

## 11.3 Binary Convolutional Codes

Our principal interest is in the performance of convolutional codes when decoded by the Viterbi algorithm. This requires:

- The free distance  $d_{free}$ .
- Weights of paths thru trellis which begin and end in the 0 state.

We begin by re-drawing the state diagram as a **signal flow graph**<sup>a</sup>.

---

<sup>a</sup>Refs: S.B. Wicker, *Error Control Systems*, Prentice-Hall, 1995, (on reserve); W.H. Huggins and D.E. Entwistle, *Introductory Systems and Design*, Blaisdell, 1968; W.H. Huggins, "Signal-Flow Graphs and Random Signals," *Proc. IRE (IEEE)*

### 11.3.1 Introduction to Signal Flow Graphs

**Definition 3** A **graph** is a collection of nodes (states) that are interconnected by branches.

**Definition 4** In a **directed graph**, a direction is associated with the branch which connects two nodes.

**Definition 5** In a **weighted graph**, each branch is assigned a value of some quantity, or **weight**.

**Definition 6** A **weighted directed graph** is one which is both weighted and directed.

**Definition 7** A **signal flow graph** is a weighted directed graph in which each node represents a **signal** and each branch value is called the **transmittance**.

**Definition 8** A **path** through a signal flow graph

- is a sequence of nodes, source  $\longrightarrow$  destination
- adjacent nodes connected by (directed) branches,
- such that all directions point from source to destination.

Further,

- a path which starts and stops at the same node is a **circuit**;
- a path which does not enter any node more than once is a **loop**.



The references contain a relatively complete algebra of signal flow graphs. We need only the basics.

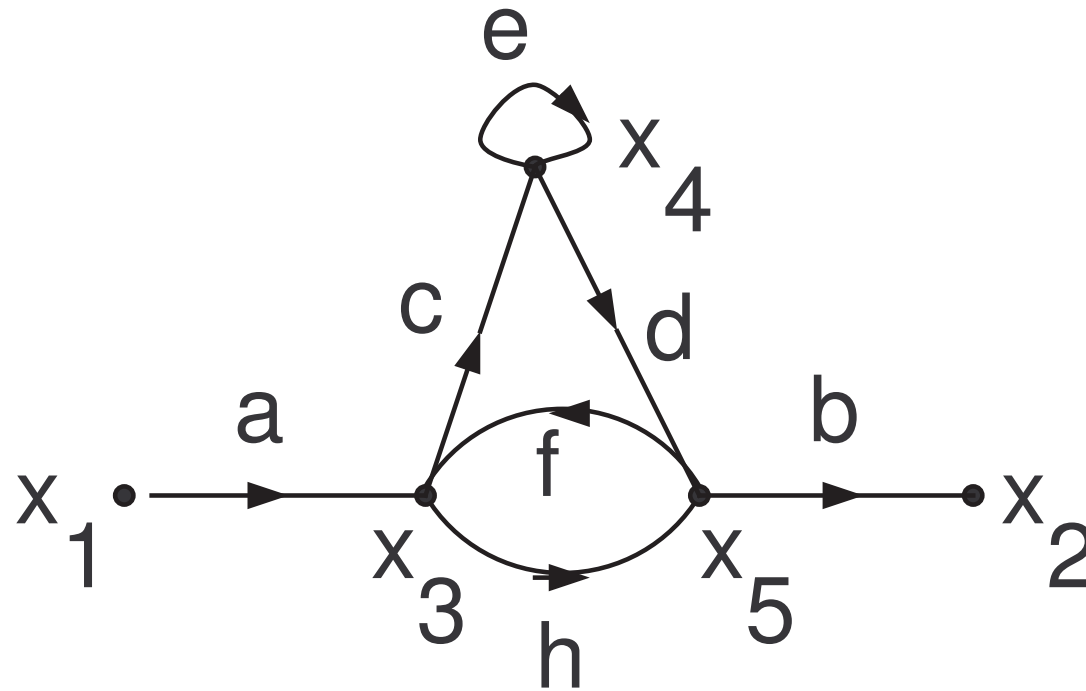
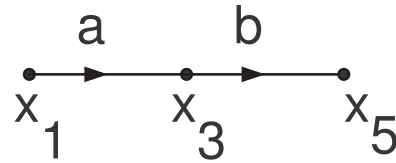


Figure 4: Simple Signal Flow Graph

## Basic rules:



$$x_3 = ax_1$$

$$x_5 = bx_3 = abx_1 \quad (6)$$

## More basic rules:

- If two or more branches **terminate** at one node, the signal at that node is the **sum** of the branch output signals.
- If two or more branches **emanate** from one node, each branch carries the signal value of the node.
- *Therefore*, two branches having **common source and common destination** nodes can be replaced by one branch with transmittance equal to the

**sum** of the transmittances of the parallel branches.

**Loops:** In Figure 4,

- there is a distinct path from  $x_3$  to  $x_5$  that goes through  $x_4$  0, 1, 2,  $\dots$  number of times;
- the path transmittances from  $x_3$  to  $x_5$  through  $x_4$  are

$$cd, ced, ce^2d, ce^3d, \dots ;$$

- the total path transmittance from  $x_3$  to  $x_5$  is the sum of the transmittances of all paths joining the two signal nodes:

$$\begin{aligned} T_{35} &= cd(1 + e + e^2 + e^3 + \dots) \\ &= \frac{1}{1 - e}; \end{aligned}$$

- therefore, the loop at node  $x_4$  can be replaced with a branch which has transmittance  $= 1/(1 - e)$ .

