolutional code is defined in several ways. The most popular one is $m + 1$.

Encoder for the Binary $(2, 1, 2)$ Convolutional Code

$\boldsymbol{u}$ is the information sequence and $\boldsymbol{v}$ is the corresponding code sequence (codeword).



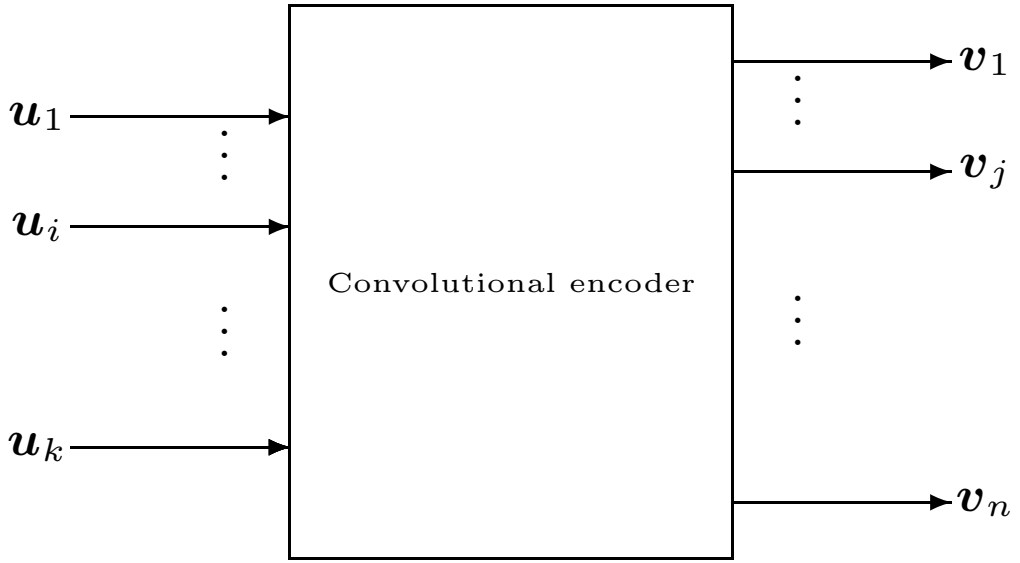
Encoder for the Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010\ 000\ 001)$

Impose Response and Convolution



1. The encoders of convolutional codes can be represented by *linear time-invariant* (LTI) systems.

2.

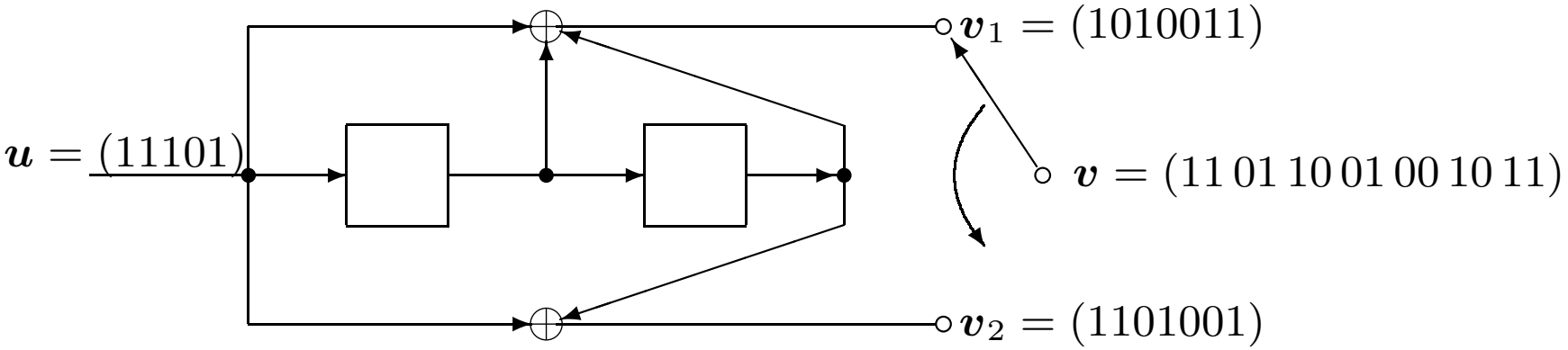
$$\boldsymbol{v}_j = \boldsymbol{u}_1 * \boldsymbol{g}_j^{(1)} + \boldsymbol{u}_2 * \boldsymbol{g}_j^{(2)} + \cdots + \boldsymbol{u}_k * \boldsymbol{g}_j^{(k)} = \sum_{i=1}^{k} \boldsymbol{u}_i * \boldsymbol{g}_j^{(k)},$$

where $*$ is the convolutional operation and $g_j^{(i)}$ is the impulse response of the ith input sequence with the response to the jth output.

3. $\boldsymbol{g}_j^{(i)}$ can be found by stimulating the encoder with the discrete impulse $(1, 0, 0, \ldots)$ at the ith input and by observing the jth output when all other inputs are fed the zero sequence $(0, 0, 0, \ldots)$.

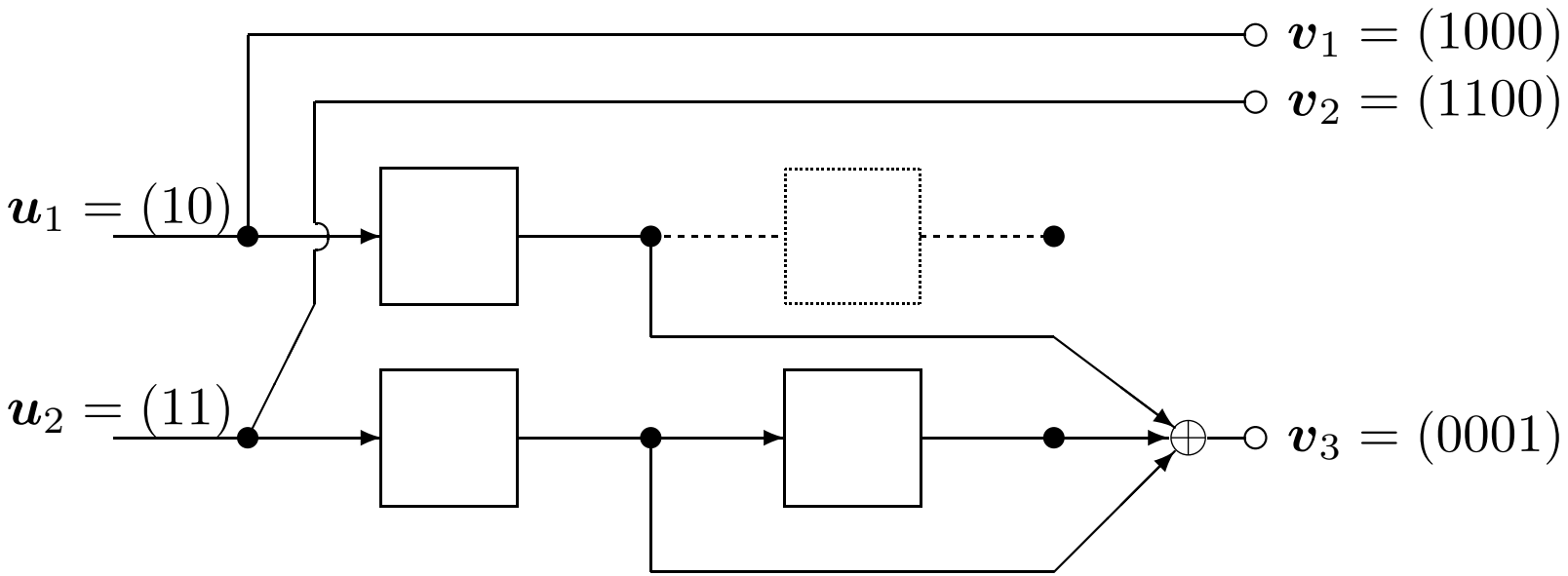
4. The impulse responses are called *generator sequences* of the encoder.

Impulse Response for the Binary $(2, 1, 2)$ Convolutional Code



$$\begin{aligned}
\boldsymbol{g}_1 &= (1,1,1,0,\ldots) = (1,1,1),\ \boldsymbol{g}_2 = (1,0,1,0,\ldots) = (1,0,1) \\
\boldsymbol{v}_1 &= (1,1,1,0,1) * (1,1,1) = (1,0,1,0,0,1,1) \\
\boldsymbol{v}_2 &= (1,1,1,0,1) * (1,0,1) = (1,1,0,1,0,0,1)
\end{aligned}$$

Impulse Response for the $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010\ 000\ 001)$

$\boldsymbol{g}_1^{(1)} = 4$ (octal)$= (1, 0, 0)$, $\boldsymbol{g}_1^{(2)} = 0$ (octal), $\boldsymbol{g}_2^{(1)} = 0$ (octal), $\boldsymbol{g}_2^{(2)} = 4$ (octal), $\boldsymbol{g}_3^{(1)} = 2$ (octal), $\boldsymbol{g}_3^{(2)} = 3$ (octal)

Generator Matrix in the Time Domain

1. The convolutional codes can be generated by a generator matrix multiplied by the information sequences.
2. Let $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k$ are the information sequences and $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ the output sequences.
3. Arrange the information sequences as

$$\begin{aligned} \boldsymbol{u} &= (u_{1,0}, u_{2,0}, \ldots, u_{k,0}, u_{1,1}, u_{2,1}, \ldots, u_{k,1}, \\ &\quad \ldots, u_{1,\ell}, u_{2,\ell}, \ldots, u_{k,\ell}, \ldots) \\ &= (\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_\ell, \ldots), \end{aligned}$$

and the output sequences as

$$\begin{aligned} \boldsymbol{v} &= (v_{1,0}, v_{2,0}, \ldots, v_{n,0}, v_{1,1}, v_{2,1}, \ldots, v_{n,1}, \\ &\quad \ldots, v_{1,\ell}, v_{2,\ell}, \ldots, v_{n,\ell}, \ldots) \\ &= (\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_\ell, \ldots) \end{aligned}$$

4. $\boldsymbol{v}$ is called a codeword or code sequence.

5. The relation between $\boldsymbol{v}$ and $\boldsymbol{u}$ can characterized as

$$\boldsymbol{v} = \boldsymbol{u} \cdot \boldsymbol{G},$$

where $\boldsymbol{G}$ is the generator matrix of the code.

6. The generator matrix is

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \boldsymbol{G}_2 & \cdots & \boldsymbol{G}_m & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m & \\ & & \boldsymbol{G}_0 & \cdots & \boldsymbol{G}_{m-2} & \boldsymbol{G}_{m-1} & \boldsymbol{G}_m \\ & & & \ddots & & & \ddots \end{pmatrix},$$

with the $k \times n$ submatrices

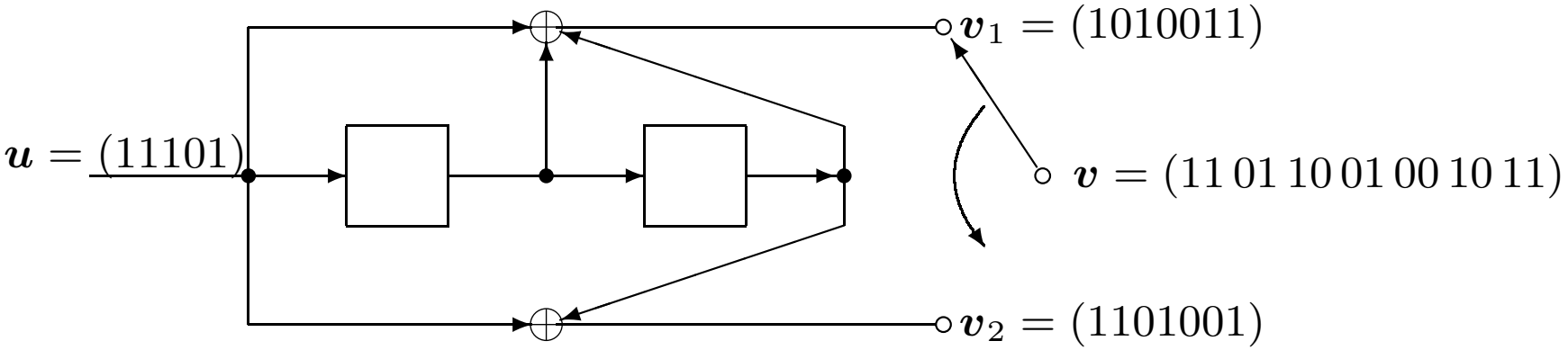
$$\boldsymbol{G}_\ell = \begin{pmatrix} g_{1,\ell}^{(1)} & g_{2,\ell}^{(1)} & \cdots & g_{n,\ell}^{(1)} \\ g_{1,\ell}^{(2)} & g_{2,\ell}^{(2)} & \cdots & g_{n,\ell}^{(2)} \\ \vdots & \vdots & & \vdots \\ g_{1,\ell}^{(k)} & g_{2,\ell}^{(k)} & \cdots & g_{n,\ell}^{(k)} \end{pmatrix}.$$

7. The element $g_{j,\ell}^{(i)}$, for $i \in [1, k]$ and $j \in [1, n]$, are the impulse

response of the ith input with respect to jth output:

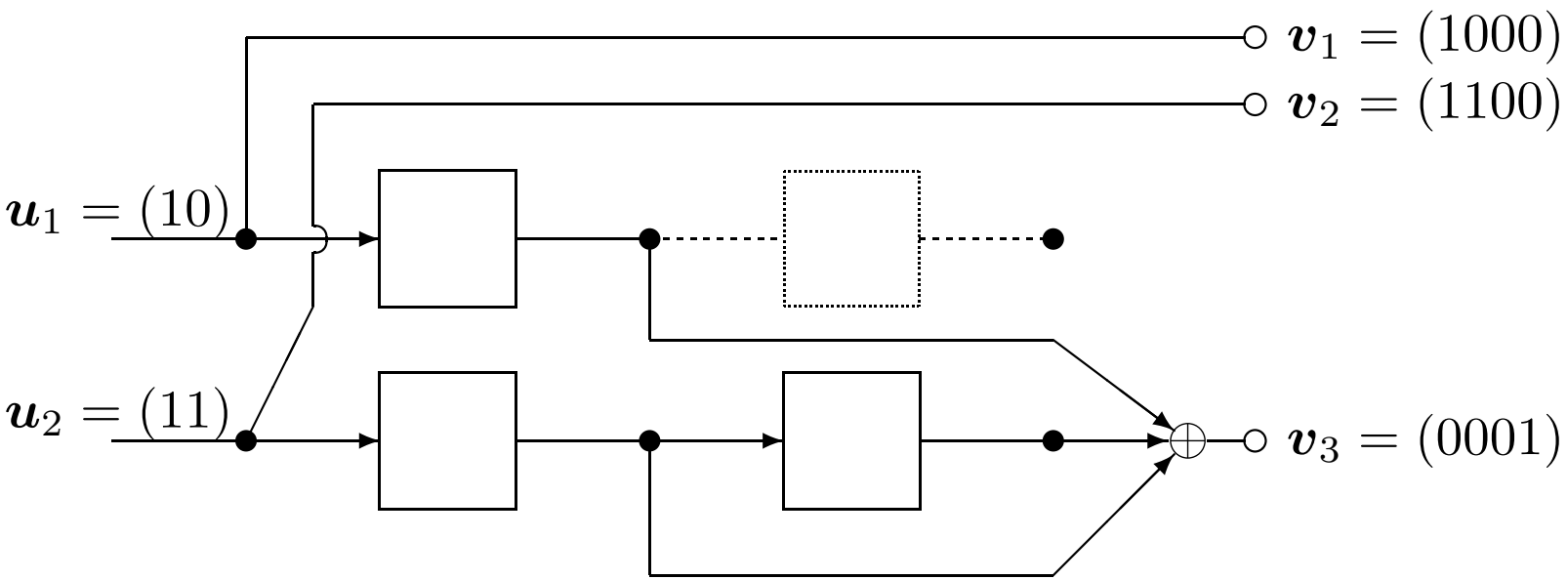
$$\boldsymbol{g}_j^{(i)} = (g_{j,0}^{(i)}, g_{j,1}^{(i)}, \ldots, g_{j,\ell}^{(i)}, \ldots, g_{j,m}^{(i)}).$$

Generator Matrix of the Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G} \\
&= (1,1,1,0,1) \cdot \begin{pmatrix}
11 & 10 & 11 & & & & \\
 & 11 & 10 & 11 & & & \\
 & & 11 & 10 & 11 & & \\
 & & & 11 & 10 & 11 & \\
 & & & & 11 & 10 & 11
\end{pmatrix} \\
&= (11,01,10,01,00,10,11)
\end{aligned}
$$

Generator Matrix of the Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$ $\boldsymbol{v} = (110\ 010\ 000\ 001)$

$\boldsymbol{g}_1^{(1)} = 4$ (octal)$= (1, 0, 0)$, $\boldsymbol{g}_1^{(2)} = 0$ (octal), $\boldsymbol{g}_2^{(1)} = 0$ (octal), $\boldsymbol{g}_2^{(2)} = 4$

(octal), $\boldsymbol{g}_3^{(1)} = 2$ (octal), $\boldsymbol{g}_3^{(2)} = 3$ (octal)

$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G} \\
&= (11, 01) \cdot \begin{pmatrix} 100 & 001 & 000 & \\ 010 & 001 & 001 & \\ & 100 & 001 & 000 \\ & 010 & 001 & 001 \end{pmatrix} \\
&= (110, 010, 000, 001)
\end{aligned}
$$

Generator Matrix in the Z Domain

1. According to the Z transform,

$$\boldsymbol{u}_i \quad \multimap \quad U_i(D) = \sum_{t=0}^{\infty} u_{i,t} D^t$$

$$\boldsymbol{v}_j \quad \multimap \quad V_j(D) = \sum_{t=0}^{\infty} v_{j,t} D^t$$

$$\boldsymbol{g}_j^{(i)} \quad \multimap \quad G_{i,j}(D) = \sum_{t=0}^{\infty} g_{j,t}^{(i)} D^t$$

2. The convolutional relation of the Z transform $Z\{\boldsymbol{u} * \boldsymbol{g}\} = U(D)G(D)$ is used to transform the convolution of input sequences and generator sequences to a multiplication in the Z domain.