**Definition 9** *The **graph transmittance** of a signal flow graph at some designated output node is the value of the signal at that node for a given unit signal at the designated input node.* □

This is equivalent to saying that the graph transmittance from  $x_1$  to  $x_2$  is  $x_2/x_1$ . We also call this the **transfer function** of the graph.

To compute the transfer function of the signal flow graph in Figure 4, it is convenient to simplify the graph using the basic rules.

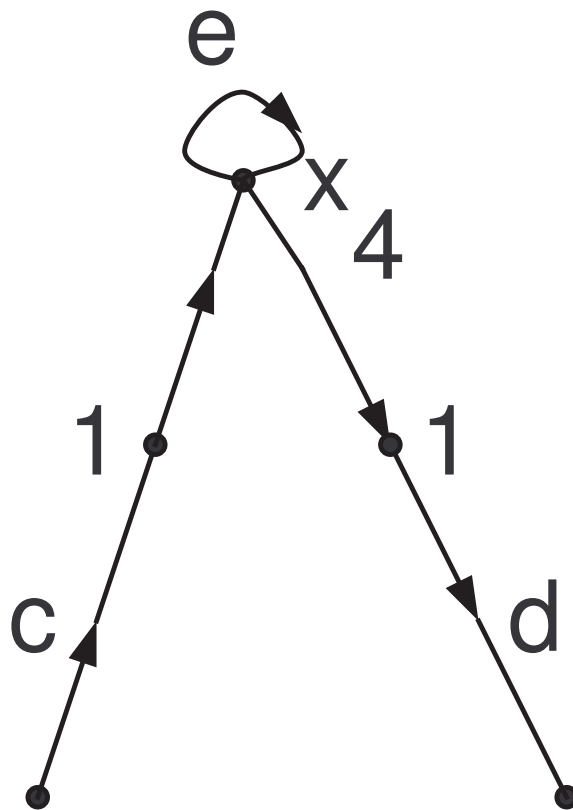


Figure 5: Partial signal flow graph

Consider the path  $x_3 \rightarrow x_4 \rightarrow x_5$ .

1. Break up branches with weights  $c$  and  $d$  as shown in Figure 5.

2. Then this subgraph reduces to a single branch with transmittance  $cd/(1-e)$ .
3. Therefore, the graph between  $x_3, x_5$  consists of three branches:
  - (a) A forward branch with transmittance  $h$ .
  - (b) A reverse branch with transmittance  $f$ .
  - (c) A forward branch with transmittance  $cd/(1-e)$ .

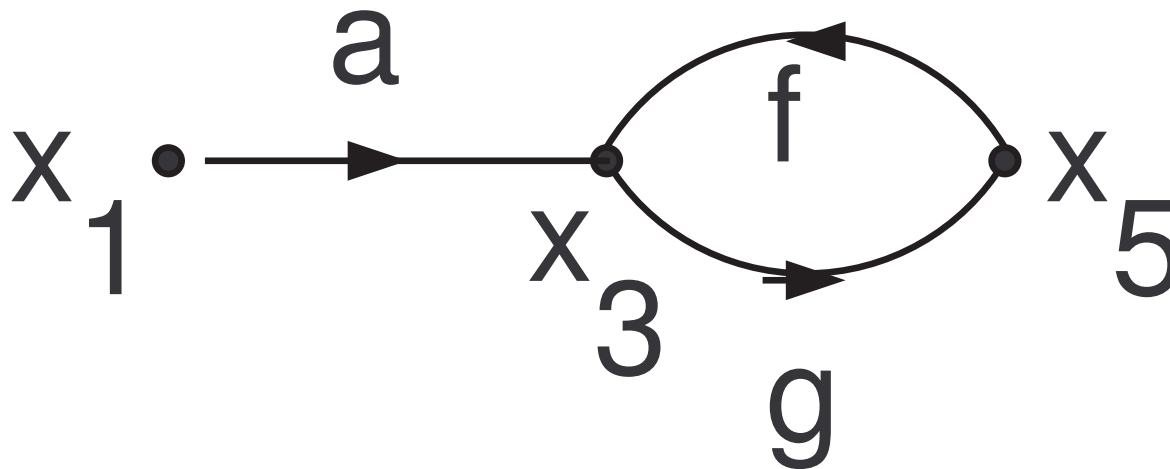


Figure 6: Partial signal flow graph.