3. $V_j(D) = \sum_{i=1}^{k} U_i(D) \cdot G_{i,j}(D)$.

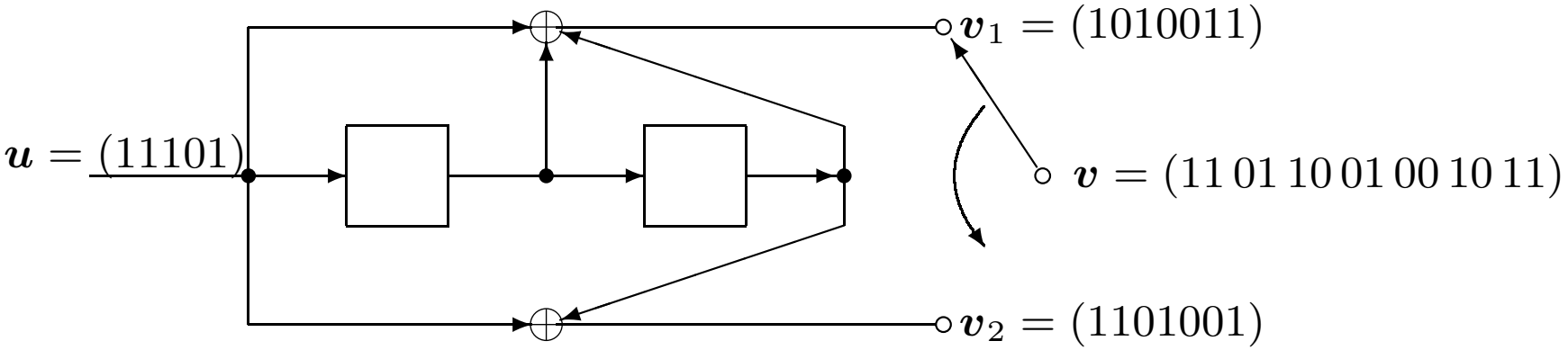
4. We can write the above equations into a matrix multiplication:

$$\boldsymbol{V}(D) = \boldsymbol{U}(D) \cdot \boldsymbol{G}(D),$$

where

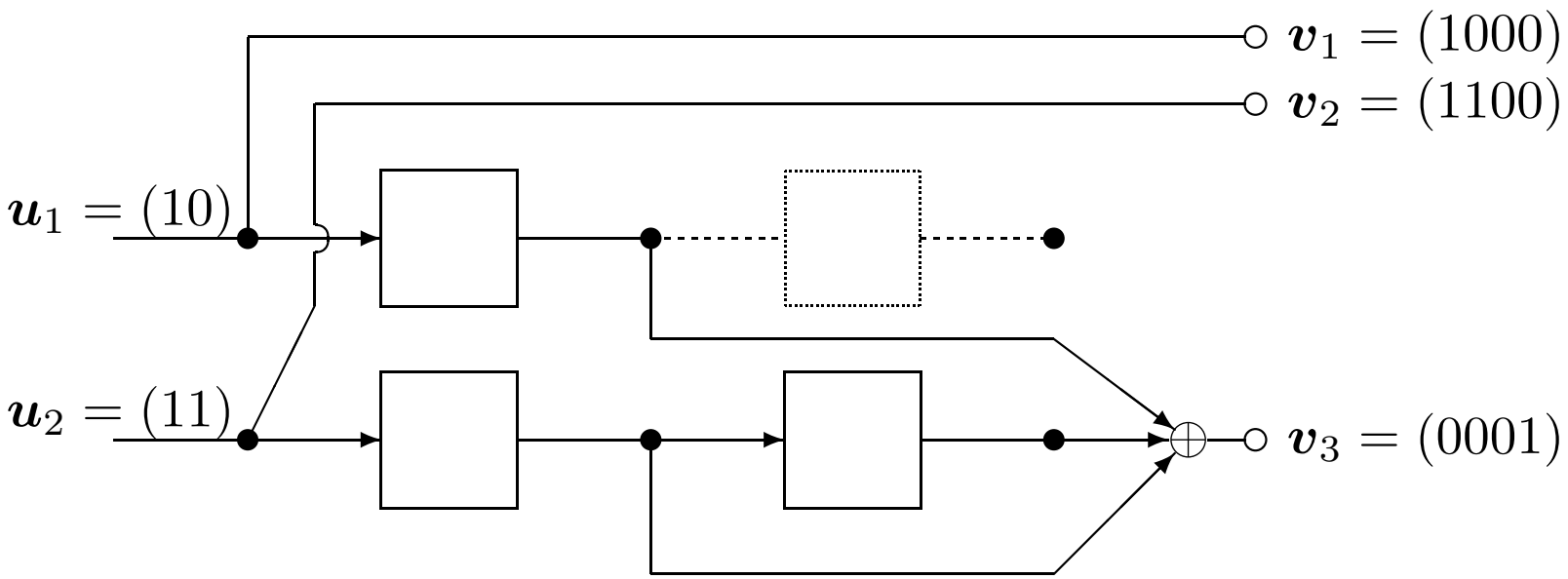
$$\begin{aligned} \boldsymbol{U}(D) &= (U_1(D), U_2(D), \ldots, U_k(D)) \\ \boldsymbol{V}(D) &= (V_1(D), V_2(D), \ldots, V_n(D)) \\ \boldsymbol{G}(D) &= \begin{pmatrix} & G_{i,j} & \\ & & \end{pmatrix} \end{aligned}$$

Generator Matrix of the Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{g}_1 &= (1,1,1,0,\ldots) = (1,1,1), \\
\boldsymbol{g}_2 &= (1,0,1,0,\ldots) = (1,0,1) \\
G_{1,1}(D) &= 1 + D + D^2 \\
G_{1,2}(D) &= 1 + D^2 \\
U_1(D) &= 1 + D + D^2 + D^4 \\
V_1(D) &= 1 + D^2 + D^5 + D^6 \\
V_2(D) &= 1 + D + D^3 + D^6
\end{aligned}
$$

Generator Matrix of the Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010\ 000\ 001)$

$$
\begin{aligned}
\boldsymbol{g}_1^{(1)} &= (1,0,0),\ \boldsymbol{g}_1^{(2)} = (0,0,0),\ \boldsymbol{g}_2^{(1)} = (0,0,0) \\
\boldsymbol{g}_2^{(2)} &= (1,0,0),\ \boldsymbol{g}_3^{(1)} = (0,1,0),\ \boldsymbol{g}_3^{(2)} = (0,1,1) \\
G_{1,1}(D) &= 1,\ G_{1,2}(D) = 0,\ G_{1,3}(D) = D \\
G_{2,1}(D) &= 0,\ G_{2,2} = 1,\ G_{2,3}(D) = D + D^2 \\
U_1(D) &= 1,\ U_2(D) = 1 + D \\
V_1(D) &= 1,\ V_2(D) = 1 + D,\ V_3(D) = D^3
\end{aligned}
$$

Termination

1. The *effective code rate*, $R_{\text{effective}}$, is defined as the average number of input bits carried by an output bit.
2. In practice, the input sequences are with finite length.
3. In order to terminate a convolutional code, some bits are appended onto the information sequence such that the shift registers return to the zero.
4. Each of the k input sequences of length L bits is padded with m zeros, and these k input sequences jointly induce $n(L+m)$ output bits.
5. The effective rate of the terminated convolutional code is now

$$R_{\text{effective}} = \frac{kL}{n(L+m)} = R\frac{L}{L+m},$$

where $\frac{L}{L+m}$ is called the *fractional rate loss.*

6. When L is large, $R_{\text{effective}} \approx R$.

7. All examples presented are terminated convolutional codes.

Truncation

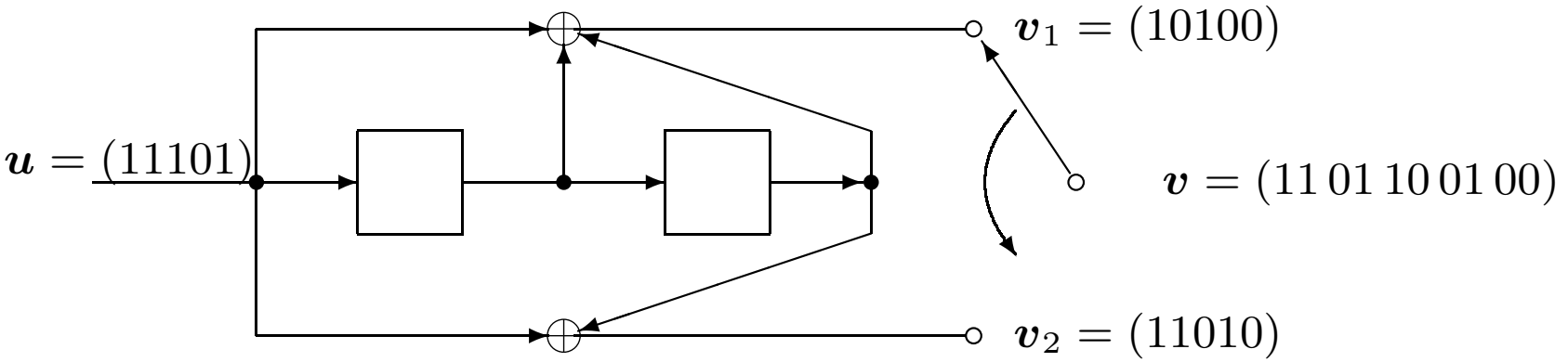
1. The second option to terminate a convolutional code is to stop for $t > L$ no matter what contents of shift registers have.
2. The effective code rate is still R.
3. The generator matrix is clipped after the Lth column:

$$\boldsymbol{G}_{[L]}^{c} = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & \\ & & \ddots & & \ddots & & \\ & & & \boldsymbol{G}_0 & & & \boldsymbol{G}_m \\ & & & & \ddots & & \vdots \\ & & & & & \boldsymbol{G}_0 & \boldsymbol{G}_1 \\ & & & & & & \boldsymbol{G}_0 \end{pmatrix},$$

where $\boldsymbol{G}_{[L]}^{c}$ is an $(k \cdot L) \times (n \cdot L)$ matrix.

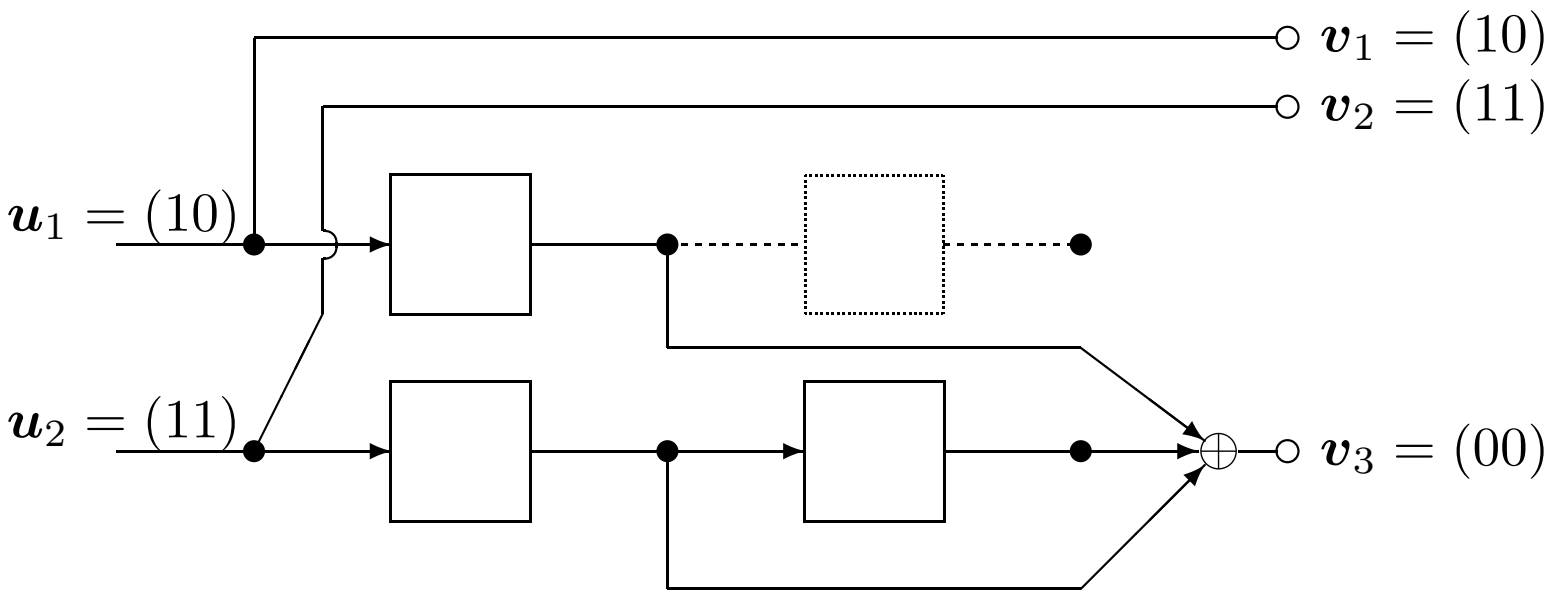
4. The drawback of truncation method is that the last few blocks of information sequences are less protected.

Generator Matrix of the Truncation Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G}^{c}_{[5]} \\
&= (1,1,1,0,1) \cdot \begin{pmatrix}
11 & 10 & 11 & & \\
 & 11 & 10 & 11 & \\
 & & 11 & 10 & 11 \\
 & & & 11 & 10 \\
 & & & & 11
\end{pmatrix} \\
&= (11, 01, 10, 01, 00)
\end{aligned}
$$

Generator Matrix of the Truncation Binary $(3, 2, 2)$ Convolutional Code



$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010)$

$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \boldsymbol{G}^{c}_{[2]} \\
&= (11, 01) \cdot \begin{pmatrix} 100 & 001 \\ 010 & 001 \\ & 100 \\ & 010 \end{pmatrix} \\
&= (110, 010)
\end{aligned}
$$

Tail Biting

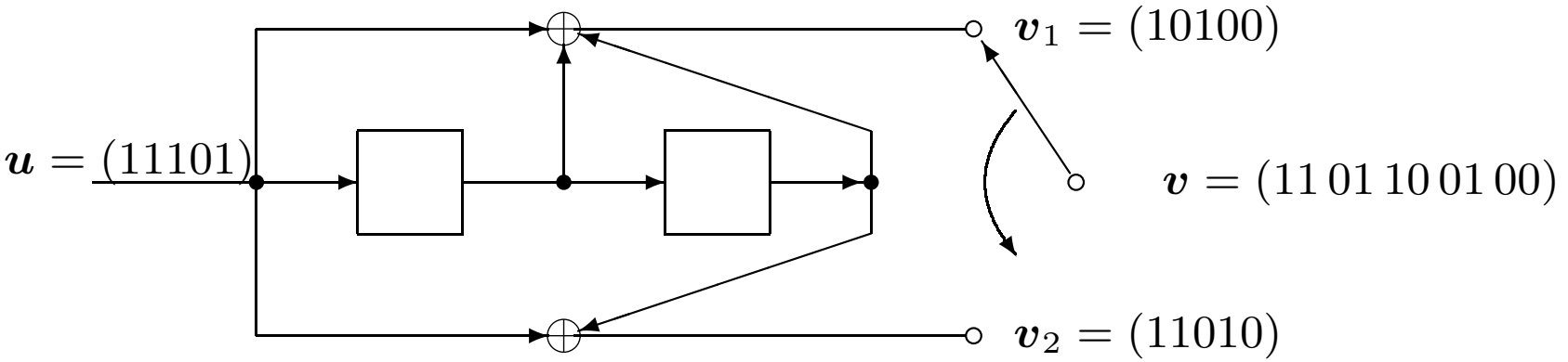
1. The third possible method to generate finite code sequences is called tail biting.
2. Tail biting is to start the convolutional encoder in the same contents of all shift registers (state) where it will stop after the input of L information blocks.
3. Equal protection of all information bits of the entire information sequences is possible.
4. The effective rate of the code is still R.
5. The generator matrix has to be clipped after the Lth column and

manipulated as follows:

$$\tilde{\boldsymbol{G}}^{c}_{[L]} = \begin{pmatrix} \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & \\ & \boldsymbol{G}_0 & \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & \\ & & \ddots & & \ddots & & \\ & & & \boldsymbol{G}_0 & & & \boldsymbol{G}_m \\ \boldsymbol{G}_m & & & & \ddots & & \vdots \\ \vdots & \ddots & & & & \boldsymbol{G}_0 & \boldsymbol{G}_1 \\ \boldsymbol{G}_1 & \cdots & \boldsymbol{G}_m & & & & \boldsymbol{G}_0 \end{pmatrix},$$

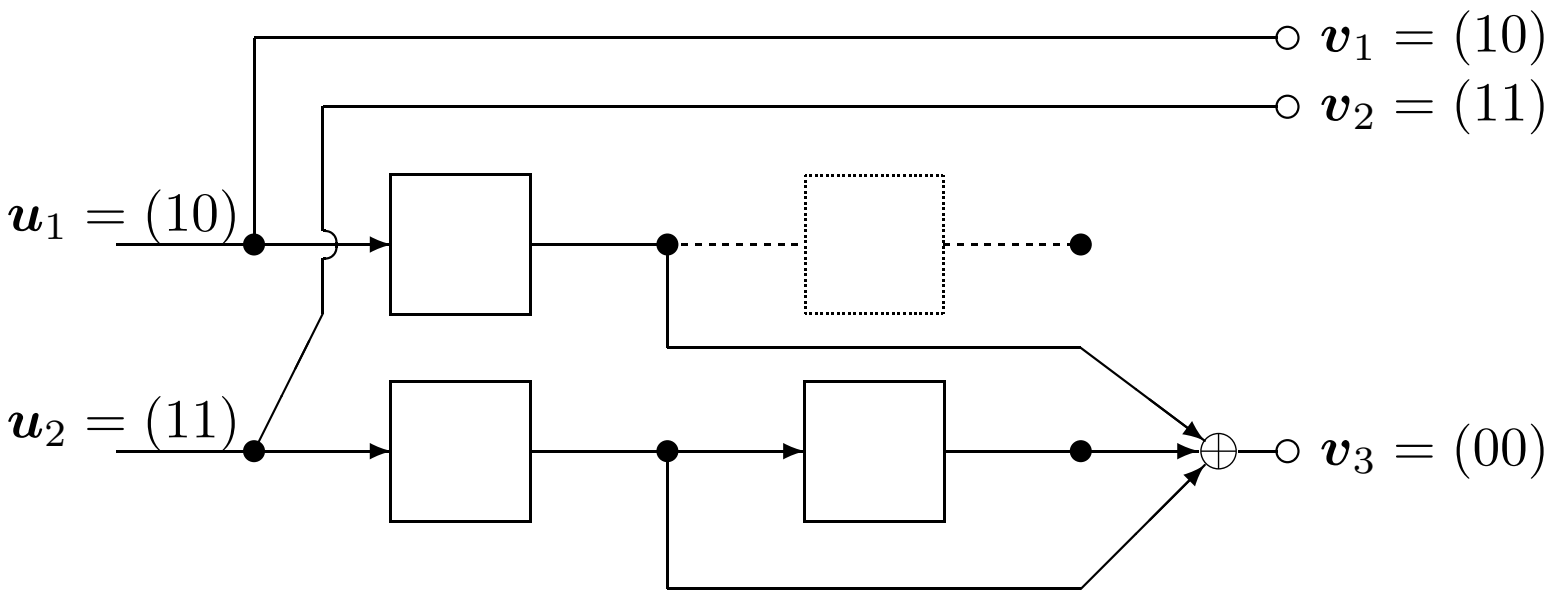
where $\tilde{\boldsymbol{G}}^{c}_{[L]}$ is an $(k \cdot L) \times (n \cdot L)$ matrix.