4. Summing the two forward transmittances reduces the signal flow graph to

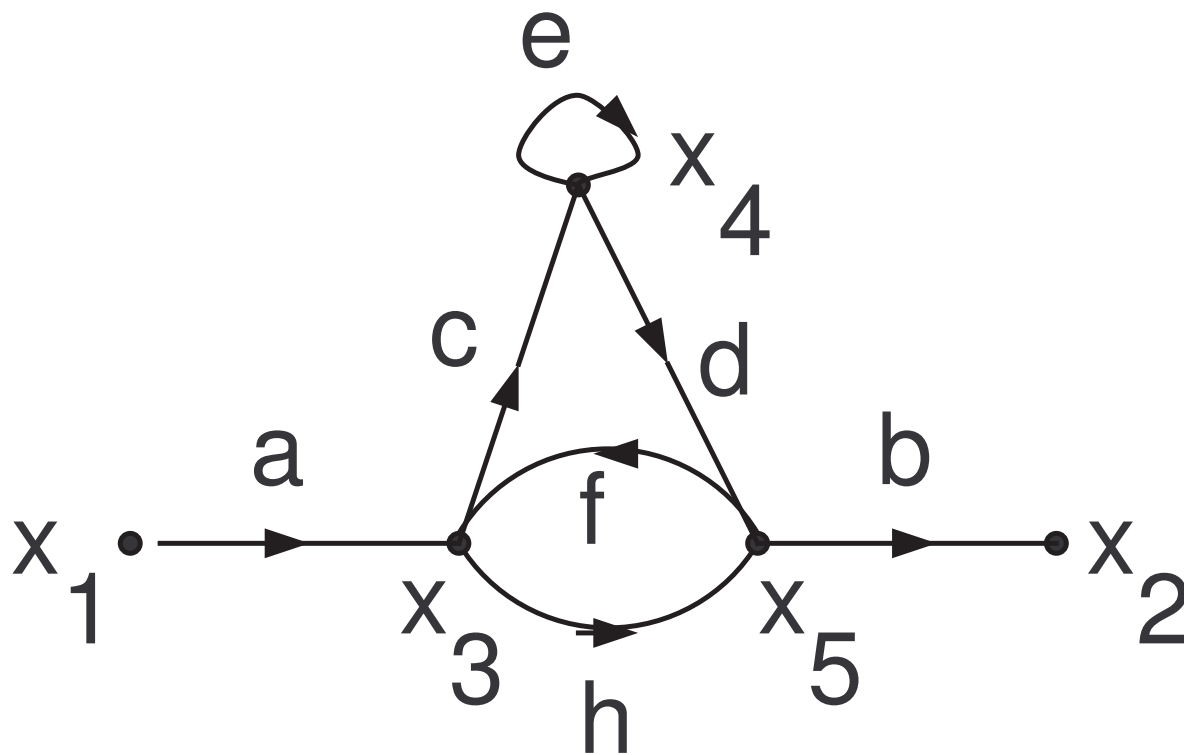
that shown in Figure 6, where

$$g = h + \frac{cd}{(1 - e)}$$

5. The  $f - g$  loop is replaced with branch having transmittance  $g/(1 - fg)$  and the transfer function from  $x_1$  to  $x_5$  is

$$\begin{aligned} T_{15} &= \frac{ag}{1 - fg} \\ &= \frac{ah - ahe + ecd}{1 - fg - e + efg} \\ &= \frac{acd + ah(1 - e)}{1 - (e + fh + cdf) + fhe} \end{aligned}$$

We show the graph and its complete transfer function together.



$$T_{12} = \frac{acdb + ahb(1 - e)}{1 - (e + fh + cdf) + fhe} \quad (7)$$

Notice the following in (7)

1. Terms  $acdb$  and  $ahb$  are transmittances of **forward** paths from  $x_1$  to  $x_2$ .
2. Terms  $e$ ,  $fh$ , and  $cdf$  are transmittances of loops. Such transmittances are called **loop gains**.
3. The term  $fhe$  is the product of two **non-touching** loop gains.

These provide examples for application of **Mason's Direct Rule** for computing the transfer function between any two points in a signal flow graph<sup>a</sup>.

---

<sup>a</sup>In addition to the previous references, one can also read: S.J. Mason, "Feedback theory: Some properties of signal flow graphs," *Proc. IRE (IEEE)*, vol 41, pp 1144-1156, Sep 1953; S.J. Mason, "Feedback theory: Further properties of signal flow graphs," *Proc. IRE(IEEE)*, vol. 44, pp 920-926, July 1956; S.J. Mason and H.J. Zimmermann, *Electronic Circuits, Signals, and Systems*, New York: Wiley, 1960

**Theorem 1 Mason's Direct Rule** *The transfer function between any two signals in a signal flow graph can be written directly as:*

$$\frac{x_{out}}{x_{in}} = \frac{1}{\Delta} \sum_{k=1}^n P_k \Delta_k$$

where

$P_k \triangleq$  the transmittance of the  $k^{th}$  **forward path** from  $x_{in}$  to  $x_{out}$ ;

$\Delta \triangleq 1 -$  (the sum of all loop gains)  $+$  (the sum of products of all possible pairs of non-touching loops)  $-$  (the sum of products of all possible triplets of non-touching loops)  $+$   $\dots$ ;

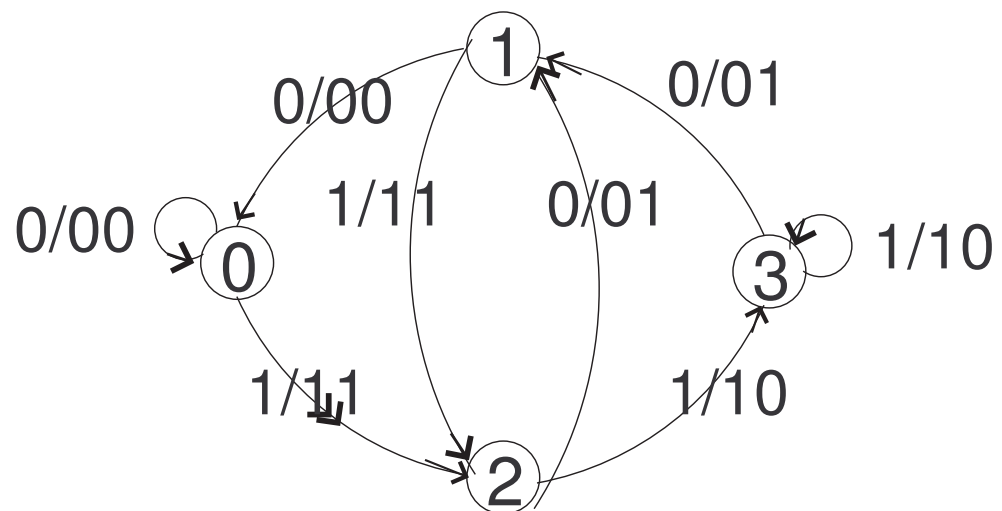
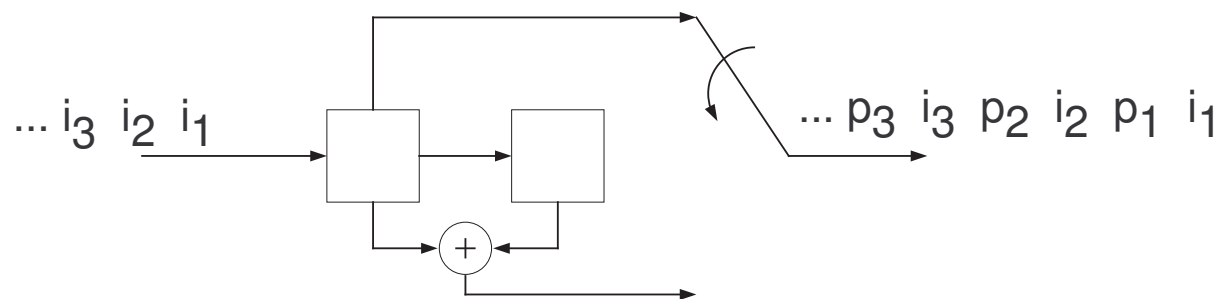
$\Delta_k \triangleq$  the value of  $\Delta$  for that portion of the graph not touching the  $k^{th}$  forward path. □