Generator Matrix of the Tail Biting Binary $(2, 1, 2)$ Convolutional Code



$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \tilde{\boldsymbol{G}}^{c}_{[5]} \\
&= (1,1,1,0,1) \cdot \begin{pmatrix}
11 & 10 & 11 & & \\
 & 11 & 10 & 11 & \\
 & & 11 & 10 & 11 \\
11 & & & 11 & 10 \\
10 & 11 & & & 11
\end{pmatrix} \\
&= (01,10,10,01,00)
\end{aligned}
$$

Generator Matrix of the Tail Biting Binary $(3, 2, 2)$ Convolutional Code

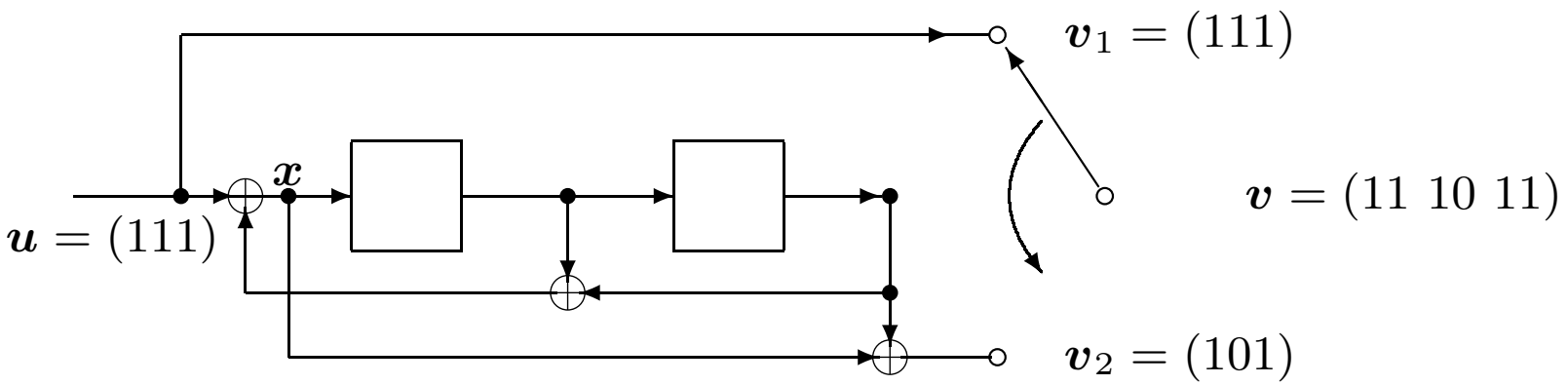


$\boldsymbol{u} = (11\ 01)$

$\boldsymbol{v} = (110\ 010)$

$$
\begin{aligned}
\boldsymbol{v} &= \boldsymbol{u} \cdot \tilde{\boldsymbol{G}}^{c}_{[2]} \\
&= (11, 01) \cdot \begin{pmatrix} 100 & 001 \\ 010 & 001 \\ 001 & 100 \\ 001 & 010 \end{pmatrix} \\
&= (111, 010)
\end{aligned}
$$

FIR and IIR systems



1. All examples in previous slides are *finite impulse response* (FIR) systems that are with finite impulse responses.
2. The above example is an *infinite impulse response* (IIR) system that is with infinite impulse response.

3. The generator sequences of the above example are

$$\begin{aligned} g_1^{(1)} &= (1) \\ g_2^{(1)} &= (1,1,1,0,1,1,0,1,1,0,\ldots). \end{aligned}$$

4. The infinite sequence of $g_2^{(1)}$ is caused by the recursive structure of the encoder.

5. By introducing the variable $\boldsymbol{x}$, we have

$$\begin{aligned} x_t &= u_t + x_{t-1} + x_{t-2} \\ v_{2,t} &= x_t + x_{t-2}. \end{aligned}$$

Accordingly, we can have the following difference equations:

$$\begin{aligned} v_{1,t} &= u_t \\ v_{2,t} + v_{2,t-1} + v_{2,t-2} &= u_t + u_{t-2} \end{aligned}$$

6. We then apply the z transform to the second equation:

$$\begin{aligned}\sum_{t=0}^{\infty} v_{2,t}D^t + \sum_{t=0}^{\infty} v_{2,t-1}D^t + \sum_{t=0}^{\infty} v_{2,t-2}D^t &= \sum_{t=0}^{\infty} u_t D^t + \sum_{t=0}^{\infty} u_{t-2}D^t, \\ V_2(D) + DV_2(D) + D^2V_2(D) &= U(D) + D^2U(D).\end{aligned}$$

7. The system function is then

$$V_2(D) = \frac{1+D^2}{1+D+D^2}U(D) = G_{12}(D)U(D).$$

8. The generator matrix is obtained:

$$\boldsymbol{G}(D) = \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix}.$$

9.

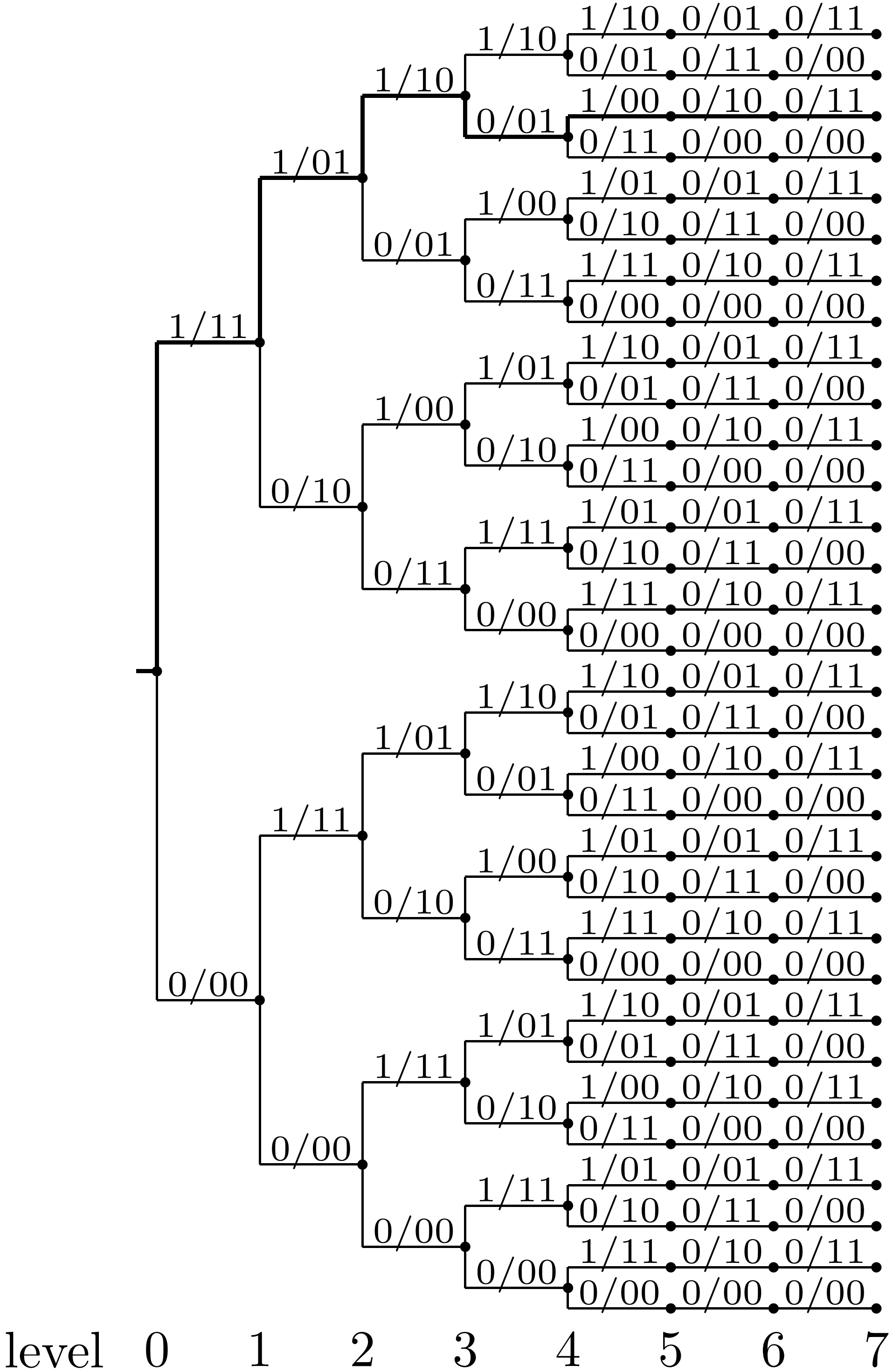
$$\begin{aligned} \boldsymbol{V}(D) &= \boldsymbol{U}(D) \cdot \boldsymbol{G}(D) \\ &= \left(1 + D + D^2\right) \begin{pmatrix} 1 & \frac{1+D^2}{1+D+D^2} \end{pmatrix} \\ &= \left(1 + D + D^2 \; 1 + D^2\right) \end{aligned}$$

10. The time diagram:

t	$\boldsymbol{\sigma}_t$	$\boldsymbol{\sigma}_{t+1}$	$\boldsymbol{v}_1$	$\boldsymbol{v}_2$
0	00	10	1	1
1	10	01	1	0
2	01	00	1	1

Code Trees of Convolutional Codes

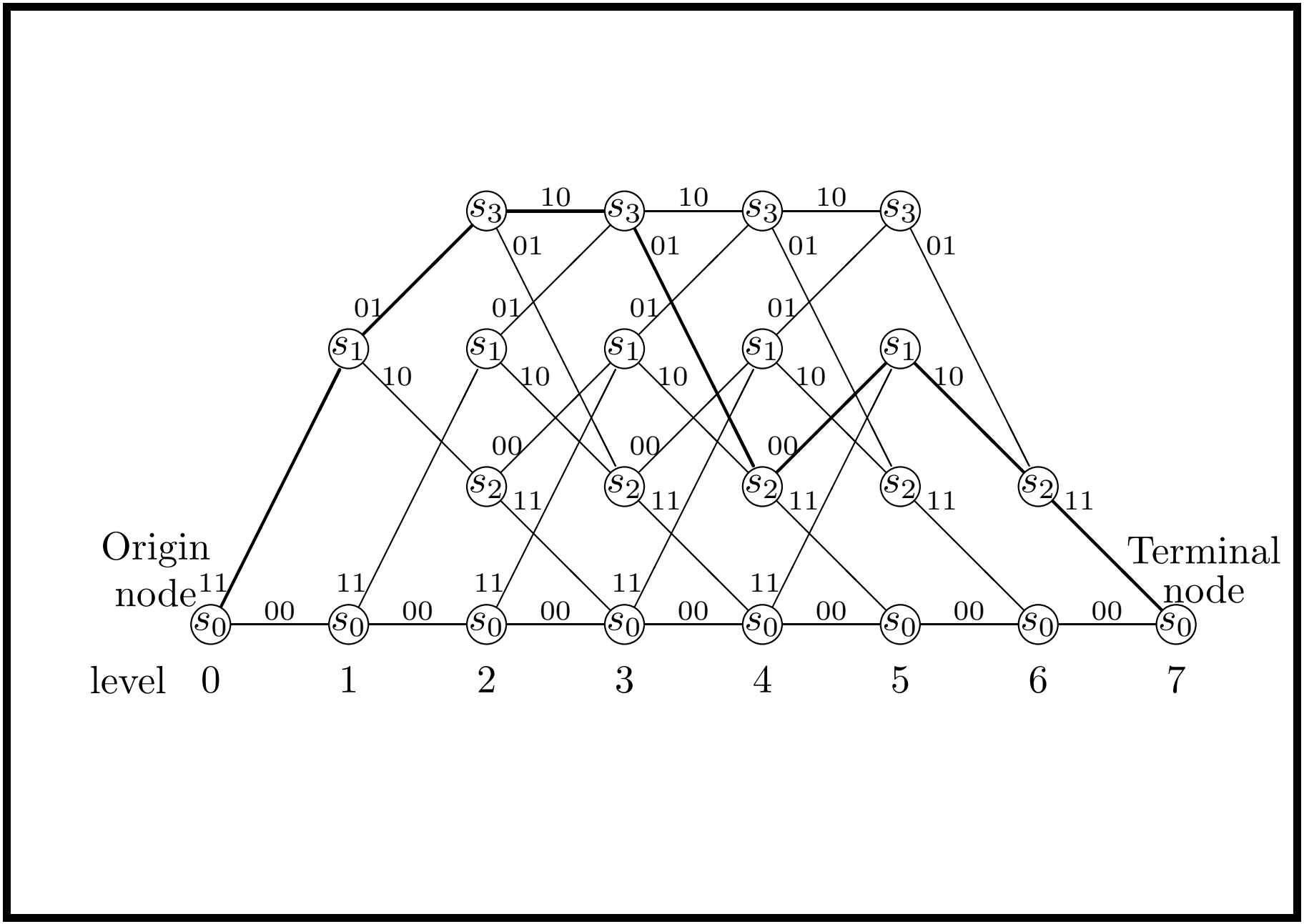
1. A *code tree* of a binary (n, k, m) convolutional code presents every codeword as a path on a tree.
2. For input sequences of length L bits, the code tree consists of $(L + m + 1)$ levels. The single leftmost node at level 0 is called the *origin node*.
3. At the first L levels, there are exactly 2^k branches leaving each node. For those nodes located at levels L through $(L + m)$, only one branch remains. The 2^{kL} rightmost nodes at level $(L + m)$ are called the *terminal nodes*.
4. As expected, a path from the single origin node to a terminal node represents a codeword; therefore, it is named the *code path* corresponding to the codeword.



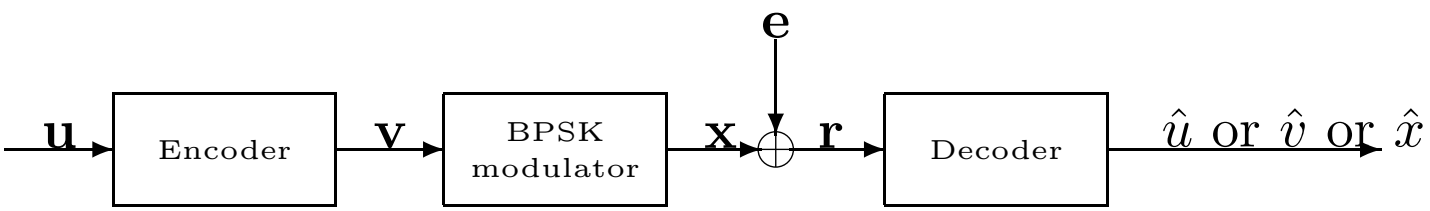
1/10
1/10 0/01 0/11
0/01 0/11 0/00
1/10
0/01
1/00 0/10 0/11
0/11 0/00 0/00
1/01
1/00
1/01 0/01 0/11
0/10 0/11 0/00
0/01
0/11
1/11 0/10 0/11
0/00 0/00 0/00
1/11
1/01
1/10 0/01 0/11
0/01 0/11 0/00
1/00
0/10
1/00 0/10 0/11
0/11 0/00 0/00
0/10
1/11
1/01 0/01 0/11
0/10 0/11 0/00
0/11
0/00
1/11 0/10 0/11
0/00 0/00 0/00
1/10
1/10 0/01 0/11
0/01 0/11 0/00
1/01
0/01
1/00 0/10 0/11
0/11 0/00 0/00
1/11
1/00
1/01 0/01 0/11
0/10 0/11 0/00
0/10
0/11
1/11 0/10 0/11
0/00 0/00 0/00
0/00
1/01
1/10 0/01 0/11
0/01 0/11 0/00
1/11
0/10
1/00 0/10 0/11
0/11 0/00 0/00
0/00
1/11
1/01 0/01 0/11
0/10 0/11 0/00
0/00
0/00
1/11 0/10 0/11
0/00 0/00 0/00
level 0 1 2 3 4 5 6 7

Trellises of Convolutional Codes

1. a code *trellis* as termed by Forney is a structure obtained from a code tree by merging those nodes in the same *state*.
2. Recall that the *state* associated with a node is determined by the associated shift-register contents.
3. For a binary (n, k, m) convolutional code, the number of states at levels m through L is 2^K, where $K = \sum_{j=1}^{k} K_j$ and K_j is the length of the jth shift register in the encoder; hence, there are 2^K nodes on these levels.
4. Due to node merging, only one terminal node remains in a trellis.
5. Analogous to a code tree, a path from the single origin node to the single terminal node in a trellis also mirrors a codeword.