## 11.3.2 Weight Enumerators of Convolutional Codes

### Summary:

- Use signal flow graphs to find the weights of error paths.
- Use Mason's rule to derive the **transfer function** or **path gain** as the quotient of two polynomials.
- Expand the quotient via polynomial long division.
- The coefficient of  $X^w$  in the expansion is the number of possible error patterns of weight  $w$ .
- Branch transmittances can include the variable  $L$  to represent **path length**.

Consider the code of **Example 1** under *Convolutional Codes*.



To construct the appropriate signal flow graph,

1. replicate the topology of the state transition diagram;
2. split the ZERO node into two identical copies (so that paths which begin and end in the ZERO state can be studied);
3. replace the branch label with a transmittance  $= X^j$  where  $j =$  the weight of the branch's output  $n$ -tuple;
4. multiply powers of  $X$  to sum their exponents, thus adding the weight along a path;
5. notice that  $d_{free}$  is given by the smallest nonzero exponent of  $X$ ;
6. The result is shown below.

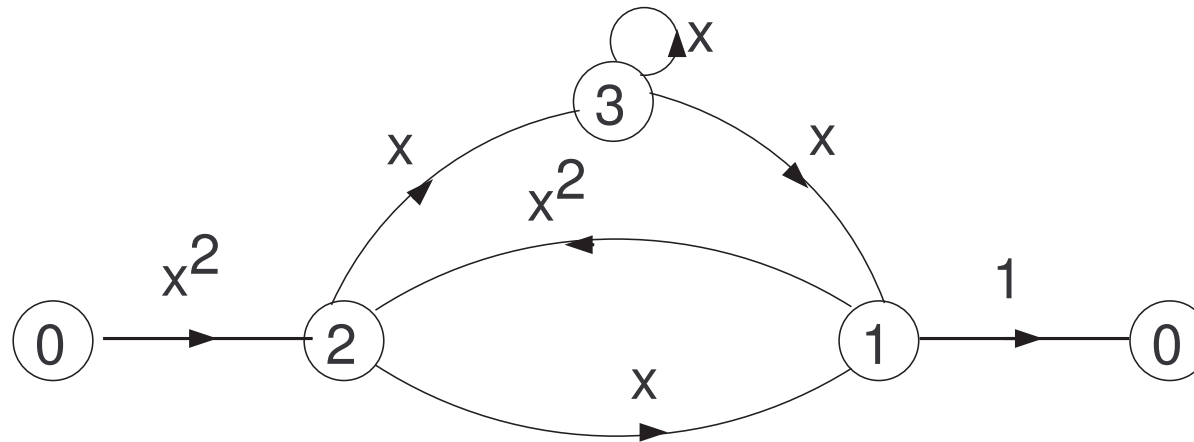


Figure 7: Signal flow graph for encoder of Example 1, CCs

**Forward paths:****Loops:**

$$P_1 : 0 \ 2 \ 1 \ 0 \quad X^3$$

$$\mathcal{L}_1 : 3 \ 3 \quad X$$

$$P_2 : 0 \ 2 \ 3 \ 1 \ 0 \quad X^4$$

$$\mathcal{L}_2 : 2 \ 3 \ 1 \ 2 \quad X^4$$

$$\mathcal{L}_3 : 2 \ 1 \ 2 \quad X^3$$

Finally,

$$\Delta = 1 - (X + X^3 + X^4) + X \cdot X^3$$

$$\Delta_1 = 1$$

$$\Delta_2 = 1 - X$$

and,

$$\begin{aligned} T_{0-0} &= \frac{X^4 + X^3(1 - X)}{1 - X - X^3} \\ &= \frac{X^3}{1 - X - X^3} \\ &= X^3 + X^4 + X^5 + 2X^6 + 3X^7 + \dots \end{aligned}$$

Therefore  $d_{free} = 3$  and there are:

1 path of weight 3 (0 2 1 0) (length = 3)

1 path of weight 4 (0 2 3 1 0) (length = 4)

1 path of weight 5 (0 2 3 3 1 0) (length = 5)

2 paths of weight 6

(0 2 3 3 3 1 0) (length = 6)

(0 2 1 2 1 0) (length = 5)

3 paths of weight 7

(0 2 3 3 3 3 1 0) (length = 7)

(0 2 3 1 2 1 0) (length = 6)

(0 2 1 2 3 1 0) (length = 6)

Weight enumerators will permit bounding of decoding error probability.

But to compute **bit error probability** we need the number of information bits that represent a given (incorrect path). This is determined from knowledge of the **path length**.

Write each branch transmittance as  $X^j L$ . The transfer function becomes a polynomial in  $X^j L^h$  where  $h =$  the number of branches in the path.

$$T(X, L) = \sum_{jh} A_{jh} X^j L^h$$

In the previous example:

$$P_1 = X^3 L^3$$

$$P_2 = X^4 L^4$$

$$\mathcal{L}_1 = XL$$

$$\mathcal{L}_2 = X^4 L^3$$

$$\mathcal{L}_3 = X^3 L^2$$

$$\Delta = 1 - (XL + X^3 L^2 + X^4 L^3) + X^4 L^3$$

$$\Delta_1 = 1 - XL$$

$$\Delta_2 = 1$$

Substituting gives

$$\begin{aligned} T(X, L) &= \frac{X^3 L^3}{1 - XL - X^3 L^2} \\ &= X^3 L^3 + X^4 L^4 + X^5 L^5 + X^6 (L^6 + L^5) \\ &\quad + X^7 (L^7 + 2L^6) + X^8 (L^8 + 3L^7) + \dots \end{aligned}$$

### 11.3.3 Error Probabilities for Viterbi Decoding I: The BSC

Let  $\mathbf{r}$  = the received sequence and  $\mathbf{c}$  be a code sequence. Then  $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}$ . If the channel is binary,  $\mathbf{e}$  is a binary sequence in which a 1 represents an error. If the channel is continuous and memoryless, then  $\mathbf{e}$  is a sequence of *i.i.d.* samples of the channel noise process (and  $\oplus$  is just  $+$ ).

**Definition 10** The **optimal** or **maximum likelihood** decoder maximizes  $P[\mathbf{r}|\mathbf{c}]$  where

$$P[\mathbf{r}|\mathbf{c}] = \prod_{i=1}^{\infty} P(\mathbf{r}_i|\mathbf{c}_i)$$

□

where  $\mathbf{c}_i = i^{th}$  branch ( $n$ -tuple) of the code sequence and  $\mathbf{r}_i = i^{th}$  branch of received sequence  $\mathbf{r}$ . Or,

$$P(\mathbf{r}|\mathbf{c}) = \prod_{i=1}^{\infty} \prod_{j=1}^n P(r_{ij}|c_{ij})$$

where the channel is understood to be memoryless.

For a sequence of  $L$  code blocks, there are  $2^{RL}$  possible code sequences,  $R =$  the code rate. So, it is important to reduce the “complexity” of the computation. This is done by:

1. finding the maximum likelihood (ML) sequence to each trellis state at time  $T_i$ ;
2. discarding all sequences arriving at each state at  $T_i$  that are not ML to that point in the trellis. *These sequences do not extend any further and are said to be eliminated.* (The algorithm for doing this is treated elsewhere.)

In what follows, we assume that the all zero code word is transmitted. Hence, the error sequences are merely the set of nonzero code sequences.

Then, if  $\nu$  errors occur in the transmission of  $N$  symbols over the discrete (e.g., the BSC) memoryless channel,

$$P(\mathbf{r}|\mathbf{c}) = p^\nu (1 - p)^{N-\nu}$$

where  $d_H(\mathbf{r}, \mathbf{c}) = \nu$ , the number of channel errors.

$$P(\mathbf{r}|P\mathbf{c}) = \left( \frac{p}{1-p} \right)^\nu (1-p)^n$$

and if  $p < 1/2$ , then  $p/(1-p) < 1$ . Therefore, optimal decoding can be achieved, on the BSC, by minimum distance decoding. □

## Analysis of $P(e)$

**Definition 11** A **first event error** is said to occur at some node  $S$  whenever a non-zero path beginning at  $S$  survives for the first time.  $\square$

- Assume that the **all-zero code word** is transmitted.
- If the incorrect path has transmittance  $X^d$ , an error event occurs whenever  $d_H(\mathbf{r}, \mathbf{y}) < d/2$ . This occurs when:
  - **odd  $d$** : a received word of weight  $\geq (d + 1)/2$  is closer to the incorrect path of weight  $d$ .
  - **even  $d$** : a received word of weight  $> d/2 + 1$  is closer to the error path; if its weight  $= d/2$ , it is closer to the incorrect path with probability  $= 1/2$ .

Now consider some received sequence  $\mathbf{r}$ .

- The all-zero path has metric  $d_H(\mathbf{r}, \mathbf{0})$ .
- Each non-zero path  $\mathbf{y}'$  which diverges from  $\mathbf{0}$  at a node of interest has path metric  $d_H(\mathbf{r}, \mathbf{y}')$ . Consider all such  $\mathbf{y}'$ .
- Whenever  $d_H(\mathbf{r}, \mathbf{y}') < d_H(\mathbf{r}, \mathbf{0})$  and  $\mathbf{y}'$  **survives**, an error event has occurred.
- **Often, however**  $\mathbf{y}'$  re-merges with  $\mathbf{0}$  prior to the end of  $\mathbf{r}$ . Not all such departures represent error events.
- But a useful upper bound to the probability of an error event is given by the sum of probabilities of all such partial path departures.
- This calls for the **union bound**:

**Theorem 2** Let  $\{E_1, E_2, \dots, E_n\}$  be events in a probability space. Then

$$P[E_1 \cup E_2 \cup \dots \cup E_n] \leq \sum_{i=1}^n P(E_i)$$

□

We will use this immediately.//

The probability of a first event error to a path of weight  $d$  is

$$\begin{aligned} P_d &= \sum_{\ell=(d+1)/2}^d \binom{d}{\ell} p^\ell (1-p)^{d-\ell}, \quad d \text{ odd} \\ &= \frac{1}{2} \binom{d}{d/2} p^{d/2} (1-p)^{d-d/2} \\ &\quad + \sum_{\ell=d/2+1}^d \binom{d}{\ell} p^\ell (1-p)^{d-\ell}, \quad d \text{ even} \end{aligned}$$