Origin node
Terminal node
level 0 1 2 3 4 5 6 7
s_0 s_1 s_2 s_3
00 01 10 11

System Model[a]



For an $(n, 1, m)$ convolutional code:

- $\mathbf{u} = (u_1, u_2, \ldots, u_L) \in \{0,1\}^L$ is the information bit sequence.[b]
- $\mathbf{v} = (v_1, v_2, \ldots, v_N) \in \{0,1\}^N$ is the code bit sequence.
- $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \{-1,1\}^N$ is the BPSK-modulated code bit sequence, where $x_j = (-1)^{v_j}$.
- $\mathbf{e} = (e_1, e_2, \ldots, e_N) \in \Re^N$ is a zero-mean i.i.d Gaussian vector

[a] All material of MAP algorithm are adapted from the lecture note by Prof. Po-Ning Chen

[b] The indices of all sequences start from 1.

(AWGN).

- $\mathbf{r} = (r_1, r_2, \ldots, r_N) \in \Re^N$ is the received vector, where $r_j = x_j + e_j$.
- $\hat{\mathbf{u}} = (\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_L) \in \{0,1\}^L$ is reconstructed information bit sequence.

BCJR (MAP) Algorithm (1)

1.

$$\begin{aligned}\Pr\{u_i = 0|\mathbf{r}\} &= \sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(0)}} \Pr\left\{ S_s^{i-1}, S_{\bar{s}}^i \middle| \mathbf{r} \right\} \\ &= \sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(0)}} \frac{\Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}\right\}}{\Pr\left\{\mathbf{r}\right\}},\end{aligned}$$

where $\mathcal{B}_i^{(0)}$ is the set of transition from nodes at level $i-1$ to nodes at level i, which corresponds to input $u_i = 0$.

$$\begin{aligned}\Pr\{u_i = 1|\mathbf{r}\} &= \sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(1)}} \Pr\left\{ S_s^{i-1}, S_{\bar{s}}^i \middle| \mathbf{r} \right\} \\ &= \sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(1)}} \frac{\Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}\right\}}{\Pr\left\{\mathbf{r}\right\}},\end{aligned}$$

where $\mathcal{B}_i^{(1)}$ is the set of transition from nodes at level $i-1$ to nodes at level i, which corresponds to input $u_i = 1$. Therefore,

$$\Lambda(i) = \log \frac{\displaystyle\sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(1)}} \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}\right\}}{\displaystyle\sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(0)}} \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}\right\}}$$

2. Define

$$\begin{aligned}
\alpha(S_{\bar{s}}^{i}) &\stackrel{\triangle}{=} \Pr\{S_{\bar{s}}^{i}, \mathbf{r}_1^{in}\} \\
\beta(S_{\bar{s}}^{i}) &\stackrel{\triangle}{=} \Pr\left\{ \mathbf{r}_{in+1}^{N} \middle| S_{\bar{s}}^{i} \right\} \\
\gamma(u, S_s^{i-1}, S_{\bar{s}}^{i}) &\stackrel{\triangle}{=} \Pr\left\{ u_i = u, S_{\bar{s}}^{i}, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1} \right\}, \quad u = 0, 1,
\end{aligned}$$

where $\mathbf{r}_a^b = (r_a, r_{a+1}, \ldots, r_b)$.

$$
\begin{aligned}
& \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}\right\} \\
= \quad & \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}_1^{(i-1)n}, \mathbf{r}_{(i-1)n+1}^{in}, \mathbf{r}_{in+1}^{N}\right\} \\
= \quad & \Pr\left\{\mathbf{r}_{in+1}^{N} \middle| S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}_1^{(i-1)n}, \mathbf{r}_{(i-1)n+1}^{in}\right\} \\
& \times \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}_1^{(i-1)n}, \mathbf{r}_{(i-1)n+1}^{in}\right\} \\
= \quad & \Pr\left\{\mathbf{r}_{in+1}^{N} \middle| S_{\bar{s}}^i\right\} \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}_1^{(i-1)n}, \mathbf{r}_{(i-1)n+1}^{in}\right\} \\
= \quad & \beta(S_{\bar{s}}^i) \cdot \Pr\left\{S_s^{i-1}, S_{\bar{s}}^i, \mathbf{r}_1^{(i-1)n}, \mathbf{r}_{(i-1)n+1}^{in}\right\} \\
= \quad & \beta(S_{\bar{s}}^i) \cdot \Pr\left\{S_{\bar{s}}^i, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1}, \mathbf{r}_1^{(i-1)n}\right\} \Pr\left\{S_s^{i-1}, \mathbf{r}_1^{(i-1)n}\right\} \\
= \quad & \Pr\left\{S_s^{i-1}, \mathbf{r}_1^{(i-1)n}\right\} \beta(S_{\bar{s}}^i) \Pr\left\{S_{\bar{s}}^i, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1}, \mathbf{r}_1^{(i-1)n}\right\} \\
= \quad & \Pr\left\{S_s^{i-1}, \mathbf{r}_1^{(i-1)n}\right\} \beta(S_{\bar{s}}^i) \Pr\left\{S_{\bar{s}}^i, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1}\right\} \\
= \quad & \alpha(S_s^{i-1}) \beta(S_{\bar{s}}^i) \sum_{u=0}^{1} \gamma(u, S_s^{i-1}, S_{\bar{s}}^i)
\end{aligned}
$$

3. **Derivation of α:** $\alpha(S_0^0) = 1$; $\alpha(S_1^0) = \alpha(S_2^0) = \alpha(S_3^0) = 0$.
   For $i = 1, \ldots, L$,

$$
\begin{aligned}
\alpha(S_{\overline{s}}^i) &\stackrel{\triangle}{=} \Pr\left\{S_{\overline{s}}^i, \mathbf{r}_1^{in}\right\} \\
&= \sum_{s=0}^{3} \Pr\left\{S_s^{i-1}, S_{\overline{s}}^i, \mathbf{r}_1^{in}\right\} \\
&= \sum_{s=0}^{3} \Pr\left\{S_s^{i-1}, S_{\overline{s}}^i, \mathbf{r}_1^{(i-1)n}, \mathbf{r}_{(i-1)n+1}^{in}\right\} \\
&= \sum_{s=0}^{3} \Pr\left\{S_s^{i-1}, \mathbf{r}_1^{(i-1)n}\right\} \Pr\left\{ S_{\overline{s}}^i, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1}, \mathbf{r}_1^{(i-1)n} \right\} \\
&= \sum_{s=0}^{3} \alpha(S_s^{i-1}) \Pr\left\{ S_{\overline{s}}^i, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1}, \mathbf{r}_1^{(i-1)n} \right\} \\
&= \sum_{s=0}^{3} \alpha(S_s^{i-1}) \Pr\left\{ S_{\overline{s}}^i, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1} \right\} \\
&= \sum_{s=0}^{3} \alpha(S_s^{i-1}) \sum_{u=0}^{1} \gamma(u, S_s^{i-1}, S_{\overline{s}}^i).
\end{aligned}
$$

4. **Derivation of β:** $\beta(S_0^L) = 1$; $\beta(S_1^L) = \beta(S_2^L) = \beta(S_3^L) = 0$.

For $i = L-1, \ldots, 0$,

$$
\begin{aligned}
\beta(S_{\bar{s}}^{i}) &\stackrel{\triangle}{=} \Pr\left\{ \mathbf{r}_{in+1}^{\bar{N}} \middle| S_{\bar{s}}^{i} \right\} \\
&= \sum_{s=0}^{3} \Pr\left\{ S_{s}^{i+1}, \mathbf{r}_{in+1}^{\bar{N}} \middle| S_{\bar{s}}^{i} \right\} \\
&= \sum_{s=0}^{3} \frac{\Pr\left\{ S_{\bar{s}}^{i}, S_{s}^{i+1}, \mathbf{r}_{in+1}^{\bar{N}} \right\}}{\Pr\left\{ S_{\bar{s}}^{i} \right\}} \\
&= \sum_{s=0}^{3} \frac{\Pr\left\{ S_{\bar{s}}^{i}, S_{s}^{i+1}, \mathbf{r}_{in+1}^{(i+1)n}, \mathbf{r}_{(i+1)n+1}^{\bar{N}} \right\}}{\Pr\left\{ S_{\bar{s}}^{i} \right\}} \\
&= \sum_{s=0}^{3} \frac{\Pr\left\{ \mathbf{r}_{(i+1)n+1}^{\bar{N}} \middle| S_{\bar{s}}^{i}, S_{s}^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \right\} \Pr\left\{ S_{\bar{s}}^{i}, S_{s}^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \right\}}{\Pr\left\{ S_{\bar{s}}^{i} \right\}} \\
&= \sum_{s=0}^{3} \frac{\Pr\left\{ \mathbf{r}_{(i+1)n+1}^{\bar{N}} \middle| S_{s}^{i+1} \right\} \Pr\left\{ S_{\bar{s}}^{i}, S_{s}^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \right\}}{\Pr\left\{ S_{\bar{s}}^{i} \right\}}
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{s=0}^{3} \frac{\beta(S_s^{i+1}) \Pr\left\{ S_{\bar{s}}^{i}, S_s^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \right\}}{\Pr\left\{ S_{\bar{s}}^{i} \right\}} \\
&= \sum_{s=0}^{3} \frac{\beta(S_s^{i+1}) \Pr\left\{ S_s^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \middle| S_{\bar{s}}^{i} \right\} \Pr\left\{ S_{\bar{s}}^{i} \right\}}{\Pr\left\{ S_{\bar{s}}^{i} \right\}} \\
&= \sum_{s=0}^{3} \beta(S_s^{i+1}) \Pr\left\{ S_s^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \middle| S_{\bar{s}}^{i} \right\} \\
&= \sum_{s=0}^{3} \beta(S_s^{i+1}) \sum_{u=0}^{1} \Pr\left\{ u, S_s^{i+1}, \mathbf{r}_{in+1}^{(i+1)n} \middle| S_{\bar{s}}^{i} \right\} \\
&= \sum_{s=0}^{3} \beta(S_s^{i+1}) \sum_{u=0}^{1} \gamma(u, S_{\bar{s}}^{i}, S_s^{i+1}).
\end{aligned}
$$