For  $d$  odd,

$$\begin{aligned}
 P_d &= \sum_{\ell=(d+1)/2}^d \binom{d}{\ell} p^\ell (1-p)^{d-\ell} \\
 &< \sum_{\ell=(d+1)/2}^d \binom{d}{\ell} p^{d/2} (1-p)^{d-d/2} \\
 &< p^{d/2} (1-p)^{d/2} \sum_{\ell=(d+1)/2}^d \binom{d}{\ell} \\
 &< p^{d/2} (1-p)^{d/2} \sum_{\ell=0}^d \binom{d}{\ell} \\
 &< 2^d p^{d/2} (1-p)^{d/2}
 \end{aligned}$$

The same bound can be shown for even  $d$  as well.

Then, at time  $t$ , the first event error probability is bounded as follows:

$$P_f[e, t] \leq \sum_{d=d_{free}}^{\infty} A_d P_d$$

where  $A_d =$  the number of paths of weight  $d$ . Substitution from above gives

$$P_f[e, t] < \sum_{d=d_{free}}^{\infty} A_d 2^d p^{d/2} (1-p)^{d/2}$$

or,

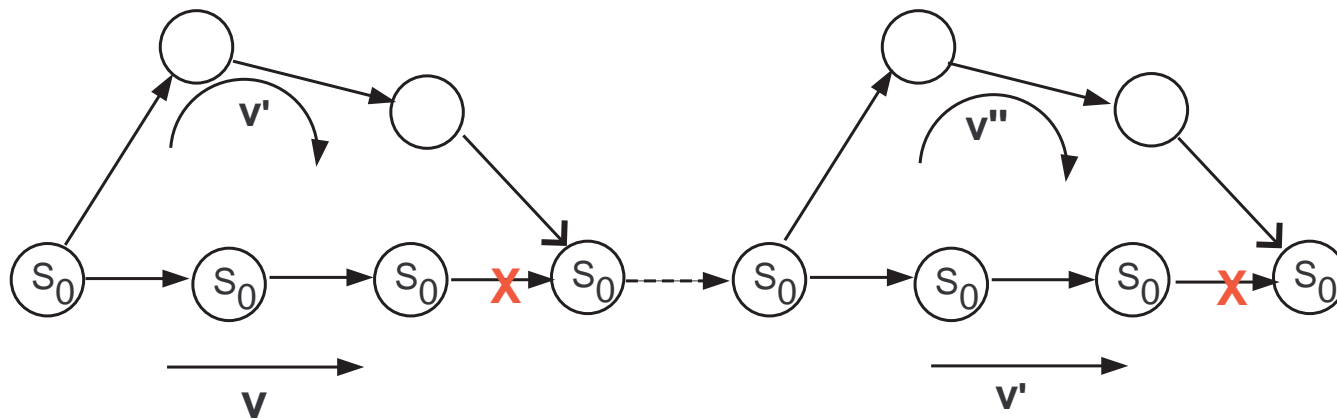
$$P_f[e, t] \leq \sum_{d=d_{free}}^{\infty} A_d \left[ 2\sqrt{p(1-p)} \right]^d$$

Both of these can be shown for even  $d$  as well.

**Lemma 3** *All incorrect paths of length  $t$  branches (or fewer) can cause a first event error at time  $t$ .*

□

To see this, see the figure.



- Path  $v'$  eliminates  $v$  at time  $t - \ell$ .
- Path  $v''$  eliminates  $v'$  at time  $t$ .
- Therefore  $v''$  eliminates  $v$  at  $t$ .

*i.e.*,  $t$  is the first time that  $v''$  eliminates  $v$ , so an event error (not the first) occurs at  $t$ . Since this event is less likely than the first (it requires 2 breaks),

$$P(e, t) \leq P_f(e, t)$$

Therefore,

$$P[e] < \sum_d A_d P_d$$

Recall the codeword WEF:

$$T(X) = \sum_d A_d X^d.$$

Combining  $P[e]$  with  $T(X)$ , we can write

$$P[e] < T(X)|_{x=2\sqrt{p(1-p)}}.$$

because of the independence from  $t$  shown above.

**Question:** *What is the expected number of bit errors per decoded bit ?*  
*i.e., what is the*

- average **bit error probability** or **bit error rate**?

To answer, we generalize the weight enumerator function again.

$$T(W, X, L) = \sum_{w,d,\ell} A_{w,d,\ell} W^w X^d L^\ell$$

where  $A_{w,d,\ell}$  = the number of sequences

- of weight  $d$
- of length  $\ell$  (blocks)
- specified by  $w$  **nonzero information bits** ( $w$  ONEs).

We call  $T(W, X, L)$  the **codeword input/output weight enumeration function (IOWEF)**.

- IOWEF is a property of the **encoder** since it depends upon the **mapping** of information sequences to code sequences.
- The WEF  $T(X)$  is a property of the **code** since it merely enumerates the codeword weights.

Consider all the code sequences that begin at some fixed node  $S_0$  on the **all zero** path, and let

$$T(W, X) = T(W, X, L)|_{L=1} = \sum_{w,d} A_{w,d} W^w X^d$$

where  $A_{w,d}$  = the number of weight  $d$  code sequences that are produced by weight  $w$  information sequences.