5. **Derivation of γ:** $\beta(S_0^L) = 1$; $\beta(S_1^L) = \alpha(S_2^L) = \alpha(S_3^L) = 0$.

$$
\begin{aligned}
\gamma(u, S_s^{i-1}, S_{\bar{s}}^{i}) &\triangleq \Pr\left\{ u, S_{\bar{s}}^{i}, \mathbf{r}_{(i-1)n+1}^{in} \middle| S_s^{i-1} \right\} \\
&= \frac{\Pr\left\{ u, S_s^{i-1}, S_{\bar{s}}^{i}, \mathbf{r}_{(i-1)n+1}^{in} \right\}}{\Pr\{S_s^{i-1}\}} \\
&= \frac{\Pr\left\{ \mathbf{r}_{(i-1)n+1}^{in} \middle| u, S_s^{i-1}, S_{\bar{s}}^{i} \right\} \Pr\left\{ u, S_s^{i-1}, S_{\bar{s}}^{i} \right\}}{\Pr\{S_s^{i-1}\}} \\
&= \frac{\Pr\left\{ \mathbf{r}_{(i-1)n+1}^{in} \middle| u \right\} \Pr\left\{ u, S_s^{i-1}, S_{\bar{s}}^{i} \right\}}{\Pr\{S_s^{i-1}\}} \\
&= \frac{\Pr\left\{ \mathbf{r}_{(i-1)n+1}^{in} \middle| \mathbf{x}_{(i-1)n+1}^{in} \right\} \Pr\left\{ u, S_s^{i-1}, S_{\bar{s}}^{i} \right\}}{\Pr\{S_s^{i-1}\}} \\
&= \frac{\Pr\left\{ \mathbf{r}_{(i-1)n+1}^{in} \middle| \mathbf{x}_{(i-1)n+1}^{in} \right\} \Pr\left\{ u \middle| S_s^{i-1}, S_{\bar{s}}^{i} \right\} \Pr\left\{ S_s^{i-1}, S_{\bar{s}}^{i} \right\}}{\Pr\{S_s^{i-1}\}}
\end{aligned}
$$

$$
\begin{aligned}
&= \Pr\left\{ \mathbf{r}_{(i-1)n+1}^{in} \middle| \mathbf{x}_{(i-1)n+1}^{in} \right\} \Pr\left\{ u \middle| S_s^{i-1}, S_{\bar{s}}^{i} \right\} \Pr\left\{ S_{\bar{s}}^{i} \middle| S_s^{i-1} \right\} \\
&= \begin{cases} \Pr\left\{ S_{\bar{s}}^{i} \middle| S_s^{i-1} \right\} \Pr\left\{ \mathbf{r}_{(i-1)n+1}^{in} \middle| \mathbf{x}_{(i-1)n+1}^{in} \right\}, & (S_s^{i-1}, S_{\bar{s}}^{i}) \in \mathcal{B}_i^{(u)}; \\ 0, & \text{otherwise}; \end{cases} \\
&= \begin{cases} \dfrac{\Pr\left\{ S_{\bar{s}}^{i} \middle| S_s^{i-1} \right\}}{\sqrt{2\pi}\sigma} \exp\left\{ -\dfrac{\sum_{j=(i-1)n+1}^{in} (r_j - x_j(u))^2}{2\sigma^2} \right\}, & (S_s^{i-1}, S_{\bar{s}}^{i}) \in \mathcal{B}_i^{(u)}; \\ 0, & \text{otherwise}; \end{cases} \\
&= \begin{cases} \dfrac{\Pr\{u_i = u\}}{\sqrt{2\pi}\sigma} \exp\left\{ -\dfrac{\sum_{j=(i-1)n+1}^{in} (r_j - x_j(u))^2}{2\sigma^2} \right\}, & (S_s^{i-1}, S_{\bar{s}}^{i}) \in \mathcal{B}_i^{(u)}; \\ 0, & \text{otherwise}. \end{cases}
\end{aligned}
$$

Note:

- x_j is indeed a function of u (or u_i); hence, we denote it by $x_j(u)$.
- $\Pr\left\{ u \middle| S_{\bar{s}}^{i-1}, S_s^{i} \right\}$ is either 1 or 0.
- The initial value of $\Pr\left\{ S_{\bar{s}}^{i} \middle| S_s^{i-1} \right\}$ can be chosen as $1/K$, where K is the number of edges on trellis, which starts from node $S_{\bar{s}}^{i-1}$ and ends at a node at level i.

BCJR (MAP) Algorithm (2)

step 1: Forward recursion

step 1-1:

- $\alpha(S_0^0) = 1;\ \alpha(S_1^0) = \alpha(S_2^0) = \alpha(S_3^0) = 0.$

step 1-2:

- For $i = 1, \ldots, L,\ \ s = 0, 1, 2, 3,\ \ \bar{s} = 0, 1, 2, 3, \quad u = 0, 1,$ compute

$$\gamma(u, S_s^{i-1}, S_{\bar{s}}^i)$$

$$= \begin{cases} \dfrac{\Pr\{u_i = u\}}{\sqrt{2\pi}\sigma} e^{-\dfrac{\sum_{j=(i-1)n+1}^{in} (r_j - x_j(u))^2}{2\sigma^2}}, & (S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(u)}; \\ 0, & \text{otherwise.} \end{cases}$$

step 1-3:

- For $i = 1, \ldots, L$ and $\bar{s} = 0, 1, 2, 3$,

$$\alpha(S_{\bar{s}}^i) = \sum_{s=0}^{3} \alpha(S_s^{i-1}) \sum_{u=0}^{1} \gamma(u, S_s^{i-1}, S_{\bar{s}}^i).$$

step 2: Backward recursion

step 2-1:

- $\beta(S_0^L) = 1$; $\beta(S_1^L) = \beta(S_2^L) = \beta(S_3^L) = 0$.

step 2-2:

- For $i = L-1, \ldots, 0$ and $\bar{s} = 0, 1, 2, 3$, compute

$$\beta(S_{\bar{s}}^i) = \sum_{s=0}^{3} \beta(S_s^{i+1}) \sum_{u=0}^{1} \gamma(u, S_{\bar{s}}^i, S_s^{i+1}).$$

step 3: Soft decision. For $i = 1, \ldots, L$,

$$\Lambda(i) = \log \frac{\displaystyle\sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(1)}} \alpha(S_s^{i-1}) \beta(S_{\bar{s}}^i) \sum_{u=0}^{1} \gamma(u, S_s^{i-1}, S_{\bar{s}}^i)}{\displaystyle\sum_{(S_s^{i-1}, S_{\bar{s}}^i) \in \mathcal{B}_i^{(0)}} \alpha(S_s^{i-1}) \beta(S_{\bar{s}}^i) \sum_{u=0}^{1} \gamma(u, S_s^{i-1}, S_{\bar{s}}^i)}.$$

Some Important Codes

1. Algebraic codes– cyclic codes, Bose-Chaudhuri-Hocquenghem codes (BCH codes), Reed-Solomon codes (RS codes), Algebraic-Geometric codes (AG codes).
2. Codes for bandwidth-limited channels- Ungerboeck codes.
3. Codes approaching Shannon bound- turbo codes, low-density-parity-check codes (LDPC codes).
4. Codes for multi-input (multiple transmit antennas) multi-output (multiple receive antennas) channels- Space-time codes.

References

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate *IEEE Trans. Inform. Theory*, pp. 284–287, March 1974.

[2] M. Bossert, *Channel Coding for Telecommunications*. New York, NY: John Wiley and Sons, 1999.

[3] J. H. Conway and N. J. A. Sloane, Soft Decoding Techniques for Codes and Lattices, Including the Golay Code and the Leech Lattice *IEEE Trans. Inform. Theory*, pp. 41–50, January 1986.

[4] G. D. Forney, Jr., Coset Codes–Part II: Binary Lattices and Related Codes *IEEE Trans. Inform. Theory*, pp. 1152–1187, September 1988.

[5] Y. S. Han, *Efficient Soft-Decision Decoding Algorithms for Linear Block Codes Using Algorithm A**, PhD thesis, School of Computer and Information Science, Syracuse University, Syracuse, NY 13244, 1993.

[6] A. J. Viterbi, Error bound for convolutional codes and an asymptotically optimum decoding algorithm *IEEE Trans. Inform. Theory*, vol. IT-13, no. 2, pp. 260–269, April 1967.

[7] J. K. Wolf, Efficient Maximum Likelihood Decoding of Linear Block Codes Using a Trellis *IEEE Trans. Inform. Theory*, pp. 76–80, January 1978.