We also recall the codeword WEF:

$$\begin{aligned} T(X) &= \sum_d A_d X^d \\ &= T(W, X)|_{X=1} \\ &= T(W, X, L)|_{W=L=1} \end{aligned}$$

where

$$A_d = \sum_w A_{w,d}$$

Notice that we can also form a **codeword information WEF**:

$$T_w(X) = \sum_d A_{w,d} X^d$$

and can write  $T(W, X)$  in terms of  $T_w(X)$ :

$$T(W, X) = \sum_w W^w T_w(X)$$

Let

$$\mathcal{B}_{w,d} \triangleq w A_{w,d}.$$

Clearly,  $\mathcal{B}_{w,d}$  is the number of information bits associated with all codewords of weight  $d$  and information weight  $w$ . (Let's call these codewords  $(w, d)$  **codewords**.)

Now form

$$\begin{aligned} \frac{\partial T(W, X)}{\partial W} &= \sum_{w,d} w A_{w,d} W^{w-1} X^d \\ \left. \frac{\partial T(W, X)}{\partial W} \right|_{W=1} &= \sum_{w,d} \mathcal{B}_{w,d} X^d \\ &\triangleq T_b(W, X) \end{aligned}$$

where  $T_b(W, X)$  is an **information bit WEF** which we can manipulate to write

$$T_b(W, X) = \sum_d X^d \sum_w \mathcal{B}_{w,d}$$

Define

$$\mathcal{B}_d \triangleq \sum_w \mathcal{B}_{w,d}$$

and notice that  $\mathcal{B}_d$  = the **total number of nonzero information bits (i.e., errors)** associated with the code words of weight  $d$ .

- But as soon as the decoder has selected the **first** branch of a weight  $d$  (incorrect) codeword, it has **decoded  $k$  information bits**.
- Therefore, for that weight  $d$  codeword, the number of bit errors per information bit is  $\mathcal{B}_d/k$ .
- Further, the **average** number of bit errors over all code words is

$$\sum_d \mathcal{B}_d P_d$$

where  $P_d$  is the probability of a weight  $d$  codeword being selected when the all zero codeword was transmitted.

- Finally, then, the average bit error probability is

$$P_b[e] = \frac{1}{k} \sum_{d=d_{free}}^{\infty} \mathcal{B}_d P_d$$

But

$$\begin{aligned}
 \mathcal{B}_d &= \sum_w \mathcal{B}_{w,d} \\
 &= \sum_w w A_{w,d} \\
 &= \left. \frac{\partial T(W, X)}{\partial W} \right|_{W=X=1}
 \end{aligned}$$

and, substituting the upper bound  $2^d p^{d/2} (1-p)^{d/2}$  for  $P_d$  gives

$$P_b[e] < \frac{1}{k} \left. \frac{\partial T(W, X)}{\partial W} \right|_{W=1, X=2\sqrt{p(1-p)}}$$

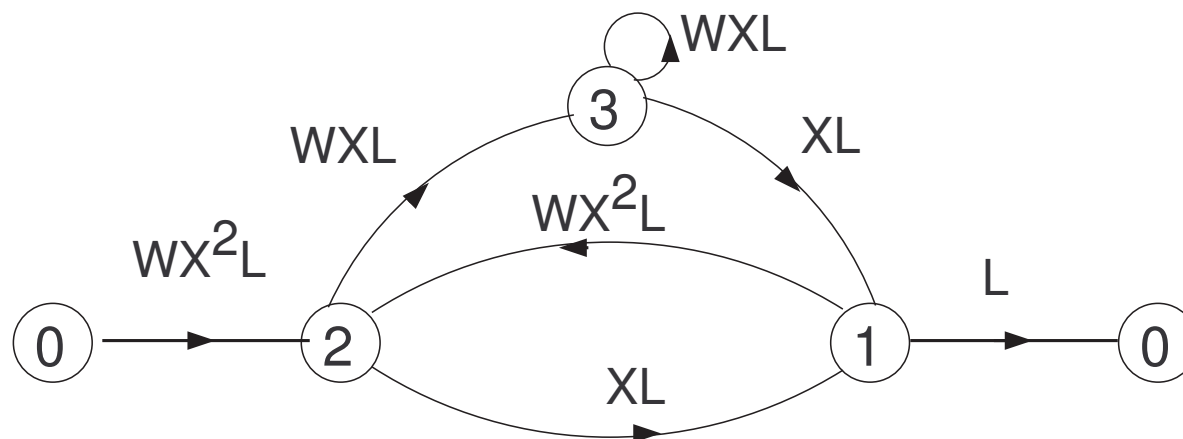
where the inequality in the last line comes from the upper bound to  $P_d$ . Finally,

$$P_b[e] < \frac{1}{k} \sum_d \mathcal{B}_d \left[ 2\sqrt{p(1-p)} \right]^d .$$

For relatively quiet channels (small  $p$ ), the foregoing simplifies to

$$P_b[e] \sim B_{d_{free}} \left[ 2\sqrt{p(1-p)} \right]^{d_{free}} \sim B_{d_{free}} 2^{d_{free}} p^{d_{free}/2}$$

**Example:** For the code used throughout this section,



we derive the codeword IOWEF as follows:

$$\begin{aligned}
P_1 &= WX^3L^3 \\
P_2 &= W^2X^4L^4 \\
\mathcal{L}_1 &= WXL \\
\mathcal{L}_2 &= W^2X^4L^3 \\
\mathcal{L}_3 &= WX^3L^2 \\
\Delta &= 1 - (WXL + WX^3L^2 + W^2X^4L^3) + X^4L^3 \\
&= 1 - WXL - WX^3L^2 \\
\Delta_1 &= 1 - WXL \\
\Delta_2 &= 1
\end{aligned}$$

Substituting gives

$$T(W, X, L) = \frac{WX^3L^3}{1 - WXL - WX^3L^2}$$

The modified WEF is given by setting  $L = 1$  in the IOWEF. The **upper bound** to bit error probability becomes

$$\begin{aligned}
 P_b[e] &< \frac{1}{k} \frac{\partial T(W, X)}{\partial W} \Big|_{X=2\sqrt{p(1-p)}, W=1} \\
 &< WX^3 + W^2X^4 + W^3X^5 + (W^2 + W^4)X^6 \\
 &\quad + (2W^3 + W^5)X^7 + \dots \Big|_{X=2\sqrt{p(1-p)}, W=1} \\
 &< 8p^{3/2}(1-p)^{3/2} + 16p^2(1-p)^2 + 32p^{5/2}(1-p)^{5/2} \\
 &\quad + 128p^3(1-p)^3 + 384p^{7/2}(1-p)^{7/2} + \dots
 \end{aligned}$$

To obtain a **lower bound** to bit error rate, notice that

- the most likely error pattern has weight  $d_{free}$ ;
- when an error event occurs, assume  $k$  bits are in error;
- the probability of an error event **exceeds**  $P_{d_{free}}$  because other error patterns have finite probability.

Therefore,

$$P_b \geq \frac{1}{k} P_{d_{free}}.$$

For the example,  $d_{free} = 3$ , so

$$P_b \geq 2^{d_{free}} p^{d_{free}/2} (1-p)^{d_{free}/2} > P_{d_{free}}$$

or

$$P_b > 8p^{3/2}(1-p)^{3/2}$$

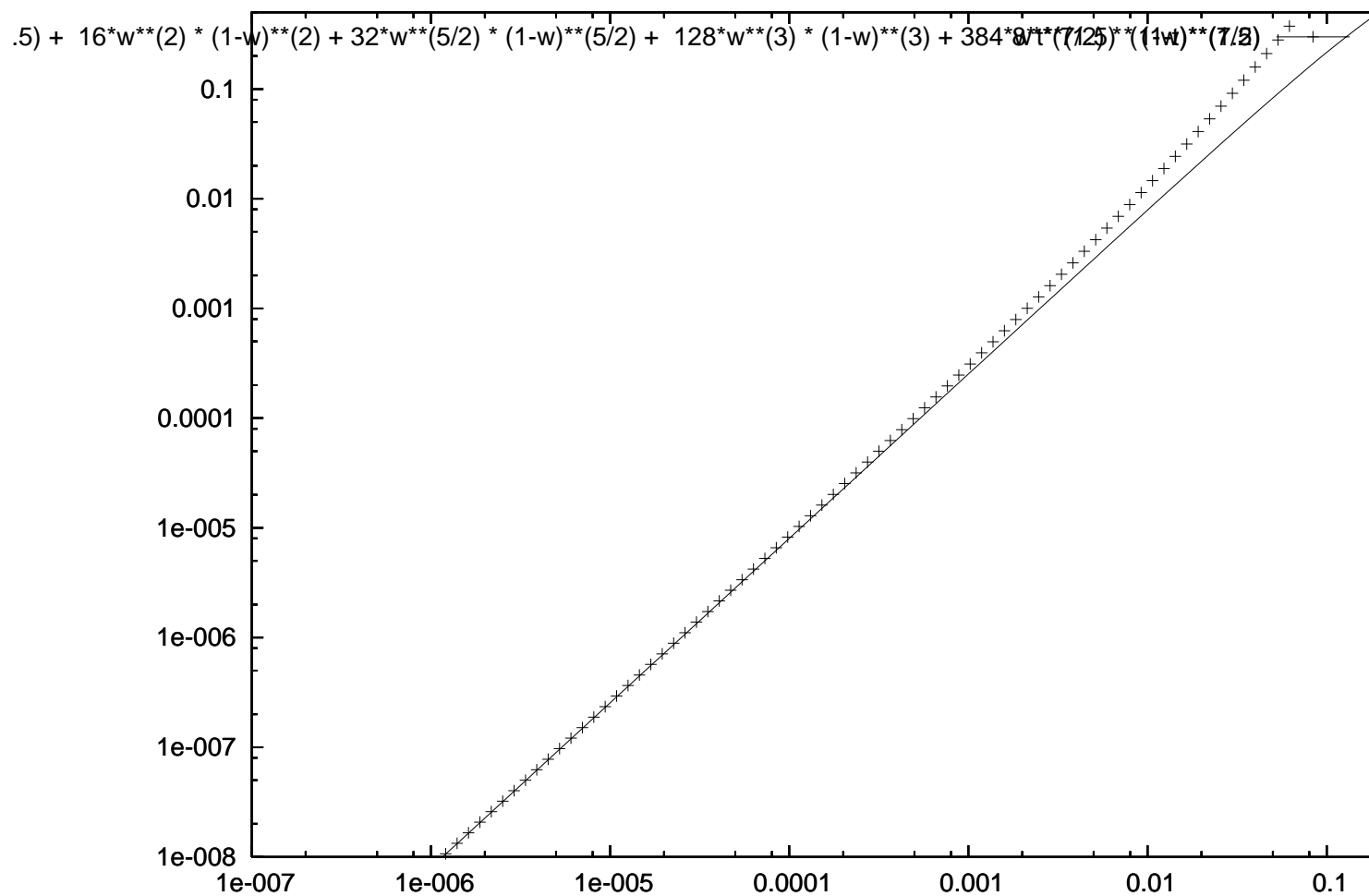


Figure 8: Upper and lower bounds for the example